TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM

KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN KỸ THUẬT MÁY TÍNH – VIỄN THÔNG

NGÀNH CÔNG NGHỆ KỸ THUẬT MÁY TÍNH

BÁO CÁO ĐỒ ÁN MÔN HỌC

MÔN HỌC: ĐỒ ÁN 1

THIẾT KẾ VÀ THI CÔNG HỆ THỐNG PHÁT HIỆN NGỦ GẬT TRÊN Ô TÔ KẾT HỢP CẢNH BÁO SMS.

GVHD: TS. Huỳnh Thế Thiện

SVTH: Hồng Lý Trung Nhân

MSSV: 21119109

TP.HCM, tháng 5 năm 2024

BẢN NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Tp.HCM, ngày tháng năm 2024

Giảng viên ký tên

TS. Huỳnh Thế Thiện

LỜI CẢM ƠN

Để hoàn thành được đồ án môn học này, em xin chân thành cảm ơn các thầy, cô ở khoa Điện - Điện tử ở trường đại học Sư Phạm Kỹ Thuật TP.HCM nói chung và bộ môn Kỹ thuật máy tính – viễn thông nói riêng vì đã hỗ trợ em trong việc trang bị các kiến thức nền tảng và các trang thiết bị trong suốt quá trình học tập, nghiên cứu vừa qua.

Đặc biệt, em xin chân thành cảm ơn sự hướng dẫn nhiệt tình của thầy TS. Huỳnh Thế Thiện cùng với những lời góp ý, giải đáp thắc mắc và đưa ra những nhận xét giúp em có cái nhìn đúng hơn về kiến thức chuyên ngành. Qua những điều đó, cho em được những kiến thức, kinh nghiệm trong việc hoàn thành đồ án này.

Em xin chân thành cảm ơn!

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

DANH MỤC CÁC TỪ VIẾT TẮT

VIẾT TẮT	NGUYÊN NGHĨA
SMS	Short Message Services
AI	Artificial Intelligence
USB	Universal Serial Bus
CNN	Convolutional Neural Network
CDC	Communication Device Class
RAM	Random Access Memory
CPU	Central Processing Unit
GPU	Graphics Processing Unit
I/O	Input/Output
LPDDR4	Low-Power Double Data Rate Synchronous Dynamic Random Access Memory
SDRAM	Synchronous Dynamic Random Access Memory
GPIO	General Purpose Input Output
UART	Universal Asynchronous Receiver / Transmitter
SPI	Serial Peripheral Interface
I2C	Inter – Integrated Circuit
ADC	Analog-to-Digital Converter
HDMI	High-Definition Multimedia Interface
GPS	Global Positioning System

CHƯƠNG 1: GIỚI THIỆU.

1.1. GIỚI THIỆU.

Hiện nay các hình thức di chuyển đường dài bằng ô tô ngày càng phổ biến, từ đó sinh ra trên báo đài ghi lại các trường hợp các tài xế rơi vào trạng thái "Giấc ngủ trắng" khi đi đường dài nhiều hơn. Hiện này này là khi tài xế rơi vào trạng thái ngủ nhưng lại không biết mình đang ngủ, do là bộ não và cơ thể vẫn còn hoạt động vì bộ não phải ra lệnh cho cơ thể không ngừng lái xe nhưng các bộ phận này không nghe lời tuyệt đối.

Thay vì biết mình phải nên nghỉ ngơi thì một vài tài xế vẫn cố tình sử dụng các biện pháp tạm thời để giúp mình thoát khỏi buồn ngủ nhưng không có hiệu quả ví dụ như là mở cửa sổ, hoặc mở nhạc to để giúp bản thân tỉnh ngủ, và cứ cố gắng đặt mình và những người cùng trên chuyến xe đó vào trong trạng thái nguy hiểm.

Trong số thanh thiếu niên, 50 đến 70 phần trăm thừa nhận lái xe buồn ngủ trong năm qua và 15 phần trăm báo cáo làm như vậy ít nhất một lần mỗi tuần. Buồn ngủ tự báo cáo trong khi lái xe có liên quan đến sự gia tăng 2,5 lần nguy cơ tai nạn xe cơ giới tương đối.

Lái xe buồn ngủ chiếm khoảng một trong sáu căn nguyên tai nạn nghiêm trọng và một trong tám căn nguyên tai nạn dẫn đến nhập viện tài xế hoặc hành khách. Tỷ lệ phần trăm cao này phù hợp với việc quan sát các vụ tai nạn lái xe buồn ngủ xảy ra ở tốc độ cao, mà không có các thao tác tránh né như phanh hoặc chuyển hướng, có thể giảm thiểu mức độ nghiêm trọng của tai nạn.

Chính vì thế em chọn đề tài **"Thiết kế hệ thống phát hiện ngủ gật trên ô tô kết hợp cảnh báo SMS."** có thể giúp giảm thiểu tình trạng ngủ gật trên xe tránh gây các tai nạn giao thông.

1.2. MỤC TIÊU ĐỀ TÀI.

Dựa trên các tình huống thực tế, các tình huống ngủ gật thường xảy ra nhiều hơn trên ô tô, xe tải, xe khách,… Ngoài ra, các trường hợp ngủ gật thường sẽ theo chiều hướng cố gắng mở mắt trong trạng thái buồn ngủ hoặc rất buồn ngủ, hệ thống này sinh ra là để nhắc nhở tài xế có thể nghỉ ngơi ở bên lề đường, trạm dừng chân,… Ngoài ra nếu như trường hợp tài xế có nguy cơ như là ngất xỉu, đột quỵ, hoặc đơn giản chỉ là ngủ quá lâu thì sẽ gửi tin nhắn SMS để cứu hộ gần đó, cơ bản nhất là gửi biển số xe của xe đó trong tin nhắn để giúp các nhân viên cứu hộ can thiệp kịp thời. Ngoài ra, sẽ có những trường hợp các xe đi vào đường tối, điều này làm cho tài xế dễ bị buồn ngủ hơn.

Ở đề tài này nhằm giải quyết tốt nhất có thể những vấn đề trên khi có xảy ra những vấn đề được kể ở trên:

- Theo dõi trạng thái buồn ngủ của tài xế thông qua camera.
- Bật đèn tạm thời tự động khi đi vào đường tối.
- Cảnh báo cho tài xế biết là mình đang bị ngủ và cần được nghỉ ngơi.
- Gửi cảnh báo thông qua SMS đến các trạm cứu hộ để có thể kịp thời can thiệp.
- Gửi cảnh báo bằng đèn tín hiệu rẽ của xe để cho những xe xung quanh biết là xe này đang có thể gây ra nguy hiểm cho các xe khác.

1.3. GIỚI HẠN ĐỀ TÀI.

Tuy vậy, hệ thống này vẫn còn rất nhiều rất nhiều những giới hạn, điều này gây ra nhiều phiền phức cho người sử dụng trong quá trình sử dụng.

- Không thể phát hiện được mặt tài xế khi mang khẩu trang, kính mát.
- Hệ thống này hiện tại chưa thể tự động hoạt động khi cấp nguồn (cần phải thông qua quá trình khởi động bởi người phát triển).
- Đôi khi hệ thống này sẽ hoạt động không tốt khi sử dụng mắt kính thông thường như là kính cần (không phát hiện được mắt do ánh sáng phản xạ qua mắt kính,…).
- Chưa có camera phục vụ cho môi trường tối thay vì phải sử dụng đèn trợ sáng.
- Chưa phát hiện hoàn toàn mọi tư thế ngủ của tài xế.

1.4. PHƯƠNG PHÁP NGHIÊN CỨU.

Để có thể tiếp cận được với đề tài này thì sẽ có rất nhiều khó khăn xảy ra, để giải quyết những khó khăn đó thì em sẽ liệt kê ra những phương pháp sau đây:

- **Phương pháp tìm kiếm tài liệu, lý thuyết:** Ở phương pháp này, em sẽ tham khảo những tài liệu, bài báo trước đó về đề tài này, thu thập các tập dữ liệu, cách thức thiết kế mô hình AI, nhúng mô hình này vào kit Raspberry Pi 4 và giao tiếp Kit này với STM32F103C6T8 để mục đích xử lí các cảnh báo theo mỗi mức độ thông qua chuẩn truyền nối tiếp bằng USB.
- **Phương pháp thực nghiệm và kiểm thử:** Ở phương pháp này thì em sẽ thực hiện bằng cách giao tiếp với từng module riêng lẻ sau đó sẽ kiểm tra đạt yêu cầu hay chưa, nếu chưa thì thực hiện lại cho đúng yêu cầu sau đó mới qua bước tiếp theo là sẽ gôm các module này chạy chung với nhau, đánh giá độ chính xác, nếu chưa đạt sẽ tinh chỉnh cho phù hợp nếu được sau đó sẽ test toàn bộ các trường hợp với hệ thống này và đánh giá một cách trực quan nhất có thể.

- **Phương pháp phân tích và đưa ra kết luận:** ở phương pháp này em sẽ thu thập các dữ liệu nhận được thông qua quá trình thử nghiệm trước đó. Phân tích và đánh giá độ tin cậy của hệ thống này.

1.5. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU.

- Nghiên cứu về mạng CNN cho dự đoán mắt nhắm, mắt mở, face recognition cho phát hiện khuôn mặt.
- Nghiên cứu về tập face landmarks detection.
- Nghiên cứu cách cài đặt môi trường và thực thi mô hình này trên raspberry pi 4.
- Nghiên cứu về stm32f103c8t6 cùng với các ngoại vi có liên quan.
- Nghiên cứu về giao thức USB CDC trên kit stm32f103c8t6.
- Nghiên cứu về module SIM900A cùng với các linh kiện còn lại như ULN2003.

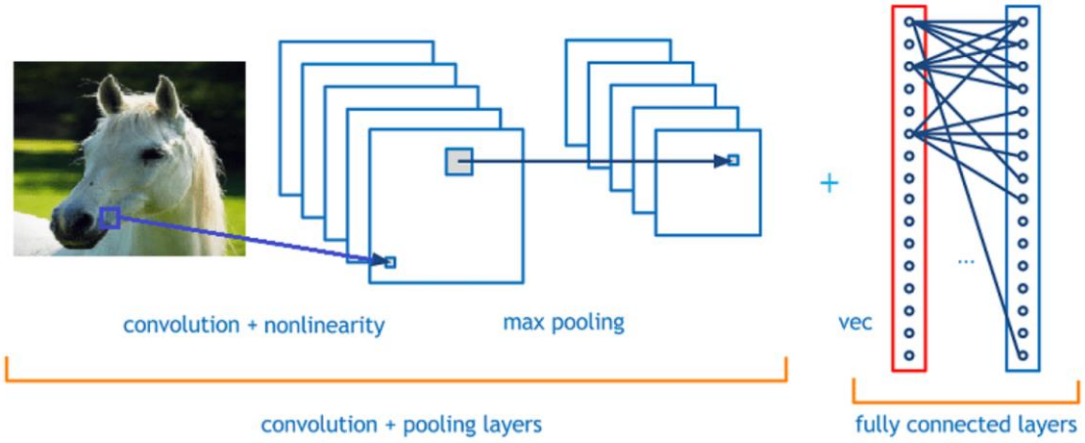
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT.

2.1. MẠNG NƠ RON TÍCH CHẬP VÀ ỨNG DỤNG TRONG PHÁT HIỆN NGỦ GẬT.

2.1.1. Mạng nơ ron tích chập (Convolution neural network).

CNN sử dụng phép toán tích chập để tạo liên kết giữa các layers trong mạng neural. Mỗi neuron, thay vì kết nối đến tất cả các neurons khác thì sẽ chỉ kết nối đến một vài neurons đại diện. Điều này giúp cho CNN học được các mối liên hệ không gian của dữ liệu tốt hơn. Đó là lý do mà vì sao CNN lại được sử dụng rất phổ biến trong các bài toán về Computer Vision như phân loại hình ảnh, phát hiện đối tượng trong ảnh/video, …

Một vài kiến trúc mạng CNN kinh điển có thể kể đến như: LeNet, AlexNet, VGG, ResNet, MobileNet,Yolo, SSD,…



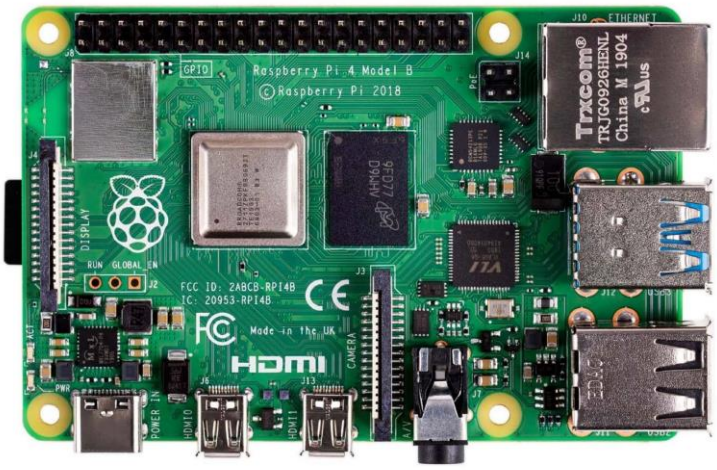
Hình 1: Cấu trúc mạng neurals tích chập.

