

# MONITORAMENTO COM SCADABR

**Sistema de Monitoramento de Ambiente em rede local e remoto utilizando do ScadaBR.**

**Danilo Braz – danilobrazsilva@gmail.com**

## Sumário

Ambiente 1 - Sala de Tecnologia de Informação.....	2
Código Arduino:.....	3
DataSource.....	5
Ambiente 2 - Sala Técnica .....	7
Código Arduino.....	8
DataSouce .....	10
Monitoramento do áudio no ScadaBR, .....	11
Ambiente 3 – Casa de Equipamentos de Transmissão .....	12
Código Arduino.....	13
DataSource.....	19

Ambiente 1 - Sala de Tecnologia de Informação.

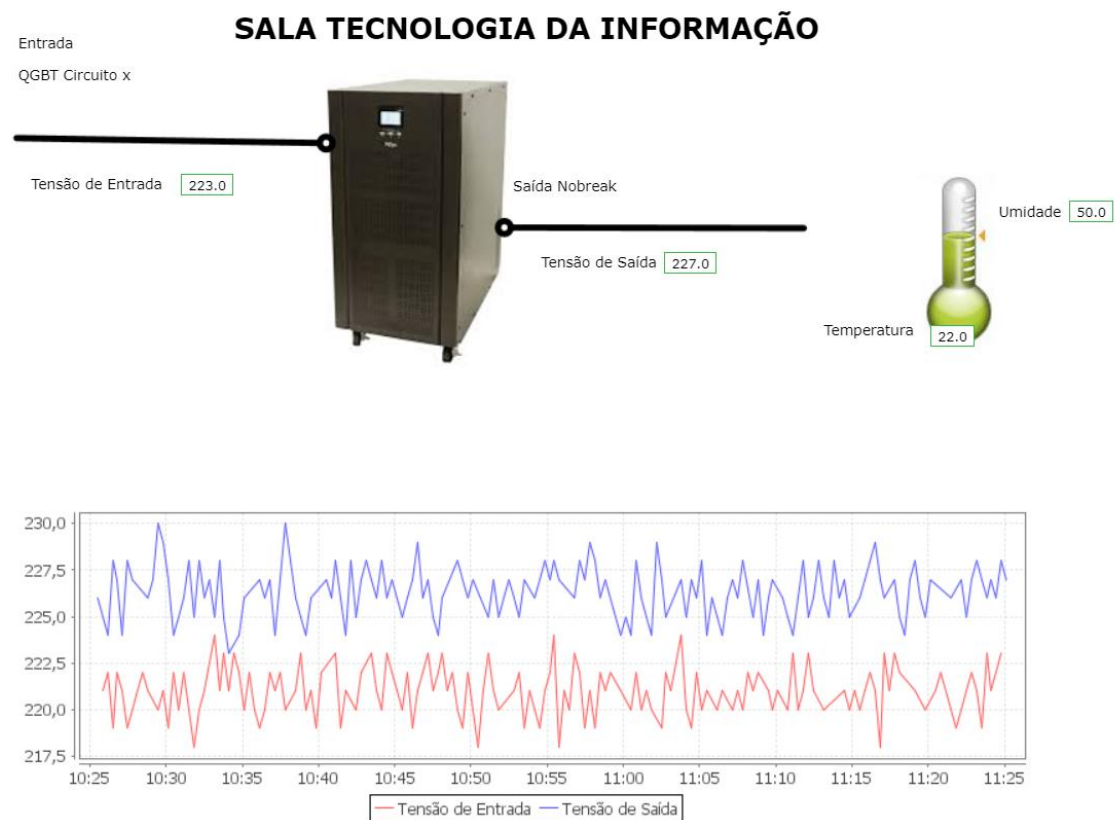


Figura 1 - Representação Gráfica ScadaBR - Sala T.I

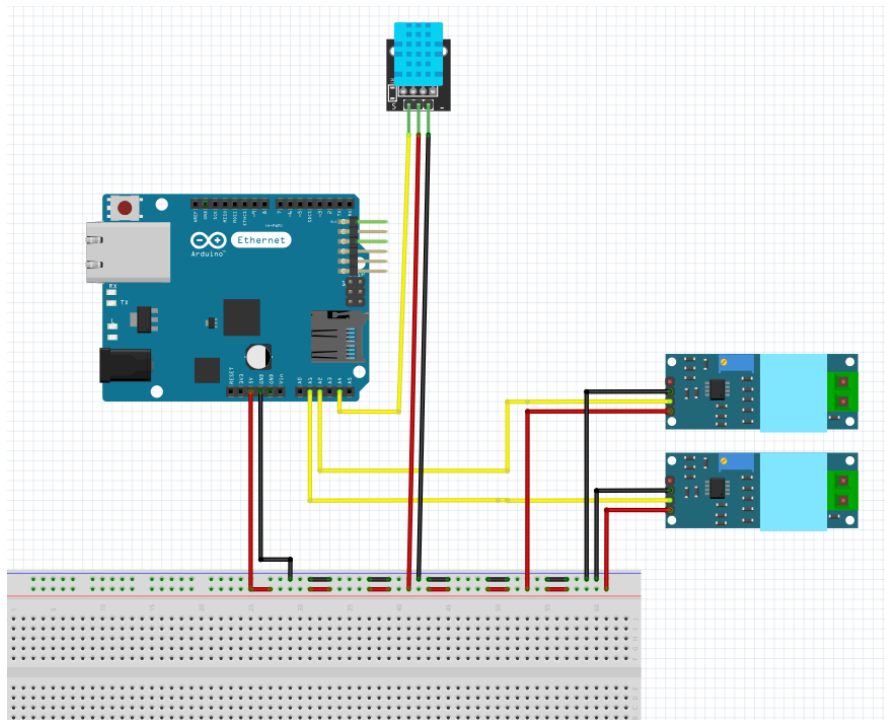


Figura 2 Esquemático - Sala TI

Itens Utilizados:

