Laboratório de instrumentação eletrônica e controle (LIEC)

Relatório de Projeto TESS

Desenvolvimento de um sistema de acionamento de cargas via wifi utilizando microcontrolador

Ricardo Soares Chinarro

ricardo.chinarro@ee.ufcg.edu.br

Campina Grande, Agosto de 2015

Índice de Ilustrações

Sumário

1. Introdução

Neste relatório é descrito as etapas relativas ao desenvolvimento e produção de um módulo o qual aciona cargas industriais e verifica seu estado (ligado / desligado).

- Descrição do funcionamento;
- Desenvolvimento do esquema elétrico dos circuitos de acionamento das cargas e detecção se as cargas estão ligadas ou desligadas;
- Desenvolvimento do layout;
- Desenvolvimento de um web server utilizando os módulos ZG2100MC e MRF24WB0MA
- Desenvolvimento do código fonte utilizando o modulo wifi ESP8266;
- Teste de funcionamento;
- Conclusão.

2. Descrição do funcionamento

O sistema de acionamento de cargas funciona a partir de dados recebidos de um servidor via *wifi* ou quando acontece alguma alteração do estado na carga (Carga sendo ligada ou desliga). Quando umas das ações acontecem no sistema ele responde das seguinte forma:

- Liga ou desliga a carga;
- Lê o estados da carga e envia o estado das cargas ao servidor;
- Quando a carga é ligada ou desligada manualmente o sistema detecta e envia o estado das cargas ao servidor.

No diagrama de blocos a seguir observamos a arquitetura do sistema o qual temos a comunicação entre o microcontrolador com o módulo wifi e a placa de interface para o ligar e desligar as cargas e verificar o estado das mesmas.

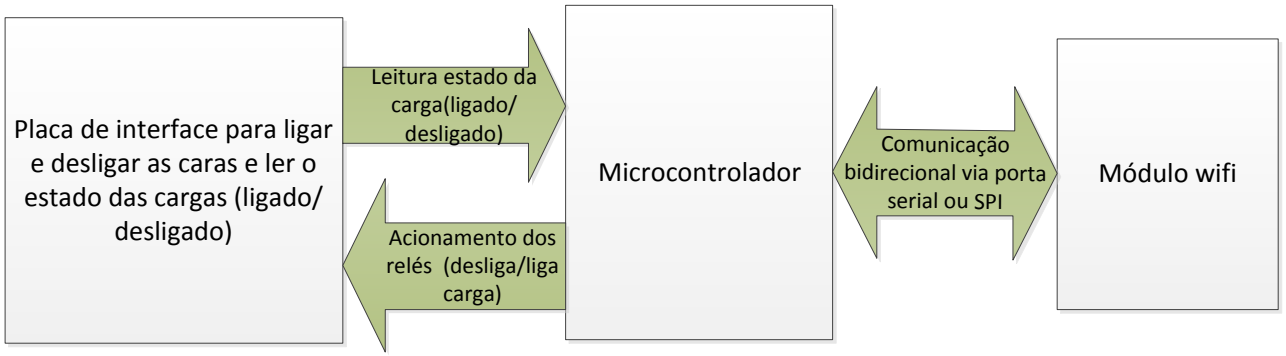

Figura 1: Diagrama de blocos

3. Desenvolvimento do esquema elétrico dos circuitos de acionamento das cargas e detecção se as cargas estão ligadas

Para detectar se a carga estava ligada ou não tivemos que considerar que o sistema de acionamento poderia ser um *three way*. Para isto utilizamos relés para o acionamento *three way* utilizamos o relé como o segundo interruptor *three way* como pode ser visto na Figura 2.

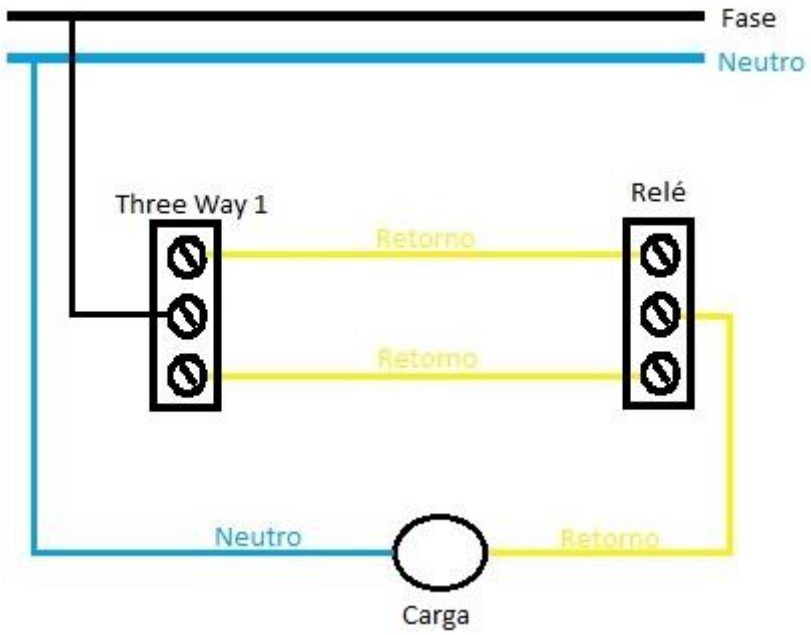

Figura 2: Ligação entre os interruptores

Para detectar se a carga está ligada consideramos que sempre que existir tensão na saída do

relé é porque a carga está ligada. Para isto utilizamos o circuito da Figura 3.

