



INSTITUTO FEDERAL DO ESPÍRITO SANTO – CAMPUS SERRA
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

JOABE RUELLA DA SILVA

PEDRO HENRIQUE AGRIMPIO COUTINHO

COMUNICAÇÃO DE DADOS
PROJETO FINAL

SERRA

2018

SUMÁRIO

1. SIMULAÇÕES E ANÁLISES DA CAMADA FÍSICA.....	3
1.1. Principais protocolos e tecnologias que utilizam RS-485.....	3
1.2. Sinal digital do padrão RS-485.....	4
1.3. Problema do eco.....	6
1.4. Interferências causadas por motores de indução.....	8
2. COMUNICAÇÃO MESTRE ESCRAVO VIA PADRÃO RS485.....	12
2.1. O protocolo.....	12
2.2. Custo do projeto.....	14
2.3. Diagrama de interligação.....	14
3. COMUNICAÇÃO VIA WIFI.....	16
3.1. NodeMcu.....	16
3.2. CloudMQTT.....	19
3.3. Protocolo comunicação CloudMQTT x NodeMcU.....	20
4. CÓDIGOS MESTRE E ESCRAVOS.....	22
5. CONCLUSÃO.....	22

1. SIMULAÇÕES E ANÁLISES DA CAMADA FÍSICA

1.1. Principais protocolos e tecnologias que utilizam RS-485

O padrão RS-485 especifica detalhes físicos da rede comunicação, como o número de dispositivos, níveis de tensão de operação e distância máxima. Suas especificações de camada física são utilizadas por diversos protocolos, tais como: Profibus, Modbus, CompoNet e EnDat.

O RS-485 é o padrão mais utilizado no protocolo Profibus DP (Decentralized Peripherals). O Profibus é utilizado em grande escala nas indústrias químicas, petroquímicas, siderúrgicas, mineração e em outros setores. Este protocolo tem como principais características a alta velocidade de comunicação e baixo tempo de resposta. Está presente em equipamentos como CLP's e remotas de sinais analógicos e digitais.

Permite adicionar e remover dispositivos da rede sem influências em dispositivos que já estão em operação, utilizando a topologia do tipo barramento. Outras vantagens estão na transmissão assíncrona NRZ, taxas de comunicação selecionáveis entre 9.6 Kbits/s a 12Mbit/s e cabo par trançado com blindagem.

As limitações do RS-485 ao Profibus estão relacionadas ao elevado custo de implementação do projeto e taxa de transmissão para longas distâncias. Acima de 400 metros, a taxa de transmissão do RS-485 é muito baixa, o que resulta na necessidade de repetidores. Além disso, o RS-485 considera cada dispositivo da rede como uma "unidade de carga". Sendo assim, há uma limitação no número de dispositivos que podem ser conectados, sendo o máximo de 32.

1.2. Sinal digital do padrão RS-485.

Antes de reproduzir o sinal digital do RS-485, é necessário compreender os níveis de tensão do padrão, tanto de transmissão quanto de recepção. O padrão RS-485 especifica um sinal diferencial de tensão entre um par de fios, onde o transmissor oferece uma tensão de no mínimo 1.5v/-1.5v e o receptor deve possuir uma sensibilidade de no mínimo 200mv/-200mv.

A diferença do sinal é determinada de forma que um dos fios transmite o sinal negativo, enquanto o outro fio transmite o sinal positivo. Essa separação permite a eliminação de ruídos admitidos ao longo da transmissão de dados.

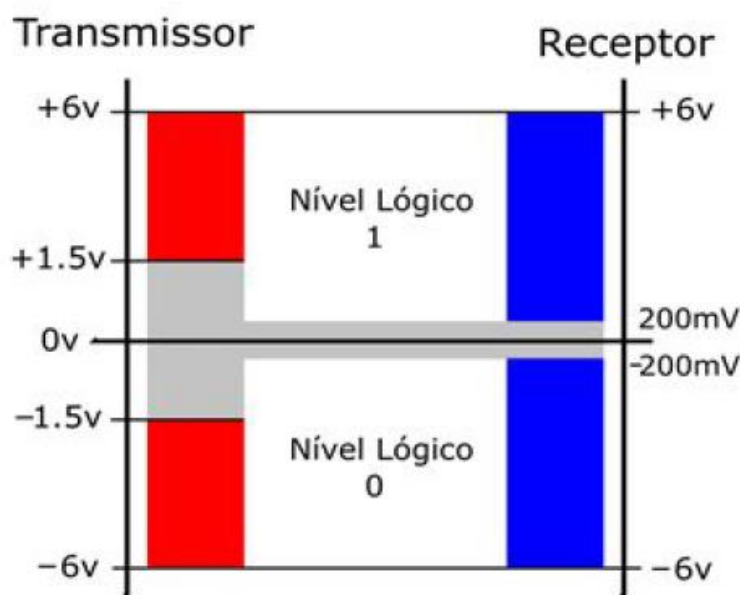


Figura 1 – Níveis lógicos do transmissor e do receptor.

A Figura acima mostra os níveis lógicos do transmissor e do receptor, mostrando os valores mínimos de tensão exigidos pelo padrão para correto funcionamento.

O próximo passo é gerar o sinal periódico do padrão RS-485, entretanto na prática é impossível dispor de infinitas frequências para formar uma onda quadrada perfeita, pois existe um limite superior para as frequências das senóides que serão utilizadas no somatório, gerando assim uma onda entendida pelo receptor como quadrada.

A amplitude do sinal diferencial escolhida foi de 5V e a frequência do sinal foi de 10MHz, como mostra o gráfico abaixo.

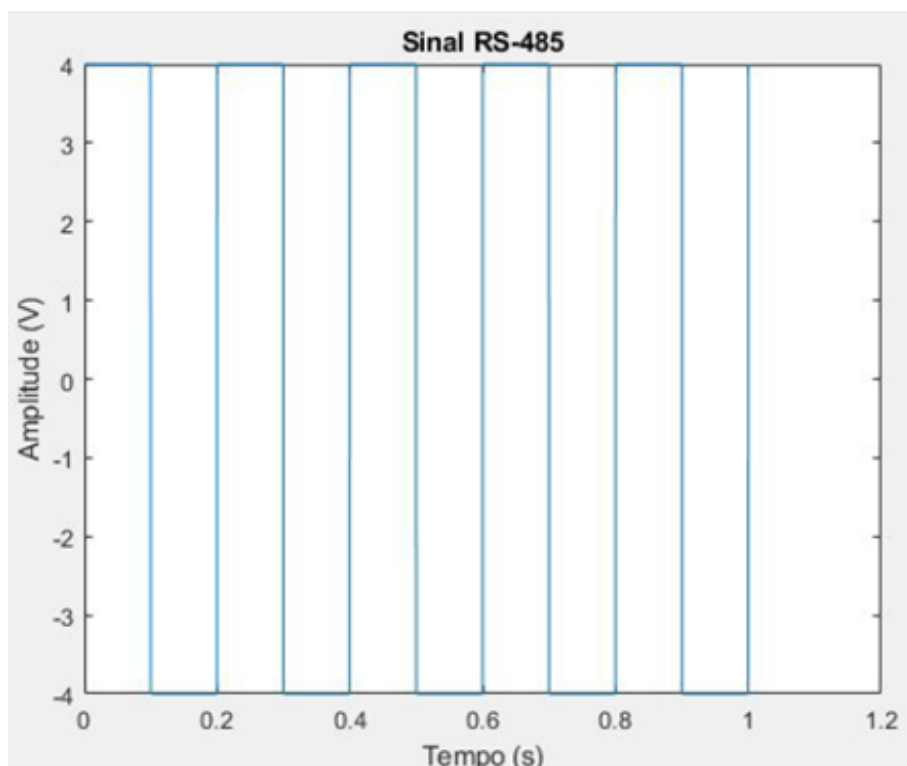


Figura 2 – Sinal Digital periódico

Logo após foi plotado a largura de banda necessária para gerá-lo no domínio da frequência.