1 Arduino Uno com Shield Ethernet;

2 Sensores de tensão - ZMPT101B;

1 Sensores de temperatura - DHT11.

1 Shilel V3 com bateria 18650. Para manter o equipamento ligado mesmo com falta de energia.

## Código Arduino:

```
#include <SPI.h>
#include <Ethernet.h>
#include <Modbus.h>
#include <ModbusIP.h>
#include "EmonLib.h" //INCLUSÃO DE BIBLIOTECA Sensores de Tensão
#include "dht.h" //INCLUSÃO DE BIBLIOTECA DE TEMPERATURA DHT11
```

//Pinos Utilizados

```
const int pintensao0 = 1; //PINO ANALÓGICO UTILIZADO PELO SENSOR
const int pintensao1 = 2; //PINO ANALÓGICO UTILIZADO PELO SENSOR
const int pintemp = A4; //PINO ANALÓGICO UTILIZADO PELO SENSOR
const int pinaudio = A5; //PINO ANALÓGICO UTILIZADO PELO SENSOR
```

```
#define CALIB_SENSOR 216.2 //VALOR DE CALIBRAÇÃO (DEVE SER AJUSTADO EM PARALELO COM UM
MULTÍMETRO)
```

```
ModbusIP mb;
```

```

//Modbus Registers Offsets
const int SENSOR_TENSAO1 = 100;
const int SENSOR_TENSAO2 = 101;
const int SENSOR_TEMP = 103;
const int SENSOR_UMID = 104;
const int SENSOR_AUDIO = 110;

//ModbusIP object
long ts;

EnergyMonitor emon1; //CRIA UMA INSTÂNCIA
EnergyMonitor emon2; //CRIA UMA INSTÂNCIA
dht DHT; //VARIÁVEL DO TIPO DHT

void setup() {
  //Configuracao da rede
  byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xEA };
  byte ip[] = { 192, 168, 19, 215 };
  mb.config(mac,ip);

  Serial.begin(115200); //Serial

  //SENSORES DE TENSÃO
  emon1.voltage(pintensao0, CALIB_SENSOR, 1.7); //PASSA PARA A FUNÇÃO OS PARÂMETROS (PINO ANALÓGICO /
  VALOR DE CALIBRAÇÃO / MUDANÇA DE FASE)
  emon2.voltage(pintensao1, CALIB_SENSOR, 1.7); //PASSA PARA A FUNÇÃO OS PARÂMETROS (PINO ANALÓGICO /
  VALOR DE CALIBRAÇÃO / MUDANÇA DE FASE)

  //Add SENSOR_IREG register - Use addIreg() for analog Inputs
  mb.addIreg(SENSOR_TENSAO1);
  mb.addIreg(SENSOR_TENSAO2);
  mb.addIreg(SENSOR_TEMP);
  mb.addIreg(SENSOR_UMID);
  mb.addIreg(SENSOR_AUDIO);
}

void loop() {
  //Call once inside loop() - all magic here
  mb.task();

  //sensores de tensão
  emon1.calcVI(17,500); //FUNÇÃO DE CÁLCULO (17 SEMICICLOS, TEMPO LIMITE PARA FAZER A MEDIÇÃO)
  emon2.calcVI(17,500); //FUNÇÃO DE CÁLCULO (17 SEMICICLOS, TEMPO LIMITE PARA FAZER A MEDIÇÃO)
  float t0 = emon1.Vrms; //VARIÁVEL RECEBE O VALOR DE TENSÃO RMS OBTIDO
  float t1 = emon2.Vrms; //VARIÁVEL RECEBE O VALOR DE TENSÃO RMS OBTIDO

  //sensor de temperatura
  DHT.read11(pintemp); //LÊ AS INFORMAÇÕES DO SENSOR
  float temperatura = DHT.temperature; //VARIÁVEL RECEBE A TEMPERATURA MEDIDA
  float umidade = DHT.humidity; //VARIÁVEL RECEBE A UMIDADE MEDIDA

  //Read each two seconds
  if (millis() > ts + 2000) {
    ts = millis();

    //Setting raw value (0-1024)
    mb.Ireg(SENSOR_TENSAO1, t0);
    mb.Ireg(SENSOR_TENSAO2, t1);
    mb.Ireg(SENSOR_TEMP, temperatura);
    mb.Ireg(SENSOR_UMID, umidade);
  }
}

```

```

    }
    // IMPRIME O TEXTO NA SERIAL
    // Serial.println("Tensão do sensor1 : ");
    // Serial.println(t0, 0);
    // Serial.println("");
    // Serial.println("Tensão do sensor2 : ");
    // Serial.println(t1, 0);
    // Serial.println("");
    // Serial.println("Tensão de Temperatura : ");
    // Serial.println(temperatura);
    // Serial.println("");
    // Serial.println("-----");
    // Serial.print("Umidade: "); //IMPRIME O TEXTO NA SERIAL
    // Serial.print(umidade, 0); //IMPRIME NA SERIAL O VALOR DE UMIDADE MEDIDO
    // Serial.print("%"); //ESCREVE O TEXTO EM SEGUIDA
    // Serial.print(" / Temperatura: "); //IMPRIME O TEXTO NA SERIAL
    // Serial.print(temperatura, 0); //IMPRIME NA SERIAL O VALOR DE UMIDADE MEDIDO E REMOVE A PARTE
    // DECIMAL
    // Serial.println("*C"); //IMPRIME O TEXTO NA SERIAL

    delay(100);

}

```

## DataSource

### Novo DataSource – Modbus IP

Nome

ArduinoSalaTI

Export ID (XID)

DS\_158574

Período de atualização

20

segundo(s)

Quantização

☐

Timeout (ms)

