ARDUINO **KIT** DESENVOLVIMENTO **V0.2**

ARDUINO

**KIT** DESENVOLVIMENTO **V0.2**

Está apostila é uma introdução na área da robótica. Formada por dez projetos que irão do fácil ao difícil, com seis níveis de dificuldades que eu mesmo classifiquei. O objetivo desta apostila é repassar o conhecimento de eletrônica e programação, para desafiar o usuário e fazer com que o mesmo adquira certo gosto pela área. Mas também está apostila tem como objetivo explicar em seus projetos a teoria de como funciona diversos equipamentos que utilizamos em nosso cotidiano, que diversos usuários não fazem a menor idéia como eles funcionam.

Os experimentos serão divididos em 6 níveis de dificuldade, sendo representados da seguinte forma:

: Iniciante;

: Fácil;

: Intermediário

: Difícil;

: Muito difícil;

: Avançado;

Dificuldade mínima:

Dificuldade máxima:

Abaixo temos as descrições dos projetos e os projetos que iremos fazer nesta apostila:

* **Módulo 1**

Componentes: Arduino + Cabo de comunicação

Descrição: Ao executar o comando *Blink* do Arduino, um led acoplado no Arduino irá ligar e desligar variando o estado conforme o tempo.

Dificuldade:

* **Módulo 2**

Componentes: Arduino + Cabo de comunicação

Descrição: Usaremos o monitor serial do Arduino para ligar e desligar o Led acoplado no Arduino.

Dificuldade:

* **Módulo 3**

Componentes: Arduino + Cabo de comunicação + Led + Protoboard + Resistor de 2200Ω + Jumper’s + Push Button

Descrição: Com o acionamento do Push Button o Led irá ligar ou desligar.

Dificuldade:

* **Módulo 4**

Componentes: Arduino + Cabo de comunicação + Led + Resistor de 2200Ω + Protoboard + Potenciômetro + Jumper’s

Descrição: Conforme variamos o Potenciômetro, o Led Liga e desliga em uma velocidade variada.

Dificuldade:

* **Módulo 5**

Componentes: Arduino + Cabo de comunicação + 4 Leds + 4 Resistores de 2200Ω + Protoboard + Jumper’s + Potenciômetro

Descrição: Conforme variamos o potenciômetro, os leds ligam e desligam gradativamente.

Dificuldade:

* **Módulo 6**

Componentes: Arduino + Cabo de comunicação + Led + Resistor de 2200Ω + Protoboard + Resistor de 10KΩ + LDR + Jumper’s + Buzzer

Descrição: Conforme a claridade da sala, o LDR manda um sinal ao Arduino ligando ou desligando o Led. O Buzzer é acionado quando não tem claridade suficiente na sala.

Dificuldade:

* **Módulo 7**

Componentes: Led + Emissor de Infravermelho + Receptor de Infravermelho + 2 Resistores 2200Ω + Jumper’s + Protoboard + Arduino + Cabo de comunicação

Descrição: Conforme o Infravermelho detecta o objeto o Led liga.

Dificuldade:

* **Módulo 8**

Componentes: 2 Emissor de Infravermelho + 2 Receptor de Infravermelho + 2 Resistores 2200Ω + Jumper’s + Protoboard + Arduino + Cabo de comunicação + Buzzer

Descrição: Conforme cada um dos 2 emissores detectam um objeto, o Buzzer é acionado. Para cada emissor IR, o Buzzer irá tocar uma nota diferente.

Dificuldade:

* **Módulo 9**

Componentes: Arduino + Cabo de comunicação + QTI + Jumper’s + Protoboard

Descrição: Conforme o QTI detecta a fita de alto contraste da mesa, o Monitor serial mostra o valor desta variação.

Dificuldade:

* **Módulo 10**

Componentes: Arduino + Cabo de comunicação + Potenciômetro + Servo motor + Jumper’s + Protoboard

Descrição: Ao variar o Potenciômetro o sentido de rotação do Servo motor oscila.

Dificuldade:

* **Módulo 11**

Componentes: Arduino + Cabo de comunicação + QTI + Jumper’s + Servo Motor + Protoboard

Descrição: Conforme o QTI detecta a fita de alto contraste da mesa, o Servo motor varia seu sentido de rotação.

Dificuldade:

* **Módulo 12**

Componentes: Arduino + Cabo de comunicação + Ultra HC-SR04 + Jumpers + Protoboard

Descrição: Conforme o Ultra detecta um objeto na distancia X, o Monitor serial mostra o valor da distancia X.

* **Módulo 13**

Componentes: Arduino + Cabo de comunicação + Ultra HC-SR04 + Jumper’s + Protoboard + Servo motor

Descrição: Conforme a distancia que o Ultra detecta o objeto, o Servo motor varia sua rotação.

Dificuldade:

* **Módulo 14**

Componentes: Arduino + Cabo de comunicação + Ultra HC-SR04 + Jumper’s + Protoboard + QTI + Led + Resistor 2200Ω

Descrição: O QTI irá mandar um sinal ao Arduino se encontrar algum contraste na superfície plana, acionando o servo motor em sentindo horário. O HC-SR04, melhor dizendo o Ultrassónico, se detectar algum objeto em sua direção, irá anular o comando do QTI, fazendo o Servo motor parar acionando um led que irá ligar e desligar a cada 1 segundo, até o ultrassónico não identificar mais nenhum objeto.

Dificuldade:

|  |  |  |
| --- | --- | --- |
| Nome | Descrição | Foto |
| Arduino | È uma plataforma com uma programação padrão com entradas e saídas. |  |
| Cabo de comunicação | Cabo que comunicamos o Arduino com um computador para a programação. |  |
| Protoboard | Matriz de contatos que usamos para fazer experimentos eletrônicos. |  |
| Resistor | Componente eletrônico usado para a filtragem de corrente elétrica. |  |
| Led’s | È um componente eletrônico chamado de Led. Um diodo emissor de luz. |  |
| Jumper’s | São condutores de eletricidade que se usam na eletrônica para experimentos e projetos. |  |
| Potenciômetro | É um resistor variável mecanicamente usado na eletrônica. |  |
| Push Button | É um interruptor eletrônico usado para intervir na corrente elétrica mecanicamente. |  |
| Buzzer | É um componente usado na eletrônica para sinalização sonora. |  |

Tabela de todos os componentes utilizados na apostila:

|  |  |  |
| --- | --- | --- |
| Nome | Descrição | Foto |
| Infravermelho | Led emissor de infravermelho usado na eletrônica para emitir uma frequência de sinal infravermelho. |  |
| Receptor Infravermelho | Receptor infravermelho usado para receber e identificar a frequência do emissor infravermelho. |  |
| QTI | Componente usado na robótica para sensoriamento de contraste em uma superfície plana. |  |
| HC-SR04 (Ultrassónico) | Componente usado na robótica que funciona como um sonar identificando objetos. |  |
| Servo motor | Servo motor transforma a corrente elétrica em movimento mecânico. |  |
| LDR | Componente eletrônico que identifica a luminosidade do ambiente como uma fotocélula. |  |

Primeiramente iremos entender melhor um dos componentes mais utilizados na eletrônica, os resistores.



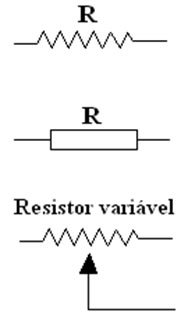
Os**resistores** são encontrados em diversos aparelhos eletrônicos como, por exemplo, televisores, rádios e amplificadores.

Um **resistor** pode ser definido como sendo um dispositivo eletrônico que tem duas funções básicas: ora transforma energia elétrica em energia térmica (efeito joule), ora limita a quantidade de corrente elétrica em um circuito, ou seja, oferece resistência à passagem de elétrons.

Os **resistores** são fabricados basicamente de carbono, podendo apresentar resistência fixa ou variável. Quando os resistores apresentam resistência variável passam a ser chamados de potenciômetros ou reostatos.

Encontramos resistores mais comumente nos chuveiros elétricos, nos filamentos das lâmpadas incandescentes, em aparelhos eletrônicos, etc.

Basicamente os resistores são representados da seguinte maneira:



Representação de resistores

Podemos então definir a resistência elétrica da seguinte maneira:



Onde:

**R** – é a resistência elétrica medida em *ohm* (**Ω**)

**U** – é a tensão medida em *volt*(**V**)

**i** – é a corrente elétrica medida em *ampère* (**A**)

Fonte: <http://www.brasilescola.com/fisica/resistores.htm>

Como os resistores são componentes muito pequenos e devem ter os valores de suas resistências facilmente identificados, costuma-se codificar este valor com o uso de uma série de faixas coloridas no corpo do resistor. Cada cor representa um algarismo. Para lermos o código do valor da resistência impresso no resistor devemos considerar que:

1.      As faixas coloridas são lidas a partir da que está mais próxima de uma extremidade.

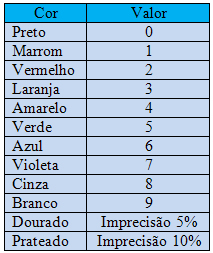
2.      A primeira faixa colorida representa o primeiro algarismo do valor da resistência.

3.      A segunda faixa colorida indica o segundo algarismo.

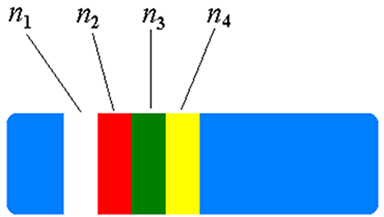
4.      A terceira faixa representa a potência de dez pela qual devemos multiplicar os dois algarismos.

