CENTRO UNIVERSITÁRIO DE BRÁSILIA

Aluno: Lucas Freitas Lemos

Professor: Aderbal Botelho

Disciplina: Sistema em Tempo Real e Embarcados

LABORATÓRIO - SENSORES E ATUADORES

1. INTRODUÇÃO

Os sensores e atuadores estão cada vez mais presentes e integrados aos sistemas operacionais de tempo real e embarcados. Em geral, os sistemas de tempo real e embarcados costumam utilizar-se de tais dispositivos para controlar todo o fluxo de execução de tarefas, seja periódica ou não, como para aumentar a eficiência de seu sistema. Os sensores são dispositivos conversores sensíveis à algum tipo de energia ambiente que pode ser luminosa, térmica, cinética, relacionando sempre informações sobre grandezas físicas que precisa ser mensurada(medida) em sinais elétricos correspondentes. Já os atuadores são componentes capazes de realizar a conversão da energia elétrica, hidráulica, pneumática em energia mecânica.

2. RESUMO

Este trabalho tem como objetivo a construção de um circuito simples com o Kit Arduíno, com a utilização de apenas um led e um buzzer que por sua vez, serão controlados de acordo com a intensidade da luz que incide sobre o sensor de luminosidade LDR(Light Dependent Resistor). A ideia é que quanto mais incidência de luz o LDR receber, menor será a resistência, portanto mais corrente elétrica irá fluir e mais alto será o valor lido. E quanto menos luz o LDR receber, maior será a resistência, então menos corrente elétrica irá fluir e menor será o valor lido fazendo que a campainha dispare e o led acenda.

3. OBJETIVOS

Objetivos

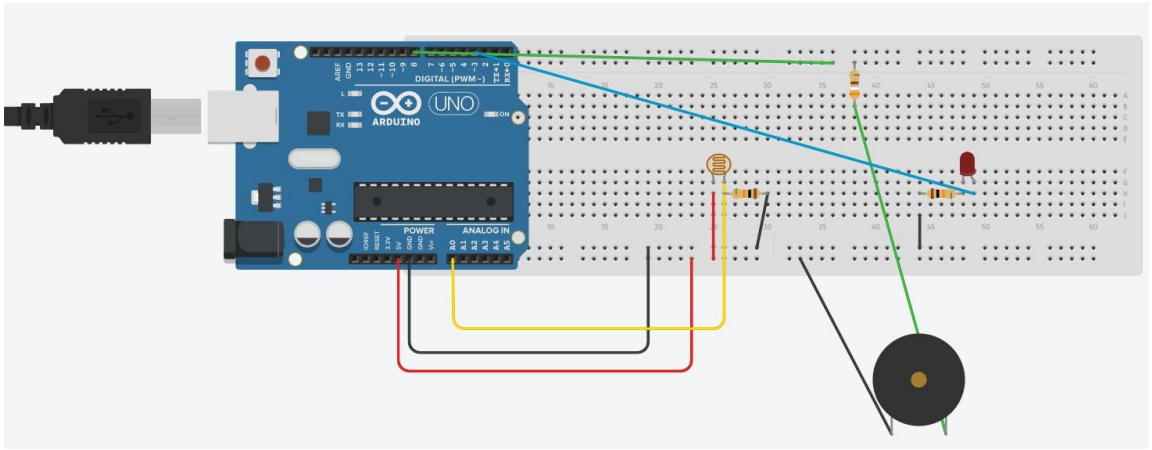
1. Definir um valor limite de disparo adequado para a luminosidade ambiente
2. Acender a luz do led quando passar valor limite de disparo
3. Acionar campanhia do buzzer quando passar valor limite de disparo
4. Mostrar valor da luminosidade medido pelo sensor

4. METODOLOGIA

Para construção desse circuito, iremos utilizar o detector de luminosidade(fotoresistor) presente no Kit Arduíno para disparar a campainha(Buzzer) e o led presente no Kit. Nesse projeto, ao todo, serão utilizados os seguintes componentes:

- Fotocélula LDR 5mm: Sensor de luminosidade.
- Buzzer 5V: Campainha que dispara um som quando ativada.
- Corrente: ≤ 42mA;
- Som de saída: ≥ 85DB;
- Frequência de Resonância: 2300 ± 300HZ;
- Temperatura de Operação: -20ºC ~ +45ºC;
- Temperatura de armazenamento: -20ºC ~ +60ºC;
- Sinalizador piezoelétrico de 12mm.
- Resistor Filme de Carbono 10kΩ
- Resistor Filme de Carbono 10kΩ
- Resistor Filme de Carbono 390Ω
- LED

Para resolução do exercício, utilizei o Tinkercad para montagem do circuito (conforme Figura 1) com o Arduino UNO e seus respectivos sensores e atuadores citados no problema, como também utilizei a linguagem C++ para elaborar um código capaz de integrar o circuito e realizar todas as ações devidas para resolução. Esse código está hospedado do GIT com os devidos comentários explicando cada parte dele, além de um roteiro dividido por sessões.



5. Explicando o código

```
//Atribui um "valor limite de disparo" para achar um valor adequado,
//ligando e desligando o LED quando passar a mão sobre o fotoresistor (LDR).
int triggerLimit = 930;

// Ligue o LED ao pino digital 3
const int ledPin = 3;

// Ligue o Piezo no pino 8
const int buzzerPin = 8;

// Altera a frequência do tom
const int freq=5;

// O fotoresistor (LDR) é conectado ao pino analógico A0
int ldrPin = A0;

// Armazena e inicializa o valor de leitura analógica
int ldrValue = 0;
```

1. Começamos declarando as constantes e as variáveis do projeto.

- 1.1. Utilizamos as constantes e variáveis tipo "int" que podem contar números inteiros de -32767 até 32767.
- 1.2. A constante do tipo inteiro nomeada buzzerPin refere-se ao buzzer que deverá estar conectado à porta digital 8 do microcontrolador Arduino.
- 1.3. A constante do tipo inteiro nomeada ldrPin refere-se ao Sensor de luminosidade LDR que deverá estar conectado à porta analógica A0.
- 1.4. A constante do tipo inteiro nomeada ledPin refere-se ao Led vermelho que deverá estar conectado à porta 3 do microcontrolador Arduino.
- 1.5. Declaramos e inicializamos a variável ldrValue como 0 (zero).
- 1.6. Declaramos e inicializamos a constante freq como 5 (esse valor deve ser diferente de zero e poderá ser alterado para aumentar ou diminuir a frequência de bips do alarme).

```
void setup() {
   // Inicia a comunicação serial com uma taxa de transmissão de 9600 boud rate
  Serial.begin(9600);

  // Define o fotoresistor como uma entrada
  pinMode(ldrPin, INPUT);

  // Define o LED como uma saída
  pinMode(ledPin, OUTPUT);

  // Define o Piezo como uma saída
  pinMode(buzzerPin, OUTPUT);
}
```

2. Através da estrutura void setup(), foi definido:

- 2.1. A função Serial.begin() serve para dizer ao Arduino que será coletado dados para o computador a partir da porta serial e o cabo USB. O número entre os parênteses significa qual é a taxa de dados que a placa vai se comunicar com o computador. Utilizaremos a taxa padrão de 9600bps (bits-per-second).

- 2.2. Observe que portas analógicas não precisam ser definidas, pois por padrão, já são definidas como INPUT. Entretanto, você pode utilizar a linha de código: pinMode(ldrPin, INPUT);
- 2.3. Define-se a constante ledPin como saída do controlador Arduino (OUTPUT) conectada à porta digital 3.
- 2.4. Define-se a constante buzzerPin como saída do controlador Arduino (OUTPUT) conectada à porta digital 7.

```
void loop(){

  // Lê o valor atual do fotoresistor
  ldrValue = analogRead(ldrPin);

  // Se o valor da luminosidade estiver abaixo do valor "limite de disparo",
  //então o LED liga, caso contrário o LED permanece desligado.
  if (ldrValue < triggerLimit) {
      digitalWrite(ledPin, HIGH);

    //Toca o alarme
    tone(buzzerPin,300); // toca um tom de 300 Hz do piezo
    delay(30); // espera alguns segundos
    noTone(buzzerPin); // interrpompe o tom do buzzer
    digitalWrite(ledPin,LOW); // apaga a led
    delay(ldrValue/freq); /* espera agora a quantidade de milisegundos
    em idrValue */
  }
  else {
      digitalWrite(ledPin,LOW); // apaga a led
      noTone(buzzerPin); // interrompe o tom do buzzer
  }

  // Imprime as leituras do sensor no monitor serial da IDE do Arduino
  Serial.print ("Leitura atual do sensor: ");
  Serial.println(ldrValue);
  delay(130);
}
```

