Health Kit - Nicoară Laura-Elena 333CA

Introducere

Proiectul constă în obținerea unor rezultate legate de propriul corp, mai precis:

- nivelul de oxigen din sânge
- pulsul
- măsurarea activității electrice cardiace (EKG).

Sănătatea, deși este cea mai importantă, este neglijată de mulți dintre noi din lipsa timpului. Un kit ce oferă posibilitatea de a oferi informații cu privire la metrici legate de sănătatea personală este util pentru monitorizarea mai rapidă și mai la îndemână. Întotdeauna mi-am dorit să știu date despre puls, tensiune și să pot preveni orice problemă legată de vreo ieșire din parametri a acestora.

Este util în orice moment simțim că suntem stresați sau lipsiți de putere. Acesta ar putea fi primul semn că trebuie să mergem la un control la medic. În acest fel putem preveni diferite probleme medicale și ne putem recăpăta starea de bine și bună dispoziție

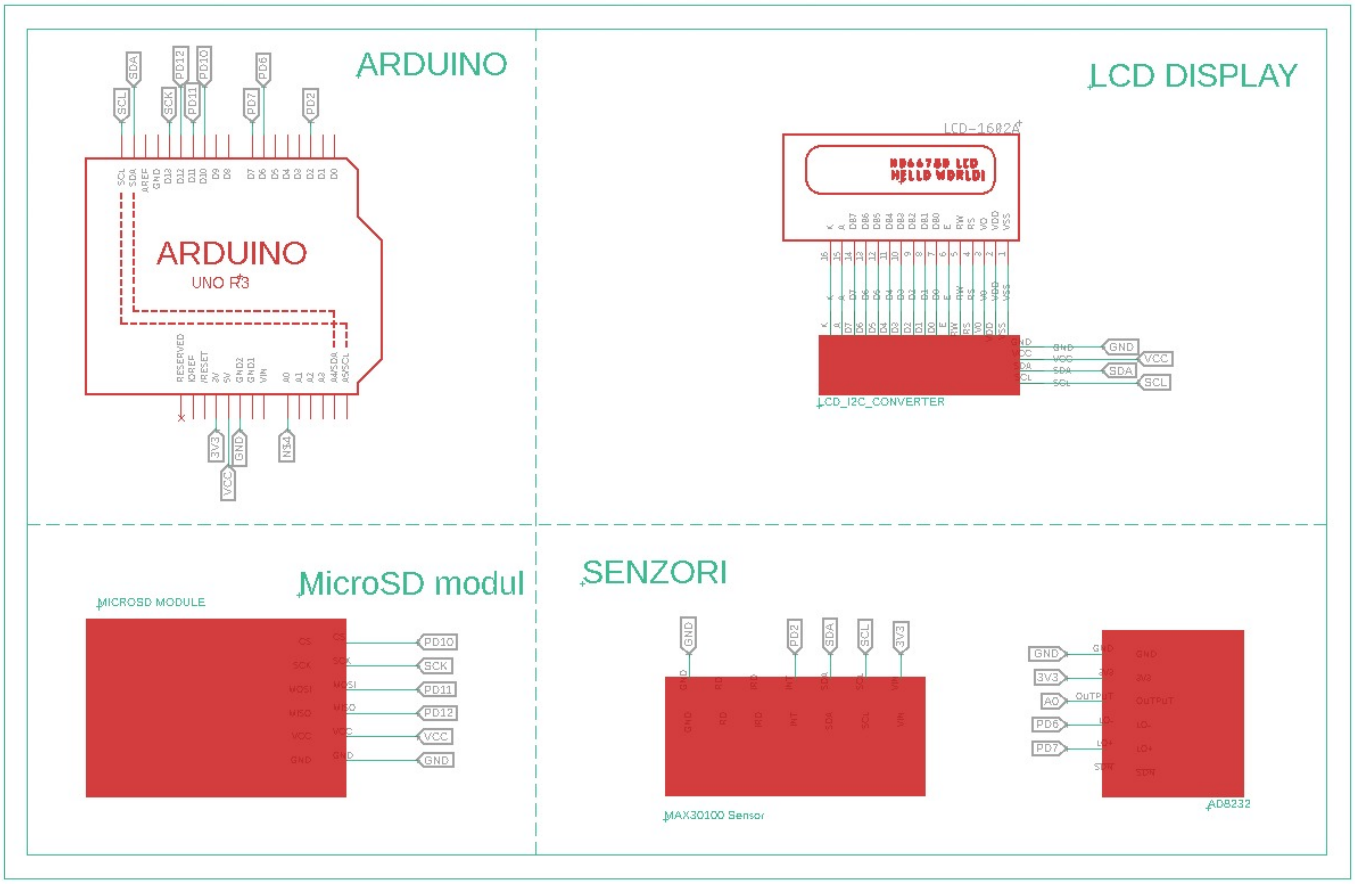
Descriere generală

Pe un ecran LCD voi afișa pulsul și nivelul oxigenului odată ce sunt înregistrate date. Cu rezultatele obținute prin senzorul de EKG voi realiza o diagramă care va afișa ritmul inimii. Datele vor fi înregistrate și salvate pe un card microSD, specificându-se momentul de timp când s-au făcut măsurătorile.

Schemă bloc

Hardware Design

Schemă electrică:



Componente hardware folosite:

- Arduino UNO R3 ATmega328P
- Modul senzor EKG AD8232
- Modul senzor ritm cardiac MAX30100
- Modul microSD
- Ecran LCD 1602
- Breadboard

Software Design

Biblioteci folosite:

- **LiquidCrystal_I2C.h** → LCD display
- **MAX30100_PulseOximeter.h** → senzorul de pulsoximetru
- **SD.h** + **SPI.h** → modulul de card microSD

Define-uri:

```
// Chip select pin for the microSD module
#define SD_CS_PIN               10

// Constant to delay the measurements
#define REPORTING_PERIOD_MS     1000
```

```
// Connect ECG LO- to PD6 pin
#define LO_MINUS PD6

// Connect ECG LO+ to PD7 pin
#define LO_PLUS PD7

// Connect ECG OUTPUT to A0 pin
#define OUTPUT_ECG A0
```

Obiectele componentelor:

```
// LCD display
extern LiquidCrystal_I2C lcd;

// Pulse oximeter sensor
extern PulseOximeter pox;

