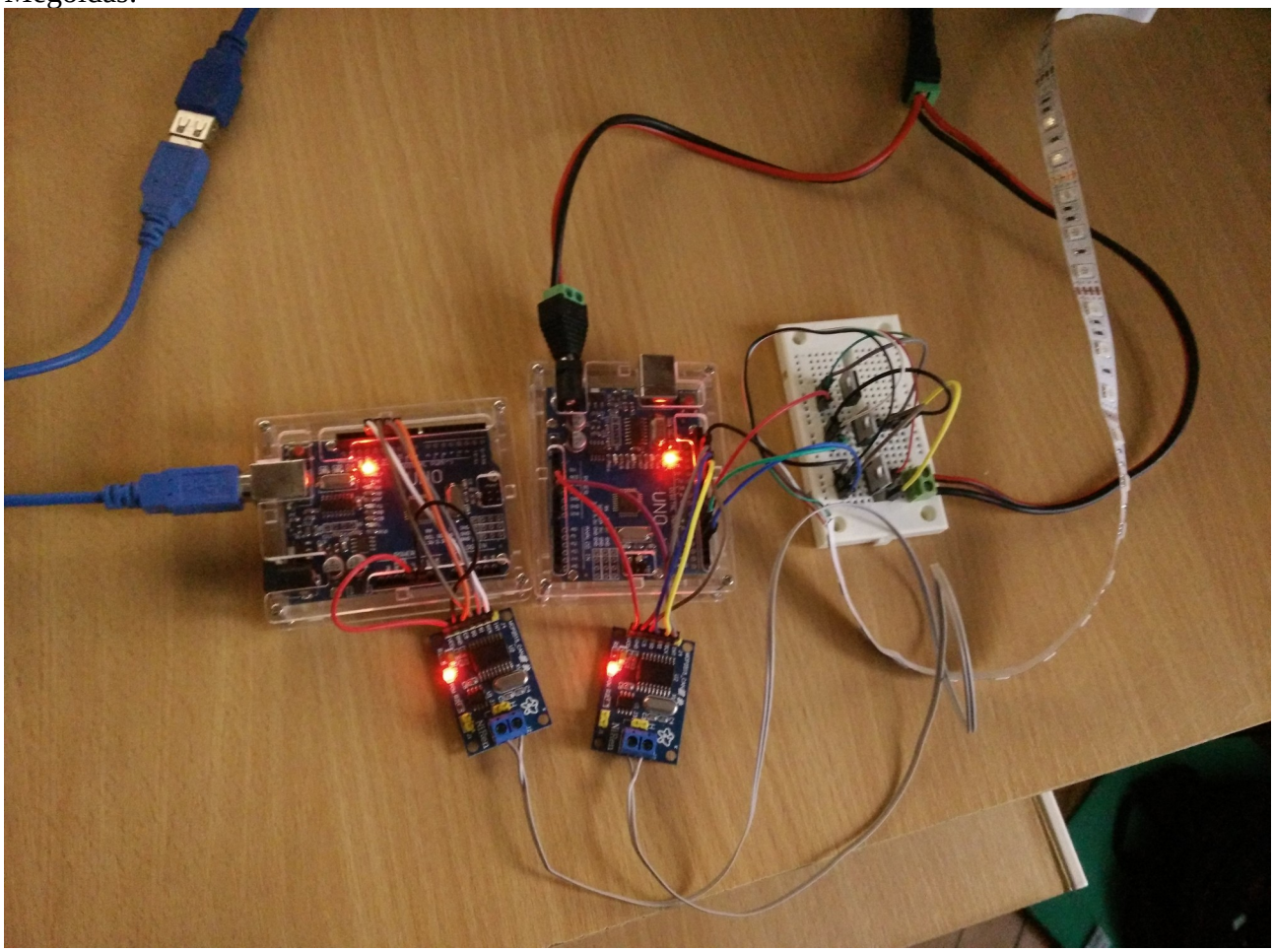
Feladat: A feladat SPI-s CAN illesztő modul használatával vezérlés megvalósítása. Egy LED lámpát kell vezérelni egy másik eszközről.

Megoldás:



Eszközök:
- 2 db arduino uno
- 2db mcp21515_can
- 3db irlz24n mosfet
- 12v tápegység
- rgb led szallag

SPI-vel csatlakoztatom az mcp21515_can modulokat az arduino unok-hoz.
Az mcp21515_can modulokat csatlakoztatom egymáshoz.
Mosfetekkel hajtom meg a ledeket az arduino pwm kimeneteivel.

A mcp21515_can modulhoz a következő könytárat használtam:
https://github.com/Seeed-Studio/CAN_BUS_Shield

Fogadó eszköz amire led szalag van kötve 8 byte-os adat-ot tud fogadni:
1. byte: vörös intenzitása (0-255)
2. byte: zöld intenzitása (0-255)
3. byte: kék intenzitása (0-255)
4. byte: led bekapcsolási ideje másodperceben (1-255) ha 0 az érték akkor folyamatosan bekapcsolás

5. byte: led villogás 100 ms-ben (1-255) ha 0 az érték akkor nincs villogás
6. byte nem használt
7. byte nem használt
8. byte nem használt

A küldő a következő parancsokat ismeri:
red szám - piros intenzitását beállítja
green szám - zöld intenzitását beállítja
blue szám - kék intenzitását beállítja
blink szám - villogás intervallumát állítja be beállítja
on szám - világítás idejét állítja be
send - elküldi a beállításokat
nino - felváltva villogtatja a szalagot kéken és pirosan
offnino - kikapcsolja a piros és kék villogást

Fogadó kódja:

```
#include <SPI.h>
#include "mcp_can.h"

const int spiCSPin = 10;

const int redPin = 6;
const int greenPin = 5;
const int bluePin = 3;

int blinkTime = 0;
boolean blink = false;
long blinkMillis = 0;


int onTime = 0;
boolean on = false;
boolean constantOn = false;
long onMillis = 0;

int redIntesitivity = 0;
int greenIntesitivity = 0;
int blueIntesitivity = 0;

boolean ledState = false;

boolean changed = false;

MCP_CAN CAN(spiCSPin);

void setup() {
  Serial.begin(115200);
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  analogWrite(redPin, redIntesitivity);
  analogWrite(greenPin, greenIntesitivity);
  analogWrite(bluePin, blueIntesitivity);
  while (CAN_OK != CAN.begin(CAN_500KBPS)) {
    Serial.println("CAN BUS Init Failed");
    delay(100);
  }
  Serial.println("CAN BUS  Init OK!");
}


void loop() {
  unsigned char len = 0;
  unsigned char buf[8];

  if (CAN_MSGAVAIL == CAN.checkReceive()) {
    CAN.readMsgBuf(&len, buf);

    unsigned long canId = CAN.getCanId();

    Serial.println("-----------------------------");
    Serial.print("Data from ID: 0x");
    Serial.println(canId, HEX);

    for (int i = 0; i < len; i++) {
      Serial.print(buf[i]);
      Serial.print("\t");
    }
```

```
    Serial.println();
    if (len > 4) {
      redIntesitivity = buf[0];
      greenIntesitivity = buf[1];
      blueIntesitivity = buf[2];

      onTime = buf[3];
      constantOn = onTime == 0 ? true : false;

      blinkTime = buf[4];
      blink = blinkTime == 0 ? false : true;
      on = true;
      analogWrite(redPin, redIntesitivity);
      analogWrite(greenPin, greenIntesitivity);
      analogWrite(bluePin, blueIntesitivity);
      onMillis = millis();
      ledState = false;
    }
  }

  unsigned long currentMillis = millis();

  if (on && !constantOn) {
    if (currentMillis - onMillis > (onTime * 1000)) {
      analogWrite(redPin, 0);
      analogWrite(greenPin, 0);
      analogWrite(bluePin, 0);
      ledState = false;
      on = false;
    }
  }

  if (blink && (on || constantOn))
  {
    if (currentMillis - blinkMillis > (blinkTime * 100)) {
      blinkMillis = currentMillis;
      if (ledState) {
        analogWrite(redPin, 0);
        analogWrite(greenPin, 0);
        analogWrite(bluePin, 0);
        ledState = false;
      }
      else {
        analogWrite(redPin, redIntesitivity);
        analogWrite(greenPin, greenIntesitivity);
        analogWrite(bluePin, blueIntesitivity);
        ledState = true;
      }

    }
  }



}
```

Küldő kódja:

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

const int spiCSPin = 10;
int ledHIGH    = 1;
int ledLOW     = 0;

boolean redblue = false;

boolean usenino = false;
long blinkMillis = 0;

char buf[80];
unsigned char stmp[8] = {0, 0, 0, 0, 0, 0, 0, 0};

MCP_CAN CAN(spiCSPin);

void setup()
{
  Serial.begin(115200);

  while (CAN_OK != CAN.begin(CAN_500KBPS))
  {
    Serial.println("CAN BUS init Failed");
    delay(100);
  }
  Serial.println("CAN BUS Shield Init OK!");
}

int readline(int readch, char *buffer, int len) {
  static int pos = 0;
  int rpos;

  if (readch > 0) {
    switch (readch) {
      case '\r': // Ignore CR
        break;
      case '\n': // Return on new-line
        rpos = pos;
        pos = 0;  // Reset position index ready for next time
        return rpos;
default:
        if (pos < len - 1) {
          buffer[pos++] = readch;
          buffer[pos] = 0;
        }
    }
  }
  return 0;
}



void loop()
{

  if (readline(Serial.read(), buf, 80) > 0) {
    String string = String(buf);
    Serial.println(string);
    if (string.startsWith("red"))
    {
      Serial.println(string.substring(3).toInt());
```

```
    stmp[0] = string.substring(3).toInt();
  }
  else if (string.startsWith("green"))
  {
    Serial.println(string.substring(5).toInt());
    stmp[1] = string.substring(5).toInt();
  }
  else if (string.startsWith("blue"))
  {
    Serial.println(string.substring(4).toInt());
    stmp[2] = string.substring(4).toInt();
  }
  else if (string.startsWith("on"))
  {
    Serial.println(string.substring(2).toInt());
    stmp[3] = string.substring(2).toInt();
  }
  else if (string.startsWith("blink"))
  {
    Serial.println(string.substring(5).toInt());
    stmp[4] = string.substring(5).toInt();
  }
  else if (string.startsWith("send"))
  {
    Serial.println("OK");
    CAN.sendMsgBuf(0x43, 0, 8, stmp);
    usenino = false;
  }
  else if (string.startsWith("nino"))
  {
    Serial.println("OK");
    usenino = true;
  }
     else if (string.startsWith("offnino"))
  {
    Serial.println("OK");
    usenino = false;
    for (int i = 0; i < 8; i++)
        stmp[i] = 0;
    CAN.sendMsgBuf(0x43, 0, 8, stmp);
  }
}

if (usenino)
{
  unsigned long currentMillis = millis();
  if (currentMillis - blinkMillis > 500) {
    blinkMillis = currentMillis;
    if (!redblue)
    {
      for (int i = 0; i < 8; i++)
        stmp[i] = 0;
      stmp[2] = 0;
      stmp[0] = 50;
      CAN.sendMsgBuf(0x43, 0, 8, stmp);
      redblue = true;
    }
    else
    {
      for (int i = 0; i < 8; i++)
        stmp[i] = 0;
      stmp[2] = 50;
      stmp[0] = 0;
      CAN.sendMsgBuf(0x43, 0, 8, stmp);
```

```
            redblue = false;
        }

    }
  }

}
```