2.1.2. Ứng dụng mạng CNN trong phát hiện ngủ gật.

Với ý tưởng là để phân loại xem là tài xế có đang ngủ hay không, bằng cách là sẽ trích xuất 2 vùng mắt của tài xế thành 1 ảnh sau đó được qua mô hình mạng CNN với 2 lớp phân loại để phân loại xem ảnh đó thuộc lớp mắt nhắm hay mắt mở. CNN có công dụng trong việc trích xuất đặc trưng xem coi ảnh đó đang là nhắm mắt hay mở mắt. Khi đó có thể ứng dụng mạng CNN cho việc dự đoán mắt mở hay nhắm.

2.2. KIT RASPBERRY PI 4.

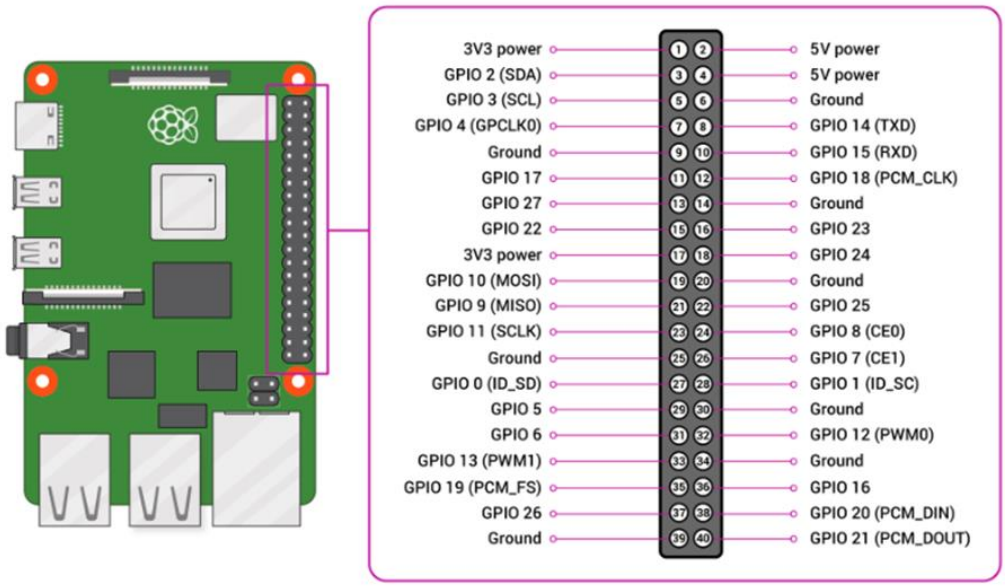
Raspberry Pi 4 Model B là phiên bản đầu tiên trong thế hệ mới của máy tính Raspberry Pi, hỗ trợ thêm RAM và có hiệu suất CPU, GPU và I/O được cải thiện đáng kể; tất cả trong một hình dạng tương tự, nguồn điện và chi phí như thế hệ trước đó của Raspberry Pi 3B+. Pi4B có sẵn với 1, 2 và 4 Gigabyte của LPDDR4 SDRAM.



Hình 2: Raspberry pi 4 model B.

• Thông số kỹ thuật:

- Bộ xử lý: Quad core 64-bit ARM-Cortex A72 chạy với tốc độ lên đến 1.5GHz.
- Bộ nhớ: RAM LPDDR4 được nâng cấp, có các tùy chọn là 1GB, 2GB hoặc 4GB.
- Đầu ra Video: Hỗ trợ đầu ra video 4K qua cổng micro HDMI (2 cổng).
- Hỗ trợ mạng không dây dual-band 2.4/5.0 GHz và Bluetooth 5.0.
- Cổng Gigabit Ethernet (hỗ trợ PoE với tiện ích PoE HAT).
- 2 cổng USB 3.0.
- 2 cổng USB 2.0.
- Khe cắm thẻ nhớ microSD.
- 40 chân GPIO.
- Nguồn điện: Nguồn cung cấp qua cổng Micro-USB hoặc cổng GPIO. Bộ cấp nguồn USB Type-C, ở mức 5V / 3A. Nếu các thiết bị USB được kết nối tới tiếp theo tiêu thụ dòng điện dưới 500mA, thì nguồn điện có thể sử dụng là 5V, 2.5A.
- Hệ điều hành hỗ trợ: Raspberry Pi OS (trước đây là Raspbian), Ubuntu, các phiên bản Linux khác, Windows 10 IoT Core,...
- Hệ thống lưu trữ: Không có bộ nhớ trong tích hợp nhưng có thể sử dụng thẻ nhớ microSD hoặc kết nối với ổ cứng ngoại vi thông qua các cổng USB.



Hình 3: Sơ đồ chân của Raspberry Pi 4 Model B.

• Mô tả chân:

- Raspberry Pi 4 có thể sử dụng trong hệ thống nhúng bên ngoài để giao tiếp tín hiệu. Có tổng cộng 40 chân , trong đó 28 chân là chân GPIO và các chân còn lại là chân nguồn.
- Các chân GPIO không chỉ thực hiện các chức năng I/O đơn giản mà còn hỗ trợ giao thức UART, SPI và I2C. Các giao thức này dành riêng cho mọi chân và tất cả chức năng của chúng như sau:

+ Chân cấp nguồn: 2 chân 2, 4 cấp nguồn 5V.

2 chân 1, 17 cấp nguồn 3.3V.

+ Chân nối đất: 8 chân 6, 9, 14, 20, 25, 30, 34 và 39.

- Chức năng:

Raspberry Pi 4 là một máy tính nhỏ gọn và mạnh mẽ trong dòng sản phẩm Raspberry Pi. Nó có nhiều chức năng và ứng dụng đa dạng. Dưới đây là một số chức năng chính của Raspberry Pi 4:

+ Máy tính nhúng: Raspberry Pi 4 có thể hoạt động như một máy tính nhúng đa năng, với khả năng chạy các hệ điều hành như Raspbian (một phiên bản tùy chỉnh của Linux), Ubuntu, Windows 10 IoT Core và nhiều hệ điều hành khác.

+ Quảng cáo kỹ thuật số: Raspberry Pi 4 có thể được sử dụng để tạo và quản lý hệ thống quảng cáo kỹ thuật số, giúp hiển thị nội dung đa phương tiện trên các màn hình thông qua kết nối HDMI.

+ Trung tâm giải trí đa phương tiện: Với khả năng xử lý đa phương tiện mạnh mẽ, Raspberry Pi 4 có thể được sử dụng như một trung tâm giải trí đa phương tiện. Bạn có thể cài đặt các ứng dụng như Kodi để xem phim, nghe nhạc và xem hình ảnh trên TV hoặc màn hình.

+ Thiết bị mạng: Raspberry Pi 4 có thể được sử dụng như một router hoặc điểm phát Wi-Fi. Bạn có thể cài đặt các phần mềm như Pi-hole để làm DNS server và chặn quảng cáo trên toàn mạng.

+ Hệ thống giám sát: Với các cảm biến và giao diện GPIO tích hợp, Raspberry Pi 4 có thể được sử dụng để xây dựng các hệ thống giám sát như hệ thống an ninh, giám sát môi trường, theo dõi nhiệt độ, độ ẩm, ánh sáng, và nhiều hơn nữa.

+ Học tập và phát triển: Raspberry Pi 4 là một công cụ tuyệt vời để học tập và phát triển trong lĩnh vực công nghệ. Người dùng có thể tìm hiểu về lập trình, điều khiển thiết bị, phát triển ứng dụng IoT và nhiều lĩnh vực khác.

+ Máy tính cá nhân: Với cấu hình mạnh mẽ, Raspberry Pi 4 có thể được sử dụng như một máy tính cá nhân để thực hiện các nhiệm vụ thông thường như duyệt web, xem phim, soạn thảo văn bản.

2.3. KIT STM32F103C8T6 VÀ GIAO THỨC USB CDC.

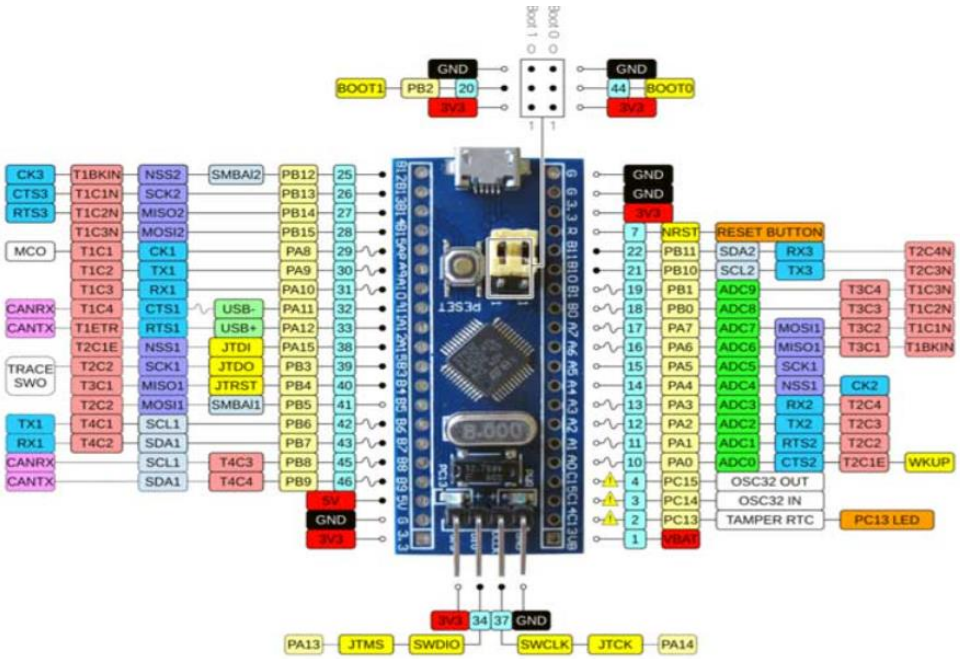
2.3.1. Giới thiệu kit STM32F103C8T6.

Bo mạch STM32F103C8T6 blue-pill được trang bị bộ xử lí STM32F103C8T6 lõi ARM 32-bit Cortex-M3 RISC và thạch anh ngoại 8MHz và thạch anh nội 32kHz . Đây là chip flash CMOS với 37 chân GPIO và 10 chân Analog, hỗ trợ các giao thức giao tiếp tiên tiến như CAN và USB.

Các thiết bị ngoại vi trên bo mạch cung cấp khả năng điều khiển vượt trội, hoạt động với điện áp rất thấp 3.3V, làm cho nó lý tưởng cho các ứng dụng tiêu thụ điện năng thấp. Bo mạch còn được trang bị bộ watchdog và window watchdog timer, giúp đảm bảo hoạt động chính xác của các dòng lệnh.

Thông số kỹ thuật:

- Điện áp hoạt động: từ 2.0V - 3.6V.
- Nhiệt độ hoạt động: từ -40°C đến 105°C.
- Bộ tạo dao động bên trong: 4-16 MHz.
- SRAM: 20 kB.
- Bộ nhớ flash: 64 kB.
- Tốc độ CPU: 72 MHz (tối đa).
- Dòng điện sink/source: 6 mA.
- Tích hợp thạch anh ngoài 8MHz.
- Có cổng USB tích hợp để nạp Bootloader hoặc sử dụng các thiết bị ngoại vi USB.
- Đèn LED tích hợp trên chân PC13.
- Hỗ trợ chuẩn giao tiếp SWD cho các mạch nạp như ST-link và J-link.
- Được tích hợp thêm bộ đồng hồ thời gian thực nội trong bộ xử lý.



Hình 4: Sơ đồ chân của STM32F103C8T6 blue-pill.

