**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA ĐIỆN TỬ - VIỄN THÔNG**

**ĐỒ ÁN CHUYÊN NGÀNH**

**ĐỀ TÀI: HỆ THỐNG GIÁM SÁT VÀ BÁO CÁO TAI NẠN PHƯƠNG TIỆN**



**GVHD: TS. NGUYỄN DUY NHẬT VIỄN**

**SVTH: HOÀNG TIẾN ĐẠT**

**LỚP : 17DT2**

**MSSV: 106170079**

**Đà Nẵng, 2021**

ĐẠI HỌC ĐÀ NẴNG **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA** Độc lập - Tự do - Hạnh phúc

KHOA ĐIỆN TỬ - VIỄN THÔNG

**NHIỆM VỤ ĐỒ ÁN CHUYÊN NGÀNH**

Họ tên sinh viên: Hoàng Tiến Đạt. Số thẻ sinh viên: 106170079

Lớp: 17DT2. Khoa: Điện Tử - Viễn Thông. Ngành: Kỹ Thuật Điện Tử

1. *Tên đề tài đồ án:*

Hệ thống giám sat và báo cáo tai nạn phương tiện

1. *Họ tên người hướng dẫn:* Nguyễn Duy Nhật Viễn

Đà Nẵng, ngày 24 tháng 8 năm 2021

**Trường Bộ môn: ……………….. Người hướng dẫn**

# TÓM TẮT

Mục đích của đề tài là tạo ra một thiết bị hộp đen cho xe máy cũng như một số phương tiện khác. Giúp người dùng có thể dễ dàng giám sát xe hay tài sản của mình cũng như báo cáo cho người dùng về tình trạng của xe khi gặp sự cố.

Sử dụng hệ thống mạng di động toàn cầu GSM, ở đây dùng module sim900a và hệ thống định vị toàn cầu GPS, dùng module gps\_neo7, đồng thời kết hợp với cảm biến gia tốc gy-61, cùng board STM32F411VE để xác định chính xác vị trí của xe khi đang gặp sự cố. Hệ thống cảnh báo cho mọi người xung quanh biết tình trạng của xe khi gặp sự cố như còi hú, để kịp thời giúp đỡ. Đồng thời cũng giúp người dùng cảnh báo tình trạng của xe thông qua cảm biến bằng tin nhắn và gọi điện nhờ hệ thống mạng di động toàn cầu GSM.

Điểm nổi bật của đề tài là hệ thống dễ dàng sử dụng cho người dùng. Ưu điểm là khi khởi đông hệ thống thì hệ thống hoạt động tự động. Sản phẩm có giá thành vừa phải, có tính ứng dụng cao không chỉ cho xe máy mà còn phát triển lên cho tất cả các phương tiện vận tải khác.

Báo cáo đồ án được tổ chức như sau:

**Chương 1:** Tổng quan đề tài

**Chương 2:** Cơ sở lý thuyết

**Chương 3:** Thiết kế phần cứng

**Chương 4:** Thiết kế phần mềm

**Chương 5:** Kết quả, nhận xét, đánh giá

# LỜI CAM ĐOAN

Đồ án này do tôi tự nghiên cứu và thực hiện dựa vào một số tài liệu trước đó, một số tài liệu có trong sách và trên mạng. Các kết quả công bố trong Đồ án Chuyên ngành “Hệ thống giảm sát và báo cáo tai nạn phương tiện” này là trung thực, không sao chép từ một công trình nào khác.

# LỜI CẢM ƠN

“Báo cáo Đồ Án Chuyên Ngành” được viết và thực hiện dưới sự hướng dẫn của TS. Nguyễn Duy Nhật Viễn, Khoa Điện Tử Viễn Thông, Đại học Bách Khoa Đà Nẵng.

Tôi xin gửi lời cảm ơn sâu sắc tới người dẫn tôi, TS. Nguyễn Duy Nhật Viễn, đã tận tình hướng dẫn và động viên trong suốt thời gian tôi thực hiện Đồ Án Chuyên Ngành này.

Cuối cùng, tôi xin gửi lời cảm ơn đến toàn thể cán bộ Khoa Điện Tử Viễn Thông đã nhiệt tình giúp đỡ tôi trong thời gian thực tập tại đây

Đà Nẵng, ngày 24 tháng 8 năm 2021

**Hoàng Tiến Đạt**

**MỤC LỤC**

[TÓM TẮT 3](#_Toc80780703)

[LỜI CAM ĐOAN 5](#_Toc80780704)

[LỜI CẢM ƠN 6](#_Toc80780705)

[DANH MỤC HÌNH ẢNH 9](#_Toc80780706)

[DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TĂT 11](#_Toc80780707)

[DANH SÁCH BẢNG BIỂU 12](#_Toc80780708)

[CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI 13](#_Toc80780709)

[1.1 Đặt vấn đề: 13](#_Toc80780710)

[1.2 Mục đích đề tài: 13](#_Toc80780713)

[1.3 Nội dung nghiên cứu: 14](#_Toc80780714)

[1.4 Giới hạn đề tài: 14](#_Toc80780715)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT: 15](#_Toc80780716)

[2.1 Tìm hiểu về hệ thống GPS: 15](#_Toc80780717)

[2.2 Tìm hiểu về hệ thống mạng GSM: 15](#_Toc80780718)

[2.2.1 Mạng thông tin di động toàn cầu (GSM): 15](#_Toc80780719)

[2.2.2 Một số tập lệnh AT cơ bản sử dụng cho ứng dụng GSM: 16](#_Toc80780720)

[2.3 Tìm hiểu về chuẩn giao tiếp UART: 19](#_Toc80780721)

[2.3.1 Khái quát về UART: 19](#_Toc80780722)

[2.3.2 Các hoạt động của UART: 20](#_Toc80780723)

[2.4 Tìm hiểu về đọc ADC: 21](#_Toc80780724)

[2.4.1 Khái quát về ADC: 21](#_Toc80780725)

[2.4.2 Các chức năng chính của ADC trong STM32: 21](#_Toc80780726)

[2.4.3 ACD một kênh chế độ Single và Continuous: 22](#_Toc80780727)

[2.4.4 ADC nhiều kênh sử dụng chế độ Scan Mode: 22](#_Toc80780728)

[2.5 Giới thiệu phần cứng: 23](#_Toc80780729)

[2.5.1 STM32F411 Discovery: 23](#_Toc80780730)

[2.5.2 Module sim900a: 25](#_Toc80780731)

[2.5.3 Module GPS\_NEO6M: 26](#_Toc80780732)

[2.5.4 Cảm biến gia tốc GY-61: 27](#_Toc80780733)

[CHƯƠNG 3: THIẾT KẾ PHẦN CỨNG 28](#_Toc80780734)

[3.1 Sơ đồ khối: 28](#_Toc80780735)

[3.2 Tính toán và thiết kế: 29](#_Toc80780736)

[3.2.1 Khối nguồn: 29](#_Toc80780737)

[3.2.2 Khối vi xử lý: 30](#_Toc80780738)

[3.2.3 Khối Module sim900A: 31](#_Toc80780739)

[3.2.4 Khối Module GPS\_NEO6M: 32](#_Toc80780740)

[3.2.5 Khối cảm biến gia tốc: 33](#_Toc80780741)

[3.2.6 Khối điều khiển cảnh báo: 33](#_Toc80780742)

[3.3 Mạch nguyên lý: 35](#_Toc80780743)

[3.4 Mạch in 3D: 36](#_Toc80780744)

[3.5 Mạch thi công: 36](#_Toc80780745)

[CHƯƠNG 4: THIẾT KẾ PHẦN MỀM: 37](#_Toc80780746)

[4.1 Lưu đồ: 37](#_Toc80780747)

[4.1.1 Lưu đồ báo ngã phương tiện: 37](#_Toc80780748)

[4.1.2 Lưu đồ giám sát phương tiện 38](#_Toc80780749)

[4.2 Phần mềm lập trình cho vi điều khiển: 39](#_Toc80780750)

[4.2.1 Giới thiệu phần mềm STM32 CubeIDE: 39](#_Toc80780751)

[4.2.2 Cấu hình: 39](#_Toc80780752)

[CHƯƠNG 5: KẾT QUẢ, NHẬN XÉT, ĐÁNH GIÁ: 42](#_Toc80780753)

[5.1 Kết quả: 42](#_Toc80780754)

[5.2 Nhận xét – đánh giá: 48](#_Toc80780755)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN: 49](#_Toc80780756)

[TÀI LIỆU THAM KHẢO: 50](#_Toc80780759)

[PHỤ LỤC: 51](#_Toc80780760)

# DANH MỤC HÌNH ẢNH

[Hình 2. 1: Kết nối UART](#_Toc80772034)

[Hình 2. 2: Khung truyền của giao tiếp UART](#_Toc80772035)

[Hình 2. 3: Định dạng gói tin](#_Toc80772036)

[Hình 2. 4: Độ phân giải ADC](#_Toc80772037)

[Hình 2. 5: Sơ đồ tần số ADC lấy từ bộ ABP2](#_Toc80772038)

[Hình 2. 6: ADC chế độ Scan Mode](#_Toc80772039)

[Hình 2. 7: Module STM32F411RE Discovery](#_Toc80772040)

[Hình 2. 8: Module sim 900A](#_Toc80772041)

[Hình 2. 9: Module NEO-6M GPS](#_Toc80772042)

[Hình 2. 10: Module gia tốc GY-61 DXL335](#_Toc80772043)

[Hình 3. 1: Sơ đồ khối hệ thống](#_Toc80772053)

[Hình 3. 2: Adapter 12V - 2A](#_Toc80772054)

[Hình 3. 3: Sơ đồ nguyên lý khối nguồn](#_Toc80772055)

[Hình 3. 4: Sơ đồ nguyên lý khối vi điều khiển](#_Toc80772056)

[Hình 3. 5: Sơ đồ nguyên lý khối Sim900A](#_Toc80772057)

[Hình 3. 6: Sơ đồ nguyên lý khối GPS](#_Toc80772058)

[Hình 3. 7: Sơ đồ nguyên lý khối cảm biến gia tốc](#_Toc80772059)

[Hình 3. 8: Sơ đồ nguyên lý khối cảnh báo](#_Toc80772060)

[Hình 3. 9: Sơ đồ nguyên lý mạch tổng quát](#_Toc80772061)

[Hình 3. 10: Mạch Altium 3D](#_Toc80772062)

[Hình 3. 11: Mạch thực tế](#_Toc80772063)

[Hình 4. 1: Lưu đồ báo ngã phương tiện](#_Toc80772067)

[Hình 4. 2: Lưu đố giám sát phương tiện](#_Toc80772068)

[Hình 4. 3: Cửa sổ cài đặt 1 STM32CubeIDE](#_Toc80772069)

[Hình 4. 4: Cửa sổ cài đặt 2 STM32CubeIDE](#_Toc80772070)

[Hình 4. 5: Cửa sổ cấu hình 1 STM32](#_Toc80772071)

[Hình 4. 6: Cửa sổ cấu hình STM32](#_Toc80772072)

[Hình 5. 1: Mặt trên 1 của sản phẩm](#_Toc80772087)

[Hình 5. 2: Mặt trên 2 của sản phẩm](#_Toc80772088)

[Hình 5. 3: Mặt dưới sản phẩm](#_Toc80772089)

[Hình 5. 4: Kết quả trả về 1](#_Toc80772090)

[Hình 5. 5: Kết quả trả về 2](#_Toc80772091)

[Hình 5. 6: Kết quả trả về 3](#_Toc80772092)

# 

# DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TĂT

\* Kí hiệu:

|  |  |  |
| --- | --- | --- |
| STT | Từ viết tắt | Nghĩa |
| 1 | Vref | Điện áp tham chiếu |
| 2 | mv/g | Kí hiệu độ nhạy cảm biến gia tốc |
| 3 | Vcc | điện thế + cho cực C của BJT |
| 4 | Xout | Giá trị trục X của cảm biến gia tốc |
| 5 | Yout | Giá trị trục Y của cảm biến gia tốc |
| 6 | Zout | Giá trị trục Z của cảm biến gia tốc |
| 7 | IRL | Dòng điện vào relay |
| 8 | VoutPC817 | Điện áp ngõ ra của Opto PC817 |
| 9 | VinPC817 | Điện áp ngõ vào của Opto PC817 |
| 10 | Igh | Dòng điện ngưỡng |
| 11 | Vbe | Điện áp Vb – Ve của transitor |
| 12 | Ib | Dòng điện vào cực B của transitor |
| 13 |  | Hệ số khuếch đại theo đó dòng điện ở cực gốc chuyển thành dòng điện khuếch đại |

