TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Vật Lý cho Công Nghệ Thông Tin

HẦM GIỮ XE THÔNG MINH

BÁO CÁO CÁ NHÂN

Sinh viên thực hiện
Nguyễn Duy Hoàng - 22127126

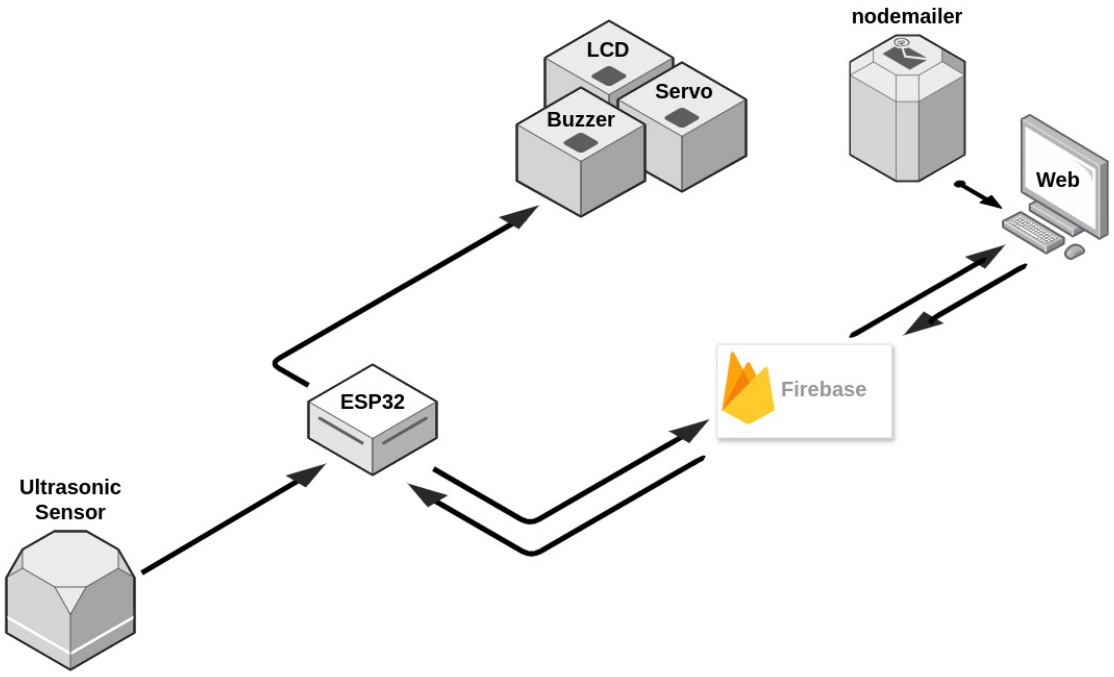
Giảng viên
Thầy Cao Xuân Nam

Ngày 17 tháng 12 năm 2024

1 Thiết bị và chức năng phụ trách

Yêu cầu	Thiết bị / Công cụ
Một thiết bị căn bản (OUTPUT)	Buzzer
Một thiết bị INPUT trong nội dung dạy	Ultrasonic Sensor
Một thiết bị OUTPUT trong nội dung dạy	LCD
Một thiết bị OUTPUT ngoài nội dung dạy	Servo
Gửi thông báo bằng email	SMTP (nodemailer)
Lưu dữ liệu cảm biến trên cloud	Firebase
Hiển thị lịch sử dữ liệu đã lưu trên web	Node.js (Express)