5.      A quarta faixa, que é opcional, indica imprecisão no valor da resistência. Prateado indica 10% de imprecisão, dourado indica 5% e a ausência desta faixa representa imprecisão de 20%.

Na tabela a seguir estão as cores e seus respectivos valores.



Um resistor, com as cores branca, vermelha e verde, terá uma resistência de 92 x 105 Ω. Um resistor de 47 Ω, com imprecisão de 5%, terá as cores: amarela, violeta, preta e dourada.



Fonte: <http://www.brasilescola.com/fisica/codigo-cores-para-resistores.htm>

**EXPERIMENTOS**

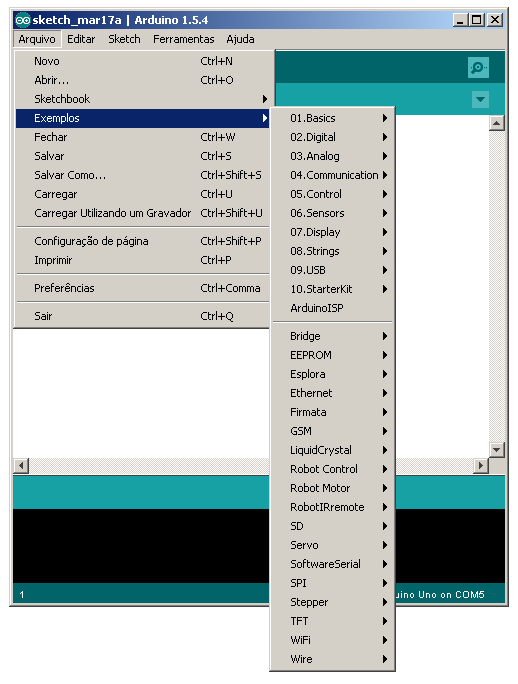
* **Módulo 1**

Componentes: Arduino + Cabo de comunicação

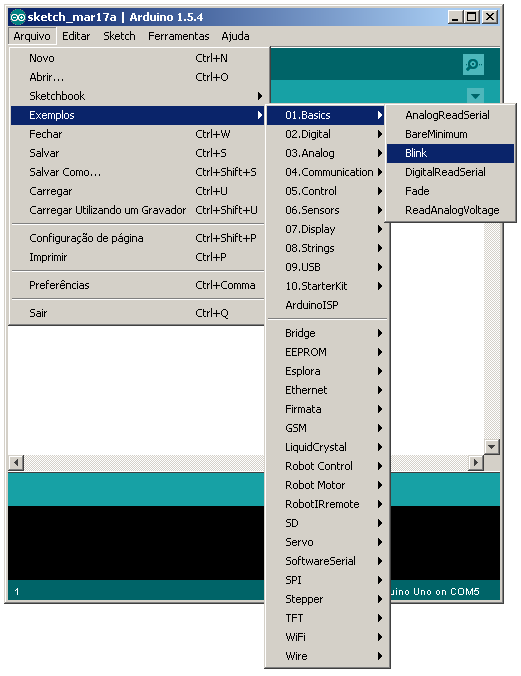
Descrição: Ao executar o comando *Blink* do Arduino, um led acoplado no Arduino irá ligar e desligar variando o estado conforme o tempo.

Dificuldade:

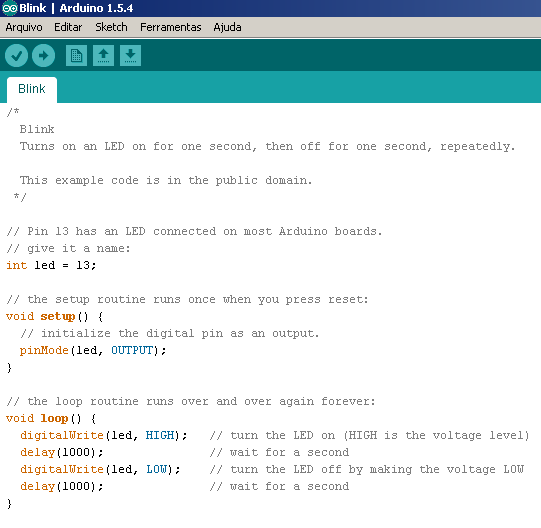
Primeiramente entramos na IDE do Arduino, vamos a *Arquivo****>****Exemplos****>****01.Basics*.



Dentro do *01.Basics* clicamos em *Blink.*



Assim irá ficar nosso código:



Irei agora explicar cada linha de código:

No comando “ int led = 13; ”, especificamos para o Arduino que no pino 13, ou melhor, na porta 13 do Arduino, a um Led conectado.

Como no Arduino quando acionamos a porta 13 como saída, liga o led acoplado no Arduino, pois este led está acoplado na porta 13. Ao invés de utilizarmos um led externo em uma protoboard, iremos usar neste exemplo o led acoplado na porta 13 do Arduino, para iniciarmos no básico.

“void **setup** () {}“ Dentro destas chaves iremos especificar para o Arduino, se o pino x é uma entrada “INPUT” ou uma saída “OUTPUT“, neste Modulo especificamos da seguinte maneira “pinMode(led, OUTPUT);”, especificando para o Arduino, que o “led” é uma saída.

“Void **loop** () {}” Dentro destas chaves á o loop do nosso código. No comando “digitalwrite(led, HIGH);” estamos acionando o led, já especificado no “void setup” que está na porta 13 do Arduino como uma saída. A sintaxe “digitalwrite” vem de porta digital, significando o acionamento da porta digital.

“delay(1000);” Interrompe o programa para o período de tempo (em milissegundos) especificado como parâmetro. (Há 1000 milissegundos em um segundo.)

“digitalwrite(led, LOW);” Especifica que a porta digital esta desligando o led.

Conclusão do código: o led acoplado no Arduino na porta digital 13 irá ligar, esperar 1000 milissegundos, que equivale a 1 segundo, e desligar por 1000 milissegundos. Está programação está no void loop, então será executada e depois voltara para o ponto inicial do código, assim fazendo um looping, ligando e desligando o led acoplado no Arduino.

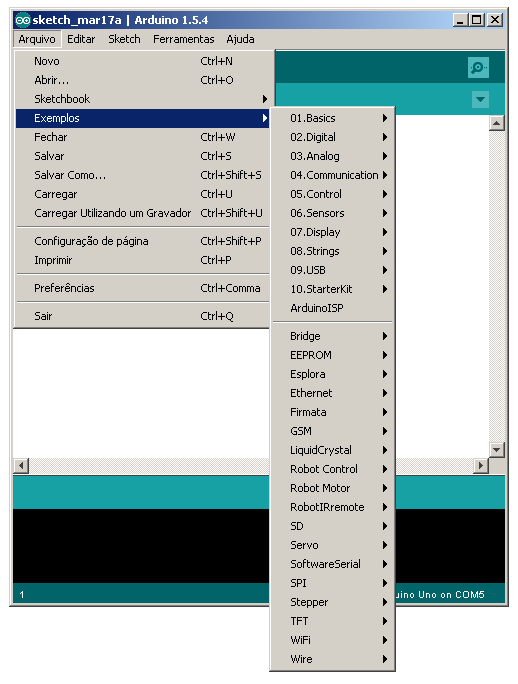
* **Módulo 2**

Componentes: Arduino + Cabo de comunicação

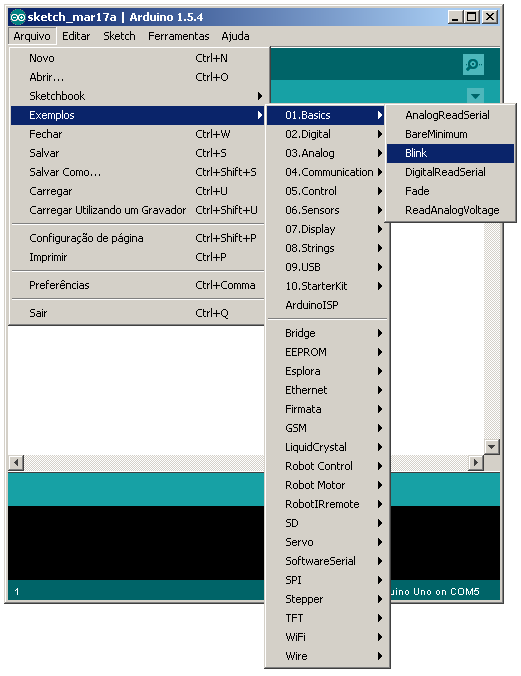
Descrição: Usaremos o monitor serial do Arduino para ligar e desligar o Led acoplado no Arduino.

Dificuldade:

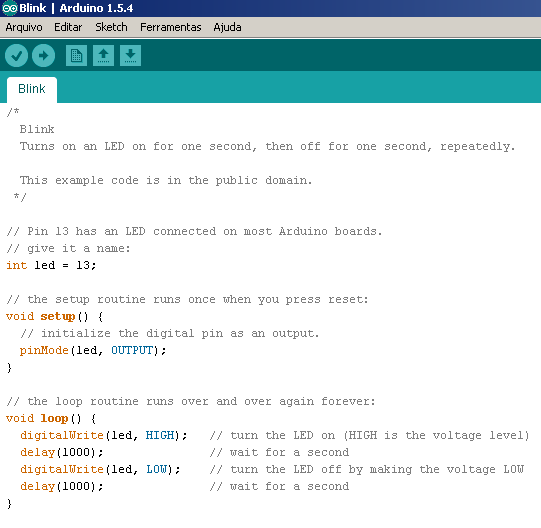
Primeiramente iremos utilizar o mesmo exemplo que o modulo 1, só que faremos algumas alterações. Entramos na IDE do Arduino, vamos a *Arquivo****>****Exemplos****>****01.Basics*.



