

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

DÉBORA FURIERI DA SILVEIRA
RAVENA MONTEIRO SOARES

RELATÓRIO REDES INDUSTRIAIS

SERRA
2019

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

DÉBORA FURIERI DA SILVEIRA
RAVENA MONTEIRO SOARES

RELATÓRIO REDES INDUSTRIAIS

IMPLEMENTAÇÃO DE INTEGRAÇÃO, EM PROVA DE CONCEITO, DE SISTEMA
INTEGRADO DE SUPERVISÃO, CONTROLE E ATUAÇÃO

Relatório referente ao projeto proposto pelo professor Rafael Emerick, desenvolvido durante a disciplina de Redes Industriais, referente ao semestre 2019/02.

SERRA
2019

SUMÁRIO

1.	INTRODUÇÃO.....	4
2.	OBJETIVO.....	4
3.	METODOLOGIA.....	5
	3.1 Planta.....	5
	3.2 OpenPLC.....	6
	3.3 OpenPLC Editor.....	6
	3.4 ScadaBR.	6
	3.5 Esp32.....	7
4.	DESENVOLVIMENTO.....	8
	4.1 Planta.....	8
	4.2 OpenPLC.....	8
	4.3 OpenPLC Editor.....	12
	4.4 ScadaBR.	13
	4.5 Esp32.....	17
5.	CONCLUSÃO.....	18
6.	BIBLIOGRAFIA.....	21

1. INTRODUÇÃO

Nesse trabalho, espera-se desenvolver e conhecer práticas e desafios inerentes a comunicação sem fio no contexto industrial usando recursos de baixo custo para fins educacionais. Alguns dos principais softwares utilizados foram OpenPLC, OpenPLC Editor, ScadaBR e Arduino IDE.

O OpenPLC é o primeiro CLP de código aberto padronizado e totalmente funcional, tanto em software quanto em hardware. Segundo informação retirada do site oficial desse software, o foco é fornecer uma solução industrial de baixo custo para automação e pesquisa. O OpenPLC tem sido usado em muitos trabalhos de pesquisa como uma estrutura para pesquisa em segurança cibernética industrial, uma vez que é o único controlador a fornecer todo o código-fonte.

Além disso, esse software conversa com o software ScadaBR. O ScadaBR é um sistema de controle supervisão e aquisição de dados (SCADA) que permite criar telas interativas, também chamadas de Interface Homem Máquina (HMI), para seus projetos de automação.

Os principais hardwares utilizados foram o Raspberry Pi 3 e o Esp32 que são, respectivamente, uma série de microcomputadores e uma série de microcontroladores de baixo custo e baixo consumo de energia.

O Modbus, protocolo utilizado nesse projeto, 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.

2. OBJETIVO

Nesse projeto, precisamos implementar uma prova de conceito de sistema de supervisão, controle e atuação/sensoriamento integrado pelas soluções de redes de comunicação disponíveis (aka, TCP/IP, Modbus, Hart, CAN, OPCUA, etc.) utilizando-se de microcontroladores(mcu) ou dispositivos comerciais para controle e sensoriamento e um computador de supervisão.

3. METODOLOGIA

A rede utilizada neste projeto foi disponibilizada por um roteador Mi Router, da marca Xiaomi, modelo 4A, que suporta bandar WiFi de 2,4 GHz a 5,0 GHz.



Figura 1 - Roteador modelo Mi 4A

3.1. Planta

A planta foi baseada em uma simulação de uma planta de nível, com característica de primeira ordem. Para isso, utilizamos um circuito RC, um led para indicar que o setpoint foi atingido e um botão para inicializar o processo.



Figura 2 - Tanque de nivelamento

3.2. OpenPLC

Para a instalação do OpenPLC, configuramos a Raspberry Pi com sistema Raspbian. Todo o passo a passo seguido se encontra no site oficial do OpenPLC. Através do IP desse dispositivo, pudemos acessar o software remotamente na porta 8080.

3.3. OpenPLC Editor

O OpenPLC Editor foi instalado em uma máquina Windows. O download foi feito no site oficial do OpenPLC, assim como o passo a passo de instalação.

3.4. ScadaBR

Para a instalação do ScadaBR foi necessária a prévia instalação do Tomcat, um servidor web em Java. Posteriormente, foi feito o clone de um repositório do github e feita a instalação em uma máquina macOS. Assim, acessamos o software localmente na porta 9090.

Register Type	Usage	PLC Address	Modbus Address	Register Size	Value Range	Access
Coil Registers	Digital Outputs	%QX0.0 - %QX99.7	0 - 799	1 bit	0 or 1 OFF or ON	read and write
Discrete Input Registers	Digital Inputs	%IX0.0 - %IX99.7	0 - 799	1 bit	0 or 1 OFF or ON	read only
Input Registers	Analog Inputs	%IW0 - %IW99	0 - 1023	16 bits	0 to 65,535	read only
Holding Registers	Analog Outputs	%QW0 - %QW99	0 - 1023	16 bits	0 to 65,535	read and write

Figura 3 - Endereçamento Modbus

3.5. ESP32

O site oficial do OpenPLC disponibiliza o código condizente para utilização do ESP8266. Conseguimos usar o mesmo código, adaptando as pinagens para a nossa realidade do ESP32, conseguindo assim abrir a conexão wifi.

Na imagem abaixo, temos as pinagens referentes ao ESP32.