\* Chữ viết tắt:

|  |  |  |
| --- | --- | --- |
| STT | Từ viết tắt | Nghĩa |
| 1 | GPS | Global Positioning System |
| 2 | GSM | Global System for Mobile Communications |
| 3 | TDMA | Time Division Multiple Access |
| 4 | CDMA | *Code Division Multiple Access* |
| 5 | AT | Attention |
| 6 | MT | Mobile Terminal |
| 7 | TE | Terminal Equipment |
| 8 | GPRS | General Packet Radio Service |
| 9 | UART | Universal Asynchronous Receive/Transmit |
| 10 | ADC | Analog to Digital Convert |
| 11 | EOC | Emergency Operations Center |
| 12 | EOCR | Electronic over current relay |
| 13 | DR | Digital Research |
| 14 | SRAM | static random access memory |
| 15 | RAM | Random Access Memory |
| 16 | ARM | Advanced RISC Machine |

# DANH SÁCH BẢNG BIỂU

[Bảng 1. 1: Các tập lệnh AT chung](#_Toc80772117)

[Bảng 1. 2: Các tập lệnh AT về cuộc gọi](#_Toc80772118)

[Bảng 1. 3: Các tập lệnh AT về gửi tin nhắn SMS](#_Toc80772119)

[Bảng 1. 4: Các tập lệnh AT về GPRS](#_Toc80772120)

[Bảng 5. 1: Bảng đánh giá sản phẩm](#_Toc80772133)

# CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

## 1.1 Đặt vấn đề:

Với sự phát triển mạnh mẽ của khoa học và công nghệ ngày nay đi cùng với công cuộc công nghiệp hóa - hiện đại hóa đất nước đã thúc đẩy nền công nghiệp sản xuất nước nhà phát triển. Các phương tiện giao thông như xe máy,ô tô cũng được chế tạo ra không những ngày càng tinh tế mang tính hiện đại mà còn thông minh tăng tính tự động hóa nhằm nâng cao chất lượng sống của chúng ta.

Với một đất nước mà xe máy là phương tiện di chuyển chính và còn là tài sản quý giá của mỗi con người, mỗi gia đình thì ngoài việc đảm bảo cho chiếc xe khỏi sự dòm ngó của những tên trộm như hệ thống chống trộm, cũng cần đòi hỏi một chiếc xe có tính thông minh, hàm lượng tự động hóa cao, đảm bảo an toàn cho người điều khiển phương tiện tránh khỏi những rủi ro đáng tiếc như va quẹt hay tai nạn và kịp thời cứu chữa khi xảy ra tai nạn bằng hệ thống thông báo tới người thân, bệnh viện nhằm đưa số lượng các vụ tai nạn, số lượng người chết giảm xuống mức tối đa có thể.

Ngày nay, công nghệ kỹ thuật đang phát triển mạnh mẽ, các phương pháp lập trình rất đa dạng và lập trình nhúng là một hướng đi của xu thế hiện nay. Trong lập trình nhúng thì ngôn ngữ lập trình với STM32 là một trong những phương pháp sử dụng nhiều nhất. STM32 là một bo mạch xử lý hay còn gọi là vi điều khiển được dùng để lập trình tương tác với các thiết bị phần cứng như cảm biến, động cơ, màn hình, bluetooth, GPS. Tuy chỉ là một bo mạch nhỏ gọn, chi phí thấp vì bo đã được thi công sẵn, tuy nhiên STM32 có thể được ứng dụng vào rất nhiều lĩnh vực khác nhau, ta có thể đo đạc nhiệt độ để điều khiển tưới tiêu trong các nhà vườn, hoặc có thể dùng để điều khiển xe cân bằng rất phổ biến hiện nay, hay có thể ứng dụng nó vào việc điều khiển cánh tay Robot. Ngoài những lợi ích to lớn trên, việc sử dụng và lập trình trên STM32 cũng rất đơn giản, môi trường phát triển đa dạng, hỗ trợ trên nhiều hệ điều hành khác nhau, có thể học 1 cách nhanh chóng. Vì những điểm mạnh ưu việt như vậy nên tôi đã quyết định chọn STM32 làm bộ xử lý trung tâm cho việc “HỆ THỐNG GIÁM SÁT VÀ BÁO TAI NẠN PHƯƠNG TIỆN”.

## 1.2 Mục đích đề tài:

Thiết kế và thi công mạch giám sát hoạt động phương tiện xe máy và cảnh báo cho người dùng.

Ứng dụng vi điều khiển STM32 để kết nối và điều khiển.

Ứng dụng các tập lệnh của Module SIM900A để gửi các thông tin giám sát và cảnh báo về Smartphone.

## 1.3 Nội dung nghiên cứu:

Quá trình nghiên cứu được thực hiện qua các bước sau:

* Cài đặt phần mềm Altium để vẽ mạch và phần mềm STM32Cube IDE để lập trình xử lý.
* Nghiên cứu về GPS và GSM.
* Nghiên cứu về STM32F411VE, Module SIM900A và GPS\_NEO6.
* Thiết kế, thi công mô hình phần cứng.
* Lập trình giao tiếp STM32 với SIM900A và GPS.
* Lập trình STM32 để giám sát thiết bị và cảnh báo cho người dùng qua Smartphone.
* Lắp ráp các khối vào mô hình.
* Chạy thử nghiệm hệ thống giám sát và cảnh báo phương tiện xe máy.
* Cân chỉnh hệ thống.

## 1.4 Giới hạn đề tài:

* Định vị vị trí xe máy trên bản đồ sai lệnh so với thực tế ở mức chấp nhận được.
* Tốc độ cập nhật vị trí và các thông số của xe máy tùy theo chất lượng sóng của nhà mạng mà Smartphone sử dụng và phụ thuộc vị trí của thiết bị.
* Thiết kế và thi công hệ thống giám sát và cảnh báo phương tiện xe máy bằng lý thuyết, mô phỏng và bản demo, chưa áp dụng được thực tế.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT:

## 2.1 Tìm hiểu về hệ thống GPS:

Hệ thống Định vị Toàn cầu (tiếng Anh: Global Positioning System - GPS) là hệ thống xác định vị trí dựa trên vị trí của các vệ tinh nhân tạo, do Bộ Quốc phòng Hoa Kỳ thiết kế, xây dựng, vận hành và quản lý. Trong cùng một thời điểm, tọa độ của một điểm trên mặt đất sẽ được xác định nếu xác định được khoảng cách từ điểm đó đến ít nhất ba vệ tinh. GPS là một hệ thống gồm 27 vệ tinh (kể cả 3 cái sơ cua) chuyển động trên các quĩ đạo chung quanh trái đất. Mỗi vệ tinh nặng khoảng 2 tấn, sử dụng năng lượng mặt trời, chuyển động cách mặt đất khoảng 19300km [2]. Mỗi vệ tinh quay quanh trái đất 2 vòng một ngày đêm. Quỹ đạo của các vệ tinh được tính toán sao cho ở bất kỳ nơi nào trên trái đất, vào bất kỳ thời điểm nào, cũng có thể “nhìn thấy” tối thiểu 4 vệ tinh. Công việc của một máy thu GPS là xác định vị trí của 4 vệ tinh hay hơn nữa, tính toán khoảng cách từ các vệ tinh và sử dụng các thông tin đó để xác định vị trí của chính nó. Quá trình này dựa trên một nguyên lý toán học đơn giản.

* Vĩ độ (ký hiệu: φ) của một điểm bất kỳ trên mặt Trái Đất là góc tạo thành giữa đường thẳng đứng (phương của dây dọi, có đỉnh nằm ở tâm hệ tọa độ-chính là trọng tâm của địa cầu) tại điểm đó và mặt phẳng tạo bởi xích đạo. Đường tạo bởi các điểmcó cùng vĩ độ gọi là vĩ tuyến, và chúng là những đường tròn đồng tâm trên bề mặt Trái Đất.
* Kinh độ (ký hiệu: λ) của một điểm trên bề mặt Trái Đất là góc tạo ra giữa mặtphẳng kinh tuyến đi qua điểm đó và mặt phẳng kinh tuyến gốc. Những đường thẳng tạo bởi các điểm có cùng kinh độ gọi là kinh tuyến. Bằng cách phối hợp hai góc này, ta có thể xác định được vị trí nằm ngang của bất kỳ điểm nào trên Trái Đất.

## 2.2 Tìm hiểu về hệ thống mạng GSM:

### 2.2.1 Mạng thông tin di động toàn cầu (GSM):

- Định nghĩa GSM:

Mạng thông tin di động toàn cầu là tiêu chuẩn chung cho các thuê bao di động di chuyển giữa các vị trí địa lý khác nhau mà vẫn giữ được liên lạc.

- Các mạng điện thoại GSM ở Việt Nam:

Ở Việt Nam và các nước trên Thế Giới, mạng điện thoại GSM vẫn chiếm đa số, Việt Nam có các mạng điện thoại GSM lớn đó là:

+ Mạng vinaphone.

+ Mạng mobiphone.

+ Mạng Viettel.

- Công nghệ của mạng GSM

Các mạng điện thoại GSM sử dụng công nghệ TDMA.

TDMA (Time Division Multiple Access). Phân chia các truy cập theo thời gian. Công nghệ TDMA là công nghệ cho phép 8 máy di động có thể sử dụng chung môt kênh để đàm thoại, mỗi máy sẽ sử dụng 1/8 khe thời gian để truyền và nhận thông tin.

- Công nghệ CDMA

Khác với công nghệ TDMA của các mạng GSM là công nghệ CDMA của các mạng như:

+ Mạng Sfone 095.

+ Mạng HT Mobile 092.

CDMA Phân chia các truy cập theo mã. Công nghệ CDMA sử dụng mã số cho mỗi cuộc gọi, và nó không sử dụng một kênh để đàm thoại như công nghệ TDMA mà sử dụng cả một phổ tần (nhiều kênh một lúc) vì vậy công nghệ này có tốc độ truyền dẫn tín hiệu cao hơn công nghệ TDMA.

### 2.2.2 Một số tập lệnh AT cơ bản sử dụng cho ứng dụng GSM:

1. ***Các thuật ngữ:***

<CR>: Carriage return (0x0D).

<LF>: Line Feed (0x0A).

MT: Mobile Terminal. Thiết bị đầu cuối mạng (trong trường hợp này là module SIM548).

TE: Terminal Equipment. Thiết bị đầu cuối (máy tinh, hệ vi điều khiển).

