



# Introdução à Internet das Coisas e à Robótica Educacional

Este trabalho foi idealizado e orientado pelo prof. Carlos Eduardo Dantas de Menezes, como um subproduto de oficinas realizadas na ***Faculdade de Tecnologia (Fatec) do Ipiranga***. Vários alunos estagiários participaram da pesquisa, escrita e elaboração dos experimentos aqui relatados. São eles: Danrley Dimas Santos, Guilherme Novais Longue, João Victor Venancio Lopes dos Reis, Johny Eskelsen Carebe, Marcelo Keiji Doi, Marcos Vinicius Vieira, Pedro Roberto Furuguem e Wellington Cruz.

## Sumário

<b>Arduino, ESP32, Raspberry, Jetson Nano... Tantas opções.... Quando usar um ou outro?</b>	<b>8</b>
<b>Modelos de Placas do Arduino</b>	<b>10</b>
<b>Estrutura de um programa em C para o Arduino</b>	<b>10</b>
<b>Guia de Componentes</b>	<b>11</b>
ARDUINO	11
RASPBERRY PI	11
PROTOBOARD	11
JUMPER	11
RESISTOR	12
CABO USB	12
BATERIA	12
CAPACITOR	12
LED	12
BOTÃO	13
MICRO SERVO MOTOR	13
POTENCIÔMETRO	13
DISPLAY LCD	13
SENSOR TOQUE	14
MOTOR DE PASSO	14
<b>Adicionando bibliotecas ao Arduino IDE</b>	<b>15</b>
<b>Pisca-pisca (Blink)</b>	<b>16</b>
Componentes Necessários	17
Montagem	17
Código Utilizado	18

Diagrama	18
<b>Projeto Pisca Ardublock</b>	<b>20</b>
Componentes necessários:	21
Diagrama dos Componentes	21
Montagem:	21
Código utilizado:	21
Explicação sobre o programa	22
<b>Semáforo Autônomo</b>	<b>23</b>
Componentes Necessários	23
Montagem	23
Código Utilizado	24
Codificação Via Ardublock	25
Diagrama	25
<b>Cancela com um servo-motor</b>	<b>26</b>
Componentes necessários	26
Montagem	27
Diagrama	27
Código utilizado	28
<b>Sensor de Temperatura e Umidade com I2C</b>	<b>29</b>
Componentes necessários	29
Montagem	30
Diagrama	30
Código utilizado	31
<b>Relógio Digital</b>	<b>34</b>
Componentes necessários	34
Montagem	34

Diagrama	35
Código utilizado	35
<b>Relógio Digital com RTC</b>	<b>38</b>
Componentes necessários	38
Montagem	38
Diagrama	39
Código utilizado	39
<b>Relógio Digital com botões e RTC</b>	<b>41</b>
Componentes necessários	41
Montagem	41
Diagrama	42
Código utilizado	42
<b>Projeto GPS em Arduino</b>	<b>45</b>
Componentes necessários	45
Montagem	45
Diagrama	46
Código utilizado	47
<b>Projeto Braço Mecânico controlado por Joystick</b>	<b>49</b>
Componentes necessários	49
Montagem	49
Diagrama elétrico	50
Diagrama mecânico	51
Código utilizado	51
<b>Cancela automática</b>	<b>54</b>
Componentes necessários	54
Montagem	54

Diagrama Elétrico	56
Código utilizado	56
<b>Jogo Genius</b>	<b>58</b>
Componentes necessários	59
Montagem	59
Diagrama Elétrico	60
Código utilizado	60
<b>Jogo do Botão</b>	<b>65</b>
Componentes necessários	66
Diagrama	66
Montagem	67
Código utilizado	69
<b>Theremin Ultrassônico</b>	<b>70</b>
Componentes necessários	70
Diagrama	70
Montagem	71
Código utilizado	73
<b>Controle de PlayStation 3 no Raspberry Pi 4</b>	<b>74</b>
Preparando o Raspberry	74
Mapeando o joystick	76
<b>Carrinho de controle remoto auto dirigível</b>	<b>81</b>
Componentes necessários	81
Montagem	82
Código utilizado	82
<b>Conexão com Arduino esp32 e Firebase</b>	<b>87</b>
Componentes necessários	88

Utilizando o Firebase	88
<b>1- Adicione um projeto</b>	<b>88</b>
<b>2- Coloque o nome do projeto</b>	<b>89</b>
Configurando a IDE do Arduino para o esp32	91
<b>1 - Baixando o drive da placa esp32</b>	<b>91</b>
<b>2 - Colocando a URL para gerenciar o esp 32</b>	<b>91</b>
<b>3 - Baixando a biblioteca do esp32</b>	<b>92</b>
<b>4 - Selecionando a placa esp32</b>	<b>93</b>
<b>5 - Colocando a biblioteca do firebase</b>	<b>94</b>
Código utilizado	94
Conexão com Arduino esp32 e MQTT	97
Componentes necessários	98
Gerenciando um broker	98
<b>1 - Baixando o drive da placa esp32</b>	<b>99</b>
Código utilizado	101
Integração do sensor de temperatura e umidade com a nuvem	103
Componentes necessários	103
Código utilizado	103
<b>Configurando a IDE do Arduino para o esp-01</b>	<b>114</b>
Código utilizado	117
Conexão com Arduino esp-01 e MQTT	120
Componentes necessários	121
Gerenciando um broker	121
Código utilizado	124
<b>ApplInventor - Primeiras experiências com Android</b>	<b>128</b>
Sensores de Humidade e Temperatura com ApplInventor	129

**Arduino, ESP32, Raspberry, Jetson Nano... Tantas  
opções.... Quando usar um ou outro?**

Muitas das aplicações que apresentaremos neste livro poderiam ser implementadas por uma placa da família Arduino, por uma ESP32, por uma Raspberry PI ou até por uma Jetson Nano, é verdade. Porém, há situações em que uma delas é mais recomendada que a outra:

- Se precisarmos de uma solução que gaste pouca energia, que já começa a trabalhar assim que energizada (com tempo de boot mínimo), que não precisa de conectividade com rede de dados ethernet ou wi-fi, que seja bem barata e que não exige um processamento tão complexo, uma placa Arduino poderá dar conta.
- Se precisarmos de conectividade de dados em wi-fi, bluetooth, ainda gastando pouca energia e com tempo de boot mínimo, poderemos optar por uma ESP32.
- Se a solução exige um processamento mais intenso, como o uso de algoritmos de Inteligência Artificial para classificação de imagens, uma placa Raspberry PI pode ser a melhor opção.
- Mas se o processamento de Inteligência Artificial roda lento na Raspberry PI, talvez porque processa muitas imagens, ou imagens com muita resolução, através de redes convolucionais, a placa Jetson Nano pode ser a solução, pois ela conta com 128 núcleos Maxwell de GPU, além de uma CPU de 4 núcleos com 64 bits (Arm A57) e 4 Gbytes de RAM DDR4, entregando um máximo de 472 GFLOPS (472 bilhões de operações de ponto flutuante por segundo).

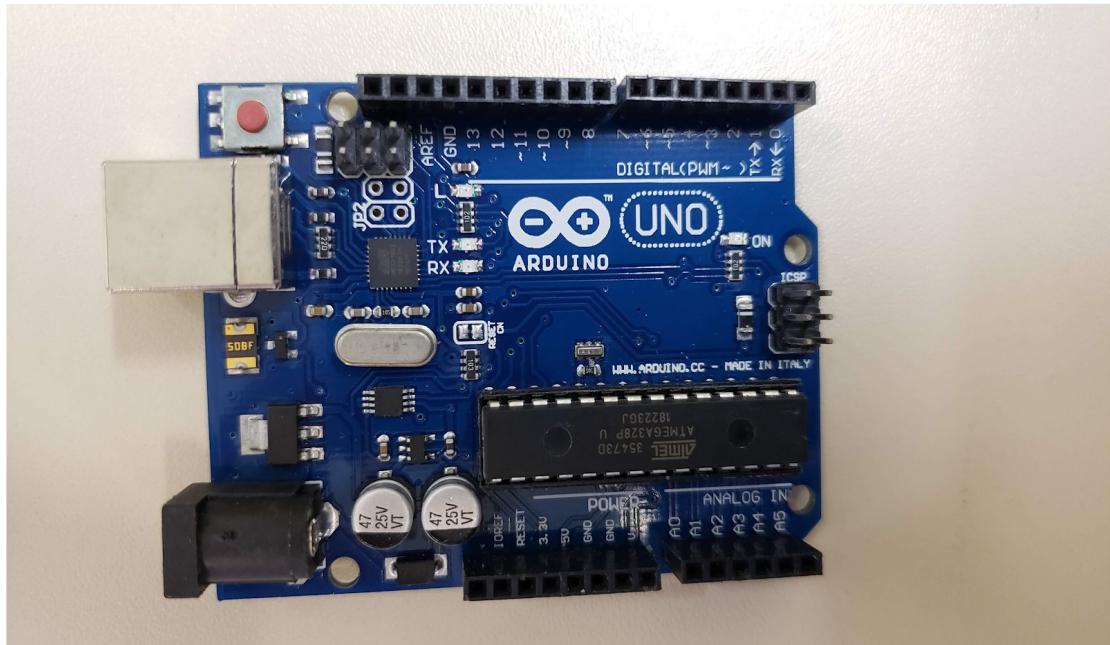
## Arduino

O Arduino é uma plataforma para criação de pequenos protótipos eletrônicos, criada em 2005 por um grupo de 5 pesquisadores: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis.

A placa eletrônica é barata, fácil de programar, e tem especificação de hardware e software completamente livre e aberta; é composta por um microcontrolador e uns poucos componentes eletrônicos. A programação pode ser feita em linguagem C em um ambiente integrado de desenvolvimento (Arduino IDE), que roda em qualquer computador, ou com a união de blocos que lembram um diagrama Nassi-Shneiderman (ArduBlockly).

Com esta placa podem ser construídos brinquedos, equipamentos e mecanismos de automatização.

**Figura:001- placa do Arduino UNO R3**



O ecossistema Arduino é composto por uma grande quantidade de sensores e atuadores, disponíveis como módulos, que são pequenas placas que contém os sensores e outros componentes auxiliares como resistores, capacitores, reguladores de tensão, transistores, mosfets, leds. Há também

placas de expansão chamadas *shields*, que aumentam os recursos da placa original, como o número de portas e a conectividade (bluetooth, rede ethernet, wi-fi).

## **Modelos de Placas do Arduino**

A placa básica ficou conhecida como Arduino UNO, possuindo 14 portas digitais e 6 analógicas; uma outra, com maior poder de processamento e uma quantidade bem maior de portas (54 portas digitais), é a placa Arduino Mega, usando o microcontrolador ATmega2560. A família conta com o Arduino Nano, cujo diferencial é o tamanho reduzido, o que pode ser interessante para a elaboração de um produto final.

## **Estrutura de um programa em C para o Arduino**

Todo código em C para o Arduino tem de ter 2 procedimentos no mínimo: `setup()` e `loop()`.

No primeiro procedimento, `setup()`, como o nome já dá a entender, ocorre toda a inicialização da placa, dos sensores e dos atuadores. Também, todo comando que deve ocorrer apenas uma única vez, logo ao energizar a placa, deve estar lá também.

O segundo procedimento, `loop()`, deverá conter tudo que será executado repetidamente, enquanto a placa estiver energizada.

## **Guia de Componentes**

### **ARDUINO**



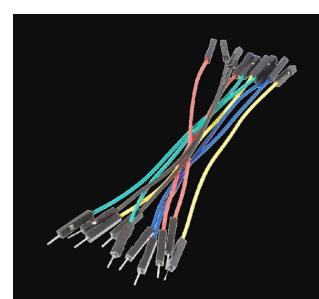
### **RASPBERRY PI**



### **PROTOBOARD**



### **JUMPER**



**RESISTOR**



**CABO USB**



**BATERIA**



**CAPACITOR**



**LED**



**BOTÃO**



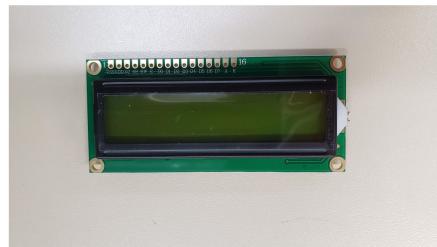
**MICRO SERVO MOTOR**



**POTENCIÔMETRO**



## DISPLAY LCD



## SENSOR TOQUE



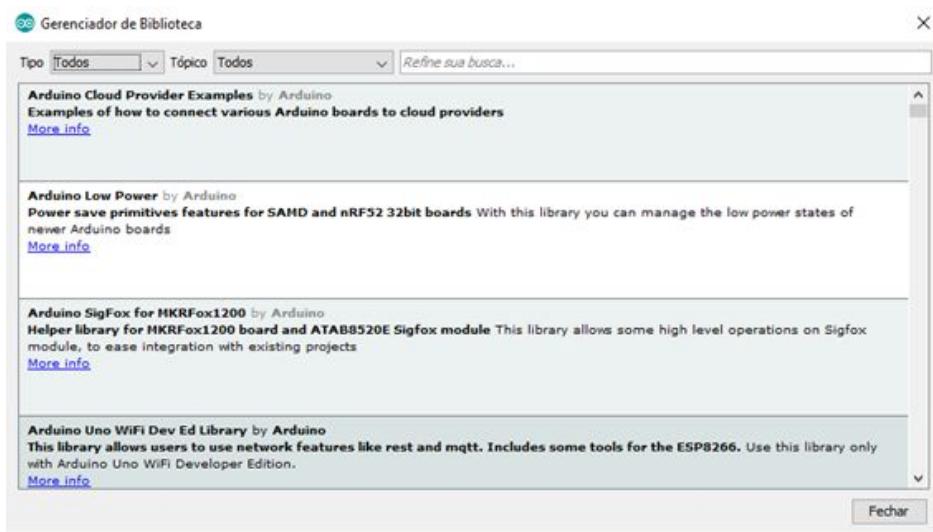
## MOTOR DE PASSO



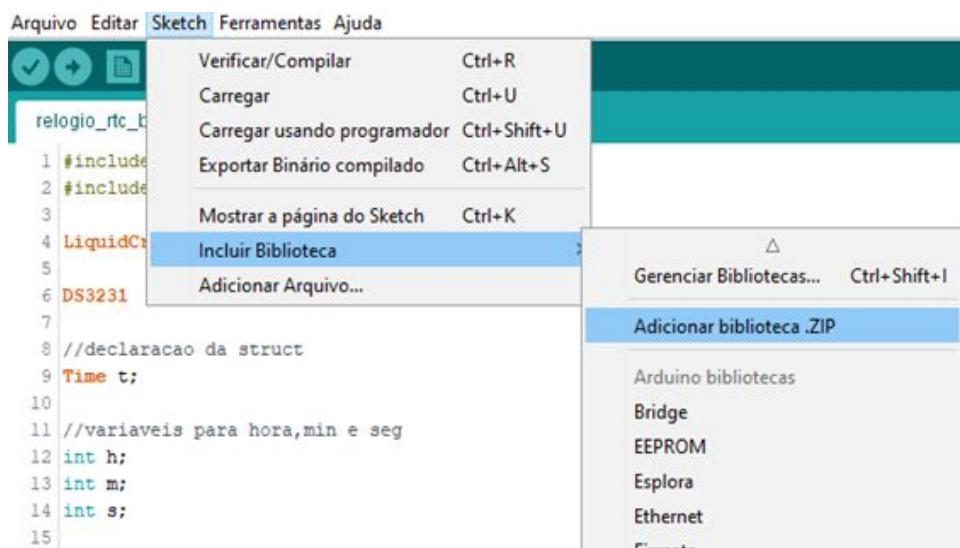
## Adicionando bibliotecas ao Arduino IDE

Para utilizar componentes como sensores ou display é necessário realizar o download de suas respectivas bibliotecas. Elas podem ser adicionadas de duas formas: pelo gerenciador de bibliotecas do Arduino IDE, ou adicionando um arquivo zip.

O gerenciador de bibliotecas pode ser encontrado em Ferramentas -> Gerenciar bibliotecas



Após encontrar uma biblioteca em um site, faça seu download. Ela virá em um arquivo compactado. No Arduino IDE selecione Sketch -> Incluir Biblioteca -> Adicionar Biblioteca .ZIP e então selecione o arquivo que você baixou. Após alguns segundos o Arduino IDE exibirá uma mensagem no canto inferior esquerdo confirmando a adição da biblioteca



# Projetos com Arduino

## Pisca-pisca (Blink)

**Descrição:** Uma experiência simples para piscar um LED (um diodo emissor de luz).

Neste projeto, iremos programar uma alternância repetitiva de dois estados de luminância de um LED: aceso por um segundo e apagado por mais um segundo, fazendo-o piscar. Descreveremos o comando de um LED externo à placa Arduino. Além disso, mostraremos como fazer o mesmo pisca-pisca com o LED que se encontra embutido na própria placa Arduino.

### Componentes Necessários

1 Unidade - Placa Arduino Uno R3

1 Unidade - Protoboard

1 Unidade - LED

1 Unidade - Resistor (você poderá usar um resistor com valor entre 330 a 1K ohms)

Conjunto de Cabos Jumper do tipo macho - macho

### Montagem

**1)** Conecte um cabo USB do Arduino para o computador.

**2)** Conecte o LED na Protoboard. As suas “perninhas” devem estar ligadas em linhas diferentes. Caso tenha dúvida na montagem, verifique o Guia de Componentes.

**3)** Conecte um cabo Jumper, uma ponta em uma entrada GND do Arduino e a outra ponta em um ponto negativo da parte inferior da Protoboard (marcada como “-”).

**4)** Conecte um resistor na Protoboard, conectando uma ponta na mesma coluna da “perninha” menor do LED, e a outra ponta na parte negativa da base lateral da Protoboard.

**5)** Conecte um cabo Jumper, uma ponta em uma das portas do Arduino e a outra na mesma linha da “perninha” maior do LED.

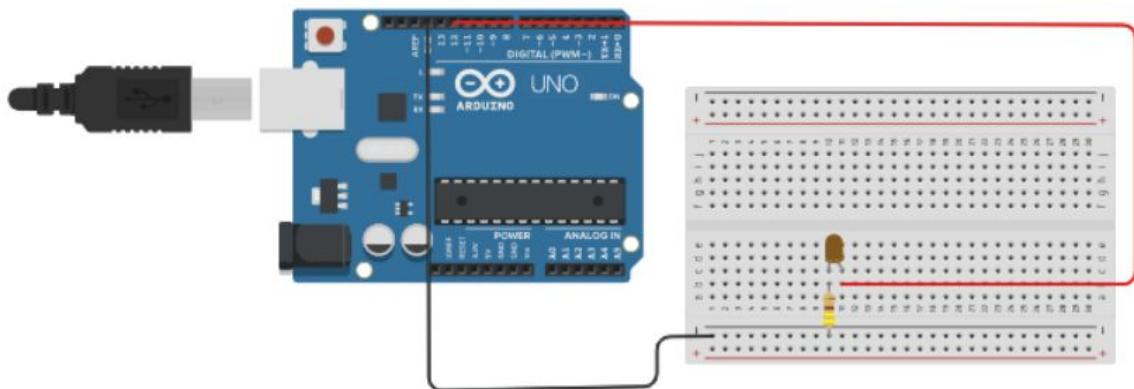
**6)** Abra o Arduino IDE e coloque o código abaixo. Verifique se você ligou a porta cujo número é citado no código. Caso contrário, você deverá corrigir isso no código ou na montagem.

**7)** Para fazer piscar o LED embutido da placa Arduino, simplesmente substitua no código o número da porta de 12 para 13. A grande maioria das placas Arduino usam essa porta para ativar seu LED interno. Neste caso você não precisará de nada além da própria placa e cabo USB para ligação com o computador.

## Código Utilizado

```
• int led = 12;
• void setup() {
•     pinMode(led,OUTPUT);
• }
•
•
• void loop() {
•     digitalWrite(led,LOW);
•     delay(1000);
•     digitalWrite(led,HIGH);
•     delay(1000);
• }
```

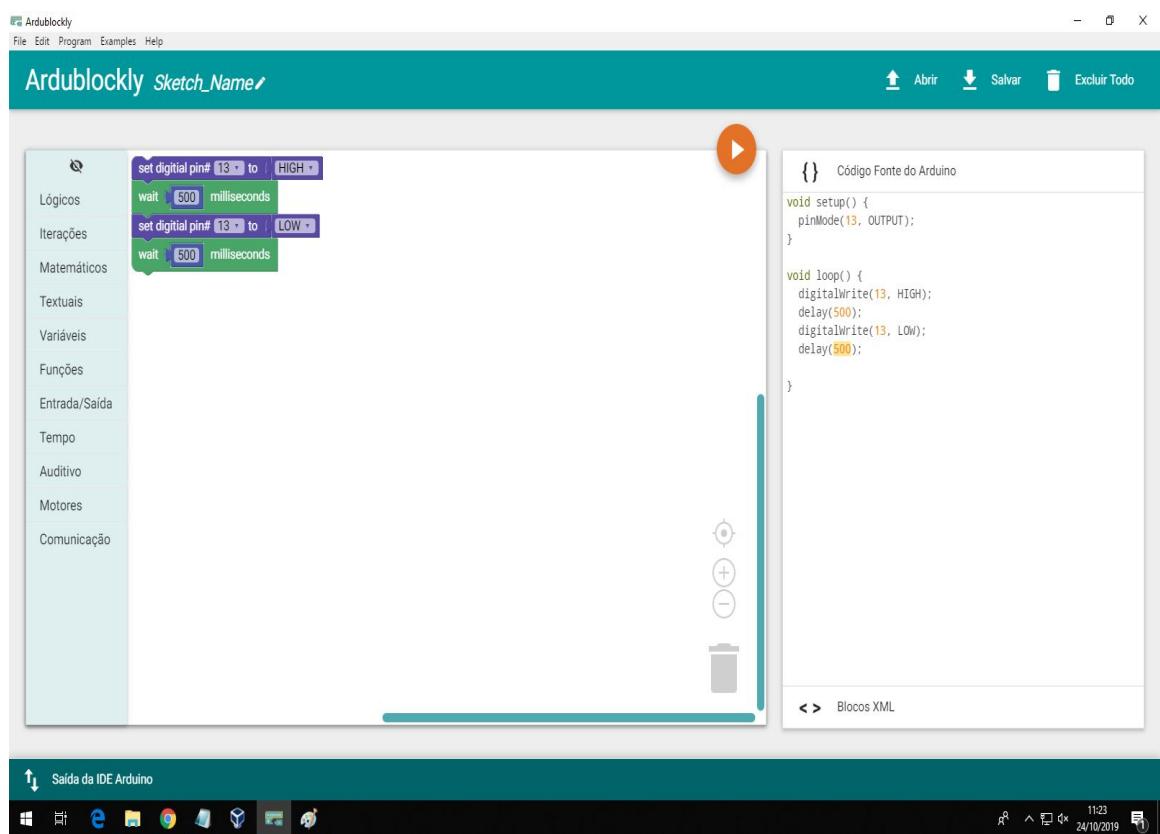
## Diagrama



## Projeto “Blink” Ardublock

**Descrição:** O projeto arduino vai mostrar como desenvolver, passo a passo, da montagem dos componentes até a programação, o projeto arduino que faz um led piscar em um intervalo de tempo definido por você.

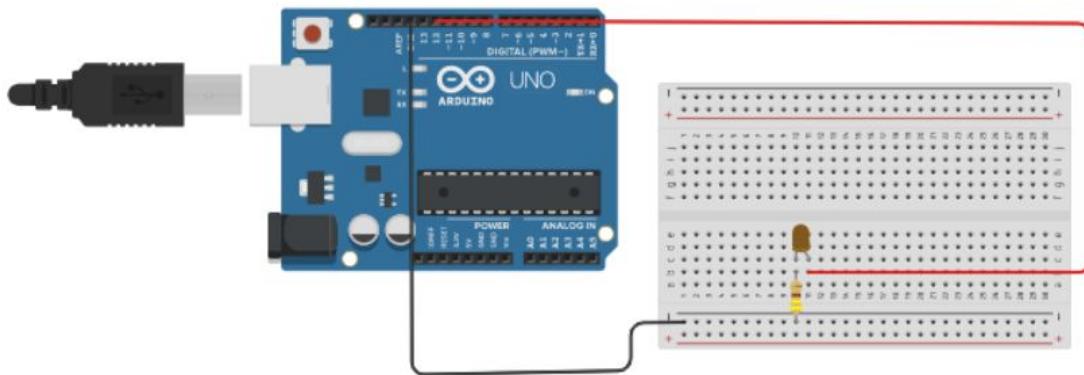
Este é um projeto introdutório ao mundo arduino, e te dará uma boa idéia do funcionamento deste microcontrolador, que também é chamado de plataforma de desenvolvimento. Também mostrará a interação do arduino, através de suas portas digitais, com componentes eletrônicos externos, como o led e o resistor. Usaremos uma protoboard como base para a ligação entre arduino e componentes eletrônicos.



## Componentes necessários:

- 1 Arduino;
- 1 cabo conector para Arduino USB;
- 1 led fica no pino 13 do Arduino;

## Diagrama



## Montagem:

- 1º conecte a placa Arduino com cabo.
- 2º conecte ao computador e ligue o resistor no pino digital 13 do arduino.

## Código utilizado:

```
• void setup() {  
•     pinMode(13,OUTPUT);  
• }  
•  
• void loop() {  
•     digitalWrite(13,HIGH);  
•     delay(1000);  
•     digitalWrite(13,LOW);  
•     delay(1000);  
• }
```

## Explicação sobre o programa

void setup( ) é um método e é executado uma vez assim que o arduino é ligado.

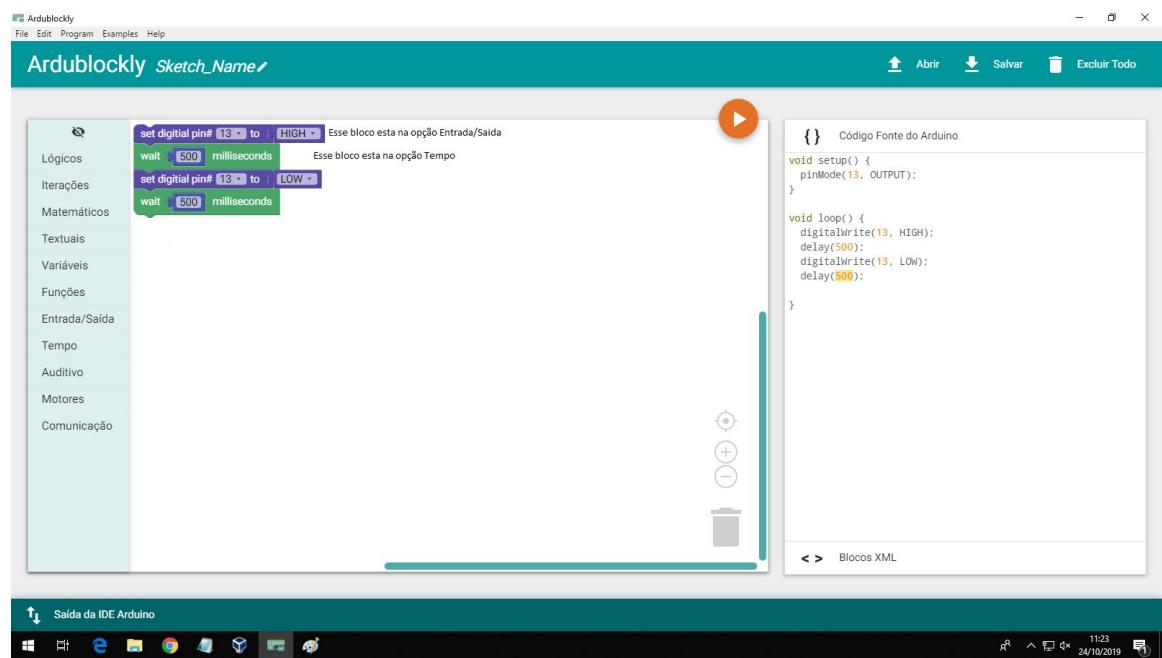
void loop( ) também é um método que é executado, como diz o próprio nome, em loop enquanto o arduino estiver ligado.

O comando pinMode(13, OUTPUT) define o pino digital 13 do arduino como um pino de saída. É o pino em que o led está ligado.

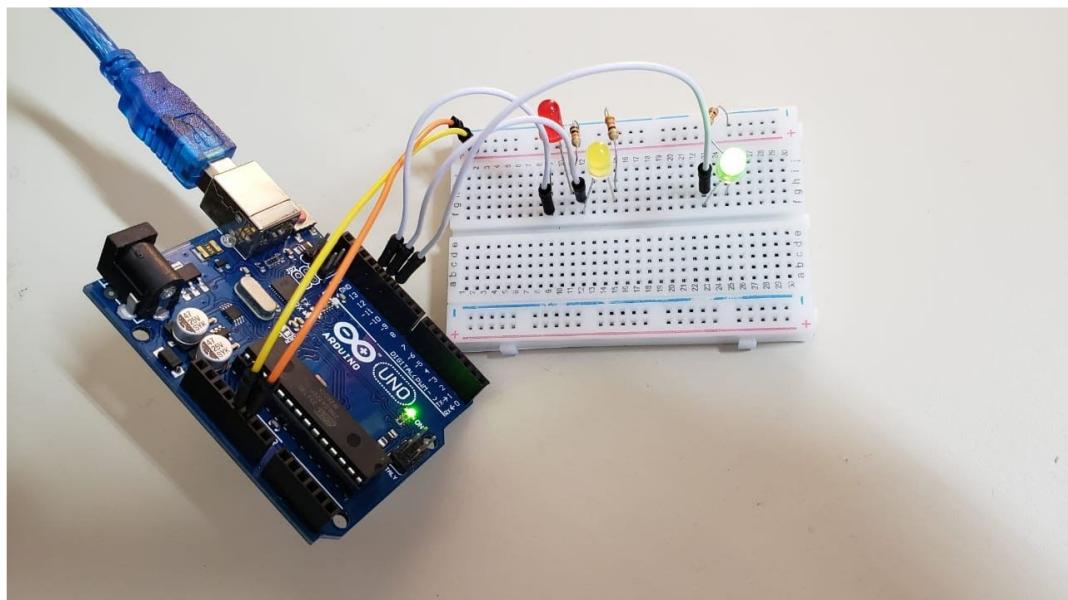
O comando digitalWrite(13,HIGH) liga o led.

O comando digitalWrite(13,LOW) desliga o led.

O comando delay(1000) faz o arduino esperar 1 segundo antes de executar o próximo comando. Para fazer o arduino esperar 2 segundos use delay(2000), para fazer esperar 5 segundos use delay(5000), para fazer esperar 1 minuto use delay(60000) , e assim por diante.



## Semáforo Autônomo



**Descrição:** Um sistema que simula o funcionamento de um semáforo, ao alternar a luminosidade de LEDs das cores verde, amarelo e vermelho.

Neste projeto, iremos adaptar o projeto Pisca-pisca mostrado anteriormente simulando um semáforo, onde as cores serão alternadas via código.

### Componentes Necessários

1 Unidade – Arduino Uno R3

1 Unidade – Protoboard

3 Unidades – Resistor (você poderá usar resistores com valor entre 330 a 1K ohms)

3 Unidades - LEDs (1 verde, 1 amarelo, 1 vermelho)

1 Unidade - Cabo USB

Conjunto de Cabos Jumper do tipo macho - macho

### Montagem

**1)** Conecte um cabo USB do Arduino para o computador

- 2)** Conecte 3(três) LEDs na Protoboard.
- 3)** Conecte 3(três) cabos Jumpers, para cada um, uma ponta deve ser ligada em uma entrada do Arduino, e a outra deve ser ligada na mesma linha da “perninha” maior de cada LED
- 4)** Conecte 3(três) resistores na Protoboard, para cada um, conecte uma ponta na mesma coluna da “perninha” menor dos LED’s, e a outra ponta na parte negativa da base lateral da Protoboard.
- 5)** Abra o Arduino IDE e coloque o código abaixo. Verifique se os cabos estão conectados nas mesmas portas, caso contrário você deverá alterá-las durante a programação

### Código Utilizado

```
•   int ledVermelho = 12;
•   int ledAmarelo = 11;
•   int ledVerde = 10;
•
•
•   void setup() {
•       pinMode(ledVermelho,OUTPUT);
•       pinMode(ledAmarelo,OUTPUT);
•       pinMode(ledVerde,OUTPUT);
•   }
•
•
•   void loop() {
•       digitalWrite(ledVermelho,HIGH);
•       digitalWrite(ledAmarelo,LOW);
•       digitalWrite(ledVerde,LOW);
•       delay(3000);
•       digitalWrite(ledVermelho,LOW);
•       digitalWrite(ledAmarelo,HIGH);
•       digitalWrite(ledVerde,LOW);
•       delay(2000);
•       digitalWrite(ledVermelho,LOW);
•       digitalWrite(ledAmarelo,LOW);
•       digitalWrite(ledVerde,HIGH);
•       delay(2000);
•   }
```

## Codificação Via Ardublock

The screenshot shows the Ardublock software interface. On the left, there's a sidebar with categories: Logic, Loops, Math, Text, Variables, Functions, Input/Output, Time, Audio, Motors, and Comms. The main workspace contains two blocks under the "Arduino loop forever:" category:

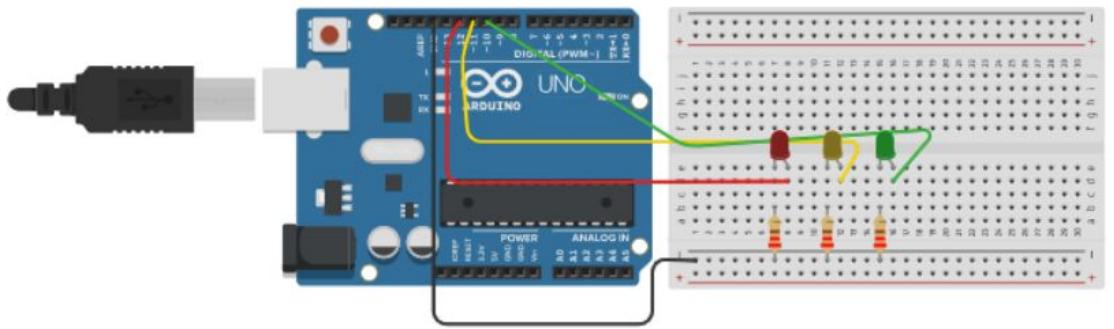
- set digital pin# 12 to HIGH
- set digital pin# 11 to LOW
- set digital pin# 10 to LOW
- wait 4000 milliseconds
- set digital pin# 12 to LOW
- set digital pin# 11 to HIGH
- set digital pin# 10 to LOW
- wait 3000 milliseconds
- set digital pin# 12 to LOW
- set digital pin# 11 to LOW
- set digital pin# 10 to HIGH
- wait 5000 milliseconds

On the right, the "Arduino Source Code" pane displays the generated C++ code:

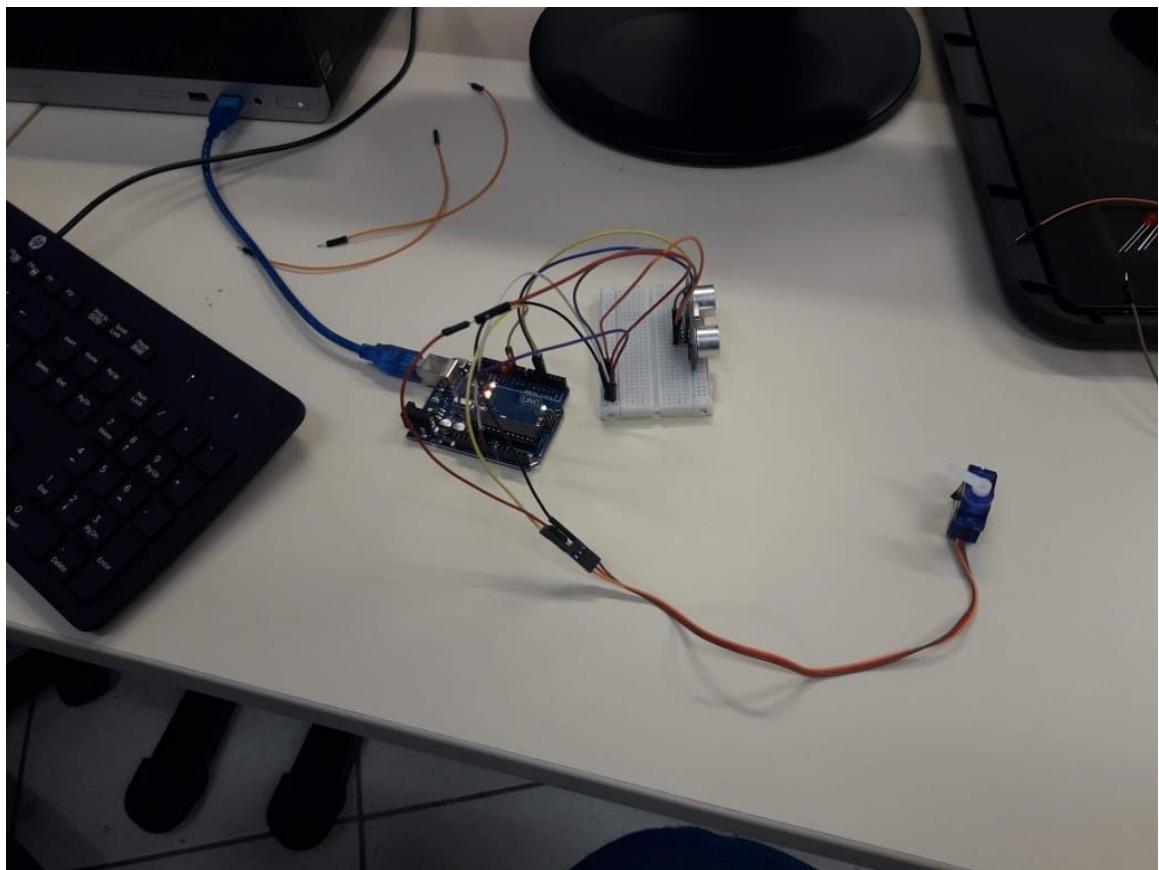
```
void setup() {  
    pinMode(12, OUTPUT);  
    pinMode(11, OUTPUT);  
    pinMode(10, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(12, HIGH);  
    digitalWrite(11, LOW);  
    digitalWrite(10, LOW);  
    delay(4000);  
    digitalWrite(12, LOW);  
    digitalWrite(11, HIGH);  
    digitalWrite(10, LOW);  
    delay(3000);  
    digitalWrite(12, LOW);  
    digitalWrite(11, LOW);  
    digitalWrite(10, HIGH);  
    delay(5000);  
}
```

Below the source code is a "Blocks XML" button.

## Diagrama



## **Cancela com um servo-motor**



**Descrição:** Um servo-motor é usado para simular a abertura e fechamento de uma cancela, como em um estacionamento.

Um servo-motor não funciona como um motor de corrente contínua comum, o qual recebe tensão e gira sem parar; o servo-motor, além da alimentação, recebe um comando, que consiste de um pulso: a duração deste pulso comanda a posição do eixo do servo, porém, sem muita precisão. A maioria dos servos tem uma faixa de variação do ângulo de seu eixo de 180°, e uns poucos chegam a variar 360°.