500

Retentativas

2

Apenas quantidades contíguas

☐

Criar pontos de monitor de escravo

☐

Máxima contagem de leitura de bits

2000

Máxima contagem de leitura de registradores

125

Máxima contagem de escrita de registradores

120

Tipo de transporte

TCP

Host

192.168.19.215

Porta

502

Encapsulado

☐

Criar ponto monitor de conexão

☐

Níveis de alarme de eventos

Exceção de data source

Urgente

Exceção de leitura de data point

Urgente

Exceção de escrita em data point

Urgente

Pesquisar por nós

Cancelar

Nós encontrados

Teste de localizador de p

Id do escravo

1

Faixa do registro

S

Tipo de dados modbus

B

Offset (baseado em 0)

0

Bit

0

Número de registradores

0

Codificação de caracteres

As

Ler

Ad

Data points

Nome	Tipo de dado	Status	Escravo	Faixa	Offset (baseado em 0)
Temperatura	Numérico		2	Registrador de entrada	103
Tensão de Entrada	Numérico		2	Registrador de entrada	100
Tensão de Saída	Numérico		2	Registrador de entrada	101
Umidade	Numérico		2	Registrador de entrada	104

Figura 3 - DataSource Sala TI

Configurar o Período de Atualização

Configurar a Rede

Adicionar cada Data point de acordo com offset configurado no código do Arduino.

## Ambiente 2 - Sala Técnica

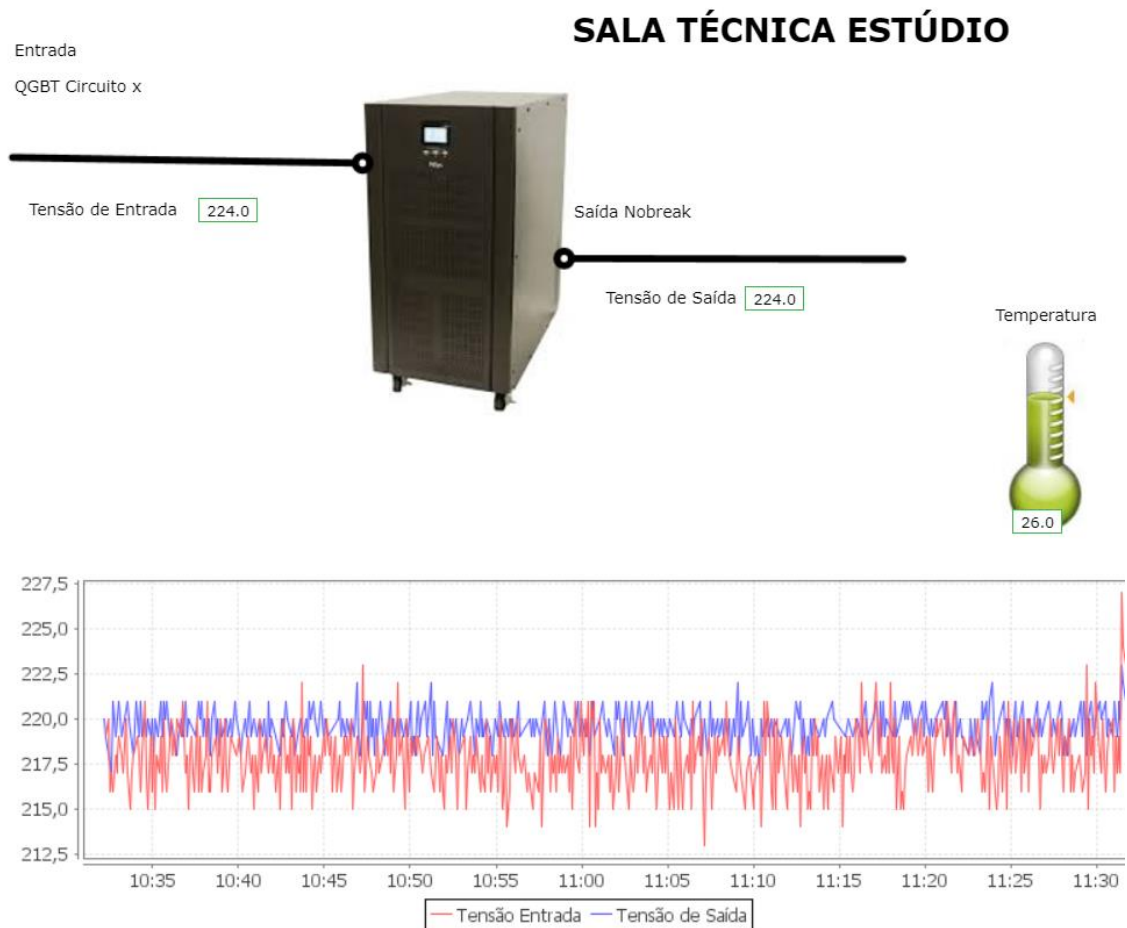


Figura 4 - Representação Gráfica Sala Técnica

- 1 Arduino Uno com Shield Ethernet;
- 2 Sensores de tensão - ZMPT101B;
- 1 Sensores de temperatura - DHT11.
- 1 Sensor de som KY-038
- 1 Sensor de Som
- 1 Shilel V3 com bateria 18650. Para manter o equipamento ligado mesmo com falta de energia.

