UNIVERSIDADE FEDERAL DO CEARÁ

Centro de Tecnologia

Departamento de Engenharia Mecânica

INSTRUMENTAÇÃO COM TINKERCAD - CAPÍTULO 6: INSTRUMENTAÇÃO APLICADA AO CONTROLE

Thiago Victor Albuquerque de Freitas - 494080

2021

Projeto Controle Remoto

Descrição:

O presente projeto irá simular o controle de 2 LED utilizando um controle remoto de raios infravermelhos. O circuito irá utilizar o sensor e o controle IR presente no Tinkercad bem como a biblioteca IRremote.h que pode ser adicionada facilmente na plataforma. Por fim, cabe ressaltar que o projeto foi baseado no Guia completo do Controle Remoto IR + Receptor IR para Arduino do autor José Gustavo Abreu Murta.

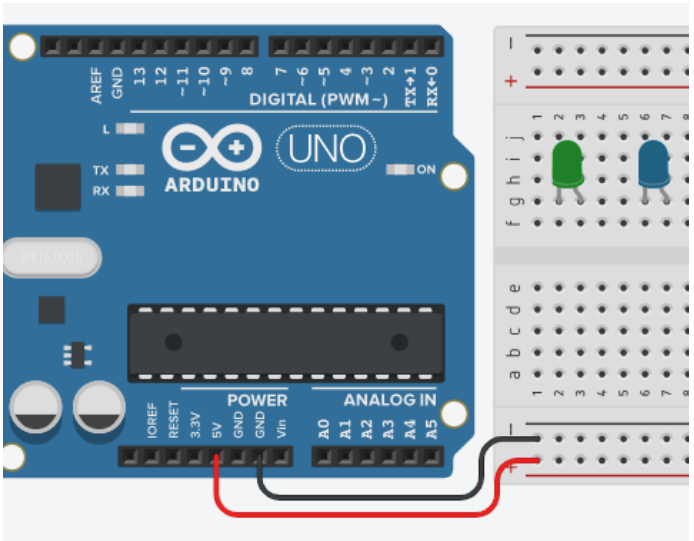
Circuito eletrônico:

- O circuito é composto:
  - 1 Arduino;
  - 1 Protoboard;
  - 1 IR sensor;
  - 1 IR controle;
  - 2 LED;
  - 2 Resistores de 220 Ω;
- Montagem do circuito:

Etapa 1: Adicione o Arduino, a protoboard, 2 LED e 2 resistores de 220 Ω ao circuito buscando “Arduino”, "Breadboard”, "LED” e “Resistor”, respectivamente, nos componentes.

Etapa 2: Conecte o 5V do arduino no “+” e o GND no “-” da protoboard, como mostrado na Fig. 1.

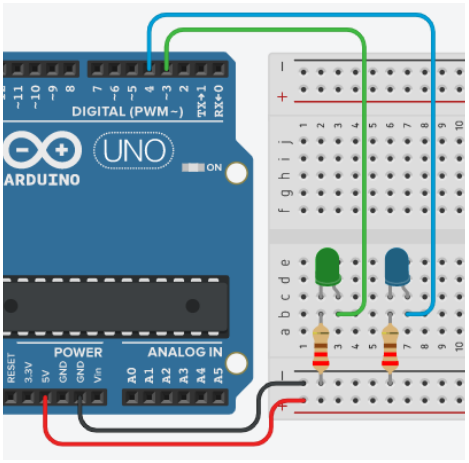
Figura 1 - Alimentação da protoboard.



Fonte: Autor.

Etapa 3: Conecte o pino positivo de um dos LED na porta 2 do Arduino e o outro pino no polo negativo através do resistor de 220 Ω. Repita o mesmo procedimento para o outro LED, porém conecte este na porta 4 do Arduino, como mostrado na Fig. 2.

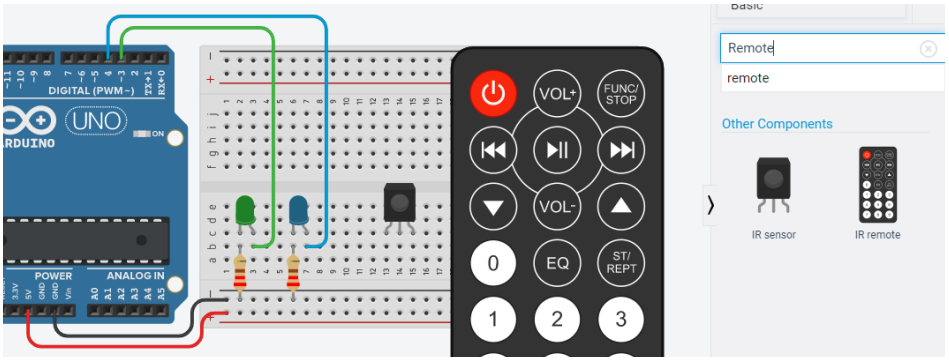
Figura 2 - Conexões dos LED ao Arduino.



Fonte: Autor.

Etapa 4: Adicione o sensor e o controle remoto buscando “IR sensor” e "IR remote”, respectivamente, nos componentes, como mostrado na Fig. 3.

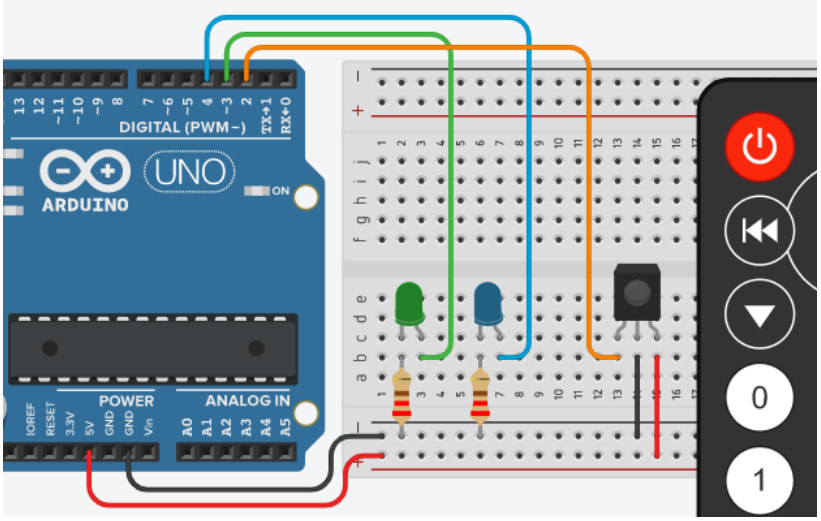
Figura 3 - Adição do controle remoto e do sensor infravermelho ao sistema.



Fonte: Autor.

Etapa 5: Alimente o sensor conectando o polo positivo da protoboard no pino mais a direita e o polo negativo no pino central do sensor. Além disso, conecte o pino mais à esquerda, que é o pino de saída de dados do sensor, na porta 2 do Arduino, como mostrado na Fig. 4.

Figura 4 - Conexões do sensor infravermelho.



Fonte: Autor.

Programação:

A função do projeto é ligar e desligar os LED pressionando os botões do controle, porém, cada botão manda um sinal infravermelho diferente que deverá ser identificado pelo sensor. Nesse caso, é necessário dois códigos com o propósito do primeiro fazer o circuito funcionar como um receptor para identificar os sinais de pelo menos 2 botões e, o segundo, fazer o circuito ligar ou apagar o LED ao receber esses sinais.

Dessa forma, utilizou-se o código disponibilizado no Guia completo do Controle Remoto IR com a adição de comentários explicativos. O código é mostrado na Fig. 5.

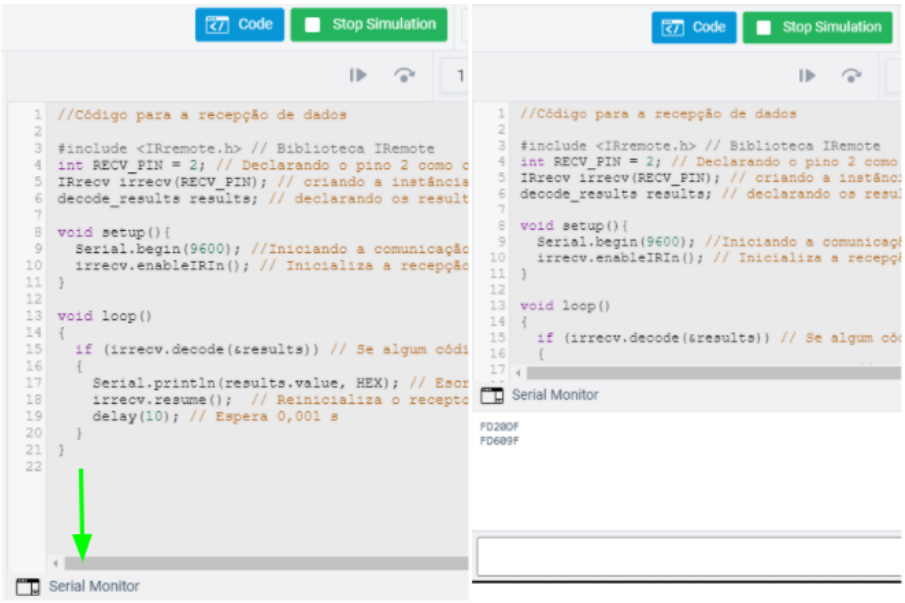
Figura 5 - Código do Arduino para identificar os sinais infravermelhos.

```
#include <IRremote.h> // Biblioteca IRemote
int RECV_PIN = 2; // Declarando o pino 2 como conectado ao Receptor IR
IRrecv irrecv(RECV_PIN); // criando a instância
decode_results results; // declarando os resultados

void setup(){
  Serial.begin(9600); //Iniciando a comunicação serial do arduino
  irrecv.enableIRIn(); // Inicializa a recepção de códigos do sinal infravermelho
}

void loop()
{
  if (irrecv.decode(&results)) // Se algum código for recebido
  {
    Serial.println(results.value, HEX); // Escreve o código no serial monitor
    irrecv.resume();  // Reinicializa o receptor
    delay(10); // Espera 0,001 s
  }
}
```

Fonte: Autor.