### **Componentes necessários**

1 Unidade – Arduino Uno;

1 Unidade – Micro Servo Motor Tower Pro 9g Sg90 ou equivalente.

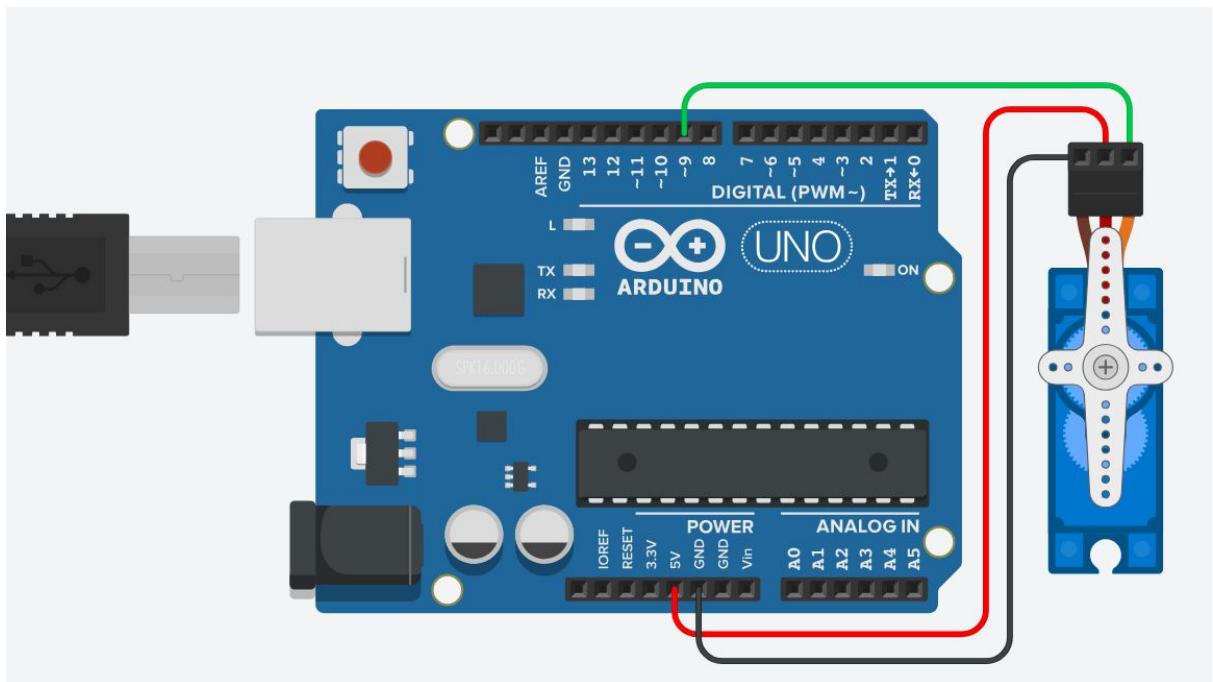
1 Unidade – Protoboard;

Conjunto de cabos Jumper macho e fêmea.

## Montagem

- 1)** Conecte o Arduino na Protoboard
- 2)** Conecte o servo-motor no Arduino
- 3)** Conecte o Arduino ao cabo USB e este ao computador
- 4)** Abra o Arduino IDE e coloque o código abaixo. Verifique se os cabos estão conectados nas mesmas portas, caso contrário você deverá alterá-las durante a programação

## Diagrama



## Código utilizado

```
● #include <Servo.h>
●
● Servo meuServo; // cria um objeto servo para controlar o dispositivo
●
● int posicao = 0; // variável para armazenar a posição do servo
●
● void setup() {
●     meuServo.attach(9); // ligo o servo no pino 9 da placa
● }
●
● void loop() {
●     for (posicao = 0; posicao <= 90; posicao++) { // vai de 0 a 90°
●         meuServo.write(posicao); // diz ao servo para ir para o ângulo
●         'posicao'
●         delay(15); // espera 15ms para o servo chegar à posição
●     }
●
●     delay(2000); // espera 2s com a cancela aberta
●     for (posicao = 90; posicao >= 0; posicao--) { // vai de 90 a 0°
●         meuServo.write(posicao); // diz ao servo para ir para o ângulo
●         'posicao'
●         delay(15); // espera 15ms para o servo chegar à posição
●     }
● }
```

## Sensor de Temperatura e Umidade com I2C



**Descrição:** Um sensor que verifica a temperatura e umidade e as pelo Serial do Arduino IDE e em um display de LCD usando o protocolo I2C.

I2C (Inter-Integrated Circuit) é um protocolo de comunicação serial utilizado para permitir a comunicação entre periféricos de baixa velocidade a um microcontrolador ou placa-mãe, via um barramento. Foi criado pela Philips na década de 1990, e tem uma característica interessante de permitir a comunicação com poucos fios: basicamente 2, o SDA (Serial Data), o SCL (Serial Clock) e a alimentação (VDD, normalmente 3,3V ou 5V) e GND (terra). Na prática, até 112 dispositivos podem ser conectados por barramento I2C (sensores, displays, outros microcontroladores, etc).

Neste exemplo, o I2C é usado para facilitar o uso do display de LCD, reduzindo o número de cabos necessários para seu funcionamento.

### Componentes necessários

1 Unidade – ESP32 D1 R32; Suas características são similares ao Arduino mas este contém módulos para WiFi e Bluetooth

1 Unidade – Sensor de temperatura e umidade DHT11 (normalmente é de cor azul) ou DHT22 (normalmente é branca); a diferença básica entre eles é a faixa de valores suportada nas medidas, a precisão das mesmas, além, é claro, do preço.

## 1 Unidade – Protoboard;

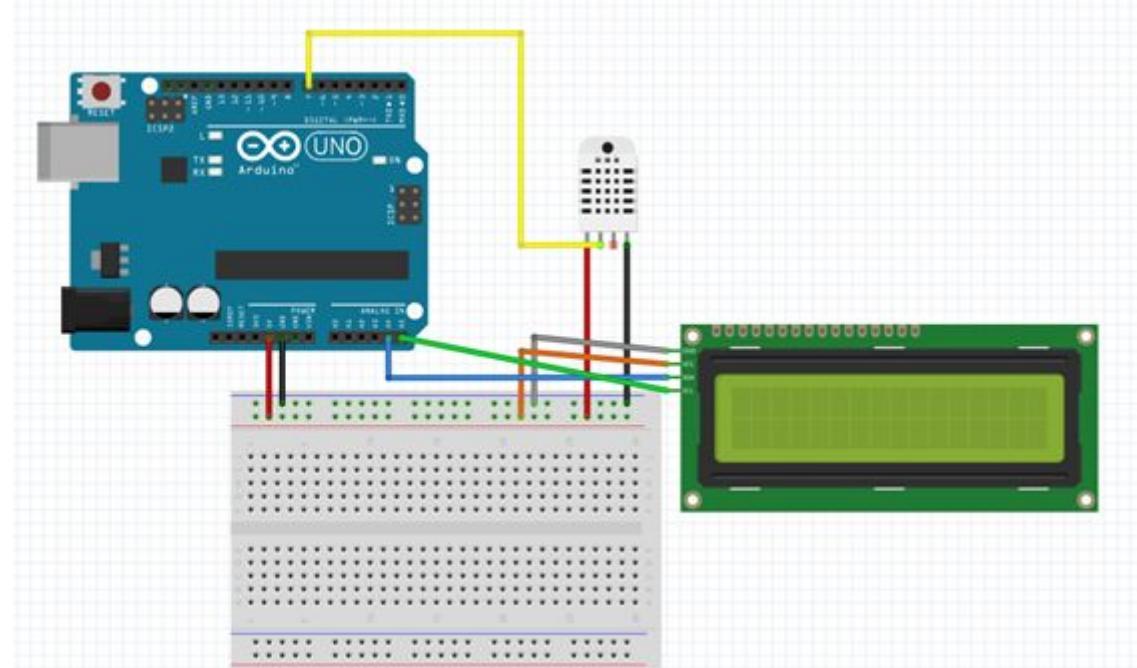
1 Unidade – Display LCD de 16x2 com adaptador I2C embutido;

Conjunto de cabos Jumper macho e fêmea.

# Montagem

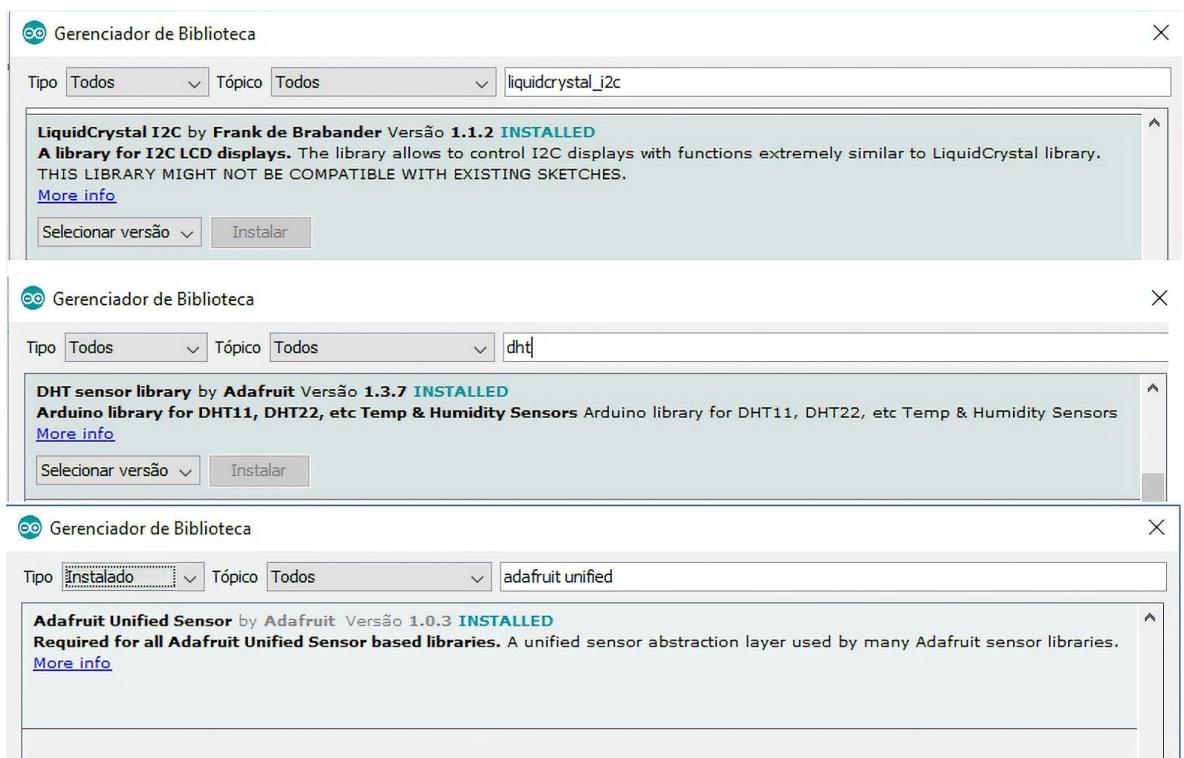
- 1) Conecte o Arduino na Protoboard**
  - 2) Conecte o sensor DHT11/DHT22 na Protoboard e no Arduino**
  - 3) Conecte o display ao Arduino**
  - 4) Conecte o Arduino ao cabo USB e este ao computador**
  - 5) Abra o Arduino IDE e coloque o código abaixo. Verifique se os cabos estão conectados nas mesmas portas, caso contrário você deverá alterá-las durante a programação**
  - 6) Caso o sensor exiba um resultado “NaN” (Not a Number), verifique se todos os cabos estão bem conectados**

## Diagrama



## Código utilizado

Antes é necessário realizar o download das bibliotecas do sensor DHT11/DHT22, Adafruit Sensor Unified e do I2C. Elas podem ser encontradas no gerenciador de bibliotecas do Arduino IDE (Sketch -> Incluir biblioteca -> Gerenciar bibliotecas)



Ao usar o código abaixo, altere adequadamente a definição existente na sexta linha para o tipo de sensor usado (DHT11 ou DHT22).

```
• #include <LiquidCrystal_I2C.h>
• #include <Adafruit_Sensor.h>
• #include <DHT.h>
• #include <DHT_U.h>
•
• #define DHTPIN 7      //Pino digital conectado ao Arduino
• #define DHTTYPE     DHT11    //Modelo do sensor a ser utilizado
• //#define DHTTYPE     DHT22
•
• // 0x27 como endereço do LCD. O LCD possui 16 colunas e 2 linhas
• LiquidCrystal_I2C lcd(0x27, 16, 2);
•
• DHT_Unified dht(DHTPIN, DHTTYPE);
```

```

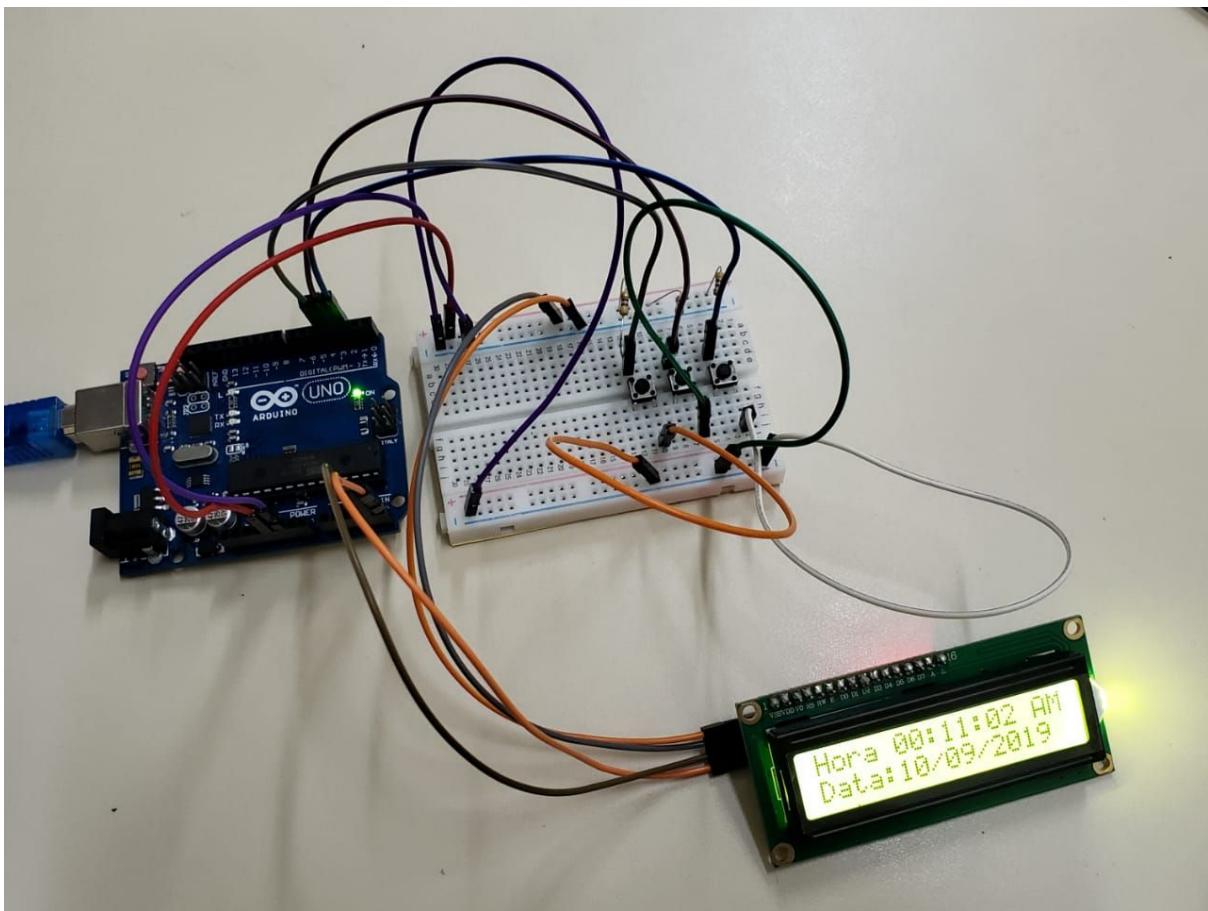
•     uint32_t delayMS;
•
•     //Montagem do simbolo "grau"
•     byte grau[8] = { B00001100,
•                      B00010010,
•                      B00010010,
•                      B00001100,
•                      B00000000,
•                      B00000000,
•                      B00000000,
•                      B00000000,
•                  };
•
•
•     void setup()
•     {
•         //Inicializa o serial
•         Serial.begin(9600);
•         Serial.print("Sensor de temperatura e umidade com I2C");
•         dht.begin();
•         sensor_t sensor;
•         //Imprime detalhes da temperatura
•         dht.temperature().getSensor(&sensor);
•
•         Serial.println(F("-----"));
•         Serial.println(F("Sensor de temperatura"));
•         Serial.print (F("Tipo de sensor: ")); Serial.println(sensor.name);
•         Serial.print (F("Versao      do      driver:      "));
•         Serial.println(sensor.version);
•         Serial.print (F("ID unico:    ")); Serial.println(sensor.sensor_id);
•         Serial.print (F("Valor maximo:    ")); Serial.print(sensor.max_value);
•         Serial.println(F("°C"));
•         Serial.print (F("Valor minimo:    ")); Serial.print(sensor.min_value);
•         Serial.println(F("°C"));
•         Serial.print (F("Resolucao:    ")); Serial.print(sensor.resolution);
•         Serial.println(F("°C"));
•         Serial.println(F("-----"));
•
•
•         //Imprime detalhes da umidade
•         dht.humidity().getSensor(&sensor);
•         Serial.println(F("Sensor de umidade"));
•         Serial.print (F("Tipo de sensor: ")); Serial.println(sensor.name);
•         Serial.print (F("Versao      do      driver:      "));
•         Serial.println(sensor.version);
•         Serial.print (F("ID unico:    ")); Serial.println(sensor.sensor_id);
•         Serial.print (F("Valor maximo:    ")); Serial.print(sensor.max_value);
•         Serial.println(F("%"));
•         Serial.print (F("Valor minimo:    ")); Serial.print(sensor.min_value);
•         Serial.println(F("%"));
•         Serial.print (F("Resolucao:    ")); Serial.print(sensor.resolution);
•         Serial.println(F("%"));
•         Serial.println(F("-----"));
•

```

```

•     //Define o delay entre leituras do sensor baseado nos detalhes do
•     sensor
•     delayMS = sensor.min_delay / 1000;
•
•
•     lcd.init();
•
•     lcd.backlight();
•     lcd.createChar(0, grau);           //Cria o caractere de grau
• }
•
• void loop()
• {
•     delay(delayMS);
•     //Pega um evento de temperatura e exibe seu valor
•     sensors_event_t event;
•     dht.temperature().getEvent(&event);
•
•     if (isnan(event.temperature)) {
•         Serial.println(F("Erro ao ler a temperatura!"));
•     }
•     else {
•         Serial.print(F("Temperatura: "));
•         Serial.print(event.temperature);
•         Serial.println(F("°C"));
•
•         lcd.setCursor(0, 0);
•         lcd.print("FATEC Ipiranga");
•         lcd.setCursor(0, 1);
•         lcd.print("T=");
•         lcd.print(event.temperature, 1);
•         lcd.write(byte(0));
•         lcd.print("C");
•
•     }
•     // Pega o evento de umidade e imprime seu valor
•     dht.humidity().getEvent(&event);
•     if (isnan(event.relative_humidity)) {
•         Serial.println(F("Erro ao ler a umidade!"));
•     }
•     else {
•         Serial.print(F("Humidity: "));
•         Serial.print(event.relative_humidity);
•         Serial.println(F("%"));
•
•         lcd.setCursor(9, 1);
•         lcd.print("U=");
•         lcd.print(event.relative_humidity, 1);
•         lcd.print("%");
•     }
• }
```

## Relógio Digital



**Descrição:** Um relógio digital Arduino utilizando um display LCD com o módulo I2C e com 3 botões para ajuste de horas, minutos e segundos. Para reduzir o consumo de energia, o backlight do display se apaga após 60 segundos em operação.

### Componentes necessários

1 Unidade – Arduino Uno;

1 Unidade – Protoboard;

1 Unidade – Display LCD de 16x2 com o módulo I2C;

3 Unidades - Botões

Conjunto de cabos Jumper macho e fêmea.

## Montagem

**1)** Conecte o Arduino na Protoboard

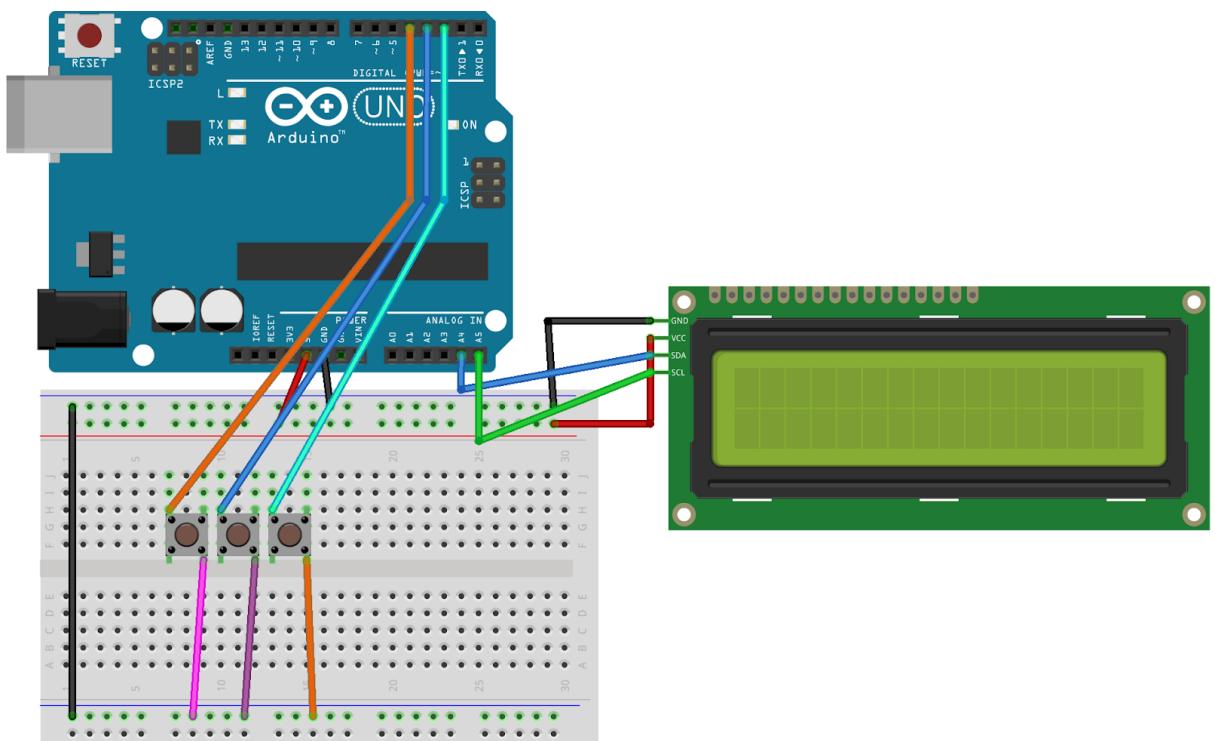
**2)** Conecte o display ao Arduino

**3)** Conecte os botões na Protoboard e ao Arduino (eles ficam ligados entre o terra ou GND e uma das portas de entrada do Arduino; usaremos as portas 2 para ajuste da hora, 3 para ajuste do minuto e 4 para zerar o mostrador de segundos)

**4)** Conecte o Arduino no cabo USB e então no computador

**5)** Abra o Arduino IDE e coloque o código abaixo.

## Diagrama



fritzing

## Código utilizado

```
• #include <LiquidCrystal_I2C.h>
• LiquidCrystal_I2C lcd(0x27, 16, 2);
• // inicializa o horario
```

```
•     int h = 16;
•     int m = 00;
•     int s = 00;
•     int flag = 1; //1 para PM, 0 para AM
•     //declara os botoes utilizados
•     int botao1;
•     int botao2;
•     int botao3;
•     //declara as portas do arduino conectadas aos botoes
•     int hs = 2; //horas
•     int ms = 3; //minutos
•     int sg = 4; //segundos
•     //Definições para o backlight
•     int setLight_ini = 60;
•     int setLight = setLight_ini;
•     int backlight = 1;
•     static uint32_t tempo_corrido, now = 0;
•
•
•     void tempo() {
•         while (now - tempo_corrido < 1000) {
•             now = millis();
•         }
•         tempo_corrido = now;
•     }
•
•     void setup() {
•         Serial.begin(9600);
•         pinMode(hs, INPUT_PULLUP);
•         pinMode(ms, INPUT_PULLUP);
•         pinMode(sg, INPUT_PULLUP);
•         lcd.init();
•         lcd.backlight();
•         pinMode(13, OUTPUT); //LED apenas para verificar se o botao esta
•         //funcionando
•     }
•
•     void loop() {
•         //ler o valor dos botoes
•         int sensorValHS = digitalRead(hs);
•         int sensorValMS = digitalRead(ms);
•         int sensorValSG = digitalRead(sg);
•
•         if (sensorValHS == HIGH) {
•             digitalWrite(13, LOW);
•         } else {
•             digitalWrite(13, HIGH);
•             Serial.println("Botao das horas pressionado");
•             h = h + 1;
•             verificador();
•         }
•         if (sensorValMS == HIGH) {
•             digitalWrite(13, LOW);
•         } else {
•             digitalWrite(13, HIGH);
•         }
•     }
• }
```

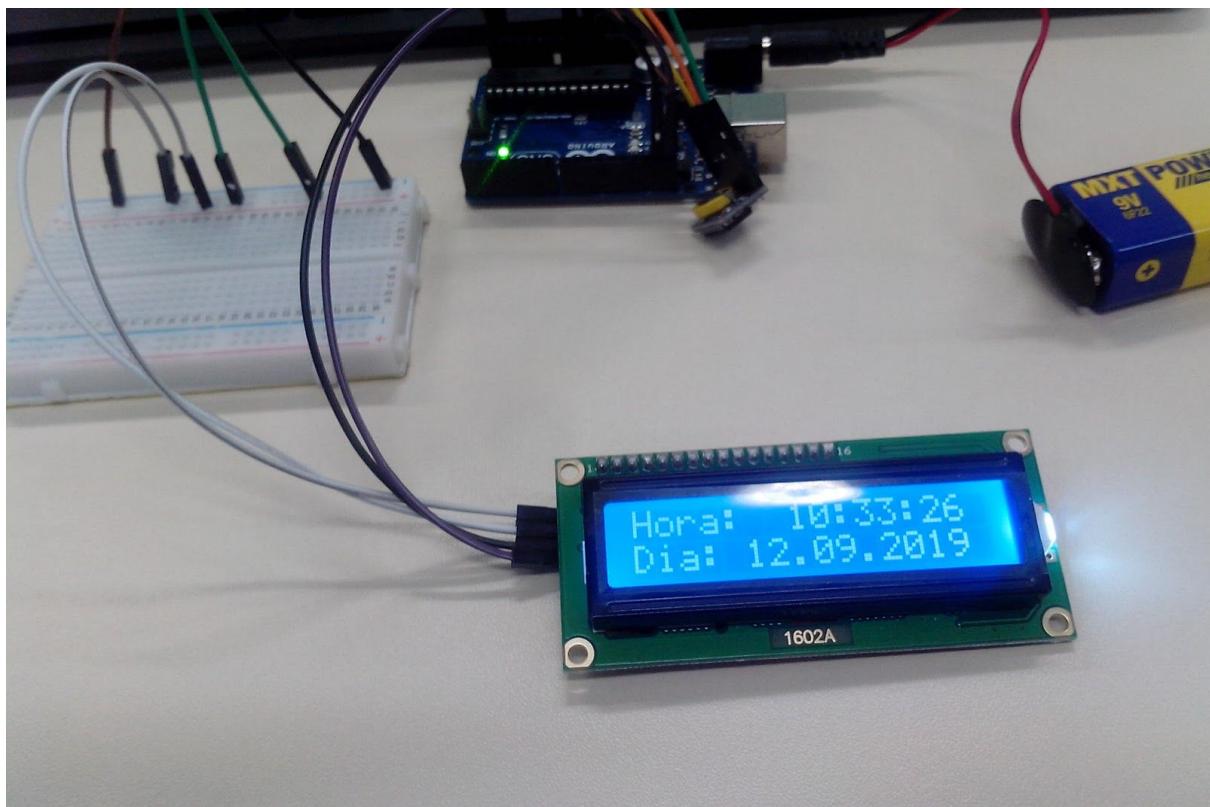
```

•     Serial.println("Botao dos minutos pressionado");
•     m = m + 1;
•     verificador();
• }
• if (sensorValsG == HIGH) {
•     digitalWrite(13, LOW);
• } else {
•     digitalWrite(13, HIGH);
•     Serial.println("Botao dos segundos pressionado");
•     s = 0;
• }
•
• lcd.setCursor(0, 0);
• lcd.print("Hora ");
• if (h < 10)lcd.print("0"); // sempre 2 dígitos
• lcd.print(h);
• lcd.print(":");
• if (m < 10)lcd.print("0");
• lcd.print(m);
• lcd.print(":");
• if (s < 10)lcd.print("0");
• lcd.print(s);
•
• if (flag == 0) lcd.print(" AM");
• if (flag == 1) lcd.print(" PM");
•
• lcd.setCursor(0, 1); // linha 2
• lcd.print("Data:10/09/2019");
•
• tempo();
• s = s + 1;
• verificador();
}

void verificador() {
    //metodo para evitar erros como 61 segundos, minutos ou 25 horas
    if (s >= 60) {
        s = 0;
        m++;
    }
    if (m >= 60)
    {
        m = 0;
        h = h + 1;
    }
    if (h == 12 && flag == 0) {
        flag++;
    }
    if (h >= 24) {
        h = 0;
        flag--;
    }
}

```

## Relógio Digital com RTC



**Descrição:** Um relógio digital feito com Arduino utilizando o módulo RTC (Real Time Clock) para armazenamento da data e hora.

### Componentes necessários

1 Unidade – Arduino Uno;

1 Unidade – Módulo RTC DS3231

1 Unidade – Protoboard;

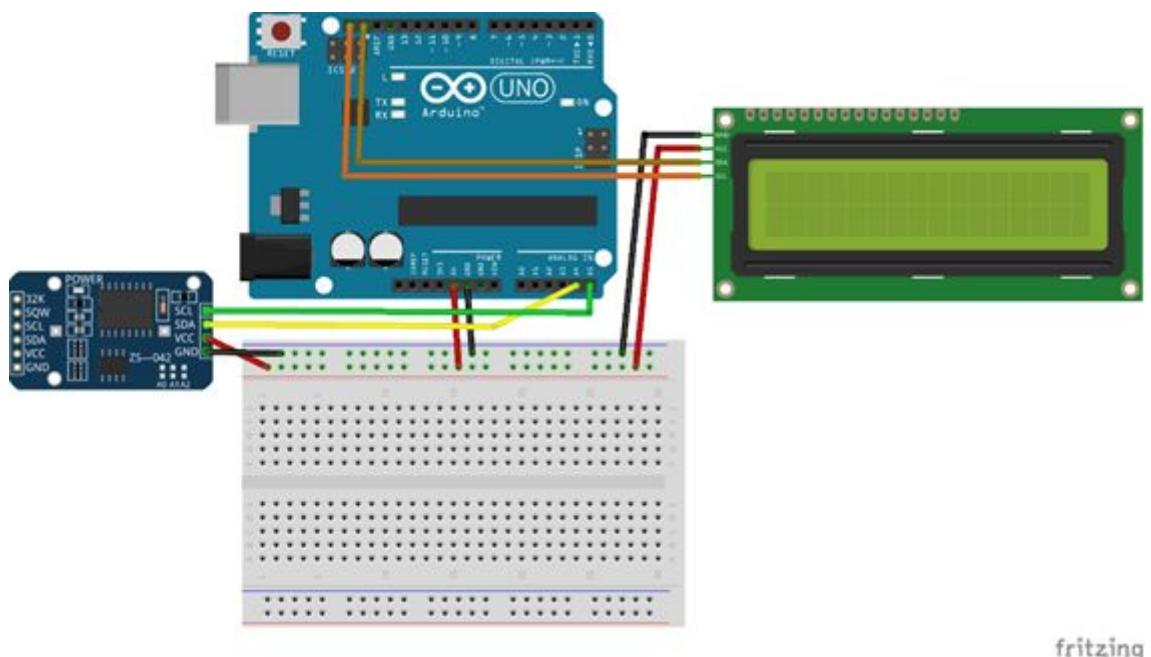
1 Unidade – Display LCD de 16x2 com o módulo I2C;

Conjunto de cabos Jumper macho e fêmea.

## Montagem

- 1)** Conecte o Arduino na Protoboard
- 2)** Conecte o módulo RTC DS3231 na Protoboard e no Arduino
- 3)** Conecte o display ao Arduino
- 4)** Conecte o Arduino no cabo USB e então no computador
- 5)** Abra o Arduino IDE e coloque o código abaixo.

## Diagrama



## Código utilizado

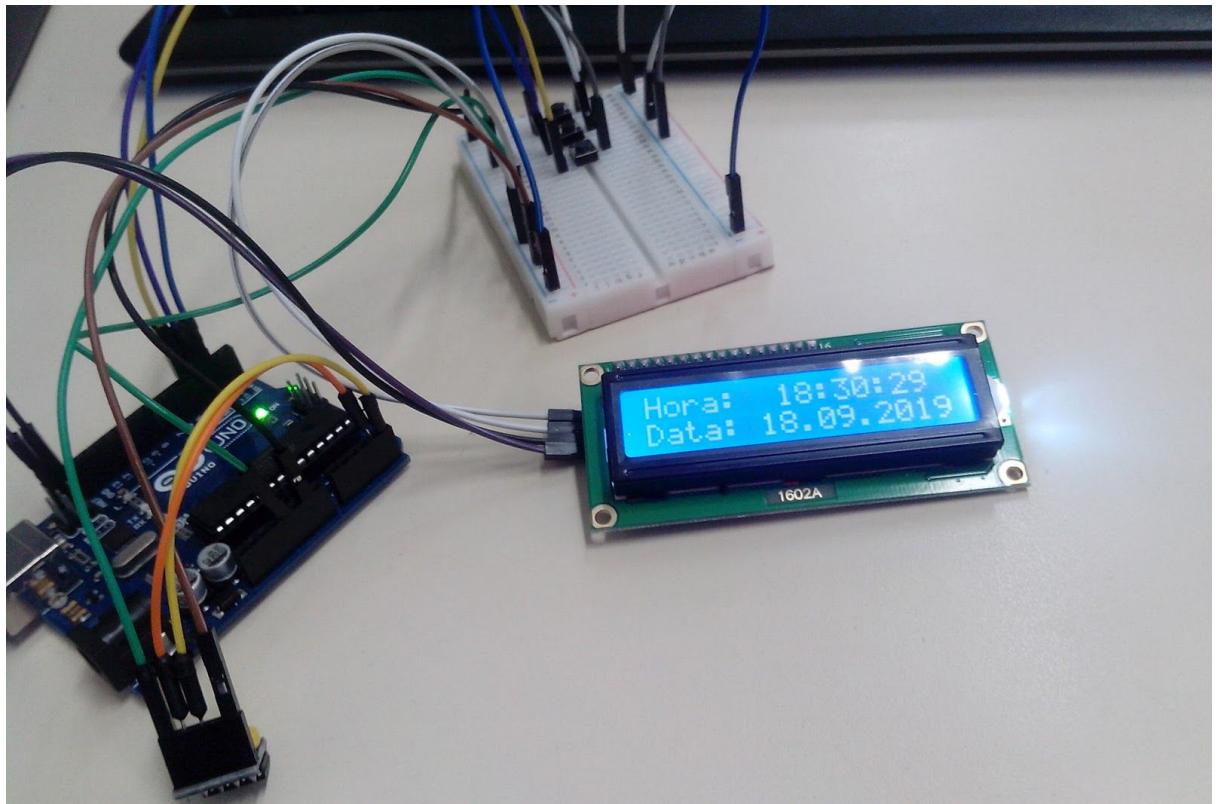
Antes é necessário baixar a biblioteca do DS3231 e adiciona-la ao Arduino IDE. Ela pode ser encontrada no link abaixo:

### Referências:

<http://www.rinkydinkelectronics.com/library.php?id=73>

```
●  #include <DS3231.h>
●  #include <LiquidCrystal_I2C.h>
●  LiquidCrystal_I2C lcd(0x27, 16, 2);
●
●  //Inicializa o DS3231
●  DS3231 rtc(SDA, SCL);
●
●  void setup()
●  {
●      //Liga o LCD
●      lcd.init();
●      lcd.backlight();
●
●      Serial.begin(9600);
●
●      //Inicializa o objeto rtc
●      rtc.begin();
●
●      // As linhas a seguir podem ser descomentadas e usadas para arrumar a
●      //data e hora
●      //rtc.setDOW(THURSDAY);          //Dia da semana
●      //rtc.setTime(10, 30, 0);        // Horario em horas, minutos e segundos,
●      //formato de 24 horas
●      //rtc.setDate(12, 9, 2019);     // Data em dia, mes, ano
●  }
●
●  void loop()
●  {
●      //Exibe dia da semana
●      Serial.print(rtc.getDOWStr());
●      Serial.print(" ");
●
●      //Exibe data
●      Serial.print(rtc.getDateStr());
●      Serial.print(" - ");
●
●      //Exibe hora
●      Serial.println(rtc.getTimeStr());
●
●      //Exibicao no lcd
●      lcd.setCursor(0, 0);
●      lcd.print("Hora: ");
●      lcd.print(rtc.getTimeStr());
●
●      lcd.setCursor(0, 1);
●      lcd.print("Data: ");
●      lcd.print(rtc.getDateStr());
●
●      delay (1000);
●  }
```

## **Relógio Digital com botões e RTC**



**Descrição:** Um relógio digital feito com Arduino utilizando o módulo RTC (Real Time Clock) para armazenamento da hora e dia atual e botões para ajuste de horas

### **Componentes necessários**

1 Unidade – Arduino Uno;

1 Unidade – Módulo RTC DS3231

1 Unidade – Protoboard;

1 Unidade – Display LCD de 16x2 com o módulo I2C;

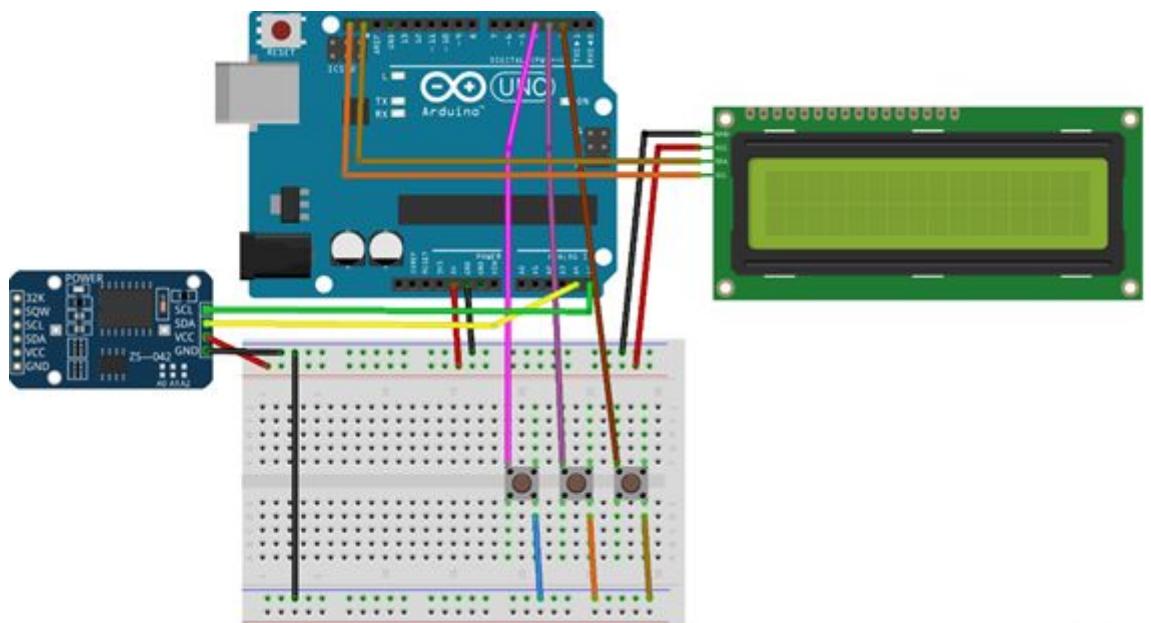
3 Unidades - Botões

Conjunto de cabos Jumper macho e fêmea.