1. ***Các tập lệnh chung:***

|  |  |
| --- | --- |
| AT | Kiểm tra module có hoạt động không   * Trả về: OK nếu hoạt động bình thường, báo lỗi hoặc không trả về nếu có lỗi xảy ra |
| ATZ | Thiết lập lại cấu hình mặc định cho Module SIM |
| ATEx | Bật (x=1) hoặc tắt (x=0) chế độ phản hồi lệnh vừa gửi (nên tắt đi) |
| AT+IPR = [baud rate] | Cài đặt tốc độ Baud của Module SIM. Chỉ hỗ trợ những tốc độ Baud sau: 0, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200. |
| AT&W | Lưu lại cài đặt mới |
| AT+CPIN? | Kiểm tra Simcard   * Trả về: +CPIN: READY OK (nếu tìm thấy simcard) |
| AT+CSQ | Kiểm tra chất lượng song   * Trả về: +CSQ: xx,0 OK (xx là chất lượng sóng, tối đa là 31) |
| AT+COPS? | Kiểm tra tên nhà mạng   * Trả về: +COPS: 0,0, ”Viettel Mobile” OK (nhận dạng được nhà mạng là Viettel Mobile) * Trả về: +COPS: 0 (không thấy nhà mạng) |

Bảng 1. 1: Các tập lệnh AT chung

#### Các lệnh gọi điện:

|  |  |
| --- | --- |
| ATD[PhoneNumber]; | Thực hiện cuộc gọi. Module sẽ trả về đáp ứng với những trường hợp sau :   * Cuộc gọi bị mất sóng: **NO DIALTONE** * Cuộc gọi bị ngắt: **BUSY** * Cuộc gọi đã kết thúc: **NO CARRIER** * Cuộc gọi không nhận được trả lời: **NO ANSWER** |
| ATDL | Thực hiện cuộc gọi đến số điện thoại cuối cùng đã liên lạc. |
| ATA | Chấp nhận cuộc gọi đến |
| ATH | Hủy cuộc gọi |
| AT+CLIP=1 | Hiển thị thông tin cuộc gọi đến (Module SIM phản hồi **RING** khi có cuộc gọi đến) |
| AT+CUSD=1 | Chuyển chế độ USD để tra số dư tài khoản |
| ATD\*101#; | Kiểm tra tài khoản  Trả về: +CUSD: 0, “Tai khoan goc cua quy khach la …. |

Bảng 1. 2: Các tập lệnh AT về cuộc gọi

#### Các lệnh gửi tin nhắn SMS:

|  |  |
| --- | --- |
| AT+CMGF=[mode] | Thiết lập định dạng đọc ghi của tin nhắn.   * 0: PDU mode. * 1: Text mode. |
| AT+CMGS= “[PhoneNumber]” | Thực hiện gửi tin nhắn.  Đợi khi có ký tự “>” được trả về thì bắt đầu viết nội dung tin nhắn. Sau khi viết xong có thể gửi bằng tổ hợp phím **Ctrl+Z** hoặc kết thúc cuối nội dung tin nhắn bằng ký tự **0x1A (<SUB>)**. |
| AT+CNMI=[mode] | Thiết lập kiểu xử lý tin nhắn.   * 2,1: mode hiển thị vị trí tin nhắn theo định dạng +CMTI. * 2,2: mode hiển thị nội dung ngay khi tin nhắn đến theo định dạng +CMT. |
| AT+CMGR=x | Đọc nội dung tin nhắn đã được gửi đến tại vị trí tin nhắn.   * x là vị trí tin nhắn cần đọc. |
| AT+CMGDA= “[type]” | Xóa tin nhắn.   * DEL READ: xóa tất cả tin nhắn đã đọc. * DEL UNREAD: xóa tất cả tin nhắn chưa đọc. * DEL SENT: xóa tất cả tin nhắn đã gửi. * DEL UNSENT: xóa tất cả tin nhắn chưa gửi được. * DEL INBOX: xóa tất cả tin nhắn nhận. * DEL ALL: xóa tất cả tin nhắn. |

Bảng 1. 3: Các tập lệnh AT về gửi tin nhắn SMS

#### ****Lệnh sử dụng trong chức năng GPRS:****

|  |  |
| --- | --- |
| AT+CIPSTART= “protocol”,“host”,“port” | Kết nối GPRS theo cấu hình được lựa chọn.   * Protocol: TCP/UDP * Host: server domain hoặc IP. * Port: cổng kết nối. |
| AT+CIPSHUT | Đóng kết nối GPRS. |
| AT+CIPSEND | Gửi yêu cầu đến host đang kết nối.  Đợi khi có ký tự “>” được trả về thì bắt đầu viết yêu cầu kết nối theo định dạng:   * GET /[url] HTTP/[version]<CR><LF> * Host: server domain hoặc IP<CR><LF> * Connection: Keep-Alive<CR><LF><CR><LF><SUB> |

Bảng 1. 4: Các tập lệnh AT về GPRS

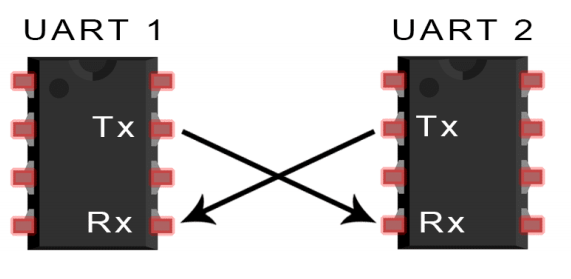
## 2.3 Tìm hiểu về chuẩn giao tiếp UART:

### 2.3.1 Khái quát về UART:

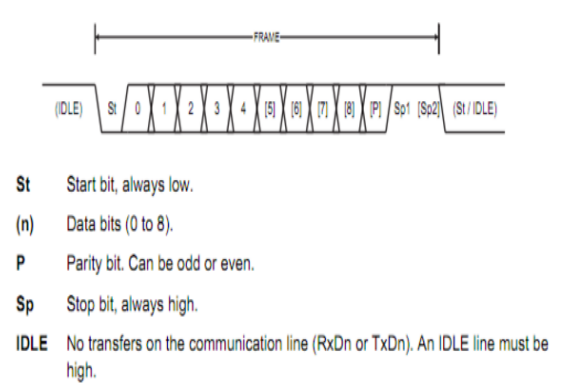
UART (Universal Asynchronous Receive/Transmit) là chuẩn giao tiếp truyền

nhận dữ liệu không đồng bộ. Đây là chuẩn giao tiếp phổ biến và dễ sử dụng, thường dùng trong giao tiếp giữa vđk với nhau hoặc với các thiết bị khác.

Cách hoạt động:



Hình 2. 1: Kết nối UART



Hình 2. 2: Khung truyền của giao tiếp UART

Hai thiết bị giao tiếp UART với nhau thông qua hai đường dẫn RX (read) và TX (transmit) (Hình 1) Vì là giao tiếp không đồng bộ nên hai thiết bị phải được cài đặt thống nhất về khung truyền (Hình 2), tốc độ truyền.

- Start bit: báo hiệu quá trình truyền dữ liệu.

- Data bits: dữ liệu cần giao tiếp, thường là 8 bit.

- Parity bit: bit kiểm tra chẵn lẻ, dùng để phát hiện lỗi. 1

- Stop bit: báo hiệu kết thúc một frame dữ liệu. Có thể tùy chọn 1 hoặc 2

### 2.3.2 Các hoạt động của UART:

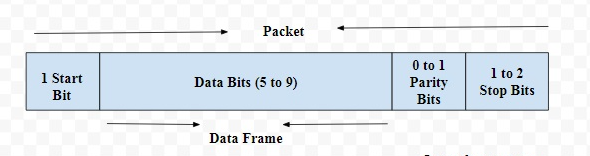
UART là giao thức truyền thông không đồng bộ, nghĩa là không có xung Clock, các thiết bị có thể hiểu được nhau nếu các Setting giống nhau.

UART là truyền thông song công (Full duplex) nghĩa là tại một thời điểm có thể truyền và nhận đồng thời.

Trong đó quan trọng nhất là Baund rate (tốc độ Baund) là khoảng thời gian dành cho 1bit được truyền. Phải được cài đặt giống nhau ở gửi và nhận.

Sau đó là định dạng gói tin.

Định dạng gói tin như sau:



Hình 2. 3: Định dạng gói tin

**Start – Bit**

Start-bit còn được gọi là bit đồng bộ hóa được đặt trước dữ liệu thực tế. Nói chung, một đường truyền dữ liệu không hoạt động được điều khiển ở mức điện áp cao. Để bắt đầu truyền dữ liệu, truyền UART kéo đường dữ liệu từ mức điện áp cao (1) xuống mức điện áp thấp (0). UART thu được thông báo sự chuyển đổi này từ mức cao sang mức thấp qua đường dữ liệu cũng như bắt đầu hiểu dữ liệu thực. Nói chung, chỉ có một start-bit.

**Stop – Bit**

Bit dừng được đặt ở phần cuối của gói dữ liệu. Thông thường, bit này dài 2bit nhưng thường chỉ sử dụng 1 bit. Để dừng sóng, UART giữ đường dữ liệu ở mức điện áp cao.

**Partity Bit**

Bit chẵn lẻ cho phép người nhận đảm bảo liệu dữ liệu được thu thập có đúng hay không. Đây là một hệ thống kiểm tra lỗi cấp thấp & bit chẵn lẻ có sẵn trong hai phạm vi như Chẵn lẻ – chẵn lẻ cũng như Chẵn lẻ – lẻ. Trên thực tế, bit này không được sử dụng rộng rãi nên không bắt buộc.

**Data frame**

Các bit dữ liệu bao gồm dữ liệu thực được truyền từ người gửi đến người nhận. Độ dài khung dữ liệu có thể nằm trong khoảng 5 & 8. Nếu bit chẵn lẻ không được sử dụng thì chiều dài khung dữ liệu có thể dài 9 bit. Nói chung, LSB của dữ liệu được truyền trước tiên sau đó nó rất hữu ích cho việc truyền.

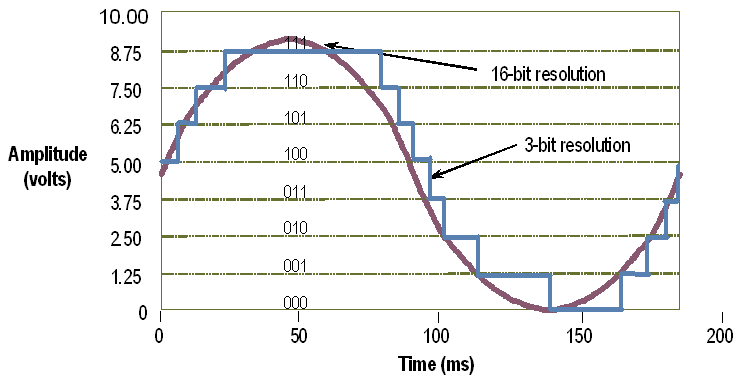
## 2.4 Tìm hiểu về đọc ADC:

### 2.4.1 Khái quát về ADC:

ADC Analog to Digital Convert là bộ chuyển đôi tương tự sang số. Đại lượng tương tự là Điện áp Vin được so sánh với điện áp mẫu Vref (giá trị lớn nhất), sau đó được chuyển đổi thành số lưu vào thanh ghi DATA của bộ chuyển đổi đó.

Có 2 tham số quan trọng của bộ ADC cần lưu ý:

* Tốc độ lấy mẫu (sampling) được tính theo số chu kì chuyển đổi
* Độ phân giải: Tính theo Bit bộ ADC có độ phân giải 10 Bit sẽ có 2^10 = 1024 giá trị



Hình 2. 4: Độ phân giải ADC

### 2.4.2 Các chức năng chính của ADC trong STM32:

Độ phân giải 12Bit

Sinh ra ngắt tại các sự kiện End of convert, End of Injected, Analog Watchdog

Chế độ Single hoặc Continuous

Chế độ Scan tự động quét từ Kênh 0 đến Kênh n (mỗi bộ có 10 kênh tối đa)

Có cơ chế cân chỉnh tay

Data Alignment (Căn chỉnh Data) căn trái hoặc căn phải

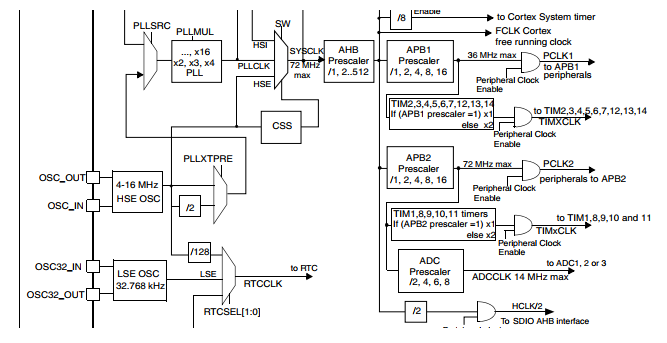
Cài đặt thời gian chuyển đổi đến từng Kênh

Có thể kích hoạt bằng xung bên ngoài

Chế độ Dual mode sử dụng cùng lúc 2 hoặc nhiều bộ ADC

Hỗ trợ DMA

Tần số chuyển đổi ADC được lấy từ bộ ABP2 thông qua ADC prescaler và phải nhỏ hơn 14mhz



Hình 2. 5: Sơ đồ tần số ADC lấy từ bộ ABP2

### ACD một kênh chế độ Single và Continuous:

Với chế độ Single bộ ADC chỉ chuyển đổi 1 lần rồi dừng, một sự kiện ngắt được sinh ra nếu bit EOCIR được set lên 1

Với chế độ Continuous bộ ADC sẽ chuyển đổi liên tục, một sự kiện ngắt được sinh ra nếu bit EOCIR được set lên 1

DMA sẽ sinh ra nếu bit DMA dc set lên 1(DMA chỉ có trên ADC1 và ADC3)

Kết quả convert được lưu vào thanh ghi DR

Bắt đầu convert bằng cách set bit ADON lên 1

Cơ chế để tạo ADC như sau:

1. Enable bộ clock cho ADC, ghi hệ số chia cho ADC prescaler
2. Chọn các kênh cần chuyển đổi
3. Chọn chế độ chuyển đổi Sinlge, Continuous, Scan, Discontinuous
4. Chọn thời gian lấy mẫu (Sampling Time)
5. Chọn Ngắt hoặc DMA
6. Start bộ chuyển đổi
7. Kiểm tra cờ EOC hoặc trong ngắt đọc dữ liệu từ thanh ghi DR về

### 2.4.4 [ADC](https://en.wikipedia.org/wiki/Analog-to-digital_converter) nhiều kênh sử dụng chế độ Scan Mode:

Chế độ Scan chỉ được bật khi sử dụng 2 kênh ADC trở lên.

