Manole Alexandru: Bratara

Introducere

Nume: Manole Alexandru
Grupa: 333CB
Indrumator: Daniel Dosaru

Proiectul constă în dezvoltarea unei brățări de monitorizare a somnului, care utilizează un microcontroller pentru a colecta și analiza datele privind activitatea cardiacă și mișcările corpului în timpul somnului si un ecran pentru a afisa aceste date.

Scopul: Scopul principal al acestei brățări este de a oferi utilizatorului informații detaliate despre calitatea somnului său, inclusiv perioadele de somn adânc și superficial, ritmul cardiac și alte aspecte relevante. Prin înregistrarea și analiza acestor date, utilizatorul poate obține o mai bună înțelegere a obiceiurilor sale de somn și poate lua măsuri pentru a îmbunătăți calitatea acestuia.

Ideea de la care am pornit: Am pornit de la dorința de a crea un dispozitiv portabil și accesibil care să ofere o soluție convenabilă pentru monitorizarea somnului.

Utilitate pentru alții:

-Promovează sănătatea și bunăstarea

-Monitorizare neinvazivă

Descriere generală

Celula reprezinta butonul folosit pentru a schimba modul de afisare pe ecranul LCD. Celula rosie este microcontrollerul, iar celelalte sunt LED-ul portocaliu si accelerometrul MPU6050.

Hardware Design

Componente:

-Placa de dezvoltare microcontroler Arduino UNO bazată pe ATmega328P

-Breadboard pentru construirea circuitului electronic

-Ecran LCD1602 cu I2C

-Accelerometru pentru detectarea miscarilor in somn

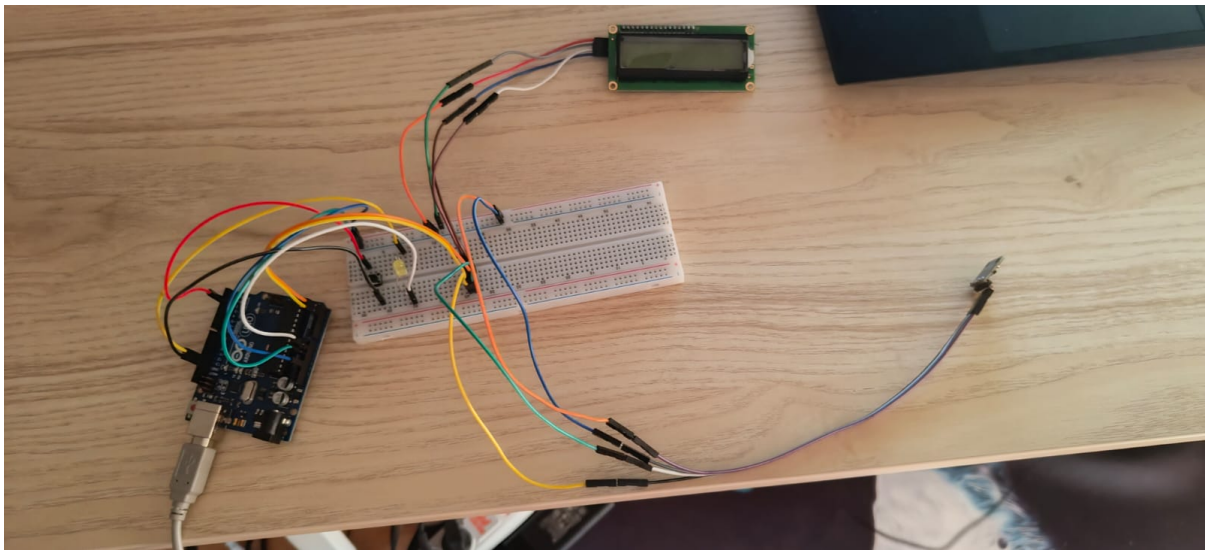
-Rezistor de 10kΩ

-LED pentru a arata starea de BAD SLEEP

-Fire de conectare pentru circuit

Diagrama circuit

Implementare



Software Design

Codul aplicatiei a fost scris in Arduino IDE. Voi detalia mai jos cateva aspecte importante legate de firmware.

Biblioteci folosite

- LiquidCrystal_I2C.h
- I2Cdev.h
- MPU6050_6Axis_MotionApps20.h

main.cpp

```
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include <LiquidCrystal_I2C.h>

MPU6050 mpu;

bool dmpReady = false;  // set true if DMP init was successful
uint8_t devStatus;      // return status after each device operation (0
= success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42
bytes)
uint8_t fifoBuffer[64]; // FIFO storage buffer
uint32_t total_sleep_movement;
uint32_t time_periods_elapsed;
uint32_t average_sleep_movement;
const int buttonPin = 2;
const int ledPin = 13;

Quaternion q;           // quaternion container
VectorInt16 aa;         // accel sensor measurements
VectorInt16 aaReal;     // gravity-free accel sensor measurements
VectorFloat gravity;    // gravity vector
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
    total_sleep_movement = 0;
    time_periods_elapsed = 0;
    pinMode(buttonPin, INPUT_PULLUP);
    pinMode(ledPin, OUTPUT);
    lcd.init();
    // Turn on the backlight
    lcd.backlight();
    // Print a message to the LCD
    lcd.print("Hello, world!");
    Wire.begin();
    Serial.begin(115200);
    mpu.initialize();
    Serial.println(mpu.testConnection() ? F("MPU6050 connection
successful") : F("MPU6050 connection failed"));
    devStatus = mpu.dmpInitialize();
    mpu.setXGyroOffset(220);
    mpu.setYGyroOffset(76);
    mpu.setZGyroOffset(-85);
```

```
    mpu.setZAccelOffset(1788);
    if (devStatus == 0) {
        mpu.CalibrateAccel(6);
        mpu.CalibrateGyro(6);
        mpu.setDMPEnabled(true);
        // attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN),
dmpDataReady, RISING);
        dmpReady = true;
        packetSize = mpu.dmpGetFIFOPacketSize();
    }
}

void ShowSleepSituation() {
  lcd.clear();
  average_sleep_movement = total_sleep_movement / time_periods_elapsed;
  if (average_sleep_movement > 1000) {
    digitalWrite(ledPin, HIGH); // Turn on the LED
  } else {
    digitalWrite(ledPin, LOW);  // Turn off the LED
  }
  if (digitalRead(buttonPin) == LOW) {
    lcd.print("Avg move: ");
    lcd.print(average_sleep_movement);
  } else {
  if(average_sleep_movement < 100) {
    lcd.print("Very Good Sleep");
  } else if(average_sleep_movement < 500) {
    lcd.print("Good Sleep");
  } else if (average_sleep_movement < 1000) {
    lcd.print("Decent Sleep");
  } else{
    lcd.print("Bad Sleep");
  }
  Serial.println(average_sleep_movement);
  }
}

void loop() {
    if (!dmpReady) return;
    if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) {
        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetAccel(&aa, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
        Serial.print("areal\t");
        Serial.print(aaReal.x);
        Serial.print("\t");
        Serial.print(aaReal.y);
        Serial.print("\t");
        Serial.println(aaReal.z);
```

```
    }
    if(abs(aaReal.x) > abs(aaReal.y)) {
      total_sleep_movement += abs(aaReal.x);
    } else {
      total_sleep_movement += abs(aaReal.y);
    }
    time_periods_elapsed++;
    if(time_periods_elapsed % 5 == 0) {
      ShowSleepSituation();
    }

    delay(500);
}

void dmpDataReady() {
    // MPU interrupt function - this just sets a flag in the main code
}
```

Show Sleep Situation este functia folosita pentru a afisa informatii la LCD si LED, atunci cand este chemata (o data la 2.5secunde).

Functiile din setup sunt folosite pentru a stabiliza accelerometrul.

In loop sunt afisate si calculate datele senzorului.

Rezultate Obţinute

Download

zipbomb.zip

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, eventual grupate pe *Resurse Software* şi *Resurse Hardware*.

Export to PDF