## Montagem

- 1)** Conecte o Arduino na Protoboard
- 2)** Conecte o módulo RTC DS3231 na Protoboard e no Arduino
- 3)** Conecte o display ao Arduino
- 4)** Encaixe os botões na Protoboard e conecte-os no Arduino
- 5)** Conecte o Arduino no cabo USB e então no computador
- 6)** Abra o Arduino IDE e coloque o código abaixo

## Diagrama



fritzing

## Código utilizado

```
• #include <LiquidCrystal_I2C.h>
• #include <DS3231.h>
•
• LiquidCrystal_I2C lcd(0x27, 16, 2);
```

```
• DS3231 rtc(SDA, SCL);
• //declaracao da struct
• Time t;
• //variaveis para hora,min e seg
• int h;
• int m;
• int s;
• //declara os botoes utilizados
• int botao1;
• int botao2;
• int botao3;
•
• //declara as portas do arduino conectadas aos botoes
• int hs = 2; //horas
• int ms = 3; //minutos
• int sg = 4; //segundos
•
• void ajustarHora(int hora, int minuto, int segundo) {
•     rtc.setTime(hora, minuto, segundo);
• }
•
• void setup() {
•     //botoes
•     pinMode(hs, INPUT_PULLUP);
•     pinMode(ms, INPUT_PULLUP);
•     pinMode(sg, INPUT_PULLUP);
•     pinMode(13, OUTPUT); //LED apenas para verificar se o botao esta
funcionando
•
•     //Liga o LCD
•     lcd.init();
•     lcd.backlight();
•
•     Serial.begin(9600);
•
•     rtc.begin();
•     t = rtc.getTime();
•     h = t.hour;
•     m = t.min;
•     s = t.sec;
• }
•
• void loop() {
•     Serial.println(rtc.getTimeStr());
•
•     s++;
•     verificador();
•     //Exibicao da hora no lcd
•     lcd.setCursor(0, 0);
•     lcd.print("Hora: ");
•     lcd.print(rtc.getTimeStr());
•
•     lcd.setCursor(0, 1);
•     lcd.print("Data: "));
```

```
•     lcd.print(rtc.getDateStr());
•
•     //ler o valor dos botoes
•     int sensorValHS = digitalRead(hs);
•     int sensorValMS = digitalRead(ms);
•     int sensorValSG = digitalRead(sg);
•
•     //botao pressionado
•     if (sensorValHS == HIGH) {
•         digitalWrite(13, LOW);
•     } else {
•         digitalWrite(13, HIGH);
•         h++;
•         verificador();
•         ajustarHora(h, m, s);
•     }
•
•     if (sensorValMS == HIGH) {
•         digitalWrite(13, LOW);
•     } else {
•         digitalWrite(13, HIGH);
•         m++;
•         verificador();
•         ajustarHora(h, m, s);
•     }
•     if (sensorValSG == HIGH) {
•         digitalWrite(13, LOW);
•     } else {
•         digitalWrite(13, HIGH);
•         s = 0;
•         ajustarHora(h, m, s);
•     }
•
•     delay (1000);
• }
•
• void verificador() {
•     //metodo para evitar erros como 61 segundos, minutos ou 25 horas
•     if (s >= 60) {
•         s = 0;
•         m++;
•         t = rtc.getTime();
•         h = t.hour;
•         m = t.min;
•         s = t.sec;
•     }
•     if (m >= 60)
•     {
•         m = 0;
•         h++;
•     }
•     if (h >= 24) {
•         h = 0;
•     }
• }
```

## Projeto GPS em Arduino



**Descrição:** Utilizando o módulo Neo7m pode-se obter latitude e longitude aproximadas, além de hora e data atuais.

### Componentes necessários

- 1 Unidade – Arduino Uno;
- 1 Unidade – Módulo Neo7m;
- 1 Unidade – Protoboard;
- 1 Unidade – Display LCD de 16x2 com o módulo I2C;

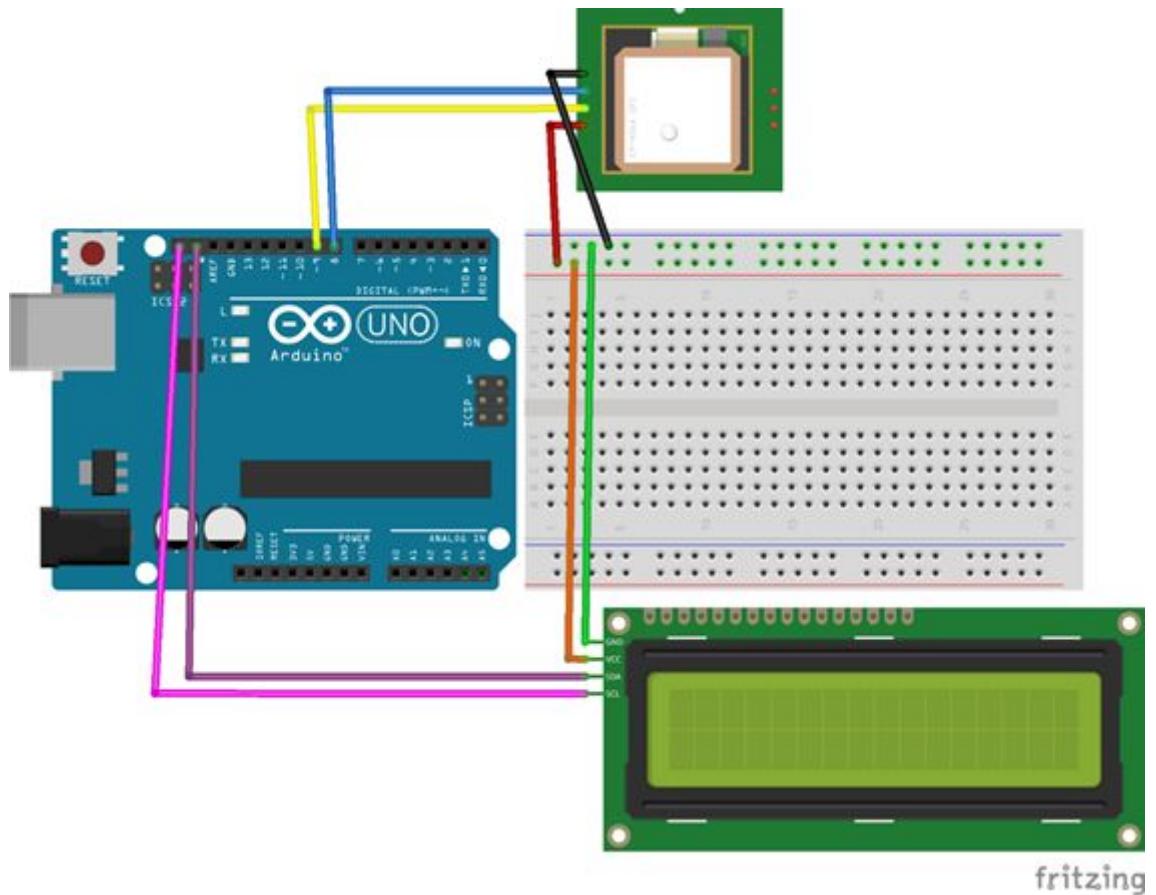
Conjunto de cabos Jumper macho e fêmea.

### Montagem

- 1)** Conecte o Arduino na Protoboard
- 2)** Conecte o módulo RTC DS3231 na Protoboard e no Arduino
- 3)** Conecte o display ao Arduino
- 4)** Conecte o Arduino no cabo USB e então no computador

**5)** Abra o Arduino IDE e coloque o código abaixo

Diagrama



Código utilizado

As seguintes bibliotecas são necessárias:

**Referências:**

TinyGPS++: disponível em <http://arduiniana.org/libraries/tinygpsplus/>

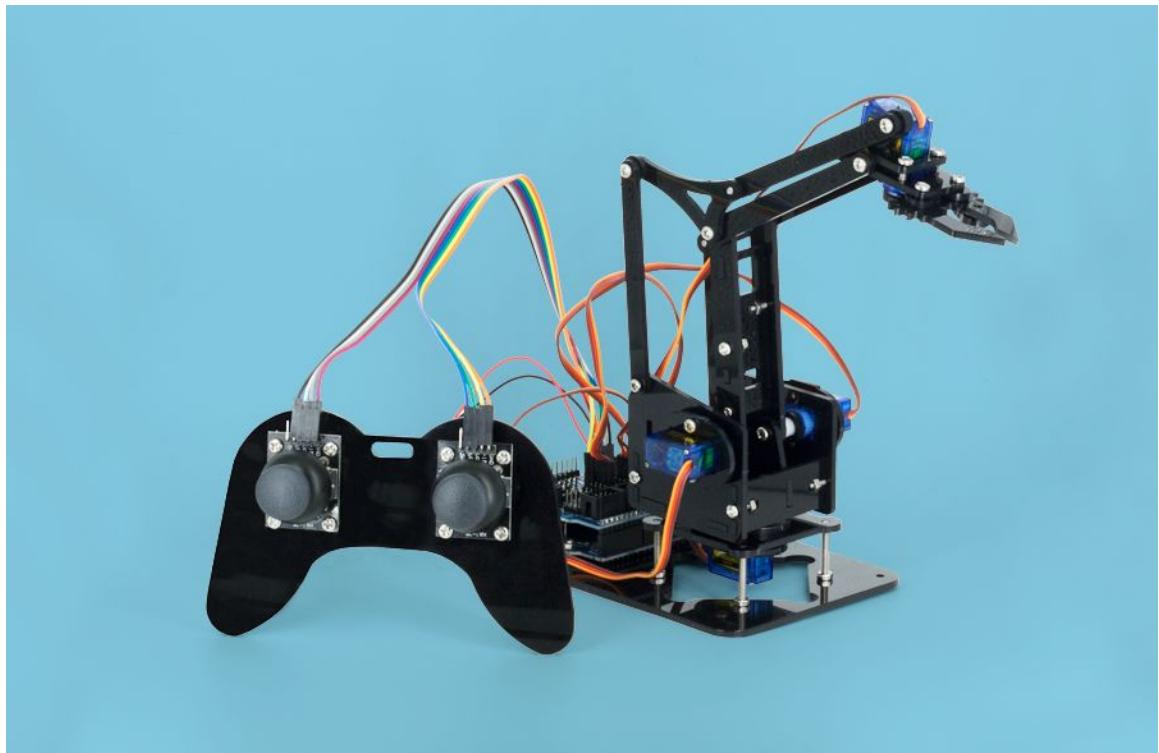
SoftwareSerial: inclusa no Arduino IDE

LiquidCrystal\_i2c: disponível no gerenciador de bibliotecas do Arduino

```
• #include <LiquidCrystal_I2C.h>
• #include <TinyGPS++.h>
• #include <SoftwareSerial.h>
•
• LiquidCrystal_I2C lcd(0x27, 16, 2);
•
• //Instancia do objeto TinyGPSPlus
• TinyGPSPlus gps;
•
• //Portas utilizadas para a conexao serial
• SoftwareSerial ss(8, 9);
•
• void setup()
• {
•     Serial.begin(9600);
•     ss.begin(9600);
•
•     lcd.init();
•     lcd.backlight();
• }
•
• void loop()
• {
•     // Exibe informacoes sempre que obtém novos dados
•     while (ss.available() > 0)
•         if (gps.encode(ss.read()))
•             displayInfo();
•
•     if (millis() > 5000 && gps.charsProcessed() < 10)
•     {
•         Serial.println(F("GPS não detectado. Verifique instalação"));
•         while (true);
•     }
• }
•
• void displayInfo()
• {
•     Serial.print(F("Localização: "));
•     if (gps.location.isValid())
•     {
•         Serial.print(gps.location.lat(), 6);
•         Serial.print(F(","));
•         Serial.print(gps.location.lng(), 6);
•         lcd.setCursor(0, 0);
•         lcd.print("LAT: ");
•         lcd.print(gps.location.lat(), 6);
•         lcd.setCursor(0, 1);
•         lcd.print("LNG: ");
•         lcd.print(gps.location.lng(), 6);
•     }
•     else
•     {
• }
```

```
•     Serial.print(F("INVALIDA"));
•
•     lcd.setCursor(0, 0);
•     lcd.print("SEM SINAL");
• }
•
• Serial.print(F(" Data/Hora: "));
• if (gps.date.isValid())
{
•     Serial.print(gps.date.day());
•     Serial.print(F("/"));
•     Serial.print(gps.date.month());
•     Serial.print(F("/"));
•     Serial.print(gps.date.year());
}
• else
{
•     Serial.print(F("INVALIDA"));
}
•
• Serial.print(F(" "));
• if (gps.time.isValid())
{
•     int horaBrasilia = gps.time.hour() - 3;
•     if (gps.time.hour() < 10) Serial.print(F("0"));
//Serial.print(gps.time.hour());
•     Serial.print(horaBrasilia);
•     Serial.print(F(":"));
•     if (gps.time.minute() < 10) Serial.print(F("0"));
•     Serial.print(gps.time.minute());
•     Serial.print(F(":"));
•     if (gps.time.second() < 10) Serial.print(F("0"));
•     Serial.print(gps.time.second());
•     Serial.print(F("."));
•     if (gps.time.centisecond() < 10) Serial.print(F("0"));
•     Serial.print(gps.time.centisecond());
}
• else
{
•     Serial.print(F("INVALIDA"));
}
•
• Serial.println();
delay(1000);
}
```

## Projeto Braço Mecânico controlado por Joystick



**Descrição:** Desenvolver um braço mecânico com uma garra, e fazer sua movimentação controlado por um dispositivo de entrada analógica (joystick) e botões.

### Componentes necessários

1 Unidade – Arduino Uno;

1 Unidade – Shield Funduino de Joystick;

1 Unidade – Protoboard;

1 Unidade – Kit de partes mecânicas cortadas a laser (MeArm versão 1.0)  
Link:

### Referências:

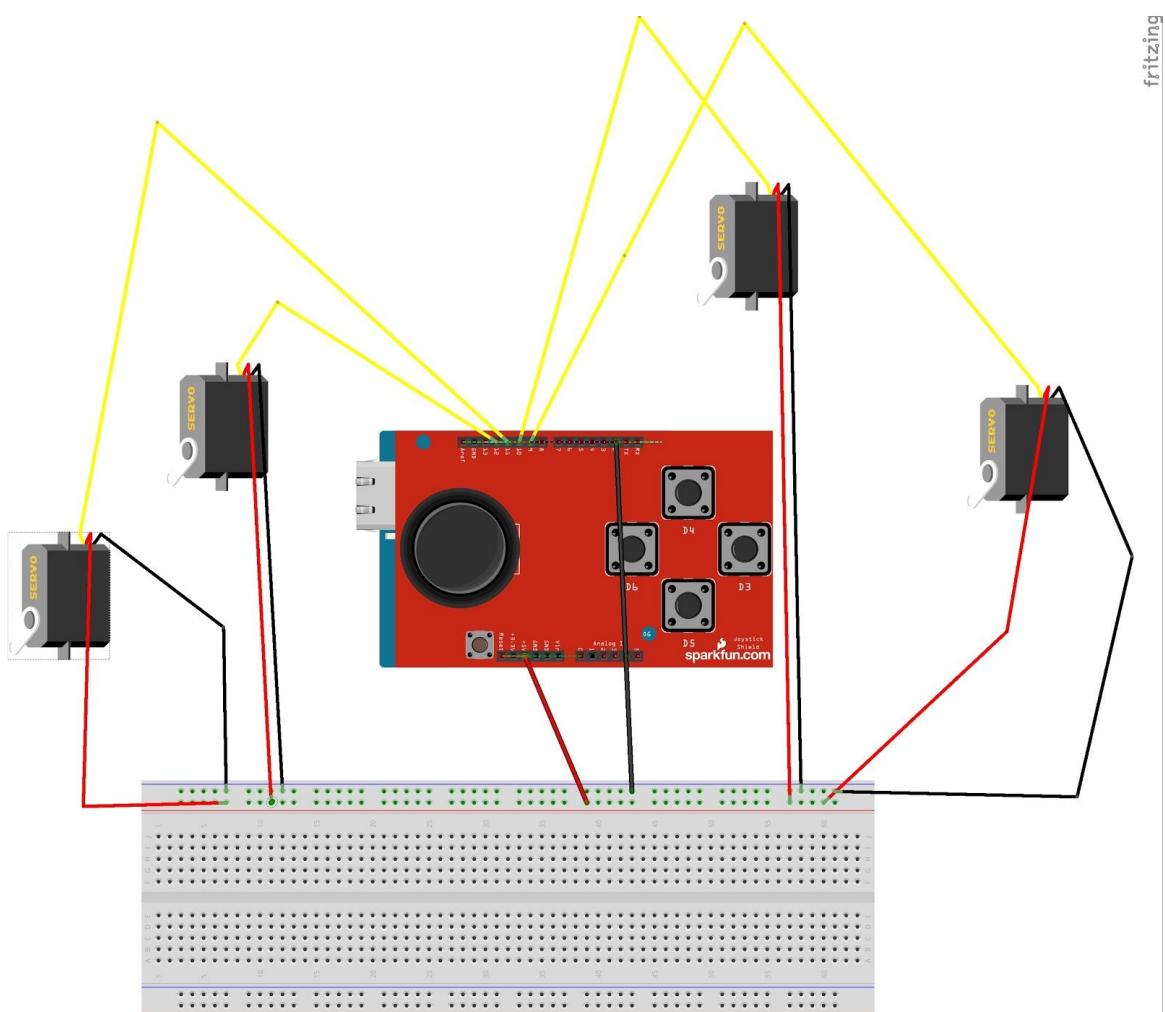
<https://www.thingiverse.com/thing:993759>

Conjunto de cabos Jumper macho e fêmea.

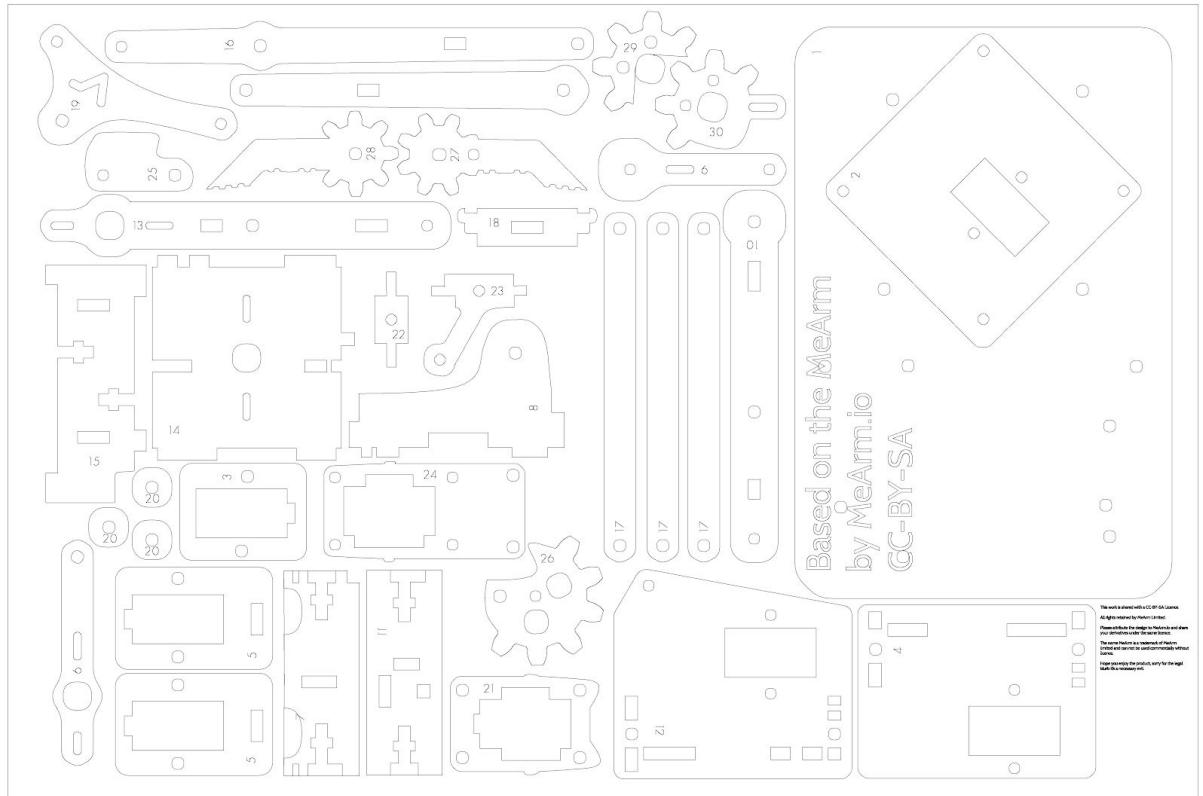
# Montagem

- 1) Conecte o Shield Funduino sobre o Arduino
  - 2) Monte as partes mecânicas conforme a documentação
  - 3) Conecte os servos na placa Funduino
  - 4) Conecte o Arduino no cabo USB e então no computador
  - 5) Abra o Arduino IDE e coloque o código a seguir

## Diagrama elétrico



## Diagrama mecânico



## Código utilizado

Desenvolvemos uma versão orientada a objetos (OO) em C++ para controlar o braço mecânico.

A motivação desta versão OO veio quando observamos que o código original, estruturado, em linguagem C, estava repetitivo, visto que todos os 4 servos tinham um tratamento computacional parecido, uma lógica de ação semelhante.

Pode-se observar no código a seguir que a classe **Servo** original foi especializada através do mecanismo da Orientação à Objetos chamado herança, dando origem à classe **MeuServo**. Os novos atributos criados memorizam as características específicas de cada um dos motores servo e o método **comandoServo** faz todo o mapeamento do que chega do joystick para a atuação segura dos motores servos, sem ultrapassar seus limites.

```

•
• #include <Servo.h>
• // Valores das portas de controle dos servos
• int const GARRA = 9;
• int const BASE = 10;
• int const ESQUERDA = 11;
• int const DIREITA = 12;
•
• // Valores limites dos ângulos dos servos
• int const minServoGarra = 105;
• int const maxServoGarra = 150;
• int const minServoBase = 105;
• int const maxServoBase = 170;
• int const minServoEsquerda = 90;
• int const maxServoEsquerda = 150;
• int const minServoDireita = 105;
• int const maxServoDireita = 160;
•
• // Funduino Joystick Shield
• int const UP_BTN = 2; //botao vermelho A
• int const DOWN_BTN = 4;
• int const LEFT_BTN = 5;
• int const RIGHT_BTN = 3;
• int const E_BTN = 6;
• int const F_BTN = 7;
• int const JOYSTICK_BTN = 8;
• int const JOYSTICK_AXIS_X = A0;
• int const JOYSTICK_AXIS_Y = A1;
• int buttons[] = {UP_BTN, DOWN_BTN, LEFT_BTN, RIGHT_BTN, E_BTN, F_BTN,
JOYSTICK_BTN};
•
• // Especialização (herança) da classe Servo
• class MeuServo: public Servo {
•     public:
•         int _min, _max, _espelhamento, _porta, memoriaPosicao;
•
•         MeuServo(int min, int max, int espelhamento, int porta)
•         { _min = min;
•             _max = max;
•             _espelhamento = espelhamento;
•             _porta = porta;
•             attach(_porta);
•             memoriaPosicao = (_min + _max)/2;
•             write(memoriaPosicao);
•         }
•         void comandoServo (int joystick)
•         { int delta = (int)((joystick * _espelhamento)/8192.0 * (_max -
_min));
•             int comando = memoriaPosicao + delta;
•             if(comando > _max) comando = _max;
•             else if(comando < _min) comando = _min;
•             memoriaPosicao = comando;
•             write(comando);
•         }
• }

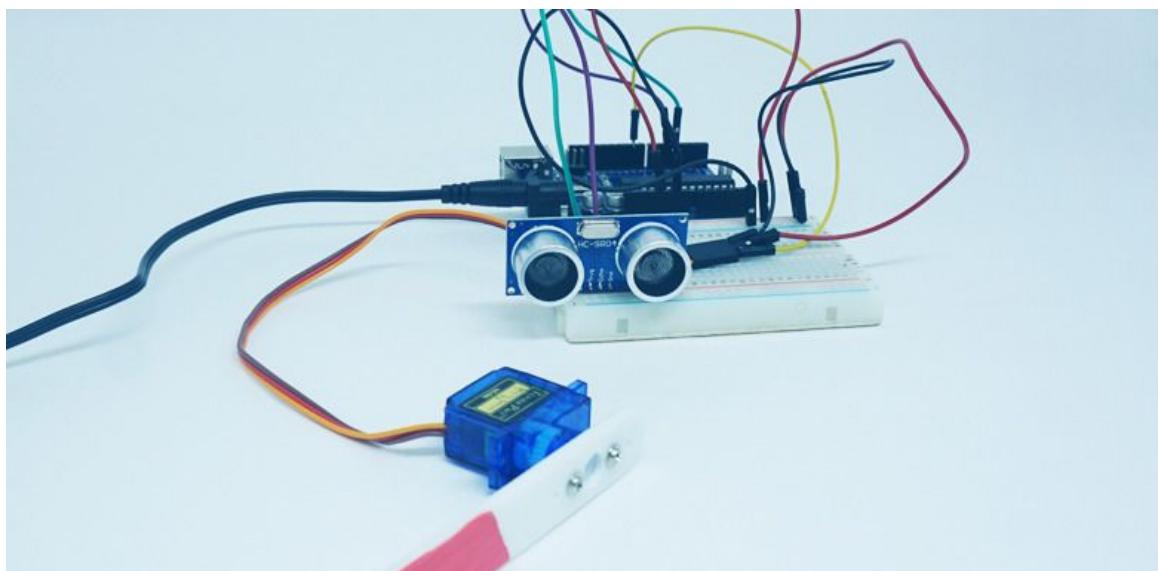
```

```

•     };
•
•     MeuServo    *servoGarra,    *servoBase,    *servoEsquerda,    *servoDireita,
•     *servoAtual; //ponteiros
•
• void setup() {
•     // Ajuste todos os botões como entradas com resistores de pullup
•     // (ligados ao positivo)
•     for (int i; i < 7; i++) pinMode(buttons[i], INPUT_PULLUP);
•
•     //Serial.begin(9600);
•
•     // Criação dos objetos MeuServo, cada um com sua configuração
•     // específica
•     MeuServo s1 = MeuServo(minServoGarra, maxServoGarra, 1, GARRA);
•     servoGarra = &s1;
•     MeuServo s2 = MeuServo(minServoBase, maxServoBase, -1, BASE);
•     servoBase = &s2;
•     MeuServo s3 = MeuServo(minServoEsquerda, maxServoEsquerda, 1,
•     ESQUERDA);
•     servoEsquerda = &s3;
•     MeuServo s4 = MeuServo(minServoDireita, maxServoDireita, 1, DIREITA);
•     servoDireita = &s4;
•
•     // No início da operação, os botões agirão sobre a Garra
•     servoAtual = servoGarra;
• }
•
• void loop() {
•     if (digitalRead(UP_BTN) == 0) servoAtual = servoGarra;
•     else if (digitalRead(DOWN_BTN) == 0) servoAtual = servoBase;
•     else if (digitalRead(LEFT_BTN) == 0) servoAtual = servoEsquerda;
•     else if (digitalRead(RIGHT_BTN) == 0) servoAtual = servoDireita;
•
•     int x = analogRead(JOYSTICK_AXIS_X) - 512;    // de -512 a +511
•     if(abs(x) < 15) x = 0; //tirar fantasma!!!
•     int y = analogRead(JOYSTICK_AXIS_Y) - 512;    // de -512 a +511
•     if(abs(y) < 15) y = 0; //tirar fantasma!!!
•     int z = abs(x) > abs(y) ? x : y; //uso sempre o maior movimento entre
•     os eixos X e Y
•     servoAtual->comandoServo(z); //executo a ação
•     delay(50);
• }

```

## Cancela automática



**Descrição:** Desenvolver um sistema que usa um sensor ultrassônico para acionar um servo-motor, com o intuito de simular o funcionamento de uma cancela automática.

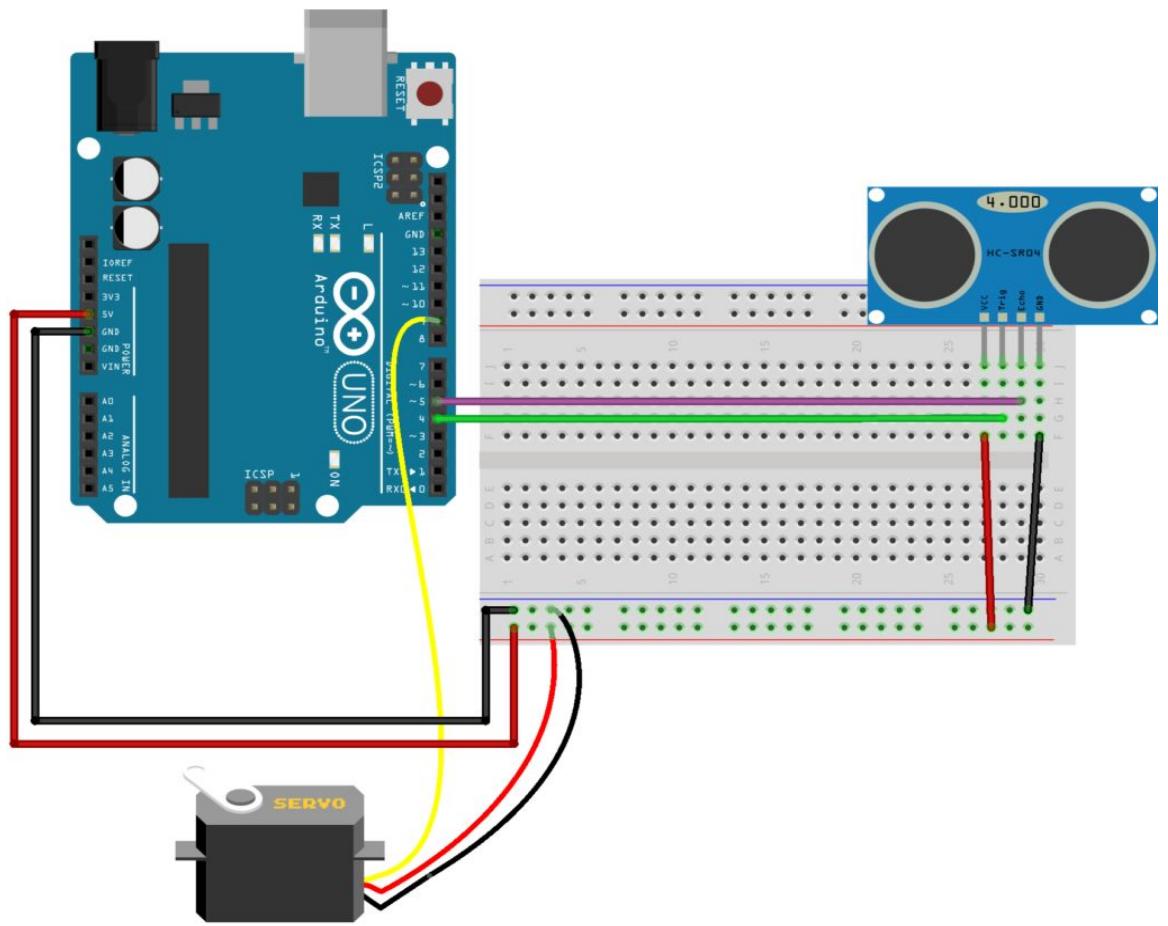
### Componentes necessários

- 1 Unidade – Arduino Uno;
- 1 Unidade – Servo motor;
- 1 Unidade – Protoboard;
- 1 Unidade – Sensor Sônico;
- Conjunto de cabos Jumper macho.

## Montagem

- 1)** Conecte o Sensor Sônico a protoboard como na figura.
- 2)** Em seguida conecte os cabos Jumpers de alimentação e o terra (vcc e gnd) na suas pinagens correspondente na protoboard, positivo para alimentação e negativo para terra .
- 3)** Conecte agora os dois cabos Jumpers correspondentes ao emissor e receptor de sinal a placa Arduino nas entradas PWM como na figura.
- 4)** Conecte os cabo de alimentação e o terra do Servo motor a protoboard como na figura.
- 5)** Em seguida conecte o cabo de pulso no Arduino nas entradas PWM como na figura.
- 6)** Conecte o cabo usb ao Arduino e então conecte ao computador.
- 7)** Abra o Arduino IDE e coloque o código a seguir .

## Diagrama Elétrico



fritzing

## Código utilizado

```
● #include <Servo.h>
● #include "Ultrasonic.h" //biblioteca importada - ULTRASONIC by Erick Simoes
● #include <HCSR04.h>
●
● //Criar objeto Servo para controlar o servo
● Servo motor_servo;
● //objeto do tipo Ultrasonic tendo, como parâmetros:
● //pinos Trig(Emissor) e pino Echo(receptor), respectivamente
● Ultrasonic ultrasonic(4,5);
● int distancia; //variavel para armazenar valor do receptor
●
● void setup()
● {
●     motor_servo.attach(10); //atribui nosso objeto servo ao pino 10
●     Serial.begin(9600);
●     Serial.println("Lendo dados do sensor...");
● }
```

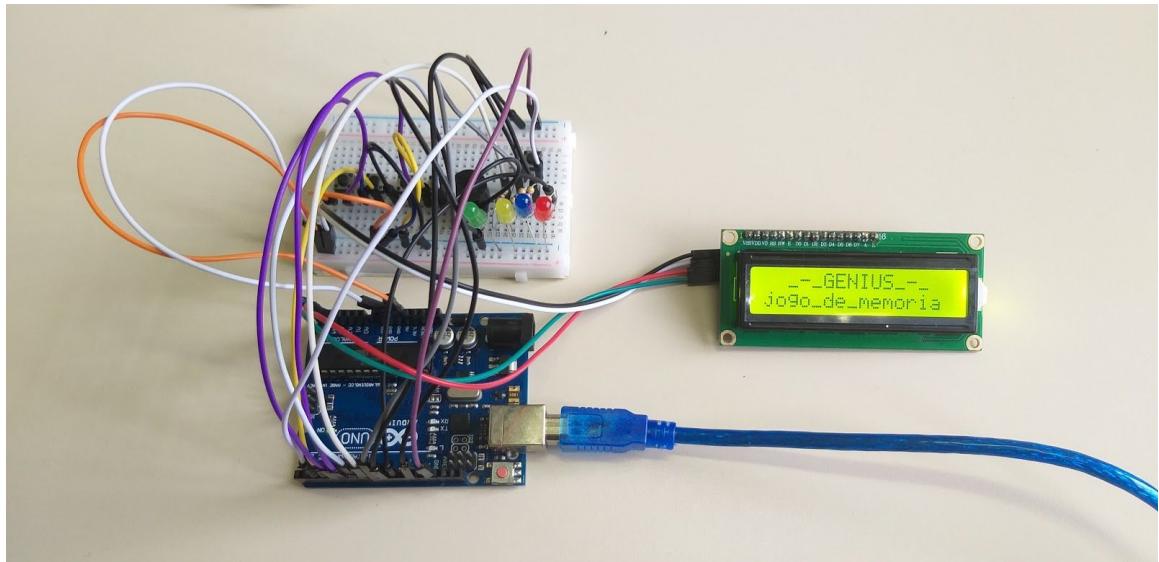
```
• void loop()
• {
•     distancia = ultrasonic.read(CM);
•     if(distancia < 10)
•     {
•         liga_servo();
•         Serial.print("Abrir cancela\n");
•         while(distancia < 10)
•         {
•             distancia = ultrasonic.read(CM);
•             delay(1000);
•         }
•     }
•     Serial.print("Fechar cancela\n");
•     fecha_servo();
•     delay(500);
• }
```

•

•

```
• void liga_servo()
• {
•     int abre_porta = 0;
•     motor_servo.write(abre_porta);
• }
• void fecha_servo()
• {
•     int fecha_porta = 100;
•     motor_servo.write(fecha_porta);
• }
```

## Jogo Genius



**Descrição:** Baseado no famoso Genius era um brinquedo muito popular na década de 1980 distribuído pela Brinquedos Estrela. O brinquedo buscava estimular a memorização de cores e sons. Com um formato semelhante a um OVNI, possuía botões coloridos que emitem sons harmônicos e se iluminavam em sequência. Nessa adaptação não usaremos os sons.

## Componentes necessários

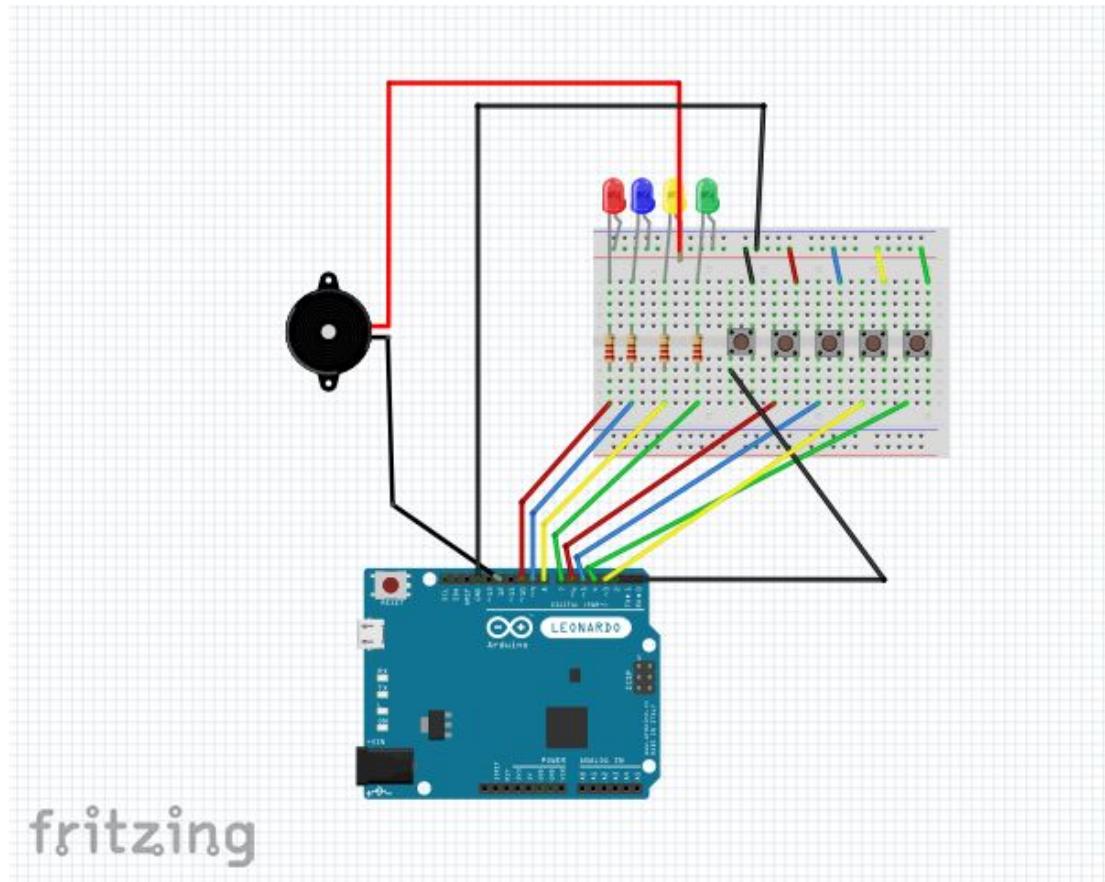
- 1 Unidade – Arduino Uno;
- 1 Unidade – Protoboard;
- 2 Unidades – Leds de cores diferentes;
- 2 Unidades – Botões;
- 4 Unidades – Resistores;
- Conjunto de cabos Jumper macho.