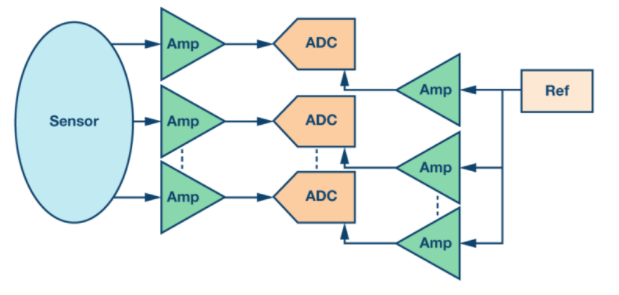
Nếu Bit CONT được set lên 1, sau khi scan hết các kênh ADC tự động lặp lại thao tác Scan.

Nếu bit CONT ko được bật, sau khi ADC scan hết các kênh sẽ dừng lại.

Sau khi thực hiện chuyển đổi ADC tại mỗi kênh, cờ EOC sẽ được bật, ngắt hoặc DMA sẽ được sinh ra, giá trị thu được sẽ lưu vào thanh ghi DR.

Khi thực hiện chuyển đổi xong tại kênh tiếp theo, giá trị sẽ ghi đè lên thanh ghi DR

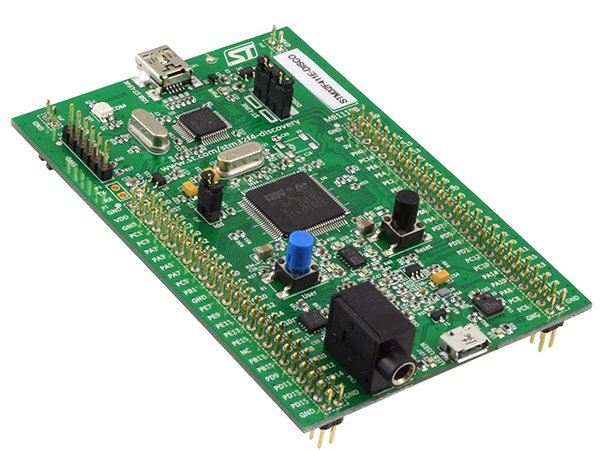
Khi sử dụng DMA giá trị chuyển đổi sẽ được chuyển tới SRAM trước khi vào bộ DMA



Hình 2. 6: ADC chế độ Scan Mode

## 2.5 Giới thiệu phần cứng:

### 2.5.1 STM32F411 Discovery:



Hình 2. 7: Module STM32F411RE Discovery

Bộ vi điều khiển STM32F411 là một phần của dòng STM32 Dynamic Efficiency ™. Các thiết bị này là cấp độ đầu vào của Dòng F4 Hiệu suất cao và cung cấp sự cân bằng tốt nhất giữa mức tiêu thụ điện năng động (ở chế độ chạy) và hiệu suất xử lý, đồng thời tích hợp nhiều tính năng giá trị gia tăng cao trong các gói nhỏ cỡ 3 x 3 mm.

MCU STM32F411 cung cấp hiệu suất của lõi Cortex®-M4 với đơn vị dấu chấm động, chạy ở tốc độ 100 MHz, đồng thời đạt được giá trị tiêu thụ điện năng thấp vượt trội ở các chế độ chạy và dừng.

Với Chế độ thu thập hàng loạt (BAM) mới, tối ưu hóa mức tiêu thụ điện năng cho việc phân phối dữ liệu, STM32F411 nâng Hiệu suất động lên một tầm cao mới. BAM này cho phép trao đổi hàng loạt dữ liệu thông qua các thiết bị ngoại vi giao tiếp với phần còn lại của thiết bị (bao gồm cả CPU) đang ở chế độ tiết kiệm năng lượng.

Lõi: ARM® 32bit Cortex®-M4

Nhiệt độ hoạt động: -40ºC – 85ºC

Bộ nhớ: Bộ nhớ Flash 512KB, RAM 128KB trong gói LQFP100

Tốc độ: 100MHz

Nguồn cung cấp cho board: thông qua bus USB hoặc từ nguồn 5V bên ngoài.

Nguồn cung cấp cho ngoại vi: 3V và 5V

8 đèn LED: LD1( đỏ/xanh) giao tiếp USB; LD2 (đỏ) tín hiệu nguồn 3,3V; 4 LED dành cho người dùng LD3(cam), LD4(xanh), LD5(đỏ), LD6(xanh dương), 2 đèn LED USB OTG: LD7(xanh) VBus và LD8( đỏ) quá dòng

2 nút nhấn (người dùng và reset)

Bộ điều khiển DMA đa mục đích với bộ điều khiển DMA 16 luồng FIFO và hỗ trợ liên tục.

11 bộ timer: 6 bộ timer 16bit , 2 bộ định thời 32bit tần số 100MHz với mỗi bộ có tối đa 4 IC/OC/PWM hoặc bộ đếm xung 2 bộ watchdog timers and một bộ ngắt System Tick

81 cổng I/0, 16line ngắt

Các giao diện giao tiếp: I2C, USART, SPI/I2S, STDIO, ISO.

### 2.5.2 Module sim900a:



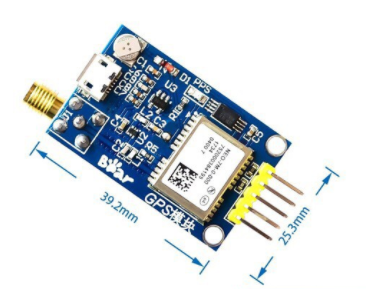
Hình 2. 8: Module sim 900A

SIM900(A) là module GSM/GPRS của hãng SIMCom được thiết kế dưới dạng module chipset, nhỏ gọn, giá thành thấp, hoạt động ổn định và phù hợp cho nhiều mục đích sử dụng. Module SIM900(A) có các tính năng cơ bản của một chiếc điện thoại như gọi điện thoại, nhắn tin, truy cập GPRS, ...

Thông số kỹ thuật:

* IC chính: Module GSM/GPRS Sim900A
* Nguồn cấp: 4.5-5V, có thể sử dụng với nguồn dòng thấp từ 500mA trở lên, nên dùng nguồn 2A để đảm bảo Sim hoạt động tốt.
* Khe Sim
* Tích hợp led báo trạng thái Sim900 hoạt động
* Tích hợp tụ bù điện dung cao giúp cho Module Sim hoạt động tốt
* Dòng khi ở chế độ chờ: 10mA
* Dòng khi hoạt động: 100mA đến 2A
* Nhiệt độ hoạt động: -40°C đến +85°C

### 2.5.3 Module GPS\_NEO6M:



Hình 2. 9: Module NEO-6M GPS

Module NEO-6M GPS là module định vị toàn cầu sử dụng hệ thống vệ tinh GPS của Mỹ. Module GPS NEO-6M có tốc độ xử lí vị trí nhanh và chính xác, có nhiều mức năng lượng hoạt động, phù hợp với các ứng dụng chạy pin. Với một số ứng dụng như:

- Xác định tọa độ (kinh tuyến, vĩ tuyến) hiện tại của module trên bề mặt trái đất với sai số nhỏ nhất < 1m.

-  Xác định thời gian quốc tế được cấp bởi đồng hồ nguyên tử trên vệ tinh gửi về. Từ đó cũng có thể suy ra thời gian đồng hồ nơi ở theo tắc trừ múi giờ. Khỏi cần module RTC.

 - Chỉ cần 3 vệ tinh là có thể xác định được tọa độ, chỉ cần 4 vệ tinh là có thể xác định được độ cao hiện tại so với mực nước biển.

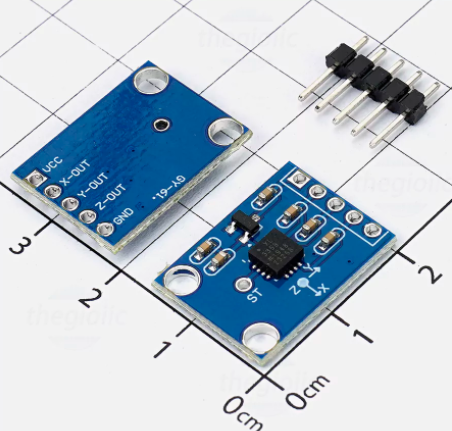
- Có thể tính toán ra tốc độ di chuyển, hướng di chuyển của vật thể được gắn module GPS.

 - Giải các bài toán về tính toán giữa 2 điểm bất kì, tính diện tích ở một không gian cực kì rộng lớn.

Thông số kỹ thuật:

* Hãng Ublox.
* Ic chính: Neo 6 nguyên chiếc của Ublox.
* Hỗ trợ GPS-QZSS-GLONASS
* Nguồn cấp : 3.5-5V
* Giao tiếp : UART
* Anten rời
* Chu kỳ cập nhật tọa độ 0,1s đến 1s tùy từng địa hình có thuận lợi hay không.
* Phần mềm hỗ trợ với rất nhiều tính năng được Ublox cho tải miễn phí.
* Ứng dụng: có tất cả khả năng như đã giới thiệu phía trên.
* Kích thước: 39,2 \* 25,3 mm

### 2.5.4 Cảm biến gia tốc GY-61:



Hình 2. 10: Module gia tốc GY-61 ADXL335

Module gia tốc kế 3 trục GY-61 DXL335 là module cảm biến gia tốc kế 3 trục dựa trên mạch tích hợp ADXL335. ADXL335 là một máy đo gia tốc với mức tiêu thụ năng lượng thấp. Cảm biến có đầy đủ dải cảm biến +/-3g. Nó có thể đo gia tốc tĩnh của trọng lực trong các ứng dụng cảm biến độ nghiêng, cũng như gia tốc động do chuyển động, sốc hoặc rung.

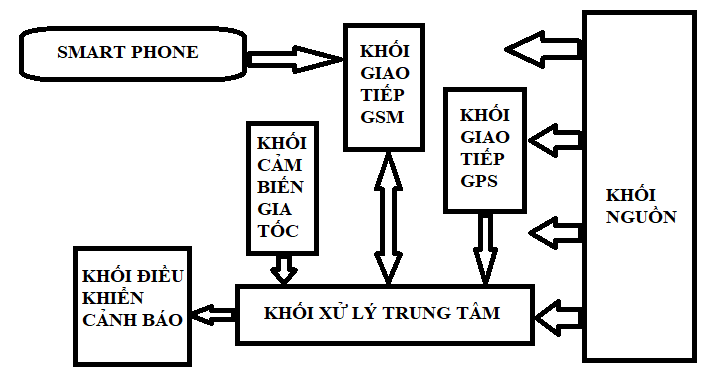
Module gia tốc với kích thước nhỏ gọn, cung cấp điện năng thấp, được ứng dụng cho các hệ thống trò chơi, thiết bị di động…

Thông số kỹ thuật:

* Chip cảm biến: ADXL335
* Dải điện áp hoạt động: 3-5V
* Dòng cung cấp: 400uA
* Giao diện: đầu ra số lượng tương tự
* Nhiệt độ hoạt động: -40oC -- 80oC
* Độ nhạy: 300mv/g

# CHƯƠNG 3: **T**HIẾT KẾ PHẦN CỨNG

## 3.1 Sơ đồ khối:



Hình 3. 1: Sơ đồ khối hệ thống

Chức năng các khối:

- Khối nguồn: cung cấp nguồn điện cho các khối xử lý trung tâm, khối giao tiếp GPS, khối giao tiếp GSM, khối điều khiển cảnh báo, khối cảm biến gia tốc.