ESP32 WROOM32 DevKit Pinout

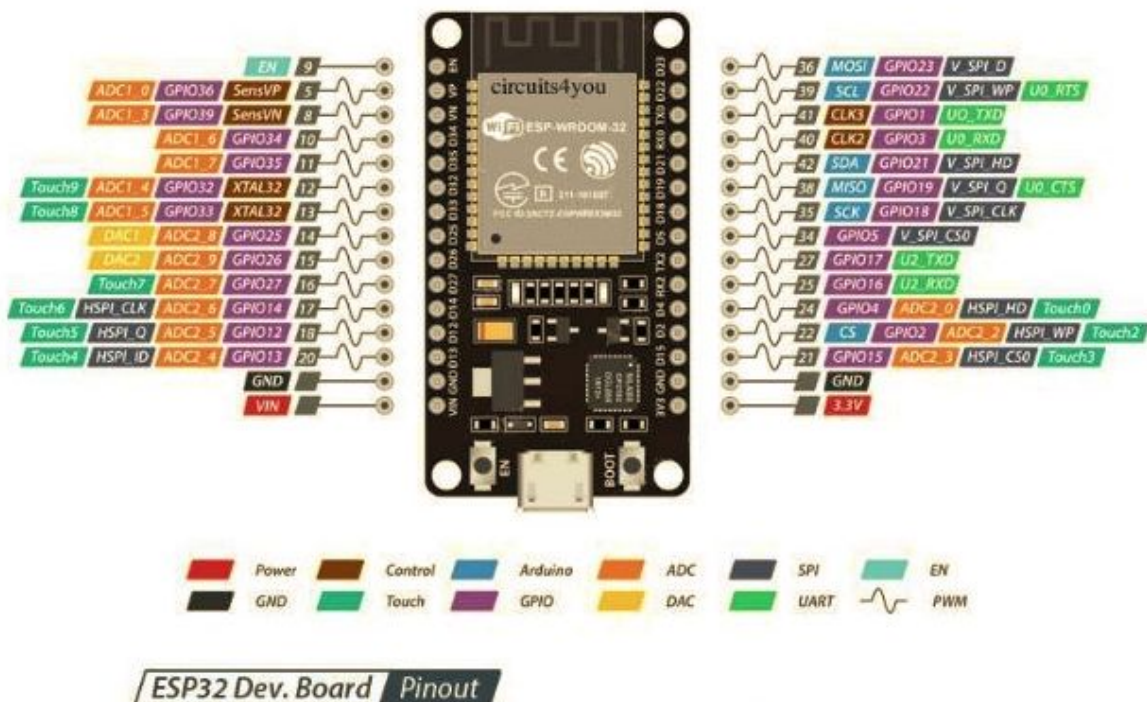


Figura 4 - Pinagem ESP32

4. DESENVOLVIMENTO

4.1. Planta

A seguir, temos a imagem da planta física, onde está exposto o escravo utilizado, o ESP32, o led aceso, indicando que o setpoint foi atingido e o botão de acionamento do processo.

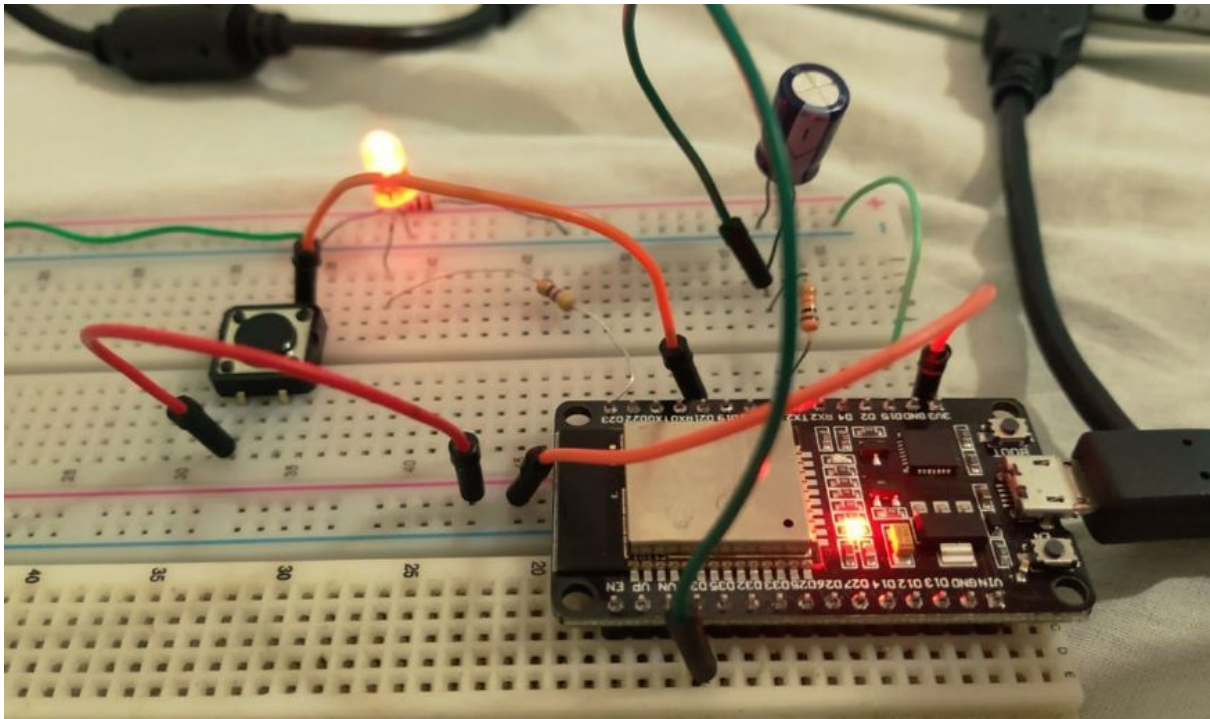


Figura 5 - Planta física

4.2. OpenPLC

Feita a instalação do OpenPLC, foi necessário adicionar um dispositivo escravo, que no nosso caso seria o ESP32. Dessa forma, fomos na aba *Slave Devices* e configuramos um dispositivo ESP32.