## Montagem

- 1)** Conecte os leds a protoboard nas suas respectivas (ou correspondentes) pinagens como no Diagrama Elétrico.
- 2)** Conecte os cabos jumpers a alimentação negativa da protoboard nas suas respectivas (ou correspondentes) pinagens como no Diagrama Elétrico e nos 2 Leds.
- 3)** Conecte agora os dois Resistores a protoboard e aos Leds nas suas respectivas (ou correspondentes) pinagens como no Diagrama Elétrico.
- 4)** Conecte os cabos Jumpers aos Resistores e a placa Arduino Uno como no Diagrama Elétrico.
- 5)** Conecte agora os dois botões a protoboard nas suas respectivas (ou correspondentes) pinagens como no Diagrama Elétrico.
- 6)** Conecte os resistores a protoboard como no Diagrama Elétrico e conecte dois cabos Jumpers para cada extremidade dos resistores correspondentes na protoboard uma para alimentação negativa (terra) e outra para a placa Arduino.
- 7)** Conecte os cabos Jumpers uma extremidade a posição correspondente na protoboard e a outra a alimentação positiva.
- 8)** Conecte o cabo usb ao Arduino e então conecte ao computador.

**9)** Conecte os cabos Jumpers uma extremidade a posição correspondente na protoboard e a outra a alimentação positiva.

Diagrama Elétrico



Código utilizado

```
• #include <LiquidCrystal_I2C.h>
• #include <EEPROM.h>
• #include <Wire.h>
• // Nossa sequência de até 100 itens vai começar vazia.
• int sequencia[100] = {};
• // Indica a rodada atual que o jogo se encontra.
• int rodada_atual = 0;
• // Indica o passo atual dentro da sequência, é usado enquanto a sequência
• // está sendo reproduzida.
• int passo_atual_na_sequencia = 0;
• // Vamos começar definindo as notas para os sons
• #define NOTE_D4 294
• #define NOTE_G4 392
• #define NOTE_A4 440
• #define NOTE_A5 880
```

```
• #define endereco_eeprom_recorde 123
• // Inicializa o display no endereço 0x27
• LiquidCrystal_I2C lcd(0x27,16 ,2);
• //criando o array para os 4 sons para sortear um som
• int tons[4] = { NOTE_A5, NOTE_A4, NOTE_G4, NOTE_D4 };
• /* Os pinos de leds e botões estão em ordem, relacionados uns aos
outros. A ordem destas
• * sequências também estão relacionadas a ordem dos tons.
• */
• #define N 4
• int pinoAudio = 12;
• int pinosLeds[N] = {8,7,9,10};
• int pinosBotoes[N+1] = {3,4,5,6,2}; //N botões para o jogo mais 1 de
reset
• // Indica se um botão foi pressionado durante o loop principal.
• int botao_pressionado = 0;
• // Flag indicando se o jogo acabou.
• int perdeu_o_jogo = true;
• // Variáveis para guardar os pontos
• int pontuacao;
•
• void setup() {
•     pontuacao = 0;
•     lcd.begin (16 ,2);
•     lcd.setBacklight(HIGH);
•     lcd.noAutoscroll();
•     lcd.init();
•     // Definindo o modo dos pinos dos Leds como saída.
•     for (int i = 0; i < N; i++) {
•         pinMode(pinosLeds[i], OUTPUT);
•     }
•     // Definindo o modo dos pinos dos Botões como entrada.
•     for (int i = 0; i < N+1; i++) {
•         pinMode(pinosBotoes[i], INPUT_PULLUP);
•     }
•     // Definindo o modo do pino de Áudio como saída.
•     pinMode(pinoAudio, OUTPUT);
•
•     // Inicializando o random através de uma leitura da porta analógica.
•     // Esta leitura gera um valor variável entre 0 e 1023.
•     randomSeed(analogRead(0));
•     //verificar reset do recorde
•     if (digitalRead(pinosBotoes[N]) == LOW) //botão reset
{
•         EEPROM.write(endereco_eeprom_recorde, 0);
•         lcd.clear();
•         lcd.setCursor(0,0);
•         lcd.print("RECORDE RESETADO");
•         tocarSomDeInicio();
•         delay(500);
•         lcd.clear();
•         delay(500);
•     }
• }
```

```
•     void loop() {
•         // Se perdeu o jogo reinicializamos todas as variáveis.
•         if (perdeu_o_jogo) {
•             int sequencia[100] = {};
•             rodada_atual = 0;
•             passo_atual_na_sequencia = 0;
•             perdeu_o_jogo = false;
•             // Exibe mensagem no lcd
•             lcd.clear();
•             lcd.setCursor(0,0);
•             lcd.print(" _-_GENIUS_-_-");
•             lcd.setCursor(0,1);
•             lcd.print("jogo_de_memoria");
•         }
•         delay(1000);
•         // Toca um som de início para anunciar que o jogo está começando quando é
•         // a primeira rodada.
•
•         if (rodada_atual == 0) {
•             tocarSomDeInicio();
•             delay(500);
•         }
•         lcd.clear();
•         delay(500);
•
•         // Chama a função que inicializa a próxima rodada.
•         proximaRodada();
•         // Reproduz a sequência atual.
•         reproduzirSequencia();
•         // Aguarda os botões serem pressionados pelo jogador.
•         aguardarJogador();
•
•         lcd.clear();
•     }
•     // Sorteia um novo item e adiciona na sequência.
•     void proximaRodada() {
•         int numero_sorteado = random(0, N);
•         sequencia[rodada_atual++] = numero_sorteado;
•     }
•     // Reproduz a sequência para ser memorizada.
•     void reproduzirSequencia() {
•         for (int i = 0; i < rodada_atual; i++) {
•             tone(pinoAudio, tons[sequencia[i]]);
•             digitalWrite(pinLeds[sequencia[i]], HIGH);
•             delay(500);
•             noTone(pinoAudio);
•             digitalWrite(pinLeds[sequencia[i]], LOW);
•             delay(100);
•         }
•         noTone(pinoAudio);
•     }
•
•     // Aguarda o jogador iniciar sua jogada.
```

```

•     void aguardarJogador() {
•         for (int i = 0; i < rodada_atual; i++) {
•             aguardarJogada();
•             verificarJogada();
•
•             if (perdeu_o_jogo) {
•                 EEPROM.write(endereco_eeprom_recorde, pontuacao);
•                 lcd.setCursor(0,0);
•                 lcd.print("NOVO RECORDISTA!");
•             }
•             lcd.clear();
•             lcd.setCursor(0,0);
•             lcd.print("--RESULTADO--");
•             lcd.setCursor(0,1);
•             lcd.print("PONTOS=");
•             lcd.setCursor(7,1);
•             lcd.print(pontuacao);
•             lcd.setCursor(10,1);
•             lcd.print("REC=");
•             lcd.setCursor(14,1);
•             lcd.print(EEPROM.read(endereco_eeprom_recorde));
•             delay(3000);
•             pontuacao = 0;
•             return;
•         }
•         passo_atual_na_sequencia++;
•     }
•     pontuacao++;
•     // Redefine a variável para 0.
•     passo_atual_na_sequencia = 0;
• }

•     void aguardarJogada() {
•         lcd.clear();
•         lcd.setCursor(0,0);
•         lcd.print("    Sigam-me ");
•         lcd.setCursor(0,1);
•         lcd.print("    os bons..");
•         lcd.setCursor(16,1);
•         lcd.autoscroll(); // Seta o display para scroll automático
•         boolean jogada_efetuada = false;
•         while (!jogada_efetuada) {
•             for (int i = 0; i < N; i++) {
•                 if (digitalRead(pinobotoes[i]) == LOW) {
•                     // Dizendo qual foi o botão pressionado.
•                     botao_pressionado = i;
•                     tone(pinoAudio, tons[i]);
•                     digitalWrite(pinoleds[i], HIGH);
•                     delay(300);
•                     digitalWrite(pinoleds[i], LOW);
•                     noTone(pinoAudio);
•                     jogada_efetuada = true;
•                 }
•             }
•         }
•     }

```

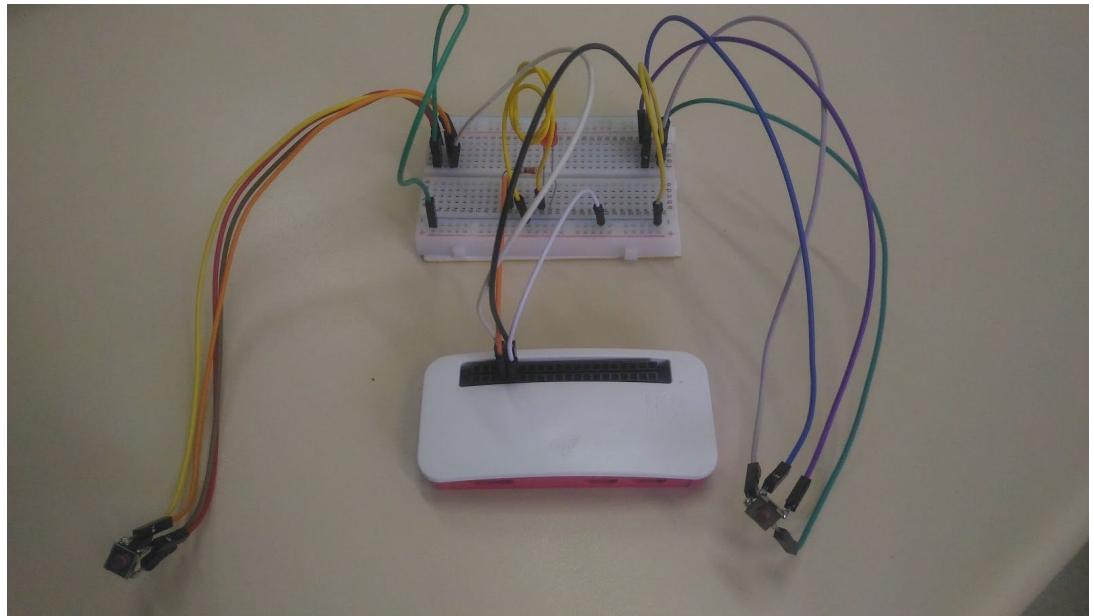
```
•     }
•     delay(300);
•     lcd.setCursor(16,1);
•     lcd.print(" ");
•     lcd.setCursor(16,0);
•     lcd.print(" ");
•   }
•   lcd.noAutoscroll();
• }

• void verificarJogada() {
•   lcd.clear();
•   lcd.setCursor(0,1);
•   lcd.print("jogo_de_memoria");
•   lcd.setCursor(0,0);

•   if((sequencia[passo_atual_na_sequencia] == botao_pressionado))
•     lcd.print(" __CONTINUE__");
•   else { // GAME OVER
•     lcd.print(" __ERROU__");
•     for (int i = 0; i < N; i++) {
•       tone(pinoAudio, tons[i]);
•       digitalWrite(pinossLeds[i], HIGH);
•       delay(200);
•       digitalWrite(pinossLeds[i], LOW);
•       noTone(pinoAudio);
•     }
•     tone(pinoAudio, tons[3]);
•     for (int j = 0; j < N; j++) {
•       for (int i = 0; i < N; i++)
•         digitalWrite(pinossLeds[i], HIGH);
•       delay(100);
•       for (int i = 0; i < N; i++)
•         digitalWrite(pinossLeds[i], LOW);
•     }
•     noTone(pinoAudio);
•     perdeu_o_jogo = true;
•   }
•   delay(200);
• }
void tocarSomDeInicio() {
  tone(pinoAudio, tons[0]);
  digitalWrite(pinossLeds[0], HIGH);
  digitalWrite(pinossLeds[1], HIGH);
  digitalWrite(pinossLeds[2], HIGH);
  digitalWrite(pinossLeds[3], HIGH);
  delay(500);
  digitalWrite(pinossLeds[0], LOW);
  digitalWrite(pinossLeds[1], LOW);
  digitalWrite(pinossLeds[2], LOW);
  digitalWrite(pinossLeds[3], LOW);
  delay(500);
  noTone(pinoAudio);
}
```

# Projetos com Raspberry PI

## Jogo do Botão



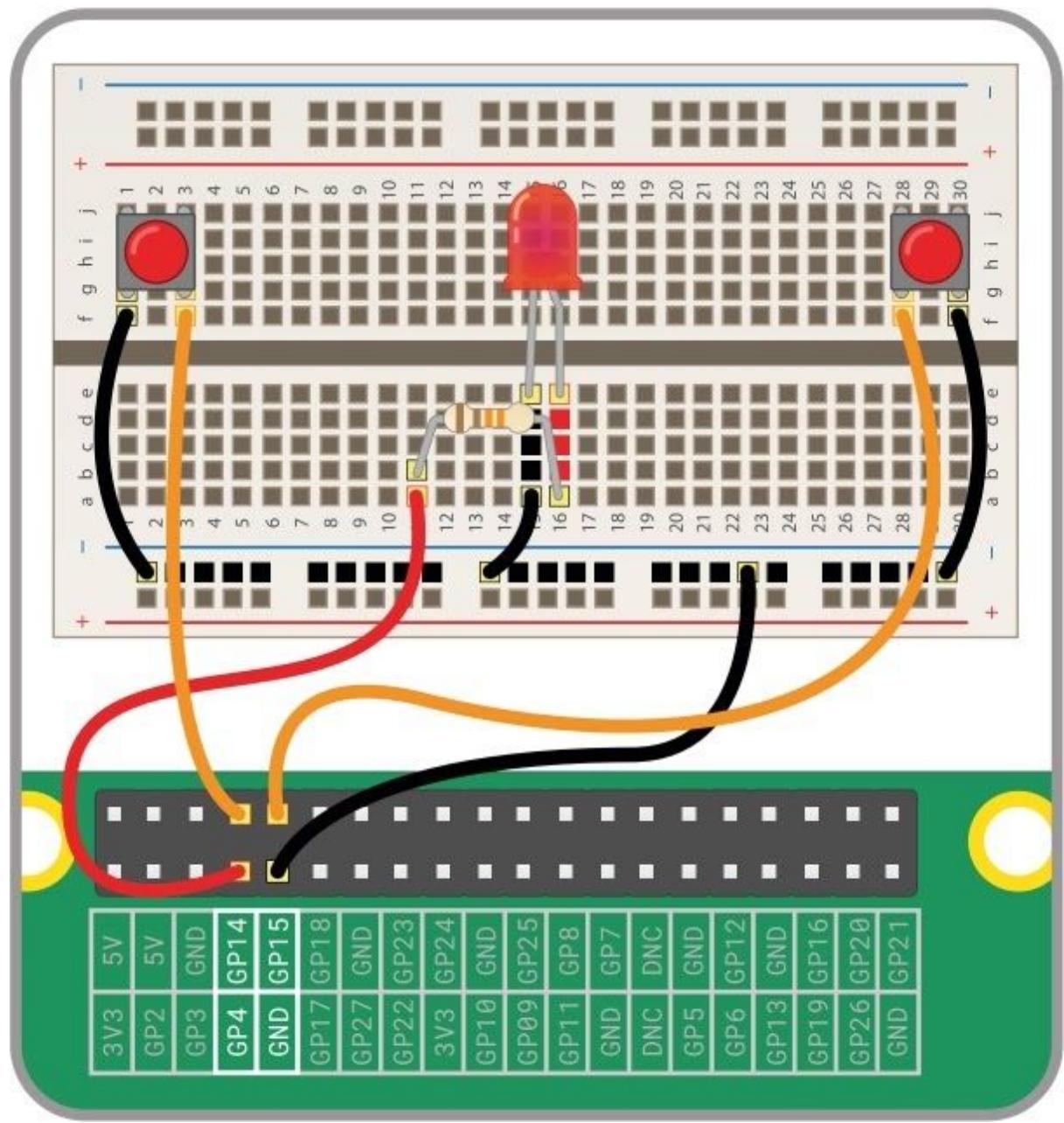
**Descrição:** Pequeno código em Python, desenvolvido no Raspberry Pi Zero. Jogo onde dois botões são dispostos a dois jogadores distintos.

Objetivos do jogo: após entrada do nome dos jogadores em tela, um LED acende e o mesmo irá apagar dentro de um intervalo uniforme de cinco a dez segundos. Após o LED apagar, o primeiro jogador a apertar o seu botão vence, com impressão em tela da mensagem “NOME DO JOGADOR venceu o jogo!”

### Componentes necessários

- 1 Unidade – Raspberry Pi (qualquer modelo);
- 2 Unidades – Botão Push Button (Chave Táctil);
- 1 Unidade – Protoboard;
- 1 Unidade – Resistor;
- 1 Unidade – LED;
- 1 Unidade – Cabo HDMI;
- 1 Ambiente Desktop (mouse USB, teclado USB e monitor HDMI);
- Conjunto de cabos Jumper macho e fêmea.

## Diagrama



**Fonte:** Raspberry Pi Beginners Guide v1, 2018, p. 146.

## Montagem

- 1)** Conecte os botões, o LED e resistor na Protoboard
- 2)** Conecte jumpers da Protoboard em ligamento aos botões, ao LED e a GPIO (pino GND) do Raspberry Pi a faixa de barramento negativa da Protoboard
- 3)** Conecte jumpers da Protoboard em ligamento aos botões e ao resistor ao GPIO (pinos GPIO) do Raspberry Pi
- 4)** Conecte o Raspberry Pi ao mouse, teclado, monitor e fonte de energia
- 5)** Ao inicializar o Sistema Operacional do Raspberry Pi abrir o menu de ferramentas encontrado na Área de Trabalho, abrir a guia de Desenvolvimento e abrir a aplicação Python 3 (IDLE)
- 6)** Dentro da Shell do Python 3, entrar em File -> New File (Ctrl+N)
- 7)** Desenvolver no Python 3 o código abaixo, salvar e entrar em Run -> Run Module (F5)
- 8)** Dentro da Shell do Python 3, entrar pelo teclado o nome dos jogadores. O LED irá acender. Quando o LED apagar e o jogo for concluído, uma mensagem será exibida na tela

## Código utilizado

```
●   from gpiozero import LED, Button
●   from time import sleep
●   from random import uniform
●   from os import _exit
●
●   print("Bem-vindo ao jogo do botão! O primeiro jogador a pressionar o
      botão vence o jogo!")
●   \nEntre com o nome dos jogadores:\n\n")
●   jogador_esquerda = input("Nome do primeiro jogador: ")
●   jogador_direita = input("Nome do segundo jogador: ")
●
●   led = LED(4)
●   botao_esquerdo = Button(14)
●   botao_direito = Button(15)
●   print("\nPreparar...\n")
●   sleep(1)
●   print("Apontar...\n")
●   led.on()
●   sleep(uniform(5, 10))
●   led.off()
●   print("Agora! Pressione o botão!\n\n")
●
●   def apertou(button):
●       if button.pin.number == 14:
●           print(jogador_esquerda + " venceu o jogo!")
●       else:
●           print(jogador_direita + " venceu o jogo!")
●       _exit(0)
●
●   botao_esquerdo.when_pressed = apertou
●   botao_direito.when_pressed = apertou
```

## Theremin Ultrassônico

**Descrição:** Um Theremin Ultrassônico com código desenvolvido em Python e Sonic Pi, no Raspberry Pi 3.

Um Theremin é:

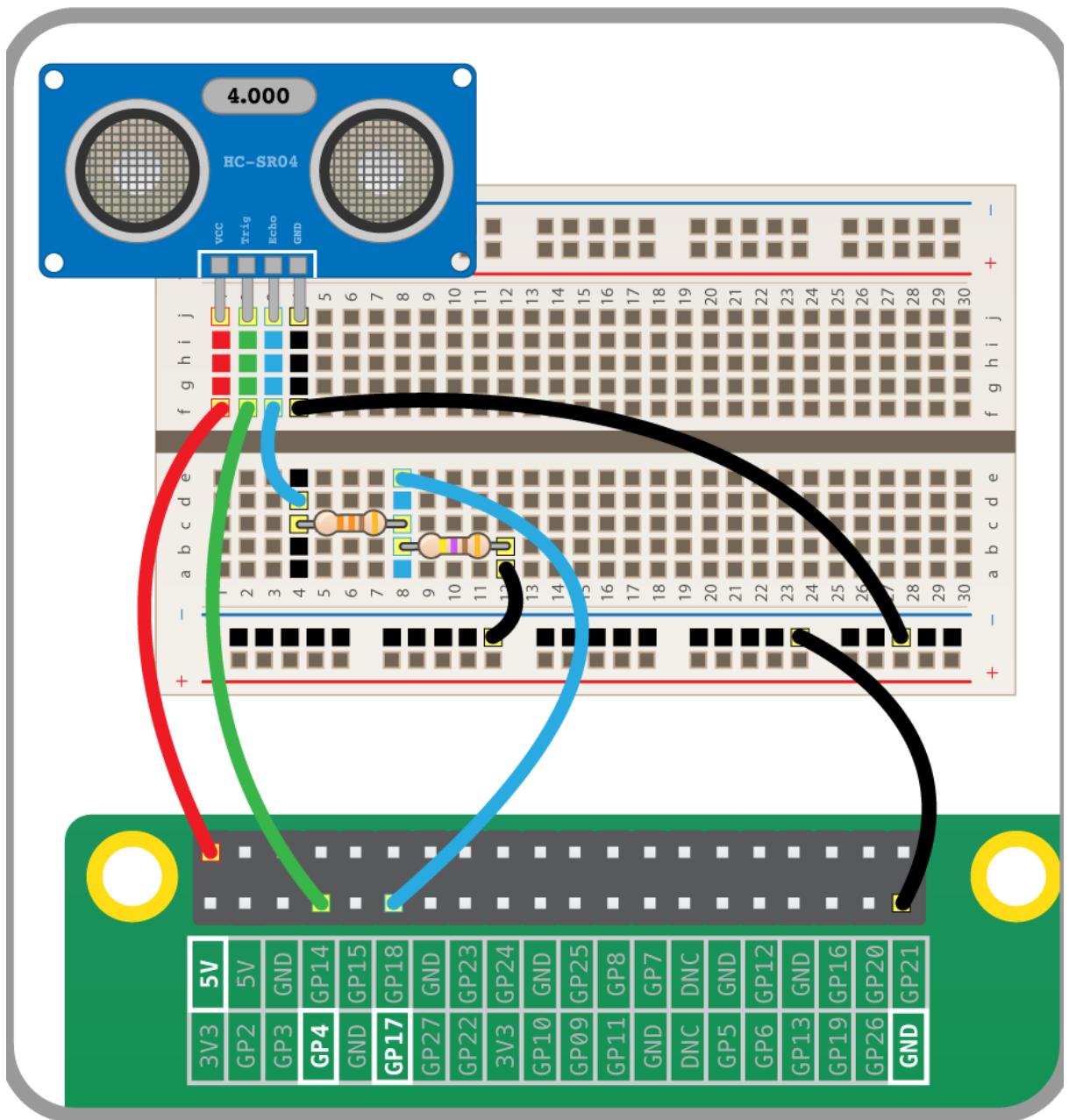
Um sensor ultrasônico emite ondas sonoras de alta frequência, que ao entrar em contato com algo material rebate eco, do qual é convertido em sinais elétricos. Através destes sinais, é possível calcular, com distância de até um metro, a que distância do sensor um corpo físico se encontra.

No Python, o código desenvolvido pega esta distância calculada, multiplica por 100 (cem) e soma mais 50 (cinquenta) e devolve, em intervalos de meio milésimo de segundo (0,05s), estes números para o software Sonic Pi. O Sonic Pi possui registros de diversos sons, em escala, correspondentes a números inteiros não negativos, com uma gama maior que uma centena de sons distintos. Cada vez que um valor é enviado ao Sonic Pi, ele retorna o som correspondente a aquele valor.

### Componentes necessários

- 1 Unidade – Raspberry Pi (qualquer modelo);
- 1 Unidade – Fonte de energia;
- 1 Unidade – Sensor de Distância Ultrasônico;
- 1 Unidade – Resistor 330Ω;
- 1 Unidade – Resistor 470Ω;
- 1 Unidade – Protoboard;
- 1 Unidade – Cabo HDMI;
- 1 Ambiente Desktop (mouse USB, teclado USB e monitor HDMI);
- Conjunto de cabos Jumper macho e fêmea.

## Diagrama



**Fonte:** <https://projects.raspberrypi.org/en/projects/ultrasonic-theremin/4>

## Montagem

- 1)** Conecte o sensor ultrassônico e os resistores na Protoboard
- 2)** Conecte jumpers da Protoboard em ligamento aos quatro pinos do sensor ultrasônico, aos resistores e a GPIO (pino GND) do Raspberry Pi a faixa de barramento negativa da Protoboard
- 3)** Conecte jumpers da Protoboard em ligamento ao sensor ultrasônico e aos resistores ao GPIO (pinos GPIO) do Raspberry Pi
- 4)** Conecte o Raspberry Pi ao mouse, teclado, monitor e fonte de energia
- 5)** Ao inicializar o Sistema Operacional do Raspberry Pi abrir o menu de ferramentas encontrado na Área de Trabalho, abrir a guia de Desenvolvimento e abrir as aplicações Python 3 (IDLE) e Sonic Pi
- 6)** Dentro do Sonic Pi, desenvolver o código abaixo e apertar o botão Run
- 7)** Dentro da Shell do Python 3, entrar em File -> New File (Ctrl+N)
- 8)** Desenvolver no Python 3 o código abaixo, salvar e entrar em Run -> Run Module (F5)
- 9)** O theremin está pronto para testes. Experimente mover sua mão próximo ao sensor ultrassônico, aproximando-a e afastando-a

## Código utilizado

Vamos precisar das últimas versões do Sonic Pi e python-osc, então precisamos rodar estas duas linhas dentro do terminal:

```
sudo apt update && sudo apt upgrade -y  
sudo pip3 install python-osc
```

### No Sonic Pi:

```
• live_loop :listen do  
•     use_real_time  
•     note = sync "/osc/play_this"  
•     play note[0]  
•     end
```

### No Python:

```
• from gpiozero import DistanceSensor  
• from time import sleep  
• from pythonosc import osc_message_builder  
• from pythonosc import udp_client  
•  
• sensor = DistanceSensor(echo=17, trigger=4)  
• sender = udp_client.SimpleUDPClient('127.0.0.1', 4559)  
•  
• while True:  
•     tonalidade = round(sensor.distance*100+50)  
•     sender.send_message('play_this', tonalidade)  
•     sleep(0.05)
```

## Controle de PlayStation 3 no Raspberry Pi 4



### Preparando o Raspberry

Primeiro, certifique-se de que o Raspberry está atualizado. Para isso, use os seguintes comandos no terminal:

1. sudo apt-get update
2. sudo apt-get upgrade

Instale os pacotes necessários:

1. sudo apt-get install libusb-dev

Baixando o código necessário e salvando em uma pasta:

1. mkdir ~/sixpair
2. cd ~/sixpair
3. wget <http://www.pabr.org/sixlinux/sixpair.c>

Compile o código baixado na etapa anterior:

1. gcc -o sixpair sixpair.c -lusb

Conekte o joystick usando o cabo USB e então execute o seguinte comando para iniciar a configuração do joystick

1. sudo ~/sixpair/sixpair

A seguinte mensagem será exibida. No lugar de “ENDERECO\_MAC” será exibido o endereço MAC do joystick. Salve o endereço pois ele será usado nos próximos passos

1. Current Bluetooth master: ENDEREKO\_MAC
2. Setting master bd\_addr to ENDEREKO\_MAC

Desconecte o joystick.

Baixe o pacote do bluetoothctl:

1. sudo bluetoothctl

Execute os seguintes comandos:

1. agent on
2. **default-agent**

Execute o comando para procurar dispositivos disponíveis

1. scan on

Após a listagem de dispositivos aparecer, aperte o botão com o logo do Playstation no centro do joystick para iniciar a comunicação com o Raspberry. Um novo dispositivo irá aparecer na lista.

Exemplo:

1. [NEW] Device ENDEREKO\_MAC B8-27-EB-A4-59-08
2. [CHG] Device ENDEREKO\_MAC Connected: no
3. [DEL] Device ENDEREKO\_MAC B8-27-EB-A4-59-08

Execute o seguinte comando, substituindo ENDEREKO\_MAC com o endereço já encontrado anteriormente. Pode ser necessário executar o comando várias vezes. Certifique-se de que o joystick está ligado.

1. connect ENDEREKO\_MAC

A conexão será bem-sucedida quando a seguinte mensagem aparecer:

1. Attempting to connect to ENDEREKO\_MAC
2. [CHG] Device ENDEREKO\_MAC Modalias: usb:v054Cp0268d0100
3. [CHG] Device ENDEREKO\_MAC UUIDs:
4. 00001124-0000-1000-8000-00805f9b34fb
5. 00001200-0000-1000-8000-00805f9b34fb

Adicione o joystick a lista de dispositivos confiáveis:

1. trust ENDEREKO\_MAC

Aparecerá a seguinte mensagem:

1. [CHG] Device ENDERECHO\_MAC Trusted: yes
2. Changing ENDERECHO\_MAC trust succeeded

Você agora pode fechar o terminal e reiniciar o Raspberry Pi.

Agora quando você precisar usar o joystick, pressione o botão do meio até os leds acenderem. A conexão é bem-sucedida quando apenas um led fica aceso.

## Mapeando o joystick

Após conectar o joystick ao Raspberry, execute esse código primeiro e pressione os botões e gatilhos. Uma mensagem exibindo um número será exibida a cada ação feita. Anote os números e seus respectivos botões.

```
•   from evdev import InputDevice, categorize, ecodes
•
•
•   #cria um objeto "gamepad" para armazenar os dados
•   gamepad = InputDevice('/dev/input/event2')
•
•   #informações do dispositivo
•   print(gamepad)
•
•   for event in gamepad.read_loop():
•       #filtrando os eventos
•       if event.type == ecodes.EV_KEY:
•           print(event)
```

Este código serve para mapear os eventos dos botões analógicos.

```
•   from evdev import InputDevice, categorize, ecodes, KeyEvent
•
•
•   gamepad = InputDevice('/dev/input/event2')
•
•   #mostra informações do dispositivo
•   print(gamepad)
•   last = {
•       "ABS_Y": 128,
•       "ABS_X": 128,
•       "ABS_RX": 128,
•       "ABS_RY": 128
•   }
•   for event in gamepad.read_loop():
•       if event.type == ecodes.EV_ABS:
```

```

•         absevent = categorize(event)
•             if ecodes.bytype[absevent.event.type][absevent.event.code] ==
'ABS_Y':
•                 last["ABS_Y"] = absevent.event.value
•
•             if ecodes.bytype[absevent.event.type][absevent.event.code] ==
'ABS_X':
•                 last["ABS_X"] = absevent.event.value
•
•                 if last["ABS_Y"] > 128:
•                     print ('Analogico esquerdo para baixo');
•                     print (last["ABS_Y"]);
•                 elif last["ABS_Y"] < 127:
•                     print ('Analogico esquerdo para cima');
•                     print (last["ABS_Y"]);
•
•                 if last["ABS_X"] > 128 :
•                     print ('Analogico esquerdo para direita');
•                     print (last["ABS_X"]);
•                 elif last["ABS_X"] < 127:
•                     print ('Analogico esquerdo para esquerda');
•                     print (last["ABS_X"]);
•
•                     if ecodes.bytype[absevent.event.type][absevent.event.code] ==
'ABS_RY':
•                         last["ABS_RY"] = absevent.event.value
•
•                         if ecodes.bytype[absevent.event.type][absevent.event.code] ==
'ABS_RX':
•                             last["ABS_RX"] = absevent.event.value
•
•                             if last["ABS_RY"] > 128:
•                                 print ('Analogico direito para baixo');
•                                 print (last["ABS_RY"]);
•                             elif last["ABS_RY"] < 127:
•                                 print ('Analogico direito para cima');
•                                 print (last["ABS_RY"]);
•
•                             if last["ABS_RX"] > 128 :
•                                 print ('Analogico direito para direita');
•                                 print (last["ABS_RX"]);
•                             elif last["ABS_RX"] < 127:
•                                 print ('Analogico direito para esquerda');
•                                 print (last["ABS_RX"]);

```

Após executar os códigos anteriores e anotar os eventos, substitua os números nas linhas 26 a 40 com os respectivos valores. Execute o código e verifique se todas as ações foram mapeadas.

```

•     from evdev import InputDevice, categorize, ecodes
•
•
•     #localiza o joystick
•     numeroEvento = 2
•     enderecoJoystick = None
•     contador = 0
•
•     try:
•         while contador <= 10:
•             endereco = '/dev/input/event' + str(numeroEvento)
•             gamepad = InputDevice(endereco)
•             if "Sony PLAYSTATION(R)3 Controller" in str(gamepad) and not
•                 "Motion Sensors" in str(gamepad):
•                 print("Joystick detectado")
•                 enderecoJoystick = ('/dev/input/event' + str(numeroEvento))
•                 break
•             else:
•                 contador += 1
•                 numeroEvento += 1
•             if contador == 10:
•                 print("joystick nao detectado")
•
•     except:
•         print("No such file or directory")
•
•
•     gamepad = InputDevice(enderecoJoystick)
•     print(gamepad)
•
•
•     cima = 544;
•     baixo = 545;
•     esquerda = 546;
•     direita = 547;
•
•     triangulo = 307;
•     quadrado = 308;
•     circulo = 305;
•     x = 304;
•
•     start = 315;
•     select = 314;
•
•     gatEsquerda = 310 ;
•     gatDireita = 311;
•
•
•     last = {
•         "ABS_Y": 128,
•         "ABS_X": 128,
•         "ABS_RX": 128,
•         "ABS_RY": 128
•     }
•
•
•     for event in gamepad.read_loop():
•         #ANALOGICO
•         if event.type == ecodes.EV_ABS:
•             absevent = categorize(event)

```

```

#ANALOGICO ESQUERDO
if ecodes.bytype[absevent.event.type][absevent.event.code] ==
'ABS_Y':
    last["ABS_Y"] = absevent.event.value

if ecodes.bytype[absevent.event.type][absevent.event.code] ==
'ABS_X':
    last["ABS_X"] = absevent.event.value

if last["ABS_Y"] > 128:
    print ('Analogico esquerdo para baixo');
    print (last["ABS_Y"]);
elif last["ABS_Y"] < 127:
    print ('Analogico esquerdo para cima');
    print (last["ABS_Y"]);

if last["ABS_X"] > 128 :
    print ('Analogico esquerdo para direita');
    print (last["ABS_X"]);
elif last["ABS_X"] < 127:
    print ('Analogico esquerdo para esquerda');
    print (last["ABS_X"]);

#ANALOGICO DIREITO
if ecodes.bytype[absevent.event.type][absevent.event.code] ==
'ABS_RY':
    last["ABS_RY"] = absevent.event.value

if ecodes.bytype[absevent.event.type][absevent.event.code] ==
'ABS_RX':
    last["ABS_RX"] = absevent.event.value

if last["ABS_RY"] > 128:
    print ('Analogico direito para baixo');
    print (last["ABS_RY"]);
elif last["ABS_RY"] < 127:
    print ('Analogico direito para cima');
    print (last["ABS_RY"]);

if last["ABS_RX"] > 128 :
    print ('Analogico direito para direita');
    print (last["ABS_RX"]);
elif last["ABS_RX"] < 127:
    print ('Analogico direito para esquerda');
    print (last["ABS_RX"]);

#BOTOES E GATILHOS
if event.type == ecodes.EV_KEY:
    if event.value == 1:
        if event.code == cima:
            print("Cima")
        elif event.code == baixo:
            print("Baixo")
        elif event.code == esquerda:

```

```
•         print("Esquerda")
•     elif event.code == direita:
•         print("Direita")
•
•     elif event.code == x:
•         print("X")
•     elif event.code == triangulo:
•         print("Triangulo")
•     elif event.code == quadrado:
•         print("Quadrado")
•     elif event.code == circulo:
•         print("Circulo")
•
•     elif event.code == start:
•         print("Start")
•     elif event.code == select:
•         print("Select")
•
•     elif event.code == gatEsquerda:
•         print("Gatilho esquierdo");
•     elif event.code == gatDireita:
•         print("Gatilho direito");
```

## Carrinho de controle remoto auto dirigível



**Descrição:** Um carrinho controlado por um joystick de PlayStation 3 que também dirige sozinho

### Componentes necessários

1 Unidade – Raspberry Pi 4

1 Unidade - Joystick de PlayStation 3 já mapeado (ver projeto “Controle de PlayStation 3 no Raspberry Pi 4”)

1 Unidade – Módulo de câmera v2

1 Unidade – Powerbank

2 Unidades – Motores DC

4 Unidades – Rodas

1 Unidade – Driver de controle L298N

1 Unidade – Carcaça

Conjunto de cabos jumper fêmea-fêmea e macho-fêmea

## Montagem

- 1) Coloque o Powerbank em cima da base do carrinho, usando fita adesiva para segura-lo
- 2) Coloque o Raspberry Pi 4 em cima do Powerbank
- 3) Conecte o driver de controle L298N ao Raspberry Pi 4
- 4) Conecte os motores ao driver de controle e coloque-as nas extremidades do carrinho, e então coloque as rodas
- 5) Verifique se tudo está bem conectado, e então coloque a carcaça
- 6) Conecte a câmera ao Raspberry Pi 4 e coloque-a em cima da carcaça, em um lugar onde ela consiga ver o que está na frente do carrinho

## Código utilizado

Antes de executar o código, ligue o joystick pois caso contrário o programa irá finalizar sua execução.