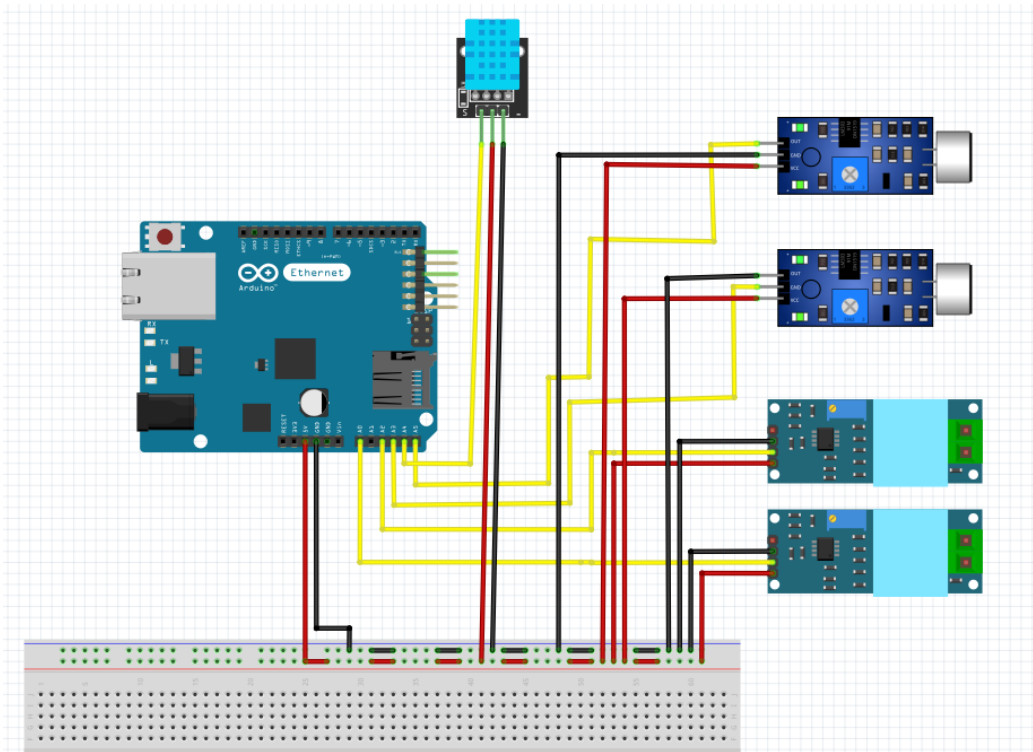


Figura 5 - Esquemático Sala Técnica

## Código Arduino

```
#include <SPI.h>
#include <Ethernet.h>
#include <Modbus.h>
#include <ModbusIP.h>
#include "EmonLib.h" //INCLUSÃO DE BIBLIOTECA Sensores de Tensão
#include "dht.h" //INCLUSÃO DE BIBLIOTECA DE TEMPERATURA DHT11

//Pinos Utilizados
const int pintensao0 = 1; //PINO ANALÓGICO UTILIZADO PELO SENSOR
const int pintensao1 = 2; //PINO ANALÓGICO UTILIZADO PELO SENSOR
const int pintemp = A4; //PINO ANALÓGICO UTILIZADO PELO SENSOR
const int pinaudio = A5; //PINO ANALÓGICO UTILIZADO PELO SENSOR

#define CALIB_SENSOR 216.2 //VALOR DE CALIBRAÇÃO (DEVE SER AJUSTADO EM PARALELO COM UM
MULTÍMETRO)

ModbusIP mb;

//Modbus Registers Offsets
const int SENSOR_TENSAO1 = 100;
const int SENSOR_TENSAO2 = 101;
const int SENSOR_TEMP = 103;
const int SENSOR_UMID = 104;
const int SENSOR_AUDIO = 110;

//ModbusIP object
long ts;
```



```

EnergyMonitor emon1; //CRIA UMA INSTÂNCIA
EnergyMonitor emon2; //CRIA UMA INSTÂNCIA
dht DHT; //VARIÁVEL DO TIPO DHT

void setup() {
  //Configuracao da rede
  byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xEA };
  byte ip[] = { 192, 168, 19, 215 };
  mb.config(mac,ip);

  Serial.begin(115200); //Serial

  //SENSORES DE TENSÃO
  emon1.voltage(pintensao0, CALIB_SENSOR, 1.7); //PASSA PARA A FUNÇÃO OS PARÂMETROS (PINO ANALÓGICO /
  VALOR DE CALIBRAÇÃO / MUDANÇA DE FASE)
  emon2.voltage(pintensao1, CALIB_SENSOR, 1.7); //PASSA PARA A FUNÇÃO OS PARÂMETROS (PINO ANALÓGICO /
  VALOR DE CALIBRAÇÃO / MUDANÇA DE FASE)

  //Add SENSOR_IREG register - Use addIreg() for analog Inputs
  mb.addIreg(SENSOR_TENSAO1);
  mb.addIreg(SENSOR_TENSAO2);
  mb.addIreg(SENSOR_TEMP);
  mb.addIreg(SENSOR_UMID);
  mb.addIreg(SENSOR_AUDIO);
}

void loop() {
  //Call once inside loop() - all magic here
  mb.task();

  //sensores de tensão
  emon1.calcVI(17,500); //FUNÇÃO DE CÁLCULO (17 SEMICICLOS, TEMPO LIMITE PARA FAZER A MEDIÇÃO)
  emon2.calcVI(17,500); //FUNÇÃO DE CÁLCULO (17 SEMICICLOS, TEMPO LIMITE PARA FAZER A MEDIÇÃO)
  float t0 = emon1.Vrms; //VARIÁVEL RECEBE O VALOR DE TENSÃO RMS OBTIDO
  float t1 = emon2.Vrms; //VARIÁVEL RECEBE O VALOR DE TENSÃO RMS OBTIDO

  //sensor de temperatura
  DHT.read11(pintemp); //LÊ AS INFORMAÇÕES DO SENSOR
  float temperatura = DHT.temperature; //VARIÁVEL RECEBE A TEMPERATURA MEDIDA
  float umidade = DHT.humidity; //VARIÁVEL RECEBE A UMIDADE MEDIDA

  //Read each two seconds
  if (millis() > ts + 2000) {
    ts = millis();

    //Setting raw value (0-1024)
    mb.Ireg(SENSOR_TENSAO1, t0);
    mb.Ireg(SENSOR_TENSAO2, t1);
    mb.Ireg(SENSOR_TEMP, temperatura);
    mb.Ireg(SENSOR_UMID, umidade);
  }

  // IMPRIME O TEXTO NA SERIAL
  // Serial.println("Tensão do sensor1 : ");
  // Serial.println(t0, 0);
  // Serial.println("");
  // Serial.println("Tensão do sensor2 : ");
  // Serial.println(t1, 0);
  // Serial.println("");
  // Serial.println("Tensão de Temperatura : ");
  // Serial.println(temperatura);