Com base nisso, é possível visualizar o valor lido no monitor serial de forma numérica clicando na opção “MONITOR SERIAL” no lado inferior direito da tela, indicado pela seta verde na Fig. 6.

Figura 6 - Seta indicando a opção do monitor serial.



Fonte: Autor.

Ao iniciar a simulação e clicar em qualquer botão do controle, o monitor serial apresentado os códigos lidos. Dessa forma, foram escolhidos os dois botões, um indicado pela cor verde e outro pela cor azul, e seus códigos são FD20DF e FD609F, como é mostrado na Fig. 7.

Figura 7 - Seta indicando a opção do monitor serial.

Fonte: Autor.

Para a segunda parte, também foi adaptado o código disponibilizado no Guia completo do Controle Remoto IR com a adição de comentários explicativos, como é mostrado nas Fig. 8 e Fig. 9.

Figura 8 - Configurações iniciais do código para identificar os sinais infravermelhos.

```
//Código adaptado de https:
//blog.eletrogate.com/guia-completo-do-controle-remoto-ir-receptor-ir-para-arduino
#include <IRremote.h> // Biblioteca IRemote
int RECV_PIN = 2; // Declarando o pino 2 como conectado ao Receptor IR
IRrecv irrecv(RECV_PIN);  // Criando a instância
decode_results results;   // Declarando os resultados
//Declarando as variaveis de cores dos LED como lógicas
bool Verde, Azul  = false;// Estado dos LEDs
int Led_verde = 3; // Declarando a porta 3 para o LED verde
int Led_azul = 4; // Declarando a porta 3 para o LED verde

void setup(){
  Serial.begin(9600);//Iniciando a comunicação serial do arduino
  pinMode(Led_verde, OUTPUT); //Configura a porta do LED verde de saida
  pinMode(Led_azul, OUTPUT); //Configura a porta do LED azul de saida
  irrecv.enableIRIn(); // Inicializa a recepção de códigos
}
```

Fonte: Autor.

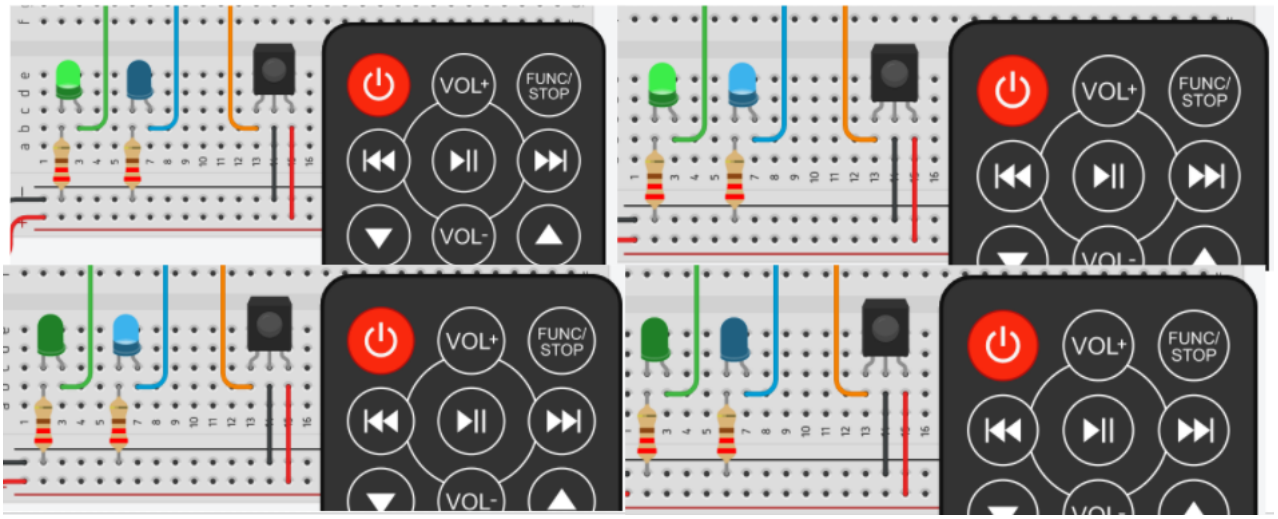
Figura 9 - Loop do código para identificar os sinais infravermelhos.

```
void loop()
{
  results.value = 0; // Zera os registradores
  if (irrecv.decode(&results))  // Se algum código for recebido
  {
    Serial.println(results.value, HEX); // Escreve o código no serial monitor
    irrecv.resume(); // Reinicializa o receptor
  }
  if (results.value == 0xFD20DF) // Se o botão do verde for pressionado
  {
    Verde = !Verde; // Alterna o estado da porta do LED verde
    digitalWrite(Led_verde, Verde); // Acende ou apaga LED verde
    delay(250);  // Espera de 0,250 s
  }
  if (results.value == 0xFD609F) // Se o botão do azul for pressionado
  {
    Azul = !Azul; // Alterna o estado da porta do LED azul
    digitalWrite(Led_azul, Azul); // Acende ou apaga LED azul
    delay(250);  // Espera de 0,250 s
  }
}
```

Fonte: Autor.

Por fim, utilizando o último código apresentado, tem-se os resultados mostrados na Fig. 10 ao pressionar o botão esquerdo, depois o direito, depois o esquerdo e, por fim, o direito novamente.

Figura 10 - Ativando os LED com o controle remoto.



Fonte: Autor.

Projeto Controle de Motor DC

Descrição:

O presente projeto irá simular o controle de um motor DC utilizando o arduino. Cabe ressaltar que o circuito foi baseado no vídeo TinkerCad 5 - Arduino Motor Transistor do canal Carlos Marques.

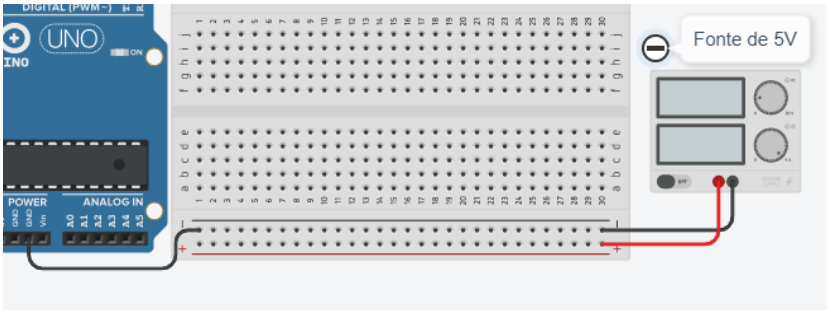
Circuito eletrônico:

- O circuito é composto:
  - 1 Arduino;
  - 1 Protoboard;
  - 1 Fonte de energia de 5 V;
  - 1 NPN Transistor;
  - 1 Motor DC;
- Montagem do circuito:

Etapa 1: Adicione o Arduino, a protoboard e 1 fonte ao circuito buscando “Arduino”, "Breadboard”, "Power supply”, respectivamente, nos componentes. É importante destacar que a fonte deve ser configurada para transmitir 5 V.

Etapa 2: Conecte a fonte no “+” e no “-” da protoboard.Por fim, conecte o GND do arduino no polo negativo da protoboard, como mostrado na Fig. 11.

