Hợp tác Đức
DEUTSCHE ZUSAMMENARBEIT

Implemented by giz Deutsche Gesellschaft für Internationale Zusammenarbeit (GIZ) GmbH

develoPPP Where business meets development.

DVET

BOSCH Sáng tạo vì cuộc sống

rexroth A Bosch Company

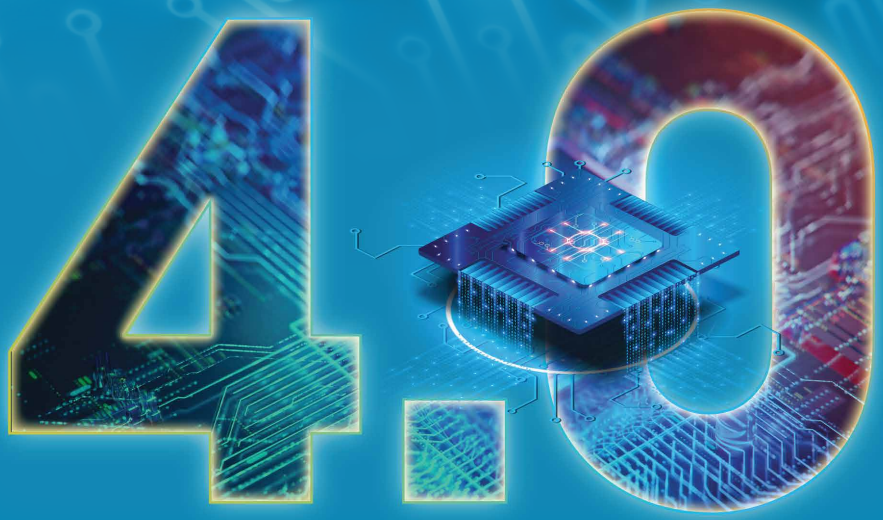
LILAMA2 COLLEGE

Hợp tác Phát triển với Khối Doanh nghiệp Tư nhân

Tích hợp các Yêu cầu của Ngành Công nghiệp 4.0 vào GDNN

Mô-đun Đào Tạo

Công Nghiệp 4.0



Lập Trình Hệ Thống Cơ Điện Tử

Chương trình *develoPPP*
của Bộ Hợp tác Kinh tế và Phát triển Liên Bang Đức

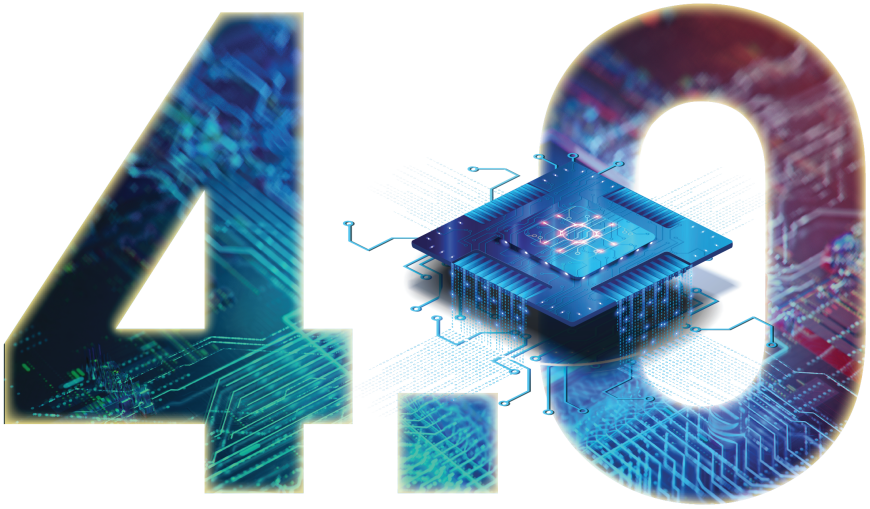
Hợp tác Phát triển với Khối Doanh nghiệp Tư nhân

Tích hợp các Yêu cầu của Ngành Công nghiệp 4.0 vào GDNN

Bosch Rexroth - GIZ - LILAMA 2

Mô-đun Đào Tạo

Công Nghiệp



Lập Trình Hệ Thống Cơ Điện Tử

Hợp tác Phát triển với Khối Doanh nghiệp Tư nhân
Tích hợp các Yêu cầu của Ngành Công nghiệp 4.0 vào GDNN
Bosch Rexroth - GIZ - LILAMA 2

Mô-đun đào tạo
Công nghiệp 4.0 - Lập trình Hệ thống Cơ điện tử
Phiên bản đầu tiên

Tài liệu đào tạo - Bài tập - Công trình dự án - Giải pháp

Ban biên tập

Stefan Paschek, Kieu Tan Thoi, Phan Hong Phuong
Nguyen Trong Tin, Le Van Hung
Frank Schulze, Ralf Hill

Đóng góp

LILAMA 2 International Technology College
Technology Engineering Faculty
Training Department
Quality Assurance Department

GIZ Programme Reform of TVET in Viet Nam
Pham Thi Thanh Truc
Nguyen Minh Cong

Phiên dịch

English language to Vietnamese
Nguyen Minh Ngoc
Tran Simon Trung Hieu

Cuốn sách Đào tạo Mô-đun Công nghiệp 4.0 - Lập trình của Mechatronic Systems được chính thức xuất bản bởi các đối tác hợp tác của liên danh phát triển
"Tích hợp các yêu cầu của Công nghiệp 4.0 vào TVET".
Nó đã được phổ biến cho mười một Viện TVET đối tác của Chương trình Cải cách TVET ở Việt Nam và cung cấp cho các bên liên quan khác trong TVET và
trong ngành sản xuất.

Địa chỉ: số 1, Ngõ 17, Phố Tạ Quang Bửu, Phường Bách Khoa, Quận Hai Bà Trưng, Hà Nội, Việt Nam
Tel: *+84 (0) 24 39 74 64 71*
Fax: *+84 (0) 24 39 74 65 70*
Website: *www.tvet-vietnam.org*
www.giz.de/vietnam

LỜI MỞ ĐẦU

Lời nói đầu cho các bên liên quan đến Công nghiệp 4.0 trong TVET và ngành sản xuất

Giáo dục và Đào tạo Kỹ thuật và Dạy nghề (TVET) đóng một vai trò quan trọng trong việc phát triển lực lượng lao động có kỹ năng và năng lực cho các lĩnh vực công nghiệp tiên tiến. Sự thích ứng của kiến thức và kỹ năng của lực lượng lao động Việt Nam với các yêu cầu năng lực mới về số hóa và Công nghiệp 4.0 là một thách thức cần được giải quyết bởi các cơ sở TVET phối hợp chặt chẽ với khu vực doanh nghiệp.

Các đối tác hợp tác Bosch Rexroth AG, Trường Cao đẳng Công nghệ Quốc tế LILAMA 2 (LILAMA 2), với sự hỗ trợ của Tổng cục Giáo dục và Đào tạo Nghề nghiệp (DVET), và Deutsche Gesellschaft für Internationale Zusammenarbeit (GIZ) GmbH đã giải quyết thách thức để đủ điều kiện kỹ thuật viên và kỹ sư đáp ứng các yêu cầu của Công nghiệp 4.0 tại LILAMA 2 và các cơ sở đào tạo nghề khác và thực hiện quan hệ đối tác phát triển "Tích hợp các yêu cầu của Công nghiệp 4.0 trong dạy nghề".

Quan hệ đối tác phát triển với Khu vực tư nhân (DPP) thúc đẩy các hoạt động của khu vực tư nhân nơi các cơ hội kinh doanh và tiềm năng chính sách phát triển gặp gỡ và được hỗ trợ bởi chương trình www.develoPPP.de của Bộ Hợp tác và Phát triển Kinh tế Liên bang Đức (BMZ).

Các hoạt động chính của các đối tác hợp tác bao gồm: nâng cao nhận thức về những thay đổi cần thiết trong hệ thống TVET ở Việt Nam; phân tích nhu cầu hiện tại và tương lai của ngành; phát triển và tích hợp các mô-đun đào tạo về Công nghiệp 4.0 vào các chương trình đào tạo ban đầu và các khóa đào tạo tiếp theo; phát triển năng lực kỹ thuật và giáo khoa của giáo viên TVET và giảng viên trong công ty cũng như các học viên thường xuyên và nhân viên kỹ thuật của công ty; phổ biến các chương trình đào tạo ban đầu và tiếp theo cho các cơ sở đào tạo nghề khác và hệ thống đào tạo nghề ở Việt Nam.

Về vấn đề này, các chuyên gia và nhà thiết kế chương trình đào tạo trong nước và quốc tế về Công nghiệp 4.0 đã phát triển mô-đun đào tạo "Lập trình hệ thống cơ điện tử" tích hợp CNTT-an ninh và các giáo viên được đào tạo thêm của LILAMA 2 và các học viện TVET đối tác khác của Chương trình cải cách chương trình TVET của Việt Đức trong Việt Nam với tư cách là giảng viên chính và là nhân tố của hệ thống TVET. Các năng lực trung gian trong mô-đun định hướng dựa trên trình độ Công nghiệp 4.0 mới nhất của pháp lệnh đào tạo sửa đổi của Đức để đào tạo các nghề điện tử và cơ điện tử trong 3,5 năm và đáp ứng nhu cầu trình độ Công nghiệp 4.0 của các công ty trong nước và quốc tế tại Việt Nam.

Mô-đun đào tạo về Công nghiệp 4.0 tuân thủ Thông tư 03/2017 của Bộ Lao động Thương binh và Xã hội và giống với cấu trúc của chương trình đào tạo ban đầu trình độ trung cấp và cao đẳng, được xây dựng trong khung Chương trình đổi mới đào tạo nghề tại Việt Nam. Chúng có thể tải xuống miễn phí tại www.tvet-vietnam.org. Tài liệu giảng dạy và học tập được xây dựng dựa trên các chương trình này và có thể được cung cấp như một mô-đun đào tạo tự chọn cho sinh viên tốt nghiệp đại học của Kỹ thuật viên Điện tử Công nghiệp, Kỹ thuật viên Cơ điện tử và Kỹ thuật viên Điện tử về Năng lượng và Công nghệ Xây dựng. Hơn nữa, mô-đun có thể được cung cấp cho các kỹ thuật viên và kỹ sư trong ngành khi được đào tạo thêm về các ứng dụng Công nghiệp 4.0 ứng dụng trong lĩnh vực điện tử, cơ điện tử và tự động hóa.

Các giáo viên TVET nên thực hiện mô-đun đào tạo "Lập trình Hệ thống Cơ điện tử" là học kết hợp hoặc học kết hợp, trong đó học viên học qua phương tiện điện tử và trực tuyến cũng như giảng dạy và đào tạo trực tiếp truyền thống.

Thay mặt các đối tác hợp tác, chúng tôi xin chân thành cảm ơn sự hỗ trợ của các bên trong việc phát triển mô-đun đào tạo "Lập trình Hệ thống Cơ điện tử" và chúc việc triển khai và phổ biến thành công.

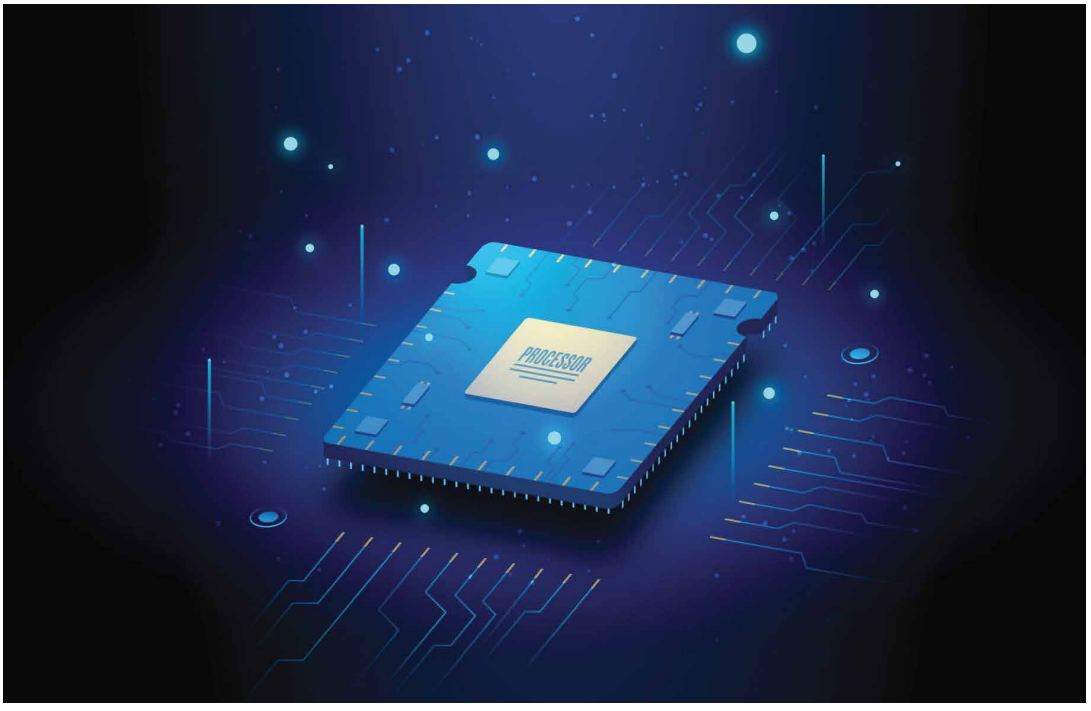
Tiến sĩ Juergen Hartwig
Giám đốc
Chương trình "Cải cách dạy nghề ở Việt Nam"

MỤC LỤC

MỤC LỤC

CÁC TỪ VIẾT TẮT

BMZ	Bộ Hợp tác và Phát triển Kinh tế Liên bang
DPP	Quan hệ đối tác phát triển với khu vực tư nhân
DVET	Tổng cục giáo dục nghề nghiệp và đào tạo
GIZ	Deutsche Gesellschaft für Internationale Zusammenarbeit GmbH
I 4.0	Công nghiệp 4.0
LILAMA 2	Trường Cao đẳng Công nghệ Quốc tế LILAMA 2
TVET	**Giáo dục và Đào tạo Kỹ thuật và Dạy nghề**



Hình minh họa 1: Bộ xử lý vi mạch thực tế

A. CHƯƠNG TRÌNH MÔ ĐUN

(Theo Thông tư số 03/2017/TT-BLĐTBXH Ngày 1 Tháng 3/2017 của Bộ Lao động - Thương binh và Xã hội)

Tên mô-đun: Công nghiệp 4.0 - Lập trình hệ thống cơ điện tử

Mã mô-đun: MD I 4.0 - Lập trình

Thời gian dự kiến: 320 giờ

Lý thuyết: 80h;

Thực hành/Phòng thí nghiệm/Thảo luận/Bài tập: 220h;

Kiểm tra/Đánh giá: 20h

I. Vị trí, tính chất của mô đun:

- Vị trí:

+ Mô-đun đào tạo trình độ công nghiệp 4.0 – Trình độ cao đẳng quốc tế. Yêu cầu tuyển sinh: Bằng tốt nghiệp cao đẳng quốc gia về nghề điện tử hoặc cơ điện tử hoặc 3 năm đã được chứng minh kinh nghiệm làm việc chuyên nghiệp trong lĩnh vực và ngành

- Tính chất:

+ Mô-đun định hướng theo tiêu chuẩn đào tạo và kiểm tra của Đức. Nó cung cấp các chủ đề đào tạo và năng lực công nghiệp 4.0 tiên tiến cũng như đào tạo với trình độ chuyên môn cao hơn trong các lĩnh vực điện tử, tự động hóa và cơ điện tử. Mô-đun được cấu trúc thành 4 đơn vị đào tạo mà nội dung được xây dựng nâng cấp và nối tiếp với nhau. Đây có thể được cung cấp như mô-đun đào tạo bổ sung cho sinh viên tốt nghiệp của các chương trình đào tạo nghề ban đầu cũng như được cung cấp cho ngành công nghiệp trong đào tạo ngắn hạn lai cho kỹ thuật viên, kỹ sư và đào tạo viên doanh nghiệp.

+ Các học viên học thuật ngữ I 4.0 chuyên nghiệp, đạt được các kỹ năng kiến thức về lập trình trong môi trường công nghiệp 4.0 - một mạng lưới ngày càng tăng gồm các nhà máy công nghiệp và một sự thôi thúc ngày càng tăng nhằm lưu trữ và hiển thị dữ liệu sản xuất theo một cách thức có ý nghĩa. Các học viên học cách triển khai các công việc và nhiệm vụ như các kỹ thuật viên điện tử và cơ điện tử trong tương lai.

+ Các chủ đề như thu thập dữ liệu, lập trình các mô-đun phần mềm với ngôn ngữ hướng đối tượng được xử lý chi tiết. Các học viên học các khả năng khác nhau để tạo ra các chương trình đo lường và kiểm soát với vi điều khiển. Qua đó học viên tích hợp các chương trình vào môi trường Công nghiệp 4.0. Học viên học các khả năng khác nhau để lưu trữ dữ liệu trong cơ sở dữ liệu, để đánh giá và hình dung dữ liệu trong bảng điều khiển để xử lý dữ liệu. Các khía cạnh của tài liệu phần mềm chính xác cũng

như việc tạo ra các yêu cầu và thông số kỹ thuật, thiết kế chính xác cơ sở dữ liệu và các khía cạnh bảo mật dữ liệu cũng được đưa vào xem xét.

- Ý nghĩa và vai trò của môn học/mô đun:

+ Sau khi hoàn thành mô-đun này, các học viên sẽ có thể xác định các ứng dụng có thể sử dụng cho Công nghiệp 4.0, từ đó đưa ra các đề xuất cho các giải pháp và thực hiện chúng. Học viên sẽ có thể hình thành một giao diện giữa phát triển và sản xuất phần mềm và liên kết cả hai thế giới một cách đồng nhất.

II. Mục tiêu của mô đun:

- Về kiến thức: Các học viên biết và quen thuộc với:

+ những phát triển lịch sử trong lĩnh vực số hóa

+ các định nghĩa về các thuật ngữ quan trọng nhất trong lĩnh vực số hóa / công nghiệp 4.0

+ Sự phân biệt rõ ràng giữa Công nghiệp 3.0 và Công nghiệp 4.0

+ các kịch bản ứng dụng khác nhau cho Hệ thống vật lý mạng

+ ngôn ngữ lập trình hướng đối tượng phù hợp với CNTT mới

+ cấu trúc và nội dung của yêu cầu và thông số kỹ thuật chức năng

+ Phương pháp tiếp cận có phương pháp để kiểm tra và ghi chép các mô-đun phần mềm

+ những nguy hiểm và rủi ro đối với các hệ thống nối mạng và các biện pháp bảo mật đầy đủ

+ hệ thống cơ sở dữ liệu quan hệ và học ngôn ngữ cơ sở dữ liệu

+ các khu vực xử lý và ứng dụng của vi điều khiển trong môi trường I4.0

+ các tùy chọn để lưu trữ và lưu giữ dữ liệu cũng như để trực quan hóa và trình bày các biến đo lường liên quan đến hệ thống trong bảng điều khiển

- Về kỹ năng: Các học viên có thể:

+ phân tích các hệ thống và thiết bị kỹ thuật và xác định tiềm năng thu thập dữ liệu và tích hợp công nghệ cảm biến bổ sung

+ xây dựng, sửa đổi và thử nghiệm các hệ thống nối mạng

+ vận hành các hệ thống nối mạng và thực hiện công việc bảo trì và tối ưu hóa

+ phân tích một vấn đề kỹ thuật và phát triển một giải pháp có tính đến các điều kiện hiện hành.

+ điều chỉnh và lên tài liệu các mô-đun phần mềm và tích hợp chúng vào các hệ thống hiện có.

+ thiết kế kế hoạch thử nghiệm và kiểm tra các mô-đun phần mềm sửa đổi trong điều kiện hoạt động

+ thực hiện phân tích lỗi / sai phạm hệ thống và chuẩn bị tài liệu toàn diện về toàn bộ quy trình

+ tích hợp các biện pháp bảo mật kỹ thuật vào hệ thống CNTT, thông báo cho người sử dụng các hệ thống này về hành vi chính xác và ghi lại các biện pháp được thực hiện phù hợp với các yêu cầu hoạt động và pháp lý.

+ kiểm tra hiệu quả của các biện pháp bảo mật đã thực hiện, giám sát việc tuân thủ các quy định bảo vệ dữ liệu và báo cáo các sự cố liên quan đến bảo mật

+ phát triển và thực hiện các ý tưởng phát thảo cho sao lưu dữ liệu

+ lựa chọn đúng quy trình cho phù hợp với tình hình và hoàn thành các bài kiểm tra phần mềm trước khi triển khai trong hệ thống sản xuất

+ tạo khả năng giám sát quy trình và dữ liệu tuân thủ các nguyên tắc bảo mật của công ty

+ sử dụng các thư viện phần mềm làm sẵn và sửa đổi chúng theo tình hình, đồng thời tuân thủ các nguyên tắc hoạt động của phiên bản phần mềm

- Về năng lực tự chủ và trách nhiệm: Các học viên có thể:

+ tự thông báo độc lập về các công nghệ mới nổi và có được kiến thức cần thiết để áp dụng trong bối cảnh công nghiệp.

+ tìm kiếm các thư viện phần mềm có sẵn và hoạt động tự do và để điều chỉnh chúng cho các nhiệm vụ nhất định

+ Xác định các ứng dụng có thể cho các ứng dụng Công nghiệp 4.0 trong công ty của họ, để đưa ra các đề xuất cho các giải pháp và thực hiện chúng.

III. Nội dung mô-đun:

1. Phân loại nội dung chung và phân bổ thời gian:

STT.	Đơn vị đào tạo mô-đun	Thời gian dự kiến (giờ)			
		Tất cả	Thuyết	Thực hành/ Phòng thí nghiệm / Thảo luận / Phân công	Kiểm tra / Đánh giá
1	**1. Lập trình hướng đối tượng** 1.1 Cơ bản về lập trình 1.2 Ngôn ngữ mô hình thống nhất (UML) 1.3 Phân tích nhiệm vụ và tìm giải pháp 1.4 Kiểm tra và triển khai phần mềm 1.5 Thực hành tiếng Anh	80	20	55	5
2	**2. Lập trình vi điều khiển** 1.1 Cơ bản về lập trình vi điều khiển 1.2 Lập trình vi điều khiển và điều khiển phần cứng 1.3 Tạo chương trình đo lường 1.4 Phương thức truyền thông và trao đổi dữ liệu 1.5 Thực hành tiếng Anh	80	20	55	5
3	**3. Hệ thống Datatabase** 1.1 Cơ bản về hệ thống cơ sở dữ liệu 1.2 Xử lý cơ sở dữ liệu ở cấp độ máy chủ 1.3 Chương trình người dùng với cơ sở dữ liệu 1.4 Thực hành tiếng Anh	80	20	55	5
4	**4. Trực quan hóa dữ liệu với Bảng điều khiển** 1. Cơ bản về bảng điều khiển 2. Những khả năng cho trực quan hóa dữ liệu 3. Tạo và triển khai bảng điều khiển 4. Thực hành tiếng Anh	80	20	55	5
	Tổng số giờ:	320	80	220	20

2. Nội dung chi tiết

Bài 1: Lập trình theo hướng đối tượng Thời gian: 80 giờ

Mục tiêu: Các học viên:

- biết sự khác biệt giữa lập trình chức năng và đối tượng
- Biết các thuật ngữ quan trọng nhất của lập trình hướng đối tượng
- Học được một ngôn ngữ lập trình hướng đối tượng
- có thể sử dụng thư viện và điều chỉnh các lớp có sẵn theo nhu cầu của họ
- Có thể phân tích một vấn đề kỹ thuật và phát triển một giải pháp có tính đến các điều kiện hiện hành.
- Có thể điều chỉnh và ghi lại các mô-đun phần mềm và tích hợp chúng vào các hệ thống hiện có.
- Thiết kế kế hoạch thử nghiệm và kiểm tra các mô-đun phần mềm sửa đổi trong điều kiện hoạt động
- Thực hiện phân tích lỗi / lỗi có hệ thống và chuẩn bị tài liệu toàn diện về toàn bộ quy trình

Nội dung:

1.1 Cơ bản về lập trình

1.1.1 Nguyên tắc cơ bản của lập trình
- 1.1.1.1 Thuật ngữ lập trình cơ bản
- 1.1.1.2 Những bước cơ bản về lập trình
- 1.1.1.3 Giới thiệu về Biến, Mảng, Điều kiện, Vòng lặp và Hàm
- 1.1.1.4 Sự khác biệt giữa IEC61131 và ngôn ngữ lập trình tiêu chuẩn
- 1.1.1.5 Tìm hiểu môi trường phát triển tích hợp (IDE)
- 1.1.1.6 Tài liệu phần mềm và nhận xét

1.1.2 Điều chỉnh mô-đun phần mềm
- 1.1.2.1 Bao gồm các mô-đun từ các thư viện được xác định trước hoặc bên ngoài
- 1.1.2.2 Hiểu mã chương trình
- 1.1.2.3 Lập ài liệu và nhận xét mã chương trình
- 1.1.2.4 Hiểu việc gọi chức năng cơ chế và lập trình mô-đun
- 1.1.2.5 Nội suy các chức năng tài liệu và chuyển giao tham số

1.1.3 Những điều cơ bản về lập trình hướng đối tượng
- 1.1.3.1 Lập trình chức năng so với lập trình hướng đối tượng
- 1.1.3.2 Sự khác biệt giữa các lớp và đối tượng
- 1.1.3.3 Quyền truy cập khía cạnh bảo mật cho các lớp
- 1.1.3.4 Tạo trình xây dựng/cấu trúc cho các lớp
- 1.1.3.5 Tạo lớp cơ sở để thừa kế
- 1.1.3.6 Hiểu được cấu trúc và cơ chế kế thừa
- 1.1.3.7 Sử dụng các cấu trúc hướng đối tượng quan trọng nhất
- 1.1.3.8 Cài đặt và sử dụng thư viện

1.1.4 Khái niệm mở rộng
- 1.1.4.1 Thực hiện đầu vào và đầu ra tệp bằng cách sử dụng cấu trúc hướng đối tượng
- 1.1.4.2 Chuẩn bị và trực quan hóa cục bộ dữ liệu bằng các phương pháp hướng đối tượng

1.1.4.3 Tạo giao diện người dùng đồ họa

1.2 Ngôn ngữ mô hình thống nhất (UML)

1.2.1 Cơ bản về UML
1.2.2 Sơ đồ lớp để mô tả các yêu cầu và tài liệu phần mềm
1.2.2.1 Đại diện cho các thuộc tính và phương pháp
1.2.2.2 Đại diện cho các khả năng đóng gói dữ liệu
1.2.3 Hiển thị và áp dụng quan hệ lớp học
1.2.3.1 Kết hợp
1.2.3.2 Tổng hợp
1.2.3.3 Các thành phần
1.2.3.4 Di sản
1.2.4 Hiển thị cây thừa kế cho tài liệu phần mềm
1.2.5 Sơ đồ trình tự để mô tả các quy trình giao tiếp của phần mềm (hệ thống PC và cơ điện tử)
1.2.6 Sử dụng Sơ đồ Trường hợp cho tài liệu cấp cao về các trường hợp sử dụng
1.2.7 Thực hiện sơ đồ UML
1.2.7.1 Tạo các lớp sử dụng sơ đồ UML, cũng như lấy sơ đồ UML từ các lớp hiện có
1.2.7.2 Thực hiện sơ đồ trình tự cho các tác vụ được xác định
1.2.7.3 Tạo Sơ đồ Trường hợp Sử dụng cho các ứng dụng mẫu

1.3 Phân tích nhiệm vụ và tìm giải pháp

1.3.1 Phân tích các đơn đặt hàng kỹ thuật và phát triển các giải pháp
1.3.1.1 Phân tích các yêu cầu của khách hàng liên quan đến chức năng cần thiết
1.3.1.2 Làm rõ thông số kỹ thuật trao đổi với khách hàng (nội bộ / bên ngoài)
1.3.1.3 Phân tích các quy trình, giao diện và điều kiện môi trường cũng như trạng thái ban đầu của hệ thống, xác định và ghi lại các yêu cầu cho các mô-đun phần mềm
1.3.1.4 Phân tích và chuẩn bị luồng dữ liệu
1.3.1.5 Tạo đặc điểm kỹ thuật yêu cầu

1.4 Kiểm tra phần mềm và triển khai

1.4.1 Làm rõ mô hình - V
1.4.2 Dự thảo kế hoạch thử nghiệm theo quy trình thử nghiệm và phát hành vận hành, đặc biệt là xác định quy trình, tiêu chuẩn và giá trị giới hạn của các thông số vận hành và tạo dữ liệu thử nghiệm
1.4.3 Tạo kế hoạch thử nghiệm dựa trên các yêu cầu (bảng thông số kỹ thuật, thông số kỹ thuật pháp lý hoặc hoạt động)
1.4.4 Mô phỏng các điều kiện môi trường kỹ thuật
1.4.5 Kiểm tra mô-đun phần mềm
1.4.6 Kiểm tra hệ thống và kiểm tra thành phần Thực hiện các thử nghiệm trong hệ thống theo thông số vận hành
1.4.7 Khắc phục sự cố phần mềm

1.4.7.1 Phân tích trục trặc, khắc phục sự cố có hệ thống, có thể điều chỉnh các yêu cầu và thông số kỹ thuật với tài liệu
1.4.8 Tài liệu và Phát hành
1.4.8.1 Những điều cơ bản về tài liệu mã và phần mềm
1.4.8.2 Phiên bản phần mềm
1.4.8.3 Chiến lược phát hành

Bài 2: Lập trình vi điều khiển Thời gian: 80 giờ

Mục tiêu: Các học viên:

- Biết sự khác biệt giữa sử dụng vi điều khiển và PLC
- Có thể điều khiển bộ truyền động và cảm biến và kết nối chúng với vi điều khiển
- có thể lưu các giá trị đo được trong tệp nhật ký
- Có thể tìm kiếm và tìm lỗi một cách có cấu trúc
- Có thể vận hành, cấu hình và tham số bộ vi xử lý
- Tạo ra các máy nhà nước để thực hiện các chương trình đo lường
- Có thể tích hợp các mô-đun phần mềm vào một chương trình trình tự
- Đã quen thuộc với khả năng trao đổi dữ liệu giữa các bộ vi xử lý
- Biết khả năng kết nối bộ vi xử lý với các hệ thống CNTT cấp cao hơn

Nội dung:

2.1 Cơ bản về lập trình vi điều khiển
2.1.1 Cơ bản về vi điều khiển
2.1.2 thuật ngữ cơ bản trong lập trình
2.1.3 Sự khác biệt giữa vi điều khiển và SPS
2.1.4 Hệ điều hành cho vi điều khiển

2.2 Lập trình vi điều khiển và điều khiển phần cứng
2.2.1 Những điều cơ bản về lập trình vi điều khiển
2.2.1.1 Cài đặt các gói phần mềm cần thiết
2.2.1.2 Lập trình vi điều khiển để điều khiển các thiết bị ngoại vi
2.2.1.3 Kiến trúc phần mềm trong môi trường vi điều khiển
2.2.2 Điều khiển phần cứng
2.2.2.1. Kiểm soát phần cứng bên ngoài (bộ truyền động và cảm biến)
2.2.2.2. Sử dụng điều chế độ rộng xung để điều khiển phần cứng
2.2.2.3. Sử dụng ADC (analog sang công cụ chuyển đổi kỹ thuật số) để đọc tín hiệu analog
2.2.2.4. Chuyển đổi đo lường từ giá trị kỹ thuật số sang số lượng vật lý
2.2.2.5. Tạo dữ liệu nhật ký với tín hiệu cảm biến

2.3. Tạo các chương trình đo lường
2.3.2. Giới thiệu về state machine
2.3.2.1. Nguyên tắc cơ bản của state machine
2.3.2.2. Triển khai state machine cho vi điều khiển
2.3.2.3. Thiết kế phần mềm của một chương trình trình tự thông qua các state machine
2.3.3. Chương trình đo lường

2.3.3.1. Thực hiện chương trình đo lường để tự động thu thập các giá trị cảm biến
2.3.3.2. Tạo chương trình lưu giá trị cảm biến trong tệp nhật ký
2.3.3.3. Mở rộng theo khả năng báo động để nhận được báo động nếu vượt quá giá trị giới hạn
2.3.3.4. Đồ họa thay thế của dữ liệu log

2.4. Phương thức truyền thông và trao đổi dữ liệu

2.4.2. Những lệnh yêu cầu HTTP Request
2.4.2.1. Những điều cơ bản về giao tiếp dựa trên web
2.4.2.2. Thiết lập phía máy chủ của một dịch vụ web để ghi lại và đánh giá các yêu cầu
2.4.2.3. Tạo chương trình giao tiếp từ phía khách hàng client bằng yêu cầu HTTP request

2.4.3. Giao tiếp với Client Server
2.4.3.1. Những điều cơ bản cho kiến trúc giao tiếp cơ sở với Client Server
2.4.3.2. Tạo trình truyền thông để thu thập dữ liệu và sao lưu dữ liệu từ phía Client bằng cách sử dụng các kết nối web socket
2.4.3.3. Tạo ra một chương trình đo lường và giao tiếp phía client, để thiết lập kết nối và truyền dữ liệu

2.4.4. Giao tiếp Publisher và Subscriber (người đăng ký và nhà cung cấp)
2.4.4.1. Cơ bản về kiến trúc truyền thông Publisher và Subscriber
2.4.4.2. Thiết kế kiến trúc truyền thông
2.4.4.3. Tạo mô-đun đo lường và sao lưu phía Publisher và Subscriber

2.4.5. Thực hiện một chương trình truyền thông và trao đổi dữ liệu của hai vi điều khiển với các khối chức năng được cung cấp

Bài 3: Hệ thống cơ sở dữ liệu Thời gian: 80 giờ

Mục tiêu: Các học viên:

- Hiểu biết cơ bản về cơ sở dữ liệu và các khái niệm hệ thống liên quan
- Có thể thiết lập và cấu hình máy chủ cơ sở dữ liệu
- Biết các trình tự lệnh cơ bản để thao tác cơ sở dữ liệu
- Có thể phát triển các chương trình ghi các giá trị đo được vào cơ sở dữ liệu và đọc chúng từ cơ sở dữ liệu

Nội dung:

3.1 Cơ bản về hệ thống cơ sở dữ liệu

3.1.1 Những điều cơ bản cơ sở dữ liệu
3.1.1.1 Khái niệm cơ bản về cơ sở dữ liệu
3.1.1.2 Các thành phần phần mềm cần thiết (máy chủ cơ sở dữ liệu)
3.1.1.3 Cấu hình thủ công của thiết đặt máy chủ cơ sở dữ liệu
3.1.1.4 Tạo cơ sở dữ liệu đang xem xét các khía cạnh bảo mật

3.1.2 Nguyên tắc lý thuyết cho cơ sở dữ liệu
3.1.2.1 Cấu trúc cơ sở dữ liệu
3.1.2.2 Giới thiệu về mô hình cơ sở dữ liệu quan hệ
3.1.2.3 Tìm hiểu Sơ đồ Quan hệ Thực thể cho thiết kế cơ sở dữ liệu và tài liệu

3.2 **Xử lý cơ sở dữ liệu ở cấp độ máy chủ**

3.2.1 Thao tác cơ sở dữ liệu

3.2.1.1 Tạo cơ sở dữ liệu trên máy chủ cơ sở dữ liệu

3.2.1.2 Tạo chính xác bảng và mục cơ sở dữ liệu cho cơ sở dữ liệu quan hệ

3.2.1.3 Tìm hiểu các lệnh cơ bản để thao tác cơ sở dữ liệu

3.2.1.4 Thực hiện các lệnh gia nhập để tham gia bảng trong cơ sở dữ liệu

3.3 **Chương trình người dùng với cơ sở dữ liệu**

3.3.1 Tạo cơ sở dữ liệu

3.3.1.1 Tìm hiểu các lớp và chức năng để kết nối với máy chủ cơ sở dữ liệu

3.3.1.2 Tạo chương trình người dùng để tạo cơ sở dữ liệu trên máy chủ cơ sở dữ liệu

3.3.1.3 Triển khai các chương trình người dùng để lưu trữ và đọc dữ liệu trên máy chủ cơ sở dữ liệu

3.3.2 Chương trình người dùng cơ sở dữ liệu

3.3.2.1 Phân tích trạng thái hiện tại của một hệ thống cơ điện tử hiện có và phát triển một khái niệm để làm cho hệ thống I4.0 phù hợp (cảm biến và bộ truyền động)

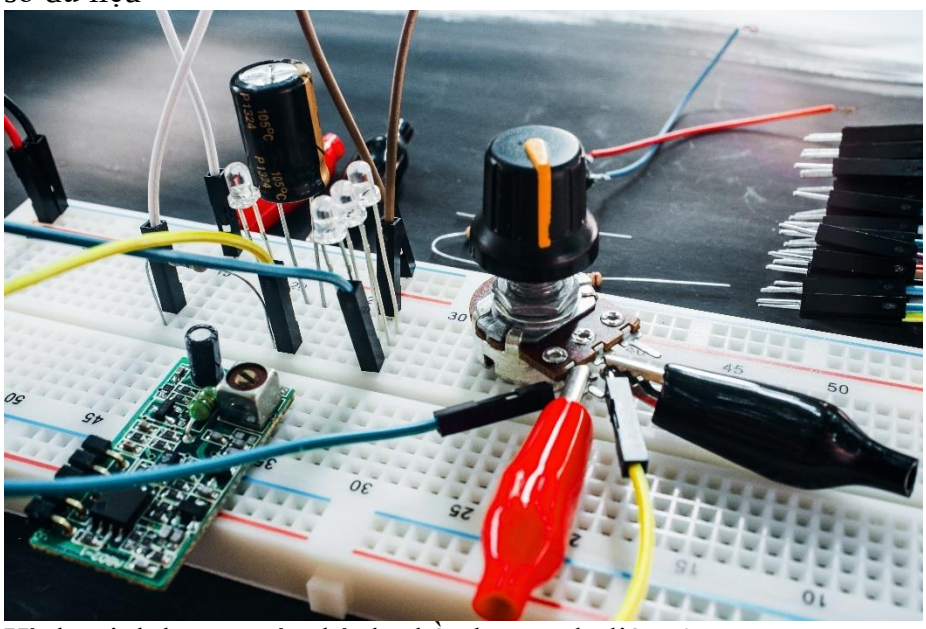
3.3.2.2 Tạo cấu trúc cơ sở dữ liệu thông qua sơ đồ ER

3.3.2.3 Tạo UML Use Case, Class ,Sơ đồ Trình tự để nắm bắt các yêu cầu của hệ thống

3.3.2.4 Thực hiện cấu trúc phần mềm sử dụng UML để liên kết chương trình đo lường với cơ sở dữ liệu

3.3.2.5 Lập trình một chương trình đo lường để lưu trữ các giá trị đo được trong cơ sở dữ liệu

3.3.2.6 Tạo một chương trình đánh giá để đọc các giá trị được đo cục bộ từ cơ sở dữ liệu



Hình minh họa 2: các thành phần bo mạch điện tử

Bài 4: Trực quan hóa dữ liệu với bảng điều khiển Thời gian:80 giờ

Mục tiêu: Các học viên:

- Tìm hiểu cách xử lý một lượng lớn dữ liệu
- Có thể sử dụng phương pháp trực quan hóa phù hợp cho các yêu cầu khác nhau
- Quen thuộc với các ứng dụng bảng điều khiển lập trình kết hợp với cơ sở dữ liệu
- Có thể triển khai bảng điều khiển trong mạng cục bộ và trên các máy chủ có thể truy cập công khai

Nội dung:

4.1 Cơ bản về bảng điều khiển

4.1.1 Xác định cách sử dụng và nhiệm vụ của bảng điều khiển
4.1.2 Nền tảng lý thuyết cho việc tạo ra các ứng dụng web
4.1.3 So sánh các phương pháp sáng tạo khác nhau
4.1.4 Hướng dẫn thiết kế cho bảng điều khiển
4.1.5 Giới hạn bảng điều khiển
4.1.6 Giải thích về quy trình tạo bảng điều khiển
4.1.6.1 So sánh các giải pháp bảng điều khiển mẫu với các giải pháp tự thực hiện
4.1.7 Các khía cạnh bảo mật cho bảng điều khiển
4.1.7.1 Triển khai nội bộ so với triển khai internet
4.1.8 Cài đặt các thư viện cần thiết
4.1.9 Giải thích về các khái niệm, chức năng và lớp học lập trình cần thiết để hiển thị bảng điều khiển
4.1.10 Làm rõ các thành phần lập trình dựa trên web cần thiết để lập trình bảng điều khiển

4.2 Các phương pháp để trực quan hóa dữ liệu

4.2.1 Nguyên tắc cơ bản của trực quan hóa dữ liệu
4.2.1.1 Hiển thị các loại sơ đồ khác nhau với các khu vực ứng dụng
4.2.1.2 Định nghĩa về cấu trúc dữ liệu cần thiết để sử dụng các loại sơ đồ
4.2.1.2.1 Dữ liệu liên tục so với các tính năng
4.2.1.3 Thay thế của sơ đồ trong bảng điều khiển
4.2.1.4 Chuyển đổi dữ liệu từ nguồn dữ liệu sang dữ liệu chìm trong sơ đồ
4.2.1.4.1 Chuẩn bị dữ liệu, lọc trước, lựa chọn nguồn

4.3 Tạo và triển khai bảng điều khiển

4.3.1 Thực hiện logic chương trình phù hợp để tạo bảng điều khiển
4.3.1.1 Thực hiện bảng điều khiển tĩnh mà không cần cập nhật dữ liệu
4.3.1.2 Thực hiện bảng điều khiển động với khoảng thời gian cập nhật
4.3.1.3 Xem xét lưu lượng mạng
4.3.2 Xem xét các tùy chọn xác thực người dùng cho các khía cạnh bảo mật
4.3.2.1 Tên người dùng, mật khẩu
4.3.3 Tạo chương trình trực quan hóa

4.3.3.1 Tạo chương trình đo lường để lưu trữ dữ liệu cảm biến trong cơ sở dữ liệu

4.3.4 Triển khai bảng điều khiển để liên tục đọc giá trị và hiển thị đồ họa các giá trị đo được từ cơ sở dữ liệu

Hình minh họa 3: đồ thị tối ưu hóa dữ liệu

IV. Yêu cầu triển khai module

1. Phòng học chuyên môn hóa, nhà xưởng

Phòng học chuyên môn hóa:

- Cung cấp quyền truy cập và nơi làm việc không bị hạn chế, tuân thủ các quy định an toàn lao động, hướng dẫn kỹ thuật và quy định pháp lý.
- Cung cấp đủ không gian làm việc cho số lượng học viên cũng như cơ sở hạ tầng CNTT hoạt động với máy trạm PC với phần mềm lập trình và kết nối internet phù hợp

Nhà xưởng:

- Xưởng cho các công trình dự án công nghiệp điện tử / cơ điện tử
  - Cung cấp quyền truy cập và nơi làm việc không bị hạn chế, tuân thủ các quy định an toàn lao động, hướng dẫn kỹ thuật và quy định pháp lý.
  - Cung cấp đủ không gian làm việc và nơi làm việc máy móc cho số lượng học viên
- Phòng vệ sinh, phòng tắm và phòng thay đồ không bị hạn chế, riêng cho phụ nữ và nam giới

2. Thiết bị và máy móc:

- Máy công cụ cố định
- Công cụ đo analog và kỹ thuật số
  - Công cụ đo chiều dài (thước đo)
  - Công cụ đo góc (thước đo)
  - Đồng hồ đo thử nghiệm
  - Máy kiểm tra điện áp lưỡng cực,máy đo vạn năng,
  - Kẹp hiện tại, máy kiểm tra cài đặt

Thiết bị

- Các nhà máy công nghiệp (hệ thống và nhà máy hoàn chỉnh để sản xuất hoặc quy trình) như nhà máy tự động hóa quy trình để sản xuất chất lỏng và chất, hệ thống chế biến chai, bàn thu gom, trạm sản xuất, trạm thử nghiệm
- Kỹ thuật quy trình, kỹ thuật sản xuất, người biểu tình kỹ thuật điều khiển cũng như người biểu tình I4.0, chức năng và có thể truy cập để lắp đặt công nghệ cảm biến vi điều khiển.

Khác

- Quản trị viên truy cập vào PC và vi điều khiển để thực hiện cài đặt phần mềm cần thiết.

3. Học liệu, dụng cụ, nguyên vật liệu:

- Dụng cụ cầm tay
- Kìm (kìm nhăn, máy cắt bên, kìm nhọn, kìm cắt dây)
- Phân loại Wrench (các) (ổ cắm lục giác / lục giác)
- Dao cáp, kéo
- Bảng Pegboards treo dụng cụ
- Các thành phần công nghiệp của công nghệ tự động hóa
  - Kệ lắp ráp linh hoạt làm bằng hồ sơ nhôm để xây dựng các nhiệm vụ con trong công nghệ tự động hóa
  - Các thành phần khí nén và điện khí nén
  - Các thành phần trong thủy lực và điện thủy lực
  - Các ổ điện như động cơ không đồng bộ ba pha, động cơ servo, động cơ bước
  - Các đơn vị nhỏ gọn PLC (có thể mạng và với AI / AO), PLC mô-đun (có thể mạng và với AI / AO), các đơn vị cung cấp năng lượng tùy thuộc vào kích thước phụ tải
  - Các mô-đun PLC và tài liệu mạng thông qua ASi và PROFIBUS, PROFINET và Ethernet, nếu cần thiết, và kết nối địa chỉ thiết bị
  - Cổng liên lạc Router và IOT để kết nối với công nghiệp 4.0
  - Máy tính xách tay hoặc máy tính để bàn mạnh mẽ, phần mềm người dùng để vẽ và mô phỏng, phần mềm PLC
  - Vi điều khiển với hệ điều hành, mô-đun WLan, ngoài ra còn kết nối mạng, ngoại vi để điều khiển bộ truyền động hoặc đọc trong cảm biến, kết nối nguồn
  - Bộ truyền động có khả năng điều khiển, tương thích trong phạm vi điện áp, giai đoạn công suất
  - Vi điều khiển phù hợp cảm biến analog, dải điện áp tương thích cần thiết, các loại khác nhau để thực hiện ứng dụng ví dụ trong nhiệt độ, khoảng cách, độ ẩm, gia tốc
  - Các thành phần mạng để kết nối vi điều khiển và cơ sở hạ tầng CNTT (cáp mạng chức năng, bộ định tuyến Wlan hoặc công tắc lan, máy chủ)
  - Cung cấp năng lượng cho vi điều khiển và bộ truyền động thích ứng với dữ liệu hiệu suất được chỉ định tương ứng

- Vật liệu phụ trợ
  - Vật liệu phụ trợ và vận hành cho công việc phân công và bảo dưỡng phù hợp với các bài tập thực hành và yêu cầu công việc, bao gồm cả các bài kiểm tra
- Vật tư tiêu hao

- cung cấp cho việc phân công công việc theo các bài tập thực hành và yêu cầu công việc, bao gồm cả các bài kiểm tra
- Cáp kết nối cho vi điều khiển và kỹ thuật sản xuất

- Thiết bị bảo hộ
  - Thiết bị an toàn cá nhân (PSA)
  - quần áo an toàn lao động, giày an toàn, nắp trực quan, bảo vệ thính giác

- Tài liệu kỹ thuật và sách bảng
  - Cẩm nang và sách giáo khoa Cơ điện tử, Tin học sổ tay

- Tài liệu kỹ thuật
  - Bản vẽ một phần, nhóm và chung, bản vẽ bố trí
  - Mô tả cài đặt, kế hoạch bảo trì, mô tả chức năng
  - Sơ đồ mạch, sơ đồ hệ thống dây điện, kế hoạch làm việc
  - bảng giá trị danh nghĩa, báo cáo đo lường, báo cáo đánh giá

- Phần mềm
  - Phần mềm người dùng để vẽ và mô phỏng,
  - Phần mềm PLC (TIA-Portal hoặc Step 7)
  - Phần mềm mô phỏng - Công nghệ tự động hóa
  - Phần mềm học tập để tự học
  - Ngôn ngữ lập trình hướng đối tượng (ví dụ: Python)
  - Thư viện cần thiết
  - Phần mềm tạo kết nối SSH với vi điều khiển (ví dụ: Putty)
  - Môi trường phát triển tích hợp Ngôn ngữ lập trình tương thích
  - Ngôn ngữ Cơ sở dữ liệu quan hệ
  - Hệ thống quản lý cơ sở dữ liệu quan hệ (ví dụ: MySql)
  - Máy chủ web (ví dụ: XAMPP)
  - Phần mềm hoặc thư viện để tạo ứng dụng bảng điều khiển (ví dụ: Dash và Thingsboard)

4. Các yêu cầu và điều kiện khác: Không có

V. Nội dung và phương pháp đánh giá

1. Nội dung:

- **Kiến thức:**
  - Để biết và tuân thủ các hướng dẫn pháp lý và hoạt động để đảm bảo chất lượng cũng như bảo vệ dữ liệu và bảo mật CNTT khi làm việc với và trong các hệ thống kỹ thuật số - Để mô tả Công nghiệp 4.0 và số hóa các quy trình sản xuất cũng như bảo vệ dữ liệu và bảo mật CNTT trong sản xuất
  - Tự động hóa máy công cụ và hệ thống sản xuất.
  - Để biết các nhà máy và hệ thống sản xuất linh hoạt cũng như các hệ thống xử lý và robot cho các nhà máy sản xuất linh hoạt và gán chúng cho mục đích dự định

- Để phân tích thông tin cần thiết cho việc xử lý đơn đặt hàng, cũng từ phương tiện kỹ thuật số và bằng tiếng Anh
- Mô tả các yêu cầu và mục tiêu kinh doanh của sản xuất và tính toán các thông số hoạt động
- để phân tích ảnh hưởng đến quá trình sản xuất và tính đến chúng trong quy hoạch
- Liên kết kiến thức về nhà máy sản xuất và chế biến với kiến thức I4.0 để kết nối và trích xuất dữ liệu quy trình

- **Kỹ năng**
  - Lập kế hoạch đặt hàng sản xuất cụ thể cho khách hàng cũng như xem xét các khía cạnh công nghệ, kinh doanh, môi trường và an toàn và bảo mật CNTT
  - Duy trì, phân tích, lưu và lưu trữ dữ liệu
  - Để biết và sử dụng các hệ thống hỗ trợ, mô phỏng, chẩn đoán và trực quan hóa
  - Thiết lập quy trình sản xuất với các công cụ máy móc thông thường và được điều khiển bằng số và / hoặc hệ thống sản xuất Máy công cụ
  - Để giám sát, kiểm soát và tối ưu hóa quy trình sản xuất - để đảm bảo sự sẵn sàng sản xuất hàng loạt của các quy trình sản xuất
  - Phát hiện và loại bỏ các xáo trộn, sai sót trong quá trình sản xuất.
  - Áp dụng các hệ thống đảm bảo chất lượng vận hành và cụ thể cho khách hàng
  - Tìm kiếm và loại bỏ một cách có hệ thống các nguyên nhân của các khiếm khuyết chất lượng
  - Quy trình sản xuất tài liệu, kiểm tra chất lượng và lỗi / trục trặc
  - Góp phần cải tiến liên tục các quy trình làm việc trong quy trình vận hành
  - Chuẩn bị sản phẩm và giao thức và bàn giao cho khách hàng bên ngoài hoặc khu vực sản xuất tiếp theo (khách hàng nội bộ)

- **Năng lực tự chủ và trách nhiệm:**
  - Tuân thủ và áp dụng các quy định chung về an toàn lao động, sức khỏe, hỏa hoạn và bảo vệ môi trường (quan trắc, danh sách kiểm tra với 90% câu trả lời đúng)
  - Phân tích đơn đặt hàng công việc của khách hàng trong và ngoài nước và đánh giá tính khả thi về kỹ thuật và kinh tế phù hợp với các quy định an toàn và bảo vệ môi trường.
  - Có tính đến các yêu cầu và thời hạn cụ thể của khách hàng và sắp xếp cho các đơn đặt hàng một phần
  - Nghiên cứu, đánh giá thông tin lập kế hoạch việc làm trong mạng số
  - Bắt đầu, giám sát và kiểm tra các phần đơn đặt hàng
  - Để bàn giao sản phẩm cho khách hàng bên ngoài hoặc cho khu vực sản xuất tiếp theo (khách hàng nội bộ) và trình bày kết quả làm việc cũng với sự trợ giúp của phương tiện kỹ thuật số

- Chịu trách nhiệm trong quá trình sản xuất và nhận thức được trách nhiệm sản phẩm trong bối cảnh quan hệ kinh doanh với khách hàng.
- Giao tiếp và hợp tác trong các nhóm liên ngành
- Sử dụng năng lượng và vật liệu dưới các khía cạnh kinh tế và môi trường và xử lý vật liệu và chất một cách thân thiện với môi trường
- Đảm bảo thời gian học tập và sáng tạo học tập (quan sát, danh sách kiểm tra). - Tham gia tích cực vào các bài học (hơn 80% lý thuyết và 100% trong các bài học thực hành).

2. Phương pháp

Việc đánh giá dựa trên công việc dự án được thực hiện và các sản phẩm do thực tập sinh/ học viên và được thực hiện theo quy định về kiến thức và kỹ năng tối thiểu cần thiết cho sinh viên tốt nghiệp trình độ trung cấp và / hoặc cao đẳng trong nghề

- **Kiến thức:**
  Kỹ năng và hành vi của học viên / người học được xác định dựa trên các bài kiểm tra bằng miệng và viết như câu hỏi, thảo luận kỹ thuật và câu hỏi trắc nghiệm, cũng như thông qua các bài tập lý thuyết - thực hành tích hợp hoặc các bài tập thực hành trong quá trình thực hiện các đơn vị giảng dạy của mô-đun. Các đánh giá được tính theo các quy tắc điểm hợp lệ

- **Kỹ năng:**
  Trên cơ sở các bài tập thực hành, công tác dự án và mệnh lệnh công tác của công ty, hiệu quả thực hành của người thực hành/người học được đánh giá liên quan đến các tiêu chí sau với sự trợ giúp của các tờ/thang đo đánh giá:
  + An toàn lao động
  + Tổ chức nơi làm việc
  + Tiêu chuẩn kỹ thuật
  + Lập kế hoạch và thực hiện
  + Thời gian thực hiện mục tiêu
  + Tự đánh giá

- **Năng lực tự chủ và trách nhiệm:**
  Theo thái độ và tính cách của học viên/người học sẽ được xác định và đánh giá bằng quan sát trong toàn bộ thời gian đào tạo: đạo đức làm việc, học tập và hợp tác, đạo đức quy định và quy định, siêng năng, tận tâm, kỷ luật, khả năng làm việc nhóm, đúng giờ, độc lập, ý thức trách nhiệm, thận trọng, chủ động, tham gia tích cực vào bài học và hỗ trợ / động lực của người khác trong quá trình chỉ đạo

VI. Hướng dẫn thực hiện mô-đun chuyên nghiệp:

1. Phạm vi áp dụng

Mô-đun đào tạo để nâng cao trình độ công nghiệp 4.0 – Trình độ cao đẳng quốc tế.
Yêu cầu tuyển sinh: Bằng tốt nghiệp cao đẳng quốc gia về các ngành nghề điện tử hoặc cơ điện tử hoặc được chứng minh 3 năm kinh nghiệm làm việc chuyên nghiệp

trong ngành.

2. Hướng dẫn về phương pháp dạy và học

- Đối với giáo viên, giảng viên và giảng viên trong công ty: Giáo viên có trách nhiệm (tại TVET)/giảng viên trong công ty (trong công ty) tuân thủ các hướng dẫn sau để thực hiện chuyên môn các bài học lý thuyết và đào tạo thực hành của toàn bộ mô-đun và từng đơn vị đào tạo duy nhất:
  - Học viên/người học phải được hướng dẫn chi tiết và tuân thủ các quy định về an toàn lao động, sức khỏe và bảo vệ môi trường cũng như phòng cháy chữa cháy và an ninh. Việc tuân thủ các quy định phải được giám sát liên tục bởi giáo viên có trách nhiệm / giảng viên có trách nhiệm trong công ty. Người học viên/người học phải được thông báo rõ ràng và nhận thức được các biện pháp và hậu quả thích hợp của việc không tuân thủ các quy định.
  - Quá trình học tập và tiến độ học tập của học viên/người học được theo dõi và đánh giá thường xuyên, đặc biệt là việc tuân thủ nhất quán các quy định về an toàn lao động và các điều kiện bảo vệ môi trường.
  - Đảm bảo chất lượng giảng dạy và đào tạo cao nhất có thể thông qua việc tham khảo cho đơn vị giảng dạy tương ứng trong việc lập kế hoạch và thực hiện bài học.
  - Trong khuôn khổ các đơn vị đào tạo thực hành, các bước làm việc cần thiết phải được giải thích cẩn thận cho người thực tập/ người học và thể hiện chính xác.
  - Trình độ kiến thức, kỹ năng cá nhân được kiểm tra, đánh giá riêng đối với từng đơn vị giảng dạy thực tiễn trên cơ sở báo cáo công tác thường xuyên do học viên soạn thảo.
  - Chất lượng giảng dạy được nâng cao và đảm bảo bằng cách tăng cường sử dụng các phương pháp dạy và học khác nhau như phương pháp 4 bước, phương pháp dự án, văn bản hướng dẫn, tự học và làm việc nhóm cũng như sử dụng hiệu quả các tài liệu dạy và học và các công cụ hỗ trợ khác.
  - Kết quả công việc của học viên/người học phải được đánh giá và thảo luận minh bạch và cùng với học viên/người học bởi giáo viên dạy nghề có trách nhiệm hoặc bởi giảng viên công ty.
- Đối với học viên/người học:
  - Các học viên/người học được hướng dẫn:
    - Thực hiện nghiêm túc hướng dẫn của giáo viên dạy nghề hoặc giảng viên công ty
    - Tham gia thường xuyên, tích cực vào các bài học và từng đơn vị giảng dạy của mô-đun đào tạo
    - Tuân thủ các quy định về an toàn vệ sinh lao động, phòng cháy chữa cháy và bảo vệ môi trường
    - Góp phần tích cực vào công tác bảo vệ môi trường
    - Tuân thủ các quy định về giảng dạy và hội thảo
    - Tham gia chu đáo trong lớp, ghi chép và đặt câu hỏi trong trường hợp không chắc chắn

- Đặt câu hỏi cho giáo viên trường nghề hoặc giảng viên công ty hoặc cho các học viên / người học khác để yêu cầu hỗ trợ các nhiệm vụ khó khăn và nêu tên các vấn đề
- Chuẩn bị nơi làm việc và giữ cho nó sạch sẽ và gọn gàng
- Chuẩn bị, xử lý và bảo dưỡng thiết bị đúng cách
- Chuẩn bị báo cáo công việc hàng ngày và hàng tuần về các bài học lý thuyết và thực tiễn của mô-đun

3. Những trọng tâm cần chú ý:

Trọng tâm đào tạo của mô-đun đào tạo đặt trên tất cả các đơn vị đào tạo: 1, 2, 3 và 4

4. Ghi chú và giải thích thêm (nếu có)

B. NỘI DUNG ĐÀO TẠO

BÀI 1: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Mục tiêu: Các học viên:

- Biết sự khác biệt giữa lập trình chức năng và đối tượng
- Biết các thuật ngữ quan trọng nhất của lập trình hướng đối tượng
- Học được một ngôn ngữ lập trình hướng đối tượng
- Có thể sử dụng thư viện và điều chỉnh các lớp có sẵn theo nhu cầu của họ
- Có thể phân tích một vấn đề kỹ thuật và phát triển một giải pháp có tính đến các điều kiện hiện hành.
- Có thể điều chỉnh và ghi lại các mô-đun phần mềm và tích hợp chúng vào các hệ thống hiện có.
- Thiết kế kế hoạch thử nghiệm và kiểm tra các mô-đun phần mềm sửa đổi trong điều kiện hoạt động
- Thực hiện phân tích lỗi / lỗi có hệ thống và chuẩn bị tài liệu toàn diện về toàn bộ quy trình

Nội dung:

1. Lập trình hướng đối tượng

1.1. Nguyên tắc cơ bản của lập trình

a. Thuật ngữ lập trình cơ bản

• Thuật toán

Thuật toán là một tập hợp các hướng dẫn hoặc quy tắc được thiết kế để giải quyết một vấn đề xác định. Vấn đề có thể đơn giản như thêm hai số hoặc một số phức, chẳng hạn như chuyển đổi tệp video từ định dạng này sang định dạng khác.

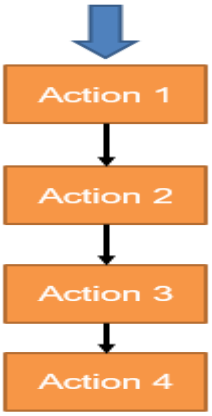
• Chương trình

Chương trình máy tính được gọi là một tập hợp các lệnh có tổ chức, khi được thực thi sẽ thực hiện một nhiệm vụ hoặc chức năng cụ thể. Một chương trình được xử lý bởi đơn vị xử lý trung tâm (CPU) của máy tính trước khi nó được thực thi. Ví dụ về chương trình là Microsoft Word, là một ứng dụng xử lý văn bản cho phép người dùng tạo và chỉnh sửa tài liệu. Các trình duyệt mà chúng ta sử dụng cũng là các chương trình được tạo ra để giúp chúng ta duyệt internet.

b. Những điều cơ bản về lập trình thủ tục

Lập trình thủ tục là một mô hình lập trình, bắt nguồn từ lập trình bắt buộc, dựa trên khái niệm về cuộc gọi thủ *tục*. Các thủ tục (một loại thông thường hoặc chương trình con) chỉ đơn giản là chứa một loạt các bước tính toán được thực hiện. Bất kỳ thủ tục nhất định nào cũng có thể được gọi tại bất kỳ thời điểm nào trong quá trình thực hiện chương trình, bao gồm cả các thủ tục khác hoặc chính nó. Các ngôn ngữ lập trình thủ tục lớn đầu tiên xuất hiện vào khoảng năm 1957-1964, bao gồm Fortran, ALGOL, COBOL, PL / I và BASIC. Pascal và C được xuất bản vào khoảng năm 1970-1972.

Bộ xử lý máy tính cung cấp hỗ trợ phần cứng cho lập trình thủ tục thông qua sổ đăng ký ngăn xếp và hướng dẫn các thủ tục gọi và trở về từ chúng. Hỗ trợ phần cứng cho các loại lập trình khác là có thể, nhưng không có nỗ lực nào thành công về mặt thương mại (ví dụ: máy Lisp hoặc bộ xử lý Java).



Biểu đồ 1: Lập trình thủ tục

c. Giới thiệu về các biến, mảng, điều kiện, vòng lặp và chức năng

- **Biến**

Biến trong Python là gì?
Biến Python là một vị trí bộ nhớ dành riêng để lưu trữ giá trị. Nói cách khác, một biến trong chương trình python cung cấp dữ liệu cho máy tính để xử lý.
Mỗi giá trị trong Python đều có một kiểu dữ liệu. Các loại dữ liệu khác nhau trong Python là Số, Danh sách, Tuple, Strings, Dictionary, v.v.
Các biến có thể được khai báo bằng bất kỳ tên hoặc thậm chí bảng chữ cái như a, aa, abc, v.v.
Quy tắc đặt tên biến đổi trong Python.
Tên biến đổi nên bắt đầu bằng chữ cái (a-zA-Z) hoặc dấu gạch dưới (_).
Giá trị : tuổi, _age, Tuổi không hợp lệ : 1age.
Trong tên biến, không có ký tự đặc biệt nào được phép ngoài dấu gạch dưới (_).
Giá trị : age_, _age
Không hợp lệ : age_*
Biến là trường hợp nhạy cảm. tuổi tác và tuổi tác là khác nhau, vì tên biến là trường hợp nhạy cảm.
Tên biến có thể có số nhưng không phải lúc bắt đầu. Ví dụ: Tên biến đổi tuổi 1
5.không nên là từ khóa Python. Từ khóa còn được gọi là từ dành riêng. Ví dụ vượt qua, phá vỡ, tiếp tục. vv được dành riêng cho ý nghĩa đặc biệt trong Python. Vì vậy, chúng ta không nên tuyên bố từ khóa là một tên biến. Cách Tuyên bố và sử dụng Biến
Hãy xem ví dụ. Chúng tôi sẽ tuyên bố biến "a" và in nó.

```
a=100
print (a)
```

- **Hoạt động toán học**
  - ✓ a = 4

- ✓ b = 3
- ✓ c = a + b = 7
- ✓ c = a - b =1
- ✓ c = a * b = 12
- ✓ c = a / b = 1.33333333
- ✓ c = a % b = 1 (modulo)
- ✓ c = a ** b = 64
- ✓ c = a // b = 1 (integer division)

- **Chức năng**

Một hàm là một khối mã có tổ chức, có thể tái sử dụng được sử dụng để thực hiện một hành động duy nhất, liên quan. Các chức năng cung cấp mô-đun tốt hơn cho ứng dụng của bạn và mức độ tái sử dụng mã cao.

Như bạn đã biết, Python cung cấp cho bạn nhiều chức năng tích hợp như in (), v.v. nhưng bạn cũng có thể tạo các chức năng của riêng mình. Các chức năng này được gọi là *hàm do người dùng xác định.*

Xác định hàm

Bạn có thể xác định các chức năng để cung cấp chức năng cần thiết. Dưới đây là các quy tắc đơn giản để xác định một hàm trong Python.

Các khối chức năng bắt đầu với từ khóa **def** được theo sau bởi tên hàm và dấu ngoặc đơn (()).

Bất kỳ tham số hoặc đối số đầu vào nào cũng nên được đặt trong các dấu ngoặc đơn này. Bạn cũng có thể xác định các tham số bên trong các dấu ngoặc đơn này.

Câu đầu tiên của một hàm có thể là một câu trả lời tùy chọn - chuỗi tài liệu của hàm hoặc *tài liệu*.

Khối mã trong mỗi hàm bắt đầu bằng một đại tràng (:) và bị thụt vào.

Câu trả lời câu trả lời [biểu thức] thoát khỏi một hàm, tùy chọn chuyển lại một biểu thức cho người gọi. Một tuyên bố trả về không có đối số cũng giống như trả về None.

Cú pháp

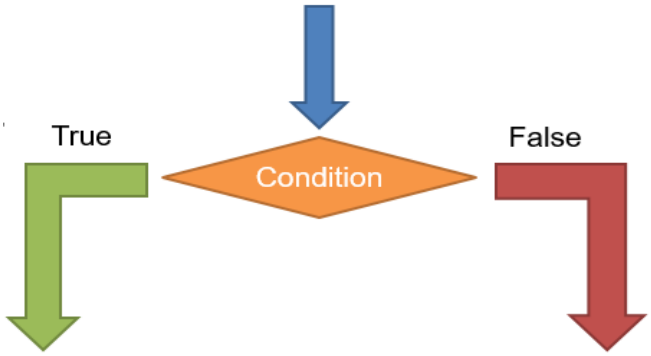
```
def functionname( parameters ):
   "function_docstring"
   function_suite
   return [expression]
```

- **Điều kiện**

Python sử dụng logic boolean để đánh giá các điều kiện. Các giá trị boolean True and False được trả về khi một biểu thức được so sánh hoặc đánh giá.

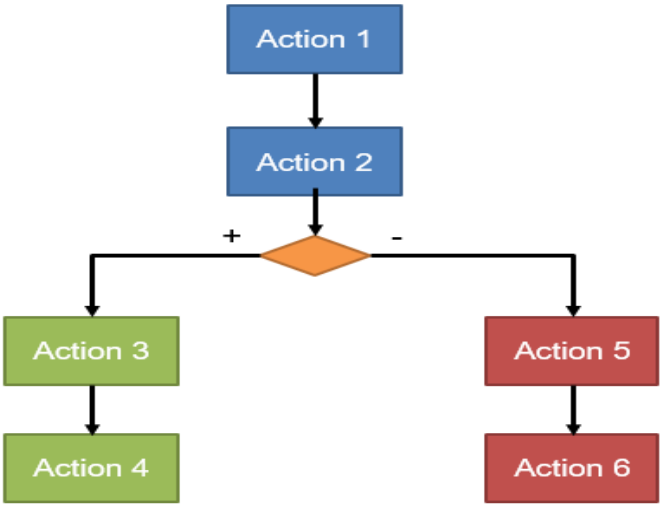
Luồng chương trình Thay đổi Điều kiện thông qua điều kiện

Chuyển hướng dòng chảy thực thi mã



Biểu đồ 2: Chức năng điều kiện

Chương trình có thể đi theo con đường đúng hoặc sai



Biểu đồ 3: Đường trình lập trình điều kiện

Điều kiện Python và nếu tuyên bố

Python hỗ trợ các điều kiện logic thông thường từ toán học:

i. Bằng: a == b
ii. Không bằng: a != b
iii. Ít hơn: một < b
iv. Ít hơn hoặc bằng: < = b
v. Lớn hơn: một > b
vi. Lớn hơn hoặc bằng: > = b

Những điều kiện này có thể được sử dụng theo nhiều cách, phổ biến nhất là trong "nếu tuyên bố" và vòng lặp.

Một "câu lệnh nếu" được viết bằng cách sử dụng từ khóa IF.

Ví dụ

Nếu tuyên bố:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

Elif

The elif keyword is pythons way of saying "if the previous conditions were not true, then try this condition".

Ví dụ

```
a = 33
b = 33
if b > a:
        print("b is greater than a")
elif a == b:
         print("a and b are equal")
```

Else

The else keyword catches anything which isn't caught by the preceding conditions.

Ví dụ

```
a = 200
b = 33
if b > a:
        print("b is greater than a")
elif a == b:
         print("a and b are equal")
else:
  print("a is greater than b")
```

In this example a is greater than b, so the first condition is not true, also the elif condition is not true, so we go to the else condition and print to screen that "a is greater than b".

Bạn cũng có thể có một cái khác mà không có elif:

Ví dụ

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

- **Vòng**

Python có hai lệnh vòng lặp nguyên thủy:

- ✓ while loops
- ✓ for loops

ví dụ:

Print i as long as i is less than 6:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Vòng lặp trong khi yêu cầu các biến có liên quan phải sẵn sàng, trong ví dụ này, chúng ta cần xác định một biến lập chỉ mục, i, mà chúng ta đặtthành 1.

A cho vòng lặp được sử dụng để lặp lại trên một chuỗi (đó là một danh sách, một tuple, một từ điển, một tập hợp, hoặc một chuỗi).

Điều này ít giống như từ khóa trong các ngôn ngữ lập trình khác và hoạt động giống như một phương pháp lặp lại như được tìm thấy trong các ngôn ngữ lập trình hướng đối tượng khác.

Với vòng lặp cho chúng ta có thể thực hiện một tập hợp các câu lệnh, một lần cho mỗi mục trong danh sách, kéo, đặt, v.v.

Ví dụ

In từng quả trong danh sách trái cây:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

- **Mảng**

Mảng được sử dụng để lưu trữ nhiều giá trị trong một biến duy nhất:

Ví dụ

Tạo một mảng chứa tên xe:

```
cars = ["Ford", "Volvo", "BMW"]
```

Một mảng là một biến đặc biệt, có thể giữ nhiều hơn một giá trị tại một thời điểm.

Nếu bạn có một danh sách các mục (ví dụ như danh sách tên xe), lưu trữ những chiếc xe trong các biến đơn lẻ có thể trông như sau:

```
car1 = "Ford"
car2 = "Volvo"
car3 = "BMW"
```

Tuy nhiên, điều gì sẽ xảy ra nếu bạn muốn vòng qua những chiếc xe và tìm một chiếc xe cụ thể? Và nếu bạn không có 3 chiếc xe, mà là 300 chiếc?

Giải pháp là một mảng!

Một mảng có thể chứa nhiều giá trị dưới một tên duy nhất và bạn có thể truy cập các giá trị bằng cách tham đề đến một số chỉ mục.

Truy nhập các Phần tử của Một Mảng

Bạn tham khảo một phần tử mảng bằng cách tham khảo *số chỉ mục*.

Example

Get the value of the first array item:

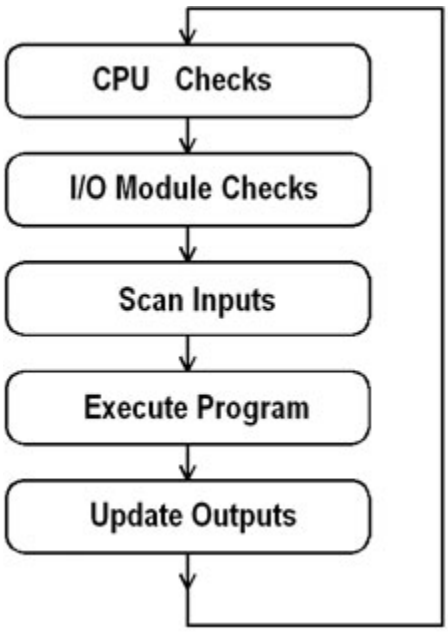
```
x = cars[0]
```

Example
Modify the value of the first array item:
cars[0] = "Toyota"

d. Sự khác biệt giữa IEC61131 và ngôn ngữ lập trình tiêu chuẩn

- Lập trình PLC so với lập trình PC

- **Lập trình PLC**
  + Xác định chu kỳ cố định
  + Hệ điều hành theo thời gian thực



Biểu đồ 4: Chu kỳ quét của PLC

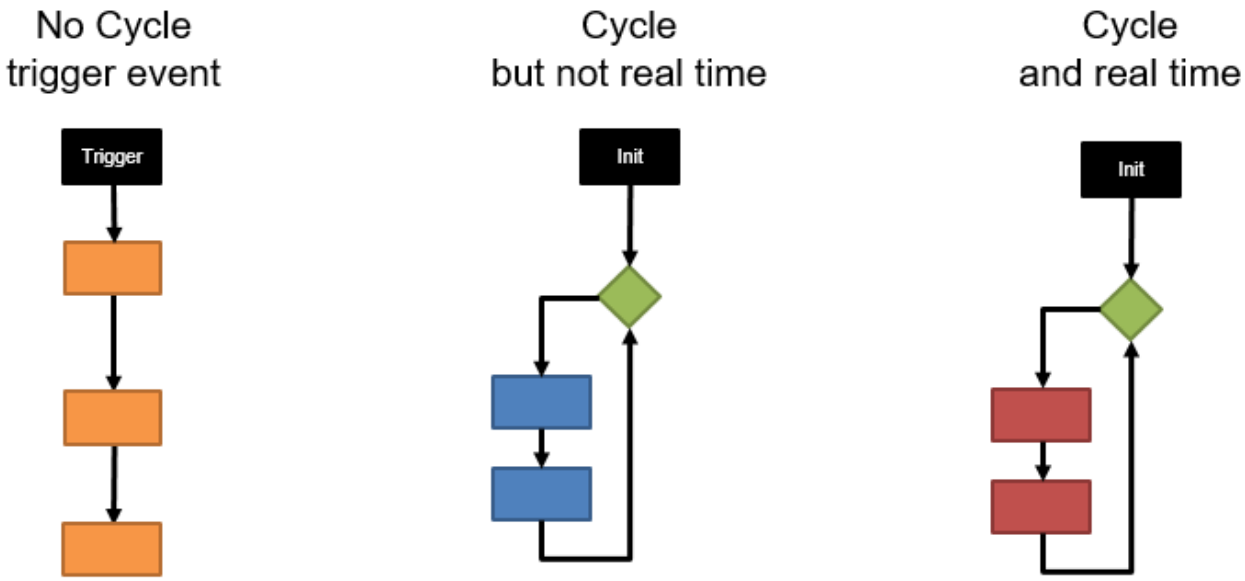
PLCs hoạt động bằng cách liên tục quét các chương trình và lặp lại quá trình này nhiều lần mỗi giây. Khi PLC bắt đầu, nó chạy kiểm tra phần cứng và phần mềm cho các lỗi, còn được gọi là tự kiểm tra. Nếu không có vấn đề gì, PLC sẽ bắt đầu chu kỳ quét. Chu kỳ quét bao gồm ba bước: quét đầu vào, thực hiện chương trình và quét đầu ra.

Quét đầuvào: Một cách đơn giản để xem xét điều này là PLC chụp nhanh các đầu vào và giải quyết logic. PLC xem xét từng thẻ đầu vào để xác định xem nó đang BẬT hay TẮT và lưu thông tin này trong bảng dữ liệu để sử dụng trong bước tiếp theo. Điều này làm cho quá trình nhanh hơn và tránh các trường hợp đầu vào thay đổi từ đầu đến cuối chương trình.

Thực hiện Chương trình (hoặc Thực thi Logic):PLC thực hiện một hướng dẫn một chương trình tại một thời điểm chỉ sử dụng bản sao bộ nhớ của các đầu vào của chương trình logic bậc thang. Ví dụ, chương trình có đầu vào đầu tiên là ON. Vì PLC biết đầu vào nào đang BẬT / tắt từ bước trước, nó sẽ có thể quyết định liệu đầu ra đầu tiên có nên được bật hay không.

Quét đầu ra: Khi quét thang hoàn tất, các đầu ra được cập nhật bằng cách sử dụng các giá trị tạm thời trong bộ nhớ. PLC cập nhật trạng thái của các đầu ra dựa trên đầu vào nào được BẬT trong bước đầu tiên và kết quả thực hiện chương trình trong bước thứ hai. PLC hiện khởi động lại quá trình bằng cách bắt đầu tự kiểm tra lỗi.