// microSD module
extern File myFile;
```

Simboluri:

```
// Symbols used for LCD display
extern const byte heart[8];
extern const byte two[8];
extern const byte gLetter[8];
```

Variabilele folosite:

```
// Flag used to mark when it's time to read the
// pulse oximeter sensor for the first time
extern bool firstRead;

// Threshold value for heart rate
extern const float threshold;

// Sum of valid heart rates
extern float heartRate;

// Sum of valid SpO2 values
extern int spo2;

// The number of relevant reads
extern int relevantReads;

// The number of irrelevant reads
extern int failures;

// Counter used for resetting the reads of sensor
extern int resetM;
```

```
// Timer to store the time at which the last beat occurred
extern uint32_t tsLastReport;

// Flags for LO+ and LO- interrupts
volatile bool flag_lo_m = 0;
volatile bool flag_lo_p = 0;
```

Inițializarea componentelor:

```
// Initialize LCD display
void setupLCD();

// Initialize pulse oximeter sensor
void setupPulseOximeter();

// Initialize microSD module
void setupSDmodule();

// Set the interrupts for ECG sensor
void setup_interrupts();
```

Pentru comunicarea cu **ecranul LCD**, voi folosi cunoștințe din laboratorul 6, I2C, ce este un protocol de comunicație serială sincron, multi-master - multi-slave. O magistrală I2C utilizează următoarele semnale:

- SDA - linia de date
- SCL - semnalul de clock

Funcțiile folosite pentru LCD display:

```
// Print initial message on the LCD
void printInitialMessage();

// Print message when the sensor starts measuring
void printStartMeasuringMessage();

// Print the heart rate
void printHeartRate();

// Print the SpO2
void printSpO2();
```

Legătura cu **modulul de card microSD** se face prin protocolul SPI ce prezintă următoarele patru semnale:

- MOSI — Master Output Slave Input (transmiterea datelor de la Master la Slave)
- MISO — Master Input Slave Output (transmiterea datelor de la Slave la Master)
- SCLK — Serial Clock (sincronizarea dintre dispozitive. Controlat de Master)
- CS/SS — Chip Select/Slave Select (selectarea dispozitivului Slave de către Master)

