Politechnika Rzeszowska
Wydział Elektrotechniki i Informatyki
Katedra Podstaw Elektroniki

Monitor zdrowia na bazie Arduino

Natalia Klocek
EFDI1 nr 177097

Spis treści

1. Wprowadzenie

Celem projektu jest zbudowanie pulsoksymetru – urządzenia medycznego mierzącego puls oraz saturację tlenu we krwi. Projekt ten został stworzony, aby lepiej zrozumieć proste układy elektryczne z wykorzystaniem Arduino oraz, aby nauczyć się korzystać z czujników i wyświetlaczy do monitorowania parametrów zdrowotnych.

Funkcje i wykorzystanie

Głównymi funkcjami projektu jest użycie czujnika tętna do wyświetlania pulsu oraz saturacji krwi na ekranie LCD. Dodatkowymi funkcjami w projekcie są:
- Tryb sportowy, który można zmieniać za pomocą przycisku.
- Diody LED RGB, które zmieniają kolor w zależności od wartości tętna.
- Brzęczyk, który ma informować kiedy puls jest w niebezpiecznych strefach.

Projekt ten można wykorzystać w warunkach domowych do monitorowania zdrowia.

2. Wykorzystane elementy

Arduino Uno R3 SMD

Jest to klon Arduino Uno R3, natomiast ma identyczne funkcje, co oryginalna płytka. Jedyną różnicą jest, że używa płaskiego mikrokontrolera ATmega328P w wersji SMD (Surface-Mount Device), który jest lutowany bezpośrednio na płytce, zamiast w wersji DIP (Dual In-line Package), która jest wtykana w podstawkę.

Specyfikacje:

- Mikrokontroler: ATmega328P
- Konwerter USB/Serial: CH340
- Napięcie pracy: 5V
- Zewnętrzne napięcie zasilania (zalecane): 7-12V
- Wyjścia cyfrowe I/O: 14 (z czego 6 obsługuje PWM)
- Wejścia analogowe: 6
- Wydajność prądowa na pin I/O: 20 mA

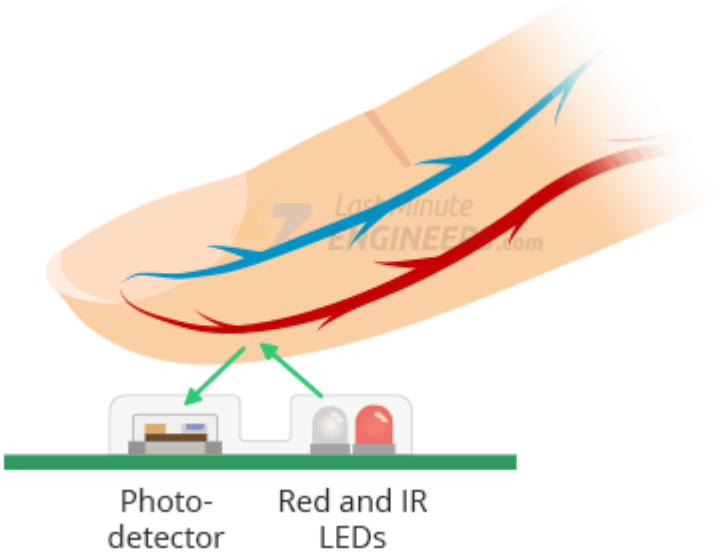
- Pamięć flash: 32 KB
- Pamięć SRAM: 2 KB
- Pamięć EEPROM: 1 KB
- Częstotliwość pracy: 16 MHz

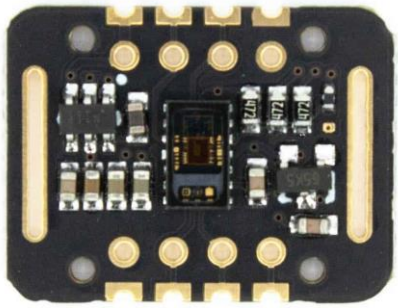
Aby pracowało można podłączyć go do komputera poprzez kabel USB lub zasilić baterią. Dodatkowo jeśli jest to potrzebne można rozszerzyć jego możliwości poprzez użycie nakładek.