Dentro do *01.Basics* clicamos em *Blink.*



Assim irá ficar nosso código:



Agora faremos algumas alterações no código.

int led = 13;

char leituraserial;

void setup() {

pinMode(led, OUTPUT);

Serial.begin(9600);

}

void loop() {

leituraserial = Serial.read();

if (leituraserial == 'l') {

digitalWrite(led, HIGH);

}

if (leituraserial == 'd') {

digitalWrite(led, LOW);

}

}

Irei agora explicar o que há de diferente nesta programação.

“char leituraserial;” Nesta segunda linha temos a sintaxe “char”, que nada mais é que um tipo de variável que armazena ao invés de números, letras. Então a variável “leituraserial”, especificada do tipo “char”, armazena nela apenas letras.

“Serial.begin(9600);” Nesta linha informamos ao Arduino a velocidade em bits da comunicação serial, ou seja, a comunicação do Arduino com o computador.

“leituraserial = Serial.read();” No void loop começo especificando para Arduino que a variável “leituraserial” é igual a “Serial.read();”. Serial.read é a leitura do serial monitor.

“if (leituraserial == 'l')” Nesta linha temos o “If”, traduzindo para o português fica “se”. Utilizamos na programação desta forma:

If (x > 10){

Digitalwrite(led, HIGH);

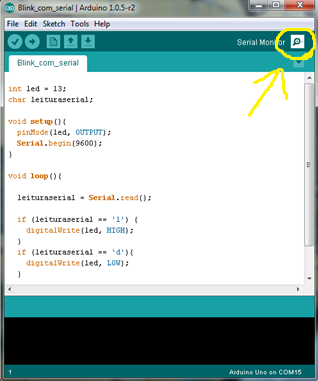
}

Se a variável x for maior que 10 o led irá ligar.

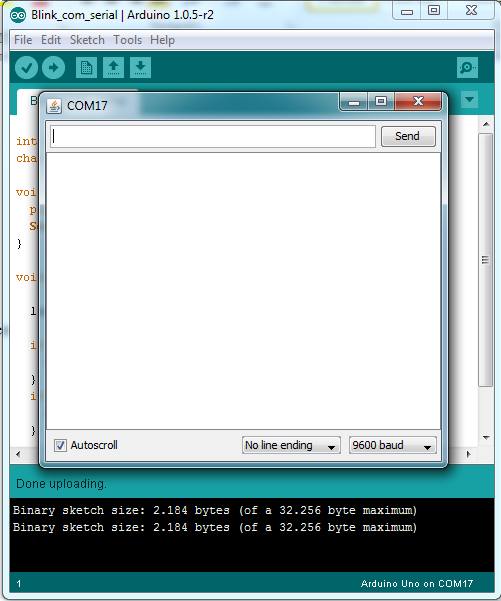
Continuando no “if (leituraserial == 'l')”, se (leituraserial for igual a ‘l’), ira executar uma função que estará dentro das chaves { }, no nosso programa irá ligar um led acoplado no arduino da porta 13.

Agora explicarei como testar seu código.

No canto superior direito a uma lupa com o nome de serial monitor, clique nele.

****

Irá aparecer desta forma:



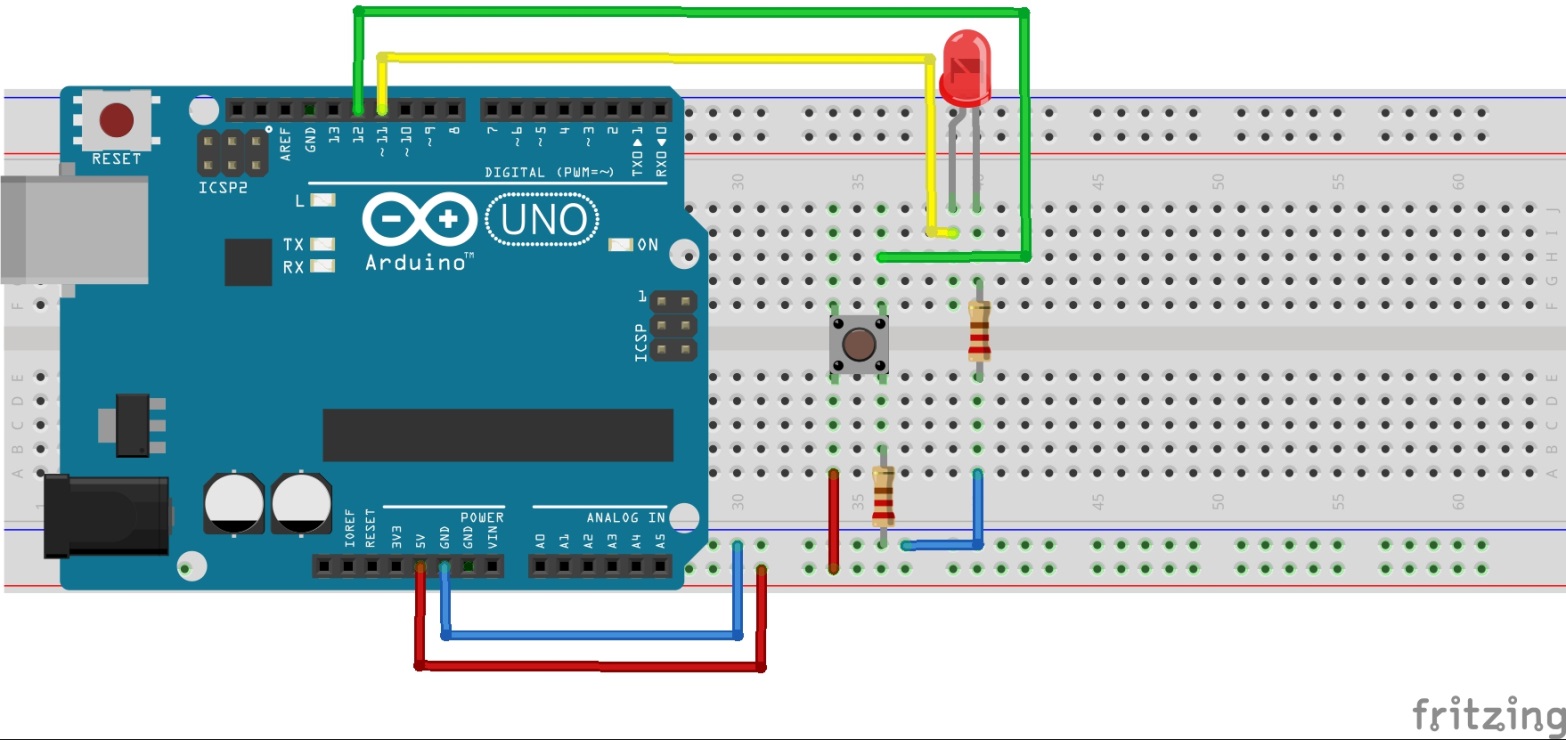
Na caixa de texto se escrever “l”, e apertar no enter do seu teclado do computador, irá acender o led acoplado no arduino, se apertar “d” e der um enter, irá desligar o led do arduino.

* **Módulo 3**

Componentes: Arduino + Cabo de comunicação + Led + Protoboard + Resistor de 150Ω + Jumper’s + Push Button

Descrição: Com o acionamento do Push Button o Led irá ligar ou desligar.

Dificuldade:



int led = 11;

int botao =12;

int estadobotao = 0;

void setup(){

pinMode(led, OUTPUT);

pinMode(led, INPUT);

}

void loop(){

estadobotao = digitalRead(botao);

if (estadobotao == HIGH) {

digitalWrite(led, HIGH);

}

if (estadobotao == LOW){

digitalWrite(led, LOW);