Scrierea pe card am realizat-o cu ajutorul funcției *writeDataToSD()*, unde am făcut media aritmetică a tuturor citirilor relevante pentru nivelul de oxigen si puls obținute de citirile senzorului:

```
void writeDataToSD()
{
  myFile = SD.open("heart.txt", FILE_WRITE);

  if (myFile) {
     char text[20] = "BPM: ";
     float result = heartRate / relevantReads;
     char buffer[6];
     dtostrf(result, 5, 2, buffer);
     strncat(text, buffer, sizeof(buffer));
     strncat(text, " - SPO2: ", sizeof(" - SPO2: "));
     result = spo2 / relevantReads;
     dtostrf(result, 3, 0, buffer);
     strncat(text, buffer, sizeof(buffer));
     strncat(text, "%", sizeof("%"));

     myFile.println(text);
     myFile.close();
  }
}
```

Senzorul **MAX30100** are următorul flow:

- Update pentru a se realiza citirile

```
// Read from the sensor
  pox.update();
```

- Citesc la fiecare secunda (lucru redat de define-ul REPORTING_PERIOD_MS) datele de pe senzor

```
if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
   // Read the values from the sensor
   float currHeartRate = pox.getHeartRate();
   uint8_t currSpO2 = pox.getSpO2();

   // ...

   tsLastReport = millis();
}
```

- Când s-a făcut o înregistrare în care una dintre cele două valori citite este diferită de 0, atunci semnalez și pe display că încep măsurătorile:

```
// When the sensor reads something for the first time
if (firstRead == 1) {
   // If one of the values is not 0, start measuring
   if (currHeartRate != 0.0 || currSpO2 != 0) {
      startMeasuring();
   }
```

```
}
```

- Dacă ambele valori sunt 0, incrementez variabila care va reseta toate valorile la 20 de astfel de citiri; când ajung la 20 de citiri pe 0, verific dacă există cel putin o citire relevantă, iar valorile le stochez pe card. Această limitare mă ajută să reiau măsurătorile de la capăt fără să țin cont de ce am calculat anterior, iar salvarea pe card se face o singură dată pentru o serie de măsurători. Altfel, dacă una dintre valori este 0, dau discard la acea citire și când ajung la 3 citiri greșite, resetez variabilele:

```
// If the sensor measures 0 for both values, increment the
// counter for resetting all values
if (currHeartRate == 0.0 && currSpO2 == 0) {
   resetM++;
   // When the counter reaches 20, reset all values
   if (resetM == 20) {
      goToFirstRead();
   }
} else if (currHeartRate == 0.0 || currSpO2 == 0) {
   // If one of the values isn't read correctly, increment the
   // counter for failed reads
   failures++;
   if (failures == 3) {
      // If there are 3 failed reads, reset the measurements
      resetValues();
   }
}
```

- Dacă trec de aceste verificări, impun ca valoarea citită pentru puls să depășească un threshold (55.0), iar nivelul de sânge să fie pozitiv, iar când aceste condiții sunt îndeplinite, afișez pe ecranul LCD valorile calculate ca media aritmetică a tuturor citirilor relevante de până atunci:

```
// If both values are read correctly, add them to the
// sum of values for calculating the average afterwards
if (currHeartRate > threshold && currSpO2 > 0) {
   addValuesToSum(currHeartRate, currSpO2);

   // Reset the counter for resetting the values
   resetM = 0;

   // Reset the counter for failed reads
   failures = 0;

   // Increment the number of relevant reads
   relevantReads++;

   // Print the values on the LCD
   printHeartRate();
   printSpO2();
}
```

Modulul de senzor EKG va folosi un protocol ADC cu ajutorul căruia se va converti o tensiune