- Khối xử lý trung tâm: là trung tâm xử lý của mạch, thực hiện công việc như: Tách tín hiệu kinh độ, vĩ độ nhận từ khối giao tiếp mạng GPS. Điều khiển khối giao tiếp mạng GSM gửi tin nhắn và gọi điện về số đã cài trước. Nhận tín hiệu từ khối cảm biến gia tốc để thực hiện nhiệm vụ rồi xử lý đưa ra kết quả là khối điều khiển cảnh báo và thực hiện tin nhắn, gọi điện.

- Khối giao tiếp mạng GSM: dùng để giao tiếp mạng di động với smartphone giúp gửi tin nhắn, cuộc gọi và giao tiếp với khối xử lý trung tâm để thực hiện các cuộc gọi và tin nhắn theo yêu cầu.

-Khối giao tiếp mạng GPS: dùng để đo kinh độ, vĩ độ gửi về khối xử lý trung tâm.

- Khối cảm biến gia tốc: dùng để đo độ nghiêng của xe nhằm để phát hiện xe bị xảy ra tai nạn.

- Khối điều khiển cảnh báo: dùng để cảnh báo khi ngã xe bằng chuông cảnh báo.

- Smarphone: là thiết bị có thể sử dụng google map để giám sát vị trí xe trên bản đồ. Đồng thời được cảnh báo ngã xe qua tin nhắn, gọi điện. Ngoài ra còn có thể thực hiện tin nhắn trong thiết bị để yêu cầu thiết bị gửi tọa độ về.

Sau khi đã biết được chức năng của từng khối, nhưng để biết chúng kết hợp với nhau ra sao và đem lại những tác dụng gì thì phải tìm hiểu hoạt động của hệ thống như thế nào sau khi gắn các khối lại với nhau.

Hoạt động của hệ thống:

Sau khi cấp nguồn, khối giao tiếp mạng GPS sẽ cập nhật kinh độ, vĩ độ gửi qua khối xử lý trung tâm và được cập nhật liên tục. Khối xử lý trung tâm lại nhận tín hiệu từ cảm biến gia tốc để đưa ra lệnh điều khiển khối giao tiếp mạng GSM gửi tin nhắn, thực hiện cuộc gọi về smartphone. Đồng thời khối điều khiển cũng gửi tín hiệu ra khối điều khiển cảnh báo. Smartphone nhận được tín hiệu từ khối giao tiếp mạng GSM để được cảnh báo kịp thời. Smartphone nhận tín hiệu cảnh báo từ khối giao tiếp mạng GSM và báo vị trí về Smartphone qua tin nhắn.

## 3.2 Tính toán và thiết kế:

### 3.2.1 Khối nguồn:

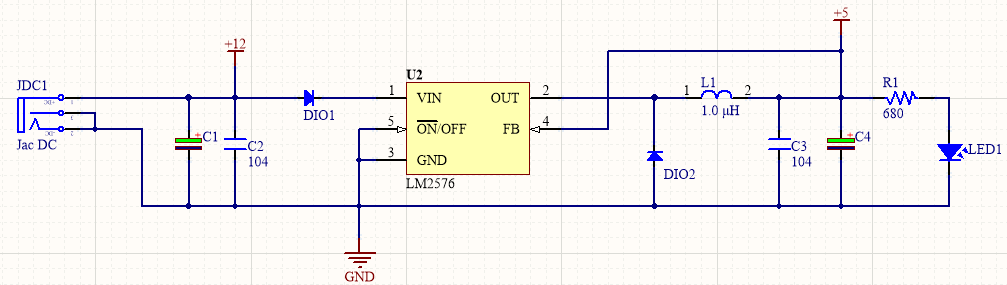
Khối nguồn sẽ cung cấp nguồn nuôi cho toàn mạch. Từ datasheet của các linh kiện trong sơ đồ nguyên lí, ta thấy điện áp hoạt động trung bình chủ yếu của toàn mạch là 12V và 5V, nên ta sẽ phải phân nguồn 12V và 5V để cấp cho toàn mạch.

Đối với nguồn 12V, sẽ dùng trực tiếp nguồn từ module Adapter 12V-2A cấp cho thiết bị cảnh báo là chuông và nguồn 5V.



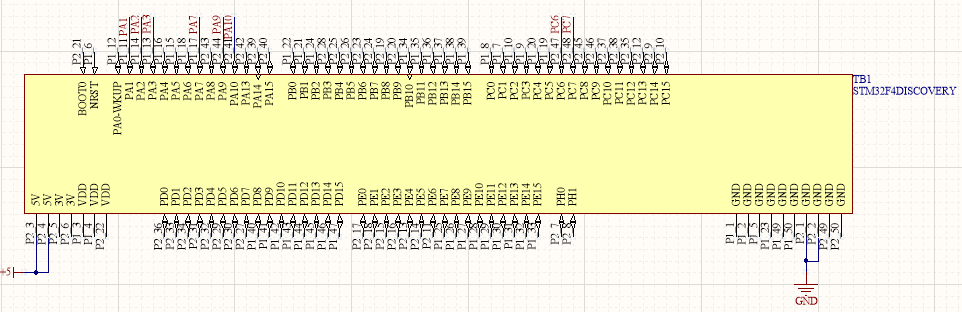
Hình 3. 2: Adapter 12V - 2A

Đối với nguồn 5V, dùng mạch nguồn đầu vào 12V, đầu ra 5V.



Hình 3. 3: Sơ đồ nguyên lý khối nguồn

### 3.2.2 Khối vi xử lý:



Hình 3. 4: Sơ đồ nguyên lý khối vi điều khiển

Sử dụng Kit vi điều khiển STM32F411 thuận tiện cho việc lập trình và tiết kiệm thời gian thiết kế vì:

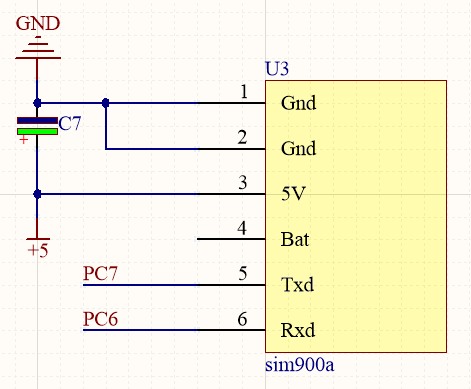
- Đáp ứng được số cổng giao tiếp UART, đọc ADC.

- STM32 hiện đang là vi điều khiển đang phát triển mạnh mẽ với cộng đồng hỗ trợ cực kì rộng lớn.

- Sử dụng Kit STM32 dễ dàng trong việc thiết kế vì không phải chạy nhiều dây như dùng IC.

Khối xử lý trung tâm với chân A7 nối với mạch điều khiển cảnh báo, chân PA1, PA2, PA3 dùng để kết nối với module cảm biến gia tốc bằng đọc ADC. Chân PA9, PA10 dùng để kết nối với module GPS\_neo6 theo chuẩn giao tiếp UART. Chân PC6, PC7 dùng kết nối với module Sim theo chuẩn giao tiếp UART.

### 3.2.3 Khối Module sim900A:



Hình 3. 5: Sơ đồ nguyên lý khối Sim900A

Để thực hiện việc cảnh báo bằng tin nhắn SMS và gọi điện để giám sát thì người ta thường dùng các chip Sim để giao tiếp điều khiển với các vi xử lý. Có rất nhiều loại chip Sim hiển nay trên thị trường, nhưng có các loại phổ biến nhất là Sim800, Sim808, Sim900, ... của hãng SimCom, Mlab.

Trong đề tài này sử dụng Module Sim900A cho việc điều khiển vì:

- Có tốc độ và độ ổn định cao.

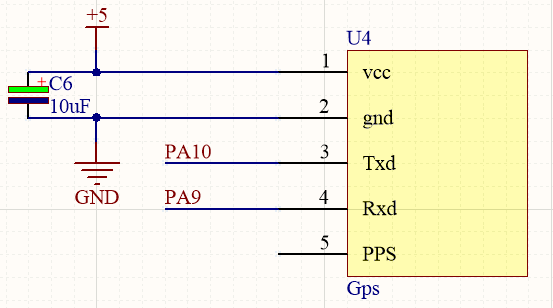
- Tích hợp thêm chức năng GSM/GPRS.

- Giá thành hợp lý

Sơ đồ nguyên lí khối Sim900A chân TX - RX được giả lập sẽ được đấu chéo nhau với VĐK.

Mắc giữa nguồn 5V với GND của Module Sim để Module hoạt động tốt và ổn định hơn.

### 3.2.4 Khối Module GPS\_NEO6M:



Hình 3. 6: Sơ đồ nguyên lý khối GPS

Để thực hiện gửi tọa độ (kinh độ, vĩ độ) về khối xử lý để xử lý dữ liệu rồi gửi về Smart phone thông qua GSM bằng tin nhắn, nhằm xác định vị trí hiện tại của xe khi gặp sự cố.

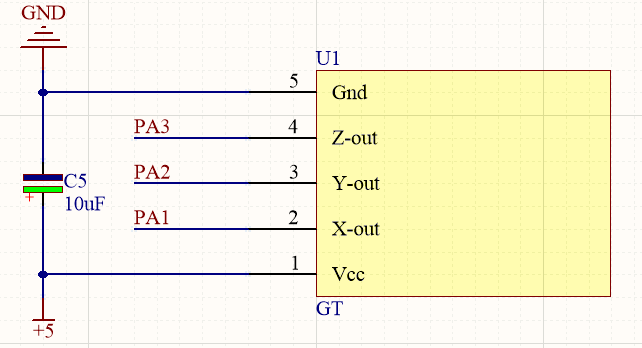
Trong đề tài chọn Module GPS\_NEO6M vì:

* Có tốc độ và ổn định cao
* Giá thành rẻ

Sơ đồ nguyên lí khối GPS chân TX - RX được giả lập sẽ được đấu chéo nhau với VĐK.

Mắc giữa nguồn Vcc với GND của Module GPS để Module hoạt động tốt và ổn định hơn.

### 3.2.5 Khối cảm biến gia tốc:



Hình 3. 7: Sơ đồ nguyên lý khối cảm biến gia tốc

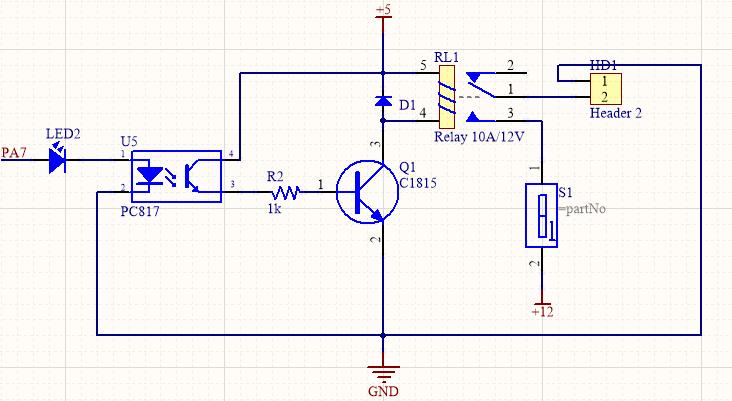
Dựa theo yêu cầu thiết kế, cần một thiết bị có khả năng đo độ nghiêng của xe. Với yêu cầu đó thì chọn cảm biến gia tốc GY-61 hỗ trợ giao tiếp và đo giá trị.

GY-61 đọc Analog nên khi giao tiếp với VĐK thì chỉ cần đọc ADC

Giá thành hợp lý và độ ổn định cao

Trong sơ đồ nguyên lý khối cảm biến gia tốc kết nối 3 chân Xout, Yout, Zout vào chân PA1, PA2, PA3 của STM32 truyền dữ liệu.

### 3.2.6 Khối điều khiển cảnh báo:



Hình 3. 8: Sơ đồ nguyên lý khối cảnh báo

Trong mạch sử dụng nguồn 12V từ adapter 12V cho chuông cảnh báo, còn mạch sử dụng nguồn 5V.

Sử dụng một cái công tắc để có thể tắt cảnh báo chuông khi không cần thiết.