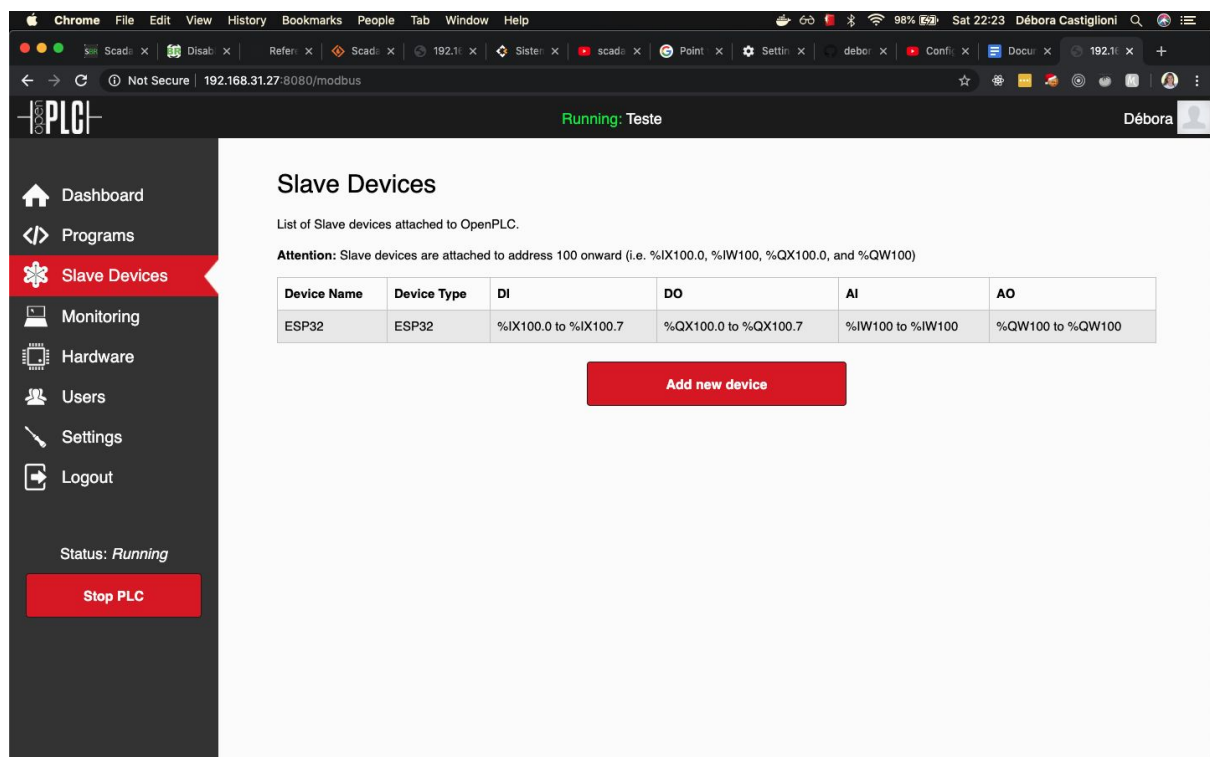


Figura 6 - ESP32 como Slave Device

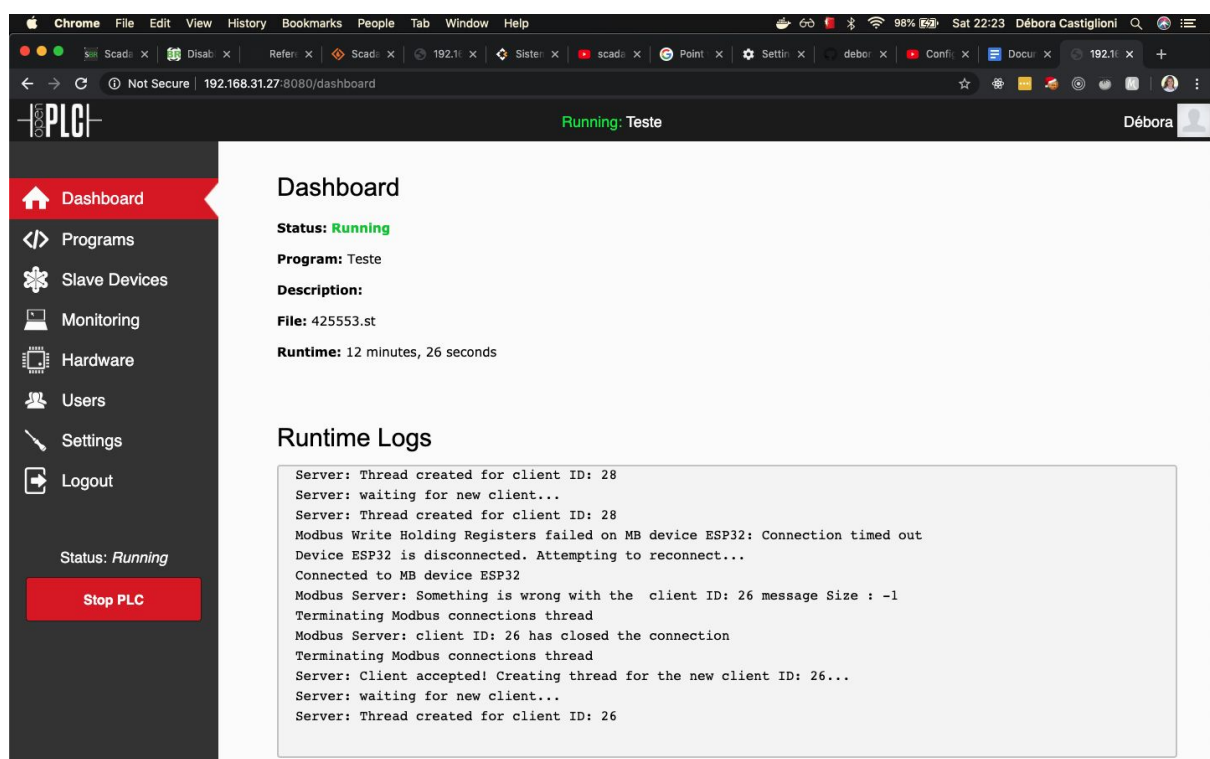


Figura 7 - Dashboard com Runtime Logs

Além disso, utilizamos a aba *Programs* para fazer o upload do nosso arquivo em ladder criado no *ScadaBR*.