Kiểu chân	Tên chân	Mô tả
Power	3.3V	Điện áp hoạt động đầu ra
	5V	Chân cấp nguồn ở cổng USB hoặc nguồn 5V bên ngoài
	GND	Chân nối đất
Chân Analog	PA0-PA7 và PB0-PB1	Chân ADC độ phân giải 10, 12-bit
Chân I / O	PA0-PA15, PB0-PB15, PC13-PC15	37 chân I / O đa chức năng
Chân ngắt	PA0-PA15, PB0-PB15, PC13-PC15	Ngắt ngoài
PWM	PA0-PA3, PA6-PA10, PB0-PB1, PB6-PB9	Điều chế độ rộng xung
UART	TX1, RX1, TX2, RX2, TX3, RX3	Chân RTS, CTS USART
SPI	MISO0, MOSI0, SCK0, MISO1, MOSI1, SCK1, CS0	2 chân SPI
CAN	CAN0TX, CAN0RX	Chân Bus của mạng CAN

I2C	SCL1, SCL2, SDA1, SD2	Chân dữ liệu I2C và chân xung nhịp
Đèn LED tích hợp	PC13	LED chỉ thị

Bảng 1: Bảng mô tả chân của kit STM32F103C8T6.

Trong đó:

Ngắt ngoài: Ngắt phần cứng được kích hoạt khi phát hiện sự thay đổi trong các tín hiệu ngoại vi.

PWM: Có 15 chân hỗ trợ điều chế độ rộng xung để tạo tín hiệu điện áp tương tự từ đầu ra PWM kỹ thuật số.

RTS/CTS: Request-to-Send/Clear-to-Send là một giao thức để kiểm soát việc truyền và nhận dữ liệu.

SPI: Giao thức để giao tiếp giữa vi điều khiển và các thiết bị ngoại vi.

CAN: Bus truyền dữ liệu theo hai hướng.

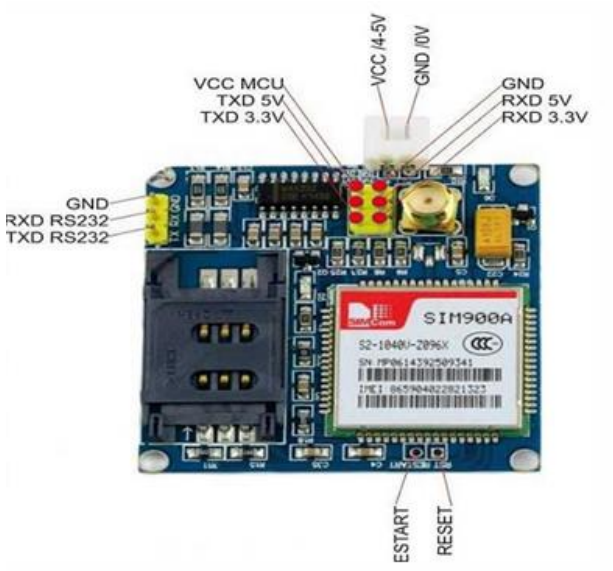
I2C: Giao thức truyền dữ liệu nối tiếp để truyền dữ liệu đồng bộ.

2.3.2. Giới thiệu giao thức USB CDC.

USB CDC hoặc là USB **C**ommunications **D**evice **C**lass là một lớp thiết bị giao tiếp mạng thông qua truyền nối tiếp trong chuẩn USB. Các thiết bị USB này đóng vai là một giao diện mạng phục vụ cho công việc giao tiếp máy tính. Các thiết bị này có thể truyền các gói tin như là dữ liệu giao tiếp với nhiều phương thức vật lý khác nhau.

Ở trong STM32F1 cũng đã có tích hợp giao thức này bằng cách sử dụng USB như là một thiết bị mạng truyền thông tin qua lại với máy tính như một cổng COM ảo. Điều này giúp đơn giản hóa việc truyền/nhận dữ liệu và giảm thiểu chi phí phần cứng so với việc sử dụng giao thức UART truyền thống.

2.4. MODULE SIM 900A.



Hình 5: Giao diện Module Sim 900A.

Thông số kỹ thuật:

- Điện áp hoạt động: 4.7-5V.
- Điện năng tiêu thụ thấp: 1.5mA (ở chế độ ngủ).
- Nhiệt độ hoạt động: -40 – 85 °C.
- Điều khiển qua tập lệnh AT (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands).
- Băng tần kép 900/ 1800 MHz.
- GPRS multi-slot class 10/8.
- GPRS mobile station class B.

Các lệnh của module SIM900A tương tự như module SIM800A :

- Lệnh: AT<CR><LF>. Là lệnh dùng để kiểm tra xem module SIM900A có hoạt động không. Nếu trả về OK tức là module hoạt động bình thường và ngược lại thì module không hoạt động.
- Lệnh: AT+IPR=[baud rate]<CR><LF>. Dùng để cài đặt tốc độ giao tiếp dữ liệu với Module Sim và chỉ cài được các tốc độ sau baud rate : 0 (auto), 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
- Lệnh: AT&W<CR><LF>. Dùng để lưu lại các câu lệnh đã cài đặt.

Các lệnh điều khiển cuộc gọi:

- Lệnh: AT+CLIP=1<CR><LF>. Dùng để hiển thị thông tin cuộc gọi đến.
- Lệnh: ATD[Số_điện_thoại];<CR><LF>. Là lệnh để thực hiện cuộc gọi.
- Lệnh: ATH<CR><LF>. Là lệnh dùng để kết thúc cuộc gọi hoặc cúp máy khi có cuộc gọi khác đến.

- Lệnh: ATA<CR><LF>. Là lệnh để chấp nhận khi có cuộc gọi đến.

Các lệnh điều khiển tin nhắn:

- Lệnh: AT+CMGF=1<CR><LF>. Là lệnh đưa SMS về chế độ Text và phải có lệnh này thì mới gửi nhận tin nhắn dưới dạng Text được.

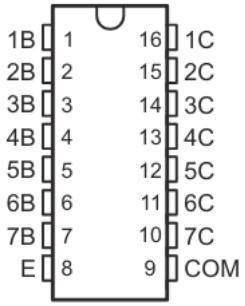
- Lệnh: AT+CMGS="Số_điện _thoại"<CR><LF>. Là lệnh dùng để gửi tin nhắn. Gửi mã Ctrl+Z hay 0x1A hoặc giá trị 26 để kết thúc nội dung và gửi tin nhắn.

- Lệnh: AT+CMGDA="DEL ALL"<CR><LF>. Dùng để xóa toàn bộ tin nhắn trong các hộp thư.

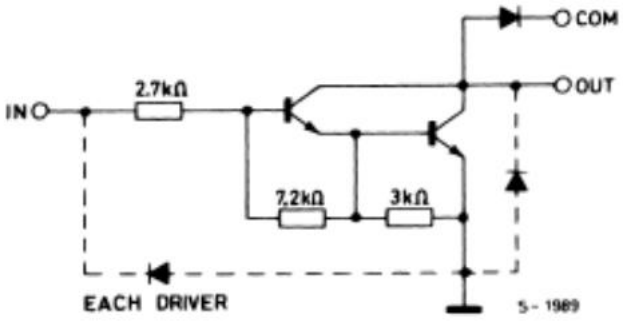
- Lệnh: AT+CNMI=2,2<CR><LF>. Dùng để hiển thị nội dung tin nhắn ngay khi có tin nhắn đến.

2.5. ULN2003.

ULN2003 là một IC phổ biến được sử dụng để điều khiển các tải dòng cao. Đây là một mạch ghép Darlington, chứa 7 cặp transistor Darlington với các diode bảo vệ tích hợp.



Hình 6: Sơ đồ chân của ULN2003.



Hình 7: Sơ đồ nguyên lý của 1 cặp In – Out của ULN2003.

Tên chân	Công dụng
Input 1 - 7	Đây là 7 chân ngõ vào của cặp Darlington, mỗi chân được kết nối với cực gốc của transistor và có thể được kích hoạt bằng cách sử dụng điện áp mức cao để điều khiển.

GND	Nối đất
COM	Được sử dụng làm chân kiểm tra hoặc chân triệt điện áp.
Output 1 - 7	Đầu ra tương ứng của bảy chân đầu vào. Mỗi chân đầu ra sẽ chỉ được nối đất khi chân đầu vào tương ứng của nó ở mức cao (+ 5V).

Bảng 2: Bảng mô tả chân của ULN2003.

Thông số kỹ thuật:

- Chứa 7 cặp transistor Darlington có khả năng xử lý điện áp cao và dòng điện cao, với mỗi cặp được đánh giá ở mức 50V và 500mA.
- Các chân đầu vào có thể được kích hoạt với điện áp +5V.
- Tất cả bảy chân đầu ra có thể được kết hợp để điều khiển tải lên đến khoảng 3.5A (7 × 500mA).

2.6. BUZZER.

Hình 8: Buzzer.

Buzzer là một thiết bị báo hiệu âm thanh có thể tạo ra các âm thanh khác nhau như báo thức, âm nhạc, chuông và còi báo động.

Thông số kỹ thuật:

- Điện áp hoạt động: 3.5V - 5.5V.
- Dòng điện tiêu thụ: <25mA.
- Tần số cộng hưởng: 2300Hz ± 500Hz.
- Biên độ âm thanh: >80 dB.
- Nhiệt độ hoạt động:-20 °C đến +70 °C.
- Kích thước : Đường kính 12mm, cao 9,7mm.

2.7. QUANG TRỞ.

Hình 9: Quang trở.

Quang trở, còn gọi là điện trở quang, photoresistor, hay photocell, là một loại linh kiện được làm từ vật liệu đặc biệt có khả năng thay đổi điện trở khi tiếp xúc với ánh sáng. Nói một cách đơn giản, nó hoạt động như một tế bào quang điện dựa trên nguyên lý quang dẫn. Điều này có nghĩa là giá trị điện trở của quang trở sẽ biến đổi theo cường độ ánh sáng mà nó nhận được.

Nguyên lý hoạt động:

Quang trở được chế tạo từ chất bán dẫn có trở kháng rất cao và không có tiếp giáp. Trong điều kiện bóng tối, điện trở của quang trở có thể lên đến vài MΩ. Và khi có ánh sáng chiếu vào thì giá trị điện trở của nó giảm xuống, thường từ một đến vài trăm Ω.

Quang trở hoạt động dựa trên nguyên lý hiệu ứng quang điện trong một khối vật liệu. Khi các photon với năng lượng đủ lớn va chạm vào sẽ làm các electron bật ra khỏi các phân tử, trở thành các electron tự do và biến chất bán dẫn thành dẫn điện. Mức độ dẫn điện của quang trở phụ thuộc chủ yếu vào số lượng photon được hấp thụ.

Khi ánh sáng chiếu vào quang trở, các electron được giải phóng, làm tăng độ dẫn điện. Tùy thuộc vào loại chất bán dẫn sử dụng, các quang trở sẽ phản ứng khác nhau với các loại sóng photon khác nhau.

CHƯƠNG 3: THIẾT KẾ HỆ THỐNG.

3.1. THIẾT KẾ PHẦN CỨNG.

3.1.1. Chức năng phần cứng.

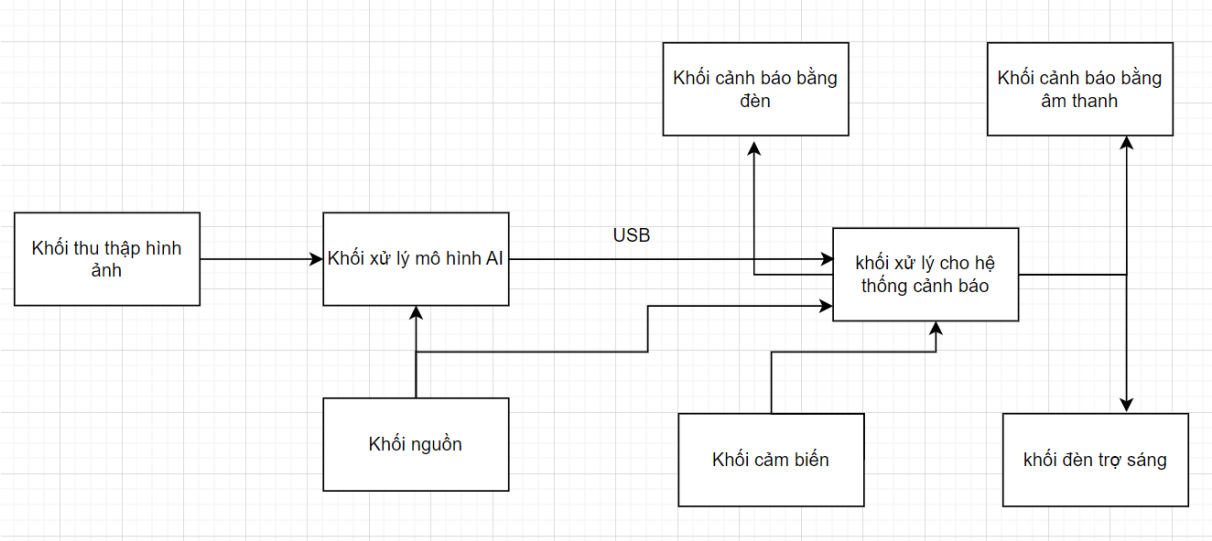
- **Đọc giá trị quang trở:** sử dụng GPIO của STM32 để đọc giá trị quang trở bằng ADC tham chiếu với 12bit có khoảng giá trị từ 0 đến 4095 và tần số lấy mẫu bằng 1.5x tần số chia cho bộ ADC (12Mhz) = 18Mhz. Khi giá trị ADC ở dưới ngưỡng thì sẽ bật đèn trợ sáng tạm thời cho ô tô, ngược lại đèn sẽ tắt.