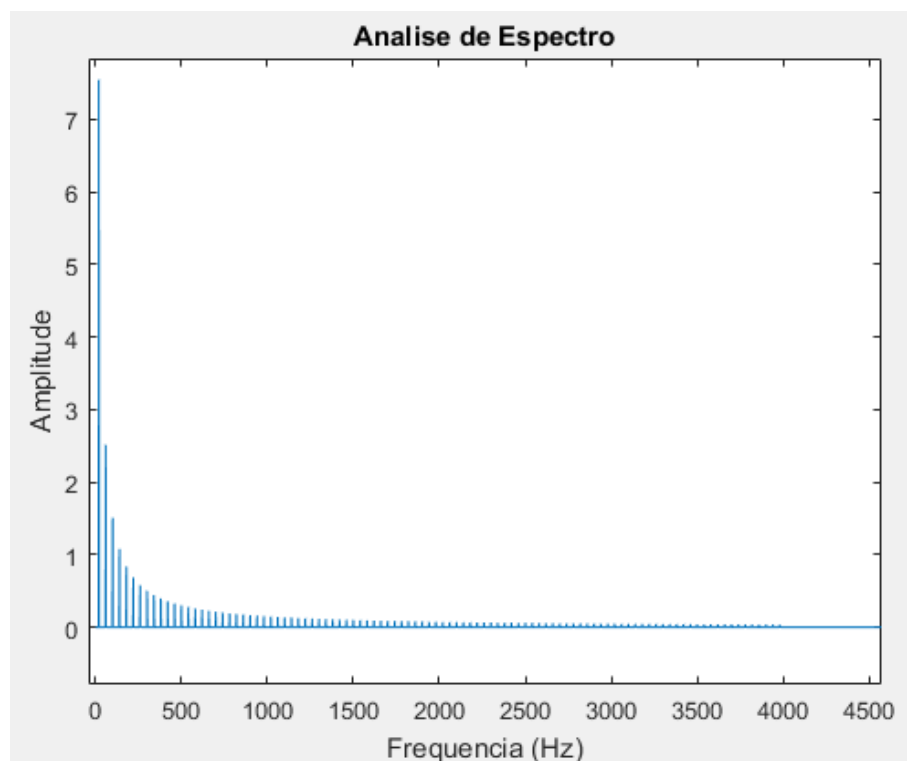


Figura 3 – Largura de banda

Agora é possível realizar a geração do sinal de forma “analógica” utilizando soma de senos, o número de harmônicos escolhido para deixar o somatório de ondas o mais próximo de uma onda quadrada sem exigir muito processamento do computador, foi o de 200 harmônicos.

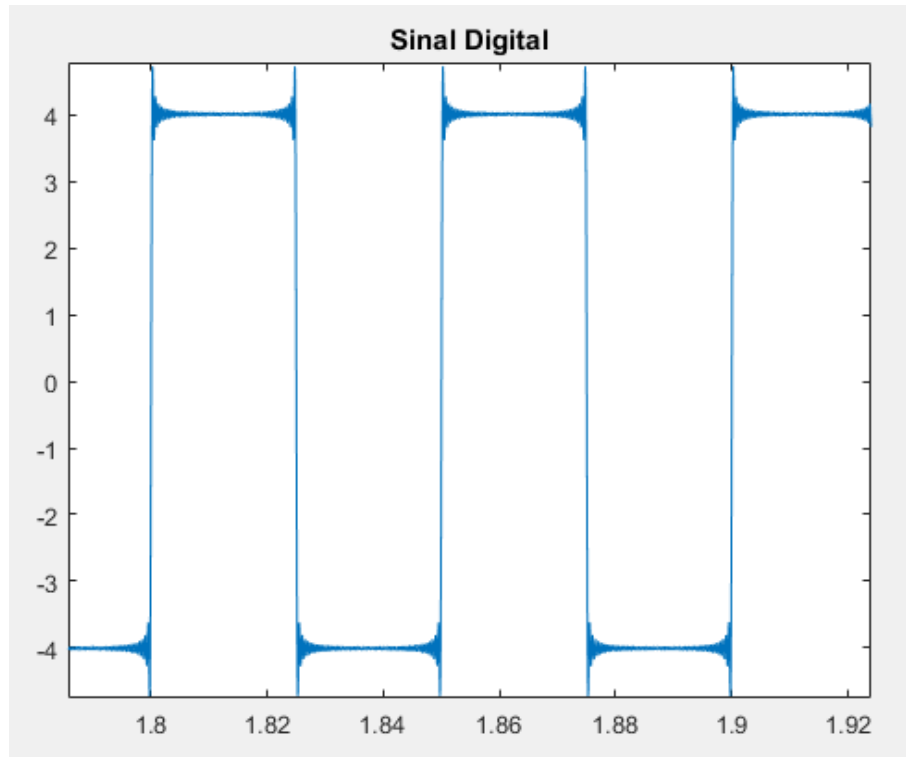


Figura 4 – Reconstrução do sinal RS-485 para o meio físico.

1.3. Problema do eco

O sinal digital e o sinal de eco sofrem atenuação ao longo do caminho de transmissão, natural do uso de conectores. A possibilidade de um ruído de eco interferir com o sinal real é alto somente para caso onde o ponto de eco esteja muito próximo da conexão do receptor. Com o aumento dessas distâncias, a própria atenuação do eco no meio de transmissão faz com que ele não seja capaz de afetar de maneira crítica o sinal. O eco precisa acontecer muito próximo da estação para ser capaz de alterar a correta recepção do sinal.

A fim de garantir que o sinal lido pelo receptor (Sinal RS485 + ECO) não deve ultrapassar o limite de $\pm 6V$ para que assim não prejudique a transmissão dados, foram geradas diferentes formas de ondas com diferentes amplitudes para verificar a qualidade do sinal enviado.