Feito isso, foi possível acompanhar o processo de ação do controle e visualização do estado das variáveis acessando a aba *Monitoring*. Nas imagens abaixo, temos o acompanhamento de alguns pontos do processo que serão descritos a seguir.

Na figura 8 podemos ver que o botão de controle foi acionado e o processo começou. Na figura 9 o processo de controle continua, podemos observar uma grande variação de algumas variáveis até atingir a estabilidade.

Monitoring

Refresh Rate (ms): 100 Update













Point Name	Type	Location	Forced	Value
button	BOOL	%IX100.0	No	 FALSE
button_scada	BOOL	%QX2.0	No	 TRUE
lamp	BOOL	%QX100.0	No	 FALSE
lamp_scada	BOOL	%QX3.0	No	 FALSE
IN	UINT	%IW100	No	12480 
IN_SCADA	UINT	%QIW0	No	12480 
OUT	INT	%QIW100	No	0 
OUT_SCADA	INT	%QIW1	No	0 
SP_SCADA	UINT	%QIW2	No	65472 
KP_SCADA	UINT	%QIW3	No	65472 
TI_SCADA	UINT	%QIW4	No	65472 
TD_SCADA	UINT	%QIW5	No	65472 

Figura 8 - Botão iniciou o processo

Monitoring

Refresh Rate (ms): Update

Point Name	Type	Location	Forced	Value
button	BOOL	%IX100.0	No	FALSE
IN	INT	%IW100	No	27456
IN_SCADA	INT	%QW0	No	27456
OUT_SCADA	INT	%QW1	No	0
SP_SCADA	INT	%QW2	No	-536
KP_SCADA	INT	%QW3	No	-5536
TI_SCADA	INT	%QW4	No	-1536
TD_SCADA	INT	%QW5	No	-536
OUT	INT	%QW100	No	24576

Figura 9 - Planta em processo de ação de controle

Nas imagens abaixo, conseguimos observar a planta controlada em ação de setpoint máximo e mínimo, respectivamente. Observa-se que as variáveis referente ao led (*lamp* e *lamp_scada*) estão em *true*.

Monitoring

Refresh Rate (ms): Update

Point Name	Type	Location	Forced	Value
button	BOOL	%IX100.0	No	 FALSE
button_scada	BOOL	%QX2.0	No	 TRUE
lamp	BOOL	%QX100.0	No	 TRUE
lamp_scada	BOOL	%QX3.0	No	 TRUE
IN	UINT	%IW100	No	65472
IN_SCADA	UINT	%QW0	No	65472
OUT	INT	%QW100	No	27276
OUT_SCADA	INT	%QW1	No	27276
SP_SCADA	UINT	%QW2	No	65472
KP_SCADA	UINT	%QW3	No	65472
TI_SCADA	UINT	%QW4	No	65472
TD_SCADA	UINT	%QW5	No	65472

Figura 10 - Planta controlada com setpoint máximo

Monitoring

Refresh Rate (ms): × Update













Point Name	Type	Location	Forced	Value
button	BOOL	%IX100.0	No	 FALSE
IN	UINT	%IW100	No	 0
IN_SCADA	UINT	%QW0	No	 0
OUT_SCADA	UINT	%QW1	No	 0
SP_SCADA	UINT	%QW2	No	 0
KP_SCADA	UINT	%QW3	No	 60
TI_SCADA	UINT	%QW4	No	 65
TD_SCADA	UINT	%QW5	No	 6500
OUT	UINT	%QW100	No	 0
button_scada	BOOL	%QX2.0	No	 TRUE
lamp	BOOL	%QX100.0	No	 TRUE
lamp_scada	BOOL	%QX3.0	No	 TRUE

Figura 11 - Planta controlada com setpoint mínimo

4.3. OpenPLC Editor

Na imagem abaixo, temos as relações das variáveis utilizadas e seus respectivos tipos e localização. A localização foi definida seguindo o endereçamento Modbus, exposto no item 3.4 deste relatório.

controle							
Descrição:				Class Filter: Todos			
#	Nome	Class	Tipo	Localização	Valor Inicial	Opção	Documentação
1	button	Local	BOOL	%IX100.0			BUTTON LIGA/RESET
2	button_scada	Local	BOOL	%QX2.0			LIGA O CONTROLE NO SCADA
3	lamp	Local	BOOL	%QX100.0			SINAL DO ERROR ZERO
4	lamp_scada	Local	BOOL	%QX3.0			SINAL DO ERRO NO SCADA
5	IN	Local	UINT	%IW100			SAIDA DA PLANTA RC
6	IN_SCADA	Local	UINT	%QW0			SAIDA DA PLANTA RC PARA S
7	OUT	Local	INT	%QW100			AÇÃO DE CONTROLE
8	OUT_SCADA	Local	INT	%QW1			AÇÃO DE CONTROLE PARA O
9	SP_SCADA	Local	UINT	%QW2			SETPPOINT DO SCADA
10	PID0	Local	PID				BLOCO PID
11	KP_SCADA	Local	UINT	%QW3			PROPORCIONAL DO SCADA
12	TI_SCADA	Local	UINT	%QW4			INTEGRAL DO SCADA
13	TD_SCADA	Local	UINT	%QW5			DERIVATIVO DO SCADA

Figura 12 - Variáveis utilizadas no processo

Construímos uma lógica em Ladder para descrever o processo de ação de controle.