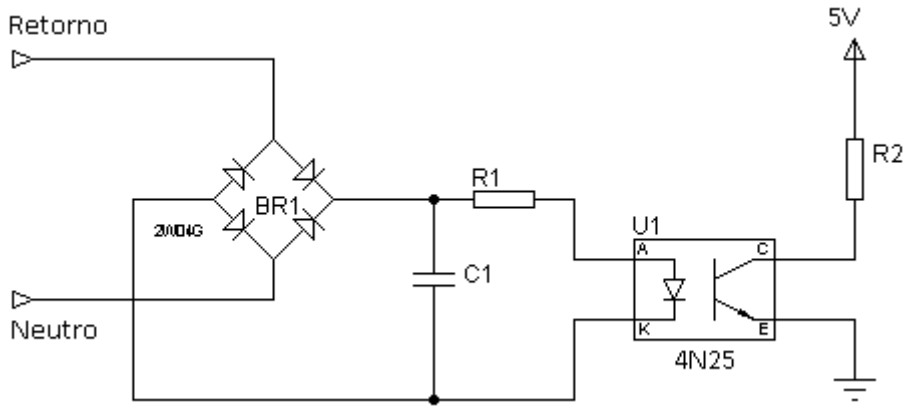

Figura 3: Detector de fase

O circuito final para a detecção de fase e acionamento dos relés pode ser observado na Figura 4.

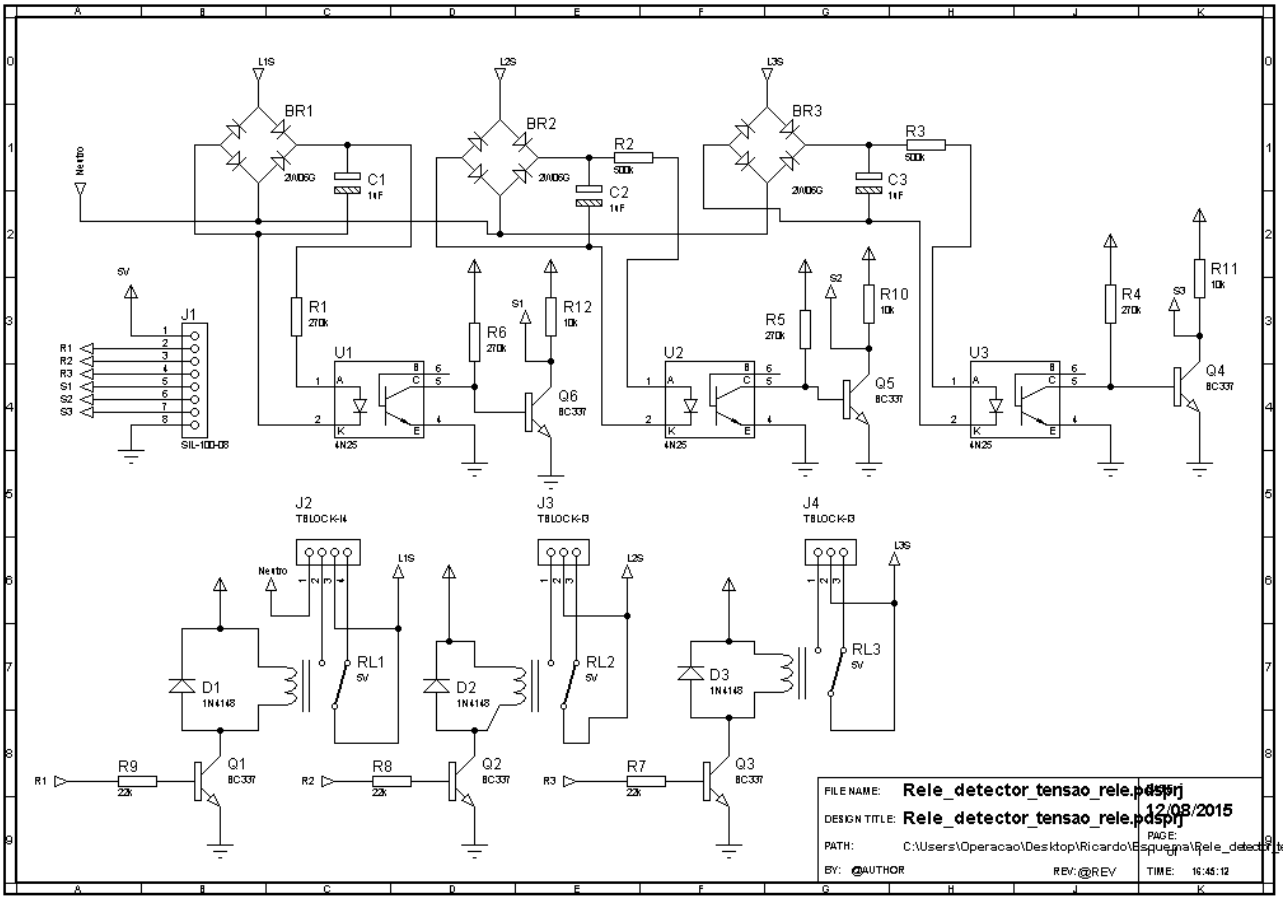

Figura 4: Circuito relés e detecção de fase

4. Desenvolvimento do layout

O desenvolvimento do *layout* da PCI foi feio pelo técnico Simões no software Orcad. O *layout* e o aspecto geral da placa podem ser encontrados na seguinte Figura 5.