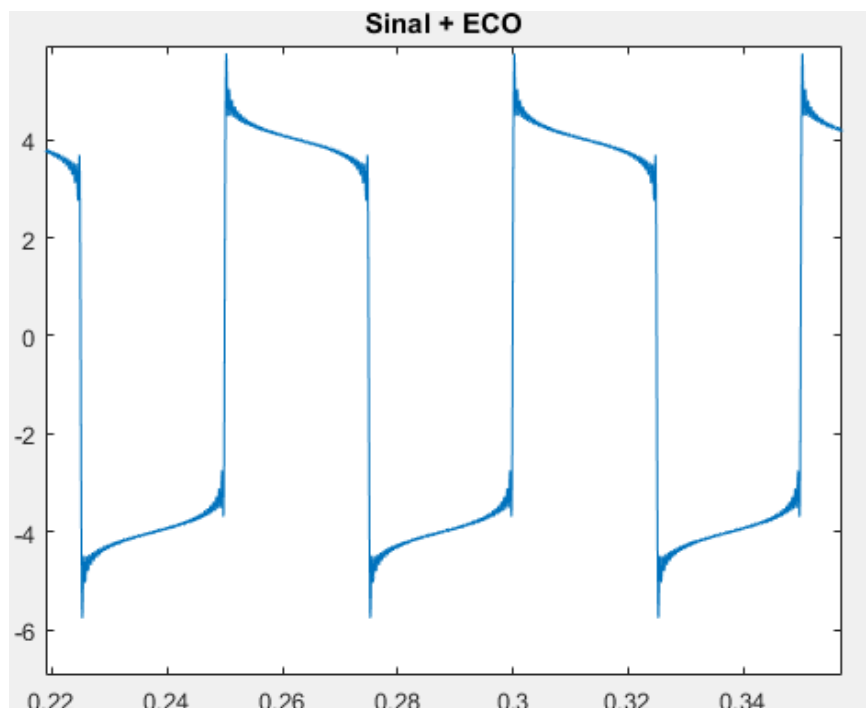


Figura 5 – Sinal + ECO com amplitude 0.5V

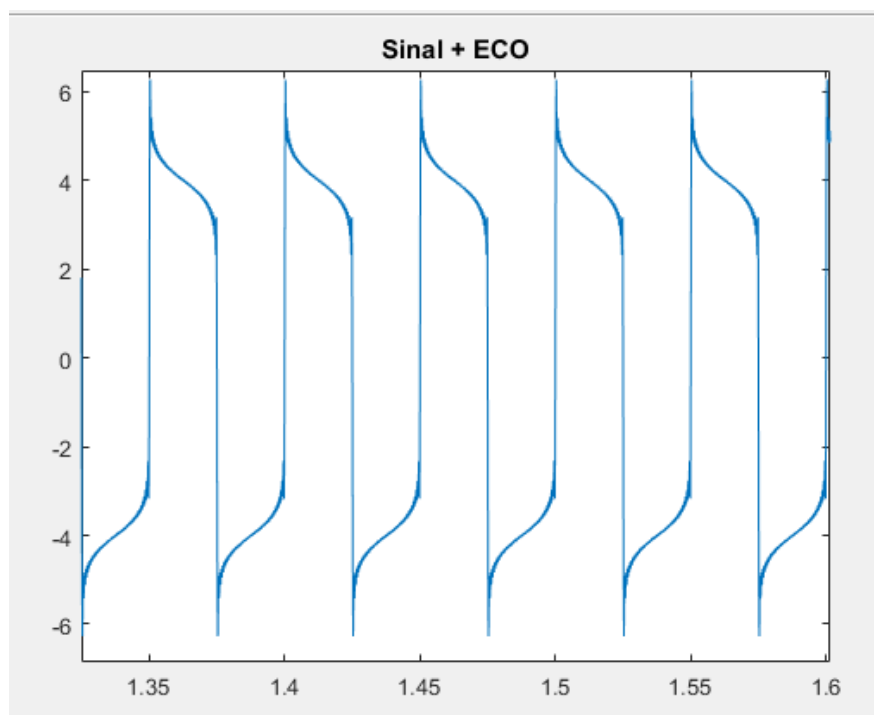


Figura 6 – Sinal + ECO com amplitude 0.75V

Analisando as figuras 4 e 5 percebemos que mesmo com a soma do eco, com amplitudes de 0.5V e 0.75V respectivamente, ao sinal digital, isso não causou erro na leitura do receptor. Agora aumentando a amplitude para 1V, temos.

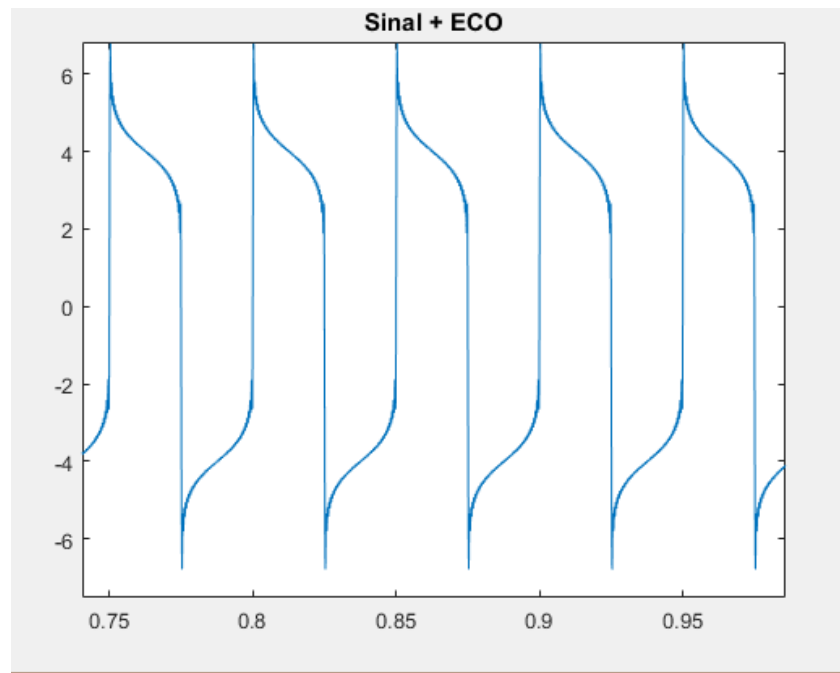


Figura 7 – Sinal + ECO com amplitude 1V

Nota-se que o sinal foi prejudicado, visto que ultrapassou os $\pm 6V$. Portanto conclui-se que a amplitude do eco não pode passar de 0.75V, senão irá causar erro na transmissão dos dados.

1.4. Interferências causadas por motores de indução

A relação sinal/ruído, ou SNR, é dado por $(\text{sinal}/\text{ruído})^2$ e representa a relação entre um sinal e o ruído presente. Para que o sistema de transmissão de dados não apresente problemas com ruídos, esta relação deve possuir um valor muito alto, pois isso representa que a amplitude do sinal é muito mais do que a amplitude do ruído.

Para comprovar essa relação foi gerado interferências eletromagnéticas provocadas pela proximidade de motores de indução e conjuntos de motores com inversores, considerando sinais senoidais de 60Hz e harmônicos até o 7º. Na figura abaixo é apresentado o formato de onda gerado por esses motores.

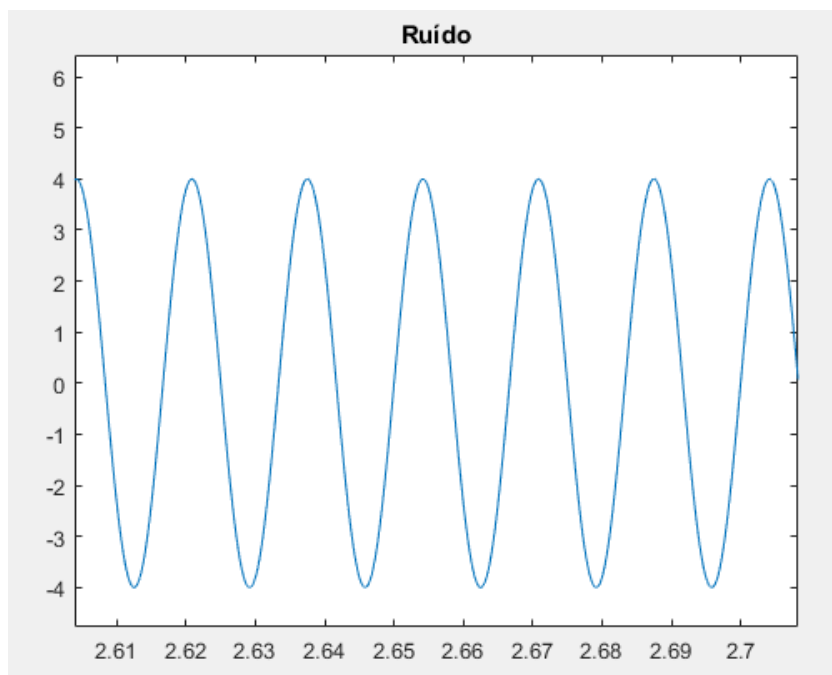


Figura 7 – Sinal de ruído eletromagnético.

Esses ruídos foram somados com o sinal digital do padrão RS-485, afim de verificar se haveria perda na transmissão de dados, provocados pelas interferências magnéticas geradas na operação dos motores de indução. Primeiramente foram somados com ruídos no 1º e 3º harmônico.