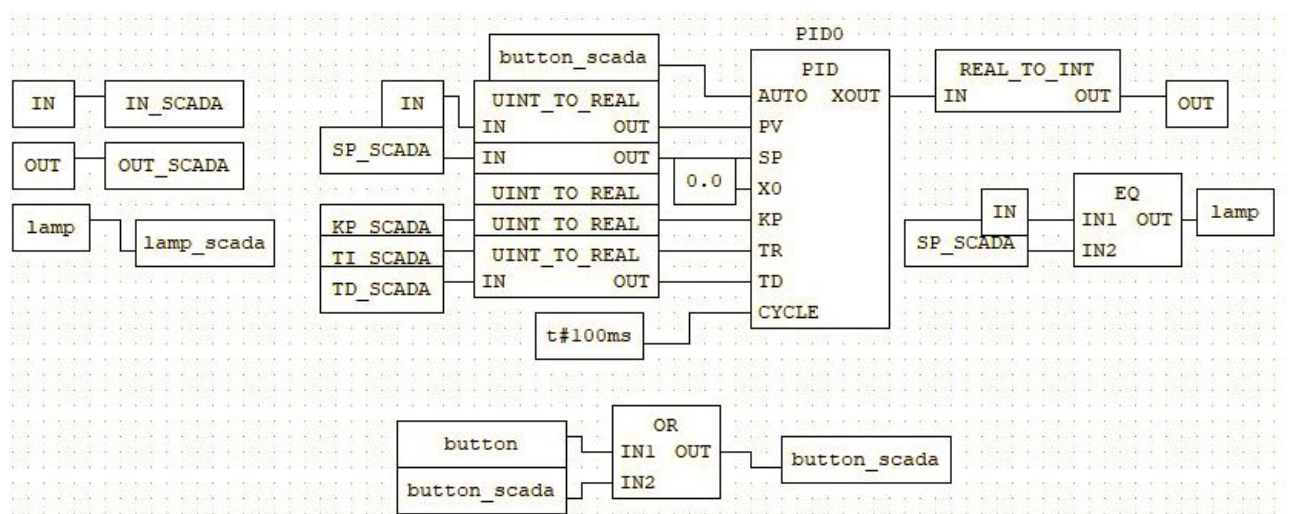


Figura 13 - Código em Ladder

4.4. ScadaBR

Na imagem abaixo, temos a configuração do ScadaBR para aceitar a conexão Modbus TCP.

Modbus IP properties

Data source has been saved

Name RC

Export ID (XID) DS_282564

Update period 500 millisecond(ms)

Quantize ☐

Timeout (ms) 20000

Retries 2

Contiguous batches only ☐

Create slave monitor points ☐

Max read bit count 2000

Max read register count 125

Max write register count 120

Transport type TCP

Host 192.168.31.27

Port 502

Encapsulated ☐

Event alarm levels

Data source exception Urgent

Point read exception Urgent

Point write exception Urgent

Modbus node scan

Scan for nodes Cancel

Scan completed

Nodes found

1
2
3
4
5
6
7
8

Modbus read data

Slave id 1

Register range Coil status

Offset (0-based) 0

Number of registers 100

Read data

85 ==> false

86 ==> false

87 ==> false

88 ==> false

89 ==> false

90 ==> false

91 ==> false

92 ==> false

93 ==> false

94 ==> false

95 ==> false

96 ==> false

97 ==> false

98 ==> false

99 ==> false

Point locator test

Slave id 1

Register range Coil status

Modbus data type Binary

Offset (0-based) 0

Bit 0

Number of registers 0

Character encoding ASCII

Read Add point

Figura 14 - Configuração Modbus

Criamos, também, uma interface de supervisão simulando o nível do tanque, o botão para iniciar o controle, além da saída, setpoint e variáveis KP, TI e TD. Nessa interface, é possível alterar os valores das variáveis.

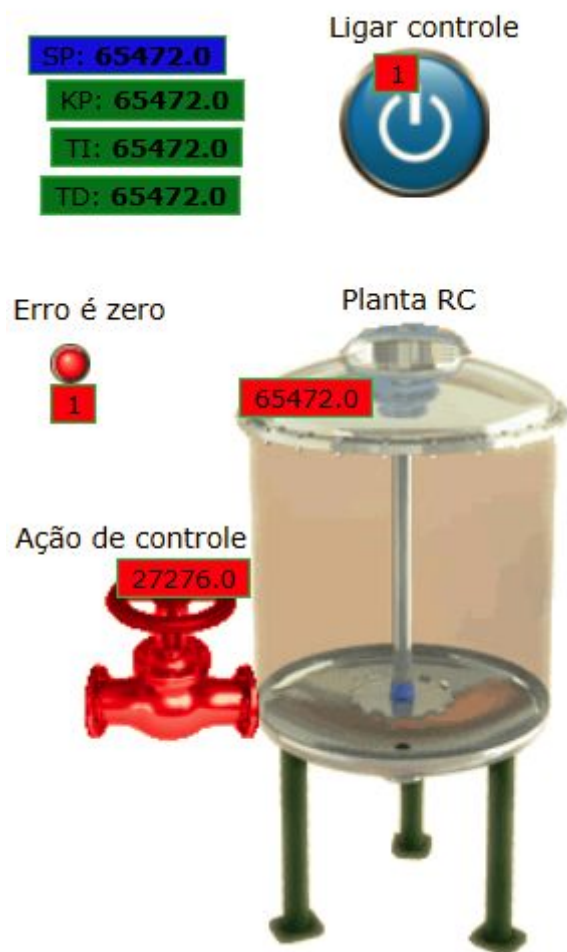


Figura 15 - Supervisório

Os *data point* utilizados estão descritos abaixo, juntamente com o tipo de dado, a faixa e o offset, baseado no endereçamento Modbus definido no OpenPLC.

Data points						
Nome	Tipo de dado	Status	Escravo	Faixa	Offset (baseado em 0)	
AÇÃO DE CONTROLE	Numérico		1	Registrador holding	1	
BUTTON_scada	Binário		1	Status do coil	16	
KP	Numérico		1	Registrador holding	3	
LED_scada	Binário		1	Status do coil	24	
SAIDA DA PLANTA RC	Numérico		1	Registrador holding	0	
Slave 1 monitor	Binário		1	Monitor de escravo		
SP	Numérico		1	Registrador holding	2	
TD	Numérico		1	Registrador holding	5	
TI	Numérico		1	Registrador holding	4	

Figura 16 - *Data points*

Na *watch list* abaixo, é possível, também, variar os valores das nossas variáveis, além de servir para acompanhar a variação em tempo real delas.

































