Cyclocomputer

Nume: Cioltan Andrei Florin

Grupa: FILS 1221A

Introducere

Proiectul consta in realizarea unui cyclocomputer care masoara cu ajutorul unui senzor de camp magnetic viteza de deplasare a unei biciclete, viteza medie si distanta totala.

Prin utilizarea unui buton se selecteaza dimensiunea rotii bicicletei. Un alt buton cicleaza printre cele doua moduri de functionare: standard si sprint. In modul standard se afiseaza pe ecran viteza curenta, viteza medie si distanta parcursa. Modul sprint functioneaza asemanator, dar in momentul in care viteza de deplasare scade sub un anumit punct, se aprinde un led, atentionand utilizatorul sa mareasca viteza.

Descriere generală

O schemă bloc cu toate modulele proiectului vostru, atât software cât şi hardware însoţită de o descriere a acestora precum şi a modului în care interacţionează.

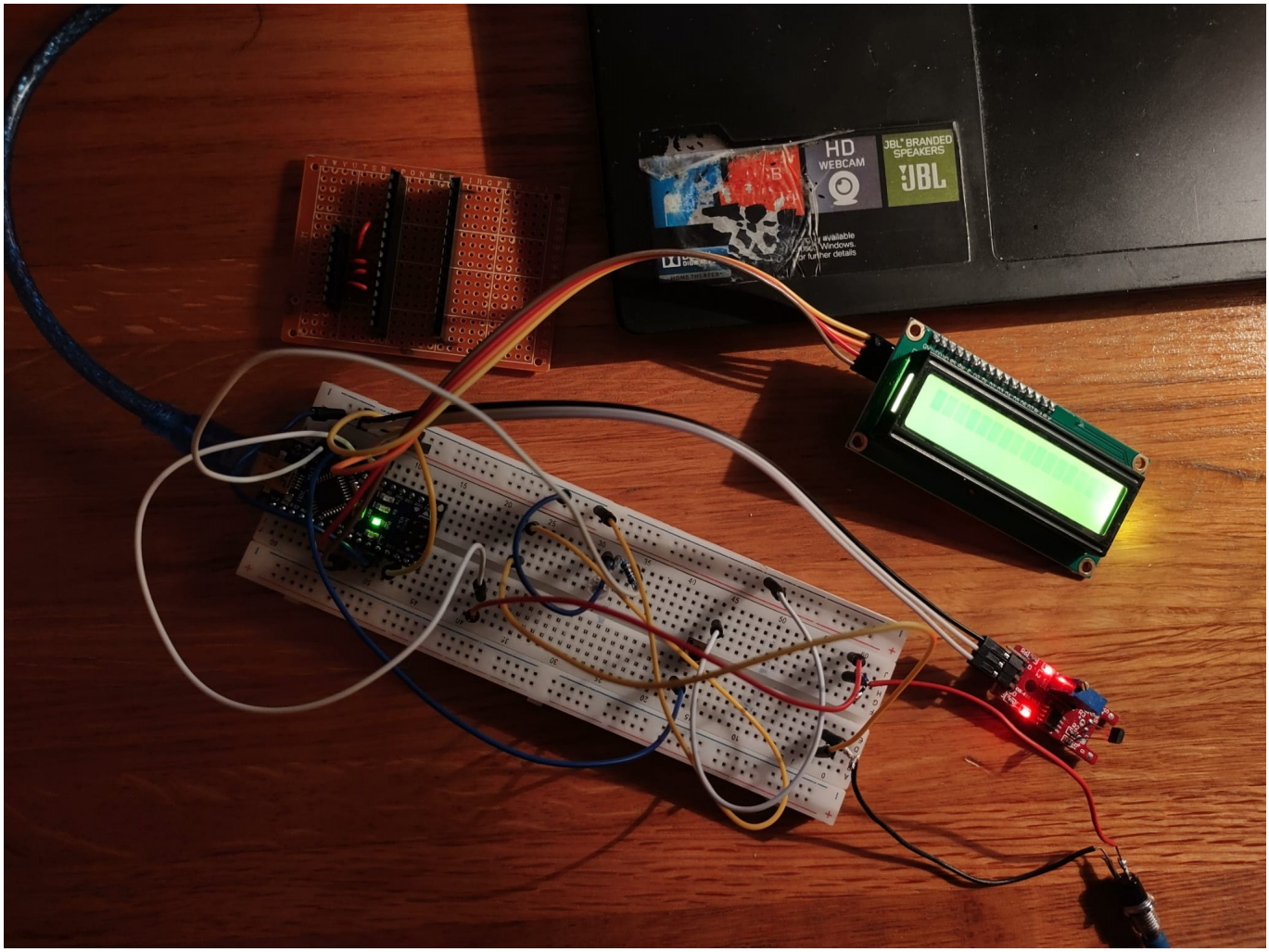
Exemplu de schemă bloc: http://www.robs-projects.com/mp3proj/newplayer.html