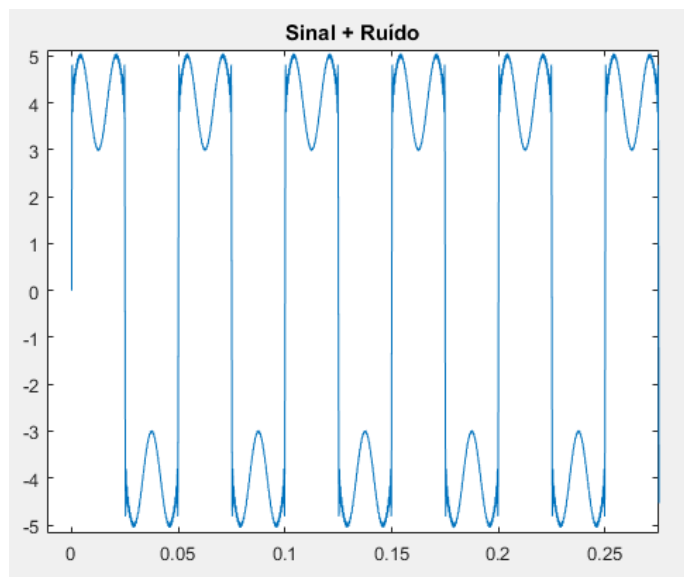


Figura 8 – Sinal digital + ruído eletromagnético no 1^o harmônico.

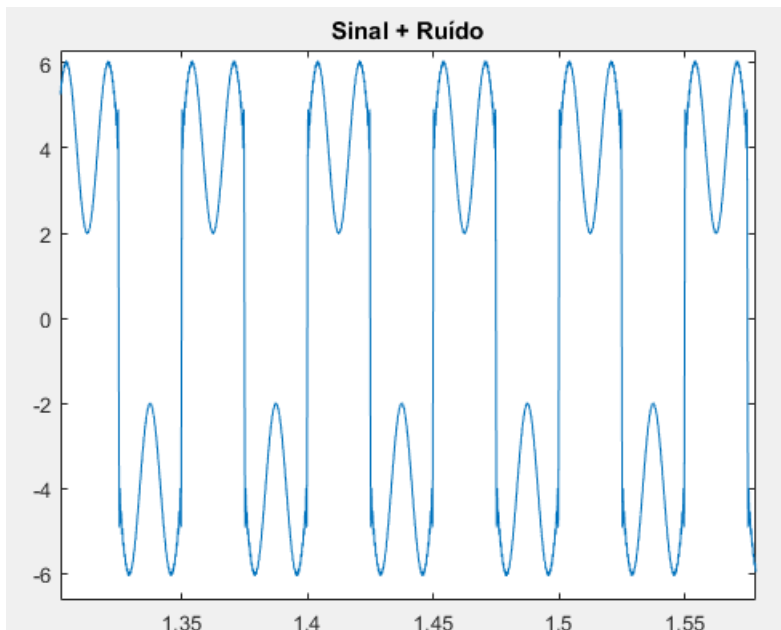


Figura 9 – Sinal digital + ruído eletromagnético no 3^o harmônico.

Nota-se que mesmo com a inserção do ruído eletromagnético o sinal digital não ultrapassou os $\pm 6V$, não prejudicando assim a transmissão dos dados. Agora foram somados ruídos no 5^o e 7^o harmônico.

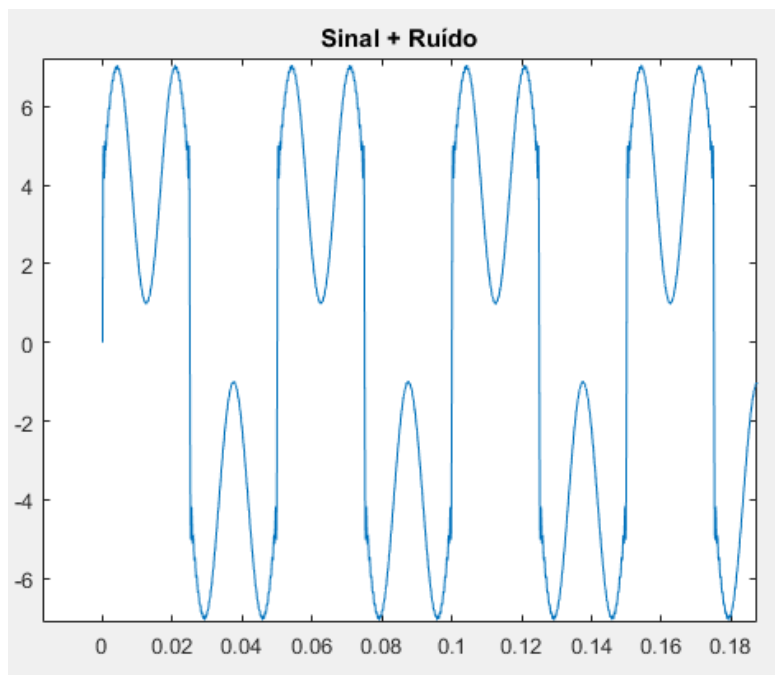


Figura 9 – Sinal digital + ruído eletromagnético no 5^o harmônico.

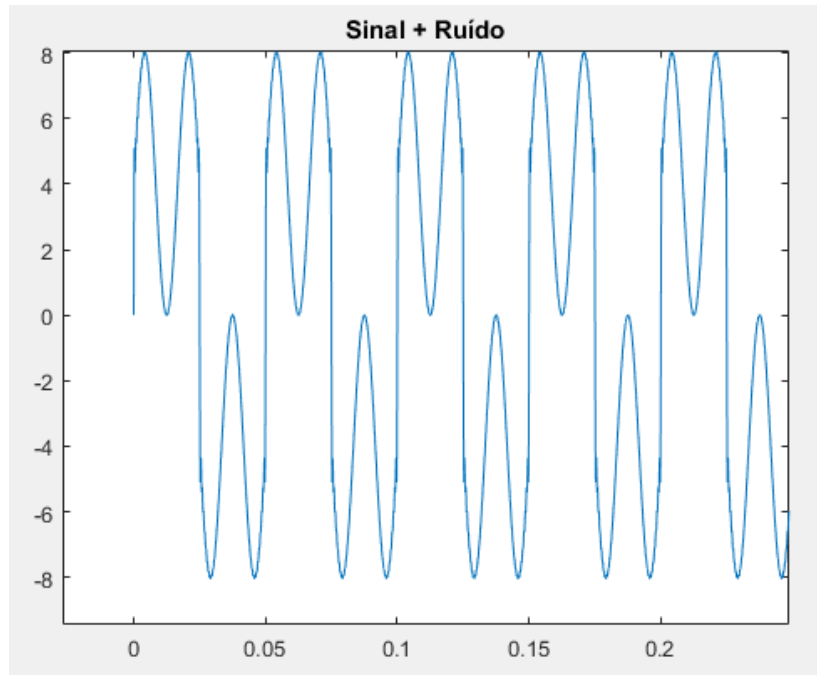


Figura 10 – Sinal digital + ruído eletromagnético no 7^o harmônico.

Fica evidente que a partir do 5º harmônico o sinal digital do padrão RS-485 fica comprometido, pois ele ultrapassa os limites de tensão especificados, +/-6V. Acarretando erros de transmissão de dados, inviabilizando a comunicação.

2. COMUNICAÇÃO MESTRE ESCRAVO VIA PADRÃO RS485

A segunda etapa do projeto consiste em desenvolver a comunicação entre um mestre e dois escravos utilizando o barramento do padrão RS-485. O mestre foi conectado via cabo USB ao PC, logo, pelo computador são enviados os comandos e se recebe as informações provenientes dos escravos obedecendo o protocolo desenvolvido.

2.1. O protocolo

O protocolo desenvolvido possui um formato projetado especificamente para esta aplicação, onde um frame de cinco bytes foi proposto e possui o seguinte formato:

Endereço	Comando	Registrador	Dados
1 Byte	1 Byte	1 Byte	2 Bytes

No byte de Endereço é definido qual dispositivo deve receber a mensagem. Em Comando é estabelecido se é leitura ou escrita. O byte Registrador remete ao endereço da informação desejada e os bytes de Dados são onde se armazena as informações para que sejam interpretadas pelo dispositivo que se destina a mensagem. Segue abaixo a relação completa dos símbolos utilizados no protocolo.

