David Foanene - ESP Code Sniffer

Introducere

Acest proiect își propune să dezvolte un sistem de comunicație CAN utilizând un microcontroler ESP32 și un shield MCP2515. Scopul principal este de a intercepta și afișa mesaje CAN transmise pe magistrala CAN.

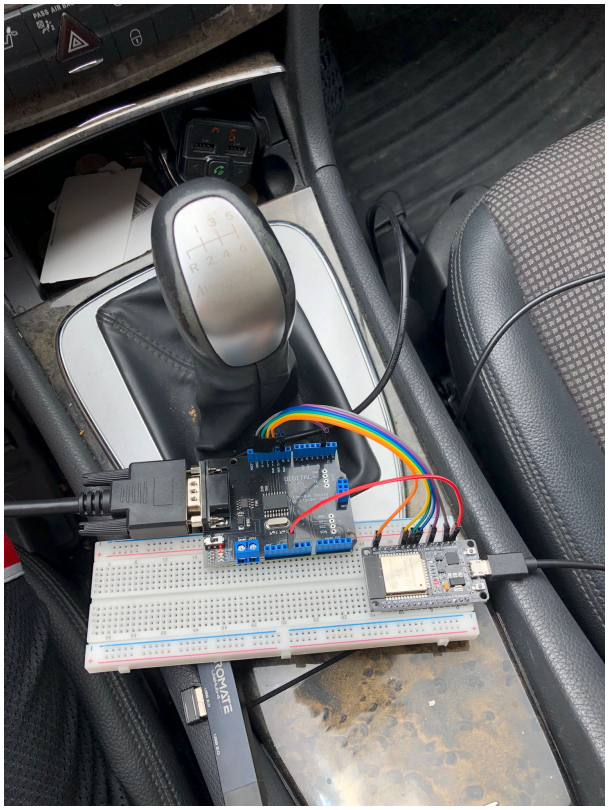
Descriere generală

Proiectul va utiliza un microcontroler ESP32 împreună cu un CAN shield pentru a citi și interpreta codurile de la modulele de control ale vehiculului. ESP32 va fi programat să stabilească o conexiune CAN (Controller Area Network) cu sistemul de diagnoză a bordului (OBD) al vehiculului.

Hardware Design

Componenete :

```
1. Conector OBD2 - DB9.
2. CAN Shield (MCP 2551 + MCP 2515).
3. ESP32.
```



Software Design

```
#include <SPI.h>
#include <mcp2515.h>
```

```
MCP2515 mcp2515(5); // Chip Select pin
```

```
void setup() {
   Serial.begin(115200);
   while (!Serial);
```

```
   SPI.begin();
   mcp2515.reset();
   mcp2515.setBitrate(CAN_500KBPS, MCP_8MHZ);

   // Dezactivează modul Loopback pentru a interacționa cu alte noduri CAN
   mcp2515.setNormalMode();
```

```
   Serial.println("MCP2515 Initialized");
}
```

```
void loop() {
   sendRandomCANMessage();
```

```
    struct can_frame canMsg;
```

```
    // Verifică pentru mesaje primite
    MCP2515::ERROR result = mcp2515.readMessage(&canMsg);
    if (result == MCP2515::ERROR_OK) {
    Serial.print("Received message with ID: 0x");
    Serial.print(canMsg.can_id, HEX);
    Serial.print(" Data: ");
    for (int i = 0; i < canMsg.can_dlc; i++) {
       Serial.print(canMsg.data[i], HEX);
       Serial.print(" ");
    }
    Serial.println();
    } else {
       Serial.print("Error Receiving Message: ");
       Serial.println(result);
    }
```

```
    // Verifică starea MCP2515
    uint8_t status = mcp2515.getStatus();
    Serial.print("MCP2515 Status: ");
    Serial.println(status, HEX);
```

```
    delay(1000); // O întârziere mai lungă între cicluri
  }
```

```
  void sendRandomCANMessage() {
    struct can_frame canMsg;
```

```
    // Generează un ID random între 0x100 și 0x7FF (ID standard de 11 biți)
    canMsg.can_id = random(0x100, 0x7FF);

    // Setează o lungime random pentru datele trimise (de la 1 până la 8)
    canMsg.can_dlc = random(1, 9);
```

```
    // Generează date random
    for (int i = 0; i < canMsg.can_dlc; i++) {
       canMsg.data[i] = random(0, 256); // Fiecare byte de date este între
0x00 și 0xFF
    }
```

```
    // Trimite mesajul
    MCP2515::ERROR result = mcp2515.sendMessage(&canMsg);
    if (result == MCP2515::ERROR_OK) {
        Serial.println("Message Sent Successfully!");
        Serial.print("Sent message with ID: 0x");
        Serial.print(canMsg.can_id, HEX);
        Serial.print(" Data: ");
        for (int i = 0; i < canMsg.can_dlc; i++) {
            Serial.print(canMsg.data[i], HEX);
```

```
            Serial.print(" ");
        }
        Serial.println();
    } else {
        Serial.print("Error Sending Message: ");
        Serial.println(result);
    }
```

```
    delay(100); // Mică întârziere pentru a asigura recepția mesajului
}
```

Rezultate Obţinute

Am reusit sa trimit si sa primesc mesaje pe magistrala CAN.
https://youtube.com/shorts/y7ZGwtQQasY?feature=share

Concluzii

Utilizarea CAN este mai grea decat am crezut la inceput, fiind nevoie de cunostiinte avansate in acest domeniu. Conectarea la magistrala CAN a unei masini a esuat din motive necunoscute. Am folosit comunicarea SPI intre shield ul CAN si ESP32.

Download

proiect_pm_david_foanene.zip

Bibliografie/Resurse

Componente achizitionate :

```
1. ESP32 - Sigmanortec [[https://www.sigmanortec.ro/]]
2. Shield CAN V1.2 - Optimus Digital [[https://www.optimusdigital.ro/ro/]]
3. Cablu OBD2 - DB9 - RoboFun [[https://www.robofun.ro/]]
```

Biblioteci folosite :

```
1. SPI.h
2. mcp2515.h
```

Export to PDF