Para que ele dirija sozinho, monte um caminho usando duas faixas com uma cor contrastante (fita isolante em um chão branco, etc para delimitar a pista. Ele usará a câmera para tirar fotos e então verificar se deve ir reto ou fazer uma curva para esquerda ou direita.

```
•  #!/usr/bin/python
•  import RPi.GPIO as GPIO
•  from time import sleep
•  from PIL import Image
•  import picamera.array
•  import numpy as np
```

```

•     import picamera
•     import _thread
•     import curses
•     import argparse
•     import sys
•     import csv
•     import os
•     from evdev import InputDevice, categorize, ecodes
•     from gpiozero import Motor, Servo
•
•
•     GPIO.setmode(GPIO.BCM)
•     GPIO.setwarnings(False)
•
•
•     contador = 0
•
•     try:
•         numeroEvento = 2
•         enderecoJosystick = None
•         while contador <= 10:
•             endereco = '/dev/input/event' + str(numeroEvento)
•             gamepad = InputDevice(endereco)
•             if "Sony PLAYSTATION(R)3 Controller" in str(gamepad) and not
•             "Motion Sensors" in str(gamepad):
•                 print("Joystick detectado")
•                 enderecoJoystick = ('/dev/input/event' + str(numeroEvento))
•                 gamepad = InputDevice(enderecoJoystick)
•                 print(gamepad)
•                 break
•             else:
•                 contador += 1
•                 numeroEvento += 1
•             if contador == 10:
•                 print("Joystick nao detectado")
•
•     except:
•         print("Joystick nao detectado, saindo...")
•         sys.exit()
•
•
•     select = 314;
•     last = {
•         "ABS_Y": 128,
•         "ABS_X": 128,
•         "ABS_RX": 128,
•         "ABS_RY": 128
•     }
•
•
•     class Camera:
•         imgsPath = "/media/pi/UUI1/"
•
•
•         def __init__(self):
•             self.camera = picamera.PiCamera()
•             self.camera.resolution = (800, 600)
•             self.camera.start_preview()
•             sleep(2)
•             self.stream = picamera.array.PiYUVArray(self.camera)

```

```

•     def capture(self):
•         self.camera.capture(self.stream, format='yuv')
•         img = self.stream.array[270:,:,0]
•         self.stream.seek(0)
•         self.stream.truncate()
•         return img
•     def save_img(self,img):
•         im = Image.fromarray(img)
•         imgPath = self.imgsPath + str(len(os.listdir(self.imgsPath))) +
".jpeg"
•         im.save(imgPath)
•     def preprocess_img(self,img):
•         b_min = 0
•         b_max = 135
•         b_binary = np.zeros_like(img)
•         b_binary[(img >= b_min) & (img <= b_max)] = 1
•         return b_binary
•
• class Controller:
•     Motor1Pin1 = 7
•     Motor1Pin2 = 24
•     Motor2Pin1 = 25
•     Motor2Pin2 = 8
•
•     def __init__(self):
•         #motores
•         GPIO.setup(self.Motor1Pin1, GPIO.OUT)
•         GPIO.setup(self.Motor1Pin2, GPIO.OUT)
•         GPIO.setup(self.Motor2Pin1, GPIO.OUT)
•         GPIO.setup(self.Motor2Pin2, GPIO.OUT)
•         self.direction = 0
•     #def front(self,f1,f2,f):
•     def front(self):
•         print("andando sozinho para frente")
•         GPIO.output(self.Motor1Pin1, GPIO.HIGH)
•         GPIO.output(self.Motor1Pin2, GPIO.LOW)
•     def rear(self):
•         print("andando sozinho para tras")
•         GPIO.output(self.Motor1Pin1, GPIO.LOW)
•         GPIO.output(self.Motor1Pin2, GPIO.HIGH)
•     def steering(self):
•         for event in gamepad.read_loop():
•             if event.type == ecodes.EV_ABS:
•                 absevent = categorize(event)
•                 if ecodes.bytype[absevent.event.type][absevent.event.code] ==
'ABS_Y':
•                     last["ABS_Y"] = absevent.event.value
•
•                     if ecodes.bytype[absevent.event.type][absevent.event.code] ==
'ABS_X':
•                         last["ABS_X"] = absevent.event.value
•
•                     if last["ABS_Y"] > 128:
•                         print ('Analogico esquerdo para baixo');

```

```

        GPIO.output(self.Motor1Pin1, GPIO.LOW)
        GPIO.output(self.Motor1Pin2, GPIO.HIGH)
        self.direction = "0"

    elif last["ABS_Y"] < 127:
        print ('Analogico esquerdo para cima');
        GPIO.output(self.Motor1Pin1, GPIO.HIGH)
        GPIO.output(self.Motor1Pin2, GPIO.LOW)
    if last["ABS_Y"] == 127:
        print("analogico esquerdo solto")
        GPIO.output(self.Motor1Pin1, GPIO.LOW)
        GPIO.output(self.Motor1Pin2, GPIO.LOW)
        GPIO.output(self.Motor2Pin1, GPIO.LOW)
        GPIO.output(self.Motor2Pin2, GPIO.LOW)
    if last["ABS_X"] > 128 :
        print ('Analogico esquerdo para direita');
        GPIO.output(self.Motor1Pin1, GPIO.HIGH)
        GPIO.output(self.Motor1Pin2, GPIO.LOW)
        GPIO.output(self.Motor2Pin1, GPIO.LOW)
        GPIO.output(self.Motor2Pin2, GPIO.HIGH)
        self.direction = "1"
    if last["ABS_X"] == 127:
        print("analogico dos lados solto")
    elif last["ABS_X"] < 127:
        print ('Analogico esquerdo para esquerda');
        GPIO.output(self.Motor1Pin1, GPIO.HIGH)
        GPIO.output(self.Motor1Pin2, GPIO.LOW)
        GPIO.output(self.Motor2Pin1, GPIO.LOW)
        GPIO.output(self.Motor2Pin2, GPIO.HIGH)
        self.direction = "2"

    class Collector:
        def __init__(self, fileName='steering.csv'):
            file = open(fileName, 'a')
            self.writer = csv.writer(file)
        def write(self, direction):
            self.writer.writerow(direction)

    def main():
        parser = argparse.ArgumentParser()
        group = parser.add_mutually_exclusive_group()
        group.add_argument("-d", "--drive", help="Dirigir sozinho",action="store_true")
        group.add_argument("-c", "--collect", help="Coleta imagens e armazena informacoes",action="store_true")
        args = parser.parse_args()
        if args.drive:
            return args
        elif args.collect:
            return args
        else:
            parser.print_help()
            sys.exit(2)

```

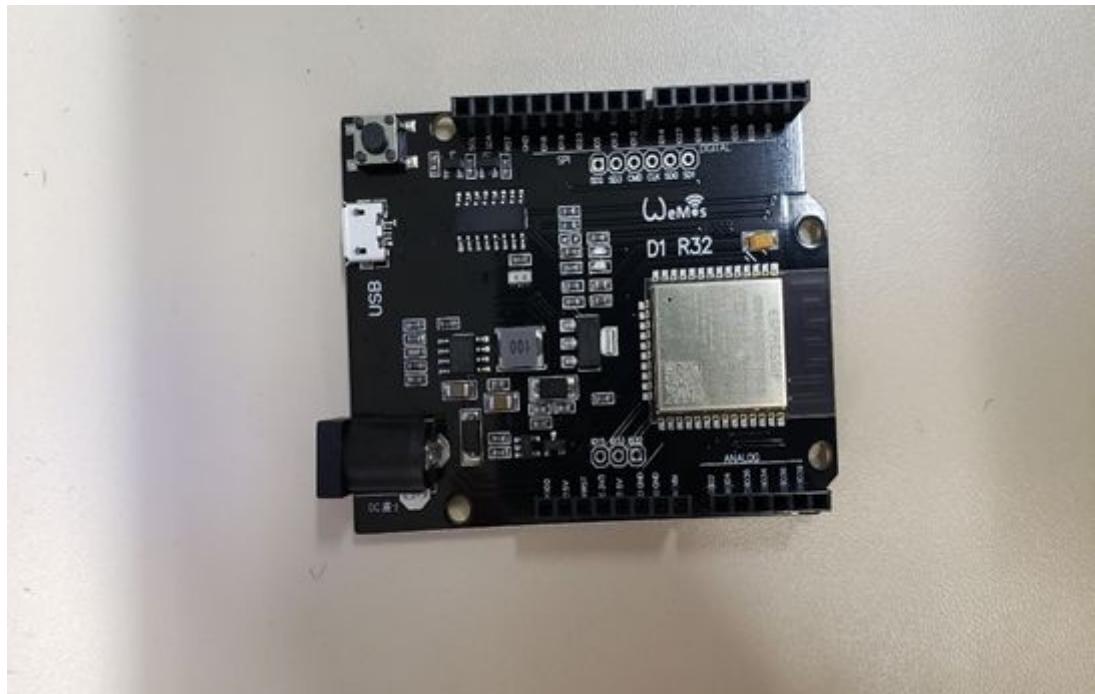
```
if __name__ == '__main__':
    args = main()
    try:
        screen = curses.initscr()
        curses.noecho()
        curses.cbreak()
        screen.keypad(True)
        carCtrl = Controller()
        carCam = Camera()

        if args.collect:
            carCol = Collector()
            _thread.start_new_thread(carCtrl.steering, ())
            while True:
                carCam.save_img(carCam.capture())
                carCol.write(str(carCtrl.direction))

        elif args.drive:
            carCtrl.rear()
            while True:
                img = carCam.preprocess_img(carCam.capture())
                histogram = np.sum(img[img.shape[0]//2:,:,:], axis=0)
                right = np.sum(histogram[220:], dtype=int)
                left = np.sum(histogram[:100], dtype=int)
                if (right - left) > 200 or left == 0:
                    print("virando para direita")
                    carCtrl.front()
                elif (right - left) < -200 or right == 0:
                    print("virando para direita")
                    carCtrl.front()
                else:
                    print("parando")

    except:
        curses.nocbreak()
        screen.keypad(0)
        curses.echo()
        curses.endwin()
        GPIO.cleanup()
        raise
```

## Conexão com Arduino esp32 e Firebase



**Descrição:** Uma forma de enviar dados para nuvem e receber com o esp32, utilizando o banco de dados em tempo real do Firebase.

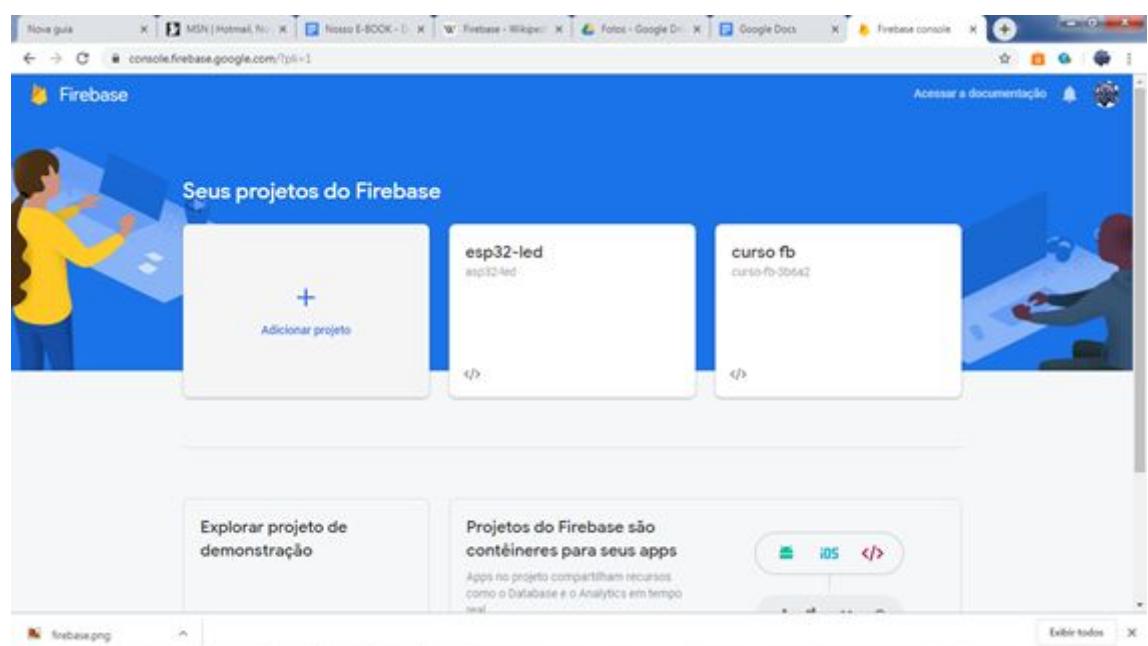
## Componentes necessários

1 Unidade – Arduino Esp32;

Uma conta no Google;

## Utilizando o Firebase

### 1- Adicione um projeto



## 2- Coloque o nome do projeto

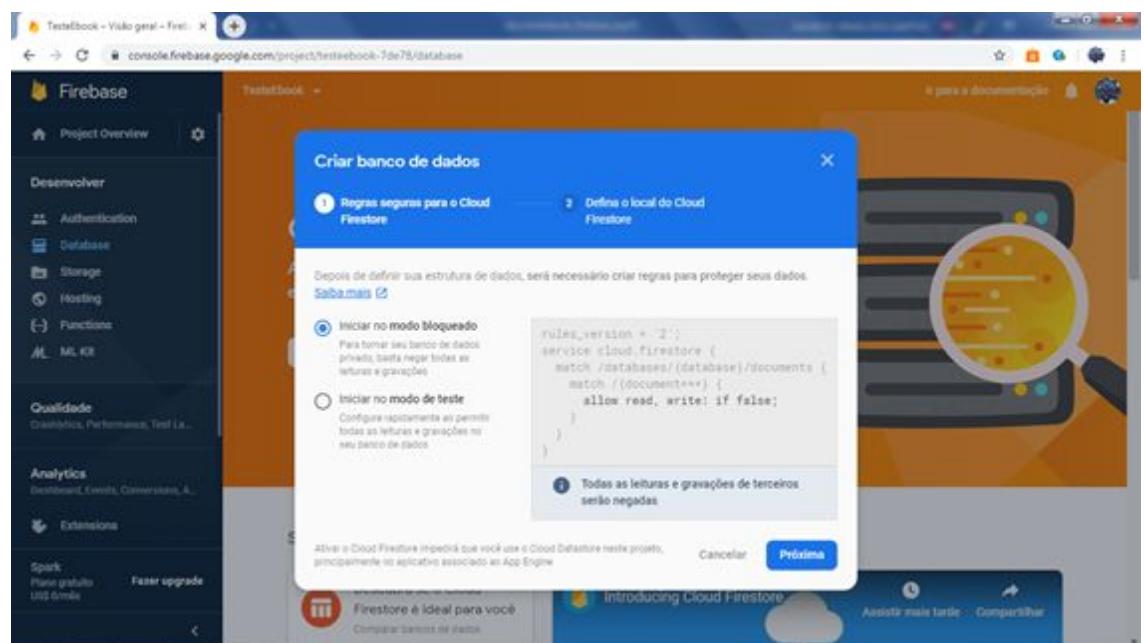


## 3- Criando um banco de dados

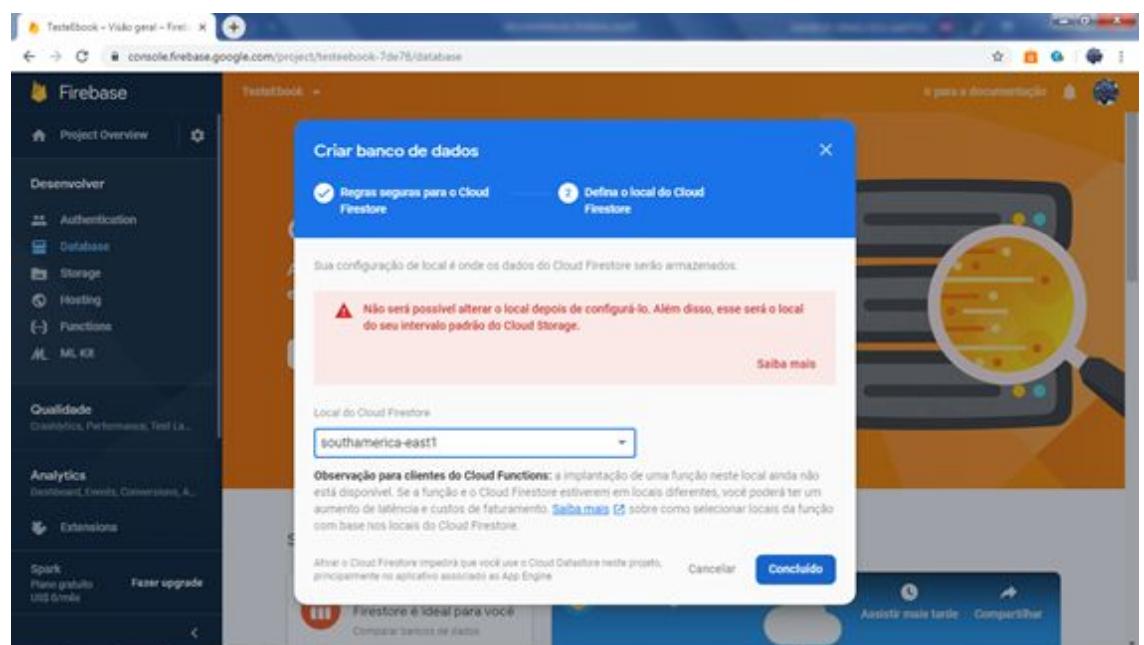
### 3.1

A screenshot of the Firebase Project Overview page for the 'TesteEbook' project. The left sidebar shows various services: Desenvolver (Authentication, Database, Storage, Hosting, Functions, ML Kit), Qualidade (Analytics, Performance, Test Lab), Analytics (Dashboard, Events, Conversions, etc.), Extensions, and Spark (Planos gratuitos, Fazer upgrade). The main content area is titled 'Cloud Firestore' and features the text 'Atualizações em tempo real, consultas eficientes e escalonamento automático'. It includes a 'Criar banco de dados' button and a 'Saiba mais' section. On the right, there's a large orange graphic with a magnifying glass over a database icon. At the bottom, there's a 'Descubra se o Cloud Firestore é ideal para você' section with a 'Comparar banco de dados' link, and a 'Introducing Cloud Firestore' video thumbnail with 'Assistir mais tarde' and 'Compartilhar' buttons. The URL in the address bar is 'console.firebaseio.google.com/project/testeebook-7de7b/database'.

### 3.2



### 3.3



# Configurando a IDE do Arduino para o esp32

## 1 - Baixando o drive da placa esp32

Caso seu sistema operacional não achar o drive do usb automaticamente, você terá que baixar manualmente.

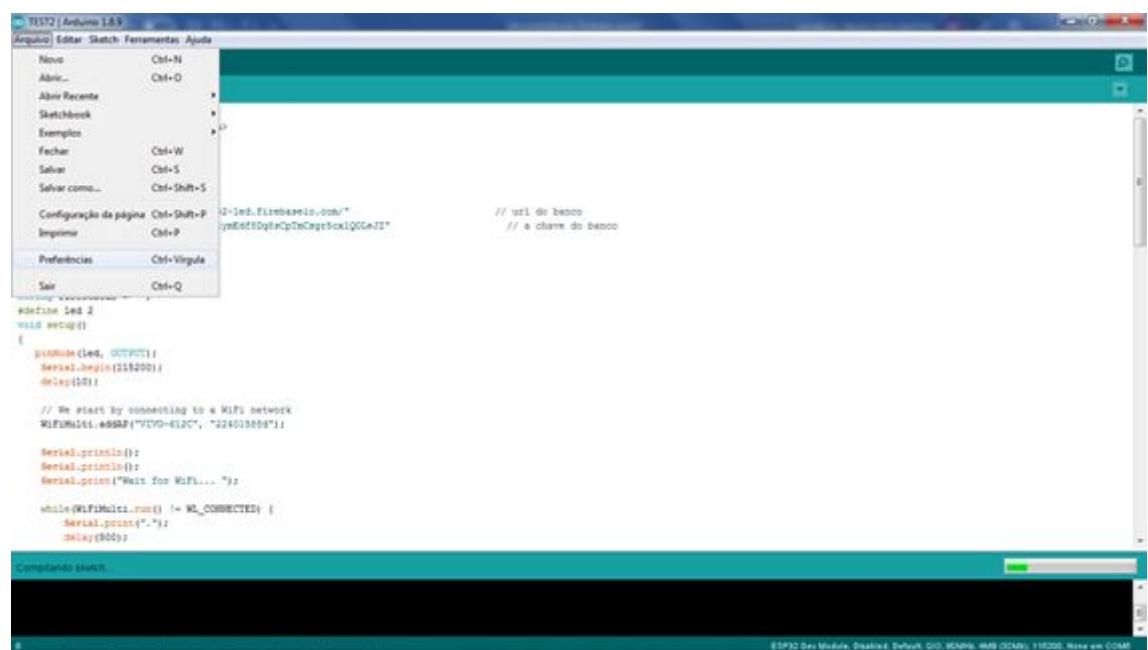
### Referências:

Link do drive da placa:

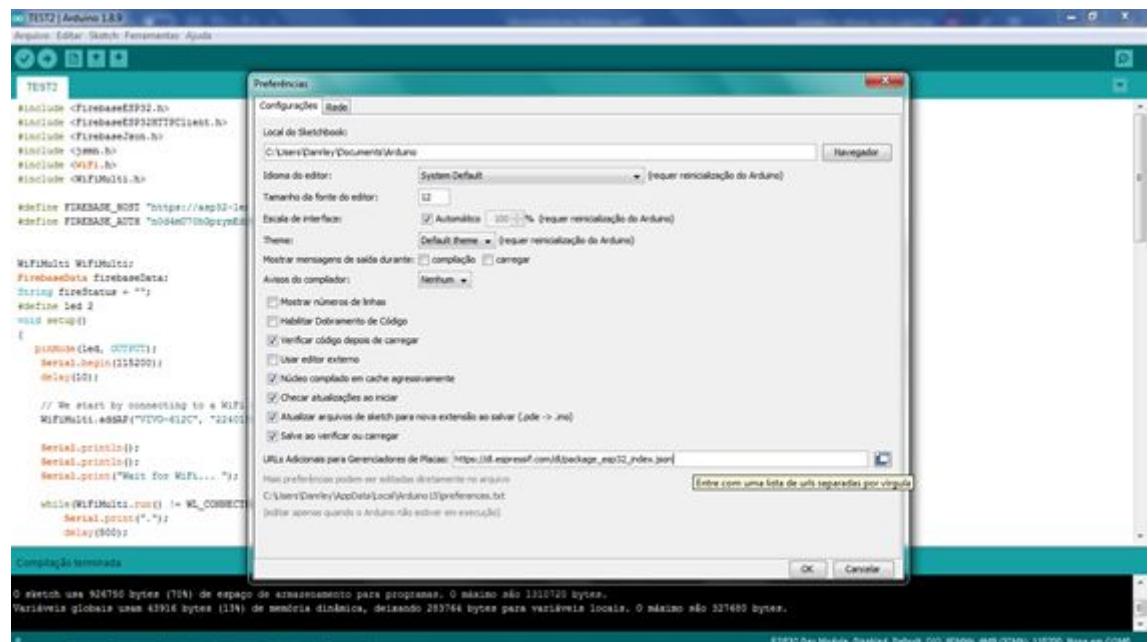
<http://blogmasterwalkershop.com.br/embarcados/wemos/wemos-d1-instalacao-no-windows/>

## 2 - Colocando a URL para gerenciar o esp 32

### 2.1



## 2.2

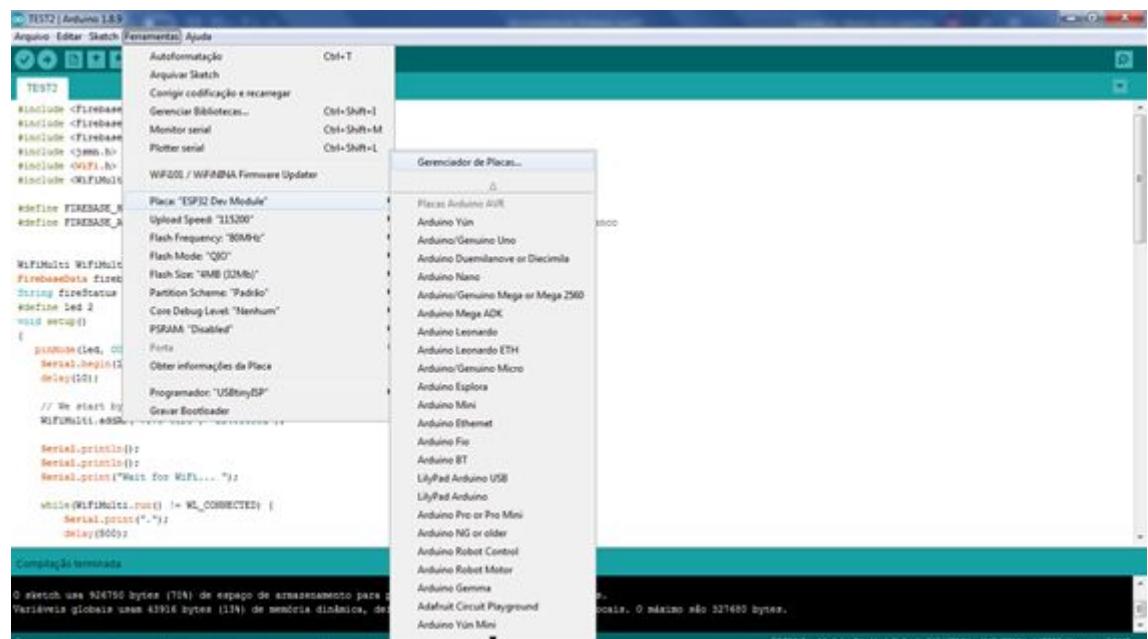


## Referência:

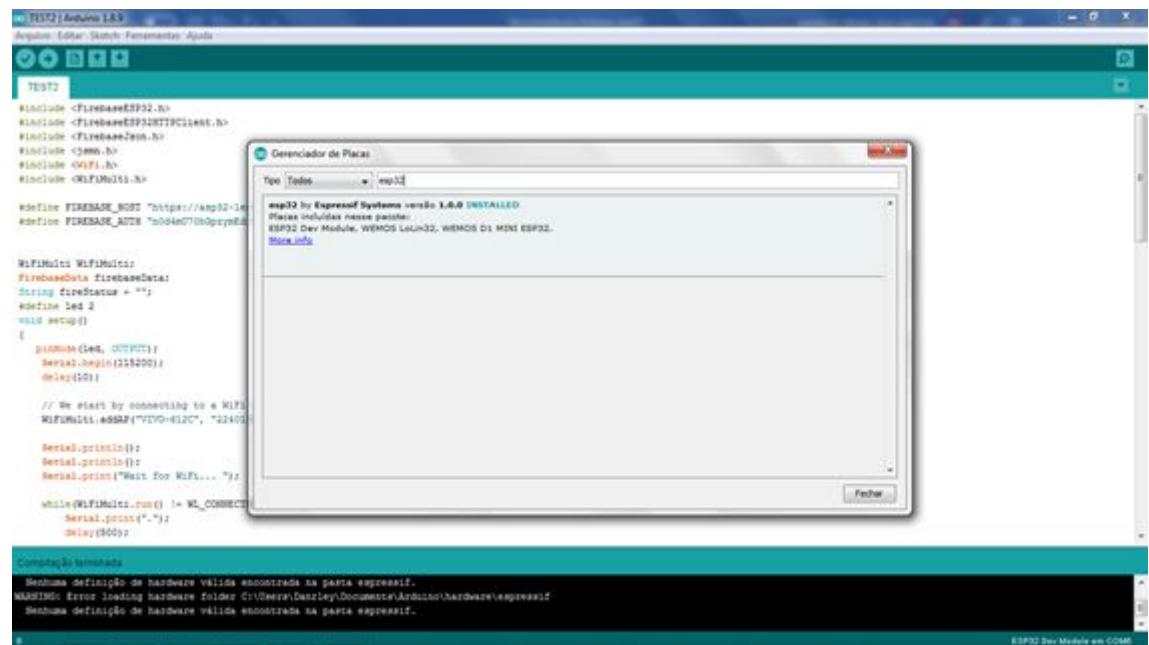
[https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json)

## 3 - Baixando a biblioteca do esp32

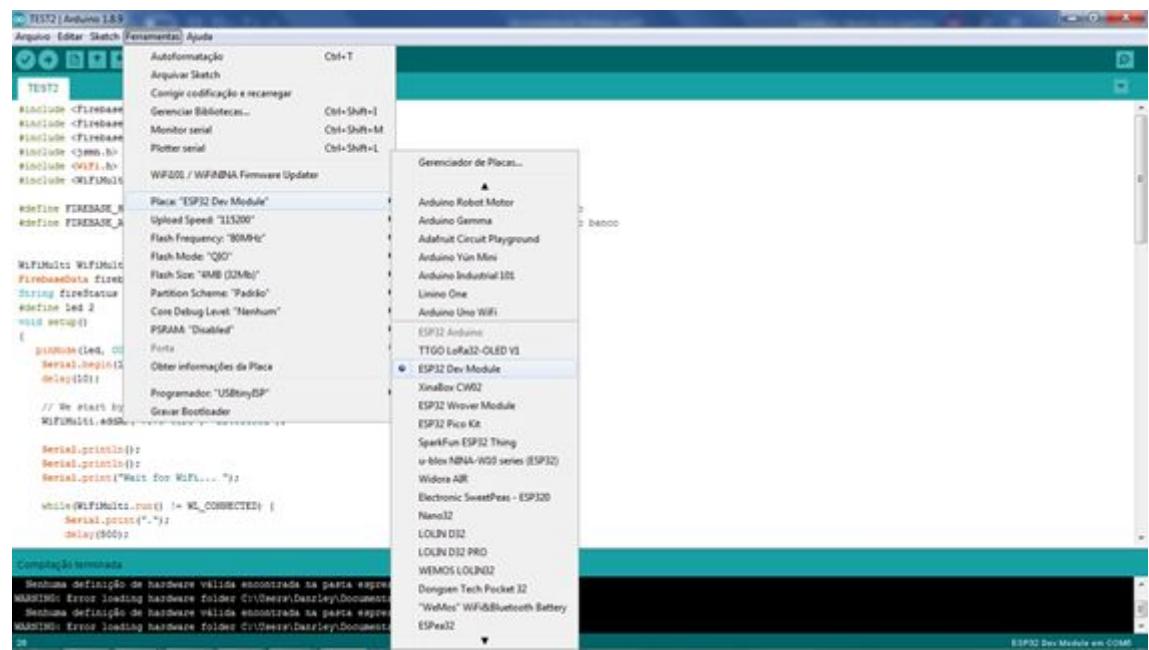
### 3.1



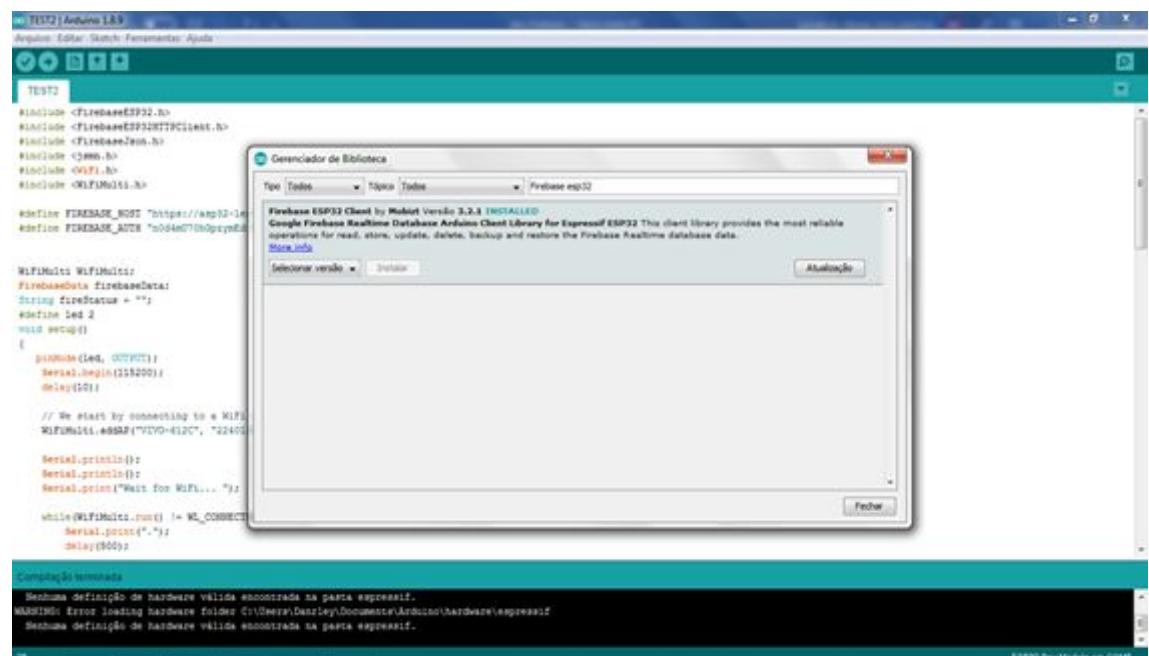
### 3.2



## 4 - Selecionando a placa esp32

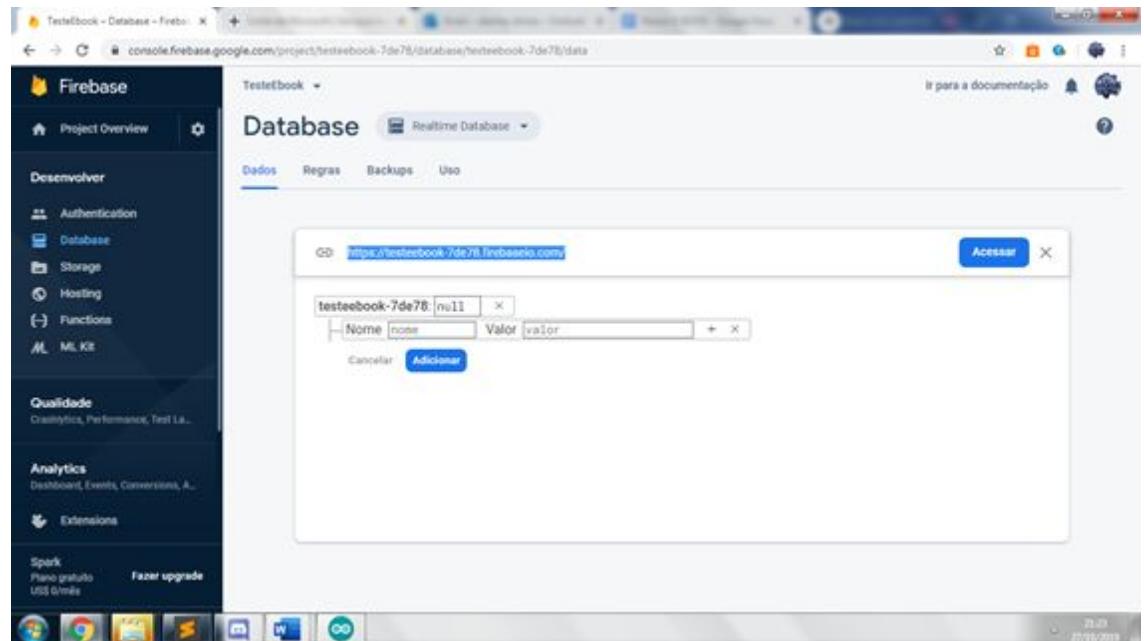


## 5 - Colocando a biblioteca do firebase

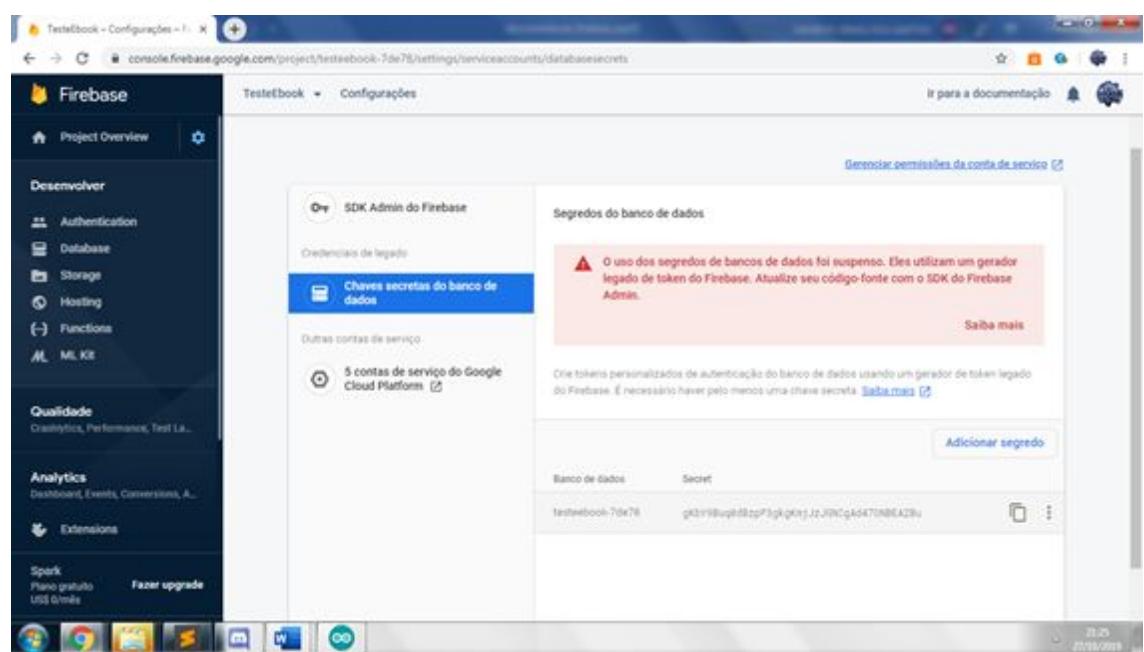


## Código utilizado

Antes é necessário pegar a URL e a chave do banco de dados



The screenshot shows the Firebase Database console for the project 'Testebook'. A modal dialog is open in the center, titled 'Dados' (Data). It contains a URL field with 'https://testebook-7de78.firebaseio.com/' and a table with one row: 'Nome' (Name) and 'Valor' (Value), both set to 'valor'. There are 'Cancelar' (Cancel) and 'Adicionar' (Add) buttons at the bottom. The background shows the database structure with a single node 'testebook-7de78/null'.



