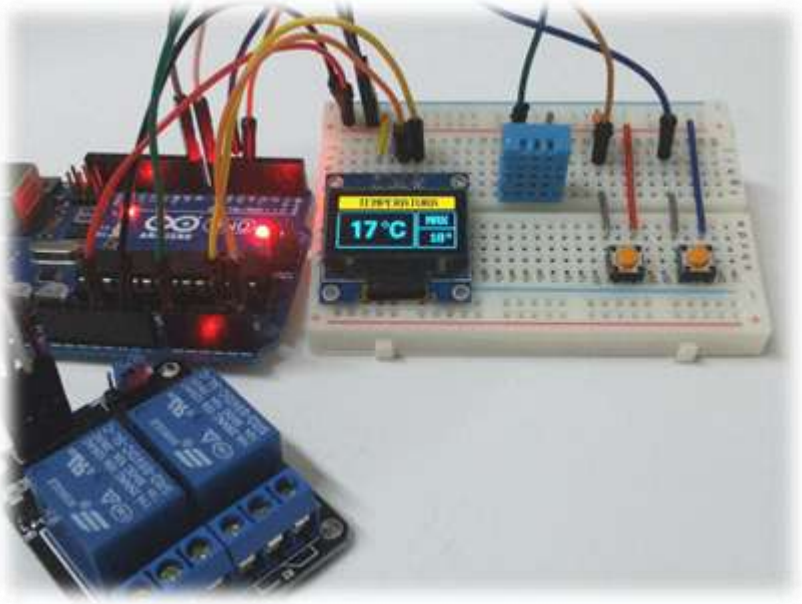
Governo do Estado de Pernambuco
Secretaria de Educação
Secretaria Executiva de Educação Profissional
Escola Técnica Estadual Professor Agamemnon Magalhães
ETEPAM

Projetos com Arduino utilizando módulos



Prof. JENER TOSCANO LINS E SILVA

Projetos Propostos

1 - Carrinho autônomo com Arduíno

(Thaenny e Jeyfferson)

2 - Contador de visitantes usando led IR e fototransistor

(Carlos e Flávia)

3 - Braço robótico com 4 articulações controlado por LDRs

(Lucas e Jailson)

4 - Arduino Parking Lot

(Junior e Luciano)

5 - Arduino Radar

(Fillipe)

6 - Arduino Bluetooth

(Pedro Luiz e Alan)

7 - Acionando uma Lâmpada pela rede Ethernet

8 - Robô Industrial com sensor de presença

Plataformas Arduino Profissionais

Projetos com Arduino ocupando menos espaço

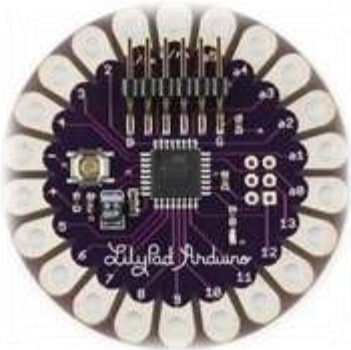
Arduino Nano

Arduino Pro Mini – 5 Volts

Dimensões: 33.3 x 18.0 (mm)

Adaptador USB/TTL

Arduino Lilypad



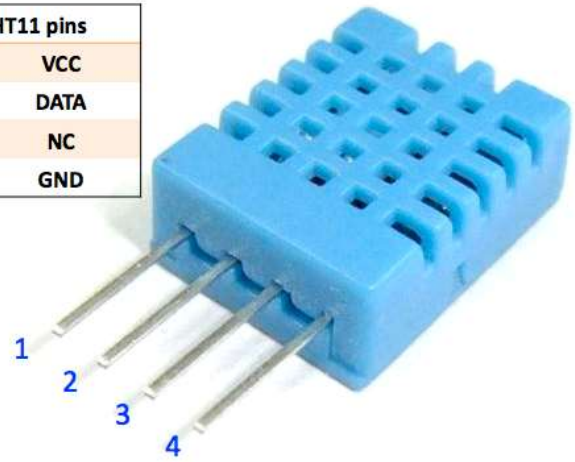
Módulos a serem estudados

- Sensor de Temperatura e Umidade
- Sensor de Gás
- Sensor de Presença
- Sensor Ultrassônico
- Controle Remoto Infravermelho
- Módulos Transmissores Rádio Frequência
- Acionamento via Bluetooth
- Comunicação por Ethernet
- Comunicação via Wi-Fi

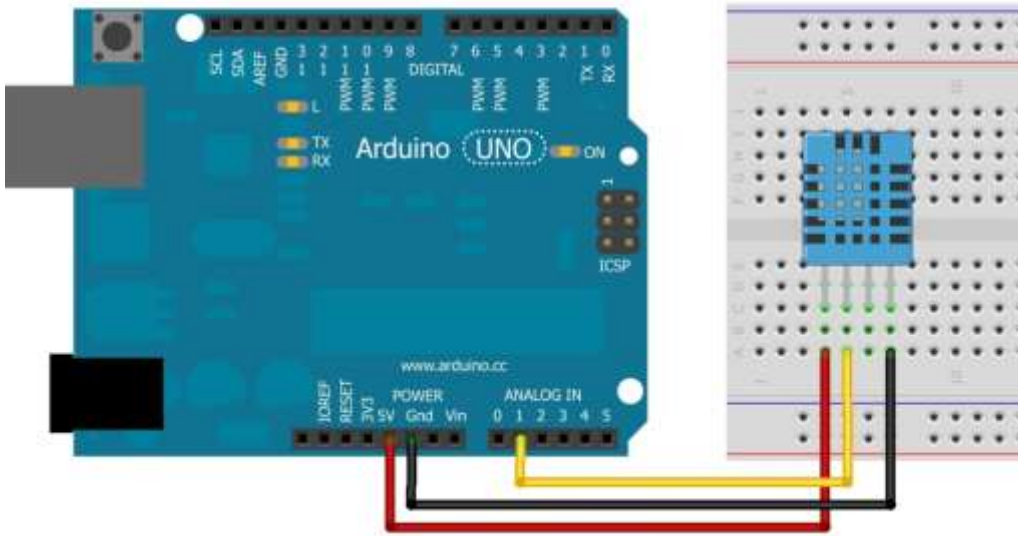
Sensor de Temperatura e Umidade (DHT11)

- O **sensor DHT11** é um sensor de temperatura e umidade , que permite medir temperaturas de **0 a 50 Celsius**, e **umidade na faixa de 20 a 90%.**
- Não é um sensor extremamente rápido e preciso, por isso não é recomendada a utilização em ambientes de alto risco.
- Sua faixa de precisão para **temperatura é de 2 gr**aus, e de **umidade, 5%.**
- O mais comum é encontrá-lo em forma de **módulo de três pinos**: Vcc, Data e Gnd.

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND



Montagem do Sensor de Umidade



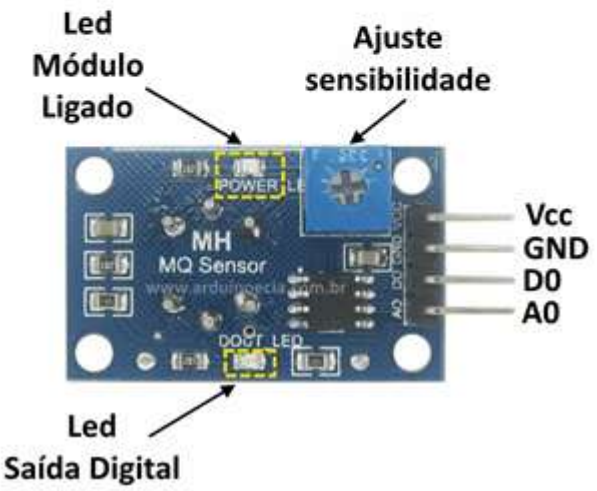
	DHT11	DHT22
		
Alimentação	3 - 5.5V	3.3 - 6V
Faixa de leitura - Umidade	20 - 80 %	0 - 100 %
Precisão - Umidade	5%	5%
Faixa de leitura - Temperatura	0 - 50 °C	-40 - 125 °C
Precisão - Temperatura	+/- 2 °C	+/- 0,5 °C
Intervalo entre medições	1s	2s

Programa do Sensor de Umidade

```
//Programa : Sensor de umidade e temperatura DHT11

#include <dht.h>
#define dht_dpin A1 //Pino DATA do Sensor ligado na porta Analogica A1

dht DHT; //Inicializa o sensor

void setup()
{
  Serial.begin(9600);
  delay(1000);//Aguarda 1 seg antes de acessar as informações do sensor
}

void loop()
{
  DHT.read11(dht_dpin); //Lê as informações do sensor
  Serial.print("Umidade = ");
  Serial.print(DHT.humidity);
  Serial.print(" %  ");
  Serial.print("Temperatura = ");
  Serial.print(DHT.temperature);
  Serial.println(" Celsius  ");

  //Não diminuir o valor abaixo. O ideal é a leitura a cada 2 segundos
  delay(2000);
}
```

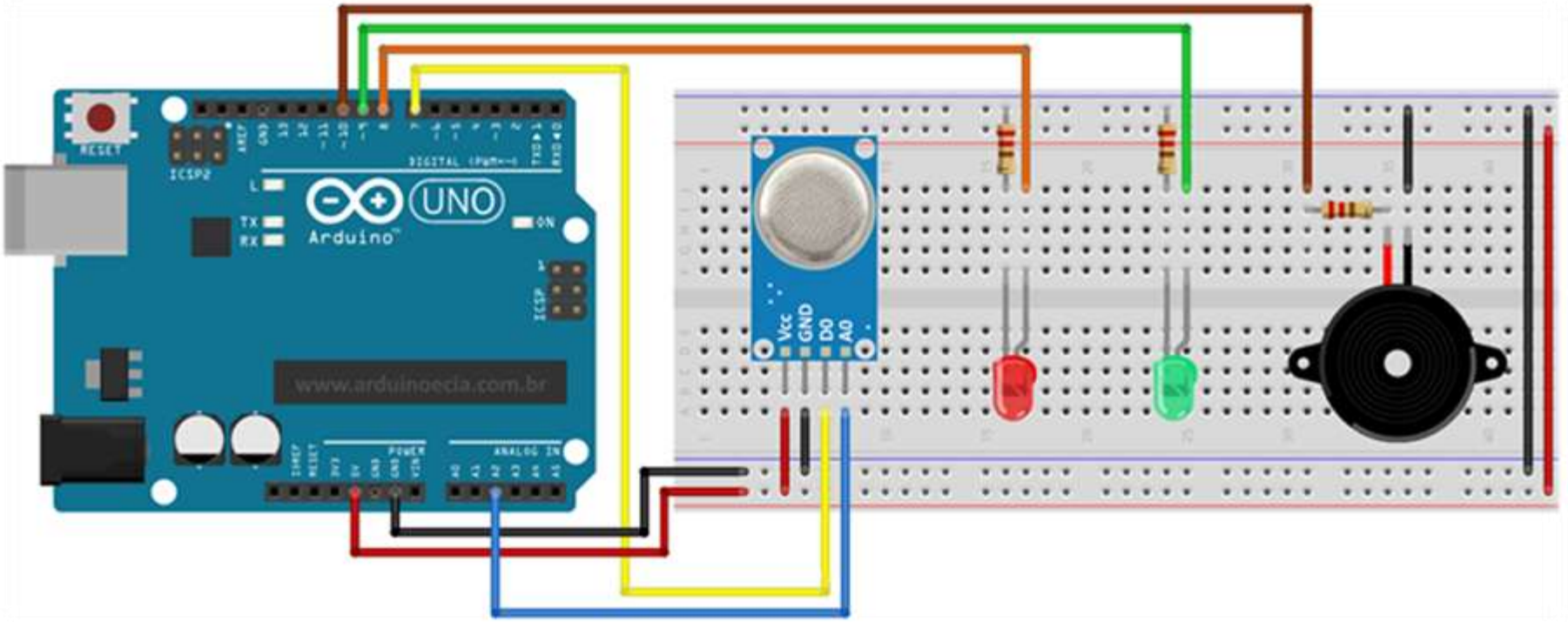
Sensor de Gás
com o Módulo MQ-2

- Sistema de detecção de gás importante item de segurança no projeto de automação residencial.
- O **sensor MQ-2** é um detector de gás e fumaça que pode indicar a presença de gás de cozinha.
- Além de outros gases: **Propano, Metano, Hidrogênio**.
- Seu nível de detecção vai de **300 a 10.000 ppm** (partes por milhão), ajustáveis por um potenciômetro na parte de trás do módulo.
- Um chip comparador **LM393** é responsável por ler as informações do sensor e converter essas informações em sinais para o microcontrolador.



