ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

THUYẾT MINH

THIẾT KẾ VÀ CHẾ TẠO
BỘ ĐỌC DỮ LIỆU OBD-II
SỬ DỤNG VI ĐIỀU KHIỂN ESP32

TPHCM, 2021

TÓM TẮT

Để đảm bảo giới hạn phát thải ô nhiễm, đồng thời ngăn chặn những hư hỏng nghiêm trọng sẽ xảy ra, mọi hệ thống EFI thương mại hiện nay đều được trang bị tính năng tự chẩn đoán thế hệ 2, gọi tắt là OBD-II (On-board Diagnostics-II). Bằng các công cụ chẩn đoán dựa trên sự thay đổi của thông số vận hành (data-based-FDI) hay dựa trên sự sai lệch giữa kết quả đo thực tế với kết quả ước lượng từ mô hình toán (model-based-FDI), bộ điều khiển phun xăng điện tử (ECU-Electronic Control Unit) sẽ cảnh báo người lái xe khi phát hiện ra những lỗi có thể làm cho mức phát thải của xe đạt 1,5 lần mức phát thải cho phép.

Mặc dù OBD-II đã được quốc tế chuẩn hoá và sử dụng trên mọi động cơ EFI thương mại, khả năng ECU tự truy tìm chính xác lỗi gốc làm phát sinh những lỗi điều khiển phức tạp vẫn còn rất hạn chế. Khi lỗi phát sinh xảy ra, ECU thường đưa ra một danh sách các lỗi gốc khả nghi. Nhiệm vụ tìm và khắc phục đúng lỗi gốc vẫn hoàn toàn do kỹ thuật viên đảm nhận. Trong nhiều trường hợp, do không thể xác định chính xác lỗi gốc, kỹ thuật viên thường đề xuất thay thế toàn bộ hệ thống điện-điện tử điều khiển động cơ, thậm chí thay thế cả động cơ.

Hầu hết các gara ở Việt Nam hiện nay đang sử dụng các loại máy chẩn đoán lỗi bằng cách kết nối với mạng ECU trên xe thông qua cổng kết nối J1962 và giao tiếp theo chuẩn OBD-II để đọc và hiển thị các kết quả tự chẩn đoán của ECU, song với lý do đã trình bày ở trên, bộ đọc dữ liệu OBD-II được thiết kế và chế tạo nhằm thu thập dữ liệu vận hành của động cơ tại tất cả các điều kiện vận hành có thể xảy ra dùng làm bộ dữ liệu để huấn luyện trí tuệ nhân tạo cho phép chẩn đoán lỗi gốc trên động cơ EFI.

MỤC LỤC

DANH MỤC HÌNH

DANH MỤC BẢNG

CHƯƠNG 1: GIỚI THIỆU

Bộ đọc dữ liệu OBD-II bằng vi điều khiển ESP32 bao gồm:

- Board mạch sử dụng vi điều khiển ESP32 và mô đun CAN Transceivers SN65HDV230 kết nối với ECU qua CAN-bus, thông qua cổng kết nối J1962.
- Chương trình giao tiếp với ECU qua các giao thức CAN-bus của chuẩn OBD-II.

1.1. OBD - OBD-II

OBD (On-Board Diagnostics) là một thuật ngữ đề cập đến khả năng phát hiện và tự chẩn đoán các lỗi liên quan đến điều khiển động cơ và điều khiển giám sát phát thải độc hại của động cơ.

Vấn đề phát thải ô nhiễm của động cơ đốt trong trên xe ngày càng được quan tâm; các quy định, tiêu chuẩn về mức độ phát thải của động cơ ngày càng khắt khe đòi hỏi các hãng sản xuất ô tô phải đưa ra giải pháp để giám sát mức độ phát thải và cảnh báo khi mức độ phát thải vượt cao so với ngưỡng quy định, đồng thời với đó là ngăn ngừa hư hỏng nặng nề cho các bộ phận của hệ thống xử lý khí thải cũng như bản thân động cơ.

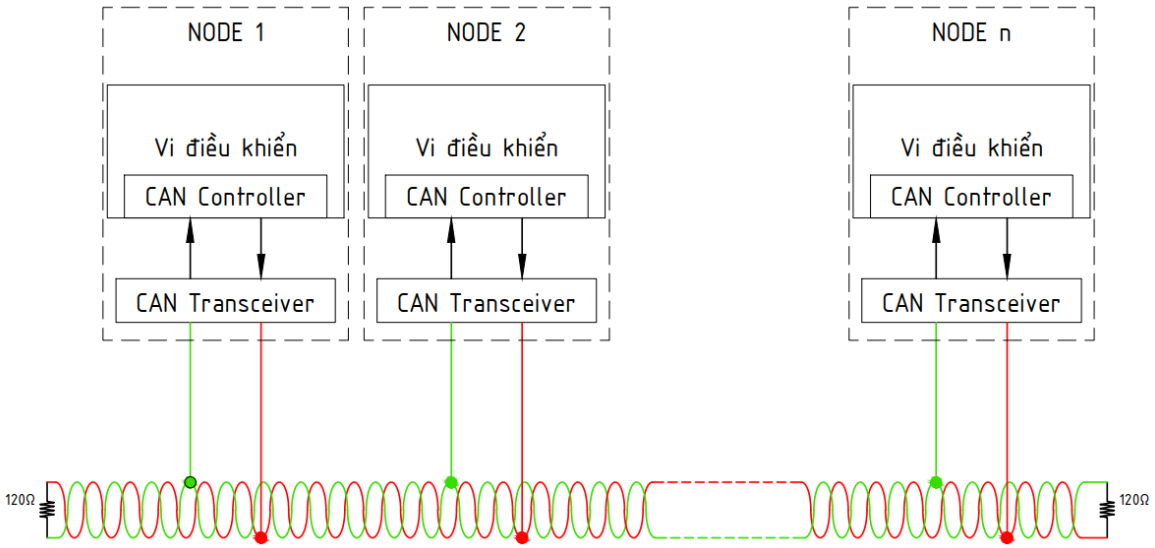
Bằng cách theo dõi các thông số vận hành của động cơ và tình trạng các mạch điện, hệ thống OBD cho phép xác định và cảnh báo các lỗi, các nguyên nhân có thể gây ra lỗi đó, cảnh báo người dùng bằng cách bật sáng đèn MIL (Malfunctions Indicator Light) và lưu trữ các mã chẩn đoán lỗi DTC (Diagnostic Trouble Code) trong bộ nhớ của ECU, cho phép kỹ thuật viên truy cập để nhanh chóng xác định và khắc phục các trục trặc. Năm 1988, OBD là hệ thống trang bị bắt buộc cho tất cả ô tô được bán ra tại Mỹ

OBD-II là bản cải tiến của OBD với nhiều tính năng mới và được chuẩn hoá quốc tế, được phát hành lần đầu năm 1994 bởi CARB và SAE (California Air Resources Board and Society of Automotive Engineer) và trở thành trang bị tiêu chuẩn bắt buộc cho tất cả các xe được bán ra tại Mỹ.

1.2. CAN-bus

CAN-bus là một chuẩn giao tiếp được thiết kế cho phép các vi điều khiển giao tiếp với nhau trong các ứng dụng mà không cần máy chủ.

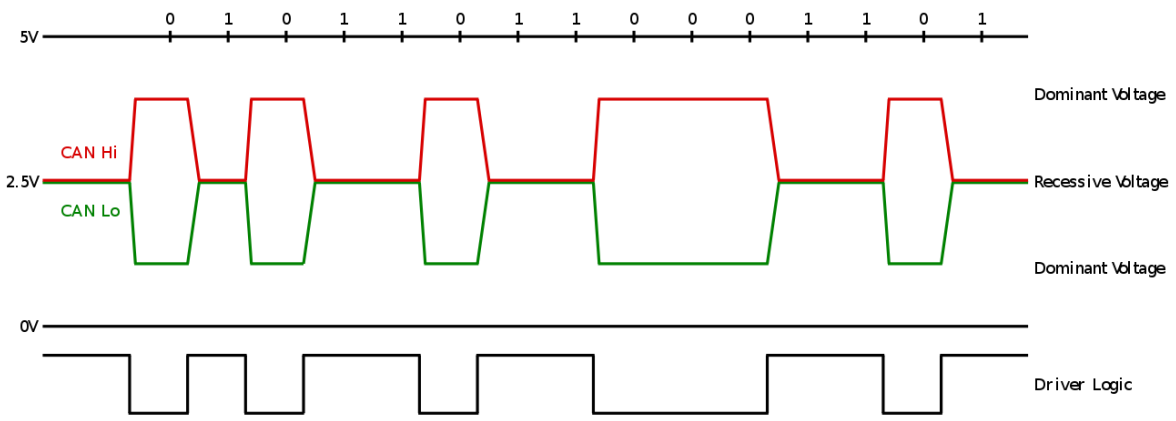
Tất cả các nút giao tiếp trong mạng Can-bus kết nối với mạng thông qua 2 dây tín hiệu, đường bus xuyên suốt mạng sử dụng dây xoắn kép (CAN Hi và CAN Lo) làm đường dẫn tín hiệu vi sai, với 2 điện trở 120Ω giới hạn ở 2 đầu.



Hình 1.1: Sơ đồ liên kết mạng CAN-bus.

Tín hiệu logic (mức 0, mức 1) giao tiếp giữa các vi điều khiển trong mạng CAN-bus được truyền dưới dạng tín hiệu vi sai (chênh lệch điện áp giữa 2 dây tín hiệu CAN Hi và CAN Lo). Trong đó, mạng CAN-bus tốc độ cao sử dụng trên ô tô (~500kbps) tồn tại 2 trạng thái:

- Trạng thái dominant tương ứng với tín liệu logic mức 0:
  + Điện áp trên dây CAN Hi so với GND là V_{CANH} = 3.5V.
  + Điện áp trên dây CAN Lo so với GND là V_{CANL} = 1.5V.
  + Chênh lệch điện áp giữa 2 dây bus V_{diff} = 2V.
- Trạng thái recessive tương ứng với tín liệu logic mức 1:
  + Điện áp trên dây CAN Hi so với GND là V_{CANH} = 2.5V.
  + Điện áp trên dây CAN Lo so với GND là V_{CANL} = 2.5V.
  + Chênh lệch điện áp giữa 2 dây bus V_{diff} = 0V.