Moduł czujnika tętna i SPO2 MAX30102

Jest to czujnik optyczny, który mierzy absorpcję pulsującej krwi za pomocą fotodetektora po wyemitowaniu dwóch długości fal światła z dwóch diod LED - jednej czerwonej i jednej podczerwonej. Dane można odczytać przy użyciu palca, ponieważ obie długości światła łatwo przenikają przez cienką tkankę, a ilość odbitego światła jest mierzona za pomocą fotodetektora. Krew jest pompowana przez palec przy każdym uderzeniu serca, ilość odbitego światła się zmienia, co tworzy zmieniającą się falę na wyjściu fotodetektora. Kontynuując naświetlanie i pobieranie odczytów z fotodetektora, otrzymuje się odczyt pulsacji serca. Pulsoksymetry opierają się na zasadzie, że ilość światła czerwonego i podczerwonego absorbowana przez krew zmienia się w zależności od ilości tlenu we krwi, co daje odczyt SPO2. Aby wykorzystać wszystkie potrzebne funkcje tego czujnika należy użyć bibliotek MAX30105 oraz spo2_algorithm. W projekcie wykorzystywane są 4 piny modułu: VIN

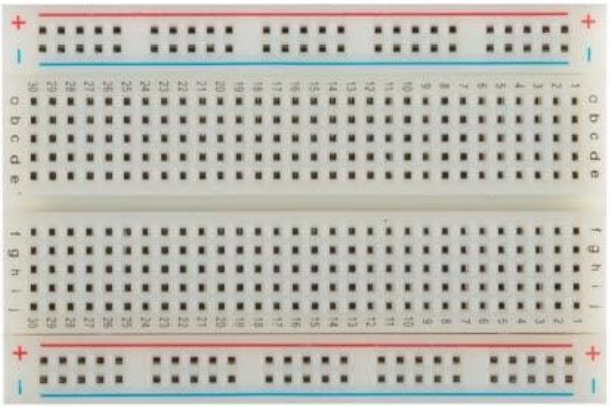
odpowiedzialny za zasilanie, SCL i SDA odpowiedzialne za komunikację między mikrokontrolerem oraz urządzeniem poprzez interfejs I2C, oraz GND odpowiedzialny za masę.





Płytka stykowa

Można ją wykorzystać jako zamiennik lutowania, jest wielorazowa i bardzo prosta w użyciu. Mają one różne rozmiary, natomiast płytka wykorzystana w projekcie posiada 830 otworów. W jej plastikowej obudowie znajdują się blaszki, które można potraktować jako krótkie przewody, które łączą elementy.



Ekran LCD

Wybrany model może wyświetlić dwie linie po 16 znaków. Posiada on 16 pinów oznaczających odpowiednio:

VSS - masa

VDD - zasilanie dodatnie, 5V

V0 - regulacja kontrastu

RS - wybór rejestrów

RW - wybór opcji odczyt/zapis

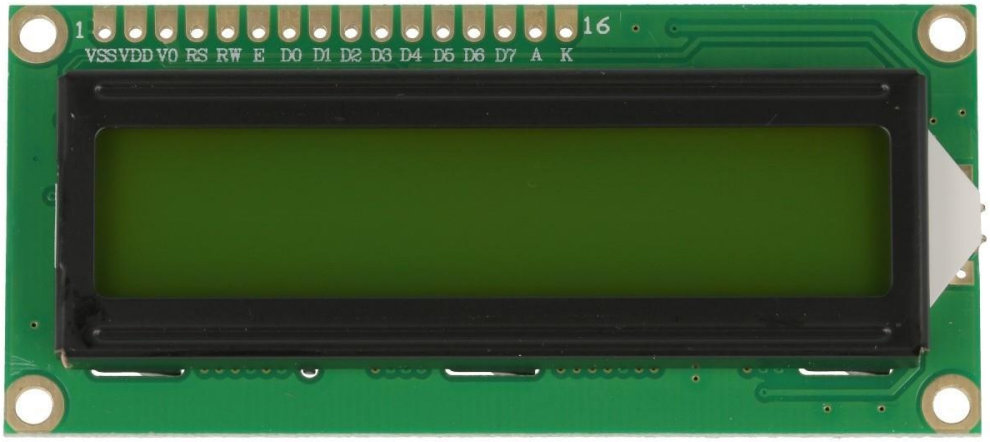
E - zezwolenie na zapis do rejestrów

D0 do D7 - dane

A - zasilanie dodatnie podświetlenia

K - masa podświetlenia

Na potrzeby projektu wykorzystane zostaną wszystkie piny oprócz D0 do D3. Do sterowania ekranem wykorzystuje się bibliotekę LiquidCrystal. Aby poprawnie działał należy dodać potencjometr 10k oraz opcjonalnie rezystor. Potencjometr jest ważny aby ustawić odpowiedni kontrast na ekranie, ponieważ bez niego nie będzie widać znaków. Natomiast rezystor można podłączyć do zasilania podświetlenia aby ekran nie był za jasny.