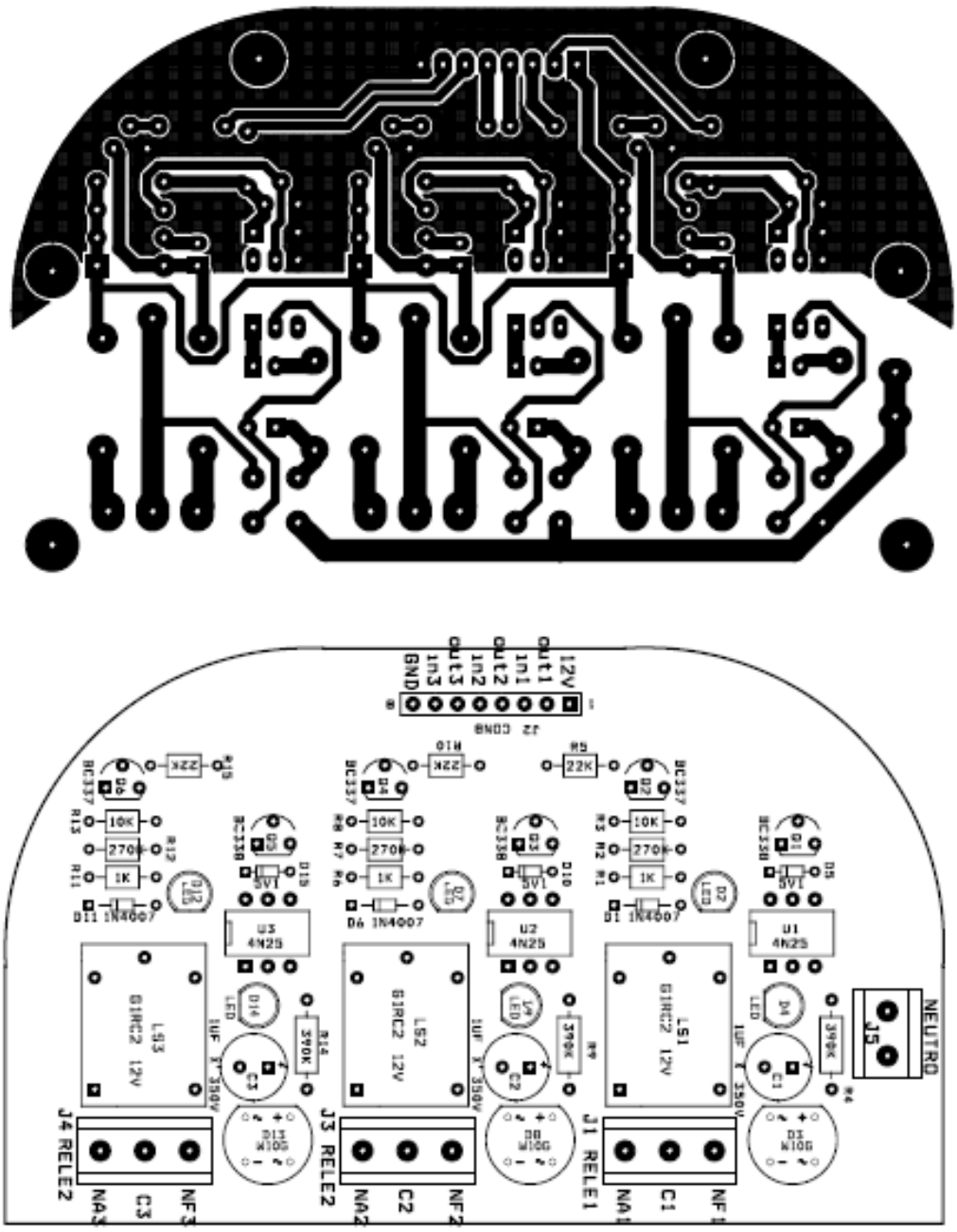

Figura 5: Layout da PCI

5. Desenvolvimento de um web server utilizando os módulos ZG2100MC e MRF24WB0MA

Para testes da placa de detecção de tensão e acionamento de cargas foi desenvolvido um web server com arduino uno, os módulos ZG2100MC e MRF24WB0MA. Para o desenvolvimento do web server foi necessário utilizar uma versão antiga da IDE de programação do arduino para que ouve-se compatibilidade com as bibliotecas para estes módulos. A versão utilizada da IDE foi a 0.23 e encontra disponível para download https://www.arduino.cc/en/Main/OldSoftwareReleases#previous .

No desenvolvimento do web server utilizando os módulos ZG2100MC e MRF24WB0MA encontramos um grande problema que foi por causa que a TCP/IP stack tinha que rodar em um microcontrolador de 8 bits de baixa capacidade de processamento deixando o sistema lento e ineficaz para esta aplicação.

6. Compreensão do módulo ESP8266

Como percebemos que os módulos ZG2100MC e MRF24WB0MA eram ineficazes para tarefas e apresentavam muitos problemas decidimos utilizar o modulo ESP8266 que já tem o TCP/IP stack nele fazendo que utilizássemos apenas comandos AT para configuração e envio de dados desta forma deixando o software mais leve e sem travamentos.