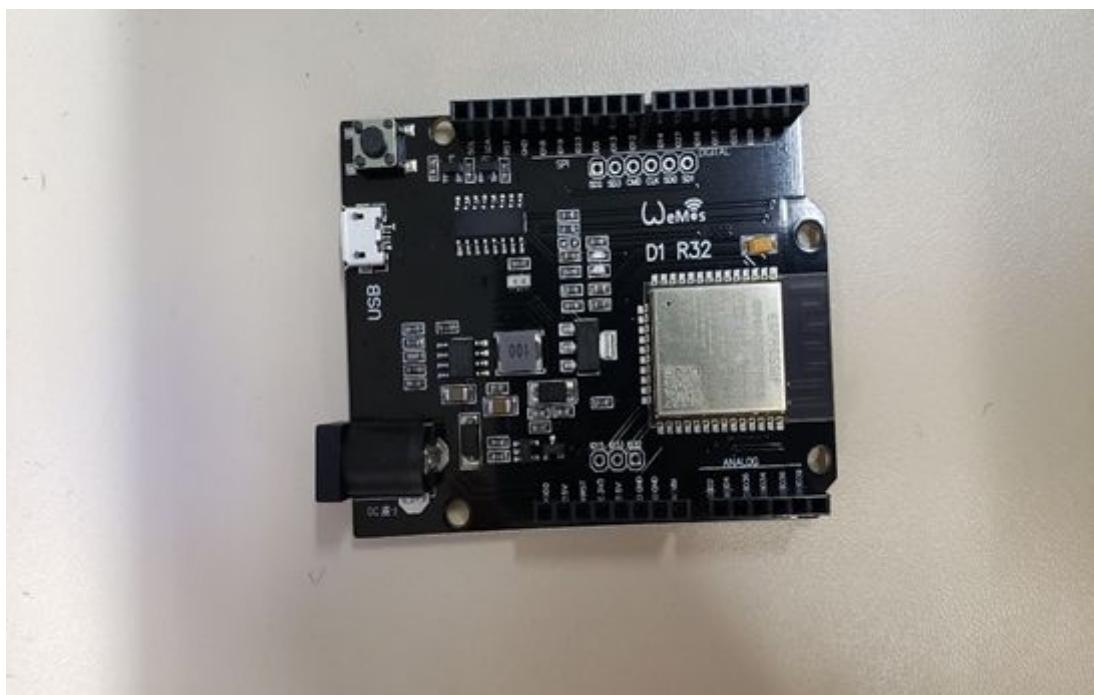
The screenshot shows the Firebase Configuration console for the project 'Testebook'. On the left, the navigation bar includes 'Database' under 'Desenvolver'. In the main area, under 'Segredos do banco de dados' (Database secrets), there is a warning message: 'O uso dos segredos de bancos de dados foi suspenso. Eles utilizam um gerador legado de token do Firebase. Atualize seu código-fonte com o SDK do Firebase Admins.' (The use of database secrets was suspended. They use a legacy Firebase token generator. Update your code with the Firebase Admins SDK.) Below this, a table lists a single secret: 'testebook-7de78' with the value 'gk3rV9ugld8tpZ3gkpk0jJzJ9CgA5470N8EA23u'. There is a 'Saiba mais' (Learn more) link and an 'Adicionar segredo' (Add secret) button.

```

●  #include <FirebaseESP32.h>
●  #include <FirebaseESP32HTTPClient.h>
●  #include <FirebaseJson.h>
●  #include <jsmn.h>
●  #include <WiFi.h>
●  #include <WiFiMulti.h>
●
●  #define FIREBASE_HOST ""                                // url do banco
●  #define FIREBASE_AUTH ""                             // a chave do banco
●
●  WiFiMulti WiFiMulti;
●  FirebaseData firebaseData;
●  String fireStatus = "";
●  #define led 2
●  void setup(){
●      pinMode(led, OUTPUT);
●      Serial.begin(115200);
●      delay(10);
●      // We start by connecting to a WiFi network
●      //COLOCANDO O WIFI
●      WiFiMulti.addAP("SSID", "SENHA");
●
●      Serial.println();
●      Serial.println();
●      Serial.print("Wait for WiFi... ");
●      while(WiFiMulti.run() != WL_CONNECTED) {
●          Serial.print(".");
●          delay(500);
●      }
●      Serial.println("");
●      Serial.println("WiFi connected");
●      Serial.println("IP address: ");
●      Serial.println(WiFi.localIP());
●
●  void loop() {
●      //pegando o campo do banco de dados
●      Firebase.getString(firebaseData, "/led");
●      String estado = firebaseData.stringData();
●      //verificando se o valor do campo contém o texto "ligado"
●      if(estado == "ligado"){
●          digitalWrite(led, HIGH);
●      }
●
●      //verificando se o valor do campo contém o texto "desligado"
●      else if(estado == "desligado"){
●          digitalWrite(led, LOW);
●      }
●  }

```

## Conexão com Arduino esp32 e MQTT



**Descrição:** Uma forma de enviar dados para nuvem e receber com o esp32, utilizando o protocolo MQTT.

## Componentes necessários

1 Unidade – Arduino Esp32;

Uma conta no Google;

## Gerenciando um broker

Link para baixar o gerenciador de broker:

[http://workswithweb.com/html/mqttbox/installing\\_apps.html#install\\_on\\_windows](http://workswithweb.com/html/mqttbox/installing_apps.html#install_on_windows)

Link para o host do broker do mosquito:

<http://test.mosquitto.org>

### 1 - Baixando o drive da placa esp32

Caso seu sistema operacional não achar o drive do usb automaticamente, você terá que baixar manualmente, esse é o link do drive da placa d1 r32 da marca

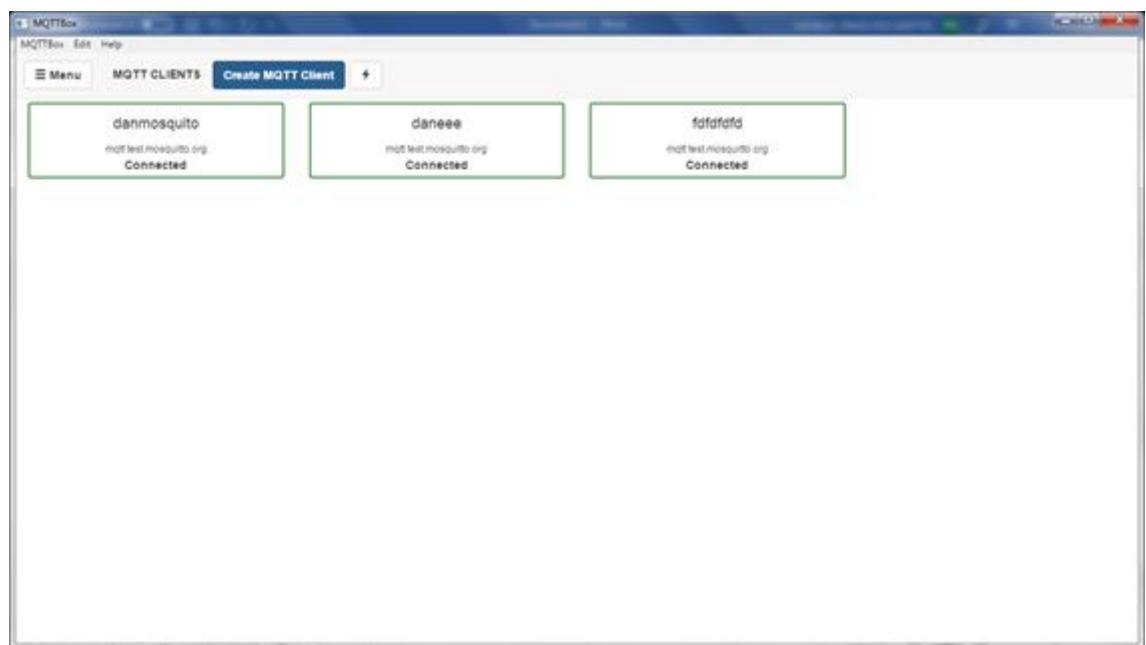
#### Referências:

link para baixar o driver:

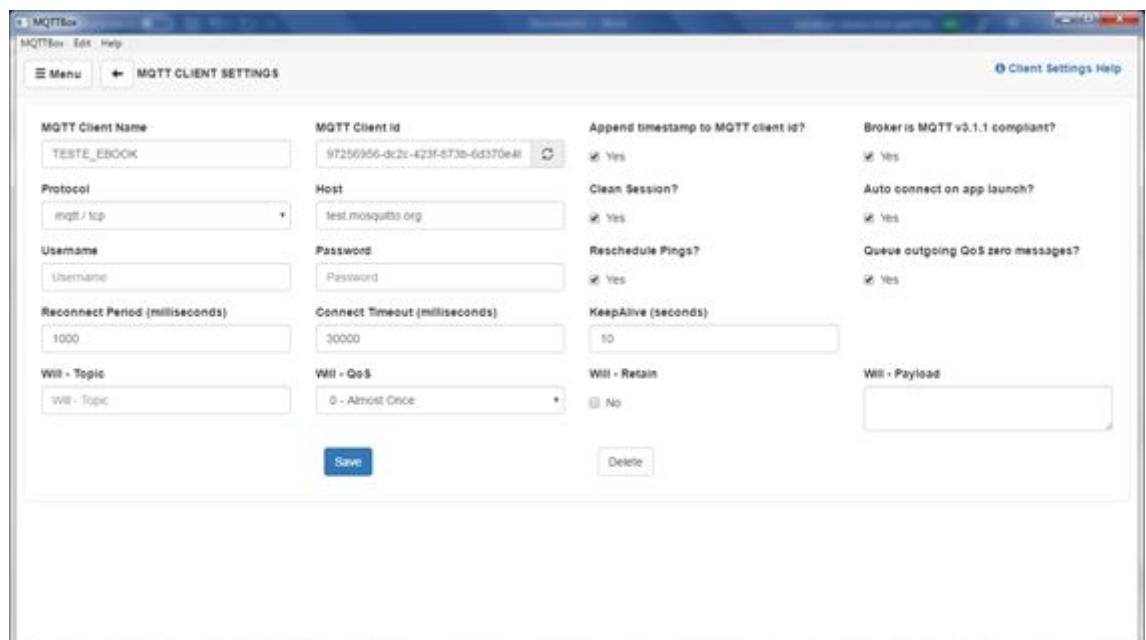
<http://blogmasterwalkershop.com.br/embarcados/wemos/wemos-d1-instalação-no-windows/>

### 2 - Criando um Cliente MQTT

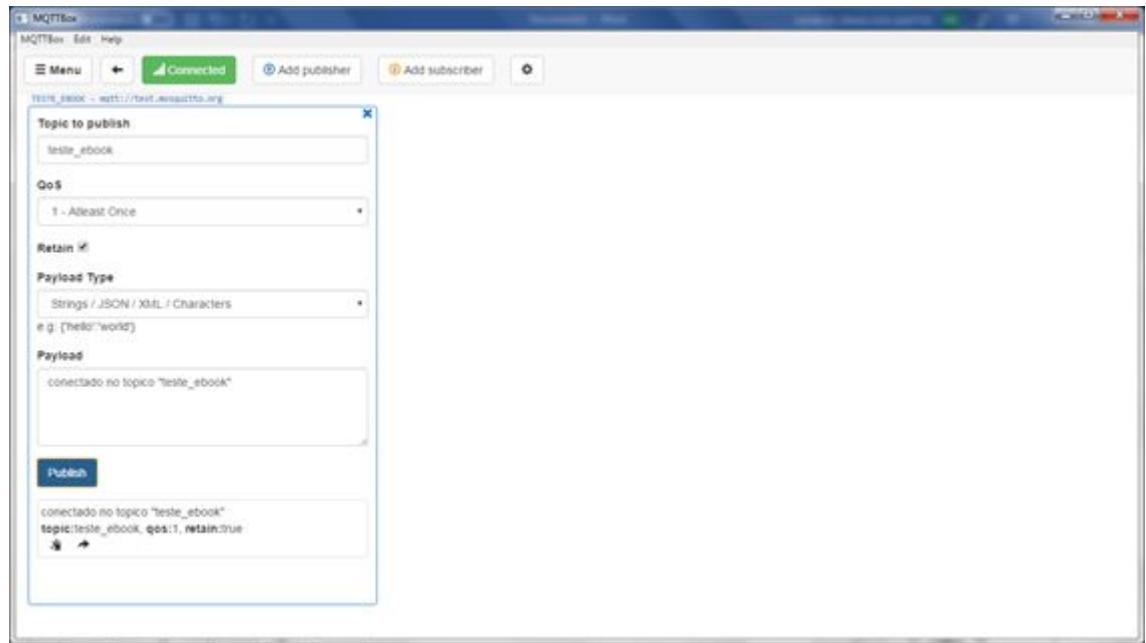
## 2.1



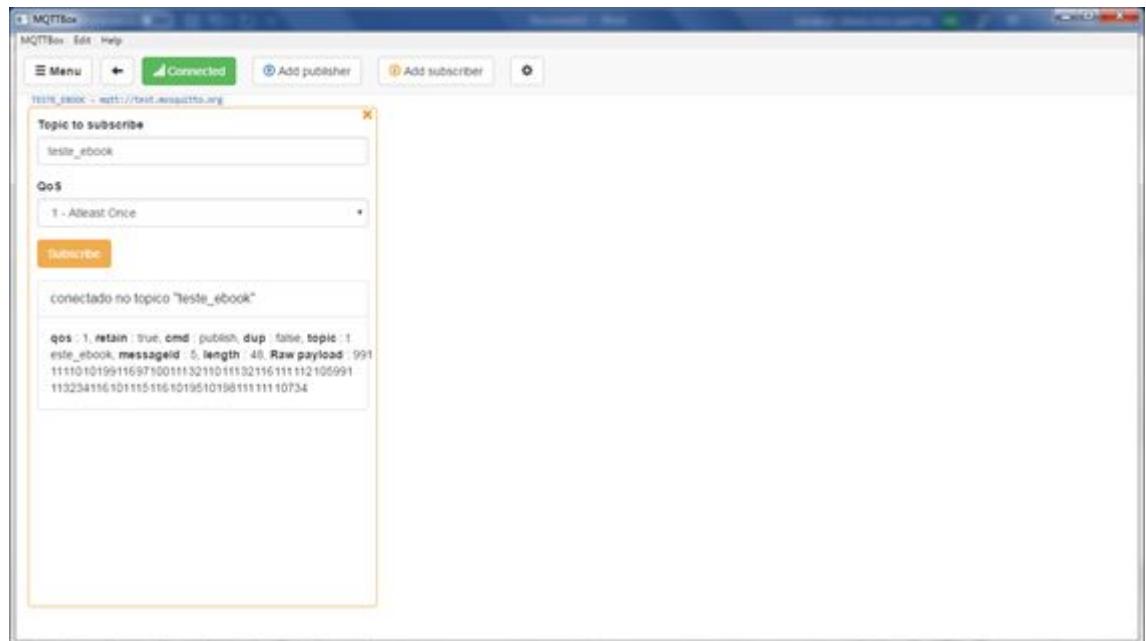
## 2.2



## 3 - Criando um tópico e definindo um Rentain



#### 4 - Adicionando um assinante



## Código utilizado

Antes é necessário baixar a biblioteca para a utilização do mqtt no Arduino.



```
• #include <PubSubClient.h>
• #include <WiFi.h>
• #include <WiFiMulti.h>
•
• #define WIFI_SSID "nome do seu wifi(ssid)"
• #define WIFI_PASSWORD "sua senha do wifi"
•
• void callback(char* topic, byte* payload, unsigned int length) {
•     Serial.print("Message arrived [");
•     Serial.print(topic);
•     Serial.print("] ");
•     for (int i=0;i<length;i++) {
•         Serial.print((char)payload[i]);
•     }
•     Serial.println();
• }
•
• WiFiMulti WiFiMulti;
• WiFiClient wifi;
• PubSubClient client(wifi);
•
• void reconnect() {
•     while (!client.connected()) {
•         Serial.print("Attempting MQTT connection...");
•         if (client.connect("arduinoClient")) {
•             Serial.println("connected");
•             // publicando na assinatura outTopic um texto "hello world"
•             client.publish("outTopic","hello world");
•             // assinando um topico no broker
•             client.subscribe("esp32/Teste2");
•         } else {
•             Serial.print("failed, rc=");
•             Serial.println(client.state());
•         }
•     }
• }
```

```
•         Serial.print(client.state());
•         Serial.println(" try again in 5 seconds");
•         // Wait 5 seconds before retrying
•         delay(5000);
•     }
• }
}

void setup(){
    Serial.begin(115200);

    WiFiMulti.addAP(WIFI_SSID,WIFI_PASSWORD);
    Serial.println();
    Serial.println();
    Serial.print("Wait for WiFi... ");

    while(WiFiMulti.run() != WL_CONNECTED) {
        Serial.print(".");
        delay(500);
    }

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());

    //utilizando um host para gerenciar nosso broker
    client.setServer("test.mosquitto.org", 1883);
    client.setCallback(callback);

}
void loop(){
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
}
```

## **Integração do sensor de temperatura e umidade com a nuvem**



**É importante antes de começar a ver esse tópico você ver os outros tópicos acima, “Sensor de Temperatura e Umidade com I2C” e “Conexão com Arduino esp32 e Firebase”**

**Descrição:** Um sensor que verifica a temperatura e umidade, mandando esses dados para nuvem, utilizando o esp32 e o banco de dados de tempo real do Firebase.

### **Componentes necessários**

1 Unidade – Arduino Esp32;

1 Unidade – Sensor de temperatura e umidade DHT11 (normalmente é de cor azul) ou DHT22 (normalmente é branca); a diferença básica entre eles é a faixa de valores suportada nas medidas, a precisão das mesmas, além, é claro, do preço.

1 Unidade – Display LCD de 16x2 com adaptador I2C embutido;

Conjunto de cabos Jumper macho e fêmea.

## Código utilizado

```
• include <LiquidCrystal_I2C.h>
•
• //Umidade e temperatura
• #include <DHT_U.h>
•
• //FireBase Access
• #include <FirebaseESP32.h>
• #include <FirebaseESP32HTTPClient.h>
• #include <FirebaseJson.h>
•
• //WiFi
• //https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/station-examples.html
• //https://www.arduino.cc/en/Reference/WiFiStatus
• #include <WiFi.h>
• #include <WiFiMulti.h>
•
• #define DHTPIN 14      //Pino digital conectado ao Arduino
• #define DHTTYPE DHT22
• #define led 12
•
• #define FIREBASE_HOST "https://inventapp-26876.firebaseio.com/"
// url do banco
• #define FIREBASE_AUTH "C0mSItk4MVz7LkKUQdA902PCb5XkNDbpBlxi6Mat"
// a chave do banco
•
• WiFiMulti WiFiMulti;
• FirebaseData firebaseData;
•
• //definindo tempo para mandar os dados
//do sensor para o historico do banco de dados
• int tempo_env = 60000;
//definindo variavel auxiliar para calcular com a função "millis()"
• int aux_tempo;
•
• // 0x27 como endereço do LCD. O LCD possui 16 colunas e 2 linhas
• LiquidCrystal_I2C lcd(0x27, 16, 2);
•
• DHT_Unified dht(DHTPIN, DHTTYPE);
•
• //Delay usado pelo sensor de umidade e temperatura
• uint32_t delayMS;
•
• //Montagem do simbolo "grau"
• byte grau[8] = { B00001100,
•                 B00010010,
•                 B00010010,
•                 B00001100,
•                 B00000000,
•                 B00000000,
```

```
•             B00000000,
•             B00000000,
•         };
•
•     // 
•     const char* ntpServer = "pool.ntp.br";
•
•
•     void setup()
•     {
•         lcd.init();
•         lcd.backlight();
•         lcd.createChar(0, grau); //Cria o caractere de grau
•         pinMode(led, OUTPUT);
•         //Inicializa o serial
•         Serial.begin(115200);
•         // iniciando a conexão com wifi
•
•         connectToWiFi();
•         Serial.println("IP address: ");
•         Serial.println(WiFi.localIP());
•
•         // conectando no banco do firebase
•         Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
•
•         //sensor init
•         dht.begin();
•         sensor_t sensor;
•         //Imprime detalhes da temperatura
•         dht.temperature().getSensor(&sensor);
•
•         //Define o delay entre leituras do sensor baseado nos detalhes do
•         //sensor
•         delayMS = sensor.min_delay / 2000;
•         lcd.clear();
•     }
•     void loop()
•     {
•         //Se ESP32 desconectado da rede
•         if(WiFi.status() != WL_CONNECTED){
•             lcd.clear();
•             lcd.setCursor(0,0);
•             lcd.print("sem conexao");
•             delay(2000);
•             lcd.clear();
•             mostraDados();
•             delay(2000);
•         }
•
•         //LED
•         estadoLED();
•
•         delay(delayMS); // /1000 -> 2 segundos; /2000 -> 1 segundo;
•         Serial.print(delayMS);
```

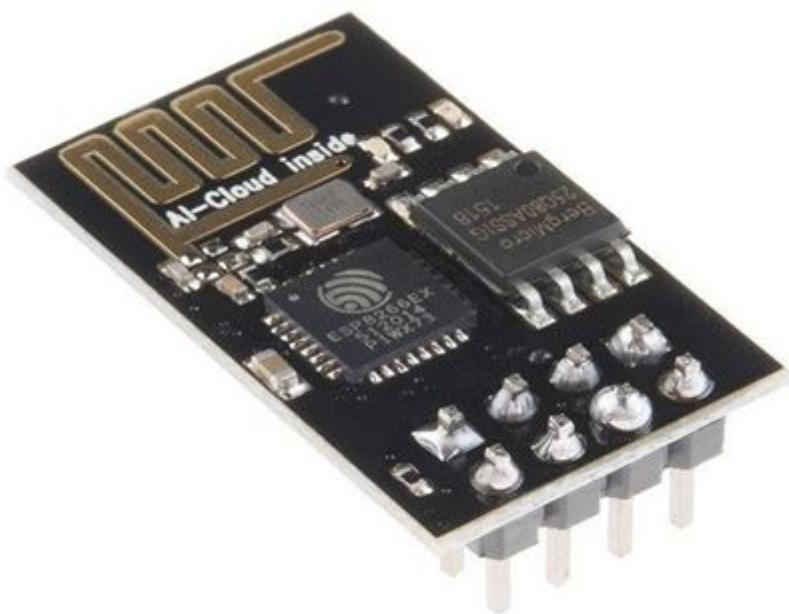
```

•     //Atualizar dados no LCD
•     mostraDados();
•
• } //loop
•
•
• void estadoLED(){
•     // "pegar" campo associado à condição do LED
•     Firebase.getString(firebaseData,"teste2_firebase/Nome");
•     String estado = firebaseData.stringData();
•     if(estado == "\\\\"Acesso\\\"") {
•         digitalWrite(led, HIGH);
•     }
•     else if(estado == "\\\\"Apagado\\\"") {
•         digitalWrite(led, LOW);
•     }
• }
•
• void connectToWiFi(){
•     //acesso ao wifi. login e senha
•     WiFiMulti.addAP("g", "12345678");
•     WiFiMulti.addAP("cogroo", "qwertyuiop");
•
•     lcd.setCursor(0,0);
•     lcd.print("aguardando wifi...\n");
•
•     while(WiFiMulti.run() != WL_CONNECTED) {
•         lcd.setCursor(0,1);
•         lcd.print(".");
•         delay(600);
•     }
•     lcd.println("WiFi connected");
• }
•
• void mostraDados(){
•     lcd.setCursor(0, 0);
•     lcd.print("FATEC Ipiranga");
•     sensors_event_t event;
•
•     // Pega o evento de temperatura e imprime seu valor
•     dht.temperature().getEvent(&event);
•     if (!isnan(event.temperature)) {
•         lcd.setCursor(0, 1);
•         lcd.print("T=");
•         lcd.print(event.temperature, 1);
•         lcd.write(byte(0));
•         lcd.print("C");
•     }
•     Firebase.setFloat(firebaseData,"teste2_firebase/Temperatura",
•     event.temperature); //muda o valor do campo "Temperatura" para a
•     temperatura atual
•
•     // Pega o evento de umidade e imprime seu valor
•     dht.humidity().getEvent(&event);

```

```
•     if (!isnan(event.relative_humidity)) {  
•         lcd.setCursor(9, 1);  
•         lcd.print("U=");  
•         lcd.print(event.relative_humidity, 1);  
•         lcd.print("%");  
•     }  
•     Firebase.setFloat(firebaseData, "teste2_firebase/Humidade",  
event.relative_humidity); //muda o valor do campo "Umidade" para a  
umidade atual  
• }
```

## Conexão com Arduino esp-01 e Firebase



**Descrição:** Uma forma de enviar dados para nuvem e receber com o esp-01, utilizando o banco de dados em tempo real do Firebase.

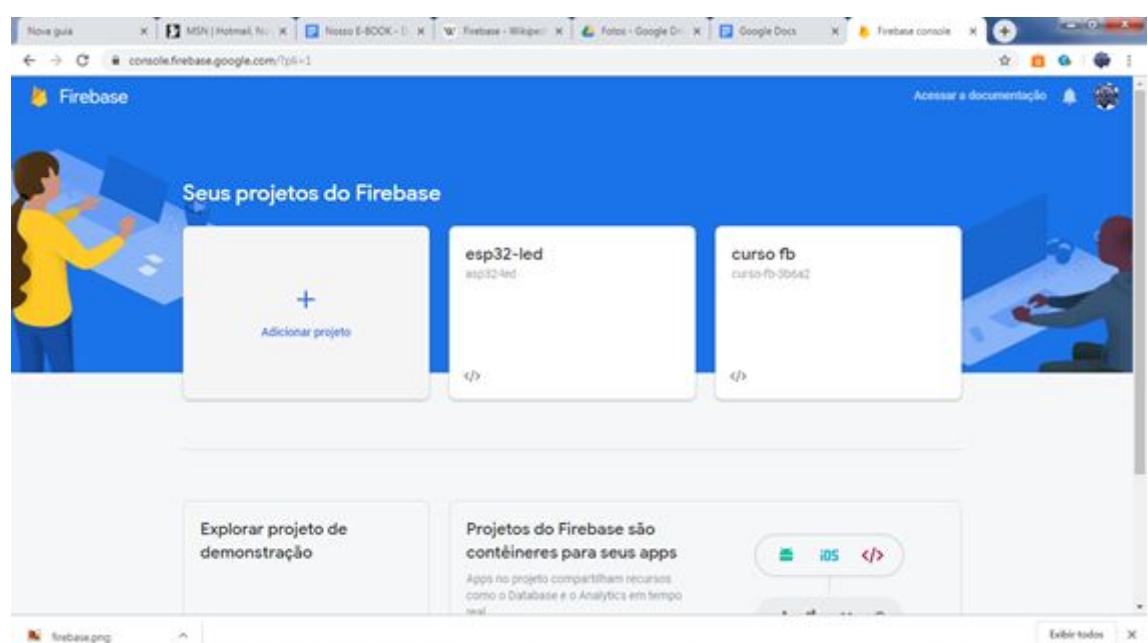
## Componentes necessários

1 Unidade – Arduino Esp-01;

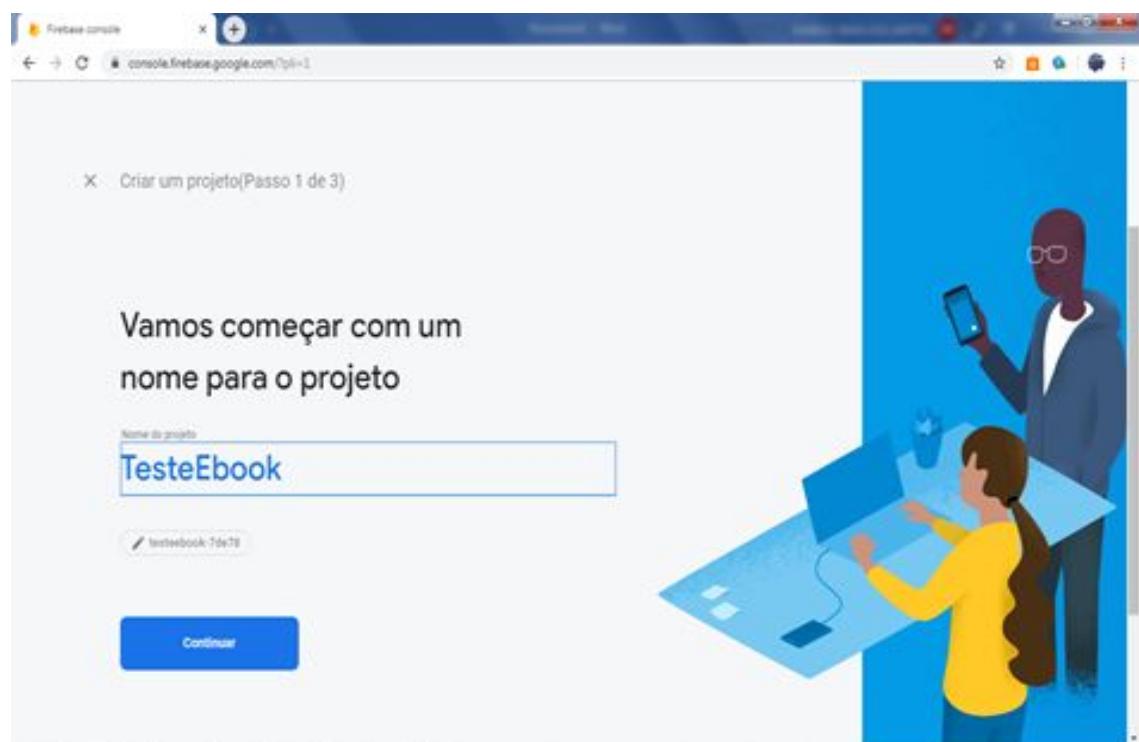
Uma conta no Google;

## Utilizando o Firebase

### 1- Adicione um projeto

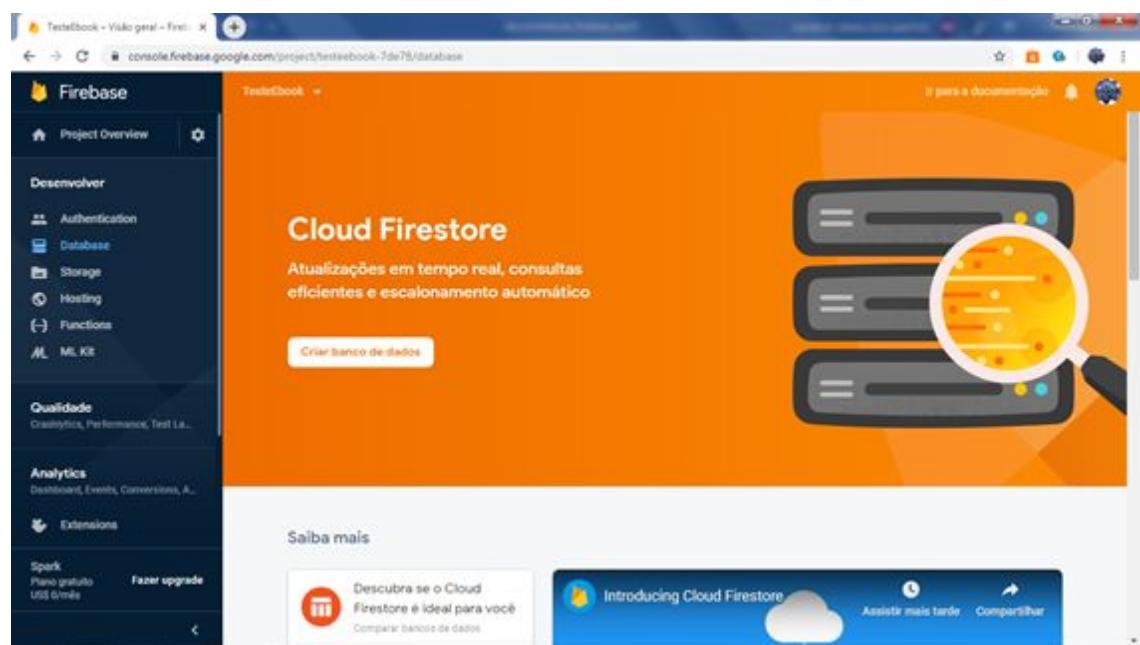


## **2- Coloque o nome do projeto**

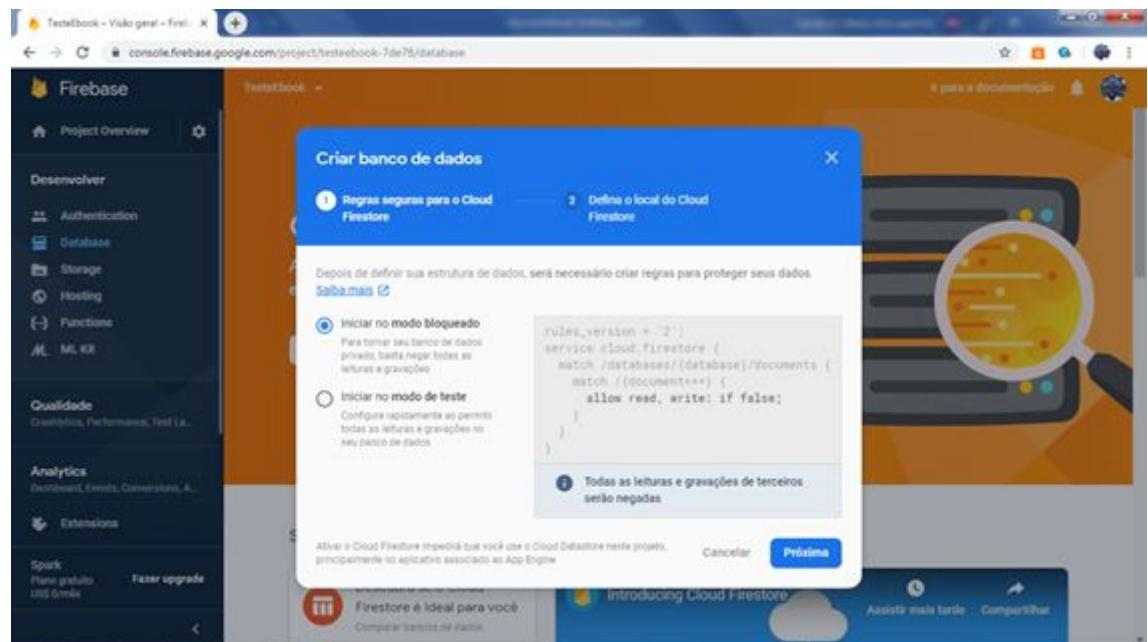


## **3- Criando um bando de dados**

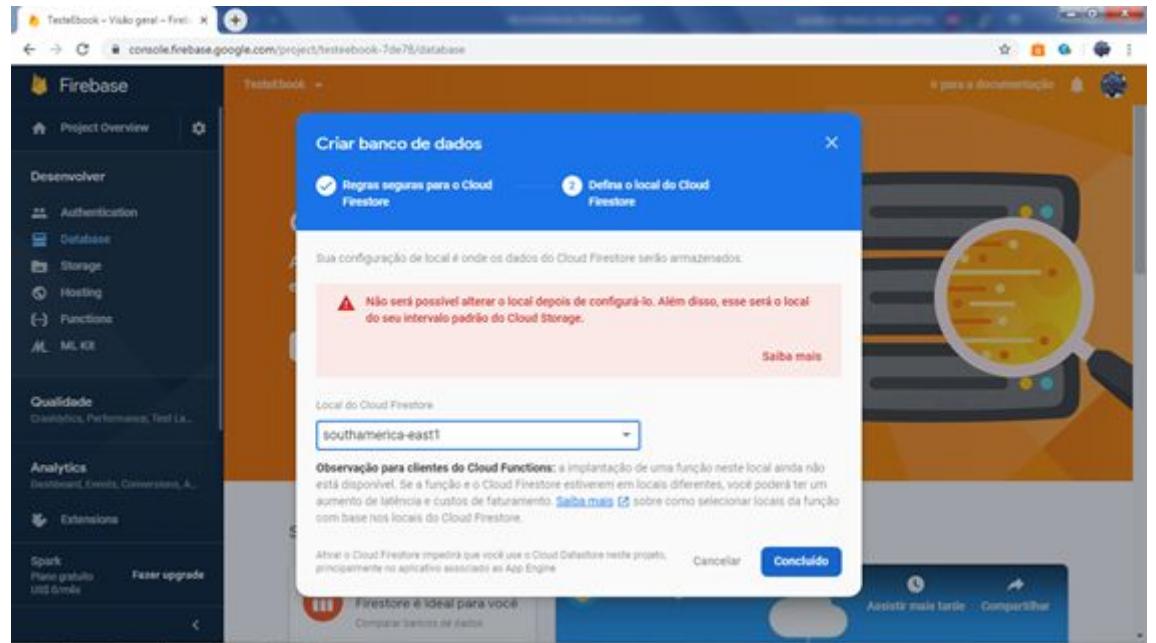
### **3.1**



### 3.2



### 3.3



## Configurando a IDE do Arduino para o esp-01

### 1 - Baixando o drive da placa esp-01

Caso seu sistema operacional não achar o drive do usb automaticamente, você terá que baixar manualmente, esse é o link do drive do serial:

<http://blogmasterwalkershop.com.br/embarcados/wemos/wemos-d1-instalacao-no-windows/>

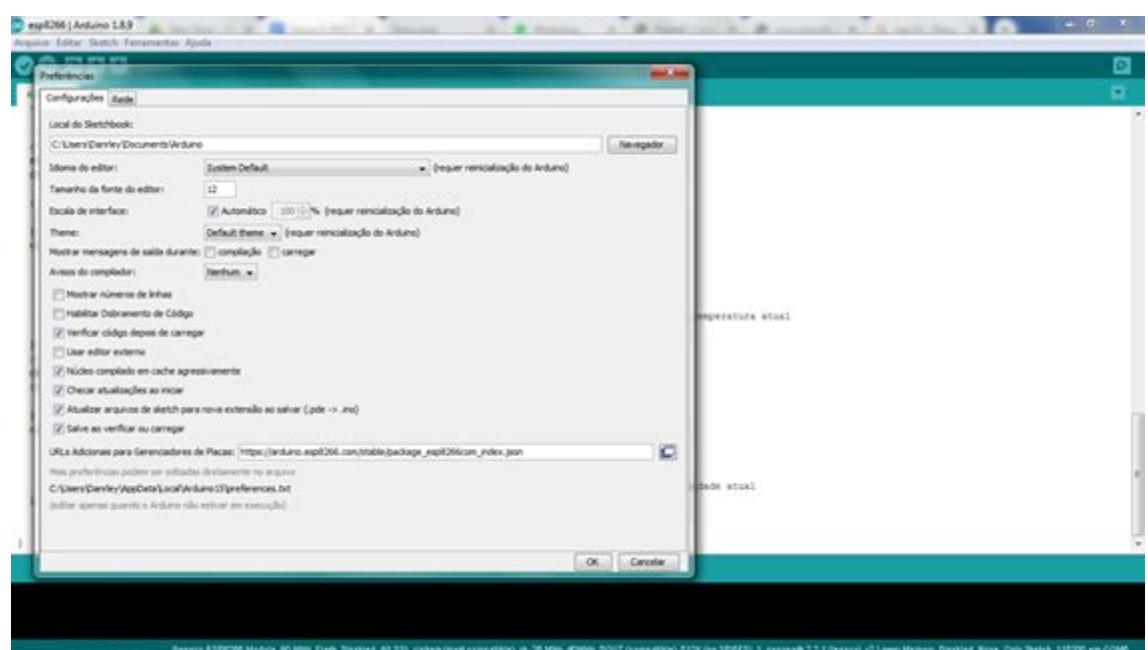
### 2 - Colocando a URL para gerenciar o esp-01

```
esp8266 | Arduino 1.8.9
Arquivo Editar Sketch Ferramentas Ajuda
Novo Ctrl+N
Abre... Ctrl+O
Arquivo Recente
Sketchbook
Exemplos
Fechar Ctrl+W
Salvar Ctrl+S
Salvar como... Ctrl+Shift+S
Configuração da página Ctrl+Shift+P
Imprime Ctrl+P
Prefe... Ctrl+Vírgula ,,
Sair Ctrl+Q

Firebase.setRef(firebaseData, "/Temperature", event.temperature); //Muda o valor do campo "Temperatura" para a temperatura atual

}
// Pega o evento de umidade e imprime seu valor
dbt.humidity().getEvent(event);
if (event.relative_humidity) {
  Serial.println(F("Erro ao ler a umidade"));
}
else {
  Serial.print(F("Humidity: "));
  Serial.print(event.relative_humidity);
  Serial.println(F("%"));
  Firebase.setRef(firebaseData, "/Umidade", event.relative_humidity); //Muda o valor do campo "Umidade" para a umidade atual
}

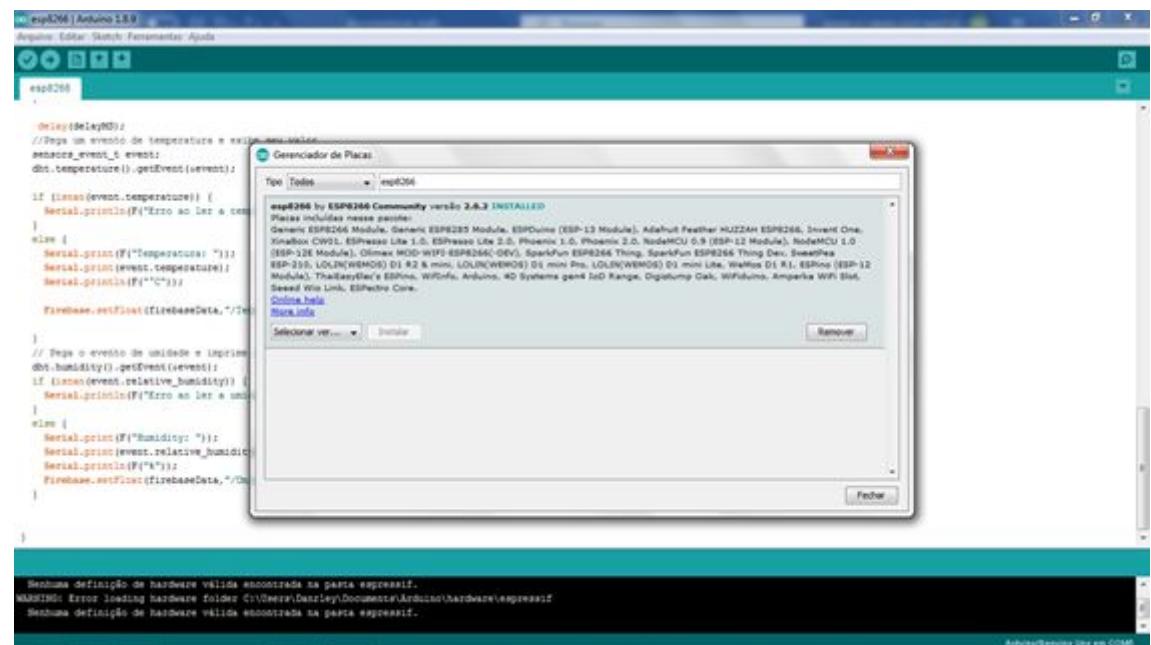
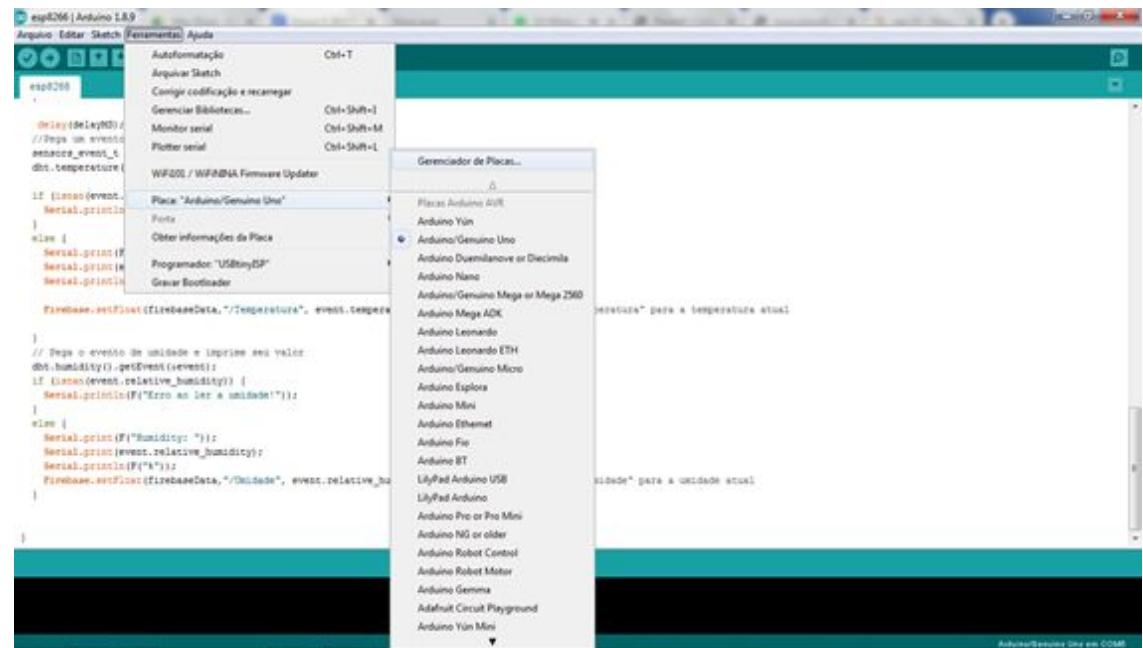

