

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KỸ THUẬT GIAO THÔNG
BỘ MÔN KỸ THUẬT Ô TÔ VÀ MÁY ĐỘNG LỰC

_____o-0-o_____



Project

**PHÁT TRIỂN PHƯƠNG PHÁP CHUẨN ĐOÁN
QUA GIAO TIẾP OBD II TRÊN CHIẾC
MITSHUBISHI XPANDER 2019 VÀ ỨNG DỤNG
IOT ĐỂ HIỂN THỊ DỮ LIỆU CHUẨN ĐOÁN**

GIẢNG VIÊN HƯỚNG DẪN: ThS. Phạm Trần Đăng Quang

DANH SÁCH NHÓM		
STT	TÊN	MSSV
1	Nguyễn Nhật Duy	1910088
2	Trịnh Quang Khải	1910257

TP.HỒ CHÍ MINH, THÁNG 3 NĂM 2023

This image shows a full page of white paper with horizontal dotted lines. The lines are evenly spaced and run across the width of the page, providing a guide for handwriting practice. There are no margins, text, or other markings on the page.

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin chân thành cảm ơn Thầy Phạm Trần Đăng Quang đã tận tâm hướng dẫn chúng em thực hiện đề tài này. Nếu không có những lời chỉ bảo và phản hồi tỉ mỉ trong suốt quá trình thực hiện thì đề tài sẽ rất khó để có thể hoàn thiện được. Bước đầu đi vào thực tế, tìm hiểu về lĩnh vực nghiên cứu về hệ thống CAN BUS trên ô tô, kiến thức của chúng em còn hạn chế và còn nhiều bỡ ngỡ. Chính nhờ sự nhiệt tình và những tài liệu bổ ích mà Thầy cung cấp đã giúp chúng em hình dung ra con đường cần đi để thực hiện đề tài.

Và nhóm cũng xin gửi lời cảm ơn đến Thầy Phạm Trần Đăng Quang đã hỗ trợ cung cấp thiết bị, tài liệu và những kiến thức chuyên sâu, tạo điều kiện tốt nhất để chúng em có thể hoàn thành tốt project này.

Dù đã thật sự cố gắng nhưng do điều kiện thời gian cũng như kinh nghiệm, kiến thức còn hạn chế nên không tránh khỏi những sai sót trong quá trình thực hiện, chúng em mong nhận được sự góp ý từ phía thầy cô, từ bạn bè cũng như những cá nhân có đọc được bản đồ án này với nhu cầu tham khảo. Chúng em xin cảm ơn quý thầy đã dành thời gian ra đọc, chỉnh sửa và đóng góp ý kiến để đề tài được hoàn chỉnh, làm nền tảng vững chắc cho việc thực hiện các hướng phát triển tiếp theo. Em xin cảm ơn quý Thầy Cô!

Mục lục

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN.....	2
LỜI CẢM ƠN	3
Mục lục	4
Chương 1: Giới thiệu.....	6
1.1. Mở đầu.....	6
1.2. Giới thiệu đề tài	6
1.3. Mục tiêu đề tài	7
1.4 . Giới hạn đề tài	7
1.5. Bố cục đề tài	7
Chương 2: Cơ sở lý thuyết	8
2.1. Hệ thống tín hiệu trên ô tô	8
2.1.1. Hệ thống đo tốc độ động cơ và tốc độ xe	8
2.1.2. Hệ thống đo tín hiệu ắc quy.....	10
2.1.3. Giới thiệu OBD-2.....	10
2.1.4. Giới thiệu về CAN.....	11
2.2. Trình bày phương án và chọn phương án.....	15
2.2.1 Khởi nhận tín hiệu.....	15
2.2.2. Khởi đọc và xử lý tín hiệu	21
2.2.3. Khởi xuất kết quả.....	23
Chương 3: Thiết kế bố trí chung.....	27
3.1. Bố trí chung hệ thống	27
3.2. Nguyên lý hoạt động hệ thống	28
Chương 4: Thiết kế kỹ thuật	30
4.1. Thiết kế phần mềm	30
4.2. Thiết kế giao diện hiển thị.....	32

4.2.1. Lưu đồ giải thuật	32
4.2.2. Lập trình Arduino	33
4.2.3. Tạo giao diện hiển thị trên Freeboard	37
Chương 5: Kết luận	39
5.1. Kết quả đạt được	39
5.1.1. Tín hiệu đọc được và xử lý	39
5.1.2. Dữ liệu trên Cloud Server và giao diện hiển thị trên Cloud dashboard ..	42
5.2. Đánh giá	42
5.3. Hướng phát triển	43
Tài liệu tham khảo	43

Chương 1: Giới thiệu

1.1. Mở đầu

Có thể xem ngành công nghiệp ô tô đang là thước đo cho sự phát triển kinh tế của đất nước. Nếu như chỉ vài năm trước đây, ngành công nghiệp này được xem là “xa xỉ” ở nước ta thì giờ đây, cùng với sự phát triển nhanh chóng của đất nước, ngành công nghiệp ô tô đang bùng lên mạnh mẽ.

Việc sử dụng ô tô làm phương tiện di chuyển chính đang dần trở nên phổ biến hơn, nhất là ở các thành phố lớn và khu đông dân cư. Với sự phát triển ngày càng hiện đại thì đòi hỏi cần một hệ thống giám sát các thông số, tình trạng xe đang hoạt động hỗ trợ người lái trong quá trình điều khiển hoặc cho các doanh nghiệp có thể giám sát thời gian thực với các đội xe từ xa. Từ những ý tưởng thảo luận và sự hướng dẫn của giảng viên hướng dẫn, nhóm đã hình thành một ý tưởng đề tài thực tế là “Phát triển phương pháp chuẩn đoán qua giao tiếp OBD II trên chiếc Mitshubishi Xpander 2019 và ứng dụng hiển IOT để hiển thị dữ liệu chuẩn đoán”.

Để hoàn thành dự án với đề tài “Phát triển phương pháp chuẩn đoán qua giao tiếp OBD II trên chiếc Mitshubishi Xpander 2019 và ứng dụng hiển IOT để hiển thị dữ liệu chuẩn đoán”, nhóm em đã vận dụng các kiến thức được học trong 4 năm tại trường cũng như tham khảo thêm nhiều tài liệu từ bạn bè, từ đàn anh đi trước, từ các tài liệu nước ngoài, các nghiên cứu trong nước trên báo đài, internet. Đi kèm với đó không thể không kể đến sự hướng dẫn tận tình của thầy Phạm Trần Đăng Quang - là giảng viên bộ môn Kỹ thuật ô tô. Dù đã thật sự cố gắng nhưng do điều kiện thời gian cũng như kinh nghiệm, kiến thức còn hạn chế nên không tránh khỏi những sai sót trong quá trình thực hiện, nhóm em mong nhận được sự góp ý từ phía thầy phản biện, từ bạn bè cũng như những cá nhân có đọc được bản đồ án này với nhu cầu tham khảo. Nhóm em xin cảm ơn quý thầy đã dành thời gian ra đọc, chỉnh sửa và đóng góp ý kiến để đề tài được hoàn chỉnh, làm nền tảng vững chắc cho việc thực hiện các hướng phát triển tiếp theo.

1.2. Giới thiệu đề tài

Với sự phát triển của công nghệ, hệ thống giám sát và cảnh báo của các dòng xe ngày càng hiện đại. Từ cảm biến tốc độ xe, cảm biến áp suất, cảm biến nhiệt độ nước làm mát, ... thông qua quá trình xử lý tín hiệu, có thể thông báo cho người dùng cho

người dùng các thông số trên xe và thu thập các dữ liệu thói quen người dùng. Chính vì vậy, có thể sử dụng máy học để cung cấp dữ liệu vào mô hình đào tạo nhằm dự đoán chi phí và thậm chí phân tích các đặc điểm của người lái xe, giúp hỗ trợ và phát triển những tính năng cần thiết cho người lái.

1.3. Mục tiêu đề tài

- **Ý tưởng:** nghiên cứu bộ sản phẩm kết nối thu thập dữ liệu từ xe gồm tốc độ động cơ, tốc độ xe, điện áp acqui, nhiệt độ nước làm mát, nhiệt độ khí nạp, vị trí bướm ga, vị trí bàn đạp ga, tín hiệu cảm biến oxy và ứng dụng IOT (Internet Of Thing) có thể truyền, lưu trữ, hiển thị giao diện người dùng trên cloud.

- **Mục tiêu** Phát triển phương pháp chuẩn đoán qua giao tiếp OBD II trên chiếc Mitsubishi Xpander 2019 và ứng dụng hiển IOT để hiển thị dữ liệu chuẩn đoán.

- **Nội dung:**

- + Chọn và lắp đặt phần cứng có nhiệm vụ đọc dữ liệu thông qua cổng OBDII.
- + Thiết kế giải thuật đọc và xử lý tín hiệu.
- + Thiết kế giải thuật truyền dữ liệu từ vi điều khiển lên cloud qua wifi.
- + Lưu trữ và xây dựng giao diện hiển thị trên cloud.

1.4. Giới hạn đề tài

Đề tài dừng lại ở việc:

- + Xây dựng mô hình mạch thu thập và hiển thị dữ liệu.
- + Thử nghiệm mô hình và đánh giá độ chính xác.
- + Xây dựng giao diện thu thập dữ liệu.
- + Đề tài được thử nghiệm trên xe Xpander 2019.
- + Dụng cụ nghiên cứu nằm trong giới hạn tài chính sinh viên.
- + Những tín hiệu có thể đọc được và xử lý phụ thuộc vào loại xe thực nghiệm.

1.5. Bố cục đề tài

Bố cục dự án bao gồm 5 phần:

Phần 1: Giới thiệu đề tài

Phần 2: Cơ sở lý thuyết

Phần 3: Thiết kế bố trí chung

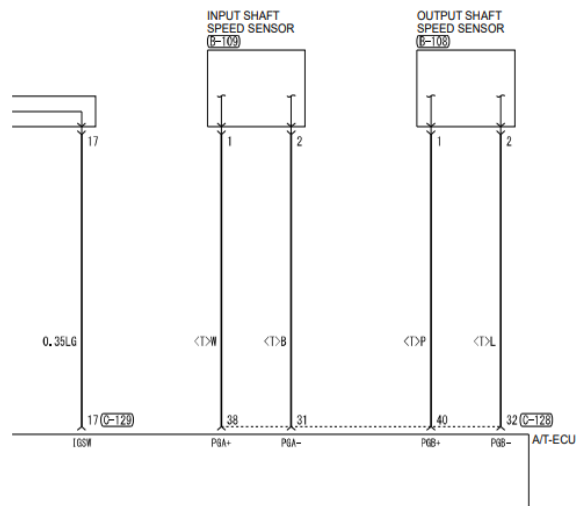
Phần 4: Thiết kế kỹ thuật

Phần 5: Kết luận

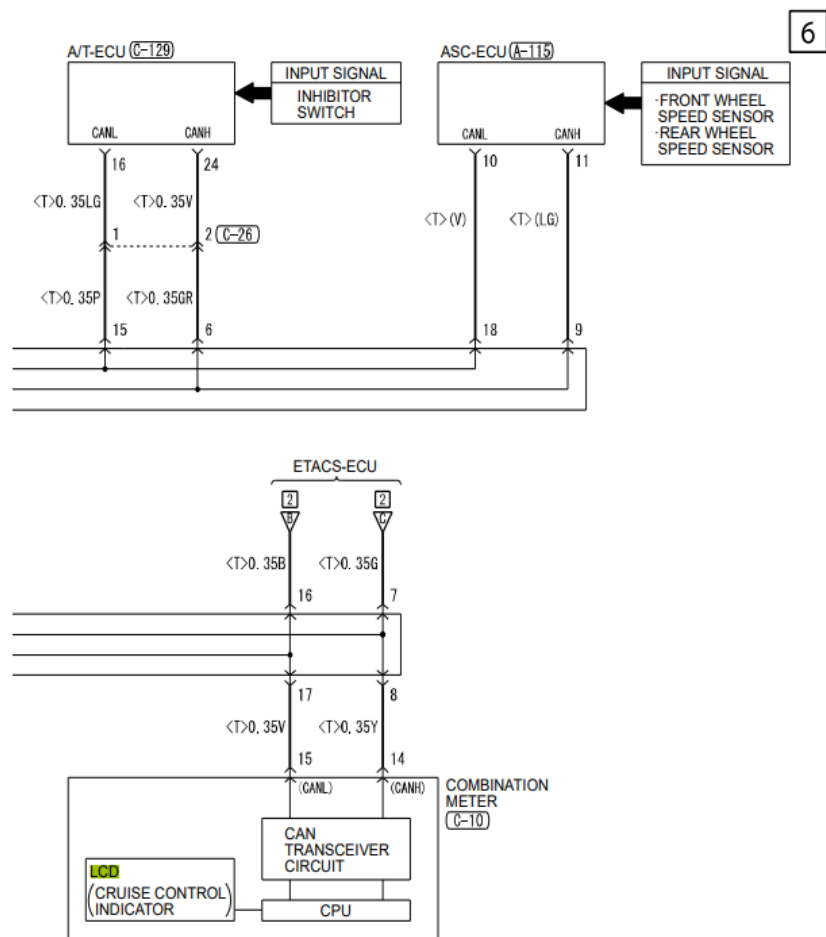
Chương 2: Cơ sở lý thuyết

2.1. Hệ thống tín hiệu trên ô tô

2.1.1. Hệ thống đo tốc độ động cơ và tốc độ xe



Sơ đồ mạch điện tín hiệu tốc độ động cơ Mitsubishi Xpander 2019

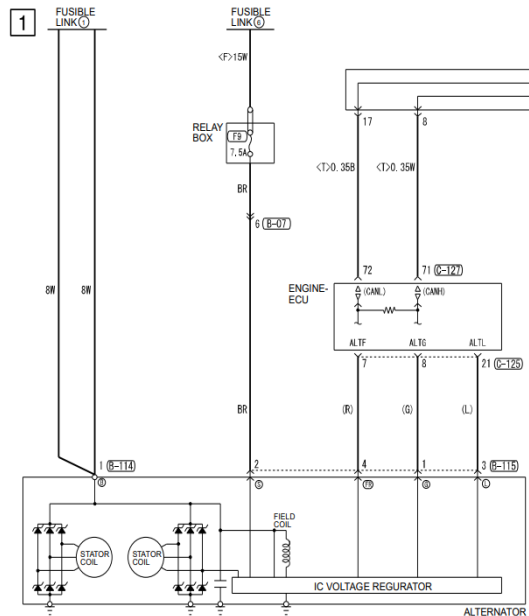


Sơ đồ mạch điện tín hiệu tốc độ bánh xe Mitsubishi Xpander 2019



Đồng hồ trên taplo Mitsubishi Xpander 2019

2.1.2. Hệ thống đo tín hiệu ắc quy



Sơ đồ mạch điện tín hiệu ắc quy Mitsubishi Xpander 2019

2.1.3. Giới thiệu OBD-2

❖ OBD-2

OBD -2 là một sự cải tiến của OBD -1 với các chức năng đa dạng hơn. Các tiêu chuẩn của O B D -2 quy định các loại giắc nối và vị trí các chân, cũng n ư quy định giao thức của tín hiệu và định dạng tin thông báo. Có một chân trong giắc chẩn đoán cấp nguồn cho máy chẩn đoán từ bìn h ắc quy của xe. Điều n ày giúp hạn chế việc cấp nguồn bên ngoài cho m áy chẩn đoán. O B D -2 sử dụng nhiều giao thức khác nhau. Một số giao thứ c cơ bản như: J1850- VPW , J1850-PWM , ISO 9141, KWP 2000, ISO 15765.

❖ Mô hình OSI

- Mô tả tiêu chuẩn chung các chức năng truyền thông. Mô hình OSI sắp xếp một giao thức với chức năng của nó vào một hoặc một nhóm các lớp tương ứng. Mỗi một tầng cấp có một đặc tính là nó chỉ sử dụng chức năng của tầng dưới nó, đồng thời chỉ cho phép tầng trên sử dụng các chức năng của mình, được chia thành 7 lớp:

- + Lớp ứng dụng.
- + Lớp trình bày.
- + Lớp phiên.
- + Lớp vận chuyển.

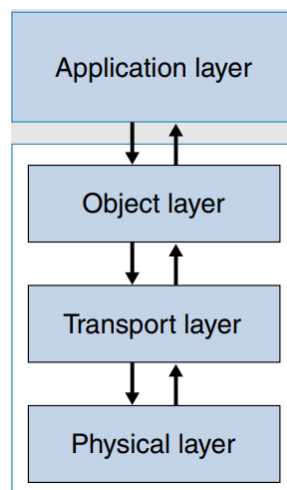
- + Lớp mạng.
- + Lớp liên kết dữ liệu.
- + Lớp vật lý.

2.1.4. Giới thiệu về CAN

❖ Chức năng

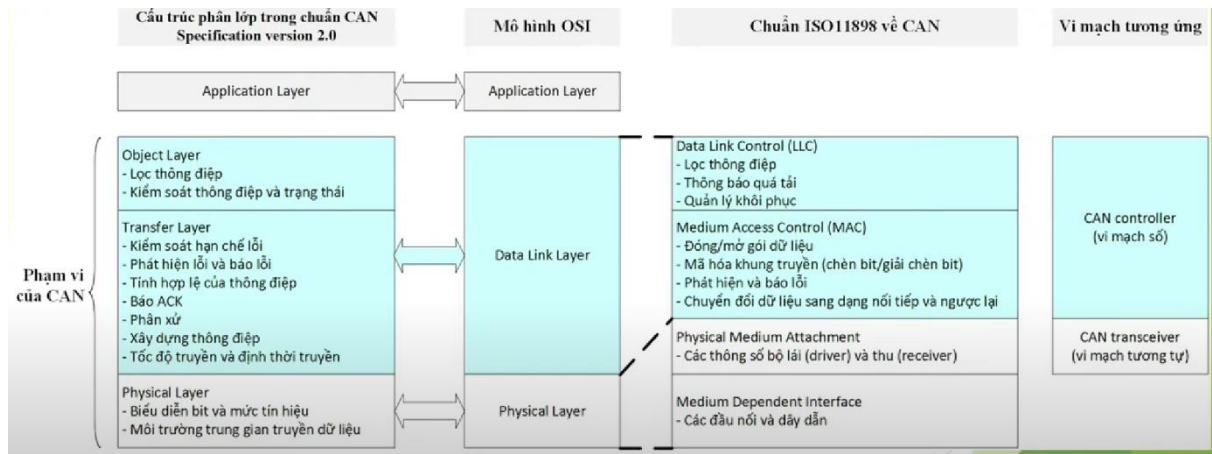
- CAN là một trong những giao thức truyền bus được dùng nhiều nhất.
 - Cơ chế phân xử không phá hủy: không bị mất dữ liệu, khi hộp ECU dùng chung dữ liệu.
 - Bus truy cập vào khung với mức độ ưu tiên cao nhất mà không có bị chậm trễ.
 - Cơ chế phát hiện lỗi:
- + Ngắt kết nối các nút bus bị lỗi để duy trì liên lạc giữa các nút còn lại.
- + Các khung được truyền không được xác định bằng địa chỉ nút của máy phát khung hoặc của máy thu khung mà bởi nội dung của chúng.
- + Định danh đại diện cho tải trọng của khung cũng có chức năng chỉ định mức ưu tiên của khung.

❖ CAN protocol



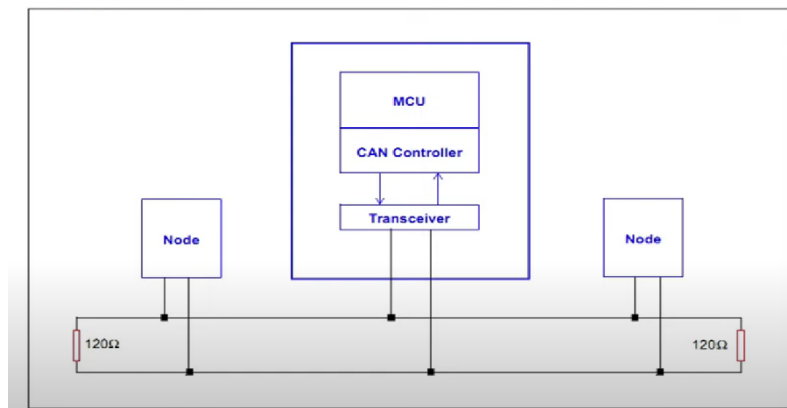
Can protocol layers

- Can ứng dụng 3 tầng: Tầng Application physical, Data link
- Bên gửi: tầng 7 nhận dữ liệu tầng 2 chia nhỏ dữ liệu thành từng frame. Tầng 1 mã hóa thành các chuỗi nhị phân.
- Bên nhận: Tầng 1 nhận chuỗi nhị phân vào vùng đệm. Tầng 2 kiểm tra các trạm lỗi, kiểm tra địa chỉ datalink, tầng 7 gửi data đến ECU.



Mô hình tổng quát về CAN BUS

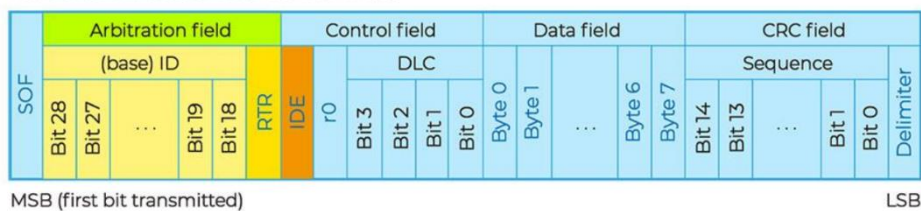
- ❖ **Cấu tạo:** Transceiver (chuyển đổi tín hiệu TTL sang tín hiệu vi sai), CAN controller (thực hiện các chức năng giao thức CAN: phân quyền ưu tiên, xử lý...), MCU (mạch điều khiển).



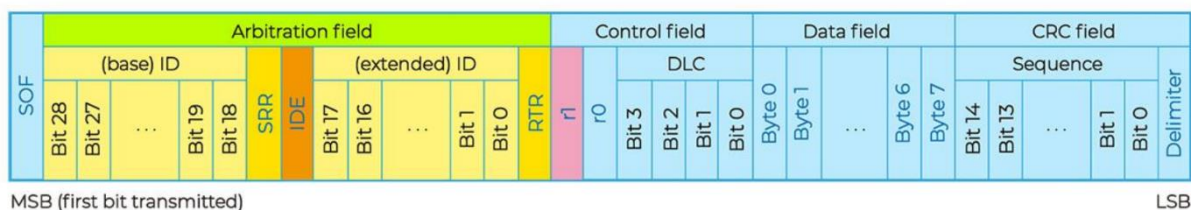
Cấu tạo hệ thống CAN BUS

- ❖ **Nguyên lý hoạt động:** tín hiệu vi xử lý là nhị phân chuyển thành tín hiệu điện áp trên 2 đường dây CAN-high, CAN-low, rồi tín hiệu điện áp được đưa vào bộ nhận. Tín hiệu chuyển hóa sau cùng giống như lúc gửi.
- ❖ **CAN BUS data frame format**

Base CAN data frame format



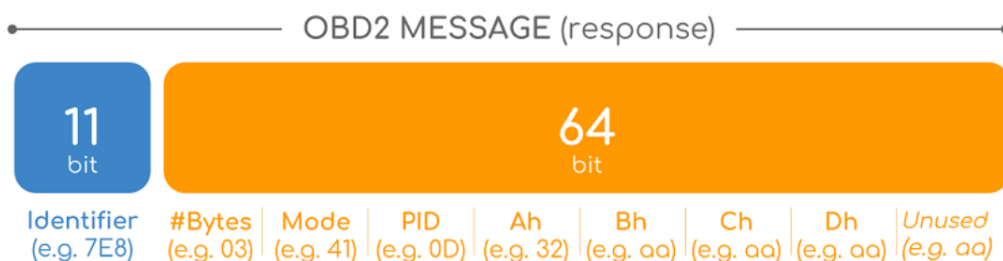
Extended CAN data frame format



- Gồm 2 loại CAN Bus chính: CAN 2.0A (Base format), CAN 2.0B (Extended format). CAN có dữ liệu đóng khung gọi là data frame và được truyền trên mạng theo dạng từng khung dữ liệu rời rạc.