Moduł z diodą LED RGB - Iduino SE010

Diody LED RGB są najczęściej sterowane za pomocą modulacji PWM. Zazwyczaj stosuje się 24 bitową metodę opisu koloru gdzie na każdą z trzech składowych barw przypada 8 bitów. Przez co poszczególne barwy składowe mogą przyjmować jedną z wartości w skali od 0 do 255. Dlatego wykorzystany moduł posiada złącze 4-pinowe, 3 ustalające kolor i 1 do masy. Dzięki sposobowi wykonania modułu nie trzeba dopinać do niego żadnych rezystorów, co normalnie zawsze jest potrzebne w diodach LED aby ich nie uszkodzić.

Brzęczyk aktywny

Posiada wbudowany generator, więc nie potrzebuje zewnętrznego sygnału sterującego i wytwarza dźwięk już od podania zasilania. Ma różne poziomy zasilania, natomiast w projekcie jest użyty typ 5V. Częstotliwość dźwięku jest ciągła i nie można jej zmieniać. Jeśli jest za głośny to można go przyciszyć dodając rezystory między jego pin zasilania, co zostało użyte w projekcie podłączając dwa rezystory 1k Ohm szeregowo.

Przycisk

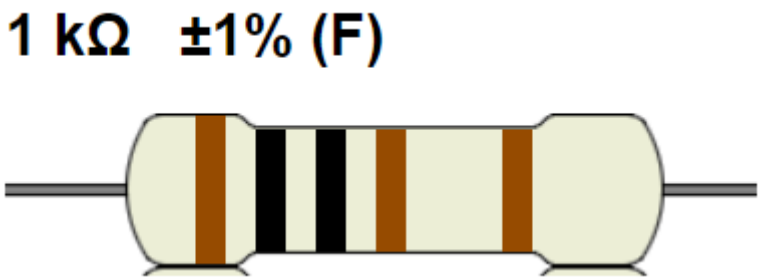
Przycisk podpina się do pinu, z którego będzie odczytywany jego stan, oraz masy. Podczas naciskania przycisku jego stan się zmienia, co idealnie pasuje do instrukcji warunkowych. W projekcie dodano rezystor między pinem przycisku a masą, aby zapewnić stabilny i niezakłócony odczyt stanu przycisku.

Potencjometr obrotowy

Potencjometr w projekcie jest wykorzystywany jedynie jako dzielnik napięcia do ustawienia kontrastu ekranu LCD. Dzielnik napięcia to urządzenie, dzięki któremu możliwe jest uzyskanie pożądanego stosunku między napięciem wejściowym a wyjściowym. Wartość tą zmienia się za pomocą ślizgacza.