	Mestre 1	Escravo 1	Escravo 2
Endereço	M	1	2
Comando		R (read) W (write) A (answer)	R (read) W (write) A (answer)
Registrador		T (Temperatura) U (Set Point) L (LED) M (Status do LED) S (Status do sistema)	T (Temperatura) U (Set Point) L (LED) M (Status do LED) S (Status do sistema)

Dados		Temperatura: 0 - 99 graus Set Point: 0 - 99 graus LED: 1 = ON, 0 = OFF Status do sistema: 0 = running, 1 = Set point atingido	Temperatura: 0 - 99 graus Set Point: 0 - 99 graus LED: 1 = ON, 0 = OFF Status do sistema: 0 = running, 1 = Set point atingido
-------	--	---	---

O Comando Z reseta a comunicação UART em todo barramento, zerando os contadores de bytes recebidos.

Na Figura 11 é possível ver um exemplo prático do frame sendo enviado a um escravo. Trata-se do frame 2WL01, onde, W é o Comando de escrita e L é o Registrador de um LED do dispositivo que está registrado com Endereço igual a 2. Como os dados estão definidos como 01, então, isto significa “ligar” o LED do escravo 2.

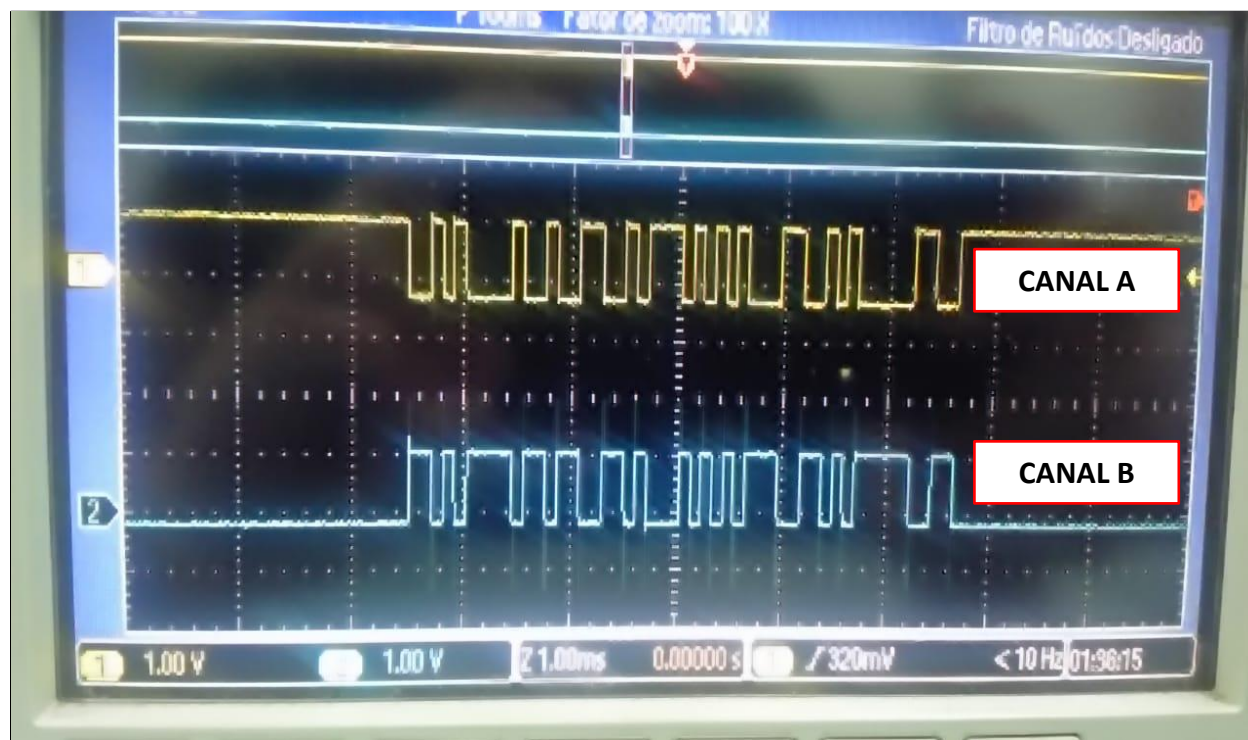


Figura 11 – Sinal real referente à palavra 2WL01.

Fica evidente a comunicação entre mestre e escravo através da imagem acima, visto que a amplitude e qualidade do sinal estão de acordo com o padrão RS-485.

2.2. Custo do projeto

O custo aproximado dos componentes do projeto pode ser visto na tabela abaixo:

Produto	Quant.	Preço un.	Valor
Arduino MEGA	2	60,00	120,00
NodeMCU	1	30,00	30,00
Cabos	30	0,50	15,00
Conversor 3.3-5V	1	10,00	10,00
Módulo MAX485	3	12,00	36,00
Placas Protoboard	6	30,00	180,00
		TOTAL	391,00

Outros componentes não entraram na lista, pois foram utilizados por empréstimos, tais como: fonte de alimentação regulada, osciloscópio, computadores, sensor de temperatura, e outros.

2.3. Diagrama de interligação

A estrutura principal do barramento de comunicação foi desenvolvida seguindo o projeto mostrado na Figura 12.

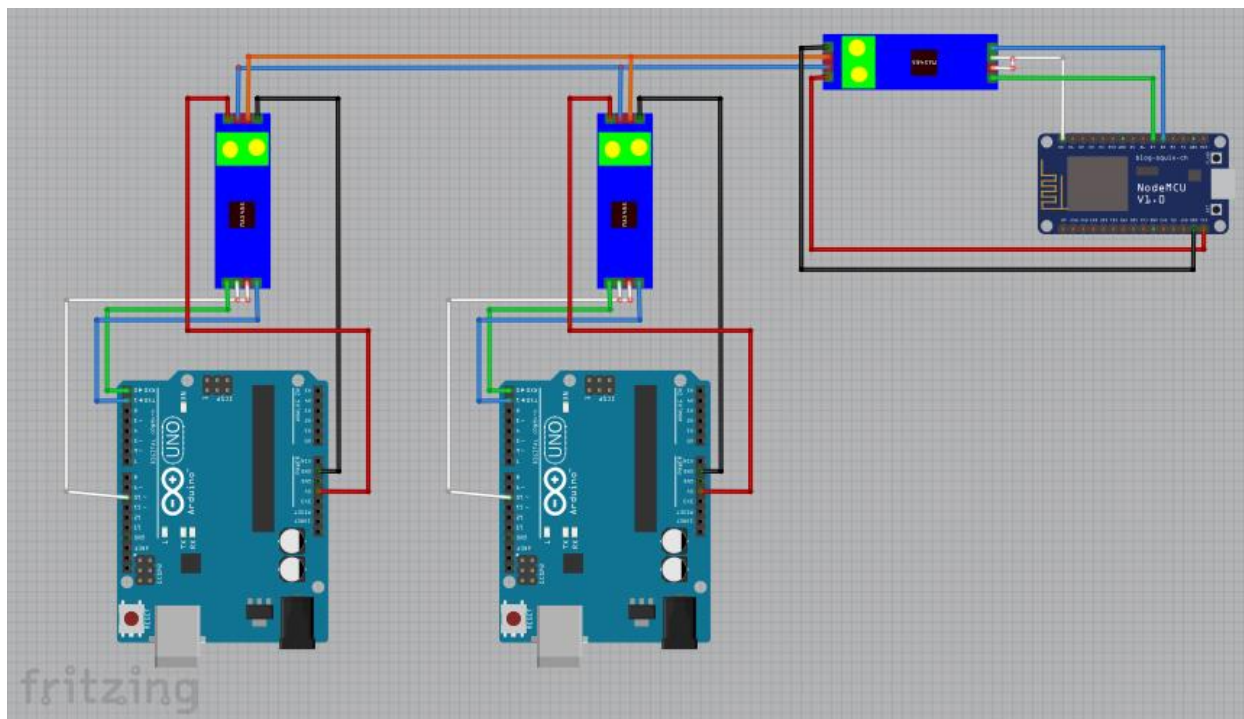


Figura 12 – Estrutura projetada da instalação do barramento RS485.