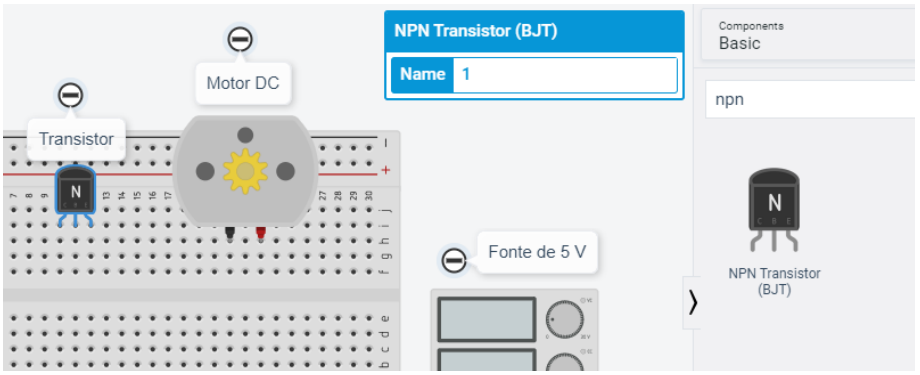
Figura 11 - Alimentação da protoboard.



Fonte: Autor.

Etapa 3: Adicione o transistor e o motor DC buscando “NPN Transistor” e "DC Motor”, respectivamente, nos componentes, como é mostrado na Fig. 12.

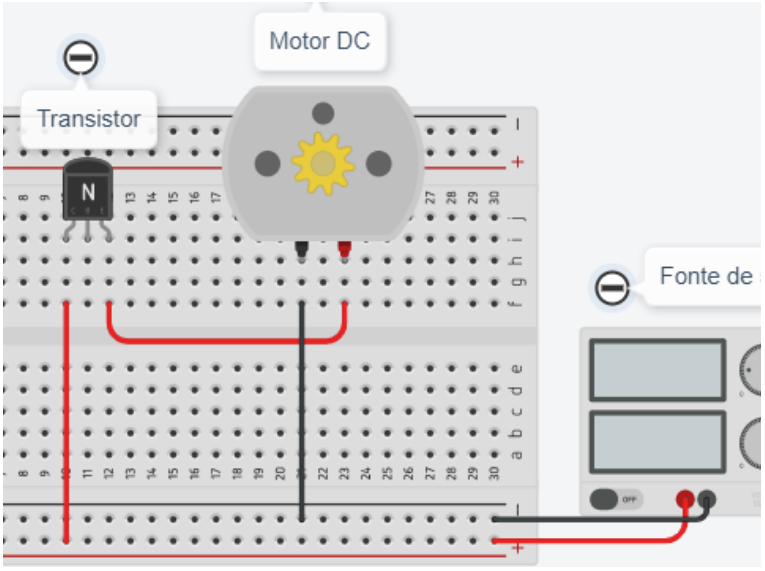
Figura 12 - Adicionando transistor e motor DC ao circuito.



Fonte: Autor.

Etapa 4: Conecte o pino positivo da bateria no pino mais a esquerda do transistor, já o pino mais a direita do transistor deve ser conectado à parte positiva do motor DC, como mostrado na Fig. 13. Por fim, conecta-se o polo negativo da bateria no outro pino do motor DC para finalizar a alimentação dele.

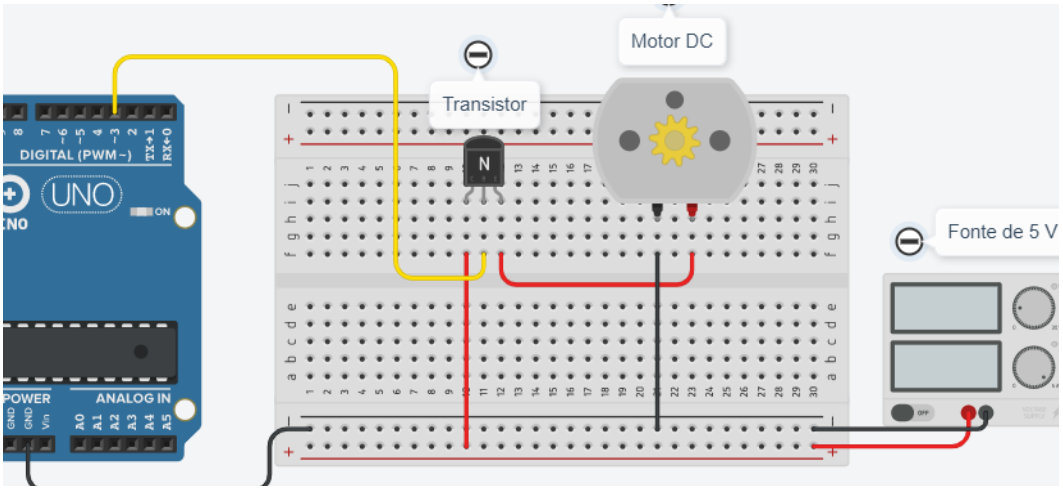
Figura 13 - Alimentação do motor DC.



Fonte: Autor.

Etapa 5: Conecte a porta 3 do arduino ao pino central do transistor, como é mostrado na Fig. 14.

Figura 14 - Conexão do arduino ao transistor.



Fonte: Autor.

Programação:

A programação é bem simples, basta configurar, inicialmente, uma porta PWM para fornecer tensões entre 0 a 5V para o transistor. Neste caso, foi configurado a porta 3 como saída de dados e, no void loop, foi utilizado o comando for para mudar a cada 100 milissegundos o valor de uma variável i. Além disso, dentro do for foi utilizado o comando analogWrite para transmitir o sinal PWM, como por ser visto na Fig. 15.

Figura 15 - Código utilizado para controlar a velocidade do motor.

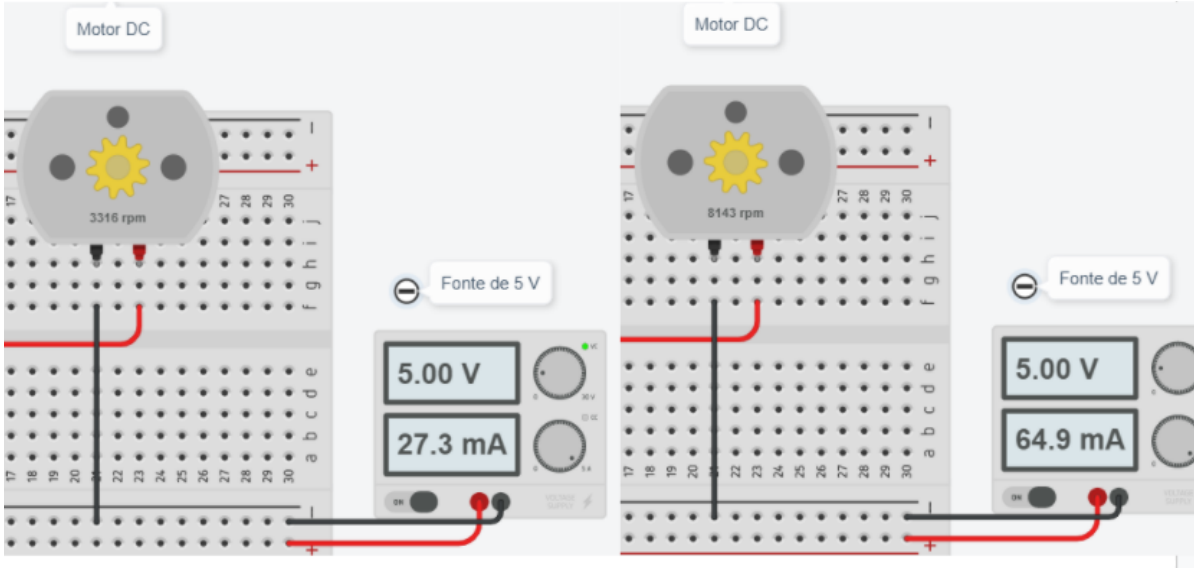
```
//Autor: Thiago Victor A. de Freitas - Estudante de Engenharia Mecânica (UFC)

//Inicialmente, devemos nomear quem vai está conectado a cada porta
int pwm = 3;

void setup()
{
  pinMode(pwm, OUTPUT);//Configura a porta pwm como porta de saida
}

void loop()
{
  for(int i = 0;i<256;i++){
/*
Comando analogWrite(porta,valor): Aciona uma onda PWM em um pino. Pode ser usada
para variar o brilho de um LED ou acionar um motor a diversas velocidades.
*/
    analogWrite(pwm,i);//Faz o motor girar proporcionalmente a i
    delay(100);//Espera 100 millisegundos
  }
}
```

Fonte: Autor.

Comentário:

Ao iniciar a simulação, aparece a quantidade de rotações por minuto do eixo do motor em cima dele, como pode ser visto na Fig. 16. Com base nessa figura, é possível perceber que, quanto maior for o RPM, maior vai ser a corrente necessária no circuito.

Figura 16 - Quantidade de rotações por minuto do eixo em dois momentos da simulação.



Fonte: Autor.

Projeto Motor DC Controlado por Infravermelho

Descrição:

O presente projeto irá simular o comando de um motor DC utilizando um controle remoto disponibilizado pelo Tinkercad. Para identificar o sinal, será usado o sensor infravermelho conectado ao arduino e este, por sua vez, irá mandar sinal PWM para modificar a velocidade do motor ou desligá-lo.

