ANKARA UNIVERSITY
ELECTRICAL AND ELECTRONICS ENGINEERING

FINAL REPORT

Controlling the Vehicle Door Locks using Reed Relays

Embedded Systems - EEE3330

31.05.2023

Lecturer: Yüksek Mühendis Atilla YENİDOĞAN

Student: Nurullah MERTEL

Contents

Introduction

The goal of this final project is to develop a comprehensive system for CAN bus communication using the MCP2515 CAN controller and the Arduino platform. The CAN (Controller Area Network) protocol is widely used in various industries for real-time communication between microcontrollers and devices, offering robustness, reliability, and high-speed data transfer capabilities.

In this project, we will explore the capabilities of the MCP2515 CAN controller and implement a transmitter and receiver system. The transmitter will send CAN messages containing data packets, while the receiver will listen for these messages and display the received data on an LCD screen.

By the end of this project, We aim to gain a deeper understanding of CAN bus communication, enhance my programming skills with the Arduino platform, and demonstrate a functional transmitter and receiver system that can be utilized in various applications such as automotive systems, industrial automation, and more.

Through this project, I hope to showcase the potential of CAN bus communication and provide a foundation for further exploration and development in this field.

Method

The hardware setup involved connecting the MCP2515 CAN controller module to the Arduino board using the SPI interface. The necessary libraries, including "SPI" "mcp2515" were installed. Also, the library "LiquidCrystal_I2C" in order to interface with the LCD screen and display the received data were installed. I referred to the Appendix A: Code Samples for the Arduino coding.

The transmitter code was developed to initialize the MCP2515 controller, create CAN messages, and send them using the sendMessage() function. The receiver code initialized the MCP2515 controller and the LCD screen, continuously checked for incoming CAN messages using the readMessage () function, and displayed the received data on the LCD screen.

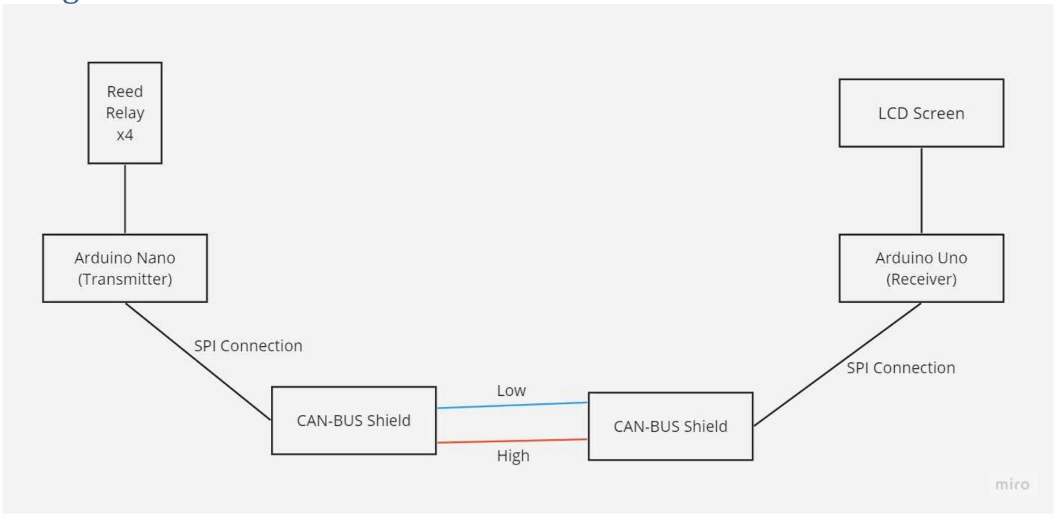
Components

The components I used in this project:

- 2 Canbus Shields v1.2
- 1 Arduino Nano
- 1 Arduino Uno R3
- 4 Reed Relays
- Breadboard
- Jumper Cables
- 9 V Battery
- LCD screen

Total Cost: 717 TL

Block Diagram

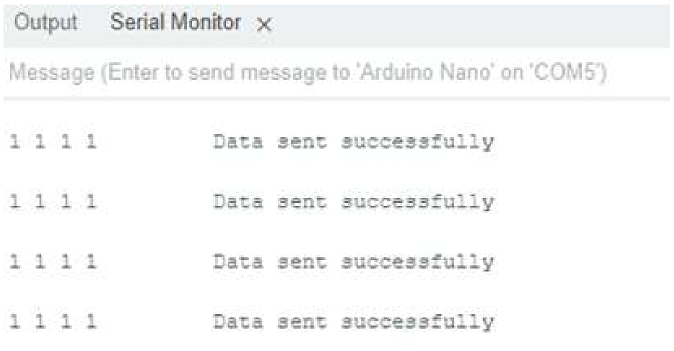


Results

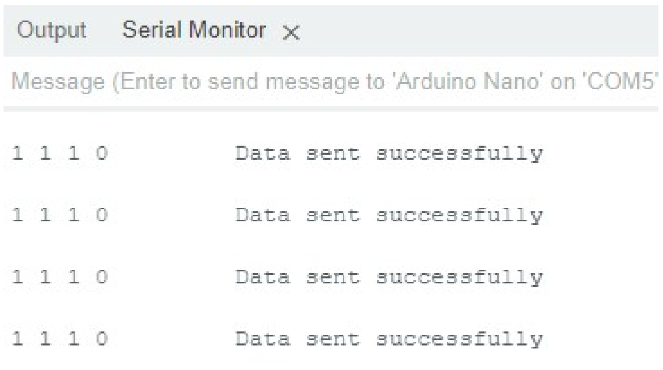
The system successfully transmitted CAN messages containing the data frame from the transmitter to the receiver. The receiver accurately received the messages and displayed the received data on the LCD screen. Experimental data and measurements on Serial Monitor and LCD Screen showed reliable and consistent performance in terms of message transmission and reception.

Serial Monitor

Serial Monitor allows us to view data from serial communication in ASCII format. Each character in the ASCII table has a corresponding numeric value. In serial communication this value is sent and the Serial Monitor displays the characters corresponding to these numbers.



```
1 1 1 1          Data sent successfully

1 1 1 1          Data sent successfully

1 1 1 1          Data sent successfully

1 1 1 1          Data sent successfully
```



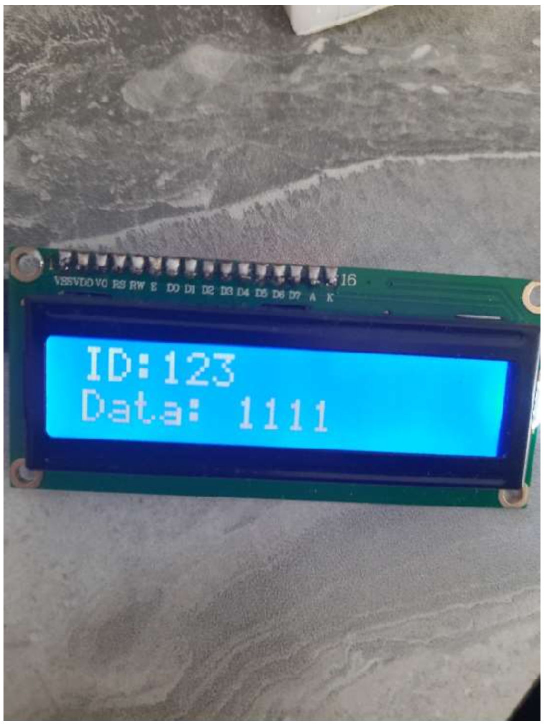
```
1 1 1 0          Data sent successfully

1 1 1 0          Data sent successfully

1 1 1 0          Data sent successfully

1 1 1 0          Data sent successfully
```

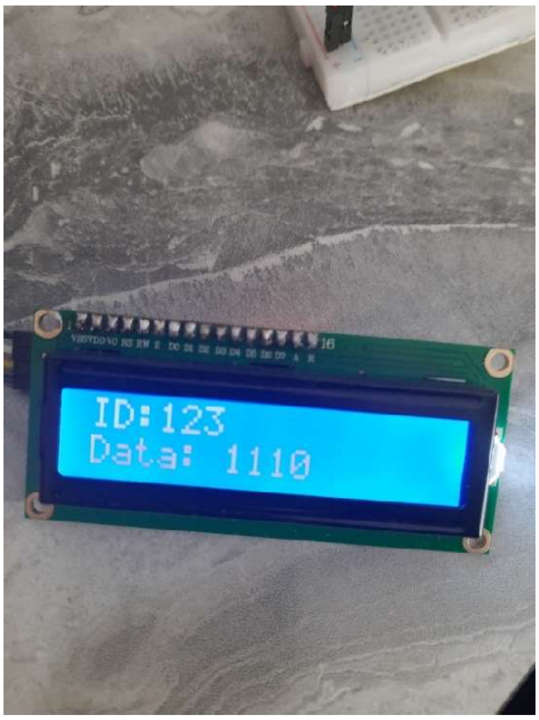
When all magnetic switches are open, we read the value of ‘1’ in the entire data frame depending on the arduino code I prepared. On the other hand, if any of the magnetic switches are closed due to the zooming of the magnet, ‘0’ value is read on the related data.

LCD Screen

The display has an LED backlight and can display 32 ASCII characters in two lines, 16 characters per line. The structure of LCDs consists of different layers as seen in the picture above.

Panels are formed when layers of LCD come together. The working logic of the panels, in its simplest form, is that the specialized cells on it are shaped by the ion layer and an image is formed by electric current.





We observed the same results on the LCD Screen as we got on the serial monitor. When all magnetic switches are open, we read the value of '1' in the entire data frame depending on the arduino code I prepared. On the other hand, if any of the magnetic switches are closed due to the zooming of the magnet, '0' value is read on the related data.

Discussion

The results indicate that the developed system effectively enables CAN bus communication using the MCP2515 CAN controller and the Arduino platform. The system's robustness and reliability were demonstrated through successful transmission and reception of CAN messages. However, limitations were identified, such as the need for proper termination and consideration of longer cable lengths in real-world applications. The comparison with existing solutions showcased the cost-effectiveness and flexibility of the Arduino-based implementation.

Conclusion

In conclusion, the final project successfully developed a comprehensive system for CAN bus communication using the MCP2515 CAN controller and the Arduino platform. The project demonstrated the ability to transmit and receive CAN messages, displaying the received data on an LCD screen. The system's performance and functionality provide a solid foundation for further exploration and application in areas such as automotive systems, industrial automation, and more.

References

- https://wiki.seeedstudio.com/CAN-BUS_Shield_V1.2/
- https://cdn-reichelt.de/documents/datenblatt/A300/113030021_01.pdf
- https://www.robotistan.com/
- https://www.robo90.com/
- https://github.com/autowp/arduino-mcp2515

Appendix A: Code Samples

A.1 Transmitter Code

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

const int cs_pin = 10; // Chip select pin for MCP2515 CAN controller
MCP2515 can(cs_pin); // Create an MCP2515 object with the chip select pin

void setup() {
  Serial.begin(9600);
  while (!Serial); // Wait for serial connection

  can.reset();
  can.setBitrate(CAN_500KBPS); // Initialize CAN controller with a baud rate
of 500kbps
  can.setNormalMode();

  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  pinMode(4, INPUT_PULLUP);
  pinMode(5, INPUT_PULLUP);
}

void loop() {
  can_frame canMsg1;
  canMsg1.can_id  = 0x123;  // Replace with the ID of your message buffer
  canMsg1.can_dlc = 4;      // Length of message data in bytes

  for(int i=0; i<canMsg1.can_dlc; i++) {
    canMsg1.data[i] = digitalRead(i+2); // Read the digital input values from
pins 2, 3, 4, 5 and store them in the message data
    Serial.print(canMsg1.data[i]); // Print each data value to the serial
monitor
    Serial.print(" ");
  }

  if (can.sendMessage(&canMsg1) == MCP2515::ERROR_OK) {
    Serial.println("\tData sent successfully\n"); // If the message is sent
successfully, print a success message to the serial monitor
  } else {
    Serial.println("\tError sending data\n"); // If there is an error sending
the message, print an error message to the serial monitor
  }

  delay(500); // Wait for 0.5 second before sending the next frame
}
```

A.2 Receiver Code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <mcp2515.h>

MCP2515 can(10); // Create an MCP2515 object with chip select pin 10
LiquidCrystal_I2C lcd(0x27, 16, 2); // Address of I2C LCD, number of columns, 
number of rows

void setup() {
  Serial.begin(9600);

  // Initialize MCP2515 chip
  can.reset();
  can.setBitrate(CAN_500KBPS);
  can.setNormalMode();

  // Initialize I2C LCD
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.print("Merhaba"); // Display "Merhaba" on the LCD
  delay(1000);
  lcd.print(" Dunya"); // Display " Dunya" on the LCD
  delay(1500);
  lcd.clear();
}

void loop() {
  can_frame canMsg1;
  if (can.readMessage(&canMsg1) == MCP2515::ERROR_OK) {
    // Print message data to I2C LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("ID:");
    lcd.print(canMsg1.can_id, HEX); // Print the message ID in hexadecimal 
format

    lcd.setCursor(0, 1);
    lcd.print("Data: ");

    for(int i=0; i<canMsg1.can_dlc; i++) {
      lcd.print(canMsg1.data[i]); // Print each byte of the message data on 
the LCD
    }
  }
}
```