UFAC FUNDAPE CITS

Introdução a IoT
Internet das Coisas

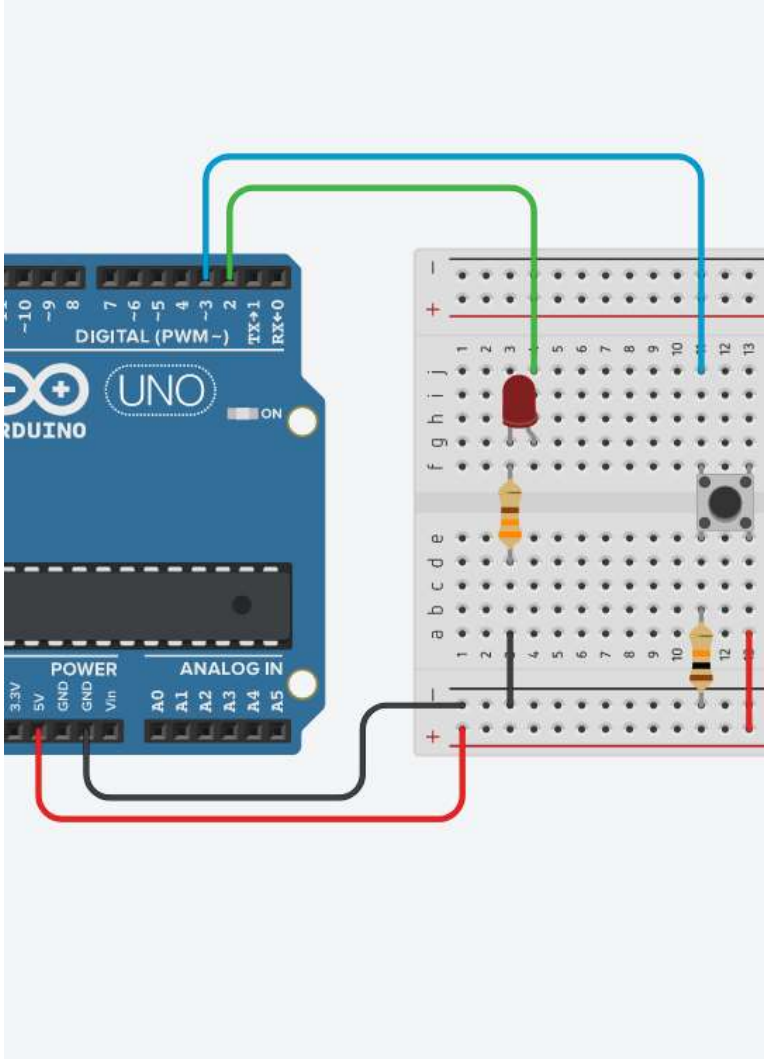
Prof. André Nasserala
andre.nasserala@ufac.br

Push-Button

- Os push-buttons (chaves botão) e leds são elementos presentes em praticamente qualquer circuito eletrônico.
- As chaves são usadas para enviar comandos para o Arduino e os Leds são elementos de sinalização luminosa.
- Esses dois componentes são trabalhados por meio das entradas e saídas digitais do Arduino.

Push-Button

- Neste exemplo vamos fazer uma aplicação básica que você provavelmente vai repetir muitas vezes.
- Vamos ler o estado de um push-button e usá-la para acender ou apagar um led.
- Ou seja, sempre que o botão for acionado, vamos apagar ou acender o Led.



