Cyclocomputer


```
Nume: Toma Bogdan-Gabriel
```

```
Grupa: 333AC
```


Introducere

Scopul acestui proiect este de a crea un ciclocomputer. Acesta este un dispozitiv ce se monteaza, de obicei, pe ghidonul bicicletei si furnizeaza date despre traseul curent (kilometri parcursi, timpul care a trecut de la inceperea traseului precum si caloriile arse).

Cine poate beneficia, cel mai mult, de pe urma utilizarii unui astfel de dispozitiv:

- Cei ce isi doresc sa piarda greutate
- Cei ce isi doresc sa piarda greutate + sa afiseze, in mod ostentativ, pe retelele de socializare efortul depus (acum putand fi cuantificat acest lucru)

}

Descriere generală

“Inima” acestui proiect este o placuta Arduino Nano, ce utilizeaza microcontroller-ul ATmega328.

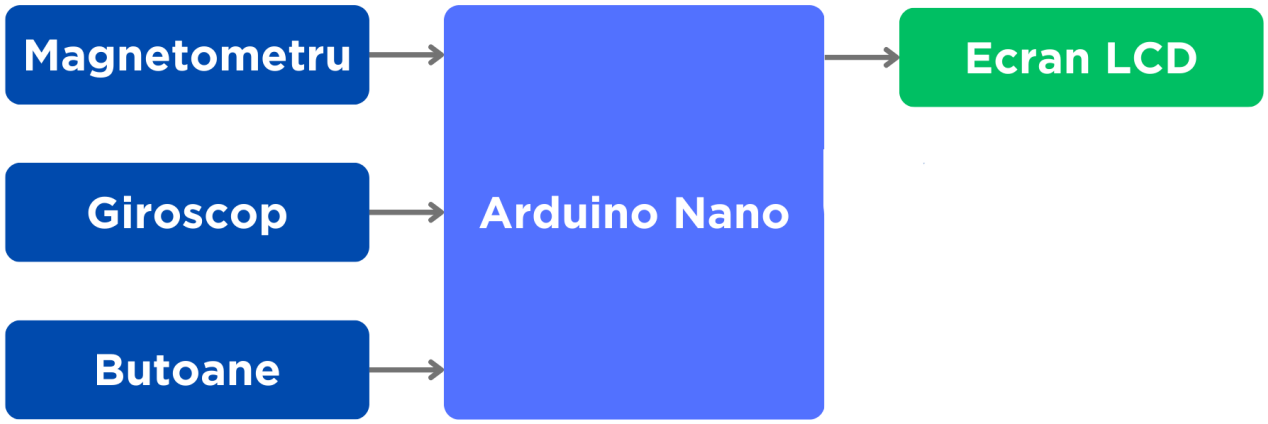
Principalele modulele folosite sunt:

- Magnetometru
- Senzor Giroscop
- Ecran LCD

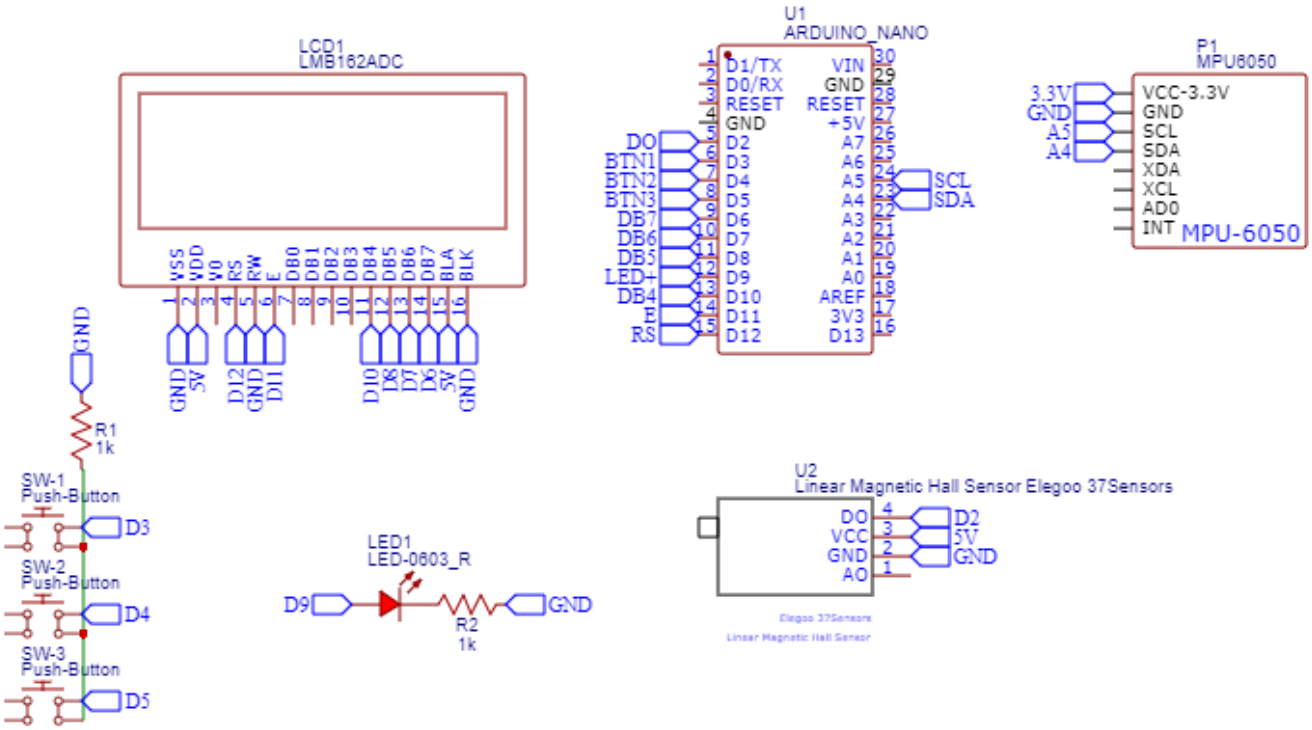
- Magnetometrul are sarcina de a contoriza turele rotii. Cunoscand acest numar si diametrul rotii putem calcula distanta parcursa pentru fiecare tura / pentru intregul traseu.

- Senzorul Giroscop are sarcina de a detecta panta / rampa pentru a putea calcula corect caloriile arse

- Ecranul LCD are rolul de a afisa datele pe care le manipulam utilizand butoanele ce vor fi incluse in proiect



Hardware Design



Software Design

Biblioteci folosite:

- Wire.h
- LiquidCrystal.h

Mediu de dezvoltare:

- Arduino IDE
- Wokwi

Laboratoare folosite (fara GPIO):

- Intreruperi
- Timere / PWM
- SPI
- I2C

Pentru inceput, functiile clasice arduino (si anume setup() si loop()) arata in felul urmator:

```
void setup() {
  Serial.begin(9600);
  setupButtons();
  setupI2C();
  initGyro();
  setupLCD();
  cli();
  setupInterrupt();
  sei();
}
```

```
void loop() {
  interrupt = true;
  gyro();
  listenForCycleModesOrRec();
  resetSpeedIfIdleLong();
}
```

Acum, sa le luam pe fiecare in parte si sa le discutam putin.

```
void setupButtons() {
  PORTD |= ((1 << PD3) | (1 << PD4) | (1 << PD5));
  DDRB |= (1 << PB1);
}
```

Aceasta functie configureaza pinii D3, D4, D5 ca fiind input (nealterand registrul DDRD) si activeaza rezistentele de pull-up (registrul PORTD). Pinul D9 este configurat ca output (deoarece este folosit pentru PWM la led).

```
void setupI2C() {
  Wire.begin();
}
```

Aceasta functie porneste comunicatia I2C prin initializarea Wire.

```
void initGyro() {
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6, true);
  AccX = (Wire.read() << 8 | Wire.read()) / 16384.0;
  AccY = (Wire.read() << 8 | Wire.read()) / 16384.0;
  AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0;
  AngleY = (atan(-1 * AccX / sqrt(pow(AccY, 2) + pow(AccZ, 2))) * 180 / PI)
+ 1.58 - 3.1;
}
```