3. Através da estrutura void loop(), obtemos:

- 3.1. A variável ldrValue receberá os valores lidos e atualizados diretamente pelo pino analógico onde está conectado o sensor LDR, através da função analogRead() que faz a conversão de analógico para digital. Esta leitura é feita pelo ADC (Analog to Digital Converter - conversor analógico para digital) sem tratamento nenhum. A variável foi definida localmente como tipo inteiro (int), e portanto, vai de 0 a 1023, ou seja, possui 210 = 1024 valores inteiros (referente à resolução de 10 bits do ADC para controladores Arduino UNO, Mega e Leonardo). Assim, quando o sensor LDR não estiver recebendo pouca ou nenhuma luz do ambiente, o valor lido será próximo de zero, e quando sensor receber muita luz, o valor será próximo de 1023, fazendo assim a leitura da luminosidade de um ambiente. Observação: Nesse circuito, o sensor LDR varia de 0V a 5V(leitura analógica), ou seja, de 0 a 1023 quando convertido em leitura digital através do ADC do controlador Arduino. Quanto mais luz o LDT receber, menor será a resistência, portanto mais corrente elétrica irá fluir e mais alto será o valor lido. E quanto menos luz o LDR receber, maior será a resistência, então menos corrente elétrica irá fluir e menor será o valor lido.

- 3.2. Utilizamos a estrutura condicional If (ldrValue < triggerLimit). Portanto, se a variável ldrValue for menor que 930 (número que utilizamos como referência quando o sensor estiver recebendo pouca luz) o led vermelho acenderá e o alarme será acionado, conforme explicação a seguir:
    - 3.2.1. A função digitalWrite(ledPin, HIGHT) faz com que acenda o led vermelho.
    - 3.2.2. Dispara o alarme, conforme explicação a seguir:
    - 3.2.3.1. A função tone() define um tom para o buzzer. Vamos utilizar tone(buzzerPin,1000) que gera um tom com frequência de 300Hz. (Você pode alterar este valor definindo tons diferentes para o buzzer).
    - 3.2.3.2. Através da função digitalWrite(ledPin, HIGH) acendemos o Led vermelho.
    - 3.2.3.3. Através da função delay(30) esperamos 30ms. Este valor define o tempo que o led e o buzzer ficaram ativos, gerando um tipo de bip.
    - 3.2.3.4. Interrompemos o som do buzzer através da função noTone().
    - 3.2.3.5. Através da função digitalWrite(ledPin, LOW) apagamos o Led vermelho.
    - 3.2.3. A função delay(ledrValue/freq) define o intervalo entre os bips. Quanto maior a incidência de luz sobre o sensor, menor será o valor de ledrValue e, portanto, menor o intervalo entre os bips, fazendo com que eles sejam emitidos de forma mais rápida. Quanto menor for a incidência de luz sobre o sensor, maior será o valor lido e, portanto, maior o intervalo entre os bips, portanto, emitidos de forma mais lenta.
    - 3.2.4. Já a constante freq faz com que o intervalo entre os bips sejam maiores ou menores, dependo do seu valor. Quanto maior for o seu valor, menor será o tempo de espera. Portanto os bips serão emitidos de forma mais rápida.
    - 3.3. Escrevemos na tela do Monitor Serial o valor da variável ldrValue através do comando Serial.printn(). O comando println() diz ao monitor que se deve pular uma linha após escrever o valor definido entre parêntesis.
    - 3.4. Através da função delay(130), definimos um tempo de espera de 130 ms entre cada ciclo do loop.