```

```
// Serial.println("");
// Serial.println("-----");
// Serial.print("Umidade: "); //IMPRIME O TEXTO NA SERIAL
// Serial.print(umidade, 0); //IMPRIME NA SERIAL O VALOR DE UMIDADE MEDIDO
// Serial.print("%"); //ESCREVE O TEXTO EM SEGUIDA
// Serial.print(" / Temperatura: "); //IMPRIME O TEXTO NA SERIAL
// Serial.print(temperatura, 0); //IMPRIME NA SERIAL O VALOR DE UMIDADE MEDIDO E REMOVE A PARTE
DECIMAL
// Serial.println("*C"); //IMPRIME O TEXTO NA SERIAL

delay(100);

}
```

## DataSouce

Nome

Sala Técnica Estúdio

Export ID (XID)

DS\_908527

Período de atualização

5

segundo(s)

Quantização

☐

Timeout (ms)

500

Retentativas

2

Apenas quantidades contíguas

☐

Criar pontos de monitor de escravo

☐

Máxima contagem de leitura de bits

2000

Máxima contagem de leitura de registradores

125

Máxima contagem de escrita de registradores

120

Tipo de transporte

TCP

Host

192.168.19.216

Porta

502

Encapsulado

☐

Criar ponto monitor de conexão

☐

Pesquisar por nós

Cancelar

Nós encontrados

Teste de localizador de p

Id do escravo

1

Faixa do registro

:

Tipo de dados modbus

1

Offset (baseado em 0)

0

Bit

0

Número de registradores

0

Codificação de caracteres

A

Ler

Ar

Níveis de alarme de eventos

Exceção de data source

Nenhum alarme

Exceção de leitura de data point

Nenhum alarme

Exceção de escrita em data point

Nenhum alarme

Data points

Nome	Tipo de dado	Status	Escravo	Faixa	Offset (baseado em 0)
AudioInternet	Numérico		1	Registrador de entrada	110
AudioTransmissor	Numérico		1	Registrador de entrada	111
Temperatura	Numérico		1	Registrador de entrada	103
Tensão de Saída	Numérico		1	Registrador de entrada	101
Tensão Entrada	Numérico		1	Registrador de entrada	100
Umidade	Numérico		1	Registrador de entrada	104

Figura 6 - Datasource Sala Técnica

Tempo de Atualização foi de 5 segundos, nesse exemplo para monitorar o áudio não é ideal ter um tempo de atualização grande pois ele pode entender que está em silêncio as vezes, não captando os momentos de áudio.

Configurar a rede e os datapoints de acordo com a configuração do código do arduino.

### Monitoramento do áudio no ScadaBR,

2 sensores de áudio adaptado com uma entrada p2, verifica sempre que tem áudio, quando tem um período de silêncio o ScadaBR envia uma notificação.