}

}

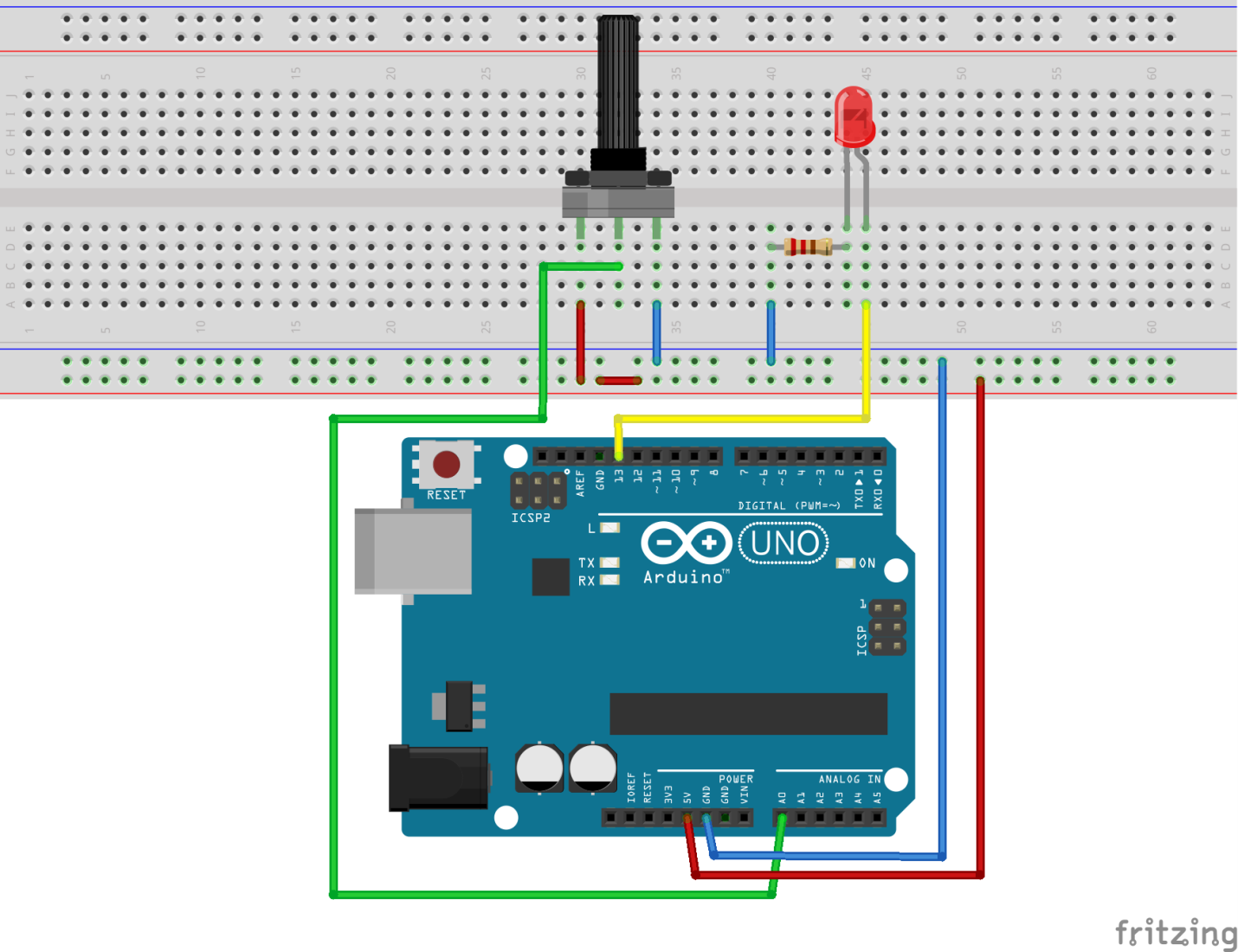
O que há de novo nesta programação é o comando “digitalRead”, que é a leitura da porta digital especificada nos parênteses, nesta programação está especificado o “(botão)”, assim ficando desta maneira “digitalRead(botão)”, leitura da porta digital onde o botão está especificado como pino 12.

* **Módulo 4**

Componentes: Arduino + Cabo de comunicação + Led + Resistor de 150Ω + Protoboard + Potenciômetro + Jumper’s

Descrição: Conforme variamos o Potenciômetro, o Led Liga e desliga em uma velocidade variada.

Dificuldade:



const int led = 13;

const int pinopotenciometro = 0;

int valorpotenciometro = 0;

void setup() {

pinMode(led, OUTPUT);

}

void loop() {

valorpotenciometro = analogRead(pinopotenciometro);

digitalWrite(led, HIGH);

delay(valorpotenciometro);

digitalWrite(led, LOW);

delay(valorpotenciometro);

}

Agora vou explicar os comandos desta programação:

Na primeira linha do código temos uma declaração de variável, “const int led = 13;” nada mais é que uma variável que permanecera sempre com o valor que o programador atribuiu, no nosso caso 13. Quando declarado uma variável “int”, ela irá armazenar valores diferentes até o seu limite de armazenamento, se necessário.

No “void setup()” identificamos o que é “OUTPUT” saida, “INPUT” entrada.

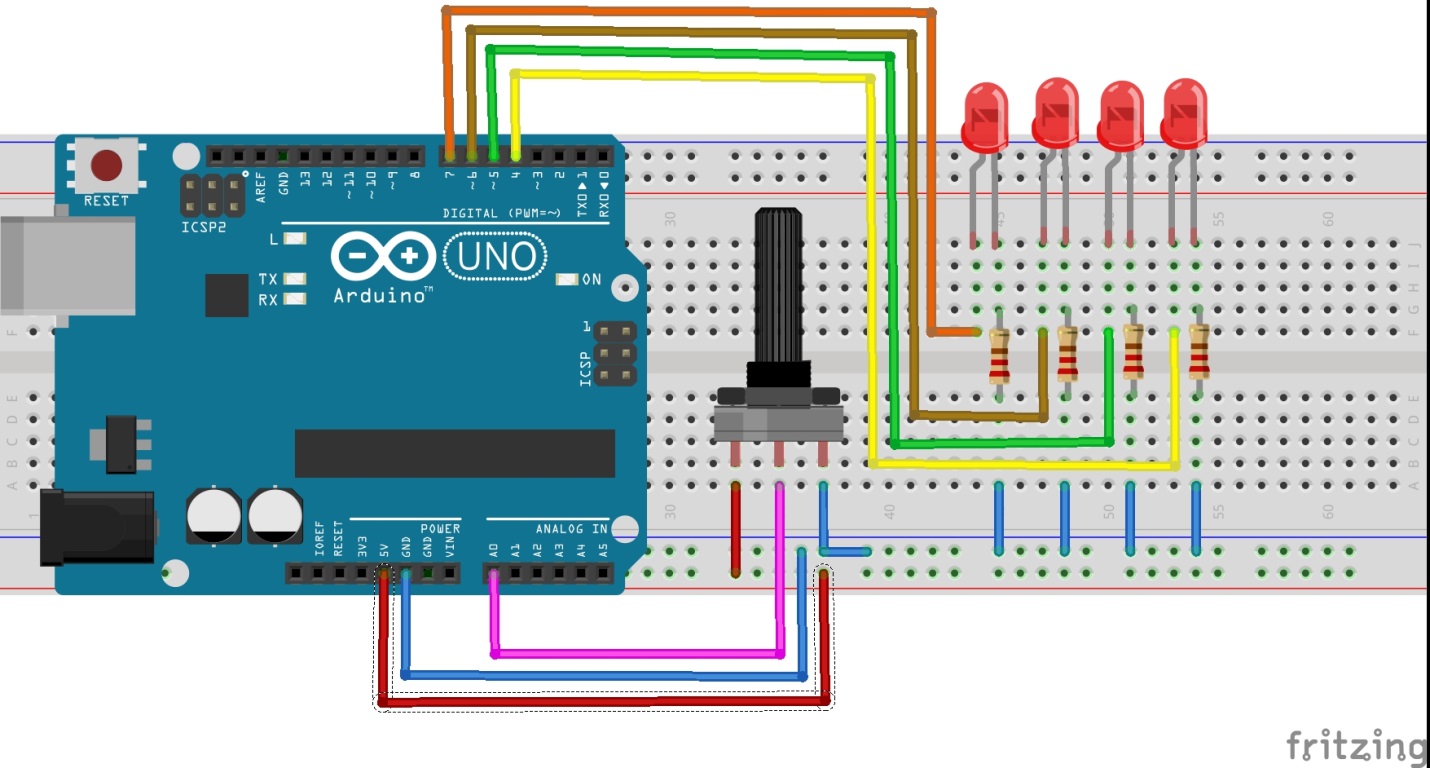
No “void loop() “ especificamos que a variável “valorpotenciometro” é igual a leitura do “pinopotenciometro”. Ligamos o led “digitalWrite(led, HIGH);” e com o potenciômetro variamos o tempo em *ms*  “delay(valorpotenciometro)”. Desligamos o led, e com o valor do potenciômetro ajustamos o tempo de desligamento. Concluindo, variamos o potenciômetro mecanicamente, assim ajustando o tempo que o led permanecerá ligado e desligado. Esse ajuste é o mesmo valor para o tempo que o led permanecera ligado ou desligado.

* **Módulo 5**

Componentes: Arduino + Cabo de comunicação + 4 Leds + 4 Resistores de 150Ω + Protoboard + Jumper’s + Potenciômetro

Descrição: Conforme variamos o potenciômetro, os leds ligam e desligam gradativamente.

Dificuldade:



const int led1 = 7;

const int led2 = 6;

const int led3 = 5;

const int led4 = 4;

const int pinopotenciometro = 0;

int valorpotenciometro = 0;

void setup() {

pinMode(led1, OUTPUT);

pinMode(led2, OUTPUT);

pinMode(led3, OUTPUT);

pinMode(led4, OUTPUT); }

void loop() {

valorpotenciometro = analogRead(pinopotenciometro);

if(valorpotenciometro > 0){

digitalWrite(led1, HIGH);

}else{

digitalWrite(led1, LOW);

}

if(valorpotenciometro > 255){

digitalWrite(led2, HIGH);

}else{

digitalWrite(led2, LOW);

}

if(valorpotenciometro > 510){

digitalWrite(led3, HIGH);

}else{

digitalWrite(led3, LOW);

}

if(valorpotenciometro > 765){

digitalWrite(led4, HIGH);

}else{

digitalWrite(led4, LOW);