- Lập trình PC
  + Không xác định chu kỳ
  + Chu kỳ có thể được lập trình
  + Thời gian thực phụ thuộc vào hệ điều hành và thiết kế chương trình



Biểu đồ 5: Chu kỳ và thời gian thực

e. Tìm hiểu môi trường phát triển tích hợp (IDE)

Môi **trường phát triển tích hợp (IDE)**là một ứng dụng phần mềm cung cấp các cơ sở toàn diện cho các lập trình viên máy tính để phát triển phần mềm. IDE thường bao gồm ít nhất một trình chỉnh sửa mã nguồn, xây dựng các công cụ tự động hóa và gỡ lỗi. Một số IDE, chẳng hạn như NetBeans và Eclipse, chứa trình biên dịch, thông dịch viên hoặc cả hai cần thiết; Những người khác, chẳng hạn như SharpDevelop và Lazarus, thì không.

Ranh giới giữa IDE và các bộ phận khác của môi trường phát triển phần mềm rộng lớn hơn không được xác định rõ ràng; đôi khi một hệ thống điều khiển phiên bản hoặc các công cụ khác nhau để đơn giản hóa việc xây dựng giao diện người dùng đồ họa (GUI) được tích hợp. Nhiều IDEs hiện đại cũng có trình duyệt lớp, trình duyệt đối tượng và sơ đồ phân cấp lớp để sử dụng trong phát triển phần mềm hướng đối tượng.

f. Tài liệu và nhận xét phần mềm

- Python 3.9

Python là một ngôn ngữ lập trình máy tính thường được sử dụng để xây dựng các trang web và phần mềm, tự động hóa các tác vụ và tiến hành phân tích dữ liệu. Python là một ngôn ngữ mục đích chung, có nghĩa là nó có thể được sử dụng để tạo ra một loạt các chương trình khác nhau và không chuyên biệt cho bất kỳ vấn đề cụ thể nào. Tính linh hoạt này, cùng với sự thân thiện với người mới bắt đầu, đã làm cho nó trở thành một trong những ngôn ngữ lập trình được sử dụng nhiều nhất hiện nay. Một cuộc khảo sát được thực hiện bởi công ty phân tích công nghiệp RedMonk cho thấy đây là ngôn ngữ lập trình phổ biến nhất trong số các nhà phát triển vào năm 2020.

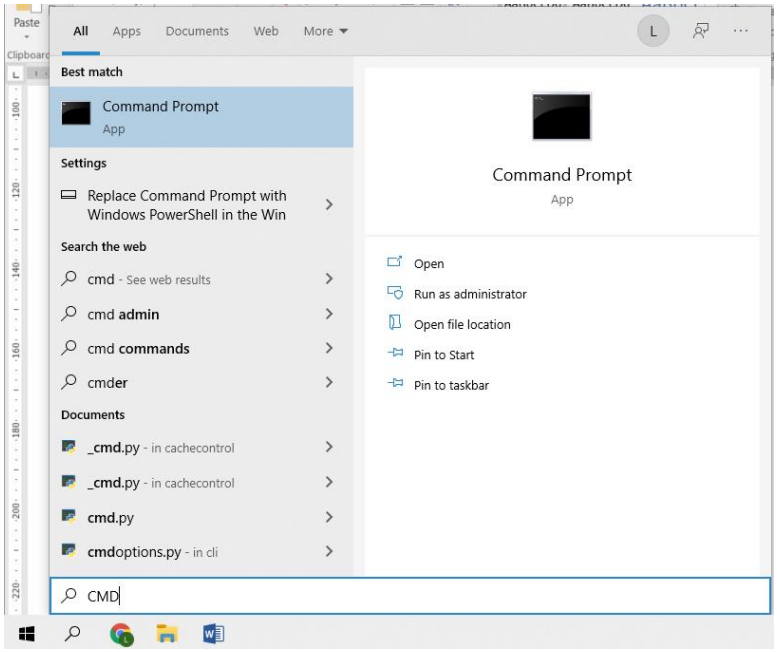
- Pycharm 3.9

PyCharm là một môi trường phát triển tích hợp (IDE) được sử dụng trong lập trình máy tính, đặc biệt cho ngôn ngữ Python. Nó được phát triển bởi công ty Séc JetBrains (trước đây gọi là IntelliJ). Nó cung cấp phân tích mã, gỡ lỗi đồ họa, kiểm tra đơn vị tích hợp, tích hợp với các hệ thống điều khiển phiên bản (VCSes) và hỗ trợ phát triển web với Django cũng như khoa học dữ liệu với Anaconda.
PyCharm là nền tảng đa nền tảng, với các phiên bản Windows, macOS và Linux. Community Edition được phát hành theo Giấy phép Apache, và cũng có Phiên bản chuyên nghiệp với các tính năng bổ sung - được phát hành theo giấy phép độc quyền.

1.2 Điều chỉnh mô-đun phần mềm

a. Bao gồm các mô-đun từ các thư viện được xác định trước hoặc bên ngoài

- Mở lệnh hệ thống windows (Nhập "cmd" vào tìm kiếm windows)



Hình 1: Giao diện lệnh hệ thống Windows

- Cửa sổ sẽ trông như thế này:

Hình 2: Màn hình cửa sổ lệnh của Microsoft windows

- In system command type (press enter after every command):
  i. pip3 install numpy
  ii. pip3 install matplotlib
  iii. pip3 install pandas
  iv. pip3 install scipy
  v. pip3 install tk

b. Hiểu mã chương trình

- **Tương tác người dùng đơn giản**

Những nhiệm vụ này được tạo ra cho một cấp độ mới bắt đầu trong lập trình. Dự định của họ là làm quen với một ngôn ngữ lập trình mới. Bên cạnh kiến thức nền tảng lý thuyết đã được cung cấp trong khóa học không cần thêm kiến thức. Các nhiệm vụ chỉ được giải quyết với sự trợ giúp của tài liệu khóa học và không có internet, trừ khi được nêu khác nhau.

- ***Đầu ra Bảng điều khiển***

Yêu cầu:

Nhiệm vụ của chương trình là xuất gia đình và tên của người dùng, tuổi và địa chỉ. Đầu ra của chương trình phải được hiển thị trong bảng điều khiển. Dữ liệu phải được mã hóa cứng. Không cần thêm đầu vào của người dùng. Để thay đổi dữ liệu, người dùng phải thay đổi mã chương trình.

Đầu ra bắt buộc:

First name: XXX
Family name: YYY
Age: DD.MM.YY
Address: XXYY

- Sulution:

```
firstName = "XXX"
familyName = "YYY"
```

```
Age = 23
Address = "XXXYYY"
print("First name:" + firstName)
print("Family name:" + familyName)
print("Age:" + str(Age))
print("Address:" + Address)
```

- ***Đầu vào Bảng điều khiển***

Yêu cầu:

Cập nhật chương trình từ 3.1 để người dùng nhập liệu từ bảng điều khiển có thể được sử dụng để đặt tên, địa chỉ và tuổi vào thời gian chạy. -

Thông tin bổ sung:

- Đầu vào chức năng **()** đọc trong đầu vào của người dùng từ bảng điều khiển cho đến khi nhấn nút trả về.
- Dữ liệu được lưu trữ dưới dạng chuỗi chuỗi dữ liệu.
- Tương tác của người dùng sẽ được đánh dấu bằng <<>>

Đầu ra bắt buộc:

```
Vui lòng thêm tên: <<nhân thêm tên>>
Vui lòng thêm tên gia đình: <<nguục thêm vào tên đầu tiên>>
Vui lòng thêm tuổi: <<nguất thêm tên>>
Vui lòng thêm địa chỉ: <<nguục thêm tên>>
------------------------------
First name: XXX
Family name: YYY
Age: DD.MM.YY
Address: XXYY
------------------------------
```

- Sulution:

```
firstName = input("Please add first name: ")
familyName = input("Please add family name:  ")
Age = input("Please add age: ")
Address = input("Please add address: ")
print("------------------------------")
print("First name:" + firstName)
print("Family name:" + familyName)
print("Age:" + Age)
print("Address:" + Address)
```

c. Hiểu cuộc gọi chức năng cơ chế và lập trình mô-đun

Chức năng trong Python là gì?

Trong Python, một hàm là một nhóm các tuyên bố liên quan thực hiện một nhiệm vụ cụ thể.

Các chức năng giúp chia chương trình của chúng tôi thành các khối nhỏ hơn và mô-đun. Khi chương trình của chúng tôi phát triển ngày càng lớn hơn, các chức năng làm cho nó có tổ chức và dễ quản lý hơn.

Hơn nữa, nó tránh lặp lại và làm cho mã có thể tái sử dụng.

Cú pháp hàm

```
def function_name(parameters):
        """docstring"""
        statement(s)
```

Ở trên được hiển thị là một định nghĩa hàm bao gồm các thành phần sau đây.

i. Từ khóa def đó đánh dấu sự khởi đầu của tiêu đề chức năng.
ii. Một tên chức năng để xác định duy nhất hàm. Đặt tên chức năng tuân theo các quy tắc tương tự của việc viết mã định danh trong Python.
iii. Tham số (đối số) mà qua đó chúng ta chuyển các giá trị đến một hàm. Chúng là tùy chọn.
iv. Đại tràng (:) để đánh dấu cuối tiêu đề hàm.
v. Chuỗi tài liệu tùy chọn (docstring) để mô tả những gì chức năng làm.
vi. Một hoặc nhiều câu nói python hợp lệ tạo nên cơ thể chức năng. Các câu trả lời phải có cùng mức thụt lề (thường là 4 khoảng trống).
vii. Câu trả về tùy chọn để trả về giá trị từ hàm.

Ví dụ về hàm

```
def greet(name):
    """    This function greets to  the person passed in as  a parameter"""
    print("Hello, " + name + ". Good morning!")
```

Làm thế nào để gọi một hàm trong python?

Một khi chúng ta đã xác định một hàm, chúng ta có thể gọi nó từ một chức năng, chương trình hoặc thậm chí là lời nhắc Python khác. Để gọi một hàm, chúng tôi chỉ cần nhập tên hàm với các tham số thích hợp.

```
>>> greet('Paul')
Hello, Paul. Good morning!
```

Hãy thử chạy mã trên trong chương trình Python với định nghĩa hàm để xem đầu ra.

```
def greet(tên):
    """    This function greets to  the person passed in as  a parameter"""
```

```
    print("Hello, " + name + ". Good morning!")
greet('Paul')
```

Note: Trong python, định nghĩa hàm phải luôn luôn có mặt trước cuộc gọi chức năng. Nếu không, chúng ta sẽ nhận được một sai lầm. Chẳng hạn

```
# function call
greet('Paul')
# function definition
def greet(name):
    """    This function greets to  the person passed in as  a parameter"""
    print("Hello, " + name + ". Good morning!")
# Erro: name 'greet' is not defined
```

Báo cáo trả lại

Câu trả lời được sử dụng để thoát khỏi một hàm và quay trở lại nơi mà nó được gọi.

Cú pháp trả về

```
return [expression_list]
```

Câu này có thể chứa một biểu thức được đánh giá và giá trị được trả về. Nếu không có biểu thức trong câu lệnh hoặc bản thân câu trả về không có mặt bên trong một hàm, thì hàm sẽ trả về đối tượng None.

For example:

```
>>> print(greet("May"))
Hello, May. Good morning!
None
```

Ở đây, Không có giá trị trả về vì chào hỏi () trực tiếp in tên và không sử dụng sao kê trả lại.

Ví dụ về sự trở lại

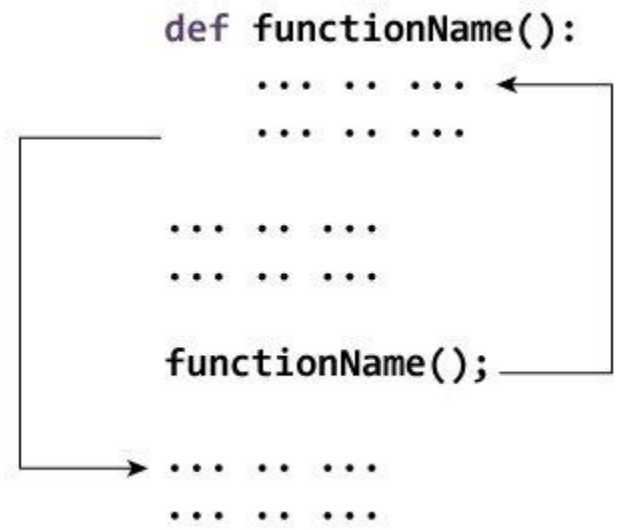
```
def absolute_value(num):
    """    This function greets to  the person passed in as  a parameter"""
    if num >= 0:
        return num
    else:
        return -num
print(absolute_value(2))
```

```
print(absolute_value(-4))
```

Output

```
2
4
```

Chức năng hoạt động như thế nào trong Python?



- *Giới thiệu lập trình mô-đun*

Lập trình mô-đun là một công cụ thiết yếu cho nhà phát triển hiện đại. Đã qua rồi cái thời mà bạn chỉ có thể ném một cái gì đó lại với nhau và hy vọng rằng nó hoạt động. Để xây dựng các hệ thống mạnh mẽ kéo dài, bạn cần hiểu cách tổ chức các chương trình của mình để chúng có thể phát triển và phát triển theo thời gian. *Spaghetti coding* không phải là một lựa chọn. Các kỹ thuật lập trình mô-đun, và đặc biệt là việc sử dụng các mô-đun và gói Python, sẽ cung cấp cho bạn các công cụ bạn cần để thành công như một chuyên gia trong bối cảnh lập trình thay đổi nhanh chóng.

Trong chương này, chúng tôi sẽ:

- Nhìn vào các khía cạnh cơ bản của lập trình mô-đun
- Xem cách sử dụng các mô-đun và gói Python để sắp xếp mã của bạn
- Khám phá những gì xảy ra khi các kỹ thuật lập trình mô-đun không được sử dụng
- Tìm hiểu cách lập trình mô-đun giúp bạn đứng đầu quá trình phát triển
- Hãy xem thư viện tiêu chuẩn Python như một ví dụ về lập trình mô-đun
- Tạo một chương trình đơn giản, được xây dựng bằng cách sử dụng các kỹ thuật mô-đun, để xem nó hoạt động như thế nào trong thực tế

Hãy bắt đầu bằng cách tìm hiểu về các mô-đun và cách chúng hoạt động.

Đối với hầu hết các lập trình viên mới bắt đầu, chương trình Python đầu tiên của họ là một số phiên bản của chương trình *Hello World* nổi tiếng. Chương trình này sẽ trông như thế này:

```
print("Hello World!")
```

Chương trình một dòng này sẽ được lưu trong một tệp trên đĩa, thường được đặt tên là một cái gì đó như hello.py, và nó sẽ được thực hiện bằng cách nhập lệnh sau đây vào một thiết bị đầu cuối hoặc cửa sổ dòng lệnh:

```
python hello.py
```

Thông dịch viên Python sau đó sẽ nghiêm túc in ra thông điệp mà bạn đã yêu cầu nó:

```
Hello World!
```

Tệp hello.py này được gọi là tệp nguồn **Python**. Khi bạn mới bắt đầu, đặt tất cả mã chương trình của bạn vào một tệp nguồn duy nhất là một cách tuyệt vời để tổ chức chương trình của bạn. Bạn có thể xác định các chức năng và lớp học và đặt hướng dẫn ở phía dưới bắt đầu chương trình của bạn khi bạn chạy nó bằng cách sử dụng thông dịch Viên Python. Lưu trữ mã chương trình của bạn bên trong tệp nguồn Python giúp bạn không phải nhập lại nó mỗi khi bạn muốn nói với thông dịch viên Python phải làm gì.

Tuy nhiên, khi các chương trình của bạn trở nên phức tạp hơn, bạn sẽ thấy rằng việc theo dõi tất cả các chức năng và lớp khác nhau mà bạn xác định ngày càng khó khăn hơn. Bạn sẽ quên nơi bạn đặt một đoạn mã cụ thể và thấy ngày càng khó nhớ tất cả các phần khác nhau phù hợp với nhau như thế nào.

Lập trình mô-đun là một cách tổ chức các chương trình khi chúng trở nên phức tạp hơn. Bạn có thể tạo một **mô-đun**Python, một tệp nguồn có chứa mã nguồn Python để làm điều gì đó hữu ích, sau đó **nhập** mô-đun này vào chương trình của bạn để bạn có thể sử dụng nó. Ví dụ: chương trình của bạn có thể cần theo dõi các số liệu thống kê khác nhau về các sự kiện diễn ra trong khi chương trình đang chạy. Cuối cùng, bạn có thể muốn biết có bao nhiêu sự kiện của mỗi loại đã xảy ra. Để đạt được điều này, bạn có thể tạo tệp nguồn Python có tên stats.py có chứa mã Python sau:

```
def init():

def event_occurred(event):

def get_stats():
```

Tệp nguồn Python stats.py xác định một mô-đun có tên số liệu thống kê —như bạn có thể thấy, tên của mô-đun chỉ đơn giản là tên của tệp nguồn mà không có hậu tố .py. Chương trình chính của bạn có thể sử dụng mô-đun này bằng cách nhập nó và sau đó

gọi các chức năng khác nhau mà bạn đã xác định là cần thiết. Ví dụ phù phiếm sau đây cho thấy cách bạn có thể sử dụng mô-đun thống kê để thu thập và hiển thị số liệu thống kê về các sự kiện:

số liệu thống kê nhập khẩu

```
stats.init()
stats.event_occurred("meal_eaten")
stats.event_occurred("snack_eaten")
stats.event_occurred("meal_eaten")
stats.event_occurred("snack_eaten")
stats.event_occurred("meal_eaten")
stats.event_occurred("diet_started")
stats.event_occurred("meal_eaten")
stats.event_occurred("meal_eaten")
stats.event_occurred("meal_eaten")
stats.event_occurred("diet_abandoned")
stats.event_occurred("snack_eaten")
for event,num_times in stats.get_stats():
```

Tất nhiên, chúng tôi không quan tâm đến việc ghi lại các bữa ăn và đồ ăn nhẹ - đây chỉ là một ví dụ - nhưng điều quan trọng cần chú ý ở đây là cách mô-đun thống kê được nhập, và sau đó làm thế nào các chức năng khác nhau bạn xác định trong tệp stats.py được sử dụng. Ví dụ, hãy xem xét dòng mã sau:

```
stats.event_occurred("snack_eaten")
```

Bởi vì hàm event_occurred() được xác định trong mô-đun thống kê, bạn cần bao gồm tên của mô-đun bất cứ khi nào bạn đề cập đến chức năng này.

- Chú ý

Có nhiều cách mà bạn có thể nhập các mô-đun để bạn không cần phải bao gồm tên của mô-đun mỗi lần. Chúng ta sẽ xem xét điều này trong Chương 3, *Sử dụng Mô-đun và Gói,*khi chúng ta nhìn vào không gian tên vàcách lệnh nhập hoạt động chi tiết hơn. Như bạn có thể thấy, tuyên bố nhập được sử dụng để tải một mô-đun và bất cứ khi nào bạn thấy tên mô-đun theo sau là một khoảng thời gian, bạn có thể nói rằng chương trình đang đề cập đến một cái gì đó (ví dụ: một hàm hoặc lớp) được xác định trong mô-đun đó.

Ví dụ

```
from math import pi
r = float(input(""))
A = r**2 * pi
print("The area of the circle is: " +str(A))
```

1.3 Những điều cơ bản về lập trình hướng đối tượng

a. Lập trình thủ tục so với lập trình hướng đối tượng

Lập trình hướng đối tượng và lập trình thủ tục đều được sử dụng để phát triển các ứng dụng. Cả hai đều là ngôn ngữ lập trình cấp cao. Đây là hai khái niệm quan trọng, và điều quan trọng là phải biết sự khác biệt giữa chúng..

STT.	Trên cơ sở	Lập trình thủ tục	Lập trình hướng đối tượng
1.	Definition	Nó là một ngôn ngữ lập trình có nguồn gốc từ lập trình cấu trúc và dựa trên khái niệm về thủ tục gọi. Nó tuân theo cách tiếp cận từng bước để chia nhỏ một nhiệm vụ thành một tập hợp các biến và thói quen thông qua một chuỗi các hướng dẫn.	Lập trình hướng đối tượng là một triết lý hoặc phương pháp thiết kế lập trình máy tính tổ chức / mô hình thiết kế phần mềm xung quanh dữ liệu hoặc đối tượng chứ không phải là các chức năng và logic.
2.	Security	Nó ít an toàn hơn OOP.	Ẩn dữ liệu có thể xảy ra trong lập trình hướng đối tượng do trừu tượng. Vì vậy, nó an toàn hơn lập trình thủ tục.
3.	Approach	Nó theo một cách tiếp cận từ trên xuống.	Nó theo cách tiếp cận từ dưới lên.
4.	Data movement	Trong lập trình thủ tục, dữ liệu di chuyển tự do trong hệ thống từ chức năng này sang chức năng khác.	Trong OOP, các đối tượng có thể di chuyển và giao tiếp với nhau thông qua các chức năng thành viên.
5.	Orientation	It is structure/procedure-oriented.	Nó hướng đối tượng.
6.	Access modifiers	Không có trình sửa đổi truy cập trong lập trình thủ tục.	Các sửa đổi truy cập trong OOP được đặt tên là tư nhân, công cộng và được bảo vệ.
7.	Inheritance	Lập trình thủ tục không có khái niệm về kế thừa.	Có một tính năng kế thừa trong lập trình hướng đối tượng.

8.	Code reusability	Không có khả năng tái sử dụng mã có trong lập trình thủ tục.	Nó cung cấp khả năng tái sử dụng mã bằng cách sử dụng tính năng thừa kế.
9.	Overloading	Quá tải là không thể trong lập trình thủ tục.	Trong OOP, có một khái niệm về quá tải chức năng và quá tải người vận hành.
10.	Importance	Nó mang lại tầm quan trọng cho các chức năng trên dữ liệu.	Nó mang lại tầm quan trọng cho dữ liệu so với các chức năng.
11.	Virtual class	Trong lập trình thủ tục, không có lớp học ảo.	In OOP, there is an appearance of virtual classes in inheritance.
12.	Complex problems	Nó không thích hợp cho các vấn đề phức tạp.	Nó thích hợp cho các vấn đề phức tạp.
13.	Data hiding	Không có cách nào thích hợp để ẩn dữ liệu.	Có khả năng ẩn dữ liệu.
14.	Program division	Trong lập trình thủ tục, một chương trình được chia thành các chương trình nhỏ được gọi là hàm.	Trong OOP, một chương trình được chia thành các phần nhỏ được gọi là đối tượng..
15.	Examples	Ví dụ về lập trình Thủ tục bao gồm C, Fortran, Pascal và VB.	Các ví dụ về lập trình hướng đối tượng là -.NET, C#, Python, Java, VB.NET và C++.

Bảng 1: So sánh lập trình thủ tục và lập trình hướng đối tượng

b. Sự khác biệt giữa các lớp và đối tượng

- ***Class học là gì?***

Một lớp là một thực thể xác định cách một đối tượng sẽ hoạt động và những gì đối tượng sẽ chứa. Nói cách khác, nó là một bản thiết kế hoặc một tập hợp các hướng dẫn để xây dựng một loại đối tượng cụ thể. Nó cung cấp các giá trị ban đầu cho các biến thành viên và các chức năng hoặc phương pháp thành viên.

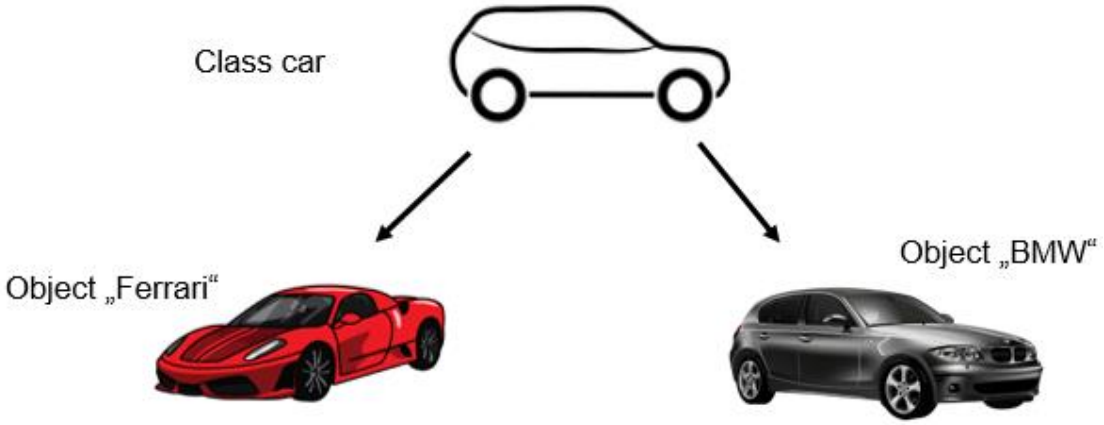
- ***Object là gì?***

Một đối tượng không là gì ngoài một thành phần khép kín bao gồm các phương pháp và thuộc tính để làm cho dữ liệu hữu ích. Nó giúp bạn xác định hành vi của lớp học.

Ví dụ: khi bạn gửi tin nhắn đến một đối tượng, bạn đang yêu cầu đối tượng gọi hoặc thực hiện một trong các phương pháp của nó.
Từ quan điểm lập trình, một đối tượng có thể là cấu trúc dữ liệu, biến hoặc hàm có vị trí bộ nhớ được phân bổ. Đối tượng được thiết kế dưới dạng phân cấp lớp.
Đây là sự khác biệt quan trọng giữa lớp và đối tượng:

Class	Object
Lớp học là một mẫu để tạo đối tượng trong chương trình.	Đối tượng là một ví dụ của một lớp.
Một lớp là một thực thể logic	Đối tượng là một thực thể vật lý
Một lớp không phân bổ không gian bộ nhớ khi nó được tạo ra.	Đối tượng phân bổ không gian bộ nhớ bất cứ khi nào chúng được tạo.
Bạn chỉ có thể khai báo lớp học một lần.	Bạn có thể tạo nhiều hơn một đối tượng bằng cách sử dụng một lớp.
Ví dụ: Car.	Ví dụ: Jaguar, BMW, Tesla, etc.
Lớp tạo đối tượng	Các đối tượng cung cấp cuộc sống cho lớp học.
Các lớp học không thể bị thao túng vì chúng không có sẵn trong bộ nhớ.	Chúng có thể bị thao túng.
Nó không có bất kỳ giá trị nào được liên kết với các trường.	Mỗi đối tượng đều có những giá trị riêng, được liên kết với các trường.
Bạn có thể tạo lớp bằng từ khóa "lớp".	Bạn có thể tạo đối tượng bằng từ khóa "mới" trong Java

Bảng 2: So sánh Lớp và Đối tượng

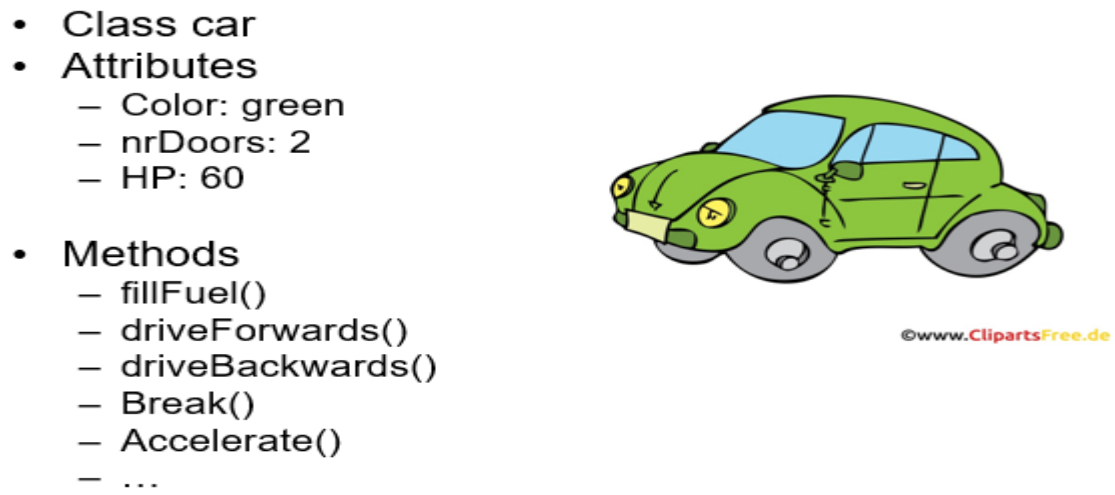


Hình 3: Trình diễn giữa "Lớp" và "Đối tượng"

- Thành viên lớp
- Thuộc tính
  - Biến
  - "thuộc tính đối tượng"
  - Có thể phục vụ như một "ký ức"
- Phương pháp

- Chức năng
- "Xác định những gì một đối tượng có khả năng làm"

- Ví dụ về OOP



Hình 4: Ví dụ về OOP

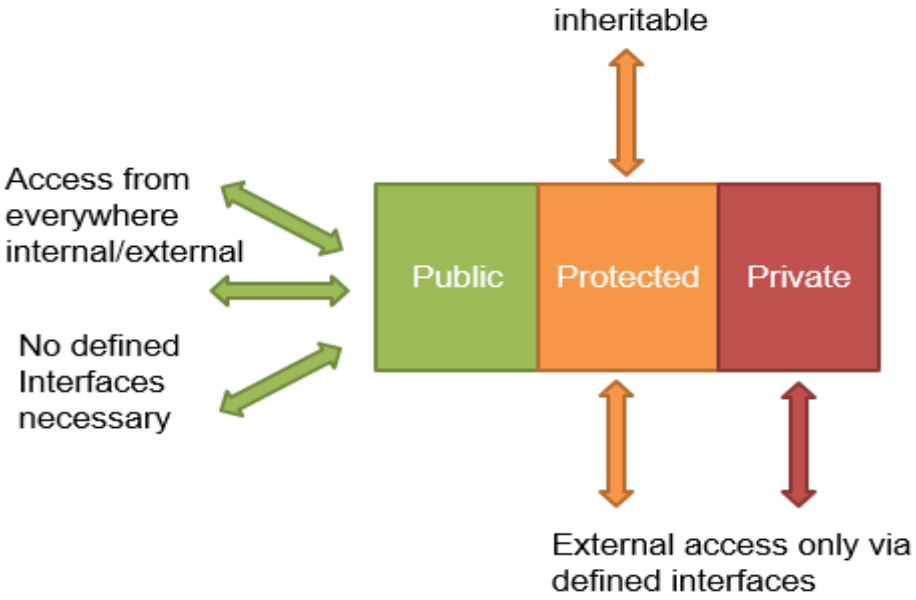
c. Quyền truy cập khía cạnh bảo mật cho các lớp

- Khả năng tiếp cận của các thành viên trong lớp
- Đóng gói dữ liệu
- Kiểm soát quyền truy cập trên các thuộc tính và phương pháp
- Phương pháp bảo mật dữ liệu
- Tạo giao lộ được xác định để sử dụng Khả năng lớp

+ Public: : Truy cập luôn có thể

Private: Truy cập từ lớp phụ huynh và các lớp có nguồn gốc có thể

- Protected: Chỉ có thể truy cập từ lớp phụ huynh



Biểu đồ 6: Khả năng của Lớp

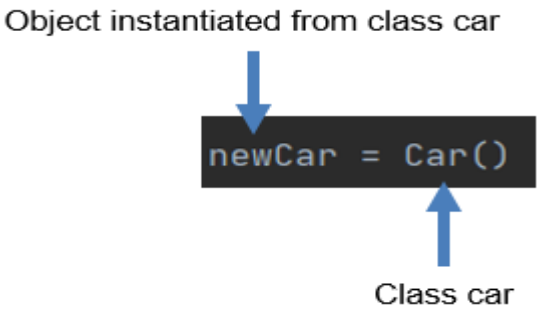
Phương pháp Nhận và thiết lập:

- Chức năng công cộng
- Giao diện được xác định
- Truy cập các thành viên riêng tư và được bảo vệ

d. Tạo nhà xây dựng/kết cấu cho các lớp

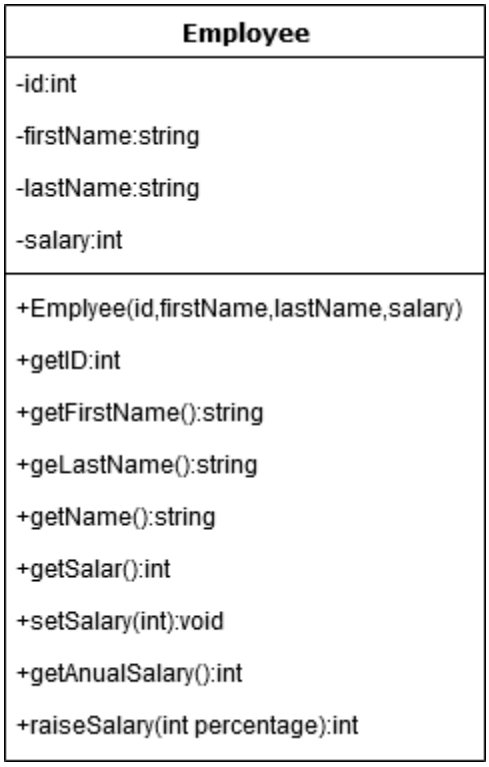
```
class Car:
    color = 'green'
    fuelLevel = 0

    def refuel(self):
        self.fuelLevel = 50
```

Ví dụ về đối tượng instantiate



Ví dụ

Tạo một lớp bằng cách sử dụng sơ đồ UML đã cho bên dưới. Tất cả các phương pháp getter trả về các thuộc tính tương ứng. Phương pháp Setter với khoảng trống kiểu dữ liệu không trả về bất kỳ giá trị nào. Phương pháp getName sẽ trả về tên đầy đủ với "Họ tên". Phương pháp tăng tỷ lệ phần trăm sẽ trả lại mức lương mới được tăng theo tỷ lệ phần trăm.



Giải pháp:

```
class Employee:
    def __init__(self, id, firstName, lastName, salary):
        self._id = id
        self._firstName = firstName
        self._lastName = lastName
        self._salary = salary

    def getID(self):
        return self._id

    def getFirstName(self):
        return self._firstName

    def getLastName(self):
        return self._lastName

    def getName(self):
        return self._firstName + " " + self._lastName

    Creation of constructor/structure for classes def getSalary(self):
        return self._salary

    def setSalary(self, value):
```

```
        self._salary = value

    def getAnnualSalary(self):
        return self._salary * 12

    def raisePercentage(self, percentage):
        self._salary = self._salary * percentage / 100 + self._salary
        return self._salary
```

e. Tạo lớp cơ sở để thừa kế

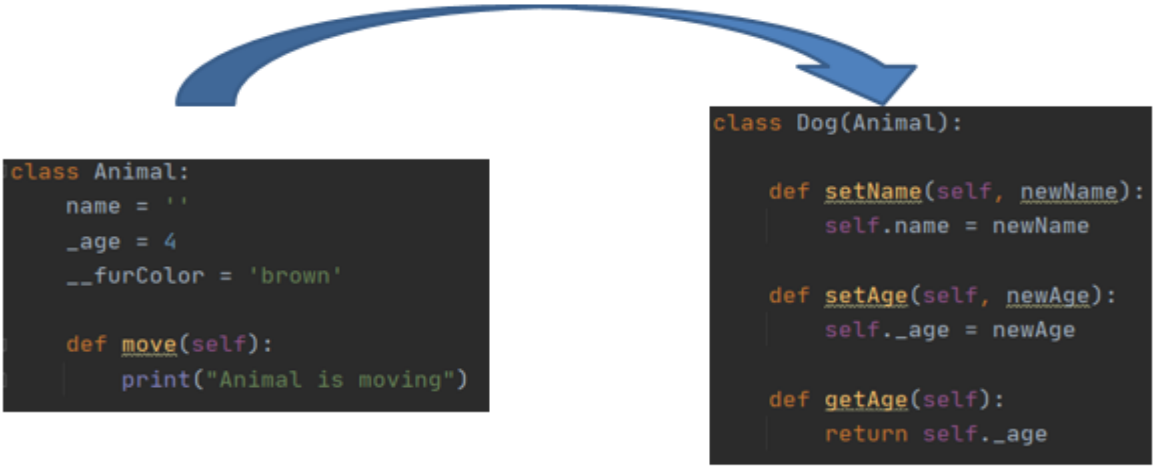
- Giới thiệu

Kế thừa là một trong những khía cạnh quan trọng nhất của Lập trình hướng đối tượng (OOP). Chìa khóa để hiểu kế thừa là nó cung cấp khả năng tái sử dụng mã. Thay vì viết cùng một mã, hết lần này đến lần khác, chúng ta chỉ có thể kế thừa các thuộc tính của lớp này sang lớp khác.

f. Hiểu rõ cấu trúc và cơ chế kế thừa

- Lớp học trẻ em thừa hưởng các thành viên từ lớp phụ huynh
- Ngăn chặn mã dư thừa
- Mô-đun hóa
- Bảo vệ không được thừa kế

- Python thừa kế



Hình 5: Sự kế thừa trong Python

- Thừa kế – Overriding

- Ghi đè
  1.Mới của phương pháp từ lớp cơ sở
  2.Thêm chức năng
  3.Thay đổi chức năng
  4.Thích ứng với các phương pháp sang lớp mới
- Sáng tạo trong lớp cơ sở như kỷ niệm

- Ví dụ ghi đè

Hình 7: Logo của pandas

```
class Animal:
    name = ''
    _age = 4
    __furColor = 'brown'

    def move(self):
        print("Animal is moving")


class Dog(Animal):

    def move(self):
        print("Dog is moving")
```

Animal method getting overridden in Dog class

Hình 6: Kế thừa - Ghi đè

1.4. Khái niệm mở rộng

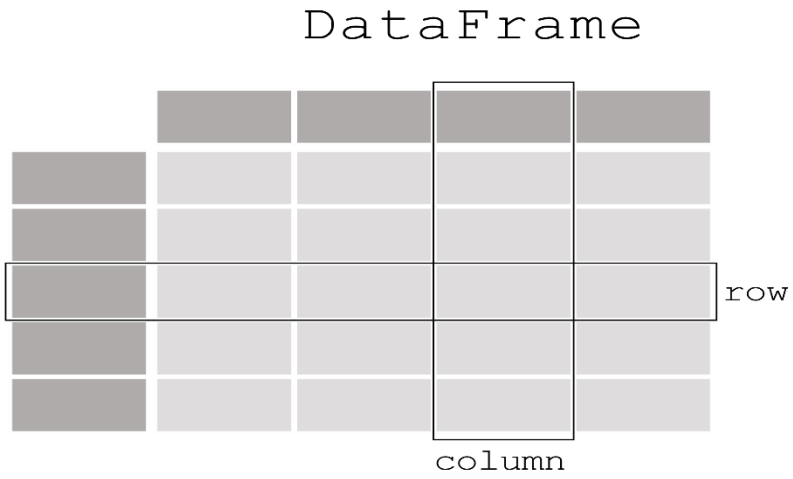
a. Thực hiện đầu vào và đầu ra tệp bằng cách sử dụng cấu trúc hướng đối tượng

- DataIO

- Đọc dữ liệu từ
  1. Tập tin
  2. Suối
  3. Lưu dữ liệu vào tệp
- Gấu trúc thư viện phổ biến
  - Nối kết nội bộ: https://pandas.pydata.org/pandas-docs/sBảng/getting_started/intro_tutorials/01_Bảng_oriented.html

- Pandas Dataframe

- Có cấu trúc như bàn
    1. Hàng
    2. Cột
- Có thể truy cập qua tên
- Thông tin thống kê



Hình 8: Mẫu khung dữ liệu của pandas

- **Khung dữ liệu mẫu trong python**

	datetime	Vancouver	Portland
0	2012-10-01 13:00:00	284.630000	282.080000
1	2012-10-01 14:00:00	284.629041	282.083252
2	2012-10-01 15:00:00	284.626998	282.091866
3	2012-10-01 16:00:00	284.624955	282.100481
4	2012-10-01 17:00:00	284.622911	282.109095

Hình 9: Mẫu khung dữ liệu của Python

• Các giá trị được phân tách bằng dấu phẩy – CSV
- Loại tệp đặc biệt cho dữ liệu cảm biến
- Định dạng tệp dễ dàng
    i. Có thể chỉnh sửa trên mỗi bàn tay
- Các giá trị được phân tách bằng dấu phẩy
    1. Phụ thuộc vào cài đặt quốc gia
    ii. Dấu chấm phẩy Đức
    iii. Dấu phẩy Mỹ
    iv. Có thể đọc trực tiếp trong excel
- • Sử dụng thư viện pandas

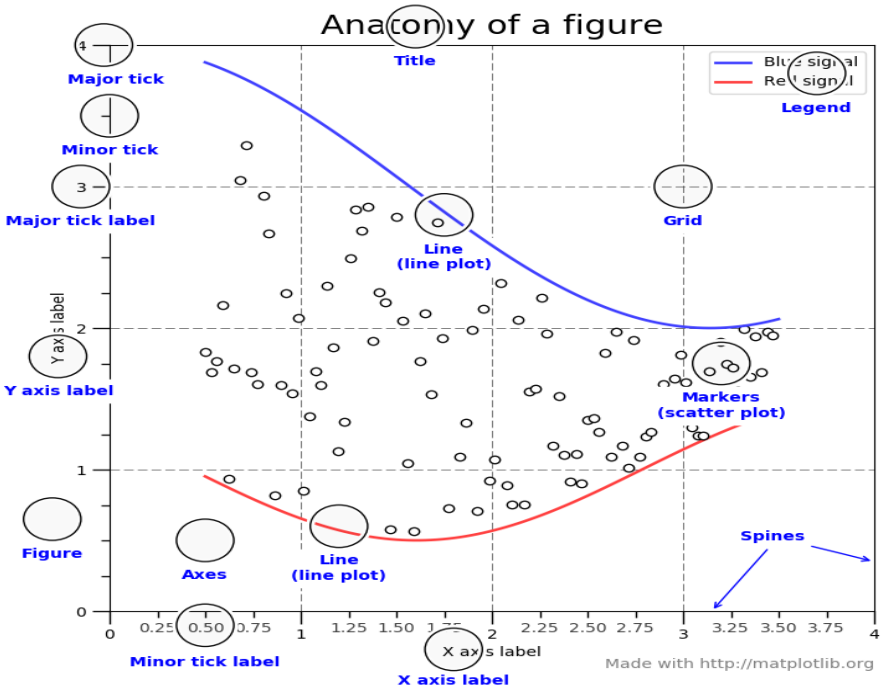
- Cài đặt gấu trúc: pip3 cài đặt pandas
- Nhập gấu trúc làm lớp: nhập pandas làm pd
- Sử dụng hàm thành viên: df = pd.readcsv("filename.csv")
- Lưu trữ dataframe trong biến: df.tocsv("tên tệp.csv")

b. Chuẩn bị và trực quan hóa cục bộ dữ liệu bằng **phương pháp** định hướng đối tượng

- Matplotlib
- Thư viện để vẽ và hình dung dữ liệu
- Tương thích với nhiều thư viện khác
- Được sử dụng trong nhiều thư viện khác
- Trang web: https://matplotlib.org/tutorials/index.html

- Trực quan hóa dữ liệu

- Dữ liệu được hiển thị trong số Hình
  - Khu vực xác định
  - Cấu hình
  - Chứa cốt truyện
- Âm mưu
  - Sơ đồ được sử dụng để hiển thị dữ liệu
- Các bộ phận của hình



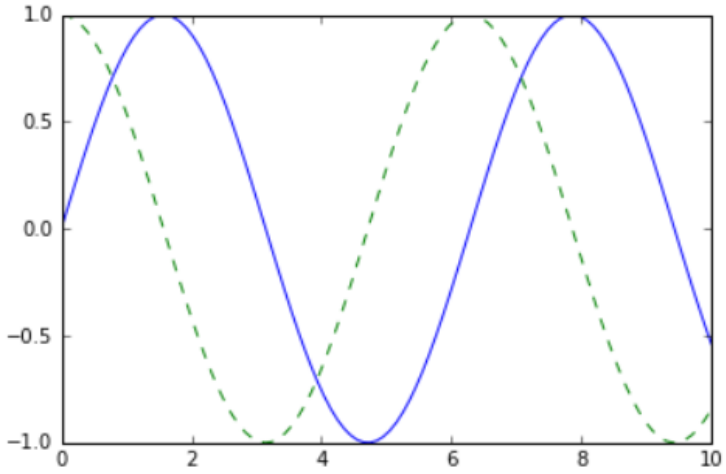
Hình 10: Giải thích các bộ phận của hình

- Tạo giao diện người dùng đồ họa

- Cài đặt Matplotlib: pip3 cài đặt Matplotlib
- Nhập pandas dưới dạng lớp : nhập Matplotlib dưới dạng mpl

- **Ví dụ**

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 10, 100)
fig = plt.Hình()
plt.plot(x, np.sin(x), '-')
plt.plot(x, np.cos(x), '--');
```



Hình 11: Giao diện người dùng đồ họa

2. Ngôn ngữ mô hình hóa thống nhất UML

2.1 Kiến thức cơ bản về UML?

Đây là ngôn ngữ mô hình hóa thống nhất được sử dụng để hình dung hệ thống. Đây là một ngôn ngữ đồ họa là tiêu chuẩn cho ngành công nghiệp phần mềm để chỉ định, hình dung, xây dựng và ghi lại các hiện vật của các hệ thống phần mềm, cũng như cho mô hình kinh doanh.

Lợi ích của UML:

Đơn giản hóa thiết kế phần mềm phức tạp, cũng có thể thực hiện OOP như một khái niệm được sử dụng rộng rãi.

Nó làm giảm hàng ngàn từ giải thích trong một vài sơ đồ đồ họa có thể làm giảm thời gian tiêu thụ để hiểu.

Nó làm cho giao tiếp rõ ràng hơn và thực tế hơn.

Nó giúp để có được toàn bộ hệ thống trong một cái nhìn.

Nó trở nên rất dễ dàng cho các lập trình viên phần mềm để thực hiện nhu cầu thực tế một khi họ có một bức tranh rõ ràng về vấn đề.

Các loại UML: Các sơ đồ UML được chia thành hai phần: Sơ đồ UML cấu trúc và sơ đồ UML hành vi được liệt kê dưới đây:

Sơ đồ UML cấu trúc

Sơ đồ lớp

Sơ đồ gói

Sơ đồ đối tượng

Sơ đồ thành phần
Sơ đồ cấu trúc composite
Sơ đồ triển khai
Sơ đồ UML hành vi
Sơ đồ hoạt động
Sơ đồ trình tự
Sử dụng sơ đồ trường hợp
Sơ đồ trạng thái
Sơ đồ truyền thông
……..

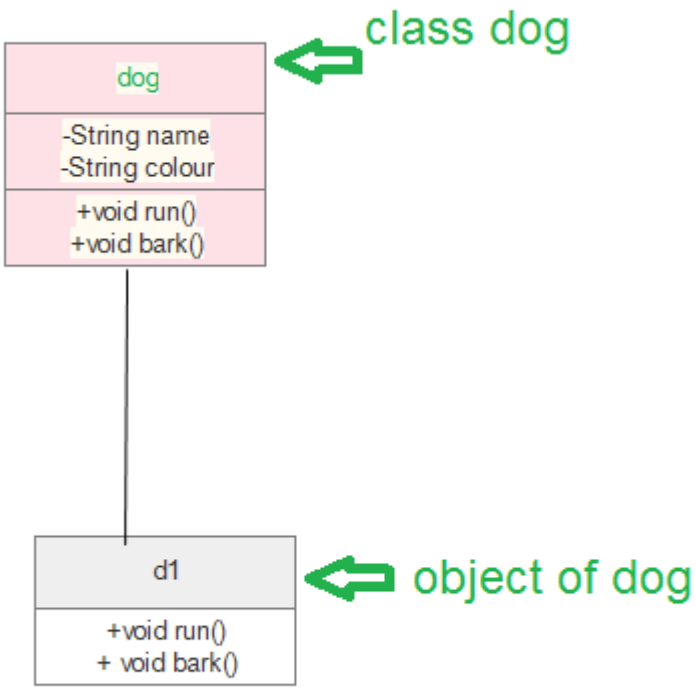
2.2 Sơ đồ class để mô tả các yêu cầu và tài liệu phần

Sơ đồ lớp UML: Sơ đồ lớp là khối xây dựng chính của mọi phương pháp hướng đối tượng. Sơ đồ lớp có thể được sử dụng để hiển thị các lớp, mối quan hệ, giao diện, liên kết và cộng tác. UML được tiêu chuẩn hóa trong sơ đồ lớp học. Vì các lớp là khối xây dựng của một ứng dụng dựa trên OOP, vì vậy sơ đồ lớp có cấu trúc thích hợp để đại diện cho các lớp, kế thừa, mối quan hệ và mọi thứ mà OOP có trong bối cảnh của chúng. Nó mô tả các loại đối tượng khác nhau và mối quan hệ tĩnh giữa chúng.
Mục đích chính để sử dụng sơ đồ lớp là:
Đây là UML duy nhất có thể mô tả thích hợp các khía cạnh khác nhau của khái niệm OOPs.
Thiết kế và phân tích ứng dụng thích hợp có thể nhanh hơn và hiệu quả hơn.
Nó là cơ sở để triển khai và sơ đồ thành phần.
Có một số phần mềm có sẵn có thể được sử dụng trực tuyến và ngoại tuyến để vẽ các sơ đồ này như Edraw max, lucid Biểu đồ, v.v. Có một số điểm cần được giữ tập trung trong khi vẽ sơ đồ lớp học. Đây có thể được nói như cú pháp của nó:
Mỗi lớp được đại diện bởi một hình chữ nhật có một phân khu gồm ba tên, thuộc tính và hoạt động.
Có ba loại sửa đổi được sử dụng để quyết định khả năng hiển thị của các thuộc tính và hoạt động.

+ được sử dụng cho khả năng hiển thị công cộng (cho tất cả mọi người)
# được sử dụng để hiển thị được bảo vệ (cho bạn bè và có nguồn gốc)
– được sử dụng cho khả năng hiển thị riêng tư (chỉ dành cho tôi)

Dưới đây là ví dụ về lớp Động vật (phụ huynh) có hai lớp con là chó và mèo đều có tính chất thừa kế đối tượng d1, c1 của lớp mẹ.
Sơ đồ lớp cũng được coi là nền tảng cho các sơ đồ thành phần và triển khai. Sơ đồ lớp không chỉ được sử dụng để hình dung chế độ xem tĩnh của hệ thống mà còn được sử dụng để xây dựng mã thực thi cho kỹ thuật chuyển tiếp và đảo ngược của bất kỳ hệ thống nào.
Sơ đồ lớp thể hiện rõ bản đồ với các ngôn ngữ hướng đối tượng như Java, C++, v.v. Từ kinh nghiệm thực tế, một sơ đồ lớp học thường được sử dụng cho mục đích xây dựng.

Tóm lại, có thể nói, sơ đồ lớp được sử dụng cho −
Mô tả chế độ xem tĩnh của hệ thống.
Hiển thị sự hợp tác giữa các yếu tố của chế độ xem tĩnh.



Biểu đồ 7: Ví dụ về kế thừa các thuộc tính của lớp cha

Mô tả các chức năng được thực hiện bởi hệ thống.
Xây dựng các ứng dụng phần mềm sử dụng ngôn ngữ hướng đối tượng.

```
import java.io.*;
class GFG {
	public static void main(String[] args)
	{
		dog d1 = new dog();
		d1.bark();
		d1.run();
		cat c1 = new cat();
		c1.meww();
	}
}
class Animal {
	public void run()
	{
		String name;
		String colour;
		System.out.println("animal is running");
```

```
        }
}
class dog extends Animal {
        public void bark()
        {
                System.out.println("wooh!wooh! dog is barking");
        }
        public void run()
        {
                System.out.println("dog is running");
        }
}
class cat extends Animal {
        public void meww()
        {
                System.out.println("meww! meww!");
        }
}
```

Quy trình thiết kế sơ đồ lớp học: Trong Edraw max (hoặc bất kỳ nền tảng nào khác có thể vẽ sơ đồ lớp) làm theo các bước:
Mở một tài liệu trống trong phần sơ đồ lớp.
Từ thư viện chọn sơ đồ lớp và bấm vào tùy chọn tạo.
Chuẩn bị mô hình của lớp trên trang mẫu đã mở.
Sau khi chỉnh sửa theo yêu cầu, hãy lưu nó.
Có một số thành phần sơ đồ có thể được sử dụng hiệu quả trong khi thực hiện / chỉnh sửa mô hình. Như sau:

Lớp { tên, thuộc tính, phương pháp}
Đối tượng
Giao diện
Các mối quan hệ {thừa kế, liên kết, khái quát hóa}
Liên kết

Sơ đồ lớp là một trong những sơ đồ được sử dụng rộng rãi nhất trong các lĩnh vực kỹ thuật phần mềm cũng như mô hình hóa kinh doanh.

2.3 Hiển thị và áp dụng các mối quan hệ giai cấp

Association (Liên kết)

Một hiệp hội có thể được dán nhãn bằng cách đặt tên liên kết ở giữa hiệp hội hoặc bằng cách đặt tên vai trò hoặc cả hai đầu của hiệp hội.
Nếu không có tên liên kết hoặc tên vai trò nào được chỉ định, thì tên liên kết mặc định 'có' được giả định

Aggregation (Kết tập)

Tổng hợp là một độ nhớt trong đó một lớp thuộc về một bộ sưu tập

Ví dụ: Đơn hàng có bộ sưu tập orderDetails

Nó được đại diện bởi biểu tượng kim cương được đặt bên cạnh tổng hợp

Kết tập đại diện cho mối quan hệ 'có một' hoặc 'toàn bộ / phần'

Composition (Hợp thành)

Đôi khi một mối quan hệ tổng hợp có thể là một tập hợp mạnh mẽ

Các bộ phận không thể có một cuộc sống của riêng họ

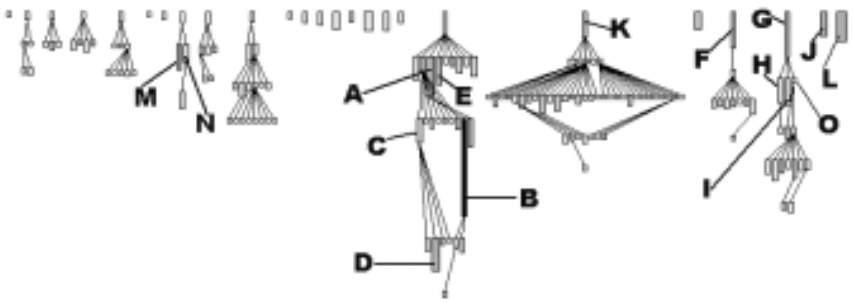
Điều đó có nghĩa là nếu cốt liệu bị phá hủy, thì các bộ phận cũng bị phá hủy.

Một tập hợp mạnh mẽ còn được gọi là một thành phần

Một tập hợp mạnh mẽ được thể hiện bằng một biểu tượng kim cương rắn

2.4 Hiển thị cây kế thừa đối với tài liệu phần mềm

Tổng quan về thừa kế: Tree. Để đánh giá kích thước và độ phức tạp của hệ thống, chúng tôi xây dựng một cây thừa kế. Chúng tôi sử dụng làm kích thước nút số lượng biến phiên bản (chiều rộng) và số lượng phương pháp (chiều cao), trong khi tông màu đại diện cho các dòng mã của lớp.



Hình 12. Tổng quan về kế thừa qua Cây; chiều rộng nút = NIV, chiều cao nút = NOM và màu = WLOC.

Giải thích. Chúng tôi quan sát một vài hệ thống phân cấp chính với tỷ lệ cao các lớp rất nhỏ. Sau đó, sử dụng num-ber của các phương pháp mỗi lớp như một criterium chúng tôi xác định một số

Các lớp thí sinh để điều tra thêm: (1) lớp nhỏ nhất (O) hoàn toàn trống rỗng và (2) 13 lớp khá lớn (từ 40 đến 175 phương pháp) (B).

13 lớp này có thể được phân loại theo po-sition của họ trong cây thừa kế: là một chiếc lá (D, I), nằm trên một hệ thống phân cấp (F, G, K), ở giữa hier-

archy (A, B) hoặc ở một mình (E, J, L). Các lớp anh chị em lớn như (H, I) và (N, M) là những ứng cử viên tốt để phân tích lại vì một số mã có thể được di chuyển lên trong của homsiêu lớp chung. Một ví dụ về khả năng điều tra thêm là lớn lớp được gọi là BrowserNavigator thực hiện 175 meth-ods (đánh dấu B) trong khi siêu lớp Navigator (A) al-ready thực hiện 70 phương pháp. Một trường hợp thú vị khác là lớp gọi là BRScanner (tên L), thực hiện 49 phương pháp và xác định 14 biến phiên bản

2.5 Sơ đồ trình tự để mô tả các quá trình giao tiếp của phần mềm (PC và hệ thống cơ điện tử)

Sơ đồ trình tự – Một sơ đồ trình tự chỉ đơn giản mô tả sự tương tác giữa các đối tượng theo thứ tự tuần tự tức là thứ tự diễn ra các tương tác này. Chúng ta cũng có thể sử dụng các thuật ngữ sơ đồ sự kiện hoặc kịch bản sự kiện để tham khảo sơ đồ trình tự. Sơ đồ trình tự mô tả cách thức và thứ tự các đối tượng trong một chức năng hệ thống. Những sơ đồ này được sử dụng rộng rãi bởi các doanh nhân và nhà phát triển phần mềm để ghi lại và hiểu các yêu cầu cho các hệ thống mới và hiện có.

Ký hiệu Sơ đồ Trình tự

Diễn viên - Một diễn viên trong sơ đồ UML đại diện cho một loại vai trò mà nó tương tác với hệ thống và các đối tượng của nó. Điều quan trọng cần lưu ý ở đây là một diễn viên luôn nằm ngoài phạm vi của hệ thống mà chúng tôi nhắm đến để mô hình hóa bằng cách sử dụng sơ đồ UML.

Chúng tôi sử dụng các diễn viên để mô tả các vai trò khác nhau bao gồm người dùng của con người và các đối tượng bên ngoài khác. Chúng tôi đại diện cho một diễn viên trong sơ đồ UML bằng cách sử dụng một ghi chú người que. Chúng ta có thể có nhiều diễn viên trong một sơ đồ trình tự.

Ví dụ : ở đây người dùng trong hệ thống đặt chỗ được hiển thị như một diễn viên nơi nó tồn tại bên ngoài hệ thống và không phải là một phần của hệ thống

Lifelines – Một huyết mạch là một yếu tố được đặt tên mô tả một người tham gia cá nhân trong một sơ đồ trình tự. Vì vậy, về cơ bản mỗi phiên bản trong một sơ đồ trình tự được thể hiện bằng một đường dây nóng. Các yếu tố đường dây nóng được đặt ở trên cùng trong sơ đồ trình tự. Tiêu chuẩn trong UML để đặt tên cho một đường dây nóng theo định dạng sau đây – Tên phiên bản : Tên lớp

Chúng tôi hiển thị một đường dây nóng trong một hình chữ nhật được gọi là đầu với tên và loại của nó. Đầu nằm trên đỉnh của một đường đứt đoạn thẳng đứng (được gọi là thân cây) như được hiển thị ở trên. Nếu chúng ta muốn mô hình hóa một phiên bản không tên, chúng ta theo cùng một mẫu ngoại trừ bây giờ phần tên của đường dây nóng bị bỏ trống.

Sự khác biệt giữa huyết mạch và diễn viên - Một đường dây nóng luôn miêu tả một đối tượng bên trong hệ thống trong khi các diễn viên được sử dụng để mô tả các đối tượng bên ngoài hệ thống. Sau đây là một ví dụ về sơ đồ trình tự:

Thông điệp – Giao tiếp giữa các đối tượng được mô tả bằng cách sử dụng tin nhắn. Các tin nhắn xuất hiện theo thứ tự tuần tự trên đường dây nóng. Chúng tôi đại diện cho các thư bằng mũi tên. Phao cứu sinh và thông điệp tạo thành cốt lõi của sơ đồ trình tự.

Thư không đồng bộ

Xóa Thư

Tự nhắn tin.

Thư trả lời.

Thư được tìm thấy

Mất tin nhắn.

Bảo vệ – Để mô hình hóa điều kiện chúng tôi sử dụng bảo vệ trong UML. Chúng được sử dụng khi chúng ta cần hạn chế luồng tin nhắn với lý do một điều kiện được đáp ứng. Bảo vệ đóng một vai trò quan trọng trong việc cho các nhà phát triển phần mềm biết những hạn chế gắn liền với một hệ thống hoặc một quy trình cụ thể.

Sử dụng sơ đồ trình tự –

Được sử dụng để mô hình hóa và hình dung logic đằng sau một chức năng, hoạt động hoặc thủ tục tinh vi.

Chúng cũng được sử dụng để hiển thị chi tiết về sơ đồ trường hợp sử dụng UML.

Được sử dụng để hiểu chức năng chi tiết của các hệ thống hiện tại hoặc tương lai.

Hình dung cách các thư và tác vụ di chuyển giữa các đối tượng hoặc thành phần trong một hệ thống.

2.6 Sơ đồ use case để ghi lại các use case cấp cao

Sử dụng sơ đồ trường hợp là một loại sơ đồ UML hành vi và thường được sử dụng để phân tích các hệ thống khác nhau. Chúng cho phép bạn hình dung các loại vai trò khác nhau trong một hệ thống và cách các vai trò đó tương tác với hệ thống. Hướng dẫn sơ đồ trường hợp sử dụng này sẽ bao gồm các chủ đề sau đây và giúp bạn tạo trường hợp sử dụng tốt hơn .

Tầm quan trọng của sơ đồ trường hợp sử dụng

Như đã đề cập trước khi sử dụng sơ đồ trường hợp được sử dụng để thu thập yêu cầu sử dụng của một hệ thống. Tùy thuộc vào yêu cầu của bạn, bạn có thể sử dụng dữ liệu đó theo những cách khác nhau. Dưới đây là một vài cách để sử dụng chúng.

Để xác định các hàm và cách vai trò tương tác với chúng - Mục đích chính của việc sử dụng sơ đồ trường hợp.

Để có cái nhìn cao về hệ thống - Đặc biệt hữu ích khi trình bày cho các nhà quản lý hoặc các bên liên quan. Bạn có thể làm nổi bật các vai trò tương tác với hệ thống và chức năng được cung cấp bởi hệ thống mà không đi sâu vào hoạt động bên trong của hệ thống.

Để xác định các yếu tố bên trong và bên ngoài - Điều này nghe có vẻ đơn giản nhưng trong các dự án phức tạp lớn, một hệ thống có thể được xác định là một vai trò bên ngoài trong một trường hợp sử dụng khác.

Sử dụng đối tượng use case:

Sử dụng sơ đồ use case bao gồm 4 đối tượng.

Tác nhân

Trường hợp sử dụng

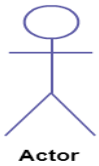
Hệ thống

Gói

Các đối tượng được giải thích thêm dưới đây.

Tác nhân

Diễn viên trong sơ đồ trường hợp sử dụng là bất kỳ thực thể nào thực hiện vai trò trong một hệ thống nhất định. Đây có thể là một người, tổ chức hoặc một hệ thống bên ngoài và thường được vẽ như bộ xương được hiển thị bên dưới.

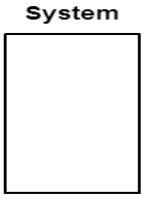


Trường hợp Sử dụng

Một trường hợp sử dụng đại diện cho một chức năng hoặc một hành động trong hệ thống. Nó được vẽ như một hình bầu dục và được đặt tên với chức năng.

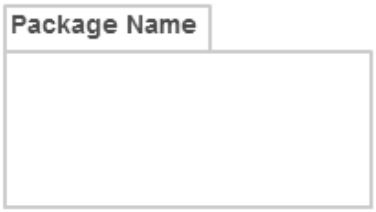
Hệ thống

Hệ thống được sử dụng để xác định phạm vi của trường hợp sử dụng và được vẽ dưới dạng hình chữ nhật. Đây là một yếu tố tùy chọn nhưng hữu ích khi bạn hình dung các hệ thống lớn. Ví dụ: bạn có thể tạo tất cả các trường hợp sử dụng và sau đó sử dụng đối tượng hệ thống để xác định phạm vi được bao phủ bởi dự án của bạn. Hoặc bạn thậm chí có thể sử dụng nó để hiển thị các khu vực khác nhau được đề cập trong các bản phát hành khác nhau.



Gói

Gói là một yếu tố tùy chọn khác cực kỳ hữu ích trong các sơ đồ phức tạp. Tương tự như sơ đồ lớp học, các gói được sử dụng để nhóm các trường hợp sử dụng cùng nhau. Chúng được vẽ giống như hình ảnh được hiển thị dưới đây.



2.7 Thực hiện sơ đồ UML

2.7.1 Tạo lớp bằng sơ đồ UML

Cho đến nay, bạn đã tìm hiểu về các đối tượng, mối quan hệ và hướng dẫn rất quan trọng khi vẽ sơ đồ trường hợp sử dụng. Tôi sẽ giải thích các quy trình khác nhau bằng cách sử dụng một hệ thống ngân hàng làm ví dụ.

Xác định tác nhân

Tác nhân là các thực thể bên ngoài tương tác với hệ thống của bạn. Nó có thể là một người, một hệ thống khác hoặc một tổ chức. Trong một hệ thống ngân hàng, tác nhân rõ ràng nhất là khách hàng. Các diễn viên khác có thể là nhân viên ngân hàng hoặc nhân viên thu ngân tùy thuộc vào vai trò bạn đang cố gắng thể hiện trong trường hợp sử dụng. Một ví dụ về một tổ chức bên ngoài có thể là cơ quan thuế hoặc ngân hàng trung ương. Bộ xử lý khoản vay là một ví dụ điển hình về một hệ thống bên ngoài được liên kết như một diễn viên.

Xác định các trường hợp sử dụng

Bây giờ là lúc để xác định các trường hợp sử dụng. Một cách tốt để làm điều này là xác định những gì các diễn viên cần từ hệ thống. Trong một hệ thống ngân hàng, khách hàng sẽ cần mở tài khoản, gửi và rút tiền, yêu cầu sổ séc và các chức năng tương tự. Vì vậy, tất cả những điều này có thể được coi là trường hợp sử dụng.

Các trường hợp sử dụng cấp cao nhất phải luôn cung cấp một chức năng hoàn chỉnh theo yêu cầu của một diễn viên. Bạn có thể mở rộng hoặc bao gồm các trường hợp sử dụng tùy thuộc vào độ phức tạp của hệ thống.

Một khi bạn xác định các diễn viên và trường hợp sử dụng cấp cao nhất, bạn có một ý tưởng cơ bản về hệ thống. Bây giờ bạn có thể tinh chỉnh nó và thêm các lớp chi tiết bổ sung vào nó.

Tìm kiếm chức năng phổ biến để sử dụng Bao gồm

Tìm kiếm các chức năng phổ biến có thể được tái sử dụng trên toàn hệ thống. Nếu bạn tìm thấy hai hoặc nhiều trường hợp sử dụng chia sẻ chức năng chung, bạn có thể trích xuất các chức năng phổ biến và thêm nó vào một trường hợp sử dụng riêng biệt. Sau đó, bạn có thể kết nối nó thông qua mối quan hệ bao gồm để cho thấy rằng nó luôn được gọi khi trường hợp sử dụng ban đầu được thực hiện. (xem sơ đồ cho một ví dụ).

Có thể khái quát hóa các diễn viên và sử dụng các trường hợp

Có thể có những trường hợp diễn viên có liên quan đến các trường hợp sử dụng tương tự trong khi kích hoạt một vài trường hợp sử dụng duy nhất chỉ dành cho họ. Trong những trường hợp như vậy, bạn có thể khái quát hóa diễn viên để hiển thị sự kế thừa của các hàm. Bạn cũng có thể làm một điều tương tự cho trường hợp sử dụng.

Một trong những ví dụ tốt nhất về điều này là trường hợp sử dụng "Thanh toán" trong hệ thống thanh toán. Bạn có thể khái quát thêm nó thành "Thanh toán bằng thẻ tín dụng", "Thanh toán bằng tiền mặt", "Thanh toán bằng séc", v.v. Tất cả chúng đều có các thuộc tính và chức năng thanh toán với các kịch bản đặc biệt duy nhất cho chúng.

Hàm tùy chọn hoặc Hàm bổ sung

Có một số chức năng được kích hoạt tùy chọn. Trong những trường hợp như vậy, bạn có thể sử dụng mối quan hệ mở rộng và đính kèm quy tắc mở rộng vào nó. Trong ví dụ hệ thống ngân hàng dưới đây "Tính tiền thưởng" là tùy chọn và chỉ kích hoạt khi một điều kiện nhất định được khớp.

Mở rộng không phải lúc nào cũng có nghĩa là nó là tùy chọn. Đôi khi trường hợp sử dụng được kết nối bằng cách mở rộng có thể bổ sung cho trường hợp sử dụng cơ

bản. Điều cần nhớ là trường hợp sử dụng cơ sở sẽ có thể tự thực hiện một chức năng ngay cả khi trường hợp sử dụng mở rộng không được gọi.

2.7.2. Thực hiện sơ đồ trình tự cho các nhiệm vụ được xác định

Một sơ đồ trình tự được cấu trúc theo cách mà nó đại diện cho một dòng thời gian bắt đầu ở trên cùng và giảm dần để đánh dấu chuỗi tương tác. Mỗi đối tượng có một cột và các thông điệp trao đổi giữa chúng được thể hiện bằng mũi tên.

Tổng quan nhanh về các phần khác nhau của sơ đồ trình tự

Lifeline Ghi nhận

Một sơ đồ trình tự được tạo thành từ một số ký hiệu huyết mạch này nên được sắp xếp theo chiều ngang trên đầu sơ đồ. Không có hai niên cẩm nào nên chồng chéo lên nhau. Chúng đại diện cho các đối tượng hoặc bộ phận khác nhau tương tác với nhau trong hệ thống trong chuỗi.

Thanh Kích hoạt

Thanh kích hoạt là hộp được đặt trên đường dây nóng. Nó được sử dụng để chỉ ra rằng một đối tượng đang hoạt động (hoặc tức thời) trong quá trình tương tác giữa hai đối tượng. Chiều dài của hình chữ nhật cho biết thời gian của các đối tượng vẫn hoạt động.

Mũi tên Thư

Mũi tên từ Người gọi Thư đến Người nhận Thư chỉ định thư trong sơ đồ trình tự. Một thông điệp có thể chảy theo bất kỳ hướng nào; từ trái sang phải, từ phải sang trái hoặc trở lại chính Người gọi Thông điệp. Mặc dù bạn có thể mô tả thư được gửi từ đối tượng này sang đối tượng khác trên mũi tên, nhưng với các đầu mũi tên khác nhau, bạn có thể chỉ ra loại thư được gửi hoặc nhận

Thư đồng bộ

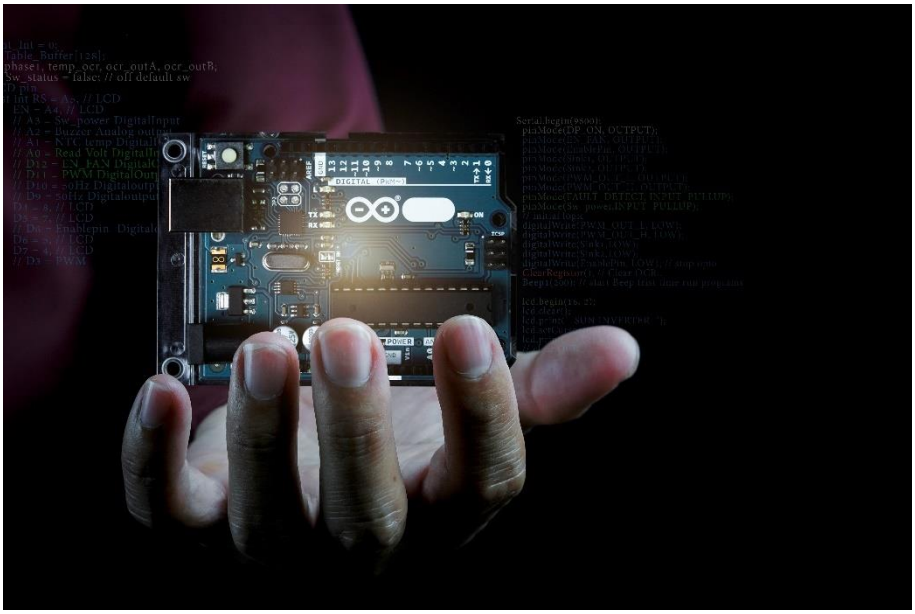
Thông điệp không đồng bộ

Thư trả về

Thông báo tạo người tham gia

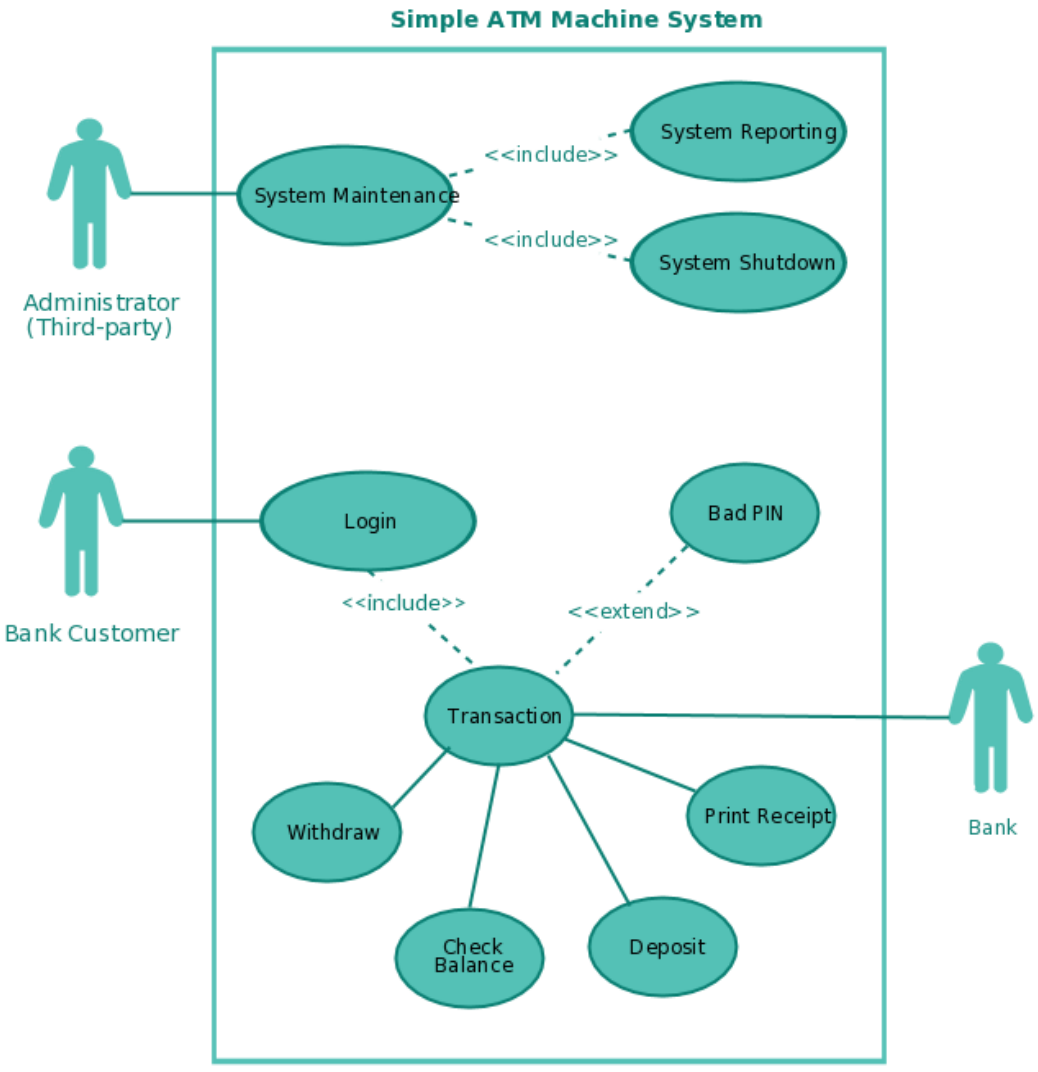
Thông báo hủy diệt người tham gia

Thông điệp phản xạ



Hình minh họa 4: phần tử bảng điều khiển điện tử

2.7.3 Tạo Sơ đồ trường hợp sử dụng cho các ứng dụng mẫu



Hình 13: Minh họa hệ thống máy ATM

Chúng tôi đã đi trước và tạo các mẫu sơ đồ trường hợp sử dụng cho một số tình huống phổ biến. Mặc dù vấn đề hoặc kịch bản của bạn sẽ không chính xác như thế này, bạn có thể sử dụng chúng như một điểm khởi đầu.

3. Phân tích nhiệm vụ và tìm giải pháp

Phân tích các đơn đặt hàng kỹ thuật và phát triển các giải pháp:

Phân tích yêu cầu của khách hàng liên quan đến chức năng cần thiết, Làm rõ thông số kỹ thuật khi trao đổi với khách hàng

Ví dụ: Tạo mã QR bằng Python

Mã QR là viết tắt của Mã phản hồi nhanh. Mã QR có thể trông đơn giản nhưng chúng có khả năng lưu trữ nhiều dữ liệu. Bất kể chúng chứa bao nhiêu dữ liệu khi mã QR được quét cho phép người dùng truy cập thông tin ngay lập tức. Đó là lý do tại sao chúng được gọi là Mã phản hồi nhanh.

Những thứ này đang được sử dụng trong nhiều trường hợp ngày nay. Nó xuất hiện lần đầu tiên ở Nhật Bản vào năm 1994. Mã QR có thể được sử dụng để lưu trữ (mã hóa) rất nhiều dữ liệu và nhiều loại khác nhau. Ví dụ, chúng có thể được sử dụng để mã hóa:

i. Chi tiết liên hệ

ii. Id Facebook, id Instagram, Twitter, WhatsApp, và hơn thế nữa.

iii. Chi tiết sự kiện

iv. Liên kết Youtube

v. Chi tiết sản phẩm

vi. Liên kết trực tiếp để tải xuống ứng dụng trên Apple App Store hoặc Google Play.

vii. Chúng cũng đang được sử dụng để thực hiện các giao dịch kỹ thuật số chỉ bằng cách quét mã QR.

viii. Truy cập Wi-Fi bằng cách lưu trữ các chi tiết mã hóa như SSID, mật khẩu và loại mã hóa.

……

Chúng ta vừa thấy một số ưu điểm của mã QR. Bây giờ chúng ta sẽ tìm hiểu ở đây cách chúng ta có thể tạo mã QR bằng Python.

Để tạo mã QR bằng python, chúng tôi sẽ sử dụng một mô-đun python được gọi là QRcode.

Link: https://pypi.org/project/qrcode/

Cài đặt nó bằng lệnh này: **pip install qrcode**

Chúng tôi sẽ tạo Mã QR để mã hóa liên kết youtube và chúng tôi cũng sẽ khám phá thêm. Tạo mã QR rất đơn giản. Chỉ cần chuyển văn bản, liên kết hoặc bất kỳ nội dung nào đến chức năng ‘make’ của mô-đun QRcode.