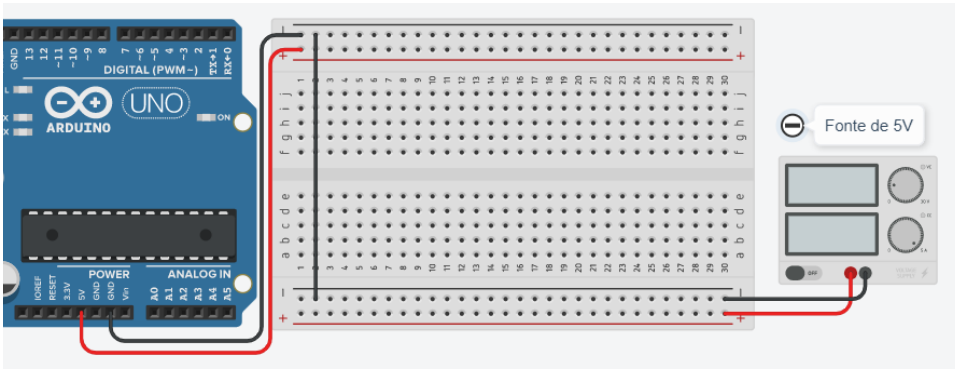
Circuito eletrônico:

- O circuito é composto:
  - 1 Arduino;
  - 1 Protoboard;
  - 1 Fonte de energia de 5 V;
  - 1 NPN Transistor;
  - 1 Motor DC;
  - 1 IR sensor;
  - 1 IR controle;
  - 2 LED;
  - 2 Resistores de 220 Ω;
- Montagem do circuito:

Etapa 1: Adicione o Arduino, a protoboard e 1 fonte ao circuito buscando “Arduino”, "Breadboard”, "Power supply”, respectivamente, nos componentes. É importante destacar que a fonte deve ser configurada para transmitir 5 V.

Etapa 2: Conecte a fonte no “+” e no “-” da protoboard. Por fim, conecte o GND do arduino no polo negativo da protoboard, como mostrado na Fig. 17. É importante ressaltar que a protoboard terá duas fontes de alimentação, o 5V do Arduino, que será utilizado apenas para o sensor infravermelho, e o 5V da fonte, que será usada para alimentar o motor DC. Isso é necessário pois este último puxa muita corrente à medida que aumenta o rpm e o Arduino não foi fabricado para alimentar circuitos, mas sim para ser o cérebro do sistema.

Figura 17 - Alimentação da protoboard.

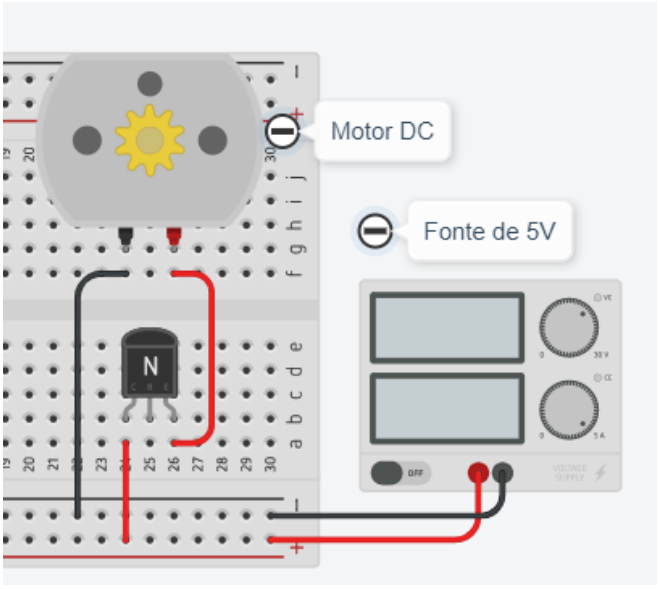


Fonte: Autor.

Etapa 3: Adicione o transistor e o motor DC buscando “NPN Transistor” e "DC Motor”, respectivamente, nos componentes.

Etapa 4: Conecte o pino positivo da fonte no pino mais a esquerda do transistor, já o pino mais a direita do transistor deve ser conectado à parte positiva do motor DC, como mostrado na Fig. 19. Por fim, conecta-se o polo negativo da bateria no outro pino do motor DC para finalizar a alimentação dele.

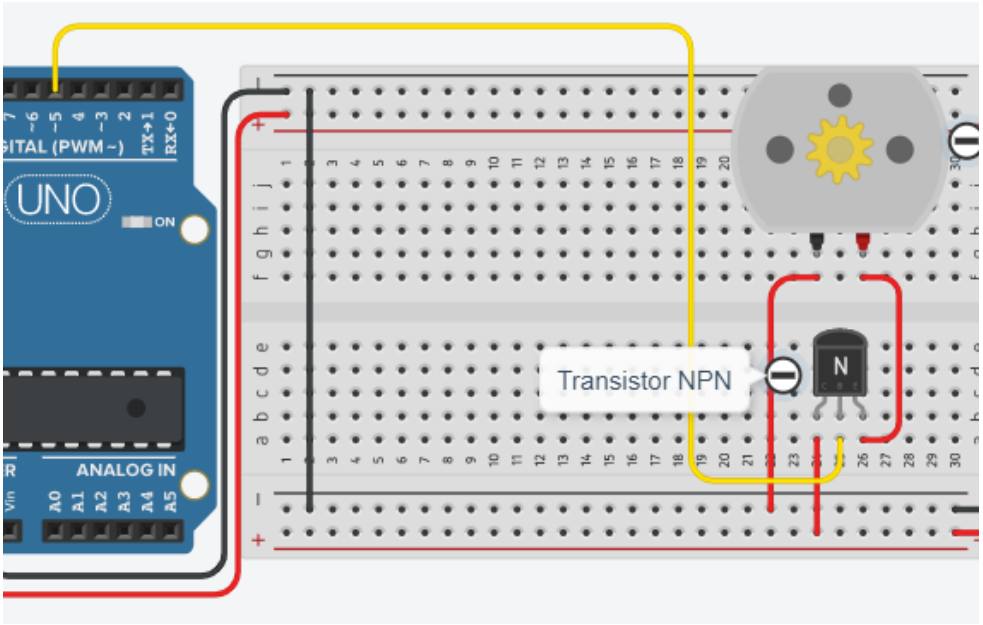
Figura 18 - Alimentação do Motor DC Controlado por Infravermelho.



Fonte: Autor.

Etapa 5: Conecte a porta 5 do arduino ao pino central do transistor, como é mostrado na Fig. 19.

Figura 19 - Conexão do arduino ao transistor.

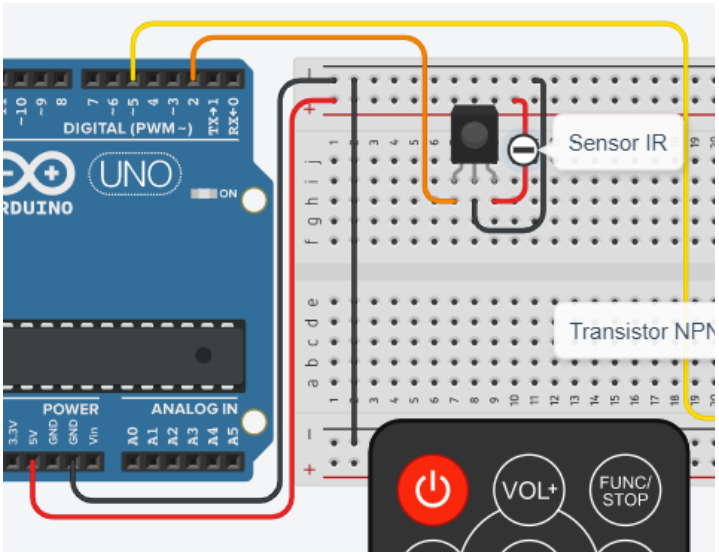


Fonte: Autor.

Etapa 6: Adicione o sensor e o controle remoto buscando “IR sensor” e "IR remote”, respectivamente, nos componentes.

Etapa 7: Alimente o sensor conectando o 5V do Arduino no pino mais a direita e o polo negativo no pino central do sensor. Além disso, conecte o pino mais à esquerda, que é o pino de saída de dados do sensor, na porta 2 do Arduino, como mostrado na Fig. 20.