O passo inicial foi aprender a utilizar os comandos AT. Para isso utilizamos o arduino mega por possuir diversas portas seriais e configuramos suas portas para um *baud rate* de 115200. Os comandos AT utilizados foram:

- AT+RST\r\n (Reseta o modulo)
- AT+CWMODE=1\r\n (Configura o módulo como *station mode*)
- AT+CWDHCP=1,1\r\n (Configura o módulo como *station mode* e desabilita o DHCP)
- AT+CIPMUX=1\r\n (Seta o módulo conexões simultâneas)
- AT+CIPSTA="10.0.0.10"\r\n (Seta o endereço de IP do módulo)
- AT+CWJAP="LIEC_WIFI","LIEC_123"\r\n (Conecta à rede wifi)
- AT+CIPSERVER=1,502\r\n (Abre a porta 502 para recepção e envio de dados)
- AT+CIPSTART=1,"TCP","10.0.0.15",502\r\n (Abre uma conexão TCP/IP com o

endereço de IP 10.0.0.15 através da porta 502)

- AT+CIPSEND=1,11\r\n (Prepara o módulo para enviar 11 bytes)
- AT+CIPCLOSE=1\r\n (Após o envio dos 11 bytes fecha a conexão TCP/IP)

A lista de todos os comandos AT existentes estão disponíveis no anexo A.

7. Desenvolvimento do código fonte utilizando o arduino nano e o ESP8266

Para o desenvolvimento do software inicialmente foi definido as portas de entrada, saída e o mapa de memória eeprom estes dados podem ser encontrados no anexo B.

Após definidas as entradas e saídas e o mapa de memória foram desenvolvidos dois códigos para o sistema. Um código é responsável pela execução do envio de dados, recebimento de dados, e monitoramento das cargas. O outro código é responsável pela configuração dos parâmetros de conexão wifi:

- SSID
- Senha
- IP do modulo
- IP do Servidor

Sempre na primeira utilização do sistema é necessário fazer a configuração do mesmo. Este modo é chamado de modo de configuração e mudando uma chave no sistema vamos para o modo de execução chamado de *Run_mode*.

7.1 Frame de Dados

Para a transmissão e recepção dos dados foi necessário a criação de um padrão de transmissão. O padrão criado foi baseado no modbus.

Foram criados dois tipos de frames, um frame para leitura de dados e outro para escrita de dados. Os frames seguem o seguinte padrão, 7 bytes para o endereço de ip, 1 byte para sinalizar escrita ou leitura, e um byte de dados. Para a escrita de dados é utilizado o frame no formato da Figura 6.

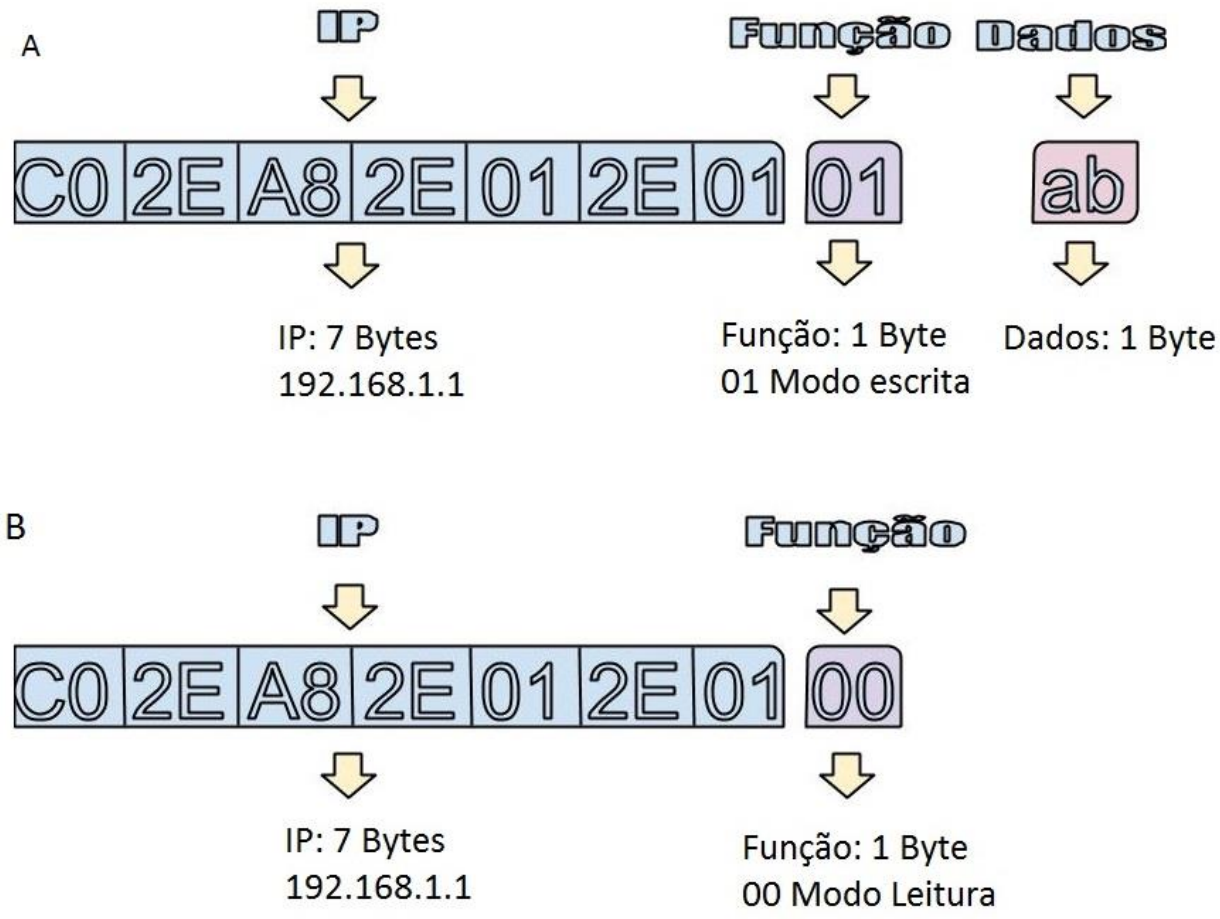

Figura 6: Frame de dados A: Mudar estado das saídas ou escrever no servidor B: Requisitar estado das cargas