- CAN data frame format gồm 7 trường: trường bắt đầu khung (Start Of Frame Field – SOF), trường xác định quyền ưu tiên (Arbitration Field), trường xác định quyền ưu tiên (Arbitration Field), Trường dữ liệu (Data Field), Trường kiểm tra (Cyclic Redundancy Check Field – CRC)

*Cấu trúc OBD II Message được sử dụng trong đề tài



Cấu trúc đoạn mã OBDII

- CAN ID:

+ Định danh: Đối với thông báo OBD2, định danh là chuẩn 11-bit và được sử dụng để phân biệt giữa "thông báo yêu cầu" (ID 7DF) và "thông báo phản hồi" (ID 7E8 đến 7EF).

- CAN DATA:

->Trường này có độ dài từ 0 đến 8 byte tùy vào giá trị của DLC ở trường điều khiển.

- + Byte 1 - Độ dài: Điều này chỉ phản ánh độ dài theo số byte của dữ liệu còn lại (03 đến 06). Đối với ví dụ về Tốc độ Xe, nó là 02 cho yêu cầu (vì chỉ có 01 và 0D theo sau), trong khi đối với phản hồi, nó là 03 vì cả 41, 0D và 32 đều tuân theo.
- + Byte 2 - Chế độ: Đối với các yêu cầu, giá trị này sẽ nằm trong khoảng từ 01-0A. Đối với các câu trả lời, số 0 được thay thế bằng 4 (tức là 41, 42 - 4A).
- + Byte 3 - PID: Đối với mỗi chế độ, tồn tại một danh sách các PID OBD2 tiêu chuẩn ví dụ: trong Chế độ 01, PID 0C là Tốc độ động cơ.
- Ah, Bh, Ch, Dh: Đây là 4 byte dữ liệu có giá trị HEX, cần được chuyển đổi sang dạng thập phân DEC trước khi chúng được sử dụng trong tính toán công thức PID. Byte dữ liệu cuối cùng (sau Dh) không được sử dụng.

Nguyên lý hoạt động:

- PID request:

- + Tin nhắn luôn có độ dài 8 byte cho dù yêu cầu ít thông tin hơn. Byte đầu tiên chỉ định độ dài bên trong trọng tải (trong trường hợp này byte đầu tiên là 2 vì chỉ byte mode và PID được sử dụng).
- + Yêu cầu thông báo PID (byte): 0x02 <mode> <pid> 0x00 0x00 0x00 0x00 0x00
- + Ví dụ: để yêu cầu chế độ 0x01 PID 0x0C (RPM) chỉ cần gửi:
0x02 0x01 0x0C 0x00 0x00 0x00 0x00 0x00
- + ID thông báo: **0x7DF** cho địa chỉ tiêu chuẩn (11 bit) và **0x18DB33F1** cho địa chỉ mở rộng (29 bit).

- PID respond:

- + ID thông báo **0x7E8** (định địa chỉ tiêu chuẩn) hoặc **0x18DAF111** (mở rộng).

Ví dụ: 0x04 0x41 0x0C 0x31 0x64 0x00 0x00 0x00

- + Độ dài 8 bit. Byte đầu tiên chỉ định độ dài bên trong trọng tải 03 (bởi vì lúc này thông tin bao gồm 04 dữ liệu: 1 byte mod, 1 byte địa chỉ và 2 byte dữ liệu phản hồi):
- + Byte Mode: - nó là phản hồi đối với chế độ 0x01 PID (request:01, respond: 41 (4->1)).
- + Byte PID: chứa địa chỉ các biến cần lấy thông tin.

Ví dụ: RPM thì địa chỉ lấy thông tin là 0C, speed địa chỉ lấy thông tin là 0D.

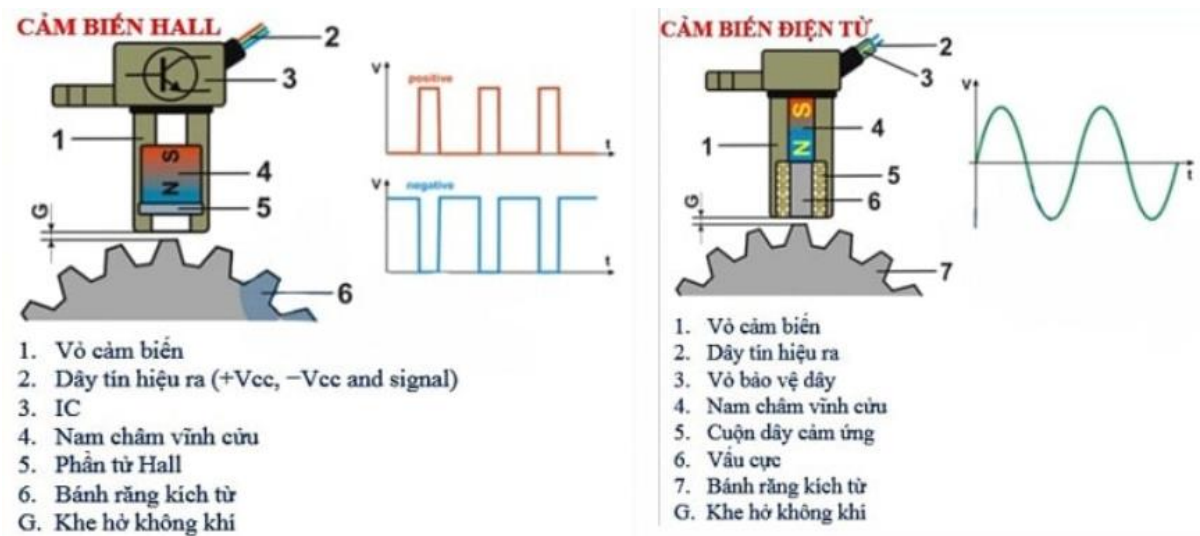
+ Byte dữ liệu: gồm 2 byte 0x31 0x64

2.2. Trình bày phương án và chọn phương án

2.2..1 Khởi nhận tín hiệu

a) Cảm biến tốc độ trực khuỷu

Chức năng: dùng để báo tốc độ động cơ để tính toán hoặc tìm góc đánh lửa tối ưu và lượng nhiên liệu sẽ phun cho từng xy lanh. Cảm biến này cũng được dùng vào mục đích điều khiển tốc độ không tải hoặc cắt nhiên liệu ở chế độ không tải cưỡng bức. Có 3 loại: cảm biến điện từ, cảm biến hall và loại cảm biến quang.

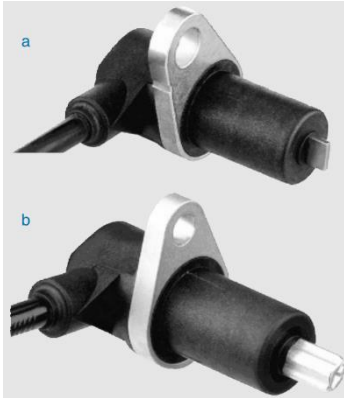


- Đối với cảm biến trực khuỷu loại điện từ sẽ có xung hình sin.
- Đối với cảm biến trực khuỷu loại Hall và Quang sẽ cho xung vuông.

b) Cảm biến tốc độ bánh xe

Chức năng: nhận biết tốc độ xe đang chạy sau đó gửi tín hiệu về ECU để điều khiển tốc độ không tải và tỉ lệ hòa khí phù hợp khi tăng tốc hoặc khi giảm tốc.

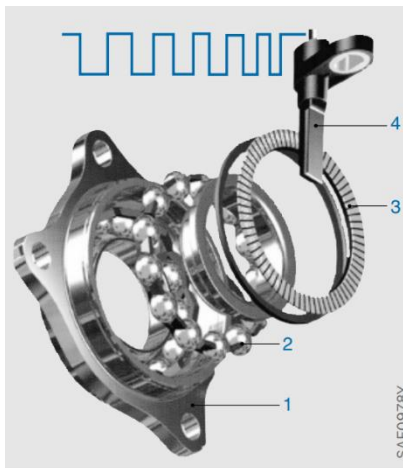
Có 2 loại chính: Cảm biến tốc độ bánh xe kiểu thụ động (cảm ứng), Cảm biến tốc độ bánh xe kiểu chủ động.



Cảm biến tốc độ bánh xe kiểu thụ động



Cảm biến tốc độ bánh xe kiểu chủ động



1. Ổ trục bánh xe.
2. Ổ bi.
3. Vòng đa cực.
4. Cảm biến tốc độ xe.

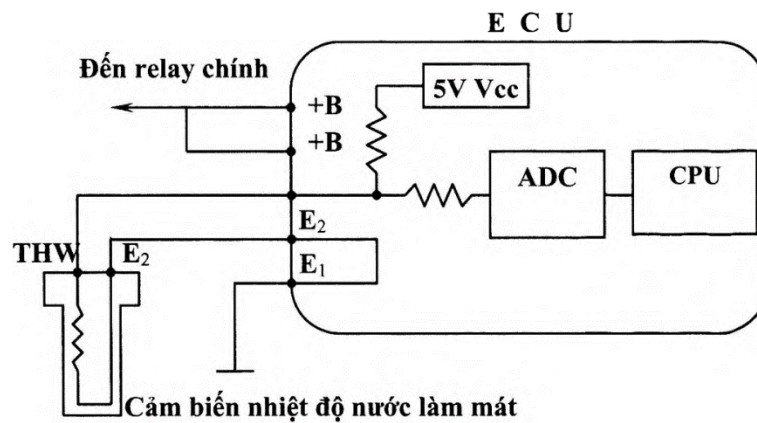
- Cảm biến tốc độ bánh xe kiểu thụ động sẽ có xung hình sin.
- Cảm biến tốc độ bánh xe kiểu chủ động sẽ cho xung vuông.

c) Cảm biến nhiệt độ nước làm mát

Chức năng: Cảm biến nhiệt độ nước làm mát sử dụng để đo nhiệt độ nước làm mát của động cơ và gửi tín hiệu về ECU để ECU điều khiển lượng xăng phun, góc đánh lửa sớm, tốc độ không tải, và cả quạt làm mát két nước.



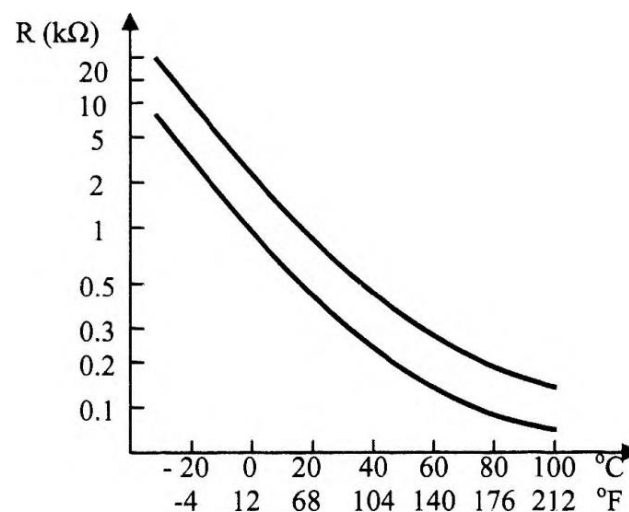
Mạch điện



Mạch điện cảm biến nhiệt độ nước làm mát

Nguyên lý: Khi nhiệt độ động cơ thấp, giá trị điện trở cảm biến cao và điện áp gửi đến bộ biến đổi ADC lớn. Tín hiệu điện áp được chuyển đổi thành một dãy xung vuông và được giải mã nhờ bộ vi xử lý để thông báo cho ECU biết động cơ đang lạnh. Khi động cơ nóng, giá trị điện trở cảm biến giảm kéo theo điện áp đặt giảm, báo cho ECU biết là động cơ đang nóng.