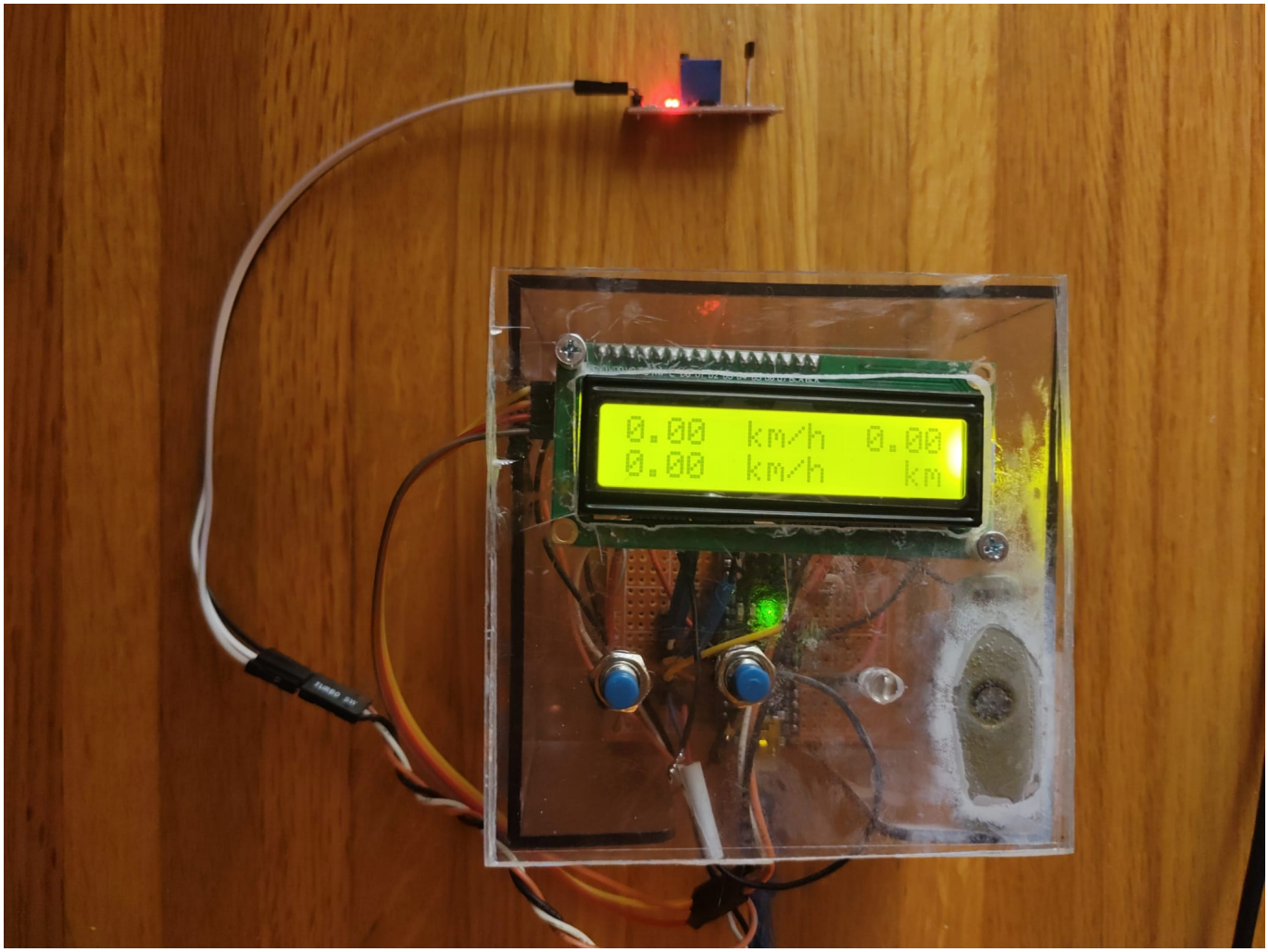
Hardware Design