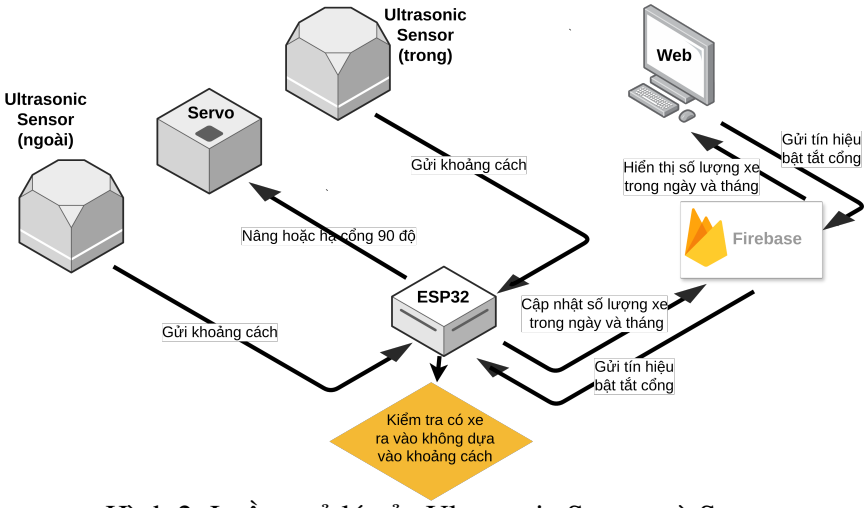
2 Sơ đồ tổng quan của các yêu cầu phụ trách



Hình 1: Luồng xử lý của các thiết bị và chức năng phụ trách

3 Giải thích chi tiết

3.1 Ultrasonic Sensor và Servo – Hệ thống cổng ra vào



Hình 2: Luồng xử lý của Ultrasonic Sensor và Servo

Ultrasonic Sensor được sử dụng để liên tục gửi lên ESP32 khoảng cách từ thiết bị này đến vật gần nhất. Việc liên tục gửi khoảng cách giúp chúng ta kiểm tra xem có xe ra hoặc vào hay không. Có 2 thiết bị này, 1 cái ở ngoài giúp ta kiểm tra xe đi vô và 1 cái ở trong giúp ta kiểm tra xe đi ra.

Servo được sử dụng để nâng hạ cổng cho xe ra vào. Việc nâng hạ được thực hiện dựa vào tín hiệu từ ESP32, khi xe ra vào Servo sẽ nâng lên góc $90°$ so với mặt đất, còn khi bình thường Servo sẽ ở góc cùng phương với mặt đất.

Lắp đặt: – Ultrasonic Sensor bên ngoài có chân **trig** là 33 và chân **echo** là 34.
– Ultrasonic Sensor bên trong có chân **trig** là 23 và chân **echo** là 39 (VN).
– Servo được cắm vào chân số 16.

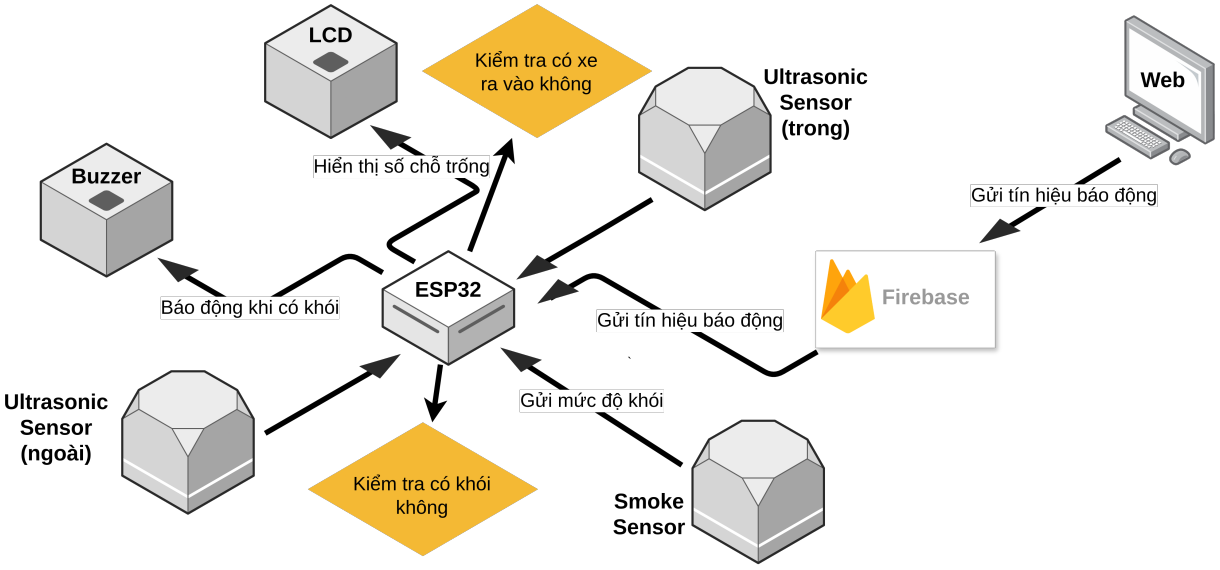
Luồng xử lý:

Khi xe đi vào hầm:
– Xe đi qua Ultrasonic Sensor bên ngoài, khoảng cách thích hợp được gửi đến ESP32.
– ESP32 kiểm tra đủ điều kiện mở cổng và gửi tín hiệu cho Servo nâng cổng lên.
– Xe đi vào và qua Ultrasonic Sensor bên trong, khoảng cách thích hợp được gửi đến ESP32.
– ESP32 kiểm tra đủ điều kiện đóng cổng và gửi tín hiệu cho Servo đóng cổng xuống.
– ESP32 tăng số lượng xe trong ngày và trong tháng rồi cập nhật lên Firebase.
– Firebase tự động gửi dữ liệu mới cho Web để hiển thị cho người dùng.

Khi xe đi ra khỏi hầm:
– Luồng này cũng giống như khi xe đi vào, chỉ là đảo ngược vai trò của Ultrasonic Sensor bên ngoài và bên trong, và việc xe ra sẽ không thay đổi dữ liệu lên Firebase.

Khi nhấn nút bật hoặc tắt cổng trên Web:
– Khi nút được ấn, Web sẽ gửi dữ liệu mới về Firebase và Firebase tự động cập nhật về ESP32.
– Lúc đó chức năng của Ultrasonic Sensor và Servo sẽ bị vô hiệu hóa hoặc kích hoạt.

3.2 LCD và Buzzer



Hình 3: Luồng xử lý của LCD và Buzzer

LCD được sử dụng để hiển thị số lượng chỗ đậu xe còn trống trong hầm. Số lượng chỗ trống nếu không thay đổi thì là giá trị mặc định, giá trị chỉ thay đổi khi có xe ra vào.

Buzzer có chức năng báo động khi cảm biến có khói và nhiệt độ hầm tăng cao. Buzzer kêu khi có tín hiệu khói hoặc tín hiệu từ Website gửi báo động.

Lắp đặt: – LCD có chân **SDA** là 21 và chân **SCL** là 22.
– Buzzer được cắm vào chân số 5.

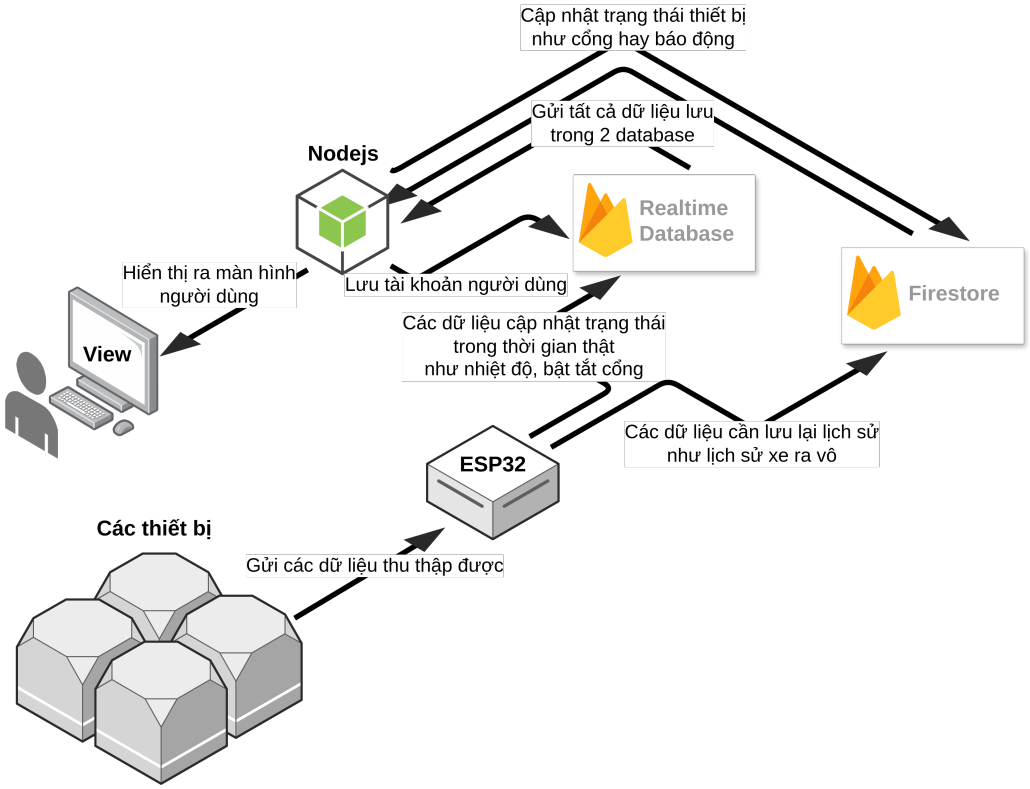
Luồng xử lý:

Hiển thị số lượng chỗ trống trên LCD:
– Ban đầu, LCD sẽ hiện số lượng chỗ trống mặc định.
– Khi có xe ra vào (biết nhờ hệ thống cổng), số lượng sẽ được cập nhật.
– LCD dùng thông tin đó để hiển thị lại cho người dùng.

Báo động khi có khói:
– ESP32 nhận giá trị đo mức độ khói từ Smoke Sensor liên tục và sử dụng nó để kiểm tra.
– Khi ESP32 kiểm tra thấy có khói, tín hiệu được gửi đến cho Buzzer kêu.
– Khi ESP32 kiểm tra hết khói, ESP32 sẽ dừng Buzzer.

Báo động khi có tín hiệu từ Web:
– Người dùng ấn nút SIREN trên Web để kích hoạt báo động.
– Web gửi dữ liệu mới về Firebase và Firebase cập nhật dữ liệu đó xuống ESP32.
– Khi ESP32 kiểm tra thấy tín hiệu được bật, Buzzer sẽ được bật lên báo động.

3.3 Lưu trữ dữ liệu trên cloud và hiển thị ra web



Hình 4: Luồng xử lý lưu trữ trên cloud và hiển thị ra web

Firebase Realtime Database được sử dụng để lưu các giá trị thay đổi trong thời gian thực ví dụ như nhiệt độ, trạng thái của cổng, trạng thái báo động. Các giá trị được lưu vào các Node trên database này, ví dụ như trạng thái cổng được lưu vào Node ‘state‘ có Node cha là ‘BARRIER‘. Vì thế, nếu ta muốn lấy ra giá trị này, ta sẽ sử dụng ‘BARRIER\state‘.

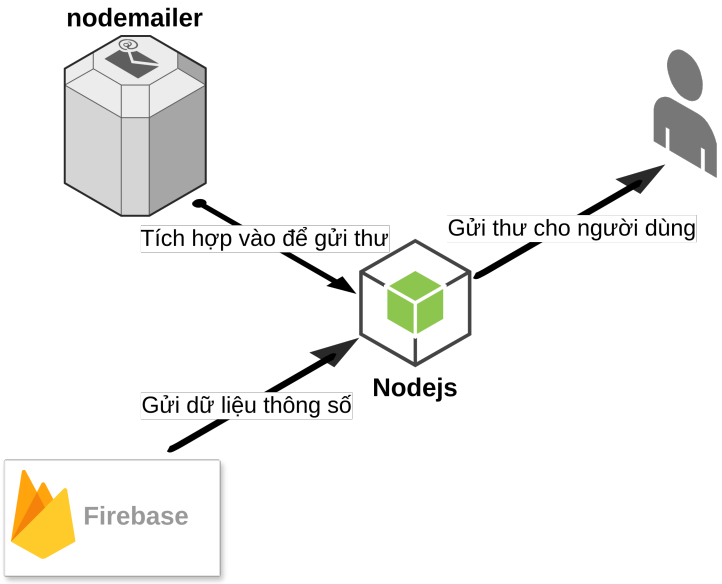
Firestore Database được sử dụng để lưu các dữ liệu nhiều thông tin hoặc lịch sử như tài khoản người dùng hay lịch sử xe ra vào. Một giá trị được lưu vào Database được gọi là một Doc, và các Doc thì có thể thuộc một Collection nào đó, ví dụ như Collection ‘users‘ để lưu tài khoản người dùng, lúc đó, 1 người dùng tương ứng 1 Doc có id là ‘abc‘ có các thông tin như tên, email,...