Đường đặc tính:



d) Tín hiệu điện áp máy phát/acqui

Ắc quy khởi động còn cung cấp điện cho các tải điện quan trọng khác trong hệ thống điện, cung cấp từng phần hoặc toàn bộ trong trường hợp động cơ chưa làm việc hoặc đã làm việc mà máy phát điện chưa phát đủ công suất (động cơ đang làm việc ở chế độ số vòng quay thấp): cung cấp điện cho đèn đậu (parking lights), radio cassette, CD, các bộ nhớ (đồng hồ, hộp điều khiển...), hệ thống báo động... Ngoài ra, ắc quy còn đóng

vai trò bộ lọc và ổn định điện thế trong hệ thống điện ô tô khi điện áp máy phát dao động.

Điện áp cung cấp của ắc quy là 6V, 12V hoặc 24V. Điện áp ắc quy thường là 12V đối với xe du lịch hoặc 24V cho xe tải.

e) Cảm biến nhiệt độ khí nạp

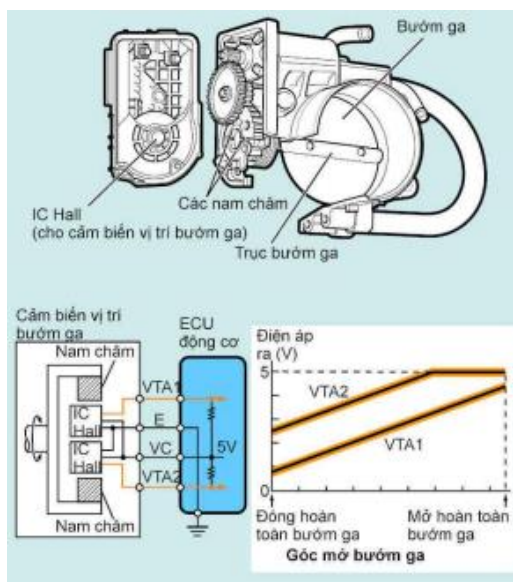
Cảm biến nhiệt độ khí nạp dùng để xác định nhiệt độ khí nạp. Cũng giống như cảm biến nhiệt độ nước, nó gồm có một điện trở được gắn trong bộ khí nạp hoặc trên đường ống nạp. Tỷ trọng của không khí thay đổi theo nhiệt độ. Nếu nhiệt độ không khí cao, hàm lượng oxy trong không khí thấp. Khi nhiệt độ không khí thấp, hàm lượng oxy trong không khí tăng.

f) Cảm biến vị trí bướm ga

** Cảm biến vị trí bướm ga loại Hall:*

Chức năng: Cảm biến vị trí bướm ga được lắp ở trên trục bướm ga. Cảm biến này chuyển vị trí góc mở cánh bướm ga thành tín hiệu điện thế gửi đến ECU.

Cấu tạo:



Cấu tạo và đặc tính cảm biến vị trí bướm ga

1. Nam châm
2. Mạch IC Hall
3. Hai dây VTA
4. Dây nguồn VC

5. Dây mass

Nguyên lý hoạt động:

Hiện tượng Hall là một hiệu ứng vật lý được thực hiện khi áp dụng một từ trường vuông góc lên một bản làm bằng kim loại hay chất bán dẫn hay chất dẫn điện nói chung (thanh Hall) đang có dòng điện chạy qua. Lúc đó người ta nhận được hiệu điện thế (hiệu thế Hall) sinh ra tại hai mặt đối diện của thanh Hall.

Khi đạp ga càng mạnh, trục bướm ga xoay nhiều, bướm ga mở lớn, số đường sức từ sinh ra bởi nam châm đi qua IC (phần tử Hall) càng nhiều, sinh ra điện trường tăng tuyến tính, tín hiệu này truyền về ECU.

Cảm biến vị trí bướm ga có 2 phần tử Hall để đặt đối diện nhau, cho ra 2 tín hiệu VT1, VT2 để tăng độ tin cậy, hai tín hiệu này nếu không cùng hệ số góc thì cảm biến có lỗi.

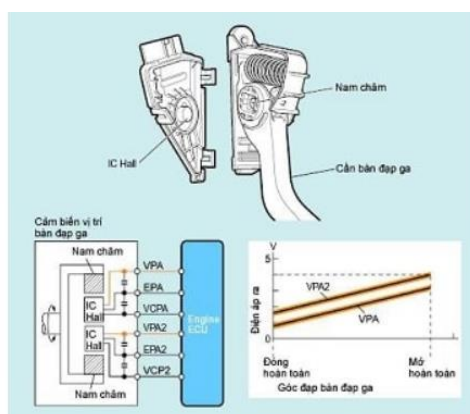
g) Cảm biến vị trí bàn đạp ga

Cảm biến chân ga được bố trí dưới bàn đạp ga.

Chức năng: Cảm biến bàn đạp chân ga được sử dụng để đo độ mở của bàn đạp chân ga khi người lái xe nhấn vào bàn đạp. Lúc này, tín hiệu từ cảm biến bàn đạp ga sẽ được gửi về ECU và ECU sẽ sử dụng các dữ liệu này để điều khiển mô tơ bướm ga mở bướm ga cho động cơ tăng tốc theo độ mở của bàn đạp chân ga và theo chế độ lái chính xác nhất.

Đa số là loại sử dụng biến trở kép, một số cảm biến bàn đạp ga sử dụng hiệu ứng Hall.

Cấu tạo và nguyên lý: (loại Hall)



cảm biến bàn đạp ga cũng được cấp nguồn VC (5V), và Mass, có 2 dây tín hiệu, điện áp của 2 chân tín hiệu (Signal) cảm biến cũng thay đổi theo độ mở của bướm ga nhưng dựa trên nguyên lý hiệu ứng Hall (có 2 loại):

* Loại thuận: 2 tín hiệu cùng tăng cùng giảm.

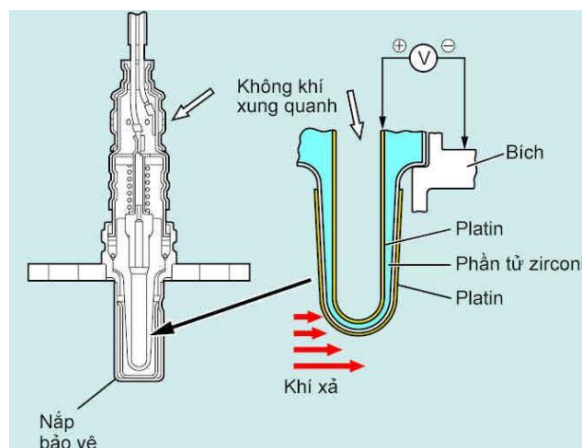
* Loại nghịch: 1 tín hiệu tăng 1 tín hiệu giảm.

h) Cảm biến oxy

Chức năng: Cảm biến oxy được dùng để xác định thành phần hòa khí tức thời của động cơ đang hoạt động. Nó phát ra một tín hiệu điện áp gửi về ECU để điều chỉnh tỉ lệ hòa khí thích hợp trong một điều kiện làm việc nhất định.

Cảm biến oxy được gắn ở đường ống thải.

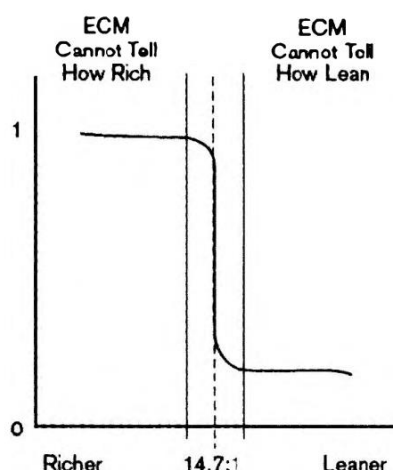
Cấu tạo: Chế tạo từ dioxide zirconium (ZrO_2)



Cấu tạo cảm biến oxy

Nguyên lý làm việc:

Cảm biến oxy một pin điện có sức điện động phụ thuộc vào nồng độ oxy trong khí thải với ZrO_2 là chất điện phân. Sự thay đổi nồng độ oxy gây ra phân cực ion tạo ra điện áp.



Điện áp phát ra của cảm biến oxy phụ thuộc vào tỷ lệ hòa khí.

2.2.2. Khối đọc và xử lý tín hiệu

❖ Mục đích:

- Gửi và nhận tín hiệu hiện tại của xe nhanh chóng và chính xác.
- Xử lý tín hiệu để bộ đọc có thể đọc được và có thể gửi lên ứng dụng đám mây.
- Bộ sản phẩm lắp đặt dễ dàng lên ô tô, để người lái theo dõi được tình trạng mức nhiên liệu qua điện thoại và nó sẽ có tính năng phát cảnh báo khi vận hành.

2.2.2.1 Mạch vi điều khiển

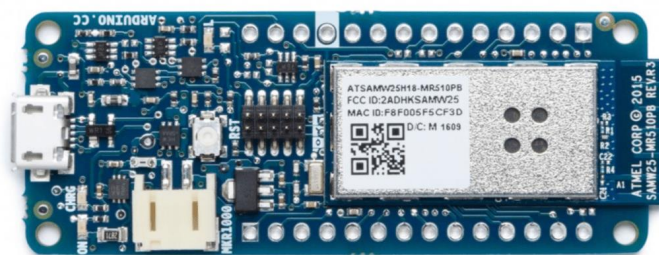
*Vi điều khiển Arduino MKR1000

Arduino MKR1000 đã được thiết kế để cung cấp một giải pháp thiết thực và tiết kiệm chi phí cho các nhà sản xuất đang tìm cách thêm kết nối Wi-Fi vào các dự án của họ với ít kinh nghiệm trước đây về mạng.

Không giống như hầu hết các bo mạch Arduino & Genuino, Arduino MKR1000 chạy ở mức 3,3V. Điện áp tối đa mà các chân I/O có thể chịu được là 3,3V. Đặt điện áp cao hơn 3,3V vào bất kỳ chân I/O nào có thể làm hỏng bo mạch. Mặc dù có thể xuất ra các thiết bị kỹ thuật số 5V, nhưng giao tiếp hai chiều với các thiết bị 5V cần chuyển mức phù hợp.

Các giao tiếp trên mạch	UART	Yes
	I2C	Yes
	SPI	Yes

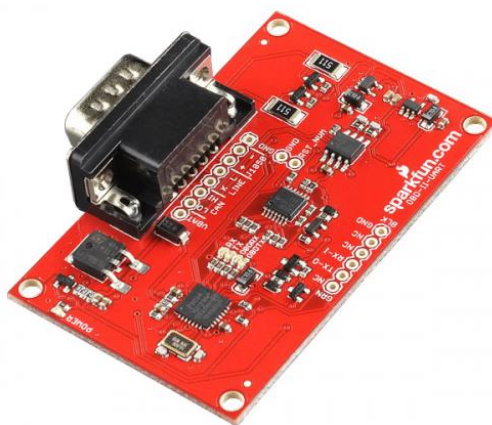
Các kết nối	Wi-Fi	ATWINC1500 (part of ATSAMW25 SoC)
	Secure element	ATECC508A



Vi điều khiển Arduino MKR1000 thực tế

2.2.2.2 Mạch chuyển đổi tín hiệu CanBus (SparkFun OBD-II UART)

Bảng này cho phép bạn giao tiếp với xe buýt OBD-II trên ô tô của bạn. Nó cung cấp cho bạn một giao diện nối tiếp sử dụng bộ lệnh ELM327 và hỗ trợ tất cả các tiêu chuẩn chính của OBD-II như CAN. Bo mạch chứa chip STN1110, đây là trình thông dịch từ OBD sang UART có thể được sử dụng để chuyển đổi thông báo giữa bất kỳ giao thức OBD-II nào hiện đang được sử dụng và UART.