```
import qrcode
img = qrcode.make("https://www.youtube.com/")
```

```
img.save("youtubeQR.jpg")
```

Khi thực thi đầu ra mã này là:

Bạn có thể quét nó và xác minh.

Bạn có thể thấy chỉ cần 3 dòng mã để tạo Mã QR này. Một điều nữa cần đề cập là bạn không nhất thiết phải cung cấp liên kết đến hàm qrcode.make (). Bạn cũng có thể cung cấp văn bản đơn giản.

Ví dụ:

Bạn có thể mã hóa: Ấn Độ là một quốc gia có nhiều tôn giáo. Tôi yêu nước Ấn Độ.

Hãy thử nó ra:

```
import qrcode
img = qrcode.make("India is a country with many religions. I love India.")
img.save("youtubeQR.jpg")
```

Mã QR đầu ra cho văn bản này là:

Quét nó từ điện thoại di động của bạn và bạn sẽ nhận được nội dung.

Vì vậy, đây là một phần, liên quan đến việc tạo mã QR và quét nó. Nhưng điều gì sẽ xảy ra nếu chúng ta muốn đọc Mã QR này, tức là bây giờ chúng ta muốn biết những gì đã được mã hóa trong Mã QR mà không cần quét nó. Đối với điều này, chúng tôi sẽ sử dụng OpenCV. OpenCV là một thư viện các hàm lập trình tập trung vào các tác vụ thị giác máy tính thời gian thực.

Cài đặt opencv: **pip install opencv-python**
Mã để giải mã các mã QR trở lại để biết chuỗi ban đầu.

```
import cv2
d = cv2.QRCodeDetector()
val, _, _ = d.detectAndDecode(cv2.imread("myQRCode.jpg"))
print("Decoded text is: ", val)
```

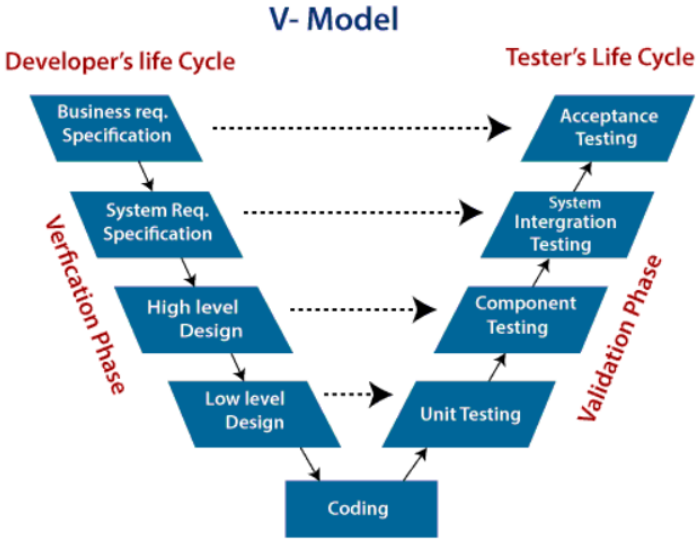
Ngõ ra:

```
India is a country with many religions. I love India.
```

4 Kiểm tra phần mềm và triển khai

4.1 Mô hình V

V-Model còn được gọi là Mô hình xác minh và xác thực. Trong điều này, mỗi giai đoạn của Vòng đời phát triển phần mềm (SDLC) phải hoàn thành trước khi giai đoạn tiếp theo bắt đầu. Nó tuân theo một quy trình thiết kế tuần tự giống như mô hình thác nước. Việc thử nghiệm thiết bị được lên kế hoạch song song với một giai đoạn phát triển tương ứng.



Hình 14: Minh họa về Mô hình xác minh và xác thực

Verfication: Nó liên quan đến một phương pháp phân tích tĩnh (xem xét) được thực hiện mà không cần thực thi mã. Đây là quá trình đánh giá quá trình phát triển sản phẩm để tìm xem liệu các yêu cầu cụ thể có đáp ứng hay không.
Validation: Nó liên quan đến phương pháp phân tích động (chức năng, phi chức năng), kiểm tra được thực hiện bằng cách thực thi mã. Xác nhận là quá trình phân loại phần mềm sau khi hoàn thành quá trình phát triển để xác định xem phần mềm có đáp ứng được các yêu cầu và mong đợi của khách hàng hay không.
Vì vậy, V-Model chứa các giai đoạn Xác minh ở một bên của các giai đoạn Xác thực ở phía bên kia. Quá trình xác minh và xác thực được tham gia bởi giai đoạn mã hóa theo hình chữ V. Vì vậy, nó được gọi là V-Model.

Có các giai đoạn khác nhau của Giai đoạn xác minh của mô hình V:

i. Phân tích yêu cầu kinh doanh: Đây là bước đầu tiên mà các yêu cầu về sản phẩm được hiểu từ phía khách hàng. Giai đoạn này bao gồm thông tin liên lạc chi tiết để hiểu mong đợi của khách hàng và yêu cầu chính xác.

ii. Thiết kế hệ thống: Trong giai đoạn này, các kỹ sư hệ thống phân tích và diễn giải hoạt động kinh doanh của hệ thống được đề xuất bằng cách nghiên cứu tài liệu yêu cầu của người dùng.

iii. Thiết kế kiến trúc: Cơ sở trong việc lựa chọn kiến trúc là nó phải hiểu tất cả những gì thường bao gồm danh sách các mô-đun, chức năng ngắn gọn của từng mô-đun, các mối quan hệ giao diện của chúng, sự phụ thuộc, bảng cơ sở dữ liệu, sơ đồ kiến trúc, chi tiết công nghệ, v.v. Kiểm thử tích hợp mô hình được thực hiện trong một giai đoạn cụ thể.

iv. Thiết kế mô-đun: Trong giai đoạn thiết kế mô-đun, hệ thống chia thành các mô-đun nhỏ. Thiết kế chi tiết của các mô-đun được chỉ định, được gọi là Thiết kế cấp thấp

v. Giai đoạn mã hóa: Sau khi thiết kế, giai đoạn mã hóa được bắt đầu. Dựa trên các yêu cầu, một ngôn ngữ lập trình phù hợp được quyết định. Có một số hướng dẫn và tiêu chuẩn để mã hóa. Trước khi kiểm tra trong kho lưu trữ, bản dựng cuối cùng được tối ưu hóa để có hiệu suất tốt hơn và mã trải qua nhiều lần đánh giá mã để kiểm tra hiệu suất.

Có các giai đoạn khác nhau của Giai đoạn xác thực của mô hình V:

i. Kiểm thử đơn vị: Trong V-Model, các Kế hoạch kiểm tra đơn vị (UTP) được phát triển trong giai đoạn thiết kế mô-đun. Các UTP này được thực thi để loại bỏ lỗi ở cấp mã hoặc cấp đơn vị. Đơn vị là thực thể nhỏ nhất có thể tồn tại độc lập, ví dụ: mô-đun chương trình. Kiểm tra đơn vị xác minh rằng thực thể nhỏ nhất có thể hoạt động chính xác khi bị cô lập với phần còn lại của mã / đơn vị.

ii. Kiểm tra Tích hợp: Các Kế hoạch Kiểm tra Tích hợp được phát triển trong Giai đoạn Thiết kế Kiến trúc. Các bài kiểm tra này xác minh rằng các nhóm được tạo và kiểm tra độc lập có thể cùng tồn tại và giao tiếp với nhau.

iii. Kiểm tra Hệ thống: Các Kế hoạch Kiểm tra Hệ thống được phát triển trong Giai đoạn Thiết kế Hệ thống. Không giống như Kế hoạch kiểm tra đơn vị và tích hợp, Kế hoạch kiểm tra hệ thống do nhóm kinh doanh của khách hàng soạn thảo. Kiểm tra Hệ thống đảm bảo rằng các kỳ vọng từ một nhà phát triển ứng dụng được đáp ứng.

iv. Kiểm tra chấp nhận: Kiểm tra chấp nhận có liên quan đến phần phân tích yêu cầu nghiệp vụ. Nó bao gồm việc thử nghiệm sản phẩm phần mềm trong môi trường người dùng. Kiểm tra chấp nhận cho thấy các vấn đề tương thích với các hệ thống khác nhau, có sẵn trong môi trường người dùng. Nó đồng thời phát hiện ra các vấn đề phi chức năng như lỗi tải và hiệu suất trong môi trường người dùng thực.

Khi nào thì sử dụng V-Model?

o Khi yêu cầu được xác định rõ ràng và không mơ hồ.

o Mô hình hình chữ V nên được sử dụng cho các dự án quy mô vừa và nhỏ, nơi các yêu cầu được xác định rõ ràng và cố định.

o Mô hình hình chữ V nên được chọn khi có sẵn nguồn lực kỹ thuật mẫu với chuyên môn kỹ thuật thiết yếu.

Ưu điểm (Ưu điểm) của V-Model:

- Dễ hiểu.
- Kiểm tra Các phương pháp như lập kế hoạch, thiết kế kiểm thử diễn ra tốt trước khi viết mã.
- Điều này giúp tiết kiệm rất nhiều thời gian. Do đó, cơ hội thành công cao hơn so với mô hình thác nước.
- Tránh dòng chảy xuống của các khuyết tật.
- Hoạt động tốt cho các kế hoạch nhỏ, nơi các yêu cầu dễ hiểu.

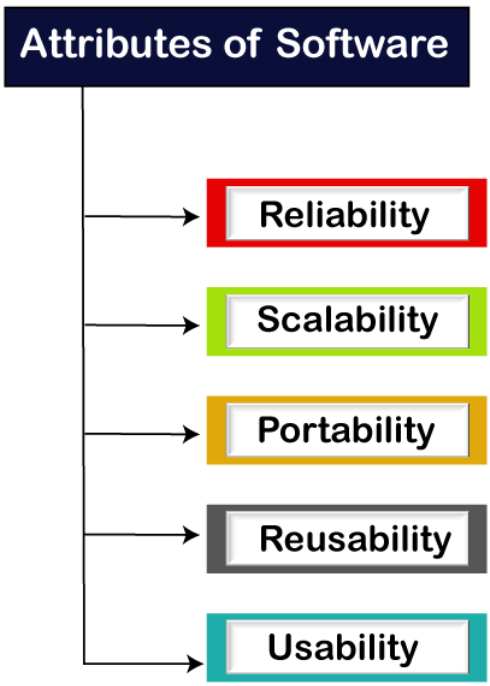
Nhược điểm (Nhược điểm) của V-Model:

- Rất cứng nhắc và kém linh hoạt nhất.
- Không tốt cho một dự án phức tạp.
- Phần mềm được phát triển trong giai đoạn thực hiện, do đó không có nguyên mẫu ban đầu của phần mềm được sản xuất.
- Nếu có bất kỳ thay đổi nào xảy ra giữa chừng, thì các tài liệu kiểm tra cùng với các tài liệu bắt buộc phải được cập nhật.

4.2 Kiểm thử phần mềm

Kiểm thử phần mềm là gì

Kiểm thử phần mềm là một quá trình xác định tính đúng đắn của phần mềm bằng cách xem xét tất cả các thuộc tính của nó (Độ tin cậy, Khả năng mở rộng, Tính khả chuyển, Khả năng sử dụng lại, Khả năng sử dụng) và đánh giá việc thực thi của các thành phần phần mềm để tìm ra lỗi hoặc lỗi hoặc khiếm khuyết của phần mềm.



Biểu đồ 8: Quy trình kiểm thử phần mềm

Kiểm thử phần mềm cung cấp một cái nhìn độc lập và mục tiêu của phần mềm và đảm bảo tính phù hợp của phần mềm. Nó liên quan đến việc kiểm tra tất cả các thành phần trong các dịch vụ được yêu cầu để xác nhận rằng liệu nó có đáp ứng các yêu cầu được chỉ định hay không. Quá trình này cũng cung cấp cho khách hàng thông tin về chất lượng của phần mềm.

Kiểm tra là bắt buộc vì nó sẽ là một tình huống nguy hiểm nếu phần mềm bị lỗi bất kỳ lúc nào do thiếu kiểm tra. Vì vậy, không có phần mềm kiểm tra không thể được triển khai cho người dùng cuối.

Thử nghiệm là gì

Kiểm thử là một nhóm các kỹ thuật để xác định tính đúng đắn của ứng dụng theo tập lệnh được xác định trước, nhưng kiểm thử không thể tìm thấy tất cả các khiếm khuyết của ứng dụng. Mục đích chính của kiểm thử là phát hiện các lỗi của ứng dụng để các lỗi có thể được phát hiện và sửa chữa. Nó không chứng minh rằng một sản phẩm hoạt động bình thường trong mọi điều kiện mà chỉ là nó không hoạt động trong một số điều kiện cụ thể.

Thử nghiệm cung cấp phép so sánh so sánh hành vi và trạng thái của phần mềm với các cơ chế vì cơ chế có thể nhận ra vấn đề. Cơ chế này có thể bao gồm các phiên bản trước đây của cùng một sản phẩm được chỉ định, các sản phẩm có thể so sánh và các giao diện có mục đích mong đợi, các tiêu chuẩn liên quan hoặc các tiêu chí khác nhưng không giới hạn ở những điều này.

Kiểm tra bao gồm kiểm tra mã và cũng như thực thi mã trong các môi trường, điều kiện khác nhau cũng như tất cả các khía cạnh kiểm tra của mã. Trong kịch bản hiện tại của phát triển phần mềm, một nhóm kiểm thử có thể tách biệt với nhóm phát triển để Thông tin thu được từ kiểm thử có thể được sử dụng để điều chỉnh quá trình phát triển phần mềm.

Sự thành công của phần mềm phụ thuộc vào việc chấp nhận đối tượng mục tiêu của nó, giao diện người dùng đồ họa dễ dàng, kiểm tra tải chức năng mạnh mẽ, v.v. Ví dụ, đối tượng của ngân hàng hoàn toàn khác với đối tượng của trò chơi điện tử. Do đó, khi một tổ chức phát triển một sản phẩm phần mềm, tổ chức có thể đánh giá liệu sản phẩm phần mềm đó có mang lại lợi ích cho người mua và các đối tượng khác hay không.

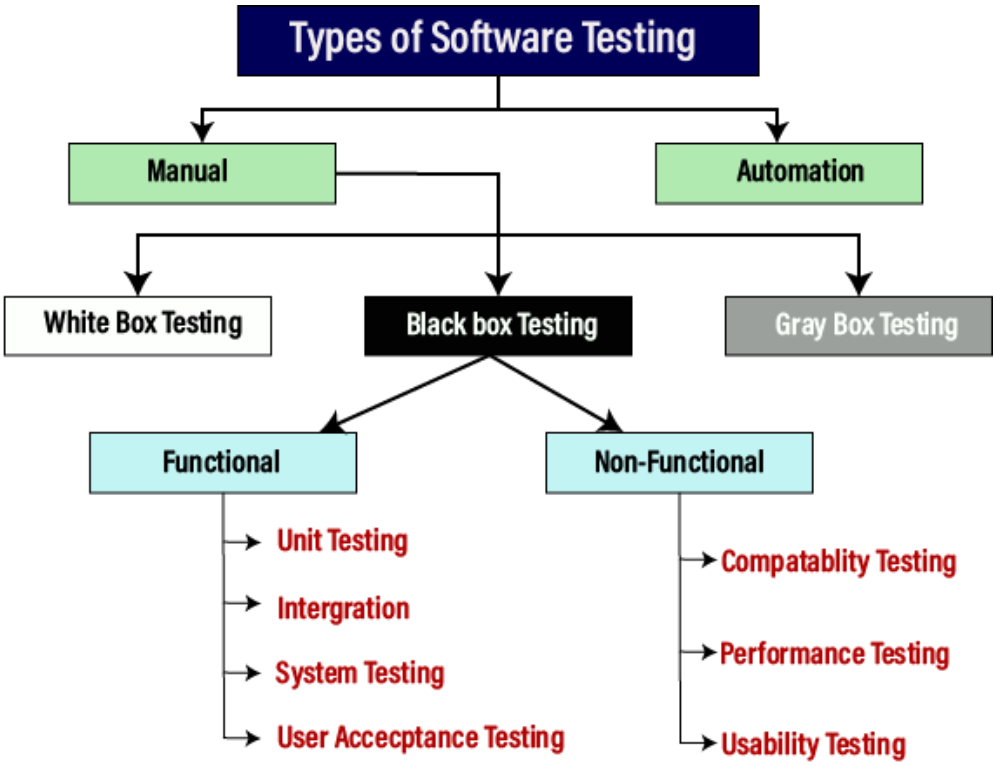
Các mục tiêu kiểm tra

- Đánh giá, đạt được và bảo toàn chất lượng và an toàn
- Tìm càng nhiều lỗi nghiêm trọng càng tốt
- Tìm lỗi càng sớm càng tốt
- Tránh chi phí cao hơn từ việc tìm kiếm lỗi muộn
- Trên thực tế, kiểm tra là cách duy nhất để chứng minh rằng hệ thống hoạt động

Type of Software testing

Chúng tôi có nhiều loại thử nghiệm khác nhau có sẵn trên thị trường, được sử dụng để kiểm tra ứng dụng hoặc phần mềm.

Với sự trợ giúp của hình ảnh dưới đây, chúng ta có thể dễ dàng hiểu loại kiểm thử phần mềm:



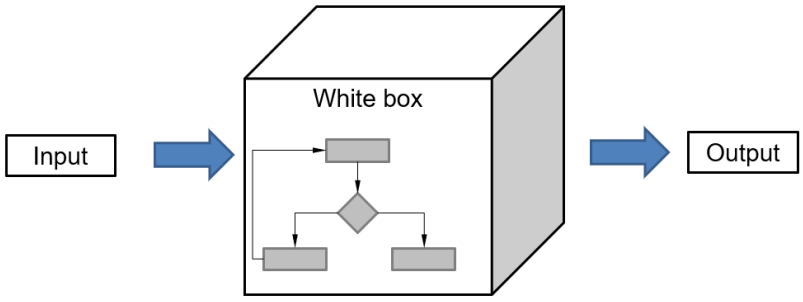
Biểu đồ 9: Các loại kiểm thử phần mềm

Kiểm tra thủ công

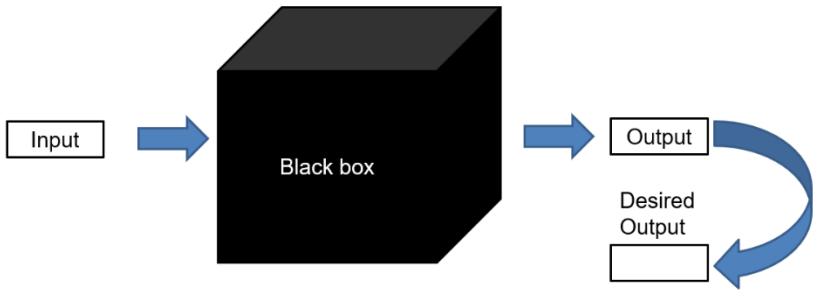
Quá trình kiểm tra chức năng của một ứng dụng theo nhu cầu của khách hàng mà không cần đến bất kỳ sự trợ giúp nào của các công cụ tự động hóa được gọi là kiểm tra thủ công. Trong khi thực hiện kiểm tra thủ công trên bất kỳ ứng dụng nào, chúng tôi không cần bất kỳ kiến thức cụ thể nào về bất kỳ công cụ kiểm tra nào, thay vì phải hiểu đúng về sản phẩm để chúng tôi có thể dễ dàng chuẩn bị tài liệu kiểm tra.

Thử nghiệm thủ công có thể được chia thành ba loại thử nghiệm, như sau:

- Kiểm tra hộp trắng



- Kiểm tra hộp đen



- Kiểm tra hộp xám

Hình 15: Các loại kiểm tra thủ công khác nhau

Kiểm tra tự động hóa

Kiểm thử tự động hóa là một quá trình chuyển đổi bất kỳ trường hợp kiểm thử thủ công nào thành các kịch bản kiểm thử với sự trợ giúp của các công cụ tự động hóa hoặc bất kỳ ngôn ngữ lập trình nào được gọi là kiểm thử tự động hóa. Với sự trợ giúp của kiểm thử tự động, chúng tôi có thể nâng cao tốc độ thực hiện kiểm thử của mình bởi vì ở đây, chúng tôi không yêu cầu bất kỳ nỗ lực nào của con người. Chúng ta cần viết một đoạn script thử nghiệm và thực thi những đoạn script đó.

Phân loại lỗi

- Yêu cầu, tính năng, lỗi chức năng
- Lỗi cấu trúc
- Lỗi dữ liệu
- Lỗi mã hóa
- Lỗi giao diện, tích hợp và hệ thống
- Kiểm tra và thử nghiệm các lỗi thiết kế

Kiểm tra tĩnh so với động

• Kiểm tra tĩnh

- Kiểm tra Yêu cầu và Mã
- Không thực thi mã

• ưu điểm

- Phát hiện lỗi sớm
- Giảm thử nghiệm sau này
- Toàn bộ đội tham gia

• nhược điểm

- Có vấn đề với các tương tác phức tạp của các thành phần hệ thống
- Một số lỗi chỉ có thể được phát hiện trong quá trình thực thi

• Thử nghiệm động

- Phát hiện lỗi thông qua thực thi chương trình
- Tạo các trường hợp thử nghiệm cụ thể

• ưu điểm

- Thực hiện đối tượng thử nghiệm
- Sự tương tác của các thành phần hệ thống được kiểm tra
- Các bài kiểm tra bổ sung, chẳng hạn như hiệu suất, tải…

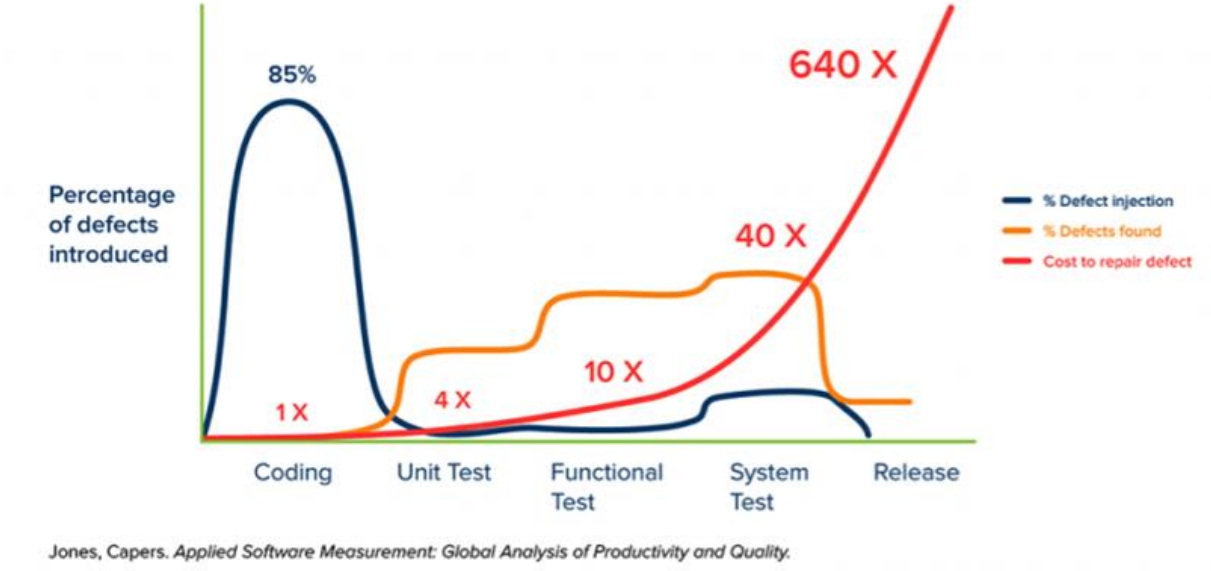
• Nhược điểm

- Cần đối tượng và môi trường thử nghiệm thực thi
- Chỉ các chương trình sẽ được thực thi mới được kiểm tra
- Chỉ các hiệu ứng lỗi được phát hiện, nguyên nhân lỗi cần được tìm thấy trong một bước riêng biệt (gỡ lỗi)

Tại sao nên kiểm thử phần mềm

Chi phí cao của lỗi phần mềm

Mặc dù việc tìm ra các lỗi phần mềm luôn có giá trị, nhưng việc phát hiện ra chúng sớm trong vòng đời phát triển là cách các tổ chức thu được nhiều giá trị nhất từ việc đầu tư vào phân tích tĩnh của họ. Biểu đồ sau đây cho thấy chi phí tìm kiếm các khiếm khuyết liên quan đến Vòng đời phát triển phần mềm (SDLC).



Hình 16: Minh họa chi phí cao của lỗi phần mềm

Trong nghiên cứu điển hình của họ, làm nổi bật giá trị của việc tìm ra lỗi trong giai đoạn mã hóa. Tận dụng phân tích tĩnh được kích hoạt để tìm lỗi lập trình trước khi phần mềm bắt đầu sản xuất, tiết kiệm chi phí liên quan đến việc kiểm tra lại, xác nhận lại và triển khai lại. Tuy nhiên, quan trọng nhất, việc phát hiện lỗi sớm giữ được vị thế tốt với khách hàng của họ. Cách tiếp cận chủ động của họ để bắt càng nhiều lỗi càng sớm càng tốt cho phép công ty nhanh chóng cung cấp phần mềm chất lượng cao mà khách hàng của họ mong đợi, đồng thời tránh chi phí liên quan đến việc phát hiện lỗi ở giai đoạn cuối.

4.3 Kế hoạch kiểm thử phần mềm

Kế hoạch kiểm thử phần mềm là gì?

Kế hoạch kiểm tra là một tài liệu đưa ra phạm vi, cách tiếp cận và lịch trình của các hoạt động kiểm tra dự kiến. Kế hoạch kiểm thử cũng có thể liệt kê các tài nguyên mà người kiểm thử phần mềm cần để hoạt động hiệu quả.

Kế hoạch kiểm tra thường bao gồm các thông tin sau:

i. Mục tiêu tổng thể của nỗ lực thử nghiệm.

ii. Một phác thảo chi tiết về cách thức kiểm tra sẽ được tiến hành (cách tiếp cận kiểm tra).

iii. Các tính năng, ứng dụng hoặc thành phần sẽ được kiểm tra.

iv. Lập lịch trình chi tiết và kế hoạch phân bổ tài nguyên cho người thử nghiệm và nhà phát triển trong suốt tất cả các giai đoạn thử nghiệm.

Các mục tiêu của một kế hoạch kiểm thử phần mềm là gì?

Mục tiêu chính của kế hoạch thử nghiệm là tạo ra tài liệu mô tả cách người thử nghiệm sẽ xác minh rằng hệ thống hoạt động như dự kiến. Tài liệu phải mô tả những gì cần phải được kiểm tra, cách nó sẽ được kiểm tra và ai chịu trách nhiệm thực hiện việc đó.

Bằng cách viết ra một kế hoạch thử nghiệm, tất cả các thành viên trong nhóm có thể làm việc đồng bộ và truyền đạt vai trò của họ cho nhau. Bạn nên cân nhắc việc tạo một số mục tiêu THÔNG MINH cho kế hoạch thử nghiệm của mình.

Hình 17: Mục tiêu của kế hoạch kiểm thử phần mềm

Trường hợp kiểm thử là gì?

Trường hợp kiểm thử là tài liệu do người kiểm thử phần mềm tạo ra chứa thông tin chi tiết về những gì kiểm thử sẽ đạt được. Đó là một phần thiết yếu của việc ghi lại thông tin về các hoạt động và kết quả thử nghiệm.

Các trường hợp kiểm thử được sử dụng cùng với các kế hoạch kiểm thử. Một trường hợp thử nghiệm nên bao gồm các thông tin sau.

i. Tên hoặc số duy nhất để xác định nó.

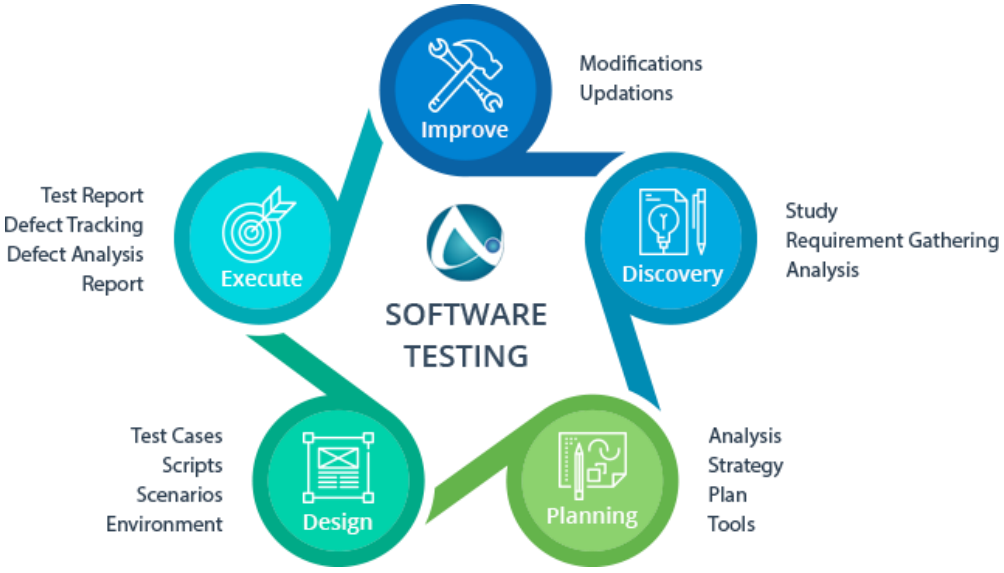
ii. Các tính năng, ứng dụng hoặc thành phần có trong trường hợp thử nghiệm.
iii. Các giá trị dữ liệu cụ thể cần thiết cho các trường đầu vào và các nút điều khiển sẽ được kiểm tra.
iv. Kết quả dự đoán của các hành động được thực hiện trong quá trình thử nghiệm (kết quả mong đợi).
v. Mô tả các kết quả thực tế sau mỗi hành động được thực hiện trong quá trình thử nghiệm (kết quả thực tế).
vi. Một dấu hiệu cho biết trường hợp thử nghiệm có thành công hay không.
vii. Mọi sai sót được phát hiện.

Tầm quan trọng của kế hoạch kiểm tra là gì?

Kế hoạch kiểm tra là nền tảng của mọi nỗ lực kiểm tra. Nó giúp đặt ra cách phần mềm sẽ được kiểm tra, những gì cụ thể sẽ được kiểm tra và ai sẽ thực hiện kiểm tra. Bằng cách tạo ra một kế hoạch kiểm tra rõ ràng mà tất cả các thành viên trong nhóm có thể làm theo, mọi người có thể làm việc cùng nhau một cách hiệu quả.

Cho dù bạn đang xây dựng một ứng dụng hay phát triển phần mềm nguồn mở, thì một kế hoạch thử nghiệm là điều cần thiết để mang lại kết quả cuối cùng.

Một kế hoạch chất lượng cao giúp xác định các khu vực rủi ro, xác định thứ tự của các hoạt động kiểm tra và phân bổ nguồn lực một cách hiệu quả. Kế hoạch thử nghiệm trở thành một tài liệu tham khảo hữu ích có thể được tham khảo lại trong suốt chu kỳ phát triển của sản phẩm.



Hình 18: Kế hoạch kiểm thử phần mềm

Cách ghi nhớ đối tượng khi tạo kế hoạch thử nghiệm

Trước khi bắt đầu tạo kế hoạch thử nghiệm, bạn cần xác định khách hàng mục tiêu của mình và đảm bảo nhu cầu của họ đang được đáp ứng. Điều này sẽ cải thiện chất lượng kế hoạch kiểm tra của bạn gấp mười lần.

Dưới đây là những điều chính để đảm bảo kế hoạch thử nghiệm của bạn là:

- Ngắn gọn. Kế hoạch kiểm tra của bạn không được dài hơn một trang với các gạch đầu dòng.

• Được tổ chức. Đảm bảo rằng tất cả thông tin được nhóm một cách hợp lý.
• Có thể đọc được. Tài liệu phải dễ đọc, tránh sử dụng ngôn ngữ kỹ thuật nếu có thể.
• Linh hoạt. Kế hoạch thử nghiệm của bạn phải phù hợp và không cố định trong đá. Bạn muốn tạo tài liệu sẽ không kìm hãm bạn nếu có thông tin mới hoặc cần thực hiện các thay đổi.
• Chính xác. Đảm bảo rằng tất cả thông tin bạn đã bao gồm là chính xác.

Cách viết kế hoạch kiểm tra

Đây có thể là công việc đầu tiên trong CV nhà phát triển phần mềm của bạn và nếu đúng như vậy, bạn có thể cần một bảng gian lận để viết thành công kế hoạch thử nghiệm ban đầu của mình.

May mắn thay, chúng tôi đã có bạn bảo hiểm. Phần này sẽ cung cấp cho bạn 14 điều cần thiết để đưa vào kế hoạch kiểm thử phần mềm của bạn như một phần của quy trình QA.

1) Tìm hiểu về phần mềm

Trước khi bắt đầu thử nghiệm, điều quan trọng là phải tìm hiểu mọi thứ bạn có thể về phần mềm. Đặt câu hỏi về cách nó được phát triển để tìm hiểu về mục đích dự định, cách thức hoạt động và thu thập thông tin có thể giúp bạn hiểu chức năng của nó.

Bằng cách hiểu đúng phần mềm của mình, bạn có thể tạo các trường hợp thử nghiệm có liên quan và hữu ích để thử nghiệm sản phẩm của mình.

2) Xác định phạm vi thử nghiệm

Không ích gì khi tạo tài liệu thử nghiệm dài hơn bản thân sản phẩm. Trước bất kỳ điều gì khác, hãy thiết lập chính xác những gì sẽ được kiểm tra trong quá trình này, mô-đun hoặc chức năng nào cần được đề cập chi tiết và bất kỳ khía cạnh thiết yếu nào khác mà bạn nên biết.

3) Tạo các trường hợp thử nghiệm

Một trong những nhiệm vụ chính khi phát triển một tài liệu kiểm thử phần mềm là tạo các trường hợp kiểm thử. Trường hợp thử nghiệm là một tài liệu mô tả các bước thực hiện để thực hiện thử nghiệm của bạn. Nó phải bao gồm các thông tin như:

• Những gì cần được kiểm tra
• Nó sẽ được kiểm tra như thế nào
• Ai sẽ làm bài kiểm tra
• Kết quả mong đợi

Dưới đây là một bảng tính đơn giản để thiết lập các trường hợp thử nghiệm:

Test Case Type	Description	Test Step	Expected Result	Status
Functionality	Area should accommodate up to 20 characters	Input up to 20 characters	All 20 characters in the request should be appropriate	Pass or Fail
Security	Verify password rules are working	Create a new password in accordance with rules	The user's password will be accepted if it adheres to the rules	Pass or Fail
Usability	Ensure all links are working properly	Have users click on various links on the page	Links will take users to another web page according to the on-page URL	Pass or Fail

Hình 19: Trường hợp kiểm thử cho phần mềm

4) Phát triển một chiến lược kiểm tra

Chiến lược thử nghiệm xác định cách bạn dự định triển khai thử nghiệm. Tất cả những người thử nghiệm của bạn phải thực hiện cùng một kế hoạch trò chơi, vì vậy hãy đảm bảo rằng mọi thành viên trong nhóm đều nhận thức được những gì họ phải làm vào bất kỳ thời điểm nào.

5) Xác định mục tiêu kiểm tra

Mỗi trường hợp kiểm tra phải được liên kết với một mục tiêu kiểm tra. Mục tiêu đảm bảo mọi hành động đều phù hợp và góp phần làm cho phần mềm của bạn có giá trị hơn đối với khách hàng. Các mục tiêu kiểm tra có thể bao gồm những thứ như:

• Kiểm tra các tính năng đã biết
• Thử nghiệm các tính năng mới được triển khai
• Thực hiện các bài kiểm tra thăm dò
• Đảm bảo sự ổn định trong suốt vòng đời của sản phẩm

6) Chọn công cụ kiểm tra

Bạn sẽ cần đảm bảo rằng mình có giải pháp kiểm thử phần mềm phù hợp để thực hiện các hoạt động kiểm tra của mình. Một số công cụ này có thể dựa trên phần mềm, trong khi những công cụ khác có thể yêu cầu tài nguyên vật lý như máy kiểm tra. Điều quan trọng là chọn các công cụ thích hợp cho từng công việc cụ thể và không dựa vào giải pháp một kích cỡ phù hợp với tất cả.

7) Tìm lỗi sớm

Dành thời gian trong tài liệu lập kế hoạch của bạn cho các phiên 'sửa lỗi'. Những điều này cho phép bạn xác định sớm các vấn đề với phần mềm trước khi chúng trở nên quá khó hoặc tốn kém để khắc phục. Điều này làm cho chúng dễ dàng hơn và rẻ hơn để giải quyết. Kiểm tra mọi biện pháp bảo mật của ứng dụng, sử dụng mọi tính năng và tìm kiếm những gì không hoạt động tốt.

8) Xác định tiêu chí kiểm tra của bạn

Đây nên là một phần của trường hợp thử nghiệm, nhưng tốt nhất là bạn nên chia nhỏ nó ra một cách riêng biệt. Tiêu chí kiểm tra về cơ bản là các mục tiêu của bạn được chia thành các phần nhỏ hơn. Chúng bao gồm thông tin cụ thể về cách đáp ứng từng mục tiêu, giúp bạn theo dõi tiến trình kiểm tra của mình.

Tiêu chí đình chỉ là tiêu chí cần được đáp ứng trước khi thử nghiệm có thể dừng lại. Ví dụ: bạn có thể muốn tạm ngừng thử nghiệm nếu một số lỗi nhất định đã được tìm thấy hoặc nếu phần mềm không thể chạy do các vấn đề về hiệu suất.

Tiêu chí thoát là tiêu chí cần được đáp ứng trước khi thử nghiệm có thể kết thúc. Ví dụ: trường hợp kiểm thử sẽ kết thúc khi mỗi mục tiêu đã được đáp ứng và tất cả các lỗi đã được giải quyết.

9) Lập kế hoạch nguồn lực

Trong tài liệu kiểm thử phần mềm của bạn, hãy bao gồm một kế hoạch tài nguyên liệt kê số lượng người cần thiết cho quá trình kiểm thử. Điều này sẽ nêu chi tiết vai trò của mỗi người là gì và bất kỳ khóa đào tạo nào họ sẽ yêu cầu để hoàn thành vai trò đó một cách hiệu quả.

Hình 20: Lập kế hoạch nguồn lực trong quản lý dự án

10) Lập kế hoạch cho môi trường thử nghiệm của bạn

Trong kế hoạch thử nghiệm của bạn, hãy bao gồm thông tin về môi trường nơi thử nghiệm sẽ diễn ra, chẳng hạn như:

• Kiểm tra phần cứng cần thiết để kiểm tra sản phẩm.
• Yêu cầu về kích thước cho phần mềm và máy chủ.
• Nền tảng được hỗ trợ bởi sản phẩm.
• Thông tin thiết yếu khác liên quan đến môi trường có thể ảnh hưởng đến quá trình thử nghiệm của bạn.

11) Lập kế hoạch hậu cần cho nhóm kiểm tra

Quản lý kiểm tra là một trong những phần quan trọng nhất của quá trình thực hiện. Nếu bạn không thể giao tiếp với những người kiểm tra của mình một cách hiệu quả, tiến trình của họ sẽ bị ảnh hưởng và tài liệu kiểm tra của bạn sẽ không hữu ích như nó có thể.

12) Lịch trình & ước tính

Trong kế hoạch thử nghiệm của bạn, hãy bao gồm một lịch trình cho phép bạn vạch ra các mốc và thời hạn thử nghiệm cụ thể. Các mốc quan trọng có thể bao gồm việc phát hành sản phẩm lần đầu, các phiên thử nghiệm nội bộ, thử nghiệm beta công khai hoặc bất kỳ điểm quan trọng nào khác trong thời gian mà nhóm của bạn cần tập trung nỗ lực vào thử nghiệm.

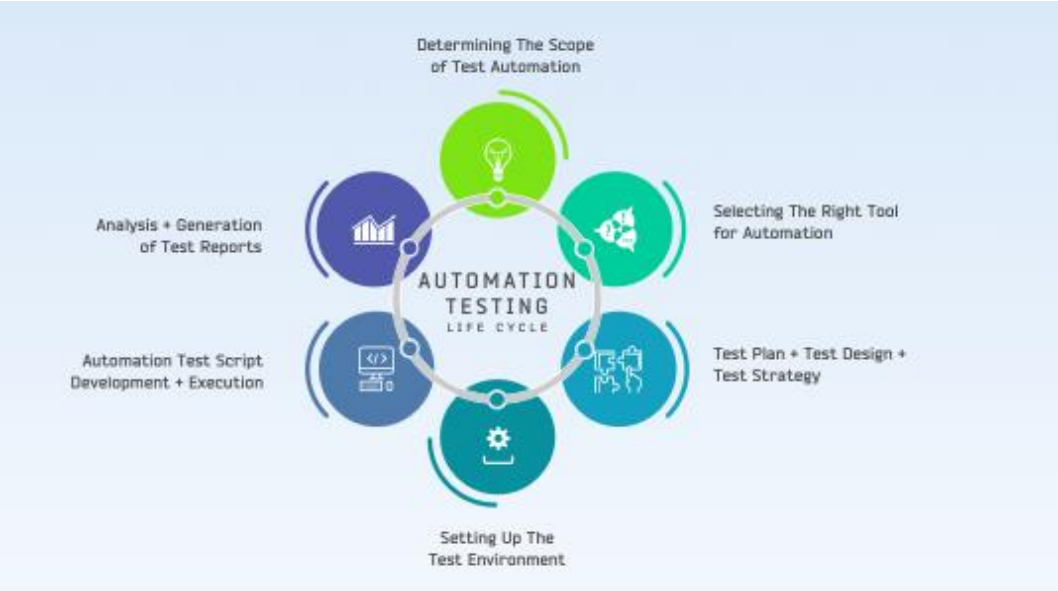
13) Thử nghiệm phân phối

Tài liệu thử nghiệm của bạn nên bao gồm danh sách tất cả các sản phẩm cần thiết để thử nghiệm. Các bước này phải được liên kết với các bước trong lịch trình của bạn để mọi người biết chính xác khi nào họ cần sẵn sàng hành động.

14) Tự động hóa kiểm tra

Nếu phần mềm của bạn đặc biệt phức tạp và đòi hỏi một số lượng lớn các trường hợp kiểm thử, bạn có thể muốn xem xét tự động hóa kiểm tra phần mềm.

Tự động hóa quy trình có nghĩa là người kiểm tra có thể hoàn thành nhiều việc hơn trong thời gian ngắn hơn, điều này giúp tăng năng suất và giảm đáng kể chi phí kiểm tra tổng thể. Bạn thậm chí có thể sử dụng một bot di động để tăng tốc các hoạt động thử nghiệm.



Hình 21: Vòng đời của thử nghiệm tự động hóa

Bài tập: So sánh Kiểm tra hộp đen so với Kiểm tra hộp trắng và Kiểm tra hộp xám

Chỉ số	Kiểm tra hộp đen	Kiểm tra hộp trắng	Kiểm tra hộp xám

1	Không cần phải có kiến thức về cấu trúc làm việc bên trong (Mã) đối với loại thử nghiệm này. Chỉ cần có GUI (Giao diện người dùng đồ họa) cho các trường hợp thử nghiệm.	Cần phải có kiến thức về cấu trúc làm việc bên trong (Mã hóa phần mềm) cho loại thử nghiệm này.	Cần phải có kiến thức một phần về cấu trúc làm việc bên trong.
2	Thử nghiệm Hộp đen còn được gọi là thử nghiệm chức năng, thử nghiệm theo hướng dữ liệu và thử nghiệm hộp kín.	Thử nghiệm Hộp trắng còn được gọi là thử nghiệm cấu trúc, thử nghiệm hộp rõ ràng, thử nghiệm dựa trên mã và thử nghiệm minh bạch.	Thử nghiệm Hộp xám còn được gọi là thử nghiệm trong mờ vì người thử nghiệm có kiến thức hạn chế về mã hóa.
3	Cách tiếp cận hướng tới kiểm thử bao gồm các kỹ thuật dùng thử và phương pháp đoán lỗi vì người kiểm thử không cần kiến thức về mã hóa nội bộ của phần mềm.	Kiểm thử Hộp trắng được tiến hành bằng cách xác minh ranh giới hệ thống và miền dữ liệu vốn có trong phần mềm vì không thiếu kiến thức mã hóa nội bộ.	Nếu người kiểm tra có kiến thức về mã hóa, thì quá trình này sẽ được tiến hành bằng cách xác nhận các miền dữ liệu và ranh giới hệ thống nội bộ của phần mềm.
4	Không gian thử nghiệm của các bảng cho các đầu vào (đầu vào được sử dụng để tạo các trường hợp thử nghiệm) là khá lớn và lớn nhất trong số tất cả các không gian thử nghiệm.	Không gian thử nghiệm của các bảng cho các đầu vào (đầu vào được sử dụng để tạo các trường hợp thử nghiệm) ít hơn so với thử nghiệm Hộp đen.	Không gian thử nghiệm của các bảng cho các đầu vào (đầu vào được sử dụng để tạo các trường hợp thử nghiệm) nhỏ hơn so với thử nghiệm Hộp đen và Hộp trắng.
5	Rất khó để phát hiện ra các lỗi ẩn của phần mềm vì lỗi có thể do hoạt động bên trong chưa biết để kiểm tra Black Box.	Rất đơn giản để phát hiện ra các lỗi ẩn vì nó có thể là do hoạt động bên trong được khám phá sâu trong thử nghiệm Hộp trắng.	Khó phát hiện ra lỗi ẩn. Có thể được tìm thấy trong thử nghiệm cấp độ người dùng.

6	Nó không được xem xét để kiểm tra thuật toán.	Nó rất phù hợp và được khuyến nghị để kiểm tra thuật toán.	Nó không được xem xét để kiểm tra thuật toán.
7	Thời gian tiêu thụ trong thử nghiệm Hộp đen phụ thuộc vào tính khả dụng của các thông số kỹ thuật chức năng.	Kiểm thử Hộp trắng mất nhiều thời gian để thiết kế các trường hợp kiểm thử do mã dài.	Việc thiết kế các trường hợp thử nghiệm có thể được thực hiện trong một khoảng thời gian ngắn.
8	Người kiểm tra, nhà phát triển và người dùng cuối có thể là một phần của thử nghiệm.	Chỉ người thử nghiệm và nhà phát triển mới có thể là một phần của thử nghiệm; người dùng cuối không thể tham gia.	Người kiểm tra, nhà phát triển và người dùng cuối có thể là một phần của thử nghiệm.

Bảng 3: So sánh Kiểm tra hộp đen, hộp trắng và hộp xám

BÀI 2: LẬP TRÌNH VI ĐIỀU KHIỂN

Mục tiêu:

Sau khi học xong bài học này các học viên có khả năng:

- Biết sự khác biệt giữa việc sử dụng vi điều khiển và PLC
- Có thể điều khiển bộ truyền động và cảm biến và kết nối chúng với bộ vi điều khiển
- Có thể lưu các giá trị đo được trong các tệp nhật ký
- Có thể tìm kiếm và tìm lỗi theo cách có cấu trúc
- Có thể vận hành, cấu hình và tham số hóa một bộ vi xử lý
- Tạo máy trạng thái để thực hiện các chương trình đo
- Có thể tích hợp các mô-đun phần mềm vào một chương trình tuần tự
- Quen thuộc với các khả năng trao đổi dữ liệu giữa các bộ vi xử lý
- Biết các khả năng kết nối bộ vi xử lý với các hệ thống CNTT cấp cao hơn

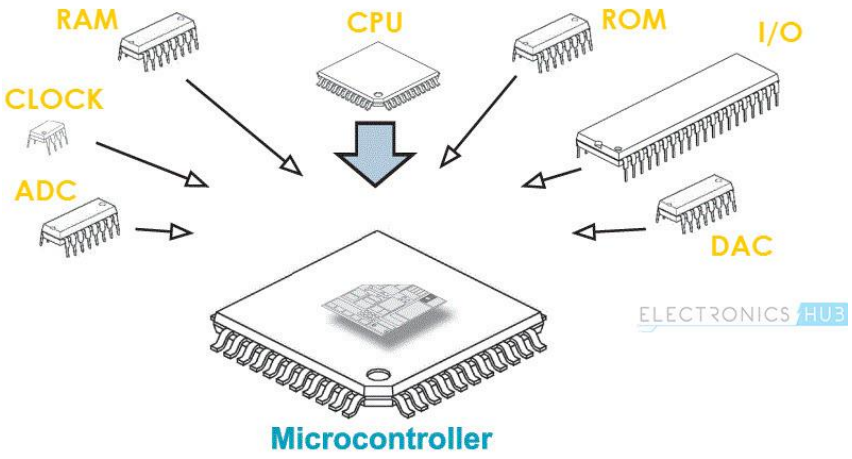
Nội dung chính:

1. Kiến thức cơ bản về lập trình vi điều khiển

1.1. Kiến thức cơ bản về vi điều khiển

a. Vi điều khiển là gì?

Vi điều khiển là một mạch tích hợp (IC) VLSI (Tích hợp quy mô rất lớn) có chứa đơn vị tính toán điện tử và đơn vị logic (được gọi chung là CPU), Bộ nhớ (Bộ nhớ chương trình và Bộ nhớ dữ liệu), Cổng I / O (Cổng vào / ra) và một số thành phần khác được tích hợp trên một con chip duy nhất.



Hình 22: Minh họa của Vi điều khiển

Đôi khi, Vi điều khiển còn được gọi là Máy tính trên chip hoặc Máy tính đơn chip. Vì Bộ vi điều khiển và mạch hỗ trợ của nó thường được nhúng vào thiết bị mà nó điều khiển, nên Bộ vi điều khiển còn được gọi là Bộ điều khiển nhúng.

Bộ vi điều khiển có mặt khắp nơi. Nếu một thiết bị hoặc một ứng dụng liên quan đến việc đo lường, lưu trữ, tính toán, kiểm soát hoặc hiển thị thông tin thì thiết bị đó có chứa một Bộ vi điều khiển trong đó. Hãy để chúng tôi xem một số lĩnh vực mà vi điều khiển được sử dụng.

Người sử dụng Vi điều khiển nhiều nhất có lẽ là Ngành công nghiệp ô tô. Hầu hết mọi ô tô ra khỏi nhà máy lắp ráp đều chứa ít nhất một Vi điều khiển nhằm mục đích điều khiển động cơ. Bạn có thể tìm thấy nhiều Bộ vi điều khiển khác để điều khiển các hệ thống bổ sung.

Điện tử gia dụng là một lĩnh vực khác được tải với Bộ vi điều khiển. Bộ vi điều khiển là một phần của Máy ảnh kỹ thuật số, Máy quay video, Đầu đĩa CD và DVD, Máy giặt, Lò nướng, v.v.

Bộ vi điều khiển cũng được sử dụng trong các thiết bị đo lường và kiểm tra như Đồng hồ vạn năng, Máy hiện sóng, Máy phát chức năng, v.v. Bạn cũng có thể tìm thấy các bộ vi điều khiển gần máy tính để bàn của mình như Máy in, Bộ định tuyến, Modem, Bàn phím, v.v.

b. Sự trỗi dậy của bộ vi điều khiển

Bộ vi xử lý, phát minh đã gây bão lĩnh vực tính toán. Bộ vi xử lý là một mạch tích hợp (IC) có chứa Bộ xử lý trung tâm (CPU). Bộ vi xử lý được biết đến sớm nhất là Intel's 4004 và Texas Instruments 'TMS1000.

Kể từ đó, sức mạnh tính toán, độ phức tạp và mức tiêu thụ điện năng tiếp tục tăng để mang lại hiệu suất tối ưu (Mức tiêu thụ điện năng phải được thảo luận riêng do những phát triển như VLSI công suất thấp, v.v.).

Để một Bộ vi xử lý hoạt động, nó cần một loạt phần cứng hỗ trợ có thể tìm thấy trên bo mạch chủ. Phần cứng bao gồm bộ nhớ, IC cho các thiết bị ngoại vi, v.v.

Ngay từ đầu, khả năng của Bộ vi xử lý có thể điều khiển các thiết bị điện tử khác như Máy photocopy. Sự nhấn mạnh ở đây không phải là sức mạnh tính toán của Bộ vi xử lý mà là cơ chế điều khiển với phần cứng ít phức tạp hơn và tăng độ tin cậy.

Yêu cầu này mở đường cho việc tích hợp phần cứng tối thiểu cần thiết để hoạt động hoàn chỉnh của Bộ xử lý vào một chip, tức là cùng một chip với bộ xử lý, chính xác.

Đây là sự ra đời của Vi điều khiển, một vi mạch tích hợp, chứa tất cả các chức năng và phần cứng để tạo nên một hệ thống máy tính hoàn chỉnh. Ở đây, sức mạnh tính toán của thiết bị ít quan trọng hơn sự tích hợp của tất cả các thành phần, bao gồm cả bộ nhớ.

c. Khái niệm cơ bản về vi điều khiển

Về cơ bản, một Vi điều khiển bao gồm các thành phần sau.

- Bộ phận xử lý trung tâm (CPU)

- Bộ nhớ chương trình (ROM - Bộ nhớ chỉ đọc)
- Bộ nhớ dữ liệu (RAM - Bộ nhớ truy cập ngẫu nhiên)
- Bộ hẹn giờ và bộ đếm
- Cổng I / O (I / O - Đầu vào / Đầu ra)
- Giao diện truyền thông nối tiếp
- Mạch đồng hồ (Mạch tạo dao động)
- Cơ chế ngắt

Hầu hết các Bộ vi điều khiển hiện đại có thể chứa nhiều thiết bị ngoại vi hơn như SPI (Giao diện ngoại vi nối tiếp), I2C (Mạch tích hợp liên thông), ADC (Bộ chuyển đổi tương tự sang kỹ thuật số), DAC (Bộ chuyển đổi kỹ thuật số sang tương tự), CAN (Mạng vùng điều khiển), USB (Nối tiếp đa năng Xe buýt), và nhiều hơn nữa.

CPU (Bộ xử lý trung tâm) trong Vi điều khiển thực hiện chức năng số học, logic, toán học và hướng dữ liệu, tương tự như CPU trong Bộ vi xử lý. Sự khác biệt giữa Bộ vi xử lý và Vi điều khiển là Bộ vi xử lý cần có giao diện với bộ nhớ ngoài và các Giao diện I / O khác để hoạt động như một máy tính trong khi Vi điều khiển có tất cả các thiết bị ngoại vi cần thiết trên cùng một chip như CPU.

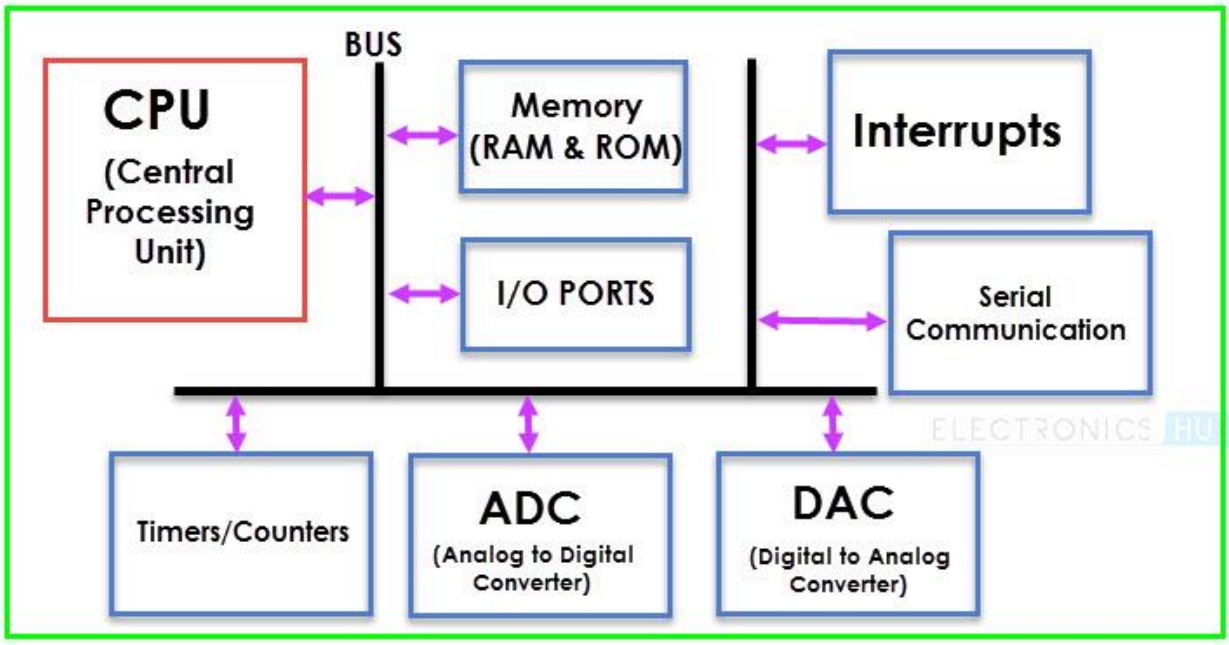
Việc tích hợp các tính năng như ADC, DAC, v.v. trên cùng một chip với CPU làm cho nó hoạt động hiệu quả hơn và rẻ hơn so với việc sử dụng một chip ADC riêng biệt.

Việc phát triển Hệ thống điều khiển bằng máy tính liên quan đến việc thiết kế Phần cứng và cũng như viết một Chương trình Phần mềm hiệu quả. Vì Vi điều khiển có tất cả phần cứng, được yêu cầu để tạo ra một hệ thống được điều khiển bằng máy tính trên một chip duy nhất, việc sử dụng Vi điều khiển sẽ giảm đáng kể nỗ lực và thời gian dành cho việc thiết kế phần cứng và đi dây.

d. Cấu trúc cơ bản của vi điều khiển

Bạn có thể đã thấy cấu trúc cơ bản của Vi điều khiển nhiều lần. Nếu bạn đã xem cấu trúc của Vi điều khiển và các thành phần cơ bản của Vi điều khiển trước đó, thì hãy coi đây là một bản sửa đổi. Nếu bạn chưa xem, thì điều rất quan trọng là bạn phải tìm hiểu về cấu trúc cơ bản của Vi điều khiển.

Hình ảnh sau đây cho thấy cấu trúc cơ bản của vi điều khiển.



Hình 23: Cấu trúc cơ bản của vi điều khiển

Từ hình ảnh trên, bạn có thể hiểu rằng ba thành phần quan trọng (hoặc chính) của Vi điều khiển là:

- CPU (Bộ xử lý trung tâm)
- Bộ nhớ
- Cổng I/O

Điều này không có nghĩa là các thành phần khác ít quan trọng hơn. Nhưng có thể coi đây là những thiết bị hỗ trợ.

e. Ưu điểm của Vi điều khiển

- Vi điều khiển là một thiết bị thực sự phù hợp với ý tưởng máy tính trên chip.
- Không cần bất kỳ giao diện bên ngoài nào của các thành phần cơ bản như Bộ nhớ, Cổng I / O, v.v.
- Bộ vi điều khiển không yêu cầu hệ điều hành phức tạp vì tất cả các hướng dẫn phải được ghi và lưu trữ trong bộ nhớ. (RTOS là một ngoại lệ).
- Tất cả các Cổng vào / ra đều có thể lập trình được.
- Tích hợp tất cả các thành phần thiết yếu làm giảm chi phí, thời gian thiết kế và diện tích của sản phẩm (hoặc ứng dụng).

f. Nhược điểm của Vi điều khiển

- Bộ vi điều khiển không được biết đến với khả năng tính toán của chúng.
- Dung lượng bộ nhớ giới hạn các lệnh mà vi điều khiển có thể thực thi.

- Không có Hệ điều hành và do đó, tất cả các hướng dẫn phải được viết.

g. Các ứng dụng của vi điều khiển

Có rất nhiều ứng dụng của Vi điều khiển. Trên thực tế, toàn bộ ngành công nghiệp hệ thống nhúng phụ thuộc vào Vi điều khiển. Sau đây là một vài ứng dụng của Vi điều khiển.

- Bảng điều khiển phía trước trong các thiết bị như Lò nướng, máy giặt, v.v.
- Máy phát chức năng
- Báo động khói và cháy
- Hệ thống tự động hóa gia đình
- Đèn pha tự động BẬT trong Ô tô
- Hệ thống khóa cửa cảm biến tốc độ

1.2. Các thuật ngữ cơ bản của lập trình

a. Thuật ngữ lập trình là gì?

Thuật ngữ lập trình là tập hợp những từ ngữ biểu thị về khái niệm khoa học, công nghệ. Thông thường, chúng sẽ được sử dụng trong các lĩnh vực về lập trình và biểu thị chính xác về khái niệm chuyên môn.

Đa phần, thuật ngữ lập trình không chứa tính biểu cảm, đồng thời phải được cơ quan có thẩm quyền phê duyệt và ban hành.

b. Thuật ngữ chỉ chung

- Software Engineering: Kỹ thuật phần mềm.
- HDSE (Higher Diploma in Software Engineering): Chứng Chỉ kỹ sư phần mềm Quốc tế.
- Structured Programming: Lập trình cấu trúc .
- OOP (Object-Oriented Programming): Lập trình hướng đối tượng.
- Programmer: Lập trình viên.
- Programming: Lập trình.
- Program: Chương trình.
- Tester: Người kiểm thử chương trình.
- Designer: Chuyên viên thiết kế.
- Developer: Người phát triển phần mềm.
- Project: Dự án.
- Project Manager (PM): Quản lý dự án.
- Coder: Người viết Code.

c. Thuật ngữ về mã nguồn

- Nguồn code: Mã nguồn.
- Open Nguồn: Mã Nguồn mở.
- Code: Mã.
- Design: Thiết kế.
- Nguồn file: File nguồn.
- Library: Thư viện.
- Header file: File chứa các nguyên mẫu hàm.
- Implementation File: File chứa nội dung thực thi, mã lệnh của các hàm.

d. Thuật ngữ về công cụ, chương trình dịch

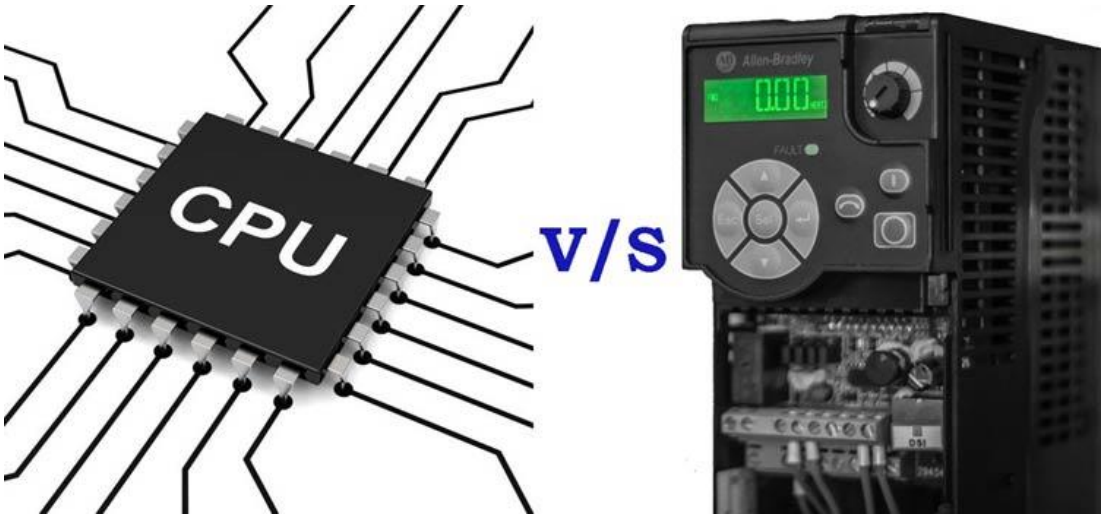
- Run: Chạy chương trình.
- Debug: Gỡ rối, sửa lỗi.
- Error: Lỗi.
- Compile error: Lỗi khi dịch chương trình.
- Runtime error: Lỗi khi chạy chương trình.
- Integrated-Development-Environment (IDE): Môi trường tích hợp phát triển.
- Compiler: Trình biên dịch.
- Compile: Dịch chương trình.
- Interpreter: Trình thông dịch.
- Line: dòng.
- Editor: Trình soạn thảo.

e. Thuật ngữ khi viết code

- Operator: Toán tử .
- Function: Hàm .
- Character: Ký tự .
- Digits: Chữ số .
- Argument: Đối số.
- Selection: Chọn lựa, rẽ nhánh.
- Statement: Câu lệnh .
- Declaration: Khai báo.
- Initialization: Khởi tạo.
- Definition: Định nghĩa.
- Condition: Điều kiện.
- Control structure: Cấu trúc điều khiển.
- Value: Giá trị.
- Syntax: Cú pháp.
- Function Call: Lời gọi hàm.
- Expression: Biểu thức.
- Operand: Toán hạng .
- Dynamic Variable: Biến động .
- Memory leak: Lỗi xảy ra khi con trỏ ra khỏi phạm vi khi chưa giải phóng bộ nhớ.
- Pointer: Con trỏ .

- Reference: Tham chiếu.
- Parameter: Tham số .
- Prototype: Nguyên mẫu hàm .
- Comment: Ghi chú, chú thích .
- Code block: Khối lệnh .
- Assign: Gán .
- Allocate (memory): Cấp phát bộ nhớ.
- Deallocate (memory): Giải phóng và thu hồi bộ nhớ.
- Dynamic Memory: Bộ nhớ động .
- Static Memory: Bộ nhớ tĩnh.
- Static variable: Biến tĩnh .

1.3. Sự khác biệt giữa vi điều khiển và SPS

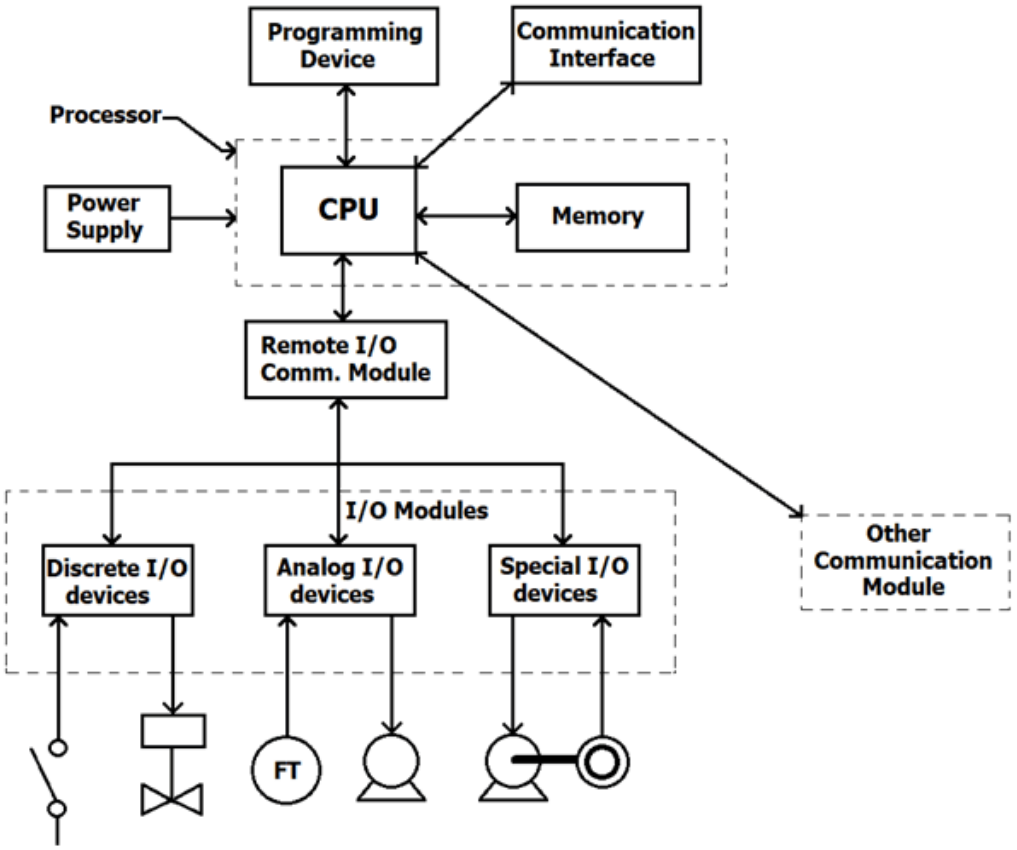


Hình 24: Bộ vi điều khiển và PLC

a. Kiến trúc

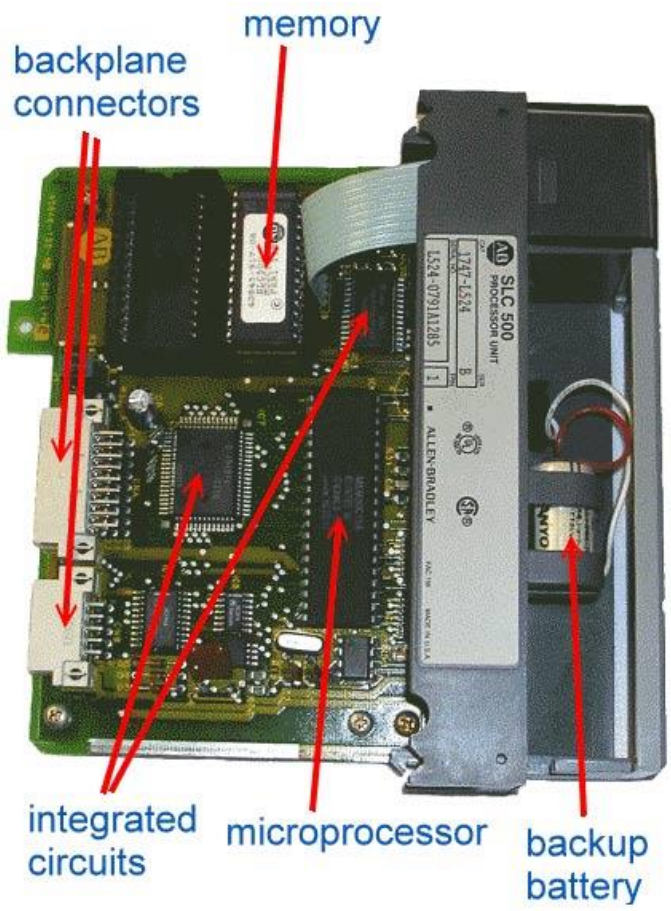
Kiến trúc PLC:

PLC nói chung có thể được coi là một bộ vi điều khiển cấp cao. Về cơ bản, chúng được tạo thành từ một mô-đun bộ xử lý, bộ nguồn và các mô-đun I / O. Mô-đun bộ xử lý bao gồm khối xử lý trung tâm (CPU) và bộ nhớ. Ngoài bộ vi xử lý, CPU cũng chứa ít nhất một giao diện mà qua đó nó có thể được lập trình (USB, Ethernet hoặc RS232) cùng với các mạng truyền thông. Nguồn cung cấp thường là một mô-đun riêng biệt và các mô-đun I / O tách biệt với bộ xử lý. Các loại mô-đun I / O bao gồm rời rạc (bật / tắt), Analog (biến liên tục) và các mô-đun đặc biệt như điều khiển chuyển động hoặc bộ đếm tốc độ cao. Các thiết bị hiện trường được kết nối với các mô-đun I / O.



Hình 25: Kiến trúc PLC

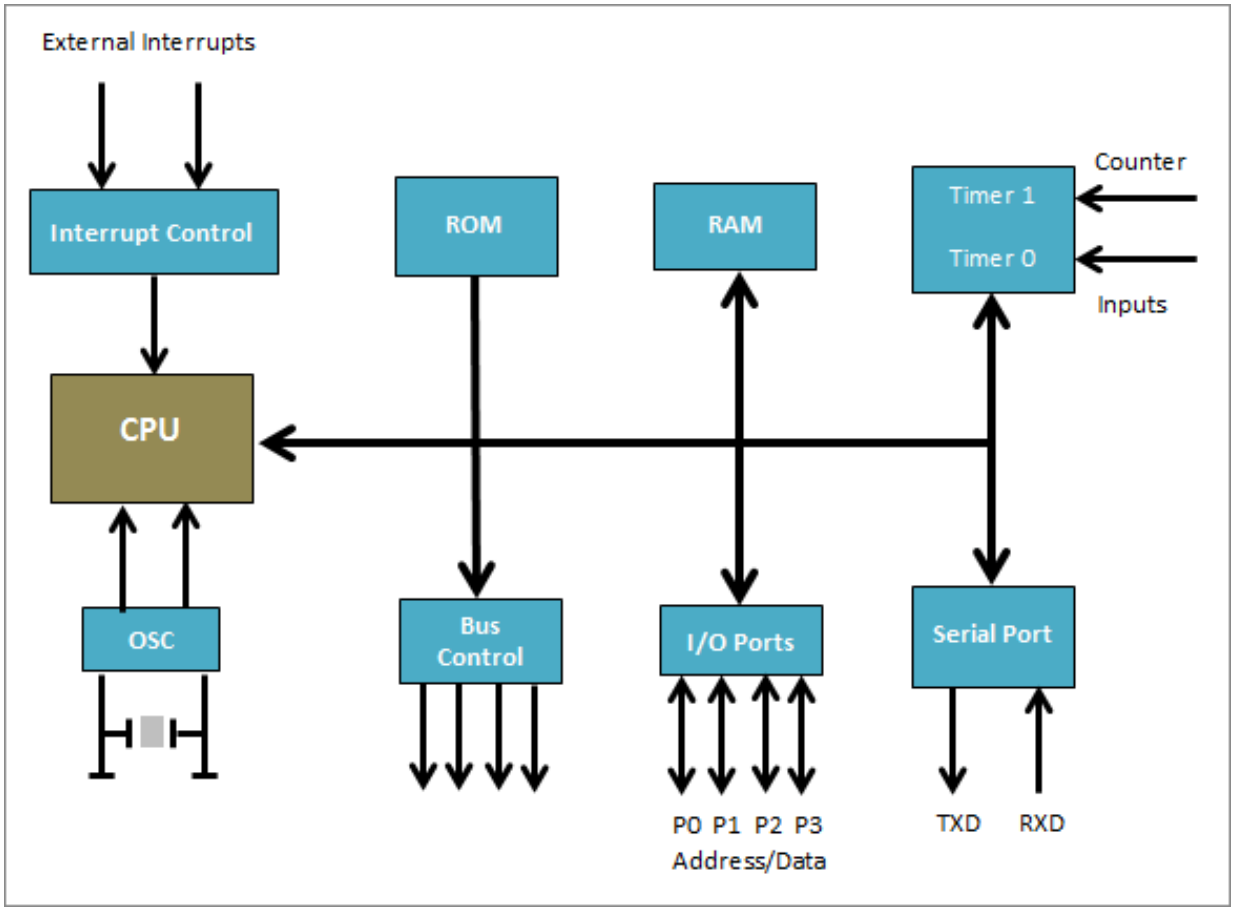
Tùy thuộc vào số lượng mô-đun I/O mà PLC sở hữu, chúng có thể nằm trong cùng một vỏ với PLC hoặc trong một vỏ riêng biệt. Một số PLC nhỏ được gọi là PLC nano/micro thường có tất cả các bộ phận của chúng bao gồm nguồn, bộ xử lý, v.v. trong cùng một vỏ.



Hình 26: Cấu trúc phần cứng PLC

Kiến trúc của vi điều khiển

Kiến trúc của PLC được mô tả ở trên hơi giống với vi điều khiển về mặt cấu thành, nhưng vi điều khiển thực hiện mọi thứ trên một chip duy nhất, từ CPU đến các cổng I / O và các giao diện cần thiết để giao tiếp với thế giới bên ngoài. Kiến trúc của vi điều khiển được hiển thị bên dưới.



Hình 27: Kiến trúc của vi điều khiển

Cũng giống như vi điều khiển có kiến trúc đa dạng từ kiến trúc AVR đến kiến trúc 8051, các PLC cũng có các biến thể trong thiết kế hỗ trợ cấu hình và mong muốn của một nhà sản xuất cụ thể nhưng nhìn chung tất cả đều tuân theo tiêu chuẩn công nghiệp (IEC 61131-3) cho PLC . Tiêu chuẩn này thúc đẩy khả năng tương tác giữa các mô-đun và các bộ phận.

b. Giao diện

PLC được thiết kế tiêu chuẩn để giao tiếp với các cảm biến, bộ truyền động và mô-đun truyền thông cấp công nghiệp và do đó được cung cấp xếp hạng dòng điện và điện áp thường không tương thích với bộ vi điều khiển mà không có phần cứng bổ sung.

PLC thường sử dụng Ethernet và một số biến thể của dòng RS-serial như RS-232, RS-485 để giao tiếp. Sự ra đời của internet vạn vật ngày nay đang tạo ra sự gia tăng về số lượng các thiết bị PLC được kết nối có khả năng truyền dữ liệu qua các giao diện truyền thông không dây.

Như đã đề cập trước đó, chúng có nhiều kích cỡ khác nhau, từ các thiết bị nhỏ (với ít chân IO / mô-đun) được coi là khối xây dựng đến các PLC lớn, gắn trên giá đỡ khổng lồ với hàng trăm IO.

Bộ vi điều khiển cũng có các cảm biến, bộ truyền động và mô-đun được thiết kế để đáp ứng các nhu cầu cụ thể của chúng mà có thể khó giao tiếp với PLC. Tuy nhiên, chúng thường được thiết kế để xử lý chỉ vài 100 IO. Trong khi một số kỹ thuật có thể được khám phá để tăng IO của vi điều khiển, điều này vẫn có thể thực hiện được với PLC và do đó không phải là duy nhất đối với vi điều khiển, vì nó làm tăng toàn bộ ngân sách dự án.

c. Hiệu suất, Độ bền và Độ tin cậy

Đây là điểm mà PLC phân biệt chính nó nhất. Như đã đề cập ban đầu, PLC được thiết kế để sử dụng trong các thiết lập công nghiệp và do đó đã được củng cố để có thể chịu được một số điều kiện bất lợi liên quan đến môi trường đó như, phạm vi nhiệt độ khắc nghiệt, tiếng ồn điện, xử lý thô và lượng rung cao. PLC cũng là một ví dụ điển hình về hệ thống hoạt động theo thời gian thực nhờ khả năng tạo ra kết quả đầu ra trong thời gian ngắn nhất có thể sau khi đánh giá đầu vào. Điều này rất quan trọng trong hệ thống công nghiệp vì thời gian là một phần quan trọng của quá trình / nhà máy sản xuất.

Tuy nhiên, vi điều khiển kém cứng cáp hơn. Theo thiết kế, chúng không được thiết kế để phục vụ như các thiết bị độc lập như PLC. Chúng được thiết kế để nhúng vào một hệ thống. Điều này cung cấp một lời giải thích cho vẻ ngoài kém chắc chắn hơn của chúng so với PLC. Vì những lý do này, bộ vi điều khiển có thể bị lỗi khi triển khai trong một số trường hợp nhất định vì các chip này rất mỏng manh và có thể dễ bị hỏng.

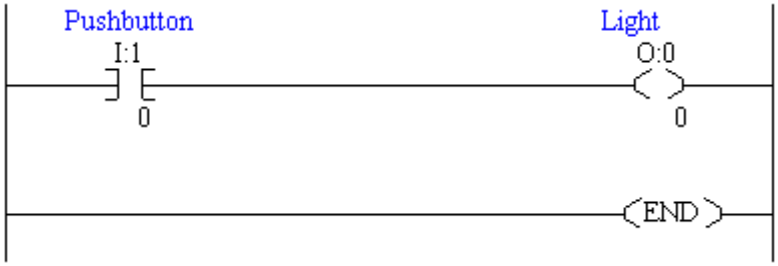
d. Yêu cầu kỹ năng để sử dụng

Một trong những thuộc tính quan trọng của PLC là kiến thức kỹ thuật cần thiết để lập trình và vận hành nó nói chung. PLC được thiết kế để sử dụng bởi cả các chuyên gia tự động hóa có tay nghề cao và các kỹ thuật viên nhà máy, những người ít hoặc không được đào tạo chính quy. Nó tương đối dễ dàng để khắc phục sự cố và chẩn đoán lỗi. Các thiết bị PLC hiện đại thường đi kèm với màn hình hiển thị giúp giám sát mọi thứ dễ dàng hơn mà không cần các công cụ phức tạp.

Tuy nhiên, vi điều khiển lại yêu cầu xử lý khéo léo. Người thiết kế cần có kiến thức tốt về nguyên lý kỹ thuật điện và lập trình để có thể thiết kế các mạch bổ trợ cho vi điều khiển. Bộ vi điều khiển cũng yêu cầu các công cụ đặc biệt (ví dụ: Máy hiện sóng) để chẩn đoán lỗi và xử lý sự cố phần sụn. Mặc dù một số nền tảng đơn giản hóa như Arduino hiện đang tồn tại, nó vẫn phức tạp hơn nhiều so với các PLC cắm và chạy cả từ quan điểm kết nối, quan điểm lập trình và tính dễ sử dụng.

e. Lập trình

Vì lợi ích của sự đơn giản và dễ sử dụng bởi tất cả các lớp kiến thức, PLC ban đầu được thiết kế để lập trình bằng cách sử dụng hình ảnh lập trình mô phỏng các kết nối / sơ đồ của sơ đồ logic rơle. Điều này làm giảm các yêu cầu đào tạo đối với các kỹ thuật viên hiện có. Ngôn ngữ lập trình chính, phổ biến nhất được sử dụng cho PLC là Ngôn ngữ lập trình Logic bậc thang và ngôn ngữ lập trình danh sách lệnh. Logic bậc thang sử dụng các ký hiệu, thay vì từ ngữ, để mô phỏng điều khiển logic rơle trong thế giới thực, vốn là một di tích từ lịch sử của PLC. Các ký hiệu này được kết nối với nhau bằng các đường để chỉ dòng điện chạy qua rơ le, giống như các tiếp điểm và cuộn dây. Số lượng các ký hiệu đã tăng lên rất nhiều trong những năm qua cho phép các kỹ sư dễ dàng triển khai các chức năng ở mức độ cao.



Hình 28: Biểu đồ / logic bậc thang

Ví dụ về mã dựa trên sơ đồ / logic bậc thang được hiển thị ở trên. Nó thường trông giống như một cái thang, đó là lý do đằng sau tên của nó. Cái nhìn đơn giản này làm cho PLC rất dễ lập trình, vì vậy nếu bạn có thể phân tích một giản đồ, bạn có thể lập trình PLC.

Do sự phổ biến gần đây của các ngôn ngữ lập trình cấp cao hiện đại, các PLC hiện đang được lập trình bằng các ngôn ngữ này như C, C ++ và cơ bản nhưng tất cả các PLC nói chung vẫn tuân thủ tiêu chuẩn hệ thống điều khiển IEC 61131/3 của ngành và hỗ trợ các ngôn ngữ lập trình được quy định bởi tiêu chuẩn bao gồm; Sơ đồ bậc thang, Văn bản có cấu trúc, Sơ đồ khối chức năng, Danh sách lệnh và Lưu đồ tuần tự.

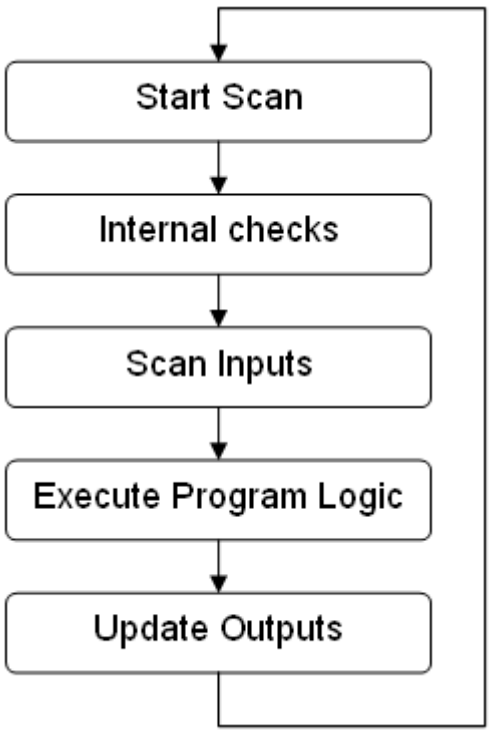
PLC ngày nay thường được lập trình thông qua phần mềm ứng dụng dựa trên bất kỳ ngôn ngữ nào được đề cập ở trên, chạy trên PC được kết nối với PLC bằng bất kỳ giao diện nào, USB, Ethernet, RS232, RS-485, RS-422,.

Mặt khác, các bộ vi điều khiển được lập trình bằng các ngôn ngữ cấp thấp như hợp ngữ hoặc các ngôn ngữ cấp cao như C và C ++ trong số những ngôn ngữ khác. Nó thường đòi hỏi kinh nghiệm cao với ngôn ngữ lập trình đang được sử dụng và hiểu biết chung về các nguyên tắc phát triển phần sụn. Các lập trình viên thường cần phải hiểu các khái niệm như cấu trúc dữ liệu và hiểu biết sâu sắc về kiến trúc vi điều khiển là cần thiết để phát triển một phần sụn rất tốt cho dự án.

Bộ vi điều khiển cũng thường được lập trình thông qua phần mềm ứng dụng chạy trên PC và chúng thường được kết nối với PC đó thông qua một phần cứng bổ sung thường được gọi là bộ lập trình.

Tuy nhiên, hoạt động của các chương trình trên PLC rất giống với hoạt động của vi điều khiển. PLC sử dụng một bộ điều khiển chuyên dụng do đó chúng chỉ xử lý lặp đi lặp lại một chương trình. Một chu kỳ thông qua chương trình được gọi là quét và nó tương tự như một bộ vi điều khiển đi qua một vòng lặp.

Chu trình hoạt động thông qua chương trình chạy trên PLC được hiển thị bên dưới.



Biểu đồ 10: Chu kỳ hoạt động trên PLC

f. Ứng dụng

PLC là phần tử điều khiển chính được sử dụng trong các hệ thống điều khiển công nghiệp. Họ tìm thấy ứng dụng trong điều khiển máy công nghiệp, băng tải, rô bốt và các máy móc dây chuyền sản xuất khác. Chúng cũng được sử dụng trong các hệ thống dựa trên SCADA và trong các hệ thống yêu cầu mức độ tin cậy cao và khả năng chịu đựng các điều kiện khắc nghiệt. Chúng được sử dụng trong các ngành công nghiệp bao gồm;

- Hệ thống chiết rót chai liên tục
- hệ thống trộn hàng loạt
- hệ thống điều hòa không khí tầng
- kiểm soát lưu lượng

Mặt khác, vi điều khiển được ứng dụng trong các thiết bị điện tử hàng ngày. Chúng là nền tảng chính của một số thiết bị điện tử tiêu dùng và thiết bị thông minh.

2. Lập trình vi điều khiển và điều khiển phần cứng

2.1. Khái niệm cơ bản về lập trình vi điều khiển

2.1.1. Cài đặt các gói phần mềm cần thiết

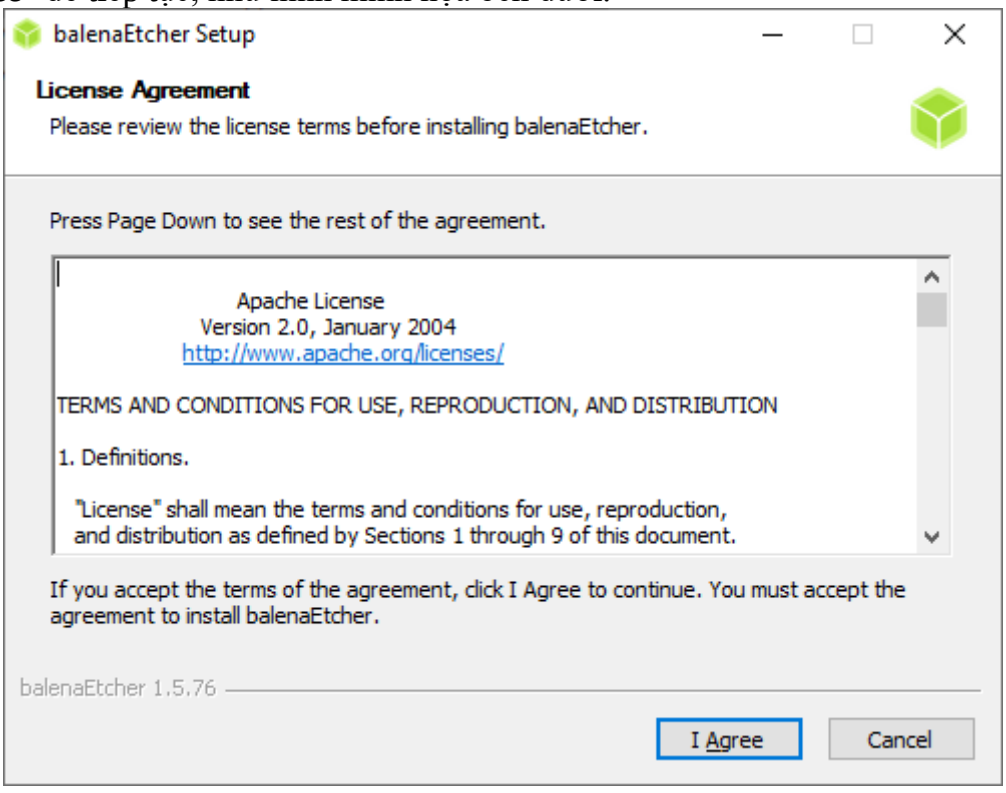
Đầu tiên chúng ta cần phải tải tập tin nén “raspbian-buster.zip” được dùng để cài đặt hệ điều hành Linux Raspbian theo đường liên kết sau đây

https://www.raspberrypi.org/downloads/raspbian/, về máy tính.

Tiếp theo chúng ta tải phần mềm balenaEtcher dành cho hệ điều hành Windows theo đường liên kết sau đây, **https://www.balena.io/etcher/**, về máy tính. Phần mềm này được sử dụng nhằm mục đích sao chép các tập tin OS Image vào thẻ nhớ. Ngoài ra, chúng ta cũng có thể sử dụng các phần mềm khác để sao chép tập tin OS Image vào thẻ nhớ giúp cài đặt hệ điều hành Linux cho máy tính nhúng Raspberry Pi.

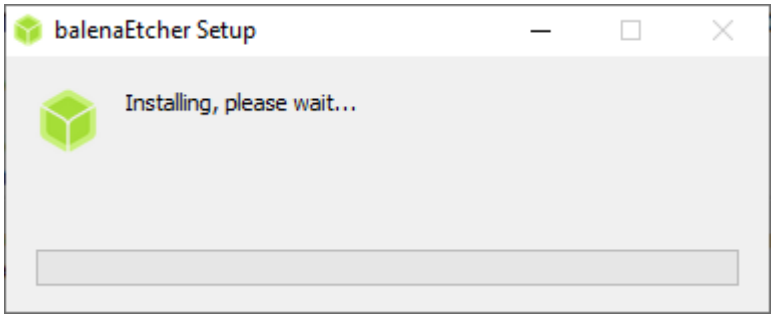
Hình 29: Hướng dẫn tải xuống phần mềm

Nhấp đúp chuột vào tập tin mới tải về được để bắt đầu tiến trình cài đặt phần mềm balenaEtcher. Trên màn hình sẽ xuất hiện một cửa sổ cài đặt và ta nhấp chuột vào nút "I Agree" để tiếp tục, như hình minh họa bên dưới.



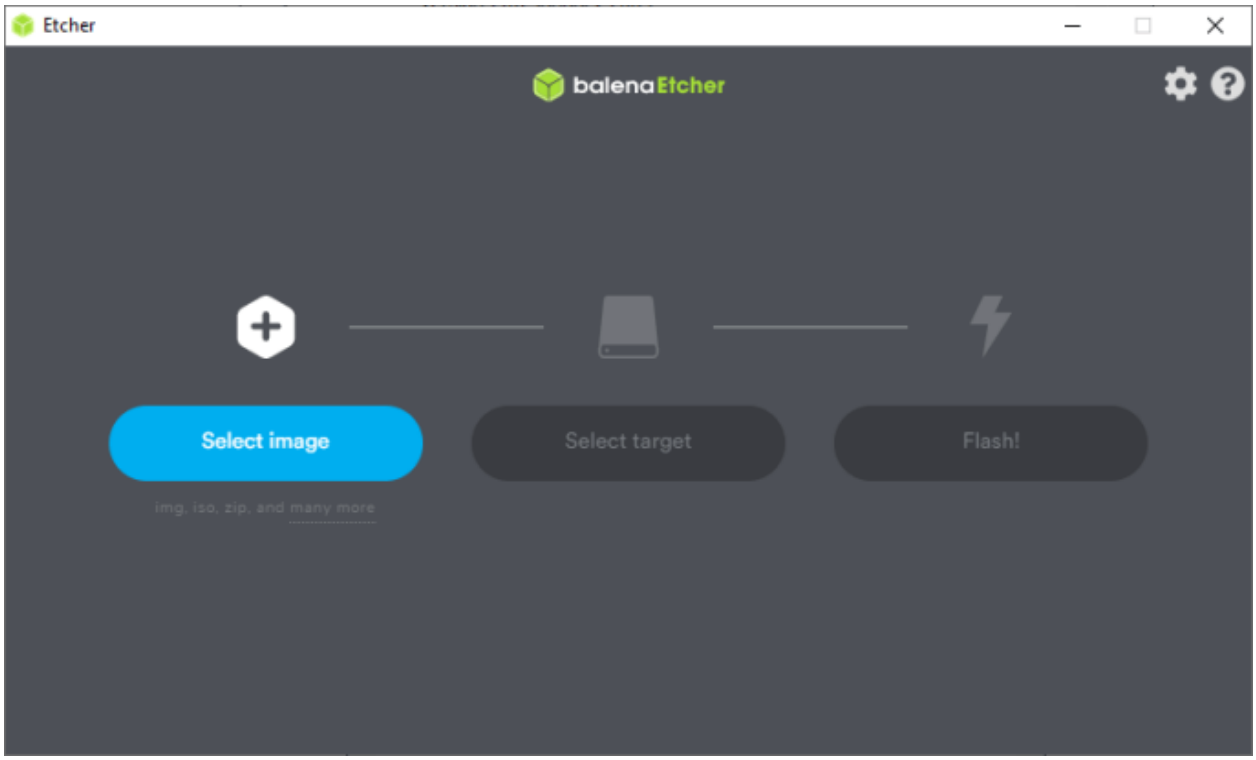
Hình 30: Hướng dẫn cài đặt balenaEtcher 1

Chờ một khoảng thời gian để quá trình cài đặt hoàn tất phần mềm balenaEtcher.



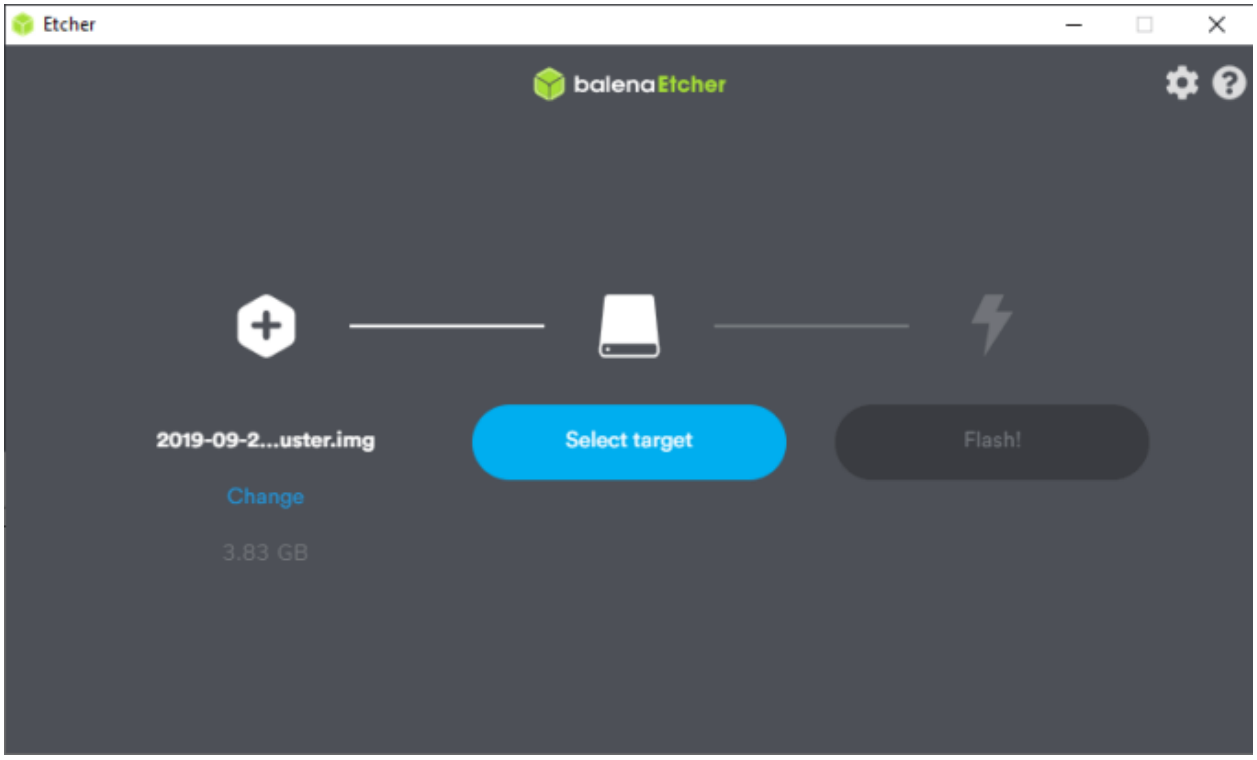
Hình 31: Hướng dẫn cài đặt balenaEtcher 2

Sau khi đã cài đặt phần mềm balenaEtcher dành cho hệ điều hành Windows lên máy tính, tiếp tục thì chúng ta sẽ sử dụng phần mềm này để sao chép tập tin OS Image vào thẻ nhớ. Đầu tiên, khởi động phần mềm VirtualBox như minh họa trong hình bên dưới.



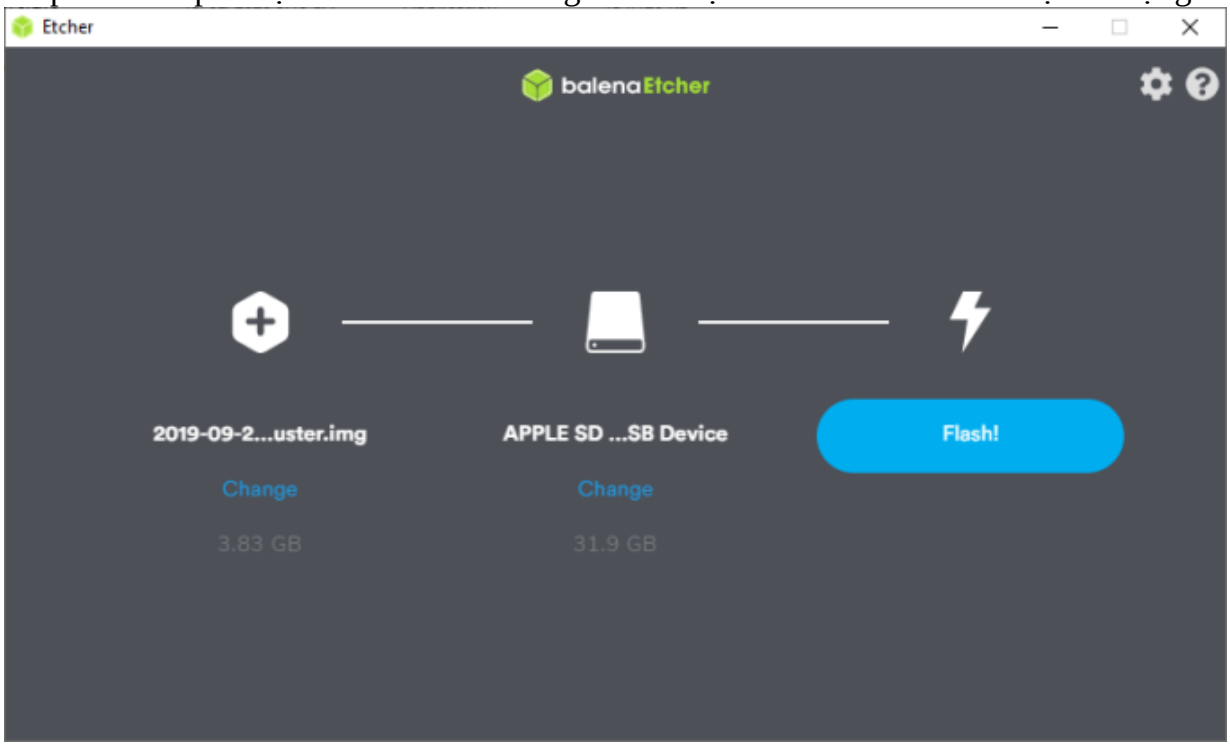
Hình 32: Hướng dẫn cài đặt balenaEtcher 3

Tiếp theo gắn thẻ nhớ vào đầu đọc trên máy tính, nhấp chuột vào nút “Select Image” và chọn tập tin OS Image “2019-09-26-raspbian-buster.zip” đã được tải về trước đó, như hình minh họa bên dưới.



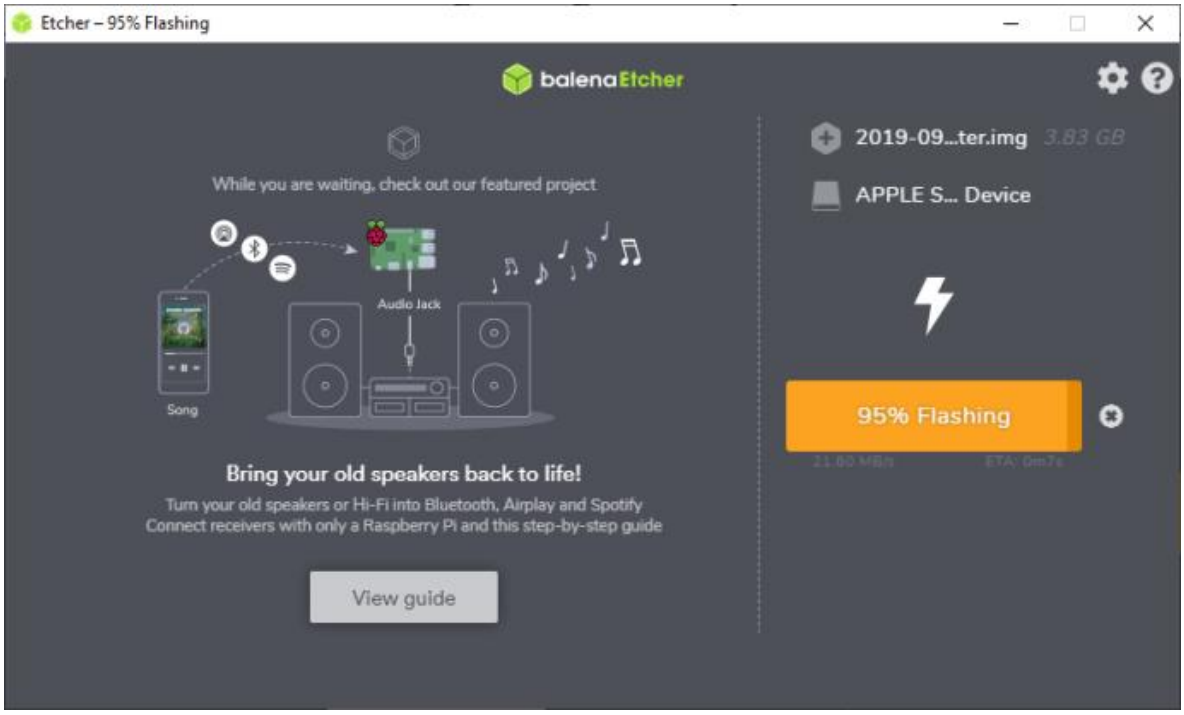
Hình 33: Hướng dẫn cài đặt balenaEtcher 4

Tiếp theo nhấp chuột vào nút “Select target” để chọn chính xác thẻ nhớ được sử dụng.



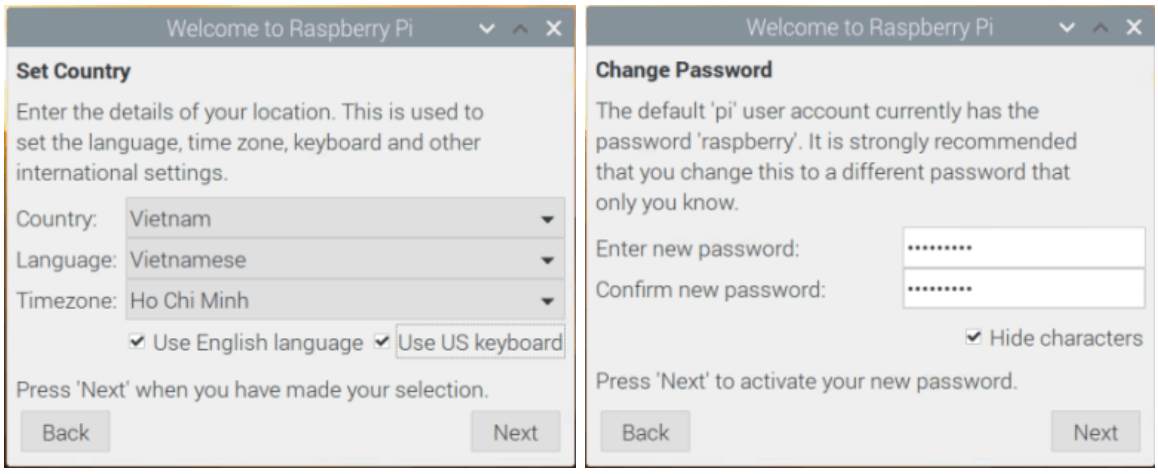
Hình 34: Hướng dẫn cài đặt balenaEtcher 5

Cuối cùng nhấp chuột vào nút “Flash!” để thực hiện việc chép tập tin OS Image vào thẻ nhớ. Chờ đợi một khoảng thời gian để quá trình được hoàn tất.



Hình 35: Hướng dẫn cài đặt balenaEtcher 6

Đến đây là hoàn tất các bước để sao chép hệ điều hành Linux vào trong thẻ nhớ. Tiếp theo chúng ta chỉ việc cắm thẻ nhớ này vào máy tính nhúng Raspberry Pi và bật điện khởi động máy, thực hiện tiếp một số thao tác thiết lập ban đầu cho Raspberry Pi như múi giờ, quốc gia, ngôn ngữ, tên đăng nhập, mật khẩu đăng nhập, kết nối mạng truyền thông,…



Hình 36: Hướng dẫn thiết lập Raspberry Pi 1

Sau khi quá trình thiết lập ban đầu hoàn tất, ta cần phải tiến hành khởi động lại Raspberry Pi và bắt đầu có thể sử dụng hệ điều hành Linux trên máy tính nhúng Raspberry Pi.

Cài đặt phần mềm Python trên hệ điều hành Linux

Để có thể mở cửa sổ Terminal nhanh chóng từ GUI, chúng ta nhấn tổ hợp phím "Ctrl + Alt + T". Sau đó thực hiện nhập vào các dòng lệnh Linux để cập nhật phiên bàn hệ điều hành Linux mới nhất và cài đặt phần mềm Python.

Để cập nhật phiên bàn hệ điều hành Linux mới nhất, nhập vào dòng lệnh sau:

~$ **sudo apt-get update**

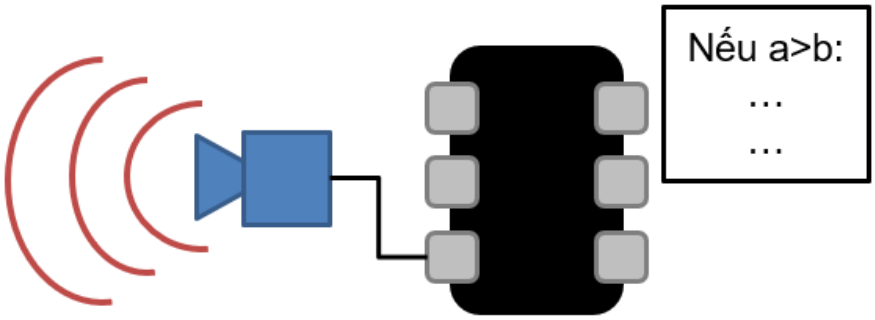
Để cài đặt phần mềm Python, nhập vào dòng lệnh sau:

~$ **sudo apt-get install python**

2.1.2. Lập trình điều khiển thiết bị ngoại vi vi điều khiển

- Điều khiển theo Sự kiện

- Chương trình trên Vi điều khiển chỉ được thực thi khi sự kiện xảy ra
- Phát hiện kích hoạt trên số PIN ngắt
- Các ứng dụng năng lượng thấp
- Thời gian thực thi thường không rõ ràng



Hình 37: Minh họa điều khiển sự kiện

- Điều khiển theo Độ trễ

- Chương trình trên vi điều khiển được thực thi định kỳ
  - Với độ trễ phần mềm, không có khả năng đa nhiệm
- Độ trễ của phần mềm vẫn cần thời gian tính toán
- Điều khiển thuần túy theo độ trễ không đáp ứng đầy đủ các yêu cầu thời gian thực