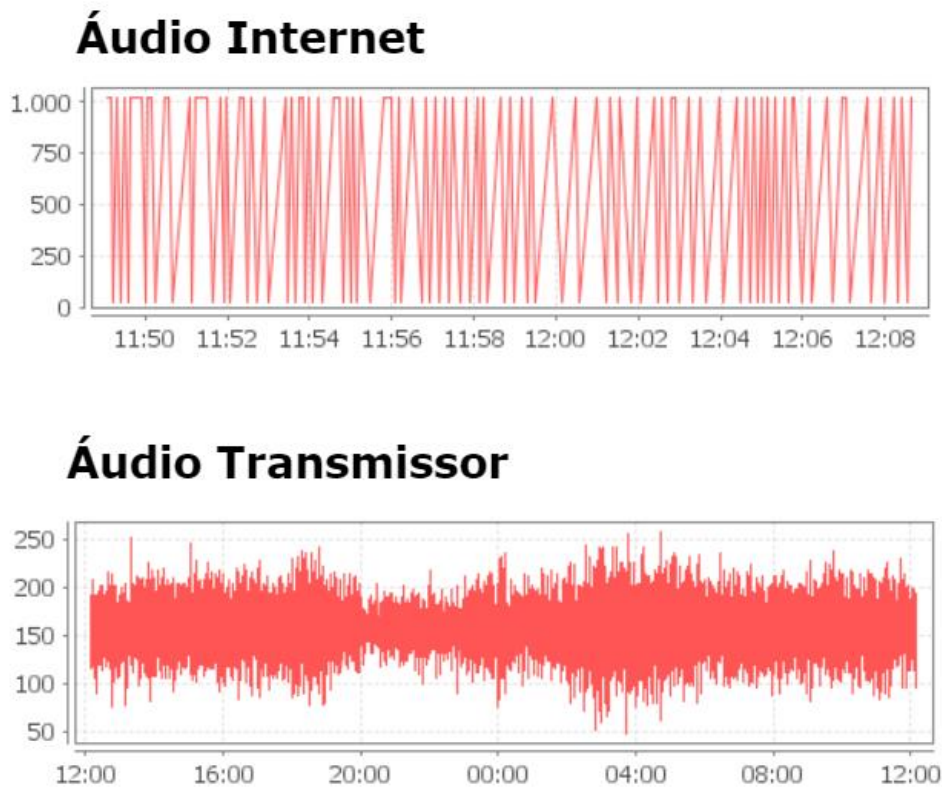


Figura 6 - Representação Gráfica ScadaBr - Áudio

## Ambiente 3 – Casa de Equipamentos de Transmissão

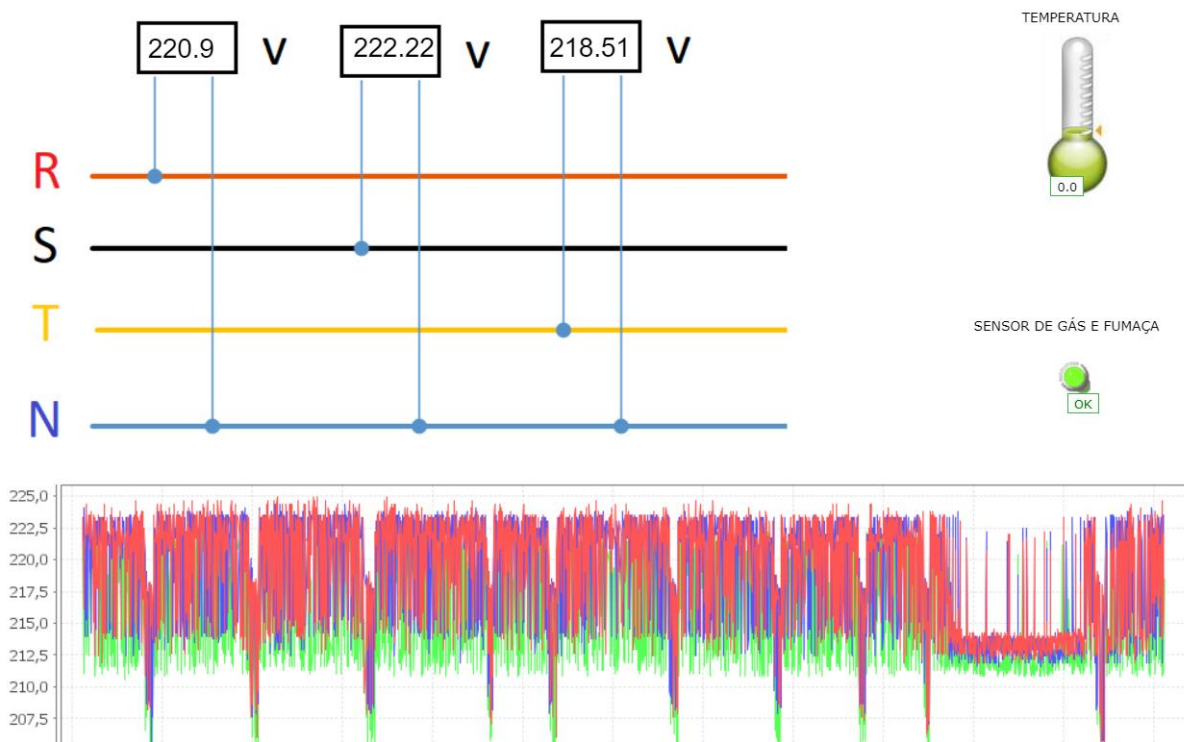
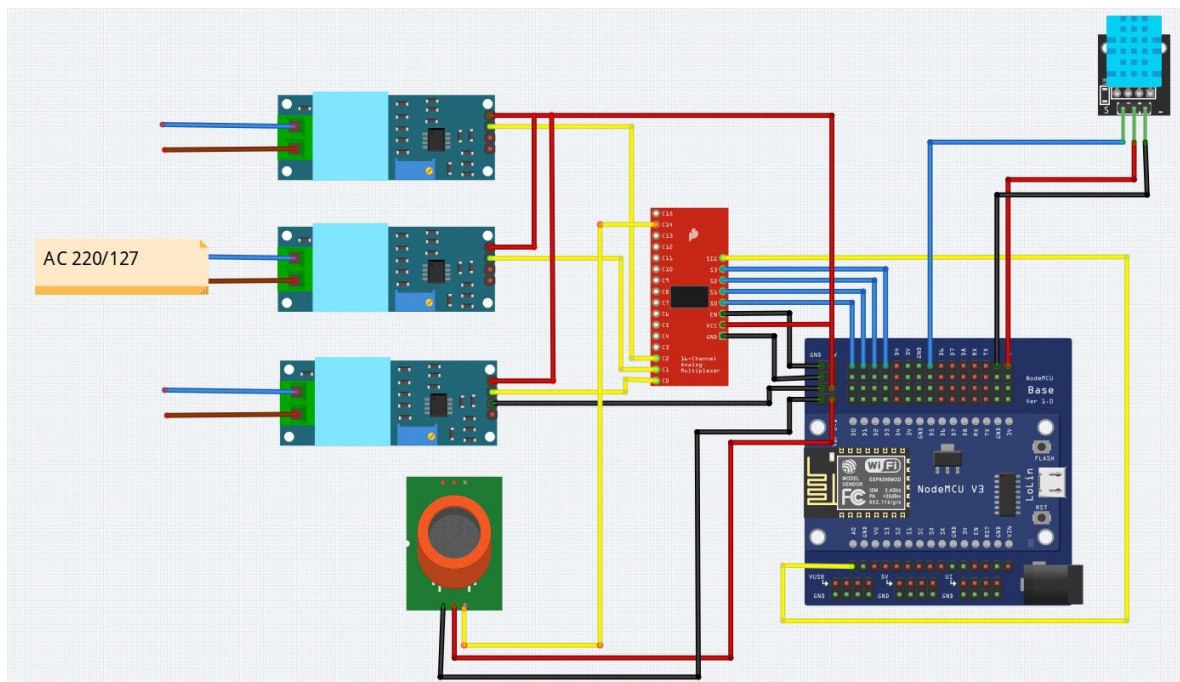


Figura 7 - Representação Gráfica ScadaBR - Transmissor

Itens utilizado:

- 1 Base para Esp8266;
- 1 Esp8266;
- 1 Carregador de Bateria 18650;
- 1 Bateria 18650 3,7 V;
- 3 Sensores Detector de Tensão;
- 1 Sensor de Fumaça e Gás MQ-2;
- 1 Sensor de Temperatura DHT11;
- 1 Multiplexador 16 canais.