7.2 Fluxograma do código

Aqui podemos analisar o fluxograma principal e de suas subrotinas:

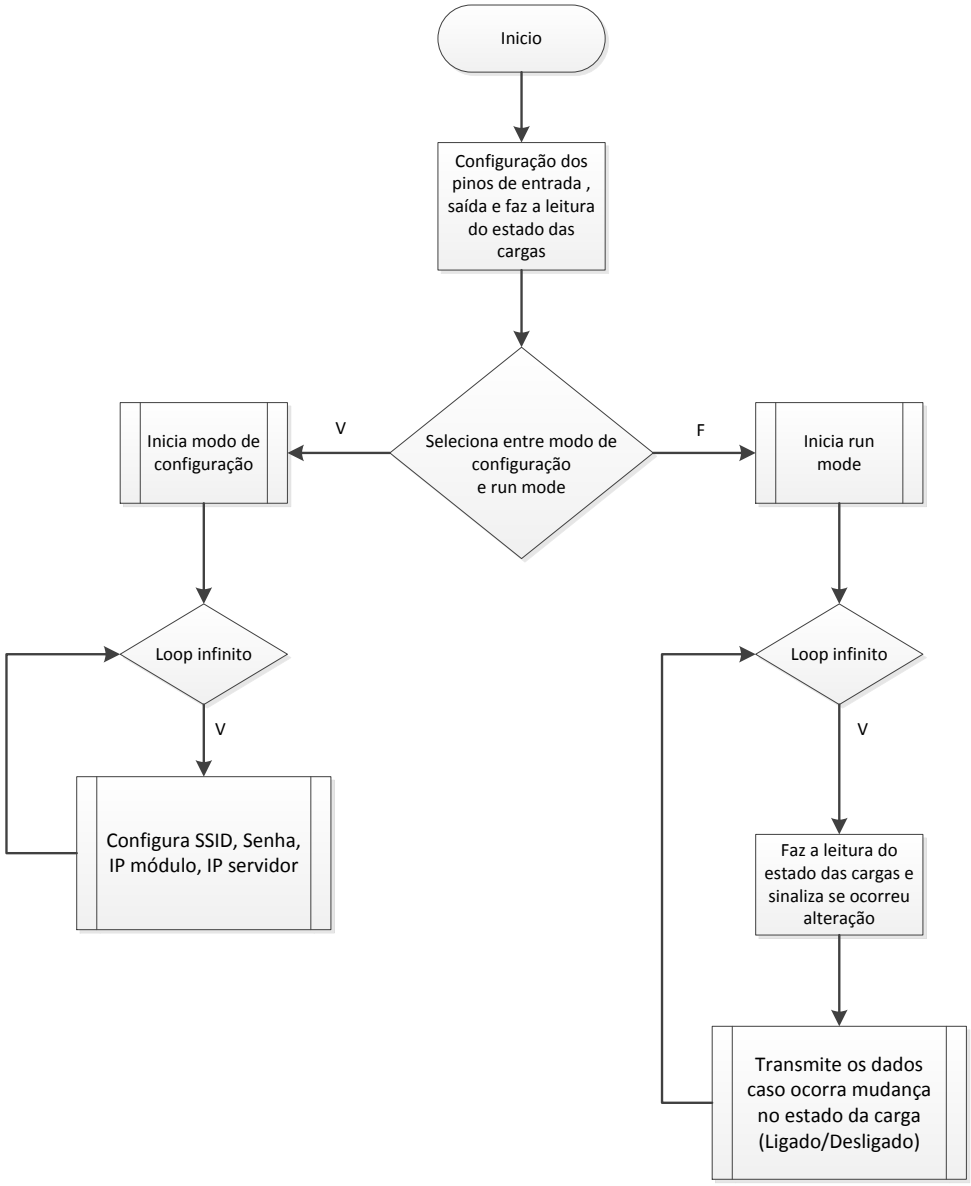

Figura 7: Código principal

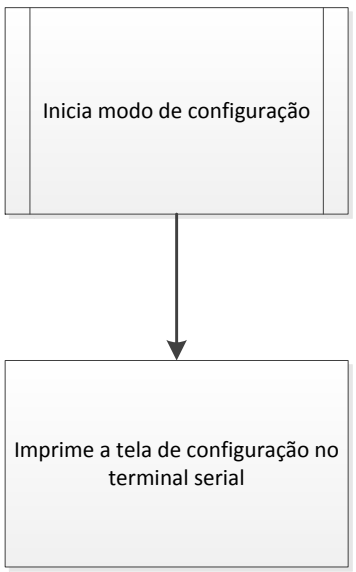

Figura 8: Inicia o modo de configuração

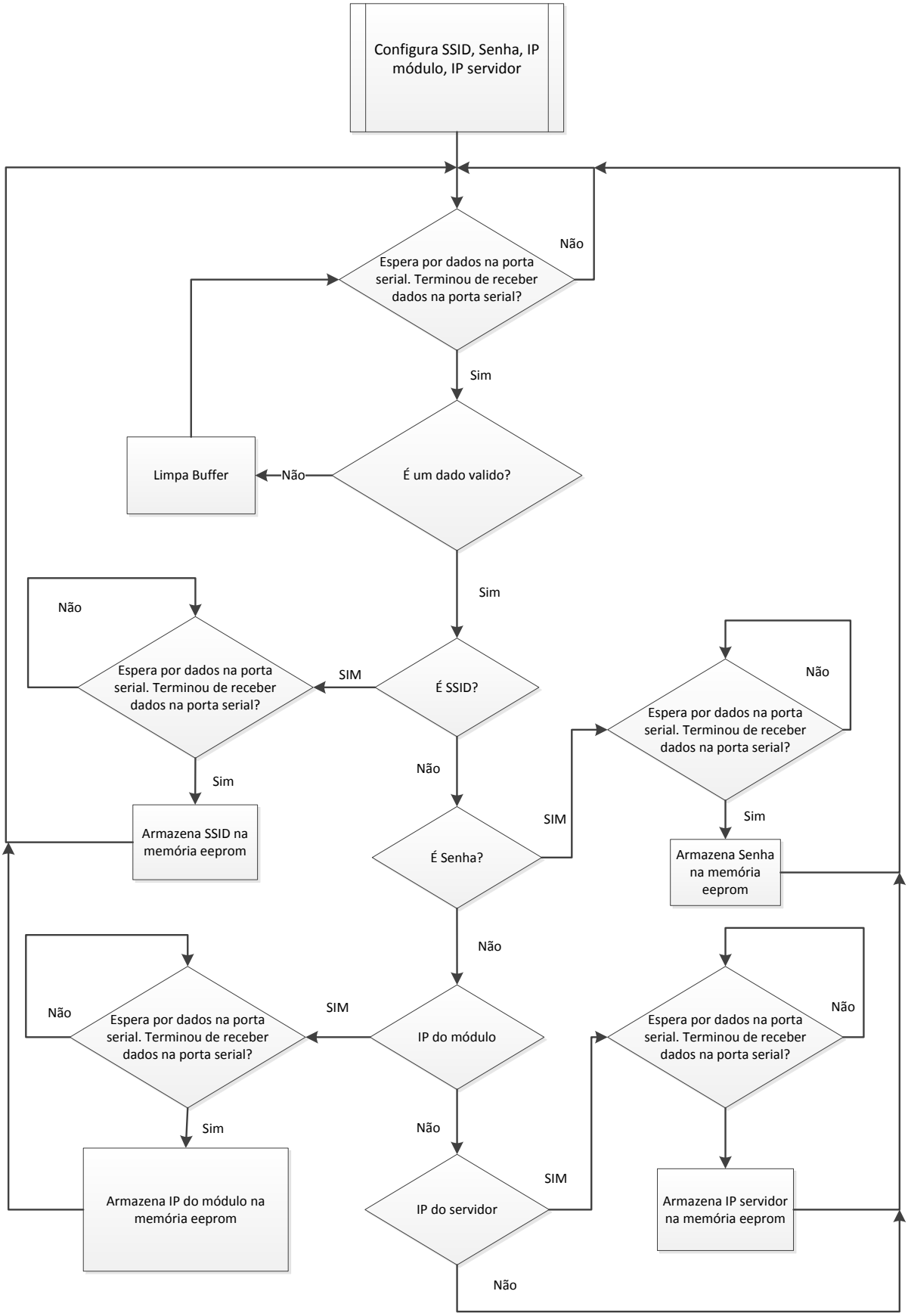

Figura 9: Subrotina configuração do módulo

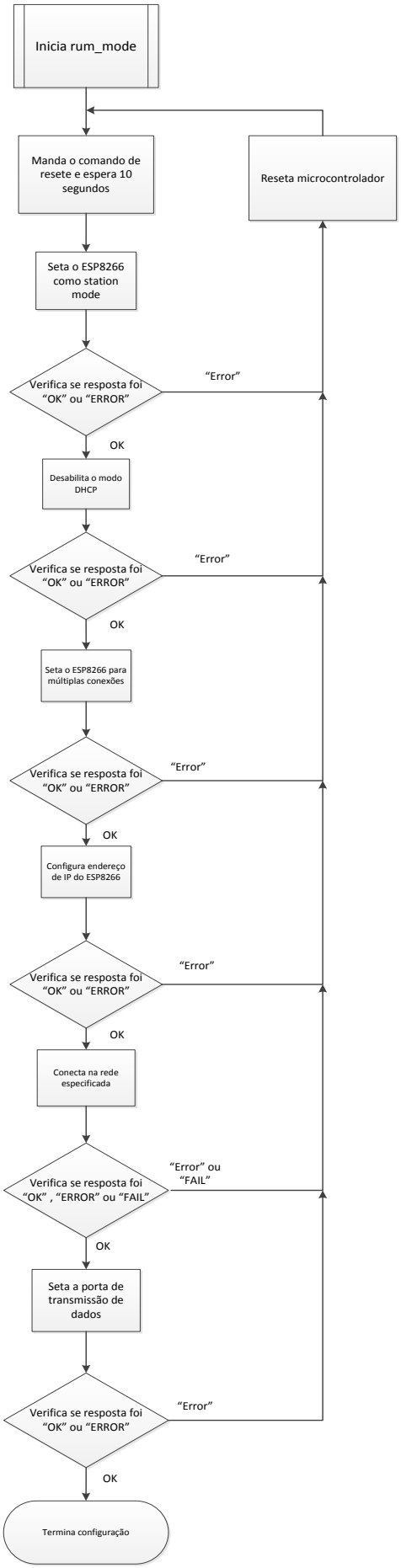

Figura 10: Subrotina inicialização em rum_mode

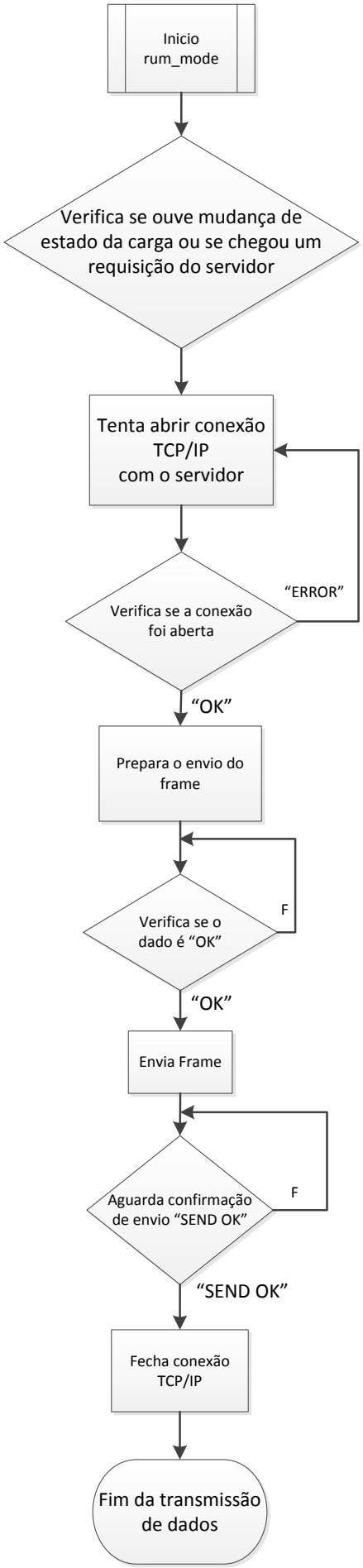
Inicio
rum_mode
Verifica se ouve mudança de estado da carga ou se chegou um requisição do servidor
Tenta abrir conexão TCP/IP com o servidor
Verifica se a conexão foi aberta
"ERROR"
"OK"
Prepara o envio do frame
Verifica se o dado é "OK"
F
"OK"
Envia Frame
Aguarda confirmação de envio "SEND OK"
F
"SEND OK"
Fecha conexão TCP/IP
Fim da transmissão de dados

Figura 11: Subrotina de transmissão de dados

Para o desenvolvimento do código fonte foi utilizado a IDE do arduino versão 1.65. Nesta IDE encontramos diversos problemas com as interrupções do sistema, tais problemas só podem ser resolvidos utilizando outra IDE de programação.

8. Teste de funcionamento

Para verificar o funcionamento do sistema utilizamos o software hercules_3-2-8 para simular um servidor e receber e transmitir os dados quando ocorrer uma mudança nos estados das cargas. Na Figura 12 verificamos a abertura da conexão e transmissão dos dados dos estados das cargas para o servidor utilizando a porta 502.