```
Trong khi Đúng:
    …
    …
    …
    Chờ (1 giây)
```

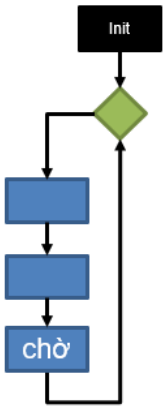
- Điều khiển theo Bộ hẹn giờ

- Chương trình trên vi điều khiển được thực thi định kỳ
  - Bộ hẹn giờ dùng để gọi tác vụ sau khoảng thời gian xác định
- Tác vụ phải được thực thi trong khoảng thời gian quy định
  - Bao gồm cả thời gian thực thi code
  - Độ trễ do truy cập phần cứng
  - Độ trễ do giao tiếp

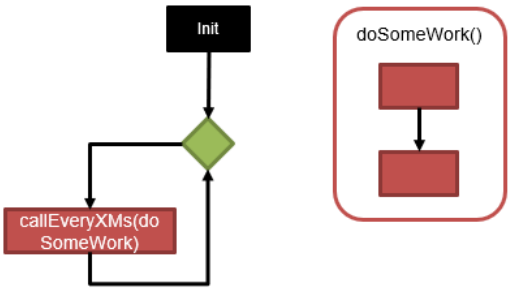
```
Hàm doSomeWork():
    …
    …
    …
callEveryXMs(doSomeWork())
```

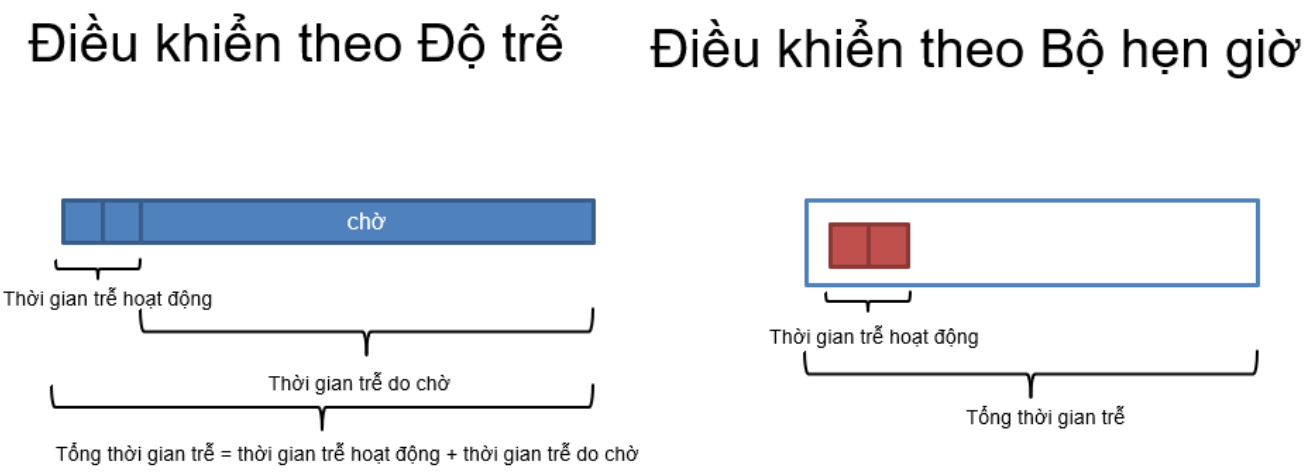
- Điều khiển theo Độ trễ so với điều khiển theo Bộ hẹn giờ

Điều khiển theo Độ trễ



Điều khiển theo Bộ hẹn giờ





Hình 38: So sánh Điều khiển trễ và Điều khiển theo thời gian

Bài tập: Truy cập tín hiệu số của đèn LED kép

Yêu cầu:

Các thư viện vi điều khiển cung cấp khả năng truy cập dễ dàng vào các cổng xuất nhập tín hiệu số. Trong nhiệm vụ này, việc sử dụng các đầu ra tín hiệu số sẽ được đào tạo. Trong ví dụ này, một đèn led màu kép sẽ được truy cập thông qua các cổng xuất tín hiệu số.

Đèn LED màu kép, được đặt trong một bóng epoxy 3mm, phát ra hai màu ánh sáng, thường là đỏ và xanh lá cây. Đèn LED hai màu có 3 chân, tức là 2 kết nối, và một cực âm chung và cực dương chung. Hai mối nối được bố trí đối song trong mạch và được nối với cực âm/cực dương. Điện áp dương được đặt vào một trong hai cực, khi đó đèn sẽ phát ra màu ánh sáng tương ứng. Nếu đảo cực của điện áp thì màu đèn khác sẽ sáng và chỉ một trong các kết nối có thể nhận điện áp. Loại đèn LED này thường được sử dụng làm đèn báo cho nhiều loại thiết bị, bao gồm tivi, máy ảnh kỹ thuật số và Điều khiển từ xa.

Nhiệm vụ của bạn là tạo một chương trình theo trình tự sau:

1. Không có màu LED nào sáng
2. Màu LED đỏ sáng, màu xanh lá không sáng
3. Màu LED xanh lá sáng, màu đỏ không sáng
4. Cả hai màu LED đều sáng
5. Quay lại 1

Giữa mỗi bước trong trình tự phải có độ trễ 1 giây. Thư viện times sẽ cung cấp cho bạn chức năng tạo độ trễ. Bạn có thể sử dụng chân (pin) GPIO hợp lệ bất kỳ. Đảm bảo rằng chân phần mềm tương ứng với chân phần cứng. Để tìm các chân hợp lệ, hãy nghiên cứu mô tả mô-đun GPIO.

Các bước:

1. Kết nối với vi điều khiển của bạn

2. Lưu trữ các số chân phần cứng tương ứng trong một biến
3. Thiết lập
    a. Khởi tạo bảng mạch GPIO
    b. Đặt các chân GPIO tương ứng làm đầu ra
    c. Đặt trạng thái đầu ra của chân GPIO thành thấp
4. Trong vòng lặp chính
    a. Triển khai trình tự của đèn LED lúc trước
5. Chuẩn bị một quy trình hủy trong trường hợp việc thực thi chương trình bị dừng lại

Đáp án tham khảo:

Lời giải (Số chân phụ thuộc vào việc triển khai phần cứng)

- redPin = Nr1
- greenPin = Nr2
- Trong hàm setup
    - Đặt redPin và greenPin làm đầu ra
    - Đặt redPin và greenPin là trạng thái thấp
- Trong vòng lặp chính
    - While true
        - Đặt redPin và greenPin là thấp
        - Chờ 1 giây
        - Đặt redPin là cao, greenPin là thấp
        - Chờ 1 giây
        - Đặt redPin là thấp, greenPin là cao
        - Chờ 1 giây
        - Đặt redPin và greenPin là cao
        - Trì hoãn 1 giây

2.1.3. Kiến trúc phần mềm trong môi trường vi điều khiển

Kiến trúc phần mềm

Kiến trúc đóng vai trò như một bản thiết kế cho một hệ thống. Nó cung cấp một sự trừu tượng để quản lý độ phức tạp của hệ thống và thiết lập một cơ chế giao tiếp và phối hợp giữa các thành phần.

- Nó xác định một giải pháp có cấu trúc để đáp ứng tất cả các yêu cầu kỹ thuật và hoạt động, đồng thời tối ưu hóa các thuộc tính chất lượng chung như hiệu suất và bảo mật.

- Hơn nữa, nó liên quan đến một tập hợp các quyết định quan trọng về tổ chức liên quan đến phát triển phần mềm và mỗi quyết định này có thể có tác động đáng kể đến chất lượng, khả năng bảo trì, hiệu suất và thành công chung của sản phẩm cuối cùng. Những quyết định này bao gồm

    - Lựa chọn các phần tử cấu trúc và các giao diện của chúng mà hệ thống được cấu thành.

- Hành vi như được chỉ định trong sự hợp tác giữa các yếu tố đó.
- Thành phần của các yếu tố cấu trúc và hành vi này thành hệ thống con lớn.
- Các quyết định về kiến trúc phù hợp với các mục tiêu kinh doanh.
- Phong cách kiến trúc hướng dẫn tổ chức.

Hệ điều hành vi điều khiển

- Yêu cầu:
  - Có khả năng bắt đầu, dừng và giám sát các tác vụ
  - Kiểm soát chính xác việc gọi hàm
  - Hỗ trợ đa nhiệm
  - Đáp ứng các yêu cầu thời gian thực

Thời gian thực

- Định nghĩa thời gian thực theo DIN 44300 Phần 9 (bản dịch):

“Thời gian thực là hoạt động của một hệ thống máy tính trong đó các chương trình xử lý dữ liệu gửi đến luôn hoạt động theo cách mà kết quả xử lý sẽ có trong một khoảng thời gian nhất định. Tùy thuộc vào ứng dụng, dữ liệu có thể được tạo ra theo phân phối ngẫu nhiên theo thời gian hoặc tại các thời điểm xác định trước ”.

Hệ điều hành thời gian thực (RTOS)

- Quản lý việc xử lý an toàn các yêu cầu từ một chương trình ứng dụng hoặc từ các tín hiệu gửi đến thông qua giao diện phần cứng trong một khoảng thời gian có thể xác định trước
- Khoảng thời gian có thể dài hoặc ngắn tùy thuộc vào yêu cầu ứng dụng
- Điều quan trọng là hoàn thành việc thực thi trong khoảng thời gian đó

RTOS mềm

- HĐH được sử dụng khi việc không thực thi trong giới hạn thời gian dẫn đến
  trạng thái hệ thống có vấn đề
- Phần lớn thời gian việc thực thi phải tuân thủ hạn định (deadline)

Hình 39: Raspberry Pi4

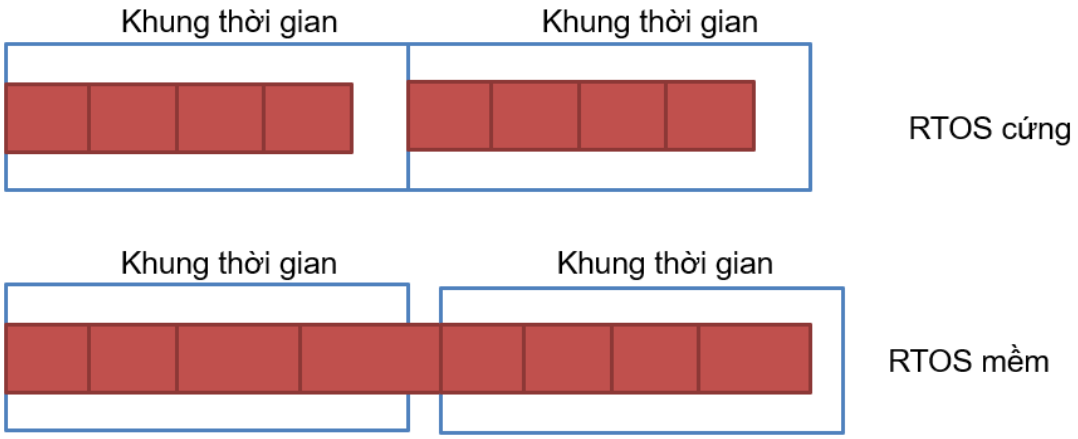
RTOS cứng

- HĐH được sử dụng khi việc không thực thi trong giới hạn thời gian dẫn đến
  hậu quả nghiêm trọng
- Phải luôn luôn tuân thủ deadline

https://www.bosch-connectivity.com/media/product_detail_xdk/xdk_016_res_1984x1116.jpg

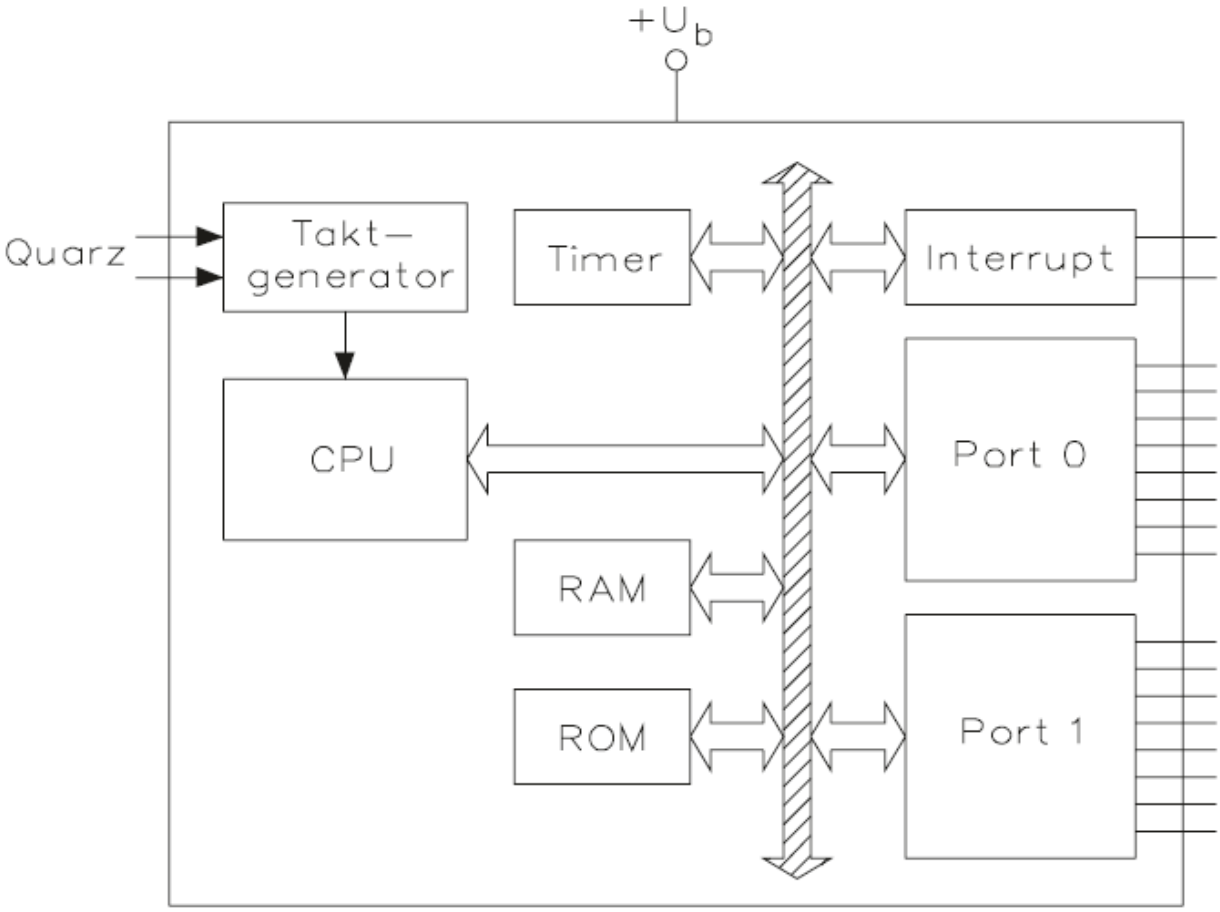
Hình 40: Cảm biến XDK

RTOS cứng và mềm



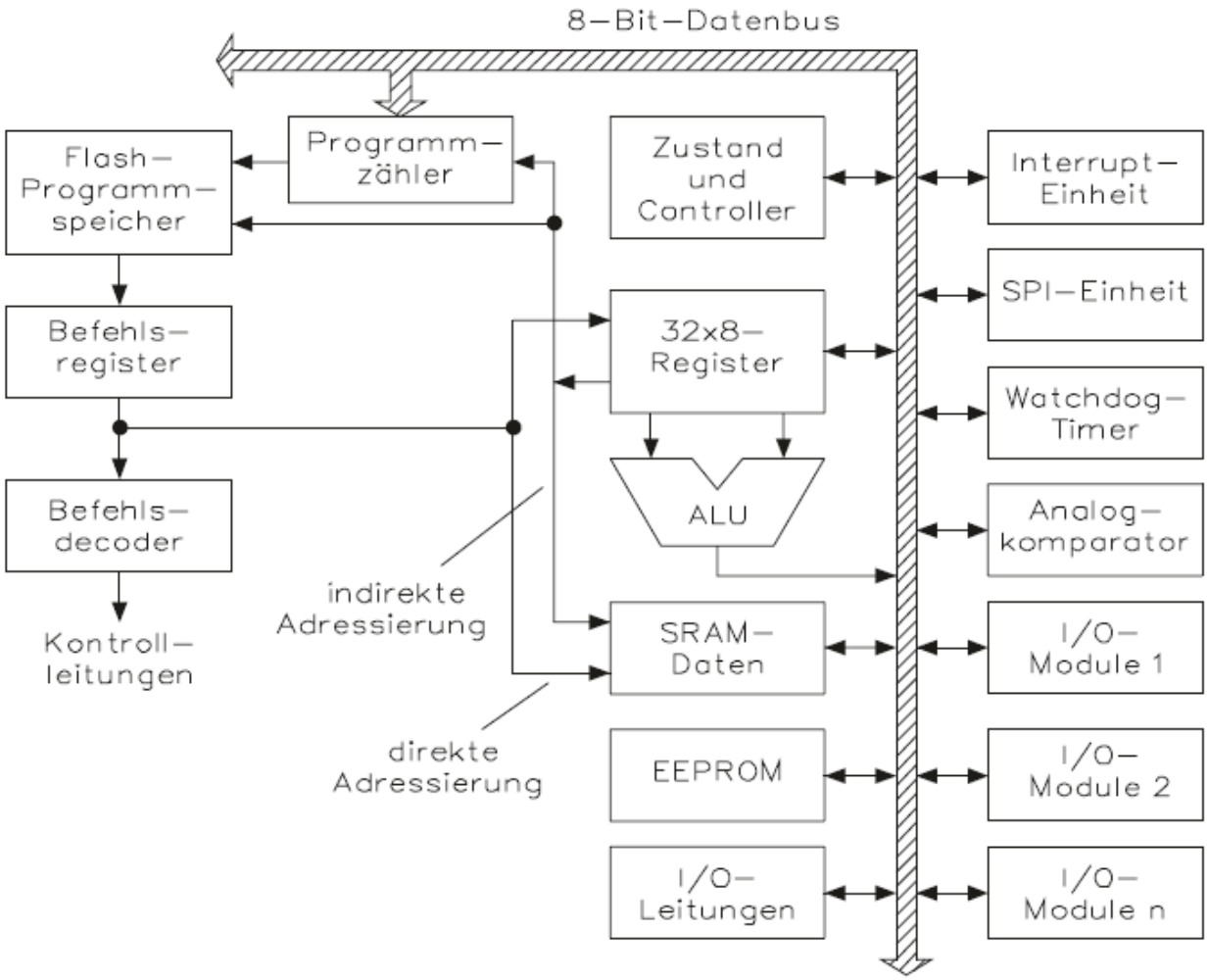
Hình 41: Minh họa RTOS cứng và mềm

Cấu trúc cơ bản vi điều khiển



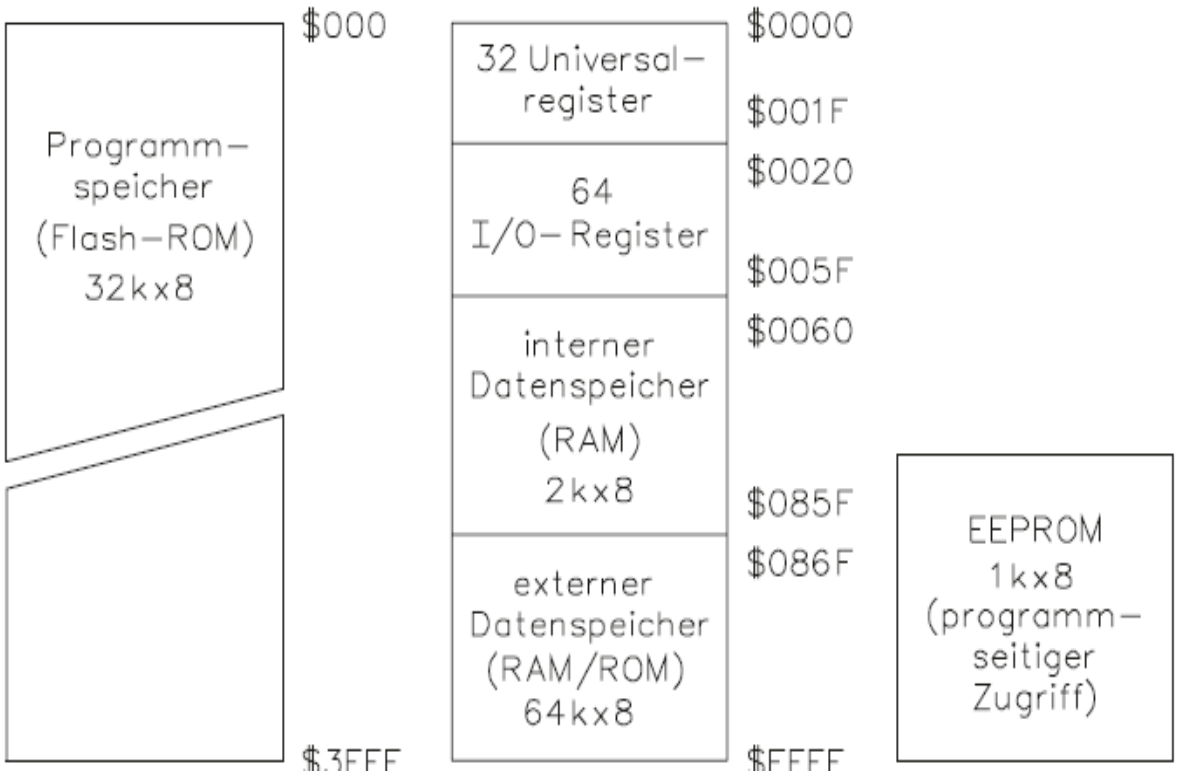
Biểu đồ 11: Cấu trúc I / O của vi điều khiển

Các khối chức năng vi điều khiển



Biểu đồ 12: Sơ đồ khối chức năng của vi điều khiển

Sự phân bố bộ nhớ



Biểu đồ 13: Phân bố bộ nhớ trong vi điều khiển

2.2. Kiểm soát phần cứng

Chương này sẽ đưa ra các nhiệm vụ điều khiển phần cứng cơ bản. Việc triển khai thực sự sẽ phụ thuộc vào kiểu phần cứng đang có. Các nhiệm vụ có thể được thay đổi tùy thuộc vào phần cứng được cung cấp. Ban đầu, bài tập này được thiết kế cho vi điều khiển sử dụng điện áp đầu ra 3,3V. Nếu dùng vi điều khiển có điện áp đầu ra 5V, vẫn sử dụng được các hướng dẫn nhiệm vụ cơ bản mà không cần thay đổi, nhưng phải đảm bảo sử dụng đúng cảm biến và bộ truyền động tương thích với điện áp 5V. Không sử dụng phần cứng 3,3V trên phần cứng 5V và ngược lại.

2.2.1. Kiểm soát phần cứng bên ngoài (thiết bị truyền động và cảm biến)

Đọc vào đầu vào tín hiệu số

Có hai cách chính để phát hiện các thay đổi trên phần cứng và phản ứng với chúng. Cách đầu tiên là thông qua thăm dò (polling) và cách thứ hai là thông qua ngắt.

Phát hiện qua thăm dò

Yêu cầu:

Khi một đầu vào tín hiệu số được phát hiện thông qua thăm dò, người dùng cần tạo một chương trình trong đó vi điều khiển đọc tuần tự đầu vào tín hiệu số và phát hiện xem nó được đặt thành cao hay thấp. Cách lập trình này sẽ tốn điện và năng lượng

tính toán với mỗi lần lặp lại. Đối với các chương trình đơn giản và không bắt buộc phải tiết kiệm năng lượng, kỹ thuật này có thể được sử dụng.

Tạo một chương trình đặt đèn LED kép thành màu xanh lá nếu nút không được nhấn và sang màu đỏ nếu nút không được nhấn nút. Sử dụng các chân GPIO của bảng mạch theo mong muốn của bạn.

Các bước:

1. Nối nút và đèn LED với bảng
2. Lưu trữ các số chân phần cứng tương ứng trong một biến
3. Thiết lập
    a. Khởi tạo bảng mạch GPIO
    b. Đặt các chân GPIO LED tương ứng làm đầu ra
    c. Đặt trạng thái đầu ra của các chân GPIO LED thành thấp
    d. Đặt các chân GPIO nút tương ứng làm đầu vào
        i. Đặt điện trở kéo lên (pull up resistor) bên trong để kéo lên "pull_up_down=GPIO.PUD_UP"
4. Vòng lặp
    a. Đọc vào đầu vào tín hiệu số
    b. Kiểm tra xem nút đã được nhấn chưa
    c. Triển khai logic trên tùy thuộc vào kết quả

Đáp án tham khảo:

Lời giải (Số chân phụ thuộc vào việc triển khai phần cứng)

- redPin = Nr1
- greenPin = Nr2
- btnPin = Nr3
- Trong hàm setup
    - Đặt redPin và greenPin làm đầu ra
    - Đặt redPin và greenPin là trạng thái thấp
    - Đặt btnPin làm đầu vào và kích hoạt điện trở kéo lên
- Trong vòng lặp chính
    - While true
        - btnVal = digitalRead(btnPin)
        - if btnVal == pressed
            - Đặt redPin là cao
            - Đặt greenPin là thấp
        - else
            - Đặt redPin là thấp
            - Đặt greenPin là cao
        - Chờ 100ms

Phát hiện qua ngắt

Yêu cầu:

Khi một đầu vào tín hiệu số được phát hiện qua ngắt thì không cần thiết phải gọi hàm read in (đọc vào) theo cách thủ công. Thay vào đó, một hàm interrupt (ngắt) được tạo ra để lắng nghe sườn lên hoặc xuống (tùy thuộc vào việc triển khai) trên chân tương ứng. Nếu trạng thái thay đổi thì hàm interrupt sẽ gọi một hàm callback (gọi lại) để thực thi. Kiểu lập trình này cho phép lập trình hiệu quả về năng lượng và khả năng tính toán.

Đáp án tham khảo:

Lời giải (Số chân phụ thuộc vào việc triển khai phần cứng)

- redPin = Nr1
- greenPin = Nr2
- btnPin = Nr3
- Trong hàm setup
    - Đặt redPin và greenPin làm đầu ra
    - Đặt redPin và greenPin là trạng thái thấp
    - Đặt btnPin làm đầu vào và kích hoạt điện trở kéo lên
    - Tạo quy trình ngắt, gán hàm detect, đặt thời gian thoát ra là 200ms, kích hoạt phát hiện sườn lên và sườn xuống
- Trong hàm detect
    - btnVal = analogeRead(btnPin)
        - if btnVal == pressed
            - Đặt redPin là cao
            - Đặt greenPin là thấp
        - else
            - Đặt redPin là thấp
            - Đặt greenPin là cao
- Trong vòng lặp chính
    - While true
        - Bỏ qua

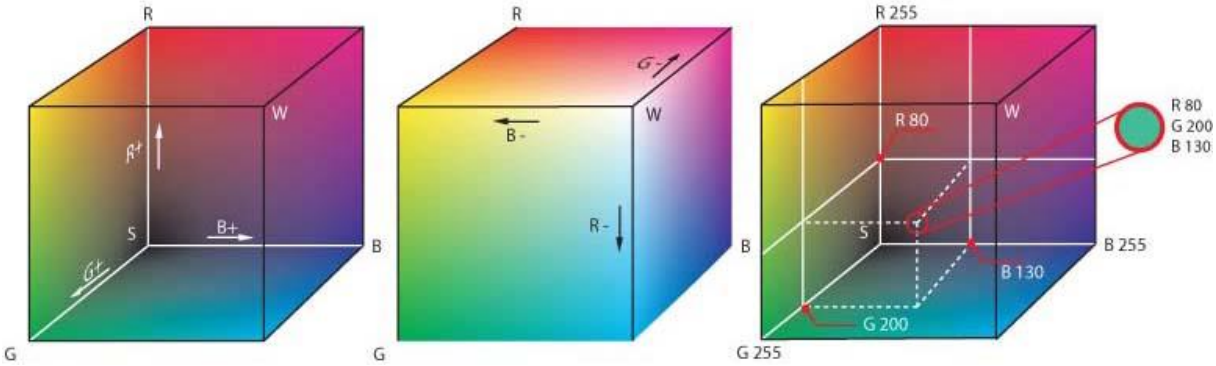
2.2.2. Sử dụng điều chế độ rộng xung để điều khiển phần cứng

Yêu cầu:

Điều chế độ rộng xung (PWM) là một phương pháp rất phổ biến để (ví dụ) làm mờ ánh sáng hoặc điều khiển tốc độ động cơ. Trong nhiệm vụ này, đèn LED RGB sẽ được sử dụng. Đèn LED RGB phát ra ánh sáng với nhiều màu sắc khác nhau. Bên trong vỏ nhựa trong hoặc mờ, với bốn chân, đèn được trang bị ba màu LED: đỏ, xanh lá và xanh lam. Với các độ sáng khác nhau, ba màu cơ bản trộn lẫn tạo ra các màu khác nhau và bằng cách điều khiển mạch, bạn có thể làm cho đèn LED RGB phát ra ánh sáng nhiều màu .

Tạo một chương trình để thiết lập đèn LED RGB thành 10 chế độ (pattern) màu khác nhau. Để đèn LED chuyển đổi giữa các chế độ với độ trễ 300ms. **Error! Reference**

source not found. hiển thị phổ màu RGB ở đây, bạn có thể lấy nó làm cơ sở để quyết định các màu bạn muốn chọn.



Hình 42: Điều chế độ rộng xung
https://upload.wikimedia.org/wikipedia/commons/0/03/RGB_farbwuerfel.jpg

Thông thường RGB được định trong khoảng từ 0 đến 255 cho mỗi màu. Tùy thuộc vào thư viện được sử dụng, có thể PWM nhận giá trị từ 0 đến 100, trong đó 0 có nghĩa là Không có xung và 100 có nghĩa là đầu ra DC đầy đủ, hoặc nhận giá trị 0 và 255, trong đó 0 có nghĩa là không có xung và 255 nghĩa là đầu ra 100%. Cân nhắc để nghiên cứu xem vi điều khiển của bạn cần loại đầu vào nào cho PWM. Nếu có thể, hãy sử dụng 2000hz làm tần số PWM.

Các bước:

1. Nối đèn LED RGB đến vi điều khiển
2. Quyết định các chế độ màu
3. Lưu các chế độ màu vào danh sách
4. Thiết lập
    a. Đặt các chân GPIO LED tương ứng làm đầu ra
    b. Đặt trạng thái đầu ra của các chân GPIO LED thành thấp
    c. Đặt các chân đã chọn làm chân PWM
5. Vòng lặp
    a. Tạo vòng lặp for
    b. Lặp lại mọi sự kết hợp màu sắc
    c. Sử dụng "ChangeDutyCycle (dc)" để thay đổi màu sắc thành giá trị mong muốn
    d. Sử dụng độ trễ 300ms

Đáp án tham khảo:

Lời giải (Số chân phụ thuộc vào việc triển khai phần cứng)

- redPin = Nr1
- greenPin = Nr2
- bluePin = Nr2
- colorListR = [cR1,…, cR10]
- colorListG = [cG1,…, cG10]
- colorListB = [cB1,…, cB10]

- Trong hàm setup
  - redPWM = đặt redPin, greenPin và bluePin làm đầu ra
  - greenPWM = đặt redPin, greenPin và bluePin là trạng thái cao
  - bluePWM = đặt redPin, greenPin và bluePin làm chân PWM với tần số 2000hz
  - Bắt đầu PWM với 100% chu kỳ nhiệm vụ
- Trong vòng lặp chính
  - While true
    - For i = 0 to 9
      - Đặt chu kỳ nhiệm vụ redPWM thành colorListR [i]
      - Đặt chu kỳ nhiệm vụ greenPWM thành colorListG [i]
      - Đặt chu kỳ nhiệm vụ bluePWM thành colorListB [i]
      - Chờ 300ms

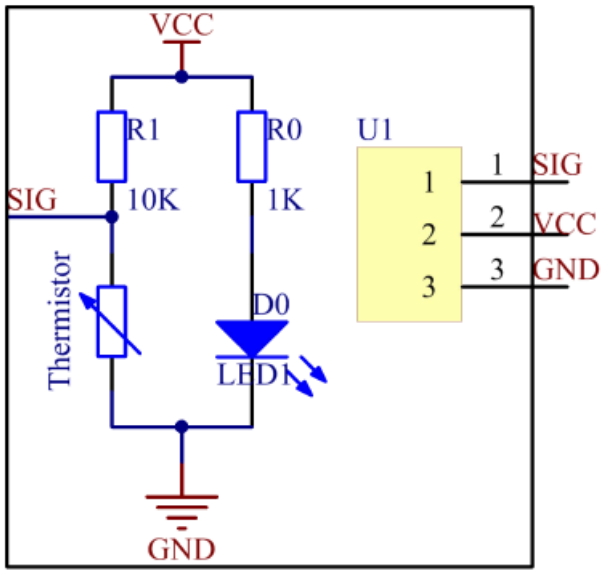
2.2.3. Sử dụng ADC (bộ chuyển đổi tương tự sang kỹ thuật số) để đọc tín hiệu tương tự

Đọc vào giá trị cảm biến

Vi điều khiển hoặc bảng mở rộng thường có khả năng đọc vào các giá trị cảm biến thông qua bộ chuyển đổi tương tự-số. Các bộ chuyển đổi này chuyển đổi một điện áp thành một biểu diễn số (nếu chúng là giao diện điện áp). Độ phân giải phụ thuộc vào loại bộ chuyển đổi ADC. Ngày nay các bộ chuyển đổi phổ biến là 8 - 12 Bit.

Trong ví dụ này, một điện trở nhiệt sẽ được sử dụng như một cảm biến nhiệt độ. Cảm biến nhiệt độ ghi lại nhiệt độ và chuyển nó thành tín hiệu đầu ra. Cảm biến nhiệt độ có thể được chia thành hai loại theo đặc điểm vật liệu và thành phần: Điện trở nhiệt (thermistor) và cặp nhiệt điện (thermocouple). Điện trở nhiệt là một trong những loại ra đời sớm hơn, được làm bằng vật liệu bán dẫn và thường có hệ số nhiệt độ âm (NTK), nghĩa là điện trở của chúng giảm khi nhiệt độ tăng. Vì điện trở thay đổi mạnh theo sự thay đổi nhiệt độ nên điện trở nhiệt là cảm biến nhiệt độ nhạy nhất.

Đọc vào điện áp của cảm biến nhiệt độ. Tính giá trị điện trở nhiệt dựa trên điện áp đọc được từ nó. Bạn có thể giả định rằng điện trở nhiệt nằm trong bộ chia điện áp như trong **Error! Reference source not found.**.



Hình 43: Bộ cảm biến 2.0 cho Raspberry Pi 4 Modell B

Nguồn: German-Sensor Set 2.0 für Raspberry Pi 4 Modell B 2020.07.08.pdf

Giá trị điện trở cơ bản của điện trở nhiệt có thể được đọc trong bảng dữ liệu (datasheet). Sau đó, sử dụng bảng dữ liệu và hoặc internet để nghiên cứu phương trình tính nhiệt độ từ điện trở tính được của điện trở nhiệt.

Các bước:

1. Nối điện trở nhiệt vào bảng vi điều khiển
2. Thiết lập
    a. Khởi động bộ chuyển đổi ADC
    b. Thiết lập chân ADC làm đầu vào
3. Vòng lặp
    a. Đọc giá trị điện trở nhiệt
    b. Tính toán điện áp điện trở nhiệt (cẩn thận nếu là 5V hay 3,3V)
    c. Dựa vào điện áp của điện trở nhiệt, tính điện trở của điện trở nhiệt
    d. Tìm phương pháp để tính giá trị điện trở nhiệt dựa trên datasheet và hoặc tìm kiếm trên internet (ví dụ gần đúng với các giá trị trong datasheet)
    e. In các giá trị nhiệt độ
    f. Độ trễ 500ms giữa mỗi lần đo

Đáp án tham khảo:

Lời giải (Số chân phụ thuộc vào việc triển khai phần cứng)

Điện áp và độ phân giải phụ thuộc vào phần cứng

- aInPin = Nr1
- Trong hàm setup
    - Bắt đầu ADC

- Đặt aInPin làm đầu vào
- Trong vòng lặp chính
  - While true
    - adcVal = analogeRead (aInPin)
    - voltageVal = Voltage * float(adcVal) /resolution
    - Rt = 10000 * voltageVal / (5 - voltageVal)
    - temp = 1/(((math.log(Rt / 10000)) / 3950) + (1 / (273.15+25)))
    - temp = temp - 273.15
    - print(temp)
    - Chờ 500ms

2.2.4. Chuyển đổi phép đo từ giá trị kỹ thuật số sang đại lượng vật lý

Đo khoảng cách

Yêu cầu:

Code đã cho đọc vào cảm biến siêu âm. Đầu tiên, cần hiểu rõ code đã cho. Sau đó, chương trình sẽ được cập nhật để nếu khoảng cách của một đối tượng dưới 7cm thì thông báo cảnh báo: “Danger object is too close” sẽ được xuất ra.

Code:

```
import RPi.GPIO as GPIO
import time
TRIG = 11
ECHO = 12
# Setup Pins
def setup():
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(TRIG, GPIO.OUT)
        GPIO.setup(ECHO, GPIO.IN)
#Calculate Distance
def calcDistance():
         GPIO.output(TRIG, 0)
         time.sleep(0.000002)
         GPIO.output(TRIG, 1)
         time.sleep(0.00001)
         GPIO.output(TRIG, 0)
        while GPIO.input(ECHO) == 0:
                 a = 0
        time1 = time.time()
         while GPIO.input(ECHO) == 1:
                 a = 1
         time2 = time.time()
         #calculate time between start and stop
```

```
        timeDel = time2 - time1
        # convert time to distance
        return timeDel * 340 / 2 * 100
def loop():
        while True:
                dis = calcDistance()
                time.sleep(0.3)
        def destroy():
        GPIO.cleanup()
if __name__ == "__main__":
        setup()
        try:
                loop()
        except KeyboardInterrupt:
        destroy()
```

Các bước:

1. Hiểu code đã cho
2. Cập nhật code để xuất cảnh báo khi khoảng cách dưới 7cm

Đáp án tham khảo:

```
import RPi.GPIO as GPIO
import time
TRIG = 11
ECHO = 12
# Setup Pins
def setup():
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(TRIG, GPIO.OUT)
        GPIO.setup(ECHO, GPIO.IN)
#Calculate Distance
def calcDistance():
        GPIO.output(TRIG, 0)
        time.sleep(0.000002)
        GPIO.output(TRIG, 1)
        time.sleep(0.00001)
        GPIO.output(TRIG, 0)
        while GPIO.input(ECHO) == 0:
                a = 0
        time1 = time.time()
        while GPIO.input(ECHO) == 1:
                a = 1
        time2 = time.time()
        #calculate time between start and stop
```

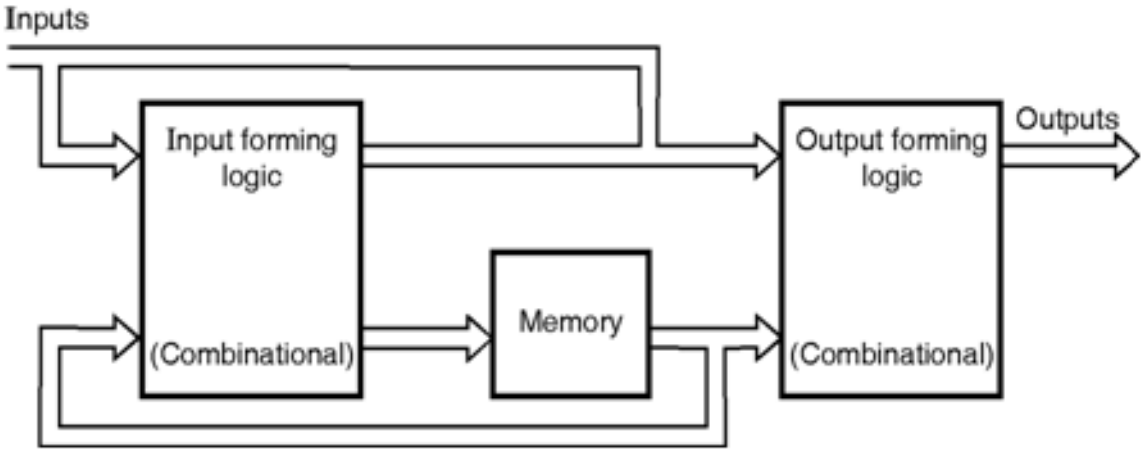
```
        timeDel = time2 - time1
        # convert time to distance
        return timeDel * 340 / 2 * 100
def loop():
        while True:
                dis = calcDistance()
                if dis < 7:
                        print("Danger object is too close")
                time.sleep(0.3)
        def destroy():
         GPIO.cleanup()
if __name__ == "__main__":
         setup()
          try:
                loop()
         except KeyboardInterrupt:
         destroy()
```

3. Tạo các chương trình đo lường

3.1. Giới thiệu về máy trạng thái (state machine)

3.1.1. Các nguyên tắc cơ bản của máy trạng thái

Máy trạng thái là một hệ thống có thể được mô tả dưới dạng một tập hợp các trạng thái mà hệ thống có thể nhập vào. Trạng thái tiếp theo đạt được phụ thuộc vào các yếu tố đầu vào và trạng thái hiện tại. Đầu ra cũng phụ thuộc vào đầu vào và trạng thái hiện tại. Trạng thái có thể là một tập hợp các giá trị được đo tại các điểm khác nhau trong mạch. Một ván lật đơn giản có hai trạng thái mà nó có thể tồn tại. Phần lớn các máy trạng thái thực tế sử dụng flip-flops có khóa làm yếu tố lưu trữ. Mã xác định mỗi trạng thái sau đó tương ứng trực tiếp với mã được chứa bởi flip-flops. Mô hình chung của máy tuần tự được thể hiện trong hình dưới đây.



Hình 44: Các nguyên tắc cơ bản của máy trạng thái

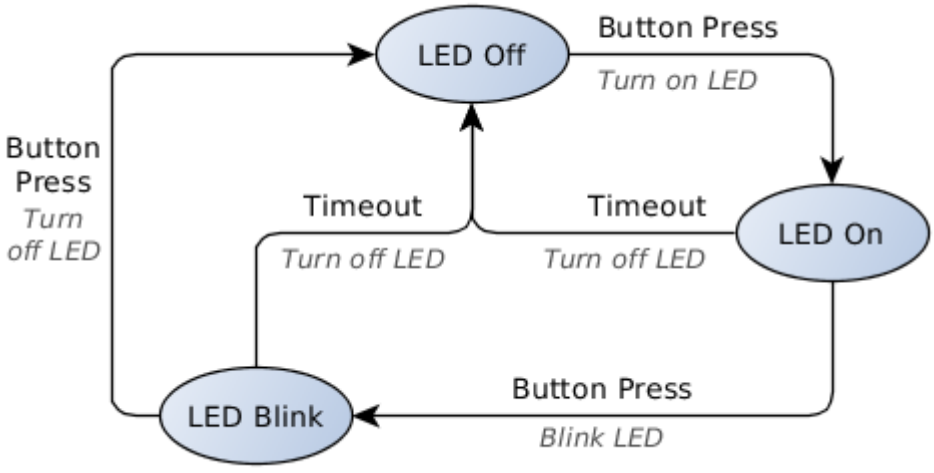
Mô hình chung của mạch tuần tự

Mô hình này còn được gọi là máy Mealy theo tên người đầu tiên đề xuất mô hình này. Phần logic hình thành đầu vào (IFL) và logic hình thành đầu ra (OFL) được tạo thành từ các mạch logic tổ hợp. Phần bộ nhớ chứa trạng thái của hệ thống. Một biến thể nhỏ của máy Mealy là máy Moore chỉ sử dụng bộ nhớ để điều khiển OFL. Trong trường hợp này, đầu ra là một hàm chỉ trạng thái của hệ thống. Khi một biến được điều khiển bằng đồng hồ, nó được coi là một biến đồng bộ. Nếu một máy thay đổi trạng thái để đáp ứng với đồng hồ và tất cả các đầu vào là đồng bộ, chúng tôi sẽ phân loại mạch đó là hệ thống đồng bộ. Nếu các thay đổi trạng thái xảy ra theo phản ứng của đồng hồ, nhưng một hoặc nhiều đầu vào không được điều khiển bằng đồng hồ, máy sẽ được gọi là hệ thống chủ yếu đồng bộ. Nếu các thay đổi trạng thái được hướng vào đầu vào thay vì theo hướng đồng hồ, hệ thống là một hệ thống không đồng bộ.

3.1.2. Thực hiện các trạng thái máy cho vi điều khiển

Máy trạng thái là công cụ hữu ích mà trong ứng dụng phù hợp có thể đơn giản hóa việc thiết kế phần firmware vi điều khiển. Chúng cho phép bạn tạo một hệ thống hướng sự kiện có thể thay đổi phản ứng của nó với các đầu vào dựa trên trạng thái bên trong của nó. Ví dụ dưới đây giới thiệu một cách để cấu trúc một máy trạng thái trong môi trường vi điều khiển.

Hãy xem xét một hệ thống ví dụ có chứa một bộ vi điều khiển được kết nối với đèn LED và một nút nhấn. Trong hệ thống ví dụ này, một lần nhấn nút sẽ bật đèn LED, một lần nhấn nút thứ hai sẽ làm đèn LED nhấp nháy và nếu nhấn nút một lần nữa, đèn LED sẽ tắt. Ngoài ra, hệ thống phải tắt đèn LED sau một thời gian không hoạt động. Nếu nút không được nhấn trong 10 giây qua, đèn LED sẽ tắt.



Hình 45: ví dụ Biểu đồ trạng thái hệ thống

Trong ví dụ trên, mỗi khi nhấn nút, hệ thống phải thực hiện một trong ba hành động có thể (bật đèn LED, nhấp nháy đèn LED hoặc tắt đèn LED). Phản ứng với một lần nhấn nút phụ thuộc vào trạng thái hiện tại của đèn LED, trạng thái này có thể bị ảnh

hưởng bởi một sự kiện khác trong hệ thống (thời gian chờ 10 giây). Do đó, chúng ta phải tạo một hệ thống theo dõi đèn LED và sử dụng trạng thái của nó để quyết định những gì cần làm để phản hồi lại đầu vào.

Một máy trạng thái sẽ tạo ra một đầu ra được xác định bởi cả trạng thái bên trong hiện tại của hệ thống và đầu vào mà nó nhận được. Bằng cách thay đổi trạng thái, hệ thống có thể tạo ra một đầu ra khác với cùng một đầu vào như trước đây. Nếu chúng ta sử dụng các trạng thái để theo dõi đèn LED, thì chúng ta có thể xác định đầu ra nào sẽ tạo ra khi nhấn nút.

Cấu trúc máy trạng thái

Đối với các ứng dụng vi điều khiển, chúng ta hãy sử dụng định nghĩa của một máy trạng thái hữu hạn. Một máy trạng thái hữu hạn có một tập hợp các đầu vào, đầu ra và trạng thái đã biết. Biểu đồ trạng thái trong Hình 1 chứa tất cả các phần tử chúng ta cần để mô tả máy trạng thái của mình; hãy xác định bốn phần tử khác nhau của một máy trạng thái.

Đầu vào — bất kỳ sự kiện nào yêu cầu hệ thống của chúng tôi tạo đầu ra hoặc thay đổi hành vi của nó đều là đầu vào. Ví dụ của chúng tôi có hai đầu vào: thời gian chờ 10 giây và nút bấm. Trong biểu đồ trạng thái của chúng tôi, các đầu vào được liệt kê phía trên các mũi tên kết nối các trạng thái.

Chuyển đổi trạng thái — các mũi tên trong biểu đồ trạng thái biểu thị các chuyển đổi trạng thái. Những điều này xác định khi nào hệ thống của chúng tôi sẽ thay đổi hành vi của nó bằng cách thay đổi trạng thái bên trong của nó. Chuyển đổi trạng thái chỉ có thể được kích hoạt bởi một đầu vào. Trong ví dụ của chúng tôi, chúng tôi kích hoạt chuyển đổi trạng thái mỗi khi đầu vào tạo ra đầu ra thay đổi trạng thái của đèn LED. Bằng cách này, trạng thái của hệ thống luôn theo dõi đèn LED. Các chuyển đổi trạng thái cũng xác định cách hệ thống của chúng tôi được phép thay đổi hành vi. Ví dụ: không có mũi tên nào kết nối trạng thái tắt của đèn LED và trạng thái nhấp nháy của đèn LED, do đó đèn LED không bao giờ có thể thay đổi trực tiếp từ trạng thái tắt sang trạng thái nhấp nháy.

Đầu ra — các hành động cần được hệ thống thực hiện để đáp ứng với đầu vào là đầu ra. Trong Hình 1, các đầu ra được liệt kê in nghiêng dưới các mũi tên chuyển đổi trạng thái. Giống như chuyển đổi trạng thái, hệ thống của chúng tôi chỉ có thể tạo ra một đầu ra sau một sự kiện đầu vào. Hệ thống của chúng tôi có ba đầu ra: làm cho đèn LED bật, nhấp nháy hoặc tắt.

Các trạng thái — các vòng tròn trong biểu đồ trạng thái đại diện cho các trạng thái. Trạng thái là một danh sách các quy tắc cho hệ thống biết phải làm gì khi một sự kiện đầu vào xảy ra. Khi một đầu vào xảy ra, hệ thống sẽ xem xét tập hợp các quy tắc được xác định trong trạng thái bên trong hiện tại của nó và tìm kiếm đầu ra hoặc chuyển đổi trạng thái nào mà nó cần tạo ra hoặc liệu nó không nên làm gì để đáp ứng với đầu vào.

Trạng thái của ví dụ là ba hành vi LED khác nhau mà chúng tôi đã nêu trước đó; máy trạng thái chỉ được phép tồn tại ở một trong những trạng thái này.

Nhưng làm thế nào để chúng ta quyết định trạng thái của chúng ta sẽ như thế nào, hoặc thậm chí là chúng ta nên có bao nhiêu? Mỗi trạng thái là một tập hợp các quy tắc khác nhau xác định cách hệ thống sẽ phản ứng với các đầu vào, vì vậy tất cả các trạng thái cần bao hàm các hành vi đầu ra khác nhau mà chúng ta cần. Chúng ta hãy xem xét các mối quan hệ đầu vào-đầu ra của ví dụ của chúng ta trong một bảng thay vì biểu đồ trạng thái ở trên.

State	Input	Output
LED Off	Timeout	None
LED On / LED Blink	Timeout	Turn off LED
LED Off	Button Press	Turn on LED
LED On	Button Press	Start blinking LED
LED Blink	Button Press	Turn off LED

Đầu vào nhấn nút có thể tạo ra ba đầu ra có thể dựa trên mô tả ví dụ của chúng tôi. Bởi vì đầu ra phụ thuộc vào trạng thái và đầu vào, cách duy nhất một đầu vào được phép tạo ra ba đầu ra khả thi là có ba trạng thái. Sự kiện thời gian chờ dẫn đến hai hành động có thể xảy ra, nhưng những hành động này có thể được đưa vào ba trạng thái khác của chúng tôi.

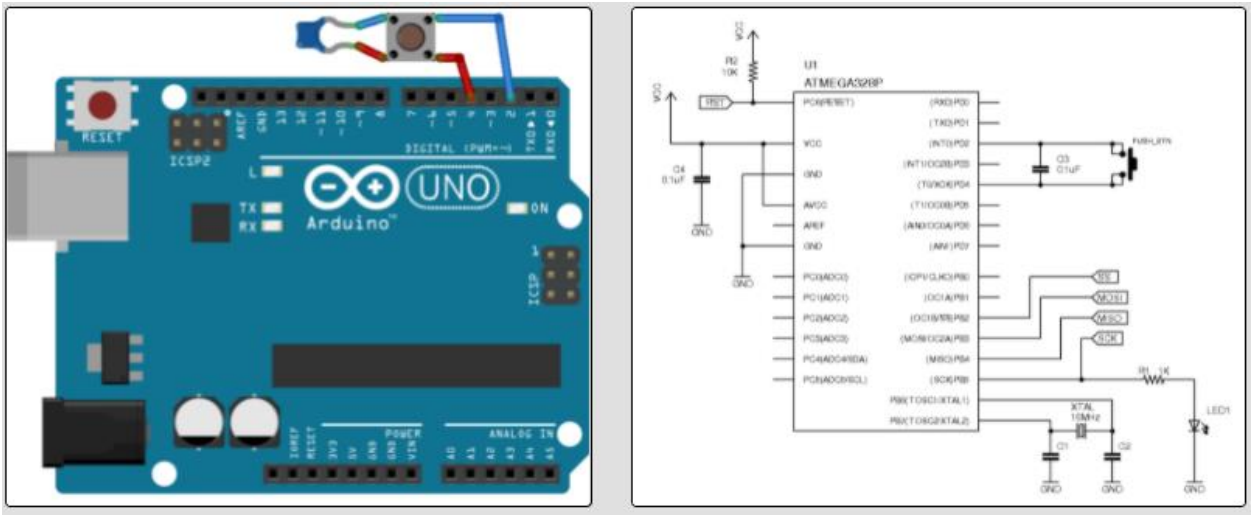
Cuối cùng, điều quan trọng là phải xác định trạng thái ban đầu của hệ thống của chúng tôi. Đây là trạng thái mà hệ thống sẽ bắt đầu khi bật nguồn hoặc nếu một thiết lập lại xảy ra. Máy trạng thái của chúng ta sẽ luôn khởi động ở trạng thái tắt đèn LED.

3.1.3. Thiết kế phần mềm của một chương trình tuần tự thông qua các trạng thái máy

Phần cứng

Trước khi chúng ta triển khai các phần tử khác nhau của máy trạng thái của chúng ta trong mã, hãy xem xét phần cứng chúng ta sẽ sử dụng.

Phần cứng ví dụ có thể được đặt cùng với Arduino, một công tắc tạm thời và một tụ điện để gỡ lỗi công tắc. Công tắc được kết nối qua các chân kỹ thuật số 2 và 4 trên Arduino (PD2 / INT0 và PD4, tương ứng trên Atmega) với một tụ điện gỡ lỗi có giá trị phù hợp (100nF) song song. Chân 4 được đặt làm bồn rửa và chân 2 được thiết lập làm đầu vào với tính năng kéo lên bên trong được bật. Ví dụ này cũng sử dụng đèn LED trên bo mạch được kết nối với chân 13 (PB5 / SCK trên Atmega).



Hình 46: Phần cứng được sử dụng để thực hiện ví dụ của chúng tôi

Ngoài ra, phần cứng mẫu có thể được xây dựng trên breadboard. Sơ đồ bên phải hiển thị mạch Atmega328 có liên quan được sử dụng trong ví dụ này.

Lập trình Máy trạng thái

Dựa trên định nghĩa của chúng ta về máy trạng thái, chúng ta cần lập trình mỗi trạng thái dưới dạng danh sách các lệnh để thực thi (đầu ra, chuyển đổi trạng thái) cho mỗi đầu vào có thể và sau đó để mã điều hướng đến tập hợp các lệnh chính xác. Một cách đơn giản để thực hiện việc này là sử dụng các câu lệnh switch lồng nhau như hình dưới đây.