Uma imagem real do que foi desenvolvido pode ser visto na Figura 13.

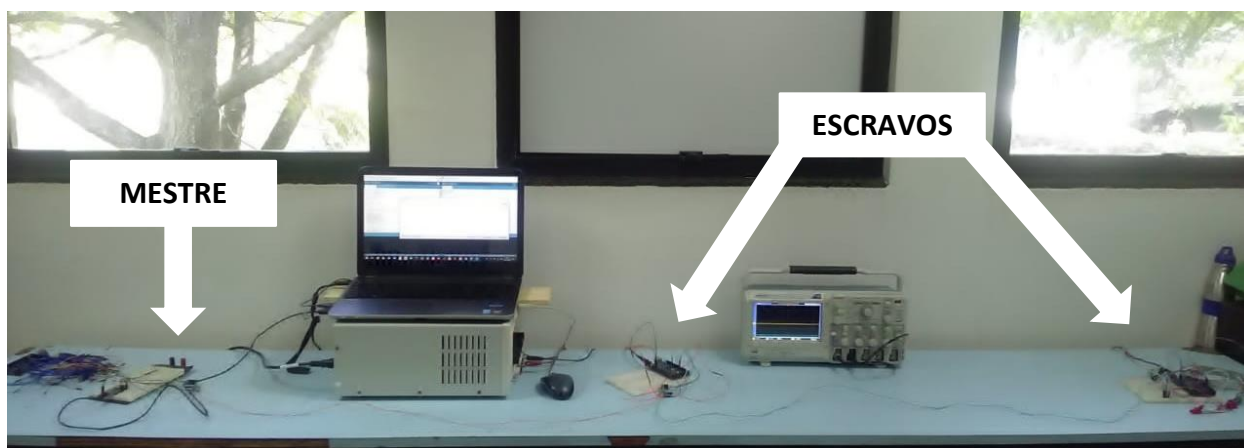


Figura 13 – Estrutura real da instalação do barramento RS485.

Durante os testes práticos, foram detectados ruídos ao longo do barramento, então, com a finalidade de tentar elimina-los, um capacitor cerâmico de 100nF foi adicionado entre o canal A e o GND e outro entre o canal B e o GND. Os resultados significativos da adição dos capacitores pode ser visto na Figura 14.

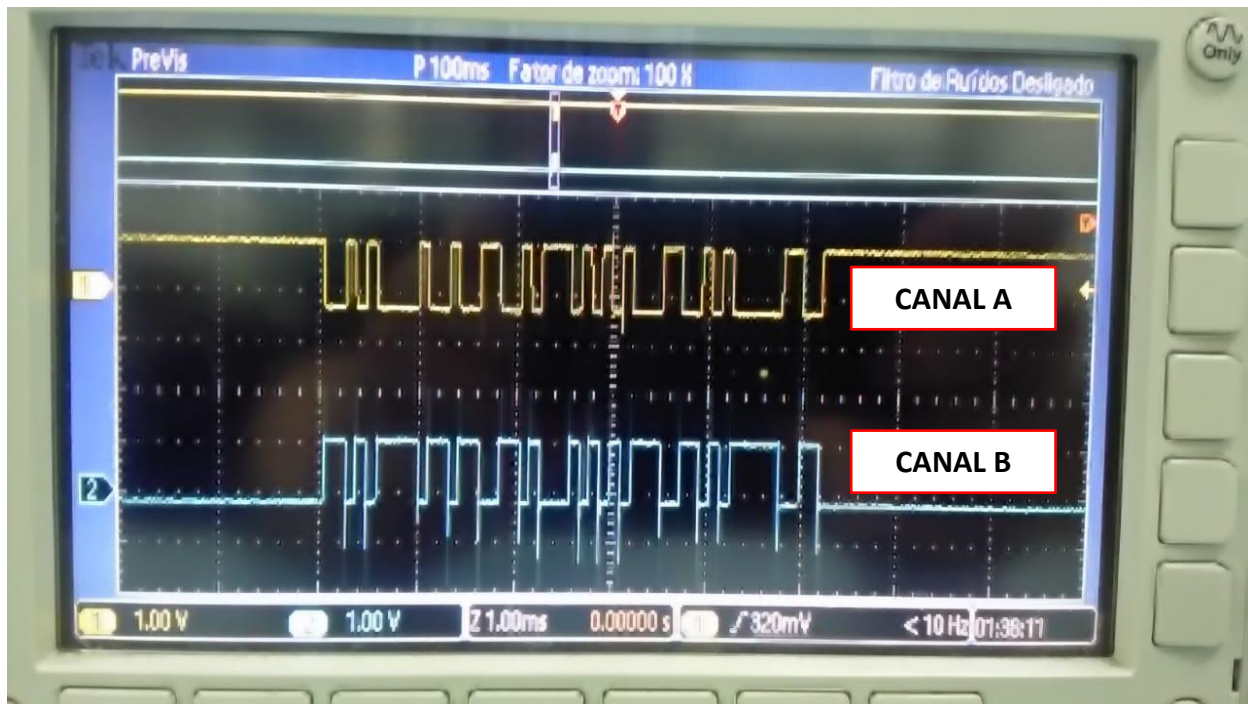


Figura 14 – Sinal do barramento com a adição de capacitor (Canal A) e sem capacitor (Canal B).

3. COMUNICAÇÃO VIA WIFI

A terceira etapa do projeto consisti no desenvolvimento de uma comunicação com o mestre via rede WiFi para realizar o controle e as leituras através de uma interface web.

3.1. NodeMCU

O NodeMCU (Figura 15) é uma placa de prototipagem que possui o chip ESP8266, que é um módulo de WiFi que pode ser configurado para se conectar à Internet para projetos de Internet das coisas (IoT) e projetos similares. O NodeMCU pode ser programado diretamente através da porta USB usando a IDE do Arduino.

Através de algumas linhas programação pode-se estabelecer uma conexão WiFi e definir pinos de entrada e saída de acordo com a necessidade do projeto (como no Arduino), transformando o NodeMCU em um servidor web. Ele combina os recursos do ponto de acesso WiFi e microcontrolador. Esses recursos tornam o NodeMCU uma ferramenta extremamente poderosa para projetos utilizando redes WiFi de baixo custo.