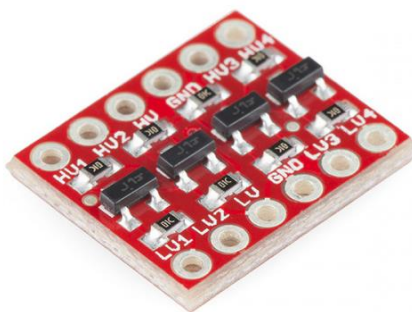


SparkFun OBD-II UART thực tế

2.2.2.3 Bộ chuyển đổi mức logic SparkFun - Hai chiều

Tuy nhiên, cần lưu ý rằng bảng thông dịch có điện áp I/O là 5 V, có thể làm hỏng I/O của bảng Arduino MKR, nếu kết nối chúng trực tiếp. Arduino MKR WiFi 1000 chạy ở điện áp thấp hơn và điện áp I/O của nó là 3,3 V. Do đó, cần có bộ dịch mức để chuyển đổi tín hiệu từ 5 V sang 3,3 V và ngược lại.

Thiết bị có thể chuyển 3,3V lên 5V hoặc 5V xuống 3,3V. Điều này được gọi là dịch chuyển mức logic.



Bộ chuyển đổi mức logic SparkFun thực tế

2.2.3. Khối xuất kết quả

Tín hiệu OBD-II sau khi đã được xử lý thành dạng dữ liệu sẽ được gửi trực tuyến, lưu trữ trên cloud server có sẵn và hiển thị dưới các dạng đồ thị trên cloud dashboard.

Yêu cầu: Dữ liệu được cho phép truy cập với bất kì trình duyệt web hoặc ứng dụng trên toàn cầu.

a) Cloud server

❖ Dweet.io



Dweet.io cơ bản là một API đơn giản mà có thể được gọi từ bất kì trình duyệt Web hoặc ứng dụng nào. Địa chỉ trang chủ: <https://dweet.io/>

Chức năng: Dùng để lưu dữ liệu từ dự án Arduino và lấy lại dữ liệu đó từ ứng dụng khác.

- **Ưu điểm:**

+ Dễ dàng truy cập, sử dụng.

- + Dễ dàng để lưu dữ liệu từ dự án Arduino và lấy lại dữ liệu đó từ ứng dụng khác.
- + Phổ biến trong các dự án của học sinh, sinh viên.

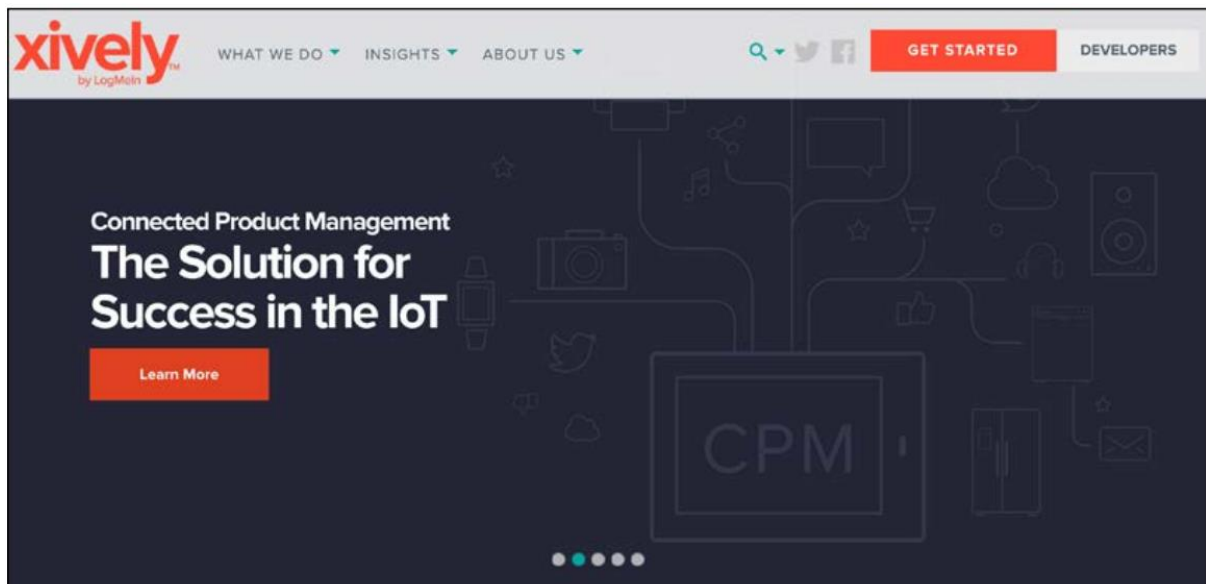
- Nhược điểm:

- + Không có nhiều tính năng như: quản lý, vẽ đồ thị.

❖ **Xivey**

Xivey là một nền tảng IOT do Google sở hữu. Xively cung cấp cho các công ty sản phẩm cách kết nối các sản phẩm, quản lý các thiết bị được kết nối và dữ liệu mà họ sản xuất cũng như tích hợp dữ liệu đó vào các hệ thống khác. Địa chỉ trang chủ:

<https://xively.com/>



- Ưu điểm:

- + Là nền tảng hoàn chỉnh, phổ biến trong kinh doanh doanh nghiệp.

+ Nhiều tính năng hơn chỉ là chỉ lưu trữ dữ liệu đơn thuần, chẳng hạn có thể vẽ đồ thị ngay trên nền tảng đó.

- Nhược điểm:

+ Truy cập, sử dụng phức tạp hơn.

+ Có thể trả phí.

❖ **SparkFun**

SparkFun tương tự Dweet.io và cho phép người dùng dễ dàng lưu trữ dữ liệu online trên bất kì trình duyệt Web hoặc ứng dụng nào. Địa chỉ trang chủ:

<https://data.sparkfun.com/>

Chức năng: Dùng để lưu trữ dữ liệu từ dự án Arduino và lấy lại dữ liệu đó từ ứng dụng khác.



- Ưu điểm:

+ Dễ dàng truy cập, sử dụng.

+ Dễ dàng để lưu trữ dữ liệu từ dự án Arduino và lấy lại dữ liệu đó từ ứng dụng khác.

- Nhược điểm:

+ Không có nhiều tính năng như: quản lý, vẽ đồ thị.

b) Cloud dashboard

❖ **Freeboard**

Là một trang web cung cấp dịch vụ hiển thị các thông tin của các sản phẩm IoT, đơn giản và trực quan nhất.

Freeboard cung cấp mã nguồn mở sản phẩm, có thể tự dựng phần giao diện trong sản phẩm của mình.

Địa chỉ trang chủ: <http://freeboard.io/>



Freeboard và Dweet.io kết nối với nhau tối ưu do cùng công ty phát triển. Freeboard có nhiều tool vẽ đồ thị đa dạng.

⇒ Vì phần cloud Server đã chọn là Dweet.io nên nhóm chọn Freeboard là nền tảng hiển thị giao diện.

Chương 3: Thiết kế bố trí chung

3.1. Bố trí chung hệ thống

*** Mục đích**

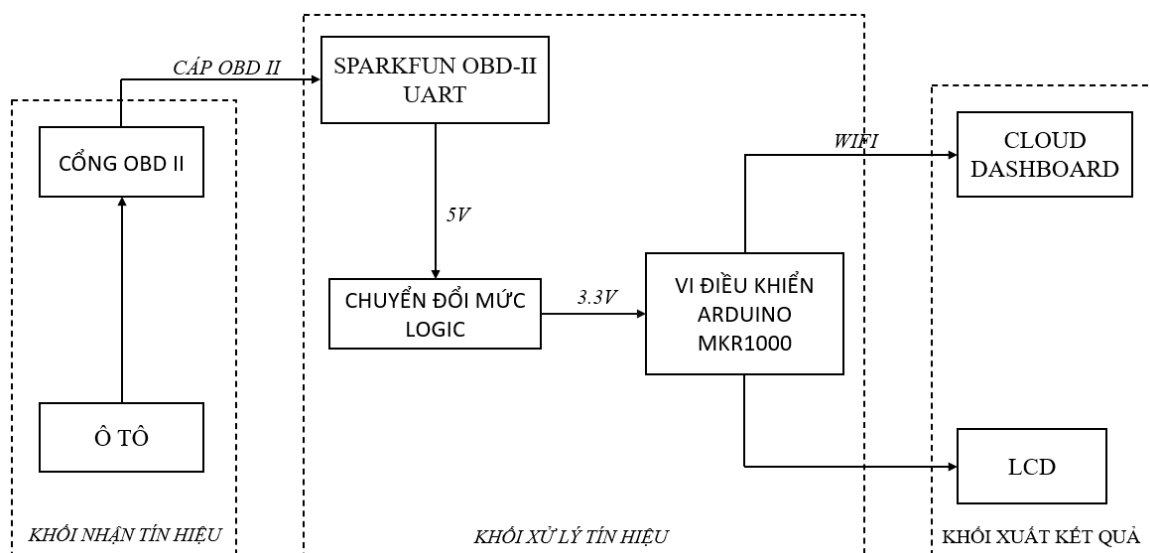
- Sơ đồ nguyên lý: Thể hiện nguyên lý hoạt động của các linh kiện điện tử, sắp xếp vị trí lắp đặt, lắp ráp của các linh kiện theo đúng chức năng của chúng và thể hiện được các sơ đồ khối chức năng của hệ thống.
- Sơ đồ kết nối mạch điện giúp hiểu rõ được cách sắp xếp vật lý của các thành phần linh kiện trong mạch và các mối quan hệ dây giữa chúng. Sơ đồ mạch được tạo ra như một hướng dẫn để thực hiện thiết kế mạch điện và giao tiếp giữa các linh kiện.

*** Yêu cầu**

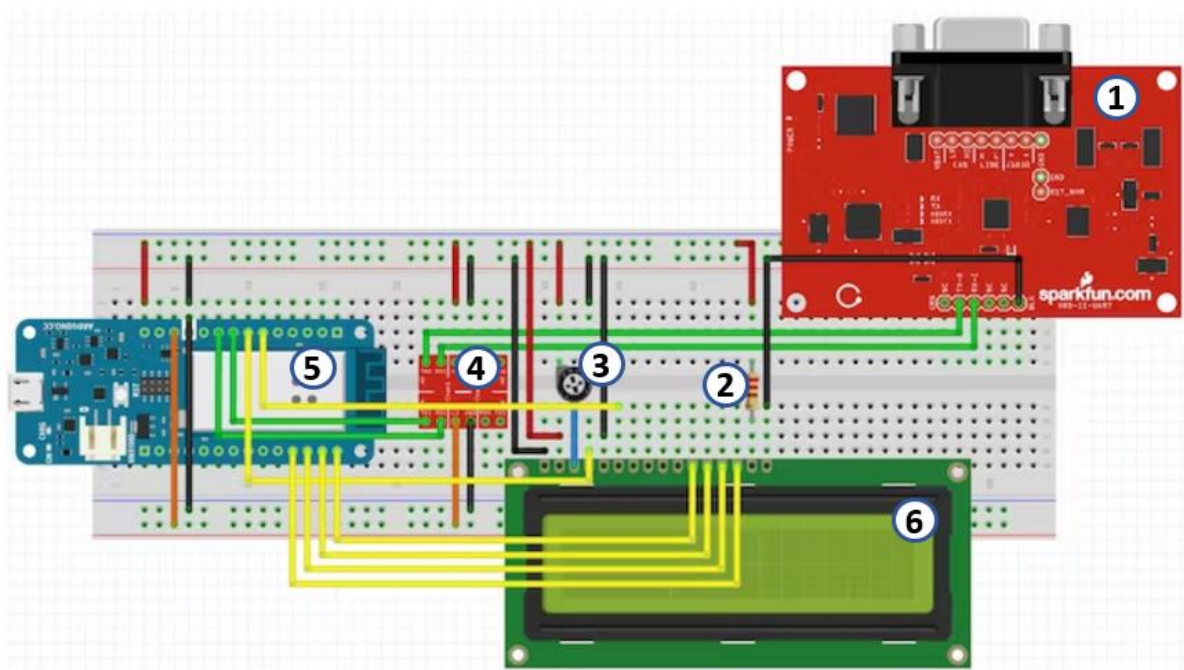
- Gồm 3 khối chức năng để vận hành hệ thống.
- Đường kết nối giữa các thiết bị với nhau, giữa các khối phải đúng với chức năng và nguyên lý hoạt động của phần tử linh tử.
- Các chân chức năng của các linh kiện phải kết nối đúng với nhau.
- Sơ đồ mạch điện thể hiện đúng với hướng dẫn của sơ đồ nguyên lý.