```
/*  Example of nested switch statements     */
switch(system_state){
   case led_off:
      switch(system_input){
         case button_press:
            turn the led on;        // Output
            system_state = led_on;  // State transition
            break;
         case timeout:
            break;                  // do nothing
      }
      break;
   case led_on:
      switch(system_input){
```

```
            case button_press:
                blink the led;              // Output
                system_state = led_blink;   // State transition
                break;
            case timeout:
                turn off the led;           // Output
                system_state = led_off;     // State transition
                break;
        }
        break;
    case led_blink:
        switch(system_input){
            case button_press:
                turn off led;           // Output
                system_state = led_off; // State transition
                break;
            case timeout:
                turn off led;           // Output
                system_state = led_off; // State transition
                break;
        }
        break;
}
```

Bài tập ứng dụng:

Máy trạng thái đơn giản: Tạo một máy trạng thái đơn giản bao gồm bốn trạng thái. Các trạng thái sẽ được gọi là “S1”, “S2”, “S3” và “S4”. Trạng thái S1 sẽ dẫn đến S2, S2 đến S3, S3 đến S4 và S4 quay trở lại S1. Sự chuyển tiếp sẽ chỉ xảy ra khi một nút được nhấn. Khi ở trạng thái mới, chương trình sẽ liên tục xuất ra tên trạng thái. Đảm bảo rằng trạng thái chỉ thay đổi một lần cho mỗi lần nhấn nút.

Đáp án tham khảo:

Lời giải (Số chân phụ thuộc vào việc triển khai phần cứng)

- activeState = “S1”

- btnPin = Nr1
- btnPressed = 0
- Trong hàm setup
  - Đặt btnPin làm đầu vào và kích hoạt điện trở kéo lên
-
- While true
  - If activeState == “S1”
    - Print(“S1”)
    - btnVal=digitalRead(btnPin)
    - if btnVal == pressed
      - btnPressed = 1
    - Elif btnPressed == 1 and btnVal == not pressed
      - bnPressed = 0
      - activeState = “S2”
  - Elif activeState == “S2”
    - Print(“S2”)
    - btnVal=digitalRead(btnPin)
    - if btnVal == pressed
      - btnPressed = 1
    - Elif btnPressed == 1 and btnVal == not pressed
      - bnPressed = 0
      - activeState = “S3”
  -
  - Elif activeState == “S3”
    - Print(“S3”)
    - btnVal=digitalRead(btnPin)
    - if btnVal == pressed
      - btnPressed = 1
    - Elif btnPressed == 1 and btnVal == not pressed
      - bnPressed = 0
      - activeState = “S4”
    -
  - Elif aciveState == “S4”
    - Print(“S4”)
    - btnVal=digitalRead(btnPin)
    - if btnVal == pressed
      - btnPressed = 1
    - Elif btnPressed == 1 and btnVal == not pressed
      - bnPressed = 0
      - activeState = “S1”
  - Chờ 200ms

3.2. Chương trình đo lường

Yêu cầu:

Một chương trình đo lường thực tế thường không bắt đầu ngay lập tức khi vi điều khiển được kết nối với nguồn điện. Thay vào đó, nó sẽ được chạy trong một máy trạng thái. Với trạng thái ban đầu "init", đây là khi chương trình chờ nhập vào người dùng để bắt đầu. Trạng thái đo lường "measure" và thường cả trạng thái lỗi "error" trong trường hợp có bất kỳ lỗi nào xảy ra. Các chương trình đo lường phức tạp hơn sẽ có nhiều trạng thái hơn. Chẳng hạn như các loại phép đo khác nhau, v.v.

Trong nhiệm vụ này, một chương trình đo đơn giản phải được triển khai. Chương trình sẽ chờ nguyên ở trạng thái init cho đến khi người dùng nhấn một nút. Khi vào trạng thái init, đầu tiên chương trình sẽ in ra "Read for measurement please press the button". Nếu một nút được nhấn thì chương trình sẽ in "Starting to measure" một lần. Sau đó, chương trình sẽ đọc vào cảm biến bạn chọn và in giá trị cảm biến. Chương trình sẽ thực hiện 100 phép đo với độ trễ giữa các phép đo là 500ms. Sau khi tất cả các lần đo đã kết thúc, chương trình sẽ in ra "measurement finished" và trở về trạng thái init. Nếu người dùng nhấn nút thêm một lần nữa trong quá trình đo thì chương trình sẽ dừng trạng thái đo và chuyển sang trạng thái lỗi. Khi đó chương trình sẽ xuất ra một lần "Error happened during measurement please reset". Nếu người dùng nhấn nút thêm một lần nữa, nó sẽ trở lại trạng thái init.

Đáp án tham khảo:

Lời giải (Số chân phụ thuộc vào việc triển khai phần cứng)

- activeState = "S1"
- btnPin = Nr1
- btnPressed = 0
- entry = 1
- Trong hàm setup
    - Đặt btnPin làm đầu vào và kích hoạt điện trở kéo lên
    - Bắt đầu ADC
    - Đặt aInPin làm đầu vào
    - Cnt = 0

- While true
    - If activeState == "init"
        - If entry == 1
            - Print("Read for measurement please press the button")
            - entry = 0
        - btnVal=digitalRead(btnPin)
        - if btnVal == pressed
            - btnPressed = 1
        - Elif btnPressed == 1 and btnVal == not pressed
            - bnPressed = 0
            - activeState = "measure"
            - entry = 1

- Elif activeState == “measure”
  - If entry == 1
    - Print(“Starting to measure”)
    - entry = 0
    - cnt = 0
  - sensVal = analogeRead(aInPin)
  - print(sensVal)
  - cnt = cnt+1
  - if btnVal == pressed
    - btnPressed = 1
  - Elif btnPressed == 1 and btnVal == not pressed
    - bnPressed = 0
    - activeState = “error”
  - if cnt >= 99
    - aciveState = “init”
- Elif activeState == “error”
  - If entry == 1
    - Print(“Error happened during measurement please reset”)
    - entry = 0
  - btnVal=digitalRead(btnPin)
  - if btnVal == pressed
    - btnPressed = 1
  - Elif btnPressed == 1 and btnVal == not pressed
    - bnPressed = 0
    - activeState = “init”
  -
- Chờ 500ms

4. Phương pháp giao tiếp và trao đổi dữ liệu

4.1. Yêu cầu HTTP

4.1.1. Kiến thức cơ bản về giao tiếp dựa trên web

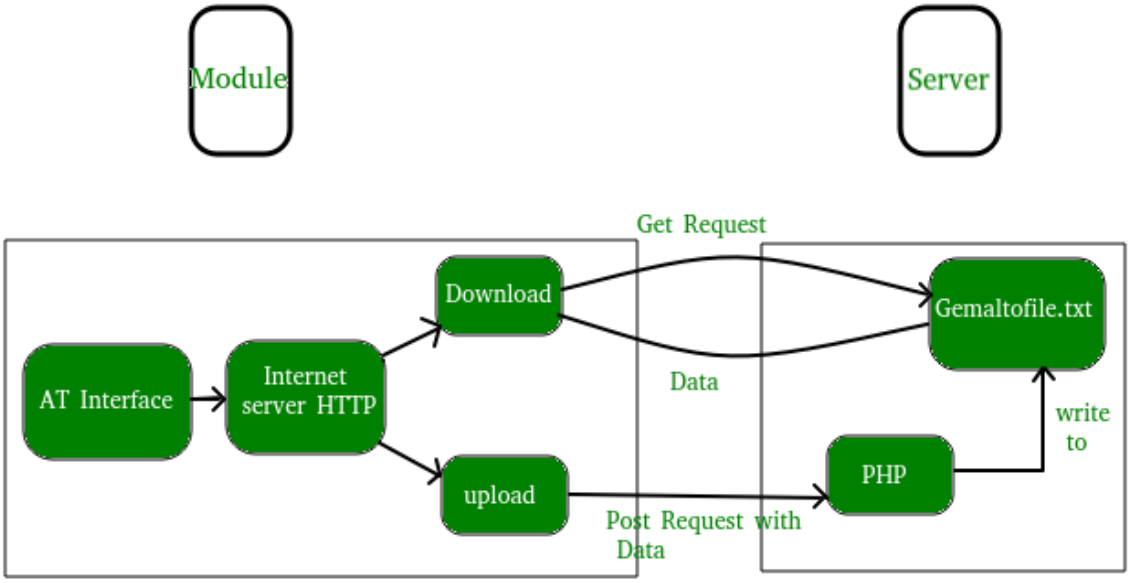
HTTP là một tập hợp các giao thức được thiết kế để cho phép giao tiếp giữa máy khách và máy chủ. Nó hoạt động như một giao thức phản hồi yêu cầu giữa máy khách và máy chủ.

Trình duyệt web có thể là máy khách và ứng dụng trên máy tính lưu trữ trang web có thể là máy chủ.

Vì vậy, để yêu cầu phản hồi từ máy chủ, chủ yếu có hai phương pháp:

- GET: để yêu cầu dữ liệu từ máy chủ.
- POST: gửi dữ liệu cần xử lý lên máy chủ.

Đây là một sơ đồ đơn giản giải thích khái niệm cơ bản của các phương thức GET và POST.



Hình 47: Giao tiếp dựa trên web

Bây giờ, để thực hiện các yêu cầu HTTP trong python, chúng ta có thể sử dụng một số thư viện HTTP như:

httplib

urllib

requests

Để tải xuống và cài đặt thư viện Yêu cầu, hãy sử dụng lệnh sau:

pip install requests

Making a Get request

```
# importing the requests library
import requests
# api-endpoint
URL = "http://maps.googleapis.com/maps/api/geocode/json"
# location given here
location = "delhi technological university"
# defining a params dict for the parameters to be sent to the API
PARAMS = {'address':location}
# sending get request and saving the response as response object
r = requests.get(url = URL, params = PARAMS)
# extracting data in json format
```

```
data = r.json()
# extracting latitude, longitude and formatted address
# of the first matching location
latitude = data['results'][0]['geometry']['location']['lat']
longitude = data['results'][0]['geometry']['location']['lng']
formatted_address = data['results'][0]['formatted_address']
 # printing the output
print("Latitude:%s\nLongitude:%s\nFormatted Address:%s"
      %(latitude, longitude,formatted_address))
```

Ví dụ trên tìm vĩ độ, kinh độ và địa chỉ được định dạng của một vị trí nhất định bằng cách gửi yêu cầu GET tới API Google Maps. API (Giao diện lập trình ứng dụng) cho phép bạn truy cập các tính năng bên trong của chương trình một cách hạn chế. Và trong hầu hết các trường hợp, dữ liệu được cung cấp ở định dạng JSON (JavaScript Object Notation) (được triển khai dưới dạng các đối tượng từ điển trong Python!).

Bây giờ, để lấy dữ liệu từ đối tượng phản hồi, chúng ta cần chuyển đổi nội dung phản hồi thô thành cấu trúc dữ liệu kiểu JSON. Điều này đạt được bằng cách sử dụng phương thức json (). Cuối cùng, chúng tôi trích xuất thông tin cần thiết bằng cách phân tích cú pháp đối tượng kiểu JSON.

Making a POST request

```
# importing the requests library
import requests
# defining the api-endpoint
API_ENDPOINT = "http://pastebin.com/api/api_post.php"
# your API key here
API_KEY = "XXXXXXXXXXXXXXXXX"

# your Nguồn code here
Nguồn_code = '''
print("Hello, world!")
a = 1
b = 2
print(a + b)
'''
# data to be sent to api
data = {'api_dev_key':API_KEY,
        'api_option':'paste',
        'api_paste_code':Nguồn_code,
        'api_paste_format':'python'}
# sending post request and saving response as response object
r = requests.post(url = API_ENDPOINT, data = data)
# extracting response text
pastebin_url = r.text
```