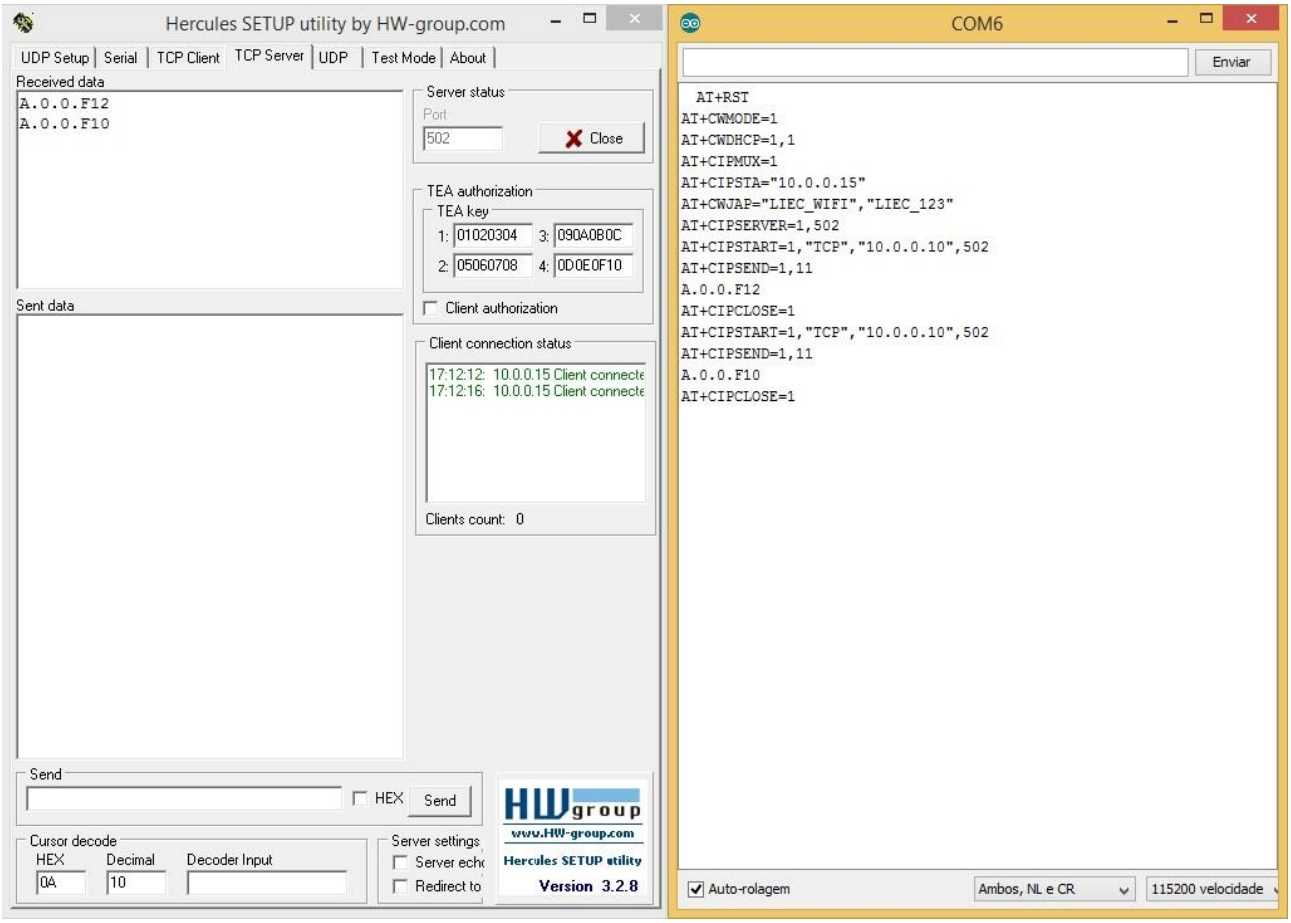

Figura 12: Transmissão dos dados para o servidor

9. Conclusão

Foi possível desenvolver um módulo que leia os estados das cargas e envie os dados para um servidor utilizando o arduino e o módulo wifi ESP8266. Ocorreram alguns problemas durante o desenvolvimento devido as limitações do software de desenvolvimento. É recomendado que para uma futura aplicação de maior porte seja utilizada outra IDE, é sugerido a utilização do Atmel Studio 6

por ser uma IDE livre e do próprio fabricante. Outro problema é o módulo wifi que é um módulo chinês o qual não temos o conhecimento do fabricante e de distribuidores autorizados. É recomendado utilizar um módulo de um fabricante conhecido e tenha distribuidores no Brasil. Os módulos com estas características são: rn -131 ou rn – 171 microchip ou o XBee Wi-Fi da digi.

Anexo A

Basic AT Instruction SET

Basic	
Instruction	**Description**
AT	Test AT startup
AT+RST	Restart module
AT+GMR	View version info
AT+GSLP	Enter deep-sleep mode
ATE	AT commands echo or not

WiFi Functions

WIFI	
Instruction	**Description**
AT+CWMODE	WIFI mode（ station/softAP/station+softAP）
AT+CWJAP	Connect to AP
AT+CWLAP	Lists available APs
AT+CWQAP	Disconnect from AP
AT+CWSAP	Set parameters under AP mode
AT+CWLIF	Get station’s ip which is connected to ESP8266 softAP
AT+CWDHCP	Enable/Disable DHCP
AT+CIPSTAMAC	Set mac address of ESP8266 station
AT+CIPAPMAC	Set mac address of ESP8266 softAP
AT+CIPSTA	Set ip address of ESP8266 station
AT+CIPAP	Set ip address of ESP8266 softAP

WiFi Functions

TCP/IP	
Instruction	**Description**
AT+ CIPSTATUS	Get connection status