* Tải: Chuông cảnh báo

Nguồn cấp: 12V

Tính toán:

Từ tải ta chọn Relay 5V chịu tải 12V/2A

Để kích dẫn relay IRLmin = 5mA

Chọn IRL = 15mA

Chọn opto PC817 có thông số:

VoutPC817/dẫn = 0.2V

VinPC817 = 1.5V

BJT Q1 là C1815 có = 70

Igh/Q1 = = = 0.214 mA

Để BJT Q1 dẫn bảo hòa thì:

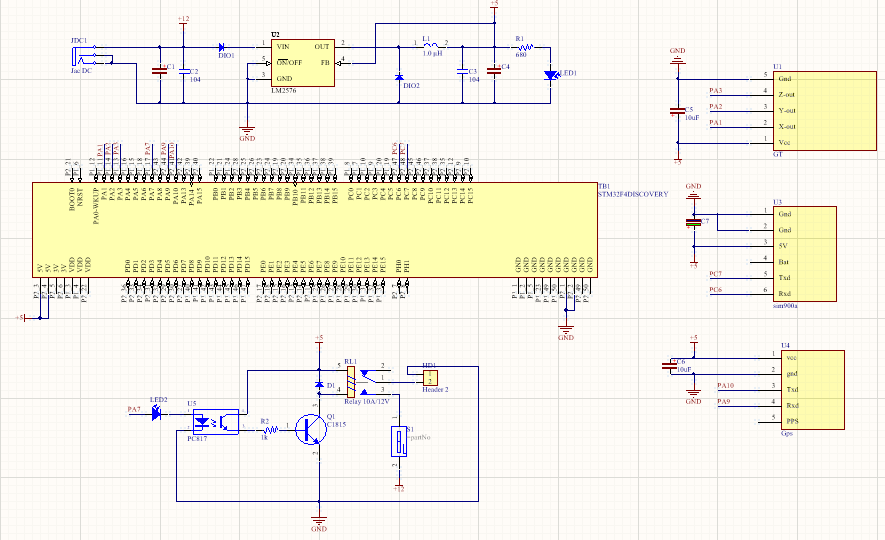
Ib/Q1 = 2 Igh/Q1 = 2\*0.214 = 0.428 mA

Ta có: VR2 = 5V - VoutPC817/dẫn –Vbe/Q1 = 5 – 0.2 – 0.7 = 4.1V

-> R2 = = = 9.5K

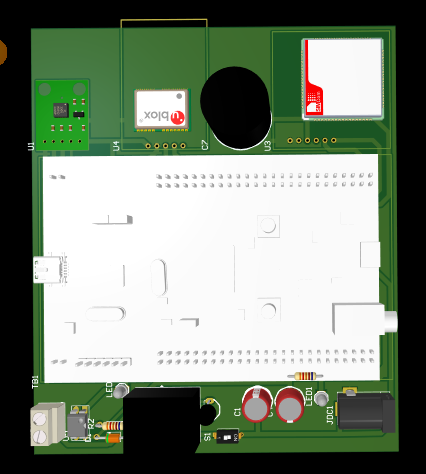
Chọn R2 = 8.6k

## 3.3 Mạch nguyên lý:



Hình 3. 9: Sơ đồ nguyên lý mạch tổng quát

## 3.4 Mạch in 3D:



Hình 3. 10: Mạch Altium 3D

## 3.5 Mạch thi công:

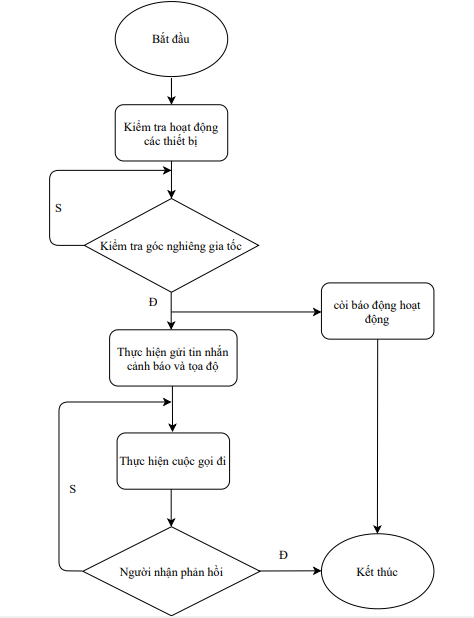


Hình 3. 11: Mạch thực tế

# CHƯƠNG 4: THIẾT KẾ PHẦN MỀM:

## 4.1 Lưu đồ:

### 4.1.1 Lưu đồ báo ngã phương tiện:

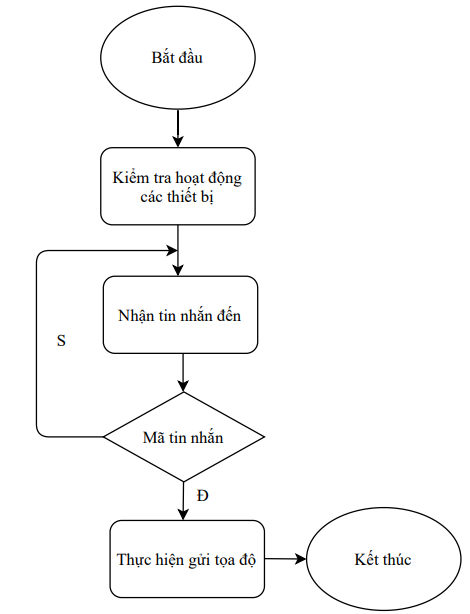


Hình 4. 1: Lưu đồ báo ngã phương tiện

* Giải thích lưu đồ:

Khi khởi động các module, thiết bị sẽ hoạt động. Khi xe ngã, chúng ta đo độ nghiêng, nếu độ nghiêng nằm vào mốc quy định thì còi báo động sẽ hoạt động, đồng thời module sim sẽ thực hiện gửi tin cảnh báo và tọa độ. Sau đó thực hiện cuộc gọi đi, nếu cuộc gọi đi không có phản hồi thì vẫn tiếp tục gọi cho đến khi có phản hồi từ người nhận thì kết thúc cuộc gọi. Kết thúc quá trình.

### 4.1.2 Lưu đồ giám sát phương tiện



Hình 4. 2: Lưu đố giám sát phương tiện

* Giải thích lưu đồ:

Khi các thiết bị, module đã hoạt động. Người gửi sẽ gửi một mã tin đi, nếu đúng với mã của hệ thống thì hệ thống sẽ lấy tọa độ từ GPS trả về, thông qua module sim và truyền đi cho người nhận. Kết thúc quá trình

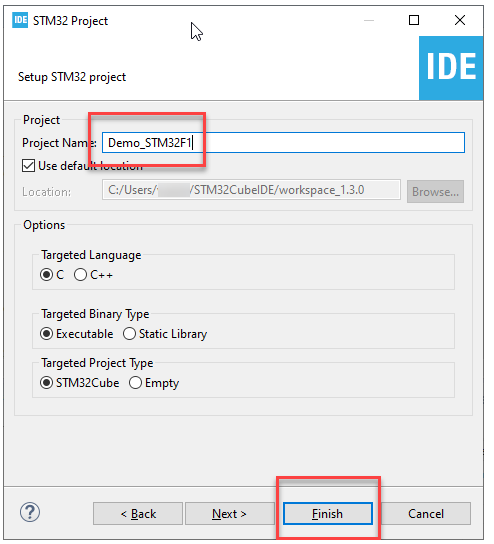
## 4.2 Phần mềm lập trình cho vi điều khiển:

### 4.2.1 Giới thiệu phần mềm STM32 CubeIDE:

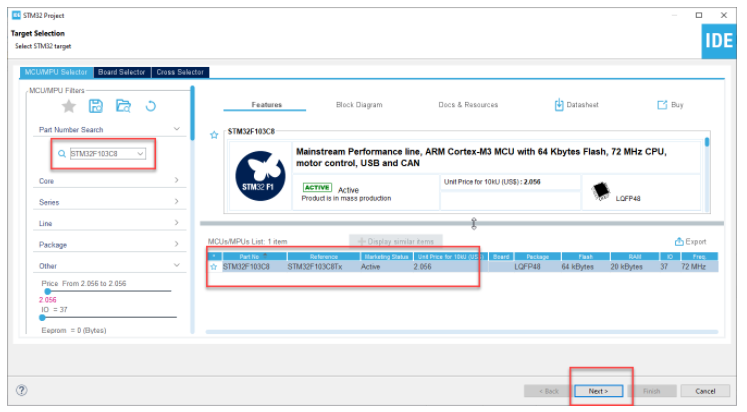
Nền tảng phát triển C / C ++ với các tính năng cấu hình ngoại vi, tạo mã, biên dịch mã và gỡ lỗi cho vi điều khiển và vi xử lý STM32. STM32CubeIDE tích hợp cấu hình STM32 và các chức năng tạo dự án từ STM32CubeMX để cung cấp trải nghiệm công cụ tất cả trong một và tiết kiệm thời gian cài đặt và phát triển. Sau khi lựa chọn MCU hoặc MPU STM32 trống hoặc bộ vi điều khiển hoặc bộ vi xử lý được cấu hình sẵn từ việc lựa chọn bo mạch hoặc chọn ví dụ, dự án sẽ được tạo và mã khởi tạo được tạo. Bất kỳ lúc nào trong quá trình phát triển, người dùng có thể quay lại quá trình khởi tạo và cấu hình thiết bị ngoại vi hoặc phần mềm trung gian và tạo lại mã khởi tạo mà không ảnh hưởng đến mã người dùng. STM32CubeIDE bao gồm các bộ phân tích xây dựng và ngăn xếp cung cấp cho người dùng thông tin hữu ích về tình trạng dự án và các yêu cầu về bộ nhớ. STM32CubeIDE cũng bao gồm các tính năng gỡ lỗi tiêu chuẩn và nâng cao bao gồm các chế độ xem thanh ghi lõi CPU, bộ nhớ và thanh ghi ngoại vi, cũng như đồng hồ biến trực tiếp, giao diện Serial Wire Viewer hoặc bộ phân tích lỗi.

### 4.2.2 Cấu hình:

Để bắt đầu với STM32CubeIDE thì ban đầu sẽ hiện cửa sổ workspace

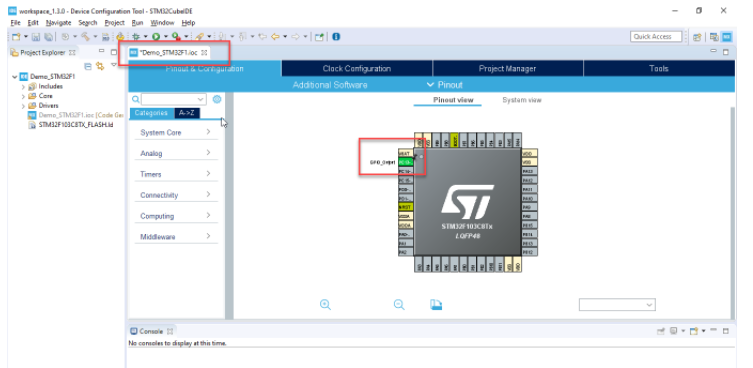
Select File > New > STM32 Project. Chọn vào Target Selection, chọn MCU/MPU Selector và tìm chip STM32F1Cx > Next 

Hình 4. 3: Cửa sổ cài đặt 1 STM32CubeIDE

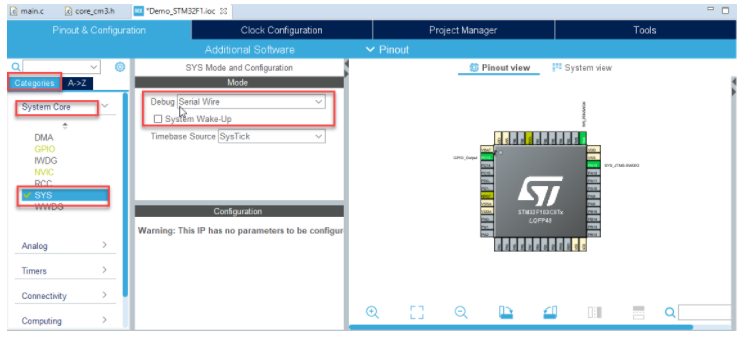


Hình 4. 4: Cửa sổ cài đặt 2 STM32CubeIDE

Sau khi xong xuôi, cửa sổ CubeMX sẽ hiện lên cấu hình chân. Điều chỉnh và sẳn sàng thực hiện code.