Aceasta functie initializeaza unghiul Y, calculandu-l folosind informatiile din accelerometru, nu giroscop. (abia dupa initializare intra in joc giroscopul)

```
void setupLCD() {
  lcd.begin(16, 2);
}
```

Aceasta functie porneste ecranul LCD prin initializarea lcd.

```
void setupInterrupt() {
  EICRA = 0;
  EICRA |= (1 << ISC01);
  EIMSK = 0;
  EIMSK |= (1 << INT0);
}
```

Aceasta functie configureaza intreruperea (de fiecare data cand magnetometrul detecteaza un camp magnetic, cu alte cuvinte roata a facut o rotatie, este declansata o intretupere). Registrul EICRA este modificat astfel incat bitul ISC01 sa fie activ. Acest lucru cauzeaza ca intreruperea sa aiba loc pe frontul descrescator al pinului INT0 (D2) (Am ales frontul descrescator pentru a fi siguri ca roata face o rotatie completa). Registrul EIMSK activeaza intreruperea pentru pinul INT0.

Acestea au fost functiile ce se gasesc in setup(). Acum ne indreptam atentia catre functiile din loop(), care sunt:

void gyro() {

```
  Wire.beginTransmission(MPU);
  Wire.write(0x45);
  Wire.endTransmission(false);
```

```
  Wire.requestFrom(MPU, 2, true);
  GyroY = (Wire.read() << 8 | Wire.read()) / 131.0 * (-1) + 2.2;
  if(abs(GyroY) >= 0.5)
    GyroAngleY += GyroY * (millis() - prevTimeGyro) / 1000;
  prevTimeGyro = millis();
}
```

Aceasta functie citeste, continuu, valoarea unghiului Y.

Mai intai este initializata transmisia catre giroscop (MPU este adresa giroscopului). Apoi accesam registrul 0x45, unde se gasesc valorile de interes, si anume GYRO_YOUT, valoare care se intinde pe 2 registrii a cate 8 biti, asadar cerem a fi cititi 2 registrii din giroscop. Aplicam niste corectii, si anume, impartim la 131, inmultim cu -1 si adunam 2.2, iar apoi daca valoarea citita este mai mare decat 0.5 o adaugam in variabila GyroAngleY tinand cont de durata acesteia. La final ar trebui sa ramanem cu unghiul ciclocomputer-ului.

```
void listenForCycleModesOrRec() {
  if(!(PIND & (1 << PD5))) {
    if(pressedPD[5] == false) {
      prevTimeButton = millis();
      pressedPD[5] = true;
    }
    else if(millis() - prevTimeButton >= 1000 && PWMOn == false && freeze ==
false && rec == false) {
      PWMOn = true;
      initPWM();
    }
  }
  else if(PIND & (1 << PD5) && pressedPD[5] == true) {
    pressedPD[5] = false;
    PWMOn = false;
    resetPWM();
    if(freeze) {
      kilometers = 0;
      calories = 0;
      seconds = 0;
      minutes = 0;
      hours = 0;
      freeze = false;
    }
    else if(millis() - prevTimeButton < 1000 || rec == true) {
      if(rec) {
        freeze = true;
        rec = false;
      }
      else if(mode < 2)  mode++;
      else  mode = 0;
    }
    else if(mode == 0) {
      rec = true;
```

```
      baseTime = millis();
    }
  }
  if(mode == 0){
    rideMode();
  }
  else if(mode == 1) {
    setWheelMode();
  }
  else if(mode == 2){
    setWeightMode();
  }
  if(rec) {
    seconds = ((millis() - baseTime) / 1000) % 60;
    minutes = (seconds / 60) % 60;
    hours = (minutes / 60) % 24;
  }
}
```

Aceasta functie are rolul de a asculta pentru apasarile butoanelor si de a actiona corespunzator. Este alcatuita din elemente simple, dar intr-adevar voluminoase, pentru a indeplini urmatoarele functionalitati:

(notam: D5 - Butonul 1, D4 - Butonul 2, D3 - Butonul 3)

- Butonul 1 apasat scurt (sub o secunda) - se trece in modul de selectie al diametrului rotii, diametru care este incrementat la apasarae butonului 3, si decrementat la apasarea butonului 2. Reapasarea butonului 1 conduce la intoarcerea la ecranul default.
- Butonul 1 tinut apasa mai mult de o secunda - se activeaza PWM pe led pana cand butonul este eliberat, apoi incepe modul de inregistrare (contorizare de km, calorii si timp)
- Daca modul inregistrare este pornit, apasarea butonului 1 duce la setarea **true** a variabilei **freeze**. Acum km, caloriile, dar si timpul raman blocate pana la reapasarea butonului, lucru ce conduce la resatarea completa a km, calorii si timp.