Watch list 				(sem nome)    
 openPLC - BUTTON_scada	1	02:36:31	<input checked="" type="checkbox"/>	 
 openPLC - LED_scada	1	02:25:30	<input checked="" type="checkbox"/>	 
 openPLC - SP	65472.0	02:51:51	<input checked="" type="checkbox"/>	 
 openPLC - KP	65472.0	02:51:54	<input checked="" type="checkbox"/>	 
 openPLC - TI	65472.0	02:51:56	<input checked="" type="checkbox"/>	 
 openPLC - TD	65472.0	02:51:59	<input checked="" type="checkbox"/>	 
 openPLC - AÇÃO DE CONTROLE	27276.0	02:57:39	<input checked="" type="checkbox"/>	 
 openPLC - SAIDA DA PLANTA RC	65472.0	02:57:42	<input checked="" type="checkbox"/>	 
 openPLC - Slave 1 monitor	0	00:34:09	<input checked="" type="checkbox"/>	 

Figura 17 - Acompanhamento dos dados supervisionados

Nos gráficos abaixo, é possível acompanhar as variações da saída da planta e da ação de controle com relação ao tempo.

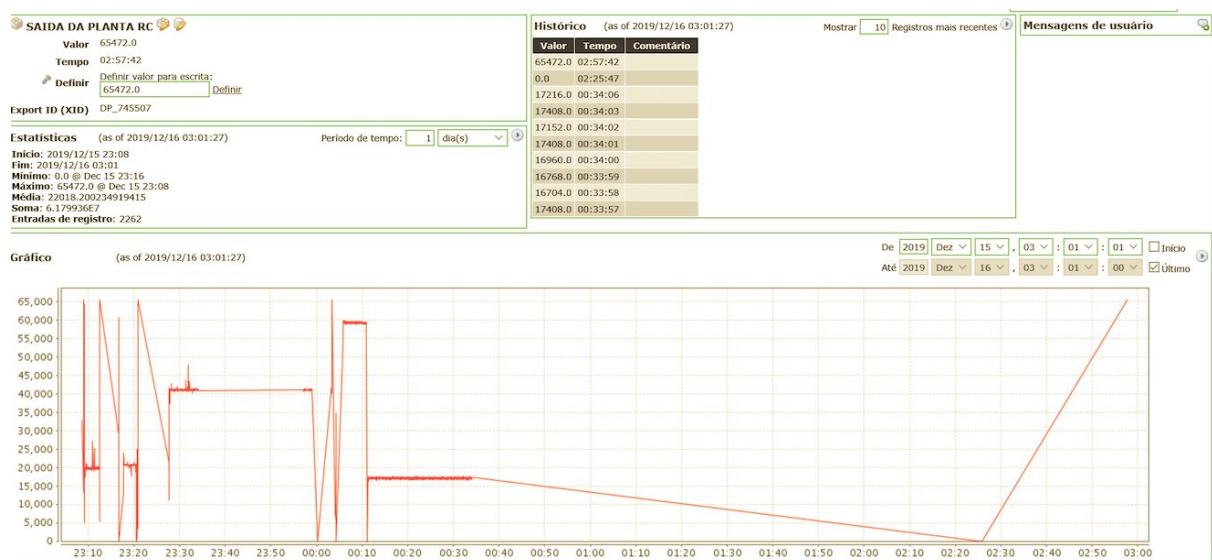


Figura 18 - Saída da Planta

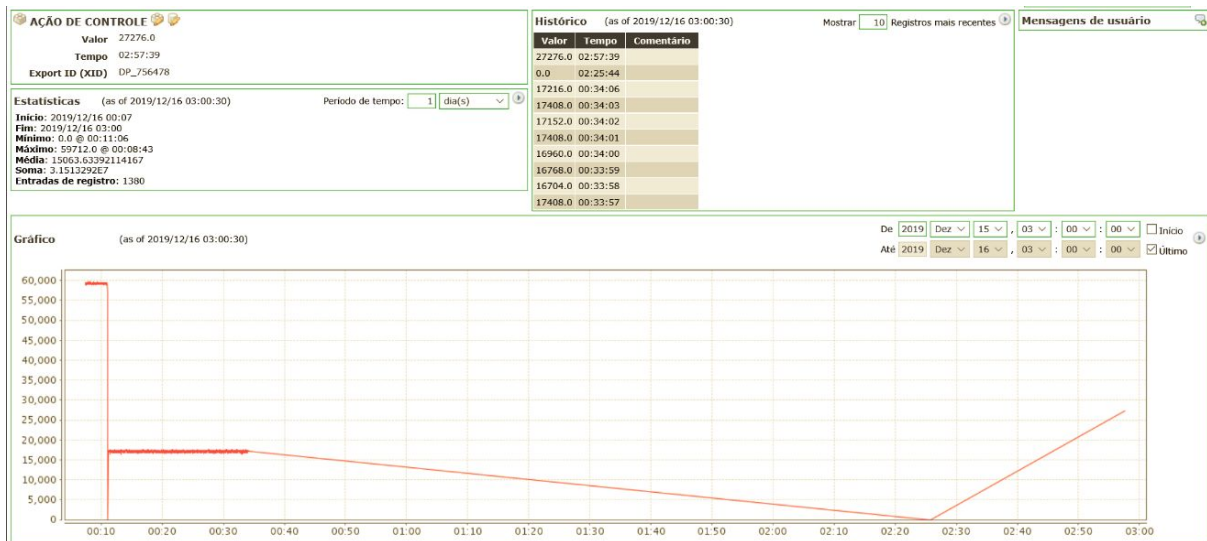


Figura 19 - Ação de controle

4.5. ESP32

Como citado na metodologia, item 3.5, precisamos fazer algumas alterações de endereçamento de pinos para adaptar o código para o ESP32, além de incluir a biblioteca *Wifi.h*.