*Figura 8- Esquemático Cada do Transmissor*

## Código Arduino

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiManager.h>
#include "DHT.h" //Biblioteca sensor de temperatura

#define DHTPIN 14 // pino D5 do Esp
#define DHTTYPE DHT11 // DHT 11 ou DHT22
#define S0 16 // Porta D0 do Esp8266
#define S1 5 // Porta D1 do Esp8266
#define S2 4 // Porta D2 do Esp8266
#define S3 0 // Porta D3 do Esp8266
#define pinA0 A0 //Porta Analógica

#define SERVER_IP "xxx.xxx.xxx.xxx:xx" //Servidor do ScadaBr, colocar o endereço ScadaBr, ex: 187.15.212.85:8080

//Cria um instância DHT sensor de tensão
DHT dht(DHTPIN, DHTTYPE);

int c = 0 ; //Váriavel contadora;
String datasource; //Variável que armazena qual datasource
float valor; //valor do datasouce, Receber o valor do datasource

void setup() {

    pinMode(pinA0, INPUT); //utilizado para Multipliex
    pinMode(DHTPIN, INPUT); //utilizado para Sensor de temperatura
```

```
pinMode(S0,OUTPUT);//utilizado para Multiplex
pinMode(S1,OUTPUT);//utilizado para Multiplex
pinMode(S2,OUTPUT);//utilizado para Multiplex
pinMode(S3,OUTPUT); //utilizado para Multiplex
```

```
dht.begin();
Serial.begin(115200);
```

```
//Conectar no Wifi
WiFi.mode(WIFI_STA); // explicitly set mode, esp defaults to STA+AP
```

```
// WiFi.mode(WIFI_STA); // it is a good practice to make sure your code sets wifi mode how you want it.
```

```
//WiFiManager, Local initialization. Once its business is done, there is no need to keep it around
WiFiManager wm;
```

```
//reseta a configuração - wipe credentials for testing
//wm.resetSettings();
```

```
// Conectar-se automaticamente usando credenciais salvas,
// se a conexão falhar, ele inicia um ponto de acesso com o nome especificado ( "AutoConnectAP"),
```

```
bool res;
// res = wm.autoConnect(); // auto generated AP name from chipid
// res = wm.autoConnect("AutoConnectAP"); // anonymous ap
//CRIA um SSID com a rede Monitoramento e a senha
res = wm.autoConnect("Monitoramento","0123456789"); // password protected ap
```

```
if(!res) {
    Serial.println("Failed to connect");
    // ESP.restart();
}
else {
    //Conectado
    Serial.println("connected... :");
}
```

```
}
```

```
void loop() {
```

```
//LEITURA DE PORTAS ANALÓGICAS A0 COM MULTIPLEX PARA 16 ENTRADAS ANALÓGICAS
//a cada loop ele irá ler uma porta analógica ou digital e enviar a valor do sensor na variável valor para o ScadaBR
remoto,
//evitando que envie várias requisições para o servidor de uma só vez
```

```
if (c > 16 ){
    delay(10000);
    c = 0;
}
```

```
/**sensores de detector de tensão, foi utilizado um multiplicador 0.2655 para se aproximar do valor da tensão
medida,
// logo esse valor depende da tensão de você está alimentando o sensor e esp, onde o sensor de detecção de
tensão não da grande precisão em valor de tensão.
```

```
//Porta Analógica C0 Sensor de Tensão AC
```

```
if (c == 0){
    digitalWrite(S0,LOW);
```

```

digitalWrite(S1,LOW);
digitalWrite(S2,LOW);
digitalWrite(S3,LOW);

valor = analogRead(pinA0)*0.2655;/**
Serial.print("Sensor de Tensão 1: ");Serial.println(valor);
datasource = "c0";
}

//Porta Analógica C1 Sensor de Tensão AC
if (c == 1){
    digitalWrite(S0,HIGH);
    digitalWrite(S1,LOW);
    digitalWrite(S2,LOW);
    digitalWrite(S3,LOW);

    valor = analogRead(pinA0)*0.2655;/**
    //valor = analogRead(pinA0)*0.21484375;
    Serial.print("Sensor de Tensão 2: ");Serial.println(valor);
    datasource = "c1";

}

if (c == 2){
    //Porta Analógica C2 Sensor de Tensão AC
    digitalWrite(S0,LOW);
    digitalWrite(S1,HIGH);
    digitalWrite(S2,LOW);
    digitalWrite(S3,LOW);
    valor = analogRead(pinA0)*0.2655;/**
    Serial.print("Sensor de Tensão 3: ");Serial.println(valor);
    datasource = "c2";

}

//Porta Analógica C3
if (c == 3){
    digitalWrite(S0,HIGH);
    digitalWrite(S1,HIGH);
    digitalWrite(S2,LOW);
    digitalWrite(S3,LOW);
    Serial.print("Sensor 4 ");Serial.println(analogRead(pinA0));
    datasource = "c3";
}

if (c == 4){
    //Porta Analógica C4
    digitalWrite(S0,LOW);
    digitalWrite(S1,LOW);
    digitalWrite(S2,HIGH);
    digitalWrite(S3,LOW);
    Serial.print("Sensor 5 ");Serial.println(analogRead(pinA0));
    datasource = "c4";
}

//Porta Analógica C5
if (c == 5){
    digitalWrite(S0,HIGH);
    digitalWrite(S1,LOW);
    digitalWrite(S2,HIGH);
    digitalWrite(S3,LOW);
    Serial.print("Sensor 6 ");Serial.println(analogRead(pinA0));
}

```

```

    datasource = "c5";
}

//Porta Analógica C6
if (c == 6){
    digitalWrite(S0,LOW);
    digitalWrite(S1,HIGH);
    digitalWrite(S2,HIGH);
    digitalWrite(S3,LOW);
    Serial.print("Sensor 7 ");Serial.println(analogRead(pinA0));
    datasource = "c6";
}

//Porta Analógica C7
if (c == 7){
    digitalWrite(S0,HIGH);
    digitalWrite(S1,HIGH);
    digitalWrite(S2,HIGH);
    digitalWrite(S3,LOW);
    Serial.print("Sensor 8 ");Serial.println(analogRead(pinA0));
    datasource = "c7";
}