- **Điều khiển Buzzer và LED**: Sử dụng giao tiếp GPIO để kết nối buzzer và LED với vi điều khiển STM32 thông qua IC driver ULN2003. Lập trình để kích hoạt buzzer và LED khi có phát hiện tài xế ngủ gật trên ô tô và cảnh báo các xe xung quanh.

- **Gửi SMS cảnh báo**: Sử dụng module SIM900A để gửi SMS thông qua UART của STM32. Lập trình để gửi nội dung tin nhắn bao gồm lời cảnh báo và biển số xe của tài xế đó đã được đăng kí trước.

- **Giao tiếp USB CDC:** Sử dụng raspberry pi 4 để gửi giá trị qua STM32 thông qua USB sau đó STM32 sẽ kiểm tra dữ liệu đó thuộc loại nào không ngủ, ngủ ngắn hay ngủ dài.

3.1.2. Sơ đồ khối.



Hình 10: Sơ đồ khối hệ thống.

***Giải thích từng khối.**

- **Khối nguồn:** Khối này cung cấp nguồn cho toàn bộ hệ thống cũng như 2 bộ xử lý chính của hệ thống.

- **Khối thu thập hình ảnh:** Khối này sẽ sử dụng 1 camera để thu thập dữ liệu khuôn mặt từ tài xế sau đó đưa qua cho bộ xử lý phân tích.

- **Khối cảm biến:** Khối này sử dụng 1 quang trở để thu thập giá trị cường độ ánh sáng thông qua ADC của STM32.

- **Khối đèn trợ sáng:** Khối này sử dụng 1 led để bật/tắt đèn khi giá trị của quang trở nằm ở ngưỡng dưới/trên.

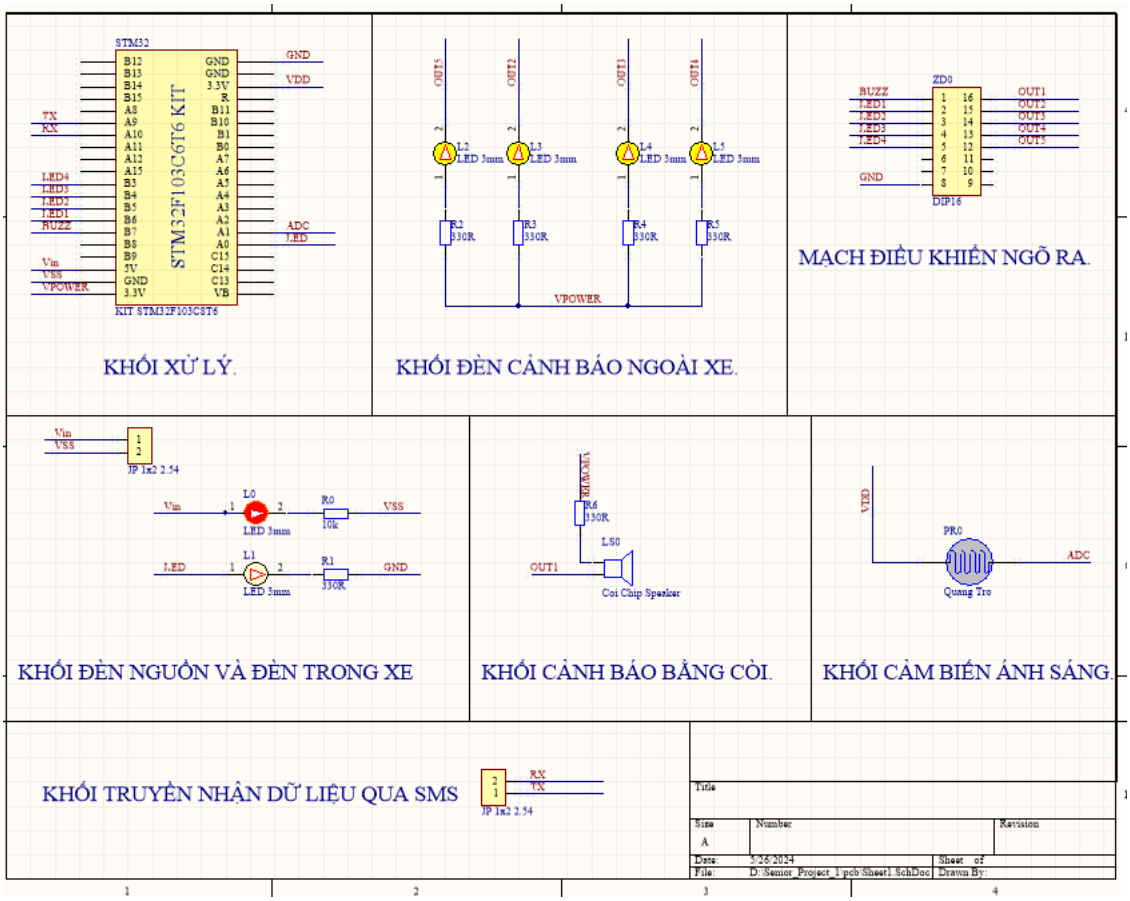
- **Khối cảnh báo bằng âm thanh:** Khối này sử dụng 1 buzzer để làm tín hiệu âm thanh khi tài xế nhắm mắt.

- **Khối cảnh báo bằng đèn:** Khối này sử dụng 4 led tương đương với 4 đèn tín hiệu ngoài xe để cảnh báo cho các xe khác khi tài xế bên trong xe đang ngủ gật.

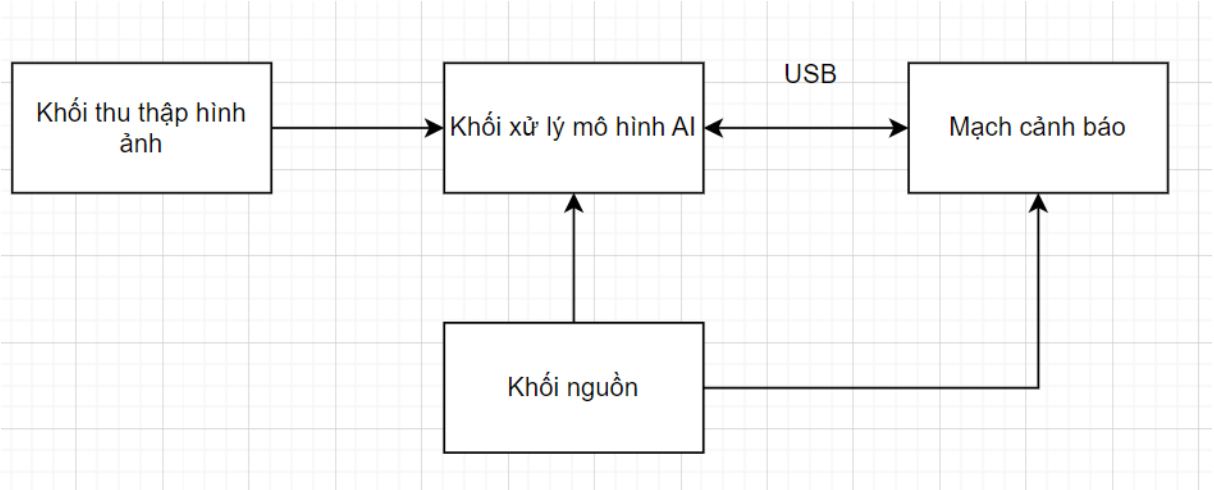
- **Khối xử lý mô hình AI:** khối này sẽ dùng 1 raspberry pi 4 để chạy mô hình AI này và gửi dữ liệu cho STM32 khi có ngủ gật xảy ra.

- **Khối xử lý cho hệ thống cảnh báo:** khối này sẽ xử lý các dữ liệu được nhận từ raspberry để phân tích và xử lý. Sau đó sẽ đưa ra cảnh báo hay không tùy thuộc vào dữ liệu nhận về từ raspberry pi.

3.1.3. Sơ đồ nguyên lý và kết nối.



Hình 11: Sơ đồ nguyên lý của mạch cảnh báo.



Hình 12: Sơ đồ khối kết nối của hệ thống.

3.2. THIẾT KẾ PHẦN MỀM.

3.2.1. Chức năng hoạt động của phần mềm.

Về phần mềm sẽ có 2 phần, phần đầu tiên sẽ xử lý ảnh mà camera thu thập được, và phần thứ 2 sẽ là chương trình xử lý khi có cảnh báo xảy ra:

Ở phần đầu tiên, camera sẽ thu thập những hình mà nó phát hiện được mặt người còn lại nó sẽ bỏ qua những tấm ảnh như là không phát hiện được mặt người, mặt người bị quá to,…Sau khi thu thập hình ảnh đạt yêu cầu như trên thì sẽ qua giai đoạn xử lý bằng cách resize lại hình ảnh thành 20x10 và có 1 kênh màu, sau đó sẽ qua giai đoạn tiền xử lý ảnh bằng Mobilenet, sau đó nó sẽ dự đoán ảnh sau khi tiền xử lý xong là mắt nhắm hay mở, nếu mắt nhắm thì sẽ tăng số frame mắt nhắm lên 1, nếu các vòng lặp sau đó nó liên tục tăng tới giá trị ngưỡng thì nó sẽ gửi tín hiệu cảnh báo thông qua USB, ngược lại thì sẽ xóa biến đếm số frame mắt nhắm lại. Trong trường hợp số frame mắt nhắm đó quá lớn đồng nghĩa với việc là mô hình này nó phát hiện tài xế ngủ quá lâu thì sẽ gửi tín hiệu qua USB để báo cho mạch cảnh báo biết là tín hiệu này cần phải gửi tin nhắn SMS đi.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 10, 20, 1)]	0	[]
conv_1 (Conv2D)	(None, 8, 18, 64)	640	['input_1[0][0]']
conv_2 (Conv2D)	(None, 6, 16, 64)	36928	['conv_1[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 3, 8, 64)	0	['conv_2[0][0]']
conv_4 (Conv2D)	(None, 3, 8, 128)	24704	['max_pooling2d_1[0][0]']
conv_3 (Conv2D)	(None, 3, 8, 128)	8320	['max_pooling2d_1[0][0]']
conv_5 (Conv2D)	(None, 3, 8, 128)	49280	['conv_4[0][0]']
add_1 (Add)	(None, 3, 8, 128)	0	['conv_3[0][0]', 'conv_5[0][0]']
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 128)	0	['add_1[0][0]']
dense_1 (Dense)	(None, 2)	258	['global_average_pooling2d_1[0][0]']