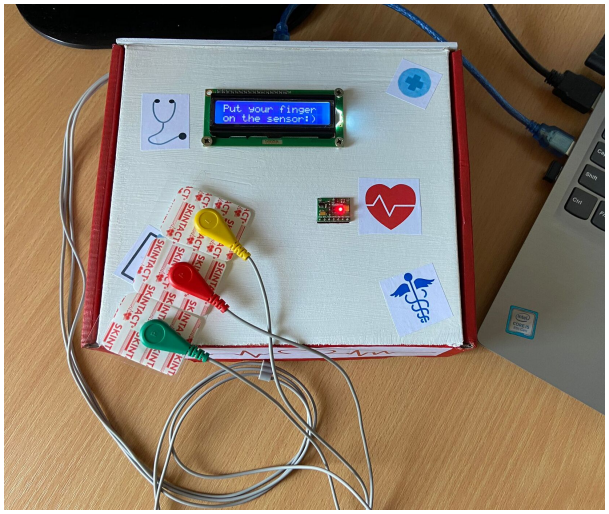
analogică de la intrare într-o valoare digitală. Am folosit întreruperi de tip pin change pentru a controla pinii LO+ și LO-. Fiecare are câte un flag corespunzător care se activează dacă se face o citire pe pin. Altfel, flag-ul se resetează.

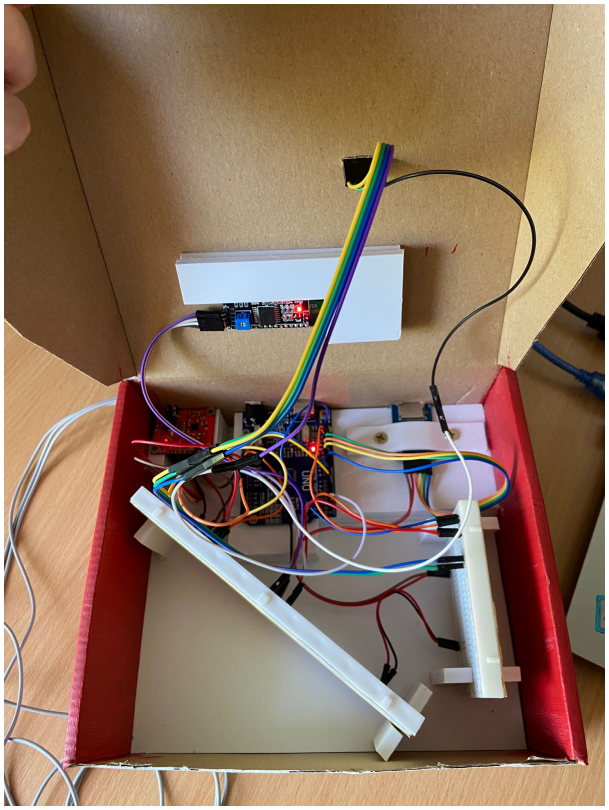
```
// Pin Change Interrupt 2
ISR(PCINT2_vect) {
  if ((PIND & (1 << LO_MINUS)) == 0) {
     flag_lo_m = 1;
  } else if ((PIND & (1 << LO_MINUS)) != 0) {
    flag_lo_m = 0;
  }

  if ((PIND & (1 << LO_PLUS)) == 0) {
     flag_lo_p = 1;
  } else if ((PIND & (1 << LO_PLUS)) != 0) {
    flag_lo_p = 0;
  }
}
```

Citirea semnalului analog se face când ambele semnale de pe pinii digitali sunt citite. Electrocardiograma va avea afișată pe ecranul laptopului o reprezentare grafică a datelor interpretate, iar conectivitatea dintre Arduino și laptop o voi realiza prin protocolul USART, folosind Serial Plotter pentru afișarea semnalului EKG.

```
void readECG() {
  // Only when both flags are set, read the analog value
  if (flag_lo_m == 1 && flag_lo_p == 1) {
    Serial.println(analogRead(OUTPUT_ECG));
  }
}
```