```
print("The pastebin URL is:%s"%pastebin_url)
```

4.1.2. Thiết lập phía máy chủ của một dịch vụ web để ghi lại và đánh giá các yêu cầu, Tạo chương trình giao tiếp phía máy khách sử dụng các yêu cầu HTTP

Yêu cầu

Trong nhiệm vụ này, vi điều khiển sẽ đọc vào các giá trị của cảm biến bạn chọn. Vi điều khiển phải đọc vào một phép đo sau mỗi 500ms. Dữ liệu cảm biến sẽ được gửi trực tiếp tới máy tính thông qua HTTP - Request. Máy tính phải có một phần mềm đo đang chạy để đọc vào các giá trị của cảm biến, chuyển đổi chúng thành các giá trị vật lý và in ra trên biểu đồ đường.

Các bước:

1. Tạo một chương trình đo lường trên vi điều khiển để đọc vào các giá trị từ một cảm biến sau mỗi 500ms
2. Cập nhật chương trình để gửi các giá trị đo được thông qua yêu cầu HTTP tới máy tính
3. Thiết lập chương trình máy chủ trên máy tính để đọc các giá trị được truyền qua HTTP - Request
4. Cập nhật chương trình máy tính để hiển thị các giá trị nhận được bằng một biểu đồ đường

Đáp án tham khảo:

Lời giải (Số chân phụ thuộc vào việc triển khai phần cứng)

Tiếp theo, ví dụ này cần có thêm thư viện http request (ví dụ: import requests)

Cũng cần phải chạy máy chủ http (ví dụ: sử dụng thư viện flask)

Máy khách (Client):

- aInPin = Nr1
- Trong hàm setup
  - Bắt đầu ADC
  - Đặt aInPin làm đầu vào
  - url = đặt URL dưới dạng chuỗi
- Trong vòng lặp chính
  - While true
    - adcVal = analogeRead (aInPin)
    - physicalVal = convertToPhysical(adcVal)
    - data = {‘SensVal’, physicalVal}
    - request.post(url, data)
    - Chờ 500ms

Máy chủ (Server)

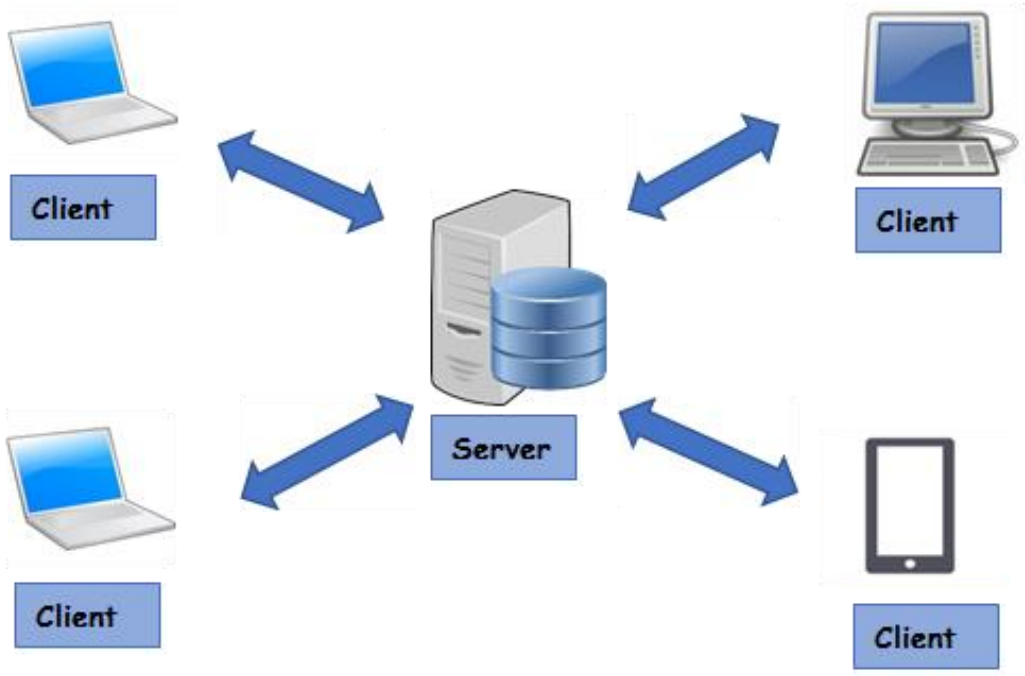
- rcvData = []
- Trong hàm callback trong khi nhận tín hiệu
  - rcvData.append (request.form ['SensVal'])
  - Vẽ biểu đồ rcvData mà không blocking

4.2. Giao tiếp Máy chủ Khách hàng (*Client Server Communication)*

4.2.1. Kiến trúc cơ bản về kiến trúc giao tiếp cơ sở của Máy chủ Khách hàng

Khi sử dụng máy tính mỗi chúng ta đều có nhu cầu kết nối thu thập và chia sẻ thông tin. Để đáp ứng nhu cầu đó mạng máy tính hay hệ thống mạng (computer network hay network system) ra đời.

Trên một hệ thống mạng, máy tính có thể đảm nhận một trong 3 vai trò sau:



Hình 48: Giao tiếp máy chủ khách hàng

- Máy tính đóng vai trò là máy chủ – Server: Là máy tính có khả năng cung cấp tài nguyên và các dịch vụ đến các máy trạm khác trong hệ thống mạng. Server đóng vai trò hỗ trợ cho các hoạt động trên máy trạm client diễn ra hiệu quả hơn.

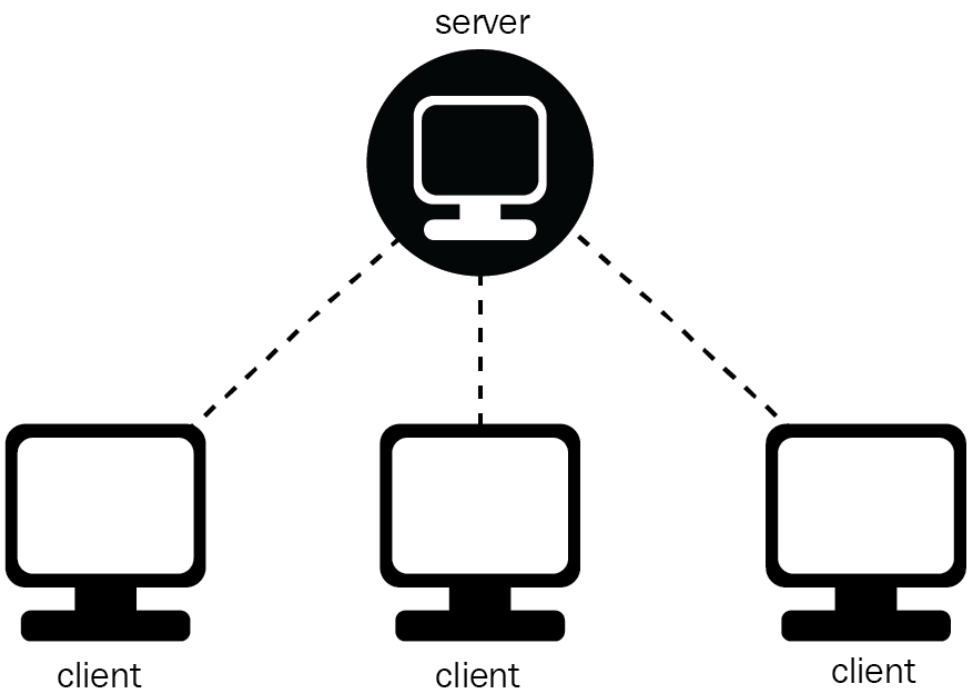
- Máy tính đóng vai trò là máy trạm – Client: Với vai trò là máy trạm, chúng sẽ không cung cấp tài nguyên đến các máy tính khác mà chỉ sử dụng tài nguyên được cung cấp từ máy chủ. Một client trong mô hình này có thể là một server cho mô hình khác, tùy thuộc vào nhu cầu sử dụng của người dùng.

- Máy tính đóng vai trò là Peer: Vừa sử dụng tài nguyên từ máy chủ cung cấp, đồng thời cũng cung cấp tài nguyên đến các máy tính khác trong mạng.

Vai trò máy tính cung cấp cho hệ thống mạng khác nhau nên đương nhiên sẽ có nhiều mô hình mạng máy tính khác nhau : Client Server, Peer-to-Peer và Hybrid. Trong đó mô hình mạng client server được sử dụng rộng rãi nhất

Mô hình Client Server là gì?

Mô hình mạng client server là mô hình mạng máy tính trong đó các máy tính con được đóng vai trò như một máy khách, chúng làm nhiệm vụ gửi yêu cầu đến các máy chủ. Để máy chủ xử lý yêu cầu và trả kết quả về cho máy khách đó.

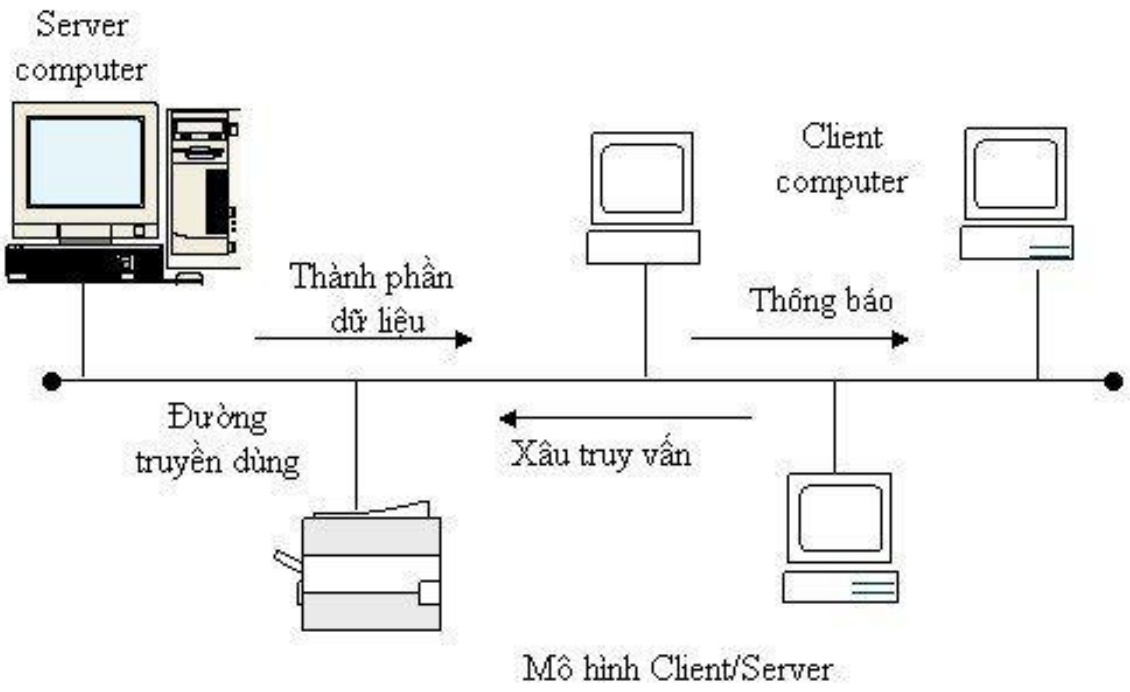


Hình 49: Mô hình Máy chủ Khách hàng

Nguyên tắc hoạt động của mô hình Client Server

Trong mô hình Client Server, server chấp nhận tấy cả các yêu cầu hợp lệ từ mọi nơi khác nhau trên mạng, sau đó trả kết quả về máy tính đã gửi yêu cầu.

Máy tính được coi là máy khách khi chúng làm nhiệm vụ gửi yêu cầu đến các máy chủ và đợi câu trả lời được gửi về.



Hình 50: Nguyên lý hoạt động của mô hình Client Server

Để máy khách và máy chủ có thể giao tiếp được với nhau thì giữa chúng phải có một chuẩn nhất định, và chuẩn đó được gọi là giao thức. Một số giao thức chuẩn được sử dụng rộng rãi hiện nay như TCP/IP, OSI, ISDN, X.25, Lan-to-Lan,.. Khi đó, nếu máy khách muốn lấy được thông tin từ máy chủ, chúng phải tuân theo một giao thức mà máy chủ đó đưa ra. Nếu yêu cầu đó được chấp nhận thì máy chủ sẽ thu thập thông tin và trả về kết quả cho máy khách yêu cầu. Bởi thông thường, server luôn trong trạng thái sẵn sàng nhận yêu cầu từ các client, nên chỉ cần client gửi tín hiệu và chấp nhận yêu cầu là server sẽ trả về kết quả trong thời gian ngắn nhất có thể.

Ưu nhược điểm của mô hình client server

ƯU điểm

- Mô hình client server giúp chúng ta có thể làm việc trên bất kì một máy tính nào có hỗ trợ giao thức truyền thông. Giao thức chuẩn này cũng giúp các nhà sản xuất tích hợp lên nhiều sản phẩm khác nhau mà không gặp phải khó khăn gì.
- Có thể có nhiều chương server cùng làm một dịch vụ, chúng có thể nằm trên nhiều máy tính hoặc một máy tính.
- Mô hình Client server chỉ mang đặc điểm của phần mềm mà không hề liên quan đến phần cứng, ngoài yêu cầu duy nhất là server phải có cấu hình cao hơn các client.

- Client server hỗ trợ người dùng nhiều dịch vụ đa dạng và sự tiện dụng bởi khả năng truy cập từ xa mà các mô hình cũ không có được.
- Mô hình mạng khách chủ cung cấp một nền tảng lý tưởng, cho phép cung cấp tích hợp các kỹ thuật hiện đại như mô hình thiết kế hướng đối tượng, hệ chuyên gia, hệ thông tin địa lý (GIS).

Nhược điểm của mô hình client- server:

Do phải trao đổi dữ liệu giữa 2 máy tính khác nhau ở 2 khu vực địa lý cách xa nhau, nên vấn đề bảo mật dữ liệu thông tin đôi khi còn chưa được an toàn lắm. Đây là nhược điểm duy nhất của mô hình này.

4.2.2. Phía máy chủ tạo một chương trình giao tiếp để thu thập dữ liệu và sao lưu dữ liệu bằng cách sử dụng web socket, Phía khách hàng tạo chương trình đo lường và truyền thông, để thiết lập kết nối và truyền dữ liệu

Websockets - Máy chủ trực quan hóa

Yêu cầu:

Trong nhiệm vụ này, vi điều khiển hoạt động như một máy khách và gửi dữ liệu đo của nó theo định kỳ tới máy tính đóng vai trò là máy chủ. Vi điều khiển sẽ gửi một phép đo cảm biến sau mỗi 200ms đến máy chủ thông qua giao tiếp WebSocket. Máy tính sẽ hoạt động như một máy chủ, đối với mỗi phép đo gửi đến, máy tính sẽ nhận dữ liệu đo và in ra để hiển thị nó trên hình ảnh trực quan.

Các bước:

1. Chuẩn bị chương trình giao tiếp với máy khách trên vi điều khiển
2. Tạo một chương trình đo lường dựa trên vi điều khiển
3. Cập nhật chương trình để gửi dữ liệu qua websocket
4. Tạo giao tiếp webserver phía máy chủ
5. Tạo hình ảnh trực quan dựa trên dữ liệu cảm biến nhận được

Đáp án tham khảo:

Lời giải (Số chân phụ thuộc vào việc triển khai phần cứng)

Tiếp theo, ví dụ này cần có thêm thư viện websocket (ví dụ: nhập vào socket)

Máy khách (Client):

- aInPin = Nr1
- Trong hàm setup
  - Bắt đầu ADC
  - Đặt aInPin làm đầu vào
  - host = địa chỉ máy chủ dưới dạng chuỗi
  - port = cổng được sử dụng dạng số nguyên (int)
- Trong vòng lặp chính
  - Socket.connect(host,port)

- While true
  - adcVal = analogeRead (aInPin)
  - physicalVal = convertToPhysical(adcVal)
  - data = chuyển đổi thành physicalVal dạng chuỗi
  - data = mã hóa thành dữ liệu byte
  - socket.sendall(data)
  - Chờ 200ms

Máy chủ (Server)

- rcvData = []
- host = địa chỉ máy chủ dưới dạng chuỗi
- port = cổng được sử dụng dạng số nguyên (int)
- socket.bind(host,port)
- socket.listen()
- conn, addr = socket.accept()
- while true
  - data = conn.recv(1024)
  - if data:
    - data = float(data.decode('utf-8))
    - rcvData.append(data)
    - Vẽ biểu đồ rcvData mà không blocking

Websockets - Máy chủ đo lường

Yêu cầu:

Yêu cầu trong nhiệm vụ này là thiết lập hệ thống giao tiếp websocket dựa trên máy chủ-máy khách. Vi điều khiển phải được cấu hình như một máy chủ đo lường. Máy tính sẽ hoạt động như một máy khách. Bất cứ khi nào máy khách gửi yêu cầu đo đến máy chủ, máy chủ sẽ đo một giá trị từ cảm biến. Giá trị này sẽ được gửi lại cho máy khách. Sau đó, máy khách sẽ lưu giá trị trong file csv. Tiếp theo, trên máy khách sẽ có một chương trình đồ họa người dùng được triển khai. Để người dùng có thể yêu cầu đo bằng cách nhấn nút.

Các bước:

1. Tạo một hàm đo lường trên vi điều khiển có thể đọc vào các giá trị cảm biến, hàm này sẽ được gọi là "readInSensor()" và nó sẽ trả về giá trị cảm biến
2. Tạo chức năng máy chủ trên vi điều khiển
   a. Khi nhận được yêu cầu của máy khách, máy chủ sẽ gọi hàm "readInSensor ()"
   b. Giá trị đọc được sẽ được gửi qua giao tiếp socket trở lại máy khách đã gửi yêu cầu
3. Tạo chức năng websocket trên máy khách
4. Tạo giao diện đồ họa người dùng trên máy tính được dùng làm máy khách
   a. Một nút sẽ được sử dụng để tạo yêu cầu từ máy khách để máy chủ trả về giá trị cảm biến

5. Cập nhật chương trình trên máy khách để lưu các giá trị nhận được trong một file csv.

Đáp án tham khảo:

Lời giải (Số chân phụ thuộc vào việc triển khai phần cứng)

Tiếp theo, ví dụ này cần có thêm thư viện websocket (ví dụ: import socket)

Ngoài ra, cần phải có thư viện để tạo giao diện đồ họa người dùng (ví dụ: tkinter) và thư viện để ghi dữ liệu vào tệp .csv (pandas)

Máy khách (Client):

- host = địa chỉ máy chủ dưới dạng chuỗi
- port = cổng được sử dụng dạng số nguyên (int)
- mở cửa sổ
- tạo đối tượng nút nhấn trên cửa sổ với hàm callback
- trong hàm callback của nút nhấn:
    - socket.connect(host,port)
    - socket.sendall("newData")
    - data = s.recv(1024)
    - data = data.decode('utf-8)
    - writeToCSV(data)

Máy chủ (Server)

- aInPin = Nr1
- host = địa chỉ máy chủ dưới dạng chuỗi
- port = cổng được sử dụng dạng số nguyên (int)
- Trong hàm setup
    - Bắt đầu ADC
    - Đặt aInPin làm đầu vào
    - host = địa chỉ máy chủ dưới dạng chuỗi
    - port = cổng được sử dụng dạng số nguyên (int)
    - socket.bind(host,port)
- while true
    -
    - socket.listen()
    - conn, addr = socket.accept()
    - data = conn.recv(1024)
    - adcVal = analogeRead (aInPin)
    - physicalVal = convertToPhysical(adcVal)
    - data = chuyển đổi physicalVal thành dạng chuỗi
    - data = mã hóa thành dữ liệu byte
    - conn.sendall(data)

Websockets - Bảng điều khiển (Control panel)

Yêu cầu:

Trong nhiệm vụ này, vi điều khiển sẽ được sử dụng làm máy chủ và được điều khiển thông qua giao diện đồ họa người dùng được tạo trên máy tính hoạt động như một máy khách. Giao diện đồ họa người dùng sẽ có hai nút - một để tăng độ sáng của đèn LED và một để giảm độ sáng của đèn LED. Bản thân đèn LED sẽ được điều khiển trên vi điều khiển thông qua PWM. Ngoài đèn LED, cũng có thể điều khiển tấm chắn động cơ và động cơ một chiều nếu có. Giao tiếp sẽ được thực hiện thông qua Websockets. Mỗi khi các nút trên GUI trên máy tính được nhấn, một lệnh sẽ được gửi tới vi điều khiển. Vi điều khiển sẽ nhận lệnh và tăng hoặc giảm độ sáng của đèn LED

Các bước:

1. Tạo phần mềm giao tiếp trên máy khách
2. Tạo phần mềm giao tiếp trên máy chủ
3. Cập nhật phần mềm máy khách để làm mờ đèn LED
    a. Tạo một hàm "dimLED(value)" để nhận giá trị làm mờ cho đèn LED và đặt PWM theo giá trị. Không cần giá trị trả lại
    b. Hãy xem xét để đảm bảo rằng không có giá trị nhập vào nào vượt quá giá trị tối đa của PWM
4. Tạo GUI phía máy chủ với hai nút để gửi lệnh tăng hoặc giảm giá trị PWM

Đáp án tham khảo:

Lời giải (Số chân phụ thuộc vào việc triển khai phần cứng)

Máy khách (Client):

- host = địa chỉ máy chủ dưới dạng chuỗi
- port = cổng được sử dụng dạng số nguyên (int)
- dc = input()
- dc.encode()
- socket.connect(host,port)
- socket.sendall(dc)

Máy chủ (Server)

- LEDPin = Nr1
- host = địa chỉ máy chủ dưới dạng chuỗi
- port = cổng được sử dụng dạng số nguyên (int)
- Trong hàm setup
    - Bắt đầu ADC
    - ledPWM = đặt LEDPin làm đầu ra
    - Bắt đầu PWM với 0% chu kỳ nhiệm vụ
    - host = địa chỉ máy chủ dưới dạng chuỗi
    - port = cổng được sử dụng dạng số nguyên (int)
    - socket.bind(host,port)
- while true
    - socket.listen()
    - conn, addr = socket.accept()

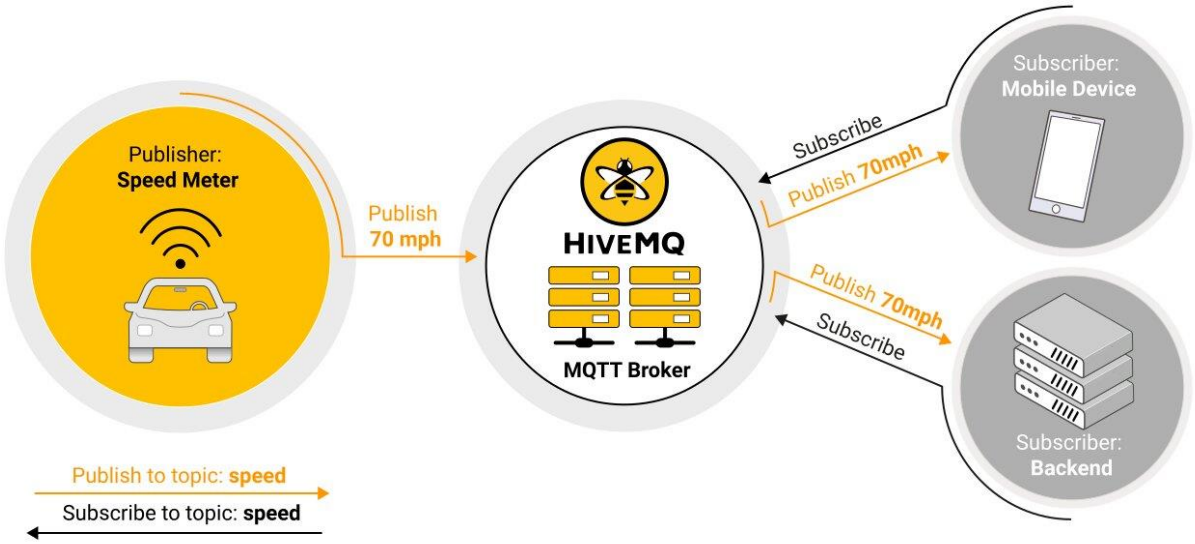
- data = conn.recv(1024)
- data = data.decode("utf-8")
- data = float(data)

đặt chu kỳ nhiệm vụ ledPWM thành dữ liệu

4.3. Giao tiếp publisher subscriber (publisher subscriber commuinication)

4.3.1. Kiến trúc cơ bản về Giao tiếp publisher subscriber

Mẫu publish/subscribe (còn được gọi là pub/sub) cung cấp một thay thế cho kiến trúc máy khách-máy chủ truyền thống. Trong mô hình máy khách-máy khách, máy khách liên lạc trực tiếp với điểm cuối. Mô hình pub / sub tách riêng máy khách gửi tin nhắn (nhà xuất bản-publish) từ máy khách hoặc máy khách nhận tin nhắn (người đăng ký-subscribe) . Các nhà xuất bản và đăng ký không bao giờ liên lạc trực tiếp với nhau. Trong thực tế, họ thậm chí không nhận thức được rằng cái kia tồn tại. Kết nối giữa chúng được xử lý bởi một thành phần thứ ba (người môi giới-broker). Công việc của nhà môi giới là lọc tất cả các tin nhắn đến và phân phối chúng một cách chính xác cho người đăng ký. Vì vậy, hãy đi sâu hơn một chút vào một số khía cạnh chung của pub/sub



Hình 51: Mẫu xuất bản / đăng ký

Khía cạnh quan trọng nhất của pub / sub là sự tách rời của nhà xuất bản tin nhắn từ người nhận (người đăng ký). Sự tách rời này có một số chiều:

- Phân tách không gian: Nhà xuất bản và người đăng ký không cần biết nhau (ví dụ: không trao đổi địa chỉ IP và cổng).
- Thời gian tách rời: Nhà xuất bản và người đăng ký không cần phải chạy cùng một lúc.
- Phân tách đồng bộ hóa: Hoạt động trên cả hai thành phần không cần phải bị gián đoạn trong quá trình xuất bản hoặc nhận.

Tóm lại, mô hình pub / sub loại bỏ giao tiếp trực tiếp giữa nhà xuất bản tin nhắn và người nhận / người đăng ký. Hoạt động lọc tin nhắn của nhà môi giới cho phép kiểm soát khách hàng / thuê bao nào nhận được tin nhắn nào. Việc tách rời có ba chiều: không gian, thời gian và đồng bộ hóa.

4.4. Thực hiện chương trình giao tiếp và trao đổi dữ liệu của hai bộ vi điều khiển với các khối chức năng được cung cấp

Giao tiếp đơn giản

Yêu cầu:

Trong nhiệm vụ này, giao thức giao tiếp MQTT sẽ được sử dụng để truyền dữ liệu từ publisher đến subcriber thông qua broker. Vi điều khiển sẽ là publisher và đọc vào nhiệt độ thông qua cảm biến nhiệt độ. Publisher sẽ xuất bản (publish) một chủ đề có tên là “TempSens1”. Publisher sẽ đọc vào các giá trị nhiệt độ mỗi giây và publish chúng về chủ đề được đặt tên. Máy tính sẽ hoạt động như một subcriber. Subcriber sẽ đọc vào các giá trị từ chủ đề “TempSens1” và in ra trên màn hình console. Tiếp theo, máy tính cũng sẽ lưu trữ broker được yêu cầu.

Các bước:

1. Thiết lập broker trên máy tính và chạy broker
2. Tạo một chương trình đo trên vi điều khiển để đọc vào phép đo cảm biến nhiệt độ và kiểm thử nó
3. Thêm các thư viện cần thiết trên vi điều khiển và tạo chủ đề để publish
4. Thêm các thư viện cần thiết trên máy tính và tạo đăng ký (subcribe) với chủ đề nơi các giá trị cảm biến được xuất bản
5. Cập nhật chương trình để đọc các giá trị đã publish và in chúng ra trên console

Đáp án tham khảo:

Lời giải (Số chân phụ thuộc vào việc triển khai phần cứng)

Tiếp theo, cần cài đặt thư viện MQTT

Broker:

Khởi động broker

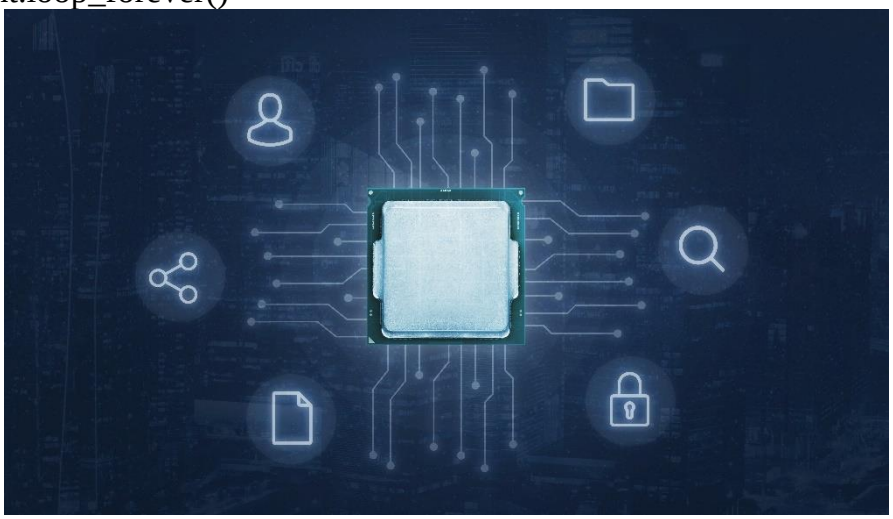
Publisher:

- aInPin = Nr1
- client = mqtt.Client()
- Trong hàm setup
  - Bắt đầu ADC
  - Đặt aInPin làm đầu vào
  - host = địa chỉ máy chủ dưới dạng chuỗi
  - port = cổng được sử dụng dạng số nguyên (int)
- Trong vòng lặp chính
  - Kết nối với máy khách bằng địa chỉ máy chủ và cổng

- Client.loop_start()
- While true
  - adcVal = analogeRead (aInPin)
  - physicalVal = convertToPhysical(adcVal)
  - client.publish("/TempSens1", physicalVal)

Subscriber:

- hàm on_connect:
  - client.subscribe("/TempSens1")
- hàm on_message:
  - print(str(msg.payload))
- client = mqtt.Client()
- client.on_connect = on_connect
- client.on_message = on_message
- serverAdr = địa chỉ máy chủ dưới dạng chuỗi
- port = cổng được sử dụng dạng số nguyên (int)
- kết nối với máy chủ bằng serverAdr và cổng
- client.loop_forever()



Hình minh họa 5: Các mạch chiếu sáng bộ xử lý CPU

Làm việc nhóm với nhiều nút cảm biến

Yêu cầu:

Trong nhiệm vụ này, giao thức giao tiếp MQTT sẽ được sử dụng để gửi dữ liệu cảm biến với ít nhất hai vi điều khiển khác nhau với vai trò là các nút cảm biến. Đây là bài tập nhóm với ít nhất 2 người. Mỗi người tham gia sẽ sử dụng một nút và hoặc máy tính. Những người tham gia phải trao đổi và phân chia nhiệm vụ trong nhóm. Các loại cảm biến đã sử dụng và tên chủ đề cho mỗi nút có thể được chọn như mong muốn nhưng không được giống nhau. Subcriber một lần nữa sẽ là một máy tính. Subcriber phải đăng ký cả hai chủ đề, đọc vào các giá trị của cả hai chủ đề và in ra các giá trị nhận được và tên chủ đề trên màn hình console.

Các bước:

1. Quyết định xem ai sẽ dùng nút / máy tính nào
2. Quyết định xem cảm biến nào sẽ được sử dụng và đặt tên cho chủ đề là gì
3. Triển khai logic của mỗi nút và của subcriber
    a. Đọc vào giá trị cảm biến
    b. Xuất bản theo chủ đề
    c. Đăng ký chủ đề
    d. In ra màn hình console

Gợi ý: Ví dụ này có thể được giải quyết chính xác như phần 4.3.1 chỉ với việc thêm một số publisher và topic

Dự án Vi điều khiển: Thu thập dữ liệu trong thiết lập I4.0

1. Giới thiệu

Tất cả các ứng dụng I4.0 đều dựa vào một sản phẩm cơ sở mà chính là dữ liệu. Dữ liệu sản xuất có giá trị nhất đối với các công ty để tối ưu hóa và đo lường chất lượng. Tuy nhiên, các công ty trên khắp thế giới vẫn còn tồn tại các nhà máy sản xuất cổ điển không có hoặc chỉ có khả năng thu thập dữ liệu độc quyền. Một trong những thách thức lớn trong tương lai là xác định và đo lường dữ liệu sản xuất và môi trường và lưu trữ chúng. Đối với các công ty, để tạo ra "dữ liệu" sản phẩm mới và nâng sản phẩm của họ lên tầm I4.0, bắt buộc phải tạo ra nhận thức và kỹ năng đo lường các giá trị vật lý quan trọng và gửi chúng qua mạng đến các trạm lưu trữ. Trong dự án này, một kịch bản thu thập dữ liệu quy mô đầy đủ sẽ được triển khai.

Yêu cầu:

- Có kinh nghiệm lập trình hướng đối tượng
- UML
    - **Sơ đồ Use case**
    - **Sơ đồ class**
    - Sơ đồ trình tự
- Lưu đồ
- Kiểm thử phần mềm
- Đã hoàn thành phần lý thuyết và thực hành của khóa học LC2

1.1 Thông tin thêm - dành cho giáo viên

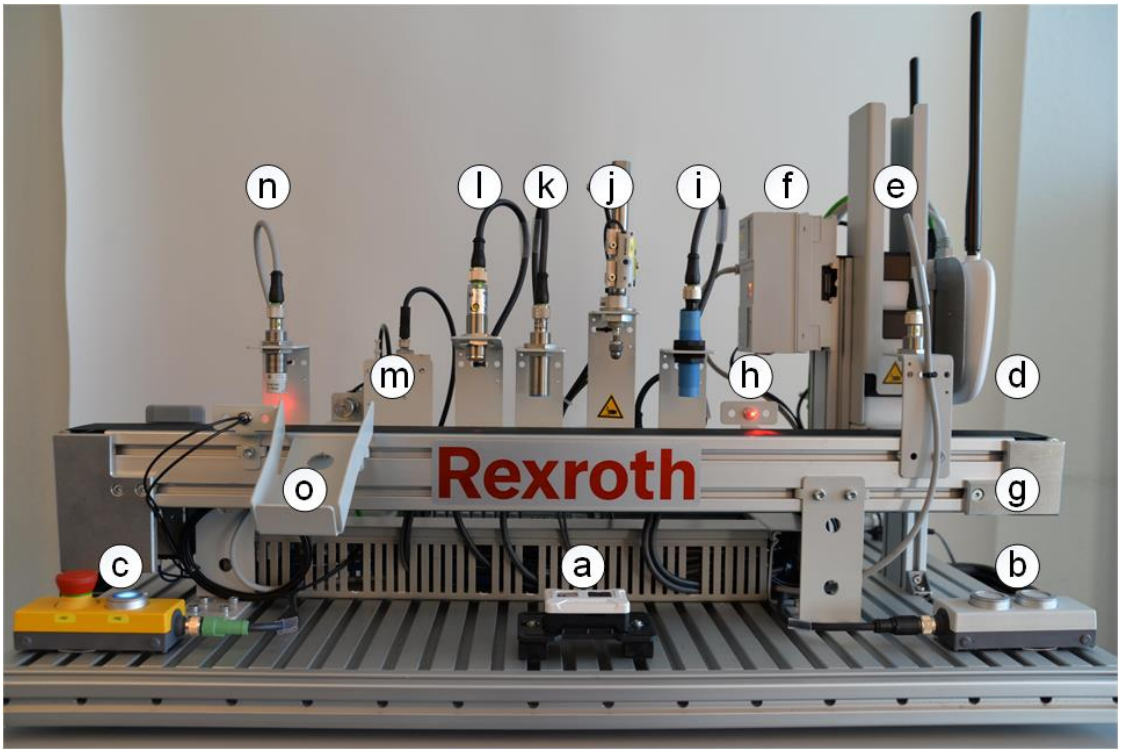
Nếu các trạm quá trình khác được sử dụng thì cần điều chỉnh mô tả phù hợp với hệ thống hiện có của bạn Dự án này được viết theo cách chung chung độc lập về phần cứng về các điều kiện hoặc các cảm biến được sử dụng. Một số bài tập sẽ đưa ra các nhiệm vụ để xác định các thông số cảm biến từ bảng dữ liệu. Chúng cần được điều chỉnh cho phù hợp với phần cứng được sử dụng. Hơn nữa, chỉ sử dụng các cảm biến khi bạn cũng có thể cung cấp một bảng dữ liệu.

2. Thiết lập cơ bản

Có hai trạm sản xuất cổ điển. Cả hai trạm sản xuất đều có các chức năng và hoạt động mà không cần các cách thu thập dữ liệu hiện đại. Ngày nay, dữ liệu sản xuất có giá trị lớn đối với mọi công ty và đều có tiềm năng tối ưu hóa. Nhiệm vụ của bạn là nâng cấp các trạm sản xuất cổ điển sang thiết lập I4.0. Vì việc lựa chọn các cảm biến phù hợp, tạo các chương trình đo và gửi dữ liệu qua mạng máy tính đến các trạm máy chủ là cơ sở của I4.0, trọng tâm chính của dự án này sẽ là thiết lập một khung cơ sở chức năng. Trong dự án này, toàn bộ quy trình công việc từ xác định hệ thống và yêu cầu cho đến triển khai và kiểm thử thực tế sẽ được thực hiện. Thiết lập Cơ bản sẽ giúp mở rộng một trạm quá trình hiện có bằng cách đo dữ liệu nhiệt độ và gia tốc. Trong các phần sau, các trạm quá trình sẽ được giải thích.

2.1 CPSi 4.0

Trạm quá trình CPSi được hiển thị bên dưới. Trạm quá trình có thể được điều khiển thông qua vi điều khiển và các nút nhấn hệ thống. Ví dụ này chỉ xem xét điều khiển bằng các nút nhấn hệ thống. Khi nhấn một nút hệ thống, trạm quá trình sẽ bắt đầu bằng cách đẩy một khối lập phương ra khỏi khay (cube magazine). Khối lập phương này được thử nghiệm ở một số trạm. Sau trạm cuối cùng, khối lập phương được đẩy ra bằng xi lanh khí nén hoặc đi qua đó.



Hình 52: Hệ thống đào tạo CPS i4.0 (nhìn từ phía trước)

a	Vi điều khiển (XDK)	Tôi	Cảm biến điện dung

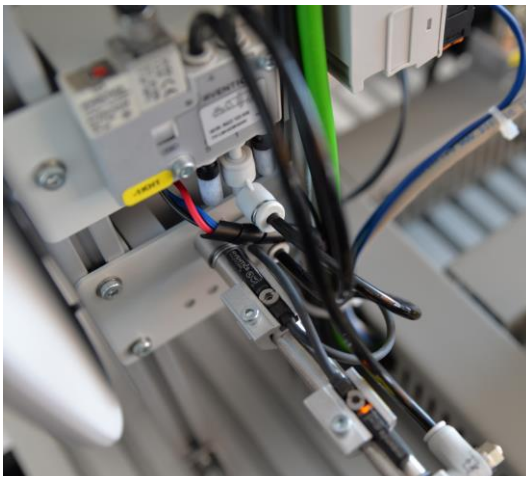
b	Các nút hệ thống	j	Cảm biến vị trí (trên xi lanh viền)
c	Nút dừng khẩn cấp	k	Cảm biến cảm ứng
d	Bộ định tuyến WLAN (Router)	l	Cản sáng
e	Khay	m	Xi lanh khí nén và van điều hướng 5/2
f	XM22 (PLC)	n	Ăng ten RFID
g	Băng chuyền	o	Dốc nhận (collecting ramp)
h	Dây dẫn ánh sáng		

Mô tả hệ thống

Trước khi thử nghiệm, các khối lập phương được lưu trữ trong một khay (hình 8 - 0). Khay được nạp đầy thủ công với các khối lập phương mới. Các nút có thể được sử dụng để chọn loại khối lập phương và bắt đầu hoạt động. Tại độ cao của khối lập phương thứ hai trong ổ, có một công tắc cơ học. Công tắc khởi động khi không có khối lập phương nào ấn vào nó (NC - thường đóng).
Với điều kiện:

- có ít nhất 2 khối lập phương trong khay
- nút dừng khẩn cấp được nhả,
- dây dẫn ánh sáng không bị cản trở,
- băng chuyền đứng yên,
- tất cả các xi lanh ở vị trí ban đầu,

Hình 53: Các khối lập phương ở dưới ăng-ten RFID lúc bắt đầu



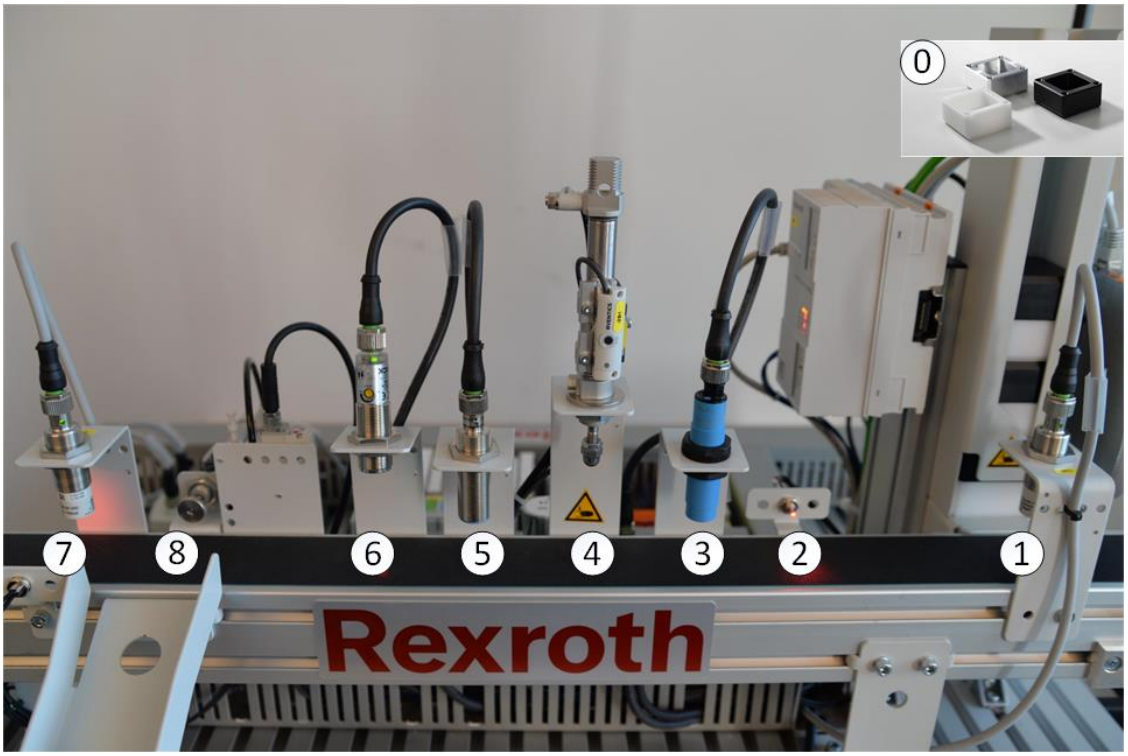
Hình 54: Xi lanh khay với hai công tắc lưỡi gà

Van nằm ở mặt sau của khay cùng với một xi lanh khí nén. Các xi lanh được vận hành bằng khí nén với áp suất ít nhất 1 bar. Khí nén được phân phối đến các van bằng hệ thống phân phối khí nén (hình 2 - f). Tùy thuộc vào vị trí van mà các van cấp khí nén vào các buồng xi lanh. Việc cung cấp khí nén cho các buồng xi lanh có thể được điều chỉnh thông qua van tiết lưu tại các điểm đầu vào. Khi piston được kéo, nó sẽ đẩy một nửa khối lập phương lên băng chuyền.

Vị trí của các xi lanh được phát hiện thông

qua các công tắc lưỡi gà. Các công tắc kích hoạt bằng vật liệu từ tính. Có hai công tắc lưỡi gà trên xylanh khay giúp phát hiện vị trí ban đầu và vị trí hoạt động của piston. Một động cơ DC (hình 2 - g) được gắn trên mặt bích ở đầu kia của băng chuyền. Động cơ dẫn động băng chuyền. Chiều quay được xác định bởi hai rơle. Trước khi băng chuyền bắt đầu dịch chuyển, xi lanh khay trở lại vị trí cơ bản và ăng ten RFID phía trên nửa khối lập phương ghi mã số loại khối lập phương trên thẻ RFID trong khối.

Các khối lập phương chuyển động theo hướng từ phải sang trái. Chúng đi qua một số trạm trên băng chuyền. Tùy thuộc vào quá trình tại mỗi trạm, băng chuyền dừng lại hoặc phôi được đánh giá khi nó đi qua.



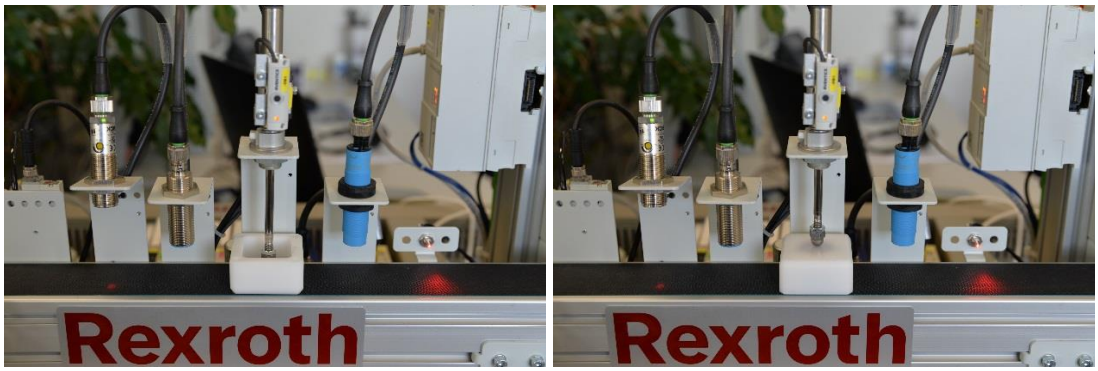
Hình 55: Vị trí phôi trong CPSi4.0

Các trạm (xem hình 8):

1. **Ăng ten RFID:** Mã số được viết trên thẻ trong khối.
2. **Dây dẫn ánh sáng:** Phát hiện vị trí khối (cạnh)
3. **Cảm biến điện dung:** Phát hiện vị trí khối (trên)
4. **Xi lanh viền:** Piston di chuyển trên phôi và cảm biến vị trí đo khoảng cách được bao phủ - suy luận về hướng (đáy/ nắp)
5. **Cảm biến cảm ứng:** Phát hiện vật liệu kim loại - suy luận về vật liệu
6. **Cản sáng:** Phát hiện các đối tượng sáng - suy luận về màu sắc
7. **Ăng ten RFID:** Mã số được đọc từ thẻ
8. **Xi lanh đẩy:** Đẩy các khối sai ra ngoài

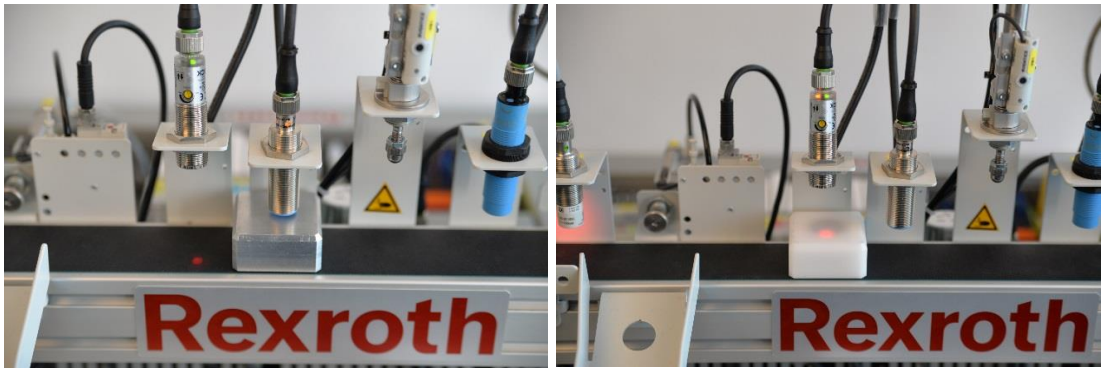
Hình 56: Khối lập phương được phát hiện bởi cảm biến điện dung

Đầu tiên phôi đi qua dây dẫn ánh sáng và sau đó là cảm biến điện dung. Phôi được phát hiện theo cạnh bên hoặc từ trên xuống. Thông qua phản hồi, chương trình biết thời điểm và vị trí của khối lập phương ở trên băng chuyền. Sau khi các cảm biến được kích hoạt, băng chuyền dừng lại để phôi được định vị dưới xi lanh viền. Việc phát hiện dư thừa đảm bảo rằng một nửa khối lập phương được định vị chính xác.



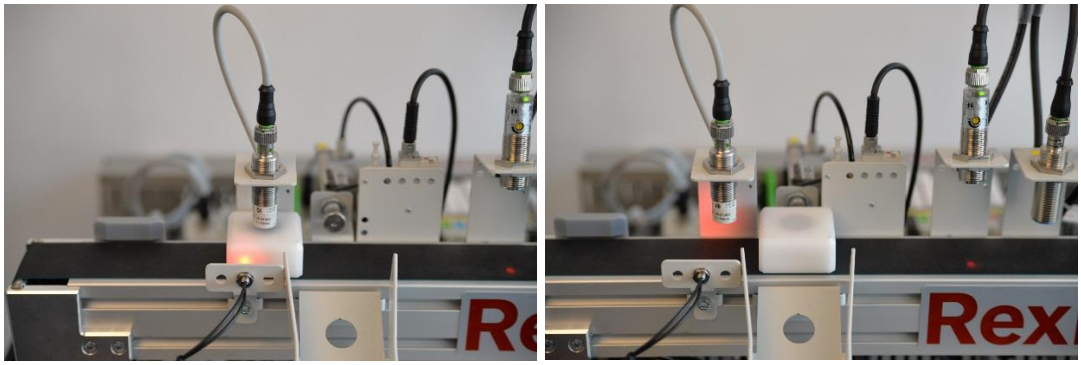
Hình 57: Xi lanh kiểm tra đường viền với cảm biến vị trí cho phần đáy (bên trái) và phần nắp (bên phải)

Nếu băng chuyền dừng ở đây, piston xi lanh sẽ giãn ra. Xi lanh này được điều khiển bởi một van khí nén để điều khiển vòng kín (hình 2 - h). Với phần đáy, xi lanh giãn hoàn toàn. Một công tắc lưỡi gà sẽ phát hiện vị trí này. Với phần nắp, piston ép lên khối lập phương. Quá trình này bị giới hạn về thời gian. Sau khi hết thời gian, piston sẽ rút lại. Khoảng cách được bao phủ được ghi lại bởi một cảm biến vị trí. Cảm biến này cũng được gắn trên vỏ xi lanh. Hướng của phôi được suy ra từ giá trị trả về.



Hình 58: Cảm biến cảm ứng với khối nhôm (trái) và tấm cản sáng với khối nhựa (phải)

Nếu piston ở vị trí ban đầu, băng chuyền tiếp tục chuyển động. Phôi đi qua một cảm biến cảm ứng và một tấm cản sáng. Cảm biến cảm ứng chỉ phản ứng với các vật thể kim loại. Tấm cản sáng phát hiện các vật thể sáng (trắng hoặc bạc). Một đèn LED chiếu sáng ở đầu các cảm biến cho biết rằng các cảm biến đã phát hiện ra điều gì đó. Băng chuyền không phải dừng ở đây. Thông tin về hướng, chất liệu và màu sắc được lưu lại.



Hình 59: Dây dẫn ánh sáng dừng hình khối dưới ăng-ten RFID (trái) và hình khối sai ở vị trí đẩy ra (phải)

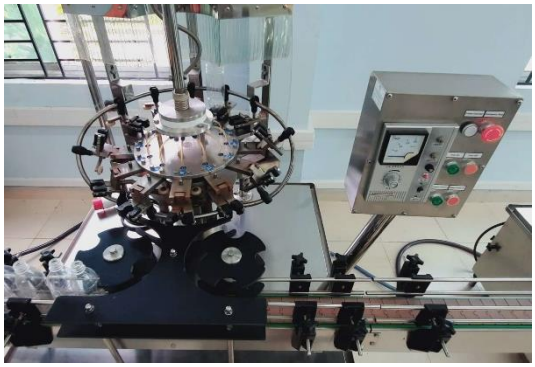
Phôi dừng ở dây dẫn ánh sáng thứ hai. Dây dẫn này được đặt đối diện với ăng-ten RFID thứ hai. Mã số trên thẻ (loại hình khối) được so sánh với các thuộc tính đã lưu. Nếu khớp, nửa khối lập phương di chuyển đến dấu màu xám (cuối băng chuyền). Trong trường hợp xảy ra lỗi, băng chuyền sẽ quay ngược trở lại xi lanh đẩy và khối lập phương được đẩy xuống dốc. Nếu có đủ khối trong khay, quá trình này sẽ tự động được lặp lại.

2.2 Trạm đóng chai

Trạm đóng chai là một trạm cơ khí thuần túy. Từ trước đến nay không có phương tiện thu thập dữ liệu nào. Trạm đóng chai lấy chai, làm sạch chúng tại trạm làm sạch, đổ đầy tại trạm chiết rót và cuối cùng đặt nắp vào chai tại trạm đóng nắp.

Hình 60: Thiết lập đầy đủ trạm đóng chai, làm sạch (trái), chiết rót (giữa) và đóng nắp (phải)

Sau khi quá trình đóng chai kết thúc, chai mới cần được đưa vào vị trí đầu vào để khởi động lại quy trình.



Hình 61: Trạm làm sạch

Trạm làm sạch là quá trình đầu tiên một chai sẽ tham gia vào trong toàn bộ trạm đóng chai. Các chai ở đúng vị trí vào trong trạm sẽ được kẹp chặt đầu một cách cơ học, di chuyển lên trên và làm sạch. Toàn bộ quá trình làm sạch sẽ được tiến hành trong vòng một vòng quay của trạm làm sạch. Sau khi làm sạch xong, chai được đẩy ra ngoài. Không có kiểm tra để xem các chai có được đặt chính xác ở vị trí đầu vào cũng như ở vị trí đầu ra hay không. Do đó, không thể phát hiện các chai bị lật ngược. Điều đó có nghĩa là tất cả các chai phải được đặt chuẩn trên băng chuyền trước khi quá trình bắt đầu

Hình 62: Trạm chiết rót

Trạm chiết rót sẽ lấy chai một cách cơ học và gắn vào một vị trí đã xác định trước. Nếu một chai được gắn vào, van chiết rót sẽ được mở ra và chất lỏng sẽ được đổ đầy vào bên trong chai. Quá trình chiết rót sẽ tự động được bắt đầu nếu một chai ở vị trí đầu vào và quá trình quay được bắt đầu. Sau khi quá trình quay kết thúc và chai ở vị trí đầu ra của trạm, van sẽ tự động được đóng kín lại. Tại vị trí đầu vào, giả định rằng các chai đã vào tuần tự và được đặt chuẩn. Không có hệ thống kiểm tra hoặc đẩy ra nếu một chai bị rơi.

Hình 63: Trạm đóng nắp

Sau khi các chai đã được làm sạch và đổ đầy nước, chúng sẽ được chuyển đến trạm cuối cùng, trạm đóng nắp. Tại đây các chai một lần nữa sẽ được chọn tuần tự. Trong một quá trình, chúng sẽ được xoay và một nắp sẽ được gắn ở đầu chai. Sau đó, các chai sẽ được chuyển xuống vị trí đầu ra của trạm. Không có kiểm tra xem nắp đã được đặt đúng trên đầu chai hay chưa. Cũng không có bất kỳ kiểm tra nào để xem các chai đến đúng vị trí ở vị trí đầu vào hay không. Ngoài ra, sau khi các chai đã được đóng nắp, không có kiểm tra xem các chai có được đặt đúng vị trí ở vị trí đầura hay không. Ngoài ra, các nắp cần được nạp thủ công vào chỗ chứa nắp

Tất cả ba trạm đều có khả năng gắn các cảm biến và đo các giá trị môi trường xung quanh.

3. Lựa chọn phần cứng - vi điều khiển chung

Để đo và phân tích giá trị vật lý một cách chính xác, cần phải có các cảm biến. Trạm quá trình hiện có sẽ được nâng cấp bằng hai loại cảm biến. Một là cảm biến nhiệt độ có điện trở nhiệt. Cái còn lại là gia tốc kế, cái còn lại là cảm biến nhiệt độ. Cảm biến nhiệt độ phải theo dõi nhiệt độ xung quanh phòng. Các cảm biến này thường được sử dụng ở các vị trí khác nhau trong dây chuyền sản xuất để theo dõi ảnh hưởng của nhiệt độ đối với các quá trình. Gia tốc kế được sử dụng để theo dõi gia tốc của các thành phần. Hơn nữa, nó cũng có thể được sử dụng để theo dõi các rung động và thậm chí được sử dụng để phát hiện các lỗi nhất định như mất thăng bằng.

3.1 Xác định các thông số cảm biến

Trong bước đầu tiên, hãy xác định cảm biến nhiệt độ và gia tốc kế được cung cấp trong bảng dữ liệu và viết loại của nó ra:

Cảm biến nhiệt độ	Gia tốc kế

Cảm biến loại điện trở nhiệt có một số thông số quan trọng, tra cứu trong bảng dữ liệu và điền vào các thông số còn thiếu trong bảng. Các bảng dữ liệu thường chứa một số danh mục khác nhau của cùng một cảm biến. Đảm bảo chỉ ghi các thông số của cảm biến của bạn:

Tham số	Giá trị

R25	
R/R25	
α	

Tìm kiếm các thông số trong bảng dữ liệu của gia tốc kế và liệt kê chúng trong bảng bên dưới:

Tham số	Giá trị
Dải quy mô cao nhất	
Hệ số quy mô nhỏ nhất	
Hệ số quy mô lớn nhất	

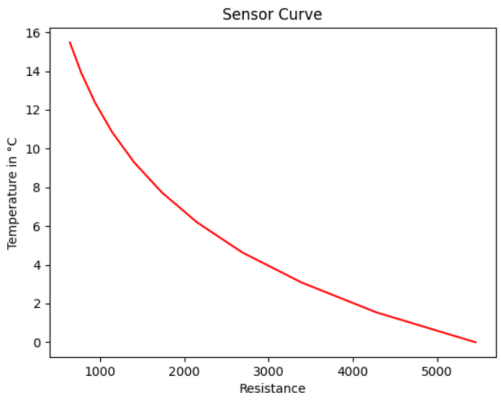
3.2 Tính toán giá trị vật lý của cảm biến

Điện trở nhiệt được cung cấp là loại NTC. Do đó, điện trở của điện trở nhiệt sẽ giảm khi nhiệt độ giảm. Nội suy tuyến tính là một phương pháp để tính toán nhiệt độ của điện trở nhiệt dựa trên điện trở đo được. Để tạo một phép nội suy tuyến tính, có thể sử dụng hàm np.interp. Hàm này sẽ được giải thích bên dưới.

3.2.1 Trực quan hóa đường cong

Trong bước đầu tiên, hãy tạo một chương trình để trực quan hóa đường cong điện trở dựa trên bảng dữ liệu trong phạm vi từ 0°C đến 50°C. Vì chúng ta muốn tính toán nhiệt độ dựa trên điện trở, hãy đảm bảo sử dụng nhiệt độ trên trục y và điện trở trên trục x. Bảng dưới đây cung cấp một đồ thị mẫu với các giá trị khác nhau. Chèn hình ảnh đồ thị của bạn trong cột “Your solution”. Đảm bảo rằng giải pháp của riêng bạn có màu sắc, nhãn và tiêu đề giống như đồ thị mẫu. Tiếp tục tải lên toàn bộ hình ảnh và mã trên nhiệm vụ được cung cấp.

Sample plot	Your solution



Hình 64: Trực quan hóa đường cong

3.2.2 Nội suy tuyến tính giá trị cảm biến

Một hàm nội suy tuyến tính thực hiện một nội suy tuyến tính giữa các điểm đã cho (nó khớp một đường giữa các điểm). Kỹ thuật này được sử dụng rất phổ biến để tính toán các giá trị cảm biến mà không cần sử dụng phương trình. Hàm np.inpterp cung cấp phép nội suy tuyến tính.

Tạo một chương trình để nội suy các giá trị nhiệt độ của điện trở nhiệt dựa trên các giá trị nhiệt độ đã cho. Chương trình sẽ được kiểm thử bằng cách tạo dữ liệu thử nghiệm. Sử dụng hàm np.linespace (start, stop, nrSteps) để tạo các giá trị thử nghiệm cho điện trở. Hãy cho start là 3000, stop là 3000 và 50 steps. Vẽ đồ thị kết quả trong bảng dưới đây. Chèn hình ảnh đồ thị của bạn trong cột "Your solution". Đảm bảo rằng giải pháp của riêng bạn có màu sắc, nhãn và tiêu đề giống như đồ thị mẫu. Tiếp tục tải lên toàn bộ hình ảnh và mã trên nhiệm vụ được cung cấp.

Mô tả:

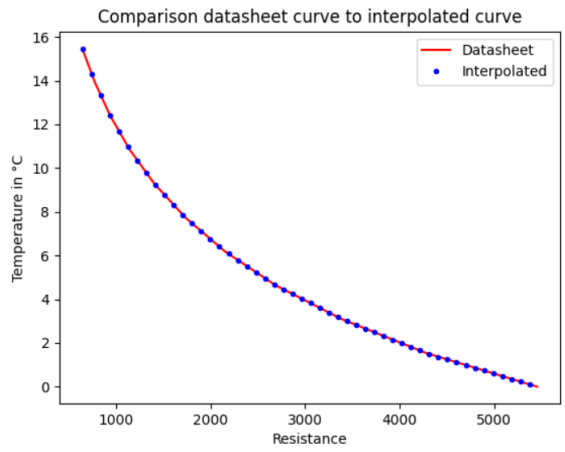
x=np.linspace(start,stop,nrSteps)

- Start… giá trị đầu tiên được tạo ra
- Stop… giá trị cuối cùng được tạo ra
- nrSteps số bước giữa giá trị đầu tiên và giá trị cuối cùng

y=np.interp(x,Xp,Yp)

- X… giá trị x trong đó phép nội suy sẽ được tính toán
- Xp… mảng các điểm đã cho theo hướng x từ bảng dữ liệu
- Yp… mảng các điểm đã cho theo hướng y từ bảng dữ liệu

Sample plot	Your solution



Hình 65: Nội suy tuyến tính

4. Chương trình đo lường - vi điều khiển cơ bản

Sau khi chọn phần cứng, một chương trình đo đơn giản sẽ được triển khai. Chương trình đo phải đọc vào các giá trị của cảm biến nhiệt độ và giá trị của cảm biến gia tốc và in chúng trực tiếp trên màn hình console. Mỗi phép đo sẽ được thực hiện trong một khoảng thời gian 2 giây. Nhập ảnh chụp màn hình console xuất ra trong bảng bên dưới. Tải lên mã của bạn và ảnh chụp màn hình console xuất ra.

Đầu ra yêu cầu sẽ là:

“Temperature sensor: XXX°C”

“AccelerometerX: XXXg”

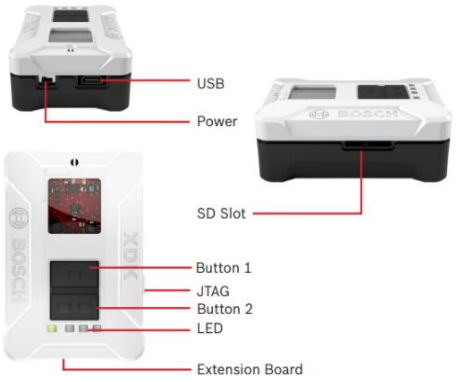
“AccelerometerY: YYYg”

“AccelerometerZ: ZZZg”

Xuất ra màn hình console

5. Chương trình đo lường - Bosch XDK

Bosch XDK là một nền tảng vi điều khiển mạnh mẽ với một mảng cảm biến rộng. Nó thường được sử dụng trong các ứng dụng công nghiệp, đặc biệt là để truy vết dữ liệu. Bosch XDK cung cấp một mảng cảm biến rộng, mô-đun wifi tích hợp, đầu đọc thẻ SD để lưu trữ dữ liệu trên thẻ nhớ, các nút nhấn có thể cấu hình tự do và đầu nối USB để kết nối XDK với PC để chạy (flashing) các chương trình mới và xuất trực tiếp các giá trị trong màn hình console IDE của riêng nó.



Hình 66: Cấu trúc cảm biến XDK

Hình 67: Chức năng cảm biến XDK

Bosch XDK còn cho phép người dùng truy cập nhiều loại cảm biến. Người dùng có thể đo gia tốc, nhiệt độ, độ ẩm, áp suất không khí và nhiều hơn nữa. Hơn nữa, XDK sử dụng "FreeRTOS" làm hệ điều hành thời gian thực. Điều này đảm bảo có thể thực hiện được các phép đo được định thời chính xác với khoảng thời gian xác định.

Ngoài ra, Bosch sử dụng một Eclipse IDE riêng được gọi là XDK-Workbench. IDE này cung cấp cho người dùng lựa chọn để lập trình XDK trực tiếp với C hoặc sử dụng ngôn ngữ lập trình MITA. Ngôn ngữ lập trình MITA được phát triển để giảm bớt gánh nặng khi tạo các ứng dụng I4.0 mà không có kiến thức về lập trình nhúng. Trang web: **https://developer.bosch.com/web/xdk/getting-started#1** được cung cấp bởi Bosch và có code mẫu để đọc vào các giá trị cảm biến. Một phần của nhiệm vụ này là đọc qua các mô tả và code mẫu và sử dụng lại chúng để tạo một chương trình làm việc.

Nhiệm vụ của phần này là tạo một chương trình đo đọc vào các giá trị cảm biến của:

- Gia tốc kế
- Cảm biến độ ẩm
- Cảm biến nhiệt độ
- Cảm biến áp suất

Giá trị đọc được sẽ được xuất ra trên màn hình console của XDK - Workbench. Dự án sẽ được lập trình bằng ngôn ngữ lập trình MITA. Các phép đo phải được thực hiện sau mỗi 3 giây. Kết quả xuất ra màn hình console mong muốn sẽ như sau:

Acceleration in X: XXXg

Acceleration in Y: YYYg

Acceleration in Z: ZZZg

Humidity: XXX%

Temperature: XXX°C

Air pressure: XXXkPa

Nhập ảnh chụp màn hình console xuất ra trong bảng bên dưới. Tải lên mã của bạn và ảnh chụp màn hình console xuất ra.

Xuất ra màn hình console

6. Vị trí đặt cảm biến và hệ thống dây điện

Sau khi các chương trình kiểm thử được thực hiện thành công, các cảm biến có thể được đặt ở vị trí mong muốn. Đối với trạm CSPi4.0, nhiệt độ môi trường xung quanh và độ rung của xi lanh đẩy phải được đo. Tương tự đối với trạm đóng chai. Tìm vị trí thích hợp để đặt cảm biến và ghi lại quyết định của bạn bằng hình ảnh. Sau đó hãy mô tả theo cách của riêng bạn tại sao bạn chọn vị trí này trong bảng dưới đây:

Hình ảnh	Thuyết minh

7. Giao thức giao tiếp

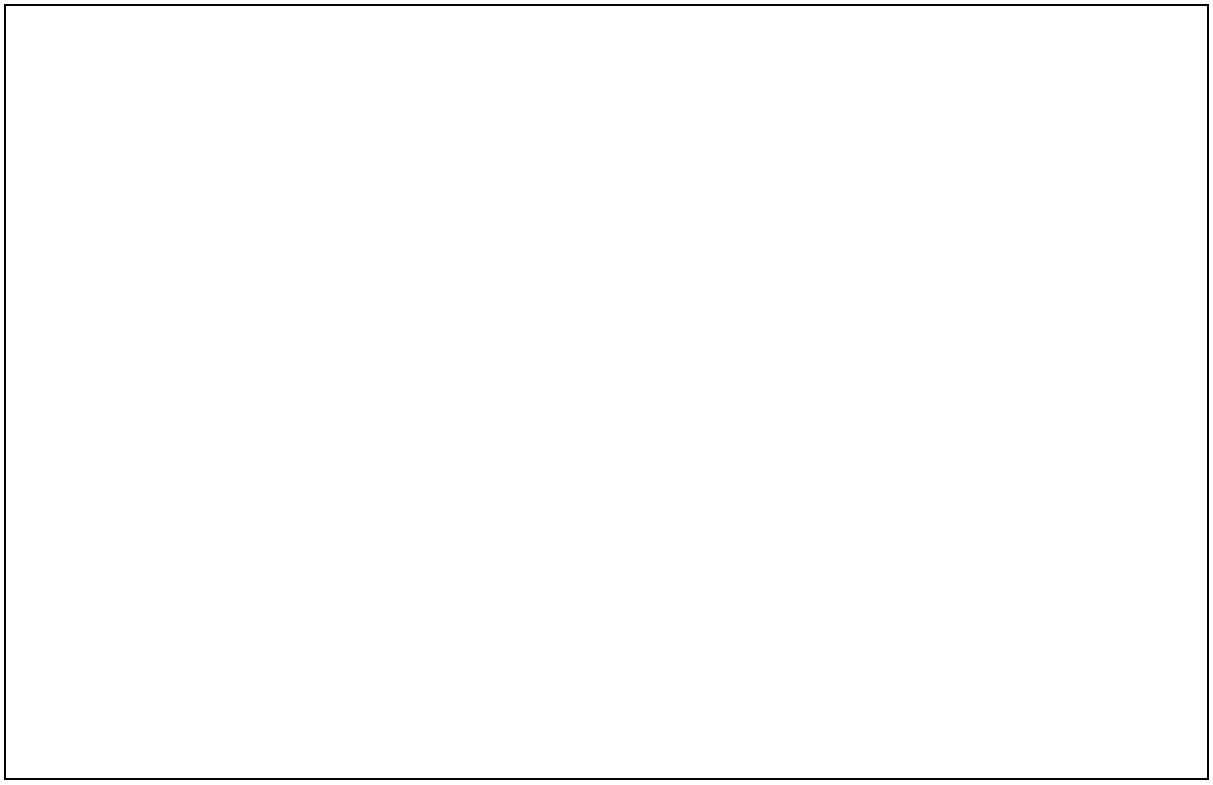
Sau khi các cảm biến và vi điều khiển có vị trí hợp lệ để thực hiện phép đo, giao thức giao tiếp sẽ được triển khai. Sau khi thực hiện một trong số các giải pháp, hãy chụp ảnh kết quả xuất ra màn hình console của máy tính và in ra trong trường bên dưới. Tiếp tục tải lên kết quả xuất ra màn hình console và tất cả code.

7.1 Vi điều khiển phổ biến

Các cảm biến được mô tả ở phần trước của vi điều khiển sẽ được sử dụng để tạo chương trình đo. Về giao thức giao tiếp, phương thức Publisher/Subcriber bằng giao thức MQTT sẽ được sử dụng.

Các bước:

6. Thiết lập broker trên máy tính và chạy broker
7. Sử dụng chương trình đo của bạn để vi điều khiển đọc vào các giá trị cảm biến. Để vi điều khiển đọc trong khoảng thời gian 1,5 giây
8. Thêm các thư viện cần thiết vào vi điều khiển và tạo chủ đề (topic) để xuất bản.
    a. Tên chủ đề cảm biến nhiệt độ
        i. Measurement/Temp
    b. Tên chủ đề gia tốc kế
        i. Measurement/AccelX
        ii. Measurement/AccelY
        iii. Measurement/AccelZ
9. Thêm các thư viện cần thiết trên máy tính và đăng ký chủ đề nơi các giá trị cảm biến được xuất bản
10. In các giá trị đã xuất bản trên màn hình console máy tính
    Xuất ra màn hình console

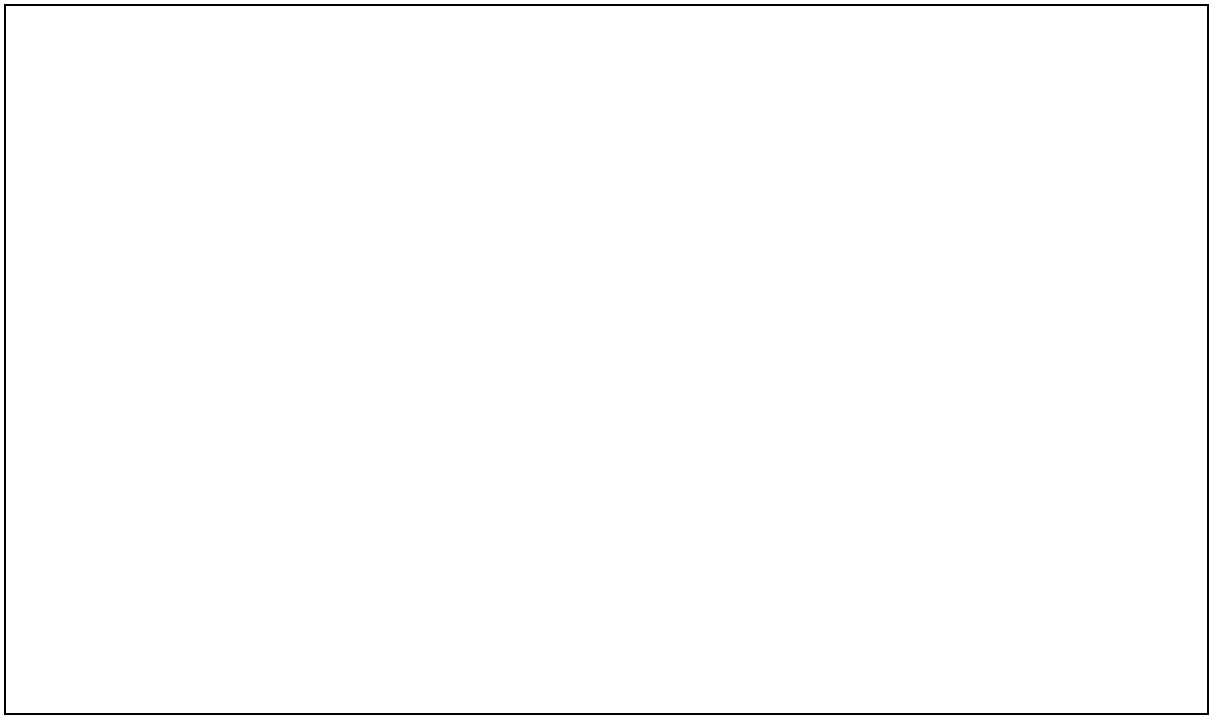


7.2 XDK

XDK cung cấp khả năng truyền dữ liệu qua phương thức Publisher/Subcriber bằng giao thức MQTT. Vì XDK cung cấp nhiều loại cảm biến nên nhiều cảm biến sẽ được truyền đến máy tính.

Các bước:

1. Thiết lập broker trên máy tính và chạy broker
2. Sử dụng chương trình đo của bạn để vi điều khiển đọc vào các giá trị cảm biến. Để vi điều khiển đọc trong khoảng thời gian 3 giây
3. Thêm các thư viện cần thiết vào vi điều khiển và tạo chủ đề (topic) để xuất bản.
    a. Tên chủ đề cảm biến nhiệt độ
        i. MeasurementXDK/Temp
    b. Tên chủ đề gia tốc kế
        i. MeasurementXDK/AccelX
        ii. MeasurementXDK/AccelY
        iii. MeasurementXDK/AccelZ
    c. Tên chủ đề độ ẩm
        i. MeasurementXDK/Humidity
    d. Tên chủ đề áp suất
        i. MeasurementXDK/Pressure
4. Thêm các thư viện cần thiết trên máy tính và đăng ký chủ đề nơi các giá trị cảm biến được xuất bản
5. In các giá trị đã xuất bản trên màn hình console máy tính
6. Xuất ra màn hình console



7.3 Tạo tệp đo

Sau khi kết nối thành công với máy tính, hãy cập nhật chương trình máy tính để ghi các giá trị cảm biến nhiệt độ nhận được vào tệp .csv. Tệp sẽ được đặt tên là "temperature.csv". Trong quá trình đo, vi điều khiển sẽ gửi một phép đo mới sau mỗi 5 giây. Sau 2 phút, phép đo sẽ dừng lại. Tính giá trị trung bình của tất cả các phép đo và tải lên tệp đo.

Giá trị trung bình:

BÀI 3: HỆ THỐNG CƠ SỞ DỮ LIỆU

Mục tiêu: Sau bài học này các học viên có thể:

- Hiểu biết cơ bản về cơ sở dữ liệu và các khái niệm hệ thống liên quan
- Có thể thiết lập và cấu hình máy chủ cơ sở dữ liệu
- Biết các trình tự lệnh cơ bản để thao tác cơ sở dữ liệu
- Có thể phát triển các chương trình ghi các giá trị đo được vào cơ sở dữ liệu và đọc chúng từ cơ sở dữ liệu

Nội dung:

1. Cơ bản về hệ thống cơ sở dữ liệu

1.1 Những điều cơ bản để sử dụng cơ sở dữ liệu

1.1.1 Khái niệm cơ bản về cơ sở dữ liệu

Một cơ sở dữ liệu dự định sẽ có một bộ sưu tập dữ liệu được lưu trữ cùng nhau để phục vụ nhiều ứng dụng nhất có thể. Do đó, một cơ sở dữ liệu thường được hình thành như một kho lưu trữ thông tin cần thiết để chạy một số chức năng nhất định trong một công ty hoặc tổ chức. Một cơ sở dữ liệu như vậy sẽ cho phép không chỉ việc truy xuất dữ liệu mà còn sửa đổi liên tục dữ liệu cần thiết để kiểm soát hoạt động. Có thể tìm kiếm cơ sở dữ liệu để có được câu trả lời cho các truy vấn hoặc thông tin cho mục đích lập kế hoạch.

Mục đích của Cơ sở dữ liệu

Cơ sở dữ liệu phải là một kho lưu trữ dữ liệu cần thiết cho việc xử lý dữ liệu của tổ chức. Dữ liệu đó phải chính xác, riêng tư và được bảo vệ khỏi bị hư hại. Nó phải chính xác để các ứng dụng đa dạng với các yêu cầu dữ liệu khác nhau có thể sử dụng dữ liệu. Các lập trình viên ứng dụng khác nhau và những người dùng cuối khác nhau có quan điểm khác nhau về dữ liệu, phải được bắt nguồn từ cấu trúc dữ liệu tổng thể chung. Phương pháp tìm kiếm và truy cập dữ liệu của họ sẽ khác nhau.

Ưu điểm của việc sử dụng cơ sở dữ liệu

Cơ sở dữ liệu giảm thiểu sự dư thừa dữ liệu ở một mức độ lớn.

Cơ sở dữ liệu có thể kiểm soát sự không nhất quán của dữ liệu ở một mức độ lớn.

Chia sẻ dữ liệu cũng có thể sử dụng cơ sở dữ liệu.

Cơ sở dữ liệu thực thi các tiêu chuẩn.

Việc sử dụng Cơ sở dữ liệu có thể đảm bảo an ninh dữ liệu.

Tính toàn vẹn có thể được quản lý bằng cách sử dụng cơ sở dữ liệu.

Các cấp độ triển khai cơ sở dữ liệu khác nhau

Cơ sở dữ liệu được thực hiện thông qua ba cấp độ chung. Các cấp độ này là:

Cấp độ nội bộ hoặc cấp độ vật lý

Cấp độ khái niệm

Mức Bên ngoài hoặc Mức dạng xem

Khái niệm độc lập dữ liệu

Vì cơ sở dữ liệu có thể được xem qua ba cấp độ trừu tượng, bất kỳ thay đổi nào ở bất kỳ cấp độ nào cũng có thể ảnh hưởng đến lược đồ của các cấp khác. Vì cơ sở dữ liệu

tiếp tục phát triển, sau đó đôi khi có thể có những thay đổi thường xuyên. Điều này không nên dẫn đến việc thiết kế lại và thực hiện lại cơ sở dữ liệu. Các khái niệm về độc lập dữ liệu chứng minh có lợi trong các loại bối cảnh như vậy.

Tính độc lập về dữ liệu vật lý

Độc lập dữ liệu logic

Các thuật ngữ cơ bản liên quan đến cơ sở dữ liệu và SQL

Quan hệ: Nói chung, một mối quan hệ là một bảng, tức là dữ liệu được sắp xếp theo hàng và cột. Một mối quan hệ có các thuộc tính sau:

Trong bất kỳ cột nhất định nào của bảng, tất cả các mục đều có cùng loại, trong khi các mục trong các cột khác nhau có thể không cùng loại.

Đối với một hàng, mỗi cột phải có giá trị nguyên tử, và cũng đối với một hàng, một cột không thể có nhiều hơn một giá trị.

Tất cả các hàng của một mối quan hệ là khác nhau.

Việc sắp xếp các hàng trong một mối quan hệ là phi vật chất.

Cột của một mối quan hệ được gán tên riêng biệt và thứ tự của các cột này là phi vật chất.

Tuple: Các hàng bảng trong một mối quan hệ thường được gọi là Tuples.

Thuộc tính: Các cột hoặc trường của bảng được gọi là Thuộc tính.

Mức độ: Số lượng thuộc tính trong một mối quan hệ xác định mức độ quan hệ. Một mối quan hệ có ba thuộc tính được cho là có mối quan hệ của mức độ 3.

Thuộc tính: Số lượng tuples hoặc hàng trong một mối quan hệ được gọi là thuộc tính.

1.1.2. Thành phần phần mềm cần thiết (cơ sở dữ liệu máy chủ)

Thành phần cơ sở dữ liệu bao gồm:

Công cụ lưu trữ

Ngôn ngữ Truy vấn

Bộ xử lý truy vấn

Công cụ tối ưu hóa

Danh mục siêu dữ liệu

Trình quản lý nhật ký

Công cụ báo cáo và giám sát

Tiện ích dữ liệu

Công cụ lưu trữ

Công cụ lưu trữ là thành phần cốt lõi của DBMS tương tác với hệ thống tệp ở cấp độ hệ điều hành để lưu trữ dữ liệu. Tất cả các truy vấn SQL tương tác với dữ liệu cơ bản đều đi qua công cụ lưu trữ.

Ngôn ngữ Truy vấn

Một ngôn ngữ truy cập cơ sở dữ liệu là cần thiết để tương tác với cơ sở dữ liệu, từ việc tạo cơ sở dữ liệu đến chỉ cần chèn hoặc truy xuất dữ liệu. Một DBMS thích hợp phải hỗ trợ một hoặc nhiều ngôn ngữ truy vấn và phương ngữ ngôn ngữ. Ngôn ngữ truy vấn có

cấu trúc (SQL) và Ngôn ngữ Truy vấn MongoDB (MQL) là hai ngôn ngữ truy vấn được sử dụng để tương tác với cơ sở dữ liệu.

Trong nhiều ngôn ngữ truy vấn, chức năng ngôn ngữ truy vấn có thể được phân loại thêm theo các tác vụ cụ thể:

Ngôn ngữ định nghĩa dữ liệu (DDL). Điều này bao gồm các lệnh có thể được sử dụng để xác định lược đồ cơ sở dữ liệu hoặc sửa đổi cấu trúc của các đối tượng cơ sở dữ liệu.

Ngôn ngữ thao tác dữ liệu (DML). Lệnh trực tiếp xử lý dữ liệu trong cơ sở dữ liệu. Tất cả các hoạt động CRUD đều thuộc DML.

Ngôn ngữ kiểm soát dữ liệu (DCL). Điều này liên quan đến các quyền và các điều khiển truy cập khác của cơ sở dữ liệu.

Ngôn ngữ kiểm soát giao dịch (TCL). Lệnh liên quan đến các giao dịch cơ sở dữ liệu nội bộ.

Bộ xử lý truy vấn

Đây là trung gian giữa các truy vấn của người dùng và cơ sở dữ liệu. Bộ xử lý truy vấn giải thích các truy vấn của người dùng và làm cho họ có thể thực hiện các lệnh có thể được hiểu bởi cơ sở dữ liệu để thực hiện chức năng thích hợp.

Công cụ tối ưu hóa

Công cụ tối ưu hóa cho phép DBMS cung cấp thông tin chi tiết về hiệu suất của cơ sở dữ liệu về tối ưu hóa chính cơ sở dữ liệu và các truy vấn. Khi kết hợp với các công cụ giám sát cơ sở dữ liệu,nó có thể cung cấp một bộ công cụ mạnh mẽ để đạt được hiệu suất tốt nhất từ cơ sở dữ liệu.

Danh mục siêu dữ liệu

Đây là danh mục tập trung của tất cả các đối tượng trong cơ sở dữ liệu. Khi một đối tượng được tạo, DBMS sẽ lưu giữ bản ghi của đối tượng đó với một số siêu dữ liệu về nó bằng danh mục siêu dữ liệu. Sau đó, bản ghi này có thể được sử dụng để:

Xác minh yêu cầu của người dùng đối với các đối tượng cơ sở dữ liệu thích hợp

Cung cấp tổng quan về cấu trúc cơ sở dữ liệu hoàn chỉnh

Quản lý nhật ký

Thành phần này sẽ giữ tất cả các nhật ký của DBMS. Các nhật ký này sẽ bao gồm thông tin đăng nhập và hoạt động của người dùng, các chức năng cơ sở dữ liệu, các chức năng sao lưu và khôi phục, v.v. Người quản lý nhật ký đảm bảo tất cả các nhật ký này được ghi lại đúng cách và dễ dàng truy cập.

Báo cáo & công cụ giám sát

Các công cụ báo cáo và giám sát là một thành phần tiêu chuẩn khác đi kèm với DBMS. Công cụ báo cáo sẽ cho phép người dùng tạo báo cáo trong khi các công cụ giám sát cho phép giám sát cơ sở dữ liệu về mức tiêu thụ tài nguyên, hoạt động của người dùng, v.v.

Tiện ích dữ liệu

Ngoài tất cả những điều trên, hầu hết phần mềm DBMS đi kèm với các tiện ích có sẵn bổ sung để cung cấp các chức năng như:

Kiểm tra tính toàn vẹn của dữ liệu
Sao lưu và khôi phục
Sửa chữa cơ sở dữ liệu đơn giản
Xác thực dữ liệu
……

1.1.3 Cấu hình thủ công cài đặt máy chủ cơ sở dữ liệu

Bạn phải tùy chỉnh các thuộc tính và tính năng của máy chủ cơ sở dữ liệu bằng cách đặt các thông số cấu hình, tạo không gian lưu trữ và định cấu hình kết nối. Bạn có thể tự động khởi động.

Bạn tùy chỉnh các thuộc tính của máy chủ cơ sở dữ liệu bằng cách đặt hoặc sửa đổi các thông số cấu hình trong tệp cấu hình trên. Bạn có thể sử dụng Công cụ IBM® OpenAdmin (OAT) cho Informix® để theo dõi và cập nhật cấu hình của bạn. OAT cung cấp các đề xuất về giá trị tham số cấu hình để tối ưu hóa cấu hình máy chủ cơ sở dữ liệu của bạn. Phiên bản hiện tại của IBM Informix không sử dụng một số tham số cấu hình được sử dụng trong các phiên bản trước của máy chủ.

Nếu bạn chọn định cấu hình máy chủ cơ sở dữ liệu trong khi cài đặt, nhiều tham số cấu hình và biến môi trường sẽ được đặt và một tập hợp không gian lưu trữ được tạo tự động. Ngoài ra, bạn có thể định cấu hình máy chủ cơ sở dữ liệu theo cách thủ công.

Khi bạn khởi động máy chủ cơ sở dữ liệu lần đầu tiên, dung lượng đĩa được khởi tạo và phần ban đầu của dbspace gốc được tạo. Mọi dữ liệu hiện có trong không gian đĩa đó đều bị ghi đè. Bộ nhớ dùng chung mà máy chủ cơ sở dữ liệu yêu cầu cũng được khởi tạo. Sau đó, khi bạn khởi động máy chủ cơ sở dữ liệu, chỉ bộ nhớ dùng chung được khởi tạo. Mặc dù dbspace gốc là vị trí mặc định của tệp nhật ký và cơ sở dữ liệu, bạn có thể lưu trữ tệp nhật ký và cơ sở dữ liệu trong không gian lưu trữ khác để ngăn dbspace gốc hết dung lượng.

Tạo và quản lý không gian lưu trữ

Bạn có thể tạo nhiều không gian lưu trữ để lưu trữ các loại đối tượng khác nhau, chẳng hạn như dữ liệu, chỉ mục, nhật ký, đối tượng tạm thời, thay vì lưu trữ mọi thứ trong dbspace gốc. Cách bạn phân phối dữ liệu trên đĩa ảnh hưởng đến hiệu suất của máy chủ cơ sở dữ liệu. Bạn có thể định cấu hình máy chủ cơ sở dữ liệu để vừa tự động thu nhỏ dung lượng lưu trữ mà dữ liệu yêu cầu vừa tự động mở rộng không gian lưu trữ khi cần thiết. Bạn có thể tách biệt các tài nguyên lưu trữ và xử lý giữa nhiều tổ chức khách hàng bằng cách định cấu hình nhiều quyền.

Điều chỉnh hiệu suất tự động

Bạn có thể đặt các thông số cấu hình và các tác vụ của Bộ lập lịch để cho phép máy chủ cơ sở dữ liệu tự động điều chỉnh các giá trị ảnh hưởng đến hiệu suất. Theo mặc định, nhiều thông số cấu hình điều chỉnh tự động và các tác vụ của Bộ lập lịch được đặt để giải quyết các vấn đề về hiệu suất phổ biến.

Cấu hình tính năng

Bạn có thể cấu hình máy chủ cơ sở dữ liệu để hỗ trợ các loại chức năng tùy chọn mà bạn cần.

Cấu hình kết nối

Thông tin kết nối cho phép ứng dụng khách kết nối với máy chủ cơ sở dữ liệu trên mạng. Bạn phải chuẩn bị thông tin kết nối ngay cả khi ứng dụng khách và máy chủ cơ sở dữ liệu nằm trên cùng một máy tính hoặc nút.

Giới hạn tài nguyên

Bạn có thể giới hạn tài nguyên có sẵn cho các phiên riêng lẻ để phân phối đồng đều hơn việc sử dụng hệ thống và ngăn độc quyền tài nguyên.

Tự động khởi động và tắt máy trên UNIX

Bạn có thể sửa đổi các tập lệnh khởi động và tắt máy trên UNIX để tự động khởi động và tắt máy chủ cơ sở dữ liệu.

Tự động khởi động trên Windows

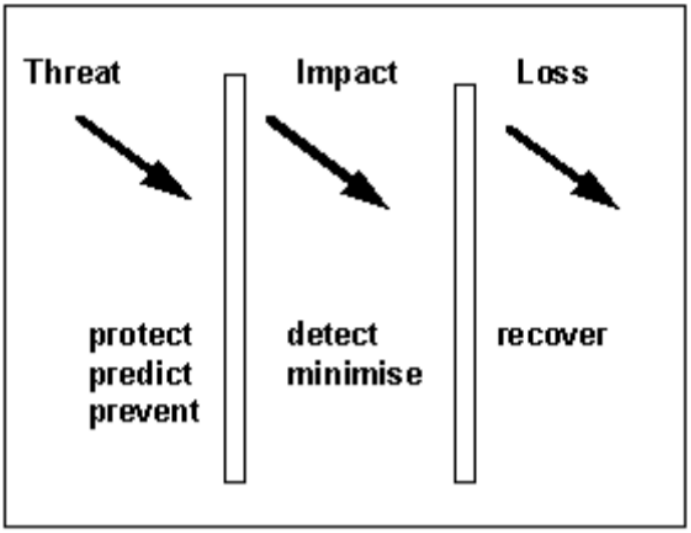
Bạn có thể tự động khởi động máy chủ cơ sở dữ liệu trên Windows.

1.1.4. Tạo cơ sở dữ liệu có xem xét các khía cạnh bảo mật

Phạm vi bảo mật cơ sở dữ liệu

Tổng quát

Tất cả các hệ thống đều có TÀI SẢN và bảo mật là bảo vệ tài sản. Sau đó, điều đầu tiên là phải biết tài sản của bạn và giá trị của chúng. Trong chương này, tập trung vào các đối tượng cơ sở dữ liệu (bảng, dạng xem, hàng), quyền truy cập vào chúng và hệ thống tổng thể quản lý chúng. Lưu ý rằng không phải tất cả dữ liệu đều nhạy cảm, vì vậy không phải tất cả dữ liệu đều cần nỗ lực bảo vệ. Tất cả tài sản đang bị đe dọa. Điều thứ hai cần biết là BA điều gì đang khiến tài sản của bạn gặp rủi ro. Chúng bao gồm những thứ như mất điện và gian lận của nhân viên. Lưu ý rằng các mối đe dọa một phần là giả thuyết, luôn thay đổi và luôn được biết đến một cách không hoàn hảo. Hoạt động bảo mật hướng vào việc bảo vệ hệ thống khỏi các mối đe dọa đã nhận biết được. Nếu một mối đe dọa tiềm ẩn, bạn phải cho phép nó trở thành hiện thực. Khi nó trở thành hiện thực thì có TÁC ĐỘNG. Tác động bạn có thể xem xét và lập kế hoạch. Nhưng trong trường hợp xấu nhất, sẽ bị MẤT. Hoạt động bảo mật ở đây hướng đến việc giảm thiểu tổn thất và khôi phục cơ sở dữ liệu để giảm thiểu tổn thất cũng như bảo vệ hơn nữa khỏi các mối đe dọa tương tự hoặc tương tự.



Hình 68: Bảo mật cơ sở dữ liệu

Một cơ chế phát triển được vạch ra là:
1. Tài sản chứng từ (chúng là gì, giá trị của chúng là bao nhiêu).
2. Xác định các món ăn (chúng là gì, khả năng xảy ra như thế nào, tác động ra sao nếu chúng xảy ra).
3. Liên kết các mối đe dọa với từng tài sản.
4. Thiết kế các cơ chế để bảo vệ từng tài sản phù hợp với giá trị và chi phí bảo vệ của chúng, nhằm phát hiện vi phạm an ninh đối với từng tài sản, nhằm giảm thiểu tổn thất phát sinh và phục hồi hoạt động bình thường.

Các mối đe dọa đối với cơ sở dữ liệu

Bạn sẽ xây dựng kỹ năng bảo mật của mình từ hai hướng. Một là từ sự đánh giá cao và nhận thức về các mối đe dọa đang thay đổi, và hai là từ các biện pháp kỹ thuật đối với chúng. Các mối đe dọa bao gồm:

- Sửa đổi trái phép: Thay đổi giá trị dữ liệu vì lý do phá hoại, tội phạm hoặc thiếu hiểu biết có thể được kích hoạt bởi cơ chế bảo mật không đầy đủ, hoặc chia sẻ mật khẩu hoặc đoán mật khẩu, chẳng hạn.
- Tiết lộ trái phép: Khi những thông tin đáng lẽ không được tiết lộ đã bị tiết lộ. Một vấn đề chung có tầm quan trọng cốt yếu, có thể là vô tình hoặc cố ý Mất khả năng cung cấp: Đôi khi được gọi là từ chối dịch vụ. Khi cơ sở dữ liệu không có sẵn, nó sẽ bị mất (nếu không thì cuộc sống sẽ tốt hơn nếu không có hệ thống!). Vì vậy, bất kỳ mối đe dọa nào làm phát sinh thời gian ngoại tuyến, thậm chí để kiểm tra xem có điều gì đó đã xảy ra hay không, đều phải tránh. Phần còn lại của phần này là tổng quan về các loại mối đe dọa quy định cụ thể đối với hệ thống cơ sở dữ liệu.
- Tính nhạy cảm về thương mại: Hầu hết các tổn thất tài chính do gian lận đều phát sinh từ nhân viên. Kiểm soát truy cập cung cấp cả biện pháp bảo vệ chống lại các hành vi phạm tội và bằng chứng về những nỗ lực (thành công hoặc theo cách khác) để thực hiện các hành vi gây bất lợi cho tổ chức, cho dù là gian lận, trích xuất dữ liệu nhạy cảm hoặc mất khả năng cung cấp.

- Quyền riêng tư cá nhân và bảo vệ dữ liệu: Trên bình diện quốc tế, dữ liệu cá nhân thường chịu sự kiểm soát của pháp luật. Dữ liệu cá nhân là dữ liệu về một cá nhân có thể nhận dạng được. Thường thì cá nhân đó phải còn sống nhưng phương pháp nhận dạng không được quy định. Vì vậy, trong một số trường hợp, mã bưu điện cho một ngôi nhà có thể xác định một cá nhân, nếu chỉ một người đang sống tại một địa chỉ có mã bưu điện. Những dữ liệu này cần được xử lý và kiểm soát cẩn thận. Để biết thêm thông tin, hãy xem Bảo vệ dữ liệu ở phần sau của chương. Các vấn đề là quá rộng để được thảo luận ở đây nhưng các tác động cần được lưu ý. Dữ liệu cá nhân cần được xác định như vậy. Kiểm soát phải tồn tại đối với việc sử dụng dữ liệu đó (có thể hạn chế các truy vấn đặc biệt). Các dấu vết đánh giá về mọi truy cập và tiết lộ thông tin cần được lưu giữ làm bằng chứng.
- Lạm dụng máy tính: Nói chung cũng có luật về việc lạm dụng máy tính. Lạm dụng bao gồm vi phạm các kiểm soát truy cập và cố gắng gây ra thiệt hại bằng cách thay đổi trạng thái cơ sở dữ liệu hoặc đưa sâu và vi rút vào để can thiệp vào hoạt động bình thường. Những tội này thường có thể bị dẫn độ. Vì vậy, một truy cập trái phép ở Hồng Kông sử dụng máy tính ở Pháp để truy cập cơ sở dữ liệu ở Đức tham chiếu đến cơ sở dữ liệu ở Mỹ có thể dẫn đến Pháp hoặc Đức hoặc Mỹ bị dẫn độ.
- Yêu cầu kiểm toán: Đây là những ràng buộc hoạt động được xây dựng xung quanh nhu cầu biết ai đã làm gì, ai đã cố gắng làm gì, và mọi thứ đã xảy ra ở đâu và khi nào. Chúng liên quan đến việc phát hiện các sự kiện (bao gồm các giao dịch CONNECT và GRANT), cung cấp bằng chứng để phát hiện, đảm bảo cũng như bào chữa hoặc truy tố. Có những vấn đề liên quan đến bằng chứng do máy tính tạo ra không được đề cập ở đây.

Khi xem xét việc truy cập hợp lý vào cơ sở dữ liệu, có thể dễ dàng bỏ qua thực tế rằng tất cả các truy cập hệ thống đều tiềm ẩn rủi ro. Nếu có quyền truy cập vào các tiện ích của hệ điều hành, bạn có thể truy cập trực tiếp vào ổ lưu trữ và sao chép hoặc làm hỏng toàn bộ cơ sở dữ liệu hoặc các thành phần của nó. Việc xem xét đầy đủ phải tính đến tất cả các quyền truy cập như vậy. Hầu hết các nhà phân tích sẽ tìm cách giảm thiểu thông tin liên lạc (trực tiếp, mạng và viễn thông) và cô lập hệ thống khỏi các mối đe dọa không cần thiết. Cũng có khả năng là mã hóa sẽ được sử dụng cả trên dữ liệu và lược đồ. Mã hóa là quá trình chuyển đổi văn bản và dữ liệu thành một dạng mà chỉ người nhận dữ liệu hoặc văn bản đó mới có thể đọc được, người đó phải biết cách chuyển đổi nó trở lại thành một thông điệp rõ ràng. Bạn sẽ thấy dễ dàng hơn khi coi bảo mật và kiểm toán là các vấn đề tách biệt với các chức năng cơ sở dữ liệu chính, tuy nhiên chúng được thực hiện. Hình dung máy chủ bảo mật và máy chủ kiểm toán dưới dạng các mô-đun chức năng riêng biệt.

Nguyên tắc bảo mật cơ sở dữ liệu

Để cấu trúc những suy nghĩ về bảo mật, bạn cần một mô hình bảo mật. Chúng có nhiều dạng tùy thuộc vào vai trò, mức độ chi tiết và mục đích. Các danh mục chính là các lĩnh

vực quan tâm (các mối đe dọa, tác động và mất mát) cũng như các hành động liên quan để đối phó với chúng. Rủi ro an ninh được nhìn thấy ở khía cạnh mất mát tài sản. Các tài sản này bao gồm:

- Phần cứng
- Phần mềm
- Dữ liệu
- Chất lượng dữ liệu
- Sự tín nhiệm
- Tính khả dụng
- Lợi ích kinh doanh

Ở đây, chúng tôi chủ yếu quan tâm đến các mối đe dọa đối với dữ liệu và chất lượng dữ liệu, nhưng tất nhiên, mối đe dọa đối với một nội dung có tác động đến các nội dung khác. Điều quan trọng luôn là bạn phải hiểu rõ tài sản nào cần được bảo vệ. Vì vậy, như một bản tóm tắt: Bạn cần chấp nhận rằng bảo mật không bao giờ có thể hoàn hảo. Luôn luôn tồn tại một yếu tố rủi ro, vì vậy cần phải thu xếp để đối phó với tình huống xấu nhất - có nghĩa là các bước để giảm thiểu tác động và phục hồi hiệu quả từ mất mát hoặc thiệt hại đối với tài sản. Những điểm cần lưu ý: 6 1. Bảo mật phù hợp - bạn không muốn chi nhiều hơn cho bảo mật so với giá trị của tài sản. 2. Bạn không muốn các biện pháp bảo mật can thiệp một cách không cần thiết vào hoạt động bình thường của hệ thống.

Các mô hình bảo mật

Mô hình bảo mật thiết lập các tiêu chí bên ngoài để kiểm tra các vấn đề bảo mật nói chung và cung cấp bối cảnh cho việc xem xét cơ sở dữ liệu, bao gồm cả việc triển khai và vận hành. Các DBMS cụ thể có các mô hình bảo mật riêng rất quan trọng trong thiết kế và vận hành hệ thống. Tham khảo mô hình SeaView để làm ví dụ.

Bạn sẽ nhận ra rằng các mô hình bảo mật giải thích các tính năng có sẵn trong DBMS cần được sử dụng để phát triển và vận hành các hệ thống bảo mật thực tế. Chúng thể hiện các khái niệm, thực hiện các chính sách và cung cấp máy chủ cho các chức năng đó. Bất kỳ lỗi nào trong mô hình bảo mật sẽ chuyển thành hoạt động không an toàn hoặc hệ thống vụng về.

Kiểm soát truy cập

Mục đích của kiểm soát truy cập phải luôn rõ ràng. Kiểm soát truy cập tốn kém về chi phí phân tích, thiết kế và vận hành. Nó được áp dụng cho các tình huống đã biết, cho các tiêu chuẩn đã biết, để đạt được các mục đích đã biết. Không áp dụng các biện pháp kiểm soát mà không có tất cả các kiến thức ở trên. Kiểm soát luôn luôn phải phù hợp với hoàn cảnh. Các vấn đề chính được giới thiệu dưới đây.

Xác thực và ủy quyền

Tất cả chúng ta là người dùng quen thuộc với yêu cầu đăng nhập của hầu hết các hệ thống. Việc truy cập vào các tài nguyên CNTT thường yêu cầu một quy trình đăng nhập được tin cậy để bảo mật. Chủ đề này là về quyền truy cập vào hệ quản trị cơ sở dữ liệu và là một cái nhìn tổng quan về quy trình từ góc độ DBA. Hầu hết những gì sau đây là

trực tiếp về hệ thống máy khách-máy chủ quan hệ. Các mô hình hệ thống khác khác nhau ở mức độ lớn hơn hoặc ít hơn, mặc dù các nguyên tắc cơ bản vẫn đúng. Để có một giản đồ đơn giản, hãy xem Sơ đồ ủy quyền và xác thực. Trong số các nguyên tắc chính cho hệ thống cơ sở dữ liệu là xác thực và ủy quyền. Xác thực 7 Máy khách phải thiết lập danh tính của máy chủ và máy chủ phải thiết lập danh tính của máy khách. Điều này được thực hiện thường xuyên bằng các bí mật được chia sẻ (kết hợp mật khẩu / user-id hoặc dữ liệu tiểu sử và / hoặc sinh trắc học được chia sẻ). Nó cũng có thể đạt được bởi một hệ thống cơ quan cao hơn đã thiết lập xác thực trước đó. Trong các hệ thống máy khách-máy chủ nơi dữ liệu (không nhất thiết là cơ sở dữ liệu) được phân phối, xác thực có thể được chấp nhận từ một hệ thống ngang hàng. Lưu ý rằng xác thực có thể được truyền từ hệ thống này sang hệ thống khác. Kết quả, theo như DBMS có liên quan, là một mã định danh ủy quyền. Xác thực không cung cấp bất kỳ đặc quyền nào cho các tác vụ cụ thể. Nó chỉ thiết lập rằng DBMS tin tưởng rằng người dùng là chính họ đã tuyên bố là ai và rằng người dùng tin tưởng rằng DBMS cũng là hệ thống dự kiến. Xác thực là điều kiện tiên quyết để ủy quyền.

Ủy quyền

Ủy quyền liên quan đến các quyền được cấp cho người dùng được ủy quyền để thực hiện các giao dịch cụ thể và do đó thay đổi trạng thái của cơ sở dữ liệu (giao dịch writeitem) và / hoặc nhận dữ liệu từ cơ sở dữ liệu (giao dịch mục đọc). Kết quả của ủy quyền, cần phải trên cơ sở giao dịch, là một vectơ: Ủy quyền (item, auth-id, operation). Vectơ là một chuỗi các giá trị dữ liệu tại một vị trí đã biết trong hệ thống. Việc này có hiệu lực như thế nào là do chức năng của DBMS. Ở cấp độ logic, cấu trúc hệ thống cần một máy chủ ủy quyền, máy chủ này cần hợp tác với một máy chủ kiểm toán. Đã xảy ra sự cố về bảo mật giữa máy chủ và máy chủ và sự cố khuếch đại khi quyền được truyền từ hệ thống này sang hệ thống khác. Khuếch đại ở đây có nghĩa là các vấn đề bảo mật trở nên lớn hơn khi số lượng lớn hơn các máy chủ DBMS tham gia vào giao dịch. Các yêu cầu kiểm toán thường được thực hiện kém. Để an toàn, bạn cần ghi nhật ký tất cả các quyền truy cập và ghi lại tất cả các chi tiết ủy quyền bằng số nhận dạng giao dịch. Cần phải đánh giá thường xuyên và duy trì một lộ trình đánh giá, thường là trong một thời gian dài.

Tiếp cận các triết lý và quản lý

Kiểm soát tùy ý là nơi các đặc quyền cụ thể được chỉ định trên cơ sở các nội dung cụ thể mà người dùng được ủy quyền được phép sử dụng theo một cách cụ thể. DBMS bảo mật phải xây dựng một ma trận truy cập bao gồm các đối tượng như quan hệ, bản ghi, dạng xem và hoạt động cho mỗi người dùng - mỗi mục nhập tách biệt các đặc quyền tạo, đọc, chèn và cập nhật. Ma trận này trở nên rất phức tạp vì quyền hạn sẽ khác nhau giữa các đối tượng. Ma trận cũng có thể trở nên rất lớn, do đó việc triển khai nó thường yêu cầu các loại thực hiện vật lý 8 liên quan đến ma trận thưa thớt. Có thể không lưu trữ được ma trận trong bộ nhớ chính của máy tính

1.2 Cấu trúc của cơ sở dữ liệu

Cấu trúc cơ sở dữ liệu: các khối xây dựng của cơ sở dữ liệu
Bước tiếp theo là trình bày trực quan cơ sở dữ liệu của bạn. Để làm được điều đó, bạn cần hiểu chính xác cơ sở dữ liệu quan hệ được cấu trúc như thế nào.
Trong cơ sở dữ liệu, dữ liệu liên quan được nhóm thành các bảng, mỗi bảng bao gồm các hàng (còn được gọi là bộ dữ liệu) và các cột, giống như một bảng tính.
Để chuyển đổi danh sách dữ liệu của bạn thành bảng, hãy bắt đầu bằng cách tạo bảng cho từng loại thực thể, chẳng hạn như sản phẩm, bán hàng, khách hàng và đơn đặt hàng.
Đây là một ví dụ:
Mỗi hàng của bảng được gọi là một bản ghi. Hồ sơ bao gồm dữ liệu về một cái gì đó hoặc một người nào đó, chẳng hạn như một khách hàng cụ thể. Ngược lại, các cột (còn được gọi là trường hoặc thuộc tính) chứa một loại thông tin duy nhất xuất hiện trong mỗi bản ghi, chẳng hạn như địa chỉ của tất cả các khách hàng được liệt kê trong bảng.

Tên	Họ	Tuổi	ZIP Code
Roger	Williams	43	34760
Jerrica	Jorgensen	32	97453
Samantha	Hopkins	56	64829

Bảng 5: Cơ sở dữ liệu

Để giữ cho dữ liệu nhất quán từ bản ghi này sang bản ghi tiếp theo, hãy gán kiểu dữ liệu thích hợp cho mỗi cột. Các kiểu dữ liệu phổ biến bao gồm:

CHAR - độ dài cụ thể của văn bản
VARCHAR - văn bản có độ dài thay đổi
TEXT - lượng lớn văn bản
INT - số nguyên dương hoặc âm
FLOAT, DOUBLE - cũng có thể lưu trữ số dấu phẩy động
BLOB - dữ liệu nhị phân

Một số hệ thống quản lý cơ sở dữ liệu cũng cung cấp kiểu dữ liệu số tự động , tự động tạo một số duy nhất trong mỗi hàng.

Với mục đích tạo tổng quan trực quan về cơ sở dữ liệu, được gọi là sơ đồ mối quan hệ thực thể, bạn sẽ không bao gồm các bảng thực tế. Thay vào đó, mỗi bảng trở thành một hộp trong sơ đồ. Tiêu đề của mỗi hộp phải cho biết dữ liệu trong bảng đó mô tả những gì, trong khi các thuộc tính được liệt kê bên dưới, như sau:

Cuối cùng, bạn nên quyết định thuộc tính nào hoặc các thuộc tính nào sẽ đóng vai trò là khóa chính cho mỗi bảng, nếu có. Khóa chính (PK) là một mã định danh duy nhất cho một thực thể nhất định, có nghĩa là bạn có thể chọn ra một khách hàng chính xác ngay cả khi bạn chỉ biết giá trị đó.

Các thuộc tính được chọn làm khóa chính phải là duy nhất, không thay đổi và luôn hiện diện (không bao giờ NULL hoặc trống). Vì lý do này, số thứ tự và tên người dùng tạo nên khóa chính tốt, trong khi số điện thoại hoặc địa chỉ đường phố thì không. Bạn cũng có thể sử dụng nhiều trường kết hợp làm khóa chính (đây được gọi là khóa tổng hợp).

Khi đến lúc tạo cơ sở dữ liệu thực tế, bạn sẽ đặt cả cấu trúc dữ liệu logic và cấu trúc dữ liệu vật lý vào ngôn ngữ định nghĩa dữ liệu được hỗ trợ bởi hệ thống quản lý cơ sở dữ liệu của bạn. Tại thời điểm đó, bạn cũng nên ước tính kích thước của cơ sở dữ liệu để đảm bảo rằng bạn có thể nhận được mức hiệu suất và không gian lưu trữ mà nó sẽ yêu cầu

1.2.1. Giới thiệu về mô hình cơ sở dữ liệu quan hệ

Mô hình quan hệ (RM) biểu diễn cơ sở dữ liệu như một tập hợp các quan hệ. Một quan hệ không là gì ngoài một bảng các giá trị. Mỗi hàng trong bảng đại diện cho một tập hợp các giá trị dữ liệu có liên quan. Các hàng này trong bảng biểu thị một thực thể hoặc mối quan hệ trong thế giới thực.

Tên bảng và tên cột rất hữu ích để giải thích ý nghĩa của các giá trị trong mỗi hàng. Dữ liệu được biểu diễn dưới dạng một tập hợp các quan hệ. Trong mô hình quan hệ, dữ liệu được lưu trữ dưới dạng bảng. Tuy nhiên, việc lưu trữ vật lý của dữ liệu độc lập với cách dữ liệu được tổ chức hợp lý.

Cách dễ nhất để hiểu cơ sở dữ liệu là một tập hợp các tệp liên quan. Hãy tưởng tượng một tệp (giấy hoặc kỹ thuật số) các đơn đặt hàng trong một cửa hàng. Sau đó, có một tệp sản phẩm khác, chứa hồ sơ kho. Để hoàn thành đơn đặt hàng, bạn cần phải tra cứu sản phẩm trong tệp đơn đặt hàng, sau đó tra cứu và điều chỉnh mức tồn kho cho sản phẩm cụ thể đó trong tệp sản phẩm. Cơ sở dữ liệu và phần mềm điều khiển cơ sở dữ liệu, được gọi là hệ quản trị cơ sở dữ liệu (DBMS), giúp thực hiện loại nhiệm vụ này

1.2.3. Tìm hiểu Sơ đồ mối quan hệ thực thể để thiết kế cơ sở dữ liệu và tài liệu

Thực thể

Các thực thể là các yếu tố thế giới thực trong hệ thống của bạn. Bạn có thể gọi chúng là danh từ của cơ sở dữ liệu của bạn. ERD hiển thị các thực thể dưới dạng hình chữ nhật:

Ví dụ: nếu bạn đang thiết kế cơ sở dữ liệu cho một cửa hàng trực tuyến, thì các thực thể là sản phẩm tạo nên khoảng không quảng cáo. Các thực thể cốt lõi khác trong cơ sở dữ liệu cửa hàng của bạn sẽ là người dùng và đơn đặt hàng.

Các mối quan hệ

Mối quan hệ là các động từ trong ERD của bạn và mô tả cách các thực thể được liên kết với nhau. ERD hiển thị các mối quan hệ dưới dạng một viên kim cương được gắn nhãn trên các đường kết nối các thực thể:
Cơ sở dữ liệu cửa hàng trực tuyến có một loại mối quan hệ giữa sản phẩm và đơn đặt hàng và một loại mối quan hệ hơi khác giữa người dùng và đơn đặt hàng.

Thuộc tính

Thuộc tính là thuộc tính hoặc đặc điểm của thực thể. Bạn có thể coi chúng như những tính từ mô tả các thực thể trong cơ sở dữ liệu của mình. ERD hiển thị các thuộc tính dưới dạng hình bầu dục được kết nối với thực thể có liên quan:
Các thực thể trong cơ sở dữ liệu cửa hàng trực tuyến của bạn sẽ có rất nhiều thuộc tính. Chỉ liệt kê một số:

Sản phẩm - tên, giá cả và mô tả
Người dùng - tên, mật khẩu, địa chỉ và địa chỉ email
Đơn đặt hàng - số lượng mặt hàng, ngày tháng, tổng số tiền

2. Xử lý cơ sở dữ liệu ở cấp độ máy chủ

2.1. Thao tác cơ sở dữ liệu

2.1.1. Tạo cơ sở dữ liệu trên máy chủ cơ sở dữ liệu

Bạn tạo một máy chủ cơ sở dữ liệu bằng cách đặt các thuộc tính bắt buộc của máy chủ cơ sở dữ liệu và sau đó khởi động máy chủ cơ sở dữ liệu.
Để tạo một máy chủ cơ sở dữ liệu:
Định cấu hình các thuộc tính bắt buộc của máy chủ cơ sở dữ liệu.
Đặt thông số cấu hình trong tệp cấu hình trên
Thêm thông tin kết nối trong tệp sqlhosts và các tệp kết nối khác.
Đặt các biến môi trường trong môi trường của bạn.
Mẹo: Trên hệ điều hành Windows, bạn có thể sử dụng Trình quản lý phiên bản máy chủ để định cấu hình các thuộc tính bắt buộc của máy chủ cơ sở dữ liệu thay vì chỉnh sửa cấu hình tệp trên và sqlhosts và thiết lập các biến môi trường.

2.1.2. Tạo chính xác mục nhập bảng và cơ sở dữ liệu cho cơ sở dữ liệu quan hệ

Chúng ta có thể tạo, đọc, cập nhật và xóa (các chức năng cơ bản của bất kỳ cơ sở dữ liệu nào) thông tin trong cơ sở dữ liệu quan hệ của mình bằng Hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS).

2.1.3. Làm quen với lệnh cơ bản để thao tác với cơ sở dữ liệu

Tất cả tài liệu hướng dẫn được nêu bên dưới mô tả một lệnh và các tham số có sẵn của nó và cung cấp một mẫu tài liệu hoặc nguyên mẫu cho mỗi lệnh. Một số tài liệu lệnh cũng bao gồm các trình trợ giúp mongosh có liên quan.
To run a command against the current database, use db.runCommand():

```
db.runCommand( { <command>
} )
```

To run an administrative command against the admin database, use db.adminCommand():