Montagem do Sensor de Gás

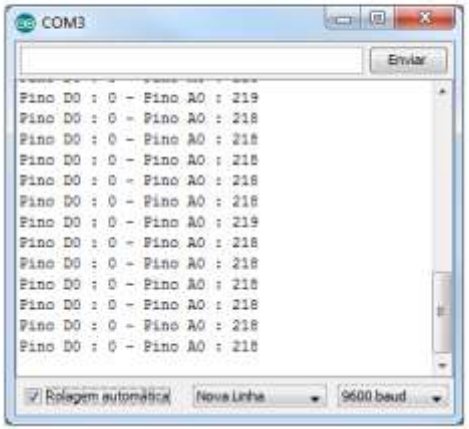
- A porta digital 7 será utilizada para ligação ao pino D0 do módulo, e a porta analógica A2 será ligada ao pino A0 do módulo.
- As portas digitais 8, 9 e 10 serão utilizadas para acionar um led vermelho e um buzzer (gás detectado), e um led verde no modo normal de operação (sem alarme).



Programa do Sensor de Gás

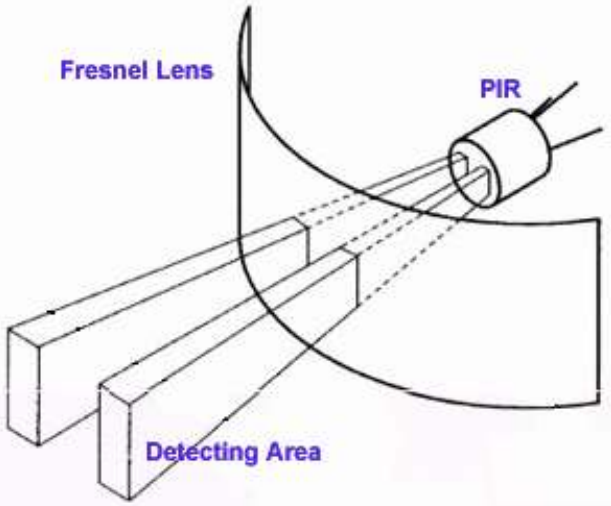
```
// Programa : Alarme com sensor de gas MQ-2

// Definicoes dos pinos dos leds e buzzer
int pin_led_verm = 8;
int pin_led_verde = 9;
int pin_buzzer = 10;
// Definicoes dos pinos ligados ao sensor
int pin_d0 = 7;
int pin_a0 = A2;
int nivel_sensor = 250;

void setup()
{
  // Define os pinos de leitura do sensor como entrada
  pinMode(pin_d0, INPUT);
  pinMode(pin_a0, INPUT);
  // Define pinos leds e buzzer como saida
  pinMode(pin_led_verm, OUTPUT);
  pinMode(pin_led_verde, OUTPUT);
  pinMode(pin_buzzer, OUTPUT);
  // Inicializa a serial
  Serial.begin(9600);
}
```

O valor das variáveis valor_digital (porta D0) e valor_analogico (porta A0) no serial monitor

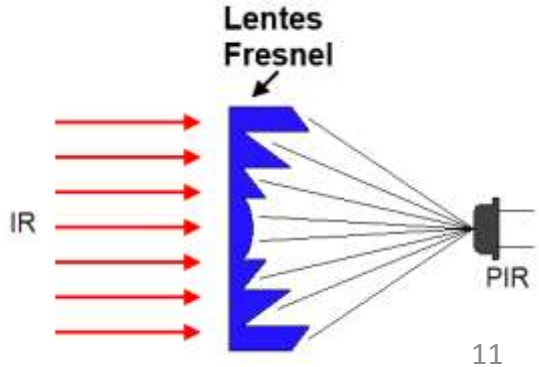



```
void loop()
{
  // Le os dados do pino digital D0 do sensor
  int valor_digital = digitalRead(pin_d0);
  // Le os dados do pino analogico A0 do sensor
  int valor_analogico = analogRead(pin_a0);
  // Mostra os dados no serial monitor
  Serial.print("Pino D0 : ");
  Serial.print(valor_digital);
  Serial.print(" Pino A0 : ");
  Serial.println(valor_analogico);
  // Verifica o nivel de gas/fumaca detectado
  if (valor_analogico > nivel_sensor)
  {
    // Liga o buzzer e o led vermelho, e
    // desliga o led verde
    digitalWrite(pin_led_verm, HIGH);
    digitalWrite(pin_led_verde, LOW);
    digitalWrite(pin_buzzer, HIGH);

  }
  else
  {
    // Desliga o buzzer e o led vermelho, e
    // liga o led verde
    digitalWrite(pin_led_verm, LOW);
    digitalWrite(pin_led_verde, HIGH);
    digitalWrite(pin_buzzer, LOW);
  }
  delay(100);
}
```

Sensor de Presença

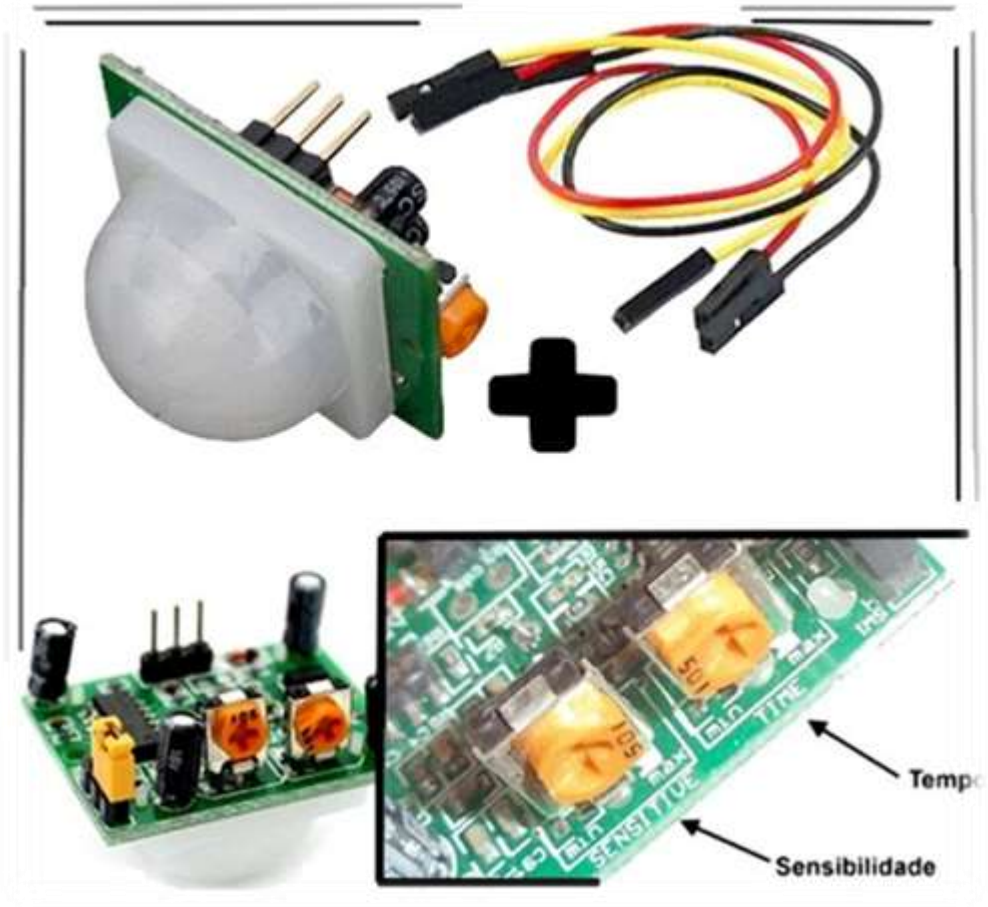


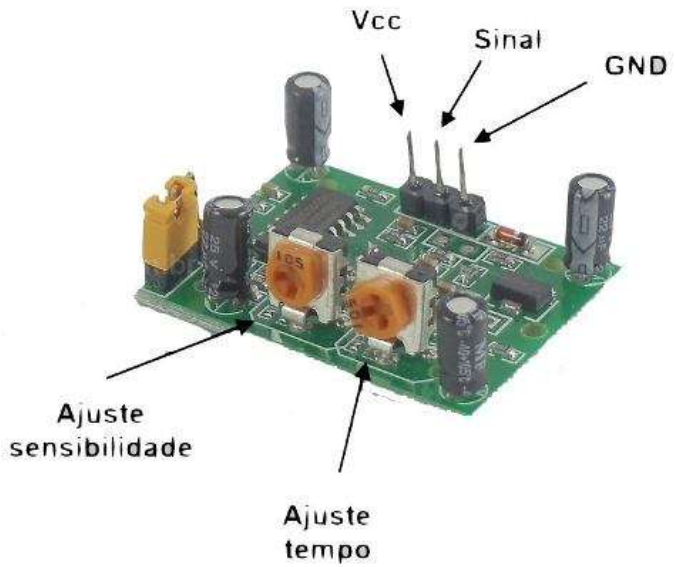
- O sensor consegue detectar o movimento de objetos que estejam em uma área de 3 até 7 metros.
- É possível ajustar a duração do tempo de espera para estabilização do PIR através do "*trimpot*" a baixo do sensor bem como sua sensibilidade.
- A estabilização pode variar entre 5-200 seg (Default: 5seg).



Sensor de Presença

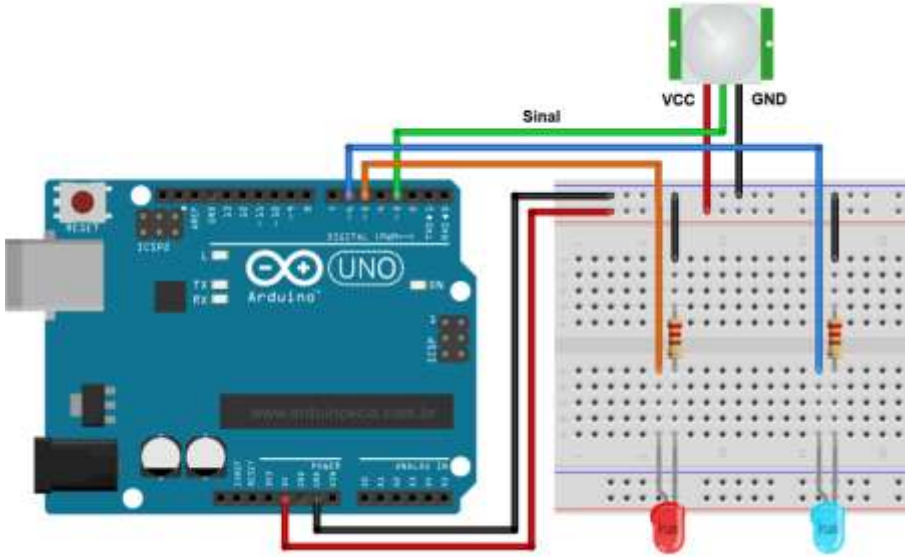
Sensor PIR





Montagem do Sensor de Presença

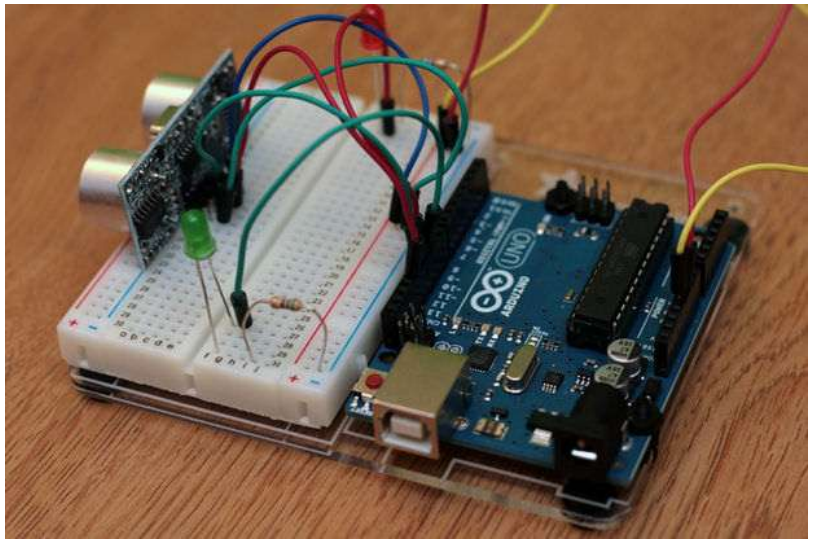
```
// Programa : Sensor de presenca com modulo PIR

int pinoledverm = 5; //Pino ligado ao led vermelho
int pinoledazul = 6; //Pino ligado ao led azul
int pinopir = 3;  //Pino ligado ao sensor PIR
int acionamento;  //Variavel para guardar valor do sensor

void setup()
{
  pinMode(pinoledverm, OUTPUT); //Define pino como saida
  pinMode(pinoledazul, OUTPUT); //Define pino como saida
  pinMode(pinopir, INPUT);   //Define pino sensor como entrada
}

void loop()
{
 acionamento = digitalRead(pinopir); //Le o valor do sensor PIR
 if (acionamento == LOW)  //Sem movimento, mantem led azul ligado
 {
    digitalWrite(pinoledverm, LOW);
    digitalWrite(pinoledazul, HIGH);
 }
 else  //Caso seja detectado um movimento, aciona o led vermelho
 {
    digitalWrite(pinoledverm, HIGH);
    digitalWrite(pinoledazul, LOW);
 }
}
```