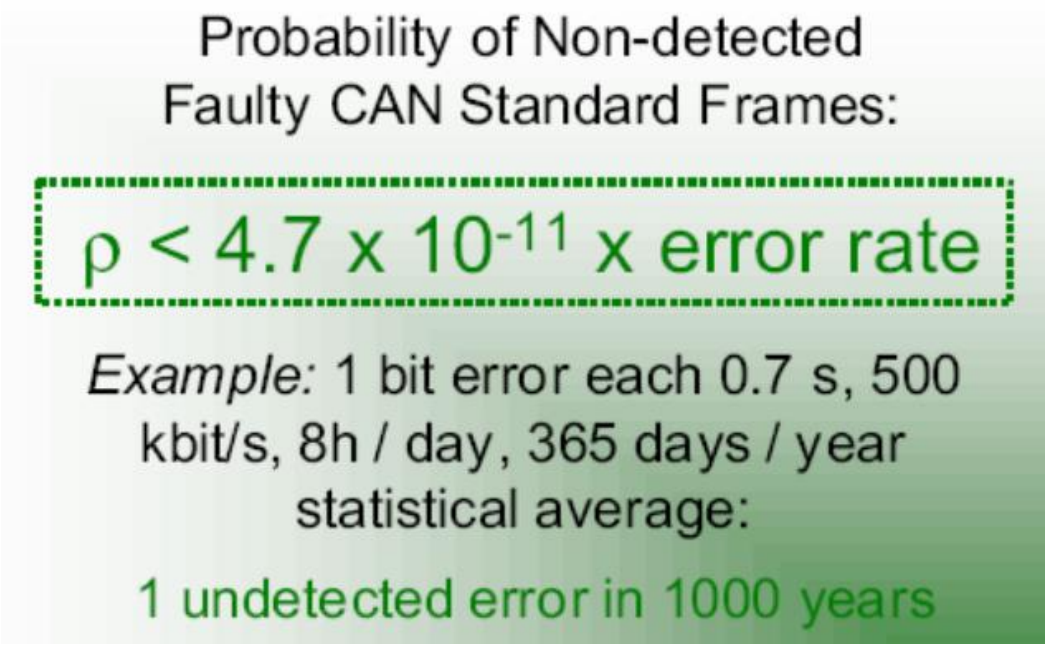


Hình 1.2: Tín hiệu trên CAN-bus và mức tín hiệu logic tương ứng.

Ngày nay, giao thức CAN đã được chuẩn hóa thành các tiêu chuẩn như ISO 11898, ISO 15765-4… Hầu như mọi nhà sản xuất chip lớn như: Intel, NEC, SIEMENS, Motorola, Maxim IC, Fairchild, Microchip, Philips, Texas Instrument, Mitsubishi, Hitachi, STMicroelectronics... đều có sản xuất ra chip CAN, hoặc có tích hợp CAN vào thành ngoại vi của vi điều khiển. Việc thực hiện giao thức CAN trở nên cực kỳ đơn giản nhờ sự hỗ trợ từ rất nhiều nhà sản xuất chip đó.

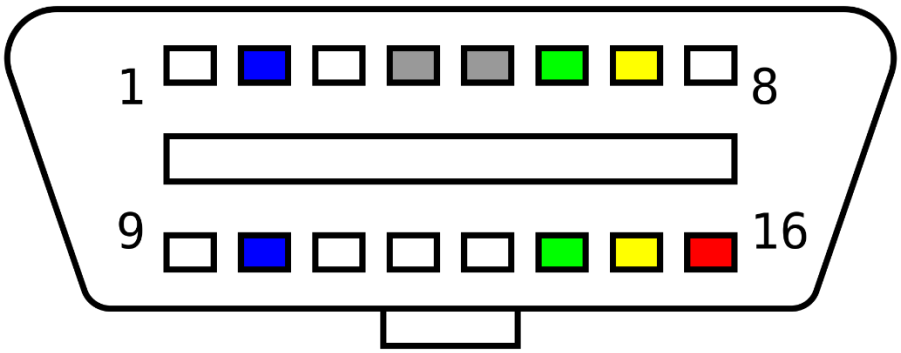
Điểm nổi trội nhất ở giao thức CAN là tính ổn định và an toàn (reliability and safety). Nhờ cơ chế phát hiện và xử lý lỗi cực mạnh, lỗi khi truyền một tin nhắn thông qua giao thức CAN hầu như được phát hiện. Theo thống kê, xác suất để một tin nhắn của giao thức CAN bị lỗi không được phát hiện chỉ là 1 khung truyền bị lỗi trong vòng 1000 năm hoạt động với các điều kiện sau:

+ Giả sử cứ 0.7s thì môi trường tác động lên đường truyền CAN-bus làm lỗi 1 bit.
+ Giả sử tốc độ truyền là 500kbps.
+ Hoạt động 8h/ngày và 365 ngày/năm.



Hình 1.3: Xác suất không phát hiện được khung truyền của giao thức CAN bị lỗi

Năm 2008, tất cả các xe bán ra tại Mỹ được yêu cầu phải sử dụng tiêu chuẩn tín hiệu ISO 15765-4 - một biến thể của CAN (Controller Area Network) để giao tiếp với OBD-II thông qua cổng kết nối J1962.



Hình 1.4: Sơ đồ chân cổng kết nối J1962 cái nhìn từ phía trước.

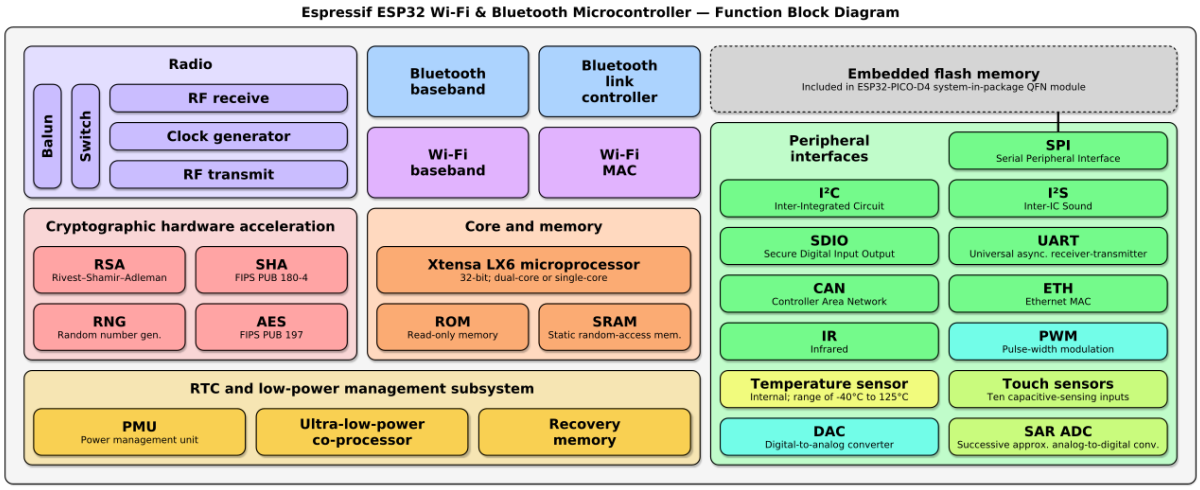
Trong đó:

+ Chân số 6 (màu xanh lá): CAN high (trong chuẩn ISO 15765-4).
+ Chân số 14 (màu xanh lá): CAN low (trong chuẩn ISO 15765-4).

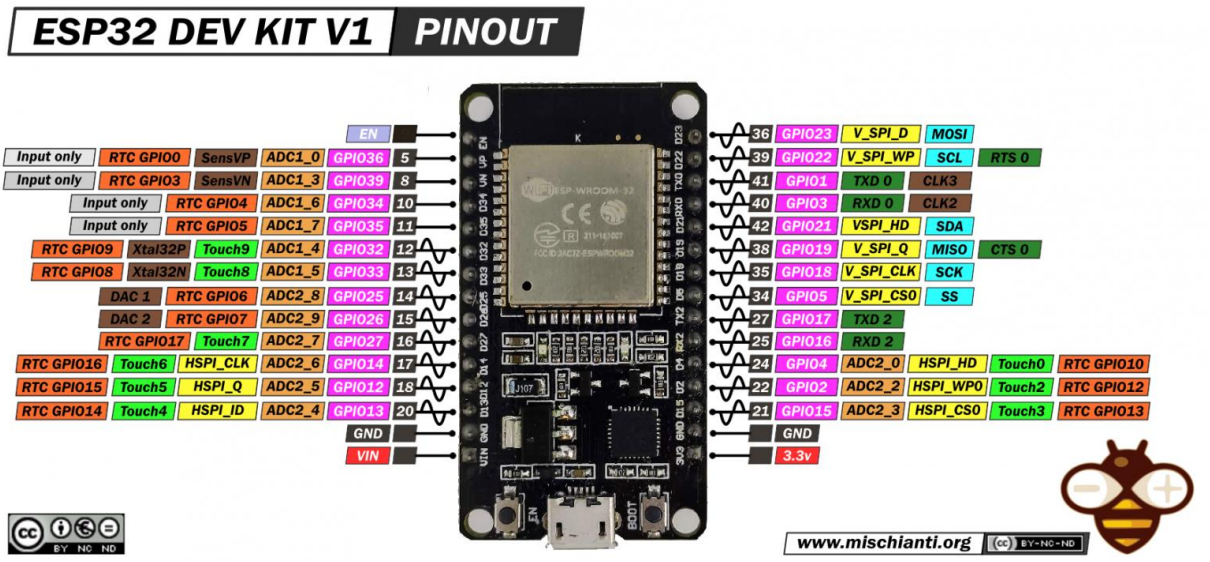
1.3. Vi điều khiển ESP32

Vi điều khiển ESP32 sử dụng cho bộ đọc dữ liệu có tên là ESP32 DEVKIT V1, thuộc sê ri vi điều khiển giá rẻ ESP32, tiêu hao năng lượng thấp, có hỗ trợ WiFi Bluetooth. Vi điều khiển họ 32-bit sử dụng bộ vi xử lý Tensilica Xtensa LX6 lõi kép và

có tích hợp ngoại vi CAN-bus 2.0, hoạt động ở điện áp 3.3V.



Hình 1.5: Sơ đồ khối chức năng của vi điều khiển ESP32

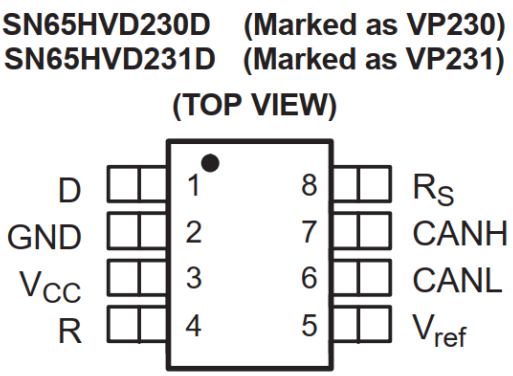


Hình 1.6: Sơ đồ chân vi điều khiển ESP32 DEVKIT V1

1.4. Mô đun CAN Transceivers SN65HDV230

Mô đun CAN Transceivers SN65HDV230 hoạt động với điện áp nguồn 3.3V, tương thích với vi điều khiển ESP32.

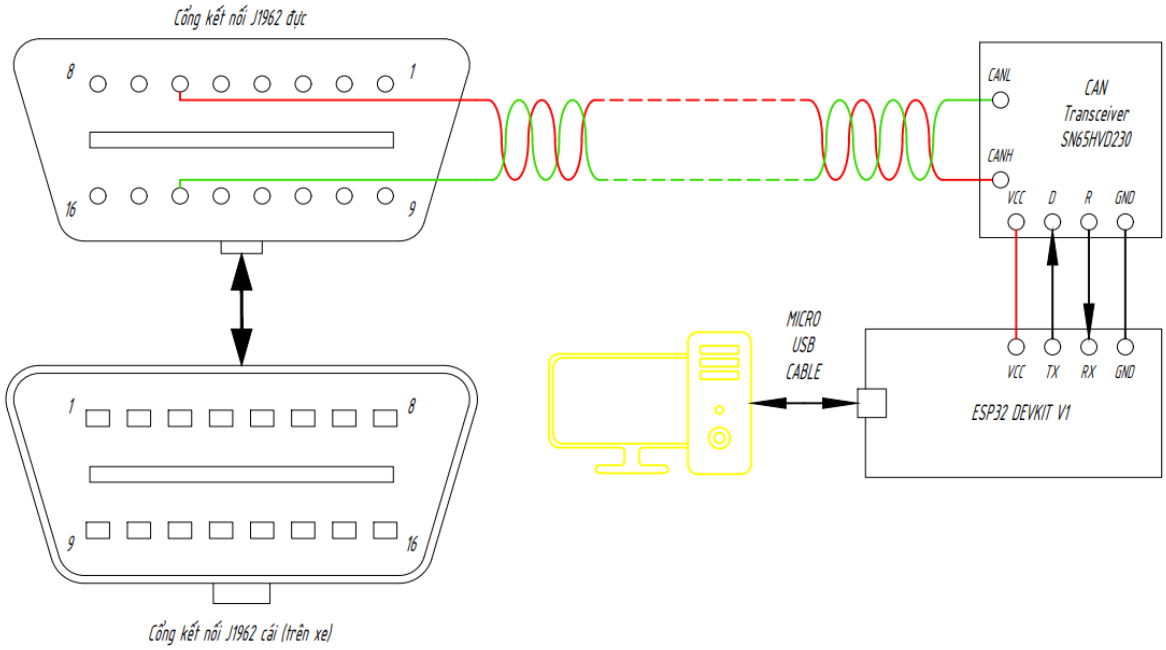
Mô đun được thiết kế cho ứng dụng trên ô tô, với điện áp chênh lệch giữa 2 dây bus lên tới ±25V, tốc độ truyền dữ liệu tối đa lên tới 1 megabit trên giây (Mbps) và cho phép tối đa 120 node trên bus.