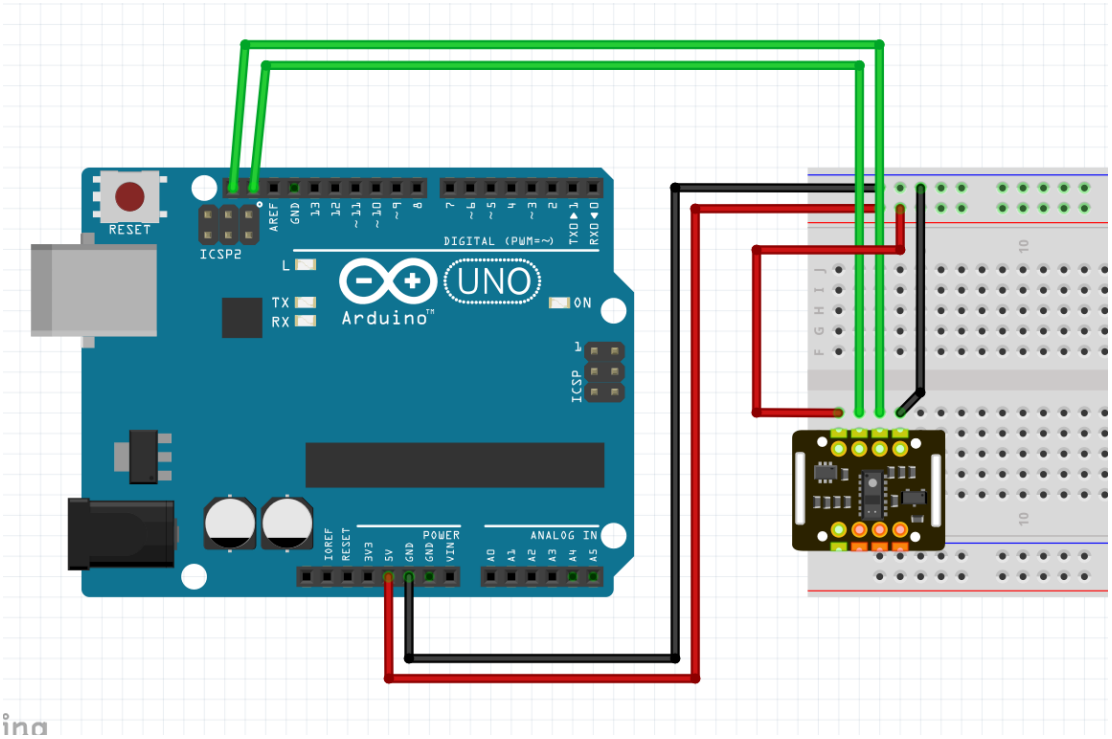
Rezystory

Ograniczają natężenie prądu płynącego w połączeniu, gdzie zostały włączone. Pracują zgodnie ze wzorami Ohma. W projekcie są używane cztery rezystory 1k Ohm.