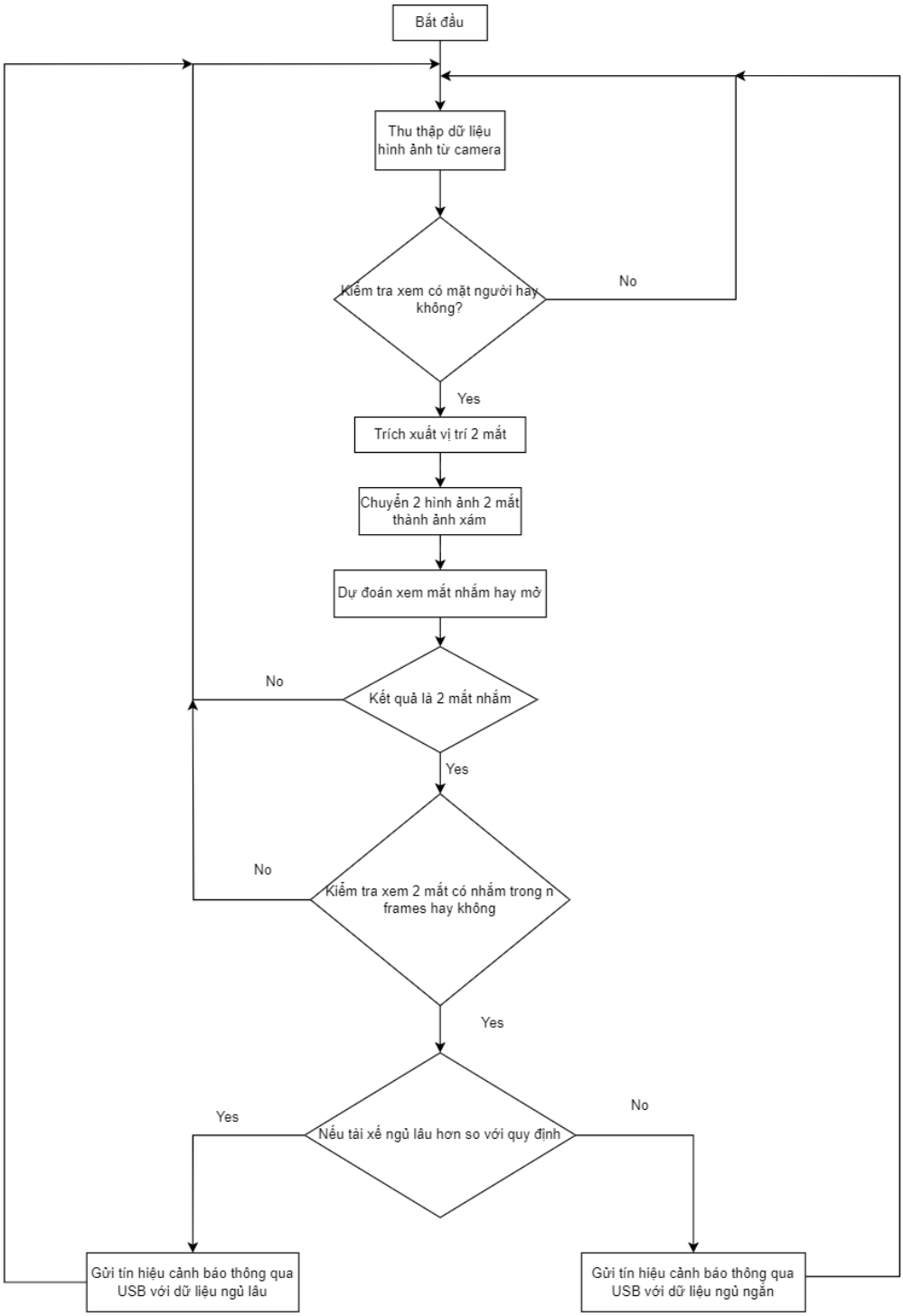
Total params: 120130 (469.26 KB)
Trainable params: 120130 (469.26 KB)
Non-trainable params: 0 (0.00 Byte)

Hình 13: Sumarry table ở của model dự đoán mắt nhắm hay mở.

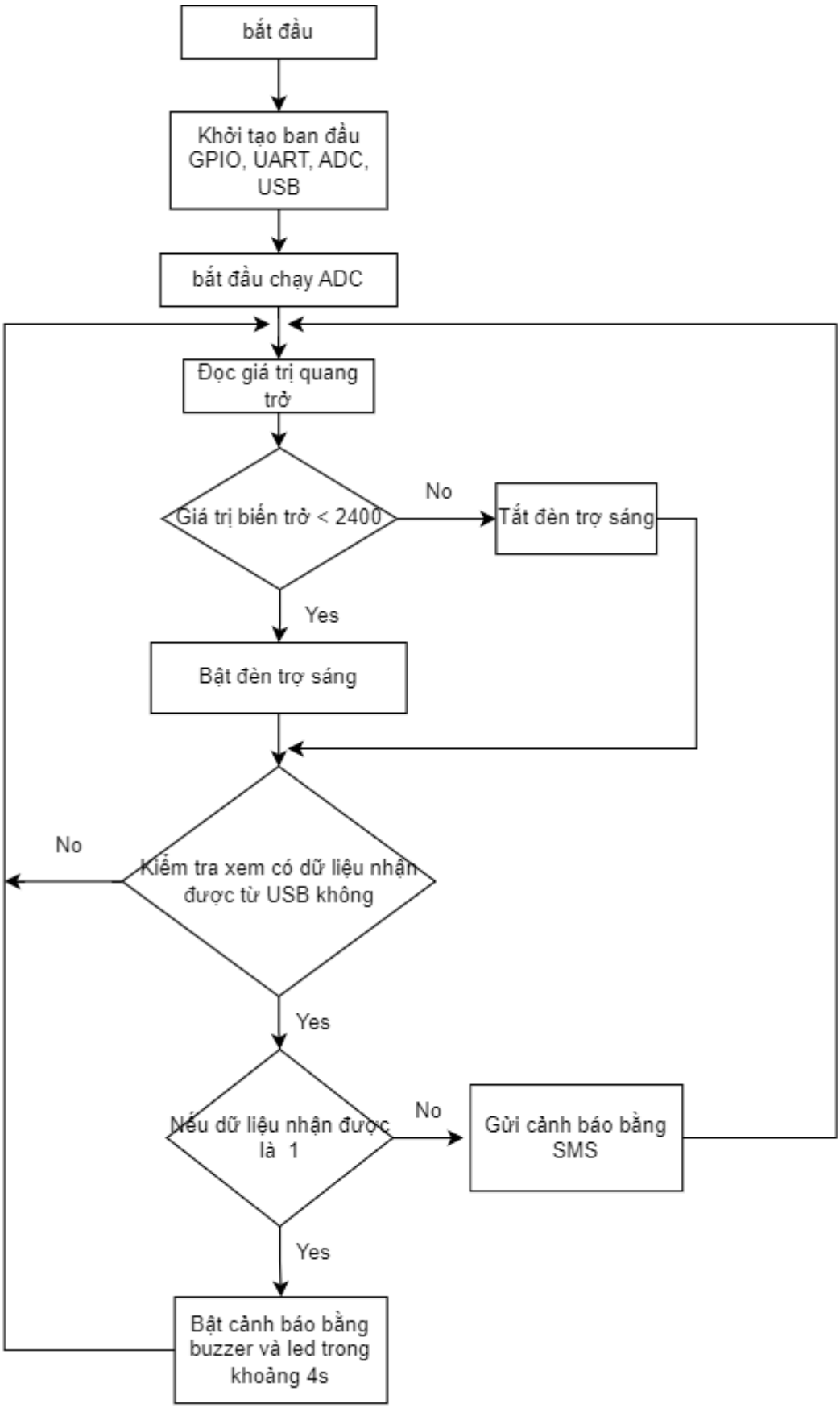
Ở phần thứ 2, sẽ là chương trình phục vụ cho mạch cảnh báo. Đầu tiên, chương trình hoạt động bằng cách khởi tạo các ngoại vi cơ bản như GPIO, UART, ADC, USB, ngoài ra còn khởi tạo các thứ khác nữa. Sau đó cho ADC bắt đầu chạy và đọc giá trị quang trở, nếu giá trị đó thấp hơn ngưỡng thì cho bật đèn trợ sáng lên, ngược lại thì tắt đèn. Sau đó, tiếp tục kiểm tra xem mạch

có nhận được tín hiệu từ USB hay không, nếu có thì kiểm tra thêm xem tín hiệu nhận được là ngủ ngắn hay dài, nếu ngắn thì bật đèn cảnh báo và âm thanh cảnh báo tài xế, nếu dài thì gửi cảnh báo bằng tin nhắn SMS. Cuối cùng sẽ quay lại kiểm tra giá trị quang trở tiếp tục như những gì đã liệt kê ra.

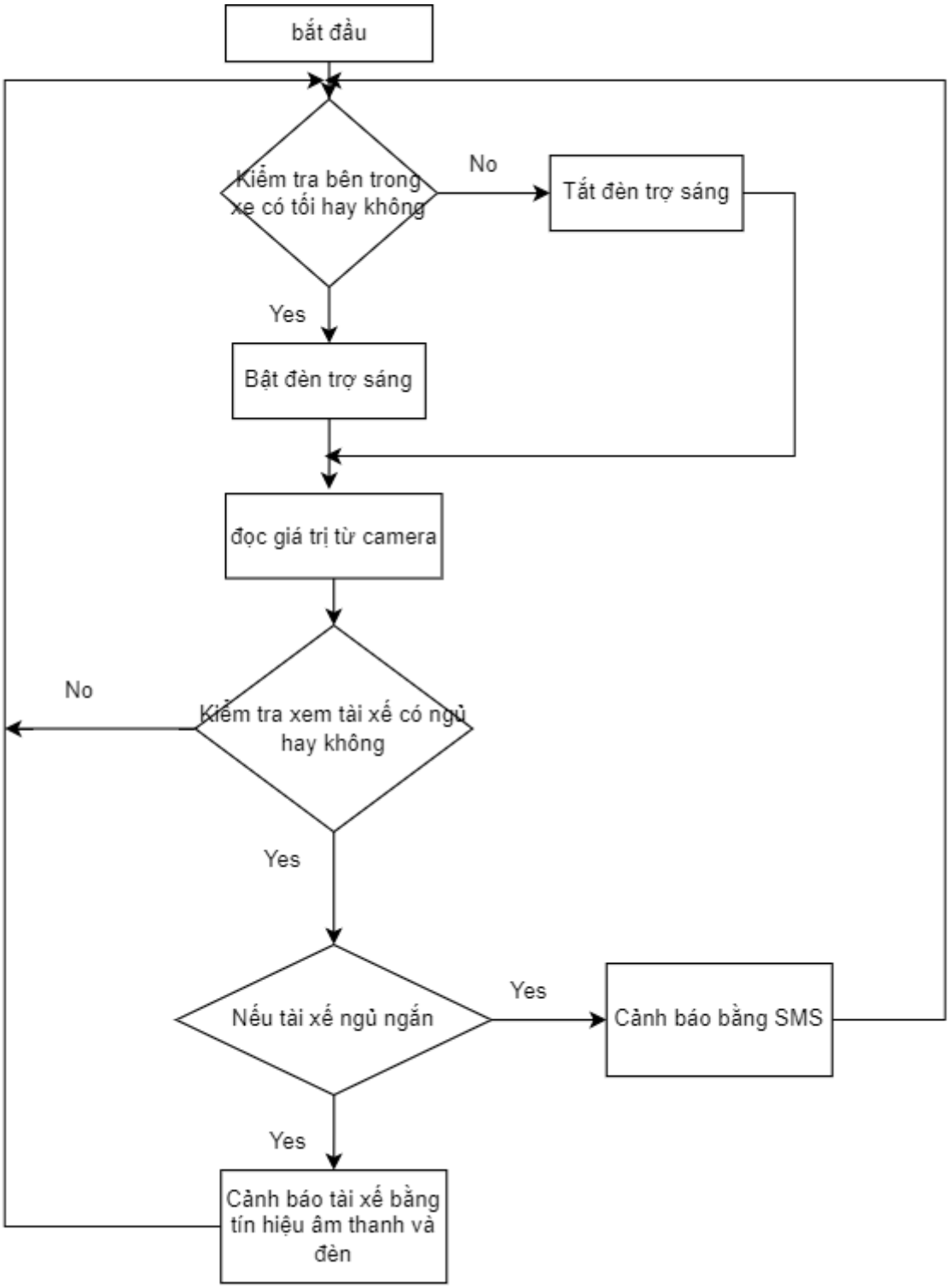
3.2.2. Lưu đồ hoạt động.



Hình 14:Lưu đồ hoạt động của mô hình AI.