Hình 1.7: Sơ đồ chân mô đun CAN Transceiver

CHƯƠNG 2: THIẾT KẾ BỐ TRÍ CHUNG

2.1. Sơ đồ bố trí chung



Hình 2.1: Sơ đồ bố trí chung bộ đọc dữ liệu OBD-II.

2.2. Bài toán cần giải quyết

- Thiết kế kỹ thuật phần cứng bộ đọc dữ liệu: Bo mạch và vỏ bộ đọc dữ liệu.
- Thiết kế kỹ thuật phần mềm bộ đọc dữ liệu.

2.3. Yêu cầu

- Có độ bền, tuổi thọ cao.
- Hoạt động ổn định, độ tin cậy cao.
- Dễ dàng thay thế, sửa chữa hư hỏng.
- Giá thành rẻ, phù hợp với công nghệ hiện có.

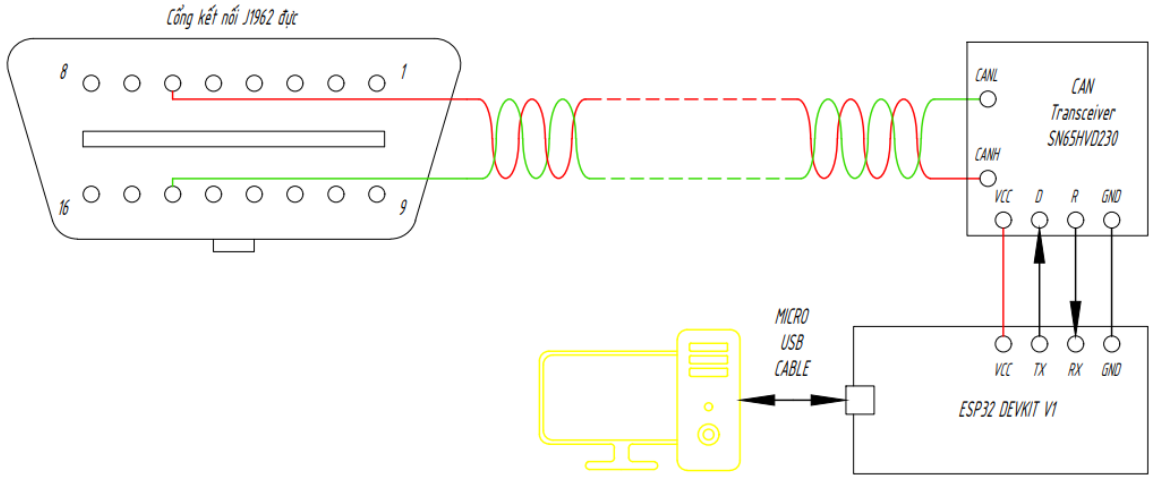
CHƯƠNG 3: THIẾT KẾ KỸ THUẬT PHẦN CỨNG

3.1. Bo mạch bộ đọc dữ liệu OBD-II

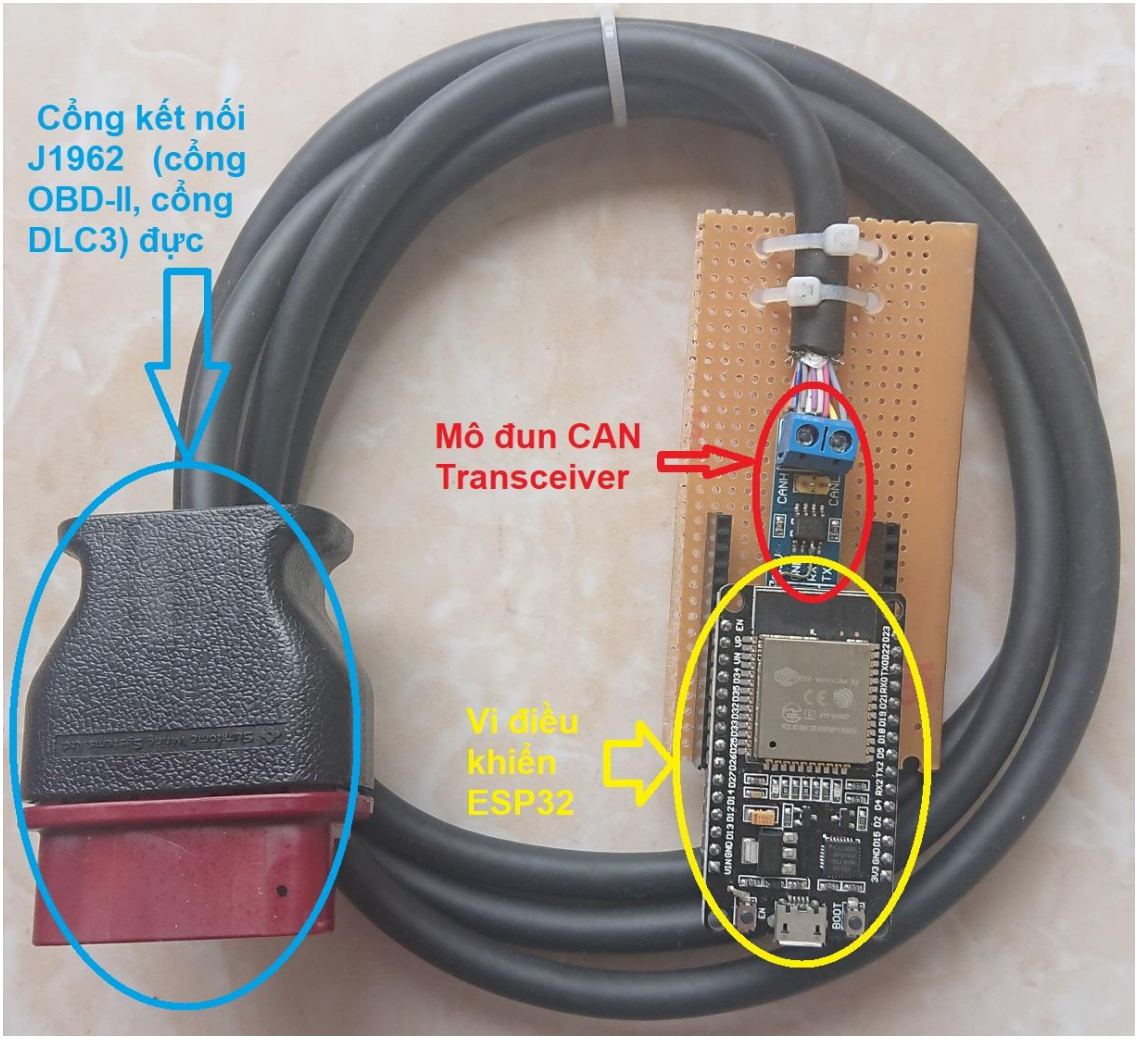
Bộ đọc dữ liệu OBD-II được kết nối vào mạng CAN-bus trên xe thông qua cổng kết nối J1962 (hay còn gọi là cổng OBD-II, cổng DLC3) cái được trang bị sẵn trên hầu như tất cả các xe đang lưu thông trên thị trường cũng như xe mới được bán ra. Tất cả các thiết bị dụng cụ chuyên dùng trong lĩnh vực chẩn đoán động cơ và ô tô hiện nay đều sử dụng cổng này để giao tiếp với ECU.

Bộ đọc dữ liệu OBD-II được thiết kế như là một node của mạng CAN-bus, do đó có cấu trúc gồm:

+ Mô đun CAN Transceiver.
+ Ngoại vi CAN Controller tích hợp trong vi điều khiển.
+ Ngoài ra còn có dây và cổng kết nối để dễ dàng kết nối vào cổng kết nối J1962 cái trang bị sẵn trên xe.



Hình 3.1: Sơ đồ bộ đọc dữ liệu OBD-II.



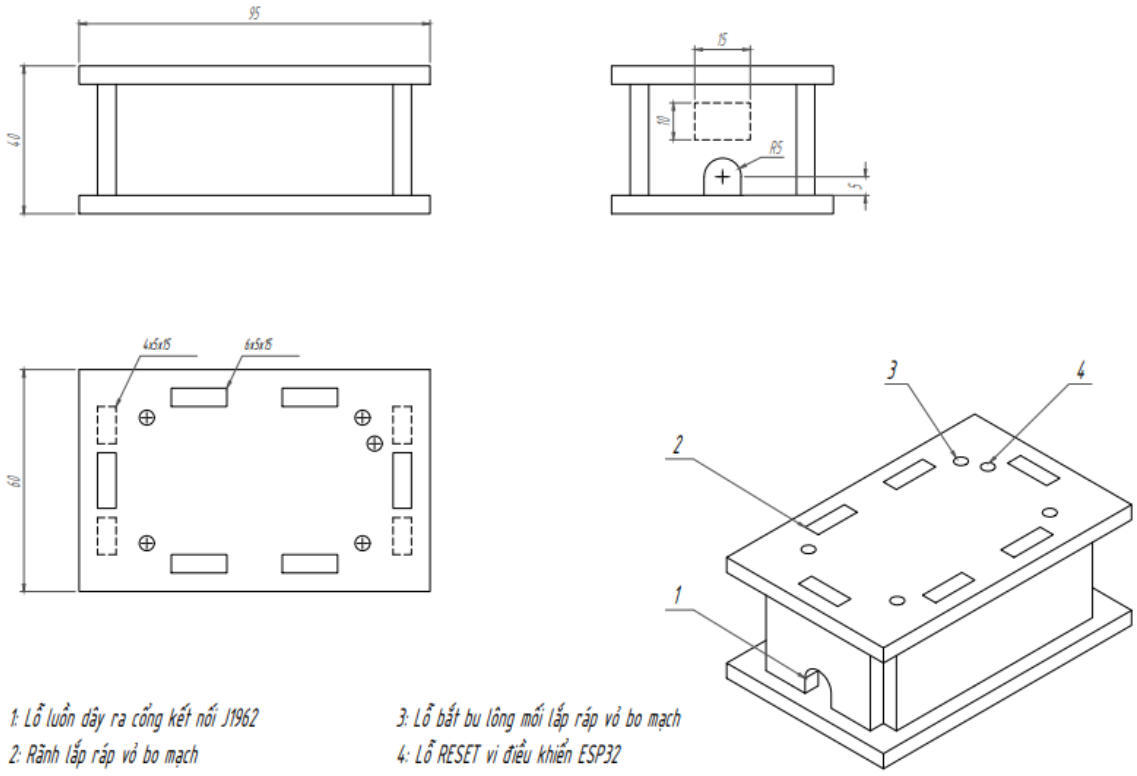
Hình 3.2: Bộ đọc dữ liệu OBD-II thực tế.