AT+CIPSTART	Establish TCP connection or register UDP port
AT+CIPSEND	Send data
AT+CIPCLOSE	Close TCP/UDP connection
AT+CIFSR	Get local IP address
AT+CIPMUX	Set multiple connections mode
AT+CIPSERVER	Configure as server

AT+CIPMODE	Set transmission mode
AT+CIPSTO	Set timeout when ESP8266 runs as TCP server

Anexo B

Mapa memoria eeprom	
Endereço	Dado
0	Número caracteres do SSID
1	SSID
2	SSID
3	SSID
4	SSID
5	SSID
6	SSID
7	SSID
8	SSID
9	SSID
10	SSID
11	SSID
12	SSID
13	SSID
14	SSID
15	Número caracteres da senha
16	Senha
17	Senha
18	Senha
19	Senha
20	Senha
21	Senha
22	Senha
23	Senha
24	Senha
25	Senha
26	Senha
27	Senha
28	Senha
29	Senha
30	Senha
31	Senha
32	Senha
33	Senha
34	Senha
35	Senha

36	Senha
37	Senha
38	Senha
39	Senha
40	Senha
41	Senha
42	Senha
43	Senha
44	Senha
45	Senha
46	Senha
47	Tamanho IP
48	IP
49	IP
50	IP
51	IP
52	IP
53	IP
54	IP
55	IP
56	IP
57	IP
58	IP
59	IP
60	IP
61	IP
62	IP
63	Tamanho IP servidor
64	IP
65	IP
66	IP
67	IP
68	IP
69	IP
70	IP
71	IP
72	IP
73	IP
74	IP
75	IP
76	IP
77	IP
78	IP

Arduino Nano					
Atmega 328P					
Pinos Arduino	I/O microcontrolador	Configuração Pino	Tipo de dado	Nome entrada/saída	Função
3		Entrada	Booleano	Status_Lampada_1	Ler estado da lâmpada 1
4		Entrada	Booleano	Status_Lampada_2	Ler estado da lâmpada 2
5		Entrada	Booleano	Status_Lampada_3	Ler estado da lâmpada 3
6		Saída	Booleano	Aciona_rele_1	Acionar Relé 1
7		Saída	Booleano	Aciona_rele_2	Acionar Relé 2
8		Saída	Booleano	Aciona_rele_3	Acionar Relé 3
1		-	-	TX	Transmite dados modulo ESP8266
2		-	-	RX	Recebe dados modulo ESP8266
9		Entrada PULL_UP	Booleano	modo_configura_ou_funciona	Seleciona se é configuração ou rum_mode