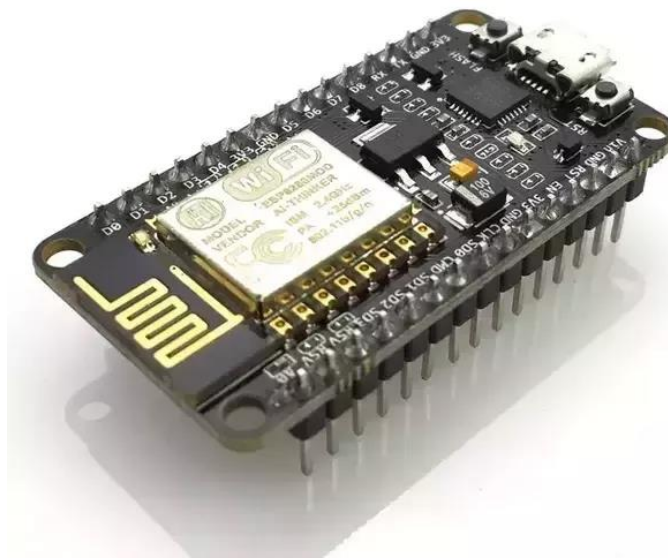
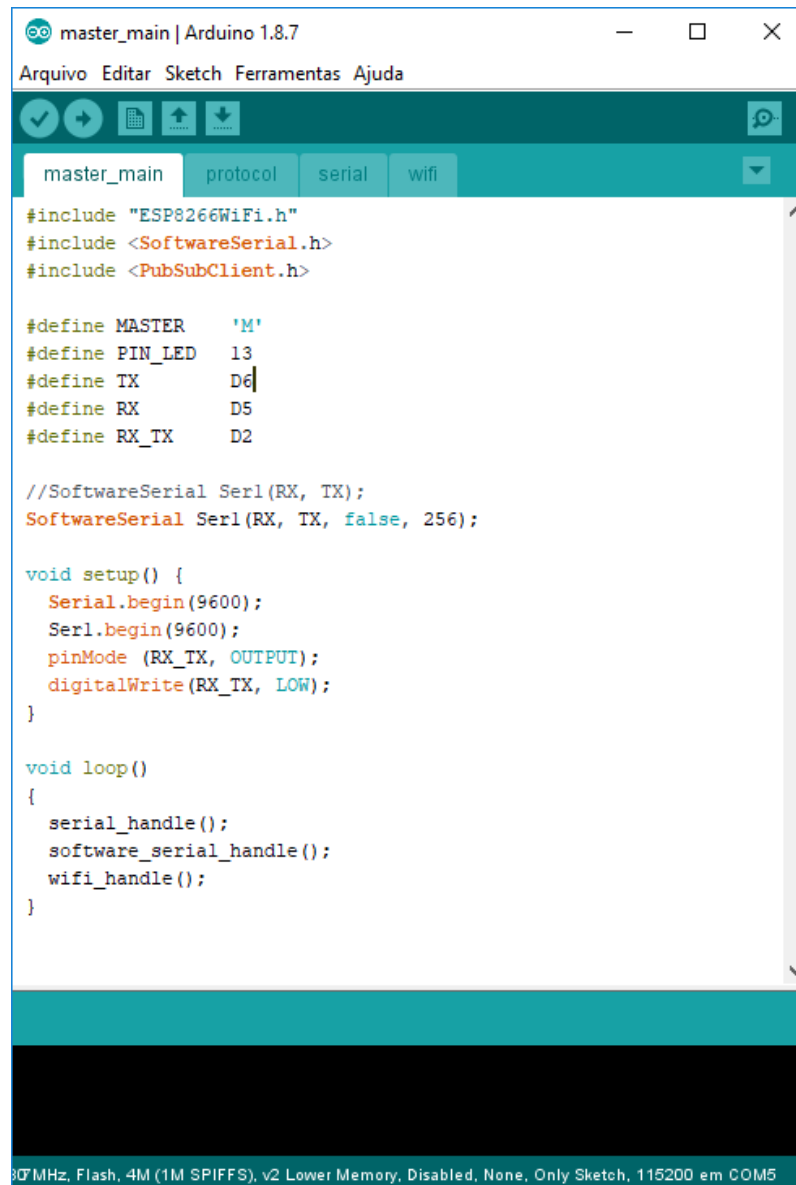


Figura 15 – Imagem ilustrativa do NodeMCU.

Uma imagem da IDE do Arduino que foi utilizada para desenvolver os códigos que rege o dispositivo mestre pode ser vista na Figura 16.



The image shows the Arduino IDE interface with the file 'master_main' open. The code is written in C++ and includes headers for ESP8266WiFi, SoftwareSerial, and PubSubClient. It defines constants for MASTER, PIN_LED, TX, RX, and RX_TX. The setup function initializes the serial port at 9600 baud, configures the TX pin as an output, and sets it to LOW. The loop function calls serial_handle, software_serial_handle, and wifi_handle.

```
master_main | Arduino 1.8.7
Arquivo  Editar  Sketch  Ferramentas  Ajuda

master_main  protocol  serial  wifi

#include "ESP8266WiFi.h"
#include <SoftwareSerial.h>
#include <PubSubClient.h>

#define MASTER    'M'
#define PIN_LED   13
#define TX        D6
#define RX        D5
#define RX_TX     D2

//SoftwareSerial Ser1(RX, TX);
SoftwareSerial Ser1(RX, TX, false, 256);

void setup() {
  Serial.begin(9600);
  Ser1.begin(9600);
  pinMode (RX_TX, OUTPUT);
  digitalWrite(RX_TX, LOW);
}

void loop()
{
  serial_handle();
  software_serial_handle();
  wifi_handle();
}
```

160MHz, Flash, 4M (1M SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 em COM5

Figura 16 – IDE do Arduino onde foi desenvolvido o código do dispositivo Mestre.

3.2. CloudMQTT

CloudMQTT é um servidor na nuvem. O servidor implementa o protocolo de transporte de telemetria MQ, MQTT, que fornece métodos leves da transmissão de mensagens usando um modelo de enfileiramento de mensagem de publicação/assinatura.



Figura 17 – Protocolo MQTT.

MQTT é ideal para o mundo da "Internet das coisas" de dispositivos conectados. Seu design minimalista torna-se perfeito para sistemas internos, telemóveis e outros que tem restrição de memória e de largura de banda.

Filas de mensagens fornecem um protocolo de comunicação assíncrona, o remetente e o receptor da mensagem não precisam interagir com a fila de mensagens ao mesmo tempo. As mensagens colocadas na fila são armazenadas até que o destinatário os recupere ou as mensagens atinjam o tempo limite.

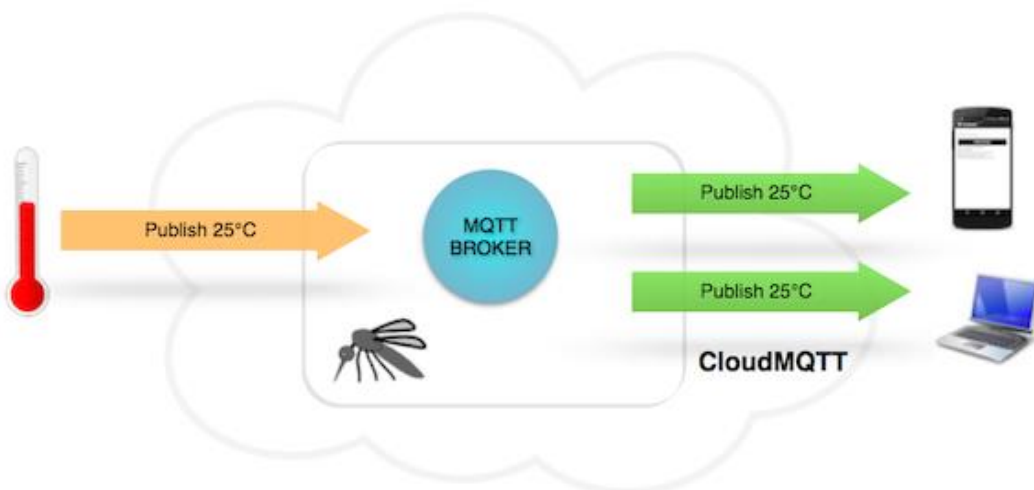


Figura 18 – Publicação e recebimento da mensagem.

3.3. Protocolo comunicação CloudMQTT x NodeMcU

Primeiramente conectamos o NodeMcu a rede wifi, que foi gerada através do 4G do celular. A figura abaixo explicita o nome da rede e senha, ao qual o master foi conectado.

```
//informações da rede WIFI  
const char* ssid = "AndroidAP";  
const char* password = "ownb7731";
```

Figura 19 – Linha do código para conexão wifi.

Para ativar a conexão do master a rede wifi, foi utilizado o comando WWWW1, via serial, que foi definido como padrão para habilitar a conexão wifi.