```



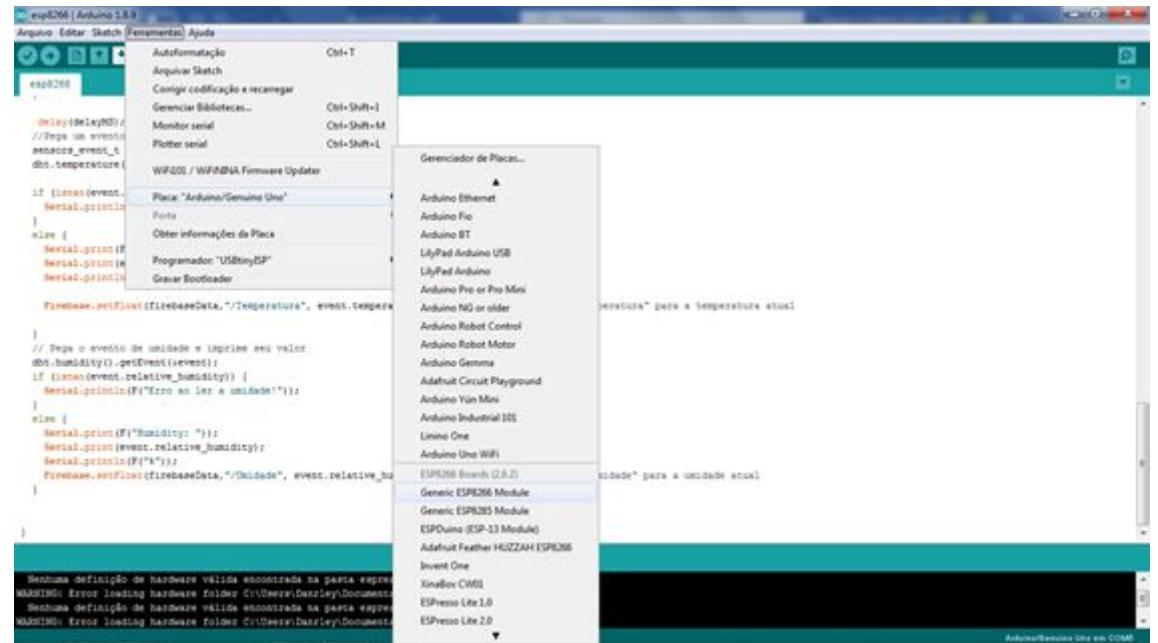
URL utilizada:

[https://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](https://arduino.esp8266.com/stable/package_esp8266com_index.json)

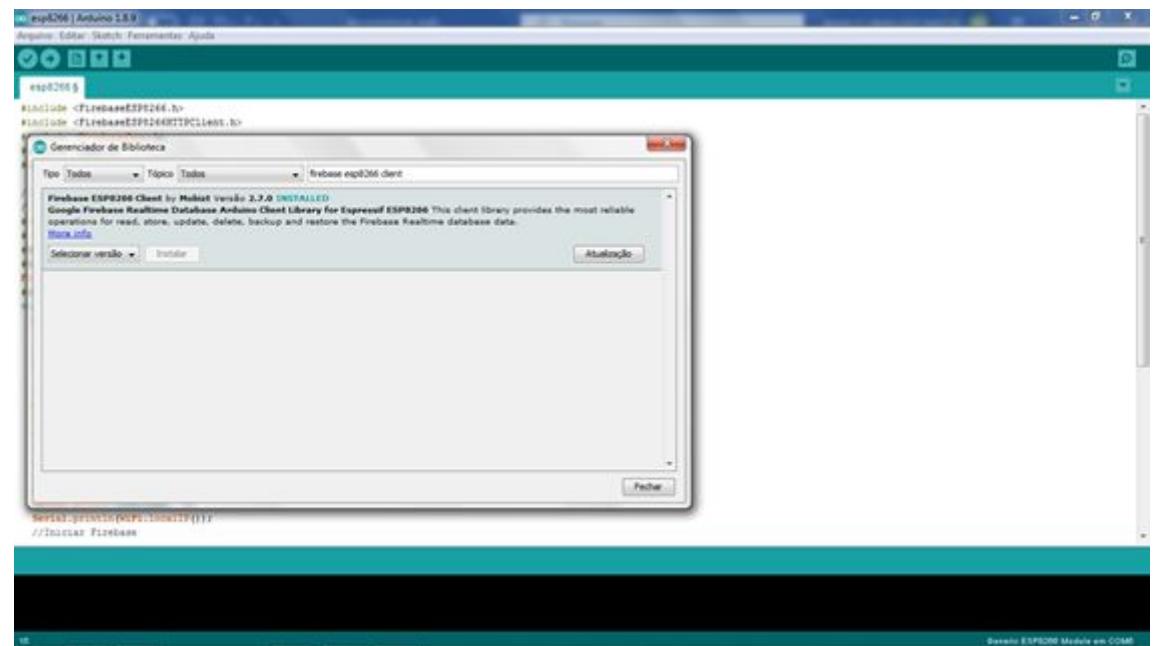
### 3 - Baixando a biblioteca do esp-01



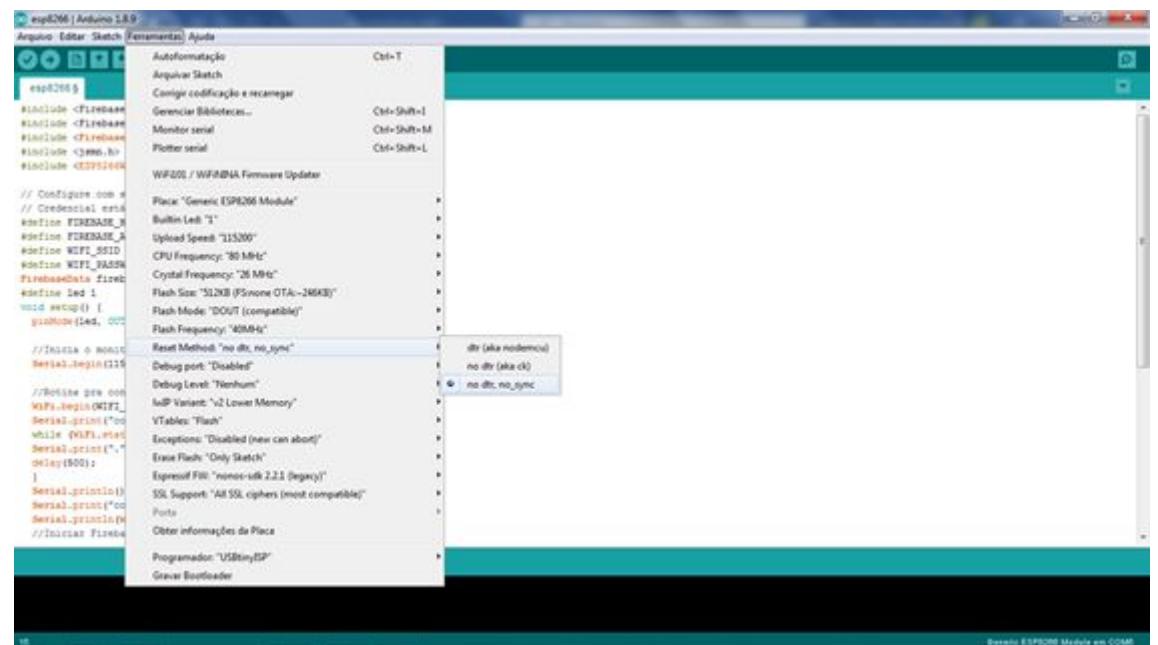
## 4 - Selecionando a placa esp-01



## 5 - Colocando a biblioteca do firebase



## 6 - Configurando o modo reset



## Código utilizado

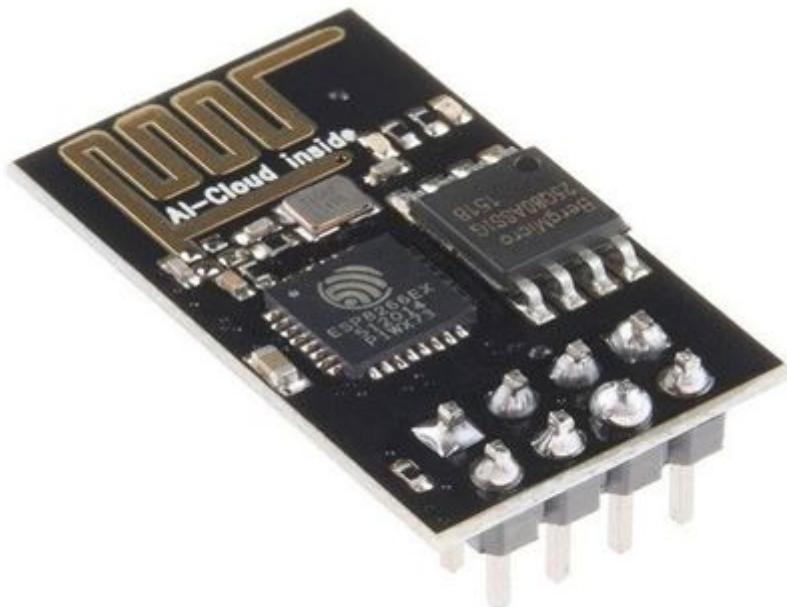
Antes é necessário pegar a URL e a chave do banco de dados

The screenshot shows the Firebase Realtime Database console for the project 'Testebook'. In the left sidebar, under 'Desenvolver', 'Database' is selected. On the right, the 'Dados' tab is active, showing a table with one row: 'testebook-7de78: null'. A modal window is open over the table, containing a form with 'Nome' set to 'teste' and 'Valor' set to 'valor'. Below the form are 'Cancelar' and 'Adicionar' buttons.

The screenshot shows the Firebase Configuration console for the project 'Testebook'. In the left sidebar, under 'Configurações', 'Database secrets' is selected. The main area displays a table titled 'Segredos do banco de dados' with one row: 'testebook-7de78' and a long secret key. A warning message at the top right states: 'O uso dos segredos de bancos de dados foi suspenso. Eles utilizam um gerador legado de token do Firebase. Atualize seu código-fonte com o SDK do Firebase Admins.' (The use of database secrets was suspended. They use a legacy token generator from Firebase. Update your code with the Firebase Admins SDK.)

```
●  #include <FirebaseESP8266.h>
●  #include <FirebaseESP8266HTTPClient.h>
●  #include <FirebaseJson.h>
●  #include <jsmn.h>
●  #include <ESP8266WiFi.h>
●
●  // Configure com suas credenciais
●  // Credencial está em configuração do projeto, contas e serviços.
●  #define FIREBASE_HOST ""//url
●  #define FIREBASE_AUTH ""//chave do banco
●  #define WIFI_SSID "ssid"
●  #define WIFI_PASSWORD "senha"
●  FirebaseData firebaseData;
●  #define led 1
●  void setup() {
●      pinMode(led, OUTPUT);
●      //Inicia o monitor serial
●      Serial.begin(115200);
●      //Rotina pra conectar ao wifi.
●      WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
●      Serial.print("conectando");
●      while (WiFi.status() != WL_CONNECTED) {
●          Serial.print(".");
●          delay(500);
●      }
●      Serial.println();
●      Serial.print("conectado: ");
●      Serial.println(WiFi.localIP());
●      //Iniciar Firebase
●      Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
●  }
●
●  void loop() {
●      Firebase.setFloat(firebaseData, "/teste", t);
●
●      //pegando o campo do banco de dados
●      Firebase.getString(firebaseData, "/led");
●      String estado = firebaseData.stringValue();
●      //verificando se o valor do campo contém o texto "ligado"
●      if(estado == "ligado"){
●          digitalWrite(led,LOW);
●          Serial.print("ligado");
●      }
●      //verificando se o valor do campo contém o texto "desligado"
●      else if(estado == "desligado"){
●          digitalWrite(led,HIGH);
●          Serial.print("desligado");
●      }
●  }
● }
```

## Conexão com Arduino esp-01 e MQTT



**Descrição:** Uma forma de enviar dados para nuvem e receber com o esp-01, utilizando o protocolo MQTT.

### Componentes necessários

1 Unidade – Arduino Esp-01;

Uma conta no Google;

## Gerenciando um broker

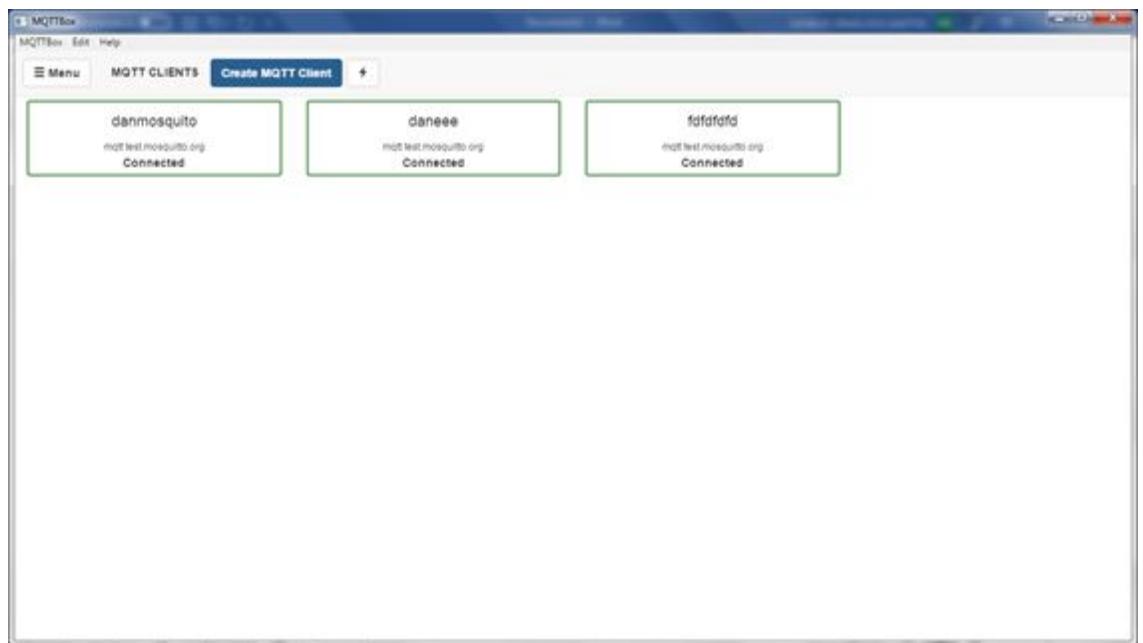
### 1 - Baixando o drive da placa esp-01

Caso seu sistema operacional não achar o drive do usb automaticamente, você terá que baixar manualmente, esse é o link do drive do serial:

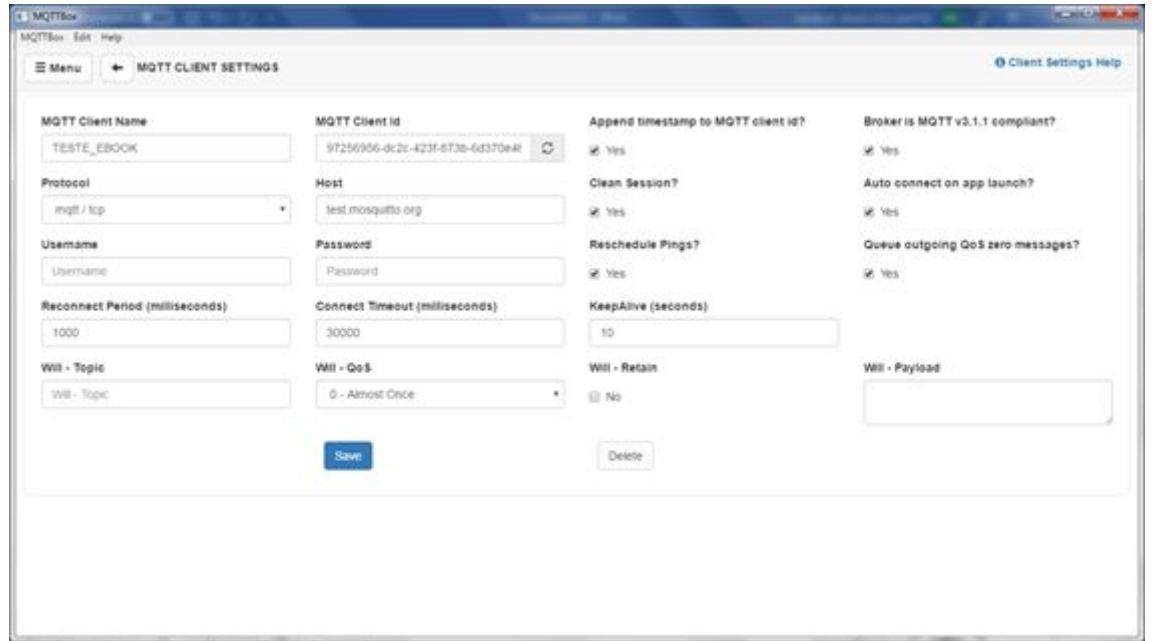
<http://blogmasterwalkershop.com.br/embarcados/wemos/wemos-d1-instalacao-no-windows/>

### 2 - Criando um Cliente MQTT

#### 2.1

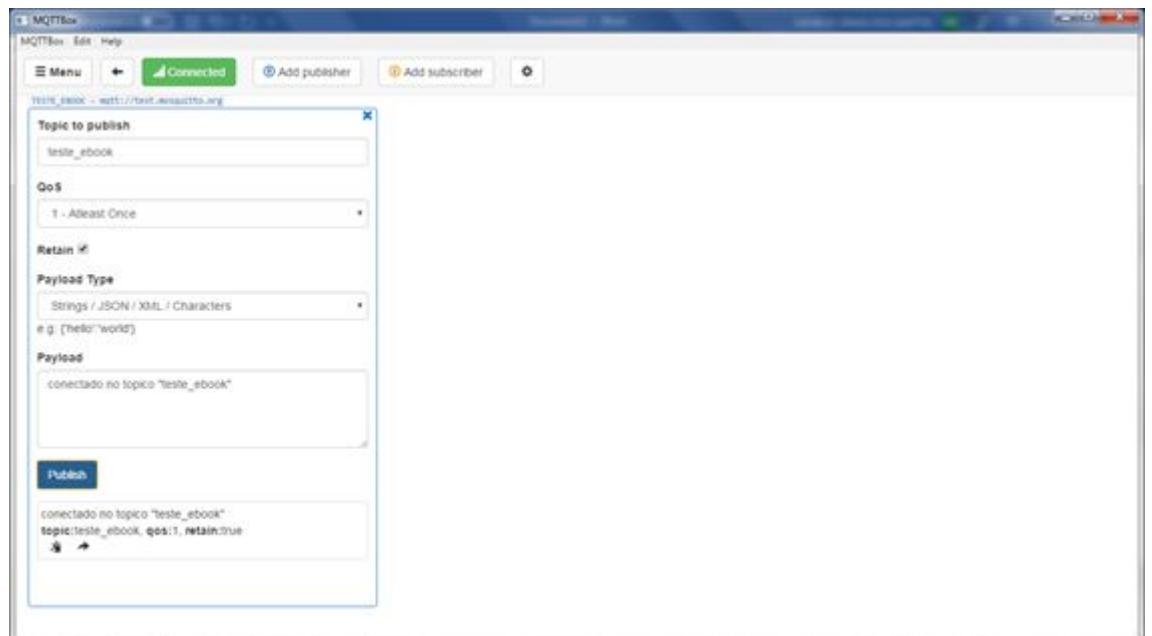


## 2.2

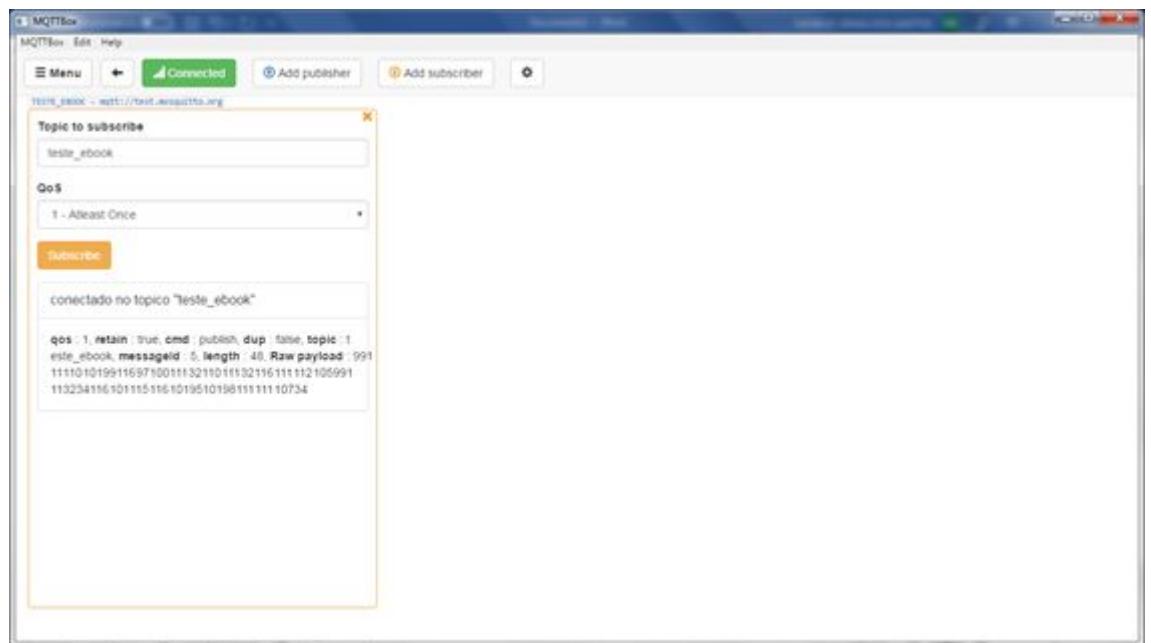


**Host utilizado:** test.mosquitto.org

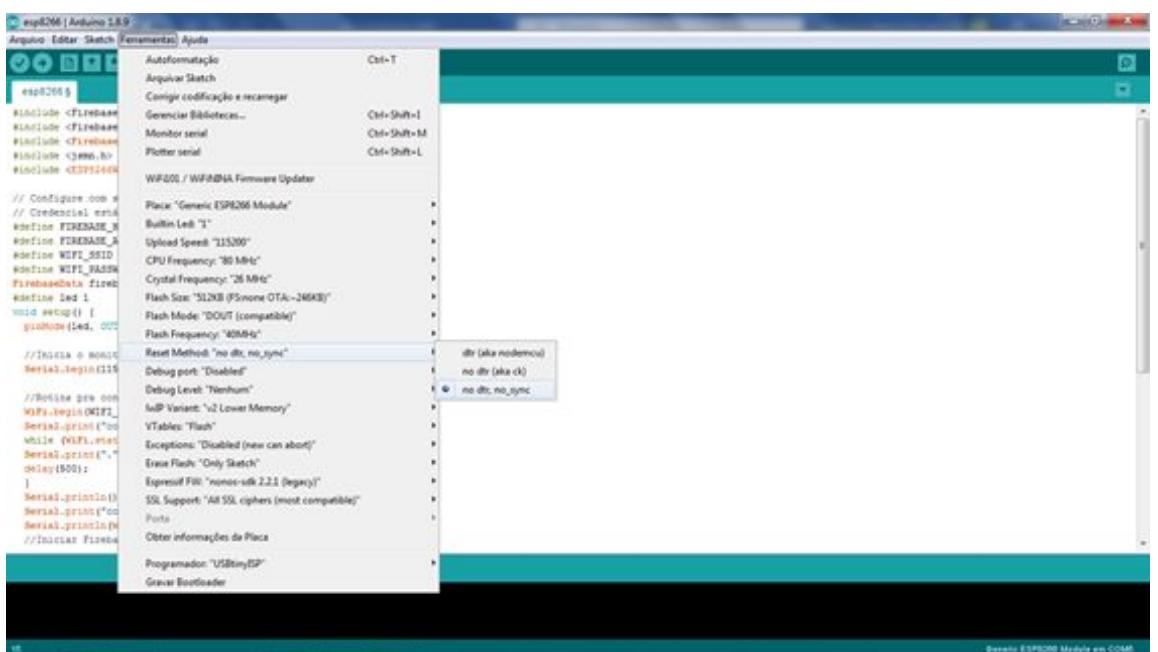
## 3 - Criando um tópico e definindo um Retain



## 4 - Adicionando um assinante



## 5 - Configurando o modo de reset



## Código utilizado

Antes é necessário baixar a biblioteca para a utilização do mqtt no Arduino.



```
• #include <ESP8266WiFi.h>
• #include <PubSubClient.h>
•
• #define WIFI_SSID "ssid" // input your home or public wifi name
•
• #define WIFI_PASSWORD "senha"
• #define led 2
•
• String valor;
•
• void callback(char* topic, byte* payload, unsigned int length) {
•
•     Serial.print("Message arrived [");
•
•     Serial.print(topic);
•
•     Serial.print("] ");
•
•
•     for (int i=0;i<length;i++) {
•
•         Serial.print((char)payload[i]);
•         valor = valor + (char)payload[i];
•
•     }
•
•     Serial.println();
•
• }
•
• WiFiClient wifi;
•
• PubSubClient client(wifi);
```

```
void reconnect() {
    // Loop until we're reconnected

    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect("arduinoClient")) {
            Serial.println("connected");
            // publishing to the topic outTopic a string "hello world"
            client.publish("outTopic", "hello world");
            // assinando um topico no broker
            client.subscribe("esp32/Teste2");
        }
        else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
}

void setup(){
    pinMode(led, OUTPUT);

    Serial.begin(115200);

    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    Serial.print("conectando");
    while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(500);
    }
    Serial.println();
    Serial.print("conectado: ");
}
```

```
●     Serial.println(WiFi.localIP());
●
●     Serial.println("");
●
●     Serial.println("WiFi connected");
●
●     Serial.println("IP address: ");
●
●     Serial.println(WiFi.localIP());
●
●     //utilizando um host para gerenciar nosso broker
●
●     client.setServer("test.mosquitto.org", 1883);
●
●     client.setCallback(callback);
●
● }
●
● void loop(){
●
●     if(!client.connected()) {
●
●         reconnect();
●     }
●
●     client.loop();
●     //String valor = client.state();
●
●     Serial.print("esse aqui é o valor:");
●     Serial.print(valor);
●
● }
```

## **ApplInventor - Primeiras experiências com Android**

Hoje em dia, aplicações para aparelhos móveis vem se tornando cada vez mais comuns e desejadas, pois nos auxiliam onde quer que estejamos, nos mais variados cenários.

Para quem está começando a se aventurar no mundo da programação e do desenvolvimento de novas aplicações e tecnologias, é compreensível que haja um certo receio e insegurança quando se adentra na área de aparelhos móveis, já que há diferenciais com relação ao desenvolvimento de aplicações para desktop ou aplicações online.

Mas, para tudo há um começo simple e rápido para auxiliá-los em seus primeiros passos. Uma das formas utilizadas para começar a desenvolver aplicações Android é a ferramenta da MIT: o ApplInventor.

O ApplInventor é um ambiente de desenvolvimento gratuito com o enfoque na criação, desenvolvimento, compartilhamento e avaliações de aplicações Android.

Sua interface é relativamente simples, onde se pode acessar os mais diversos componentes, verificar suas especificações e características que podem ou não ser editáveis.

Contudo, um dos principais diferenciais do ApplInventor para outros IDEs de programação, é o fato desta ser feita através de blocos de código para diversas sintaxes, sejam elas lógicas, matemáticas, condicionais, entre muitas outras.

Além do mais, é possível desenvolver e testar ao mesmo tempo! É possível sincronizar o ambiente e um aparelho com a mesma rede wi-fi, conectar o aparelho via cabo usb ou até mesmo utilizar um emulador, tudo para auxiliar seu cenário de desenvolvimento.

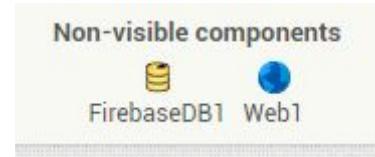
Isso acaba tornando a primeira experiência com desenvolvimento ‘mobile’ muito mais simples e prazerosa para iniciantes e até mesmo para usuários frequentes de linguagens de programação e design.

## Sensores de Humidade e Temperatura com AppInventor

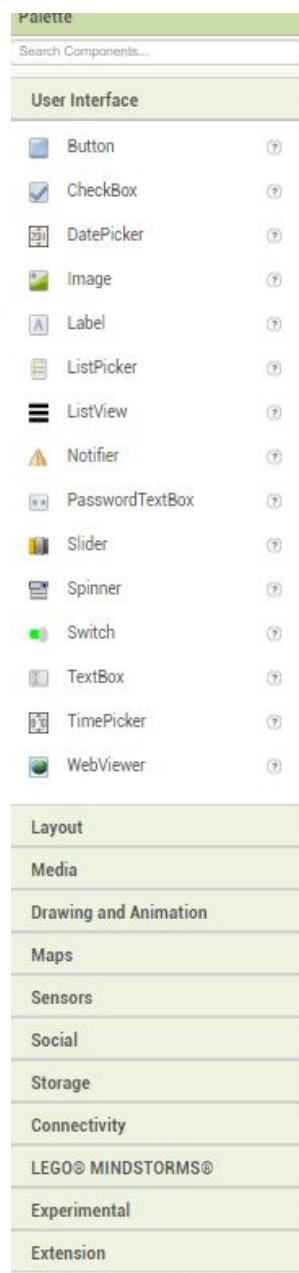
Esta é a interface principal onde são arrastados e soltos na interface os componentes disponibilizados pelo AppInventor



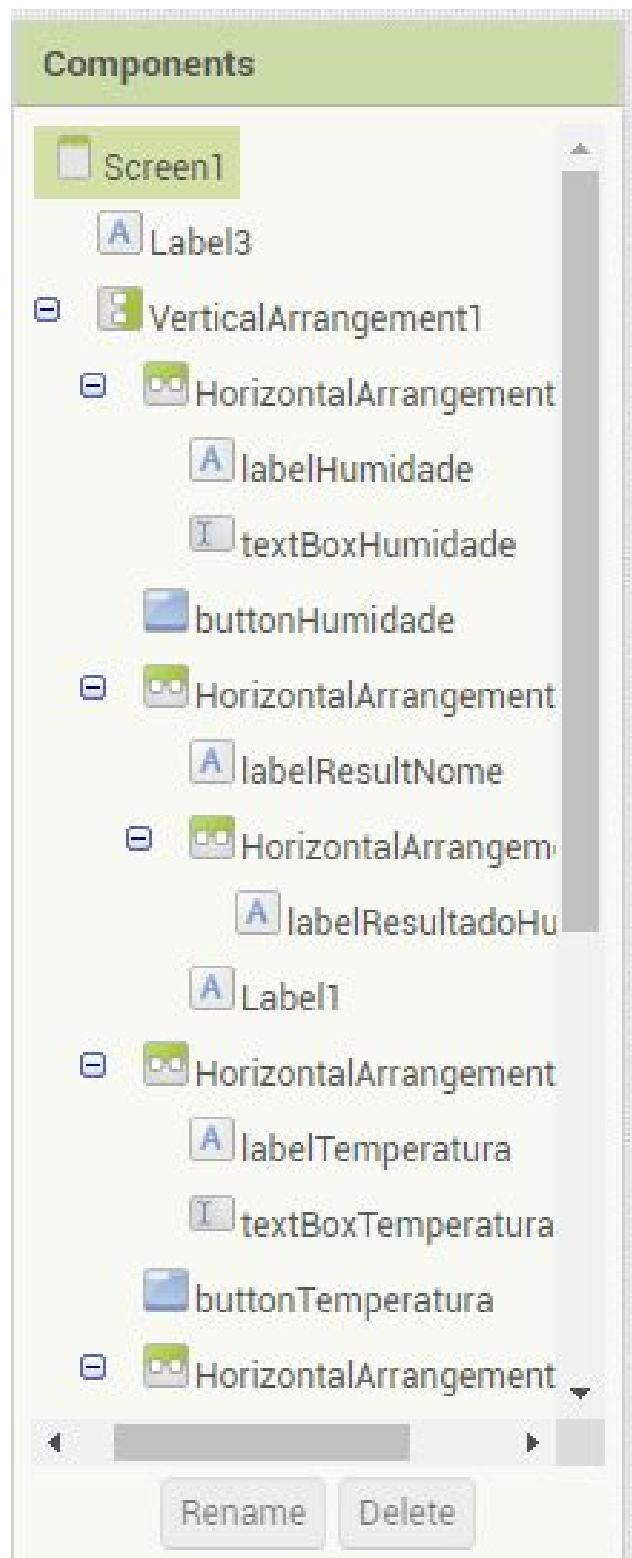
Aqui são dispostos os componentes de segundo plano OU não-visíveis da aplicação, como o experimental ‘Firebase’ e ‘Web’



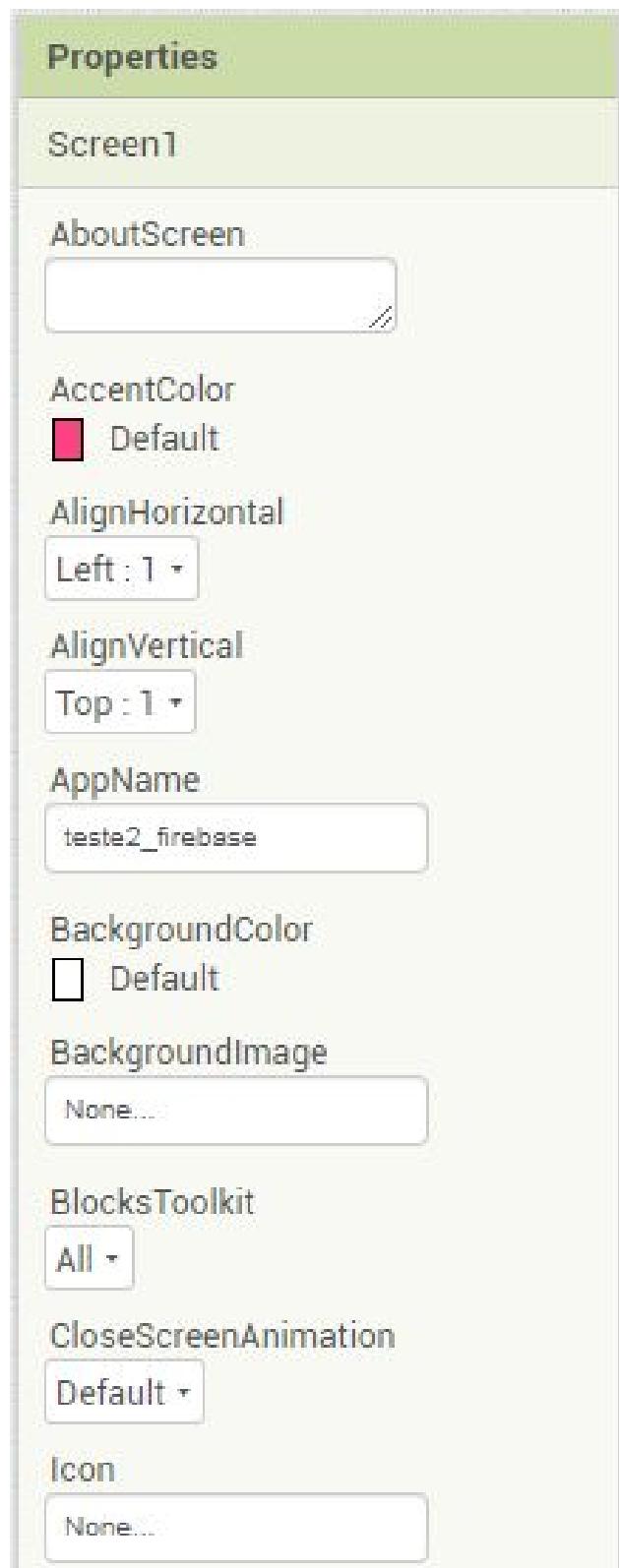
Aqui encontram-se todos os componentes disponibilizados pelo AppInventor. Os componentes são arrastados e soltos na interface da forma como desejar. Pode-se utilizar opções de layout vertical e horizontal para aprimorar o posicionamento dos componentes visuais



Cada componente que é colocado na interface é disposto aqui para acessibilidade

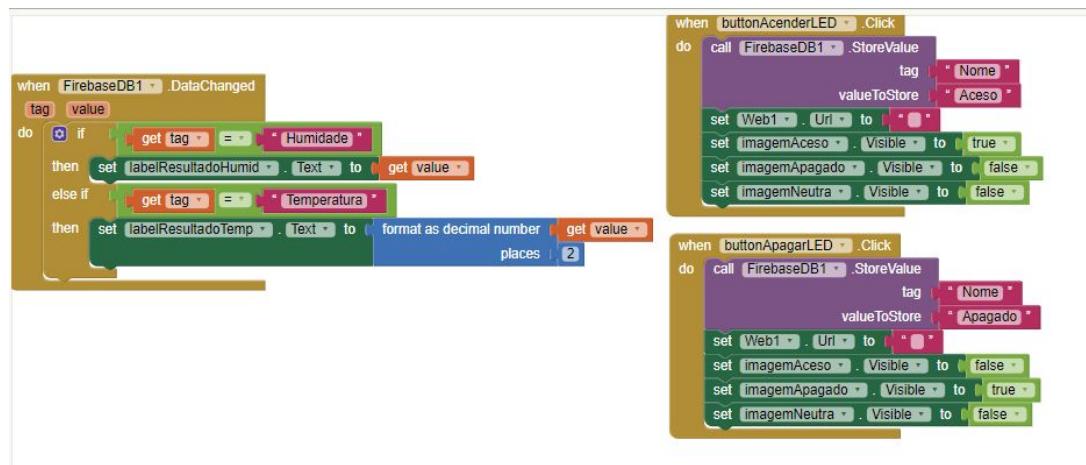


Em Propriedades/Properties, é dispostos os atributos editáveis de um componente selecionado

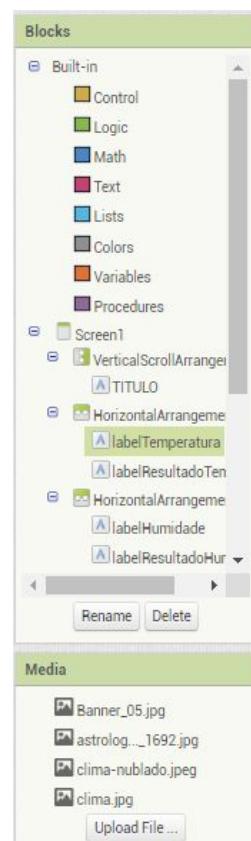


Ao acessar a aba ‘Blocks’, uma área de desenvolvimento é aberta, onde pode-se arrastar os blocos de código, e editá-los

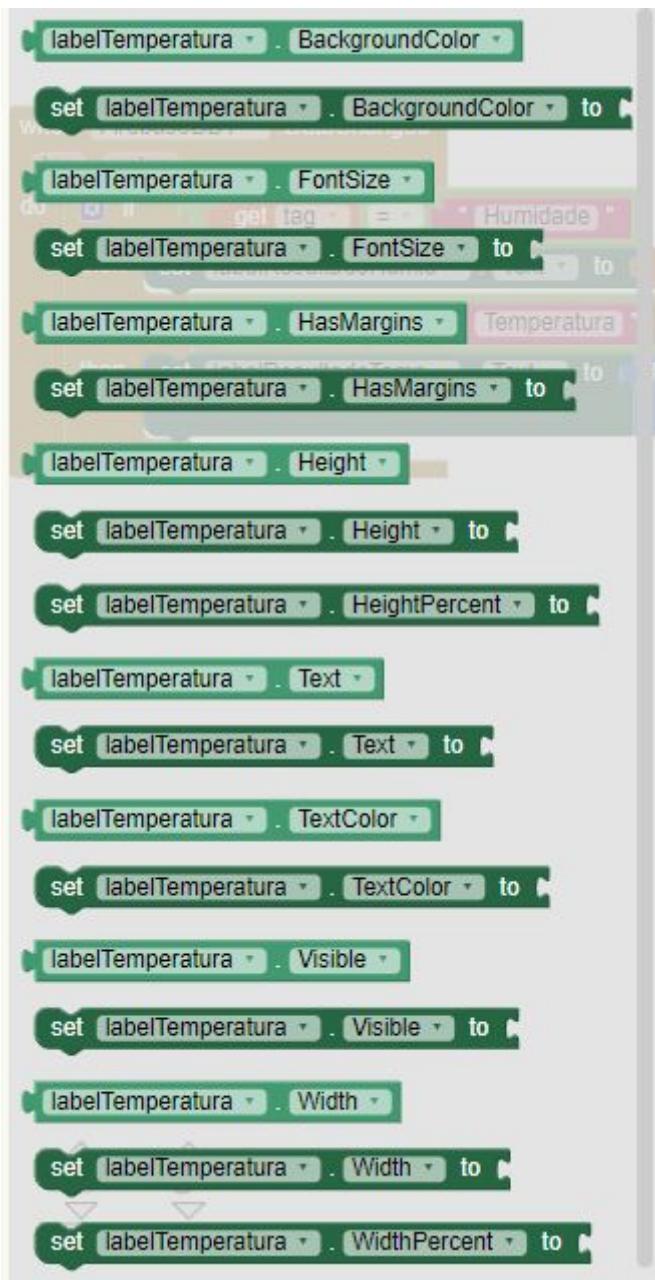
Observação: nem todo bloco pode ser acoplado a outro. Há sempre restrições que podem ser analisadas e compreendidas de acordo com o encaixe dos blocos, bem como o sentido lógico de cada um (seu papel).