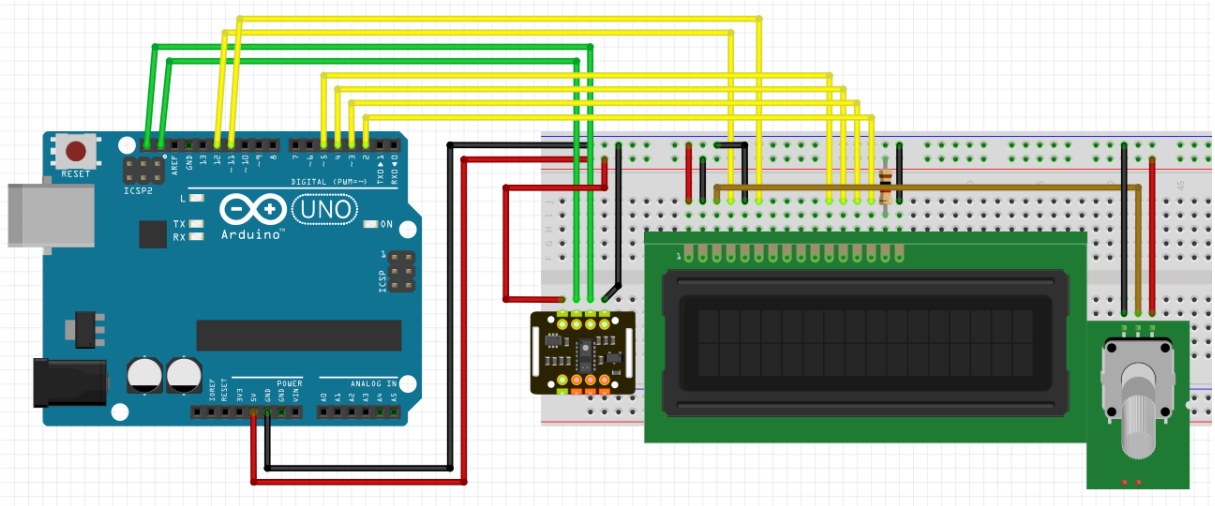


3. Montaż układu

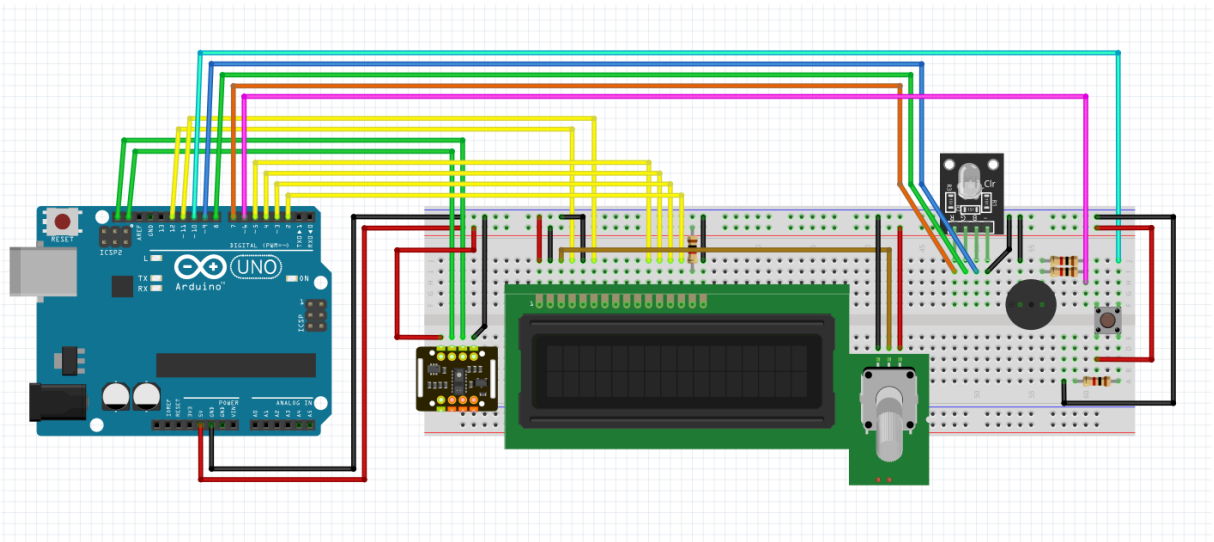
1. Na początku należy zacząć od najważniejszego elementu układu, czyli modułu czujnika tętna. Aby to zrobić trzeba najpierw podpiąć pin GND w Arduino z wierszem ze znakiem minus na płytce stykowej, oraz pin 5V z wierszem ze znakiem plus. W ten sposób Wszystko związane z masą i zasilaniem będzie podpinane do płytki stykowej, a nie do Arduino, dzięki czemu wystarczy miejsca na wszystko. Następnie pin VIN z modułu czujnika należy podłączyć do wiesza z plusem na płytce stykowej oraz GND do wiersza z minusem. SCL należy podłączyć do pinu SCL w Arduino oraz tak samo trzeba podłączyć SDA z SDA. Oba te połączenia są odpowiedzialne za komunikację między mikrokontrolerem oraz urządzeniem poprzez interfejs I2C.



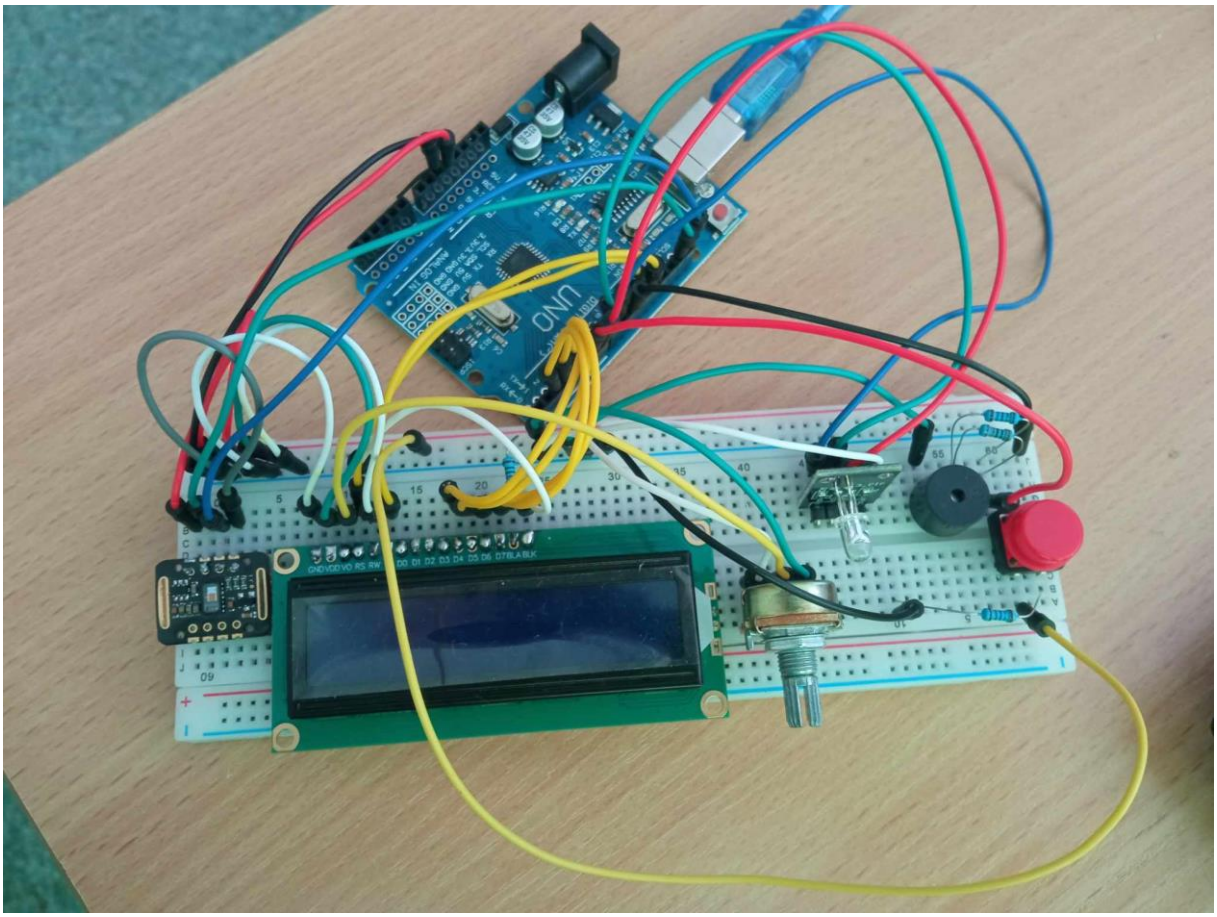
2. Następnie należy podłączyć również bardzo ważny element w projekcie – ekran LCD. VDD oraz należy podłączyć do wiersza z plusem w płytce stykowej, oraz tak samo z A, jednak ono powinno być również połączone przez rezystor 1k Ohm, aby ograniczyć napięcie, dzięki czemu ekran nie będzie za jasny. Następnie podłączamy piny GND, RW oraz K powinny zostać podłączone do płytki stykowej w wierszu z minusem. Odpowiednio pin RS podłączamy do 12, E do 11, D4 do 5, D5 do 4, D6 do 3 oraz D7 do 2, piny te można również przypisać do innych ponumerowanych wejść w Arduino, pamiętając o tym podczas następnych kroków, aby wszystko dobrze podłączyć. Na sam koniec trzeba podłączyć V0 do potencjometru, który również trzeba wcześniej podłączyć do płytki. Ekran LCD należy podłączyć do środkowej nóżki potencjometru, natomiast lewą nóżkę potencjometru należy podłączyć do wiersza z minusem, a prawą do wiersza z plusem (patrząc od strony ślizgacza).