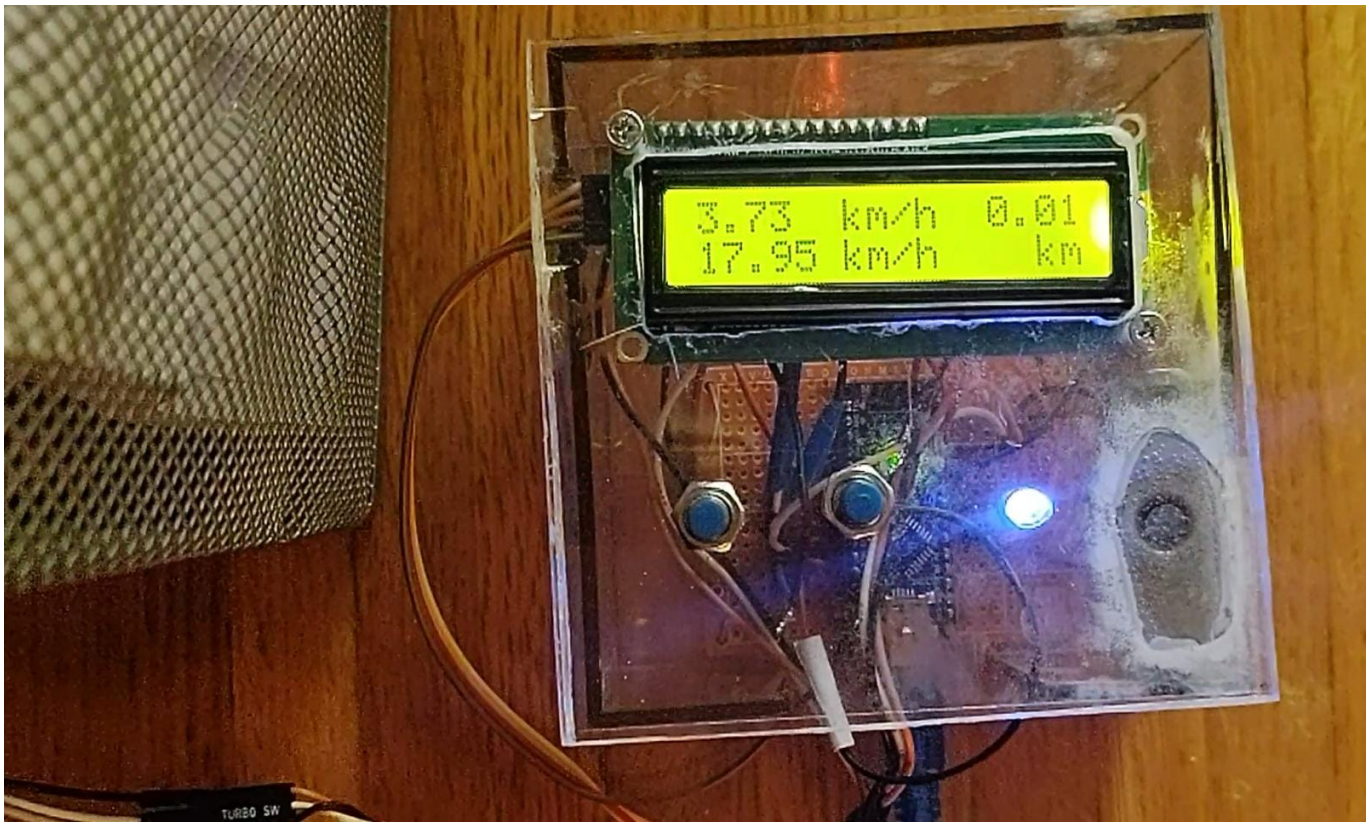
Componente:

- Arduino Nano
- Ecran LCD 1602 IIC/I2C
- Senzor Magnetic(Hall)
- Buton push fara retinere, 7mm, 2 pini
- Led 5mm
- Rezistor 1kOhm

- PCB




0.00 km/h 0.00
0.00 km/h km


3.73 km/h 0.01
17.95 km/h km
TURBO SW

Software Design

```
#include <LiquidCrystal_I2C.h> // Library for LCD
#define LED 7
#define B1 9
#define B2 8
#define reed 6

LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27, 16 column and 2
rows
float radius = 13;// tire radius (in inches)
int reedVal;
long timer = 0;// time between one full rotation (in ms)
float kph = 00.00;
float circumference;
int tire = 0;
float lastSize = radius;

int maxReedCounter = 100;//min time (in ms) of one rotation (for debouncing)
int reedCounter;

int mode = 1; // (1) --> normal mode; (-1) --> sprint mode
int lastMode = 1;
int number_of_modes = 2;
int number_of_wheel_sizes = 3;
int modeState;
int wheelState;
int old_button1_state = HIGH;
int old_button2_state = HIGH;

int totalReadings = 0;
int MAXREADINGS = 100;
float averageSpeed = 0;
float lastAverageSpeed = 0;
float totalDistanceTraveled = 0;
float tireSizes[3] = {13, 13.75, 14.5};

void setup(){
  lcd.init(); //initialize the lcd
  lcd.backlight(); //open the backlight
  lcd.begin(16, 2);
  pinMode(B1, INPUT_PULLUP);
  pinMode(B2, INPUT_PULLUP);
  pinMode(LED, OUTPUT);
  pinMode(reed, INPUT);

  reedCounter = maxReedCounter;
  circumference = 2 * 3.14 * radius;

  cli();//stop interrupts
```

```
  //set timer1 interrupt at 1kHz
  TCCR1A = 0;// set entire TCCR1A register to 0
  TCCR1B = 0;// same for TCCR1B
  TCNT1  = 0;
  // set timer count for 1khz increments
  OCR1A = 1999;// = (1/1000) / ((1/(16*10^6))*8) - 1
  // turn on CTC mode
  TCCR1B |= (1 << WGM12);
  // Set CS11 bit for 8 prescaler
  TCCR1B |= (1 << CS11);
  // enable timer compare interrupt
  TIMSK1 |= (1 << OCIE1A);

  sei();//allow interrupts
  //END TIMER SETUP

  Serial.begin(9600);
}

ISR(TIMER1_COMPA_vect) {//Interrupt at freq of 1kHz

  reedVal = digitalRead(reed);
  int button1 = digitalRead(B1);
  int button2 = digitalRead(B2);

  if ((button1 == LOW) && (old_button1_state == HIGH) && (wheelState <
number_of_wheel_sizes))
  {
    wheelState = wheelState + 1;
    digitalWrite(LED, HIGH);
    lastSize = tireSizes[tire];
    tire++;
    tire %= 3;
    radius = tireSizes[tire];
    circumference = 2 * 3.14 * radius;
  }
  else if ((button1 == LOW) && (old_button1_state == HIGH) && (wheelState >
(number_of_wheel_sizes - 1)))
  {
    wheelState = 1;
    digitalWrite(LED, HIGH);
    lastSize = tireSizes[tire];
    tire++;
    tire %= 3;
    radius = tireSizes[tire];
    circumference = 2 * 3.14 * radius;
  }
  else {
    digitalWrite(LED, LOW);
  }
  old_button1_state = button1;
```

```