}

}

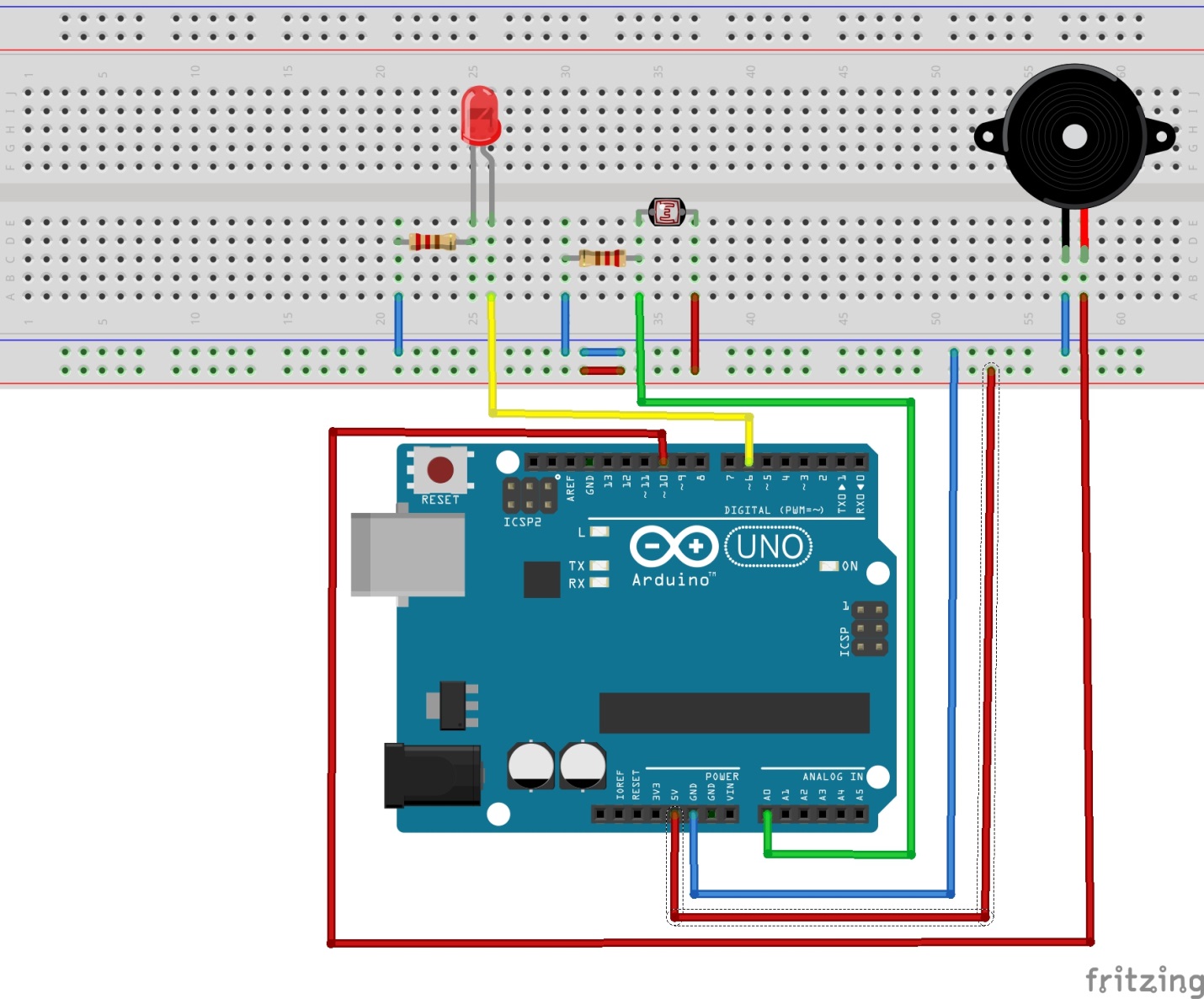
}

* **Módulo 6**

Componentes: Arduino + Cabo de comunicação + Led + Resistor de 150Ω + Protoboard + Resistor de 10KΩ + LDR + Jumper’s + Buzzer

Descrição: Conforme a claridade da sala, o LDR manda um sinal ao Arduino ligando ou desligando o Led. O Buzzer é acionado quando não tem claridade suficiente na sala.

Dificuldade:



const int LDR = 0;

const int led = 6;

const int buzzer = 5;

int valorlido = 0;

int pwm = 0;

void setup(){

pinMode(led, OUTPUT);

pinMode(buzzer, OUTPUT);

}

void loop(){

valorlido = analogRead(LDR);

if (valorlido < 100) {

analogWrite(led, pwm);

analogWrite(buzzer, pwm);

pwm++;

delay(100);

}

else{

digitalWrite(led, LOW);

digitalWrite(buzzer, LOW);

pwm = 0;

}

if(pwm > 255){

pwm = 0;

}

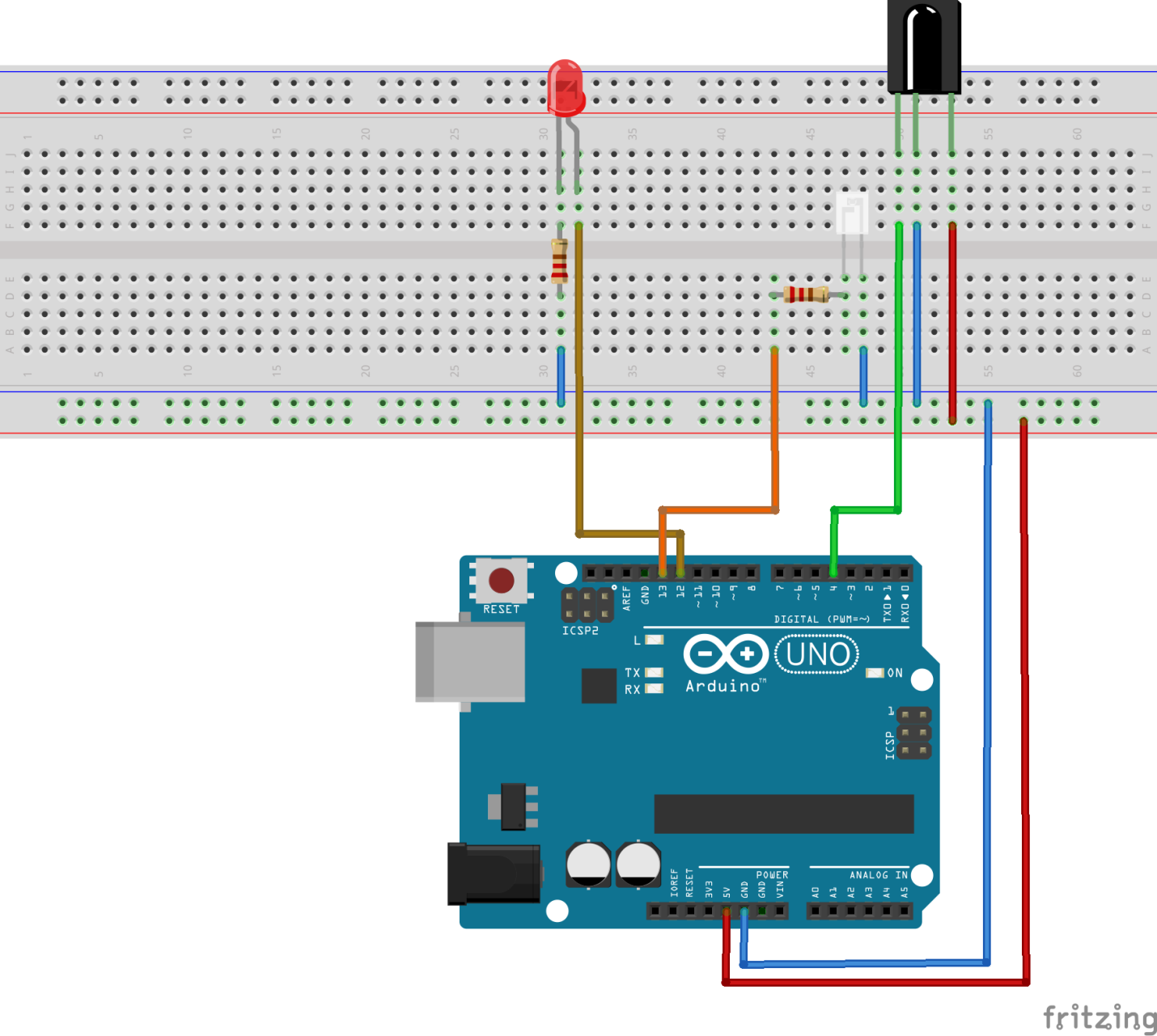
}

* **Módulo 7**

Componentes: Led + Emissor de Infravermelho + Receptor de Infravermelho + Resistor 150 Ω + Resistor 470 Ω + Jumper’s + Protoboard + Arduino + Cabo de comunicação

Descrição: Conforme o Infravermelho detecta o objeto o Led liga.