Push-Button

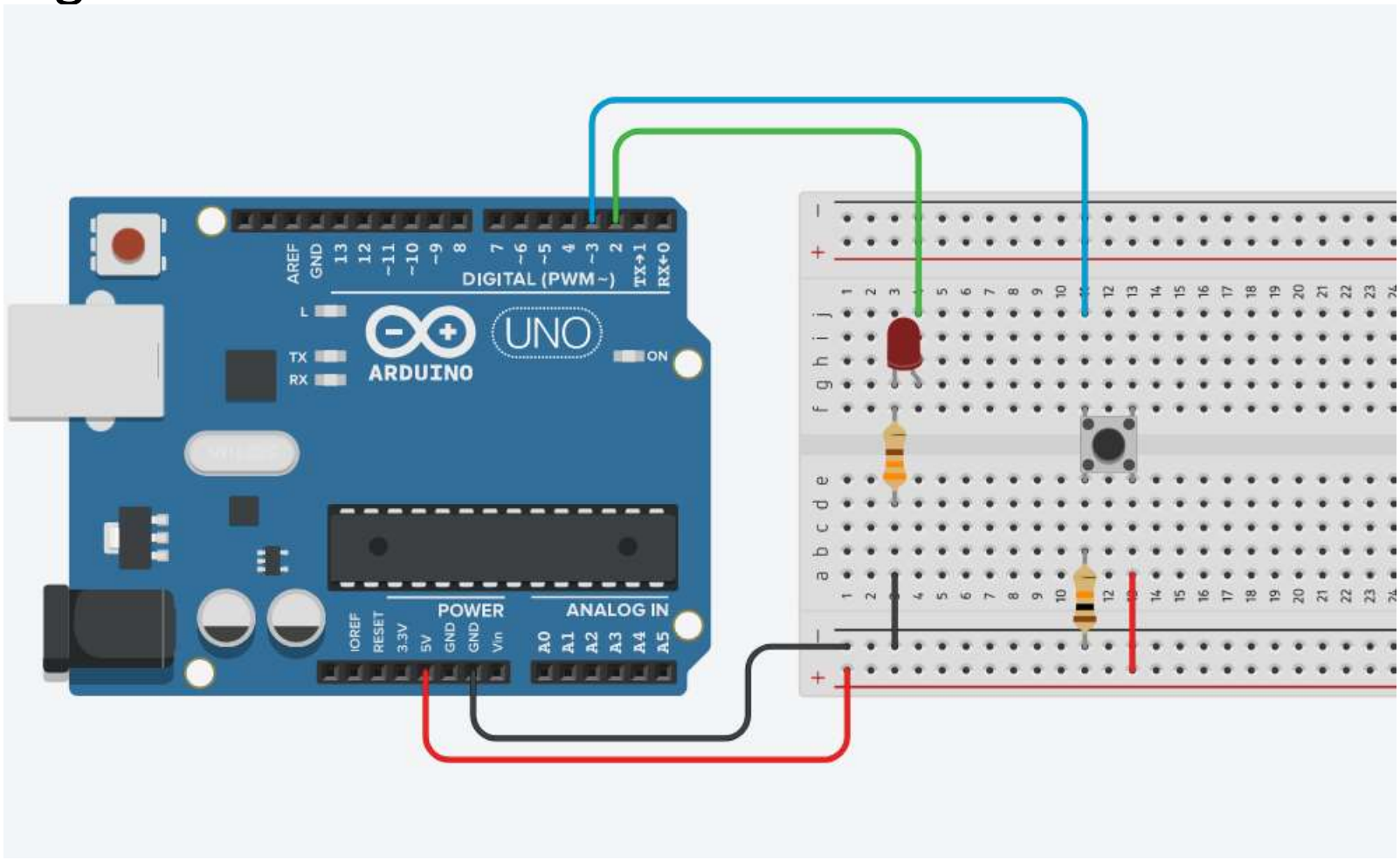
- Lista de materiais:

- Para esse exemplo você vai precisar:

- 2 resistores de 330 ohms;
- 1 Led vermelho (ou de outra cor de sua preferência);
- Push-button (chave botão);
- 1 Arduino UNO;
- Protoboard;
- Jumpers de ligação;

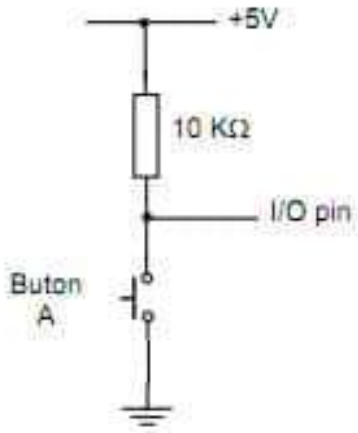
Push-Button

- Diagrama de circuito:

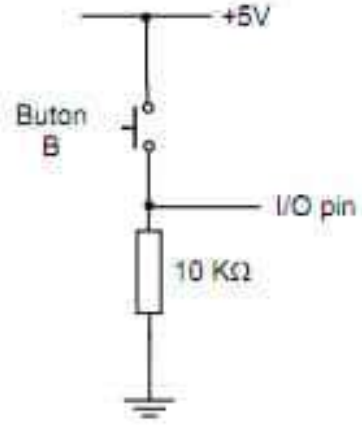


Push-Button

- Os resistores pull-up e pull-down garantem um nível lógico estável quando por exemplo uma tecla não está pressionada.
- Geralmente utiliza-se um resistor de 10KΩ para esse propósito.
- A seguir é exibida a ligação desses resistores no circuito para leitura de tecla do botão:



Pull-Up



Pull-Down

Push-Button

- Esse pequeno código abaixo mostra como ler entradas digitais e como acionar as saídas digitais.
- Na função void Setup(), é preciso configurar qual pino será usado como saída e qual será usado como entrada.
- Depois de configurar os pinos, para acioná-los basta chamar a função digitalWrite(pino,HIGH).
- A função digitalWrite() aciona ou desaciona um pino digital dependendo do valor passado no argumento:

- Se for “HIGH”, o pino é acionado.
- Se for “LOW”, o pino é desligado.

Push-Button

- Na função void Loop(), fizemos um if no qual a função digitalRead é usada para saber se o pushButton está acionado ou não.

- Caso ele esteja acionado, nós acendemos o Led, caso ele esteja desligado, nós desligamos o led.

- Carregue o código abaixo e pressione o botão para acender o LED.

Push-Button

```
#define BOTAO 8 // define pino digital D8
#define LED 7 // define pino digital D7
void setup()
{
  pinMode(BOTAO, INPUT); // configura D8 como entrada digital
  pinMode(LED, OUTPUT); // configura D7 como saída digital
}
void loop()
{
  if ( digitalRead(BOTAO) == HIGH)  { // se chave = nivel alto
    digitalWrite(LED, HIGH); // liga LED com 5V
  }
  else { // senão chave = nivel baixo
   digitalWrite(LED, LOW); // desliga LED com 0V
  }
  delay(100); // atraso de 100 milissegundos
}
```

Melhorando o Código

- Criar um circuito para atribuir duas funções em um único botão (push button).
- Neste circuito o botão servirá como um interruptor para ligar e desligar um componente eletrônico.
- No nosso exemplo vamos ligar e desligar um led utilizando o push button como interruptor.
- **Observação**: Os push-button apenas mudam seu estado enquanto estamos pressionando, voltando ao seu estado original quando o botão é liberado.
- Teremos uma rotina para atribuir a um só botão duas funções de ligar e desligar um componente eletrônico qualquer.

Melhorando o Código

- Criar um circuito para atribuir duas funções em um único botão (push button).

- Neste circuito o botão servirá como um interruptor para ligar e desligar um componente eletrônico.

- No nosso exemplo vamos ligar e desligar um led utilizando o push button como interruptor.

Melhorando o Código

- **<u>Como o projeto deve funcionar:</u>**

- Quando você rodar o programa, o led ficará apagado.

- Ao pressionar e soltar o botão o led se acenderá.

- Ao pressionar e soltar novamente o botão o led se apagará, invertendo-se assim o seu estado.

Melhorando o Código

```
#define BOTAO 8 // define pino digital D8
#define LED 7 // define pino digital D7
int ESTADO = 0; // variável para leitura do pushbutton
int G_ESTADO = LOW; // variável para armazenar valores do pushbutton
void setup()
{
  pinMode(BOTAO, INPUT); // configura D8 como entrada digital
  pinMode(LED, OUTPUT); // configura D7 como saída digital
  }
void loop()
{
  ESTADO = digitalRead(BOTAO);  // le o estado pushbutton: ligado (HIGH) ou desligado (LOW)
  if (ESTADO == HIGH) {// verifica se o botão (pushbutton) está pressionado
    G_ESTADO = !G_ESTADO;  // inverte valor da variável variable_buttonEstado
    delay(500); //esperera o tempo de 500ms para evitar que haja várias vezes alterações
  }
  if (G_ESTADO == HIGH) {
    digitalWrite(LED, HIGH); // liga o led
  }
  else {
      digitalWrite(LED, LOW); // desliga o led
  }
}
```