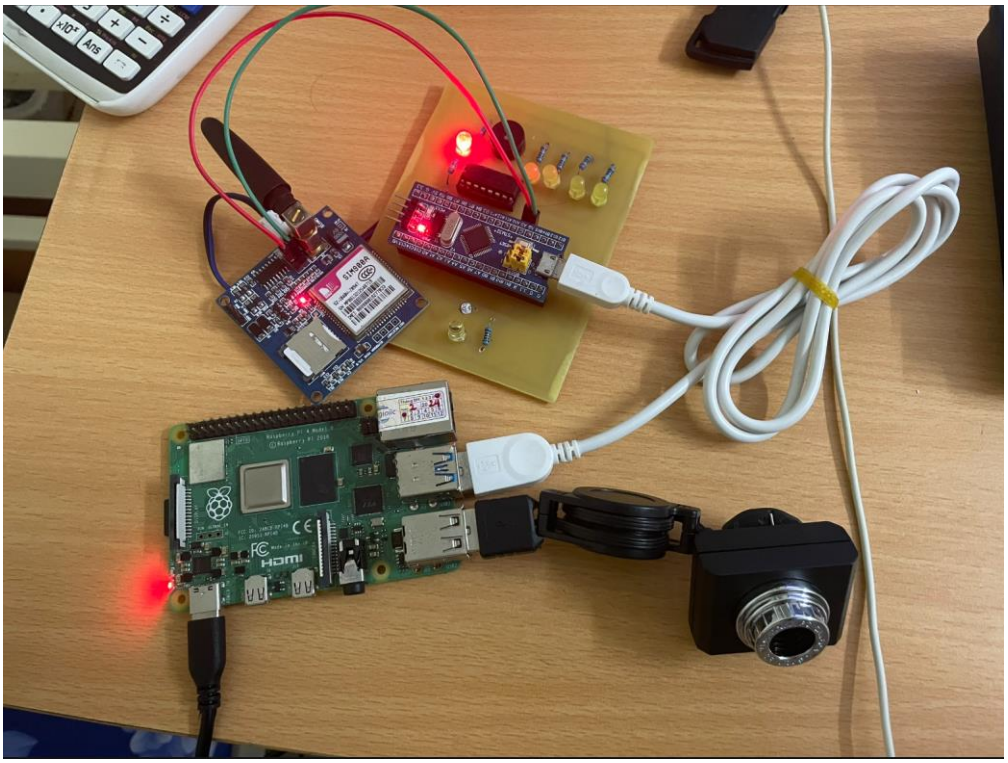


Hình 15: Lưu đồ hoạt động của mạch cảnh báo.

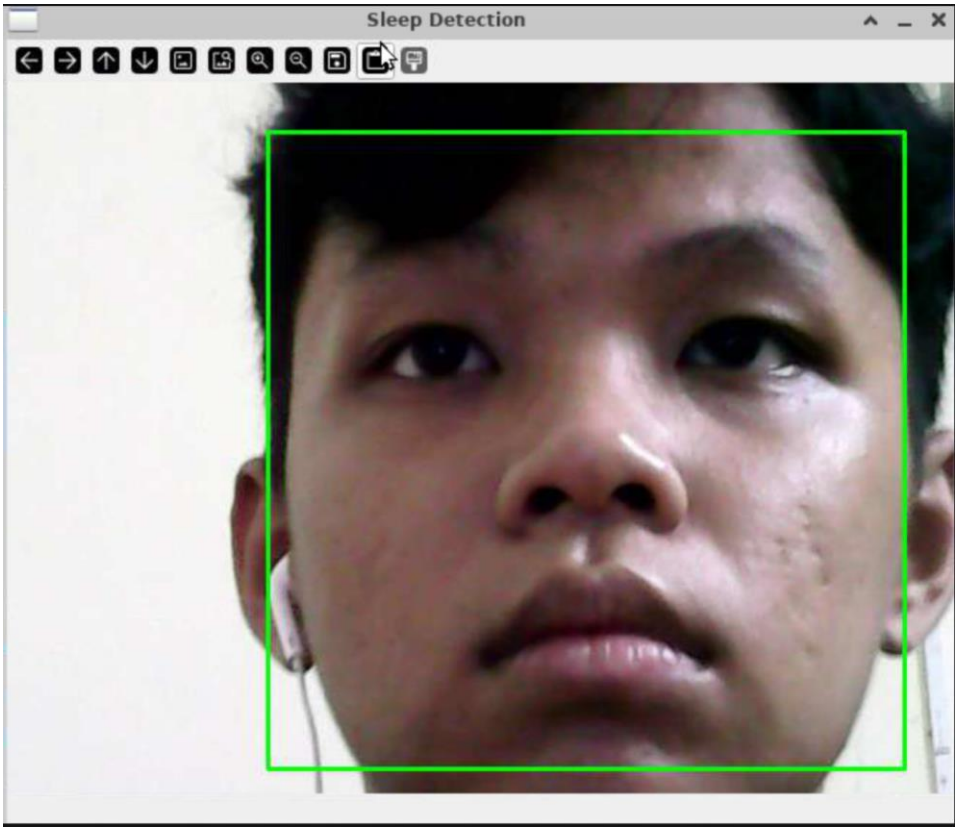


Hình 16:Lưu đồ hoạt động của toàn hệ thống.

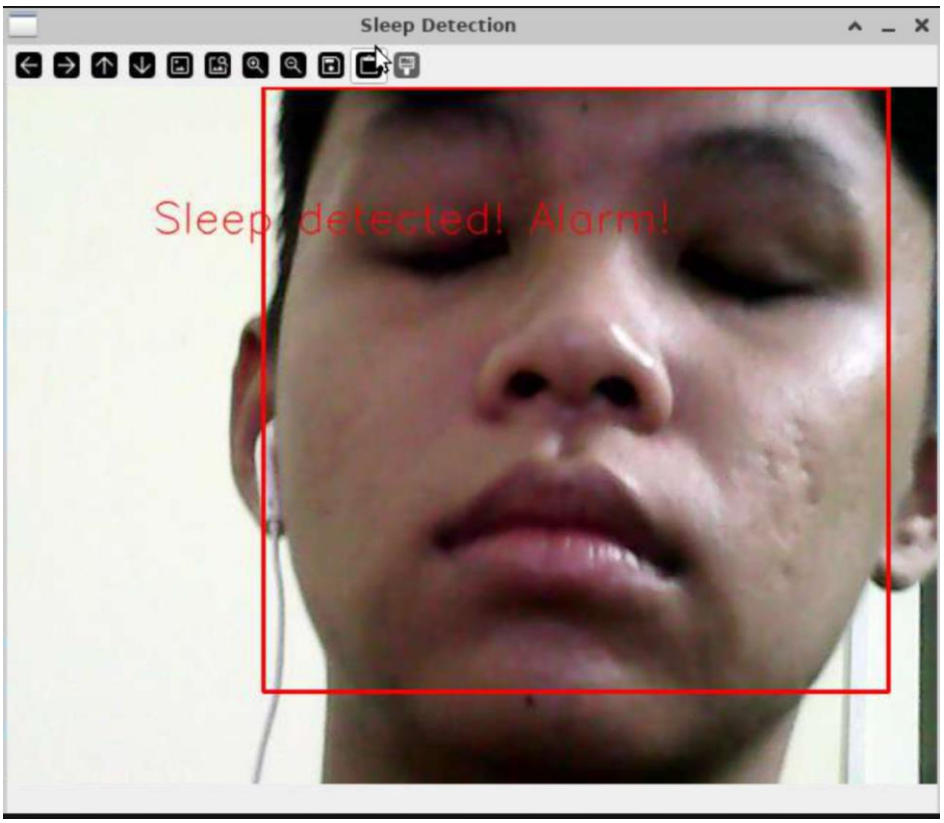
CHƯƠNG 4: KẾT QUẢ.



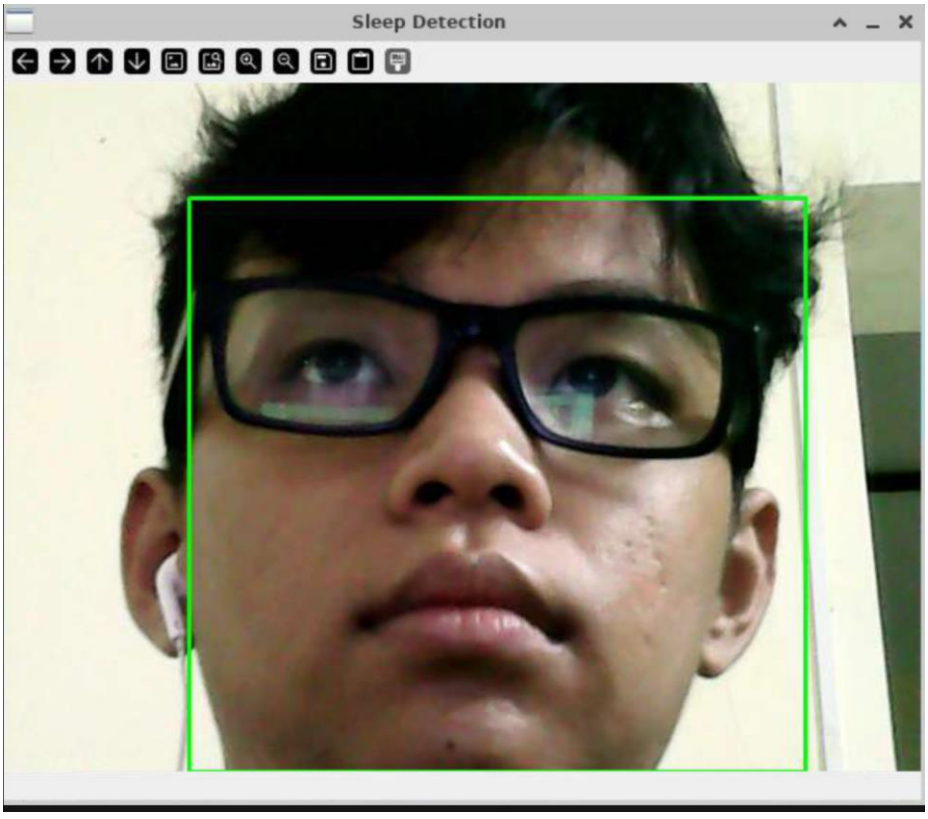
Hình 17: Triển khai hệ thống.



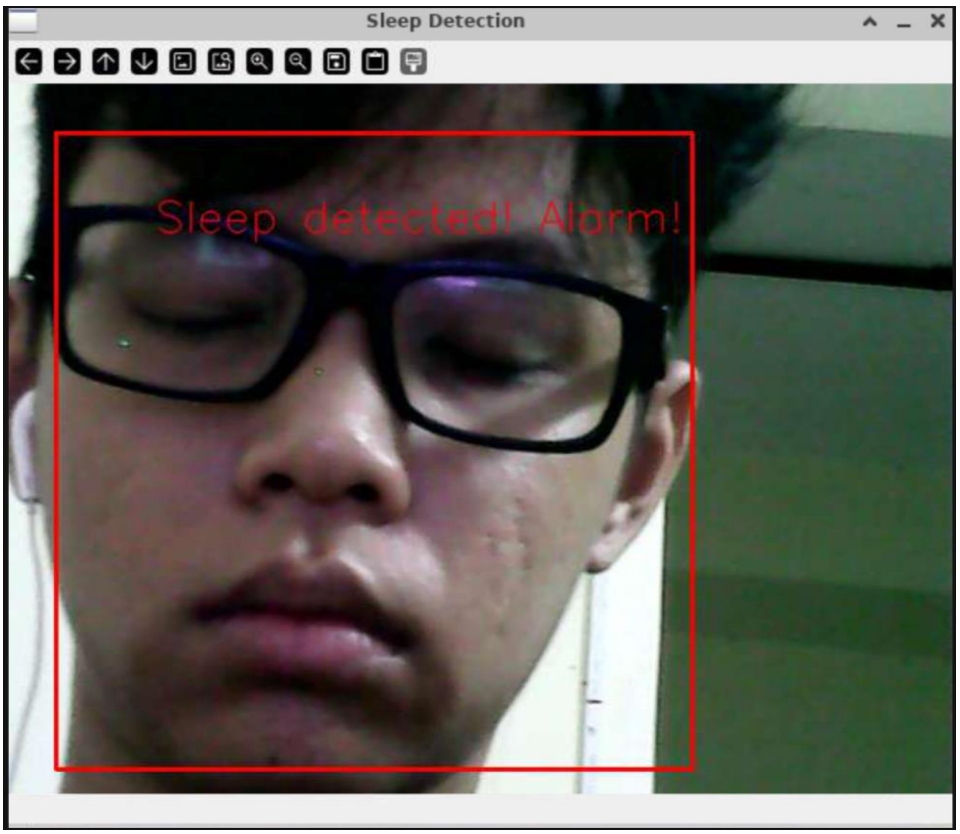
Hình 18:Khuôn mặt tài xế đang tỉnh táo không có vật cản.



