

PAS

A Pulseira de Assistência Sensorial (PAS) é um dispositivo desenvolvido em Arduino para auxiliar crianças com TEA. Ele monitora a frequência cardíaca da criança e possui um botão para situações de crise. Este guia fornecerá todas as informações necessárias para montar, programar e utilizar a PAS.

Como Usar:

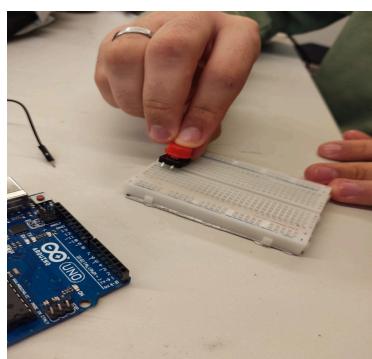
Materiais Necessários

Componentes Eletrônicos:

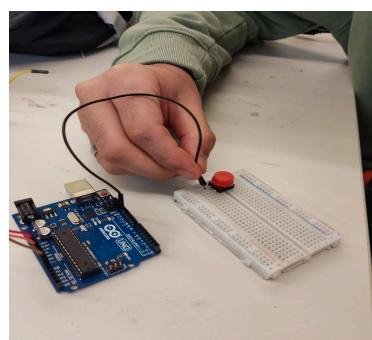
- 1 - Arduino Uno
- 1 - Sensor de frequência cardíaca
- 1 - Botão push button
- 4 - Fios Jumper
- 1 - Protoboard
- 1 - Led

Montagem do hardware

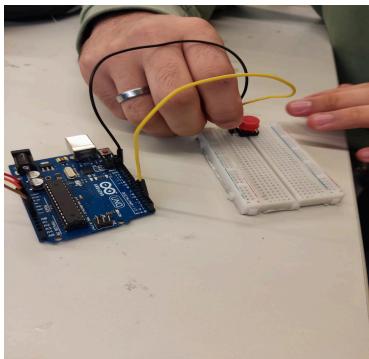
1. Conecte o botão:



- Coloque um botão no Jamboard:

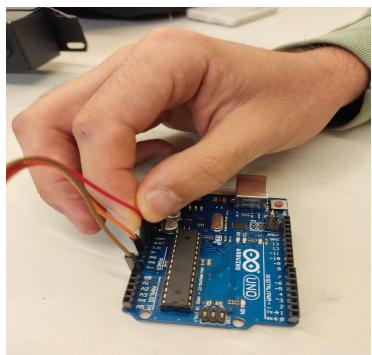


- Conecte um terminal ao GND do Arduino



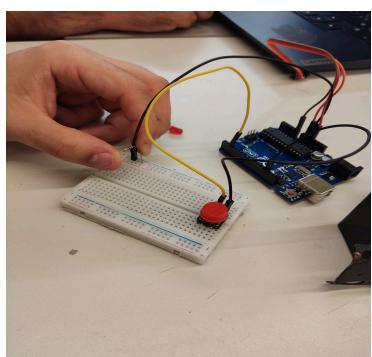
- Conecte o outro terminal ao pino digital 2 do Arduino

2. Conecte o sensor de Frequênciā cardíaca:

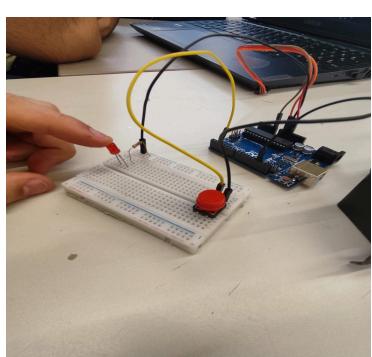


- Conecte o fio vermelho do sensor ao pino 5V do Arduino
- Conecte o fio preto do sensor ao pino GND do Arduino
- Conecte o fio roxo(sinal) ao pino A0 do Arduino

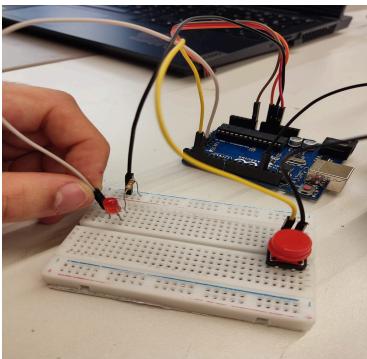
3. Conecte o Led:



- Conecte um terminal do botão ao GND do Arduino



- Conecte o resistor entre o Led e o GND



- Conecte o pino digital 3 ao terminal do Led

Configuração de Software

Programação do Arduíno

1.Bibliotecas

```
#include <PulseSensorPlayground.h>
```

2.Loop Principal

O código espera pelo botão ser pressionado ou que ocorra algum batimento no sensor de frequência cardíaca, caso o botão seja pressionado ele manda uma variável inteira representando o estado do botão pelo porte serial ao código python, caso seja detectado um batimento vai ser mandado pelo porte serial uma variável inteira representando o batimento obtido pelo sensor.

```
#include <PulseSensorPlayground.h>
```

```
// Definição dos pinos
```

```
const int pulsePin = A0; // Pino do sensor de batimento cardíaco
```

```
const int buttonPin = 2; // Pino do botão
```

```
const int ledPin = 3; // Pino da saída para o LED
```

```
PulseSensorPlayground pulseSensor;
```

```
// Variáveis
```

```
int pulseValue = 0; // Valor lido do sensor de batimento cardíaco
```

```
int buttonState = 0; // Estado do botão
```

```
void setup() {
```

```
// Inicializa a comunicação serial
Serial.begin(9600);

// Configura os pinos
pinMode(buttonPin, INPUT_PULLUP); // Usa o resistor pull-up interno do Arduino
pinMode(ledPin, OUTPUT); // Configura o pino do LED como saída
pulseSensor.analogInput(pulsePin);
pulseSensor.setThreshold(550); // Ajuste este valor conforme necessário

if (pulseSensor.begin()) {
    Serial.println("PulseSensor iniciado com sucesso.");
} else {
    Serial.println("Falha ao iniciar o PulseSensor.");
}

void loop() {
    // Lê o valor do sensor de batimento cardíaco
    int myBPM = pulseSensor.getBeatsPerMinute();

    // Verifica se há uma batida detectada
    bool pulseDetected = pulseSensor.sawStartOfBeat();

    // Lê o estado do botão
    buttonState = digitalRead(buttonPin);

    // Envia os dados via serial no formato: "pulseValue,buttonState"
    Serial.print(myBPM);
    Serial.print(",");
    Serial.println(buttonState);

    // Pisca o LED quando o botão é pressionado
    if (buttonState == LOW) {
        digitalWrite(ledPin, HIGH); // Liga o LED
        delay(100); // Aguarda 100 ms
```

```

    digitalWrite(ledPin, LOW); // Desliga o LED
}

// Pequena pausa para estabilidade da leitura
delay(100);
}

```

Programação em Python

O código em python recebe os dados do arduino quando há algum input dele, se for uma pressionada no botão o programa irá incrementar uma variável e escreverá essa variável acumuladora como string em um arquivo .txt, para o caso de que seja detectado um batimento pelo sensor o programa irá escrever o número que o sensor tiver detectado em um arquivo .txt.

1.Bibliotecas

```
import serial
```

```
import time
```

2.Código

```
import serial
```

```
import time
```

```
# Configuração da porta serial (ajuste conforme necessário)
arduino_port = 'COM9' # Substitua 'COM7' pela porta correta
baud_rate = 9600
```

```
# Inicializa a comunicação serial
ser = serial.Serial(arduino_port, baud_rate)
time.sleep(2) # Aguarda a estabilização da conexão
```

```
button_press_count = 0
last_button_state = 1 # Assume que o botão começa no estado "solto" (HIGH)
```

```
# Temporizador para controle de atualização
last_update_time = time.time()
update_interval = 5 # Intervalo de atualização em segundos
```

```
# Função para calcular os batimentos por minuto (BPM)
def calculate_bpm(pulse_value):
    # Esse cálculo é um exemplo e pode ser ajustado conforme o sensor específico usado
    return pulse_value

try:
    while True:
        if ser.in_waiting > 0:
            # Lê uma linha de dados da serial
            line = ser.readline().decode('utf-8').strip()

            # Verifica se a linha contém os dados esperados (números)
            if ',' in line:
                try:
                    pulse_value, button_state = map(int, line.split(','))
                except ValueError:
                    # Ignora linhas que não contêm dados válidos
                    continue

            # Verifica se o botão foi pressionado (transição de HIGH para LOW)
            if button_state == 0 and last_button_state == 1:
                button_press_count += 1

            last_button_state = button_state

            # Atualiza os dados a cada 5 segundos
            current_time = time.time()
            if current_time - last_update_time >= update_interval:
                # Calcula os BPM (ajuste conforme necessário)
                bpm = calculate_bpm(pulse_value)

            # Exibe os valores lidos
            print(f"Batimento Cardíaco: {bpm}, Botão Pressionado: {button_press_count} vezes")

            # Salva os valores no arquivo (sobrescrevendo o arquivo)
            with open('pedidos_ajuda.txt', 'w') as file:
                file.write("Contador de Pressionamentos,Batimentos por Minuto (BPM)\n")
                file.write(f"{button_press_count},{bpm}\n")

            # Atualiza o tempo da última atualização
            last_update_time = current_time
```

```

except KeyboardInterrupt:
    print("Interrompido pelo usuário")

finally:
    ser.close()

# Atualiza os dados a cada 5 segundos
current_time = time.time()
if current_time - last_update_time >= update_interval:
    # Calcula os BPM (ajuste conforme necessário)
    bpm = calculate_bpm(pulse_value)

    # Exibe os valores lidos
    print(f'Batimento Cardíaco: {bpm}, Botão Pressionado: {button_press_count} vezes')

    # Salva os valores no arquivo (sobrescrevendo o arquivo)
    with open('pedidos_ajuda.txt', 'w') as file:
        file.write("Contador de Pressionamentos,Batimentos por Minuto (BPM)\n")
        file.write(f'{button_press_count},{bpm}\n')

    # Atualiza o tempo da última atualização
    last_update_time = current_time

except KeyboardInterrupt:
    print("Interrompido pelo usuário")

finally:
    ser.close()

```

Programação Javascript

O código irá ler o arquivo .txt formatando o mesmo e atribuindo os valores obtidos do arquivo em suas respectivas variáveis para ser exposta no site.

Código

```

async function carregarContador() {
    try {
        const resposta = await fetch('pedidos_ajuda.txt');
        const texto = await resposta.text();
        const linhas = texto.trim().split('\n');
        const ultimaLinha = linhas[linhas.length - 1];
        const [batimentosCardiacos, pedidosAjuda] = ultimaLinha.split(',');
        document.getElementById('batimentos').textContent = `${batimentosCardiacos}`;
    }
}

```

```

        document.getElementById('contador').textContent = `${pedidosAjuda}`;
    } catch (erro) {
        console.error('Erro ao carregar o contador:', erro);
        document.getElementById('contador').textContent = 'Erro ao carregar o contador.';
        document.getElementById('batimentos').textContent = 'Erro ao carregar os batimentos
cardíacos.';
    }
}

// Chama a função ao carregar a página e a cada 5 segundos
carregarContador();
setInterval(carregarContador, 5000);

```

Testes e Depuração

1. Teste de Conexão de Hardware

- Verifique todas as conexões e certifique-se de que estão firmes e corretas.

2. Teste de Funcionamento do Sensor e do Botão

- Teste o sensor de frequência cardíaca e o botão para garantir que estão funcionando conforme esperado.

3. Verificação de Comunicação Serial

- Certifique-se de que o Arduino está enviando dados corretamente para o computador e que o código Python está processando esses dados.

4. Teste de Feedback Visual (LED)

- Verifique se o LED acende quando o botão é pressionado, indicando uma situação de crise.

Aplicação prática e exemplos

A PAS pode ser usada em ambientes escolares para monitorar crianças com TEA. Em caso de emergência, o botão de crise pode ser pressionado para enviar um sinal de alerta.

Conclusão

A Pulseira de Assistência Sensorial (PAS) é uma solução inovadora para ajudar crianças com Transtorno do Espectro Autista (TEA). Utilizando tecnologias acessíveis como Arduino, sensores de frequência cardíaca e programação em Python e JavaScript, a PAS monitora sinais de estresse e crises, permitindo intervenções rápidas.

Este projeto demonstra a importância de ferramentas assistivas que são econômicas e adaptáveis às necessidades individuais. A PAS não só melhora a qualidade de vida das crianças com TEA e seus cuidadores, mas também exemplifica como a tecnologia pode ser usada de forma prática e significativa.

O tutorial detalhado incentiva educadores, cuidadores e entusiastas a adotar e adaptar o projeto, contribuindo para a segurança e bem-estar das crianças com TEA e proporcionando tranquilidade aos pais. A PAS destaca o poder da tecnologia em contextos educativos e assistenciais, promovendo uma sociedade mais inclusiva e acessível.