Compilado o código na IDE Arduino, conseguimos visualizar o resultado exposto na imagem abaixo, que confirma a conexão realizada com sucesso.

Como visto, a comunicação com o mestre se dá via Wifi.

```

/dev/cu.SLAB_USBtoUART

Connecting to GAIN Router
...
WiFi connected
Server started
My IP: 192.168.31.105
new client!

Autoscroll Show timestamp Both NL & CR 115200 baud Clear output

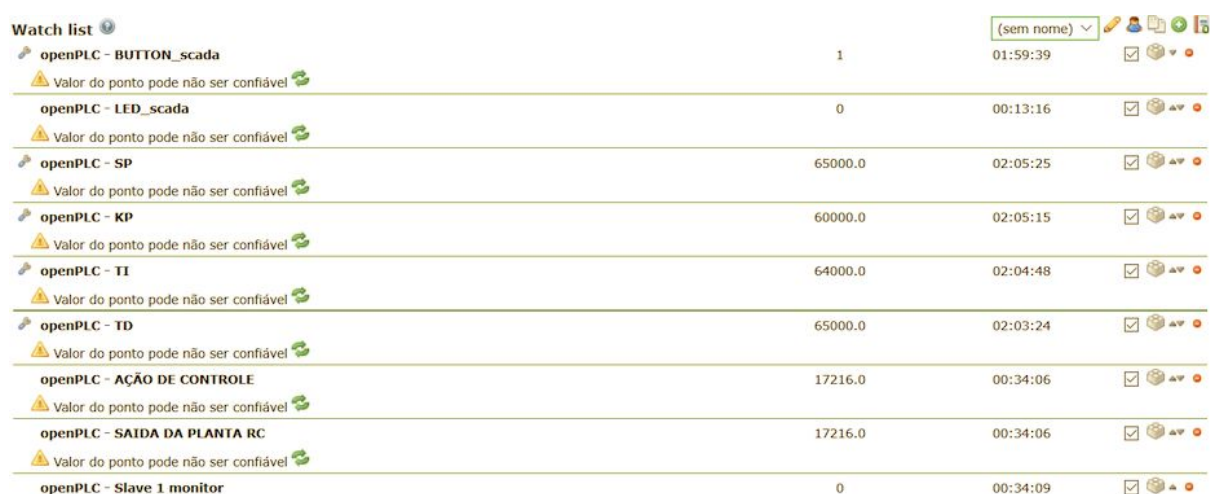
```

Figura 20 - Output IDE Arduino

5. CONCLUSÃO

Em geral, devido aos tutoriais encontrados no site oficial do OpenPLC, as instalações dos programas foram concluídas sem muita dificuldade. Com exceção do ScadaBR, onde foi necessário instalar o *Tomcat*, um servidor java.

Entretanto, encontramos algumas dificuldades para configuração dos endereços Modbus. Como pode ser visto nas imagens abaixo, foram feitas várias tentativas e bastantes pesquisas no Google para entender e endereçar corretamente nossas variáveis. Infelizmente, poucos materiais e fóruns foram encontrados que citavam os erros que nos deparamos no desenvolvimento do projeto.



Watch list		(sem nome)		
openPLC - BUTTON_scada	1	01:59:39	<input checked="" type="checkbox"/>	
openPLC - LED_scada	0	00:13:16	<input checked="" type="checkbox"/>	
openPLC - SP	65000.0	02:05:25	<input checked="" type="checkbox"/>	
openPLC - KP	60000.0	02:05:15	<input checked="" type="checkbox"/>	
openPLC - TI	64000.0	02:04:48	<input checked="" type="checkbox"/>	
openPLC - TD	65000.0	02:03:24	<input checked="" type="checkbox"/>	
openPLC - AÇÃO DE CONTROLE	17216.0	00:34:06	<input checked="" type="checkbox"/>	
openPLC - SAIDA DA PLANTA RC	17216.0	00:34:06	<input checked="" type="checkbox"/>	
openPLC - Slave 1 monitor	0	00:34:09	<input checked="" type="checkbox"/>	

Figura 21 - Erro *Valor do ponto pode não ser confiável*

Nos deparamos com erros de conexão, timeouts e exceptions.