Node.js được sử dụng để xây dựng website với Express.js để xử lý phần backend, đó cũng là phần tương tác với các Firebase Database như cập nhật, lưu trữ và lấy dữ liệu để hiển thị cho người dùng. Bên cạnh đó, để tương tác với Firebase thì phải cần thư viện ‘firebase-admin‘ để xử lý.

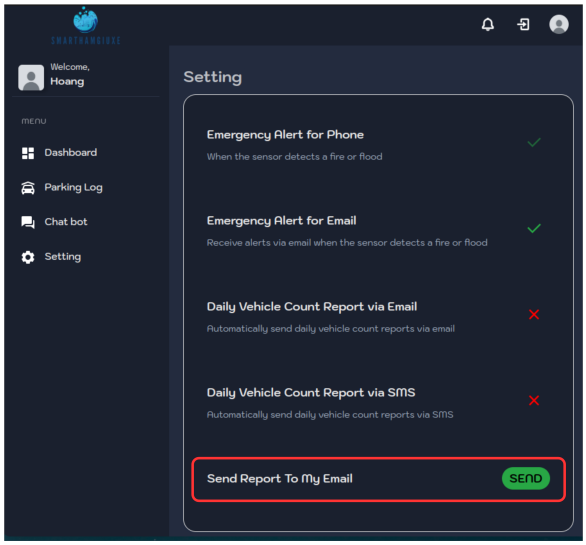
ESP32 tương tác với Firebase bằng thư viện ‘FirebaseClient.h‘. Realtime Database của Firebase có cơ chế stream tức là theo dõi sự thay đổi của các Node, khi có thay đổi xảy ra sẽ gửi tín hiệu lên cho ESP32 để thực hiện 1 hành động thích hợp.

Luồng xử lý:
– Các thiết bị gửi dữ liệu hiện tại cho ESP32.
– Nếu có thay đổi mới, ESP32 cập nhật dữ liệu xuống Firebase.
– Express.js, đang theo dõi, bắt được thông tin mới từ Firebase và hiển thị lên màn hình.

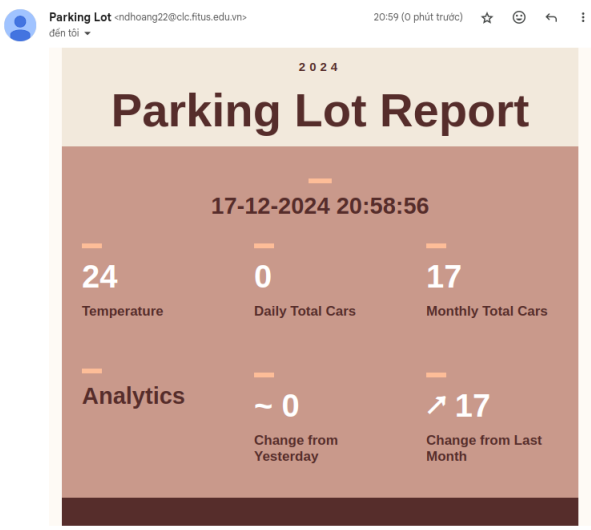
3.4 Gửi thông báo bằng email



Hình 5: Luồng xử lý của gửi email



Hình 6: Nút gửi email báo cáo cho người dùng



Hình 7: Một email báo cáo được gửi

Nodemailer là một thư viện Node.js sử dụng phương thức mặc định là SMTP để hỗ trợ gửi email. Đây cũng là phương thức được sử dụng trong đồ án. Bên cạnh đó, chúng ta cũng cần cài đặt dịch vụ API của Gmail để sử dụng.

Luồng xử lý: – Người dùng ấn nút SEND để gửi email (người dùng có email hợp lệ).
– Node.js lấy các dữ liệu từ Firebase.
– Sau đó, Node.js dùng thư viện nodemailer để gửi email cho người dùng.