Sensor de luz LDR

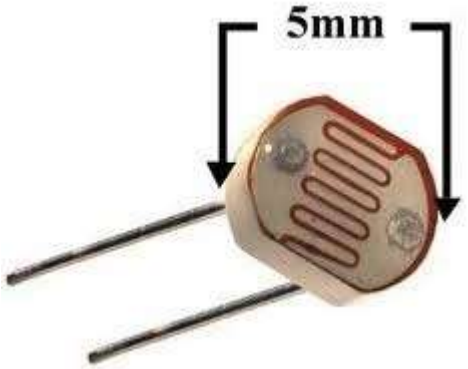
- O sensor LDR é um sensor de luminosidade.
- LDR é um Light Dependent Resistor, ou seja, um resistor cuja resistência varia com a quantidade de luz que incide sobre ele.
- Esse é seu princípio de funcionamento.

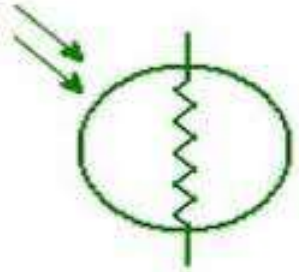
Sensor de luz LDR

- É importante considerar a potência máxima do sensor, que é de 100 mW.
- Ou seja, com uma tensão de operação de 5V, a corrente máxima que pode passar por ele é 20 mA.
- Felizmente, com 8K ohms (medido experimentalmente com o ambiente bem iluminado), que é a resistência mínima, a corrente ainda está longe disso, sendo 0,625mA.
    - Nas suas medições, pode ser que você encontre um valor de resistência mínimo diferente, pois depende da iluminação local.
- Dessa forma, podemos interfacear o sensor diretamente com o Arduino.

Especificações do LDR

- Modelo: GL5528
- Diâmetro: 5mm
- Tensão máxima: 150VDC
- Potência máxima: 100mW
- Temperatura de operação: -30°C a 70°C
- Comprimento com terminais: 32mm
- Resistência no escuro: 1 MΩ (Lux 0)
- Resistência na luz: 10-20 KΩ (Lux 10)





- Este sensor de luminosidade pode ser utilizado em projetos com arduino e outros microcontroladores para alarmes, automação residencial, sensores de presença e vários outros.

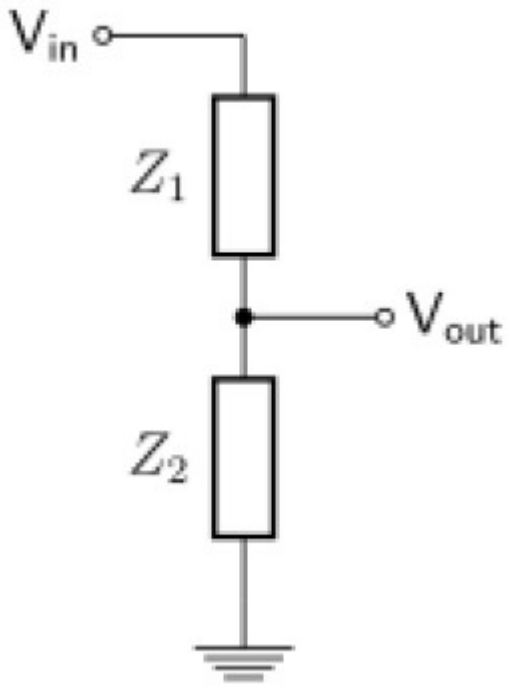
Divisor de Tensão

- O divisor de tensão consiste em dois resistores ligados em série (Z1 e Z2) , em que o sinal de 5V é aplicado a o terminal de um deles.
- O terminal do segundo resistor é ligado ao GND, e o ponto de conexão entre os dois resistores é a saída do divisor, cuja tensão é dada pela seguinte relação:

$$V_{\text{out}} = \frac{Z_2}{Z_1 + Z_2} \cdot V_{\text{in}}$$

Divisor de Tensão

- Em que Z1 e Z2 são os valores dos resistores da figura abaixo.