Nesta área, é possível visualizar todos os componentes que já foram adicionados ao seu aplicativo, incluindo outros mais genéricos que podem ser utilizados e manipulados conforme a necessidade.

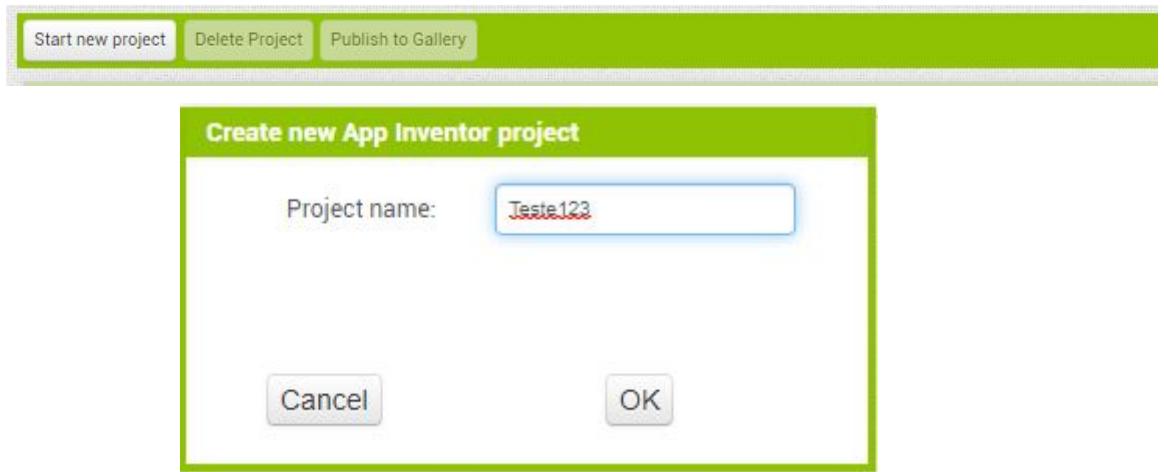


Ao selecionar um dos componentes que já foram adicionados à interface, é possível visualizar, arrastar e soltar na área de desenvolvimento blocos de código já desenvolvidos



## **Passo a Passo - Aplicação Mobile Temperatura e Humidade no AppInventor**

- 1) Após criar uma conta no AppInventor, acesse a aba ‘Start New Project’, e dê um nome a ele.



- 2) Quando um novo projeto é criado, é possível visualizar a interface de um celular.

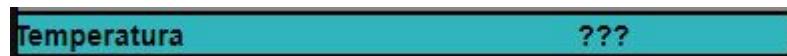


- 3) Do lado esquerdo, é possível visualizar os conjuntos de elementos disponibilizados pelo AppInventor. (Pode-se renomear o apelido de componente na aba superior direita ‘Components’ > ‘Rename’)
- Vá em Layout, selecione e arraste ‘VerticalScrollArrangement’ para dentro da tela.
  - Ao selecionar o componente na interface, ao lado direito, é possível visualizar suas Properties. Configure da seguinte forma:
    - AlignHorizontal Center : 3
    - AlignVertical Top : 1
    - BackgroundColor (Exemplo: Red)
    - Height 6 percent
    - Width 100 percent
    - Image None
    - Visible Checked
  - Vá em ‘User Interface’, selecione e arraste o componente ‘Label’ para dentro do ‘VerticalScrollArrangement’. Em Properties, configure da seguinte forma:
    - BackgroundColor None
    - FontBold Checked
    - FontItalic Unchecked
    - FontSize 20
    - FontTypeface default
    - HTMLFormat Unchecked
    - HasMargins Checked
    - Height Automatic
    - Width Automatic
    - x) Text (Exemplo: Fatec Ipiranga HT Sensor)
    - xi) TextAlignCenter : 1
    - xii) TextColor Default
    - xiii) Visible Checked

**Fatec Ipiranga HT Sensor**

- Vá em ‘Layout’, selecione e arraste o componente ‘HorizontalArrangement’ para a tela embaixo do colocado anteriormente. Configure da seguinte forma:
  - AlignHorizontal Left : 1
  - AlignVertical Top : 1
  - BackgroundColor (Exemplo: Blue, ou Customizada)
  - Height Automatic
  - Width Automatic
  - Image None

- vii) Visible Checked
- e) Vá em ‘User Interface’, selecione e arraste uma ‘Label’ para dentro do ‘HorizontalArrangement’. Configure da seguinte forma:
- i) BackgroundColor None
  - ii) FontBold Checked
  - iii) FontItalic Unchecked
  - iv) FontSize 20
  - v) FontStyle default
  - vi) HTMLFormat Unchecked
  - vii) HasMargins Checked
  - viii) Height Automatic
  - ix) Width 50 percent
  - x) Text Temperatura
  - xi) TextAlign left : 0
  - xii) TextColor Default
  - xiii) Visible Checked
- f) Vá em ‘User Interface’, selecione e arraste um ‘Label’ para dentro do ‘HorizontalArrangement’, ao lado da ‘Label’ anterior. Configure da seguinte forma:
- i) BackgroundColor None
  - ii) FontBold Checked
  - iii) FontItalic Unchecked
  - iv) FontSize 20
  - v) FontStyle default
  - vi) HTMLFormat Unchecked
  - vii) HasMargins Checked
  - viii) Height Automatic
  - ix) Width 50 percent
  - x) Text (Em branco, para testes, pode-se colocar qualquer texto para reconhecimento, Exemplo: ???)
  - xi) TextAlign Center : 1
  - xii) TextColor Default
  - xiii) Visible Checked



- g) Vá em ‘Layout’, selecione e arraste um ‘HorizontalArrangement’ para dentro da tela, embaixo do ‘HorizontalArrangement’ anterior. Configure da seguinte forma:
- i) AlignHorizontal Left : 1
  - ii) AlignVertical Top : 1
  - iii) BackgroundColor (Exemplo: Blue)

- iv) Height Automatic
  - v) Width Automatic
  - vi) Image None
  - vii) Visible Checked
- h) Vá em ‘User Interface’, selecione e arraste uma ‘Label’ para dentro do ‘HorizontalArrangement’ colocado anteriormente. Configure da seguinte forma:
- i) BackgroundColor None
  - ii) FontBold Checked
  - iii) FontItalic Unchecked
  - iv) FontSize 20
  - v) FontStyle Default
  - vi) HTMLFormat Unchecked
  - vii) HasMargins Checked
  - viii) Height Automatic
  - ix) Width 50 percent
  - x) Text Humidade
  - xi) TextAlignment left : 0
  - xii) TextColor Default
  - xiii) Visible Checked
- i) Vá em ‘User Interface’, selecione e arraste uma ‘Label1’ para dentro do ‘HorizontalArrangement’, ao lado da ‘Label’ colocada anteriormente. Configure da seguinte forma:
- i) BackgroundColor None
  - ii) FontBold Checked
  - iii) FontItalic Unchecked
  - iv) FontSize 20
  - v) FontStyle default
  - vi) HTMLFormat Unchecked
  - vii) HasMargins Checked
  - viii) Height Automatic
  - ix) Width 50 percent
  - x) Text (exemplo: ???)
  - xi) TextAlignment center : 1
  - xii) TextColor Default
  - xiii) Visible Checked

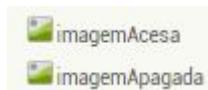


- j) Em ‘User Interface’, selecione e arraste um ‘Button’ para dentro da interface, abaixo do ‘Horizontal Arrangement’ colocado anteriormente. Configure da seguinte forma:

- i) BackGroundColor Default
- ii) Enabled Checked
- iii) FontBold Unchecked
- iv) FontItalic Unchecked
- v) FontSize 14.0
- vi) FontTypeface Default
- vii) Height 78 percent
- viii) Width 100 percent
- ix) Em ‘Image’, você deve procurar na internet duas imagens, elas irão representar o led aceso ou apagado, por exemplo, uma imagem de uma lâmpada acesa e outra de uma lâmpada apagada, onde serão trocadas dinamicamente quando a aplicação é aberta e quando o botão é clicado. Ao baixar as imagens, na Propriedade Image do Botão, você fará o Upload dessas duas imagens, mas deixará marcado ‘None’, pois trabalhará em conjunto com outros 2 elementos de imagem.
- x) Shape default
- xi) ShowFeedback Checked
- xii) Text Em Branco
- xiii) TextAlignment center : 1
- xiv) TextColor Default
- xv) Visible Checked



- k) Vá em ‘User Interface’, selecione e arraste um componente ‘Image’ para dentro da interface, abaixo do ‘Button’. Configure-o da seguinte forma:
- Para melhor identificação, sugerimos colocar o seguinte apelido para o este componente de imagem: imagemAcesa
  - Height Automatic
  - Width Automatic
  - Picture Selecione a imagem baixada que represente ACESO
  - RotationAngle 0.0
  - ScalePictureToFit Unchecked
  - Visible Unchecked (Isso, porque ela começará invisível, e se tornará visível quando for necessário, ou seja, dinamicamente)
- l) Vá em ‘User Interface’, selecione e arraste um componente ‘Image’ para dentro da interface, abaixo da ‘Image’ anterior. Configure-o da seguinte forma:
- Para melhor identificação, sugerimos colocar o seguinte apelido para o este componente de imagem: imagemApagada
  - Height Automatic
  - Width Automatic
  - Picture Selecione a imagem baixada que represente APAGADO
  - RotationAngle 0.0
  - ScalePictureToFit Unchecked
  - Visible Unchecked (Isso, porque ela começará invisível, e se tornará visível quando for necessário, ou seja, dinamicamente)



- m) Vá em ‘Experimental’, selecione e arraste ‘FirebaseDB’ para dentro da interface. Note que abaixo da tela do aparelho encontram-se os elementos ‘Non-visible components’.



Encontra-se lá pois não queremos que este componente seja visível na tela. Configure da seguinte forma:

i) FirebaseToken, Firebase URL e FirebaseBucket

- (1) Nesta etapa, é necessário que você acesse o Google, pesquise por Firebase, acesse o site, crie uma conta.
- (2) Na parte de iniciação, adicione um projeto, coloque um nome para ele, mantenha como aberto ou fechado para teste, como preferir.
- (3) Após o projeto ser criado, acesse a aba Database, lá você poderá criar sua base de dados.
- (4) Coloque um nome a tabela, bem como três colunas/linhas: Titulo: teste\_firebase, Humidade: 10, Nome: Aceso. Temperatura: 27.
- (5) No canto superior esquerdo, ao lado de Project Overview, selecione a Engrenagem > Configurações do Projeto > Contas de Serviço > Chaves secretas do Banco de Dados > Mostrar ‘Secret’, anote este valor, pois é a sua *FirebaseToken*
- (6) Voltando para a Database, um pouco acima dela é possível visualizar um link com https, anote este valor, pois é a sua *Firebase URL*
- (7) O nome da sua tabela do banco de dados, ‘teste\_firebase’, anote este nome, pois é a sua *FirebaseBucket*
- (8) Retorne para a interface do AppInventor, selecione seu componente “FirebaseDB”, e preencha os campos FirebaseToken, FirebaseURL e FirebaseBucket com os valores anotados dos passos anteriores]

ii) Us Default Unchecked

iii) Persist Unchecked



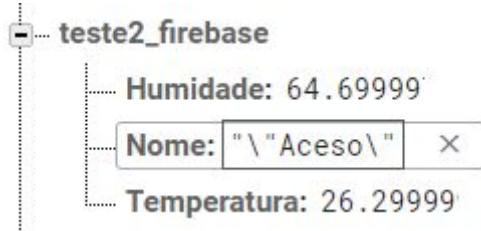
<https://inventapp-26876.firebaseio.com/>

Banco de dados

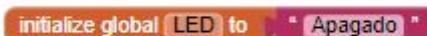
Secret

inventapp-26876

\*\*\*\*\*



- n) No canto superior direito da interface, é possível encontrar um botão chamado Designer e outro chamado Blocks. Clique no botão Blocks
- o) No lado esquerdo, estão todos os blocks e os seus componentes adicionados ao projeto. Clique em ‘Variables’, selecione e arraste ‘initialize global name to’ para a Viewer.
  - i) Edite o texto ‘name’ para ‘LED’
  - ii) Vá na aba de blocks ‘Text’, arraste um componente de texto em branco, e ligue-o ao ‘initialize global LED to’
  - iii) Edite o texto do block roxo de texto para ‘Apagado’

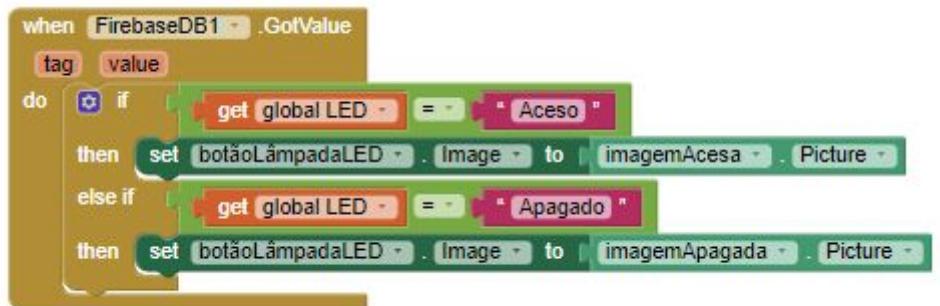


- p) Selecione o componente ‘Screen1’ (o componente da tela), pegue o block ‘When Screen1 Initialize - do’, arraste para a Viewer
  - i) Selecione o componente ‘FirebaseDB1’, pegue o block ‘call FirebaseDatabase1.GetValue - tag - valueIfTagNotThere’, arraste para a Viewer, encaixando abaixo do ‘Screen Initialize’ colocado anteriormente
  - ii) Selecione os blocks de Text em branco, coloque um encaixado em ‘tag’ e outro encaixado em ‘valueIfTagNotThere’. Edite o texto de ambos, o primeiro para ‘Nome’, e o segundo para ‘Erro de Nome’

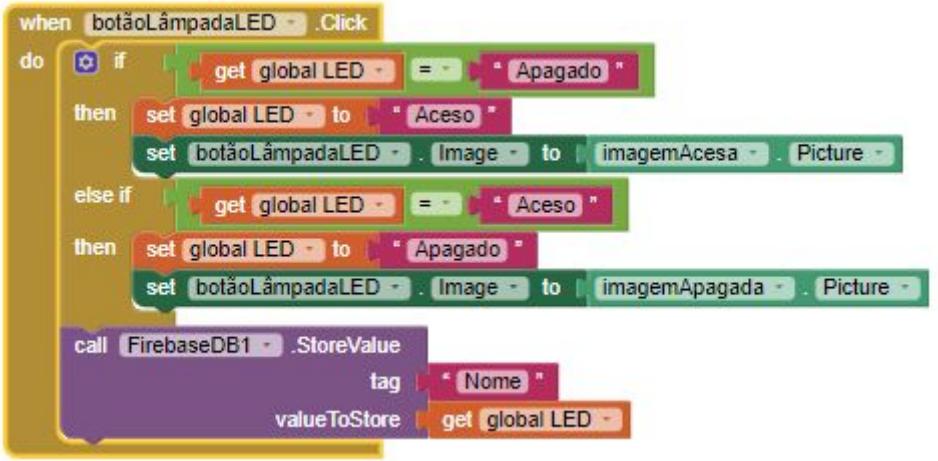


- q) Selecione o componente ‘FirebaseDB1’, pegue o block ‘when FirebaseDatabase1.GotValue - do’, arraste e solte na Viewer
  - i) Vá em blocos ‘Control’, arraste e solte dentro do ‘GotValue’ um bloco ‘ifThen’
  - ii) Para o if, coloque um block da aba ‘Logic’: VAZIO = VAZIO

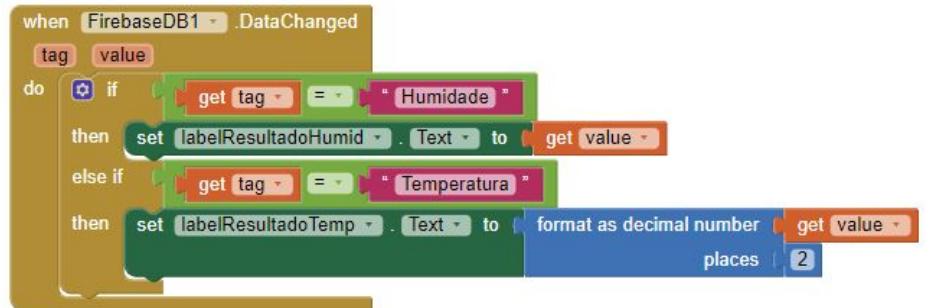
- iii) No primeiro espaço VAZIO, vá em ‘Variables’, pegue um block ‘get \_\_\_\_’, arraste-o e coloque-o la, e selecione o nome ‘LED’
- iv) No segundo espaço VAZIO, vá em Text e coloque um espaço de texto em branco. Edite-o e coloque ‘Aceso’
- v) Para o ‘then’, vá no componente ‘Button’, pegue o block ‘set button.image to’, arraste e solte no then. Depois, no componente ‘imagemAcesa’, pegue o block ‘imagemAcesa.Picture’, e coloque-o após o block anterior.
- vi) Agora, vamos precisar de um elseif, para isso, clique na pequena engrenagem azul localizada ao lado da palavra if do componente que já está na tela, abrirá um pequeno menu, arraste o elseif para embaixo do if da direita do pequeno menu, assim ele irá atualizar o bloco real da Viewer
- vii) Copie e cole os dois blocos, tanto o verde lógico comparativo quando o block do SET, cole e arraste-os para as mesmas posições, porém agora do elseif.



- r) Vá para o componente ‘Button’ e pegue o block do evento ‘click’ (when button.Click - do), arraste e solte na Viewer
  - i) Pegue um block condicional IF, arraste e solte na Viewer, dentro desse evento CLICK
  - ii) Siga o mesmo esquema de bloco comparativo e SET feitos nos passos anteriores
  - iii) Vá no componente ‘FirebaseDB1’, pegue a chamada ‘call FirebaseDB1.StoreValue - tag - valueToStore’, arraste e solte fora do bloco IF, ao final do block WHEN
  - iv) Efetue as edições nos componentes, de forma que fique assim:



- s) Vá para o componente ‘FirebaseDB1’ e selecione o componente ‘when FirebaseDatabase1.DataChanged - do’
- Arraste um bloco condicional IF para esse WHEN
  - Utilize o comparativo lógico, as variables, os campos de texto em branco, e o bloco ‘Math’ (matemático) de FORMAT, para formar o seguinte bloco de código:



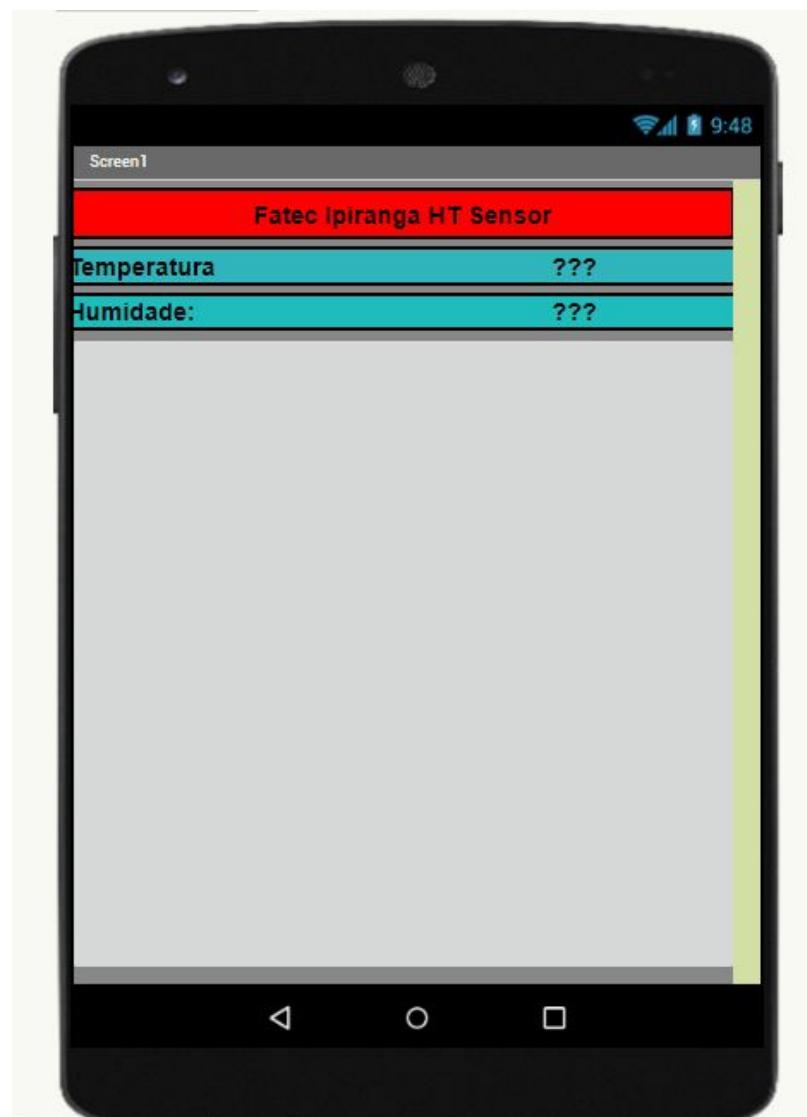
- t) Neste momento, a integração entre o Firebase e a Interface do ApplInventor já devem estar funcionando. Há diversas formas de se efetuar testes.
- u) Na Barra de Ferramentas na parte superior do ApplInventor, é possível identificar uma opção chamada Connect
- para utilizar o AI Companion, é necessário que você instale em seu celular um aplicativo gratuito chamado MIT Ai Companion, onde você poderá conectar via código ou QR Code, e sincronizar a aplicação presente no ambiente de teste com o seu aparelho celular, podendo testar em paralelo com o desenvolvimento
  - para utilizar o emulador (pouco recomendado), é necessário baixar o mesmo aplicativo, porém pelo site oficial, instalar e rodar antes de clicar em Emulator, para então sincronizar,

havendo teste em paralelo porém pelo computador. Nós não recomendamos somente pelo fato de ser mais lento, sendo que as demais opções garantem testes mais rapidamente

- iii) a opção USB permite que você conecte a aplicação com o seu celular diretamente via USB
- iv) você pode utilizar as opções na Aba Build também. Lá você pode criar um QR code para instalar o apk do projeto em seu aparelho, ou salvar um arquivo .apk em seu computador e então o transferir via cabo USB para o diretório do seu aparelho, e abrir pelo celular

v) Ao final do projeto:

w) Interface:



### x) Codificação

```
initialize global [LED] to [Apagado]
```

```
when [FirebaseDB1] .DataChanged
  tag [value]
  do [if [get [tag] = "Humidade"]
    then [set [labelResultadoHumid] .Text to [get [value]]]
    else if [get [tag] = "Temperatura"]
    then [set [labelResultadoTemp] .Text to [format as decimal number [get [value]] places 2]]]
```

```
when [Screen1] .Initialize
do [call [FirebaseDB1] .GetValue
  tag ["Nome"]
  valueIfTagNotThere "Erro de Nome"]
```

```
when [FirebaseDB1] .GetValue
  tag [value]
  do [if [get [global LED] = "Aceso"]
    then [set [botãoLâmpadaLED] .Image to [imagemAcesa] .Picture]
    else if [get [global LED] = "Apagado"]
    then [set [botãoLâmpadaLED] .Image to [imagemApagada] .Picture]]
```

```
when [botãoLâmpadaLED] .Click
do [if [get [global LED] = "Apagado"]
  then [set [global LED] to Returns the value of this variable.
        set [botãoLâmpadaLED] .Image to [imagemAcesa] .Picture]
  else if [get [global LED] = "Aceso"]
  then [set [global LED] to "Apagado"
        set [botãoLâmpadaLED] .Image to [imagemApagada] .Picture]]
call [FirebaseDB1] .StoreValue
  tag ["Nome"]
  valueToStore [get [global LED]]
```

**Referências:**

**Firebase**

[https://firebase.google.com/?gclid=EA1alQobChMlzL747qmj5QIVjoWRCh3S2wZlEAAYASAAEglwhfD\\_BwE](https://firebase.google.com/?gclid=EA1alQobChMlzL747qmj5QIVjoWRCh3S2wZlEAAYASAAEglwhfD_BwE)

**AppInventor** - <https://appinventor.mit.edu/>

## Rascunho

instalação bibliotecas alsa no raspberry  
<https://scribles.net/updating-alsa-on-raspbian-stretch/>

<https://www.bishoph.org/step-by-step-raspberry-pi-offline-voice-recognition-with-sopare/>

---

## Raspberry Pi Beginners Guide v1

14/08/19 14:00

- Dispositivo para navegar a web, jogar jogos, escrever programas ou criar os próprios circuitos e dispositivos físicos
- Computador impresso em única placa de circuito. Pequeno, possui o tamanho de um cartão de crédito
- Apesar do tamanho diminuto, possui o mesmo poder de computadores maiores, mas não tão veloz quanto
- Todos os modelos Raspberry Pi são compatíveis, softwares escritos para um modelo rodarão em qualquer outro (inclusive sistemas operacionais)
  
- Raspberry Pi 3 B+ possui: circuito intergrado (inclui CPU - unidade de processamento central - e GPU - unidade de processamento gráfico), memória principal RAM - acesso aleatório de memória -, radio (permite comunicação wireless, wifi radio e bluetooth radio), chip de controle USB e rede e chip PMIC - Circuito Integrado de Gestão de Força - (lida com a entrada USB de força, que faz o Raspberry rodar)

- Entradas Raspberry Pi 3 B+: USBs 2.0, Ethernet port (com led que acende se está conectado), conector 3.5mm audio-visual(AV) (para fone de ouvido - também pode ser conectado em TVs, projetores e outros dispositivos de vídeo compatíveis), conector para câmera CSI - Interface Serial de Câmera -, entrada HDMI, USB de força (para conectar a uma fonte de energia), conector de telas DSI - Interface Serial de Exibição - (para conexão da tela touch Raspberry Pi), 40 pinos de metal GPIO - Entrada/Saída de Propósito Geral - (para comunicar com diversos hardwares/dispositivos), conector PoE - Força sob Internet - (outro conector para conexão de internet) e entrada para MicroSD (memória secundária)

- SO Raspbian

- Usando o Raspberry Pi - porcentagem no topo direito da tela: monitor de CPU, informa quanta CPU está sendo usada

- Possui o software Scratch 2, para uso da linguagem de programação em blocos Scratch. O Scratch é uma linguagem de programação visual desenvolvida pelo MIT (Instituto de Tecnologia do Massachusetts). Possui uma interface infantil e serve como introdução a programação

- Possui o software IDE Thonny para uso da linguagem de programação Python

- Python:

- não necessita de ;) ao final das linhas de código, a quebra de linha comunica final de linha

- cadeamento não necessita de {}, apenas correta indentação

- comando print() imprime na tela. print("Mensagem") imprime "Mensagem" ao ser executado

- for i in range (): para variável i que acabou de ser definida, dentro do alcance de algum número, começa em 0 e repete X vezes - repetidor de comando(s) de linha.

```
for i in range (10):
```

```
print("Mensagem", i)      irá imprimir linha por linha  
"Mensagem 0", "Mensagem 1" até "Mensagem 9"
```

14/08/19 17:00

-Python:

- while, if, else, elif, etc. Diversos comandos funcionam como em C, não há necessidade de colocar condição dentro de (). Terminar a linha de código em (:). Exemplo: while x<0:

- não há necessidade de declarar o tipo da variável, declarada automaticamente conforme o uso da variável. Exemplo: x=3.0

- comando input() faz entrada de dados. x=input("Digite o valor") imprime "Digite o valor" na tela e ao usuário entrar com um valor ele é atribuído a variável x

- comando import importa bibliotecas. Exemplo import turtle (importa a biblioteca turtle, um robô que anda em linha reta, vira e pode levantar e abaixar uma caneta)

- é possível importar apenas parte de uma biblioteca. Exemplo: from time import sleep

- quando usando instruções contidas em uma biblioteca, deve-se entrar inicialmente como nome dela acrescido de (.) e então o nome da instrução/função. Exemplo: turtle.Turtle()

- Exemplo: pat = turtle.Turtle()

- comando def define uma função. Exemplo: def Somatorio:

- para uso do GPIO, necessário importar a biblioteca gpiotzero

- led no GPIO:

- led = LED(25) atribui a variável led que aquilo conectado em GP25 no GPIO é um led (para conectar um led, conectá-lo a um resistor e então escolher um dos GPXX para conectar o resistor, a outra ponta do led será aterrada)

- led.on() e led.off() acende e apaga o led

-

```
from gpiozero import LED
```

```
from time import sleep
```

```
led = LED(25)
```

```
while True:
```

```
    led.on()
```

```
    sleep(1)
```

```
    led.off()
```

```
    sleep(1)
```

Este programa irá importar o controle de leds da biblioteca gpiozero e a função sleep da biblioteca time. A variável 'led' recebe um led conectado em GP25. Em loop infinito, pois a condição True nunca é quebrada, o led irá acender, o programa será pausado por um segundo, o led irá apagar, o programa será pausado por um segundo.

- botão no GPIO:

- button = Button(2) atribui a variável button que aquilo conectado em GP2 no GPIO é um botão (para conectar um botão, conectar a um dos GPXX um pino do botão, o pino ao lado será aterrado)

- button.wait\_for\_press() espera o botão ser pressionado e então faz alguma coisa

```
-  
from gpiozero import LED  
  
from time import sleep  
  
from gpiozero import Button  
  
button = Button(2)  
  
led = LED(25)  
  
button.wait_for_press()  
  
led.on()  
  
sleep(3)  
  
led.off()
```

Este programa irá importar o controle de leds e de botões da biblioteca gpiozero e a função sleep da biblioteca time. A variável 'button' recebe um botão conectado em GP2 e a variável 'led' recebe um led conectado em GP25. Ao botão ser pressionado, o led se acende, o programa é pausado por 3 segundos, o led apaga.

- buzina no GPIO:

- buzzer = Buzzer(15) atribui a variável buzzer que aquilo conectado em GP15 no GPIO é uma buzina (para conectar uma buzina, conectar a um dos GPXX um pino da buzina, o pino ao lado será aterrado)

- buzzer.on() e buzzer.off() liga e desliga a buzina

- jogo, quem pressiona o botão primeiro:

```
from gpiozero import LED, Button  
  
from time import sleep  
  
from random import uniform  
  
from os import _exit
```

```

left_name = input("Left player name is ")

right_name = input ("Right player name is ")

led = LED(4)

right_button = Button(15)

left_button = Button(14)

led.on()

sleep(uniform(5, 10))

led.off()

def pressed(button):

    if button.pin.number == 14:

        print(left_name + " won the game")

    else:

        print(right_name + " won the game")

    _exit(0)

right_button.when_pressed = pressed

left_button.when_pressed = pressed

```

- Importa LED e Button da biblioteca gpiozero, sleep da time, uniform da random (a biblioteca random permite diversas ações de aleatorizar no programa, ao usar uniform as opções aleatórias são uniformes) e \_exit da os (a biblioteca os é do sistema operacional, \_exit termina o programa). A variável left\_name recebe o nome entrado por um dos jogadores e a right\_name o nome do outro jogador. Há um led na GP4 e botões nas GP14 e GP15. O led acende, o programa pausa aleatoriamente entre 5 e 10 segundos, o led apaga. É definida a função pressed, que irá verificar o botão: se o botão ativo tiver pinagem 14 (GP14) então imprime que o jogador da esquerda vence, senão imprime que o jogador da direita vence; o programa se encerra. Se o botão for apertado pelo jogador da direita, chama a função pressed. Se o botão for apertado pelo jogador da esquerda, chama a função pressed.

- Câmera:

- Entrar nas opções do Pi, entrar em Interfaces e habilitar Camera

- abrir o Terminal, os comandos raspistill e raspivid tiram foto e gravam vídeo, respectivamente

- no Terminal, raspistill -o test.jpg irá ativar a câmera, que irá mostrar uma tela por 5 segundos e então irá salvar uma foto do segundo final na home (home/pi) com o nome test.jpg

- os comandos -vt e -hf rotacionam vertical e horizontalmente. Exemplo: raspistill -vf -hf -o cam2.jpg irá tirar uma foto de ponta-cabeça

- no Terminal, os comandos para raspistill são os mesmos para raspivid, mas ao invés de tirar uma foto, irá gravar 5 segundos de vídeo. Exemplo: raspivid -o vid.h264

- raspivid -o video.h264 -t 10000 aumenta o tempo de gravação, ao invés de 5 segundos este vídeo terá 10 segundos

## - Python:

- biblioteca picamera permite controle da câmera

- PiCamera, da biblioteca picamera, permite o controle do Módulo de Câmera com o comando PiCamera()

- vamos dar um apelido para PiCamera a fim de simplificar os comandos, usando: camera = PiCamera()

- camera.start\_preview() liga a câmera e  
camera.stop\_preview() desliga a câmera

- camera.rotation permite rotacionar o display. Exemplo:  
camera.rotation = 180

- camera.capture('/home/pi/Desktop/image.jpg') salva uma imagem daquilo que está sendo exibido na câmera na home, com o nome de image.jpg

- camera.start\_recording('/home/pi/Desktop/video.h264') inicia uma gravação daquilo que está sendo exibido na câmera na home, com o nome de video.h264 . Para parar a gravação, usar o comando camera.stop\_recording(). Exemplo:

```
from picamera import PiCamera

from time import sleep

camera = PiCamera()

camera.start_preview()

camera.start_recording('/home/pi/Desktop/video.h264')

sleep(10)

camera.stop_recording()

camera.stop_preview()
```

Grava um vídeo de 10 segundos

- comando try roda tudo que tem dentro dele, exceto caso em comando except. Exemplo:

```
from time import sleep
```

```
s=0  
while True:  
    try:  
        s+=1  
    except KeyboardInterrupt:  
        break  
    sleep(1)
```

Loop infinito onde a variável s é somada +1 a cada segundo até que o usuário interrompa usando o teclado (atualho ctrl+c)

- %0Xd é um recurso que coloca n zeros a frente de um número inteiro a fim dele ter tamanho de X dígitos. Exemplo:

```
a=14  
print("%05d" %a)    imprime na tela 00014
```

- Projeto Stop Motion:

- No Python:

```
from picamera import PiCamera  
from time import sleep  
from gpiozero import Button  
  
camera = PiCamera()  
  
button = Button(2)  
  
camera.start_preview()  
  
frame = 1  
  
while True:
```

```
try:
```

```
    button.wait_for_press()
```

```
camera.capture('/home/pi/Desktop/animation/frame%03d.jpg' % frame)
```

```
    frame += 1
```

```
except KeyboardInterrupt:
```

```
    camera.stop_preview()
```

```
    break
```

Em loop infinito, através de botão conectado no GP2, cada vez que ele é pressionado, a câmera, que está ligada, salva imagem na pasta animation na Tela de Trabalho daquilo que está em display, iniciando em nome frame001.jpg e indo até frame999.jpg (se o botão for apertado mais de 999 vezes, irá começar a sobrescrever as imagens anteriores, primeiro salvando imagem inédita em frame000.jpg e então começando a sobrescrever a partir de frame001.jpg). Caso o usuário interrompa usando o teclado, a câmera é desligada e o loop quebrado, chegando ao fim do programa

- No Terminal:

```
- cd Desktop/animation          (cd      =      change  
directory)
```

```
- avconv -r 1 -i frame%03d.jpg -r 10 animation.h264
```

- avconv é um programa que converte imagens estáticas em vídeo, no caso as n imagens frameXXX.jpg em vídeo de nome animation.h264

- -r 10 são os frames do seu vídeo, no caso foram usado 10 frames

- outros comandos picamera do Python:

```
- camera.awb_mode = 'auto'      balanço     de      branco  
automático, pode ser usado qualquer um destes outros: off, auto, sunlight,  
cloudy, shade, tungsten, fluorescent, incandescent, flash, horizon
```

- camera.brightness = XX muda o brilho da câmera, entre 0 e 100

- camera.color\_effects = None remove efeitos ativos.  
camera.color\_effects = (128, 128)deixa em preto-e-branco

- camera.contrast = XX muda o contraste da câmera, entre -100 e 100

- camera.crop = (0.0, 0.0, 1.0, 1.0) diminui o tamanho do dosplay capturado, os dois primeiros são as coordenadas e os últimos o tamanho a ser cortado

- camera.exposure\_compensation = XX controla quanta luz é capturada pela câmera, entre -25 e 25