Acum este momentul potrivit pentur a prezenta functiile folosite in functia anterioara.

```
void rideMode() {
  if(millis() - prevTimeLCD >= 500){
    lcd.clear();
    lcd.setCursor(0, 0);
    if(kilometers < 10)
      lcd.print(String(kilometers, 2) + " km");
    else if(kilometers >= 10 && kilometers < 100)
      lcd.print(String(kilometers, 1) + " km");
    else if(kilometers >= 100 && kilometers < 1000)
      lcd.print(String(kilometers, 0) + " km");
    else
      lcd.print("999 km");
    lcd.setCursor(0, 1);
    (hours >= 10) ? lcd.print(String(hours)) : lcd.print("0" +
```

```
String(hours));
    lcd.setCursor(2, 1);
    lcd.print(":");
    lcd.setCursor(3, 1);
    (minutes >= 10) ? lcd.print(String(minutes)) : lcd.print("0" +
String(minutes));
    lcd.setCursor(5, 1);
    lcd.print(":");
    lcd.setCursor(6, 1);
    (seconds >= 10) ? lcd.print(String(seconds)) : lcd.print("0" +
String(seconds));
    lcd.setCursor(8, 0);
    lcd.print("Cal");
    lcd.setCursor(12, 0);
    lcd.print(String((int)calories));
    lcd.setCursor(9, 1);
    lcd.print("Kmh");
    lcd.setCursor(13, 1);
    lcd.print(String((int)speedKmh));
    prevTimeLCD = millis();
  }
}
```

Aceasta functie are rolul de a afisa pe LCD informatiile de interes modului default, si anume, Ride Mode.

```
void setWheelMode() {
  if(millis() - prevTimeLCD >= 500) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Wheel diameter:");
    lcd.setCursor(0, 1);
    lcd.print(String(wheelDiameterCm) + " cm");
    prevTimeLCD = millis();
  }
  if(!(PIND & (1 << PB3)) && pressedPD[3] == false) {
    pressedPD[3] = true;
    wheelDiameterCm += 10;
  }
  else if(PIND & (1 << PB3)) pressedPD[3] = false;
  if(!(PIND & (1 << PB4)) && pressedPD[4] == false) {
    pressedPD[4] = true;
    if(wheelDiameterCm >= 10){
      wheelDiameterCm -= 10;
    }
  }
  else if(PIND & (1 << PB4)) pressedPD[4] = false;
}
```

Aceasta functie a rolul de a afisa pe LCD informatiile de interes modului 2, si anume, Set Wheel Mode.

Tot aici apasarile butoanelor 2 si 3 conduc la modificarea diametrului rotii.

```
void setWeightMode() {
  if(millis() - prevTimeLCD >= 500) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Weight:");
    lcd.setCursor(0, 1);
    lcd.print(String(weight) + " cm");
    prevTimeLCD = millis();
  }
  if(!(PIND & (1 << PB3)) && pressedPD[3] == false) {
    pressedPD[3] = true;
    weight += 10;
    weightMultiplier = (1.6 * (weight / 80)) / 2;
  }
  else if(PIND & (1 << PB3)) pressedPD[3] = false;
  if(!(PIND & (1 << PB4)) && pressedPD[4] == false) {
    pressedPD[4] = true;
    if(wheelDiameterCm >= 10){
      weight -= 10;
      weightMultiplier = (1.6 * (weight / 80)) / 2;
    }
  }
  else if(PIND & (1 << PB4)) pressedPD[4] = false;
}
```