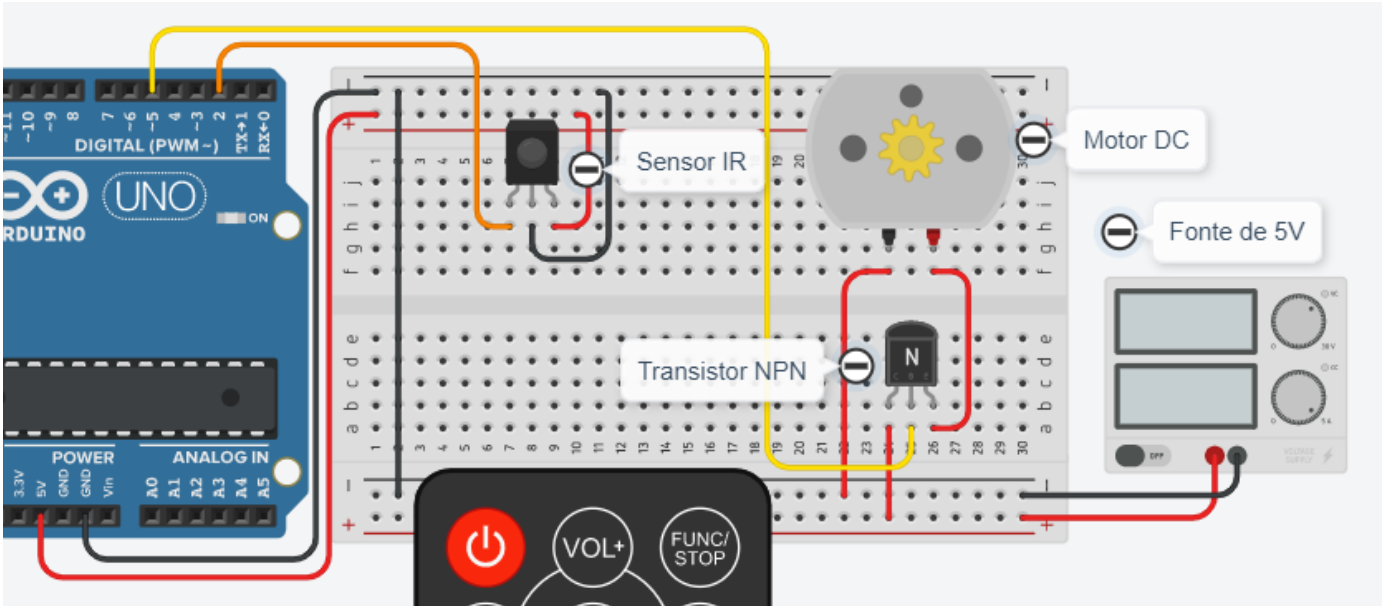
Figura 20 - Conexões do sensor infravermelho.



Fonte: Autor.

PRONTO!!! O circuito todo está criado, como é possível visualizar na Fig. 21. É válido ressaltar que foram seguidos quase que os mesmos passos dos projetos Controle Remoto e Controle de Motor DC.

Figura 21 - Circuito do projeto Controle.



Fonte: Autor.

Programação:

A programação é bem simples, foi utilizado uma adaptação dos dois códigos apresentados nos últimos projetos, alterando apenas algumas variáveis, como pode ser visto na Fig. 22. Além disso, em cada estrutura de condição, foi utilizado o comando analogWrite para indicar o RPM do eixo do motor, como é mostrado na Fig. 23.

Figura 22 - Configurações iniciais para o uso do motor junto com o infravermelho.

```
//Autor: Thiago Victor A. de Freitas - Estudante de Engenharia Mecânica (UFC)

#include <IRremote.h> // Biblioteca IRemote
int RECV_PIN = 2; // Declarando o pino 2 como conectado ao Receptor IR
IRrecv irrecv(RECV_PIN); // Criando a instância
decode_results results; // Declarando os resultados
int PWM = 5; // Declarando o pino 5 como conectado ao transitor do Motor DC

void setup(){
  Serial.begin(9600); //Iniciando a comunicação serial do arduino
  irrecv.enableIRIn(); // Inicializa a recepção de códigos
  pinMode(PWM, OUTPUT); //Configura a porta pwm como porta de saida
}
```

Fonte: Autor.

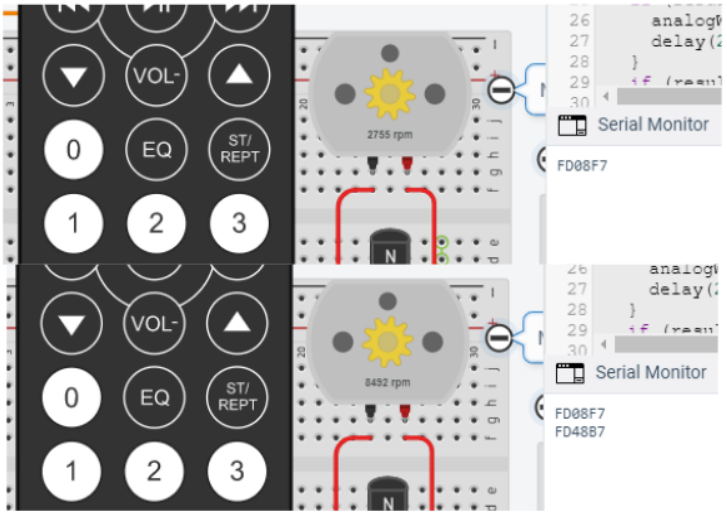
Figura 23 - Conexões do sensor infravermelho.

```
void loop() {
  results.value = 0; // Zera os registradores
  if (irrecv.decode(&results)) { // Se algum código for recebido
    Serial.println(results.value, HEX); // Escreve o código no serial monitor
    irrecv.resume(); // Reinicializa o receptor
  }
  if (results.value == 0xFD30CF) { // Se BOTAO vermelho for pressionado
    analogWrite(PWM,0); // O motor para de rodar
    delay(250);  // Espera de 0,250 s
  }
  if (results.value == 0xFD08F7) {  // Se BOTAO 1 for pressionado
    analogWrite(PWM,85); // O motor roda com proporção de 85/255
    delay(250);  // Espera de 0,250 s
  }
  if (results.value == 0xFD8877) { // Se BOTAO 2 for pressionado

    analogWrite(PWM,170); // O motor roda com proporção de 170/255
    delay(250);  // Espera de 0,250 s
  }
  if (results.value == 0xFD48B7) { // Se BOTAO 3 for pressionado
    analogWrite(PWM,255); // O motor roda com proporção de 255/255
    delay(250);  // Espera de 0,250 s
  }
}
```

Fonte: Autor.

Com base nisso, tem-se o resultado apresentado na Fig. 24, o qual, na parte superior, mostra o circuito após o botão 1 ser pressionado e, na parte inferior, mostra o circuito após o botão 3 ser pressionado.

Figura 24 - Resultados do circuito após ser pressionado os botões do controle.



Fonte: Autor.

Projeto Controle PID de Temperatura

Descrição:

O presente projeto irá simular o controle PID da temperatura de um ambiente utilizando dois motores, um para aquecer e outro para resfriar. Devido às limitações na plataforma, a temperatura do sistema deverá ser variada manualmente, porém, à medida que a temperatura se afasta do valor ideal, um dos motores roda com mais intensidade, mostrando que, em um ambiente real, o motor iria trabalhar mais para fazer o sistema atingir o valor ideal de temperatura. Cabe ressaltar que o circuito foi baseado no vídeo Arduino+PID do canal Ivan Seidel.

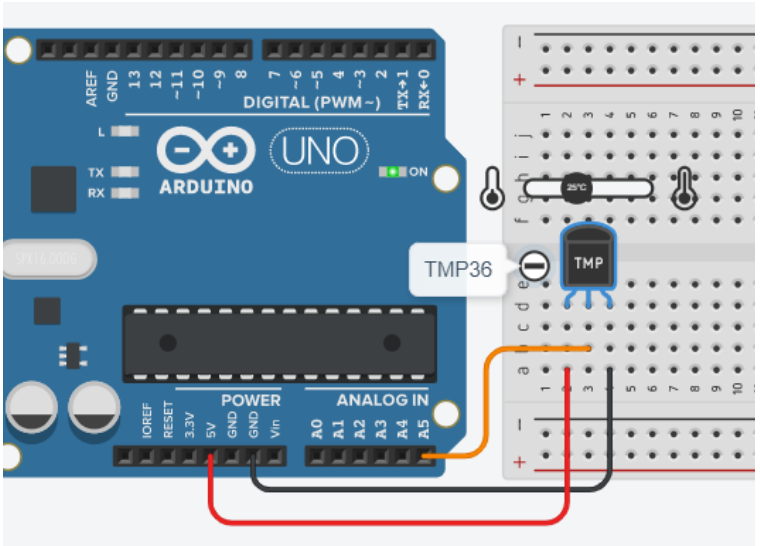
Circuito eletrônico:

- O circuito é composto:
  - 1 Arduino;
  - 1 Protoboard;
  - 1 Protoboard mini;
  - 1 Fonte de energia de 5 V;
  - 2 NPN Transistor;
  - 2 Motor DC;
  - 1 Sensor de temperatura TMP36 (Características);
- Montagem do circuito:

Etapa 1: Adicione o Arduino, a protoboard e o sensor de temperatura ao circuito buscando “Arduino”, "Breadboard” e "TMP36”, respectivamente, nos componentes.