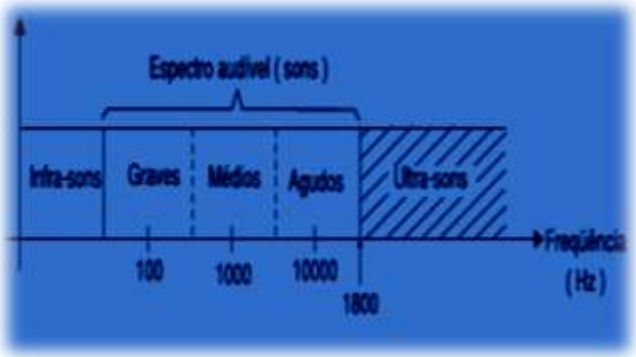


Observação:

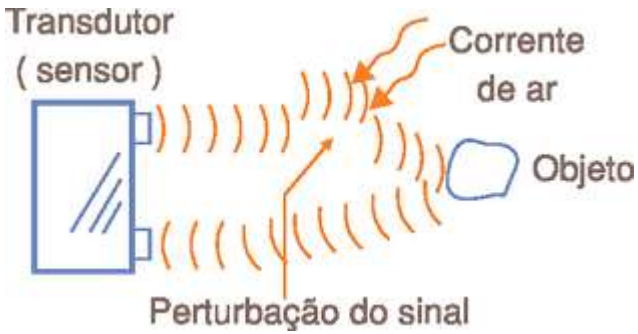
- O pino DADOS refere-se ao sinal de saída que será 'Alto' indicando movimento ou 'Baixo' indicando nenhuma movimentação.
- Quando a saída é acionada pelo movimento detectado esta ficará em alto por um curto período de tempo, mesmo se não haja mais movimento.
- O tempo em alto pode ser setado variando o potenciômetro 'Time', sendo que o outro altera a sensibilidade.

Sensor Ultrassônico HC-SR04





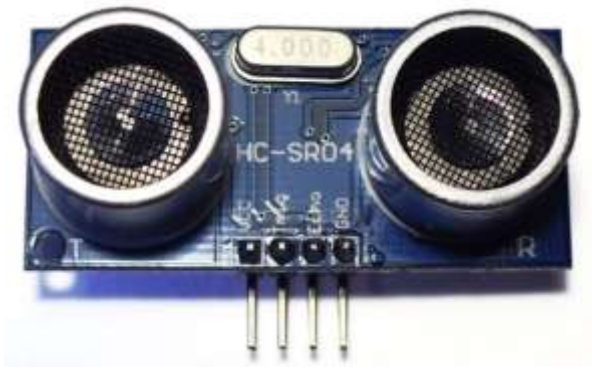
O que é o sensor HC-SR04 ?



- É um sensor ultrassônico que utiliza pulsos sonoros para determinar a distância de objetos (mesmo principio dos morcegos).
- Oferece uma excelente precisão e leituras estáveis.
- Não é afetado pela luz (do sol ou ambiente) ao contrário de sensor infravermelho.
- Porém materiais "acústicos" podem dificultar as leituras do sensor.
- O tempo que a onda leva para ir e voltar é medido pelo sensor, e como a velocidade do som é conhecida, é possível calcular a distância do objeto através da equação:

$$Distância(m) = \frac{Velocidade(m/s) \times Tempo(s)}{2}$$

Características Técnicas



- Alimentação: 5v DC
- Corrente de trabalho: <2mA
- Ângulo Efetivo: <15° (ângulo que o sensor "enxerga" objetos)
- Distância de Trabalho : 2cm até 450cm
- Resolução : 0.3 cm

Ele possui 4 pinos, sendo 2 de alimentação e dois de comando.

- **Vcc** – +5v
- **Trig** – este pino tem que receber um pulso para iniciar uma nova leitura (**emissor** do pulso ultrassônico)
- **Echo** – este pino irá para nível lógico alto(+5v) quando o pulso de ultrassom retornar (**receptor** do pulso ultrassônico)
- **Gnd** – negativo(0v)

Função *pulseIn*()

Lê um pulso (alto ou baixo) em um pino e retorna a largura do pulso (em microssegundos) ou "0" se nenhum pulso for concluída antes do tempo limite. Os pulsos podem ter duração de 10 microsegundos à 3 minutos.

Sintaxe

pulseIn(pino, valor)
pulseIn(pino, valor, tempo)

pino: número do pino para leitura do pulso (int).

valor: nível do pulso para leitura (HIGH ou LOW).

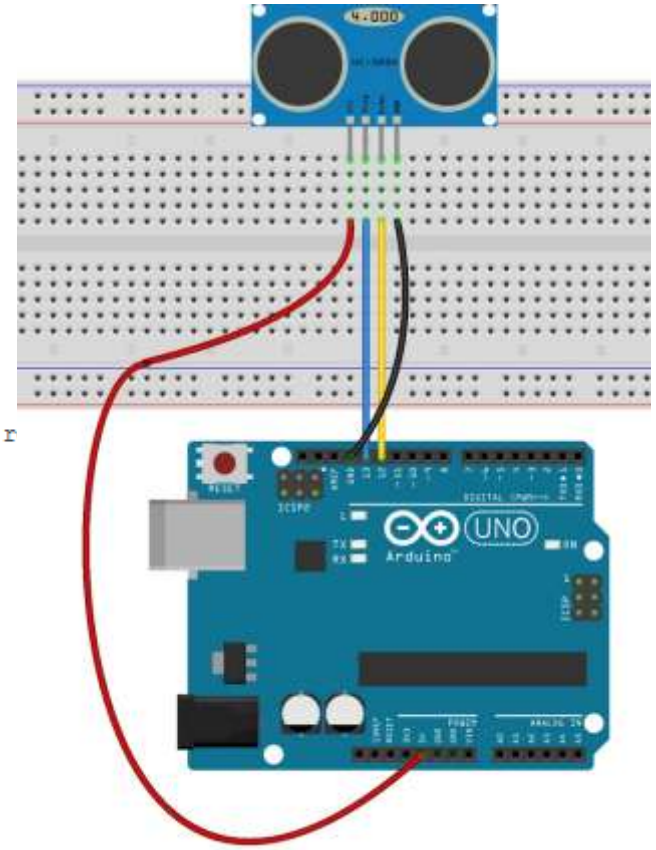
tempo (opcional): tempo de espera em microsegundos para que o pulso comece. O padrão é um segundo (*unsigned long*).

Exemplo

```
int pin = 7;

unsigned long duration;

void setup()
{
  pinMode(pin, INPUT);
}


void loop()
{
  duration = pulseIn(pin, HIGH);
}
```

Pequeno Projeto
sem biblioteca

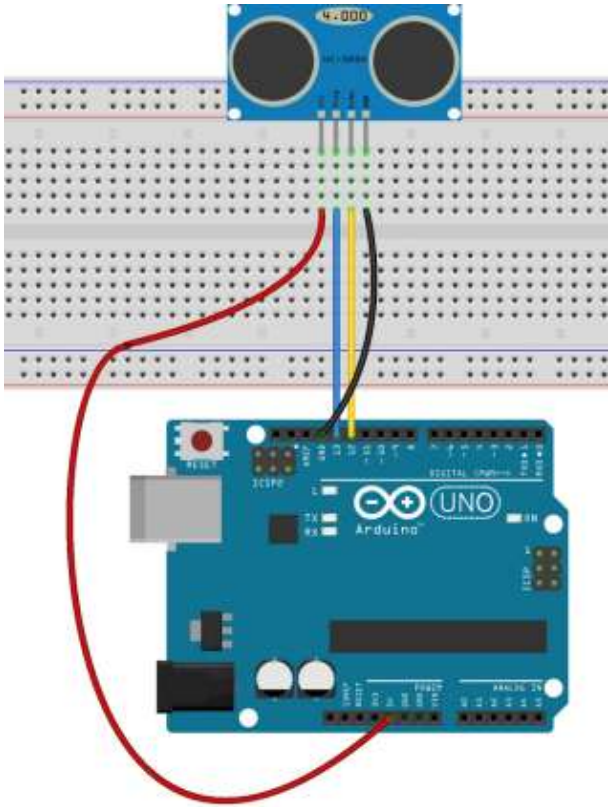
```
#define trigPin 13          //definindo que onde aparecer "trigPin" o Arduino entenda como "13"
#define echoPin 12          //definindo que onde aparecer "echoPin" o Arduino entenda como "12"
long tempo, distancia;      //declarando as variáveis que vamos utilizar
void setup() {
Serial.begin (9600);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
}
void loop()
digitalWrite(trigPin, LOW);  //garante que o pino de trigger esteja baixo (0v)
delayMicroseconds(2);        //aguarda 2us para ligar o pino trigger (5v)
//Gera um pulso de 10 microsegundos no pino de trigger
digitalWrite(trigPin, HIGH);  //seta o pino do trigger (5v)
delayMicroseconds(10);        //espera 10us em 5V
digitalWrite(trigPin, LOW);   //retorna ao 0v
//função "pulseIn" que retorna quantos us passaram desde a emissão do pulso até o r
tempo = pulseIn(echoPin, HIGH);
//A velocidade do som é de 340 m/s ou 29 microssegundos por centimetro.
//Logo distancia = (tempo/2)/29;
//testa se a distancia medida está aceitável
if (distancia >= 400 || distancia <= 0){
Serial.println("Fora de escala");
}
else {
Serial.print(distancia);
Serial.println("cm");
}
delay(500);  //espera meio segundo para fazer uma nova leitura
}
```



Após abrir a IDE do Arduino e fazer o upload para sua arduino,
clique no ícone de "serial monitor" para aparecer as medidas realizadas pela placa.

Pequeno Projeto
com biblioteca

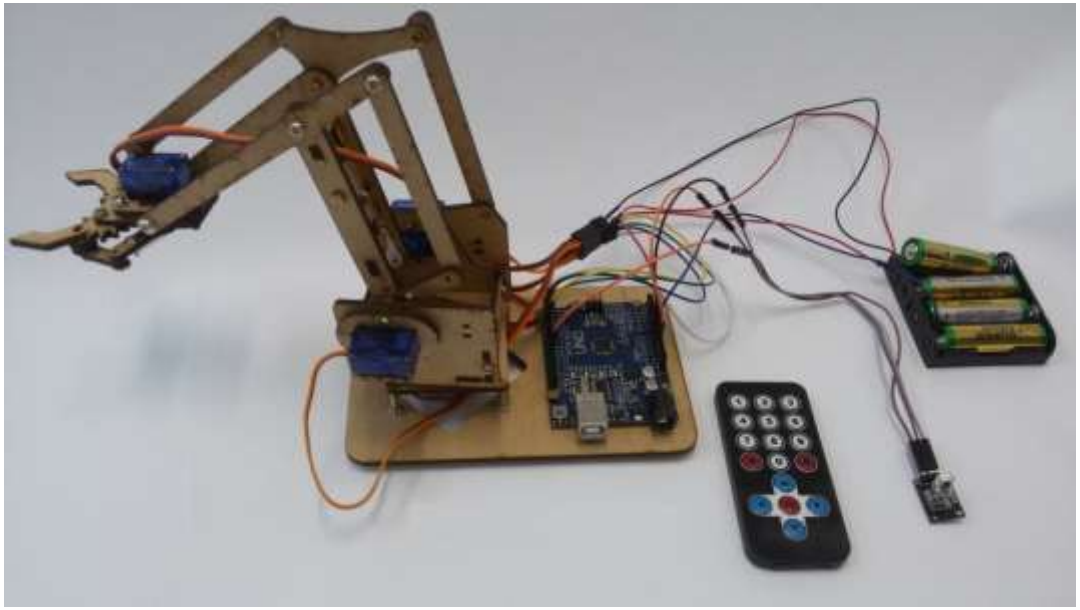
```
//Carrega a biblioteca do sensor ultrassonico
#include <Ultrasonic.h>

//Define os pinos para o trigger e echo
#define pino_trigger 13
#define pino_echo 12

//Inicializa o sensor nos pinos definidos acima
Ultrasonic ultrasonic(pino_trigger, pino_echo);

void setup()
{
  Serial.begin(9600);
  Serial.println("Lendo dados do sensor...");
}

void loop()
{
  //Le as informacoes do sensor, em cm e pol
  float cmMsec, inMsec;
  long microsec = ultrasonic.timing();
  cmMsec = ultrasonic.convert(microsec, Ultrasonic::CM);
  inMsec = ultrasonic.convert(microsec, Ultrasonic::IN);
  //Exibe informacoes no serial monitor
  Serial.print("Distancia em cm: ");
  Serial.print(cmMsec);
  Serial.print(" - Distancia em polegadas: ");
  Serial.println(inMsec);
  delay(1000);
}
```



Após abrir a IDE do Arduino e fazer o upload para sua arduino, clique no ícone de "serial monitor" para aparecer as medidas realizadas pela placa.