Aceasta functie a rolul de a afisa pe LCD informatiile de interes modului 3, si anume, Set Weight Mode. Tot aici apasarile butoanelor 2 si 3 conduc la modificarea gretutatii.

```
void initPWM() {
  TCCR1A = 0;
  TCCR1A |= ((1 << COM1A0) | (1 << WGM12));
  OCR1A = 31270;
  TCNT1 = 0;
}
```

Aceasta functie configureaza PWM pentru led. Setam in registrul TCCR1A bitii COM1A0, responsabil pentru trecerea pinului OC1A (D9) in starea low atunci cand valoarea din timer ajunge la valoarea prestabilita, si anume, 31270, si bitul WGM12, responsabil pentru setarea modului CTC.

```
void resetPWM() {
  TCCR1A = 0;
}
```

Opusul functiei prezentate anterior.

Acum, “piesa de rezistenta”, functia ce se executa in intrerupere:

```
ISR(INT0_vect)
{
  if(interrupt) {
    distance = (2 * 3.14 * (wheelDiameterCm / 2)) / 100000;
    if(rec){
      if(kilometers < 1000)
        kilometers += distance;
      if(calories < 9999) {
        slopeMultiplier = 3 * AngleY / 1.72;
        if(AngleY >= 0.6)
          calories += distance * 32 * slopeMultiplier * weightMultiplier;
        else if (AngleY >= -0.6 && AngleY < 0.6)
          calories += distance * 32;
        else
          calories += distance * 32 / (slopeMultiplier * (-1)) *
weightMultiplier;
      }
    }
    speedKmh = distance * 3600000/(millis() - prevTimeKmh);
    if(speedKmh > 999)
      speedKmh = 999;
    prevTimeKmh = millis();
    interrupt = false;
  }
}
```

Aceasta functie calculeaza distanta, in km, bazat pe diametrul rotii (variabile booleana **interrupt** este folosita pentru a nu avea intretuperi multiple (un fel de debouncing pentru intreruperi). Tot aici, in cazul in care inregistram (variabila **rec** este **true**) contorizam km, calculam caloriile si viteza. Calculul caloriilor se bazeaza pe date colectate din sursele ce urmeaza a fi citate mai jos, si pe regula de 3 simpla. Pentru calculul vitezei se tine cont de diametrul rotii si de timpul dintre 2 intreruperi. in cazul in care acesta este mai mare de 1 secunda, viteza se reseteaza folosind functia urmatoare.

```
void resetSpeedIfIdleLong() {
  if(millis() - prevTimeKmh >= 1000) speedKmh = 0;
}
```

Aceasta functie reprezinta sfarsitul lungului sir de functii folosite (ura!). Variabilele folosite de alungul programului sunt urmatoarele:

```
LiquidCrystal lcd(12, 11, 6, 7, 8, 10);
const int MPU = 0x68;
float GyroY = 0, GyroAngleY;
float prevTimeGyro = 0;
float prevTimeKmh = 0;
float prevTimeButton = 0;
float prevTimeLCD = 0;
int baseTime = 0;
float kilometers = 0, distance = 0;
int hours = 0, minutes = 0, seconds = 0;
```

```
float wheelDiameterCm = 0, weight = 0;
float calories = 0;
int mode = 0;
float speedKmh = 0;
bool rec = false;
bool interrupt = true;
bool pressedPD[7] = { false };
bool PWMOn = false;
bool freeze = false;
float slopeMultiplier = 0, weightMultiplier = 0;
```

Explicatie ecuatii calorii:

Luand ca referinta average male (175 cm, 80 kg) si o viteza medie (20 km/h) avem urmatoarele date:

- calorii arse intr-un km ~= 32
- calorii arse intr-o ora de ciclat pe teren plat ~= 300, calorii arse intr-o ora de ciclat in rampa de 3% (1.72 grade) ~= 900 (3 ori mai mult)
- calorii arse intr-o ora de ciclat ~= 300, calorii arse intr-o ora de cicalt, de aceeasi persoana, dar kilograme duble ~= 480 (1.6 ori mai mult)

Cunoscand aceste date, putem aplica regula de 3 simpla pentru a afla caloriile arse pentru diferite unghiuri de panta sau kilograme astfel:

---

caloriile arse 3 ori mai mult ... 1.72 unghi

calorii arse de x ori mai mult ... y unghi (furnizat de giroscop)

---

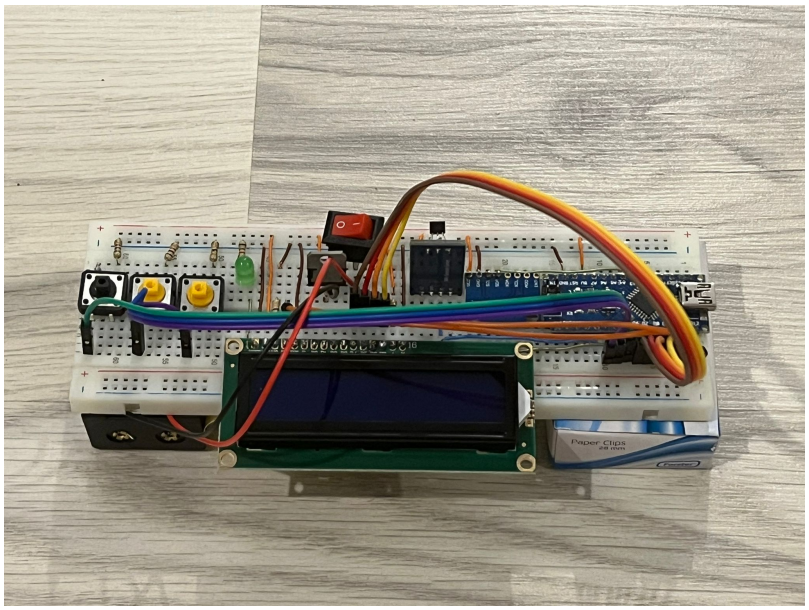
calorii arse de 1.6 ori mai mult ... avem de 2 ori 80 kg

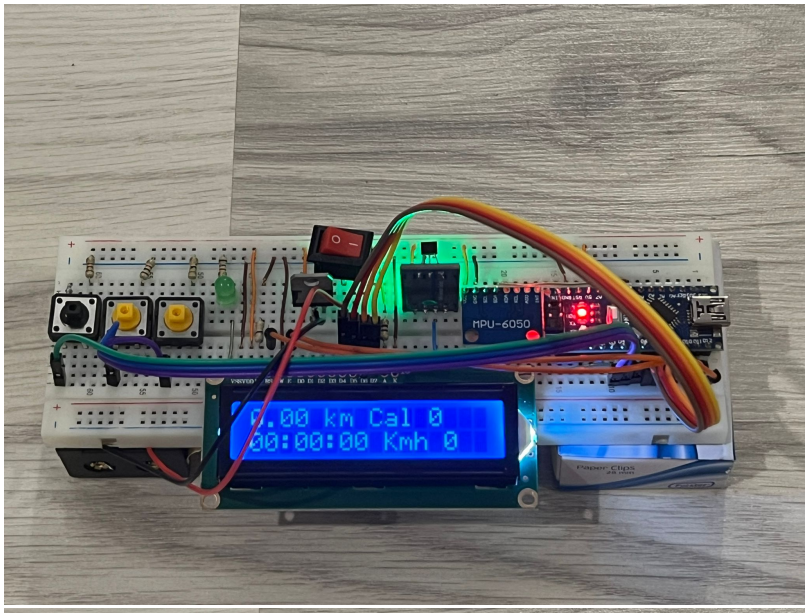
calorii arse de x ori mai mult ... avem de y ori 80 kg

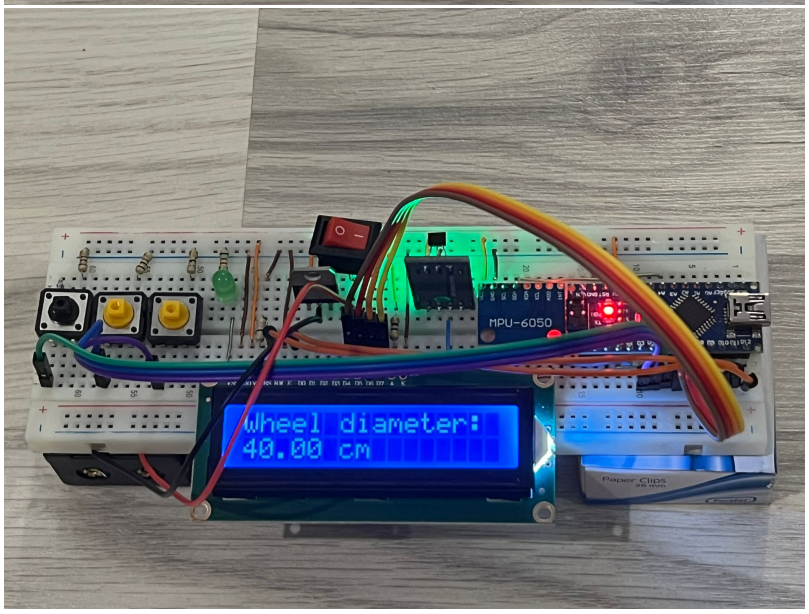
Cele doua reguli conduc la urmatoarele ecuatii:

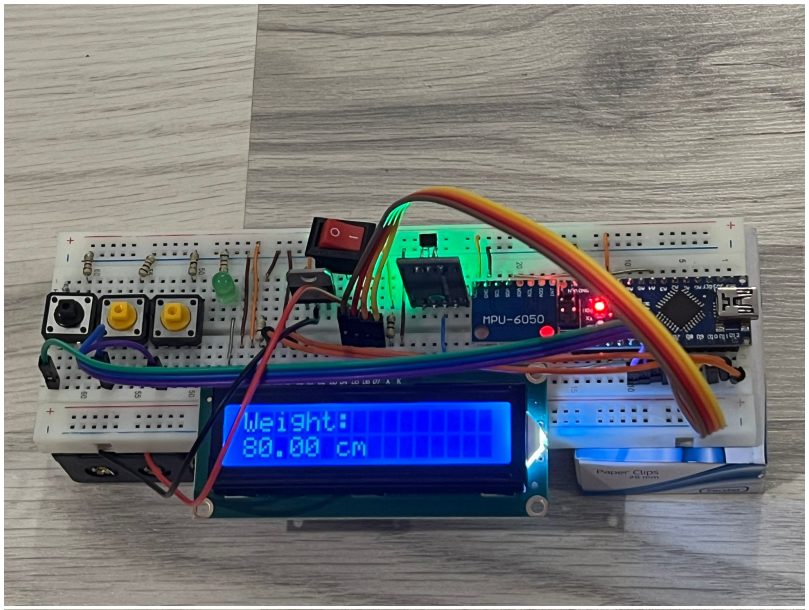
- caloriile += (distanta parcursa la o tura de roata, in km) * 32 (average calorii arse intr-un km) * ((3 * AngleY) / 1.72)
- caloriile += (distanta parcura la o tura de roata, in km) * 32 (average calorii arse intr-un km) * ((1.6 * (weight / 80) / 2);