Etapa 2: Alimente o sensor conectando o 5 V do Arduino no pino mais a esquerda e o GND no pino mais a direita, como mostrado na Fig. 25. Além disso, conecte o pino central, de transferência de dados, na porta A5 do Arduino.

Figura 25 - Conexões do sensor de temperatura ao Arduino.

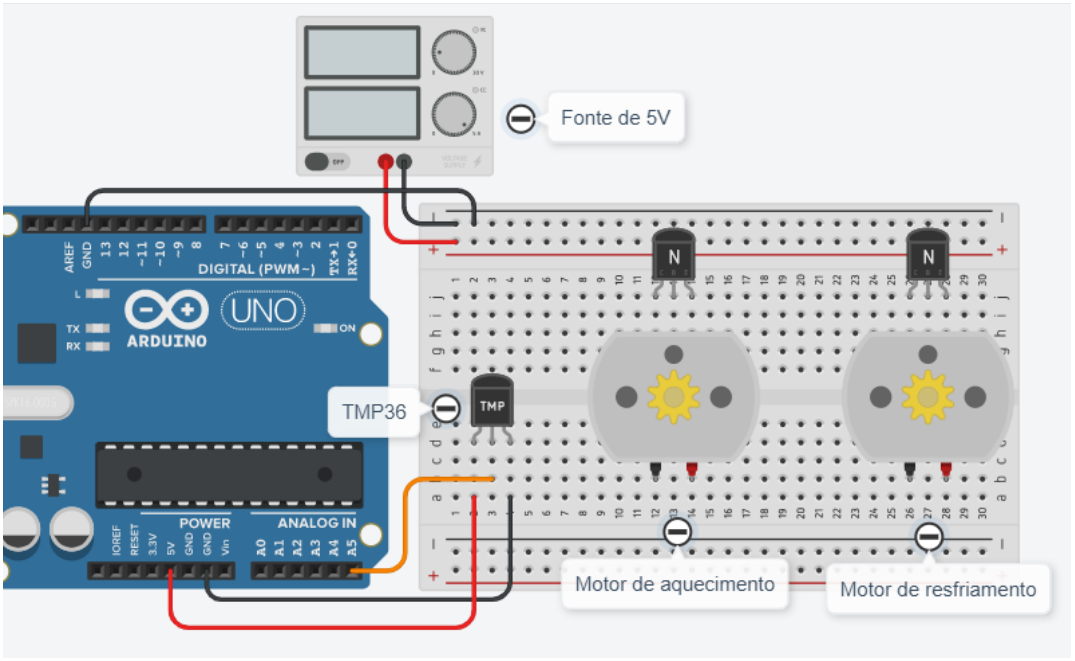


Fonte: Autor.

Etapa 3: Adicione 2 transistores e 2 motores DC ao circuito buscando “NPN Transistor” e "DC Motor”, respectivamente, nos componentes.

Etapa 4: Adicione a 1 fonte de alimentação de 5 V ao circuito buscando "Power Supply" nos componentes. Conecte a fonte no "+" e no "-" da protoboard. Por fim, conecte o GND do arduino no polo negativo da protoboard, como mostrado na Fig. 26.

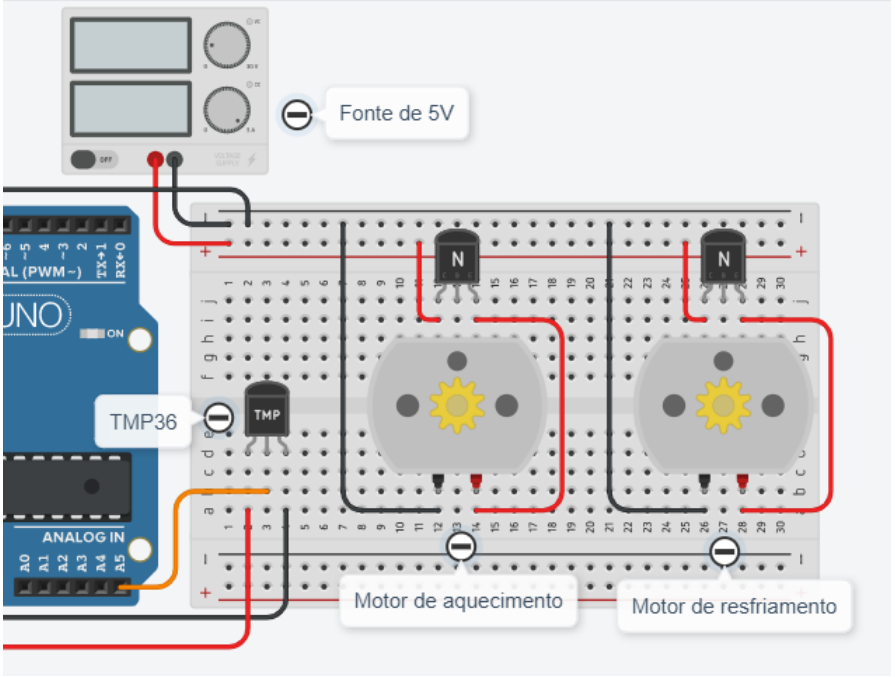
Figura 26 - Adição dos motores e alimentação da protoboard.



Fonte: Autor.

Etapa 4: Conecte o pino positivo da fonte no pino mais a esquerda do transistor, já o pino mais a direita do transistor deve ser conectado à parte positiva do motor DC, como mostrado na Fig. 27. Por fim, conecta-se o polo negativo da bateria no outro pino do motor DC para finalizar a alimentação dele. Esse processo deve ser repetido para os dois motores

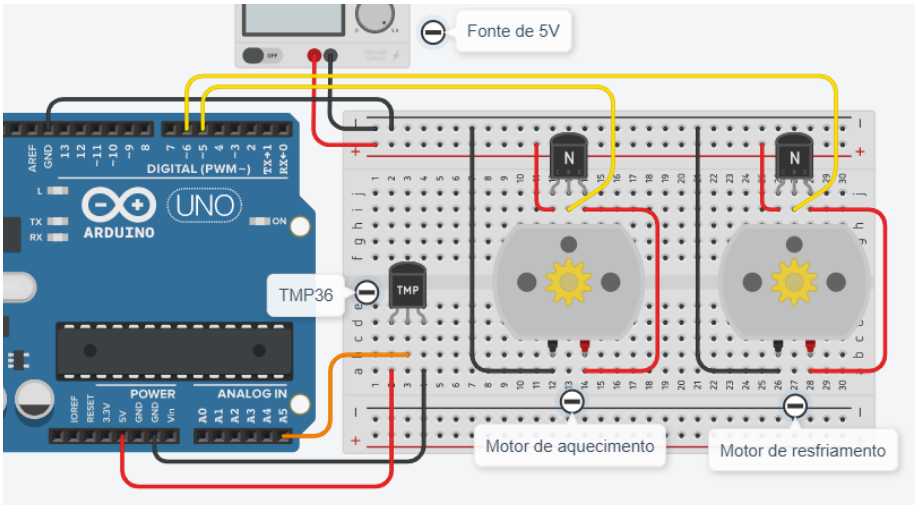
Figura 27 - Alimentação dos Motores DC.



Fonte: Autor.

Etapa 5: Conecte a porta 5 e a porta 6 do arduino ao pino central dos transistores, como é mostrado na Fig. 28.

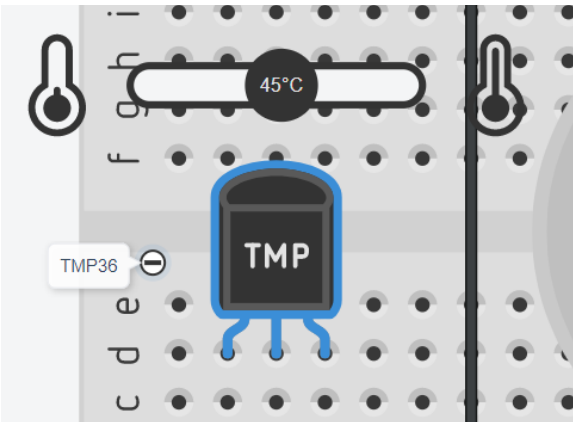
Figura 28 - Conexões entre o Arduino e os transistores.



Fonte: Autor.

Por fim, é possível variar a temperatura do meio clicando no sensor TMP36 e deslocando a esfera preta acima do sensor, conforme mostrado na Fig. 29.

Figura 29 - Ajuste de temperatura no circuito.



Fonte: Autor.

Programação:

A explicação do código foi dividida em várias partes devido às complexidades do programa. Inicialmente é mostrado na Fig. 30 a definição dos pinos conforme o circuito criado.