```
db.adminCommand( { <command>
} )
```

GHI CHÚ

Để biết chi tiết về các lệnh cụ thể, bao gồm cú pháp và ví dụ, hãy nhấp vào lệnh cụ thể để chuyển đến trang tham chiếu của lệnh đó.

2.1.4. Đặc biệt các lệnh nối để tham gia bảng trong cơ sở dữ liệu

Chúng tôi đã trình bày nhiều nội dung trong chương này, từ việc khám phá cách các phép nối hoạt động ở cấp độ khái niệm, thông qua việc làm việc với các loại phép nối khác nhau và cuối cùng là các kỹ thuật hữu ích như răng cưa và truy vấn con.

Một trong những điều quan trọng nhất cần nhớ về cách các phép nối hoạt động là chúng ta đặt điều kiện so sánh một giá trị từ bảng đầu tiên (thường là khóa chính) với một giá trị từ bảng thứ hai (thường là khóa ngoại). Nếu điều kiện sử dụng hai giá trị này được đánh giá là true, thì hàng chứa giá trị đầu tiên được nối với hàng chứa giá trị thứ hai.

Hãy nhanh chóng tóm tắt lại một số loại phép nối khác nhau mà chúng ta có thể sử dụng:

Join Type	Notes
INNER	Kết hợp các hàng từ hai bảng bất cứ khi nào điều kiện nối được đáp ứng.
LEFT	Tương tự như một phép nối bên trong, ngoại trừ các hàng từ bảng đầu tiên được thêm vào bảng nối, bất kể việc đánh giá điều kiện nối.
RIGHT	Tương tự như một phép nối bên trong, ngoại trừ các hàng từ bảng thứ hai được thêm vào bảng nối, bất kể việc đánh giá điều kiện nối.
FULL	Sự kết hợp của phép nối trái và phép nối phải.
CROSS	Không sử dụng điều kiện tham gia. Bảng nối là kết quả của việc so khớp mọi hàng từ bảng đầu tiên với bảng thứ hai, tích chéo của tất cả các hàng trên cả hai bảng.

Bảng 6: Chức năng tham gia

Khi sử dụng phép nối, đôi khi các truy vấn của chúng ta có thể khó sử dụng, đặc biệt là khi chúng ta đang xử lý 2 hoặc nhiều JOIN. Để quản lý điều này tốt hơn, chúng tôi có thể đặt bí danh tên bảng và tên cột để rút ngắn truy vấn của chúng tôi. Chúng tôi cũng có thể sử dụng bí danh để cung cấp thêm ngữ cảnh về kết quả truy vấn.

Cuối cùng, kết quả từ một truy vấn nối đôi khi có thể thu được bằng các phương pháp khác nhau. Truy vấn con cung cấp một phương pháp khác để chúng ta truy vấn cơ sở dữ liệu và lấy ra các kết quả tương tự của các kết quả tương tự, như thể chúng ta đã sử dụng mệnh đề JOIN

3. Chương trình người dùng với cơ sở dữ liệu

3.1. Tìm hiểu các lớp và chức năng kết nối các máy chủ cơ sở dữ liệu

Một lớp là một bản thiết kế của một đối tượng. Bạn có thể nghĩ về một lớp như một khái niệm, và đối tượng là hiện thân của khái niệm đó. Bạn cần có một lớp trước khi có thể tạo một đối tượng. Vì vậy, giả sử bạn muốn sử dụng một người trong chương trình của mình. Bạn muốn có thể mô tả người đó và yêu cầu người đó làm điều gì đó. Một lớp được gọi là 'người' sẽ cung cấp bản thiết kế cho một người trông như thế nào và một người có thể làm gì. Để thực sự sử dụng một người trong chương trình của bạn, bạn cần tạo một đối tượng. Bạn sử dụng lớp người để tạo một đối tượng kiểu 'người'. Bây giờ bạn có thể mô tả người này và yêu cầu người đó làm điều gì đó.

Lớp học rất hữu ích trong lập trình. Hãy xem xét ví dụ về trường hợp bạn không muốn chỉ sử dụng một người mà là 100 người. Thay vì mô tả từng chi tiết từ đầu,
bạn có thể sử dụng cùng một lớp người để tạo 100 đối tượng kiểu 'người'. Bạn vẫn phải cung cấp cho mỗi người một cái tên và các thuộc tính khác, nhưng cấu trúc cơ bản của một người trông giống nhau.

Một hàm là một tổ hợp các lệnh được kết hợp để đạt được một số kết quả. Một hàm thường yêu cầu một số đầu vào (được gọi là đối số) và trả về một số kết quả. Ví dụ, hãy xem xét ví dụ về việc lái xe ô tô. Để xác định quãng đường đi được, bạn cần thực hiện một phép tính bằng cách sử dụng quãng đường đã lái và lượng nhiên liệu đã sử dụng. Bạn có thể viết một hàm để thực hiện phép tính này. Các đối số đi vào hàm sẽ là khoảng cách và mức tiêu thụ nhiên liệu, và kết quả sẽ là số dặm. Bất cứ lúc nào bạn muốn xác định quãng đường, bạn chỉ cần gọi hàm để thực hiện phép tính.

3.2 Tạo chương trình người dùng để tạo cơ sở dữ liệu trên máy chủ cơ sở dữ liệu

B1: Tạo cơ sở dữ liệu

Đăng nhập vào Bảng điều khiển OVHcloud của bạn và chọn Web Cloud trong thanh điều hướng trên cùng. Nhấp vào Cơ sở dữ liệu trong thanh dịch vụ ở phía bên trái, sau đó chọn phiên bản SQL có liên quan. Bấm vào tab Cơ sở dữ liệu, sau đó bấm Thêm cơ sở dữ liệu.

Điền vào các trường theo các tiêu chí được liệt kê. Bạn có thể tạo người dùng trực tiếp bằng cách đánh dấu vào hộp Tạo người dùng.

Tên cơ sở dữ liệu (bắt buộc): đây sẽ là tên cơ sở dữ liệu của bạn.

Tên người dùng: Đây là tên của người dùng có thể đăng nhập vào cơ sở dữ liệu của bạn và thực hiện các yêu cầu (chỉ áp dụng nếu hộp Tạo người dùng được chọn).

Quyền (chỉ khi hộp được đánh dấu): các quyền sẽ được liên kết với người dùng trên cơ sở dữ liệu. Để sử dụng tiêu chuẩn, hãy chọn Quản trị viên. Các quyền có thể được sửa đổi như sau.

Mật khẩu / Xác nhận mật khẩu (chỉ khi hộp được đánh dấu): nhập mật khẩu, sau đó xác nhận.

Cuối cùng, bấm Xác nhận.

B2: Thêm người dùng

Để sử dụng máy chủ cơ sở dữ liệu OVHcloud, bạn cần tạo người dùng có các quyền cụ thể để kết nối với cơ sở dữ liệu.

Đăng nhập vào Bảng điều khiển OVHcloud của bạn và chọn Web Cloud trong thanh điều hướng trên cùng. Nhấp vào Cơ sở dữ liệu trong thanh dịch vụ ở phía bên trái, sau đó chọn tên cơ sở dữ liệu liên quan. Tiếp theo, chuyển sang tab Người dùng và quyền và nhấp vào Thêm người dùng.

Nhập "tên người dùng" và "mật khẩu", sau đó nhấp vào Xác nhận.

3.3. Thực hiện chương trình người dùng để lưu trữ và đọc trên máy chủ cơ sở dữ liệu

Quản lý quyền của người dùng

Để cho phép người dùng thực hiện các hành động trên cơ sở dữ liệu, cần phải gán quyền cho người dùng.

Đăng nhập vào Bảng điều khiển OVHcloud của bạn và chọn Web Cloud trong thanh điều hướng trên cùng. Nhấp vào Cơ sở dữ liệu trong thanh dịch vụ ở phía bên trái, sau đó chọn tên cơ sở dữ liệu liên quan. Tiếp theo, chuyển sang tab Người dùng và quyền. Nhấp vào nút ... ở bên phải của người dùng có liên quan, sau đó nhấp vào Quản lý quyền.

Trong cột bên trái, Cơ sở dữ liệu, bạn sẽ thấy danh sách các cơ sở dữ liệu trên máy chủ cơ sở dữ liệu của mình.

3 loại quyền được đề xuất được mô tả dưới đây:

Quản trị viên: Cấp quyền cho các truy vấn sau: Chọn / Chèn / Cập nhật / Xóa / Tạo / Thay thế / Thả

Đọc / Viết: Cấp quyền cho các truy vấn sau: Chọn / Chèn / Cập nhật / Xóa

Đọc: Cấp phép cho các truy vấn Chọn

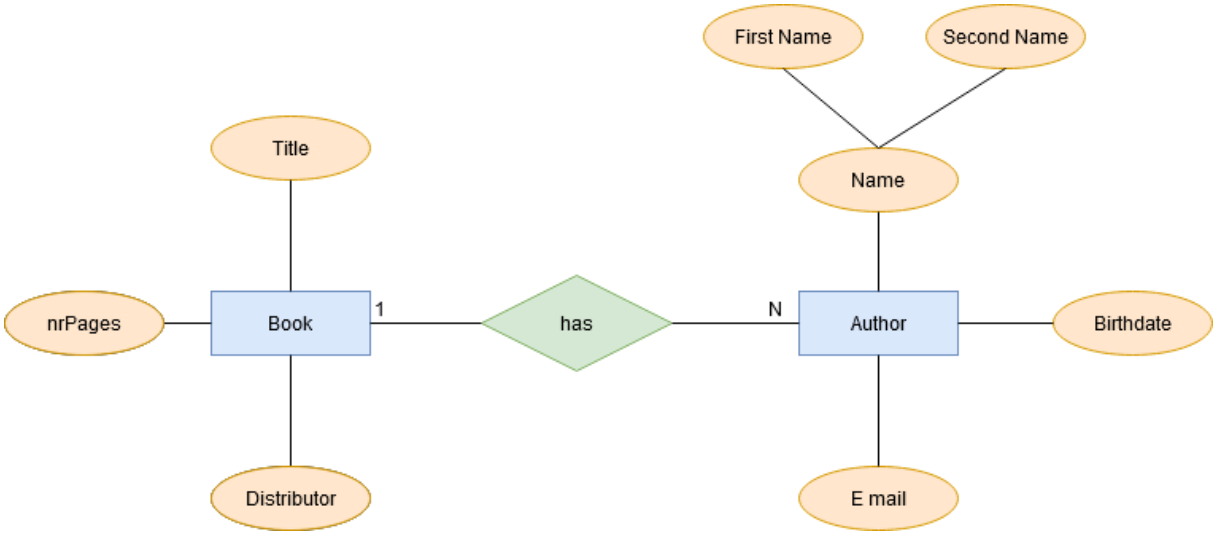
Không: Không có quyền cơ sở dữ liệu

Việc phân phối các quyền được đề cập ở trên là duy nhất đối với OVHcloud. Điều này sẽ cho phép người dùng có quyền Quản trị viên sử dụng DLL (Data_Definition_Language) và DML (Data_Manipulation_Language), trong khi người dùng có quyền Đọc / Viết sẽ chỉ sử dụng DML.

3.4 Chương trình người dùng cơ sở dữ liệu

Yêu cầu 1: Sách và tác giả

Dưới đây là mô hình ERM của hệ thống quản lý sách. Nhiệm vụ của bạn là tạo Mô hình dữ liệu quan hệ bao gồm tất cả các thuộc tính cần thiết. Đồng thời đánh dấu khóa chính và khóa ngoại.



Hình 69: Mô hình ERM của hệ thống quản lý sách

Mô hình dữ liệu quan hệ::

…… ……

Yêu cầu: Nghệ thuật

Bạn được giao nhiệm vụ thiết kế một cơ sở dữ liệu trong đó các tác phẩm nghệ thuật quan trọng nhất và vị trí của chúng có thể được quản lý. Trước đây, tất cả thông tin được lưu trữ trong một bảng duy nhất. Thiết kế bảng không thỏa mãn bất kỳ yêu cầu nào cần thiết để sử dụng phần mềm cơ sở dữ liệu quan hệ một cách an toàn. Một bảng mẫu được liệt kê dưới đây.

- Trong bước đầu tiên, một sơ đồ quan hệ thực thể sẽ được vẽ để hiển thị tất cả các thực thể và các quan hệ của chúng.
- Trong bước thứ hai, bảng sẽ được đưa về dạng N1.

Bảng:

Location	Country	Artist	Title
Prado, Madrid	Spain	Peer Paul Rubens	Die drei Grazien
Louvre, Paris	France	Leonardo Da Vinci	Mona Lisa
Museum of Modern Art, New York City	USA	Vincent van Gogh	'The Starry Night'
Upper Belvedere museum, Vienna	Austria	Gustav Klimt	'The Kiss'
Mauritshuis, The Hague	Netherlands	Johannes Vermeer	'Girl With a Pearl Earring'

Bảng 7: Mẫu cơ sở dữ liệu tác phẩm nghệ thuật

ERM

……

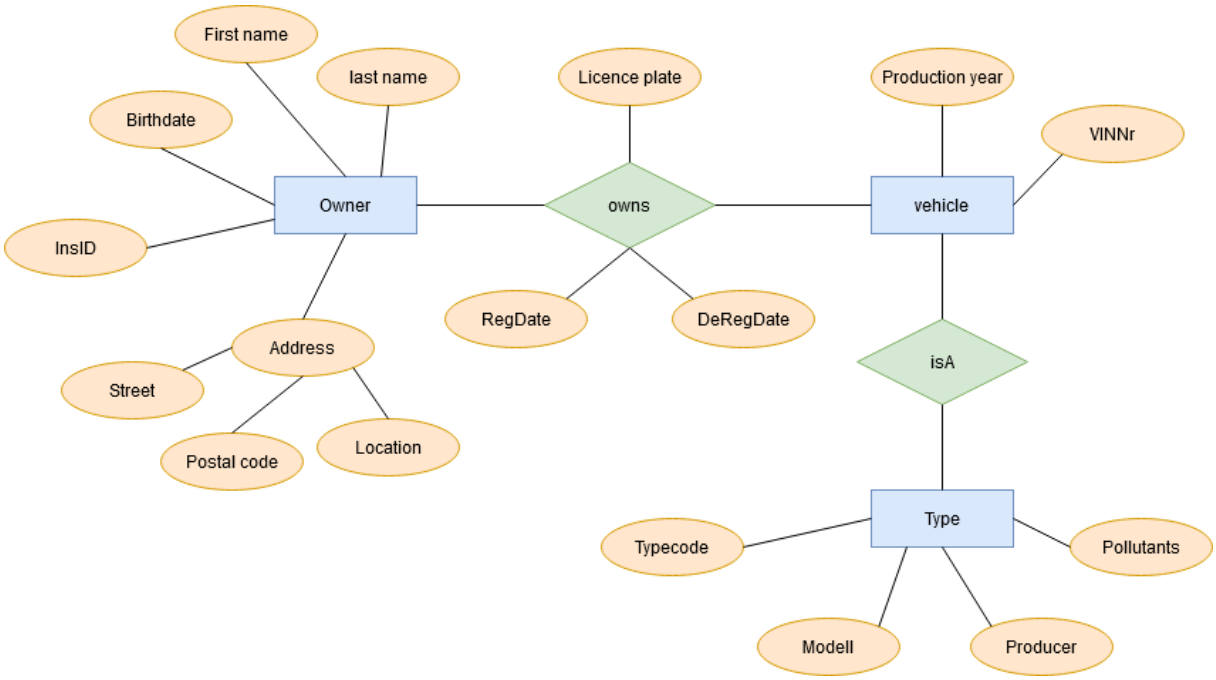
Bảng dạng N1:

……

Yêu cầu: ô tô

Một công ty bảo hiểm xe hơi có kế hoạch tạo ra một hệ thống theo dõi khách hàng dựa trên cơ sở dữ liệu. Cho đến nay, dữ liệu khách hàng được lưu trữ mà không cần xem xét các mô hình cơ sở dữ liệu quan hệ. Đầu tiên, bạn đã tạo một mô hình ER để thể hiện quan hệ của tất cả các thuộc tính và thực thể. Giải thích cho khách hàng về thuật ngữ, thực thể, quan hệ và thuộc tính. Giải thích cho khách hàng cách hoạt động của ERM

ERM



Hình 70: Mô hình ERM cho thấy tất cả các mối quan hệ

Giải thích cho Khách hàng:

……

Bảng Xe ô-tô dạng N1

Yêu cầu

Khách hàng yêu cầu một cơ sở dữ liệu thỏa mãn tất cả các yêu cầu của dạng N1. Sau đó, khách hàng muốn biết dạng N1 là gì và nó sẽ mang lại lợi ích như thế nào cho khách hàng.

Mô hình cơ sở dữ liệu quan hệ dạng N1:

……..

Trả lời câu hỏi:

……..

Bảng Xe ô-tô dạng N2

Yêu cầu:

Với tư cách là kỹ sư cơ sở dữ liệu có kinh nghiệm, bạn đã thuyết phục khách hàng đưa cơ sở dữ liệu sang dạng chuẩn cao cấp hơn. Giải thích cho khách hàng các bước cần thiết và sự khác biệt so với dạng N1 nếu cần.

Mô hình cơ sở dữ liệu quan hệ:

……

Sự khác biệt của dạng N2 so với N1

……..

Bảng Xe ô-tô dạng N3

Yêu cầu:

Trong bước cuối cùng, bạn quyết định đưa cơ sở dữ liệu sang dạng N3. Giải thích cho khách hàng tại sao việc này lại cần thiết. Cho khách hàng xem mô hình cơ sở dữ liệu quan hệ đã hoàn thiện.

Mô hình cơ sở dữ liệu quan hệ:

……

Sự khác biệt của dạng N3 so với N2

……

Hình minh họa 6: Lưu trữ dữ liệu đám mây và máy chủ đám mây lưu trữ dữ liệu an toàn

Dự án hệ thống cơ sở dữ liệu: Thu thập dữ liệu trong thiết lập I4.0

Giới thiệu

Tất cả các ứng dụng I4.0 đều dựa một sản phẩm cơ sở mà đó cũng là dữ liệu. Dữ liệu sản xuất có giá trị nhất đối với các công ty để tối ưu hóa và đo lường chất lượng. Tuy nhiên, các công ty trên khắp thế giới vẫn còn tồn tại các nhà máy sản xuất kinh điển không có hoặc chỉ có khả năng thu thập dữ liệu độc quyền. Một trong những thách thức lớn trong tương lai là xác định và đo lường dữ liệu sản xuất và môi trường và lưu trữ chúng. Đối với các công ty, để tạo ra "dữ liệu" sản phẩm mới và nâng sản phẩm của họ lên tầm I4.0, bắt buộc phải tạo ra nhận thức và kỹ năng đo lường các giá trị vật lý quan trọng và gửi chúng qua mạng đến các trạm lưu trữ. Hệ thống cơ sở dữ liệu quan hệ giúp lưu trữ một lượng lớn dữ liệu an toàn và hiệu quả. Hơn nữa, chúng cho phép một giải pháp dễ dàng truy cập và duy trì. Trong dự án này, một kịch bản thu thập và lưu trữ dữ liệu quy mô đầy đủ bằng cơ sở dữ liệu sẽ được triển khai.

Yêu cầu:

- Có kinh nghiệm lập trình hướng đối tượng
- UML
  - Sơ đồ Use case
  - Sơ đồ class
  - Sơ đồ trình tự
- Lưu đồ
- Kiểm thử phần mềm
- Lập trình vi điều khiển
  - Điều khiển phần cứng
  - Đọc vào cảm biến
  - Tạo các chương trình đo
- Các phương thức giao tiếp trong mạng máy tính
- Đã hoàn thành phần lý thuyết và thực hành của khóa học LC3

Thông tin thêm - dành cho giáo viên

Nếu các trạm quá trình khác được sử dụng thì cần điều chỉnh mô tả phù hợp với hệ thống hiện có của bạn. Dự án này được viết theo cách chung chung độc lập về phần cứng về các điều kiện hoặc các cảm biến được sử dụng. Một số bài tập sẽ đưa ra các nhiệm vụ để xác định các thông số cảm biến từ bảng dữ liệu. Chúng cần được điều chỉnh cho phù hợp với phần cứng được sử dụng. Hơn nữa, chỉ sử dụng các cảm biến khi bạn cũng có thể cung cấp một bảng dữ liệu.

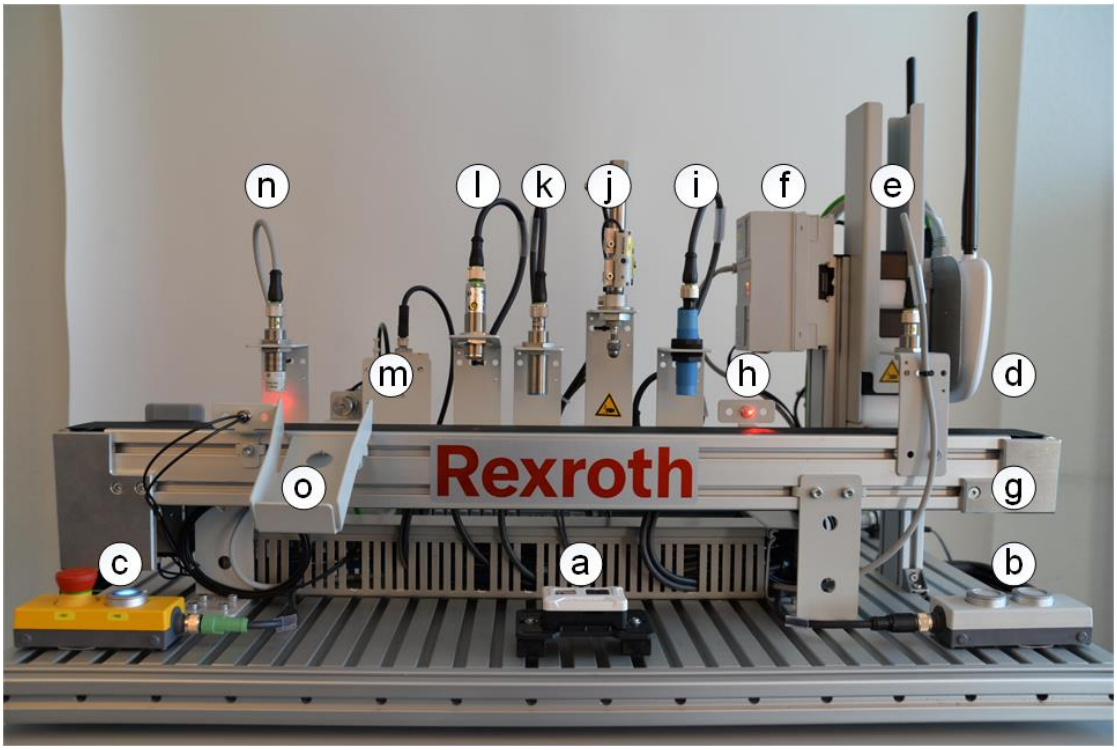
Thiết lập cơ bản

Có hai trạm sản xuất kinh điển. Cả hai trạm sản xuất đều có các chức năng và hoạt động mà không cần các cách thu thập dữ liệu hiện đại. Ngày nay, dữ liệu sản xuất có giá trị

lớn đối với mọi công ty và đều có tiềm năng tối ưu hóa. Nhiệm vụ của bạn là nâng cấp các trạm sản xuất kinh điển sang thiết lập I4.0. Vì việc lựa chọn các cảm biến phù hợp, tạo các chương trình đo và gửi dữ liệu qua mạng máy tính đến các trạm máy chủ và lưu trữ các giá trị đo được là cơ sở của I4.0, trọng tâm chính của dự án này sẽ là thiết lập một khung cơ sở chức năng. Trong dự án này, toàn bộ quy trình công việc từ xác định hệ thống và yêu cầu cho đến triển khai và kiểm thử thực tế sẽ được thực hiện. Thiết lập Cơ bản sẽ là mở rộng một trạm quá trình hiện có bằng cách đo các giá trị vật lý. Một tổ hợp các cảm biến sẽ được sử dụng để đếm số lượng hàng hóa được sản xuất. Tiếp theo, một hệ thống cơ sở dữ liệu sẽ được thiết lập để lưu trữ các giá trị đo được. Một chương trình phía máy chủ sẽ nhận các giá trị thông qua giao tiếp mạng và lưu trữ chúng an toàn trong một cơ sở dữ liệu quan hệ.

CPSi 4.0

Trạm quá trình CPSi được hiển thị bên dưới. Trạm quá trình có thể được điều khiển thông qua vi điều khiển và các nút nhấn hệ thống. Ví dụ này chỉ xem xét điều khiển bằng các nút nhấn hệ thống. Khi nhấn một nút hệ thống, trạm quá trình sẽ bắt đầu bằng cách đẩy một khối lập phương ra khỏi khay (cube magazine). Khối lập phương này được thử nghiệm ở một số trạm. Sau trạm cuối cùng, khối lập phương được đẩy ra bằng xi lanh khí nén hoặc đi qua đó.



Hình 71: Hệ thống đào tạo CPS i4.0 (nhìn từ phía trước)

a	Vi điều khiển (XDK)	i	Cảm biến điện dung
b	Các nút hệ thống	j	Cảm biến vị trí (trên xi lanh viền)
c	Nút dừng khẩn cấp	k	Cảm biến cảm ứng
d	Bộ định tuyến WLAN (Router)	l	Cản sáng

e	Khay	m	Xi lanh khí nén và van điều hướng 5/2
f	XM22 (PLC)	n	Ăng ten RFID
g	Băng chuyền	o	Dốc nhận (collecting ramp)
h	Dây dẫn ánh sáng		

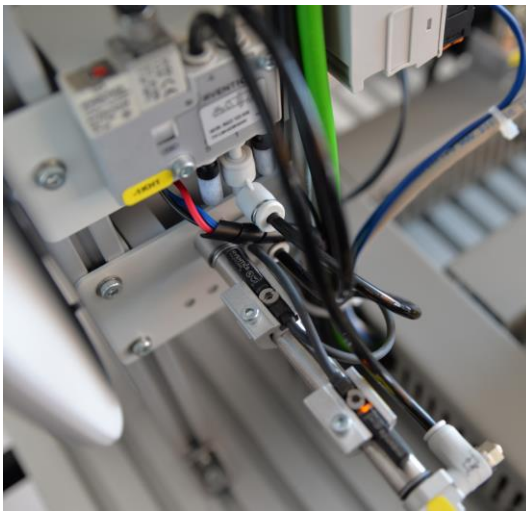
Mô tả hệ thống

Trước khi thử nghiệm, các khối lập phương được lưu trữ trong một khay (hình 8 - 0). Khay được nạp đầy thủ công với các khối lập phương mới. Các nút có thể được sử dụng để chọn loại khối lập phương và bắt đầu hoạt động. Tại độ cao của khối lập phương thứ hai trong ổ, có một công tắc cơ học. Công tắc khởi động khi không có khối lập phương nào ấn vào nó (NC - thường đóng). Với điều kiện:

Hình 72: Các khối lập phương ở dưới ăng-ten RFID lúc bắt đầu

- có ít nhất 2 khối lập phương trong khay
- nút dừng khẩn cấp được nhả,
- dây dẫn ánh sáng không bị cản trở,
- băng chuyền đứng yên,
- tất cả các xi lanh ở vị trí ban đầu,

... van vận hành khay.

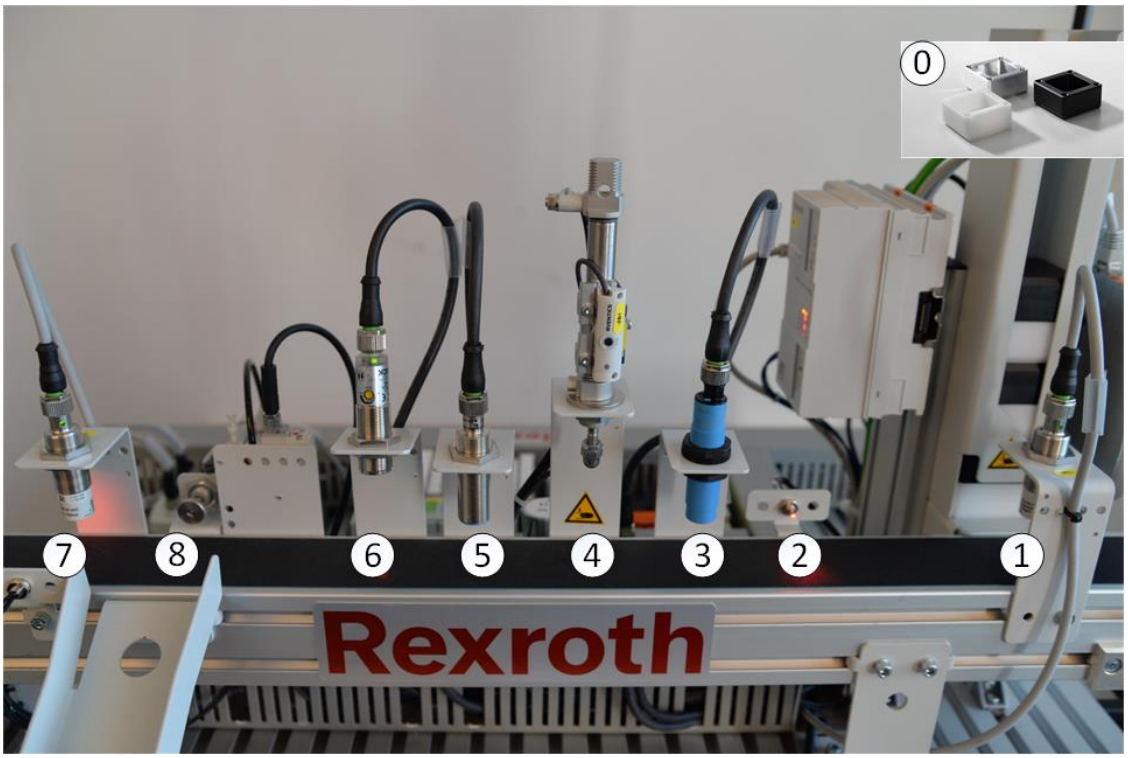


Hình 73: Xi lanh khay với hai công tắc lưỡi gà

Van nằm ở mặt sau của khay cùng với một xi lanh khí nén. Các xi lanh được vận hành bằng khí nén với áp suất ít nhất 1 bar. Khí nén được phân phối đến các van bằng hệ thống phân phối khí nén (hình 2 - f). Tùy thuộc vào vị trí van mà các van cấp khí nén vào các buồng xi lanh. Việc cung cấp khí nén cho các buồng xi lanh có thể được điều chỉnh thông qua van tiết lưu tại các điểm đầu vào. Khi piston được kéo, nó sẽ đẩy một nửa khối lập phương lên băng tải. Vị trí của các xi lanh được phát hiện thông qua các công tắc lưỡi gà. Các công tắc kích hoạt bằng vật liệu từ tính. Có hai công tắc lưỡi gà trên xylanh khay giúp phát hiện vị trí ban đầu và vị trí hoạt động của piston. Một động cơ DC (hình 2 - g) được gắn trên mặt bích ở đầu kia của băng chuyền. Động cơ dẫn động băng chuyền. Chiều quay được xác định bởi hai rơle. Trước khi băng chuyền bắt đầu dịch chuyển, xi lanh khay trở lại vị trí cơ bản và ăng ten

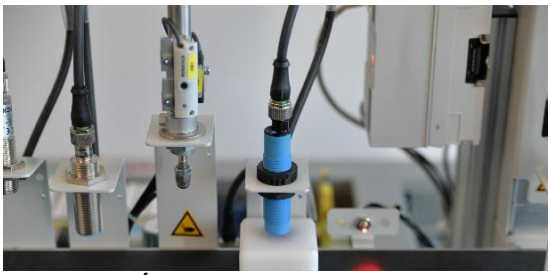
RFID phía trên nửa khối lập phương ghi mã số loại khối lập phương trên thẻ RFID trong khối.

Các khối lập phương chuyển động theo hướng từ phải sang trái. Chúng đi qua một số trạm trên băng chuyền. Tùy thuộc vào quá trình tại mỗi trạm, băng chuyền dừng lại hoặc phôi được đánh giá khi nó đi qua.



Hình 74: Vị trí phôi trong CPSi4.0

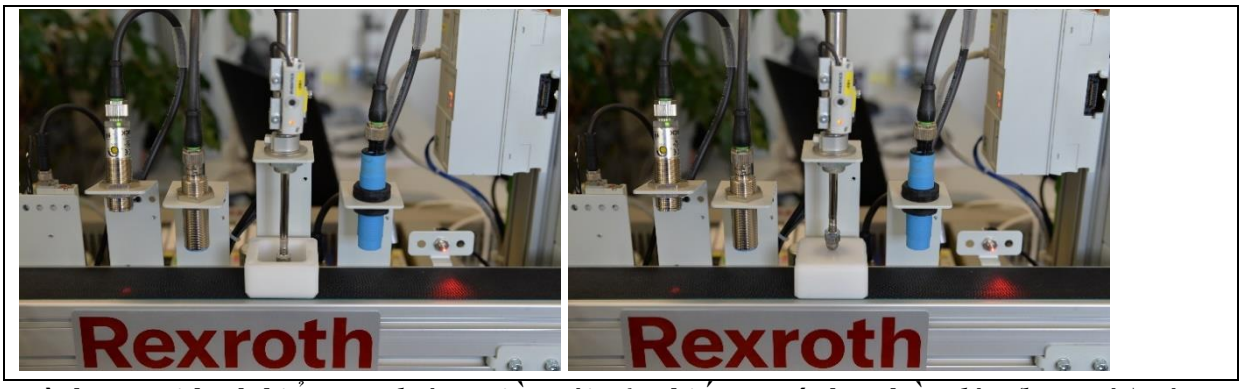
9. **Ăng ten RFID:** Mã số được viết trên thẻ trong khối.
10. **Dây dẫn ánh sáng:** Phát hiện vị trí khối (cạnh)
11. **Cảm biến điện dung:** Phát hiện vị trí khối (trên)
12. **Xi lanh viền:** Piston di chuyển trên phôi và cảm biến vị trí đo khoảng cách được bao phủ - suy luận về hướng (đáy/ nắp)
13. **Cảm biến cảm ứng:** Phát hiện vật liệu kim loại - suy luận về vật liệu
14. **Cản sáng:** Phát hiện các đối tượng sáng - suy luận về màu sắc
15. **Ăng ten RFID:** Mã số được đọc từ thẻ
16. **Xi lanh đẩy:** Đẩy các khối sai ra ngoài



Hình 75: Khối lập phương được phát hiện bởi cảm biến điện dung

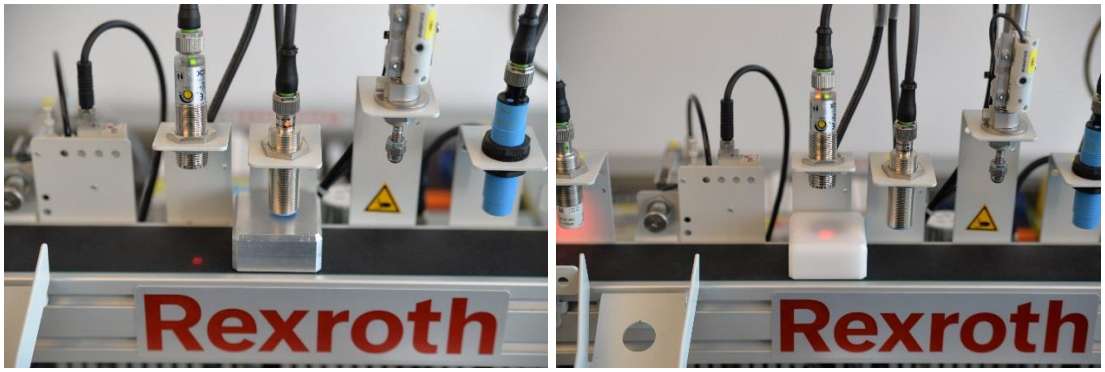
Đầu tiên phôi đi qua dây dẫn ánh sáng và sau đó là cảm biến điện dung. Phôi được phát hiện theo cạnh bên hoặc từ trên xuống. Thông qua phản hồi, chương trình biết thời điểm và vị trí của khối lập phương ở trên băng chuyền. Sau khi các cảm biến được kích hoạt, băng chuyền dừng lại để phôi

được định vị dưới xi lanh viền. Việc phát hiện dư thừa đảm bảo rằng một nửa khối lập phương được định vị chính xác.



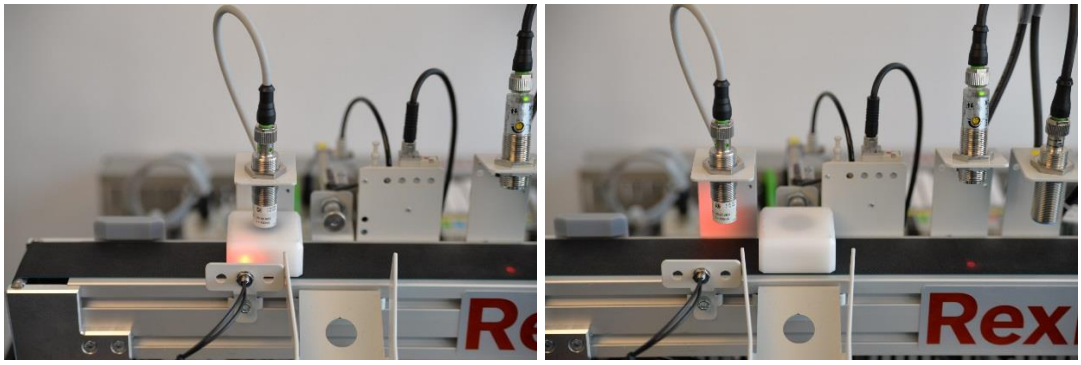
Hình 76: Xi lanh kiểm tra đường viền với cảm biến vị trí cho phần đáy (bên trái) và phần nắp (bên phải)

Nếu băng chuyền dừng ở đây, piston xi lanh sẽ giãn ra. Xi lanh này được điều khiển bởi một van khí nén để điều khiển vòng kín (hình 2 - h). Với phần đáy, xi lanh giãn hoàn toàn. Một công tắc lưỡi gà sẽ phát hiện vị trí này. Với phần nắp, piston ép lên khối lập phương. Quá trình này bị giới hạn về thời gian. Sau khi hết thời gian, piston sẽ rút lại. Khoảng cách được bao phủ được ghi lại bởi một cảm biến vị trí. Cảm biến này cũng được gắn trên vỏ xi lanh. Hướng của phôi được suy ra từ giá trị trả về.



Hình 77: Cảm biến cảm ứng với khối nhôm (trái) và tấm cản sáng với khối nhựa (phải)

Nếu piston ở vị trí ban đầu, băng chuyền tiếp tục chuyển động. Phôi đi qua một cảm biến cảm ứng và một tấm cản sáng. Cảm biến cảm ứng chỉ phản ứng với các vật thể kim loại. Tấm cản sáng phát hiện các vật thể sáng (trắng hoặc bạc). Một đèn LED chiếu sáng ở đầu các cảm biến cho biết rằng các cảm biến đã phát hiện ra điều gì đó. Băng chuyền không phải dừng ở đây. Thông tin về hướng, chất liệu và màu sắc được lưu lại.



Hình 78: Dây dẫn ánh sáng dừng hình khối dưới ăng-ten RFID (trái) và hình khối sai ở vị trí đẩy ra (phải)

Phôi dừng ở dây dẫn ánh sáng thứ hai. Dây dẫn này được đặt đối diện với ăng-ten RFID thứ hai. Mã số trên thẻ (loại hình khối) được so sánh với các thuộc tính đã lưu. Nếu khớp, nửa khối lập phương di chuyển đến dấu màu xám (cuối băng chuyền). Trong trường hợp xảy ra lỗi, băng chuyền sẽ quay ngược trở lại xi lanh đẩy và khối lập phương được đẩy xuống dốc. Nếu có đủ khối trong khay, quá trình này sẽ tự động được lặp lại.

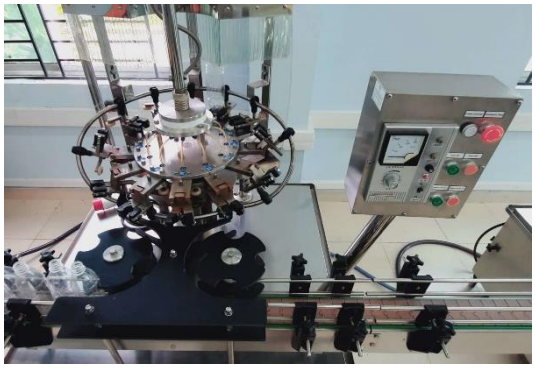
Trạm đóng chai

Trạm đóng chai là một trạm cơ khí thuần túy. Từ trước đến nay không có phương tiện thu thập dữ liệu nào. Trạm đóng chai lấy chai, làm sạch chúng tại trạm làm sạch, đổ đầy tại trạm chiết rót và cuối cùng đặt nắp vào chai tại trạm đóng nắp.

Hình 79: Thiết lập đầy đủ trạm đóng chai, làm sạch (trái), chiết rót (giữa) và đóng nắp (phải)

Sau khi quá trình đóng chai kết thúc, chai mới cần được đưa vào vị trí đầu vào để khởi động lại quy trình.

Trạm làm sạch là quá trình đầu tiên một chai sẽ tham gia vào trong toàn bộ trạm đóng chai. Các chai ở đúng vị trí vào trong trạm sẽ được kẹp chặt đầu một cách cơ học, di chuyển lên trên và làm sạch. Toàn bộ quá trình làm sạch sẽ được tiến hành trong vòng một vòng quay của trạm làm sạch. Sau khi làm sạch xong, chai được đẩy ra ngoài. Không có kiểm tra để xem các chai có được đặt chính xác ở vị trí đầu vào cũng như ở vị trí đầu ra hay không. Do đó, không thể phát hiện các chai bị lật ngược. Điều đó có nghĩa là tất cả các chai phải được đặt chuẩn trên băng chuyền trước khi quá trình bắt đầu



Hình 80: Trạm làm sạch

Hình 81: Trạm chiết rót

Trạm chiết rót sẽ lấy chai một cách cơ học và gắn vào một vị trí đã xác định trước. Nếu một chai được gắn vào, van chiết rót sẽ được mở ra và chất lỏng sẽ được đổ đầy vào bên trong chai. Quá trình chiết rót sẽ tự động được bắt đầu nếu một chai ở vị trí đầu vào và quá trình quay được bắt đầu. Sau khi quá trình quay kết thúc và chai ở vị trí đầu ra của trạm, van sẽ tự động được đóng kín lại. Tại vị trí đầu vào, giả định rằng các chai đã vào tuần tự và được đặt chuẩn. Không có hệ thống kiểm tra hoặc đẩy ra nếu một chai bị rơi.

Sau khi các chai đã được làm sạch và đổ đầy nước, chúng sẽ được chuyển đến trạm cuối cùng, trạm đóng nắp. Tại đây các chai một lần nữa sẽ được chọn tuần tự. Trong một quá trình, chúng sẽ được xoay và một nắp sẽ được gắn ở đầu chai. Sau đó, các chai sẽ được chuyển xuống vị trí đầu ra của trạm. Không có kiểm tra xem nắp đã được đặt đúng trên đầu chai hay chưa. Cũng không có bất kỳ kiểm tra nào để xem các chai đến đúng vị trí ở vị trí đầu vào hay không. Ngoài ra, sau khi các chai đã được đóng nắp, không có kiểm tra xem các chai có được đặt đúng vị trí ở vị trí đầura hay không. Ngoài ra, các nắp cần được nạp thủ công vào chỗ chứa nắp

Hình 82: trạm đóng nắp

Tất cả ba trạm đều có khả năng gắn các cảm biến và đo các giá trị môi trường xung quanh.

Lựa chọn phần cứng

Nhận thức đối tượng có thể đo được thông qua các loại cảm biến khác nhau. Một khả năng là sử dụng cảm biến siêu âm. Các loại cảm biến này được sử dụng để đo khoảng cách. Lợi ích của chúng là có thể sử dụng để phát hiện trên hầu hết các bề mặt. Tùy thuộc vào lĩnh vực ứng dụng, nhược điểm của chúng có thể là bán kính phát hiện có hình nón nên có thể dẫn đến hiện tượng xuyên âm. Ngoài ra, một cảm biến hồng ngoại (IR) sẽ được sử dụng. Lợi ích của chúng là có cơ chế phát hiện hình dạng điểm. Nhược điểm của chúng là một số bề mặt và vật liệu nhất định không phản xạ tín hiệu IR, khiến các cảm biến không thể phát hiện đối tượng.

Xác định các thông số cảm biến

Trong bước đầu tiên, hãy xác định cảm biến siêu âm được cung cấp trong bảng dữ liệu và viết loại của nó ra:

Ultrasonic sensor

Cảm biến siêu âm có một số thông số quan trọng. Tra cứu trong bảng dữ liệu và điền vào các thông số còn thiếu trong bảng. Đảm bảo chỉ ghi các thông số của cảm biến của bạn:

Parameter	Value
maxRange	
minRange	
Trigger impulse duration	

Để đo phạm vi bằng cảm biến siêu âm, cần phải tiến hành theo các bước nhất định. Các bước này được viết trong bảng dữ liệu. Tóm tắt tất cả các bước tại đây:

Phát hiện phạm vi trên cảm biến siêu âm hoạt động thông qua phép đo thời gian. Để chuyển đổi thời gian đo được thành khoảng cách, một phương trình được cung cấp trong bảng dữ liệu. Viết ra phương trình ở đây:

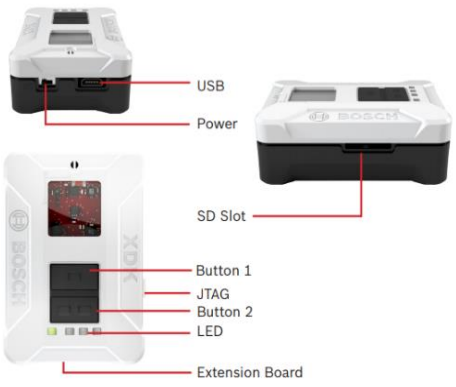
Chương trình đo lường - vi điều khiển cơ bản

Sau khi chọn phần cứng, một chương trình đo đơn giản sẽ được triển khai. Chương trình đo phải đọc vào các giá trị của cảm biến siêu âm và các giá trị đầu vào của cảm biến IR và in chúng trực tiếp trên màn hình console. Trong trường hợp này, chúng ta giả sử cảm biến IR phát tín hiệu nhị phân nếu một đối tượng được phát hiện. Mỗi phép đo sẽ được thực hiện trong một khoảng thời gian 5 giây. Nhập ảnh chụp màn hình console xuất ra trong bảng bên dưới. Tải lên mã của bạn và ảnh chụp màn hình console xuất ra.

Đầu ra yêu cầu sẽ là:
“Ultrasonic sensor: XXXcm”
“IR sensor detection: XXX”
Xuất ra màn hình console

Chương trình đo lường - Bosch XDK

Bosch XDK là một nền tảng vi điều khiển mạnh mẽ với một mảng cảm biến rộng. Nó thường được sử dụng trong các ứng dụng công nghiệp, đặc biệt là để truy vết dữ liệu. Bosch XDK cung cấp một mảng cảm biến rộng, mô-đun wifi tích hợp, đầu đọc thẻ SD để lưu trữ dữ liệu trên thẻ nhớ, các nút nhấn có thể cấu hình tự do và đầu nối USB để kết nối XDK với PC để chạy (flashing) các chương trình mới và xuất trực tiếp các



Hình 83: Ví dụ về Databoads

giá trị trong màn hình console IDE của riêng nó.

Hình 84: So sánh sự khác biệt giữa Báo cáo và Trang tổng quan

Bosch XDK còn cho phép người dùng truy cập nhiều loại cảm biến. Người dùng có thể đo gia tốc, nhiệt độ, độ ẩm, áp suất không khí và nhiều hơn nữa. Hơn nữa, XDK sử dụng "FreeRTOS" làm hệ điều hành thời gian thực. Điều này đảm bảo có thể thực hiện được các phép đo được định thời chính xác với khoảng thời gian xác định.

Ngoài ra, Bosch sử dụng một Eclipse IDE riêng được gọi là XDK-Workbench. IDE này cung cấp cho người dùng lựa chọn để lập trình XDK trực tiếp với C hoặc sử dụng ngôn ngữ lập trình MITA. Ngôn ngữ lập trình MITA được phát triển để giảm bớt gánh nặng khi tạo các ứng dụng I4.0 mà không có kiến thức về lập trình nhúng. Trang web: https://developer.bosch.com/web/xdk/getting-started#1 được cung cấp bởi Bosch và có code mẫu để đọc vào các giá trị cảm biến. Một phần của nhiệm vụ này là đọc qua các mô tả và code mẫu và sử dụng lại chúng để tạo một chương trình làm việc.

Nhiệm vụ của phần này là tạo một chương trình đo đọc vào các giá trị cảm biến của:

- Gia tốc kế
- Cảm biến độ ẩm
- Cảm biến nhiệt độ
- Cảm biến áp suất

Giá trị đọc được sẽ được xuất ra trên màn hình console của XDK - Workbench. Dự án sẽ được lập trình bằng ngôn ngữ lập trình MITA. Các phép đo phải được thực hiện sau mỗi 3 giây. Kết quả xuất ra màn hình console mong muốn sẽ như sau:
Acceleration in X: XXXg
Acceleration in Y: YYYg
Acceleration in Z: ZZZg
Humidity: XXX%
Temperature: XXX°C
Air pressure: XXXkPa
Nhập ảnh chụp màn hình console xuất ra trong bảng bên dưới. Tải lên mã của bạn và ảnh chụp màn hình console xuất ra.
Xuất ra màn hình console

Vị trí đặt cảm biến và hệ thống dây điện

Sau khi các chương trình kiểm thử được thực hiện thành công, các cảm biến có thể được đặt ở vị trí mong muốn. Đối với trạm đóng chai, các cảm biến phải được đặt sao cho cảm biến siêu âm có thể đo xem có chai hay không từ bên cạnh. Cảm biến IR sẽ được sử dụng để đo xem có nắp đậy trên chai hay không. Tìm một vị trí thích hợp để các cảm biến đáp ứng yêu cầu này. Trên trạm CPSi 4.0, quy trình tương tự sẽ được thực hiện. Thay vì chai, vị trí của các khối sẽ được tính toán. Đảm bảo điều chỉnh cảm biến IR để nó có thể phát hiện xem khối có lỗ ở mặt trên hay không. Tìm vị trí thích hợp để đặt cảm biến và ghi lại quyết định của bạn bằng hình ảnh. Sau đó hãy mô tả theo cách của riêng bạn tại sao bạn chọn vị trí này trong bảng dưới đây:

Hình ảnh	Thuyết minh

Giao thức giao tiếp

Sau khi các cảm biến và vi điều khiển có vị trí hợp lệ để thực hiện phép đo, giao thức giao tiếp sẽ được triển khai. Sau khi triển khai một trong số các lời giải, hãy chụp ảnh kết quả xuất ra màn hình console của máy tính và in ra trong trường bên dưới. Tiếp tục tải lên kết quả xuất ra màn hình console và tất cả code.

Vi điều khiển phổ biến

Các cảm biến được mô tả ở phần trước của vi điều khiển sẽ được sử dụng để tạo chương trình đo. Về giao thức giao tiếp, phương thức Publisher/Subcriber bằng giao thức MQTT sẽ được sử dụng. Trong ví dụ này, tất cả dữ liệu đo được sẽ được chuyển đổi thành đối tượng JSON. Đối tượng này sẽ được chuyển qua giao thức MQTT. Do đó, sẽ chỉ có một chủ đề để gửi dữ liệu. Dữ liệu siêu âm sẽ được đặt tên là “Distance”, dữ liệu cảm biến IR sẽ được đặt tên là “Detected”.

Các bước:

11. Thiết lập broker trên máy tính và chạy broker
12. Sử dụng chương trình đo của bạn để vi điều khiển đọc vào các giá trị cảm biến. Để vi điều khiển đọc trong khoảng thời gian 500 giây
13. Lưu tất cả các giá trị đo được trong một đối tượng JSON, chọn tên theo yêu cầu
14. Thêm các thư viện cần thiết vào vi điều khiển và tạo chủ đề (topic) để xuất bản.
    a. Tên chủ đề
        i. Measurement/States
15. Thêm các thư viện cần thiết trên máy tính và đăng ký chủ đề nơi các giá trị cảm biến được xuất bản
16. In các giá trị đã xuất bản trên màn hình console máy tính
17. Kiểm thử các khoảng thời gian cảm biến khác nhau

Xuất ra màn hình console

……..

XDK

XDK cung cấp khả năng truyền dữ liệu qua phương thức Publisher/Subcriber bằng giao thức MQTT. Vì XDK cung cấp nhiều loại cảm biến nên nhiều cảm biến sẽ được truyền đến máy tính. Trong ví dụ này, tất cả dữ liệu đo được sẽ được chuyển đổi thành đối tượng JSON. Đối tượng này sẽ được chuyển qua giao thức MQTT. Do đó, sẽ chỉ có một chủ đề để gửi dữ liệu. Dữ liệu cảm biến phải được đặt tên như sau:

- Đối với gia tốc
    - AccelX
    - AccelY
    - AccelZ
- Đối với độ ẩm
    - Humidity
- Đối với nhiệt độ
    - Temperature
- Đối với áp suất
    - Pressure

Các bước:

7. Thiết lập broker trên máy tính và chạy broker
8. Sử dụng chương trình đo của bạn để vi điều khiển đọc vào các giá trị cảm biến. Để vi điều khiển đọc trong khoảng thời gian 3 giây

9. Thêm các thư viện cần thiết vào vi điều khiển và tạo chủ đề (topic) để xuất bản.
    a. Tên chủ đề
        i. MeasurementXDK/ States
10. Thêm các thư viện cần thiết trên máy tính và đăng ký chủ đề nơi các giá trị cảm biến được xuất bản
11. In các giá trị đã xuất bản trên màn hình console máy tính
12. Xuất ra màn hình console

……

Tạo sơ đồ ERM

Sau khi hệ thống đo lường và giao tiếp được thiết lập, hệ thống cơ sở dữ liệu quan hệ có thể được khởi động. Bước đầu tiên, một sơ đồ ERM sẽ được tạo. Đối với sơ đồ ERM, cần phải mô hình hóa tất cả các bộ phận của hệ thống. Cơ sở dữ liệu phải bao gồm tên trạm, tên cảm biến, loại cảm biến, giá trị cảm biến, loại vi điều khiển và dấu thời gian khi thực hiện phép đo. Tất cả các tên phải bằng tiếng Anh. Vẽ sơ đồ ERM dưới đây:

……..

Thiết lập cơ sở dữ liệu

Sơ đồ ERM là cơ sở của mọi cơ sở dữ liệu. Sau khi đã tạo sơ đồ ERM, hãy tạo cơ sở dữ liệu. Tên cơ sở dữ liệu sẽ là “Bottling Station” hoặc “CSPi4.0” tùy thuộc bạn làm việc với trạm nào. Tất cả các tên được sử dụng phải khớp với sơ đồ ERM. Tất cả các bảng phải có dạng chuẩn N3, nếu cần thiết phải chuẩn hóa các bảng. Tải lên và thêm hình ảnh của tất cả các bảng và hàng đã tạo. Cũng tải hình ảnh lên.

Bảng

……

Chương trình cơ sở dữ liệu

Sau khi cơ sở dữ liệu được thiết lập và chương trình đo hoạt động, có thể lưu trữ dữ liệu đo bên trong cơ sở dữ liệu.

Ghi vào cơ sở dữ liệu

Bất cứ khi nào chương trình nhận được một giá trị thông qua giao thức MQTT, tất cả các giá trị cảm biến nhận được cùng với dấu thời gian nhận hiện tại sẽ được lưu vào cơ sở dữ liệu. Đảm bảo rằng các bảng và giá trị đúng được gửi đến cơ sở dữ liệu. Tiếp theo, hãy đảm bảo rằng dấu thời gian có định dạng đúng để phù hợp với cơ sở dữ liệu đã sử dụng của bạn. Dưới đây là một danh sách nhỏ các lệnh SQL cần thiết. Cũng cần phải cung cấp một thư viện để gửi các lệnh SQL đến cơ sở dữ liệu của bạn.
Kiểm thử thiết lập của bạn bằng cách tạo một chương trình đọc vào các giá trị trong 5 phút và lưu các giá trị đo được vào cơ sở dữ liệu. Trước khi bắt đầu chương trình đo, hãy đảm bảo các trạm đang hoạt động. Tải code lên.

Các lệnh SQL:

Chèn giá trị vào bảng

INSERT INTO *BảngName*
VALUES (*value1*, *value2*, *value3*, ...);

Đọc giá trị hàng từ bảng

SELECT colName FROM BảngName

Đọc từ cơ sở dữ liệu

Sau khi hoàn thành, tải tất cả các giá trị cảm biến được lưu trữ từ cơ sở dữ liệu. Tạo hình ảnh trực quan của các giá trị cảm biến. Đặt mỗi giá trị cảm biến vào một sơ đồ riêng. Chọn tên cảm biến làm tiêu đề. Chèn các sơ đồ bên dưới. Tải code và ảnh chụp sơ đồ lên.

……

BÀI 4: TRỰC QUAN HÓA DỮ LIỆU VỚI DASHBOARDS

Mục tiêu: Sau khi học xong bài học này các học viên có khả năng:
- Hiểu được kiến thức cơ bản về dashboard
- Phân biệt được các loại dashboard, phân biệt được điểm giống và khác nhau của dashboard và báo cáo
- Nắm được các bước để thiết kế một dashboard hoàn chỉnh
- Biết được những điểm giới hạn của một dashboard
- Hiểu được quy trình để tạo nên một dashboard
- Có thể tạo những dashboard đơn giản
- Có thể tạo một chương trình trực quan hóa

Nội dung

1. Kiến thức cơ bản về dashboard

1.1. Xác định cách sử dụng và nhiệm vụ của dashboard

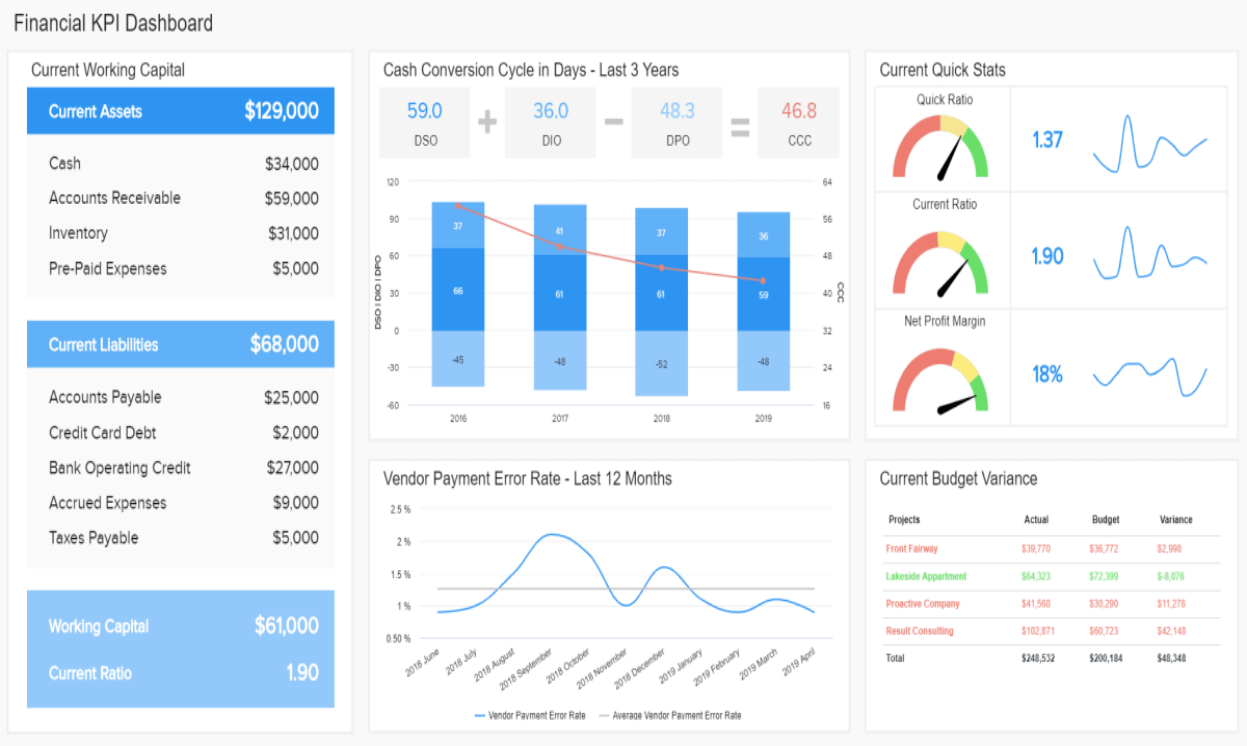
1.1.1. Dashboard là gì?

Bảng điều khiển dữ liệu là một công cụ cung cấp một phương tiện tương tác, tập trung để theo dõi, đo lường, phân tích và trích xuất thông tin chi tiết về doanh nghiệp có liên quan từ các bộ dữ liệu khác nhau trong các lĩnh vực chính đồng thời hiển thị thông tin theo cách tương tác, trực quan và trực quan.

Nó cung cấp cho người dùng một cái nhìn tổng quan toàn diện về các phòng ban, mục tiêu, sáng kiến, quy trình hoặc dự án nội bộ khác nhau của công ty họ. Chúng được đo lường thông qua các chỉ số hiệu suất chính (KPI), cung cấp thông tin chi tiết giúp thúc đẩy tăng trưởng và cải tiến.

Trang tổng quan trực tuyến cung cấp quyền truy cập có thể điều hướng ngay lập tức vào các phân tích có thể hành động có khả năng thúc đẩy lợi nhuận của bạn thông qua

quá trình phát triển thương mại liên tục.



Hình 85: So sánh sự giống nhau giữa Báo cáo và Trang tổng quan

Để xác định đúng trang tổng quan, bạn cần xem xét thực tế rằng, nếu không có sự tồn tại của trang tổng quan và các phương pháp báo cáo trang tổng quan, các doanh nghiệp sẽ cần phải sàng lọc qua những đống dữ liệu phi cấu trúc khổng lồ, vừa không hiệu quả vừa tốn thời gian.

1.1.2. Mục đích của Dashboard là gì?

Như đã đề cập trước đó, Dashboard có khả năng trả lời một loạt các câu hỏi liên quan đến kinh doanh dựa trên các mục tiêu, mục tiêu và chiến lược cụ thể của bạn.

Bằng cách lấy dữ liệu thô từ một số nguồn và hợp nhất trước khi trình bày theo cách trực quan phù hợp, tùy chỉnh, trang tổng quan dữ liệu có thể giúp hiểu rõ về dữ liệu có giá trị nhất của công ty bạn và cho phép bạn tìm câu trả lời hữu ích cho các câu hỏi kinh doanh nhức nhối nhất của bạn.

Thông qua việc liên kết với các KPI cụ thể phù hợp với mục tiêu kinh doanh của mình, bạn có thể đi sâu vào các nhóm thông tin cụ thể, tạo điểm chuẩn và liên tục đo lường thành công của mình.

Khi làm như vậy, doanh nghiệp của bạn sẽ được định hướng dựa trên dữ liệu và kết quả là trực tiếp - thành công hơn. Để tìm hiểu thêm về trang tổng quan và các chỉ số hiệu suất chính, hãy khám phá bộ sưu tập ngày càng mở rộng của chúng tôi về các mẫu và ví dụ KPI thúc đẩy kinh doanh khác nhau.

1.2. Cơ sở lý thuyết để tạo ra các ứng dụng web

Để có thể quan sát thực nghiệm, người ta cần các khái niệm lý thuyết có thể áp dụng được. Chúng tôi đang sử dụng khái niệm thông tin dựa trên các quy trình phụ khác

nhau của thông tin diễn ra trong đời sống xã hội và được hỗ trợ về mặt kỹ thuật bởi CNTT-TT. Đây là các quá trình nhận thức, giao tiếp và hợp tác.

- Các quá trình nhận thức (bao gồm cả các quá trình cảm xúc) là cá nhân, hoặc, trong trường hợp của bất kỳ cơ quan xã hội cá nhân nào được đặt tên là một chủ thể, các quá trình nội bộ chủ quan của việc tạo ra thông tin. .
- Các quá trình giao tiếp có tính tương tác, nghĩa là giữa các cá nhân với nhau hoặc các chủ thể xã hội khác. Giao tiếp qua trung gian máy tính giải quyết các quy trình này được hỗ trợ bởi ICTs
- Các quá trình hợp tác mang tính tích hợp, liên quan đến cấp độ siêu cá nhân và để thông tin xuất hiện từ các tác động tổng hợp của các chủ thể giao tiếp. Ban đầu, Công việc Hợp tác Hỗ trợ Máy tính đã nghiên cứu chủ đề này từ quan điểm về sự tham gia của CNTT-TT.

Ngày nay, cách tiếp cận này tận dụng lợi thế từ nghiên cứu về trí tuệ tập thể, trí tuệ của đám đông, v.v.

1.3. So sánh các phương pháp tạo khác nhau

1.3.1. Báo cáo là gì?

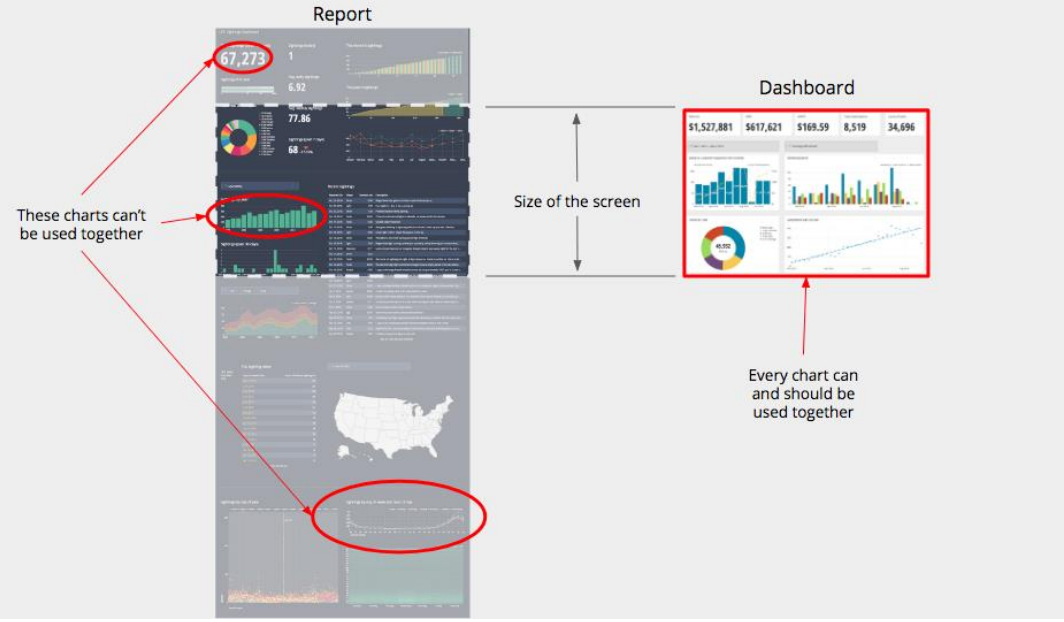
Báo cáo có thể là bản trình bày các biểu đồ tương ứng và các hình ảnh trực quan khác, hoặc chúng có thể là một tập hợp lớn các biểu đồ và hình ảnh trực quan có thể liên quan trực tiếp hoặc không. Báo cáo được sử dụng để thu thập thông tin chi tiết về các hoạt động trong một tổ chức, do đó, báo cáo có thể bao gồm rất rộng phạm vi thông tin liên quan hoặc tập trung hẹp vào chi tiết của một mục, mục đích hoặc sự kiện. Tất cả thông tin này, khi được trình bày trong một báo cáo, có nghĩa là một bản tóm tắt kịp thời.

Trong một nền tảng trực quan hóa dữ liệu, như Biểu đồio, một báo cáo thậm chí có thể được xây dựng trong cùng một môi trường với trang tổng quan, dẫn đến sự nhầm lẫn thậm chí còn nhiều hơn về sự khác biệt giữa báo cáo và trang tổng quan. Báo cáo này thậm chí có thể có giao diện của một trang tổng quan, trong trường hợp này, bạn có thể đang tạo một "trang tổng quan". Báo cáo cũng có thể là một loạt các trang tổng quan có thể liên quan đến nhau và như một tổng thể hiển thị nhiều thông tin cần thiết hơn để hiểu trạng thái của mọi thứ. Các nhóm trang tổng quan này có thể được liên kết trực tiếp hoặc được nhóm lại trong thư viện, phân loại hoặc công nghệ tổ chức của phần mềm.

1.3.2. Báo cáo và Trang tổng quan khác nhau như thế nào?

Đầu tiên, một báo cáo chứa nhiều thông tin chi tiết hơn. Trong trường hợp trang tổng quan có thể chỉ cung cấp cho Giám đốc điều hành thông tin về tiến độ bán hàng của toàn công ty, một báo cáo tương ứng sẽ cung cấp cho Giám đốc tài chính hoặc Phó Giám đốc bán hàng khả năng xem từng khu vực bán hàng hoặc thậm chí nhân viên bán hàng đang hoạt động như thế nào và đưa ra quyết định lãnh đạo. Cũng giống như trách nhiệm, dữ liệu sẽ chi tiết hơn và nhiều chi tiết nhỏ hơn khi hệ thống phân cấp tổ

chức đi xuống C Suite có thể quan tâm đến dữ liệu chi tiết, nhưng để xem ảnh chụp nhanh thông tin cấp cao, bảng điều khiển là chế độ mong muốn.
Thứ hai, Báo cáo dài hơn nhiều so với trang tổng quan. Không chỉ về số lượng chi tiết, mà còn về mặt trực quan. Các bảng và biểu đồ nằm trong một báo cáo có thể chiếm nhiều trang trên một phương tiện in và thậm chí có thể là sách hoặc nhiều tập sách. Trong các phương tiện điện tử, một báo cáo có thể sẽ yêu cầu người đọc cuộn qua nhiều màn hình hoặc nhấp từ trang này sang trang khác.
Điều này rất quan trọng vì lỗi đầu tiên được ghi nhận của Giáo sư về Trực quan hóa Dữ liệu về Thiết kế Bảng điều khiển Thông tin, "Vượt quá giới hạn của một màn hình".
Rất ít người mô tả nó theo cách này. "Việc tôi nhấn mạnh rằng bảng điều khiển nên giới hạn màn hình của nó trong một màn hình duy nhất mà không cần phải cuộn hoặc chuyển đổi giữa nhiều màn hình, có vẻ tùy tiện và hơi lắt léo, nhưng nó dựa trên cơ sở lý luận vững chắc và thực tế. Sau khi nghiên cứu trực quan hóa dữ liệu và nhận thức trực quan trong một thời gian, chúng tôi phát hiện ra rằng điều gì đó mạnh mẽ sẽ xảy ra khi chúng tôi nhìn mọi thứ cùng nhau, tất cả đều trong tầm mắt. Tương tự như vậy, một số thứ quan trọng sẽ bị xâm phạm khi chúng tôi mất dấu một số dữ liệu bằng cách cuộn hoặc chuyển sang màn hình khác để xem dữ liệu khác ".



Hình 86: Chỉ hiển thị nội dung quan trọng nhất

Khi một trang tổng quan riêng lẻ có quá nhiều thông tin cần phải cuộn, sức mạnh của trang tổng quan sẽ giảm đi vì thông tin tồn tại ở đó được dự định sẽ được xem cùng nhau. Mỗi phần thông tin trên bảng điều khiển nhằm cung cấp cho người đọc khả năng trả lời một phần câu hỏi trọng tâm của bảng điều khiển. Các biểu đồ này kết hợp với nhau để trả lời câu hỏi, vì vậy nếu người đọc không thể nhìn thấy chúng cùng nhau, thì việc làm cho chúng kết hợp với nhau sẽ khó hơn nhiều.
Cuối cùng, một báo cáo nhiều khả năng sẽ bao gồm các giải thích bằng văn bản về dữ liệu được trình bày. Nó cũng có thể được kèm theo các bản tóm tắt và thậm chí cả các

khuyến nghị cho tương lai của doanh nghiệp. Một bảng điều khiển sẽ cho rằng người đọc hiểu về chủ đề ở mức độ cao hơn và sẽ không bao gồm nhiều lời giải thích, nếu có.

1.3.3. Chúng giống nhau như thế nào?

Sự tương đồng là tương đối ít về số lượng. Đầu tiên, cả báo cáo và trang tổng quan đều trình bày thông tin. Thứ hai, chủ yếu xoay quanh nội dung mà họ có. Trong khi biểu đồ và bảng có thể xuất hiện trên cả hai, bảng ít có khả năng xuất hiện trên trang tổng quan hơn. Điều này không có nghĩa là thông tin dạng bảng là không quan trọng, điều này chỉ nhằm nhấn mạnh ý tưởng rằng bảng điều khiển phải được tiêu thụ theo thứ tự ngắn và chỉ được chứa các phần thông tin phối hợp với nhau để trả lời câu hỏi trọng tâm.

Như bạn có thể thấy trong bảng được cung cấp bên dưới, một số khía cạnh chính của trang tổng quan cũng có thể liên quan đến việc tạo báo cáo nhưng chúng không bắt buộc phải làm. Các khía cạnh chính này phải được trả lời là “Có” để nó trở thành một bảng điều khiển.

	REPORT	DASHBOARD
Presents Information	*Yes*	*Yes*
Uses Visualizations	*Maybe*	*Yes*
Single Screen View	*Maybe*	*Yes*
All Information Used Together	*Maybe*	*Yes*
Up to the Minute Information	*No*	*Yes*

Hình 87: Sử dụng đúng kích thước và vị trí

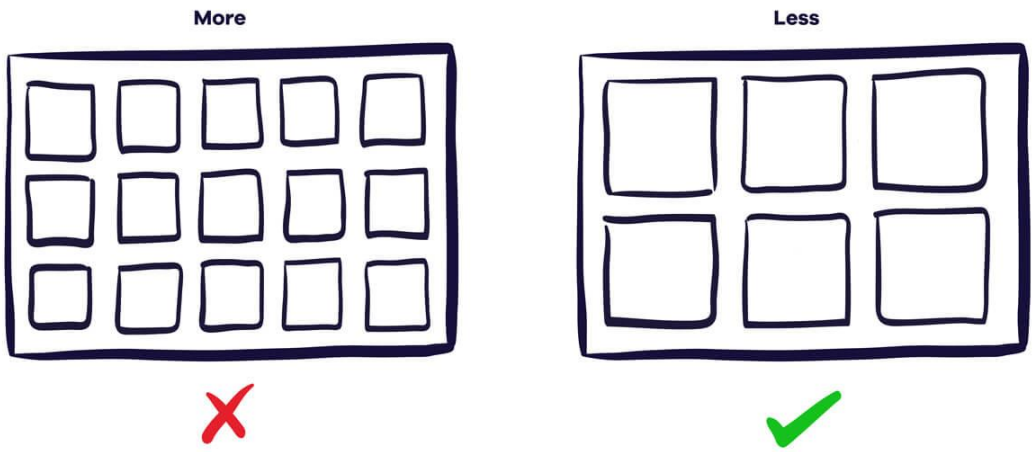
1.4. Hướng dẫn thiết kế cho trang tổng quan

1.4.1. Hãy rõ ràng về những gì bạn đang cố gắng đạt được

Bước đầu tiên để thiết kế bất kỳ trang tổng quan nào là xác định rõ những gì bạn đang cố gắng đạt được. Mục đích của trang tổng quan của bạn là gì? Ai là người? Bạn muốn họ làm khác đi vì điều gì?

Có lẽ bạn đang cố gắng tập trung nhóm của mình vào một mục tiêu cụ thể hoặc cho họ thấy cách họ đóng góp vào bức tranh toàn cảnh. Hoặc có thể bạn muốn đảm bảo một loại vấn đề cụ thể được chú ý nhanh hơn. Đây là tất cả những mục đích tốt cần ghi nhớ.

1.4.2. Chỉ bao gồm những nội dung quan trọng nhất



Hình 88: Thêm cảnh báo nếu cần

Nội dung là chìa khóa khi nói đến trang tổng quan. Nếu bạn không hiển thị các chỉ số hữu ích thì bạn sắp xếp chúng như thế nào không quan trọng.

Thông thường, bạn sẽ xác định một số mục tiêu và KPI và việc thêm những mục tiêu đó là một điểm khởi đầu tuyệt vời. Chỉ cần nhớ, tất cả mọi thứ nên gắn lại với mục đích của bảng của bạn.

Mỗi inch trên bảng điều khiển TV của bạn là bất động sản có giá trị. Việc thêm quá nhiều thông tin có thể làm giảm những gì quan trọng và khiến mọi thứ khó tìm hơn. Nếu bạn thực sự đang đấu tranh để làm cho mọi thứ phù hợp thì bạn có thể cần nhiều hơn một trang tổng quan.

Khi đặt bất kỳ số liệu nào trên trang tổng quan của mình, bạn nên đảm bảo rằng chúng:

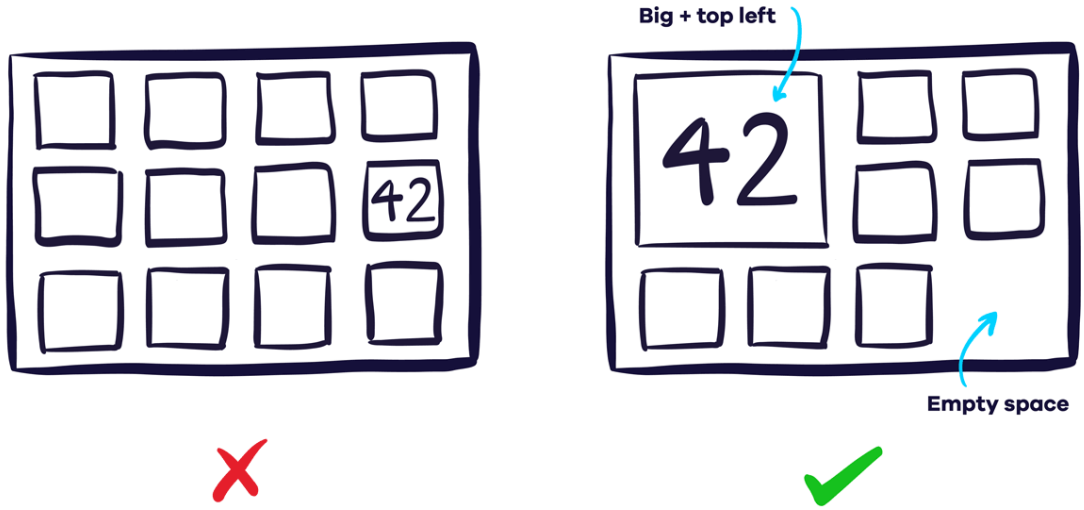
• Phù hợp với mục đích của hội đồng quản trị của bạn
• Có thể bị ảnh hưởng bởi nhóm của bạn
• Có thể dễ hiểu
• Thay đổi thường xuyên một cách hợp lý (bạn không muốn nhìn chằm chằm vào những con số không bao giờ thay đổi)
• Không thay đổi nhiều đến mức bạn không thể dễ dàng phát hiện ra xu hướng

1.4.3. Sử dụng kích thước và vị trí để hiển thị thứ bậc

Trang tổng quan cần có hệ thống phân cấp để dễ quét. Sử dụng kích thước và vị trí để nhấn mạnh thông tin quan trọng nhất và hạ thấp các chỉ số cần được xem xét ít thường xuyên hơn. Kích thước nhất quán và mối quan hệ rõ ràng giữa các yếu tố sẽ giúp tạo ra các mẫu và dòng chảy trực quan.

Về vị trí, góc trên cùng bên trái của trang tổng quan của bạn là vị trí tốt nhất vì đó là nơi mà mắt bạn bị thu hút tự nhiên đầu tiên.

Đừng sợ không gian trống. Tốt hơn là để lại một khoảng trống hơn là làm một cái gì đó lớn hơn chỉ để lấp đầy nó.



Hình 89: Nhóm các chỉ số liên quan với nhau

1.4.4. Cung cấp bối cảnh số của bạn

Để biết một con số tốt hay xấu, người xem của bạn cần có ngữ cảnh. Chẳng hạn, họ có biết rằng 42 khách hàng tiềm năng mới ngày hôm nay là khác thường không?

Một trong những cách dễ nhất để làm điều này là bao gồm dữ liệu trong quá khứ. Bạn có thể bao gồm cùng một số liệu cho ngày hôm trước, hoặc thậm chí một biểu đồ đường hoặc cột hiển thị cách số liệu theo dõi trong một khoảng thời gian dài hơn. Một kỹ thuật khác là bao gồm các mức cao và thấp trung bình hoặc trước đó.

Nếu bạn đang hướng tới một mục tiêu, hãy bao gồm mục tiêu cũng như tiến độ hiện tại của bạn.

Bạn cũng có thể thêm cảnh báo khi một chỉ số cao hơn hoặc thấp hơn một ngưỡng nhất định để giúp phát hiện vấn đề dễ dàng hơn.

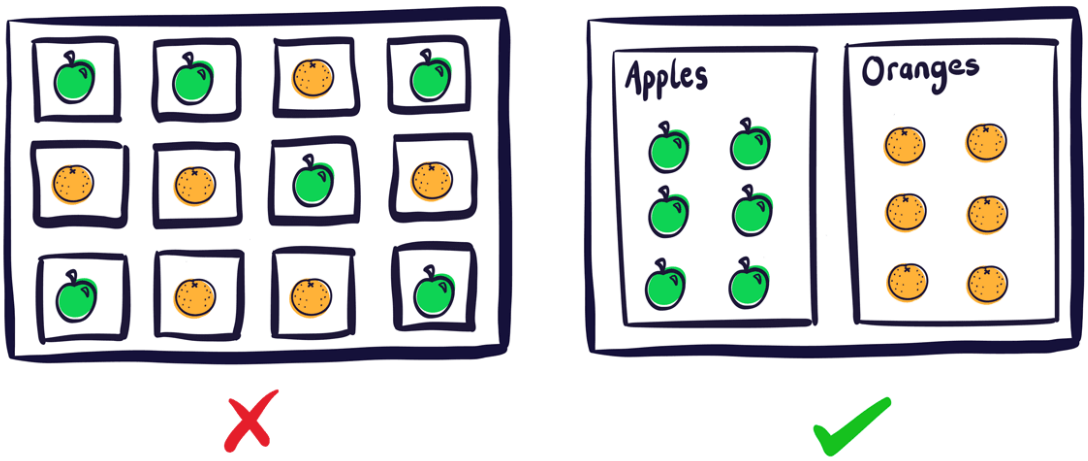
Hình 90: Trang tổng quan - dễ đọc hơn

1.4.5. Nhóm số liệu liên quan của bạn

Định vị thông tin trên trang tổng quan của bạn một cách hợp lý là điều cần thiết. Nhóm các chỉ số có liên quan bên cạnh nhau giúp bạn dễ dàng tìm thấy chúng - và làm cho thiết kế trang tổng quan của bạn hấp dẫn hơn.

Có nhiều cách khác nhau để phân nhóm, v.d. theo chỉ số, sản phẩm, thương hiệu, chiến dịch, khu vực, nhóm hoặc thậm chí khoảng thời gian. Bạn có thể cần phải thử nghiệm loại nào thích hợp nhất cho bảng của bạn.

Đặt cho các nhóm một danh hiệu giúp họ dễ phát hiện hơn.



Hình 91: Sơ lược về bảng điều khiển

1.4.6. Kiên định

Với nhiều trang tổng quan, bạn sẽ thấy có yếu tố lặp lại, chẳng hạn như bạn có thể hiển thị cùng một bộ số liệu cho nhiều thứ. Trang tổng quan của bạn sẽ dễ đọc hơn rất nhiều nếu bạn sử dụng cùng hình ảnh và bố cục giữa các nhóm. Nó cũng sẽ trông đẹp mắt hơn nhiều, vì vậy hãy tránh sử dụng biểu đồ đường thay vì một cột chỉ để thêm thắt cho mọi thứ.

Hình 92: Ví dụ về Bảng điều khiển Hoạt động

1.4.7. Sử dụng nhãn rõ ràng giúp cho khán giả của bạn hiểu

Phần quan trọng của trang tổng quan của bạn là các nhãn mô tả từng chỉ số hoặc biểu đồ. Chúng phải dễ hiểu và rõ ràng đối với người xem Đồng thời, bạn nên cố gắng giữ chúng càng ngắn càng tốt để tránh làm lộn xộn bảng của bạn và cản trở dữ liệu.
Các từ viết tắt cũng có thể hữu ích (miễn là khán giả của bạn hiểu chúng), ví dụ: "7 ngày" thay vì "7 ngày". Các ký hiệu như '%' có thể thay thế từ. Bạn cũng có thể bỏ qua một định nghĩa ngắn hơn cho một số liệu nếu mọi người đã quen thuộc với nó.
Các tiêu đề cũng có thể được sử dụng để giảm sự lặp lại. Hãy tưởng tượng bạn có cùng một số liệu cho các khung thời gian khác nhau, ví dụ: đăng ký ngay hôm nay, đăng ký trong tháng này, v.v. Nếu tất cả chúng được nhóm lại trong một tiêu đề có tên là "Đăng ký", bạn không cần phải lặp lại mỗi lần.

1.4.8. Khoanh vùng số của bạn

Khi hiển thị số, đừng bao gồm độ chính xác cao hơn mức bạn cần. Việc hiển thị tỷ lệ chuyển đổi của bạn đến 3 chữ số thập phân hoặc doanh thu của bạn chính xác đến từng xu khi bạn chỉ quan tâm đến những thay đổi lớn hơn nhiều chỉ làm sao lãng khỏi những gì quan trọng. Thêm, bao gồm quá nhiều chi tiết có thể khiến một ngọn núi chênh vênh.

Hình 93: Ví dụ về Bảng điều khiển Phân tích

1.4.9. Tiếp tục phát triển trang tổng quan của bạn

Phần cuối cùng của chúng tôi về lời khuyên thiết kế bảng điều khiển là quan trọng nhất. Khi bạn đã xây dựng xong trang tổng quan của mình, đừng bỏ nó đi. Yêu cầu nhóm của bạn cho phản hồi.

- Họ xem xét điều gì thường xuyên nhất hoặc thấy hữu ích nhất, và tại sao?
- Những gì họ không bao giờ xem xét hoặc thấy ít hữu ích nhất, và tại sao?
- Có điều gì còn thiếu mà họ thấy hữu ích không?
- Nó có thay đổi bất cứ điều gì về cách họ làm việc không?

Sử dụng phản hồi này để lặp lại trang tổng quan của bạn. Kiểm tra bảng điều khiển của bạn có đang thúc đẩy hành vi mà bạn dự định hay không. Thỉnh thoảng hãy lùi lại khỏi bảng của bạn và xem xét cách tất cả các yếu tố hoạt động cùng nhau. Nhắc nhở bản thân về thông tin mà bạn chủ yếu đang cố gắng truy cập và mức độ hiệu quả của những yếu tố quan trọng đó.

Khi các mục tiêu và ưu tiên của bạn thay đổi, hãy đảm bảo bạn cập nhật bảng của mình để nó hoạt động như nhịp tim cho bất cứ điều gì bạn đang làm.

Minh họa 7: Phân tích dữ liệu với bảng điều khiển

1.5. Các giới hạn của bảng điều khiển

1.5.1. Thiếu khả năng phát hiện bất thường trong thời gian thực ngăn cản việc quản lý sự cố chủ động

Hầu hết các bảng điều khiển BI không hiển thị dữ liệu trong thời gian thực và khi chúng hiển thị, có rất nhiều màn hình lộn xộn số liệu khiến người dùng có thể dễ dàng bỏ lỡ thông tin quan trọng nhất. Sự can thiệp kịp thời là rất quan trọng đối với các doanh nghiệp hiện đại, thường chạy hệ sinh thái tích hợp chặt chẽ của các ứng dụng và cơ sở hạ tầng trải dài trên nhiều bộ phận và xử lý lượng dữ liệu khổng lồ.

Ví dụ: nền tảng công nghệ quảng cáo hàng đầu, Dự án Rubicon, lĩnh vực hàng nghìn tỷ yêu cầu giá thầu mỗi tháng và cần phân tích các điểm dữ liệu từ hàng triệu nguồn tiềm năng. Trong một môi trường như vậy, mỗi phút có thể có tác động đáng kể. Họ phát hiện ra rằng các bảng điều khiển truyền thống không thể cung cấp khả năng phát hiện và phản hồi theo thời gian thực cần thiết để can thiệp trước khi sự bất thường ảnh hưởng đến lợi nhuận của họ.

1.5.2. Phụ thuộc quá nhiều vào dữ liệu lịch sử

Hầu hết các công ty định cấu hình và sử dụng trang tổng quan truyền thống để theo dõi KPI và các chỉ số kinh doanh quan trọng khác để hiểu hoạt động kinh doanh và hệ thống của họ hoạt động như thế nào. Một yếu tố thường bị những người ra quyết định bỏ qua là dữ liệu họ xem trong trang tổng quan truyền thống mô tả những gì đã xảy ra và có thể không phải là một chỉ báo đáng tin cậy về những gì sẽ xảy ra trong tương lai.

Chuyển từ phương thức tư duy mô tả sang dự đoán đòi hỏi sự hiểu biết sâu sắc về bối cảnh kinh doanh và tư duy phản biện, điều này có thể là thách thức đối với bất kỳ người nào hoặc thậm chí là một nhóm chuyên dụng, do sự đa dạng của tập dữ liệu, xu hướng mới và hành vi dao động.

1.5.3. Bỏ sót những sự cố nhỏ có tác động tiêu cực

Khó phát hiện ra một số sự cố, nhưng điều đó không có nghĩa là chúng sẽ không ảnh hưởng đáng kể đến hoạt động kinh doanh. Khi không bị phát hiện, các sự cố khó phát hiện có thể tích tụ và có thể gây ra tác động tương tự như các vấn đề nổi cộm hơn. Một kịch bản điển hình liên quan đến các sự cố chỉ ảnh hưởng đến một thành phần kinh doanh. Những vấn đề riêng biệt này có thể dễ dàng bị mất KPI dựa trên mức trung bình được tính toán của nhiều chỉ số. Ví dụ: một cụm máy chủ có thể đang hiển thị thời gian hoạt động trung bình 99,99%. Nếu một máy chủ trong cụm đó đang gặp phải lượng thời gian chết cao bất thường, nó có thể vẫn ẩn trên bảng điều khiển. Một máy chủ duy nhất là một điểm dữ liệu nhỏ trong một trung tâm dữ liệu với hàng nghìn máy chủ, nhưng nó có thể rất quan trọng tùy thuộc vào những gì máy chủ đó đang chạy.

1.5.4. Trang tổng quan lộn xộn và xác thực sai

Đôi khi, ngay cả khi có tất cả thông tin cần thiết, bảng điều khiển BI vẫn phải vật lộn để trình bày một bức tranh mạch lạc. Đặc biệt, với bảng điều khiển CEO, có một số phỏng đoán trong việc xác định trước thông tin nào đủ quan trọng để hiển thị trong bất động sản hạn chế có sẵn trên màn hình. Khi các cảnh báo bắt đầu bật lên, có thể khó biết dữ liệu nào là cần thiết hoặc đáng bỏ qua. Khối lượng tuyệt đối và độ phức tạp ngày càng tăng của dữ liệu có thể nhanh chóng lấn át giao diện bảng điều khiển, khiến các nhà lãnh đạo doanh nghiệp khó xử lý kịp thời, chính xác.

1.5.5. Thiếu ưu tiên thông minh

Thu thập hàng nghìn sự kiện hoặc cảnh báo mỗi phút từ các ứng dụng và cơ sở hạ tầng của bạn và hiển thị dữ liệu đó trong trang tổng quan không phải là phân tích. Người dùng áp dụng các bộ lọc trên dữ liệu này, thực hiện phân tích và công việc của riêng họ.

Khai thác thông tin thông minh từ dữ liệu không yêu cầu người dùng cuối xác định những gì cần tìm, ở đâu, hoặc đâu là KPI quan trọng nhất, bình thường hay bất thường là gì. Đây không phải là sự thông minh vì người dùng đang cho trang tổng quan biết chính xác dữ liệu nào sẽ hiển thị

1.5.6. Tận dụng sức mạnh của AI Analytics

Chiến lược kinh doanh chỉ có hiệu quả nếu được trao quyền đủ trí tuệ và sự nhanh nhẹn để vượt lên đối thủ cạnh tranh. Trang tổng quan truyền thống không cung cấp thông tin chi tiết đủ nhanh trong thế giới dựa trên dữ liệu ngày nay và khi một doanh nghiệp có thể mất hàng trăm nghìn đô la trong một giờ do trục trặc về giá trên trang Thương mại điện tử, thì tiền đặt cọc là quá cao.

Các công ty cần thông tin chi tiết có thể hành động theo thời gian thực trên tất cả các chỉ số dữ liệu liên quan đến hiệu suất. Các doanh nghiệp hoạt động tốt nhất tận dụng các giải pháp BI được hỗ trợ bởi AI và học máy để loại bỏ nhu cầu về mối tương quan giữa con người với hàng triệu chỉ số quan trọng cần thiết để hiểu hoạt động kinh doanh và hệ thống

Nhóm và liên hệ nhiều điểm bất thường theo thiết kế, phân tích do AI hỗ trợ của Anodot nâng cao những thông tin chi tiết cần thiết trước tiên. Bằng cách tìm hiểu hành vi bình thường của hàng triệu chỉ số, Anodot chỉ phát hiện các sự cố có tác động mạnh nhất và thông báo cho các nhóm liên quan ngay từ đầu

1.6. Giải thích quy trình tạo trang tổng quan

Bước 1: Đi thẳng vào vấn đề

Bảng điều khiển là một màn hình duy nhất được tạo thành với các chỉ số quan trọng nhất. Do đó, trang tổng quan không nên hiển thị tất cả các điểm dữ liệu. Nó chỉ phải bao gồm những thông tin cần thiết. Nó phải là sự lựa chọn các chỉ số phù hợp nhất về doanh nghiệp của bạn.

Khi nhìn vào trang tổng quan, điều đầu tiên bạn sẽ thấy là một bản trình bày nhanh về dữ liệu quan trọng nhất liên quan đến hoạt động của bạn.

Các tab Excel hoặc các menu phụ khác được tạo bằng các công cụ trực quan hóa dữ liệu khác sẽ cho phép bạn nghiên cứu sâu hơn dữ liệu của mình.

Bước 2: Giải quyết vấn đề liên quan đến hoạt động của bạn

Bạn phải bắt đầu bằng cách xác định nhu cầu hoạt động của mình, sau đó tìm hiểu đối tượng của bạn. Bạn phải tự hỏi mình: dữ liệu nào giải quyết tốt nhất các câu hỏi cụ thể mà bạn quan tâm? Vì các tổ chức và lĩnh vực chuyên môn rất đa dạng, cách trang tổng quan được định cấu hình sẽ phải phù hợp với các nhu cầu và nền tảng khác nhau của họ.

Tạo một màn hình hoặc một tab cho mỗi truy vấn dữ liệu. Các trang tổng quan thường chứa đầy dữ liệu không liên quan. Vấn đề với những bảng điều khiển này là chúng không miêu tả một thông điệp rõ ràng. Người dùng của bạn không cần phải xem tất cả dữ liệu có sẵn. Họ chỉ cần xem thông tin mà họ có thể sử dụng cho các nhiệm vụ của mình. Do đó, nhiệm vụ của bạn là sắp xếp dữ liệu cho chúng.

Để làm được điều này, bạn phải biết hội đồng quản trị và đồng nghiệp của bạn mong đợi điều gì.

Mẹo hữu ích: một lượng lớn dữ liệu công khai có sẵn trên trang web của Văn phòng Thống kê Quốc gia và trên các cơ sở dữ liệu công cộng khác. Với dữ liệu này, bạn có thể cung cấp cho khán giả bức tranh toàn cảnh.

Bước 3: Tạo điểm hành động

Xây dựng trang tổng quan là một quá trình phát triển không ngừng. Các dự án của bạn sẽ phát triển và công ty của bạn cũng vậy. Trang tổng quan hiệu quả là trang mà bạn có thể sửa đổi sau nếu cần. Gặp gỡ đồng nghiệp thường xuyên và xem xét các mục tiêu để đảm bảo việc lựa chọn các chỉ số vẫn đáp ứng nhu cầu của họ.

Mỗi chỉ báo phải đại diện cho một hành động cho người dùng của nó. Để đưa ra một ví dụ, số lượng người truy cập trang web ngày càng giảm có thể cảnh báo bộ phận tiếp thị để thực hiện hành động ngay lập tức. Dữ liệu phải tạo ra một hành động; dữ liệu có thể báo hiệu sự bất thường. Đây là một trong những tiêu chí quan trọng cần được xây dựng trong trang tổng quan để nó hoạt động hiệu quả.

Bước 4: Xác định các chỉ số hiệu suất

Việc lựa chọn sai các chỉ số là một lời nguyền cho tổ chức. Mỗi chỉ số phải phục vụ cho chiến lược toàn cầu của công ty. Hơn nữa, cần phải rõ ràng những hành động được yêu cầu nếu có sự thay đổi đột ngột trong dữ liệu.

Nếu bạn chọn sai chỉ báo cho trang tổng quan của mình, bạn có nguy cơ khiến nhóm của mình mất tập trung và đưa ra các quyết định sai lầm.

Để một chỉ báo xuất hiện trên trang tổng quan của bạn, nó cần phải dễ hiểu và phù hợp với tất cả những người sẽ có quyền truy cập vào nó. Ngoài ra, nó phải đáng tin cậy và có thể so sánh được theo ngữ cảnh.

Ví dụ: sẽ không có ý nghĩa gì nếu đưa ra doanh thu hàng năm của công ty mà không so sánh trước với doanh thu của những năm trước hoặc với doanh thu của một công ty khác.

Cần phải giữ logic này đối với khán giả và đối với tất cả các chỉ số.

Bước 5: Tạo trang tổng quan trên thiết bị di động

Đối tượng của bạn sử dụng thông tin như thế nào? Hầu hết người dùng thích sử dụng máy tính của họ để xem các báo cáo. Tuy nhiên, một số người dùng có thể có hồ sơ di động hơn, chẳng hạn như người quản lý và nhân viên bán hàng. Những người dùng này luôn di chuyển. Do đó, điều quan trọng là họ có thể truy cập trang tổng quan một cách tự do trên thiết bị di động.

Vì trang tổng quan là công cụ tương tác nên khả năng tương thích với thiết bị di động cho phép người dùng tương tác với dữ liệu trong tầm tay của mình. Bạn có thể chạm vào các tác phẩm của mình để hiển thị chi tiết hơn, điều hướng qua menu chính vào các danh mục phụ khác nhau. Quan trọng nhất, bạn có thể trình diễn trực tiếp khi thuyết trình với khách hàng.

Bước 6: Làm cho dữ liệu của bạn quen thuộc hơn

Nếu khán giả của bạn hiểu biết về dữ liệu mà bạn đang trình bày, họ sẽ dễ tiếp thu thông điệp mà bạn đang cố gắng truyền tải hơn.

Nhiệm vụ của bạn là tìm ra những ví dụ tốt nhất có liên quan đến khán giả của bạn. Hãy thử những điều mới, không chỉ hiển thị dữ liệu của bạn mà còn hiển thị lý do tại sao bạn hiển thị nó. Đưa ra ngữ cảnh và minh họa dữ liệu của bạn bằng những câu chuyện và trải nghiệm.

Bước 7: Làm cho nó dễ đọc

Bạn có cảm thấy mệt mỏi khi không hiểu trang tổng quan của người khác không? Đừng mắc những sai lầm tương tự.

Dữ liệu càng dễ đọc, thì dữ liệu đó sẽ được hiểu nhanh hơn. Việc ra quyết định cũng bị ảnh hưởng bởi tính rõ ràng của trang tổng quan của bạn. Nếu người đọc hiểu rõ, anh ta có thể đưa ra quyết định tức thời về kết quả. Một điều cần lưu ý là người dùng không phải lúc nào cũng quen với định dạng dữ liệu. Do đó, việc cung cấp thông tin có thể truy cập được cho người dùng là rất quan trọng.

Một biểu đồ phải truyền tải một thông điệp duy nhất. Thông thường, trang tổng quan hiển thị hàng tá chỉ báo và biểu đồ dẫn đến mất sự quan tâm của người dùng. Đừng làm khán giả choáng ngợp bằng cách trình bày hết đồ thị này đến đồ thị thông tin khác. Chỉ trình bày một ý tưởng hoặc hình ảnh trực quan trên mỗi màn hình. Bạn có thể sử dụng nhiều trang trình bày nếu bạn có nhiều thứ cho hiện tại.

Bước 8: Vẽ dữ liệu

Để thu hút khán giả của bạn, hãy kể một câu chuyện.

Dữ liệu được sử dụng để bổ sung sự thật cho những gì bạn đang nói. Nó không nên là trung tâm của các cuộc tranh luận của bạn. Để kể một câu chuyện, bạn phải ngữ cảnh hóa dữ liệu của mình. Viết một câu chuyện phát triển xung quanh dữ liệu của bạn.

Làm việc để làm cho bảng điều khiển của bạn nhiều màu sắc hơn. Nhận xét và tiêu đề của bạn phải dễ đọc. Điều này cho phép khán giả của bạn theo dõi và đoán trước dữ liệu đang được trình bày.

Bước 9: Hiểu khán giả của bạn

Khi bạn đang chuẩn bị báo cáo phân tích dữ liệu của mình, điều cần thiết là phải trả lời đúng các câu hỏi. Để làm được điều này, bạn phải đặt mình vào vị trí của khán giả. Bạn phải tự hỏi mình, 'Các công cụ báo cáo mà tôi đang sử dụng có tương ứng với thông điệp tôi muốn truyền tải không? Biểu đồ tôi đang sử dụng có đủ trực quan cho khán giả của tôi không? Chỉ số này có đủ toàn diện không?

Bước 10: Trình bày suôn sẻ

Nỗi sợ hãi về dữ liệu và số liệu thống kê thường là một trở ngại cho khán giả trong việc hiểu một báo cáo. Bạn có thể khắc phục điều này bằng cách trực quan hóa dữ liệu. Đôi khi, khán giả cần bạn hướng dẫn họ. Tình huống là để chứng minh và sử dụng dữ liệu kể chuyện.

1.7. Các khía cạnh bảo mật cho trang tổng quan

Một bảng điều khiển bảo mật tốt cần phải bao gồm những điều sau đây trong một khoảng thời gian cụ thể / được đo lường: Một dấu hiệu về mức độ đe dọa hiện tại đối với tổ chức; một dấu hiệu về các sự kiện và sự cố đã xảy ra; biên bản về lỗi xác thực; một chỉ báo về việc quét, thăm dò và truy cập trái phép, và một chỉ báo nếu các biện pháp chính đó là tăng, giảm hoặc không thay đổi; các cuộc tấn công vũ phu chống lại hệ thống và các thiết bị không tuân thủ; vi phạm chính sách; sự kiện phần mềm độc hại; và các sự kiện lừa đảo

Danh sách kiểm tra bảng điều khiển bảo mật:

• Mức độ đe dọa hiện tại đối với tổ chức.

• Các sự kiện và sự cố đã xảy ra.

• Lỗi xác thực.
• Quét, thăm dò và truy cập trái phép.
• Tấn công bạo lực chống lại hệ thống và các thiết bị không tuân thủ.
• Vi phạm chính sách.
• Sự kiện phần mềm độc hại.
• Sự kiện lừa đảo.
• Các chỉ số kỹ thuật chi tiết cụ thể cho các biện pháp kiểm soát được sử dụng để quản lý rủi ro đối với các vectơ đe dọa hiện tại và mới nổi.
• Số lượng tài sản được bảo hiểm.
• Tài sản mới phát hiện.
• Tài sản ngừng hoạt động.
• Số lượng các mối đe dọa được phát hiện và mức độ rủi ro của chúng.
• Tầm nhìn rõ ràng về bối cảnh rủi ro.
• Lĩnh vực đào tạo / nhận thức.
• Quản lý lỗ hổng bảo mật.
• Quản lý rủi ro của bên thứ ba.
• Quản lý sự cố.
• Quản lý rủi ro tổng thể.
• Thời gian trung bình để vá.
• Thời gian trung bình để phát hiện và ứng phó với các sự cố tiềm ẩn.
• Cửa sổ tiếp xúc trung bình.
• Số lượng ngoại lệ / loại ngoại lệ.
• Tỷ lệ phần trăm không thành công.
• Tác động của đào tạo khắc phục hậu quả.

1.8. Cài đặt các thư viện cần thiết

Chúng tôi sẽ bắt đầu bằng cách cài đặt một bảng điều khiển giải thích bằng cách sử dụng pip. Lệnh dưới đây sẽ làm điều đó.