3. W ostatnim kroku została do dodania reszta elementów, które służą dla celów informacyjnych i interaktywnych. Moduł diody LED RGB należy podłączyć odpowiednio tak jak są oznaczone piny, GND do wiersza z minusem, R do 7, G do 8 oraz B do 9. Tutaj również można podłączyć diodę do innych ponumerowanych wejść Arduino, pamiętając aby później dobrze je skonfigurować w kodzie. Następnie do układu należy dołączyć brzęczyk podpinając nóżkę od strony ze znakiem plus do pinu 6, łącząc je poprzez dwa rezystory 1k Ohm połączone szeregowo, aby wydawał cichsze dźwięki. Drugą nóżkę trzeba podłączyć do wiersza z minusem na płytce stykowej. Ostatnim elementem do podłączenia jest przycisk, który z jednej strony należy podłączyć do pinu 10, z drugiej poprzez rezystor 1k Ohm do wiersza z minusem w płytce stykowej, oraz z trzeciej do wiersza z plusem. Czwarta nóżka nie jest używana.



Jak powinien wyglądać skończony schemat.



Zdjęcie przedstawia poglądowy wygląd gotowego projektu.

4. Zaprogramowanie pulsoksymetru

Kod umożliwia odczyt danych z czujnika, oblicza saturację tlenem i tętno, a następnie wyświetla te dane na wyświetlaczu LCD. Dodatkowo sprawdza w jakich strefach jest tętno oraz sprawdza w jakim trybie działa. Kod został napisany w Arduino IDE przy pomocy języka C.