3.2. Vỏ bo mạch bộ đọc dữ liệu OBD-II

Vỏ bo mạch bộ đọc dữ liệu OBD-II được thiết kế bằng mica trong suốt, độ dày 5mm cho phép chống va đập, chống bụi và chống đụng chạm vào các chi tiết kim loại làm ngắn mạch các chân vi điều khiển trong quá trình sử dụng. Ngoài ra, lớp vỏ trong suốt còn cho phép quan sát đèn báo trên vi điều khiển ESP32.

Trên vỏ bo mạch của bộ đọc dữ liệu còn có 3 lỗ:

+ Lỗ luồn dây ra cổng kết nối J1962.
+ Lỗ gắn cáp USB để giao tiếp giữa bộ đọc dữ liệu với máy tính.
+ Lỗ nhấn nút RESET vi điều khiển ESP32 trong trường hợp bộ đọc dữ liệu bị treo.



Hình 3.3: Vỏ bo mạch bộ đọc dữ liệu OBD-II

CHƯƠNG 4: THIẾT KẾ KỸ THUẬT PHẦN MỀM

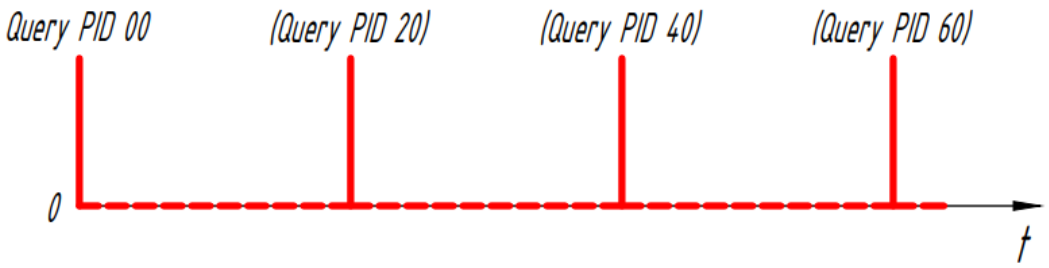
Phần mềm của bộ đọc dữ liệu OBD-II sử dụng vi điều khiển ESP32 được lập trình trên công cụ lập trình Arduino IDE, vi điều khiển thu thập các thông số vận hành của động cơ bằng cách giao tiếp với ECU thông qua giao thức CAN, chuẩn OBD-II.

Có 2 phần mềm riêng biệt được sử dụng cho bộ đọc dữ liệu OBD-II. Phần mềm thứ nhất sử dụng bộ PID tiêu chuẩn của Mode 01 (hiển thị dữ liệu hiện tại) của chuẩn OBD-II để scan xác định các PID dữ liệu mà chuẩn OBD-II có hỗ trợ trên mỗi dòng xe cụ thể. Phần mềm thứ 2 được sử dụng sau đó, có nạp các PID cần đọc mà xe có hỗ trợ để gửi truy vấn dữ liệu tới ECU. Thông qua giao thức CAN, bộ đọc dữ liệu OBD-II sử dụng khung dữ liệu tiêu chuẩn để giao tiếp với ECU:

+ Khung truy vấn dữ liệu: khung truyền mang có ID là ID của ECU (7DF), vùng dữ liệu chứa mã Mode (01) và PID của dữ liệu cần truy vấn, được vi điều khiển của bộ đọc dữ liệu OBD-II gửi lên CAN-bus.
+ Khung dữ liệu phản hồi: khung truyền mang ID theo tiêu chuẩn (7E8), vùng dữ liệu chứa mã Mode (41), PID của dữ liệu trong khung truyền và các byte dữ liệu do ECU gửi lên CAN-bus. Bộ đọc dữ liệu OBD-II kiểm tra khung dữ liệu nhận được, xác nhận ID của khung truyền và PID của dữ liệu cần đọc, tiến hành chuyển đổi thành thông số có nghĩa mà người dùng có thể sử dụng.

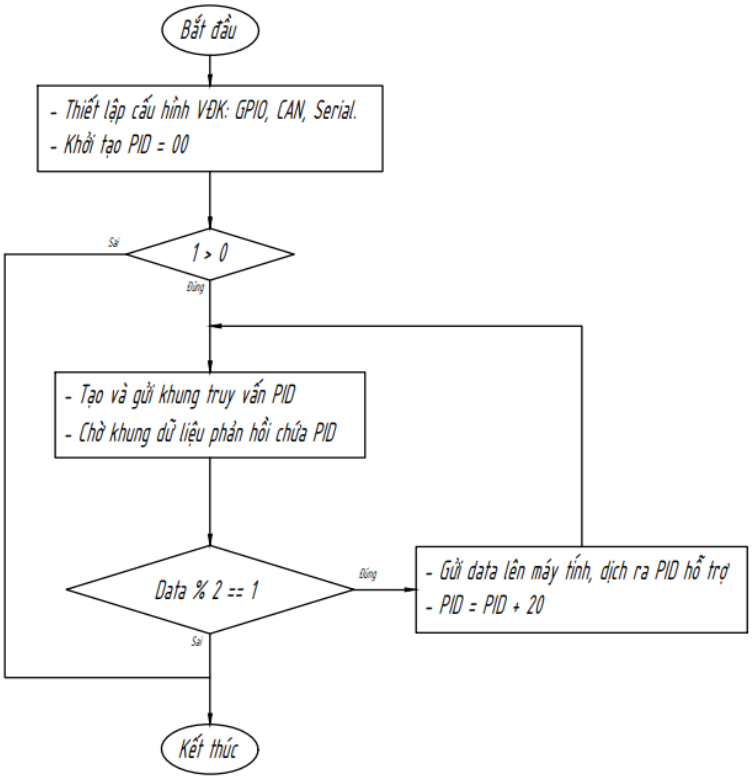
4.1. Phần mềm scan PID

4.1.1. Giản đồ thời gian chương trình scan PID



Hình 4.1: Giản đồ thời gian chương trình scan PID

4.1.2. Giải thuật chương trình



Hinh 4.2: Lưu đồ giải thuật chương trình scan PID

4.1.2.1. Khởi tạo biến, thiết lập cấu hình vi điều khiển

- PID = 00.
- GPIO:
  + Cấu hình chân output CAN TX, input CAN RX.
- CAN:
  + Cấu hình chân sử dụng làm CAN TX, CAN RX.
  + Cấu hình tốc độ mạng CAN (500kbps).
  + Cấu hình khung truyền: khung dữ liệu tiêu chuẩn với ID = 7DF, DLC = 8, data[0] = 2, data[1] = 01, data[2] = PID, data[3-7] = CC.
- Serial:
  + Tốc độ baudrate tương thích với phần mềm trên máy tính.

4.1.2.2. Tạo và gửi khung truy vấn, chờ khung phản hồi và tách data

- Tạo và gửi khung truy vấn:

+ data[2] = PID.

+ Gửi khung truy vấn dữ liệu.

- Chờ khung phản hồi, sau khi nhận được tiến hành kiểm tra:
  + ID = 7E8.
  + data[0] = 6, data[1] = 01, data[2] = PID.
  + data = data[3-6].
  + data%2 = 1 → PID = PID + 20.

4.1.2.3. Gửi data lên máy tính

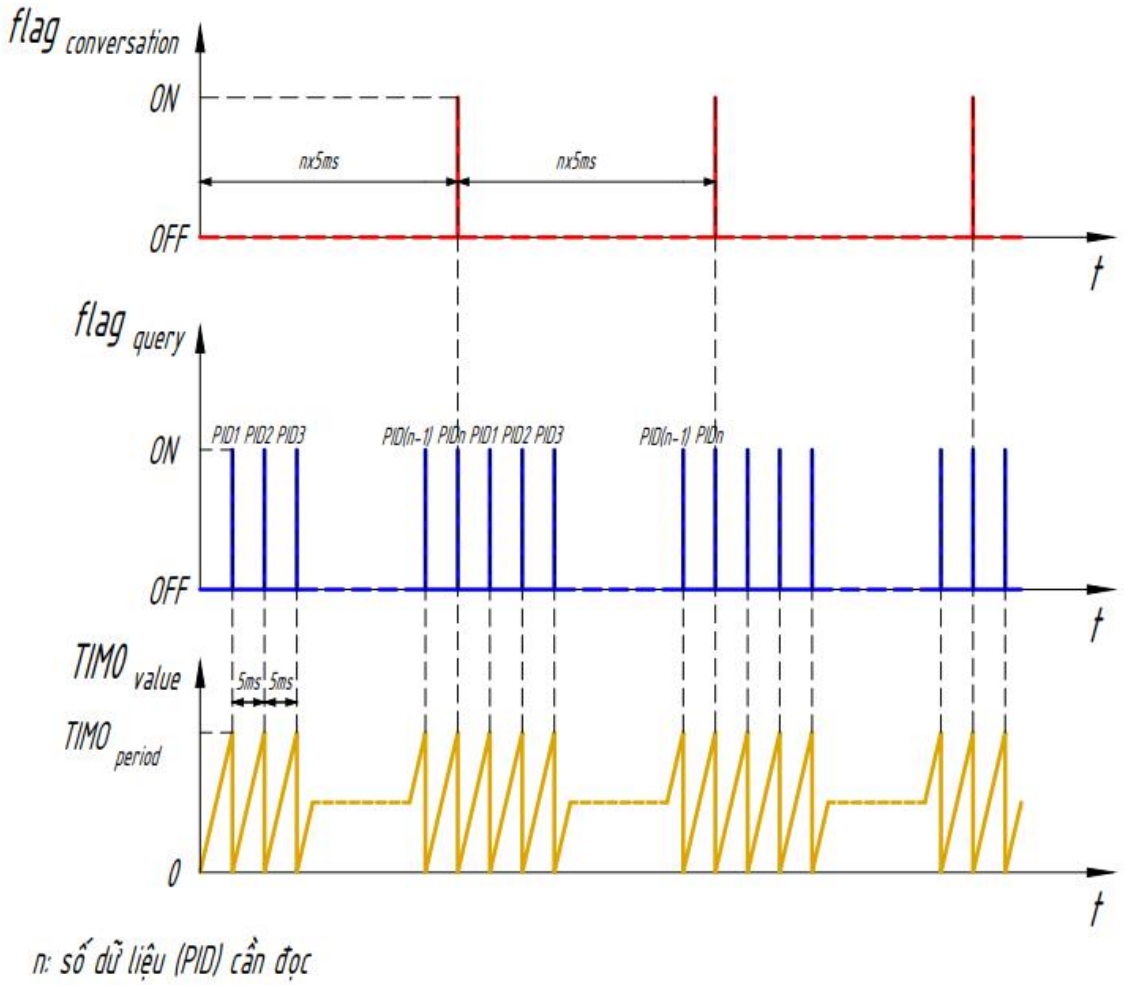
Gửi từng byte data[3-6] lên máy tính thông qua giao tiếp serial.