Figura 30 - Inclusão da biblioteca e declaração dos pinos.

```
1  //Autor: Thiago Victor A. de Freitas - Estudante de Engenharia Mecânica (UFC)
2  #include <math.h>
3  #define pino_do_sensor  A5 // Declarando o pino A5 como conectado ao sensor de temp
4  #define motor_de_aque   5  // Declarando o pino 5 como conectado ao motor de aquecimento
5  #define motor_de_resf   6  // Declarando o pino 5 como conectado ao motor de resfriamento
```

Fonte: Autor.

Após isso, tem-se a criação das variáveis utilizadas nos cálculos como o erro, o qual será dado pela diferença entre o valor ideal de temperatura e o valor medido. Aqui é válido destacar que são definidas os valores das constantes de proporcionalidade, integração e derivação, os quais seus efeitos são descritos com mais detalhes na próxima seção.

Figura 31 - Criação das variáveis importantes para o cálculo do PID.

```
7  //Criando variáveis para os calculos
8  double error; //Variável usada para calcular o erro
9  double temp; //Variável para armazenar o valor lido da temp
10 double ultima_temp; //Variável usada para armazenar o ultimo valor lido da temp
11 double kP = 1.0, kI = 0.0, kD = 0.0; //Constante de proporcionalidade, integração e derivação
12 //Aqui é valido ressaltar que, para baixos kP, tem-se uma estabilidade maior, porém
13 // o sistema irá demorar mais para atingir ao estado ideal
14 double P, I, D; //Variável para armanezar o valor proporcional, integrativo e derivativo
15 double PID; //Variável usada para armazenar o PID
16 float dt; //Variável usada para calcular a variação de tempo
17 long ultimo_tempo; //Variável usada para armazenar o tempo decorrido até a ultima medição de temp
18
```

Fonte: Autor.

A Fig. 32 mostra a criação das variáveis de controle para o código, como o set point de 50ºC , que é temperatura ideal a qual o sistema deve ficar, e o controle_ideal de 0, sendo este último proporcional ao RPM que o motor DC deve fornecer quando a temperatura do sistema é ideal e, no código, foi dado o valor de 0.

Figura 32 - Criação das varíaveis para as condições ideais.

```
19 double setPoint = 50; //Valor da temperatura ideal
20 int controle_ideal = 0; //Valor de RPM para quando temp for ideal
21 //Quando a temp for 50, o motor deve ser desligado, por isso é igual a zero
22 int controlePwm = 0; //Variavel usada para indicar a rotação do motor
23
```

Fonte: Autor.

Para finalizar a inicialização do código, tem-se as configurações do setup na Fig. 33, para declarar os as portas como de entrada ou saída de dados bem como iniciar a comunicação serial para printar os valores medidos e a variável de controle.

Figura 33 - Setup do código para as configurações das portas.

```
24 void setup() {
25     Serial.begin(9600); //Iniciando a comunicação serial do arduino
26     pinMode(pino_do_sensor, INPUT); //Configura a porta do sensor como entrada de dados
27     pinMode(motor_de_aque , OUTPUT); //Configura a porta do motor como saída de dados
28     pinMode(motor_de_resf , OUTPUT); //Configura a porta do motor como saída de dados
29 }
30
```

Fonte: Autor.

A Fig. 34 mostra que a primeira parte do loop é realizar a leitura da temperatura e convertendo para ºC utilizando a seguinte fórmula mostradas em capítulos anteriores, já que o sensor não faz esse processamento do dado.

$$Temperatura(\circ\ C) = \frac{\frac{analogRead(porta)*5000}{1024.0} - 500}{10}$$

Figura 34 - Leitura da temperatura.

```
31 void loop() {
32     // Lendo a temperatura e convertendo para graus celsius
33     temp = (analogRead(pino_do_sensor)*(5000/1024.0) - 500)/10;
34     Serial.print("Temp = "); //Escreve no monitor serial
35     Serial.print(temp); //Escreve no monitor serial
36     Serial.print("*C ; "); //Escreve no monitor serial
37
```

Fonte: Autor.

Com base no valor de temperatura, é calculado o erro pela diferença com valor de setPoint (temperatura ideal a qual o sistema deve ficar), como é mostrado na Fig. 35. Além disso, é utilizado o comando millis(), pois ele retorna o valor de tempo decorrido desde que o circuito foi ligado, para calcular o valor da variação de temperatura desde a última medição do loop passado.

Figura 35 - Computando o erro e o tempo decorrido.

```
38      // Implementação PID
39      error = setPoint - temp; //Calculo do erro
40      dt = (millis() - ultimo_tempo) / 1000.0; //Calculo da varição do tempo
41      ultimo_tempo = millis(); //Armazenando o tempo decorrido
```

Fonte: Autor.

Após isso são computados os parâmetros proporcional, integrativo e derivativo e soma-se tudo na variável PID, como é mostrado na Fig. 36.

Figura 36 - Cálculo do PID.

```
43      P = error * kP; //Calculo da proporção
44      I = I + (error * kI) * dt; //Calculo d integração
45      D = (ultima_temp - temp) * kD / dt; //Calculo da derivação
46      ultima_temp = temp; //Armazenando o ultumo valor de temp lido
47      PID = P + I + D; // Soma tudo
```

Fonte: Autor.

Na linha 52 da Fig. 37 é possível visualizar que a estimativa de controle é dada pelo PID calculado somado ao valor de controle ideal.

Além disso, foi criado uma estrutura de condição com a seguinte lógica: se o valor do controle der negativo, então a temperatura dele está acima do valor ideal, logo o motor de resfriamento deve ser ativo e o de aquecimento desligado. Com isso, foi criado uma adaptação para deixar o valor positivo, pois a função abs() estava travando o código e, caso o valor esteja negativo, o motor iria rodar inicialmente com a velocidade máxima e ir baixando com a diminuição do valor, logo o projeto ia ter uma alta sensibilidade para o motor de resfriamento, podendo causar algum erro. De forma análoga, caso o número seja positivo, isso significa que a temperatura está abaixo da ideal, logo ele vai ativar o motor de aquecimento e desligar o de resfriamento.

Figura 37 - Estrutura condicional para ativar ou desativar os motores.

```
49      Serial.print("PID = "); //Escreve no monitor serial
50      Serial.print(PID); //Escreve no monitor serial
51      Serial.print(" ; "); //Escreve no monitor serial
52      controlePwm = PID+controle_ideal;
53      Serial.print("Controle = "); //Escreve no monitor serial
54      Serial.print(controlePwm); //Escreve no monitor serial
55      Serial.println(" ;"); //Escreve no monitor serial
56
57      if (controlePwm < 0) { //Se o controle for maior que zero, ele ativa o resfriamento
58          controlePwm = sqrt((PID + controle_ideal)*(PID + controle_ideal));
59          //Foi tirado a raiz quadrada do numero ao quadrado para deixar o valor positivo
60          analogWrite(motor_de_resf, controlePwm);// Saída do controle
61          analogWrite(motor_de_aque, controle_ideal);
62      }else{ //Se o controle for maior que zero, ele ativa o aquecimento
63          analogWrite(motor_de_aque, controlePwm); // Saída do controle
64          analogWrite(motor_de_resf, controle_ideal);
65      }
```

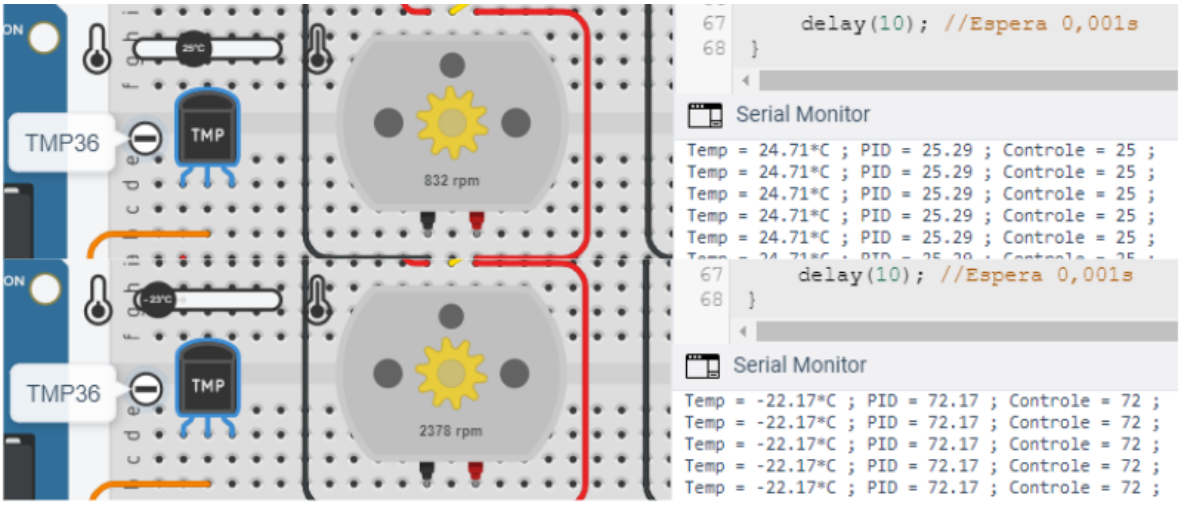
Fonte: Autor.