Dificuldade:



const int infravermelho = 13;

const int receptorInfravermelho = 4;

const int led = 11;

byte iv;

void setup() {

pinMode(infravermelho, OUTPUT);

pinMode(receptorInfravermelho, INPUT);

pinMode(led, OUTPUT);

}

void loop() {

for(byte x = 1; x <= 38; x++){

digitalWrite(infravermelho, HIGH);

delayMicroseconds(13);

digitalWrite(infravermelho, LOW);

delayMicroseconds(13);

}

iv = digitalRead(receptorInfravermelho);

if(iv == 0){

digitalWrite(led, LOW);

}else{

digitalWrite(led, HIGH);

}

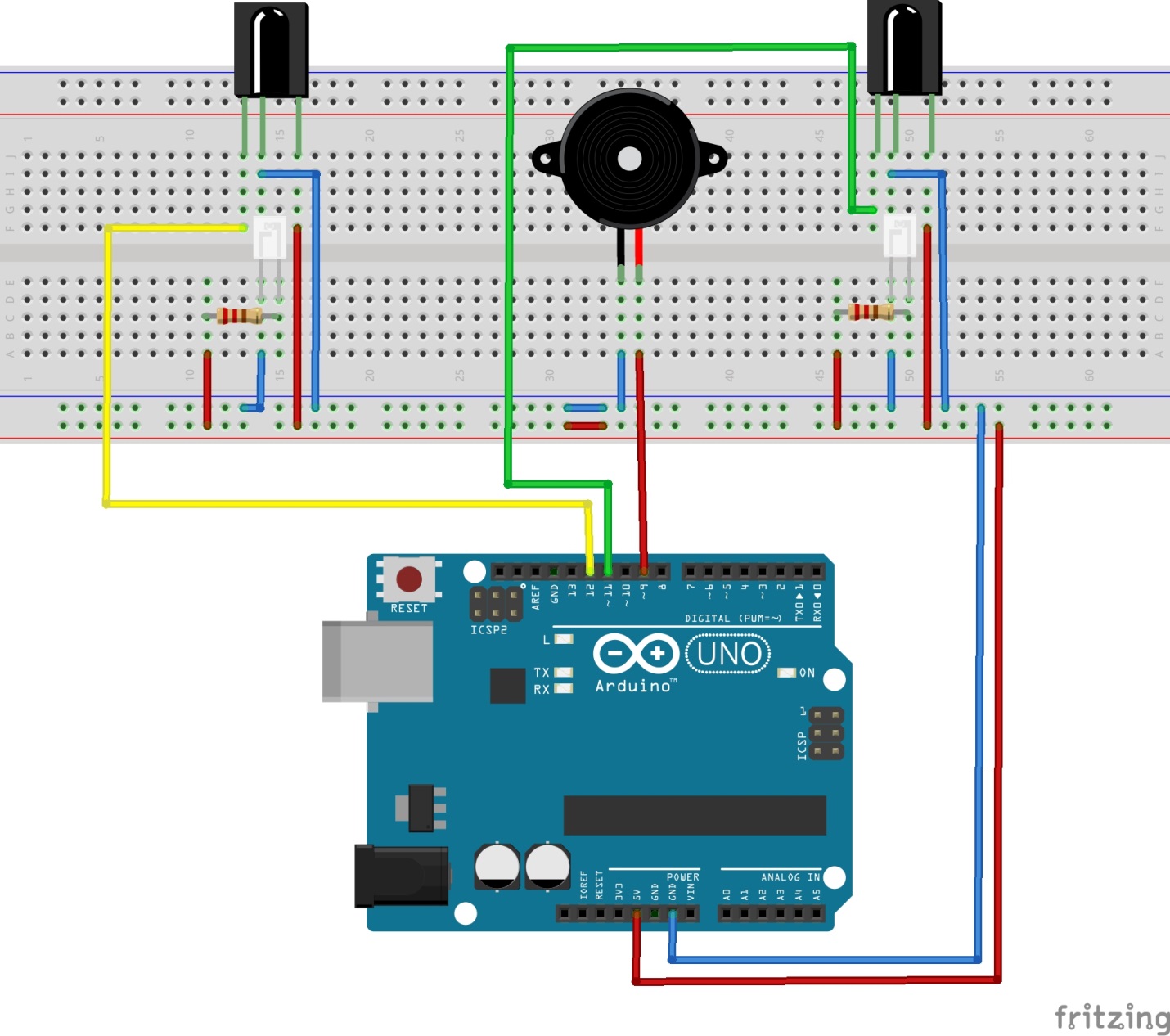
}

* **Módulo 8**

Componentes: 2 Emissor de Infravermelho + 2 Receptor de Infravermelho + Resistor Ω + Jumper’s + Protoboard + Arduino + Cabo de comunicação + Buzzer

Descrição: Conforme cada um dos 4 emissores detectam um objeto, o Buzzer é acionado. Para cada emissor IR, o Buzzer irá tocar uma nota diferente.

Dificuldade:



const int infravermelho0 = 2;

const int infravermelho1 = 4;

const int receptorInfravermelho0 = 13;

const int receptorInfravermelho1 = 12;

const int led = 9;

byte iv;

byte v;

void setup() {

pinMode(infravermelho0, OUTPUT);

pinMode(infravermelho1, OUTPUT);

pinMode(receptorInfravermelho0, INPUT);

pinMode(receptorInfravermelho1, INPUT);

pinMode(led, OUTPUT); }

void loop() {

for(byte x = 1; x <= 38; x++) {

digitalWrite(infravermelho0, HIGH);

digitalWrite(infravermelho1, HIGH);

delayMicroseconds(8.5);

digitalWrite(infravermelho0, LOW);

digitalWrite(infravermelho1, LOW);

delayMicroseconds(8.5); }

iv = digitalRead(receptorInfravermelho0);

v = digitalRead(receptorInfravermelho1);

if(iv == 0){

analogWrite(led, 0);

}else{

analogWrite(led, 255);

}

if(v == 0){

analogWrite(led, 0);

}else{

analogWrite(led, 255);

}

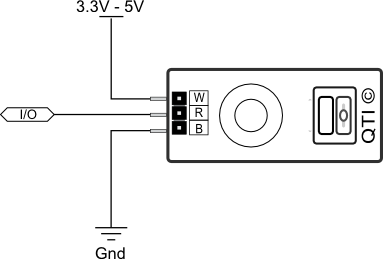
}

* **Módulo 9**

Componentes: Arduino + Cabo de comunicação + QTI + Jumper’s + Protoboard

Descrição: Conforme o QTI detecta a fita de alto contraste da mesa, o Monitor serial mostra o valor desta variação.

Dificuldade:



int SensorPin=2;

int Sensorval=LOW;

void setup() {

pinMode(2,OUTPUT);

Serial.begin(9600);

}

void loop() {

pinMode(2,OUTPUT); // Primeiro é necessário configurar o pino do Arduino (digital 2) que conecta o Vout do

digitalWrite(2,HIGH); //breakout como saída para descarregar o capacitor, colocando o pino digital 2 do Arduino

delayMicroseconds(10); // em alto (HIGH) e depois uma espera de 10 microssegundos

pinMode(2,INPUT); // E por fim configura o pino do Arduino (digital 2) como entrada

long time = micros();

// Enquanto o tempo for menor que 3000 microssegundos e o pino do sensor for alto (HIGH), então o valor do sensor

// será a diferença entre o tempo atual e o tempo anterior.

while (digitalRead(SensorPin) == HIGH && micros() - time < 3000);

int diff = micros() - time;

Sensorval=diff;

if(Serial.available()>0);

{

Serial.println(Sensorval);

}

delay(500);

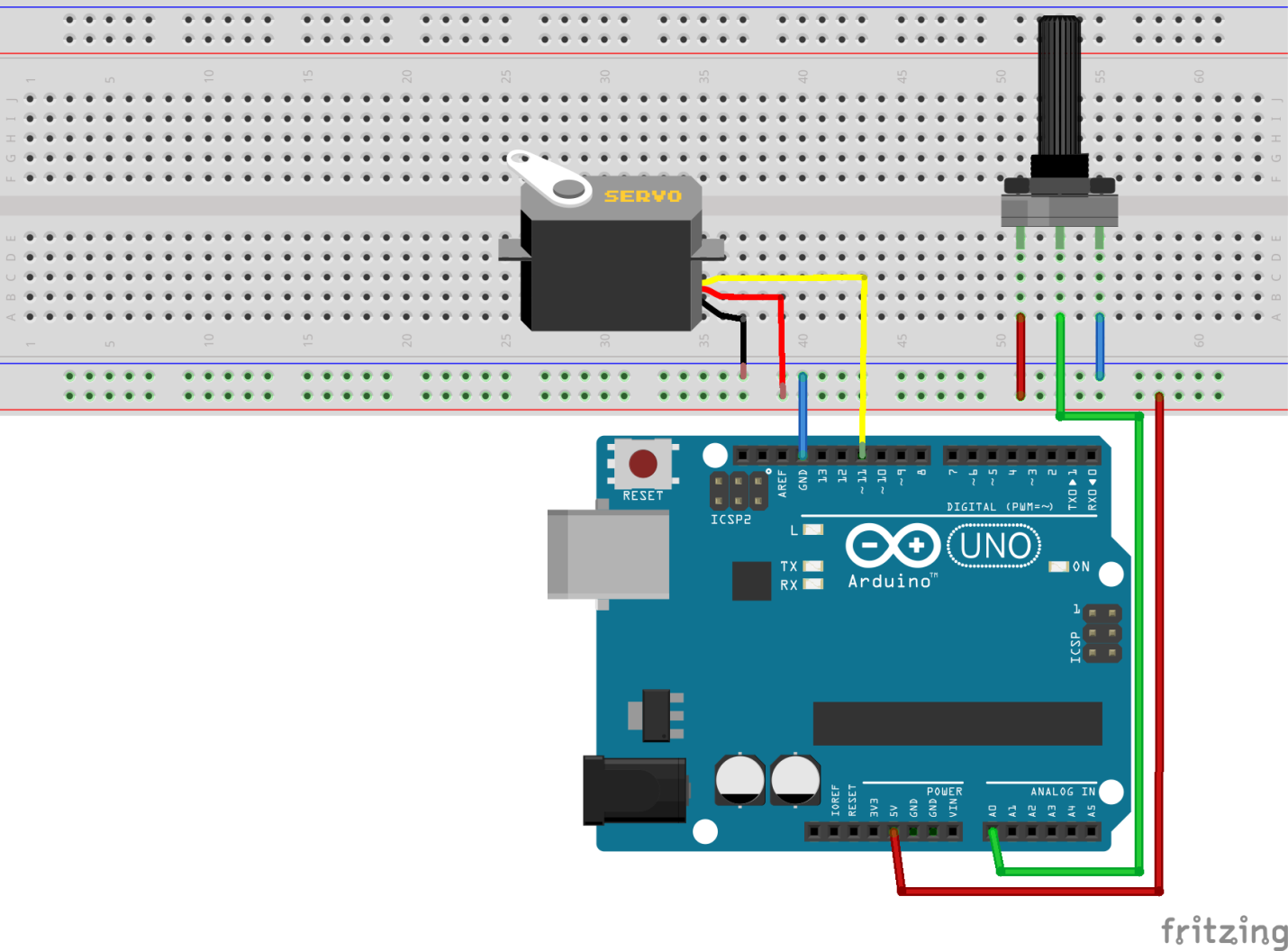
}

* **Módulo 10**

Componentes: Arduino + Cabo de comunicação + Potenciômetro + Servo motor + Jumper’s + Protoboard

Descrição: Ao variar o Potenciômetro o sentido de rotação do Servo motor oscila.