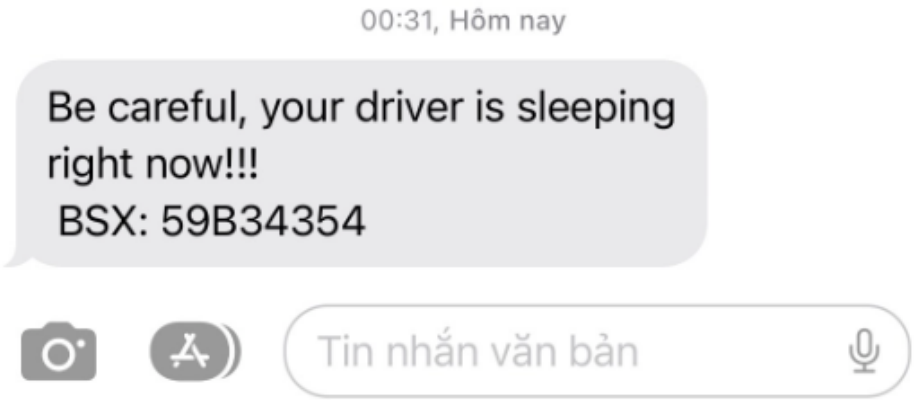
Hình 19: Phát hiện tài xế ngủ gật khi không có vật cản.



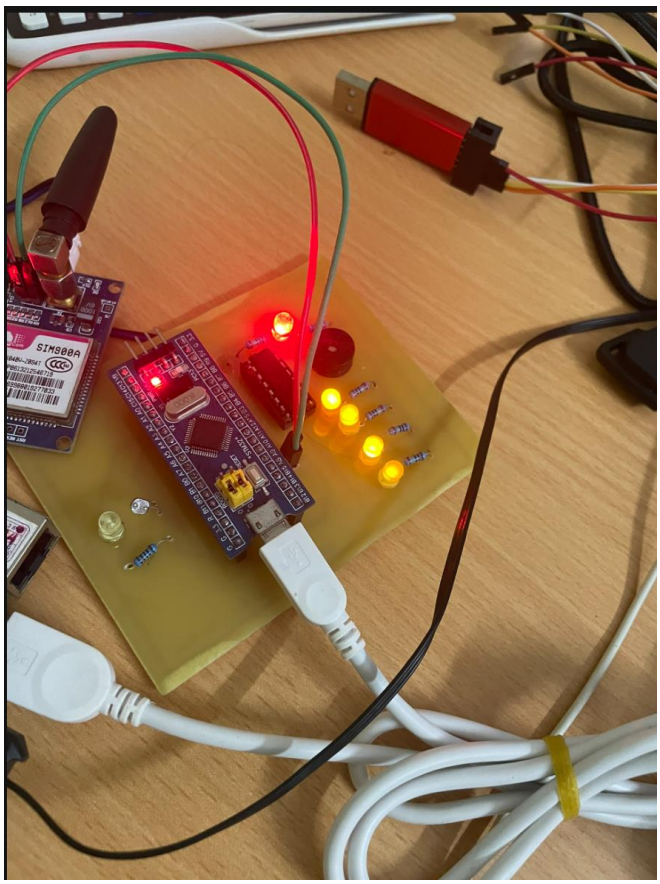
Hình 20: Khuôn mặt tài xế đang tỉnh táo có đeo thêm kính.



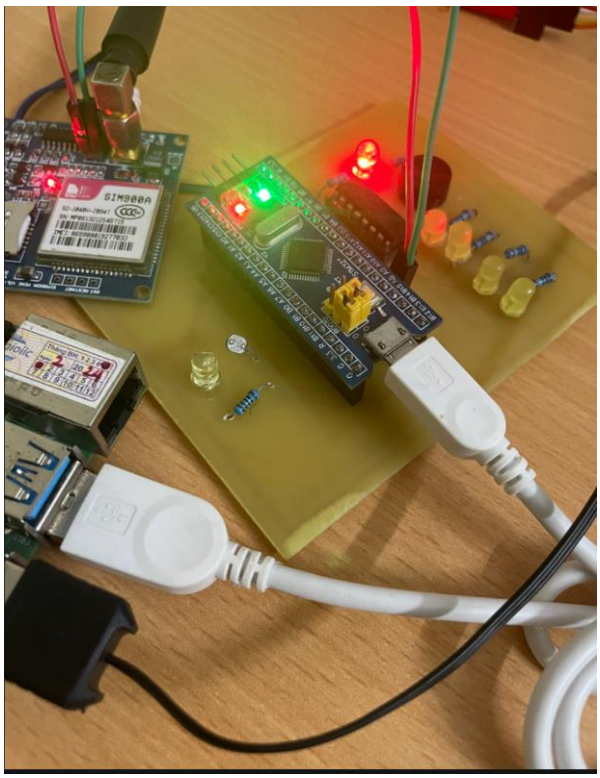
Hình 21:Phát hiện tài xế đang ngủ gật khi có đeo thêm kính.



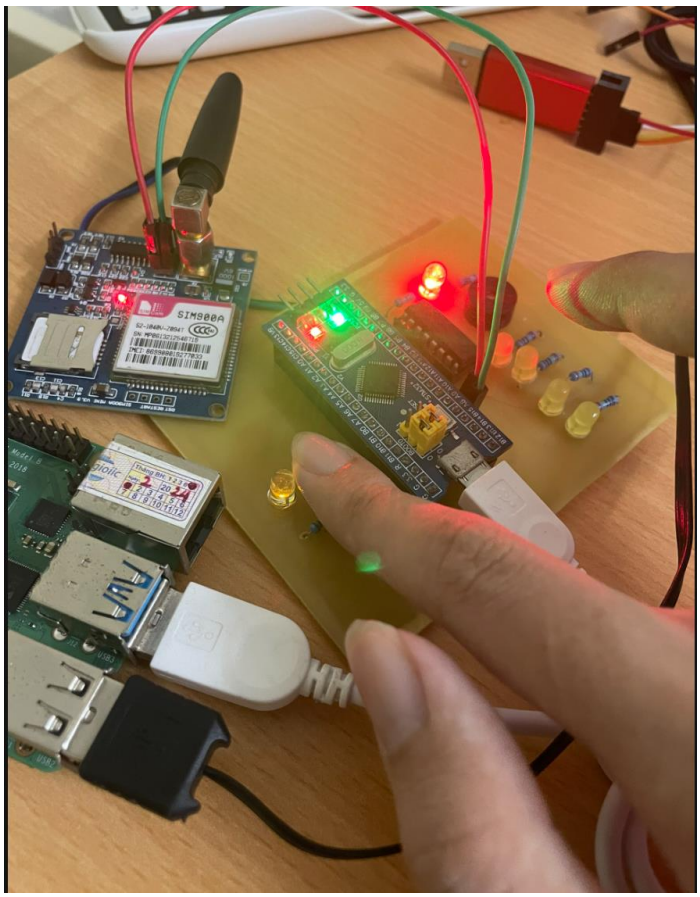
Hình 22:Tin nhắn SMS được gửi tới điện thoại khi phát hiện tài xế ngủ gật quá lâu.



Hình 23:Bật đèn cảnh báo và buzzer khi phát hiện tài xế ngủ gật.



Hình 24:Đèn khi không báo động.



Hình 25: Kiểm tra đèn trợ sáng khi trong môi trường tối.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.

5.1. KẾT LUẬN

Sau khi kiểm thử hệ thống này thì rút ra kết luận hệ thống này có chính xác chưa tốt do các môi trường làm cho ảnh nhiễu chẳng hạn như là đeo kính bị ánh sáng chá, hoặc là đeo kính trong môi trường tối, ngủ ở tư thế mà camera không thể bắt được khuôn mặt của tài xế. Tuy nhiên, nếu như ở một môi trường thuận lợi thì hệ thống này làm việc khá tốt khi cho ra độ chính xác tương đối tốt.

5.2. HƯỚNG PHÁT TRIỂN

Trong tương lai, nếu may mắn hệ thống này có khả năng được sử dụng thì cần cải tiến đặc biệt hơn về phần cứng, do trong quá trình sử dụng thì Raspberry Pi 4 cho ra một tốc độ phản hồi tương đối chậm và camera này thuộc hàng không tốt cho nên việc phát hiện tài xế ngủ gật sẽ khó khăn nhất ở điểm này. Vì thế trong tương lai, hướng phát triển của hệ thống này sẽ bao gồm những thay đổi và cải tiến như sau:

- Thay Raspberry Pi bằng 1 bộ xử lý có GPU mạnh mẽ hơn như CUDA GPU của Jetson Nano,...
- Nâng cấp camera thành 1 camera có tốc độ khung hình trên giây tốt hơn sử dụng USB 3.0 để tăng tốc độ đọc camera, đặc biệt camera phải có hồng ngoại để có thể quan sát được tài xế trong môi trường tối.
- Sử dụng thêm GPS để gửi tin nhắn vị trí của người tài xế ngủ gật đó.

- Thay vì chỉ đơn giản là phát hiện tài xế ngủ gật thì có thể nâng cấp model này thành phát hiện tài xế mất tập trung trong lái xe như là sử dụng điện thoại, vừa ăn vừa lái xe, không quay về hướng phía trước để lái xe,…
- Tối ưu vị trí lắp đặt camera để đảm bảo là chỉ nhìn thấy được tài xế.
- Nâng cấp thêm thành nhiều mức độ ngủ nữa như là thêm phát hiện ngáp,…