```
pip install explainerdashboard
```

- **Nhập các thư viện bắt buộc**

Trong bước này, chúng tôi sẽ nhập các thư viện và chức năng cần thiết để tạo mô hình học máy và bảng điều khiển.

```
from sklearn.ensemble import RandomForestClassifier
from explainerdashboard import ClassifierExplainer, ExplainerDashboard
from explainerdashboard.datasets import titanic_survive, titanic_names
```

- **Tạo Mô hình & Trang tổng quan**

Đây là bước cuối cùng trong đó chúng tôi sẽ tạo mô hình học máy và sau đó diễn giải mô hình đó bằng cách tạo trang tổng quan.

+ Tạo mô hình:

```
feature_descriptions = {
   "Sex": "Gender of passenger",
```

```
   "Gender": "Gender of passenger",
   "Deck": "The deck the passenger had their cabin on",
   "PassengerClass": "The class of the ticket: 1st, 2nd or 3rd class",
   "Fare": "The amount of money people paid",
   "Embarked": "the port where the passenger boarded the Titanic. Either Southampton,
Cherbourg or Queenstown",
   "Age": "Age of the passenger",
   "No_of_siblings_plus_spouses_on_board": "The sum of the number of siblings plus the
number of spouses on board",
   "No_of_parents_plus_children_on_board" : "The sum of the number of parents plus the
number of children on board",
}X_train, y_train, X_test, y_test = titanic_survive()
train_names, test_names = titanic_names()
model = RandomForestClassifier(n_estimators=50, max_depth=5)
model.fit(X_train, y_train).
```

+ Tạo Trang tổng quan:

```
from sklearn.ensemble import RandomForestClassifier
from explainerdashboard import ClassifierExplainer, ExplainerDashboard
from explainerdashboard.datasets import titanic_survive,
titanic_namesfeature_descriptions = {
   "Sex": "Gender of passenger",
   "Gender": "Gender of passenger",
   "Deck": "The deck the passenger had their cabin on",
   "PassengerClass": "The class of the ticket: 1st, 2nd or 3rd class",
   "Fare": "The amount of money people paid",
   "Embarked": "the port where the passenger boarded the Titanic. Either Southampton,
Cherbourg or Queenstown",
   "Age": "Age of the passenger",
   "No_of_siblings_plus_spouses_on_board": "The sum of the number of siblings plus the
number of spouses on board",
   "No_of_parents_plus_children_on_board" : "The sum of the number of parents plus the
number of children on board",
}X_train, y_train, X_test, y_test = titanic_survive()
train_names, test_names = titanic_names()
model = RandomForestClassifier(n_estimators=50, max_depth=5)
model.fit(X_train, y_train)explainer = ClassifierExplainer(model, X_test, y_test,
                        cats=['Deck', 'Embarked',
                           {'Gender': ['Sex_male', 'Sex_female', 'Sex_nan']}],
                        cats_notencoded={'Embarked': 'Stowaway'},
```

```
                descriptions=feature_descriptions,
                labels=['Not survived', 'Survived'],
                idxs = test_names,
                index_name = "Passenger",
                target = "Survival",
                )db = ExplainerDashboard(explainer,
            title="Titanic Explainer",
            shap_interaction=False,
            )
db.run(port=8050)
```

1.9. Giải thích các khái niệm, chức năng và lớp lập trình cần thiết để hiển thị trang tổng quan

- Lập trình hướng đối tượng là một cách tiếp cận để giải quyết vấn đề trong đó tất cả các tính toán được thực hiện bằng cách sử dụng các đối tượng. Đối tượng là một thành phần của chương trình biết cách thực hiện các hành động nhất định và cách tương tác với các phần tử khác của chương trình. Đối tượng là đơn vị cơ bản của lập trình hướng đối tượng. Một ví dụ đơn giản về một đối tượng sẽ là một người. Về mặt logic, bạn sẽ mong đợi một người có tên. Đây sẽ được coi là tài sản của một người. Bạn cũng có thể mong đợi một người có thể làm điều gì đó, chẳng hạn như đi bộ hoặc lái xe. Đây sẽ được coi là một phương pháp của con người.

Mã trong lập trình hướng đối tượng được tổ chức xung quanh các đối tượng. Khi bạn đã có đối tượng của mình, chúng có thể tương tác với nhau để tạo nên điều gì đó xảy ra. Giả sử bạn muốn có một chương trình trong đó một người lên ô tô và lái nó từ A đến B. Bạn sẽ bắt đầu bằng cách mô tả các đối tượng, chẳng hạn như một người và ô tô. Điều đó bao gồm các phương pháp: một người biết cách lái xe ô tô và một chiếc xe hơi biết cảm giác lái. Khi bạn đã có đồ vật của mình, bạn mang chúng lại với nhau để người đó có thể lên xe và lái.

- Một ***class*** là một bản thiết kế của một đối tượng. Bạn có thể nghĩ về một lớp như một khái niệm, và đối tượng là hiện thân của khái niệm đó. Bạn cần có một lớp trước khi có thể tạo một đối tượng. Vì vậy, giả sử bạn muốn sử dụng một người trong chương trình của mình. Bạn muốn có thể mô tả người đó và yêu cầu người đó làm điều gì đó. Một lớp được gọi là 'người' sẽ cung cấp bản thiết kế cho một người trông như thế nào và một người có thể làm gì. Để thực sự sử dụng một người trong chương trình của bạn, bạn cần tạo một đối tượng. Bạn sử dụng lớp người để tạo một đối tượng kiểu 'người'. Bây giờ bạn có thể mô tả người này và yêu cầu người đó làm điều gì đó.

Lớp học rất hữu ích trong lập trình. Hãy xem xét ví dụ về trường hợp bạn không muốn chỉ sử dụng một người mà là 100 người. Thay vì mô tả chi tiết từng đối tượng từ đầu, bạn có thể sử dụng cùng một lớp người để tạo 100 đối tượng kiểu 'người'. Bạn vẫn phải cung cấp cho mỗi người một cái tên và các thuộc tính khác, nhưng cấu trúc cơ bản của một người trông giống nhau.

- ***Function*** là một tổ hợp các lệnh được kết hợp với nhau để đạt được một kết quả nào đó. Một hàm thường yêu cầu một số đầu vào (được gọi là đối số) và trả về một số kết quả. Ví dụ, hãy xem xét ví dụ về việc lái xe ô tô. Để xác định quãng đường đi được, bạn cần thực hiện một phép tính bằng cách sử dụng quãng đường đã lái và lượng nhiên liệu đã sử dụng. Bạn có thể viết một hàm để thực hiện phép tính này. Các đối số đi vào hàm sẽ là khoảng cách và mức tiêu thụ nhiên liệu, và kết quả sẽ là số dặm. Bất cứ lúc nào bạn muốn xác định quãng đường, bạn chỉ cần gọi hàm để thực hiện phép tính.

Minh họa 8: Lập trình hướng đối tượng

1.10. Làm rõ các thành phần lập trình dựa trên web cần thiết để lập trình bảng điều khiển

Một trang tổng quan điển hình chứa ba yếu tố:

• Tiêu đề giải thích nội dung của trang tổng quan và mục đích của nó

• Sơ đồ trực quan hóa các chỉ số

• Giải thích ngắn gọn về trạng thái và thông tin trong sơ đồ

Khi thiết kế các trang của bảng điều khiển, các nguyên tắc của khả năng nhận thức nhận thức cần được tính đến.

1. Các phần tử của trang tổng quan phải liên quan đến nhau về mặt logic và khái niệm 2. Số lượng phần tử trong trang tổng quan (sơ đồ, trường văn bản, giải thích, nút) không được nhiều hơn 7 (+2 nếu cần) vì đây là số yếu tố mà một người bình thường có thể lưu giữ trong bộ nhớ ngắn hạn.

3. Việc sử dụng màu sắc nên được hạn chế ở mức tối thiểu và màu sắc nên ngoại suy sơ đồ và thông tin quan trọng trong bảng điều khiển.

Một số công nghệ và khuôn khổ tồn tại có thể hỗ trợ sự phát triển của trang tổng quan, ví dụ:

• Dashing.io (open Nguồn): http://dashing.io/ - một phần mềm bảng điều khiển sẵn sàng sử dụng dựa trên các liên kết tệp XML đến máy chủ web. Khung này rất đơn giản để thiết lập, nhưng hạn chế về khả năng đồ họa của nó. Nó cũng yêu cầu một bộ xử lý dữ liệu xương sống vì nó không thể tự xử lý dữ liệu.

• The dash (free): https://www.thedash.com/ - một sự thay thế cho dashing.io, với các yêu cầu tương tự đối với các tập lệnh của bộ xử lý xương sống, nhưng linh hoạt hơn về các hình ảnh trực quan có sẵn (ví dụ: sơ đồ). Dự án đo lường Trung tâm phần mềm 8

• Google dashboard (free): https://developers.google.com/appsscript/articles/Biểu đồs_dashboard - một tập hợp các biểu đồ dựa trên javascript và SVG đơn giản để thiết lập có thể được tùy chỉnh rất dễ dàng. Ưu điểm chính là nó đơn giản và dễ sử dụng nhưng nó cũng yêu cầu xử lý dữ liệu xương sống.

• D3 (Data Driven Documents, open Nguồn): http://d3js.org/ - một giải pháp thay thế linh hoạt hơn (mạnh mẽ và biểu cảm) cho biểu đồ / bảng điều khiển của Google.

• Tibco Spotfire: http://spotfire.tibco.com/products/spotfiredesktop?gclid=CjwKEAjwkK6wBRCcoK_tiOTzFASJAC7RArijfNQV5JgnHYXKOVyhwDlfgKdTj0b3ei4xyJBqn6VqhoCLO3w_wcB - một công cụ thông minh dành cho doanh nghiệp cho phép dễ dàng tạo các báo cáo chi tiết và trang tổng quan. Ưu điểm chính là khi dữ liệu nằm trong cơ sở dữ liệu, công cụ này có một cách đồ họa để tạo các biểu đồ (không cần lập trình như trong các kỹ thuật trước đây); nhược điểm chính là nó mang tính thương mại và việc thiết lập cơ sở dữ liệu và nhập dữ liệu đòi hỏi phải lập trình và nhiều nỗ lực hơn so với trường hợp của các tập lệnh cho các kỹ thuật trước đây.

• Bảngu: http://www.Bảngau.com/ and http://www.Bảngau.com/learn/whitepapers/5-bảng điều khiển-thực hành-hiệu quả-tốt nhất - một giải pháp thay thế cho Spotfire.

• Qlikview: http://www.qlik.com – một giải pháp thay thế khác cho Spotfire

Hình minh họa 9: Robot và người máy trí tuệ nhân tạo trong tương lai

2. Khả năng trực quan hóa dữ liệu

2.1. Các nguyên tắc cơ bản về trực quan hóa dữ liệu

2.1.1. Hiển thị các loại sơ đồ khác nhau với các khu vực ứng dụng

- Operational Dashboards: Hình ảnh một bảng điều khiển "truyền thống". Bạn có thấy các chỉ số, cập nhật theo thời gian thực, hiển thị dữ liệu hiệu suất liên quan đến các hoạt động trong ngày không? Nếu vậy, bạn đang hình dung một trang tổng quan hoạt động, được cho là loại trang tổng quan phổ biến nhất. Đây là những bảng điều khiển rất phù hợp với màn hình treo tường trên sàn nhà máy sản xuất hoặc bộ chỉ huy hoạt động toàn cầu.

Bảng điều khiển hoạt động được thiết kế để cung cấp, trong nháy mắt, ảnh chụp nhanh toàn diện về hiệu suất trong ngày. Giống như bảng điều khiển trên ô tô, bảng điều khiển hoạt động cung cấp cho người xem thông tin liên quan đến hoạt động tức thì của tổ chức. Chúng không nên yêu cầu chi tiết hóa để hữu ích, vì thường người xem sẽ không có tùy chọn để thao tác trang tổng quan qua chế độ xem ban đầu.

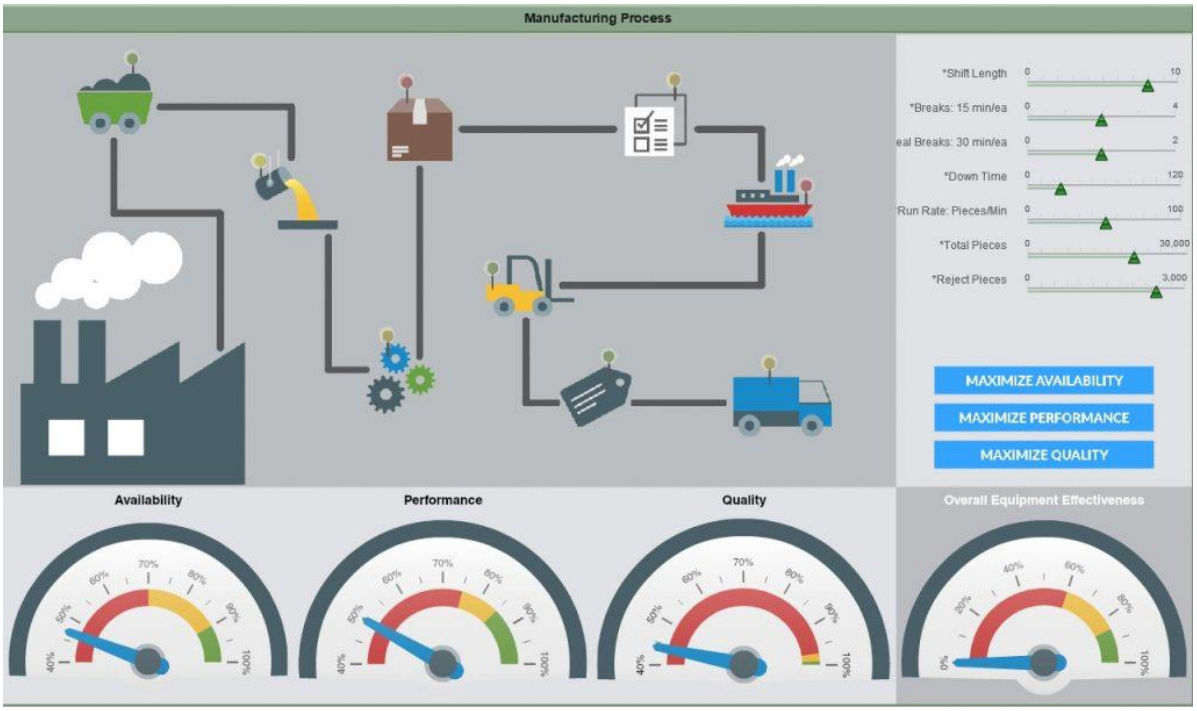
Điều này có nghĩa là bảng điều khiển hoạt động phải có một cái nhìn khá chi tiết. Đổi lại, điều quan trọng là phải đảm bảo rằng, khi lập kế hoạch tổng quan về hiệu suất hoạt động, phạm vi không trở nên quá rộng. Nếu bạn cố gắng hoàn thành quá nhiều việc với một trang tổng quan, nó có thể trở nên không rõ ràng và cuối cùng không được sử dụng. Ghi nhớ người dùng cuối sẽ hỗ trợ trong quá trình này. Bạn đang truyền đạt các chỉ số cho một dây chuyền lắp ráp hay một giám đốc điều hành? Đảm bảo tập trung nhóm người dùng cuối của bạn để bạn biết chính xác những gì họ cần xem để thực hiện các chức năng công việc của họ một cách hiệu quả.

Hình 94: Ví dụ về Bảng điều khiển Chiến lược

Analytical Dashboards: Nếu bạn đang sử dụng dữ liệu từ quá khứ để xác định các xu hướng có thể giúp bạn đưa ra quyết định về tương lai, bạn đang trên đường tạo trang tổng quan phân tích. Các trang tổng quan này là công cụ mà người dùng có thể tương tác, hỏi và khám phá. Do đó, các tính năng như bảng tổng hợp và chi tiết hóa rất phù hợp với trang tổng quan phân tích.

So sánh và đối chiếu dữ liệu trên nhiều biến là một khía cạnh quan trọng của phân tích dữ liệu. Người dùng phải có khả năng so sánh dữ liệu theo thời gian, để xem liệu sự khác biệt về hiệu suất có tương quan với hành động của công ty hay không (hoặc nếu các lực lượng bên ngoài, chẳng hạn như tính thời vụ, có tác động có thể đo lường được đối với các chỉ số). Việc cắt và chia nhỏ dữ liệu theo cách có cấu trúc cho phép người dùng xác định những nỗ lực đã mang lại hiệu quả

Bảng điều khiển phân tích có thể là một công cụ có giá trị trong tay phải, nhưng chúng đòi hỏi mức độ hiểu biết mà người dùng doanh nghiệp bình thường có thể không có. Dữ liệu trong trang tổng quan phân tích thường phức tạp, cũng như các bài tập phân tích mà trang tổng quan phù hợp. Do đó, các bảng điều khiển phân tích tốt nhất nên để các nhà phân tích cơ sở dữ liệu của bạn thay vì toàn bộ công ty. Xác định quyền của người dùng là một cách đơn giản để đảm bảo rằng các trang tổng quan phân tích của bạn đang được cung cấp cho đúng nhóm.



Hình 95: Ví dụ về Bảng điều khiển Phân tích

Strategic Dashboards: Chúng tôi dành nhiều thời gian để nói về KPI, bởi vì bằng chứng rõ ràng rằng việc thiết lập mục tiêu và hướng tới chúng là con đường chắc chắn nhất dẫn đến thành công. Nếu bạn đã xác định các chỉ số hiệu suất chính và đang theo dõi hiệu suất liên quan đến các KPI đó, thì rất có thể bạn đã có cho mình một bảng điều khiển chiến lược. Các bảng điều khiển này thường được sử dụng để điều chỉnh hiệu suất của các bộ phận với chiến lược tổng thể của công ty.

Thông thường, các bảng điều khiển chiến lược có một sự tinh tế hồi tưởng. Họ xem xét dữ liệu hiệu suất điểm chuẩn từ quý trước và so sánh với khoảng thời gian hiện tại. Mọi thứ đã được cải thiện, giữ nguyên hay xấu đi? Chúng cũng thường bao gồm dữ liệu từ nhiều nguồn, vì các mục tiêu của toàn công ty bị ảnh hưởng bởi nhiều hệ thống và hành động.

Trang tổng quan chiến lược thường chia sẻ các số liệu quan trọng đối với toàn tổ chức, vì vậy hãy cân nhắc việc cung cấp chúng cho toàn tổ chức. Rõ ràng là ban quản lý và nhóm điều hành sẽ muốn có một cái nhìn tổng quát về các KPI chiến lược, nhưng khi loại dữ liệu hiệu suất này được chia sẻ một cách minh bạch với nhân viên cấp thấp hơn, có thể có những lợi ích bất ngờ. Bạn không bao giờ biết được ý tưởng tuyệt vời tiếp theo trong tổ chức của mình sẽ đến từ đâu và khi bạn trao quyền cho cả nhóm bằng kiến thức, điều đó đặc biệt trở thành sự thật.

Hình 96: Ví dụ về Bảng điều khiển Chiến lược

2.1.2. Định nghĩa các cấu trúc dữ liệu cần thiết để sử dụng loại đường chéo:

Chúng tôi không thể nhấn mạnh đủ tầm quan trọng của việc chọn các loại hình ảnh hóa dữ liệu phù hợp. Bạn có thể hủy mọi nỗ lực của mình bằng một loại biểu đồ bị thiếu hoặc không chính xác. Điều quan trọng là phải hiểu loại thông tin bạn muốn truyền tải và chọn hình ảnh hóa dữ liệu phù hợp với nhiệm vụ.

Hình ảnh và biểu đồ tập trung vào trang tổng quan được chia thành bốn danh mục chính có liên quan đến mục đích của hình ảnh hóa: mối quan hệ, phân phối, thành phần và so sánh. Điều quan trọng là phải hiểu mục tiêu của số liệu trước khi chọn loại biểu đồ bạn muốn. Ở đây chúng ta sẽ nói về một số loại phổ biến nhất và mục đích của chúng:

Biểu đồ đường là tuyệt vời khi hiển thị các mô hình thay đổi trong một chuỗi liên tục. Chúng nhỏ gọn, rõ ràng và chính xác. Định dạng biểu đồ đường phổ biến và quen thuộc với hầu hết mọi người nên chúng có thể dễ dàng được phân tích trong nháy mắt.

Chọn biểu đồ thanh nếu bạn muốn so sánh nhanh các mục trong cùng một danh mục, ví dụ: lượt xem trang theo quốc gia. Một lần nữa các biểu đồ như vậy rất dễ hiểu, rõ ràng và cô đọng.

Biểu đồ hình tròn không phải là lựa chọn hoàn hảo. Chúng xếp hạng độ chính xác thấp vì người dùng khó so sánh chính xác kích thước của các lát bánh. Mặc dù các biểu đồ như vậy có thể được quét ngay lập tức và người dùng sẽ nhận thấy phần lớn nhất ngay

lập tức, nhưng có thể có vấn đề về tỷ lệ dẫn đến phần nhỏ nhất nhỏ đến mức không thể hiển thị chúng. Một thực tiễn tốt khi sử dụng biểu đồ hình tròn là chỉ thực hiện với một vài lát cắt, bằng cách này, bạn đảm bảo rằng thông tin dễ hiểu và sẽ mang lại giá trị cho trang tổng quan của bạn.

Biểu đồ thu nhỏ thường không có tỷ lệ, có nghĩa là người dùng sẽ không thể nhận thấy các giá trị riêng lẻ. Tuy nhiên, chúng hoạt động tốt khi bạn có nhiều chỉ số và bạn chỉ muốn hiển thị các xu hướng. Chúng có thể quét nhanh chóng và rất nhỏ gọn.

Việc giải mã các biểu đồ phân tán cũng không dễ dàng như vậy vì chúng là một kiểu trực quan hóa nâng cao dành cho những người dùng hiểu biết hơn. Họ nhằm mục đích tìm ra mối tương quan giữa hai biến số. Khi dữ liệu được phân phối trên biểu đồ, kết quả cho thấy mối tương quan là tích cực, tiêu cực hoặc không tồn tại.

Biểu đồ đo là hình ảnh trực quan có giá trị để cung cấp ngữ cảnh. Ưu điểm của các biểu đồ này nằm ở chỗ dễ hiểu vì chúng sử dụng nhiều màu sắc khác nhau để biểu thị các giá trị khác nhau của cùng một số liệu. Chúng thường được sử dụng trong các tình huống mà giá trị kỳ vọng đã được biết trước, theo cách này, các bên liên quan khác nhau sử dụng bảng điều khiển có thể hiểu vị trí của họ chỉ bằng cách nhìn vào biểu đồ đo. Ví dụ, để theo dõi mục tiêu bán hàng hoặc tăng trưởng doanh số bán hàng.

Hầu hết các chuyên gia đồng ý rằng biểu đồ bong bóng không phù hợp với trang tổng quan. Chúng đòi hỏi quá nhiều nỗ lực tinh thần từ người dùng ngay cả khi đọc thông tin đơn giản trong ngữ cảnh. Do thiếu chính xác và rõ ràng, chúng không phổ biến lắm và người dùng không quen với chúng. theo thứ tự)
- Có ba loại trang tổng quan: hoạt động, chiến lược và phân tích.
+ Các chỉ số bạn có thể theo dõi trên trang tổng quan hoạt động
• Số liệu hiệu suất trang web như người dùng mới hoặc tỷ lệ thoát
• Số lượng người theo dõi hoặc nhận xét trên các kênh truyền thông xã hội của bạn
+ Các chỉ số bạn có thể theo dõi trên trang tổng quan phân tích
• Giá trị hợp đồng hàng năm để theo dõi số tiền mà một hợp đồng khách hàng trung bình có giá trị
• Đo lường thói quen chi tiêu của công ty bạn bằng Điểm hiệu quả Bessemer
• Hiểu sự gia tăng người dùng hoạt động hàng ngày trong một khoảng thời gian
• Lợi tức chi tiêu cho quảng cáo để theo dõi hiệu quả của tiền quảng cáo kỹ thuật số của bạn
+ Các chỉ số bạn có thể theo dõi trên trang tổng quan chiến lược
• Kết quả hoạt động tài chính hàng tháng, hàng quý hoặc hàng năm
• Tỷ lệ tăng trưởng tài khoản và MRR
• Thu nhập trước lãi vay, thuế, khấu hao và khấu hao (hay còn gọi là EBITDA)

2.2. Vị trí của sơ đồ trong trang tổng quan

Các phương pháp hay nhất về trang tổng quan trong thiết kế quan tâm nhiều hơn đến các chỉ số tốt và biểu đồ được suy nghĩ kỹ lưỡng. Bước tiếp theo là vị trí của các biểu đồ trên trang tổng quan. Nếu trang tổng quan của bạn được tổ chức trực quan, người dùng sẽ dễ dàng tìm thấy thông tin họ cần. Bố cục kém buộc người dùng phải suy nghĩ nhiều hơn trước khi nắm bắt được vấn đề và không ai thích tìm kiếm dữ liệu trong một rừng biểu đồ và con số. Quy tắc chung là thông tin quan trọng phải được hiển thị đầu tiên - ở đầu màn hình, góc trên bên trái. Có một số thông minh khoa học đằng sau vị trí này - hầu hết các nền văn hóa đọc ngôn ngữ viết của họ từ trái sang phải và từ trên xuống dưới, có nghĩa là mọi người nhìn vào phần trên bên trái của trang trước một cách trực quan, bất kể bạn đang phát triển một doanh nghiệp thiết kế bảng điều khiển hoặc quy mô nhỏ hơn trong bộ - quy tắc là như nhau.

Một nguyên tắc bố trí bảng điều khiển hữu ích khác là bắt đầu với bức tranh lớn. Xu hướng chính sẽ được nhìn thấy trong nháy mắt. Sau khi tiết lộ tổng quan đầu tiên này, bạn có thể tiếp tục với các biểu đồ chi tiết hơn. Hãy nhớ nhóm các biểu đồ theo chủ đề với các chỉ số có thể so sánh được đặt cạnh nhau. Bằng cách này, người dùng không phải thay đổi tinh thần khi nhìn vào bảng điều khiển, chẳng hạn như chuyển từ dữ liệu bán hàng sang dữ liệu tiếp thị, rồi lại chuyển sang dữ liệu bán hàng. Phương pháp hay nhất về bảng điều khiển phân tích này sẽ cho phép bạn trình bày dữ liệu của mình theo cách có ý nghĩa nhất và rõ ràng cho người dùng cuối.

2.3. Chuyển đổi dữ liệu từ nguồn dữ liệu sang dữ liệu chìm trong sơ đồ

2.3.1. Chuẩn bị dữ liệu, lọc trước, chọn nguồn

- Chuẩn bị dữ liệu là quá trình làm sạch và biến đổi dữ liệu thô trước khi xử lý và phân tích. Đây là một bước quan trọng trước khi xử lý và thường liên quan đến việc định dạng lại dữ liệu, sửa chữa dữ liệu và kết hợp các tập dữ liệu để làm phong phú dữ liệu.

Chuẩn bị dữ liệu thường là công việc kéo dài đối với các chuyên gia dữ liệu hoặc người dùng doanh nghiệp, nhưng điều cần thiết là đặt dữ liệu vào ngữ cảnh để biến nó thành thông tin chi tiết và loại bỏ sự sai lệch do chất lượng dữ liệu kém.

- 76% các nhà khoa học dữ liệu nói rằng chuẩn bị dữ liệu là phần tồi tệ nhất trong công việc của họ, nhưng các quyết định kinh doanh hiệu quả, chính xác chỉ có thể được thực hiện với dữ liệu sạch. Chuẩn bị dữ liệu giúp:**Fix errors quickly** — Data preparation helps catch errors before processing. After data has been removed from its original Nguồn, these errors become more difficult to understand and correct.

  • **Tạo dữ liệu chất lượng hàng đầu** - Làm sạch và định dạng lại bộ dữ liệu đảm bảo rằng tất cả dữ liệu được sử dụng trong phân tích sẽ có chất lượng cao.

  • **Đưa ra quyết định kinh doanh tốt hơn** - Dữ liệu chất lượng cao hơn có thể được xử lý và phân tích nhanh chóng và hiệu quả hơn dẫn đến các quyết định kinh doanh kịp thời, hiệu quả và chất lượng cao hơn.

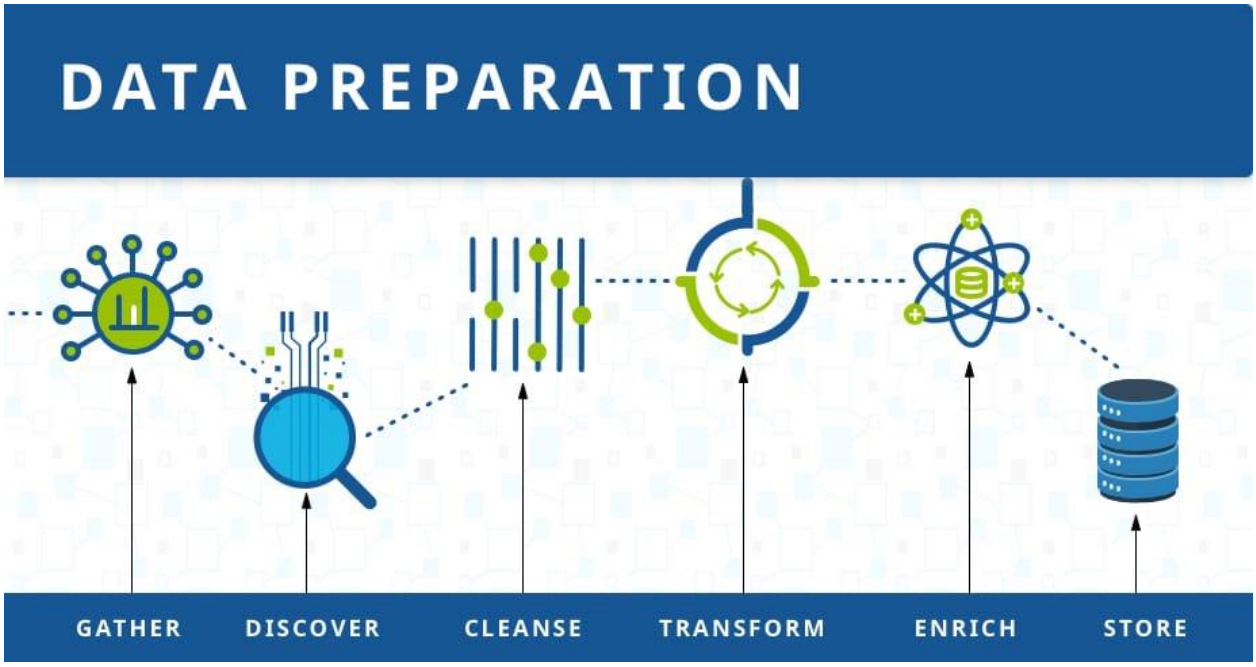
Ngoài ra, khi dữ liệu và các quy trình dữ liệu chuyển sang đám mây, việc chuẩn bị dữ liệu sẽ di chuyển theo nó vì những lợi ích lớn hơn, chẳng hạn như:

- **• Khả năng mở rộng vượt trội** - Chuẩn bị dữ liệu đám mây có thể phát triển theo tốc độ của doanh nghiệp. Doanh nghiệp không phải lo lắng về cơ sở hạ tầng cơ bản hoặc cố gắng dự đoán sự phát triển của chúng.
- **• Bằng chứng trong tương lai** - Tự động nâng cấp chuẩn bị dữ liệu đám mây để các khả năng mới hoặc các bản sửa lỗi có thể được bật ngay sau khi chúng được phát hành. Điều này cho phép các tổ chức đi trước đường cong đổi mới mà không bị chậm trễ và tăng thêm chi phí.
- **• Cộng tác và sử dụng dữ liệu được tăng tốc** — Chuẩn bị dữ liệu trên đám mây có nghĩa là nó luôn bật, không yêu cầu bất kỳ cài đặt kỹ thuật nào và cho phép các nhóm cộng tác trong công việc để có kết quả nhanh hơn.

- Ngoài ra, một công cụ chuẩn bị dữ liệu gốc đám mây tốt sẽ cung cấp các lợi ích khác (như GUI trực quan và đơn giản để sử dụng) để chuẩn bị dễ dàng và hiệu quả hơn.

Các bước chuẩn bị dữ liệu

Các chi tiết cụ thể của quá trình chuẩn bị dữ liệu khác nhau tùy theo ngành, tổ chức và nhu cầu, nhưng khuôn khổ phần lớn vẫn giống nhau.



Hình 97: Minh họa về chuẩn bị dữ liệu

i. Thu thập dữ liệu

Quá trình chuẩn bị dữ liệu bắt đầu với việc tìm kiếm dữ liệu phù hợp. Điều này có thể đến từ một danh mục dữ liệu hiện có hoặc có thể được thêm vào đặc biệt.

ii. Khám phá và đánh giá dữ liệu

Sau khi thu thập dữ liệu, điều quan trọng là phải khám phá từng tập dữ liệu. Bước này là về việc tìm hiểu dữ liệu và hiểu những gì phải làm trước khi dữ liệu trở nên hữu ích trong một ngữ cảnh cụ thể.

Khám phá là một nhiệm vụ lớn, nhưng nền tảng chuẩn bị dữ liệu của Talend cung cấp các công cụ trực quan hóa giúp người dùng lập hồ sơ và duyệt dữ liệu của họ.

iii. Làm sạch và xác thực dữ liệu

Làm sạch dữ liệu theo truyền thống là phần tốn nhiều thời gian nhất của quá trình chuẩn bị dữ liệu, nhưng điều quan trọng là loại bỏ dữ liệu bị lỗi và lấp đầy khoảng trống. Các nhiệm vụ quan trọng ở đây bao gồm:

Loại bỏ dữ liệu không liên quan và ngoại lai.

Điền vào các giá trị còn thiếu.

Chỉnh sửa dữ liệu theo một mẫu tiêu chuẩn hóa.

Che dấu các mục nhập dữ liệu riêng tư hoặc nhạy cảm.

Khi dữ liệu đã được làm sạch, nó phải được xác thực bằng cách kiểm tra các lỗi trong quá trình chuẩn bị dữ liệu cho đến thời điểm này. Thông thường, một lỗi trong hệ thống sẽ xuất hiện rõ ràng trong bước này và cần được giải quyết trước khi tiếp tục.

iv. Chuyển đổi và làm phong phú dữ liệu

Chuyển đổi dữ liệu là quá trình cập nhật các mục nhập định dạng hoặc giá trị để đạt được kết quả được xác định rõ ràng hoặc để làm cho dữ liệu dễ hiểu hơn đối với nhiều đối tượng hơn. Làm giàu dữ liệu đề cập đến việc thêm và kết nối dữ liệu với các thông tin liên quan khác để cung cấp những hiểu biết sâu sắc hơn.

v. Lưu trữ dữ liệu

Sau khi chuẩn bị xong, dữ liệu có thể được lưu trữ hoặc chuyển vào một ứng dụng của bên thứ ba — chẳng hạn như một công cụ kinh doanh thông minh — dọn đường cho quá trình xử lý và phân tích diễn ra.

3. Tạo và triển khai trang tổng quan

3.1. Thực hiện logic chương trình phù hợp để tạo bảng điều khiển

3.1.1. Triển khai trang tổng quan tĩnh mà không cần cập nhật dữ liệu

Trang tổng quan tĩnh là công cụ báo cáo được sử dụng để tóm tắt thông tin thành các dạng đồ họa có thể tiêu hóa được nhằm cung cấp khả năng hiển thị nhanh về hiệu suất kinh doanh. Giá trị nằm ở khả năng của chúng để minh họa các cải tiến hiệu suất liên tục thông qua việc phù hợp với các tính năng trực quan cho người dùng. Tùy thuộc vào mục đích và bối cảnh, cập nhật dữ liệu có thể diễn ra mỗi tháng, một tuần, một lần hoặc thậm chí theo thời gian thực. Trang tổng quan tĩnh có thể đáp ứng cả tính cấp thiết của môi trường có nhịp độ nhanh, cung cấp hỗ trợ dữ liệu thời gian thực và theo dõi các chỉ số hiệu suất chống lại các mục tiêu chiến lược toàn doanh nghiệp dựa trên dữ liệu lịch sử. Tuy nhiên, các trang tổng quan như vậy không liên quan đến người dùng trong quá trình trực quan hóa dữ liệu và có vấn đề với việc xử lý dữ liệu đa chiều và phức tạp. Bảng điều khiển tương tác có thể được coi là một bước hướng tới việc liên quan trực tiếp đến người dùng trong quá trình phân tích. Bảng điều khiển tương

tác không chỉ xem xét các tính năng trực quan mà còn giới thiệu các tính năng chức năng như tương tác điểm và nhấp chuột. Những khả năng này cho phép những người ra quyết định hoạt động thực hiện các phân tích phức tạp hơn. Giải quyết những trở ngại của trang tổng quan tĩnh, chúng được sử dụng để hiểu rõ hơn về bản chất phức tạp của dữ liệu, điều này cũng có thể thúc đẩy việc ra quyết định. Tuy nhiên, lợi ích của tính tương tác có thể làm tăng nỗ lực nhận thức của người dùng và yêu cầu thời gian phân tích thủ công, làm tăng xác suất các quyết định bị trì hoãn hoặc (thậm chí) lỗi.

3.1.2. Triển khai trang tổng quan động với khoảng thời gian cập nhật

- Trang tổng quan động là một loại trang tổng quan dữ liệu cập nhật tự động theo thời gian thực. Bạn cũng có thể nghe chúng được gọi là trang tổng quan tương tác, vì các báo cáo có thể được thay đổi, tổ chức lại và thao tác. Điều đó khác với trang tổng quan tĩnh, chỉ hiển thị một tập dữ liệu cố định.

Hầu hết khi chúng ta sử dụng từ "dashboards", những gì chúng ta thực sự đang nói đến là các bảng điều khiển động. Dưới đây, hãy xem một số cách bạn có thể sử dụng trang tổng quan động để khám phá dữ liệu EHS của mình.

i. Phân tích dữ liệu trong thời gian thực

Như chúng tôi đã nói trước đây, trang tổng quan động tự động cập nhật để hiển thị dữ liệu của bạn khi dữ liệu được thu thập. Thông tin được thu thập trong toàn công ty của bạn trên thiết bị di động và thông qua tích hợp hệ thống tự động hiển thị trong trang tổng quan của bạn gần như ngay lập tức.

Ví dụ: nếu bạn sử dụng hệ thống giám sát khí thải liên tục, bạn có thể xem dữ liệu khí thải khi nó được thu thập. Sử dụng bảng điều khiển tương tác, bạn có thể phân tích xu hướng, đi sâu vào chi tiết theo vị trí, đo lường thực tế so với giới hạn cho phép và theo dõi hiệu suất khí thải liên quan đến thông lượng sản xuất. Điều đó có nghĩa là bạn có thể nhận được thông tin chi tiết ngay sau khi thông tin được nhập vào hệ thống của bạn và phản ứng nhanh chóng.

ii. Đi sâu để có những hiểu biết sâu sắc hơn

Không giống như biểu đồ và đồ thị tĩnh, trang tổng quan động cho phép người dùng tương tác với dữ liệu của họ. Bạn có thể xem chi tiết - hoặc xem dữ liệu cụ thể hơn - về một yếu tố hoặc KPI cụ thể cho đến khi bạn tìm thấy mức độ chi tiết mà bạn yêu cầu.

Giả sử bạn muốn xem thông tin chi tiết hơn về lượng khí thải CO_2 của mình. Bằng cách nhấp vào báo cáo cụ thể đó, bạn có thể truy cập các lớp dữ liệu bổ sung. Bằng cách đó, trang tổng quan động cho phép bạn tìm hiểu sâu hơn về dữ liệu của mình mà không làm lộn xộn chế độ xem dữ liệu chính.

iii. Kéo và thả để tạo chế độ xem dữ liệu tùy chỉnh

Một ưu điểm khác của trang tổng quan động là bạn có thể kiểm soát thông tin được hiển thị. Bạn có thể kéo và thả để thêm hoặc xóa báo cáo khỏi trang tổng quan của mình. Bạn cũng có thể thay đổi phạm vi ngày, loại biểu đồ, kích thước và vị trí của

báo cáo để thu hút sự chú ý của mọi người đến các chỉ số quan trọng nhất. Theo cách đó, trang tổng quan động cho phép bạn trực quan hóa dữ liệu của mình theo cách bạn muốn.

iv. Điều chỉnh trang tổng quan cho người dùng cụ thể

Với trang tổng quan động, bạn có thể xuất bản các trang tổng quan khác nhau cho các nhóm người dùng khác nhau. Vì vậy, bạn có thể xây dựng một bảng điều hành cho các nhà lãnh đạo cấp cao. Hoặc, bạn có thể tạo một bảng điều khiển dành riêng cho người quản lý nhà máy. Điều này giải quyết vấn đề phải tạo một trang tổng quan khác nhau cho từng cá nhân hoặc buộc hàng trăm nhân viên chia sẻ một chế độ xem trang tổng quan tĩnh duy nhất.

Trang tổng quan động cũng cho phép bạn kiểm soát quyền truy cập vào thông tin nhạy cảm. Giám đốc nhà máy chỉ nên xem dữ liệu của cơ sở mà họ giám sát, trong khi giám đốc điều hành nên xem dữ liệu của toàn bộ công ty. Với trang tổng quan động, bạn có thể đặt quyền để người dùng chỉ thấy dữ liệu có liên quan đến vai trò của họ và họ được phép truy cập.

3.2. Cân nhắc các tùy chọn xác thực người dùng cho các khía cạnh bảo mật

3.2.1. Tên người dùng, mật khẩu:

- Giao diện người dùng bảng điều khiển được chia thành nhiều khối: menu tiện ích, menu chính và ngăn nội dung chính.

Phần trên cùng bên phải của màn hình chứa một menu tiện ích. Nó bao gồm các nhiệm vụ chung liên quan đến bảng điều khiển nhiều hơn là đến cụm Ceph. Bằng cách nhấp vào các mục của nó, bạn có thể truy cập các chủ đề sau:

• Thay đổi ngôn ngữ của giao diện người dùng của bảng điều khiển. Bạn có thể chọn từ tiếng Séc, tiếng Anh, tiếng Đức, tiếng Tây Ban Nha, tiếng Pháp, tiếng Bồ Đào Nha, tiếng Trung hoặc tiếng Indonesia.

• Hiển thị danh sách các tác vụ liên quan đến Ceph đang chạy ở chế độ nền.

• Xem và xóa các thông báo trang tổng quan gần đây.

• Hiển thị danh sách các liên kết đề cập đến thông tin về trang tổng quan, tài liệu đầy đủ và tổng quan về API REST của trang tổng quan.

• Quản lý người dùng và vai trò người dùng của trang tổng quan. Tham khảo Chương 14, Quản lý Người dùng và Vai trò trên Dòng lệnh để biết mô tả chi tiết hơn về dòng lệnh.

• Đăng xuất khỏi bảng điều khiển.

3.3. Tạo một chương trình trực quan hóa

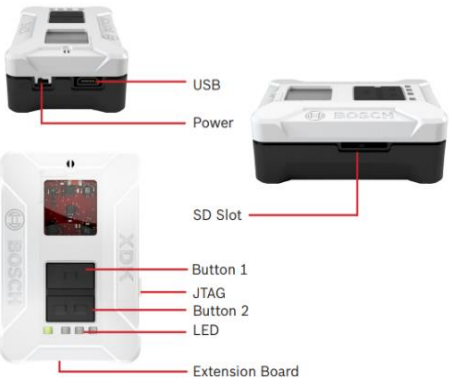
3.3.1. Tạo một chương trình đo lường để lưu trữ dữ liệu cảm biến trong cơ sở dữ liệu

- Trong dự án này, cả dữ liệu môi trường cũng như lượng hàng hóa được sản xuất sẽ được đo lường. Nhận thức đối tượng có thể được đo lường thông qua các loại cảm biến khác nhau. Một khả năng là việc sử dụng cảm biến siêu âm. Các loại cảm biến này được sử dụng để đo khoảng cách. Lợi ích của chúng là chúng có thể được sử dụng

để phát hiện trên hầu hết các bề mặt. Tùy thuộc vào lĩnh vực ứng dụng, có thể có nhược điểm là bán kính phát hiện của chúng có hình nón, có thể dẫn đến hiện tượng xuyên âm. Hơn nữa, một cảm biến hồng ngoại (IR) sẽ được sử dụng. Lợi ích của chúng là chúng có một cơ chế phát hiện hình dạng điểm. Nhược điểm của chúng là một số bề mặt và vật liệu nhất định không phản xạ tín hiệu IR, khiến các cảm biến không thể phát hiện đối tượng. Đối với dữ liệu môi trường, các loại cảm biến khác nhau có sẵn. Bạn cũng có thể chọn hai loại cảm biến. Viết loại cảm biến theo biểu dữ liệu trong bảng dưới đây.

Cảm biến1 Cảm biến2 Cảm biến3 Cảm biến4

- - Bosh XDK là một nền tảng vi điều khiển mạnh mẽ với mảng cảm biến rộng. Nó thường được sử dụng trong các ứng dụng công nghiệp đặc biệt là để theo dõi dữ liệu. Bosh XDK cung cấp mảng cảm biến rộng, mô-đun wifi tích hợp, đầu đọc thẻ sd để lưu trữ dữ liệu trên thẻ nhớ, các nút có thể cấu hình tự do và đầu nối usb để kết nối XDK với PC để nhấp nháy các chương trình mới và xuất trực tiếp các giá trị trong giao diện điều khiển IDE của riêng nó.



Hình 98: Cấu trúc cảm biến XDK

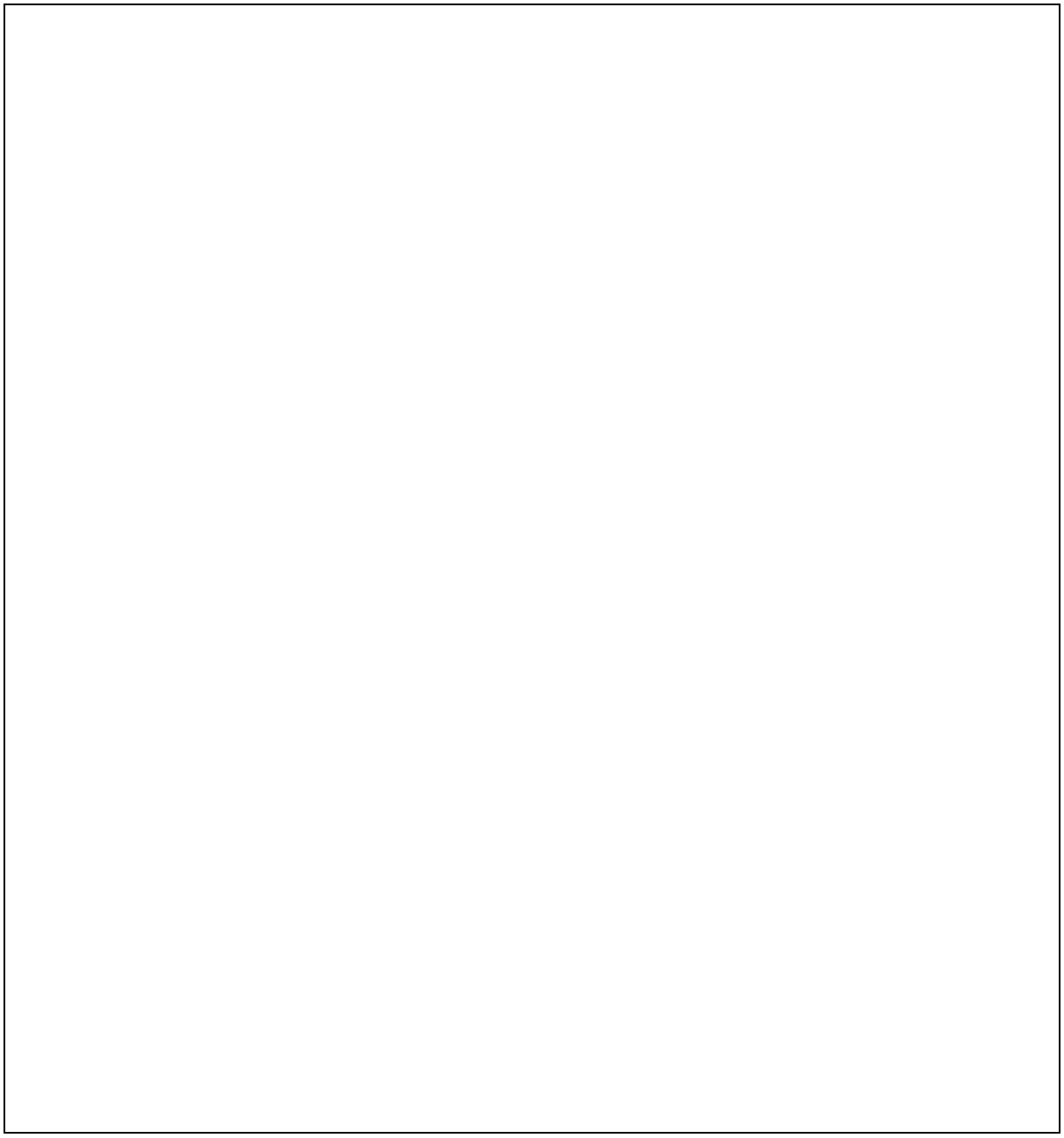
Hình 99: Chức năng cảm biến XDK

Bosh XDK còn cho phép người dùng truy cập nhiều loại cảm biến. Người dùng có thể đo gia tốc, nhiệt độ, độ ẩm, áp suất không khí và nhiều hơn nữa. Hơn nữa, XDK sử dụng “FreeRTOS” làm hệ điều hành thời gian thực. Điều này đảm bảo rằng có thể thực hiện các phép đo được định thời chính xác với các khoảng thời gian xác định.

Bosh hơn nữa sử dụng IDE dựa trên nhật thực riêng được gọi là XDK-Workbench. IDE này cho phép người dùng lựa chọn lập trình XDK trực tiếp với C hoặc sử dụng ngôn ngữ lập trình MITA. Ngôn ngữ lập trình MITA được phát triển để giảm bớt gánh nặng khi tạo các ứng dụng I4.0 mà không có kiến thức về lập trình nhúng. Trang web: https://developer.bosch.com/web/xdk/getting-started#1 do bosh cung cấp và có mã

mẫu để đọc các giá trị cảm biến. Một phần của nhiệm vụ này là đọc qua các mô tả và mã mẫu và sử dụng lại chúng để tạo một chương trình làm việc.
Nhiệm vụ của phần này là tạo một chương trình đo đọc các giá trị của cảm biến. Ít nhất hai cảm biến của XDK phải được sử dụng trong bài tập này. Bạn có thể chọn loại cảm biến nào bạn muốn.
Giá trị đọc trong sẽ được xuất ra trên bàn điều khiển của XDK - Workbench. Dự án sẽ được lập trình bằng ngôn ngữ lập trình MITA. Các phép đo phải được thực hiện sau mỗi 5 giây. Đầu ra bảng điều khiển mong muốn sẽ như sau:
“LOẠI CẢM BIẾN”: XXX ”
“LOẠI CẢM BIẾN”: XXX ”
Nhập ảnh chụp màn hình của đầu ra bảng điều khiển trong bảng bên dưới. Tải lên mã của bạn và ảnh chụp màn hình của đầu ra bảng điều khiển.
Đầu ra bảng điều khiển:

- - Sau khi hệ thống đo lường và truyền thông được thiết lập, hệ thống cơ sở dữ liệu quan hệ có thể được khởi động. Bước đầu tiên, một sơ đồ ERM sẽ được tạo. Đối với sơ đồ ERM, cần phải mô hình hóa tất cả các bộ phận của hệ thống. Cơ sở dữ liệu phải bao gồm tên trạm, tên cảm biến, loại cảm biến, giá trị cảm biến, loại vi điều khiển và dấu thời gian khi thực hiện phép đo. Tất cả các tên phải bằng tiếng Anh. Vẽ sơ đồ ERM dưới đây:



- Sơ đồ ERM là cơ sở của mọi cơ sở dữ liệu. Sau khi nó được tạo Tự tạo cơ sở dữ liệu. Tên cơ sở dữ liệu sẽ là "Trạm đóng chai" hoặc "CSPi4.0" tùy thuộc vào trạm bạn làm việc. Tất cả các tên được sử dụng phải phù hợp với sơ đồ ERM. Tất cả các bảng phải ở dạng chuẩn thứ 3, nếu cần, hãy chuẩn hóa các bảng. Tải lên và bao gồm hình ảnh của tất cả các bảng và hàng đã tạo. Cũng tải lên hình ảnh.

Bảng:

3.3.2. Triển khai trang tổng quan để liên tục đọc và hiển thị bằng đồ thị các giá trị đo được từ cơ sở dữ liệu

Sau khi cơ sở dữ liệu được thiết lập và chương trình đo hoạt động, có thể lưu trữ dữ liệu đo bên trong cơ sở dữ liệu.

Ghi vào cơ sở dữ liệu

Bất cứ khi nào chương trình nhận một giá trị trên MQTT, tất cả các giá trị cảm biến nhận được cộng với dấu thời gian nhận hiện tại sẽ được lưu vào cơ sở dữ liệu. Đảm bảo rằng các bảng và giá trị phù hợp được gửi đến cơ sở dữ liệu. Hơn nữa, đảm bảo rằng dấu thời gian có định dạng phù hợp để phù hợp với cơ sở dữ liệu đã sử dụng của bạn. Dưới đây là một danh sách nhỏ các lệnh SQL cần thiết được trình bày. Nó cũng cần thiết để cung cấp một thư viện để gửi các lệnh SQL đến cơ sở dữ liệu của bạn. Kiểm tra thiết lập của bạn bằng cách tạo một chương trình đọc các giá trị trong 5 phút và lưu các giá trị đo được vào cơ sở dữ liệu. Trước khi bắt đầu đo, hãy đảm bảo chương trình đang chạy. Tải lên mã.

Các lệnh SQL:

Chèn giá trị vào bảng

INSERT INTO *BảngName*

VALUES (*value1*, *value2*, *value3*, ...);

Reading row values from Bảng

SELECT colName FROM BảngName

Đọc từ cơ sở dữ liệu

Sau khi hoàn thành tải tất cả các giá trị cảm biến được lưu trữ từ cơ sở dữ liệu. Tạo hình ảnh trực quan về các giá trị cảm biến. Đặt mọi giá trị cảm biến vào một sơ đồ riêng. Như ô chọn tên cảm biến. Chèn các sơ đồ bên dưới. Tải lên mã và ảnh chụp màn hình của các sơ đồ.

3.3.3. Tạo cơ chế giá trị giới hạn để phát hiện việc vượt quá giá trị giới hạn

- Sau khi các chương trình thử nghiệm được hoàn thành thành công, các cảm biến có thể được đặt ở vị trí mong muốn. Đối với trạm đóng chai, các cảm biến phải được đặt theo cách mà cảm biến siêu âm đo được nếu có chai từ bên cạnh. Cảm biến hồng ngoại phải được sử dụng để đo xem có nắp đậy trên chai hay không. Tìm một vị trí hợp lệ để các cảm biến đáp ứng đầy đủ yêu cầu này. Trên trạm CPSi 4.0, quy trình tương tự sẽ được thực hiện. Thay vì chai, vị trí của các khối sẽ được tìm ra. Đảm bảo điều chỉnh cảm biến IR để nó có thể phát hiện xem khối có lỗ ở phía trên hay không. Tìm một vị trí hợp lệ cho vị trí đặt cảm biến và ghi lại quyết định của bạn bằng hình ảnh. Mô tả thêm bằng từ ngữ của riêng bạn tại sao bạn chọn vị trí này trong bảng bên dưới:

Hình ảnh	Trình bày

3.3.4. Tạo Cơ chế cảnh báo để đưa ra cảnh báo khi giá trị giới hạn bị vượt quá

XDK cung cấp khả năng truyền dữ liệu qua phương thức nhà xuất bản và người đăng ký bằng giao thức MQTT. Vì XDK cung cấp nhiều loại cảm biến nên nhiều cảm biến sẽ được truyền đến máy tính. Trong ví dụ này, tất cả dữ liệu đo được sẽ được chuyển

đổi thành một đối tượng JSON. Đối tượng này sẽ được chuyển qua MQTT. Do đó, sẽ chỉ có một chủ đề để gửi dữ liệu. Dữ liệu cảm biến phải được đặt tên như sau:

- For acceleration
  - AccelX
  - AccelY
  - AccelZ
- For humidity
  - Humidity
- For temperature
  - Temperature
- For Pressure
  - Pressure

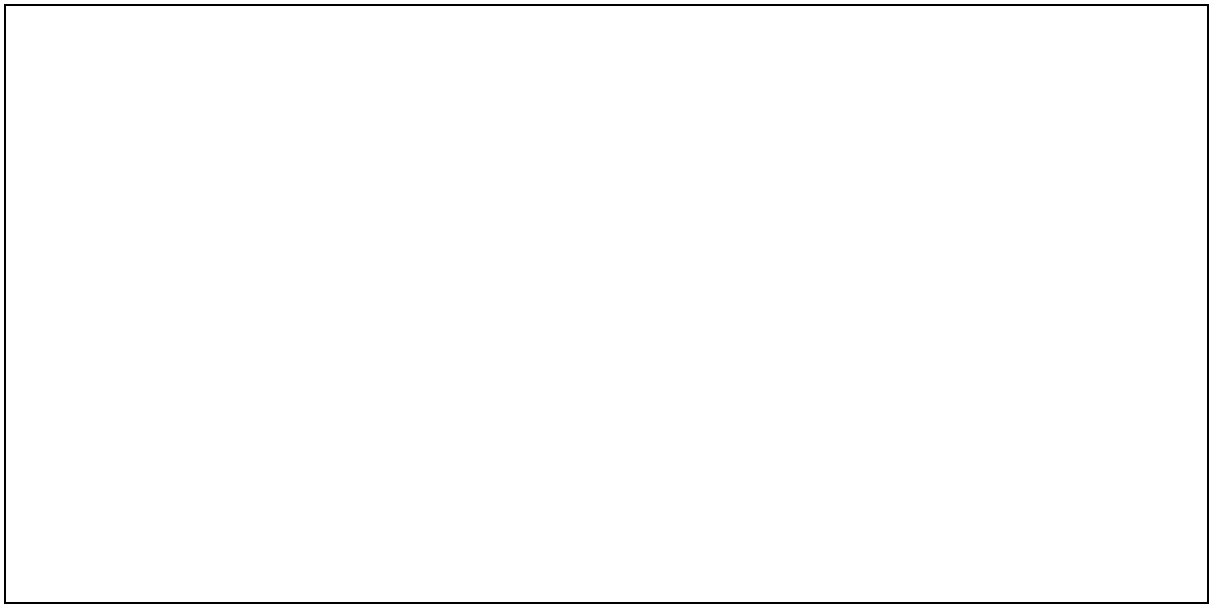
Các bước:

1. Thiết lập nhà môi giới trên máy tính và chạy nhà môi giới
2. Sử dụng chương trình đo của bạn để bộ vi điều khiển đọc các giá trị cảm biến. Để bộ vi điều khiển đọc trong khoảng thời gian 3 giây
3. Bao gồm các thư viện cần thiết trên vi điều khiển và tạo chủ đề để xuất bản.

Một. Tên chủ đề

tôi. Đo lườngXDK / Kỳ

4. Bao gồm các thư viện cần thiết trên máy tính và đăng ký chủ đề nơi các giá trị cảm biến được xuất bản tại
5. In các giá trị đã xuất bản trên bảng điều khiển máy tính
6. Đầu ra bảng điều khiển



Sau khi triển khai ứng dụng web thành công, ứng dụng bảng điều khiển sẽ được tạo. Bảng điều khiển sẽ trực quan hóa các giá trị cảm biến đo được trong cơ sở dữ liệu.

Bảng điều khiển tĩnh

Trong lần thử đầu tiên, một bảng điều khiển tĩnh sẽ được tạo. Các bước sau sẽ được thực hiện:

1. Kết nối với cơ sở dữ liệu
2. Tải các giá trị đã lưu trữ của tất cả các cảm biến và các giá trị tương ứng của chúng
3. Tạo trang tổng quan

Một. Đối với mỗi cảm biến, hãy tạo một sơ đồ

b. Đặt tiêu đề của mọi sơ đồ thành tên cảm biến

C. Đặt trục y là giá trị cảm biến

d. Đặt trục x làm dấu thời gian của giá trị cảm biến tương ứng

4. Đặt nền có màu xám
5. Đối với một trong các cảm biến cũng thêm một đồng hồ đo
6. Trước khi thử nghiệm, hãy để phép đo chạy trong 2 phút

Bao gồm hình ảnh về trang tổng quan của bạn bên dưới. Tải lên mã của bạn và hình ảnh trang tổng quan của bạn.

Bảng điều khiển động

Trong phần mở rộng này, bảng điều khiển sẽ được cập nhật 10 giây một lần. Để đạt được nhiệm vụ này, hãy sử dụng một đối tượng khoảng thời gian với khoảng thời gian 10 giây. Tạo một hàm gọi lại được gọi với khoảng thời gian xác định. Trong chức năng gọi lại, hãy đọc các giá trị cơ sở dữ liệu được cập nhật và cập nhật hình ảnh trực quan của bạn tự động.

Các bước:

1. Kết nối với cơ sở dữ liệu
2. Xác định đối tượng khoảng
3. Tạo trang tổng quan

Một. Tạo cấu trúc cơ sở của bảng điều khiển

b. Đặt nền có màu xám

C. Tạo chức năng gọi lại

tôi. Tải các giá trị đã lưu trữ của tất cả các cảm biến và các giá trị tương ứng của chúng

ii. Đối với mỗi cảm biến, hãy tạo một sơ đồ

iii. Đặt tiêu đề của mọi sơ đồ thành tên cảm biến

iv. Đặt trục y là giá trị cảm biến

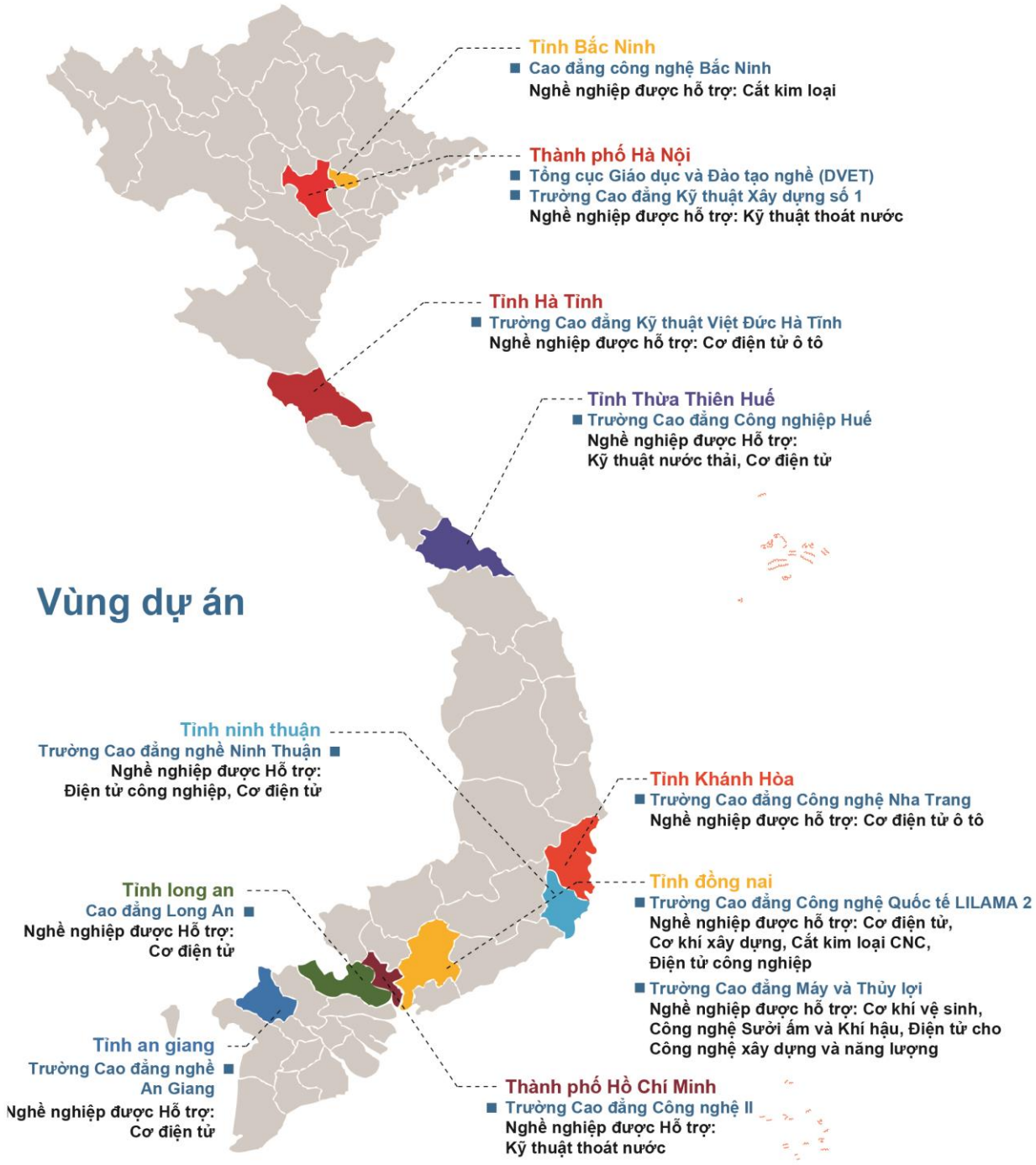
v. Đặt trục x làm dấu thời gian của giá trị cảm biến tương ứng

vi. Cập nhật cốt truyện

4. Bắt đầu đo
5. Triển khai trực quan

PHỤ LỤC

1. VÙNG DỰ ÁN VÀ CÁC TỔ CHỨC ĐỐI TÁC TVET



2. DANH MỤC BIỂU ĐỒ

3. DANH MỤC BẢNG BIỂU

4. DANH MỤC HÌNH MINH HỌA

5. DANH MỤC HÌNH

TÀI LIỆU THAM KHẢO

Stefan Paschek, Vi điều khiển
Stefan Paschek, Bài Tập vi điều khiển
Stefan Paschek , Dự án Vi điều khiển: Thu thập dữ liệu trong thiết lập I4.0
Stefan Paschek, Database
Stefan Paschek, Bài tập Database
Stefan Paschek , Database System
https://www.geeksforgeeks.org
http://www.heptapod.com
https://www.electronicshub.org
https://www.datapine.com
https://www.geckoboard.com
https://www.anodot.com
https://www.toucantoco.com
https://gupea.ub.gu.se
https://www.w3schools.in

Mô-đun đào tạo được xây dựng và phát triển trong khuôn khổ Hợp tác Phát triển với Khối Doanh nghiệp Tư nhân (DPP) Tích hợp các Yếu tố của Ngành Công nghiệp 4.0 vào GDNN. Chương trình DPP được thực hiện bởi các đối tác hợp tác gồm Công ty Bosch Rexroth AG, Trường Cao đẳng Công nghệ Quốc tế LILAMA 2 và Tổ chức Hợp tác Quốc tế Đức (GIZ), được hỗ trợ bởi chương trình *develoPPP* của Bộ Hợp tác Kinh tế và Phát triển Liên bang Đức (BMZ).

Viet Nam, 12/2021