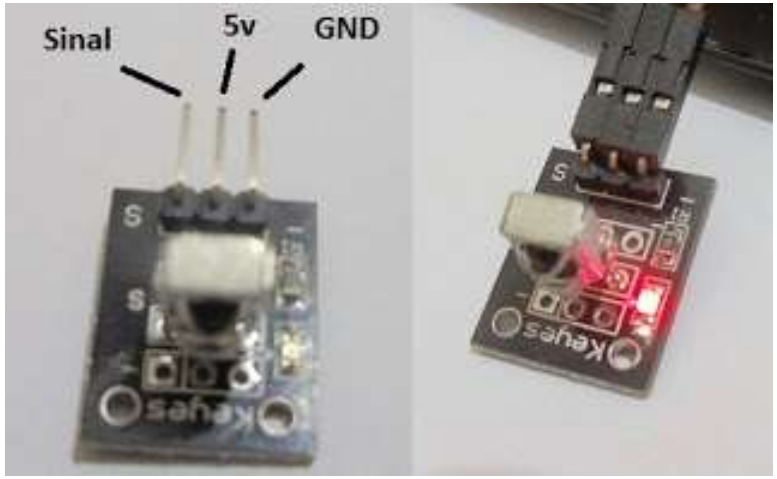
Controle Remoto Infravermelho(IR)

- Quando apertamos o botão do controle, fazemos essa luz piscar, emitindo pulsos longos e curtos que compõem um código binário, convertido em comandos pelo aparelho ao qual se destina.
- A cada botão do controle remoto corresponde um código específico.



O kit do Controle IR

- 1 módulo receptor IR;
- 1 led infravermelho;
- 1 cabo de conexão e
- 1 controle remoto (com bateria CR2025)

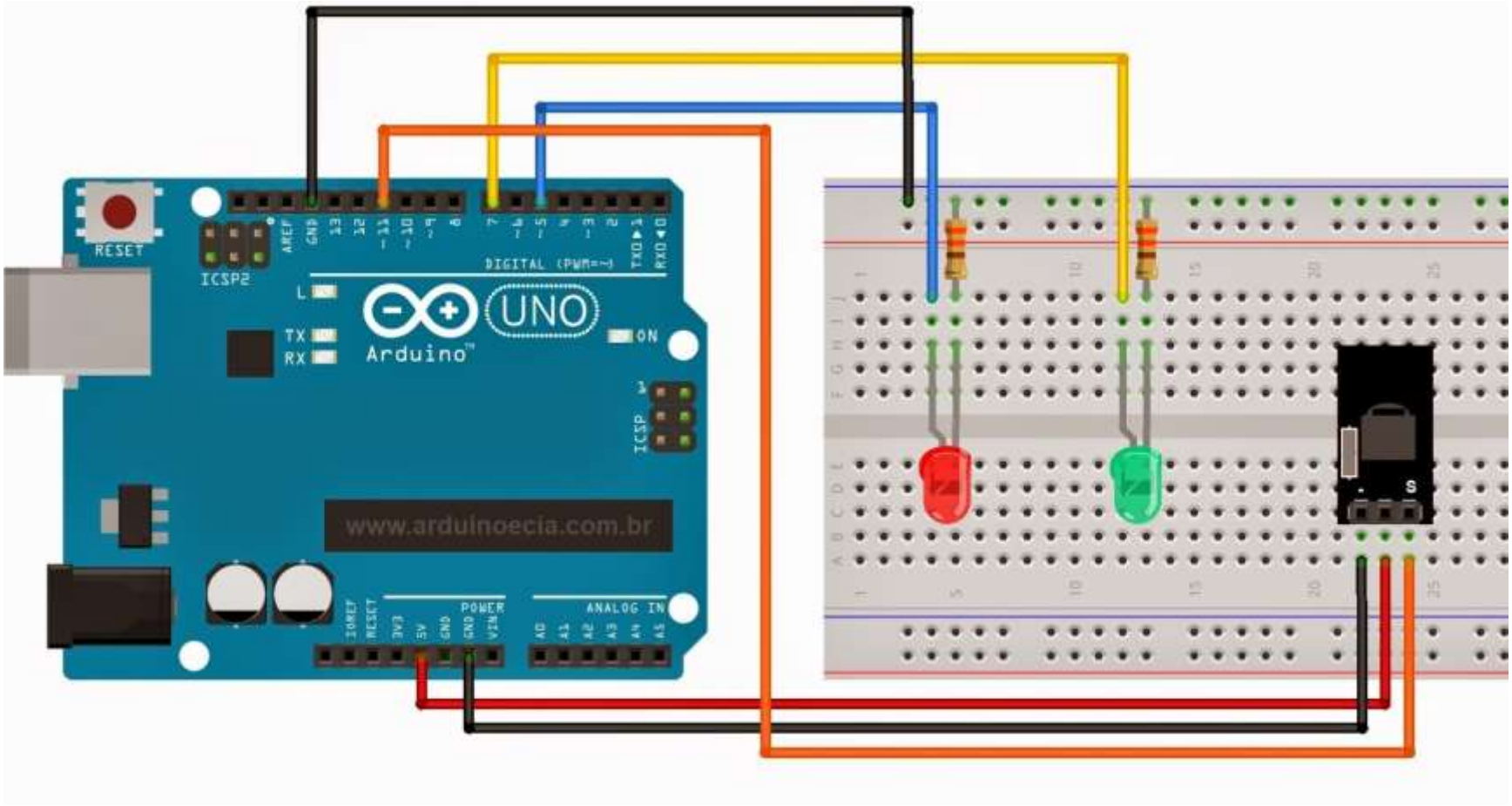


Programa Proposto

Recepção do sinal do controle remoto, acendendo e apagando os leds vermelho e verde, conforme a sequência abaixo :

- **Tecla 1** : Acende led **vermelho (FF30CF)**
- **Tecla 2** : Apaga led **vermelho (FF18E7)**
- **Tecla 4** : Acende led **verde (FF10EF)**
- **Tecla 5** : Apaga led **verde (FF38C7)**
- **Tecla 9** : Apaga os **2 leds (FF52AD)**

Montagem do circuito



Controle Remoto Infravermelho


```
#include <IRremote.h>
int RECV_PIN = 11;
float armazenavalor;
int pinoledvermelho = 5;
int pinoledverde = 7;
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup()
{
  pinMode(pinoledvermelho, OUTPUT);
  pinMode(pinoledverde, OUTPUT);
  Serial.begin(9600);
  irrecv.enableIRIn(); // Inicializa o receptor IR
}
void loop()
{
  if (irrecv.decode(&results))
  {
    Serial.print("Valor lido : ");
    Serial.println(results.value, HEX);
    armazenavalor = (results.value);
    if (armazenavalor == 0xFF30CF) //Verifica se a tecla 1 foi acionada
    {
      digitalWrite(pinoledvermelho, HIGH);  //Acende o led vermelho
    }
    if (armazenavalor == 0xFF18E7) //Verifica se a tecla 2 foi acionada
    {
      digitalWrite(pinoledvermelho, LOW); //Apaga o led vermelho
    }
    if (armazenavalor == 0xFF10EF) //Verifica se a tecla 4 foi acionada
    {
      digitalWrite(pinoledverde, HIGH); //Acende o led verde
    }
    if (armazenavalor == 0xFF38C7) //Verifica se a tecla 5 foi acionada
    {
      digitalWrite(pinoledverde, LOW); //Apaga o led verde
    }
    if (armazenavalor == 0xFF52AD) //Verifica se a tecla 9 foi acionada
    {
      digitalWrite(pinoledvermelho, LOW); //Apaga todos os leds
      digitalWrite(pinoledverde, LOW);
    }
  irrecv.resume(); //Le o próximo valor
  }
}
```

Módulos Transmissores RF (Rádio Frequência)

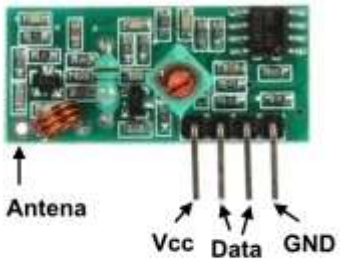
- Com um Kit Módulo RF Transmissor + Receptor de 433 MHz, pode-se conseguir enviar e receber dados sem a necessidade de uso de fios.

Modelo: MX-FS-03V
- Alcance: 20-200 metros (conforme voltagem)
- Tensão de operação: 3,5-12v
- Modo de operação: AM (Modulação em Amplitude)
- Taxa de transferência: 4KB/s
- Potência de transmissão: 10mW
- Frequência de transmissão: 433MHz
- Dimensões: 19 x 19mm

Modelo: MX-05V
- Tensão de operação: 5v DC
- Corrente de operação: 4mA
- Frequência de recepção: 433MHz
- Sensibilidade: -105dB
- Dimensões: 30 x 14 x 7mm

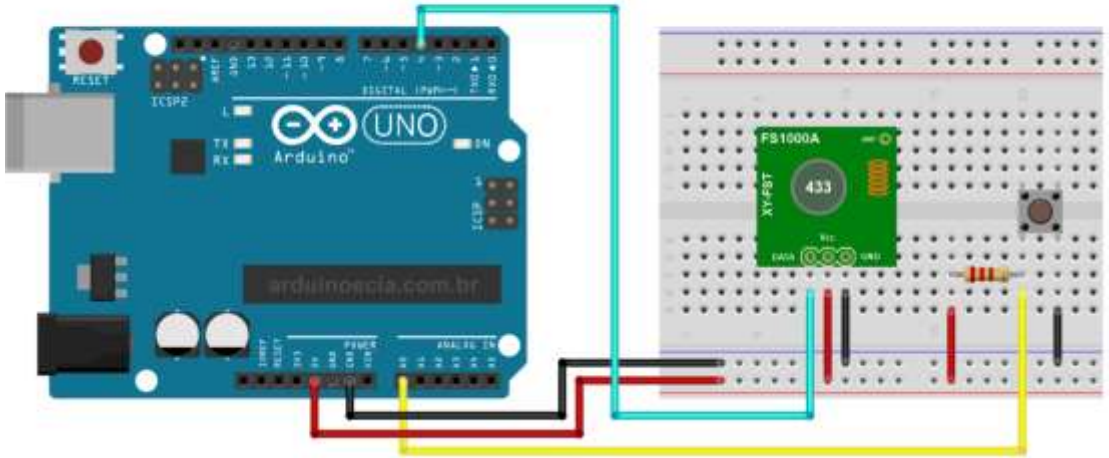
TRANSMISSOR

RECEPTOR

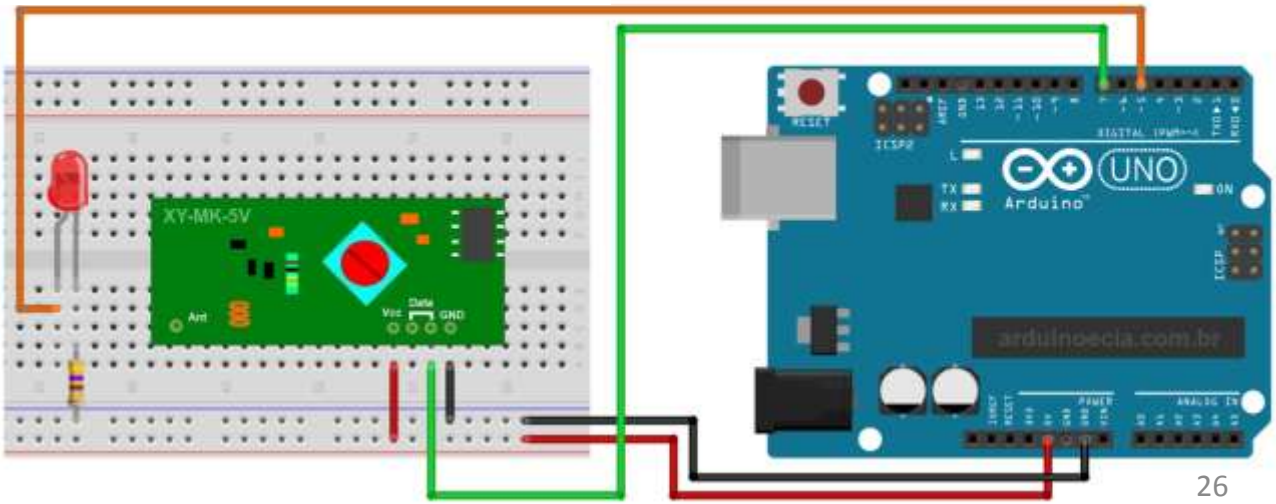


Módulos transmissores RF (Rádio Frequência)