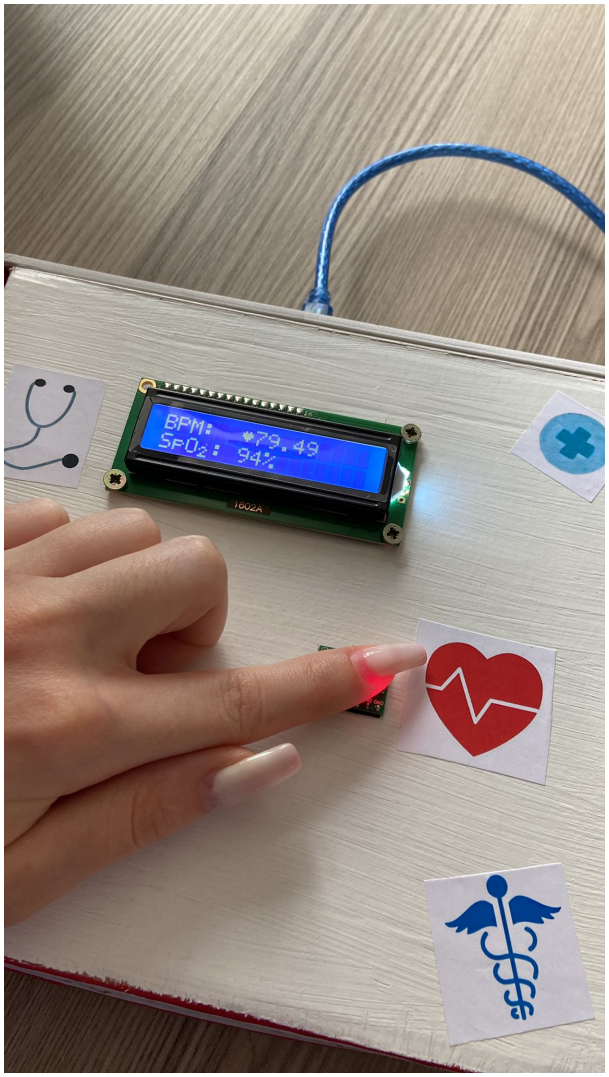
Rezultate Obţinute





Semnalul ECG:

Rezultatele returnate de senzorul de pulsoximetru:



Datele salvate pe cardul microSD:

```
BPM: 66.64 - SPO2:  98%
BPM: 61.21 - SPO2:  99%
BPM: 63.20 - SPO2:  98%
BPM: 79.26 - SPO2:  98%
BPM: 94.19 - SPO2:  98%
BPM: 76.11 - SPO2:  95%
BPM: 65.48 - SPO2:  95%
BPM: 73.26 - SPO2:  96%
BPM: 77.52 - SPO2:  96%
```

Concluzii

Mi-a făcut plăcere să lucrez la acest proiect, a fost challenging și interesant, am descoperit cât de util este să te informezi înainte cu ce componente lucrezi, ce au nevoie, cum interacționează împreună. Partea cea mai frumoasă a proiectului a fost când am început să primesc primele semne de viață ale senzorilor și faptul că funcționează și am legat bine firele. Apoi am devenit din ce în ce mai confident cu asamblarea sa, fără să mai am teama că am legat firele pe dos. Mă bucur că am realizat un proiect pe care să îl pot folosi oricând voi dori, știind că este realizat de mine.

Am învățat cum este să testezi permanent funcționalitatea și cum să fiu atentă până la cel mai mic detaliu pentru cum arată codul si partea hardware. Overall, sunt mulțumită de progresul meu și de rezultatul final al proiectului.

Download

Demo-ul proiectului:
https://drive.google.com/drive/folders/1HiiBZKtpVVICN6hZpQ_ekqxAuGifKx-2?usp=sharing

Link GitHub pentru proiect: https://github.com/lauranicoara/Health-Kit

Jurnal

13 aprilie - alegerea proiectului

24 aprilie - achizitionarea pieselor

3 mai - crearea paginii de documentatie

17 mai - milestone hardware

25 mai - milestone software

28 mai - finalizare pagină documentație

Bibliografie/Resurse

Resurse Hardware:

- https://microcontrollerslab.com/ad8232-ecg-module-pinout-interfacing-with-arduino-applications-features/
- https://github.com/sparkfun/AD8232_Heart_Rate_Monitor/blob/master/Fritzing/AD8232_Heart_Rate_Monitor_Demo_bb_Fritzing_Corrected.jpg
- https://www.analog.com/media/en/technical-documentation/data-sheets/AD8232.pdf
- https://microcontrollerslab.com/max30100-pulse-oximeter-heart-rate-sensor-esp32/
- https://www.analog.com/media/en/technical-documentation/data-sheets/MAX30100.pdf

Resurse Software:

- https://lastminuteengineers.com/max30100-pulse-oximeter-heart-rate-sensor-arduino-tutorial/
- https://lastminuteengineers.com/arduino-micro-sd-card-module-tutorial/
- https://github.com/arduino-libraries/SD
- https://github.com/oxullo/Arduino-MAX30100
- https://how2electronics.com/ecg-monitoring-with-ad8232-ecg-sensor-arduino/

Export to PDF