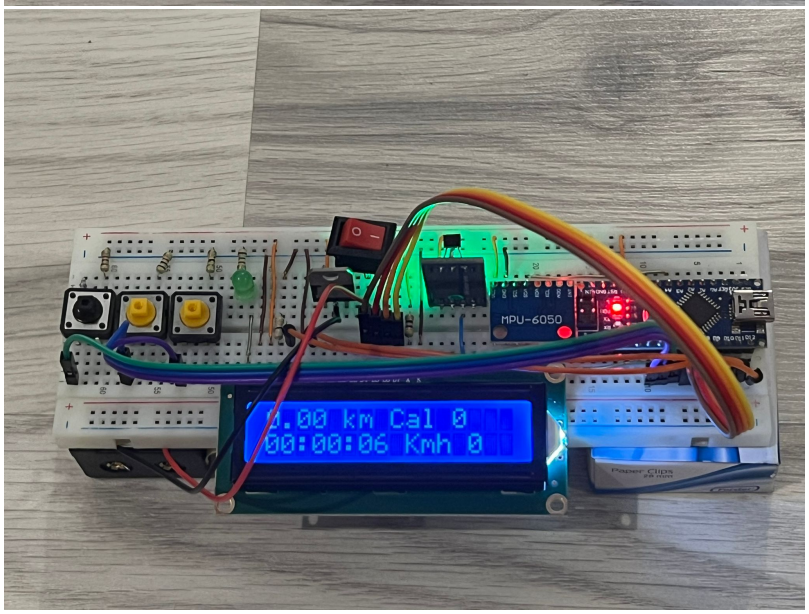
Rezultate Obţinute

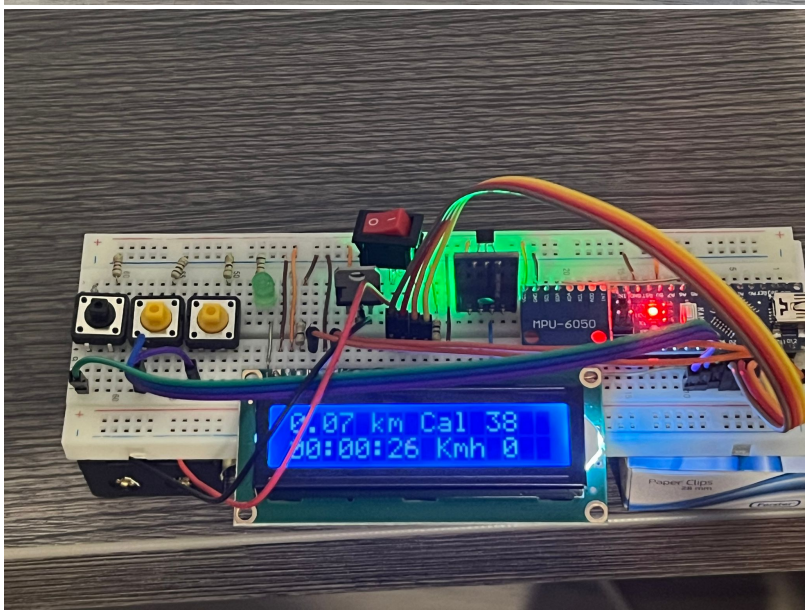







MPU-6050
Weight:
80.00 cm


MPU-6050
.00 km Cal 0
00:00:06 Kmh 0


MPU-6050
.07 km Cal 38
00:00:26 Kmh 0

Concluzii

In urma acestui proiect am realizat ca este mult mai usor sa te accidentezi stand la o masa decat ai crede. In saptamanile de proiect m-am accidentat cu:

- Foarfeca
- Letcon
- Fire
- Alta foarfeca
- Cositor
- Pini

Asadar, in urma instalarii unui usor PTSD la simplul contact vizual cu o unealta din campul electronicii / electrotehnicii / orice cuvant prefixat, sau sufixat, de termenul “electro”, am invatat sa dau importanta normelor de protectia muncii. Tot in cadrul proiectului am invatat ca led-urile cumparate de la un magazin, cel putin dubios, dintr-un imobil interbelic, a carui cale de acces (usa) se afla in curtea din spate, deloc primitoare, comuna cu alte imobile invecinate, pot fi putin, putin mai mult, peste pretul de piata.

Desigur, pe parcursul proiectului am invatat cum se configureaza un arduino folosind registri, ceea ce conduce la o viteza conisderabil mai mare a programului si la minimizarea spatiului utilizat, cum compilatorul optimizeaza codul, cum functioneaz intreruperile, timerele, etc. dar asta nu este foarte important. In viata mai des te intalnesti cu o foarfeca (nu o sa o iert) decat cu un timer pe 16 biti care trebuie configurat in asa fel incat sa genereze un semnal PWM pe pinul D9 cu un duty cycle de 50%, avand frecventa nealterata (16MHz).

Dar ca sa concluzionam, scurt, foarte frumos, mai facem™ :like:

Download

cyclocomputer_toma_bogdan_gabriel.rar

Bibliografie/Resurse[[http://example.com|External Link]]

ATmega328P Datasheet

MPU-6050 Datasheet

Calories burned cycling uphill

Cycling calories burned calculator

Slope to degrees table

Average height / weight men

Average height / weight women

Slope grade formulas

Export to PDF