Kod źródłowy

```
#include <Wire.h>
#include "MAX30105.h"
#include "spo2_algorithm.h"
#include <LiquidCrystal.h>

MAX30105 particleSensor;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
#define MAX_BRIGHTNESS 255
uint16_t irBuffer[100]; //dane z czujnika diod LED na podczerwień
uint16_t redBuffer[100];  //dane z czerwonej diody LED

int32_t bufferLength; //długość daty
int32_t spo2; //SPO2
int8_t validSPO2; //sprawdzenie czy SPO2 może być poprawne
```

```
int32_t heartRate; //puls
int8_t validHeartRate; //sprawdzenie czy puls jest poprawny

byte pulseLED = 11;
byte readLED = 13;
int redPin= 7;
int greenPin= 8;
int bluePin= 9;
int buzzerPin = 6;
int buttonPin = 10;
bool sportMode = false;

void setup()
{
  Serial.begin(115200); //115200 baud
  pinMode(redPin,  OUTPUT); //wszystkie piny
  pinMode(greenPin,  OUTPUT);
  pinMode(bluePin,  OUTPUT);
  pinMode(buzzerPin,  OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  digitalWrite(buzzerPin, LOW);
  lcd.begin(16, 2);
  pinMode(pulseLED, OUTPUT);
  pinMode(readLED, OUTPUT);

  //inicjalizacja czujnika
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //port I2C
  {
    Serial.println(F("MAX30105 nie odnaleziony"));
    while (1);
  }
  //Opcje zalecane przez producenta
  byte ledBrightness = 60; //Options: 0=Off to 255=50mA
  byte sampleAverage = 4; //Options: 1, 2, 4, 8, 16, 32
  byte ledMode = 2; //Options: 1 = Red only, 2 = Red + IR, 3 = Red + IR + Green
  byte sampleRate = 100; //Options: 50, 100, 200, 400, 800, 1000, 1600, 3200
  int pulseWidth = 411; //Options: 69, 118, 215, 411
  int adcRange = 4096; //Options: 2048, 4096, 8192, 16384

  particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate, pulseWidth,
adcRange); //konfiguracja czujnika

}

void loop()
{
  if (digitalRead(buttonPin) == HIGH) {
    sportMode = !sportMode;
  }
  bufferLength = 100;

  for (byte i = 0 ; i < bufferLength ; i++)
```

```
  {
   while (particleSensor.available() == false)
    particleSensor.check(); //sprawdź czy są nowe dane

   redBuffer[i] = particleSensor.getRed();
   irBuffer[i] = particleSensor.getIR();
   particleSensor.nextSample(); //przejdź do następnej próbki

   Serial.print(F("red="));
   Serial.print(redBuffer[i], DEC);
   Serial.print(F(", ir="));
   Serial.println(irBuffer[i], DEC);
   if (digitalRead(buttonPin) == HIGH) {
    sportMode = !sportMode;
   }
  }

  //wylicz puls i SP2 po pierwszych 100 próbkach
  maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength, redBuffer, &spo2,
&validSPO2, &heartRate, &validHeartRate);

  while (1)
  {
   for (byte i = 25; i < 100; i++)
   {
    redBuffer[i - 25] = redBuffer[i];
    irBuffer[i - 25] = irBuffer[i];
    if (digitalRead(buttonPin) == HIGH) {
   sportMode = !sportMode;
  }
   }
   for (byte i = 75; i < 100; i++)
   {
    if (digitalRead(buttonPin) == HIGH) {
   sportMode = !sportMode;
  }
    while (particleSensor.available() == false)
     particleSensor.check();

    digitalWrite(readLED, !digitalRead(readLED)); //miganie diodą kiedy pobierane są dane

    redBuffer[i] = particleSensor.getRed();
    irBuffer[i] = particleSensor.getIR();
    particleSensor.nextSample();

    //prześlij próbki i wyniki przez UART aby sprawdzić, czy dobrze działa
    Serial.print(F("red="));
    Serial.print(redBuffer[i], DEC);
    Serial.print(F(", ir="));
    Serial.print(irBuffer[i], DEC);

    Serial.print(F(", HR="));
```

```
    Serial.print(heartRate, DEC);

    Serial.print(F(", HRvalid="));
    Serial.print(validHeartRate, DEC);

    Serial.print(F(", SPO2="));
    Serial.print(spo2, DEC);

    Serial.print(F(", SPO2Valid="));
    Serial.println(validSPO2, DEC);
   }

   maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength, redBuffer, &spo2,
&validSPO2, &heartRate, &validHeartRate);
   if(heartRate != -999)
   {
    lcd.setCursor(0,0);
    lcd.print("BPM: ");
    lcd.print(heartRate);
    lcd.print("     ");
   }
   if(spo2 != -999)
   {
    lcd.setCursor(0,1);
    lcd.print("SPO2: ");
    lcd.print(spo2);
    lcd.print("     ");
   }
   checkRange(heartRate);
   if (digitalRead(buttonPin) == HIGH) {
   sportMode = !sportMode;
  }
  }
}
void setColor(int redValue, int greenValue,  int blueValue) { //zmiana koloru LED RGB
  analogWrite(redPin, redValue);
  analogWrite(greenPin,  greenValue);
  analogWrite(bluePin, blueValue);
}

void checkRange(int32_t hb){ //zmiana LED RGB i buzzera w zależności od pulsu
  if(!sportMode){
  if (hb < 50 || hb > 120) {
    setColor(255, 0, 0); // czerwony
    digitalWrite(buzzerPin, HIGH);
  } else {
    setColor(0, 255, 0); // zielony
    digitalWrite(buzzerPin, LOW);
  }
  }
  if(sportMode) {
    digitalWrite(buzzerPin, LOW);
```

```
    if (hb < 60 || hb > 220) {
    setColor(255, 0, 0); // czerwony
    digitalWrite(buzzerPin, HIGH);
   } else if (hb >= 60 && hb < 110){
    setColor(0, 255, 0); //zielony
   }
    else if (hb >= 110 && hb < 130){
    setColor(255, 255, 0); //żółty
    }
    else if (hb >= 130 && hb < 170){
    setColor(200, 200, 100); //biały
    }
    else {
    setColor(255, 0, 255); //fioletowy
   }
   }
}
```