Dificuldade:



#include <Servo.h>

Servo servo;

const int potenciometro = 0;

int valorpotenciometro = 0;

void setup() {

servo.attach(11);

}

void loop() {

int valorpotenciometro=analogRead(potenciometro);

valorpotenciometro=map(valorpotenciometro, 0, 1023, 0, 180);

servo.write(valorpotenciometro);

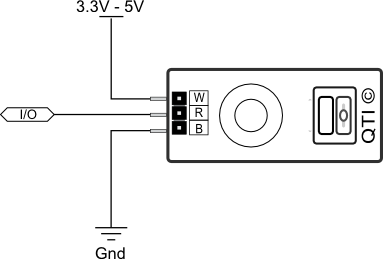
}

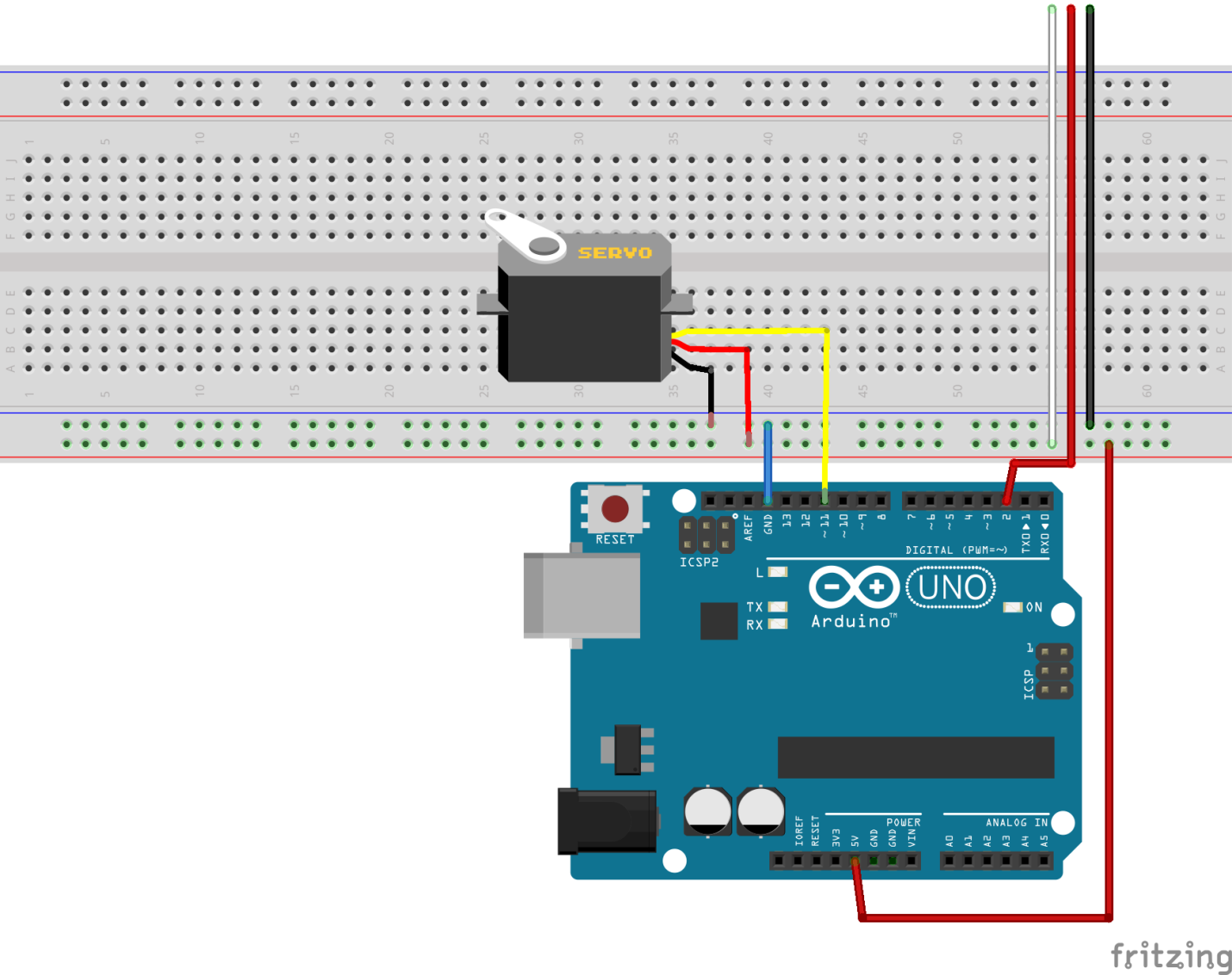
* **Módulo 11**

Componentes: Arduino + Cabo de comunicação + QTI + Jumper’s + Servo Motor + Protoboard

Descrição: Conforme o QTI detecta a fita de alto contraste da mesa, o Servo motor varia seu sentido de rotação.

Dificuldade:





#include<Servo.h>

#define servo 11

Servo s;

int SensorPin=2;

int Sensorval=LOW;

void setup() {

pinMode(2,OUTPUT);

s.attach(servo);

}

void loop() {

pinMode(2,OUTPUT);

digitalWrite(2,HIGH);

delayMicroseconds(10);

pinMode(2,INPUT);

long time = micros();

while (digitalRead(SensorPin) == HIGH && micros() - time < 3000);

int diff = micros() - time;

Sensorval=diff;

if(Serial.available()>0);

{

Sensorval=map(Sensorval, 0, 3000, 0, 180);

s.write(Sensorval);

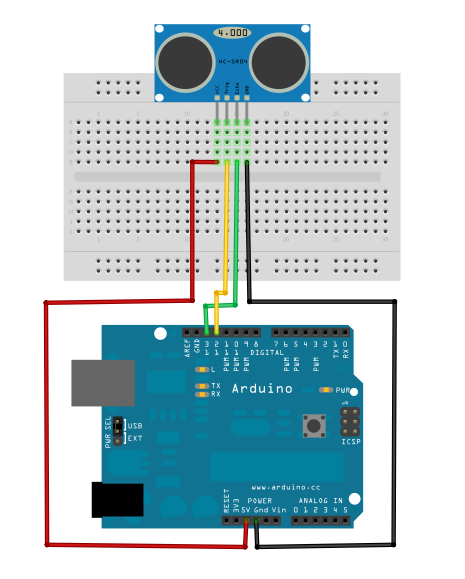
}

}

* **Módulo 12**

Componentes: Arduino + Cabo de comunicação + Ultra HC-SR04 + Jumpers + Protoboard

Descrição: Conforme o Ultra detecta um objeto na distancia X, o Monitor serial mostra o valor da distancia X.



#define trigPin 13

#define echoPin 12

void setup()

{

Serial.begin(9600);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

}

void loop()

{

float TemP, DiSt;

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

TemP = pulseIn(echoPin, HIGH);

DiSt = (TemP/2) /29.1;

Serial.print(DiSt);

Serial.print("Cm");

Serial.println();

delay(500);