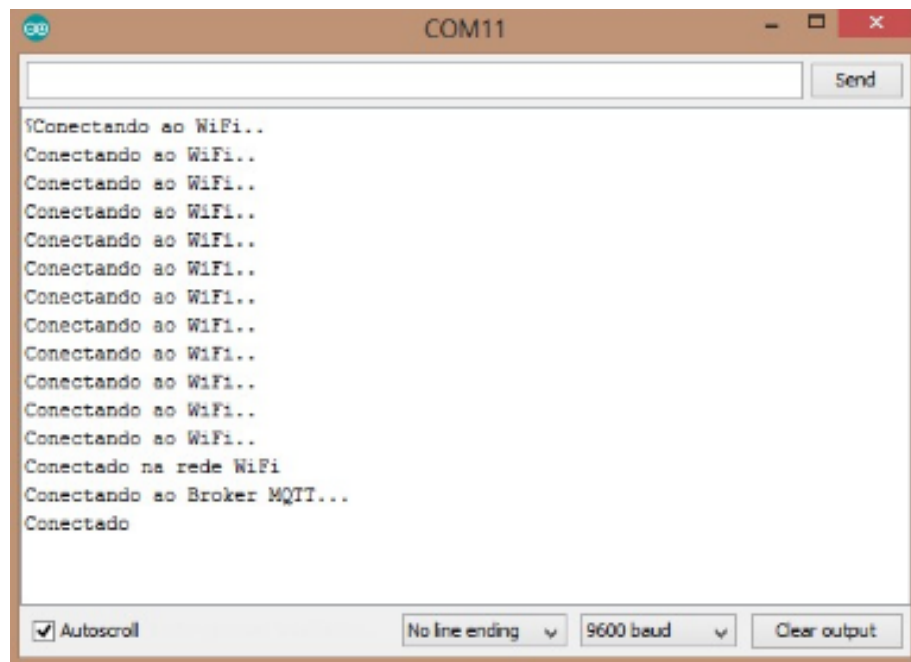


Figura 20 – Estabelecimento de conexão do NodeMcu com servidor.

Conforme pode ser visto na figura acima o master foi conectado com sucesso a rede. O próximo passo foi utilizar a interface com usuário do CloudMQTT para publicar e receber mensagens, que seguem o padrão do protocolo definido anteriormente, nos tópicos utilizados para comunicar com o NodeMcu.

O nome do tópico utilizado para enviar comando ao NodeMcu é “MASTER”. Através dele os comandos são enviados e como o NodeMcu está escutando o canal ele recebe as mensagens e toma as decisões. A figura abaixo ilustra tal procedimento.

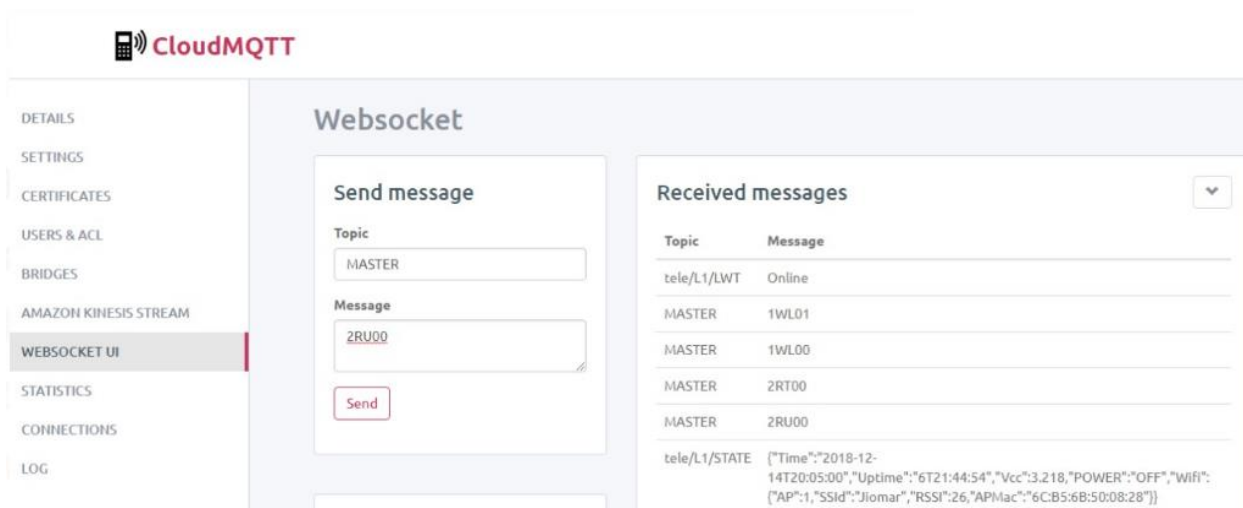


Figura 21 – Interface do usuário do servidor.

Pode observar que foi ligado (1LW01) e desligado (1LW00) o LED do slave 1. Logo após foi lido a temperatura atual (2RT00) e lido o setpoint desejado da temperatura (2RU00) no slave 2. A figura abaixo mostra que a comunicação entre o CloudMQTT e o NodeMcu foi realizada com sucesso.

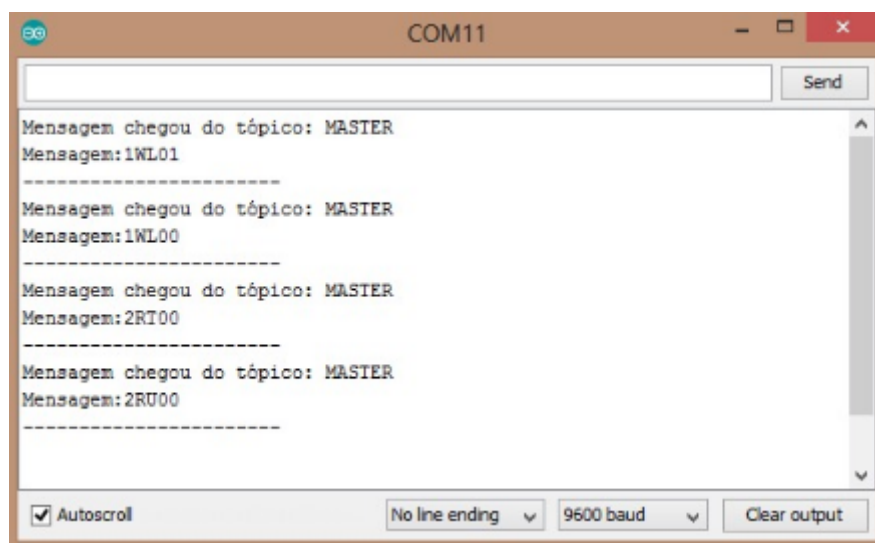


Figura 22 – Mensagens recebidas do servidor no NodeMcu.

Nota-se que após os comandos de leitura não há respostas dos escravos, devido a problemas de interferências no barramento do padrão RS-485. O sinal é somado com ruídos externos, o que resulta em falha de comunicação, mesmo que garantido a entrega da mensagem no escravo e sua resposta.

O comando não chegava ao master ao passar pelo conversor MAX485, em temperatura ambiente a mensagem respondida deveria ser “MAT25”, o que significa que o escravo respondeu a temperatura 25°C medida.

4. CÓDIGOS MESTRE E ESCRAVOS

Os códigos utilizados no projeto estão disponibilizados no repositório do github abaixo:

https://github.com/mojorim/final_project_CD.git

5. CONCLUSÃO

Neste trabalho foi realizado estudos relacionados à comunicação de dados digitais utilizando o padrão RS485. Na primeira etapa foi utilizado MATLAB para simular o sinal RS485 e foram feitas as devidas análises quando há a presença de ecos e ruídos na linha de transmissão do sinal.

Na segunda etapa foi realizado o projeto prático, onde foi verificado os efeitos observados na primeira etapa, e um tratamento simples do sinal utilizando capacitores para filtrar ruídos.

Finalizando, na terceira etapa, utilizando o NodeMcu, foi realizada a comunicação via WiFi do sistema com o servidor CloudMQTT.