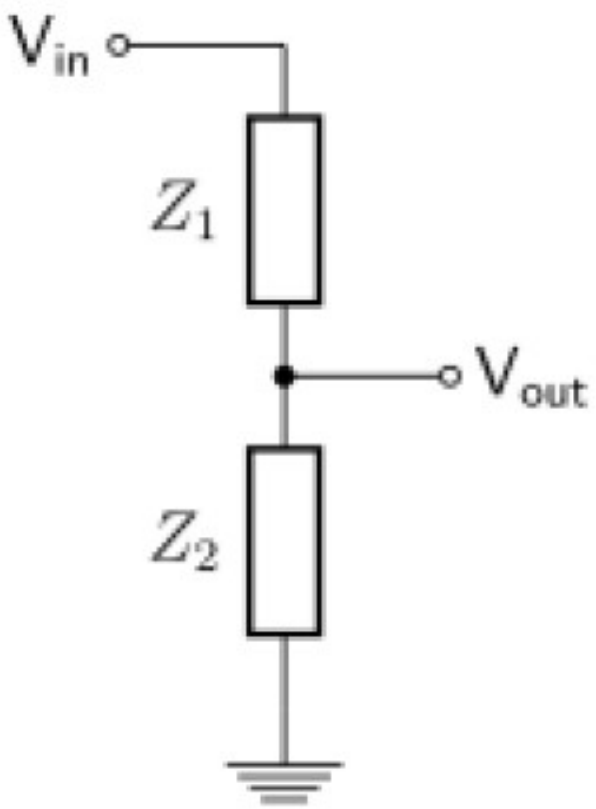


Divisor de Tensão

- Um divisor de tensão muito comum é fazer Z1 igual 330 ohms e Z2 igual 680 ohms.

- Vout = (680 /(330+680)*5
- Vout = 0,673 * 5
- Vout = 3,365 v

- Dessa forma a sáida Vout fica sendo 3.365 V.



Exemplo

- No exemplo a seguir, vamos usar uma entrada analógica do Arduino para ler a variação de tensão no LDR e, consequentemente, saber como a luminosidade ambiente está se comportando.
- Veja na especificação que com muita luz, a resistência fica em torno de 10-20 KΩ, enquanto que no escuro pode chegar a 1MΩ.
- Para podermos ler as variações de tensão resultantes da variação da resistência do LDR, vamos usar o sensor como parte de um divisor de tensão.

Exemplo

- Assim, a saída do divisor será dependente apenas da resistência do sensor, pois a tensão de entrada e a outra resistência são valores conhecidos.

- No nosso caso, vamos usar um resistor de 10K e uma tensão de operação de 5V.