RF 433 MHz - Circuito Transmissor



RF 433 MHz - Circuito Receptor



Programa Arduino sem biblioteca

- *Transmissor RF*

```
/* RF Blink - Transmit sketch
   Website: http://arduinobasics.blogspot.com
   Transmitter: FS1000A/XY-FST
------------------------------------------------------------ */

#define rfTransmitPin 4  //RF Transmitter pin = digital pin 4
#define ledPin 13        //Onboard LED = digital pin 13

void setup(){
  pinMode(rfTransmitPin, OUTPUT);
  pinMode(ledPin, OUTPUT);
}

void loop(){
  for(int i=4000; i>5; i=i-(i/3)){
    digitalWrite(rfTransmitPin, HIGH);     //Transmit a HIGH signal
    digitalWrite(ledPin, HIGH);            //Turn the LED on
    delay(2000);                           //Wait for 1 second

    digitalWrite(rfTransmitPin,LOW);       //Transmit a LOW signal
    digitalWrite(ledPin, LOW);             //Turn the LED off
    delay(i);                             //Variable delay
  }
}
```

- *Receptor RF*

```
/* RF Blink - Receiver sketch
   Website: http://arduinobasics.blogspot.com
   Receiver: XY-MK-5V
------------------------------------------------------------ */

#define rfReceivePin A0  //RF Receiver pin = Analog pin 0
#define ledPin 13        //Onboard LED = digital pin 13

unsigned int data = 0;   // variable used to store received data
const unsigned int upperThreshold = 70;  //upper threshold value
const unsigned int lowerThreshold = 50;  //lower threshold value

void setup(){
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  data=analogRead(rfReceivePin);    //listen for data on Analog pin 0

   if(data>upperThreshold){
    digitalWrite(ledPin, LOW);   //If a LOW signal is received, turn LED OFF
    Serial.println(data);
  }

  if(data<lowerThreshold){
    digitalWrite(ledPin, HIGH);   //If a HIGH signal is received, turn LED ON
    Serial.println(data);
  }
}
```

Programa Arduino com biblioteca *RCSwitch*

- *Transmissor RF*

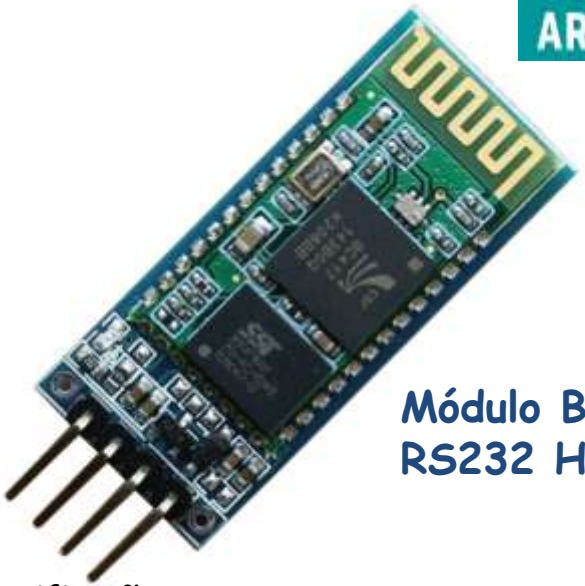
```
#include <RCSwitch.h>            //importa a biblioteca RCSwitch
RCSwitch mySwitch = RCSwitch();  //Instacia a Biblioteca RF
const int button = 2;            //Constante para os pinos dos botoes
int buttonState = 0;             //variaveis de Controle de cada botao

void setup() {
pinMode(button, INPUT);          //seta os pinos dos botoes como saida
digitalWrite(button, 1);         //seta os pinos como HIGH (Alto)
Serial.begin(9600);
mySwitch.enableTransmit(10);     //Seta como habititado para enviar dados RF o pino 10
delay(50);
}

void loop()
buttonState = digitalRead(button); //Le o Status dos botoes
if (buttonState == 0) {
  Serial.println("2");
    mySwitch.send(2, 24);
}
 if (buttonState == 1) {
  Serial.println("1");
    mySwitch.send(1, 24);
}
}
```

- *Receptor RF*

```
#include <RCSwitch.h>
RCSwitch mySwitch = RCSwitch();      //Instacia a Biblioteca
int led=13;                          //Conexao fisica do pino 13 com o LED
int value=0;                         //variavel responsavel em receber os dados do RF

void setup() {
pinMode(led,OUTPUT);                  //Seta o pino 13 como saida para o LED
Serial.begin(9600);
delay(50);                            //delay para estabilizacao do Sinal delay(50);
mySwitch.enableReceive(0);            //Seta como Receptor o pino "0"
}

void loop() {
if (mySwitch.available()) {
value = mySwitch.getReceivedValue(); //recebe na variavel value o Status
if (value == 0) {
    Serial.println("Codigo desconhecido");
}
if(value ==2){
    Serial.println(mySwitch.getReceivedValue());
    digitalWrite(led, HIGH);
}
  if(value ==1){
    Serial.println(mySwitch.getReceivedValue());
    digitalWrite(led, LOW);
  }
 mySwitch.resetAvailable();
  }
}
```

Acionamento via Bluetooth

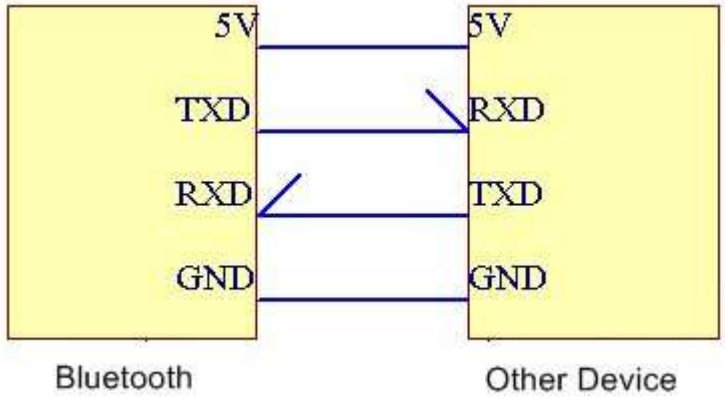
- O módulo Bluetooth HC-06 é usado para comunicação wireless entre o Arduino e algum outro dispositivo com bluetooth, como por exemplo um telefone celular, um computador ou tablet.
- As informações recebidas pelo módulo são repassadas ao Arduino (ou outro microcontrolador) via serial.
- O alcance do módulo segue o padrão da comunicação bluetooth, que é de aproximadamente 10 metros.
- O módulo HC-06 funciona apenas em modo **slave (escravo)**, ou seja, ele permite que outros dispositivos se conectem à ele, mas não permite que ele próprio se conecte à outros dispositivos bluetooth.
- O módulo HC-05 suporta o modo mestre e escravo e tem uma fácil configuração.



Módulo Bluetooth RS232 HC-05

Especificações:
- Protocolo Bluetooth: v2.0+EDR
- Firmware: Linvor 1.8
- Frequência: 2,4GHz Banda ISM
- Modulação: GFSK
- Segurança: Autentificação e Encriptação
- Banda de Onda: 2,4Hhz-2,8Ghz, Banda ISM
- Tensão: 3,3v ou 5V
- Corrente: Pareado 35mA; Conectado 8mA
- Alcance: 10m
- Baud Rate: 4800;9600;19200;38400;57600;115200;230400;460800; 921600;1382400
- Dimensões: 26,9 x 13 x 2,2mm
- Peso: 9,6g

Características e ligação ao Arduino



- O módulo possui 4 pinos :
  - **Vcc** (alimentação de 3,6 à 6v e 10 mA), **GND**, **RXD** e **TXD**, sendo os dois últimos utilizados para comunicação com o Arduino via serial.
  - O nível lógico dos pinos RXD e TXD é de 3.3v, o que significa que, para o Arduino Uno, por exemplo, vamos precisar de um divisor de tensão no pino RX para evitar que o módulo seja danificado. Isso é necessário pois o Arduino Uno trabalha com nível de sinal de 5v.

Programa no Arduino antes de montar o circuito

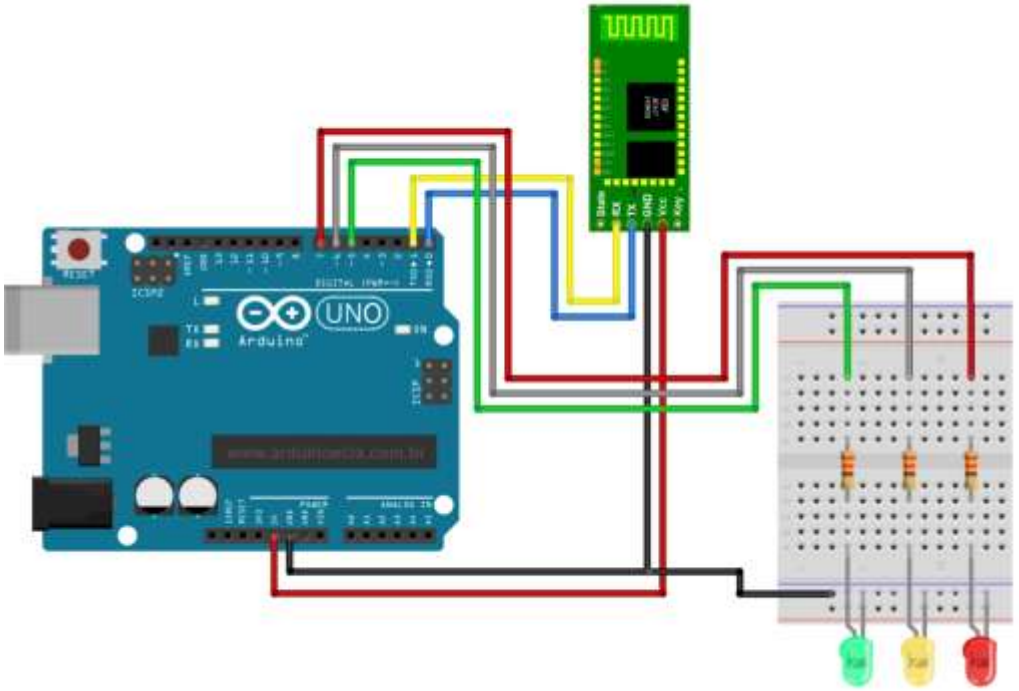

```
//Autor: Almir M Ventura
//site: www.omecatronico.com.br
//Controla LEDs pela comunicação Bluetooth.
//Aplicativo Android "Domótica JL" do autor "Júlio Lima"
//https://play.google.com/store/apps/details?id=appinventor.ai_paulistacesar.DOMOTICA_JL

char aux=0;
char led1=13;
char led2=12;
char led3=11;
char led4=10;
char led5=9;
char led6=8;
char led7=7;
char led8=6;

void setup(){
Serial.begin(9600);
Serial.println("Sistema Inicializado!");
pinMode(led1,OUTPUT);
pinMode(led2,OUTPUT);
pinMode(led3,OUTPUT);
pinMode(led4,OUTPUT);
pinMode(led5,OUTPUT);
pinMode(led6,OUTPUT);
pinMode(led7,OUTPUT);
pinMode(led8,OUTPUT);

}
```

```
//Desliga todos os lEDs...
  case 'I':
    Serial.println("TODOS OS RELES FORAM DESLIGADOS!");
    digitalWrite(led1,LOW);
    digitalWrite(led2,LOW);
    digitalWrite(led3,LOW);
    digitalWrite(led4,LOW);
    digitalWrite(led5,LOW);
    digitalWrite(led6,LOW);
    digitalWrite(led7,LOW);
    digitalWrite(led8,LOW);
    break;
//liga todos os lEDs...
  case 'E':
    Serial.println("TODOS OS RELES FORAM LIGADOS!");
    digitalWrite(led1,HIGH);
    digitalWrite(led2,HIGH);
    digitalWrite(led3,HIGH);
    digitalWrite(led4,HIGH);
    digitalWrite(led5,HIGH);
    digitalWrite(led6,HIGH);
    digitalWrite(led7,HIGH);
    digitalWrite(led8,HIGH);
    break;
  }
}
```

```
void loop(){
aux=Serial.read();

  switch(aux){
//led 1
  case 'L':
    Serial.println("RELE 1 LIGADO");
    digitalWrite(led1,HIGH);
    break;
  case 'D':
    Serial.println("RELE 1 DESLIGADO");
    digitalWrite(led1,LOW);
    break;
//lED 2
  case 'O':
    Serial.println("RELE 2 LIGADO");
    digitalWrite(led2,HIGH);
    break;
  case 'F':
    Serial.println("RELE 2 DESLIGADO");
    digitalWrite(led2,LOW);
    break;
//lED 3
  case '5':
    Serial.println("RELE 3 LIGADO");
    digitalWrite(led3,HIGH);
    break;
  case '6':
    Serial.println("RELE 3 DESLIGADO");
    digitalWrite(led3,LOW);
    break;
```

Montagem do Circuito

- Ao montar o circuito deve-se observar a conexão do módulo *bluetooth* com a placa Arduino: a conexão TX do módulo deverá ser ligada ao RX do Arduino (porta digital 0), enquanto que a conexão do RX do módulo deverá ser ligado ao TX do Arduino (porta digital 1).
- Como a conexão do módulo *bluetooth* também utiliza comunicação serial, isso pode interferir na comunicação com o computador, assim ao carregar o programa da CPU no arduino, deve-se desligar os cabos do módulo *bluetooth* com o arduino.



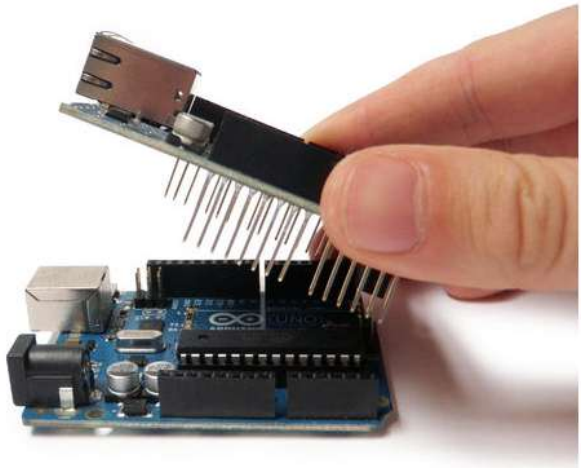
Domótica JL

- Foi criado com o propósito de oferecer ao "USUÁRIO" uma plataforma simples e de multi aplicações.
- O aplicativo é 100% compatível com os mais variáveis tipos de sistemas MICROCONTROLADOS, permitindo ao usuário ter o controle total dos comandos de envio através da comunicação serial por Bluetooth.
- Podendo ser utilizado nos mais diversos projetos, garantindo ao usuário controle total de suas aplicações, tendo em vista que o DOMÓTICA JL é capaz de enviar e receber dados através de sua interface.

Módulo Arduino Ethernet W5100

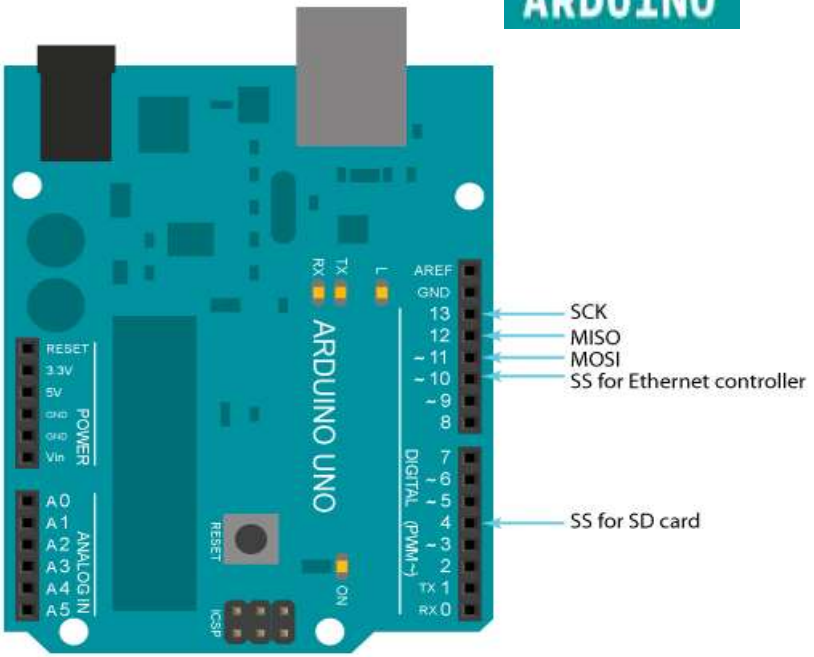
- Permite conectar o Arduino (Uno ou Mega) à rede local e também à internet, possibilitando acesso remoto, transferência de dados, verificação remota de status de sensores, e muito mais.
- Possui um leitor de cartões microSD.

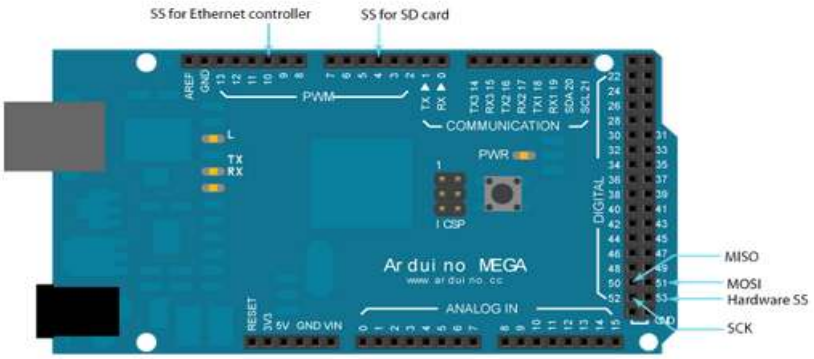
LED	INDICAÇÃO
TX	Transmissão
RX	Recepção
COLL	Colisão
FULLD	Modo de conexão Full Duplex
100M	Conexão a 100 Mbits
LINK	Conexão estabelecida
PWR	Módulo Ligado



Compatibilidade do Shield

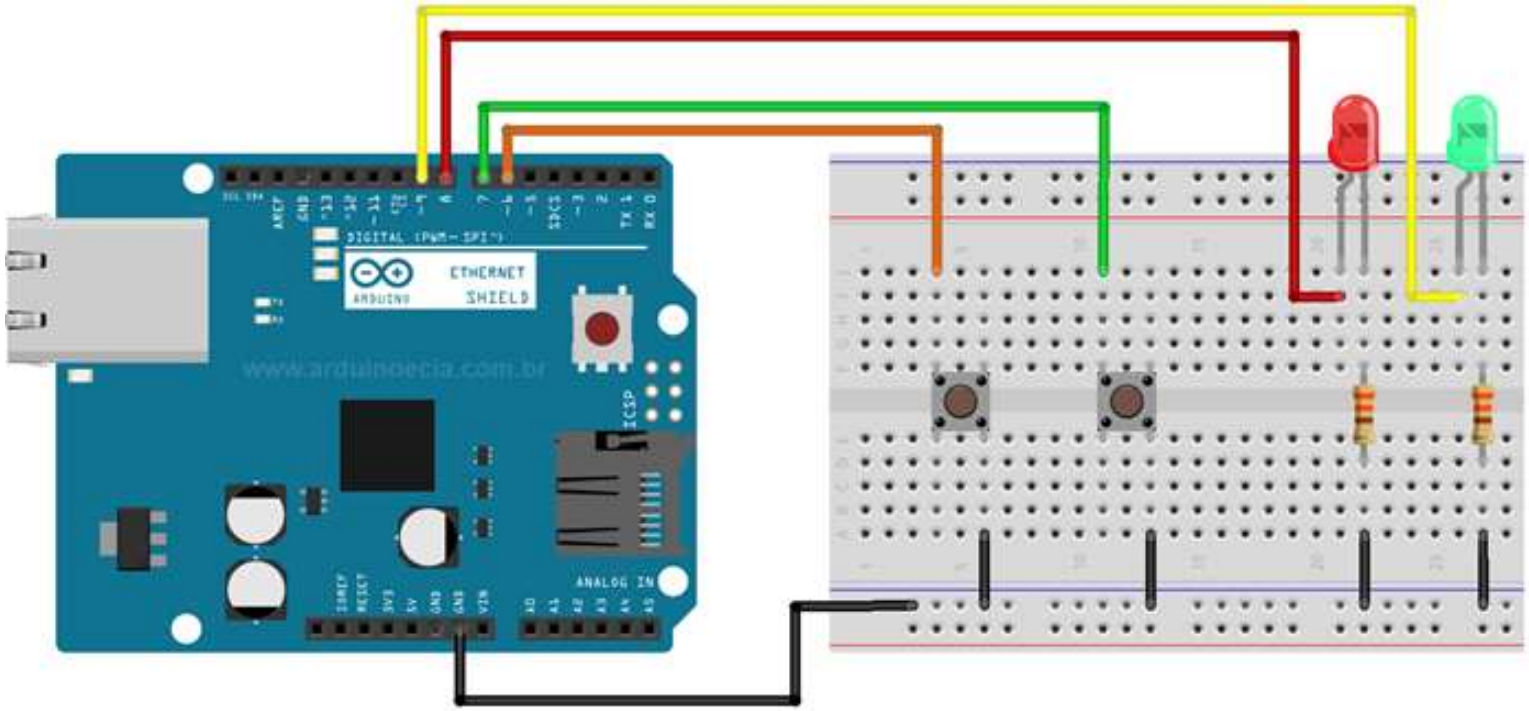
- É **compatível** tanto com o Arduino Uno e Mega.
- Possui um slot para cartão micro-SD que pode ser usado para armazenar arquivos que vão servir na rede.





Montagem do módulo W5100

- Montagem do circuito com o *ethernet shield* devidamente encaixado no Arduino.



Configuração IP

Percebe-se que os endereços IP são separados por vírgula, ao invés de ponto, como é habitual. Para verificar o funcionamento da placa, deve-se abrir o prompt de comando DOS e digitar:
ping 192.168.0.100

Endereço IP	192.168.0.100
Gateway	192.168.0.1
Máscara	255.255.255.0

```
// Programa : Ethernet Shield Wiznet W5100 - Define endereço IP
#include <SPI.h>
#include <Ethernet.h>

/* A linha abaixo permite que se defina o endereço
 fisico (MAC ADDRESS) da placa de rede */
byte mac[] = { 0xAB, 0xCD, 0x12, 0x34, 0xFF, 0xCA };

/* Os valores abaixo definem o endereço IP, gateway e máscara.
 Configure de acordo com a sua rede */
IPAddress ip(192,168,0,100);          //Define o endereço IP
IPAddress gateway(192,168,0,1);       //Define o gateway
IPAddress subnet(255, 255, 255, 0);   //Define a máscara de rede

void setup()
{
  Ethernet.begin(mac, ip, gateway, subnet); //Inicializa o Ethernet Shield
}
```

```
C:\Users\JENER>ping 192.168.0.100

Disparando 192.168.0.100 com 32 bytes de dados:
Resposta de 192.168.0.100: bytes=32 tempo=1ms TTL=128
Resposta de 192.168.0.100: bytes=32 tempo<1ms TTL=128
Resposta de 192.168.0.100: bytes=32 tempo<1ms TTL=128
Resposta de 192.168.0.100: bytes=32 tempo<1ms TTL=128

Estatísticas do Ping para 192.168.0.100:
    Pacotes: Enviados = 4, Recebidos = 4, Perdidos = 0 (0% de
             perda),
Aproximar um número redondo de vezes em milissegundos:
    Mínimo = 0ms, Máximo = 1ms, Média = 0ms
```

Envio de dados do arduino para o browser

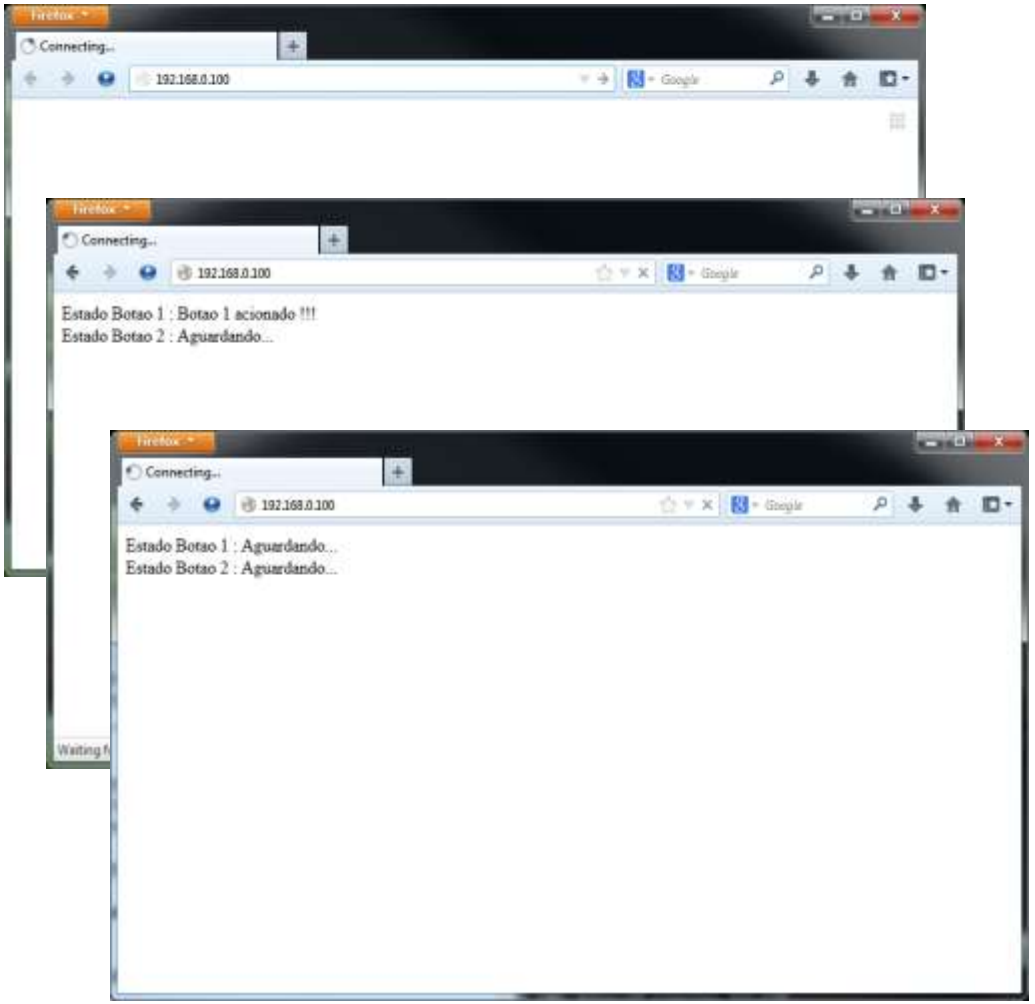

```
// Programa : Webserver com aviso de acionamento de botoes
#include <SPI.h>
#include <Ethernet.h>

/* A linha abaixo permite que voce defina o endereço fisico (MAC ADDRESS)
 da placa de rede */
byte mac[] = { 0xAB, 0xCD, 0x12, 0x34, 0xFF, 0xCA };

/* Os valores abaixo definem o endereço IP, gateway e máscara.
 Configure de acordo com a sua rede */
IPAddress ip(192,168,0,100);          //Define o endereço IP
IPAddress gateway(192,168,0,1);     //Define o gateway
IPAddress subnet(255, 255, 255, 0); //Define a máscara de rede

/* Inicializa a biblioteca da placa ethernet com as
 configurações de IP fornecidas */
EthernetServer server(80);

int botao1 = 6;   //Botao que aciona o led vermelho
int botao2 = 7;   //Botao que aciona o led verde
int pinoled=8;    //Pino ligado ao led vermelho
int pinoled2=9;   //Pino ligado ao led verde
int leitura = 0;  //Armazena o valor de leitura do botao1
int leitura2 = 0; //Armazena o valor de leitura do botao2
char mensagem[20];   //Mensagem a ser apresentada para o botao1
char mensagem2[20];  //Mensagem a ser apresentada para o botao2

void setup()
{
  pinMode(pinoled, OUTPUT); //Led
  pinMode(pinoled2, OUTPUT); //Led
  pinMode(botao1, INPUT);
  digitalWrite(botao1, HIGH);
  pinMode(botao2, INPUT);
  digitalWrite(botao2, HIGH);
  //Inicializa a conexao ethernet e o servidor web na porta 80
  Ethernet.begin(mac, ip, gateway, subnet);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}
```

```
void loop()
{
   //Verifica o status do Botao1 e imprime mensagem no browser
   leitura=digitalRead(botao1);
   if (leitura == 0)
     {
       digitalWrite(pinoled,1);
       char mensagem[] = "Botao 1 acionado !!!";
       char mensagem2[] = "Aguardando...";
       apresentadados(mensagem,mensagem2);
       delay(5000);   //Mantem o led aceso por 5 segundos
       //Imprime mensagem padrao, aguardando novo acionamento
       apresentadados("Aguardando...", "Aguardando...");
       digitalWrite(pinoled,0);
     }

   //Verifica o status do Botao2 e imprime mensagem no browser
   leitura2=digitalRead(botao2);
   if (leitura2 == 0)
     {
       digitalWrite(pinoled2,1);
       char mensagem[] = "Aguardando...";
       char mensagem2[] = "Botao 2 acionado !!!";
       apresentadados(mensagem,mensagem2);
       delay(5000);  //Mantem o led aceso por 5 segundos
       //Imprime mensagem padrao, aguardando novo acionamento
       apresentadados("Aguardando...", "Aguardando...");
       digitalWrite(pinoled2,0);
     }
}

// Rotina que recebe os valores de Mensagem e Mensagem2,
// imprimindo o resultado no browser
```

```
void apresentadados(char msg[], char msg2[])
{
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        // if you've gotten to the end of the line (received a newline
        // character) and the line is blank, the http request has ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          // the connection will be closed after completion of
          // the response
          client.println("Connection: close");

          // refresh the page automatically every 5 sec
          client.println("Refresh: 0");
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          // output the value of each analog input pin
          client.print("Estado Botao 1 : ");
          client.print(msg);
          client.println("<br />");
          client.print("Estado Botao 2 : ");
          client.print(msg2);
          client.println("<br />");
          client.println("</html>");
          break;
        }
        if (c == '\n') {
          // you're starting a new line
          currentLineIsBlank = true;
        }
        else if (c != '\r') {
          // you've gotten a character on the current line
          currentLineIsBlank = false;
        }
      }

    }
    // give the web browser time to receive the data
    delay(1);
    // close the connection:
    client.stop();
    Serial.println("client disconnected");
  }
}
```

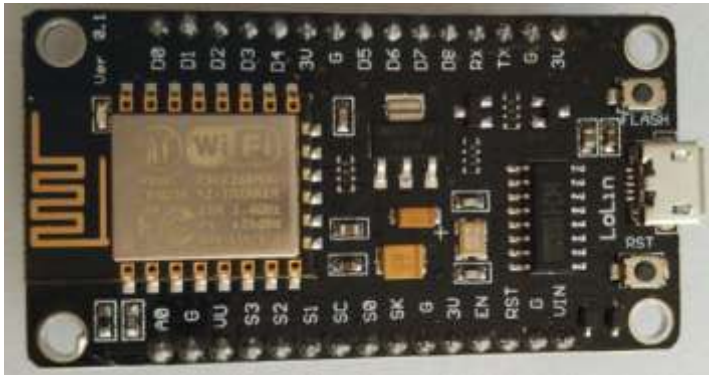
Teste do programa

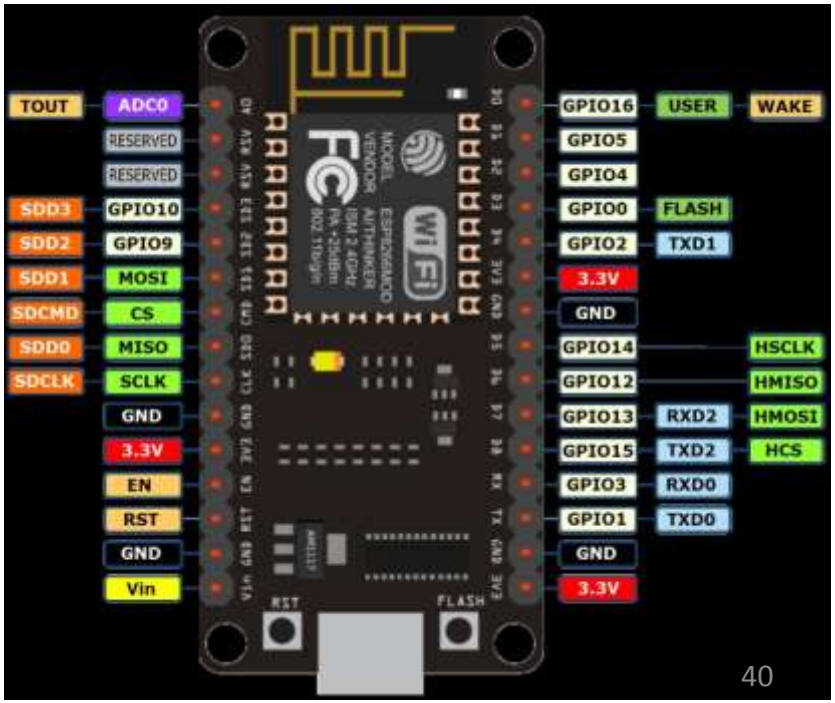
- Para testar o programa, deve-se entrar no browser (Ex.: Firefox) e digitar o endereço da placa de rede configurada no programa;
- Ao pressionar a tecla <ENTER> o browser irá acessar o servidor Web interno da placa e aguardará o pressionamento de um dos botões;
- Pressionando um dos botões do circuito o LED correspondente será aceso, indicando que o botão foi acionado. Ao mesmo tempo, será exibida na tela a mensagem abaixo, de acordo com o botão pressionado;
- O LED permanecerá aceso por 5 segundos, depois disso o browser exibirá a mensagem de "Aguardando..." para os 2 botões, sinalizando a espera de um novo sinal.



Módulo de rede Wi-Fi ESP8266

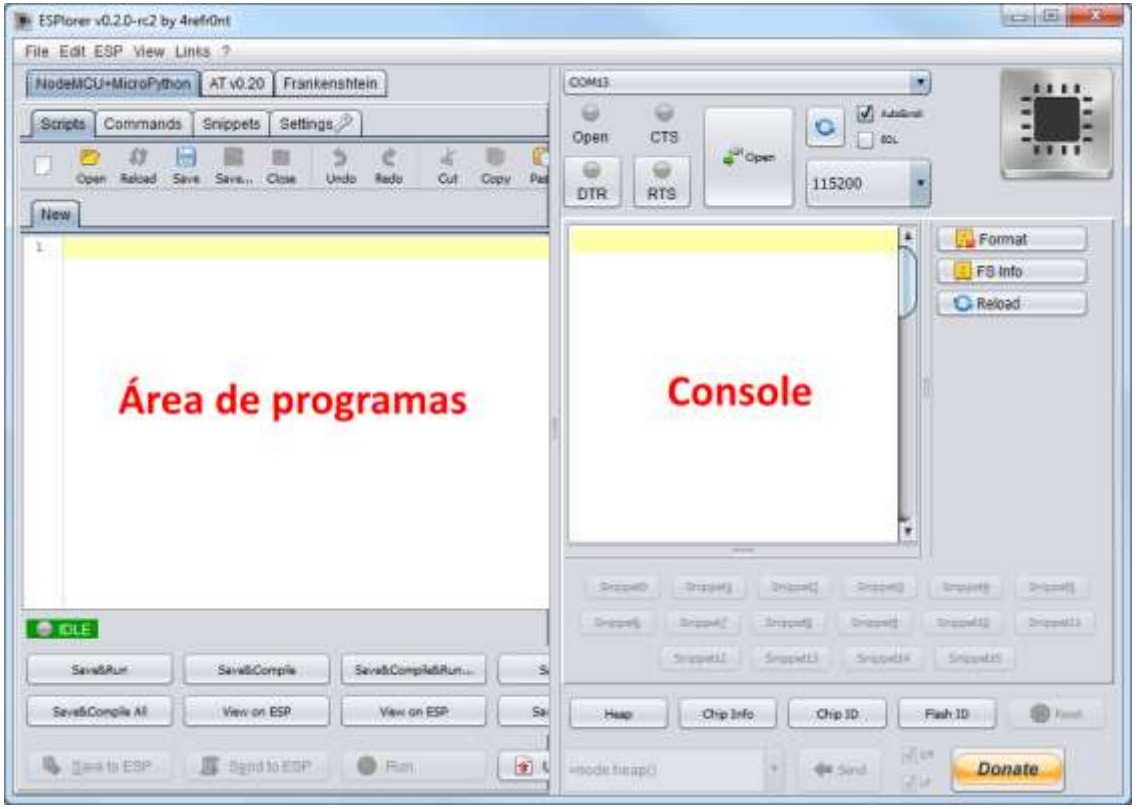
- Tem incorporado um módulo ***ESP12-E*** com antena embutida e um conversor USB-TTL (CH340), facilitando a comunicação e transferência de programas;
- Tem um formato que facilita o uso em uma protoboard, permitindo acesso às 11 portas (GPIO) do módulo, não necessitando de um microcontrolador adicional como **Arduino, PIC** ou **Raspberry** para criação dos projetos;
- O conector de alimentação é micro-usb, e serve também para comunicação com o computador;
- A programação pode tanto ser feita em Lua, como também com a própria IDE do Arduino, com algumas modificações na parte de gerenciamento da placa;
- A instalação do módulo ESP8266 NodeMCU no Windows foi feita de forma automática, tendo o módulo reconhecido na porta COM13 como um dispositivo USB-Serial CH340.

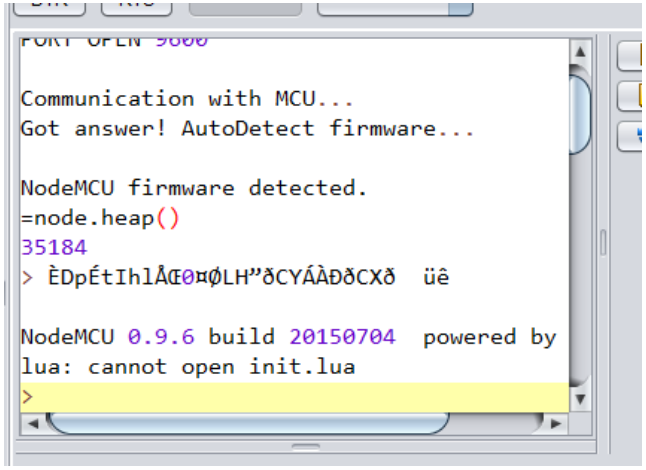




Instalação do ESPlorer

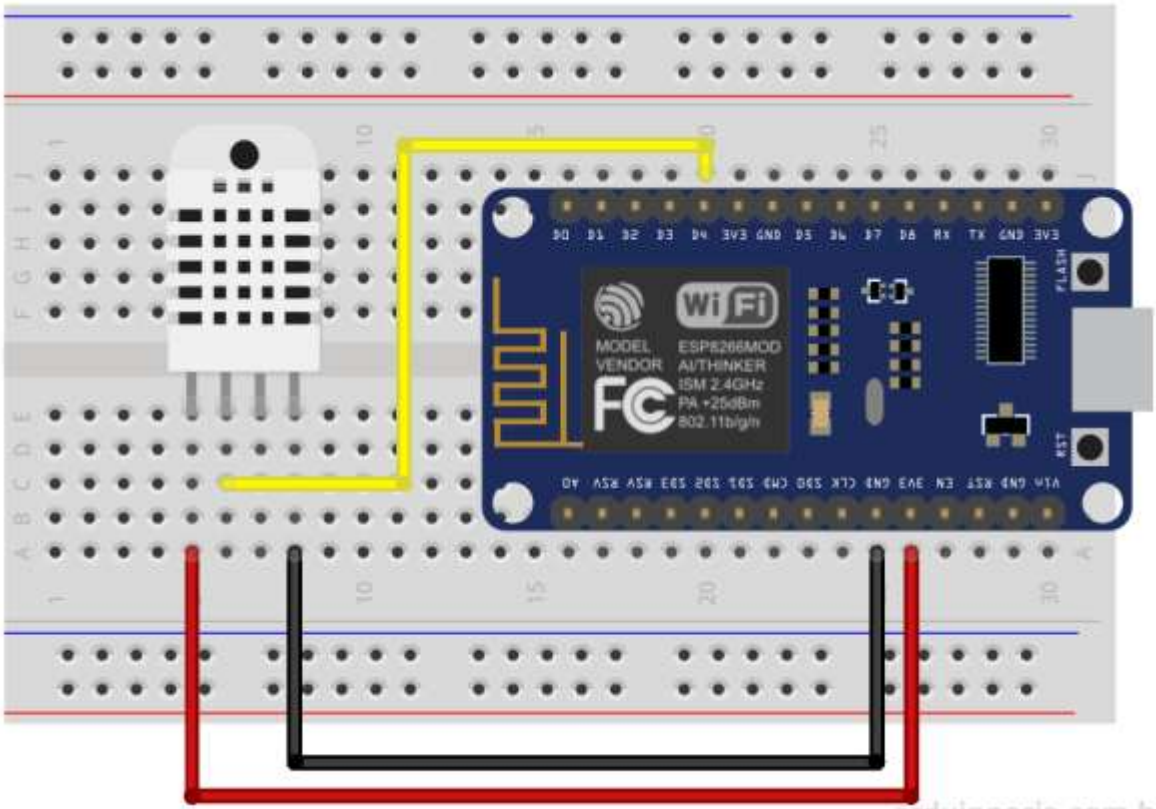
- Com o **ESPlorer** a criação e transferência de programas para o módulo ESP8266 é bastante simplificada, sendo possível: salvar programas, enviar comandos especiais ao módulo, resetar, formatar, etc. (Exige JAVA v.7 ou superior);
- Selecionar no console porta COM13, velocidade 9600 e no módulo o botão RST.





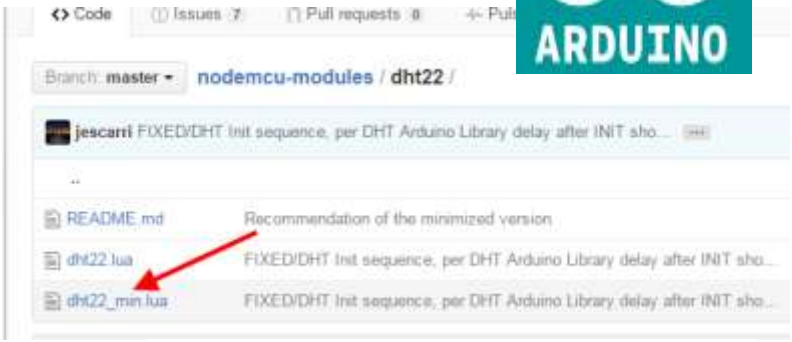
Montagem do módulo ESP8266 NodeMCU

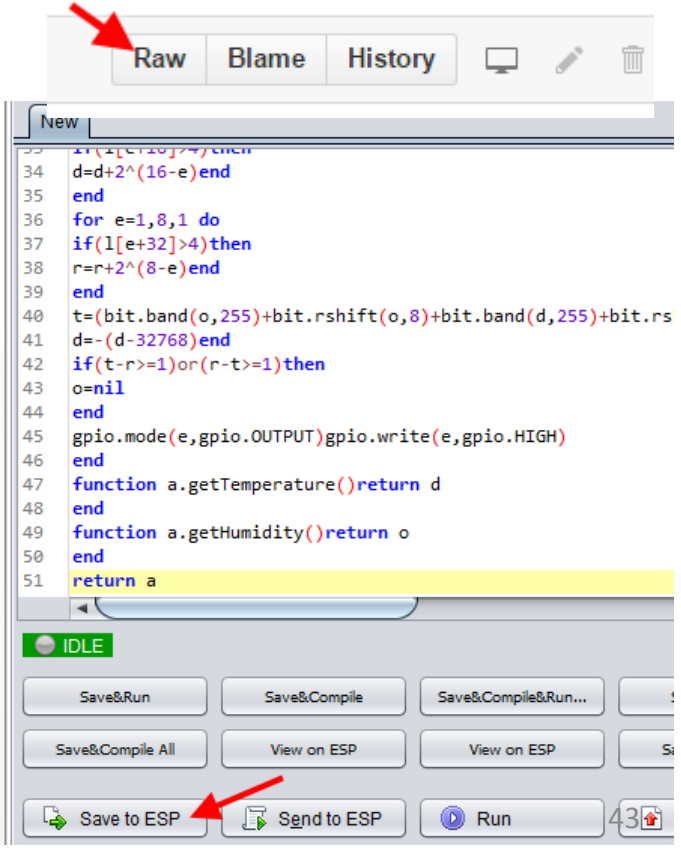
- O pino de dados do DHT22 deverá ser conectado à porta 4 do NodeMCU (GPIO02);
- A alimentação do sensor de temperatura será feita pelos pinos 3.3V e GND do módulo;
- Conectar o cabo microusb ao NodeMCU para ligar o circuito e iniciar o processo de transferência de programas.

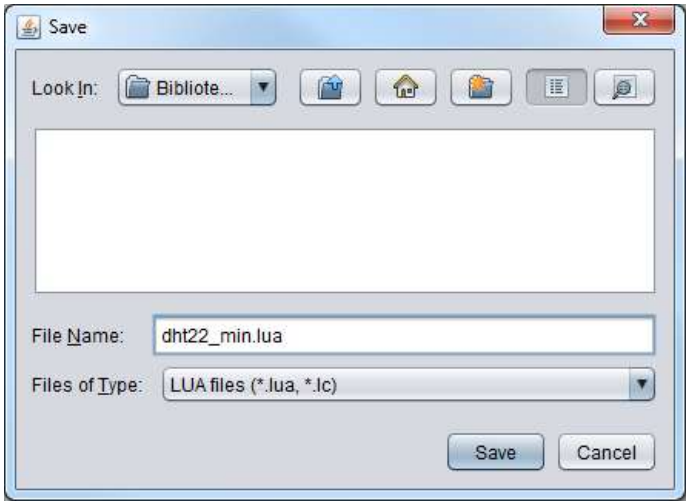


Instalação biblioteca DHT22 Lua

- Instalar a biblioteca DHT22 Lua;
- Selecionar a opção RAW;
- O código da biblioteca, "limpo", será exibido no browser;
- Copiar todo o código e colar no **ESPlorer**, de preferência em uma nova aba;
- Em seguida, clique em **Save to ESP**;
- Na janela seguinte, coloque o nome do arquivo como **dht22_min.lua** e clique em **Save**;
- O programa será então gravado localmente na sua máquina, e também transferido para o ESP8266 NodeMCU.



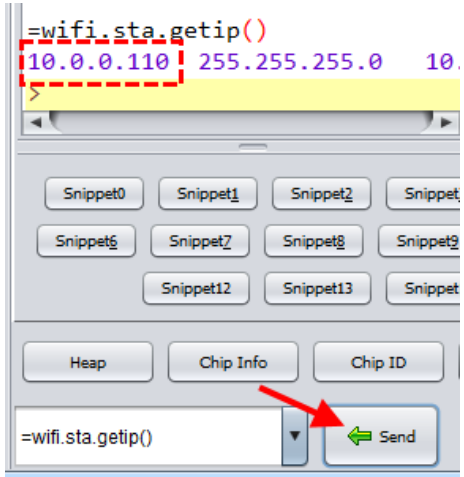




Programa Web Server com DHT22

- Copiar o programa WiFi.txt do site do Prof. para uma nova aba do ***ESPlorer***.
- Ele faz a leitura dos dados do DHT22 e cria um web server para exibição de uma página com as informações de temperatura e umidade.
- Na linha 7, trocar os campos **NOME_REDE** e **SENHA_REDE** pelas informações da rede Wi-Fi à qual o módulo vai se conectar.
- Pressionar a tecla **Send to ESP** e aguarde até que o programa seja transferido;
- Na caixa de seleção do lado direito, abaixo da console, escolhe-se a opção **=wifi.sta.getip()** e clica em **Send**. O endereço IP será exibido no console (IP = **10.0.0.110**);
- Utilize o IP no browser, digitando-o na barra de endereços. Em poucos instantes as informações de temperatura e umidade do DHT22 serão enviadas pela placa.

```
1  -- Programa: Web Server com ESP8266 NodeMCU e DHT22
2  -- Autor: Arduino e Cia
3  -- Baseado no programa original de www.beerandchips.net
4
5  -- Define as configuracoes de rede
6  wifi.setmode(wifi.STATION)
7  wifi.sta.config("NOME_REDE","SENHA_REDE")
8  wifi.sta.connect()
9
10 -- Definicao de pino do DHT22
11 PIN = 4 --  data pin, GPIO2
12 dht22 = require("dht22_min")
13 chipserial = node.chipid()
14
15 -- Cria e roda o web server
16 srv=net.createServer(net.TCP, 4)
17 print("Server created on " .. wifi.sta.getip())
18 srv:listen(80,function(conn)
19 conn:on("receive",function(conn,request)
20 print(request)
```



The End!