Distantainatorul

Student: Radu Octavian, 333CC

Introducere

Proiectul presupune realizarea unei masinute controlate prin intermediul unui accelerometru, avand la baza comunicarea a doua placute Arduino cu ajutorul a doua module bluetooth, care masoara distante cu ajutorul unui senzor cu ultrasunet.

Acest proiect are scopul de a integra cunostiintele dobandite la laboratoarele de PM. De asemenea, poate fi util in masurarea de distante in locuri inaccesibile utilizatorilor.

Descriere generală

Mod de functionare : Prin folosirea accelerometrului se realizeaza calculul directiei de mers a masinutei. Aceste date sunt trimise prin intermediul Bluetooth, masinuta trimitand inapoi valoarea distantei pe care o masoara pana la cel mai apropiat obstacol in directia de mers. Aceasta distanta este afisata pe un ecran OLED conectat la placuta Arduino care serveste ca si telecomanda.

Schema bloc:

Hardware Design

Piese folosite:

- 2 X Arduino Uno R3 ATmega328P
- 2 X Modul Bluetooth HC-05
- 1 X Ecran OLED 0.96”
- 1 X Senzor Ultrasunete HC SR-04P 3-5.5V
- 1 X Modul giroscopic si accelerometru 3 axe GY-521
- 1 X Kit sasiu Smart Car 4WD
- 1 X Arduino Shield L293D

Schema telecomanda:

Schema masina:

Software Design

Mediu de dezvoltare: Arduino IDE

Librarii folosite:

- AFMotor.h
- Arduino.h
- SPI.h
- Wire.h
- Adafruit_GFX.h
- Adafruit_SSD1306.h

Cod telecomanda:

```
#include <Arduino.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C
#define GY_ADDRESS 0x68

constexpr int16_t X_THRESHOLD = 10000;
constexpr int16_t Y_THRESHOLD = 8000;

constexpr unsigned char MODE_STATIONARY = 30;
constexpr unsigned char MODE_FORWARD = 0;
constexpr unsigned char MODE_BACKWARD = 1;
constexpr unsigned char MODE_LEFT = 2;
constexpr unsigned char MODE_RIGHT = 3;


Adafruit_SSD1306 display(OLED_RESET);
int currentMode = 30;
int16_t accelerometer_x, accelerometer_y, accelerometer_z; // variables for
accelerometer raw data
int16_t gyro_x, gyro_y, gyro_z; // variables for gyro raw data
int16_t temperature; // variables for temperature data
```

```
float distance;
char tmp_str[7]; // temporary variable used in convert function

char* convert_int16_to_str(int16_t i) { // converts int16 to string.
Moreover, resulting strings will have the same length in the debug monitor.
  sprintf(tmp_str, "%6d", i);
  return tmp_str;
}

void setup() {
  Serial.begin (38400);
  display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);

  Wire.begin();
  Wire.beginTransmission(GY_ADDRESS);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
}

bool below_X() {
  return -X_THRESHOLD > accelerometer_x;
}

bool above_X() {
  return X_THRESHOLD < accelerometer_x;
}

bool between_X() {
  return -X_THRESHOLD < accelerometer_x && accelerometer_x < X_THRESHOLD;
}

bool below_Y() {
  return -Y_THRESHOLD > accelerometer_y;
}

bool above_Y() {
  return Y_THRESHOLD < accelerometer_y;
}

bool between_Y() {
  return -Y_THRESHOLD < accelerometer_y && accelerometer_y < Y_THRESHOLD;
}

int calculateMode() {

    if (between_X() && between_Y())
      return MODE_STATIONARY;

    if (between_X() && below_Y())
      return MODE_FORWARD;
```

```
    if (between_X() && above_Y())
      return MODE_BACKWARD;

    if (above_X() && between_Y())
      return MODE_LEFT;

    if (below_X() && between_Y())
      return MODE_RIGHT;

    return MODE_STATIONARY;
}

void loop() {

   display.setTextSize(1);
   display.setTextColor(WHITE);
   display.clearDisplay();
   display.setCursor(10, 0);
   display.println("Obstacole ahead in");
   display.setCursor(50, 20);
   display.print(distance);
   display.println(" cm");
   display.display();
  display.clearDisplay();

  Wire.beginTransmission(GY_ADDRESS);
  Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H) [MPU-6000
and MPU-6050 Register Map and Descriptions Revision 4.2, p.40]
  Wire.endTransmission(false); // the parameter indicates that the Arduino
will send a restart. As a result, the connection is kept active.
  Wire.requestFrom(GY_ADDRESS, 7*2, true); // request a total of 7*2=14
registers
  Wire.endTransmission(true);
  // "Wire.read()<<8 | Wire.read();" means two registers are read and stored
in the same variable
  accelerometer_x = Wire.read()<<8 | Wire.read(); // reading registers: 0x3B
(ACCEL_XOUT_H) and 0x3C (ACCEL_XOUT_L)
  accelerometer_y = Wire.read()<<8 | Wire.read(); // reading registers: 0x3D
(ACCEL_YOUT_H) and 0x3E (ACCEL_YOUT_L)
  accelerometer_z = Wire.read()<<8 | Wire.read(); // reading registers: 0x3F
(ACCEL_ZOUT_H) and 0x40 (ACCEL_ZOUT_L)
  temperature = Wire.read()<<8 | Wire.read(); // reading registers: 0x41
(TEMP_OUT_H) and 0x42 (TEMP_OUT_L)
  gyro_x = Wire.read()<<8 | Wire.read(); // reading registers: 0x43
(GYRO_XOUT_H) and 0x44 (GYRO_XOUT_L)
  gyro_y = Wire.read()<<8 | Wire.read(); // reading registers: 0x45
(GYRO_YOUT_H) and 0x46 (GYRO_YOUT_L)
  gyro_z = Wire.read()<<8 | Wire.read(); // reading registers: 0x47
(GYRO_ZOUT_H) and 0x48 (GYRO_ZOUT_L)
```

```
  // uncomment this for debugging - better understanding of gyroscope
  // Serial.print("aX = ");
Serial.print(convert_int16_to_str(accelerometer_x));
  // Serial.print(" | aY = ");
Serial.print(convert_int16_to_str(accelerometer_y));
  // Serial.print(" | aZ = ");
Serial.print(convert_int16_to_str(accelerometer_z));
  // // the following equation was taken from the documentation
[MPU-6000/MPU-6050 Register Map and Description, p.30]
  // Serial.print(" | tmp = "); Serial.print(temperature/340.00+36.53);
  // Serial.print(" | gX = "); Serial.print(convert_int16_to_str(gyro_x));
  // Serial.print(" | gY = "); Serial.print(convert_int16_to_str(gyro_y));
  // Serial.print(" | gZ = "); Serial.print(convert_int16_to_str(gyro_z));
  // Serial.println();

  int mode = calculateMode();
  if(mode != currentMode){
    currentMode = mode;
  }
  Serial.println(currentMode);
  Serial.flush();

  while (!Serial.available()){
    // busy waiting for distance from ultrasonic sensor
  }
  distance = Serial.parseFloat();
}
```

Explicatie cod telecomanda:

In functia calculateMode() se calculeaza modul de mers a masinutei cu ajutorul giroscopului GY521 prin accesarea valoriilor intoarse pe axa Ox si Oy. Dupa ce a fost calculata modul de functionare a masinutei, se trimite catre masinuta prin interfata Serial. In final, se efectueaza un busy waiting pentru aflarea distantei masurate de catre masina a celui mai apropiat obstacol pe directia de mers si se afiseaza pe display.