Para finalizar, o código utiliza o comando delay para fazer o código esperar 0,01s antes de iniciar o próximo loop.

Comentários:

Com base nesse código tem-se o seguinte resultado mostrado na Fig. 38, o qual o Arduino liga o motor de aquecimento quando a temperatura está abaixo da ideal de 50 ºC e, como é possível verificar na parte mais abaixo da figura, o motor fornece um RPM maior a temperatura está mais distante da ideal.

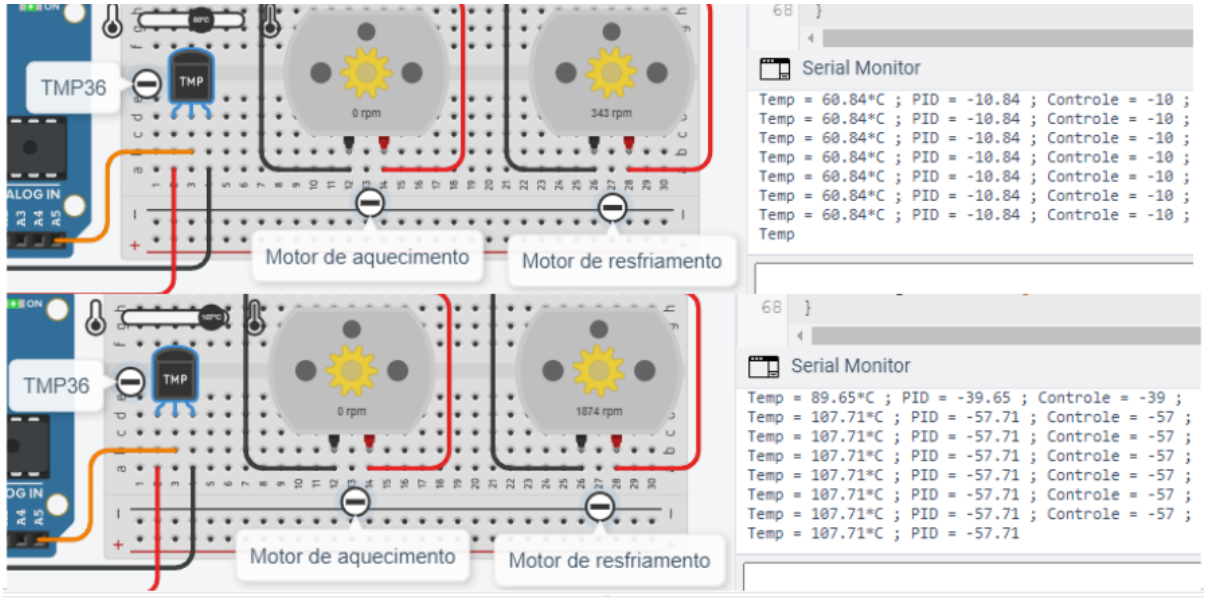
Figura 38 - Ativação do motor de aquecimento.



Fonte: Autor.

Além disso, é importante ressaltar que, caso a temperatura esteja acima de 50 ºC, o motor de aquecimento é desligado e o motor de resfriamento é ativado com a mesma lógica explicada anteriormente, como é possível visualizar na Fig. 39.

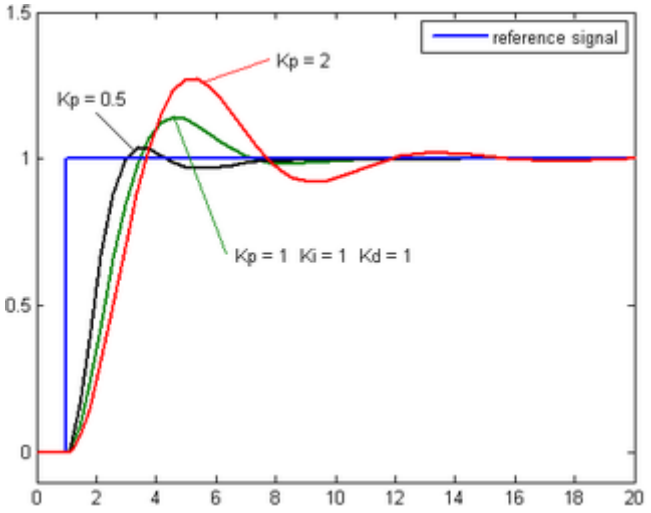
Figura 39 - Ativação do motor de resfriamento.



Fonte: Autor.

Também é válido ressaltar que as constantes utilizadas para o cálculo do PID são importantes para dar mais estabilidade para o projeto. De inicio, tem-se a comparação mostrada na Fig. 40 para a constante de proporcionalidade, o qual, segundo o site Laboratório de Garagem, para valores maiores de Kp, tem-se um sistema mais instável.

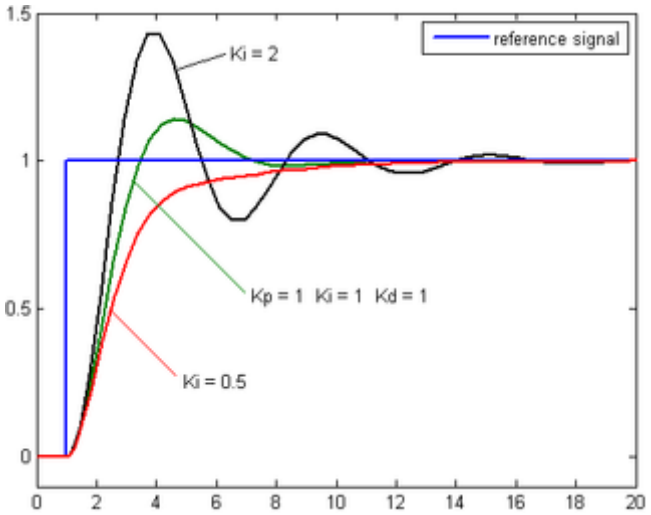
Figura 40 - Variações da constante de proporcionalidade.



Fonte: Laboratório de Garagem.

Para altos Ki, tem-se uma aceleração do movimento do processo, ou seja, o sistema atinge valores maiores e mais rápidos, como é mostrado na Fig. 41.

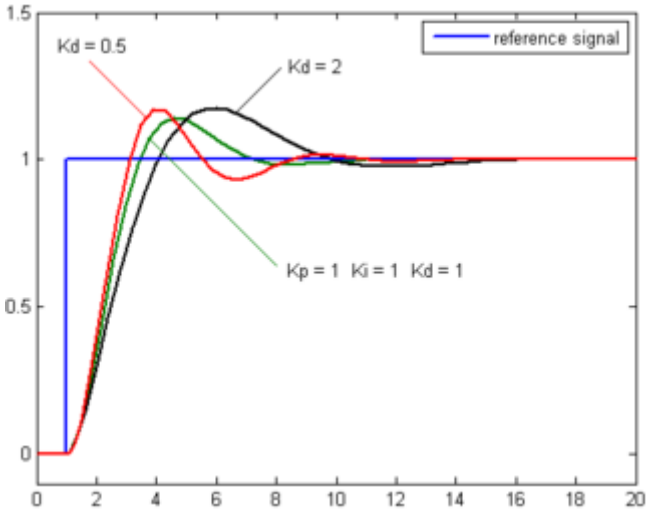
Figura 41 - Variações da constante de integração.



Fonte: Laboratório de Garagem.

Por fim, a constante Kd melhora a estabilidade do sistema, mas causa um retardo na resposta do controlador e é muito suscetível a ruídos, como é possível visualizar na Fig. 42.

Figura 42 - Variações da constante de derivação.



Fonte: Laboratório de Garagem.

É importante ressaltar que, devido às limitações na plataforma, não é possível visualizar eficientemente os efeitos dessas constantes, pois, como foi explicado anteriormente, é necessário modificar manualmente a temperatura do sistema.

Referências

Sites:

- Guia completo do Controle Remoto IR + Receptor IR para Arduino do autor José Gustavo Abreu Murta.
- **Circuito do projeto Controle Remoto**.
- **Circuito do projeto Controle de Motor DC**.
- Vídeo do canal Tecnotrônica mostrando uma aplicação de controle de velocidade do motor.
- **Circuito do projeto Motor DC Controlado por Infravermelho**.
- Artigo sobre Controle de Temperatura PID com Arduino.
- Artigo sobre Controle PID Básico.
- Artigo sobre Controle PID - Introdução
- **Circuito do projeto Controle PID de Temperatura**.
- Vídeo sobre Sistema de Bombeamento de Água | Arduino - TinkerCad + Explicação