PHỤ LỤC

***Code cho mạch cảnh báo:**

```
#include "main.h"
#include "usb_device.h"
/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;
UART_HandleTypeDef huart1;
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
static void MX_USART1_UART_Init(void);
/* USER CODE BEGIN 0 */
extern char ReceivedData[100];
extern uint8_t Rxcount;
extern uint32_t dataSize;
extern uint8_t check;
extern uint32_t time;

void SendSMS(){
	char mobileNumber[] = "+84919059121";  // Enter the Mobile Number you want to send to
	char ATcommand[80];
	uint8_t buffer[30] = {0};
	uint8_t ATisOK = 0;
	while(!ATisOK){
			sprintf(ATcommand,"AT\r\n");
			HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
			HAL_UART_Receive (&huart1, buffer, 30, 100);
			HAL_Delay(100);
			if(strstr((char *)buffer,"A")){
				ATisOK = 1;
			}
			memset(buffer,0,sizeof(buffer));
	}
	sprintf(ATcommand,"AT+CMGF=1\r\n");
	HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
	HAL_UART_Receive (&huart1, buffer, 30, 100);
	HAL_Delay(200);
	memset(buffer,0,sizeof(buffer));
	sprintf(ATcommand,"AT+CMGS=\"%s\"\r\n",mobileNumber);
	HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
	HAL_Delay(200);
```

```
        sprintf(ATcommand,"Be careful, your driver is sleeping right now!!!\n
BSX: 59B34354%c",0x1a);
        HAL_UART_Transmit(&huart1,(uint8_t
*)ATcommand,strlen(ATcommand),1000);
        HAL_UART_Receive (&huart1, buffer, 30, 100);
        memset(buffer,0,sizeof(buffer));
        //HAL_Delay(4000);
      HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
}
/* USER CODE END 0 */

int main(void) {
      /* USER CODE BEGIN 1 */

      /* USER CODE END 1 */

      /* MCU Configuration--------------------------------------------------------*/

      /* Reset of all peripherals, Initializes the Flash interface and the Systick.
*/
      HAL_Init();

      /* USER CODE BEGIN Init */

      /* USER CODE END Init */

      /* Configure the system clock */
      SystemClock_Config();

      /* USER CODE BEGIN SysInit */

      /* USER CODE END SysInit */

      /* Initialize all configured peripherals */
      MX_GPIO_Init();
      MX_ADC1_Init();
      MX_USART1_UART_Init();
      MX_USB_DEVICE_Init();
      /* USER CODE BEGIN 2 */
      uint32_t Brightness = 0;
      int cnt = 0;
      int isSleepy = 0;
      HAL_GPIO_WritePin(GPIOC, State_Pin, 1);
//    HAL_GPIO_WritePin(GPIOB, LED_4_Pin, 1);
      /* USER CODE END 2 */

      /* Infinite loop */
      /* USER CODE BEGIN WHILE */
```

```
	HAL_ADC_Start(&hadc1);

	while (1) {
		/* USER CODE END WHILE */

		/* USER CODE BEGIN 3 */

		Brightness = HAL_ADC_GetValue(&hadc1);
		if (Brightness < 2400) {
			HAL_GPIO_WritePin(GPIOA, LED_Pin,
GPIO_PIN_SET);
		}

		if (cnt == 60) {
			cnt = 0;
			HAL_GPIO_WritePin(GPIOA, LED_Pin,
GPIO_PIN_RESET);
			HAL_ADC_Start(&hadc1);

		} else {
			cnt++;
			if(HAL_GPIO_ReadPin(GPIOA, LED_Pin) ==
GPIO_PIN_SET)
				HAL_ADC_Stop(&hadc1);
			else
				HAL_ADC_Start(&hadc1);
		}
		if (check) {
			if (ReceivedData[0] == '1') {
				isSleepy = 1; cnt = 0;
			}
			else if (ReceivedData[0] == '2') {
				SendSMS();
			}
			for (int i = 0; i < dataSize; i++) {
				ReceivedData[i] = 0;
			}
			check = 0;
		}
		if (isSleepy == 1) {
			if (cnt <= 20) {
				HAL_GPIO_TogglePin(GPIOB, Buzzer_Pin);
				HAL_GPIO_TogglePin(GPIOB, LED_4_Pin);
				HAL_GPIO_TogglePin(GPIOB, LED_3_Pin);
				HAL_GPIO_TogglePin(GPIOB, LED_2_Pin);
				HAL_GPIO_TogglePin(GPIOB, LED_1_Pin);
				HAL_Delay(200);
			}
```

```
                    else {
                        isSleepy = 0;
                        HAL_GPIO_WritePin(GPIOB, LED_4_Pin, 0);
                        HAL_GPIO_WritePin(GPIOB, LED_3_Pin, 0);
                        HAL_GPIO_WritePin(GPIOB, LED_2_Pin, 0);
                        HAL_GPIO_WritePin(GPIOB, LED_1_Pin, 0);
                        HAL_GPIO_WritePin(GPIOB, Buzzer_Pin, 0);
                    }
                }
            }
}
    /* USER CODE END 3 */


/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void) {
    RCC_OscInitTypeDef RCC_OscInitStruct = { 0 };
    RCC_ClkInitTypeDef RCC_ClkInitStruct = { 0 };
    RCC_PeriphCLKInitTypeDef PeriphClkInit = { 0 };

    /** Initializes the RCC Oscillators according to the specified parameters
     * in the RCC_OscInitTypeDef structure.
     */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
    RCC_OscInitStruct.HSIState = RCC_HSI_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
    RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL6;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK) {
        Error_Handler();
    }

    /** Initializes the CPU, AHB and APB buses clocks
     */
    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK |
RCC_CLOCKTYPE_SYSCLK
                    | RCC_CLOCKTYPE_PCLK1 |
RCC_CLOCKTYPE_PCLK2;
    RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
```

```
    if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_1) != HAL_OK) {
        Error_Handler();
    }
    PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_ADC |
RCC_PERIPHCLK_USB;
    PeriphClkInit.AdcClockSelection = RCC_ADCPCLK2_DIV2;
    PeriphClkInit.UsbClockSelection = RCC_USBCLKSOURCE_PLL;
    if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK) {
        Error_Handler();
    }
}

/**
 * @brief ADC1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_ADC1_Init(void) {

    /* USER CODE BEGIN ADC1_Init 0 */

    /* USER CODE END ADC1_Init 0 */

    ADC_ChannelConfTypeDef sConfig = { 0 };

    /* USER CODE BEGIN ADC1_Init 1 */

    /* USER CODE END ADC1_Init 1 */

    /** Common config
     */
    hadc1.Instance = ADC1;
    hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
    hadc1.Init.ContinuousConvMode = DISABLE;
    hadc1.Init.DiscontinuousConvMode = DISABLE;
    hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
    hadc1.Init.NbrOfConversion = 1;
    if (HAL_ADC_Init(&hadc1) != HAL_OK) {
        Error_Handler();
    }

    /** Configure Regular Channel
     */
    sConfig.Channel = ADC_CHANNEL_1;
    sConfig.Rank = ADC_REGULAR_RANK_1;
```

```
    sConfig.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
    if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK) {
        Error_Handler();
    }
    /* USER CODE BEGIN ADC1_Init 2 */

    /* USER CODE END ADC1_Init 2 */

}

/**
 * @brief USART1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_USART1_UART_Init(void) {

    /* USER CODE BEGIN USART1_Init 0 */

    /* USER CODE END USART1_Init 0 */

    /* USER CODE BEGIN USART1_Init 1 */

    /* USER CODE END USART1_Init 1 */
    huart1.Instance = USART1;
    huart1.Init.BaudRate = 9600;
    huart1.Init.WordLength = UART_WORDLENGTH_8B;
    huart1.Init.StopBits = UART_STOPBITS_1;
    huart1.Init.Parity = UART_PARITY_NONE;
    huart1.Init.Mode = UART_MODE_TX_RX;
    huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart1.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart1) != HAL_OK) {
        Error_Handler();
    }
    /* USER CODE BEGIN USART1_Init 2 */

    /* USER CODE END USART1_Init 2 */

}
/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void) {
    GPIO_InitTypeDef GPIO_InitStruct = { 0 };
    /* GPIO Ports Clock Enable */
```

```
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOD_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(State_GPIO_Port, State_Pin,
GPIO_PIN_RESET);
    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin,
GPIO_PIN_RESET);

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOB,
                      LED_4_Pin | LED_3_Pin | LED_2_Pin | LED_1_Pin |
Buzzer_Pin,
                      GPIO_PIN_RESET);

    /*Configure GPIO pin : State_Pin */
    GPIO_InitStruct.Pin = State_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(State_GPIO_Port, &GPIO_InitStruct);

    /*Configure GPIO pin : LED_Pin */
    GPIO_InitStruct.Pin = LED_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(LED_GPIO_Port, &GPIO_InitStruct);

    /*Configure GPIO pins : LED_4_Pin LED_3_Pin LED_2_Pin
LED_1_Pin
     Buzzer_Pin */
    GPIO_InitStruct.Pin = LED_4_Pin | LED_3_Pin | LED_2_Pin |
LED_1_Pin
                      | Buzzer_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */
```

```
/**
 * @brief  This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void) {
	/* USER CODE BEGIN Error_Handler_Debug */
	/* User can add his own implementation to report the HAL error return
state */
	__disable_irq();
	while (1) {
	}
	/* USER CODE END Error_Handler_Debug */
}
```

*Code cho phát hiện ngủ gật:

```
import numpy as np

import keras

from keras import backend as K

import imutils

from keras.models import load_model

import requests

from scipy.spatial import distance as dist

from imutils import face_utils

import time

import dlib

import cv2, os, sys

import collections

import random

import face_recognition

import pickle

import math

import threading

import tensorflow as tf

import serial

# Class dinh nghia vi tri 2 mat con nguoi
```

```
class FacialLandMarksPosition:
    left_eye_start_index, left_eye_end_index = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
    right_eye_start_index, right_eye_end_index = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
# Ham du doan mat dong hay mo
def predict_eye_state(model, image):
    # Resize anh ve 20x10
    image = cv2.resize(image, (20, 10))
    image = image.astype(dtype=np.float32)

    # Chuyen thanh tensor
    image_batch = np.reshape(image, (1, 10, 20, 1))
    # Dua vao mang mobilenet de xem mat dong hay mo
    image_batch = keras.applications.mobilenet.preprocess_input(image_batch)

    return np.argmax(model.predict(image_batch)[0])

# Load model dlib de phat hien cac diem tren mat nguoi - lansmark
facial_landmarks_predictor = '/bin/drowsiness/68_face_landmarks_predictor.dat'
predictor = dlib.shape_predictor(facial_landmarks_predictor)
# Load model predict xem mat nguoi dang dong hay mo
model = load_model('/bin/drowsiness/weights.149-0.01.hdf5')
# Lay anh tu Webcam
cap = cv2.VideoCapture(0)
#get port serial
ser = serial.Serial('/dev/ttyACM0', 9600)
scale = 0.3
countClose = 0
currState = 0
```

```
alarmThreshold = 4
countFrame = 0


while (True):
    c = time.time()
    # Doc anh tu webcam va chuyen thanh RGB
    ret, frame = cap.read()
    image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    # Resize anh con 50% kich thuoc goc
    original_height, original_width = image.shape[:2]
    resized_image = cv2.resize(image, (0, 0), fx=scale, fy=scale)
    # Chuyen sang he mau LAB de lay thanh lan Lightness
    lab = cv2.cvtColor(resized_image, cv2.COLOR_BGR2LAB)
    l, _, _ = cv2.split(lab)
    resized_height, resized_width = l.shape[:2]
    height_ratio, width_ratio = original_height / resized_height, original_width
/ resized_width
    # Tim kiem khuon mat bang HOG
    face_locations = face_recognition.face_locations(l, model='hog')
    # Neu tim thay it nhat 1 khuon mat
    if len(face_locations):
        # Lay vi tri khuon mat
        top, right, bottom, left = face_locations[0]
        x1, y1, x2, y2 = left, top, right, bottom
        x1 = int(x1 * width_ratio)
        y1 = int(y1 * height_ratio)
        x2 = int(x2 * width_ratio)
        y2 = int(y2 * height_ratio)
        # Trich xuat vi tri 2 mat
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
        shape = predictor(gray, dlib.rectangle(x1, y1, x2, y2))
        face_landmarks = face_utils.shape_to_np(shape)

        left_eye_indices = face_landmarks[FacialLandMarksPosition.left_eye_start_index:
                                          FacialLandMarksPosition.left_eye_end_index]
        (x, y, w, h) = cv2.boundingRect(np.array([left_eye_indices]))
        left_eye = gray[y:y + h, x:x + w]
        right_eye_indices = face_landmarks[FacialLandMarksPosition.right_eye_start_index:
                                           FacialLandMarksPosition.right_eye_end_index]
        (x, y, w, h) = cv2.boundingRect(np.array([right_eye_indices]))
        right_eye = gray[y:y + h, x:x + w]
        left_eye_open = 'yes' if predict_eye_state(model=model, image=left_eye) else 'no'
        right_eye_open = 'yes' if predict_eye_state(model=model, image=right_eye) else 'no'
        print('left eye open: {0}  right eye open: {1}'.format(left_eye_open, right_eye_open))
        if left_eye_open == 'yes' or right_eye_open == 'yes':
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
            currState = 0
            countClose = 0
        else:
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2)
            currState = 1
            countClose +=1
    frame = cv2.flip(frame, 1)
    if countClose > alarmThreshold:
        cv2.putText(frame, "Sleep detected! Alarm!", (100, 100), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 255),
                    lineType=cv2.LINE_AA)
```

```
        if countFrame == 12:
            #for i in range(25):
            time.sleep(1.5)
            ser.write('2'.encode())
            print("Send SMS done")
            countFrame = 0
        else:
            ser.write('1'.encode())
            countFrame += 1
    cv2.imshow('Sleep Detection', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# xoa window khi thoat khoi loop
cap.release()
cv2.destroyAllWindows()
```

TÀI LIỆU THAM KHẢO

[1]. N. B. Nguyen, "*Những đối tượng có nguy cơ ngủ gật khi lái xe*" 2012. [Online]. Available: https://www.vinmec.com/vi/tin-tuc/thong-tin-suc-khoe/suc-khoe-tong-quat/nhung-doi-tuong-co-nguyco-ngu-gat-khi-lai-xe/.

[2]. LA, "*Ngủ gật, hiểm họa tiềm ẩn khi lái xe*" 2019. [Online]. Available: https://laodong.vn/xe/ngu-gat-hiem-hoa-tiem-an-khi-lai-xe-761219.ldo.

[3]. *ULN2003A Datasheet*, Texas Instruments.

[4]. *SIM900A*, SimCom.

[5]. *STM32F103x6*, STMicroelectronics.

[6]. "*Raspberry Pi 4 là gì?*", 2023. [online]. Available: https://mecsu.vn/ho-tro-ky-thuat/raspberry-pi-4-la-gi-so-do-chan-tinh-nang-va-ngoai-vi-cua.lmA

[7]. E. Magán, M. P. Sesmero, J. M. Alonso-Weber, and A. Sanchis, "*Driver Drowsiness Detection by Applying Deep Learning Techniques to Sequences of Images*," Applied Sciences, vol. 12, no. 3, pp. 1– 25, 2022.

[8]. Nguyen Vu Hai, Vu Quang Huy, Ha Van Ninh, Tran Quang Quy, Ngo Huy Huy, "*Drowsiness detection system applied on car",* TNU – University of Information and Communication Technology, 2023.