Cod masinuta:

```
#include <SoftwareSerial.h>
#include <AFMotor.h>

#define TRIG A1
#define ECHO A0
#define Speed 170
#define motor 10
#define spoint 103
#define STOP_DISTANCE 12

constexpr unsigned char MODE_STATIONARY = 30;
constexpr unsigned char MODE_FORWARD = 0;
```

```
constexpr unsigned char MODE_BACKWARD = 1;
constexpr unsigned char MODE_LEFT = 2;
constexpr unsigned char MODE_RIGHT = 3;

int mode = MODE_STATIONARY;
unsigned long previousMillis = 0;
AF_DCMotor M1(1);
AF_DCMotor M2(2);
AF_DCMotor M3(3);
AF_DCMotor M4(4);
void setup() {
  Serial.begin(38400);
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);
  M1.setSpeed(Speed);
  M2.setSpeed(Speed);
  M3.setSpeed(Speed);
  M4.setSpeed(Speed);

  M1.run(RELEASE);
  M2.run(RELEASE);
  M3.run(RELEASE);
  M4.run(RELEASE);
}

void loop() {
  digitalWrite(TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(5);
  digitalWrite(TRIG, LOW);

  float t = pulseIn(ECHO, HIGH);
  float distance = t*0.017015;

  while (!Serial.available()){
    //busy waiting for gyro direction mode
  }
  mode = Serial.parseInt();
  Serial.println(distance);
  Serial.flush();

  if(distance < STOP_DISTANCE && mode == MODE_FORWARD)
    mode = MODE_STATIONARY;

  if (mode == MODE_STATIONARY) {
    M1.run(RELEASE);
    M2.run(RELEASE);
    M3.run(RELEASE);
    M4.run(RELEASE);
```

```
  } else if (mode == MODE_FORWARD) {
    M1.run(FORWARD);
    M2.run(FORWARD);
    M3.run(FORWARD);
    M4.run(FORWARD);
  } else if (mode == MODE_BACKWARD){
    M1.run(BACKWARD);
    M2.run(BACKWARD);
    M3.run(BACKWARD);
    M4.run(BACKWARD);
  } else if (mode == MODE_LEFT){
    M1.run(BACKWARD);
    M2.run(FORWARD);
    M3.run(FORWARD);
    M4.run(BACKWARD);
  } else if (mode == MODE_RIGHT){
    M1.run(FORWARD);
    M2.run(BACKWARD);
    M3.run(BACKWARD);
    M4.run(FORWARD);
  }
}
```

Explicatie cod masina:

Se obtine distanta masurata de senzorul cu ultrasunet, dupa aceea se efectueaza un busy waiting pentru aflarea modului de mers a masinii, care este primit de la telecomanda. Dupa aceea, se transmite cu ajutorul interfetei Serial distanta masurata si se actualizeaza modul de mers.

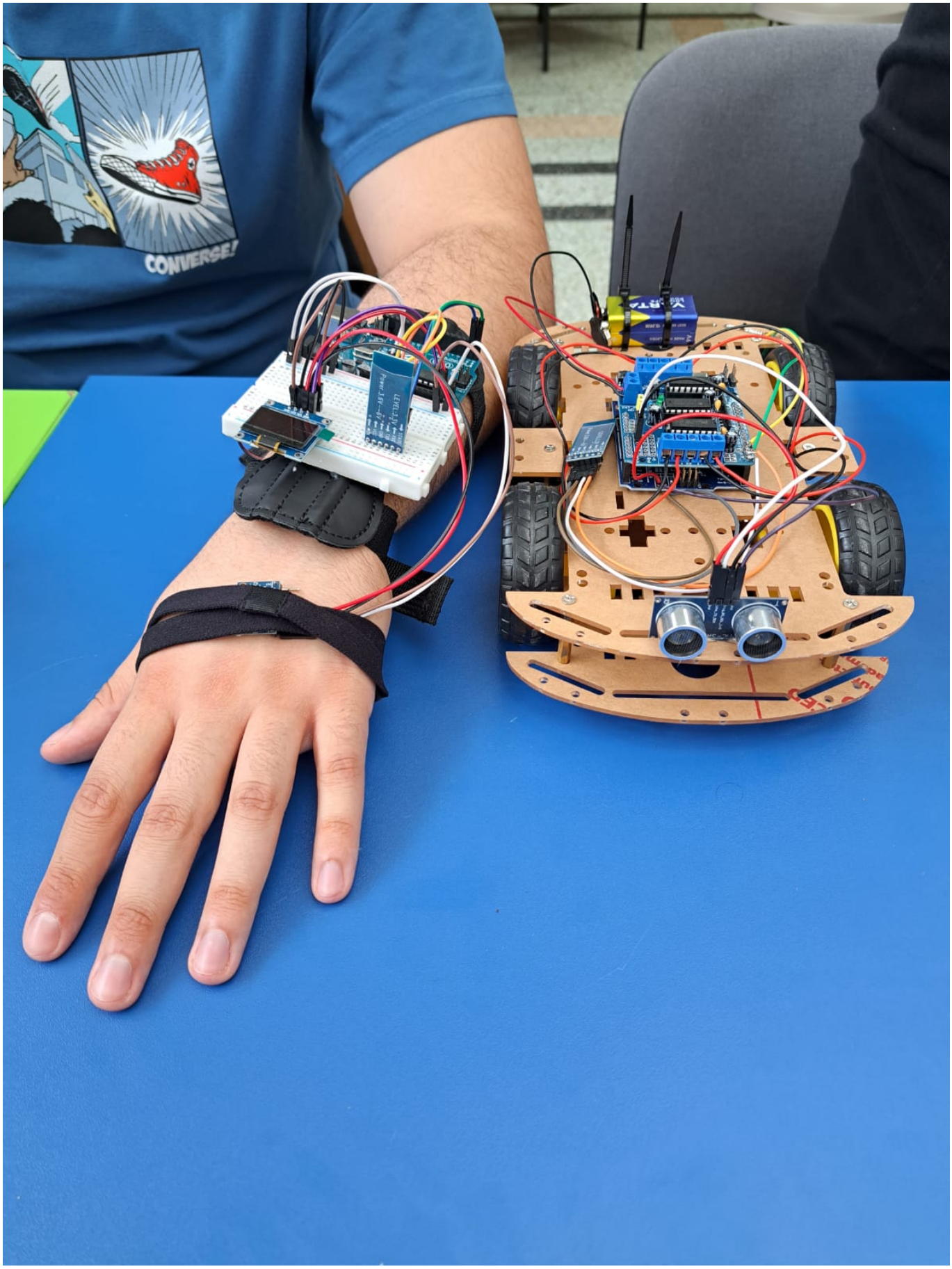
Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

Concluzii

Proiectul si-a atins scopul de a folosi cunostiintele dobandite la laborator, de asemenea am invatat aspecte practice, cum ar fi efectuarea de lipituri cu fludor.

In final, proiectul prezinta si un scop practic, poate fi folosit pentru masurarea de distante in locuri inaccesibile utilizatorilor, avand potential pentru un viitor upgrade de a crea o harta a spatiului pe care il strabate.



Aici puteti gasi un demo al proiectului.

Download

Aici se gaseste arhiva cu codul sursa al proiectului: distantainatorul.zip

Jurnal

- 5 Mai: Creare pagină Wiki + documentație
- 19 Mai: Prezentare parte Hardware
- 26 Mai: Prezentare Software + Prezentare finala
- 28 Mai Definitivare Wiki

Bibliografie/Resurse

Resurse:

- Comunicare intre 2 placute Arduino pe baza de bluetooth
- Folosirea senzorului cu ultrasunete
- Masinuta care evita obstacole
- Utilizare ecran OLED I2C
- Utilizare giroscop GY521
- laboratoarele de PM

Export to PDF