4.2. Phần mềm đọc dữ liệu OBD-II

Phần mềm đọc dữ liệu OBD-II sử dụng bộ Timer 0 của vi điều khiển ESP32 tạo ra các chu kỳ làm việc chuẩn bao gồm:

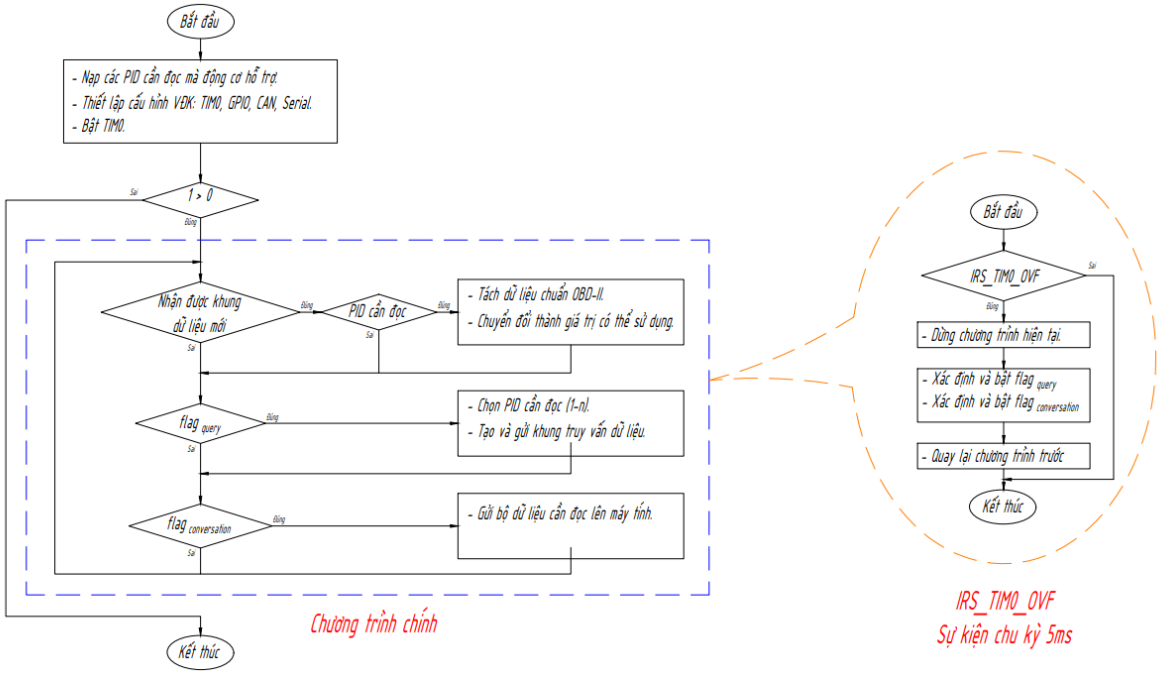
- Chu kỳ gửi khung truy vấn dữ liệu đến ECU vào mạng CAN-bus.
- Chu kỳ giao tiếp, gửi bộ dữ liệu thu thập lên máy tính.
- Ngoài ra, bộ đọc dữ liệu OBD-II còn liên tục kiểm tra và đọc khung dữ liệu phản hồi do ECU gửi lên mạng CAN-bus, tính toán ra giá trị của thông số mà người dùng có thể sử dụng thông qua các công thức tiêu chuẩn. Quá trình này được diễn ra bất cứ thời điểm nào vi điều khiển ESP32 nhận được khung dữ liệu có PID trùng với một trong số những PID cần đọc đã được lựa chọn.

4.2.1. Giản đồ thời gian chương trình đọc dữ liệu OBD-II



Hình 4.3: Giản đồ thời gian chương trình đọc dữ liệu OBD-II

4.2.2. Giải thuật chương trình



Hình 4.4: Lưu đồ giải thuật chương trình đọc dữ liệu OBD-II

4.2.2.1. Nạp PID, thiết lập cấu hình vi điều khiển

- Nạp các PID cần đọc và công thức tiêu chuẩn chuyển đổi dữ liệu ra thông số có thể sử dụng.

```
switch(rx_frame.data.u8[2])
{
  case 0x0C: //RPM
    Data[0] = (float)((rx_frame.data.u8[3]*256.0 + rx_frame.data.u8[4])/4.0);
    FlagFrame[0] = true;
    break;
  case 0x49: //PEDAL
    Data[1] = (float)((rx_frame.data.u8[3]*100.0)/255.0);
    FlagFrame[1] = true;
    break;
  case 0x11: //THROTTLE
    Data[2] = (float)(((rx_frame.data.u8[3]*100.0)/255.0));
    FlagFrame[2] = true;
    break;
  case 0x06: //STFT
    Data[3] = ((float)rx_frame.data.u8[3]/1.28 - 100);
    FlagFrame[3] = true;
    break;
  case 0x07: //LTFT
    Data[4] = ((float)rx_frame.data.u8[3]/1.28 - 100);
    FlagFrame[4] = true;
    break;
```

Hình 4.5: Một số PID hỗ trợ cho xe Mitsubishi Xpander và công thức tiêu chuẩn

- TIM0:
  - + Clock 80MHz.
  - + Prescaler = 80.
  - + Period = 5000
    - ⇨ TIM0 tràn với chu kỳ 5ms.
- GPIO:
  - + Cấu hình chân output CAN TX, input CAN RX.
- CAN:
  - + Cấu hình chân sử dụng làm CAN TX, CAN RX.
  - + Cấu hình tốc độ mạng CAN (500kbps).
  - + Cấu hình khung truyền: khung dữ liệu tiêu chuẩn với ID = 7DF, DLC = 8, data[0] = 2, data[1] = 01, data[2] = PID, data[3-7] = CC.
- Serial:
  - + Tốc độ baudrate tương thích với phần mềm trên máy tính.

4.2.2.2. Tạo và gửi khung truy vấn dữ liệu

+ data[2] = PID.
+ Gửi khung truy vấn dữ liệu.
+ Lựa chọn PID cho chu kỳ gửi truy vấn tiếp theo.

4.2.2.3. Đọc khung dữ liệu phản hồi

Khi nhận được khung dữ liệu từ mạng CAN-bus, tiến hành kiểm tra:

+ ID = 7E8.
+ data[1] = 1. data[1] = 01,
+ data[2] bằng một trong số những PID đã được nạp.
+ Sử dụng công thức tiêu chuẩn để chuyển đổi data thành thông số có thể sử dụng, lưu vào mảng dữ liệu gửi lên máy tính.

4.2.2.4. Giao tiếp với máy tính

Tạo khung dữ liệu gửi lên máy tính để kiểm tra và tránh lỗi trong quá trình gửi.

```
void creat_send_frame()
{
  send_frame = "";
  send_frame += 'N';
  send_frame += 'A';
  send_frame += 'M';
}
```

```
void send_num_float(float num_float, char id[2])
{
  send_frame += id[0];
  send_frame += id[1];
  byte* p_num_float = (byte*)&num_float;
  send_frame += (char)*(p_num_float + 3);
  send_frame += (char)*(p_num_float + 2);
  send_frame += (char)*(p_num_float + 1);
  send_frame += (char)*p_num_float;
  send_frame += ';';
}
```

```
void _send()
{
  send_frame_size = send_frame.length() + 5;
  send_frame += (char) send_frame_size;
  send_frame += 'H';
  send_frame += 'O';
  send_frame += 'A';
  send_frame += '\n';
  Serial.print(send_frame);
}
```

Hình 4.6: Tạo khung dữ liệu để tránh lỗi trong giao tiếp Serial

CHƯƠNG 5: CHẠY THỬ NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

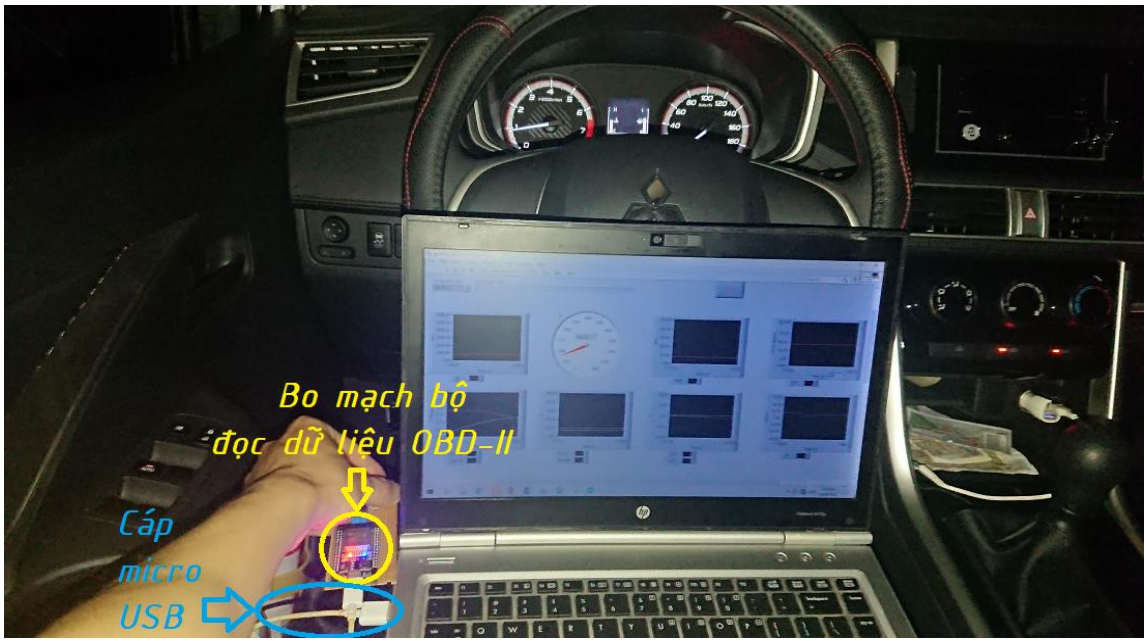
5.1. Chạy thử

5.1.1. Chuẩn bị

- Kết nối cổng J1962 đực của bộ đọc dữ liệu ODB-II với cổng J1962 cái sẵn có trên xe.

Hình 5.1: Kết nối chuẩn SAE J1962

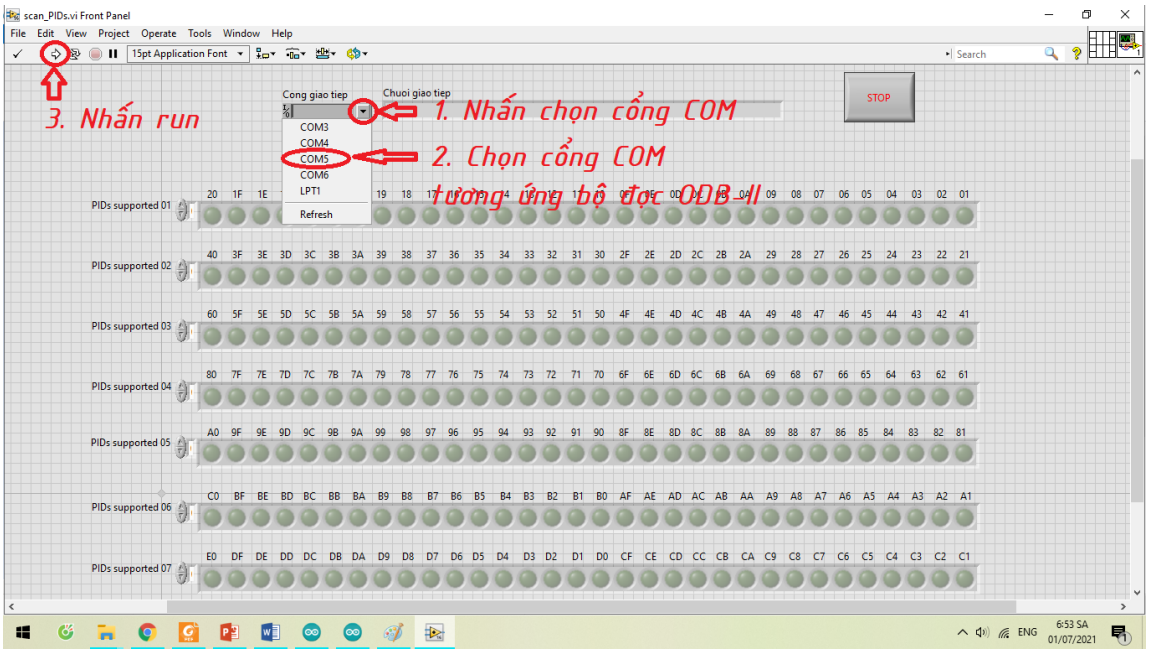
- Kết nối bộ đọc dữ liệu OBD-II với máy tính thông qua cáp micro USB.



Hình 5.2: Kết nối USB giữa bộ đọc dữ liệu OBD-II với máy tính

5.1.2. Scan PID

- Nạp phần mềm scan PID cho bộ điều khiển.
- Khởi động phần mềm giao tiếp với người dùng trên máy tính, chọn cổng COM kết nối đến bộ đọc dữ liệu OBD-II và nhấn Run.



Hình 5.3: Thao tác kết nối phần mềm giao tiếp trên máy tính với bộ đọc OBD-II

Các PID được hỗ trợ trên xe Mistsubishi Xpander 2020

5.1.3. Đọc dữ liệu OBD-II

- Nạp các PID cần đọc và nhập công thức tính toán tiêu chuẩn và phần mềm đọc dữ liệu OBD-II:
  + Từ kết quả scan PID, nhận thấy xe Mitsubishi Xpander 2020 có hỗ trợ một số PID : 01, 03, 04, 05, 06, 07, 0B, 0C, 0D, 0E, 0F, 11, 13, 14, 15…
  + Lựa chọn các dữ liệu thu thập:

    Nhiệt độ nước làm mát động cơ (PID 05).

    Short term fuel trim (PID 06).

    Long term fuel trim (PID 07).

    Áp suất tuyệt đối đường ống nạp (PID 0B).

    Tốc độ động cơ (PID 0C).

    Góc đánh lửa sớm (PID 0E).

    Vị trí van bướm ga (PID 11).

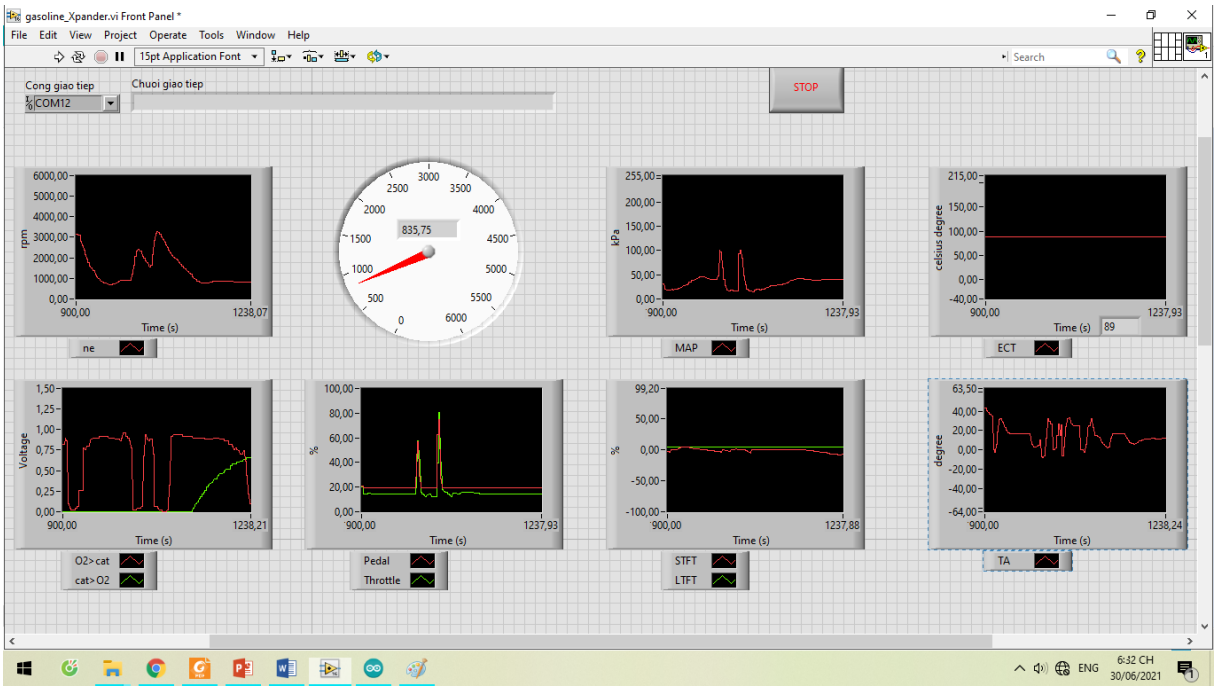
    Điện áp cảm biến oxy trước bầu xúc tác (PID 14).

    Điện áp cảm biến oxy sau bầu xúc tác (PID 15).

    Vị trí bàn đạp ga (PID 49).

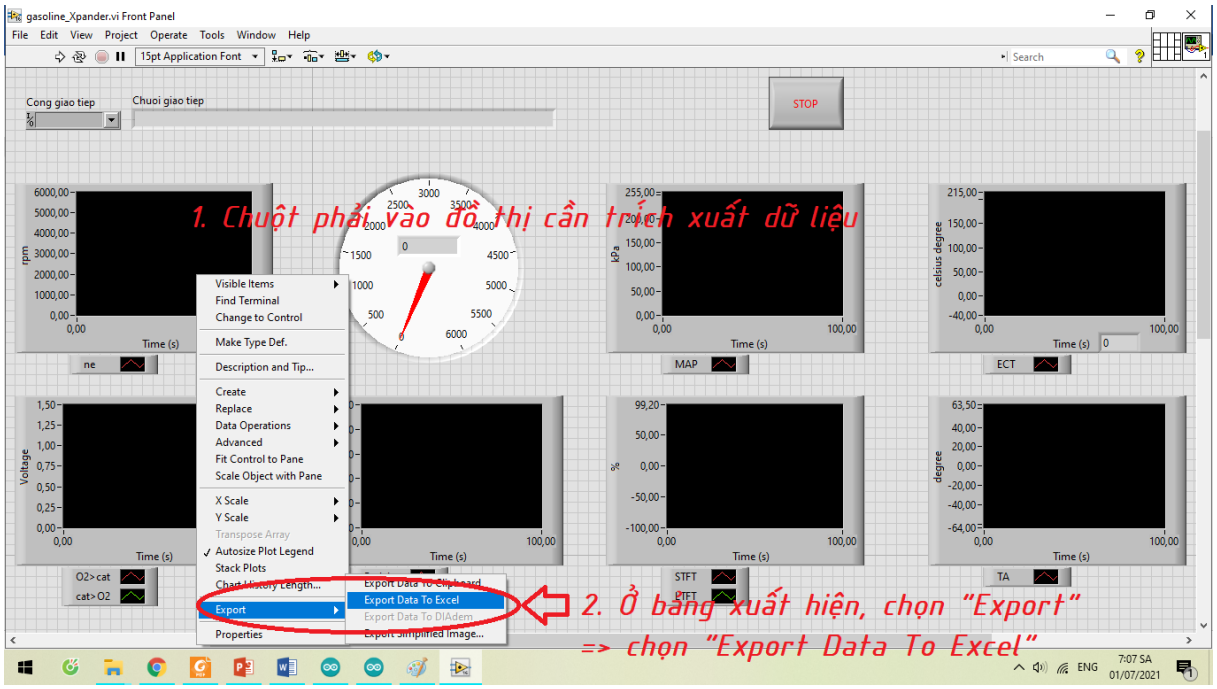
- Khởi động phần mềm giao tiếp với người dùng trên máy tính, chọn cổng COM kết nối đến bộ đọc dữ liệu OBD-II và nhấn Run.

- Dữ liệu bắt đầu được ghi nhận.



Hình 5.4: Đồ thị các số liệu thu thập được theo thời gian

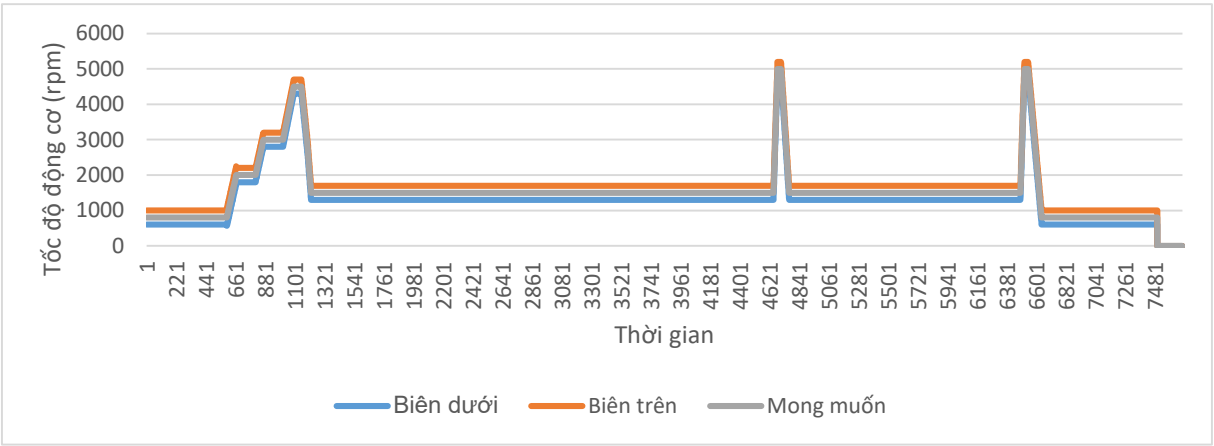
- Xuất dữ liệu ra dạng bảng tính.



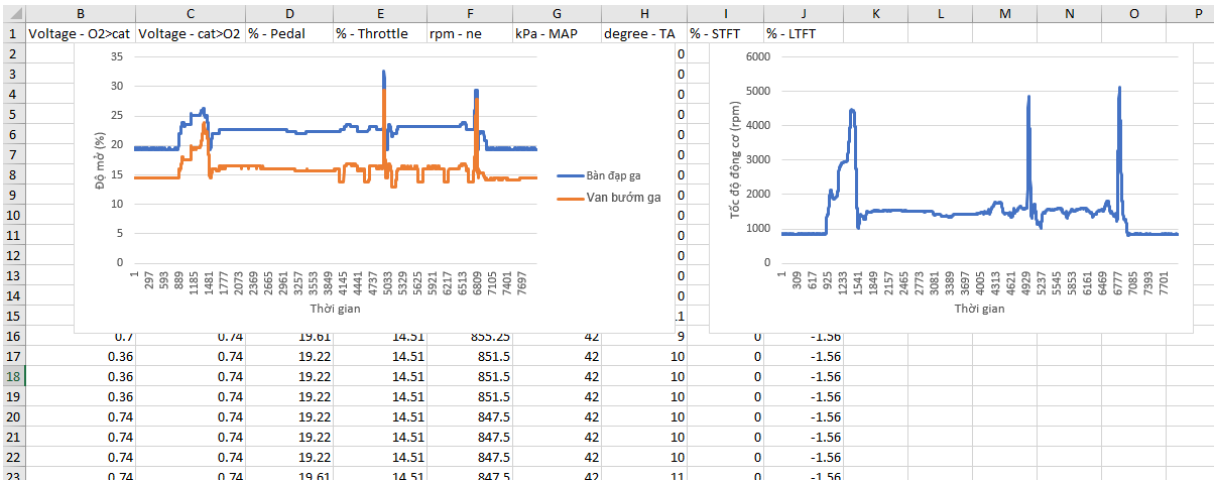
Hình 5.5: Thao tác trích xuất dữ liệu ra dạng bảng tính

5.2. Đánh giá kết quả

Theo dõi đồ thị theo thời gian của các dữ liệu cần đọc trên giao diện giao tiếp người dùng, so sánh với vận hành thực tế của động cơ nhận thấy dữ cập nhật theo thời gian thực, bộ đọc dữ liệu OBD-II hoạt động tốt và ít bị trễ.



Hình 5.6: Chu trình thử

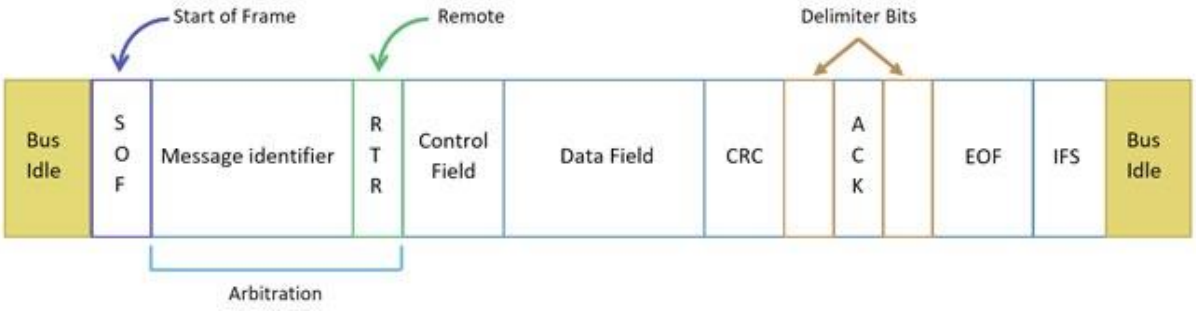


Voltage - O2>cat	Voltage - cat>O2	% - Pedal	% - Throttle	rpm - ne	kPa - MAP	degree - TA	% - STFT	% - LTFT
0.7	0.74	19.61	14.51	855.25	42	9	0	-1.56
0.36	0.74	19.22	14.51	851.5	42	10	0	-1.56
0.36	0.74	19.22	14.51	851.5	42	10	0	-1.56
0.36	0.74	19.22	14.51	851.5	42	10	0	-1.56
0.74	0.74	19.22	14.51	847.5	42	10	0	-1.56
0.74	0.74	19.22	14.51	847.5	42	10	0	-1.56
0.74	0.74	19.22	14.51	847.5	42	10	0	-1.56
0.74	0.74	19.61	14.51	847.5	42	11	0	-1.56

Hình 5.7: Dữ liệu thu được khi chạy theo chu trình thử

PHỤ LỤC 1: KHUNG TRUYỀN CỦA GIAO THỨC CAN-BUS

Một khung truyền của giao thức CAN-bus có dạng sau:



Hình 6.1: Cấu trúc khung truyền của giao thức CAN-bus

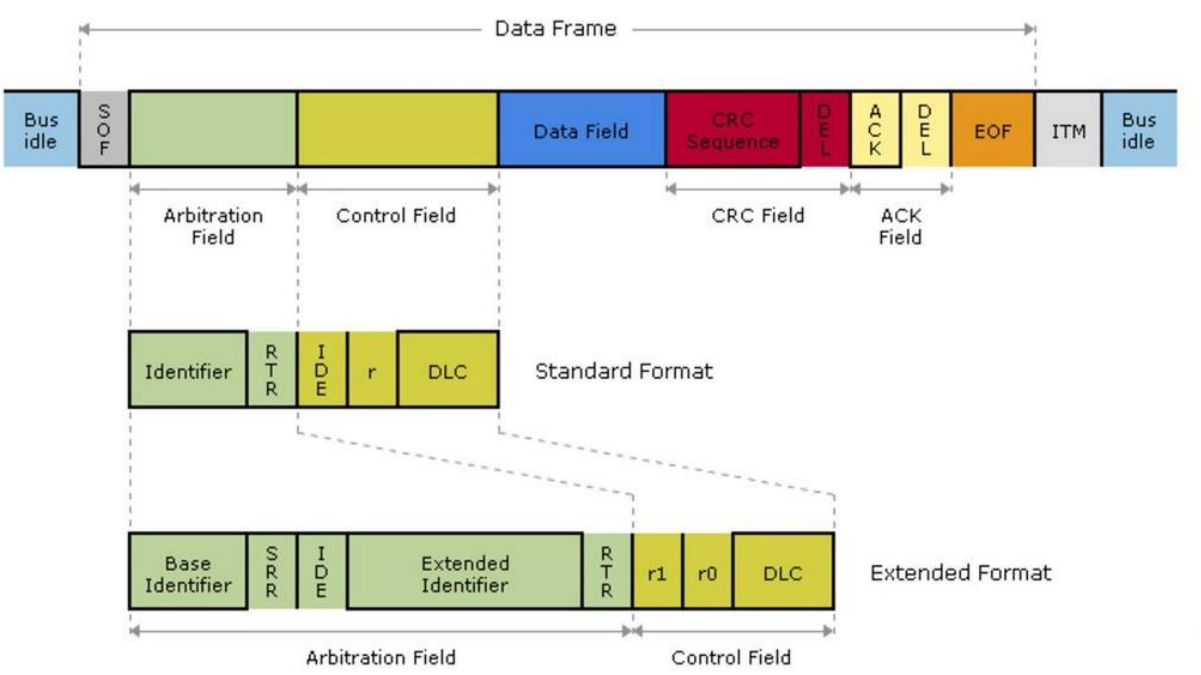
Giao thức CAN định nghĩa 4 loại khung truyền:

- Khung dữ liệu (data frame) dùng khi node muốn truyền dữ liệu tới node khác.
- Khung yêu cầu (remote frame) dùng để yêu cầu node khác truyền khung dữ liệu tương ứng với ID trong khung yêu cầu.
- Khung quá tải (overload frame) và khung lỗi (error frame) dùng trong việc xử lý lỗi.

Trạng thái bus idle là khi không có node nào trên bus phát đi khung truyền (không có node nào phát đi trạng thái dominant); khi đó, bus ở trạng thái recessive.

6.1. Khung dữ liệu

Khung dữ liệu dùng để truyền đi một tin nhắn chứa nội dung (dữ liệu). Có 2 dạng khung dữ liệu: khung dữ liệu tiêu chuẩn (standard frame) và khung dữ liệu mở rộng (extended frame).



Hình 6.2: Cấu trúc 2 dạng khung dữ liệu của giao thức CAN

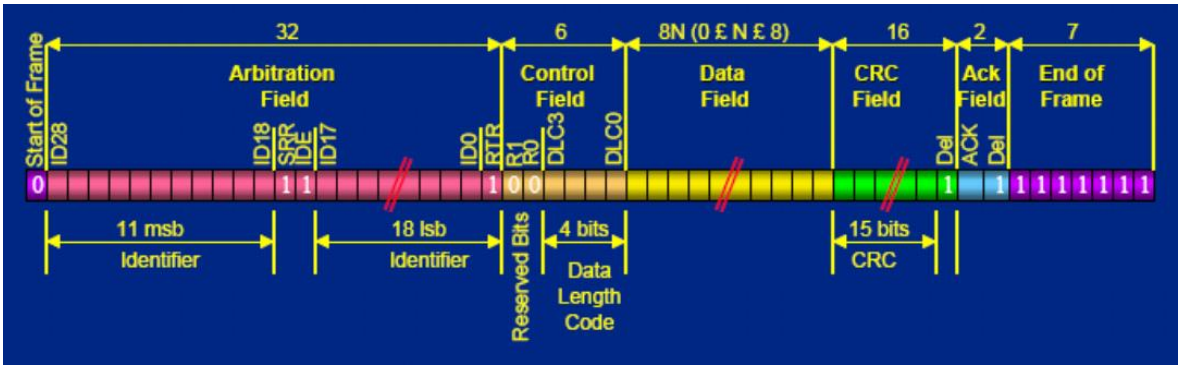
6.1.1. Khung dữ liệu tiêu chuẩn

- Khung dữ liệu tiêu chuẩn được bắt đầu bằng một bit bắt đầu khung truyền (SOF – Start Of Frame) luôn ở trạng thái dominant.
- Tiếp theo là 11 bit ID (Identifier).
- Tiếp theo là 1 bit RTR (Remote Transmit Request) để phân biệt khung dữ liệu và khung yêu cầu, nếu bit này bằng dominant nghĩa là khung dữ liệu, nếu bằng recessive nghĩa là khung yêu cầu.
- Tiếp đến là 1 bit IDE (Identifier Extension) để phân biệt giữa khung dữ liệu tiêu chuẩn (khi bit này bằng dominant) và khung dữ liệu mở rộng (khi bit này bằng recessive).
- Tiếp theo là 1 bit r luôn ở trạng thái dominant.
- Tiếp đến là 3 bit DLC (Data Length Control) cho biết số lượng byte dữ liệu của khung dữ liệu.
- Tiếp đến là từ 0 đến 8 byte dữ liệu.
- Tiếp đến là 15 bit CRC (Cyclic Redundancy Code) và 1 bit CRC DEL (Cyclic Redundancy Code Delimiter – là bit luôn ở trạng thái recessive).

- Tiếp đến là 1 bit ACK (Acknowledge) và 1 bit DEL (Delimiter – là bit luôn ở trạng thái recessive).
- Tiếp theo là 7 bit kết thúc khung truyền (EOF – End Of Frame) luôn ở trạng thái recessive.
- Cuối cùng là khoảng cách tối thiểu giữa 2 khung truyền (IFS – Inter Frame Space).

6.1.2. Khung dữ liệu mở rộng

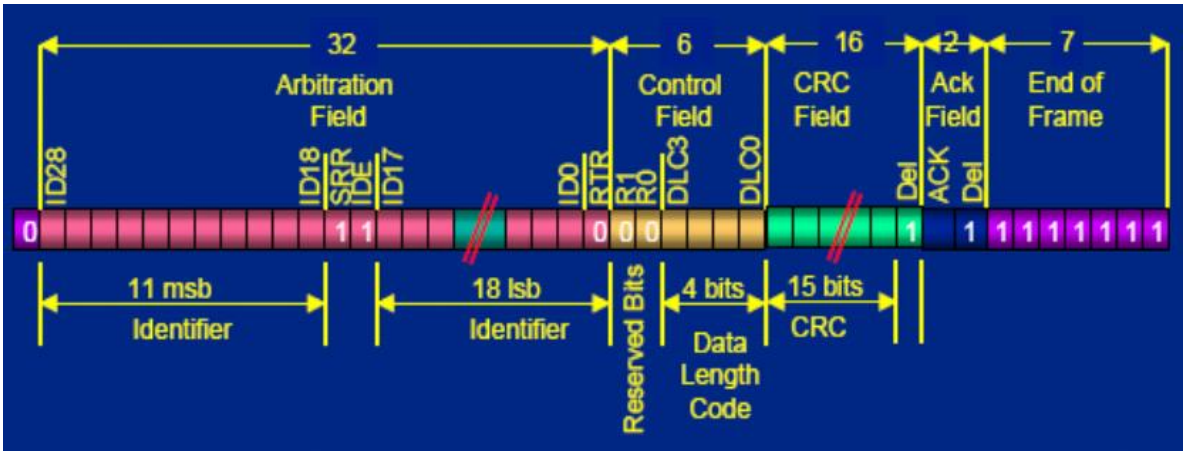
Khung dữ liệu mở rộng có cấu trúc gần giống khung dữ liệu tiêu chuẩn tuy nhiên có tới 29 bit ID.



Hình 6.3: Cấu trúc khung dữ liệu mở rộng

6.2. Khung yêu cầu

Khung yêu cầu dùng để yêu cầu một node khác gửi khung dữ liệu chứa dữ liệu có ID trùng với ID trong khung yêu cầu. khung yêu cầu có cấu trúc gần giống khung dữ liệu, tuy nhiên có vùng DLC bằng 0 và không có vùng dữ liệu.



Hình 6.4: Cấu trúc khung yêu cầu

PHỤ LỤC 2: OBD-II STANDARD

OBD-II PIDs (On-Board Diagnostics Parameter Identifications) là những ID mà các công cụ chẩn đoán chuyên dùng dùng để yêu cầu dữ liệu OBD-II từ xe ô tô.

Tiêu chuẩn SAE J1979 định nghĩa một danh sách nhiều OBD-II PID. Tất cả các xe lưu thông và được bán ra tại Mỹ phải được hỗ trợ một tập con của danh sách này, chủ yếu liên quan đến vấn đề tiêu chuẩn phát thải của xe ở mỗi tiểu bang. Các nhà sản xuất cũng có thể định nghĩa các PID bổ sung riêng cho các dòng xe khác nhau. Mặc dù không bắt buộc, nhiều dòng xe máy cũng hỗ trợ các OBD-II PID.

7.1. OBD-II Mode

Ngoài các PID tiêu chuẩn, chuẩn OBD-II còn quy định 10 chế độ chẩn đoán tiêu chuẩn (mode hay service) theo chuẩn SAE J1979, sử dụng kết hợp với các PID tiêu chuẩn để giao tiếp với ECU.

Bảng 7.1: Mô tả 10 chế độ chẩn đoán tiêu chuẩn ODB-II SAE J1979

Mode (hex)	***Mô tả***
01	Hiển thị thông số tức thời
02	Hiển thị khung dữ liệu cố định
03	Hiển thị mã lỗi chẩn đoán đã được lưu trữ
04	Xoá mã lỗi chẩn đoán và các giá trị được lưu trữ
05	Kết quả kiểm tra, giám sát cảm biến oxy
06	Kết quả kiểm tra, giám sát cá hệ thống, chi tiết khác
07	Hiển thị mã lỗi chẩn đoán đang chờ xử lý (mới nhất)
08	Kiểm soát hoạt động của các hệ thống on-board
09	Truy vấn thông tin xe
0A	Hiển thị tất cả mã lỗi chẩn đoán từng có (đã xoá)

7.2. OBD-II PIDs tiêu chuẩn

Bộ đọc dữ liệu OBD-II sử dụng trong đề tài chỉ sử dụng Mode 01 để đọc dữ liệu OBD-II sử dụng cho huấn luyện mạng chẩn đoán và chẩn đoán.

Dưới đây là một số PID chính và công thức tiêu chuẩn để tính ra thông số vận hành mà người dùng có thể trực tiếp sử dụng từ dữ liệu mà bộ đọc dữ liệu OBD-II nhận được

từ ECU. Vùng dữ liệu trong mỗi khung dữ liệu phản hồi từ ECU có độ dài từ 1-4 byte.

A								B								C								D							
A7	A6	A5	A4	A3	A2	A1	A0	B7	B6	B5	B 4	B3	B2	B1	B0	C7	C 6	C5	C4	C3	C2	C1	C0	D7	D6	D5	D4	D3	D2	D1	D0

Hình 7.1: Vùng dữ liệu mà bộ đọc dữ liệu OBD-II nhận được

Bảng 7.2: PID tiêu chuẩn Mode 01 và cách chuyển đổi dữ liệu giao tiếp về thông số có nghĩa

PID (hex)	***Số byte dữ liệu trả về***	***Mô tả***	***Giá trị nhỏ nhất***	***Giá trị lớn nhất***	***Đơn vị***	***Công thức***
00	4	Các PID có hỗ trợ				Mỗi bit [A7...D0] tương đương 1 PID từ [01...20]
04	1	Tải động cơ	0	100	%	100*A/255
05	1	Nhiệt độ nước làm mát động cơ	-40	215	^{0}C	A – 40
06	1	Short term fuel trim – bank 1	-100 (giảm nhiên liệu: hoà khí quá giàu)	100 (thêm nhiên liệu: hoà khí quá nghèo)	%	100*A/128 – 100
07	1	Long term fuel trim – bank 1				
08	1	Short term fuel trim – bank 2				
09	1	Long term fuel trim – bank 2				
0A	1	Áp suất nhiên liệu	0	765	kPa	3*A
0B	1	Áp suất chân không đường	0	255	kPa	A

		ống nạp				
0C	2	Tốc độ động cơ	0	16383.75	rpm	(256*A + B)/4
0D	1	Tốc độ xe	0	255	km/h	A
0E	1	Góc đánh lửa sớm (phun dầu sớm)	-64	63.5	^{0} trước ĐCT	A/2 – 64
0F	1	Nhiệt độ khí nạp	-40	215	^{0}C	A – 40
10	2	Lưu lượng khí nạp	0	655.35	g/s	(256*A + B)/100
11	1	Vị trí bướm ga	0	100	%	100*A/255
14	1	Điện áp cảm biến oxy trước bầu xúc tác	0	1.275	V	A/200
15	1	Điện áp cảm biến oxy sau bầu xúc tác				
1F	2	Thời gian tính từ khi động cơ khởi động	0	65535	s	256*A + B
20	4	Các PID có hỗ trợ				Mỗi bit [A7...D0] tương đương 1 PID từ [21...40]
23	2	Áp suất thanh rail nhiên liệu (diesel, xăng phun nhiên liệu trực tiếp	0	655350	kPa	10*(256*A + B)
2C	1	Tỉ lệ hồi lưu khí thải	0	100	%	100*A/255

40	4	Các PID có hỗ trợ				Mỗi bit [A7...D0] tương đương 1 PID từ [41...60]
49	1	Vị trí bàn đạp ga	0	100	%	100*A/255
60	4	Các PID có hỗ trợ				Mỗi bit [A7...D0] tương đương 1 PID từ [61...80]
7C	2	Nhiệt độ bộ lọc DPF			^{0}C	((256*A + B)/10) – 40
80	4	Các PID có hỗ trợ				Mỗi bit [A7...D0] tương đương 1 PID từ [81...A0]
A0	4	Các PID có hỗ trợ				Mỗi bit [A7...D0] tương đương 1 PID từ [A1...C0]
C0	4	Các PID có hỗ trợ				Mỗi bit [A7...D0] tương đương 1 PID từ [C1...E0]

7.3. Khung truyền chuẩn OBD-II – khung dữ liêu tiêu chuẩn giao thức CAN-bus

Chuẩn OBD-II sử dụng định dạng khung dữ liệu tiêu chuẩn chứa 11 bit ID của giao thức CAN-bus để giao tiếp với ECU. Các công cụ chẩn đoán chuyên dùng kết nối với mạng CAN-bus thông qua cổng kết nối J1962; sử dụng Mode và các PID phù hợp để gửi khung truy vấn dữ liệu (query) tới ECU và kiểm tra PID trong khung dữ liệu phản hồi (response) để quyết định có đọc dữ liệu hay không.

7.3.1. Khung truy vấn dữ liệu

Khung truy vấn dữ liệu sử dụng sử dụng ID 7DF (hex – tương đương 11 bit binary) và 8byte data.

Bảng 7.3: Cấu trúc vùng dữ liệu của khung truy vấn dữ liệu

	Byte							
Loại PID	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***
Tiêu chuẩn SAE	Số byte có nghĩa liền sau: 2	Mode Ví dụ: 01 = yêu cầu truy vấn dữ liệu tức thời	PID Ví dụ: 05 = yêu cầu truy vấn nhiệt độ nước làm mát động cơ	Không được sử dụng Tiêu chuẩn ISO 15765-2 đề xuất là CC (hex)				
Đặc trưng trên dòng xe cụ thể	Số byte có nghĩa liền sau: 3	Mã dịch vụ Ví dụ: 22 = yêu cầu truy vấn dữ liệu nâng cao	PID Ví dụ: 4980 (hex)	Không được sử dụng Tiêu chuẩn ISO 15765-2 đề xuất là CC (hex)				

7.3.2. Khung dữ liệu phản hồi

Xe sẽ gửi khung dữ liệu phản hồi lên CAN-bus khi nhận được khung truy vấn dữ liệu, ID của khung dữ liệu sẽ phụ thuộc vào mô đun gửi phản hồi. Ví dụ ECU điều khiển động cơ sẽ gửi các phản hồi với ID của khung dữ liệu CAN là 7E8 (hex). Các mô đun khác, ví dụ như bộ điều khiển pin của xe Prius sẽ gửi phản hồi ở ID 7E9, 7EA hoặc 7EB.

Bảng 7.3: Cấu trúc vùng dữ liệu của khung truy dữ liệu phản hồi

	Byte							
CAN ID	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***
Tiêu chuẩn SAE: 7E8 7E9 7EA …	Số byte có nghĩa liền sau: 3 đến 6	Mode + 40 (hex) Ví dụ: 41 = phản hồi dữ liệu tức thời	PID Ví dụ: 05 = yêu cầu truy vấn nhiệt độ nước làm mát động cơ	Giá trị của dữ liệu: byte 0	Giá trị của dữ liệu: byte 1	Giá trị của dữ liệu: byte 2	Giá trị của dữ liệu: byte 3	Không sử dụng Có thể là 00 hoặc 55
Đặc trưng trên dòng xe cụ thể: 7E8 hoặc 8 + ID của mô đun phản hồi	Số byte có nghĩa liền sau: 4 đến 7	Mã dịch vụ + 40 (hex) Ví dụ: 62 phản hồi cho truy vấn bằng mã dịch vụ 22	PID Ví dụ: 4980 (hex)		Giá trị của dữ liệu: byte 0	Giá trị của dữ liệu: byte 1	Giá trị của dữ liệu: byte 2	Giá trị của dữ liệu: byte 3

TÀI LIỆU THAM KHẢO

[1] wikipedia.org

[2] Texas Insrtruments, Introduction to the Controller Area Network (CAN), Application Report SLOA101B–August 2002–Revised May 2016

[3] SAE J1979-DA, Digital Annex of E/E Diagnostic Test Modes

[4] ISO 15765-2:2016, Road vehicles – Diagnostic communication over Controller Area Network (DoCAN) – Part 2: Transport protocol and network layer services

[5] ISO 15765-4:2016, Road vehicles – Diagnostic communication over Controller Area Network (DoCAN) – Part 4: Requirements for emissions-related systems