***Điều kiện làm việc**

- Hệ thống không hoạt động nếu sắp xếp sai vị trí các phần tử trong sơ đồ nguyên lý.
- Dễ cháy mạch hoặc không hoạt động khi sơ đồ đấu nối sai.
- Sơ đồ mạch điện cho thấy hoạt động đúng với sơ đồ nguyên lý và các tiêu chuẩn thiết kế.



Hình: Sơ đồ nguyên lý bộ phận đọc phát tín hiệu



Hình: Sơ đồ kết nối mạch điện bộ phận đọc phát tín hiệu và hiển thị LCD

- | | |
|-------------------------|--------------------------------------|
| 1: SparkFun OBD II Uart | 4: Mạch chuyển đổi mức logic 5V-3.3V |
| 2: Điện trở | 5: Arduino MKR1000 |
| 3: Nút nhấn | 6: LCD |

3.2. Nguyên lý hoạt động hệ thống

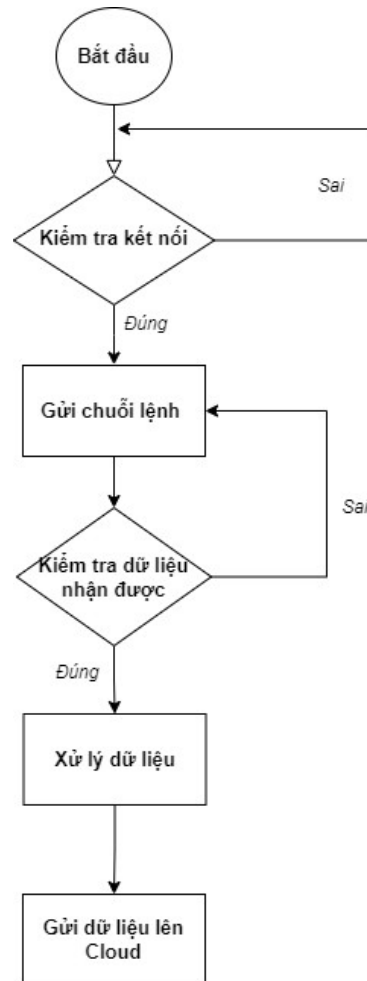
Hệ thống được thiết kế sao cho có thể đọc và hiển thị các thông số như gồm tốc độ động cơ, tốc độ xe, nhiệt độ dầu bôi trơn, điện áp acqui bằng cách kết nối Arduino với CAN Bus trên Mitsubishi xpander 2019. Quá trình làm việc của hệ thống được mô tả

như sau: Mạch SparkFun OBD II Uart giao tiếp OBD II trên xe và giao tiếp serial với vi điều khiển Arduino MKR 1000; có nhiệm vụ “thông dịch” tín hiệu gửi và nhận giữa CAN Bus và vi điều khiển. Vi điều khiển sẽ gửi tín hiệu yêu cầu và tín hiệu OBD II sẽ được gửi về. Từ đó, vi điều khiển sẽ đọc và xử lý tín hiệu thành các thông số. Khối xuất kết quả gồm 2 hình thức: Lcd và cloud dashboard.

Chương 4: Thiết kế kỹ thuật

4.1. Thiết kế phần mềm

4.1.1. Lưu đồ giải thuật



4.1.1. Chương trình Arduino

❖ Các dòng lệnh quan trọng:

+ Thư viện tương thích cho giao tiếp CAN:

```
+ #include <SPI.h>
```

+ Khởi tạo biến:

Bộ đệm ký tự sẽ lưu trữ dữ liệu từ cổng nối tiếp:

```
char rxData[40];  
char rxIndex = 0;  
int i=0,j=0;  
char text[20];
```

Các biến lưu trữ giá trị tín hiệu:

```
int vSpeed = 0;
int vRPM = 0;
int vEngine_Coolant_Temp = 0;
int vPedal_Position = 0;
float vOxygen_Sensor = 0;
int vIntake_Air_Temp = 0;
int vVoltage = 0;
int vThrottle_Position= 0;
```

+ Tạo hàm đọc dữ liệu từ CAN Bus:

```
void OBD_read(void)
```

Dùng cấu trúc vòng lặp while với hàm Serial1.available() để kiểm tra nếu nhận được số byte tín hiệu thì thực hiện đọc.

```
while(Serial1.available() > 0) {
```

Dùng cấu trúc if-else:

```
if(Serial1.peek() == '\r'){ //kiểm tra kết thúc chuỗi
```

➔ Xóa bộ đệm serial buffer

```
➔ rxData[rxIndex] = '\0';
```

Nếu không phát hiện kết thúc chuỗi, thực hiện nhận kí tự

```
else{
    // thực hiện nhận kí tự từ cổng serial
    inChar = Serial1.read();
    if(inChar != '\n'){
        // cộng ký tự mới vào chuỗi và tăng biến index
        rxData[rxIndex++] = inChar;
```

+ Tạo hàm dọn bộ đệm:

```
void clearBuffer()
```

+ Tạo hàm xử lý:

```
void xuly(void)
```

Kiểm tra 2 trường hợp:

1. Trường hợp kí tự đầu là dấu '>'
 2. Trường hợp kí tự đầu không là dấu '>'
- + Tạo các hàm nhận và tính toán tín hiệu

```
+ getRPM();  
+ getSPEED();  
+ getVOLT();  
+ getIntake_air_temperature();  
+ getThrottle_position();  
+ getOxygen_Sensor_2();  
+ getAccelerator_pedal_position_D();
```

- Gửi byte PID biến cần lấy thông tin

VD:

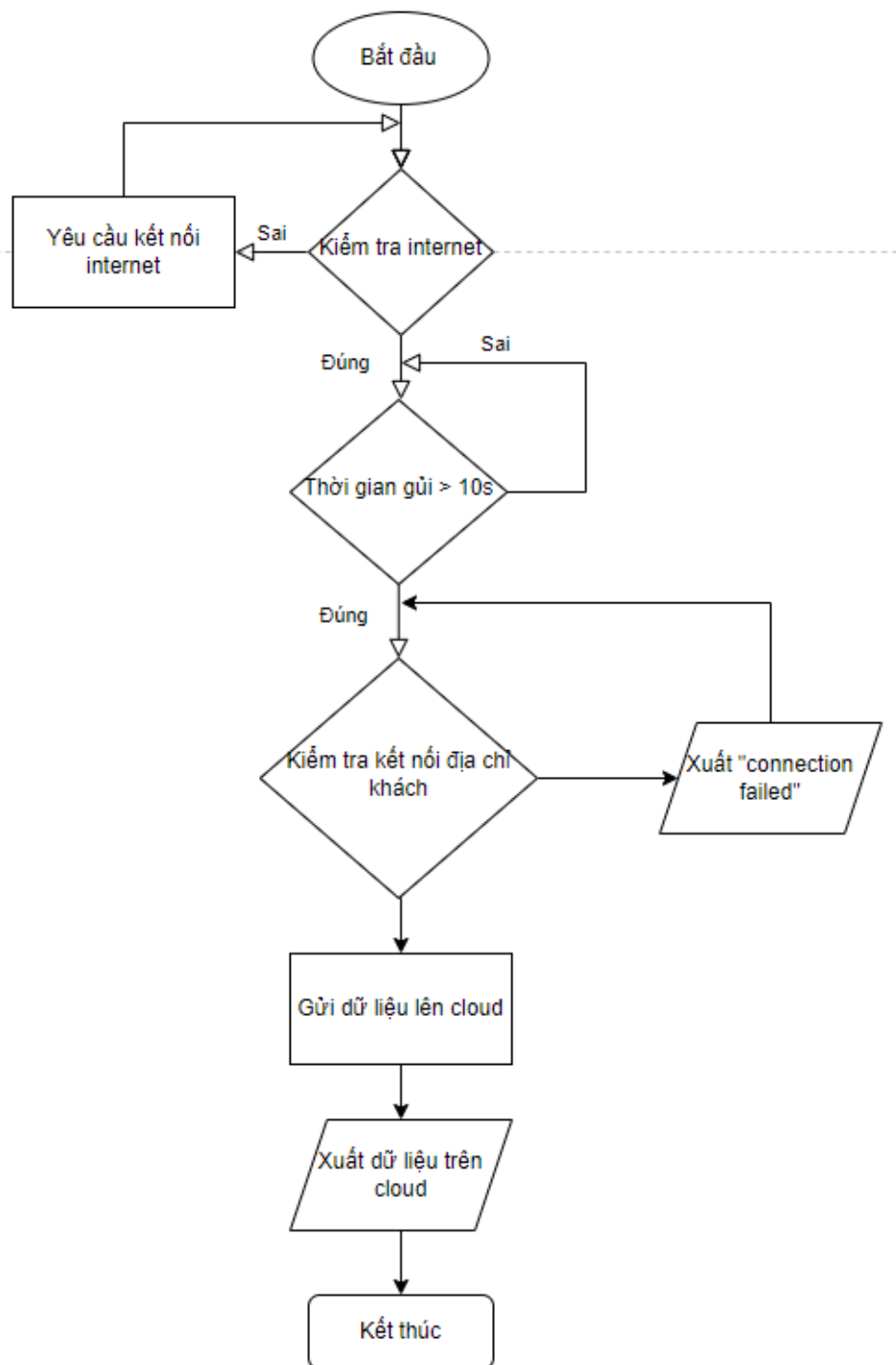
```
Serial1.print("010C\r");
```

- Đọc tín hiệu
- Tính toán thông số tín hiệu

VD: Vòng tua RPM sẽ được tính qua công thức $(256A+B)/4$.

4.2. Thiết kế giao diện hiển thị

4.2.1. Lưu đồ giải thuật



4.2.2. Lập trình Arduino

❖ Cấu hình board

Trước khi sử dụng chip Wi-Fi trên board, chúng ta cần một thư viện Arduino để có thể điều khiển nó. Thư viện dành cho chip này là WiFi101 và có thể thấy trong phần quản lý thư viện của Arduino. Để truy cập phần quản lý thư viện, chỉ cần đơn giản đến theo đường dẫn Sketch | Include Library | Manage Libraries trong Arduino IDE. Tiếp đến gõ WiFi101 để tìm kiếm thư viện:



⇒ Nhấn Install để cài đặt thư viện.

* Kết nối chip vào wifi cục bộ:

+ Thêm thư viện:

```
// Kết nối board với wifi
#include <SPI.h>
#include <WiFi101.h>

//Thư viện WiFi101 giúp việc sử dụng chip Wi-Fi onboard của board MKR1000 trở
nên dễ dàng, và dễ kết nối board vào Internet
//SPI (Serial Peripheral Bus) là một chuẩn truyền thông Master-Slave
```

+ Khởi tạo thông số mạng: SSID, pass.

```
// Thông tin mạng và server cloud
char ssid[] = ""; // your network SSID (name)// ssid: tên WiFi của điểm truy cập mà
chúng ta muốn kết nối đến, có thể có tối đa lên đến 32 ký tự.
char pass[] = ""; // your network password, nhập pass mạng truy cập
int status = WL_IDLE_STATUS; //Giai đoạn WiFi đang trong quá trình thay đổi
giữa các trạng thái, . giá trị = 0
```

+ Kết nối wifi trong hàm void setup():

```
// Kiểm tra trạng thái kết nối cho đến khi có báo cáo WL_CONNECTED
// Kiểm tra hoạt động của block wifi trong chip
if (WiFi.status() == WL_NO_SHIELD) {
// WL_NO_SHIELD = 255,
Serial.println("WiFi not ready");
while(true);
}
while ( status != WL_CONNECTED) {
// Không kết nối được mạng, WL_CONNECTED = 3
```

```
Serial.print("Attempting to connect to WPA SSID: ");  
Serial.println(ssid);
```

```
// Connect to WPA/WPA2 network:
```

```
//Chuyển đổi sang chế độ station, ta dùng hàm begin. Các tham số cần thiết sẽ là SSID  
và password, để module có thể kết nối đến một Access Point (AP) cụ thể  
status = WiFi.begin(ssid, pass);
```

```
// Chờ 5s để kết nối  
delay(5000);}
```

❖ Truyền dữ liệu lên cloud

+ Bước 1: Xác định địa chỉ server cần kết nối đến board.

```
char server[] = "dweet.io";
```

+ Bước 2: Xác định khoảng thời gian gửi dữ liệu lên server.

```
unsigned long lastConnectionTime = 0; //theo dõi thời gian cuối  
const unsigned long postingInterval = 10L * 1000L; // post data every 10 seconds, up  
data mỗi 10 s
```

+ Bước 3: Khởi tạo máy khách (vi điều khiển).

```
WiFiClient client; //Initialize the wifi client
```

+ Bước 4: Tạo hàm kết nối đến server đã xác định và truyền dữ liệu lên cloud.

```
void httpRequest(int vSpeed, int vRPM, int vEngine_Coolant_Tem; int  
vPedal_Position, float vOxygen_Sensor, int vIntake_Air_Temp, int vVoltage, int  
vThrottle_Position ) {  
    client.stop(); // khôi phục client sau khi được gọi  
    // create data string to send to freeboard. tạo chuỗi data gửi  
    if (client.connect(server, 80)){ // clien.connect: điều kiện kiểm tra kết nối địa chỉ  
khách , 80 là ô chấm máy chủ  
        Serial.println("Connected");  
        Serial.println(vRPM);  
        Serial.println(vSpeed);  
        Serial.println(vPedal_Position);
```

```

Serial.println(vOxygen_Sensor);
Serial.println(vIntake_Air_Temp);
Serial.println(vVoltage);
Serial.println(vThrottle_Position);
String data = "GET /dweet/for/arduinoMRK1000"

data.concat("&RPM=");
data.concat(vRPM); // upload engine RPM
data.concat("&Speed=");
data.concat(vSpeed); // upload car speed
data.concat("&Engine_Coolant_Temp=");
data.concat(vEngine_Coolant_Temp); // upload intake air temperature
data.concat("&Pedal_Position=");
data.concat(vPedal_Position); // upload pedal position
data.concat("&Oxygen_Sensor=");
data.concat(vOxygen_Sensor); // upload Oxygen Sensor
data.concat("&Voltage=");
data.concat(vVoltage); // upload battery voltage
data.concat("&Intake_Air_Temp=");
data.concat(vIntake_Air_Temp); // upload intake air temperature
data.concat("&Throttle_Position=");
data.concat(vThrottle_Position); // upload throttle position
client.println(data); //in ra thông số cho máy khách
client.println("Host: dweet.io");
client.println("User-Agent: ArduinoWiFi/1.1");
client.println("Connection: close");
client.println();
// Note the time that the connection was made:
  lastConnectionTime = millis();
}
else {
// if you couldn't make a connection:
Serial.println("connection failed");

```

```
}  
}
```

+ Bước 5: Trong hàm `loop()`, kiểm tra thời gian gửi dữ liệu.

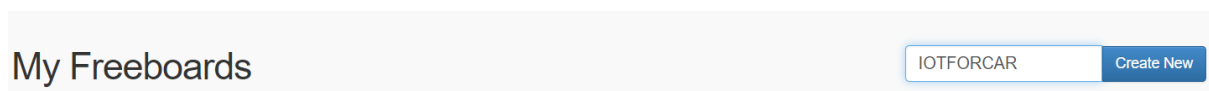
```
if (millis() - lastConnectionTime > postingInterval) {  
    httpRequest(vSpeed, vRPM, vEngine_Coolant_Temp, vPedal_Position,  
vOxygen_Sensor, vIntake_Air_Temp, vVoltage ,vThrottle_Position);// gửi yêu cầu  
  
    lastConnectionTime = millis();  
}  
// Kiểm tra kết nối network mỗi 5s  
delay(5000);  
}
```

4.2.3. Tạo giao diện hiển thị trên Freeboard

+ Truy cập địa chỉ trang chủ để tạo tài khoản:

<http://freeboard.io/>

+ Bên trong giao diện web, tạo một dashboard mới:



+ Bên trong bảng điều khiển, thêm 1 datasource mới:

DATASOURCE

A datasource for connecting to things at [dweet.io](#).

TYPE:

NAME:

THING NAME:
Example: salty-dog-1



KEY:

If the thing is not locked, you can ignore this field

SHOW FULL PAYLOAD: ☐ NO
If on, gives access to the full Dweet payload (used to obtain timestamp). If not, only the Content object is captured

SAVE CANCEL

+Data source mới xuất hiện trong bảng điều khiển:

DATASOURCES		
Name	Last Updated	
ArduinoMRK1000	never	 
ADD		

+ Vẽ đồ thị hiển thị thông số bằng có chọn ADD PANEL và cài đặt các thành phần:

Ví dụ: đồ thị hiển thị thông số tốc độ xe.

WIDGET

TYPE

Gauge


TITLE

CAR SPEED [MPH]

VALUE

datasources["ArduinoMRK1000"]["Speed"]

+ DATASOURCE

 .JS EDITOR

UNITS

MPH

MINIMUM

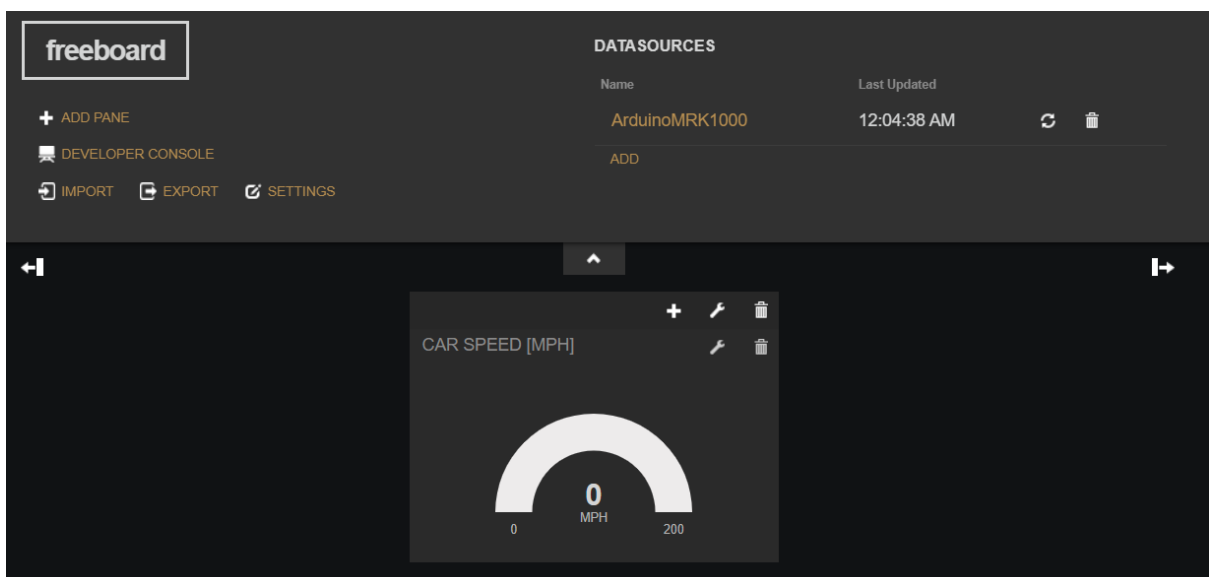
0

MAXIMUM

200

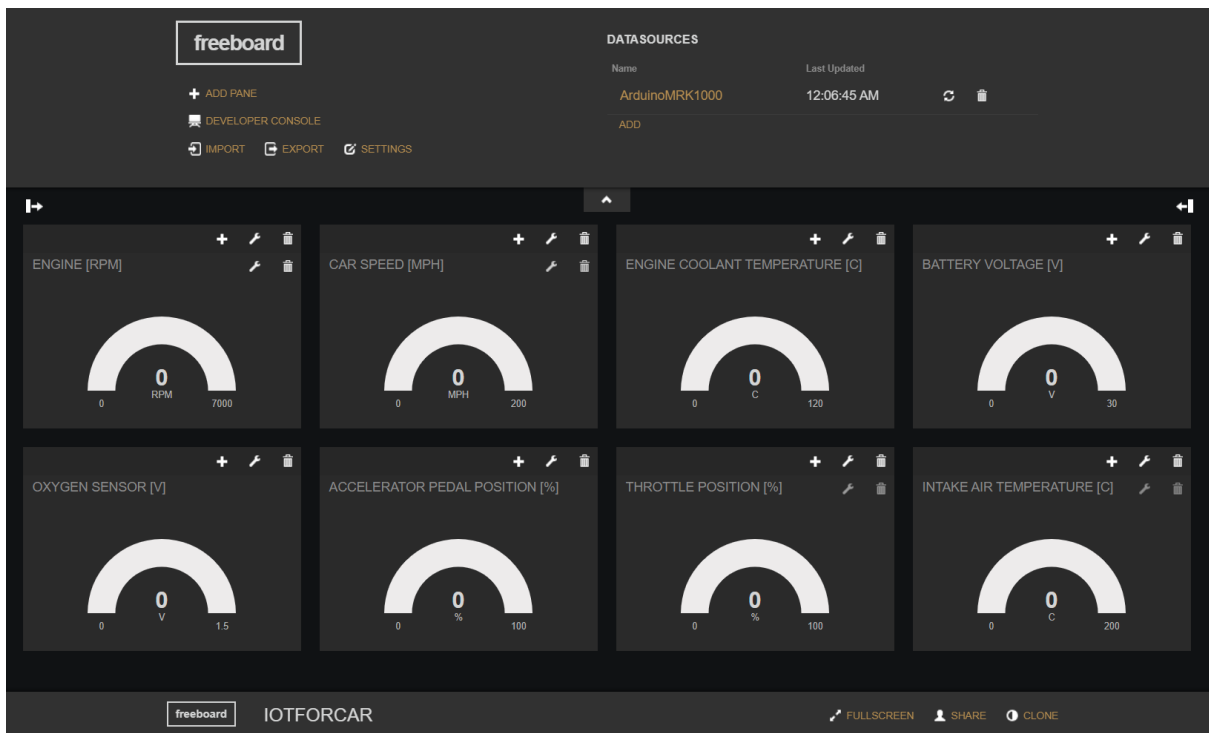
SAVE

CANCEL



⇒ Cách làm tương tự các thông số còn lại.

➤ **Kết quả:**

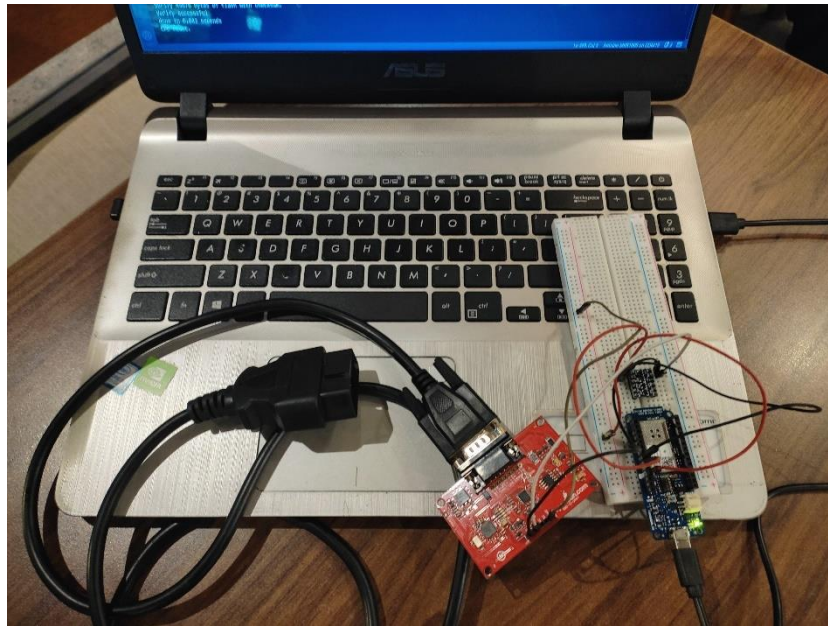


Chương 5: Kết luận

5.1. Kết quả đạt được

5.1.1. Tín hiệu đọc được và xử lý

❖ Mô hình



Mô hình đọc và xử lý tín hiệu OBD II

❖ Tốc độ bánh xe

```

toc do xe la: 0 Km/h
toc do xe la(hex): 41 0D 00
toc do xe la: 0 Km/h
toc do xe la(hex): 41 0D 00
toc do xe la: 0 Km/h
toc do xe la(hex): 41 0D 00
toc do xe la: 0 Km/h
toc do xe la(hex): 41 0D 00
toc do xe la: 0 Km/h
toc do xe la(hex): 41 0D 00

```

❖ Tốc độ động cơ

```

toc do dong co la: 851 rpm
toc do dong co la(hex): 41 0C 0D 58
toc do dong co la: 854 rpm
toc do dong co la(hex): 41 0C 0D 58
toc do dong co la: 854 rpm
toc do dong co la(hex): 41 0C 0D 3E
toc do dong co la: 847 rpm
toc do dong co la(hex): 41 0C 0D 3E
toc do dong co la: 847 rpm
toc do dong co la(hex): 41 0C 0D 50
toc do dong co la: 852 rpm

```

❖ Nhiệt độ nước làm mát

```

nhiet do nuoc(hex): 41 05 81
nhiet do nuoc: 89 do C
nhiet do nuoc(hex): 41 05 80
nhiet do nuoc: 88 do C
nhiet do nuoc(hex): 41 05 80
nhiet do nuoc: 88 do C
nhiet do nuoc(hex): 41 05 81
nhiet do nuoc: 89 do C
nhiet do nuoc(hex): 41 05 81
nhiet do nuoc: 89 do C
nhiet do nuoc(hex): 41 05 81

```


❖ Điện áp acqui

```
VON(hex): 41 42 38 C2
VON: 14 V
VON(hex): 41 42 38 C2
VON: 14 V
VON(hex): 41 42 38 93
VON: 14 V
VON(hex): 41 42 38 64
VON: 14 V
VON(hex): 41 42 39 09
VON: 14 V
VON(hex): 41 42 38 C2
VON: 14 V
VON(hex): 41 42 38 C2
VON: 14 V
```

❖ Nhiệt độ khí nạp

```
nhiet do khi nap la(hex): 41 0F 48
nhiet do khi nap la: 32 do C
nhiet do khi nap la(hex): 41 0F 48
nhiet do khi nap la: 32 do C
nhiet do khi nap la(hex): 41 0F 48
nhiet do khi nap la: 32 do C
nhiet do khi nap la(hex): 41 0F 48
nhiet do khi nap la: 32 do C
nhiet do khi nap la(hex): 41 0F 48
nhiet do khi nap la: 32 do C
nhiet do khi nap la(hex): 41 0F 48
nhiet do khi nap la: 32 do C
```

❖ Vị trí bướm ga

```
vi tri buom ga la(hex): 41 11 22
vi tri buom ga la: 13 %
vi tri buom ga la(hex): 41 11 22
vi tri buom ga la: 13 %
vi tri buom ga la(hex): 41 11 22
vi tri buom ga la: 13 %
vi tri buom ga la(hex): 41 11 22
vi tri buom ga la: 13 %
vi tri buom ga la(hex): 41 11 22
vi tri buom ga la: 13 %
vi tri buom ga la(hex): 41 11 22
vi tri buom ga la: 13 %
```

❖ Vị trí bàn đạp ga

```
vi tri buom ga la(hex): 41 11 22
vi tri buom ga la: 13 %
vi tri buom ga la(hex): 41 11 22
vi tri buom ga la: 13 %
vi tri buom ga la(hex): 41 11 22
vi tri buom ga la: 13 %
vi tri buom ga la(hex): 41 11 22
vi tri buom ga la: 13 %
vi tri buom ga la(hex): 41 11 22
vi tri buom ga la: 13 %
vi tri buom ga la(hex): 41 11 22
vi tri buom ga la: 13 %
```

❖ Cảm biến oxy

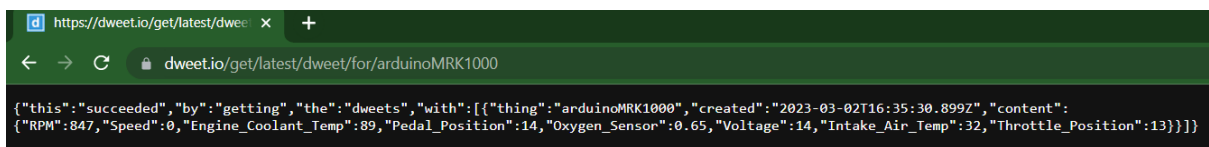
```
tin hieu cam bien oxy la(hex): 41 15 83 FF
tin hieu cam bien oxy la: 0.65 V
tin hieu cam bien oxy la(hex): 41 15 83 FF
tin hieu cam bien oxy la: 0.65 V
tin hieu cam bien oxy la(hex): 41 15 83 FF
tin hieu cam bien oxy la: 0.65 V
tin hieu cam bien oxy la(hex): 41 15 83 FF
tin hieu cam bien oxy la: 0.65 V
tin hieu cam bien oxy la(hex): 41 15 83 FF
tin hieu cam bien oxy la: 0.65 V
```

5.1.2. Dữ liệu trên Cloud Server và giao diện hiển thị trên Cloud dashboard

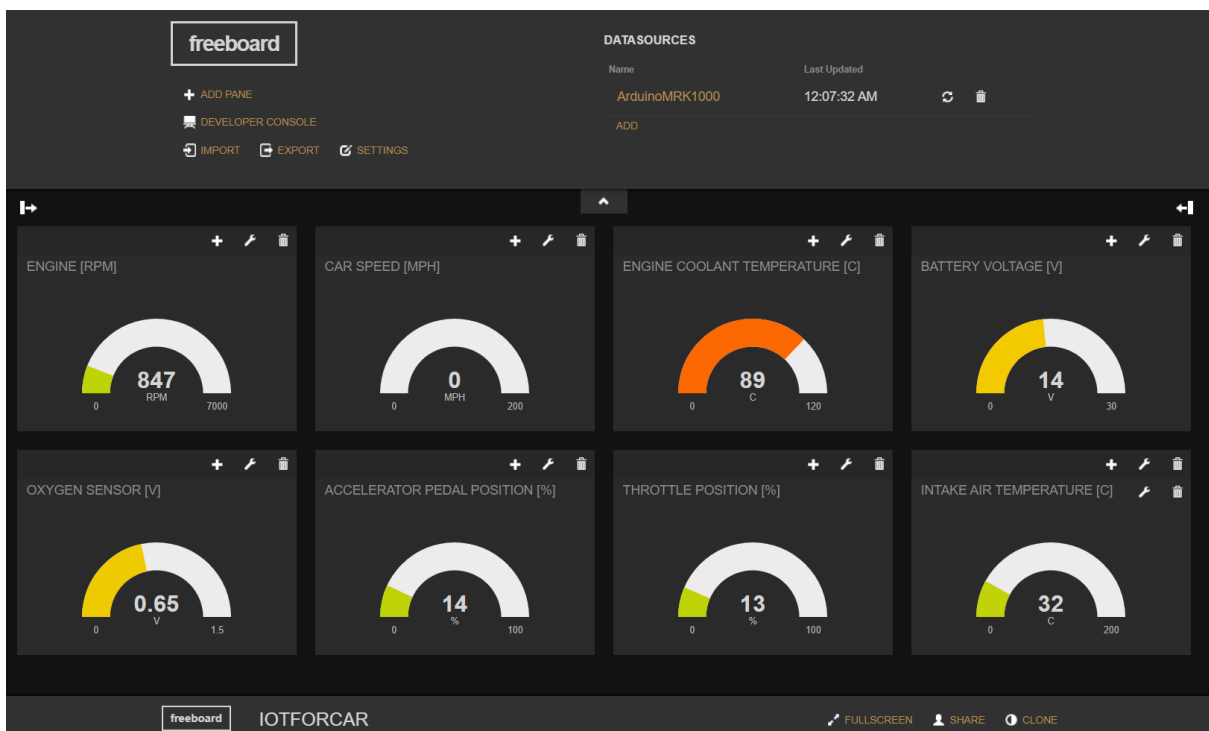
a) Lưu trữ dữ liệu trên dweet.io

- Truy cập dữ liệu mới nhất được lưu trữ trên Cloud server qua địa chỉ:

<https://dweet.io/get/latest/dweet/for/arduinoMRK1000>



b) Giao diện hiển thị trên Cloud Freeboard



⇒ Biểu đồ tương ứng với thông số tín hiệu lưu trữ trên cloud server.

5.2. Đánh giá

Mô hình xác định tín hiệu OBD II trên chiếc Mitsubishi Xpander 2019 của nhóm thực hiện đã thỏa mãn mục tiêu đề ra trong giới hạn đề tài cho phép:

- + Xây dựng mô hình mạch thu thập và hiển thị dữ liệu.

- + Thử nghiệm mô hình và đánh giá độ chính xác.
- + Đề tài được thử nghiệm trên xe Xpander 2019.
- + Dụng cụ nghiên cứu nằm trong giới hạn tài chính sinh viên.
- + Những tín hiệu có thể đọc được và xử lý phụ thuộc vào loại xe thực nghiệm.
- ✱ Với điều kiện thời gian cho phép nhóm chưa thể tiến hành thử nghiệm mô hình và đánh giá độ chính xác,

Project: “Phát triển phương pháp chuẩn đoán qua giao tiếp OBD II trên chiếc Mitshubishi Xpander 2019 và ứng dụng hiển IOT để hiển thị dữ liệu chuẩn đoán” trình bày tương đối đầy đủ, nhưng do thời gian và kiến thức liên quan đến lập trình Arduino còn hạn chế nên đề tài còn nhiều hạn chế.

5.3. Hướng phát triển

- ✓ Hướng 1: Đọc và xử lý tín hiệu OBD-II đồng thời -> dùng pointer
- ✓ Hướng 2: Xử lý tín hiệu bằng STM32F407Discovery -> Giao tiếp STM32F407Discovery Matlab -> tạo giao diện Matlab GUI & APP

Tài liệu tham khảo

[1] IOT4CAR

<https://www.hackster.io/frankzhao/iot4car-1b07f1>

[2] Automotive Electrics and Electronics - Bosh Automotive Sensor

[3] Konrad Reif. Automotive Mechatronics-Bosh Professional Automotive Infomation.

[4] PGS-TS Đỗ Văn Dũng – Điện động cơ và Điều khiển động cơ