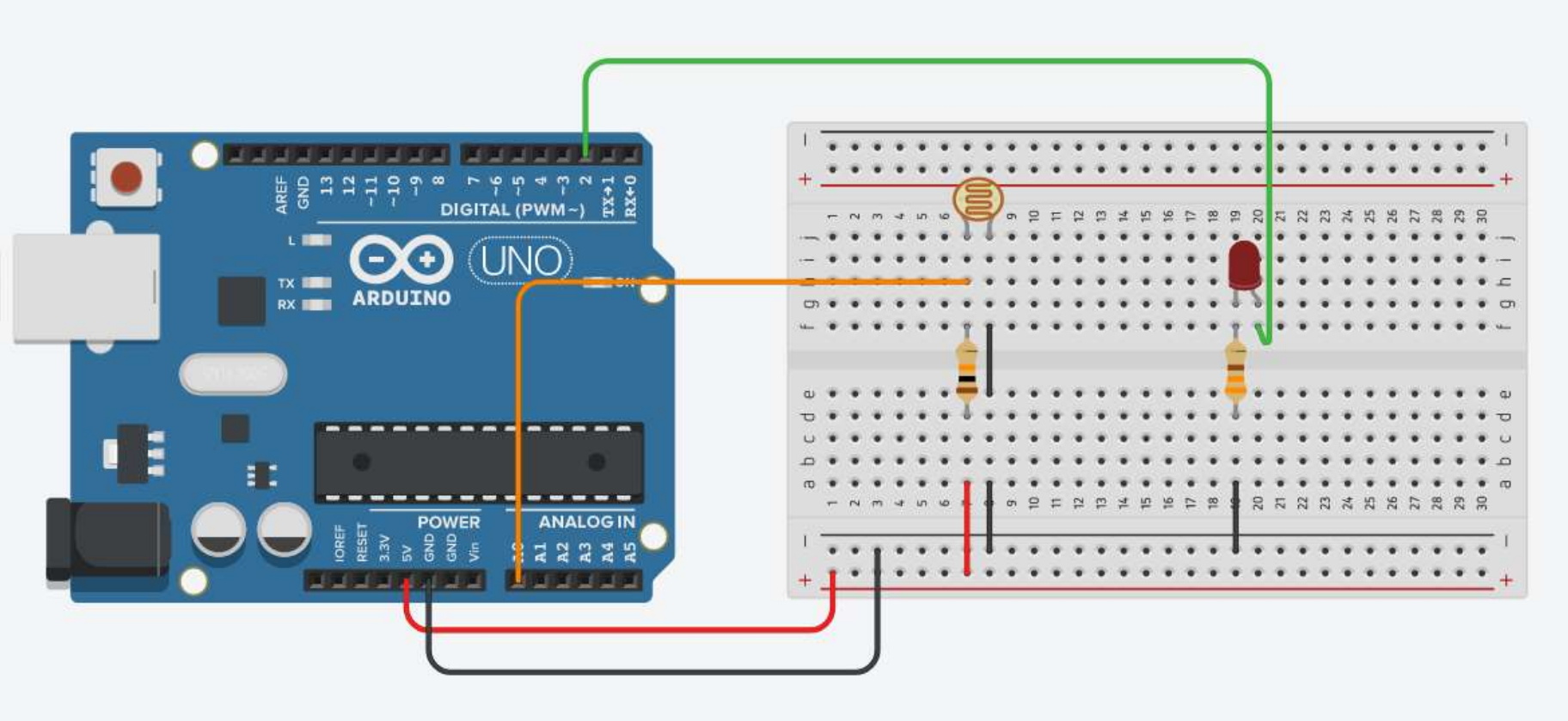
- Assim, o sinal que vamos ler no arduino terá uma variação de 2,2V (quando o LDR for 8K) e 5V (quando o LDR tiver resistências muito maiores que o resistor de 10K).

Exemplo

- Lista de materiais:

- Para esse exemplo você vai precisar:
- LDR;
- Resistor de 10k;
- 1 Arduino UNO;
- Protoboard;
- Jumpers de ligação;

Exemplo



PROJETO IOT

Exemplo

- No diagrama, o sensor é ligado como parte de um divisor de tensão no pino analógico A0, de forma que a tensão de saída do divisor varia de acordo com a variação da resistência do sensor.

- Assim, vamos identificar as variações na intensidade de luz pelas variações na tensão do sensor.

- Quanto maior a intensidade de luz, menor a resistência do sensor e, consequentemente, menor a tensão de saída.

Código

```
#define LDR A0
#define LED 2
int LEITURA = 0;
void setup()
{
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
}
void loop()
{
   LEITURA = analogRead(LDR);
   Serial.print("Leitura do LDR: ");
   Serial.println(LEITURA);
   delay(500);
   if ( LEITURA >= 100) {
     digitalWrite(LED, HIGH);
   }
   else {
      digitalWrite(LED, LOW);
   }
}
```

Referências

- MONK, Simon. Programação com Arduino. Porto Alegre – RS. Editora: Bookman – 2017. ISBN: 9788582604465
- VIDAL, Vitor, Gustavo Murta. Arduino Start. Eletrogate – 2018. Belo Horizonte – MG. Disponível em: https://conteudo.eletrogate.com/apostila-arduino-start.
- MALVINO, Albert Paul. Eletrônica: Volume 1. 4.ed. São Paulo – SP: Makron Books, 1997. ISBN: 8534603782.
- SENAI, Senai SP. FUNDAMENTOS DE ELETRÔNICA - 1ªED. Editora: Senai SP – São Paulo 2015. ISBN: 9788583932086
- WILSON, J. A. e Milton Kaufman. Eletrônica Básica - Teoria e Prática - Volume 2. São Paulo: Editora: Rideel, 1980.
- PEREZ, Anderson Luiz Fernandes, Heron Pereira, Cristiano Pereira de Abreu, Renan Rocha Darós. Oficina de Robótica. UFSC – Programação Básica em Arduino - 2015. Disponível em: http://oficinaderobotica.ufsc.br/programacao-basica-em-arduino/.