Hình 4. 5: Cửa sổ cấu hình 1 STM32



Hình 4. 6: Cửa sổ cấu hình STM32

Ấn vào File > Save hoặc Ctrl + S thì sẽ có thông báo tự gen code. Yes

# CHƯƠNG 5: KẾT QUẢ, NHẬN XÉT, ĐÁNH GIÁ:

## 5.1 Kết quả:

Sau thời gian tiến hành tìm hiểu, nghiên cứu các tài liệu chuyên ngành, tìm hiểu thêm qua mạng Internet, tổng hợp lại các kiến thức cũng như được sự hướng dẫn của thầy GVHD Ts.Nguyễn Duy Nhật Viễn. Tôi đã hoàn thành được đồ án môn học với đề tài “HỆ THỐNG GIÁM SÁT VÀ BÁO CÁO TAI NẠN PHƯƠNG TIỆN”. Sau đề tài đồ án này, tôi cũng đã nghiên cứu và tích lũy được thêm nhiều hiểu biết, kiến thức mới như:

Hiểu về sử dụng và các tính năng của STM32 như giao tiếp giữa STM32 với các module như: Module sim 900A, Module GPS Neo, cảm biến gia tốc.

Nghiên cứu biết được cách sử dụng của cảm biến gia tốc, nguyên lý hoạt động của cảm biến, các thông số kỹ thuật, các tính năng của cảm biến.

Nghiên cứu biết được cách sử dụng module Sim 900A, nguyên lý hoạt động, các thông số kỹ thuật, tính năng của Sim 808A. Biết được cách thiết lập cho module sim, sử dụng tính năng nhắn tin SMS đến điện thoại và gửi dữ liệu GPS trên module Sim để điều khiển thiết bị.

Biết cách sử dụng thêm phần mềm vẽ mạch Altium để thiết kế mạch in, làm mạch kết nối giữa kit STM32 với các module mở rộng để giảm sử dụng các dây cắm và các linh kiện gắn rời nhằm tăng tính nhỏ gọn cho mạch điều khiển.

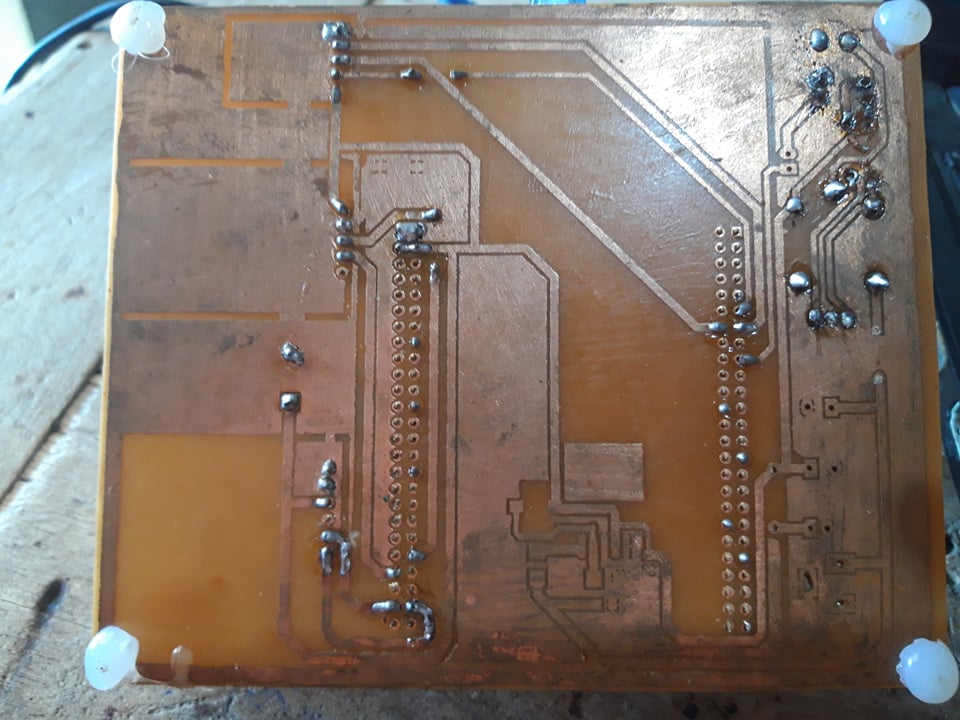
\***Một số hình ảnh thực tế:**

****

Hình 5. 1: Mặt trên 1 của sản phẩm

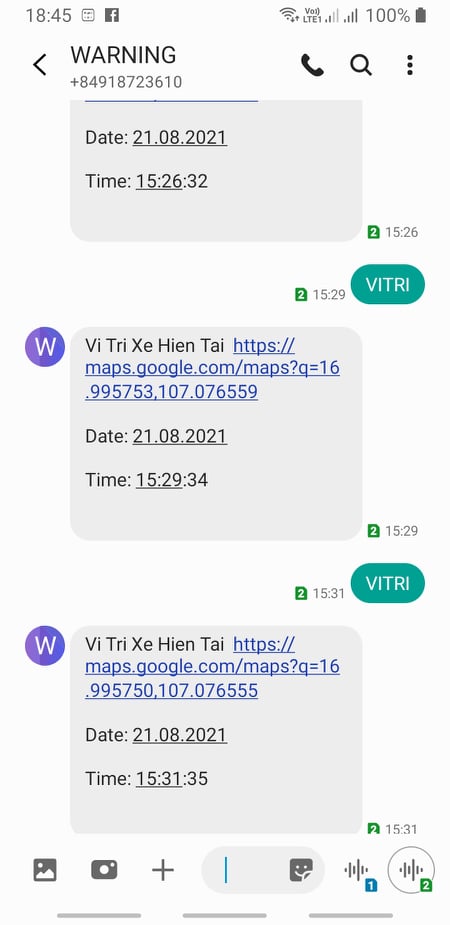
****

Hình 5. 2: Mặt trên 2 của sản phẩm

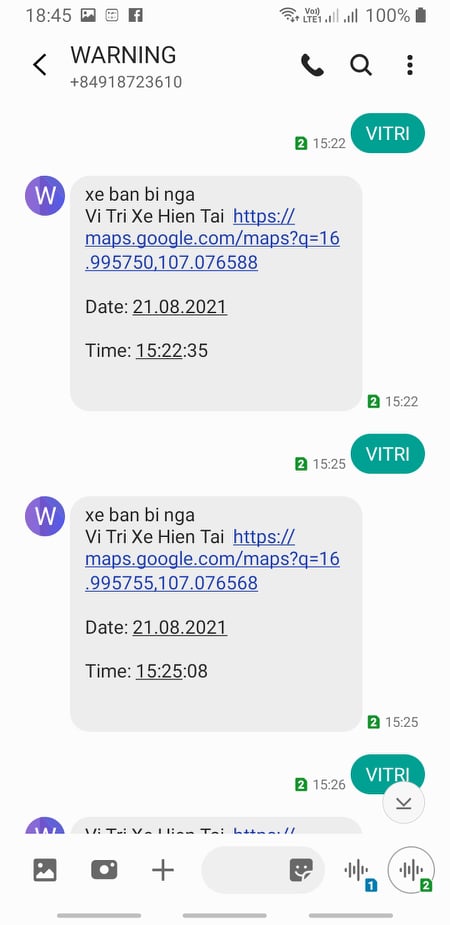
****

Hình 5. 3: Mặt dưới sản phẩm

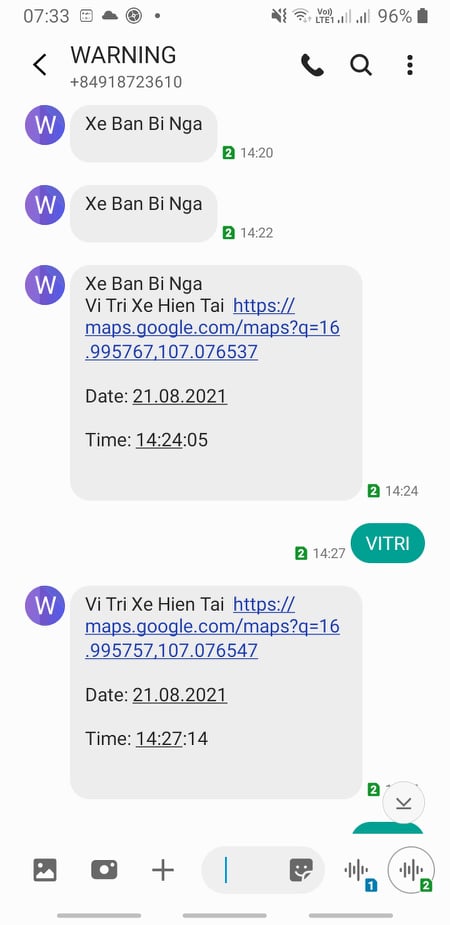
Khi nhận được tín hiệu GPS, điện thoại sẽ nhận được tin nhắn như hình.

****

Hình 5. 4: Kết quả trả về 1

****

Hình 5. 5: Kết quả trả về 2

****

Hình 5. 6: Kết quả trả về 3

## 5.2 Nhận xét – đánh giá:

Sau quá trình vận hành thử hệ thống, đã thu về những số liệu như bảng sau:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Số lần thử nghiệm | Số lần thành công | Hiệu suất (%) | Đánh giá | Ghi chú |
| Định vị GPS | 50 | 40 | 80% | Đạt |  |
| Hoạt động cảm biến gia tốc | 60 | 60 | 100% | Đạt |  |
| Hệ thống cảnh báo | 0 (do thiếu thiết bị) | 0 | 0 | Không đạt |  |
| Tin nhắn báo về điện thoại | 40 | 30 | 75 % | Đạt |  |

Bảng 5. 1: Bảng đánh giá sản phẩm

Đánh giá chung: Do tình hình covid vì vậy một số dụng cụ thiết bị không có nên hoàn thành được khoảng 75% so với quy định

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN:

## Kết luận:

Sau thời gian tìm hiểu, nghiên cứu và thực hiện mô hình, nhiệm vụ đề tài cơ bản được hoàn thành. Bằng sự cố gắng của bản thân, học hỏi từ Thầy Cô, bạn bè, Tôi đã hoàn thành đề tài “HỆ THỐNG GIÁM SÁT VÀ BÁO CÁO TAI NẠN PHƯƠNG TIỆN”.

Đề tài đã hoàn thành cụ thể:

* Kết quả thi công hoàn thiện 75%
* Hệ thống chạy ổn định, hoàn thành 75% mục tiêu ban đầu.
* Độ phản hồi của hệ thống cảnh báo nhanh.
* Tương tác giữa điện thoại và module sim ổn định

Quá trình thực hiện đề tài đã đạt được những kết quả như mục tiêu ban đầu đề ra, tuy nhiên còn gặp nhiều khó khăn và hạn chế. Tín hiệu GPS vẫn chưa được định vị ổn định và chính xác được.

## Hướng phát triển:

Từ đề tài “HỆ THỐNG GIÁM SÁT VÀ BÁO CÁO TAI NẠN PHƯƠNG TIỆN” ta có thể phát triển theo các hướng nhằm đảm bảo sự ổn định, độ bền và tính thực tế cho sản phẩm sau:

* Có thể thêm hệ thống chống trộm cho xe
* Phát triển lên ngoài việc gửi qua SMS thì có thể gửi lên và điều khiển bằng Interet
* Vị trí GPS có thể theo dõi trực tiếp chứ không phải chỉ gửi được 1 lần với độ chính xác, ổn định hơn
* Mức độ tai nạn xe sẽ được phân rõ ra các cấp.

