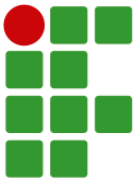


INSTITUTO FEDERAL
Brasília
Campus Brasília

TECNOLOGIA EM SISTEMAS PARA INTERNET

Mateus Lopes da Silva

Ruan Mateus de Souza Nunes

Wendel Maxuel Ribeiro Pereira

Vitor Gabriel Gaspar Rodrigues

RELATÓRIO DE PRÁTICA INTEGRADA
DE
CIÊNCIA DE DADOS E INTERNET DAS COISAS

Brasília - DF

13/12/2022

Sumário

1. Objetivos

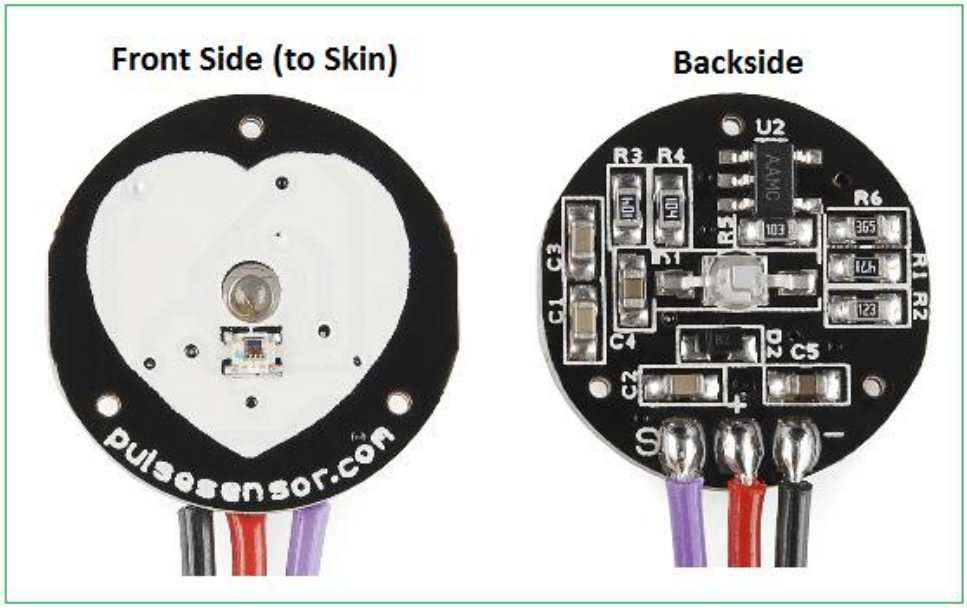
- Utilizar o sensor de **Monitoramento de Frequência Cardíaca** em conjunto com uma placa de Arduino Uno.
- Realizar a conexão do sensor na placa de Arduino.
- Implementar o código na IDE do Arduino
- Salvar os dados de teste do sensor em um arquivo csv.

2. Descrição do problema

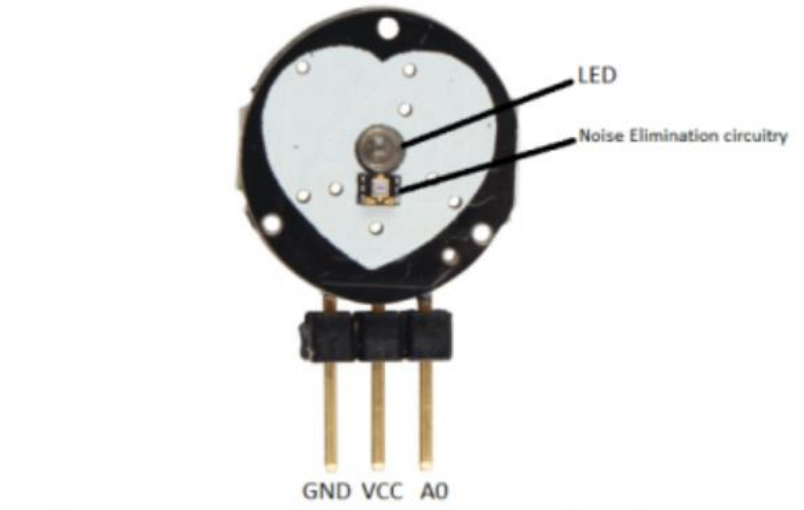
Aqui buscamos entender como fazer a montagem do hardware (sensor + Arduino), bem como conectar os 3 pinos do sensor nos lugares corretos na placa de Arduino, e instalar e configura a IDE que será utilizada para teste do código. Após isso, definir como será feito o armazenamento dos dados para a próxima etapa.

3. Desenvolvimento

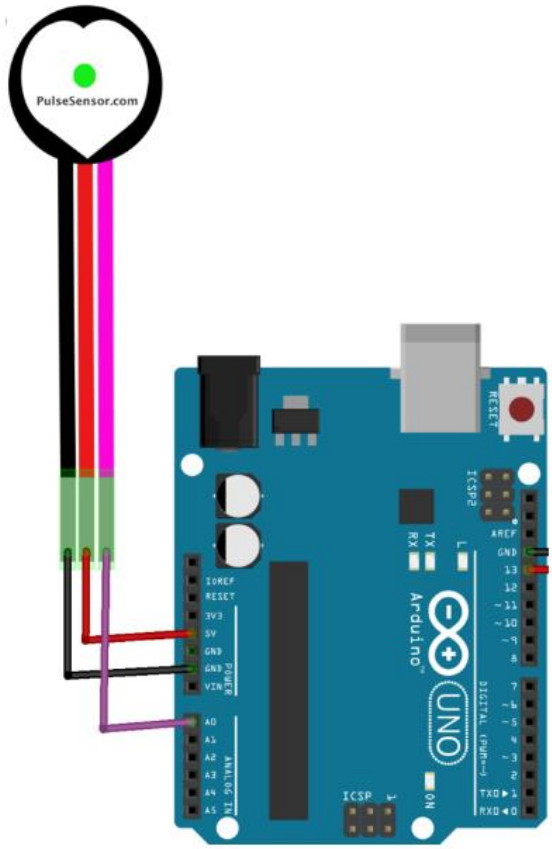
No desenvolvimento, após a montagem do hardware (sensor + Arduino) está concluída, partimos para pesquisar um código que se adequasse ao nosso objetivo. Depois de encontrar fizemos a implementação do código e verificamos o resultado conforme o vídeo visto. Após isso, ficamos em dúvida sobre qual método escolher para armazena os dados advindos do sensor, cogitamos o cartão SD mas descobrimos que para usá-lo teríamos que ter um módulo para sua utilização. No final encontramos um meio para armazena os dados em csv através do Python.



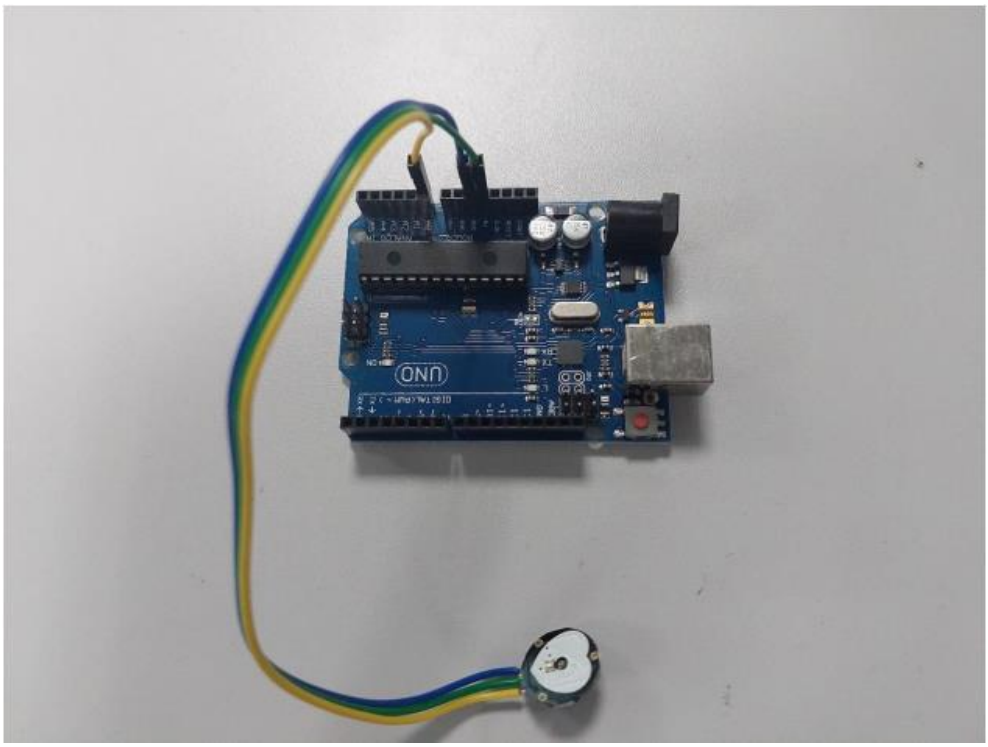
(Imagem 1 – Sensor de Pulso Cardíaco)



(Imagem 2 – Esquema de pinos (GND, VCC e Signal(A0)) do Sensor de Pulso Cardíaco)



(Imagem 3 – (sensor + Arduino) conectados de forma virtual)



(Imagem 4 – (sensor + Arduino) ambos conectados)

3.1 Código implementado

Código utilizado no Arduino para obtenção dos Batimentos Cardíacos:

```
int pin = A0;               // Definido o pino A0 como "pin"

float valorAnterior = 0;    // Definindo a variável que armazenará o valor da
leitura anterior
float valorMaximo = 0.0;    // Definindo a variável que armazenará 97% do valor
máximo obtido
int quantidadeBatidas = 0;  // Definindo a variável que armazenará a quantidade de
batimentos

float fatorFiltro = 0.75;   // Coeficiente para o filtro do valor analógico obtido
durante a leitura
int minimoEntreBatidas = 300;   // Valor mínimo de tempo entre os batimentos
cardíacos

long entreBatidas = millis();   // Definindo a variável local que armazenará o
tempo entre os batimentos
long tempoBPM = millis();       // Definindo a variável local "tempo de batimentos
por minuto em milissegundos

void setup() {
  Serial.begin(9600);           // Inicializando o monitor serial
  Serial.println("BPM");        // Imprime no monitor serial a mensagem que está
sendo passada como parâmetro
}

void loop() {
  int valorLido = analogRead(pin);    // Realizando a leitura do pino denominado A0

  float valorFiltrado = fatorFiltro * valorAnterior + (1 - fatorFiltro) *
valorLido;    // Realizando a filtragem do sinal analógico
  float valorDiferenca = valorFiltrado - valorAnterior;         // Calculando a
diferença entre a variavel valorFiltrado e valorAnterior

  valorAnterior = valorFiltrado;          // Atualizando a variavel valorAnterior
com o valor da variável valorFiltrado

  if ((valorDiferenca >= valorMaximo) &&            // Verificando se a variavel
valorDiferenca é maior que a variável valorMaximo
    (millis() > entreBatidas + minimoEntreBatidas)  // E se o tempo atual é maior
que a soma do tempo da última batida detectada e o tempo mínimo entre as batidas
  ) {
    valorMaximo = valorDiferenca;                   // Armazena na variável
valorMaximo o valor da variável valorDiferenca
    entreBatidas = millis();                        // Armazena o momento atual em
milissegundos na variável entreBatidas
    quantidadeBatidas++;                            // Incrementa mais um ao valor
armazenado na variável quantidadeBatidas
```

```
  }

  valorMaximo = valorMaximo * 0.97;                // Atualizando o valor da
variável valorMaximo com 97% do valor da variavel valorDiferenca

  if (millis() >= tempoBPM + 15000) {              // Verificando se já se
passaram 15 segundos para mostrar a quantidade de batimentos por minuto
    Serial.println(quantidadeBatidas * 4);         // Imprime no monitor serial a
quantidade de batidas multiplicadas por 4, pis a cada 15 segundos temos 1/4 do
minuto
    tempoBPM = millis();                           // Armazena o momento atual em
milissegundos na variável tempoBPM
    quantidadeBatidas = 0;                         // Atualizando a variavel
quantidadeBatidas com o valor zero para iniciar uma nova contagem
  }

  delay(50);                                       // Parando a execução do código
por 50 milissegundos
}
```

Link do código no GitHub:

https://github.com/infocbra/pratica-integrada-cd-e-ic-2022-2-g5-rwmv

4. Considerações finais

Conseguimos montar o hardware e testar o funcionamento do sensor, porém atrasamos para entregar as tarefas. Vamos focar na próxima sprint em agilizar a execução das tarefas e não repetir a postura que tivemos nessa sprint.

Referências

Pulso Cardiaco. **Blog ESPOL**, 2019.

Disponível em: http://blog.espol.edu.ec/edelros/categoy/arduino/pulso-cardiaco/

Acesso em: 20/12/2022

Monitor de taxa de pulso (BPM) usando Arduino e sensor de pulso. **Cap Sistema,** 2021.

Disponível em:https://capsistema.com.br/index.php/2021/02/03/monitor-de-taxa-de-pulso-bpm-usando-arduino-e-sensor-de-pulso/

Acesso em: 17/12/2022

Código Arduino utilizado:

https://www.youtube.com/watch?v=KGuUnhOulGw&t=526s

Código Python utilizado para criação do csv:

https://www.youtube.com/watch?v=UGjjP45wrKQ&t=903s