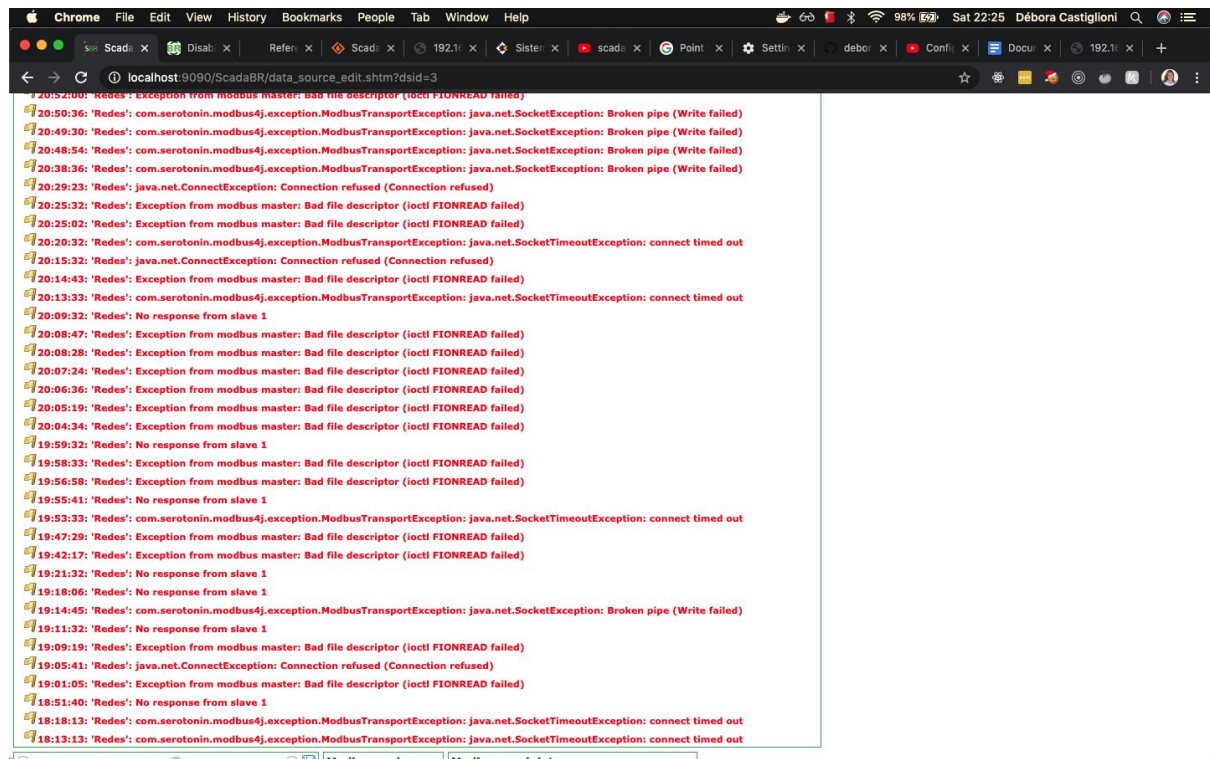


Figura 22 - Erro conexão Modbus

Utilizando os comandos *traceroute* e *ping*, tivemos os seguintes outputs.

```
deboracastigli@Suportes-MacBook-Pro-97:~$ traceroute 192.168.31.27
traceroute to 192.168.31.27 (192.168.31.27), 64 hops max, 52 byte packets
 1  192.168.31.27 (192.168.31.27)  10.583 ms  2.340 ms  1.718 ms
```

Figura 23 - Traceroute exibindo apenas um salto

Desempenho da rede foi em geral satisfatório. Como pode ser visto na imagem abaixo, ao pingar o IP 192.168.31.27, do PLC, houve 0% de perda de pacote.

```
deboracastiglioni@Suportes-MacBook-Pro-97:~$ ping 192.168.31.27
PING 192.168.31.27 (192.168.31.27): 56 data bytes
64 bytes from 192.168.31.27: icmp_seq=0 ttl=64 time=1.708 ms
64 bytes from 192.168.31.27: icmp_seq=1 ttl=64 time=5.969 ms
64 bytes from 192.168.31.27: icmp_seq=2 ttl=64 time=6.769 ms
64 bytes from 192.168.31.27: icmp_seq=3 ttl=64 time=1.840 ms
64 bytes from 192.168.31.27: icmp_seq=4 ttl=64 time=7.027 ms
64 bytes from 192.168.31.27: icmp_seq=5 ttl=64 time=1.944 ms
64 bytes from 192.168.31.27: icmp_seq=6 ttl=64 time=1.751 ms
64 bytes from 192.168.31.27: icmp_seq=7 ttl=64 time=1.933 ms
64 bytes from 192.168.31.27: icmp_seq=8 ttl=64 time=2.008 ms
64 bytes from 192.168.31.27: icmp_seq=9 ttl=64 time=1.880 ms
64 bytes from 192.168.31.27: icmp_seq=10 ttl=64 time=2.490 ms
64 bytes from 192.168.31.27: icmp_seq=11 ttl=64 time=28.375 ms
64 bytes from 192.168.31.27: icmp_seq=12 ttl=64 time=2.009 ms
64 bytes from 192.168.31.27: icmp_seq=13 ttl=64 time=2.826 ms
64 bytes from 192.168.31.27: icmp_seq=14 ttl=64 time=6.740 ms
64 bytes from 192.168.31.27: icmp_seq=15 ttl=64 time=1.908 ms
64 bytes from 192.168.31.27: icmp_seq=16 ttl=64 time=1.862 ms
64 bytes from 192.168.31.27: icmp_seq=17 ttl=64 time=7.642 ms
64 bytes from 192.168.31.27: icmp_seq=18 ttl=64 time=2.141 ms
64 bytes from 192.168.31.27: icmp_seq=19 ttl=64 time=3.746 ms
64 bytes from 192.168.31.27: icmp_seq=20 ttl=64 time=1.775 ms
64 bytes from 192.168.31.27: icmp_seq=21 ttl=64 time=5.232 ms
64 bytes from 192.168.31.27: icmp_seq=22 ttl=64 time=7.476 ms
64 bytes from 192.168.31.27: icmp_seq=23 ttl=64 time=2.106 ms
64 bytes from 192.168.31.27: icmp_seq=24 ttl=64 time=5.523 ms
64 bytes from 192.168.31.27: icmp_seq=25 ttl=64 time=2.212 ms
64 bytes from 192.168.31.27: icmp_seq=26 ttl=64 time=1.796 ms
64 bytes from 192.168.31.27: icmp_seq=27 ttl=64 time=9.631 ms
64 bytes from 192.168.31.27: icmp_seq=28 ttl=64 time=3.121 ms
64 bytes from 192.168.31.27: icmp_seq=29 ttl=64 time=2.016 ms
64 bytes from 192.168.31.27: icmp_seq=30 ttl=64 time=1.902 ms
^C
--- 192.168.31.27 ping statistics ---
31 packets transmitted, 31 packets received, 0.0% packet loss
```

Figura 24 - Ping no IP 192.168.31.27

6. BIBLIOGRAFIA

OPENPLC PROJECT, disponível em <<https://www.openplcproject.com/>>

OPENPLC RUNTIME, disponível em <<https://www.openplcproject.com/runtime>>

OPENPLC ON RASPBERRY PI, disponível em
<<https://www.openplcproject.com/getting-started-rpi/>>

OPENPLC EDITOR, disponível em
<<https://www.openplcproject.com/plcopen-editor>>

INSTALLING SCADABR, disponível em
<<https://www.openplcproject.com/reference-installing-scadabr>>

OPENPLC AND ESP8266, disponível em
<<https://www.openplcproject.com/getting-started-esp8266>>

ESP32 ADC, disponível em
<<https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/>>