Omówienie działania

Na samym początku należy dodać biblioteki z których będziemy korzystać. Następnie należy zadeklarować potrzebne zmienne oraz, dla własnej wygody, podpisać wszystkie piny numeryczne które są wykorzystywane i ustawić tryb wyjściowy aby mogły otrzymywać informacje z programu. W kolejnych linijkach jest inicjalizowany czujnik , jeśli został wykryty błąd to zostanie to powiadomione na konsoli, a program nie będzie kontynuował. W nieskończonej pętli są pobierane dane oraz są wyliczane wartości tętna i SPO2 w wbudowanej funkcji z biblioteki. Jeśli zmienna sprawdzająca, czy wyniki mogą być poprawne wynosi jeden to wtedy zostają one wypisane na ekranie LCD. Zaraz po tym jest sprawdzane tętno w funkcji checkRange, aby odpowiednio zmienić kolor diody LED RGB oraz, aby wyłączyć lub włączyć brzęczyk. Od wyniku tej funkcji zależy również, czy tryb sportowy jest włączony, co można zmieniać w wielu miejscach w kodzie, aby zapewnić, że w każdym momencie przyciśnięcia tryb zostanie zmieniony. Dodatkowo program wypisuje wszystkie pobierane i obliczane dane na monitorze w Arduino IDE w celu debugowania.

Wykorzystane biblioteki

LiquidCrystal - https://www.arduino.cc/reference/en/libraries/liquidcrystal/ (do ekranu LCD)

Wire - https://www.arduino.cc/reference/en/language/functions/communication/wire/ (komunikacja portem I2C)

MAX30105 - https://github.com/opensensinglab/max30105/blob/master/src/MAX30105.h (do czujnika)

spo2_algorithm - https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library/blob/master/src/spo2_algorithm.h (do czujnika)

5. Kompilacja i uruchomienie

Aby poprawnie uruchomić projekt należy wkleić kod do Arduino IDE oraz podłączyć płytkę do komputera poprzez USB. Następnie trzeba wybrać poprawny port oraz płytkę w oknie koło opcji „Upload” i „Debug”. Po zakończonych tych krokach wystarczy kliknąć „Upload”, aby wysłać program do płytki. Jeżeli czerwona dioda na module czujnika się zaświeciła, to znaczy, że działa on poprawnie. Aby mieć pewność czy wszystko jest dobrze podpięte należy używać czujnika i obserwować zmiany w użytych elementach oraz na monitorze w Arduino IDE. Jeżeli występuje jakiś błąd to należy jeszcze raz sprawdzić wszystkie połączenia, czy kod jest dobrze skopiowany, lub czy płytka jest dobrze podpięta.

6. Znane błędy.

Znane są obecnie dwa błędy, jeden mniej istotny i drugi bardziej.

Przycisk

Problem polega na tym, że rzadko podczas przyciskania można trafić na moment w którym program tego nie wykryje. Można to zauważyć po kolorach diody RGB czy tryb został zmieniony, wystarczy wtedy przycisnąć jeszcze raz.

Czujnik

Podczas mierzenia tętna podaje się palec na moduł czujnika, z którego światło później zapisuje dane. Przez to, że człowiek nie jest w stanie utrzymać dokładnie takiego samego nacisku przez cały czas, wyniki przesyłane przez czujnik są często błędne. Najlepszym rozwiązaniem, według producenta, jest przymocowanie czujnika do palca za pomocą gumki recepturki.

7. Źródła

https://www.arduino.cc/reference/en/libraries/

https://lastminuteengineers.com/max30102-pulse-oximeter-heart-rate-sensor-arduino-tutorial/

https://www.calculator.net/resistor-calculator.html?bandnum=5&band1=brown&band2=black&band3=black&multiplier=brown&tolerance=brown&temperatureCoefficient=brown&type=c&x=Calculate

https://docs.arduino.cc/learn/electronics/lcd-displays/

https://forbot.pl/blog/jak-dziala-plytka-stykowa-zdjecia-budowa-przyklady-id21978

https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf