**Tutorial ScadaBR/Modbus TCP**

Aluna: Taynara Costa Luz

* Protocolo Modbus TCP

O protocolo Modbus é uma estrutura de mensagem aberta desenvolvida pela Modicon na década de 70, utilizada para comunicação entre dispositivos mestre-escravo / cliente-servidor. O Modbus é um dos protocolos mais utilizados em automação industrial, graças à sua simplicidade e facilidade de implementação, podendo ser utilizado em diversos padrões de meio físico, como:

* RS-232;
* RS-485;
* **Ethernet TCP/IP (MODBUS TCP).**

Modbus TCP é uma implementação do protocolo Modbus baseado em TCP/IP. Utiliza a pilha TCP/IP para comunicação e adiciona ao quadro Modbus um cabeçalho específico chamado MBAP (MODBUS Application Protocol).

O cliente Modbus TCP deve iniciar uma conexão TCP com o servidor a fim de enviar as requisições. A porta TCP **502** é a porta padrão para conexão com servidores Modbus TCP.

Na figura abaixo vemos um exemplo de rede com o protocolo Modbus, com um gateway fazendo a conexão entre os dois tipos de Modbus, o serial em RS-485 e o TCP/IP em ethernet.

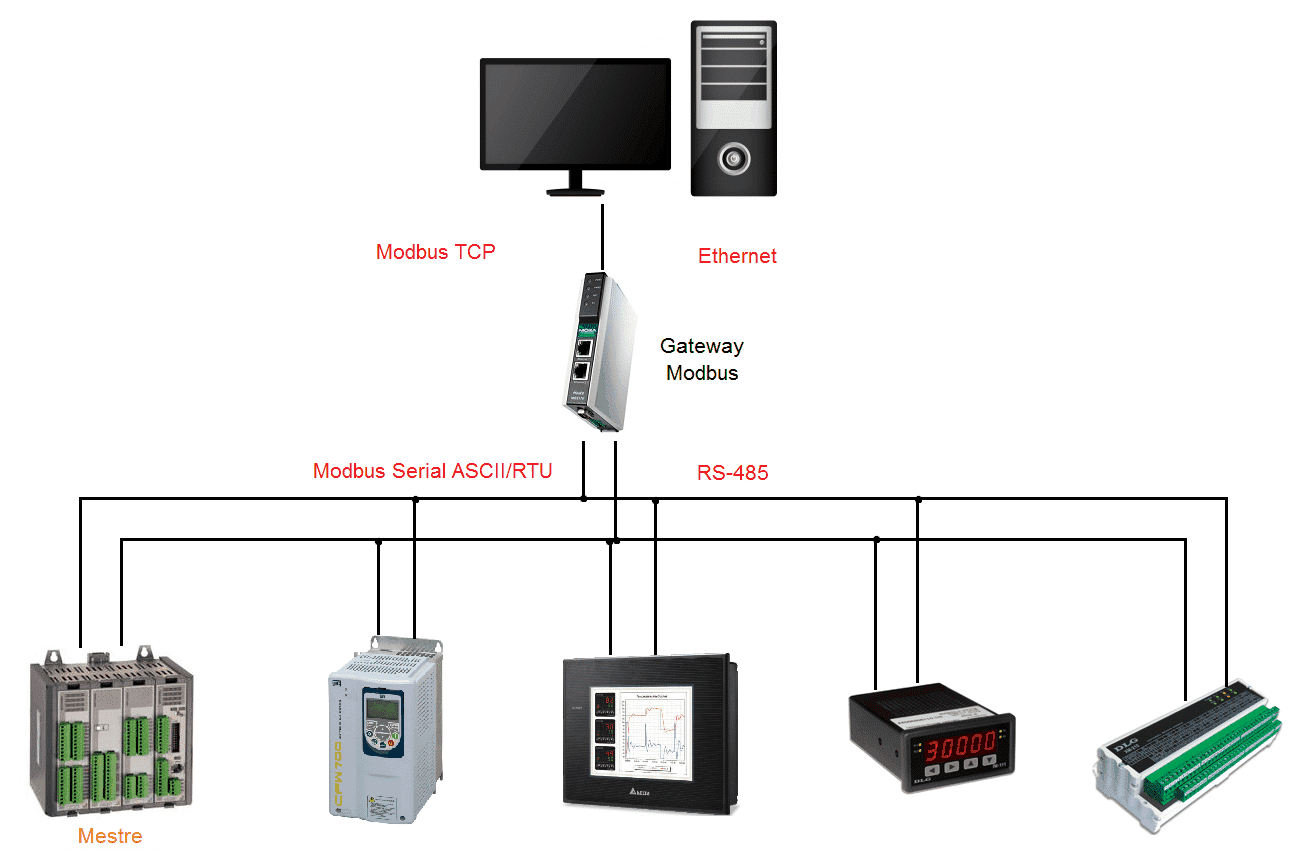


Figura 1 – Exemplo de rede com o protocolo Modbus.

A estação mestre inicia a comunicação solicitando que os escravos enviem seus dados. Os escravos, por sua vez, recebem a requisição do mestre e retornam os dados solicitados. Os dados transmitidos podem ser discretos ou numéricos, ou seja, é possível enviar valores numéricos como temperatura e pressão ou enviar um bit para ligar e desligar um motor. Na figura a seguir podemos observar como é constituído o quadro de mensagens no protocolo Modbus.

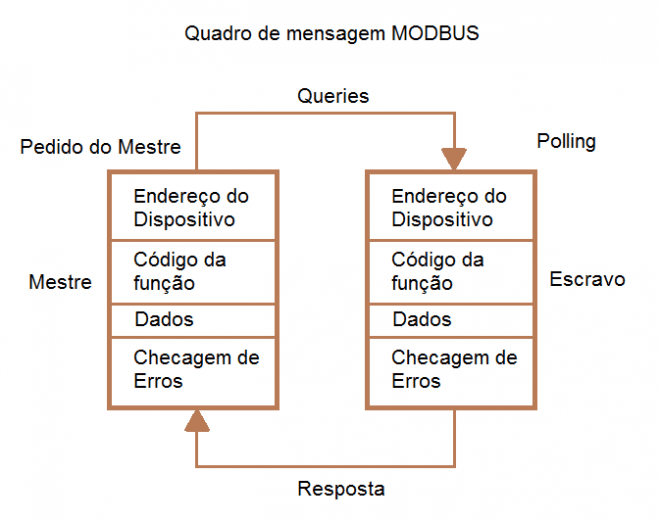


Figura 2 – Quadro de mensagens para Protocolo Modbus

Na especificação do protocolo Modbus estão definidos dois modos de transmissão:

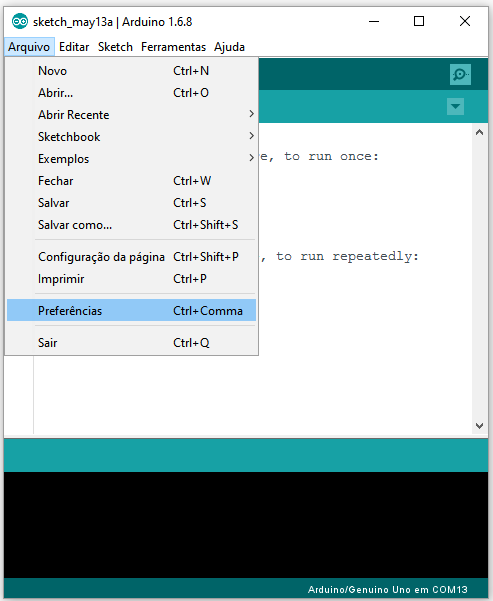
* ASCII;
* RTU.

Quando os equipamentos são configurados para se comunicarem em uma rede Modbus usando ASCII (American Standard Code for Information Interchange), cada byte em uma mensagem é enviado como dois caracteres ASCII. Apesar de gerar mensagens legíveis pela tabela ASCII esse modo consome mais recursos da rede. A principal vantagem dessa modalidade é que permite que os intervalos de tempo sejam cerca de um segundo para correr entre os caracteres sem causar erro.

* Comunicação NodeMCU/Arduino IDE

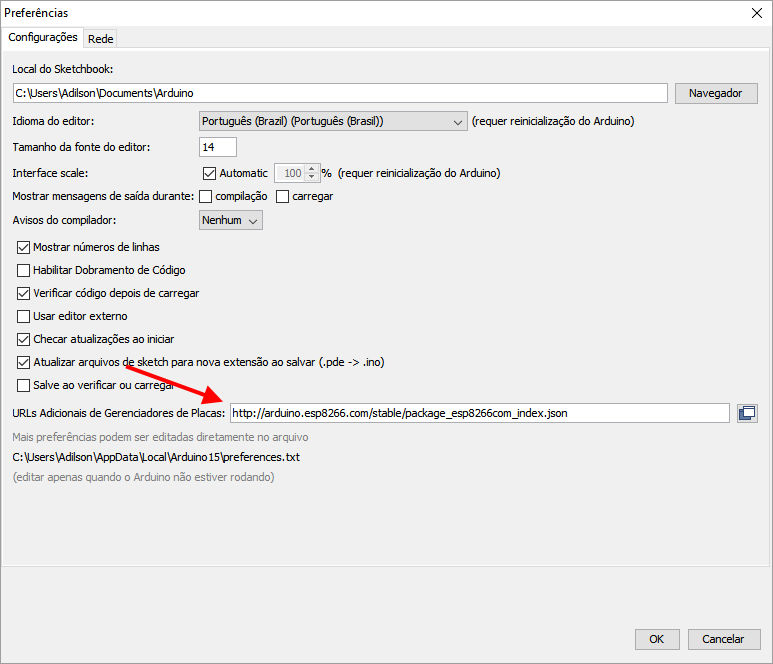
A comunicação da placa NodeMCU e a plataforma Arduino IDE pode ser feito por meio do gerenciador de placas da IDE, onde é necessário incluir não só o NodeMCU, mas também outras placas da família ESP8266.

1. Entre na IDE do Arduino e clicar em **Arquivo -> Preferências**:



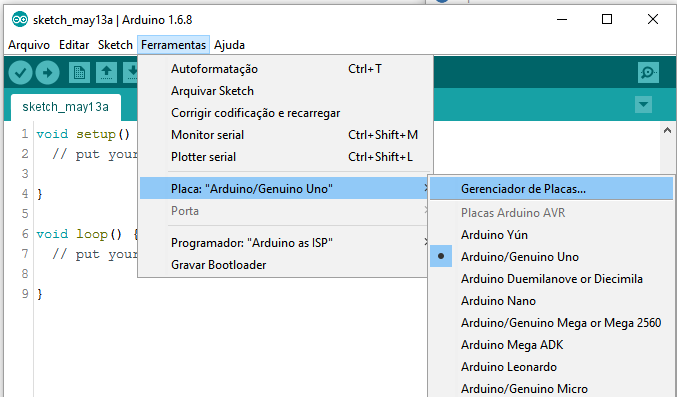
1. Na tela seguinte, digite o link abaixo no campo URLs adicionais de Gerenciadores de Placas:

**URL:** <http://arduino.esp8266.com/stable/package_esp8266com_index.json>

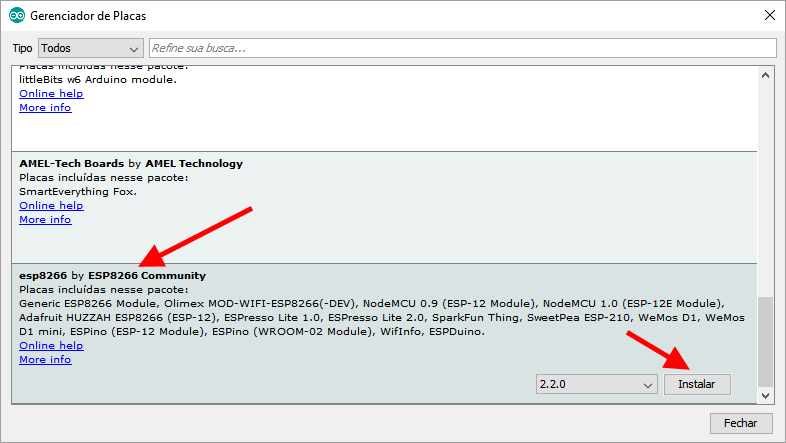


Clique em OK para retornar à tela principal da IDE.

1. Agora clique em Ferramentas -> Placa -> Gerenciador de Placas



1. Utilize a barra de rolagem para encontrar o esp8266 by ESP8266 Community e clique em INSTALAR



Após alguns minutos as placas da linha ESP8266 já estarão disponíveis na lista de placas da IDE do Arduino.

* Como configurar o ESP8266 à internet

1. Na IDE do Arduino vá em “Ferramentas”. Escolha a placa que você está em mãos. No meu caso é a “NodeMCU 1.0 (ESP-12E Module)”.

O código fonte para testes de conexão possui poucas linhas e está comentado.

Inserir o nome da sua rede e a senha, respectivamente em WLAN SSID e WLAN PASS.

/ --- Inclusão da biblioteca ---

#include <ESP8266WiFi.h>

// ======================================================================

// --- Dados de Acesso do seu roteador ---

#define WLAN SSID   "XXXXXXXX"   // nome da sua rede

#define WLAN PASS  "XXXXXXXX"   // senha de acesso do seu roteador

void setup()

{

  Serial.begin(115200); // Inicia Serial com Baud Rate de 115200

WiFi.begin(“WLAN SSID”, "WLAN PASS");

while (WiFi.status() != WL\_CONNECTED) {

        delay(500);

        Serial.print(".");

    }

    Serial.println("");

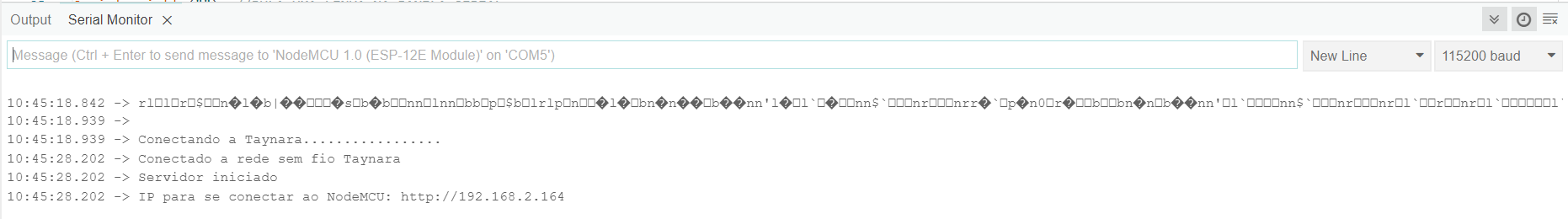
    Serial.println("WiFi connected");

    Serial.println("IP address: ");

    Serial.println(WiFi.localIP());

1. Para verificar se a conexão foi ou não bem sucedida é necessário que o monitor serial funcione. Para isso, verificar se a velocidade está 115200.

Na programação existe a ação de conectar com o roteador. Essa é a primeira etapa para podermos conectar o ESP8266 à Internet. O endereço IP que você visualizará no monitor serial é local, ou seja, atribuído pelo seu roteador.



* Comunicação Modbus/ESP8266 na plataforma Arduino IDE

1. Biblioteca Modbus para Arduino

**http://github.com/andresarmento/modbus-arduino**

**Copyright by André Sarmento Barbosa**

Esta biblioteca permite que seu arduino se comunique através do protocolo Modbus. Na versão atual a biblioteca permite que o arduino opere como escravo, suportando tando Modbus Serial quanto Modbus IP.

Características:

* Opera como escravo (Modo mestre em desenvolvimento)
* Suporta Modbus Serial (RS-232 ou RS485) e Modbus IP (TCP não keep-alive)
* Responde todos os tipos de exceção para as funções suportadas
* Suporta as seguintes funções Modbus:

0x01 - Read Coils

0x02 - Read Input Status (Read Discrete Inputs)

0x03 - Read Holding Registers

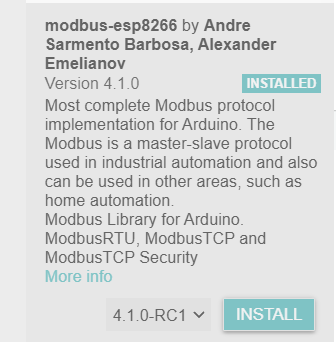
0x04 - Read Input Registers

0x05 - Write Single Coil

0x06 - Write Single Register

0x0F - Write Multiple Coils

0x10 - Write Multiple Registers

1. Biblioteca Modbus para ESP8266

Esta biblioteca permite que seu ESP8266 se comunique através do protocolo Modbus. Na versão atual a biblioteca permite que o arduino opere como escravo, suportando o Modbus IP via rede wireless.

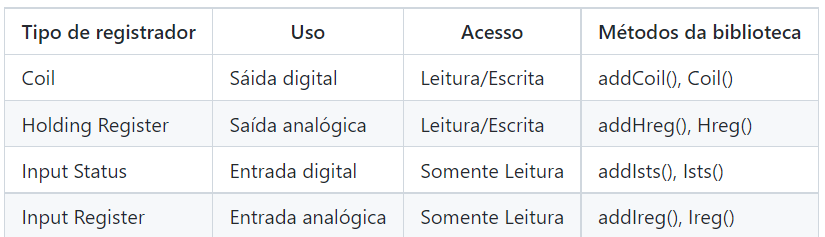
**https://github.com/emelianov/modbus-esp8266**

**Copyright by Alexander Emelianov**

Assim para início de código, temos:

 #include <ESP8266WiFi.h>

#include <ModbusIP\_ESP8266.h>

* Usando a biblioteca Modbus

Leitura de entradas e saídas discretas

• Read Coil Status: leitura de saídas discretas – código 01. Formato: endereço inicial 2 bytes e número de

saídas 2 bytes;

• Read Input Status: leitura de entradas discretas – código 02. Formato: endereço inicial 2 bytes e número de saídas 2 bytes.

Leitura de Registradores

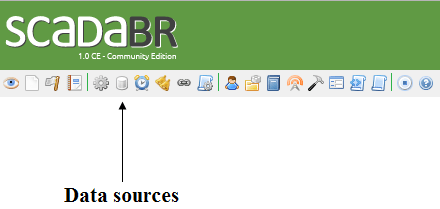
• Read Holding Registers: leitura de registradores do dispositivo escravo – código 03. Formato: endereço

inicial 2 bytes e número de registradores 2 bytes;

• Read Input Registers: leitura dos valores das entradas dos registros – código 04. Formato: endereço inicial 2 bytes e número de registradores 2 bytes;

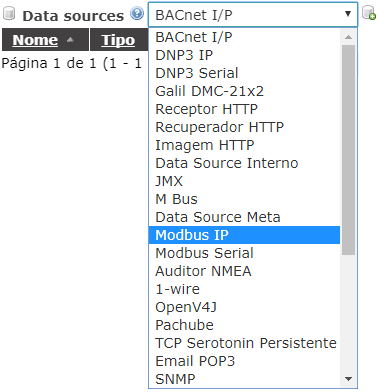
• Preset Single Register: escrita de um valor em registrador – código 06. Formato: endereço 2 bytes e valor 2 bytes

* Modbus e ScadaBR-1.2CE

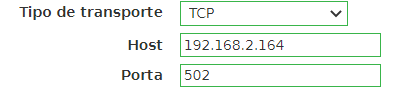


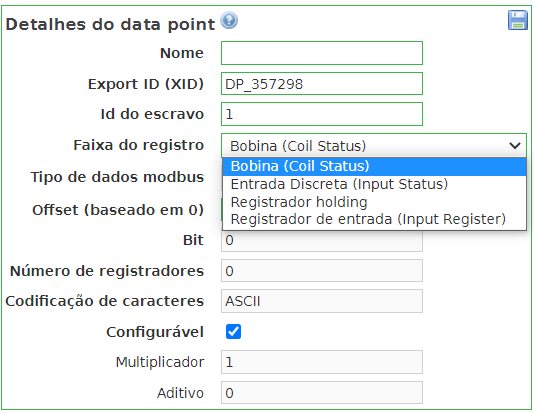
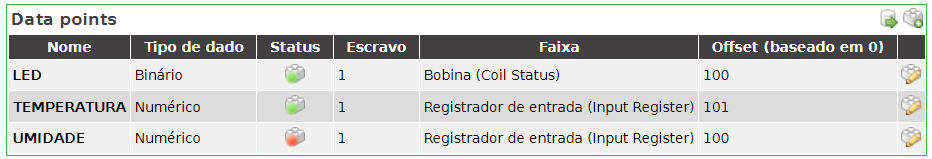
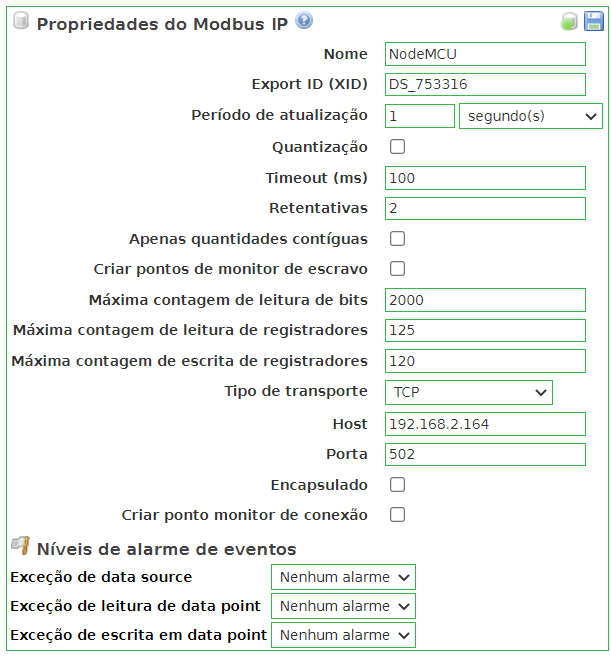
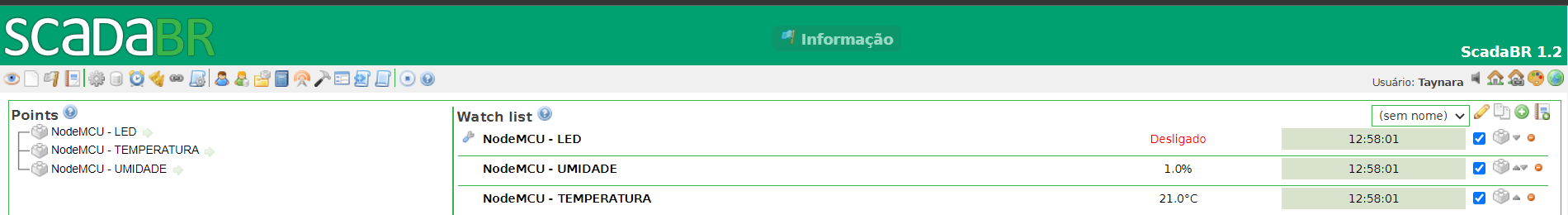
Configurando o Scadabr

1. Identifique o elemento com o qual irá se comunicar. Isto deve ser feito através da criação de um elemento chamado **Data source**, que por sua vez, será a ponte entre a placa NodeMCU e o ScadaBR.
2. Ao clicar para a criação do Data point, devemos dizer qual tipo de comunicação existirá entre os 2 elementos. Como dito anteriormente utilizaremos o protocolo Modbus através da comunicação serial, portanto basta na lista de seleção encontrar a opção **Modbus IP** e clicar no ícone à direita da lista em questão.



1. É necessário também especificar mais alguns parâmetros relativos à comunicação que será estabelecida, como por exemplo: o **endeceço** **IP** previamente configurado no código do NodeMCU e o tipo de transporte. Não há necessidade de alterarmos o campo **Porta.**



1. Após a criação e configuração do Data source, devemos criar uma pequena estrutura denominada **Data point**. Os Data points são os elementos que irão conter os valores relacionados com as entradas e saídas da nossa placa NodeMCU. Para realizar este procedimento, vamos até a parte inferior da tela e em seguida clicamos no ícone localizado à direita da tabela referente aos Data points.
2. Após clicarmos no ícone citado, o ScadaBR nos dará uma janela com algumas opções para serem definidas. Primeiramente, devemos **nomear** o Data point e em seguida, devemos garantir que o campo **Faixa do registro** esteja com algumas das opções selecionadas. (consultar funções da biblioteca Modbus), assim como o **Tipo de dados modbus** e **Offset**.
3. A figura abaixo mostra como a tabela onde estão organizados os Data points ficará após a criação realizada anteriormente. Para habilitar os data points, basta clicar no círculo vermelho presente no campo Status do Data point criado.
4. Devemos ir ao topo da página, salvar as configurações e habilitar o Data source
5. Ao retornar a **Watch list** e no lado esquerdo da interface apresentada, clicar na seta para visualizar o estado do data point criado.

/\*

    Copyright by André Sarmento Barbosa

And Alexander Emelianov

  http://github.com/andresarmento/modbus-arduino

  https://github.com/emelianov/modbus-esp8266

\*/

#ifdef ESP8266

 #include <ESP8266WiFi.h>

#else //ESP32

 #include <WiFi.h>

#endif

#include <ModbusIP\_ESP8266.h>

#nclude "DHT.h"

#define DHTPIN 4     // what digital pin the DHT22 is conected to

#define DHTTYPE DHT22   // there are multiple kinds of DHT sensors

//Modbus Registers Offsets

const int SENSOR\_IREG = 100;

const int LED\_COIL = 100;

const int ledPin = 0; //GPIO0

int LEITURA = 0;

int UMIDADE = 0;

const int SENSOR2\_IREG = 101;

DHT dht(DHTPIN, DHTTYPE);

//ModbusIP object

ModbusIP mb;

long ts;

void setup() {

    Serial.begin(115200);

    WiFi.begin("Taynara", "willa2809");

    while (WiFi.status() != WL\_CONNECTED) {

        delay(500);

        Serial.print(".");

    }

    Serial.println("");

    Serial.println("WiFi connected");

    Serial.println("IP address: ");

    Serial.println(WiFi.localIP());

    mb.server();    //Start Modbus IP

     pinMode(ledPin, OUTPUT);

    dht.begin();

    // Add SENSOR\_IREG register - Use addIreg() for analog Inputs

    mb.addIreg(SENSOR\_IREG);

    mb.addIreg(SENSOR2\_IREG);

    mb.addCoil(LED\_COIL);

    ts = millis();

}

void loop() {

   //Call once inside loop() - all magic here

   mb.task();

   digitalWrite(ledPin, mb.Coil(LED\_COIL));

    float t = dht.readTemperature();

    LEITURA = analogRead(A0);

    UMIDADE = map(LEITURA, 0, 1024, 100, 0);

   //Read each two seconds

   if (millis() > ts + 2000) {

       ts = millis();

       //Setting raw value (0-1024)

       mb.Ireg(SENSOR\_IREG, UMIDADE);

       mb.Ireg(SENSOR2\_IREG, t);

    Serial.println("Umidade: ");

        Serial.println(UMIDADE);

    Serial.println("Temperatura: ");

        Serial.print(t);

        Serial.println(" \*C ");

   }

   delay(10);  
}

**Referências Bibliográficas**

[1] <https://www.embarcados.com.br/protocolo-modbus/>

[2] <https://www.filipeflop.com/blog/programar-nodemcu-com-ide-arduino/>

[3] <https://www.filipeflop.com/blog/como-conectar-o-esp8266-a-internet/>

[4] <https://github.com/andresarmento/modbus-arduino/blob/master/README_pt_BR.md>

[5] <https://github.com/andresarmento/modbus-esp8266/blob/master/README_pt_BR.md>

[6] <https://portal.vidadesilicio.com.br/scadabr-modbus-tcp-ip/>