UNIUBE Educação e Responsabilidade Social **CAMPUS UBERLÂNDIA**	**CURSOS DE GRADUAÇÃO EM ENGENHARIA** **LISTA DE EXERCÍCIOS** **COMPONENTE: Sistemas Embarcados e ioT** **PROFESSOR: Júlio Almeida Borges**

Aluno (a): _____ RA: ._____.

Valor: 2,5 pts Data: 22/10/2024 Nota: _____

Lista de Exercícios 1

Projetos de Monitoramento e Controle com sistema embarcado

Objetivo Geral:

Aplicar conceitos de sistemas embarcados utilizando o Arduino para criar soluções de monitoramento e controle, explorando sensores, atuadores, e o uso de portas analógicas, digitais e PWM.

Material Necessário:

- 1 Placa Arduino (Uno, Nano, etc.)
- Protoboard
- Cabos Jumper
- 1 Sensor de Temperatura (LM35 ou DHT11)
- 1 Ventoinha (5V ou 12V) com transistor (TIP120 ou MOSFET) e diodo de proteção
- 1 Potenciômetro (para o Projeto 2)
- 1 LDR (Sensor de Luz)
- 1 LED
- 1 Resistor de 220Ω (para o LED)
- Fonte de alimentação de 5V (ou USB)
- Multímetro (opcional para medições)

Controle de Porta Analógica (Entrada)

As portas analógicas no Arduino são usadas para **ler** valores de sensores ou dispositivos que fornecem sinais analógicos (valores contínuos), como potenciômetros, sensores de temperatura, etc. A função usada para esse tipo de leitura é `analogRead()`.

Explicação:

- analogRead(sensorPin) lê um valor analógico da porta definida, retornando um valor entre **0 e 1023** (em função da tensão de entrada no pino, variando de 0 a 5V).
- Os pinos analógicos no Arduino são designados como **A0, A1, A2**, etc.

Controle de Porta Digital (Entrada e Saída)

As portas digitais podem ser usadas tanto para **ler** quanto para **escrever** valores digitais (0 ou 1, LOW ou HIGH). Para leitura, utilizamos a função `digitalRead()`, e para saída, `digitalWrite()`.

Explicação:

- **Saída Digital**: `digitalWrite(ledPin, HIGH)` ou `digitalWrite(ledPin, LOW)` envia um valor de **5V** ou **0V** para a porta digital, ligando ou desligando um componente, como um LED.
- **Entrada Digital**: `digitalRead(buttonPin)` lê o estado de uma porta digital, retornando **HIGH** (5V) ou **LOW** (0V), usado para monitorar o estado de botões, sensores de limite, etc.
- As portas digitais são numeradas como **2, 3, 4**, etc.

Controle de Porta Digital PWM

As portas digitais PWM permitem controlar dispositivos como motores ou LEDs com variação de intensidade, simulando uma saída analógica. A função utilizada é `analogWrite()`.

Explicação:

- `analogWrite(pwmPin, brightness)` aplica um sinal PWM (Modulação por Largura de Pulso) à porta especificada. O valor pode variar de **0 a 255**, onde **0** representa 0% do tempo no estado **HIGH** (apagado) e **255** representa 100% no estado **HIGH** (máxima intensidade).
- Portas PWM específicas no Arduino: **3, 5, 6, 9, 10, 11**.

Exercício 01: Monitoramento de Temperatura e Controle da Ventoinha

Objetivo: Monitorar a temperatura ambiente utilizando um sensor de temperatura e controlar a velocidade de uma ventoinha de acordo com a variação de temperatura, usando PWM.

Passos:

1. **Montagem do Circuito:**
    - Conecte o sensor de temperatura LM35 à porta analógica A0 do Arduino.
    - Conecte a ventoinha em uma porta digital PWM (D9), controlando-a por meio de um transistor TIP120.
    - Use um diodo de proteção no transistor para a ventoinha.
2. **Código:**
    - Leia a temperatura do sensor.
    - Converta a temperatura lida em uma saída PWM para a ventoinha.
    - Quando a temperatura for baixa (abaixo de 25ºC), a ventoinha gira devagar, e conforme a temperatura sobe, a velocidade da ventoinha aumenta proporcionalmente.

Exercício 2: Controle Automático de Iluminação com LDR

Objetivo: Criar um sistema que detecta a luminosidade ambiente usando um LDR e ajusta automaticamente o brilho de um LED, utilizando uma porta analógica para ler a luminosidade e PWM para controlar o LED.

Passos:

1. **Montagem do Circuito:**
    - Conecte o LDR à porta analógica A2 (ligue o LDR em série com um resistor de 10kΩ).
    - Conecte o LED à porta PWM (D10) com um resistor de 220Ω.
2. **Código:**
    - Leia o valor da luminosidade ambiente e ajuste o brilho do LED automaticamente.

Exercício 3: Controle de Velocidade de Motor DC com Sensor de Pressão

Objetivo: Simular o controle de velocidade de um motor DC utilizando um sensor de pressão (ou um potenciômetro), aplicando o conceito de controle proporcional em um ambiente industrial.

Materiais Necessários:

- 1 Motor DC (5V ou 12V) com transistor TIP120 ou MOSFET
- 1 Sensor de Pressão (ou potenciômetro, caso o sensor não esteja disponível)
- 1 Diodo de proteção (1N4007)
- 1 Fonte de alimentação (5V ou 12V)
- Protoboard e cabos jumper

Descrição do Projeto:

Neste projeto, a **velocidade do motor DC** será controlada de acordo com a pressão detectada por um sensor, simulando uma aplicação industrial onde o motor aumenta ou diminui sua velocidade com base na pressão do sistema. Em caso de indisponibilidade de um sensor de pressão, um **potenciômetro** pode ser usado para simular esse controle.

Montagem do Circuito:

1. **Sensor de Pressão (ou Potenciômetro):**
    - Conecte o pino de leitura do sensor de pressão à porta analógica A1 do Arduino. Se estiver usando um potenciômetro, conecte o pino central do potenciômetro à porta A1 e as extremidades no GND e 5V.
2. **Motor DC:**
    - Conecte o motor à porta D9 (porta PWM) do Arduino, através de um **transistor TIP120** ou um **MOSFET** para controlar a alimentação do motor. O diodo de proteção (1N4007) deve ser conectado em paralelo ao motor, para evitar danos ao transistor devido ao efeito de retorno de corrente.

Código:

O código abaixo lê o valor do sensor de pressão (ou potenciômetro) e converte-o em um valor de PWM para controlar a velocidade do motor DC.