  if ((button2 == LOW) && (old_button2_state == HIGH) && (modeState <
number_of_modes))
  {
    modeState = modeState + 1;
    lastMode = mode;
    mode *= -1;
  }
  else if ((button2 == LOW) && (old_button2_state == HIGH) && (modeState >
(number_of_modes - 1)))
  {
    modeState = 1;
    lastMode = mode;
    mode *= -1;
  }
  old_button2_state = button2;

  if(mode == -1 && kph <= 20) {
    digitalWrite(LED, HIGH);
  }
  else {
    digitalWrite(LED, LOW);
  }

  if (reedVal != 1){
    if (reedCounter == 0){//min time between pulses has passed
      //kph = (56.8 * float(circumference))/float(timer);//calculate miles
per hour
      //kph *= 1.60934;// calculate km per hour
      kph = (3600 * (float(circumference) * 0.0254)) / float(timer);
      totalDistanceTraveled += (circumference / 39370.07);
      totalReadings++;
      averageSpeed = (averageSpeed + kph) / 2;
      if(totalReadings == MAXREADINGS) {
        totalReadings = 1;
        averageSpeed = kph;
      }
      timer = 0;//reset timer
      reedCounter = maxReedCounter;
    }
    else{
      if (reedCounter > 0){
        reedCounter -= 1;
      }
    }
  }
  else{
    if (reedCounter > 0){
      reedCounter -= 1;
    }
  }
  if (timer > 2000){
```

```
    kph = 0;//if no new pulses from senzor - tire is still, set kph to 0
  }
  else{
    timer += 1;//increment timer
  }
}

void displayKPH(){

  if(radius != lastSize) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Wheel Size ");
    lcd.print(radius * 2);
    delay(700);
    lcd.clear();
    lastSize = radius;
    averageSpeed = 0;
  }

  if(mode != lastMode) {
    lcd.clear();
    lcd.setCursor(0, 0);
    if(mode == 1)
      lcd.print("Normal");
    else
      lcd.print("Sprint");
    lcd.print(" Mode");
    delay(700);
    lcd.clear();
    lastMode = mode;
  }

  lcd.setCursor(0, 0); // move cursor to   (0, 0)
  lcd.print(kph);
  if(kph < 10) {
    lcd.setCursor(4, 0);
    lcd.print(" ");
  }
  lcd.setCursor(6,0);
  lcd.print("km/h");
  lcd.setCursor(0, 1);
  lcd.print(averageSpeed);
  lcd.setCursor(6, 1);
  lcd.print("km/h");
  lcd.setCursor(12,0);
  lcd.print(totalDistanceTraveled);
  lcd.setCursor(14, 1);
  lcd.print("km");
  Serial.println(kph);
}
```

```
void loop(){
  displayKPH();
  delay(500);
}
```

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

Concluzii

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite:

https://en.wikipedia.org/wiki/Cyclocomputer

https://arduinogetstarted.com/tutorials/arduino-lcd-i2c

https://sensorkit.joy-it.net/en/sensors/ky-024

Export to PDF

From:
http://ocw.cs.pub.ro/courses/ - **CS Open CourseWare**

Permanent link:
http://ocw.cs.pub.ro/courses/pm/prj2023/apredescu/cyclocomputer

Last update: **2023/05/29 20:22**