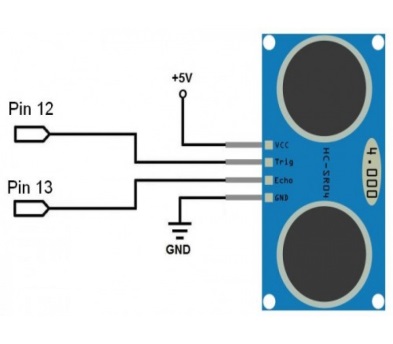
}

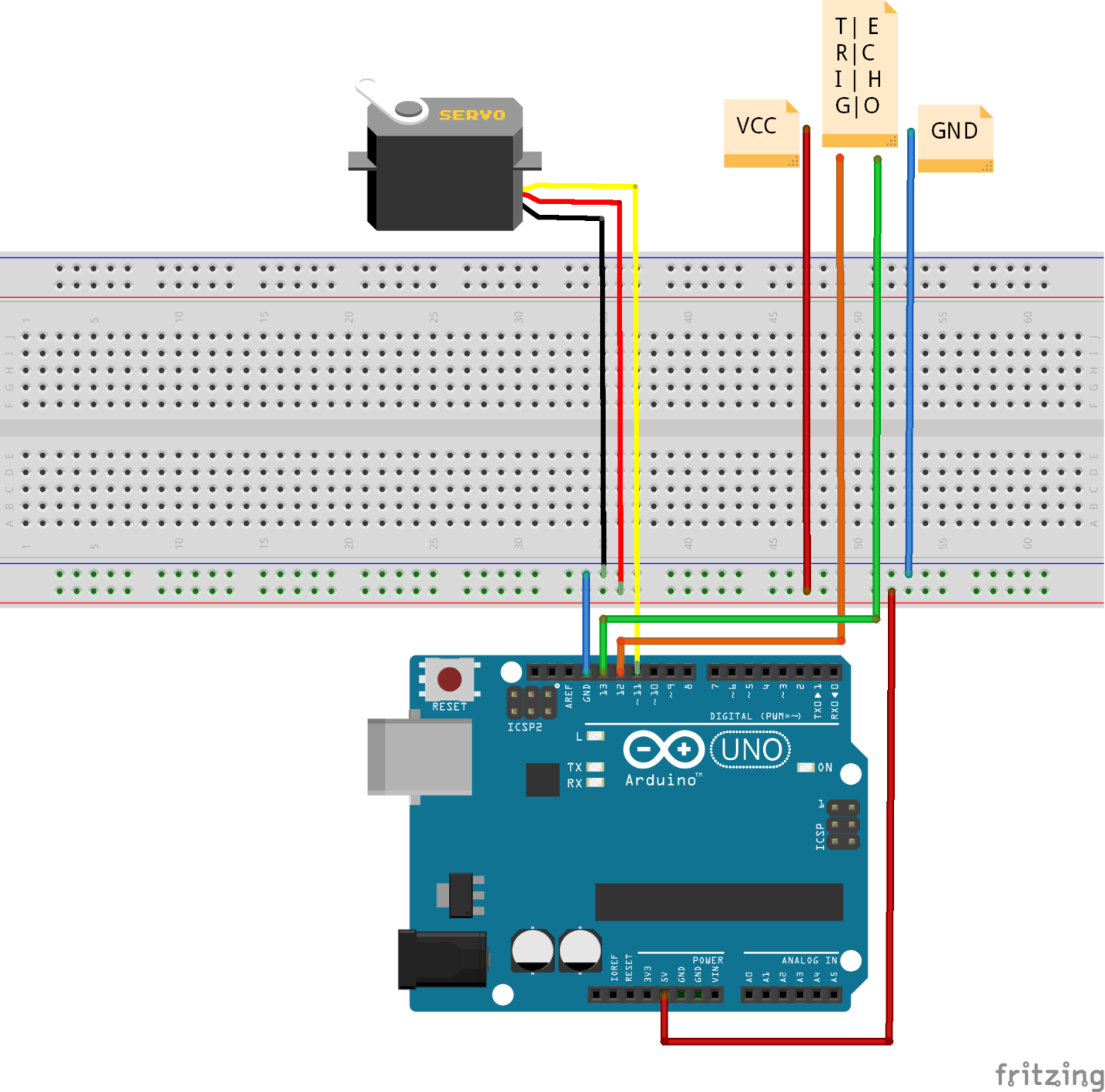
* **Módulo 13**

Componentes: Arduino + Cabo de comunicação + Ultra HC-SR04 + Jumper’s + Protoboard + Servo motor

Descrição: Conforme a distancia que o Ultra detecta o objeto, o Servo motor varia sua rotação.

Dificuldade:





#include<Servo.h>

#define SERVO 11

Servo s;

#define trigPin 13

#define echoPin 12

void setup()

{

Serial.begin(9600);

s.attach(SERVO);

s.write(0);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

}

void loop()

{

float TemP, DiSt;

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

TemP = pulseIn(echoPin, HIGH);

DiSt = (TemP/2) /29.1;

Serial.print(DiSt);

Serial.print("Cm");

Serial.println();

delay(500);

s.write(DiSt);

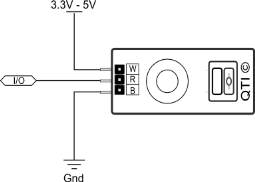
}

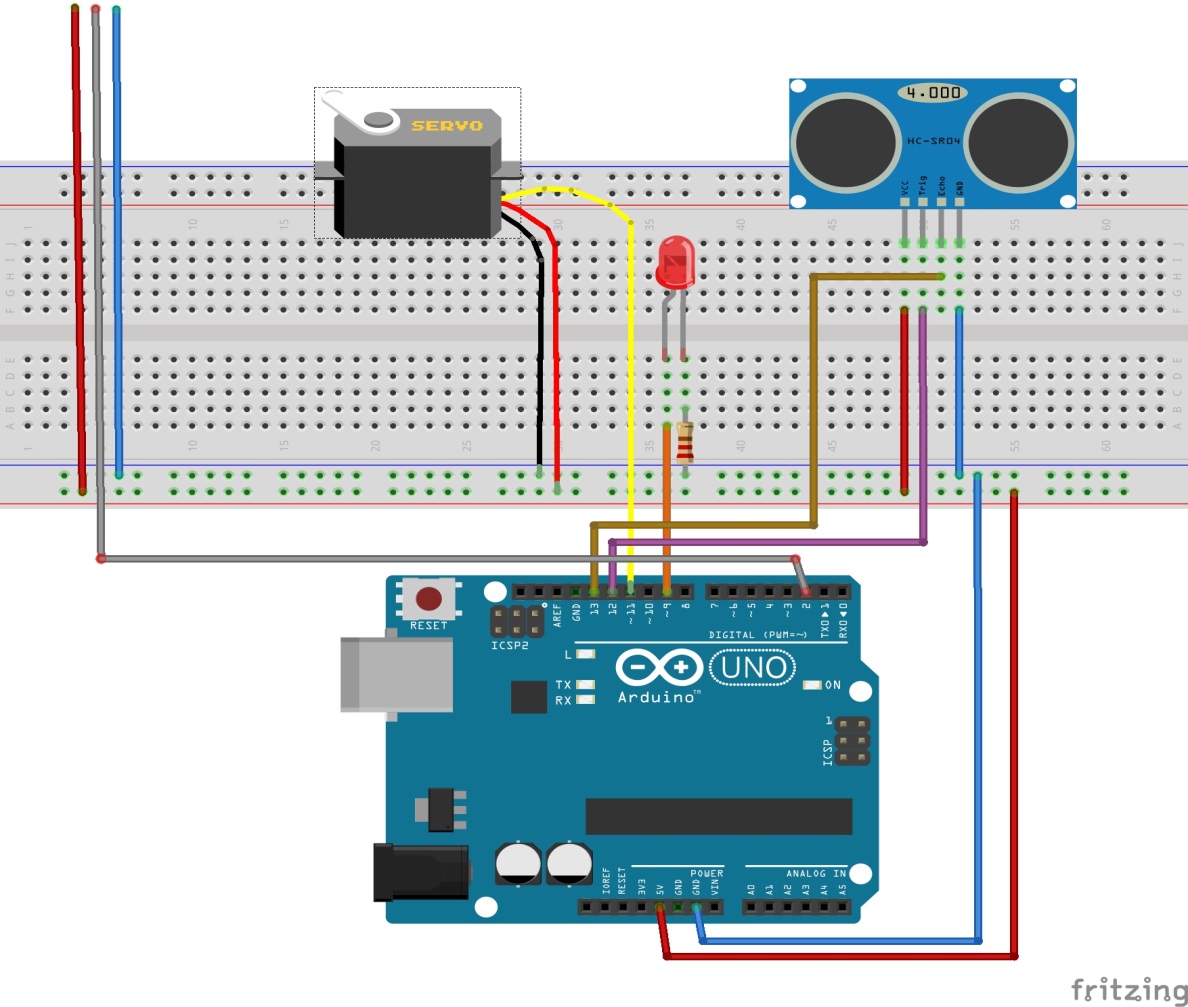
* **Módulo 14**

Componentes: Arduino + Cabo de comunicação + Ultra HC-SR04 + Jumper’s + Protoboard + QTI + Led + Resistor 2200Ω

Descrição: O QTI irá mandar um sinal ao Arduino se encontrar algum contraste na superfície plana, acionando o servo motor em sentindo horário. O HC-SR04, melhor dizendo o Ultrassónico, se detectar algum objeto em sua direção, irá anular o comando do QTI, fazendo o Servo motor parar acionando um led que irá ligar e desligar a cada 1 segundo, até o ultrassónico não identificar mais nenhum objeto.

Dificuldade:





#include<Servo.h>

#define servo 11

Servo s;

#define trigPin 13

#define echoPin 12

const int led = 9;

int SensorPin=2;

int Sensorval=LOW;

void setup()

{

s.attach(servo);

pinMode(led, OUTPUT);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(2,OUTPUT);

Serial.begin(9600);

}

void loop() {

pinMode(2,OUTPUT);

digitalWrite(2,HIGH);

delayMicroseconds(10);

pinMode(2,INPUT);

long time = micros();

while (digitalRead(SensorPin) == HIGH && micros() - time < 3000);

int diff = micros() - time;

Sensorval=diff;

if(Serial.available()>0);

{

Serial.println(Sensorval);

}

float TemP, DiSt;

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

s.write(93);

digitalWrite(led, HIGH);

delay(50);

digitalWrite(led, LOW);

delay(50);

}

Serial.println(Sensorval);

}

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

TemP = pulseIn(echoPin, HIGH);

DiSt = (TemP/2) /29.1;

if(Sensorval > 1000){

s.write(180);

}

if(DiSt < 80){

s.write(93);

digitalWrite(led, HIGH);

delay(50);

digitalWrite(led, LOW);

delay(50);

}

Serial.println(Sensorval);

}