# TÀI LIỆU THAM KHẢO:

[1] Slide môn học Kỹ thuật Vi Điều Khiển – Huỳnh Việt Thắng

[2] Ahmed El-Rabbany, “Introduction to GPS the Global Positioning System”, Artech House, Inc. Publication

[3] Wikipedia, Hệ thống định vị toàn cầu, <https://vi.wikipedia.org/>

[4] <https://www.st.com/content/st_com/en.html>

[5] <https://tapit.vn/huong-dan-su-dung-mot-chuc-nang-cua-module-sim900a/>

[6] <https://tapit.vn/tong-hop-cac-bai-huong-dan-lap-trinh-vi-dieu-khien-stm32/>

[7] Tài liệu tham khảo về datasheet linh kiện, <https://www.alldatasheet.com/>

[8] Đọc ADC <https://www.youtube.com/watch?v=tQXsUKgzSR8>

# PHỤ LỤC:

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

**#include** "main.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

**#include** "gps\_neo6.h"

**#include** "stdio.h"

**#include** "string.h"

**#define** SCALE\_MAX 90

**#define** SCALE\_MIN -90

**#define** X\_MAX 2360

**#define** Y\_MAX 2355

**#define** Z\_MAX 2380

**#define** X\_MIN 1554

**#define** Y\_MIN 1549

**#define** Z\_MIN 1566

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

ADC\_HandleTypeDef hadc1;

DMA\_HandleTypeDef hdma\_adc1;

UART\_HandleTypeDef huart1;

UART\_HandleTypeDef huart6;

/\* USER CODE BEGIN PV \*/

/\* Private variables ---------------------------------------------------------\*/

uint32\_t data[3];

**char** Message1[160] = {'\0'};

**char** Message2[160] = {'\0'};

**char** Message3[160] = {'\0'};

**char** Message4[160] = {'\0'};

int32\_t x\_angle,y\_angle,z\_angle;

**#define** SIZEARRAY 100

**char** enter[2]={0x1A};

**char** Rx\_Buffer[SIZEARRAY];

**char** Rx\_Data[SIZEARRAY];

**int** flag1 = 0, flag2 = 0;

**int** UART6\_count;

**int** UART6\_time;

**int** Status = 0;

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

**void** **SystemClock\_Config**(**void**);

**static** **void** **MX\_GPIO\_Init**(**void**);

**static** **void** **MX\_DMA\_Init**(**void**);

**static** **void** **MX\_ADC1\_Init**(**void**);

**static** **void** **MX\_USART1\_UART\_Init**(**void**);

**static** **void** **MX\_USART6\_UART\_Init**(**void**);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

**void** **HAL\_ADC\_ConvCpltCallback**(ADC\_HandleTypeDef\* hadc){

x\_angle = (int32\_t)((**float**)((SCALE\_MAX-SCALE\_MIN)\*(data[0]-X\_MIN))/(X\_MAX-X\_MIN))+SCALE\_MIN;

y\_angle = (int32\_t)((**float**)((SCALE\_MAX-SCALE\_MIN)\*(data[1]-Y\_MIN))/(Y\_MAX-Y\_MIN))+SCALE\_MIN;

z\_angle = (int32\_t)((**float**)(SCALE\_MAX-SCALE\_MIN)\*(data[2]-Z\_MIN)/(Z\_MAX-Z\_MIN))+SCALE\_MIN;

}

**void** **delBuf**(**char**\* data)

{

**int** len = **strlen**(data);

**for**( i = 0; i < len; i++)

{

data[i] = 0;

}

}

**void** **HAL\_UART\_RxCpltCallback**(UART\_HandleTypeDef \*huart)

{

**if** (huart -> Instance == USART1)

{

NEO6\_ReceiveUartChar(&GpsState);

}

**if** (huart->Instance == USART6)// kiem tra ngat o bo UART6

{

**if**((HAL\_GetTick() - UART6\_time > 100)||(UART6\_count == SIZEARRAY)){

UART6\_count = 0;

delBuf(Rx\_Buffer);// xoa bo dem nho

}

Rx\_Buffer[UART6\_count++]=Rx\_Data[0];// tang count len 1 va lay byte data tu UART6\_RXData

UART6\_time = HAL\_GetTick();

**if**(Rx\_Buffer[50]=='V' && Rx\_Buffer[51]=='I' && Rx\_Buffer[52]=='T' && Rx\_Buffer[53]=='R' && Rx\_Buffer[54]=='I' ){

flag1=1;

}

**if**(Rx\_Buffer[2]=='B' && Rx\_Buffer[3]=='U' && Rx\_Buffer[4]=='S' && Rx\_Buffer[5]=='Y' ){

flag2=1;

}

} HAL\_UART\_Receive\_IT(&huart6, (uint8\_t \*)Rx\_Data,1);

}

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

**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\_DMA\_Init();

MX\_ADC1\_Init();

MX\_USART1\_UART\_Init();

MX\_USART6\_UART\_Init();

/\* USER CODE BEGIN 2 \*/

HAL\_ADC\_Start\_DMA(&hadc1, (uint32\_t\*)data, 3);

NEO6\_Init(&GpsState, &huart1);

HAL\_UART\_Receive\_IT(&huart6, (uint8\_t \*)Rx\_Data,1);

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

**if** (x\_angle < -15 || x\_angle > 165 && flag1 == 0 ){

HAL\_Delay(5000);

**if** (x\_angle < -15 || x\_angle > 165 ){

Status ++;

**if** (Status == 1){

**for**(**int** j = 0; j<100000; j++){

GetData();

}

SendSms();

**while**(flag2 == 0){

Call();

}

}

}

HAL\_Delay(10000);

**if** (x\_angle > -15 || x\_angle < 165 ){

flag2 = 0;

Status = 0;

}

}

**if**( flag1 == 1){

**for**(**int** j = 0; j<100000; j++){

GetData();

}

SendSms1();

}

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

**void** **SystemClock\_Config**(**void**)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Configure the main internal regulator output voltage

\*/

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSI;

RCC\_OscInitStruct.PLL.PLLM = 8;

RCC\_OscInitStruct.PLL.PLLN = 100;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 4;

**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\_DIV1;

**if** (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_3) != *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 \*/

/\*\* Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)

\*/

hadc1.Instance = ADC1;

hadc1.Init.ClockPrescaler = ADC\_CLOCK\_SYNC\_PCLK\_DIV4;

hadc1.Init.Resolution = ADC\_RESOLUTION\_12B;

hadc1.Init.ScanConvMode = *ENABLE*;

hadc1.Init.ContinuousConvMode = *ENABLE*;

hadc1.Init.DiscontinuousConvMode = *DISABLE*;

hadc1.Init.ExternalTrigConvEdge = ADC\_EXTERNALTRIGCONVEDGE\_NONE;

hadc1.Init.ExternalTrigConv = ADC\_SOFTWARE\_START;

hadc1.Init.DataAlign = ADC\_DATAALIGN\_RIGHT;

hadc1.Init.NbrOfConversion = 3;

hadc1.Init.DMAContinuousRequests = *ENABLE*;

hadc1.Init.EOCSelection = ADC\_EOC\_SINGLE\_CONV;

**if** (HAL\_ADC\_Init(&hadc1) != *HAL\_OK*)

{

Error\_Handler();

}

/\*\* Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.

\*/

sConfig.Channel = ADC\_CHANNEL\_1;

sConfig.Rank = 1;

sConfig.SamplingTime = ADC\_SAMPLETIME\_480CYCLES;

**if** (HAL\_ADC\_ConfigChannel(&hadc1, &sConfig) != *HAL\_OK*)

{

Error\_Handler();

}

/\*\* Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.

\*/

sConfig.Channel = ADC\_CHANNEL\_2;

sConfig.Rank = 2;

**if** (HAL\_ADC\_ConfigChannel(&hadc1, &sConfig) != *HAL\_OK*)

{

Error\_Handler();

}

/\*\* Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.

\*/

sConfig.Channel = ADC\_CHANNEL\_3;

sConfig.Rank = 3;

**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 USART6 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_USART6\_UART\_Init**(**void**)

{

/\* USER CODE BEGIN USART6\_Init 0 \*/

/\* USER CODE END USART6\_Init 0 \*/

/\* USER CODE BEGIN USART6\_Init 1 \*/

/\* USER CODE END USART6\_Init 1 \*/

huart6.Instance = USART6;

huart6.Init.BaudRate = 9600;

huart6.Init.WordLength = UART\_WORDLENGTH\_8B;

huart6.Init.StopBits = UART\_STOPBITS\_1;

huart6.Init.Parity = UART\_PARITY\_NONE;

huart6.Init.Mode = UART\_MODE\_TX\_RX;

huart6.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart6.Init.OverSampling = UART\_OVERSAMPLING\_16;

**if** (HAL\_UART\_Init(&huart6) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN USART6\_Init 2 \*/

/\* USER CODE END USART6\_Init 2 \*/

}

/\*\*

\* Enable DMA controller clock

\*/

**static** **void** **MX\_DMA\_Init**(**void**)

{

/\* DMA controller clock enable \*/

\_\_HAL\_RCC\_DMA2\_CLK\_ENABLE();

/\* DMA interrupt init \*/

/\* DMA2\_Stream0\_IRQn interrupt configuration \*/

HAL\_NVIC\_SetPriority(*DMA2\_Stream0\_IRQn*, 0, 0);

HAL\_NVIC\_EnableIRQ(*DMA2\_Stream0\_IRQn*);

}

/\*\*

\* @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\_GPIOH\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin : PA7 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_7;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

}

/\* USER CODE BEGIN 4 \*/

**void** **GetData**(){

NEO6\_Task(&GpsState);

**if**(NEO6\_IsFix(&GpsState)){

**sprintf**(Message1, "Time: %02d:%02d:%02d\n\r", GpsState.Hour, GpsState.Minute, GpsState.Second);

**sprintf**(Message2, "Date: %02d.%02d.20%02d\n\r", GpsState.Day, GpsState.Month, GpsState.Year);

**sprintf**(Message3, "Vi Tri Xe Hien Tai https://maps.google.com/maps?q=%2f,%2f\n\r", GpsState.Latitude, GpsState.Longitude);

}

**else**{

**sprintf**(Message4, "No Fix\n\r");

}

}

**void** **SendSms**(){

HAL\_Delay(2000);

HAL\_UART\_Transmit(&huart6, (uint8\_t \*)"AT+CMGDA=\"DEL ALL\"\r\n", **sizeof**("AT+CMGDA=\"DEL ALL\"\r\n"),20);

HAL\_Delay(1000);

HAL\_UART\_Transmit(&huart6, (uint8\_t \*)"AT+CMGS=\"09\*\*\*\*\*\*\*\*\"\r\n", **strlen**("AT+CMGS=\"09\*\*\*\*\*\*\*\*\"\r\n"), 20);

HAL\_Delay(5000);

HAL\_UART\_Transmit(&huart6, (uint8\_t \*)"Xe Ban Bi Nga\r\n",**strlen**("Xe Ban Bi Nga\r\n"),20);

HAL\_UART\_Transmit(&huart6, (uint8\_t \*)(**char** \*)Message3,**strlen**((**char** \*)Message3),100);

HAL\_UART\_Transmit(&huart6, (uint8\_t \*)(**char** \*)Message2,**strlen**((**char** \*)Message2),20);

HAL\_UART\_Transmit(&huart6, (uint8\_t \*)(**char** \*)Message1,**strlen**((**char** \*)Message1),20);

HAL\_UART\_Transmit(&huart6, (uint8\_t \*)enter,1,10);

}

**void** **SendSms1**(){

HAL\_Delay(1000);

HAL\_UART\_Transmit(&huart6, (uint8\_t \*)"AT+CMGDA=\"DEL ALL\"\r\n", **sizeof**("AT+CMGDA=\"DEL ALL\"\r\n"), 20);

HAL\_Delay(1000);

HAL\_UART\_Transmit(&huart6, (uint8\_t \*)"AT+CMGS=\"09\*\*\*\*\*\*\*\*\"\r\n",**strlen**("AT+CMGS=\"09\*\*\*\*\*\*\*\*\"\r\n"), 20);

HAL\_Delay(1000);

HAL\_UART\_Transmit(&huart6, (uint8\_t \*)(**char** \*)Message3,**strlen**((**char** \*)Message3),100);

HAL\_UART\_Transmit(&huart6, (uint8\_t \*)(**char** \*)Message2,**strlen**((**char** \*)Message2),20);

HAL\_UART\_Transmit(&huart6, (uint8\_t \*)(**char** \*)Message1,**strlen**((**char** \*)Message1),20);

HAL\_UART\_Transmit(&huart6, (uint8\_t \*)enter,1,10);

flag1=0;

}

**void** **Call**(){

HAL\_UART\_Transmit(&huart6, (uint8\_t \*)"ATD09\*\*\*\*\*\*\*\*;\r\n",**strlen**("ATD094\*\*\*\*\*\*\*\*;\r\n"), 20);

HAL\_Delay(20000);

}

/\* 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 \*/

**#ifdef** USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

**void** assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

**#endif** /\* USE\_FULL\_ASSERT \*/