//Porta Analógica C8
if (c == 8){
    digitalWrite(S0,LOW);
    digitalWrite(S1,LOW);
    digitalWrite(S2,LOW);
    digitalWrite(S3,HIGH);
    Serial.print("Sensor 9 ");Serial.println(analogRead(pinA0));
    datasource = "c8";
}

//Porta Analógica C9
if (c == 9){
    digitalWrite(S0,HIGH);
    digitalWrite(S1,LOW);
    digitalWrite(S2,LOW);
    digitalWrite(S3,HIGH);
    Serial.print("Sensor 10 ");Serial.println(analogRead(pinA0));
    datasource = "c9";
}

//Porta Analógica C10
if (c == 10){
    digitalWrite(S0,LOW);
    digitalWrite(S1,HIGH);
    digitalWrite(S2,LOW);
    digitalWrite(S3,HIGH);
    Serial.print("Sensor 11 ");Serial.println(analogRead(pinA0));
    datasource = "c10";
}

//Porta Analógica C11
if (c == 11){
    digitalWrite(S0,HIGH);
    digitalWrite(S1,HIGH);
    digitalWrite(S2,LOW);
    digitalWrite(S3,HIGH);
    Serial.print("Sensor 12 ");Serial.println(analogRead(pinA0));
    datasource = "c11";
}

```



```

}

//Porta Analógica C12
if (c == 12){
    digitalWrite(S0,LOW);
    digitalWrite(S1,LOW);
    digitalWrite(S2,HIGH);
    digitalWrite(S3,HIGH);
    Serial.print("Sensor 13 ");Serial.println(analogRead(pinA0));
    datasource = "c12";
}
//Porta Analógica C13
if (c == 13){
    digitalWrite(S0,HIGH);
    digitalWrite(S1,LOW);
    digitalWrite(S2,HIGH);
    digitalWrite(S3,HIGH);
    Serial.print("Sensor 14 ");Serial.println(analogRead(pinA0));
    datasource = "c13";
}

//Porta Analógica C14 Sensor de Fumaça
if (c == 14){
    digitalWrite(S0,LOW);
    digitalWrite(S1,HIGH);
    digitalWrite(S2,HIGH);
    digitalWrite(S3,HIGH);

    if(analogRead(pinA0) > 420)
    {
        valor = analogRead(pinA0);
        datasource = "c14";
    }
    else
    {
        c++;
    }

    Serial.print("Sensor 15 - Sensor de Fumaça: ");Serial.println(analogRead(pinA0));

}

//Porta Analógica C15 Sensor de Temperatura
if (c == 15){
    digitalWrite(S0,HIGH);
    digitalWrite(S1,HIGH);
    digitalWrite(S2,HIGH);
    digitalWrite(S3,HIGH);
    Serial.print("Sensor 16: ");Serial.println(pinA0);
    datasource = "c15";
}

//ENTRADAS DIGITAIS

if(c == 16){
    //Porta digital D5 pin 14

    //LEITURA DA TEMPERATURA
    // A leitura da temperatura e umidade pode levar 250ms!
    // O atraso do sensor pode chegar a 2 segundos.
    //float h = dht.readHumidity();
    float t = dht.readTemperature();

```

```

    valor = t;
    // testa se retorno é valido, caso contrário algo está errado.
    if (isnan(valor))// || isnan(h))
    {
        Serial.println("Failed to read from DHT");
    }
    else
    {
        //Serial.print("Umidade: ");
        //Serial.print(valor);
        //Serial.print(" %t");

        Serial.print("Sensor 17 - Temperatura: ");
        Serial.print(valor);
        Serial.println(" *C");
    }

    datasource = "c16";
}

c++; //Variável de contagem

String valorString = String(valor); //Converte a Variável para string

// Verifica a conexão com wifi
if ((WiFi.status() == WL_CONNECTED)) {

    WiFiClient client;
    HTTPClient http;

    // Serial.print("[HTTP] begin...\n");
    // configure traged server and url

    // Envia o valor do Sensor para o Servidor ScadaBR
    http.begin(client, "http://" SERVER_IP "/ScadaBR/https?" + datasource + "=" + valorString); //HTTP
    http.addHeader("Content-Type", "application/json");

    //Serial.print("[HTTP] POST...\n");
    // start connection and send HTTP header and body
    Serial.println("http://" SERVER_IP "/ScadaBR/https?" + datasource + "=" + valorString);

    int httpCode = http.GET();
    // httpCode will be negative on error
    if (httpCode > 0) {
        // HTTP header has been send and Server response header has been handled
        //Serial.printf("[HTTP] POST... code: %d\n", httpCode);

        // file found at server
        if (httpCode == HTTP_CODE_OK) {
            const String& payload = http.getString();
            //Serial.println("received payload:\n<<");
            //Serial.println(payload);
            //Serial.println(">>");
        }
        else {
            Serial.printf("[HTTP] .. failed, error: %s\n", http.errorToString(httpCode).c_str());
        }



        http.end();
    }

    delay(1000); //configuração o tempo para envio de cada envio de requisição.

```

}

## DataSource

 **Propriedades do receptor HTTP** 


Nome

Transmissor

Export ID (XID)

DS\_356346

Lista branca de IP remotos



\*


\*

\*


\*

\*


Lista branca de ID de dispositovo



\*

 **Níveis de alarme de eventos**

Não existem eventos para este tipo de data source

**Data points** 










Nome	Tipo de dado	Status	Parâmetro	
Fumaça	Numérico		c14	
Temperatura	Numérico		c16	
Tensão 1	Numérico		c0	
Tensão 2	Numérico		c1	
Tensão 3	Numérico		c2	

Figura 9 - DataSource - Casa do Transmissor

Configurar o Parâmetro com o mesmo nome que está no código do esp8266.