
Sistema Eletrónicos para a Internet das Coisas

Engenharia Eletrotécnica e de Computadores – 3ºAno

Ano Letivo de 2021/2022

2ºSemestre-Regime Diurno

Elaborado por:

Gedião Manaça Nº 2172302

Augusto de lara Nº 2213335

Índice

Índice	2
Índice de Figuras	3
1. Introdução	4
2. Funcionamento Geral.....	5
2.1 Diagrama de blocos geral	5
3. Sistema de Temperatura	6
4. Sistema de Luminosidade.....	8
5. Logica Sensorial no Node-Red.....	10
6. Código para o envio da notificação.....	11
7. Emails recebidos com as notificações.....	12
8. Implementação Broker e Node-RED	12
9. SaveData.....	15
10. Comunicação ESP32 BROKER	18
11. Conclusão	20
12. Bibliografia	21

Índice de Figuras

Figura 1 - Sensor de Temperatura.....	7
Figura 4 - Amplificador Não Inversor	8
Figura 5 - Temperatura padrão	9
Figura 7 - Temperatura de Alarme	9
Figura 9 - Sensor de luminosidade	10
Figura 11 - Alarme de luminosidade	10
Figura 12 - Blocos desenvolvidos para o sensor de temperatura e luminosidade	11
Figura 13 - Código da função que salva a temperatura em uma variável global.....	11
Figura 14 - Código da função que salva a luminosidade em uma variável global.....	12
Figura 15 - Bloco node-red com a lógica para envio do email de alerta.....	12
Figura 16 - Função lógica para envio do email de alerta	12
Figura 17 - E Mails de alerta recebidos.	13
Figura 18 - Broker Aedes utilizado no Node-Red	13
Figura 19 - Blocos desenvolvidos para ligar e desligar led na raspberry Pi.....	14
Figura 20 - Código da função que verifica o payload enviado através do broker.....	14
Figura 21 - Execução do comando para desligar o Led.	14
Figura 22 - Execução de pub através de outro computador na mesma rede local.	15
Figura 23 - Controle do led pela UI	15
Figura 24 - Blocos desenvolvidos para salvar os dados na nuvem.	16
Figura 25 - Código da função que busca os dados e envia para a url request.	16
Figura 26 - Execução URL request.	17
Figura 27 - Senha de app na conta do Google.	17
Figura 28 - Ativando saveData através do mosquitto_pub.....	17
Figura 29 - Dados sendo salvos na planilha online a cada 1 segundo.	18
Figura 30 - Protoboard montada com o sensor de humidade.....	19
Figura 31 - O circuito foi utilizado para montar o sensor de humidade a Raspberry Pi.	20
Figura 32 - Gráfico gerado no Node-Red UI - Humidade.	20
Figura 33 - Gráfico gerado no Node-Red UI - Luminosidade e Temperatura	20
Figura 34 - Gráfico gerado no Node-Red UI - Temperatura.....	21

1. Introdução

No âmbito da unidade curricular de Sistema Eletrónicos para Internet das Coisas foi proposto o desenvolvimento de um Sistema com diversos dispositivos ligados entre si. O sistema desenvolvido foi no intuito de simular o funcionamento básico de uma estufa. O objetivo foi de interligar equipamentos eletrónicos de modo que seja possível controlar localmente e remotamente os vários sensores, e para que seja executado corretamente foi necessário recorrer a matéria lecionada ao longo da unidade curricular.

No presente relatório está descrito o processo executado para cumprir o objetivo da unidade curricular, isto é tanto o dimensionamento bem como as montagens executadas.

Para que este seja bem-sucedido foi necessário elaborar um diagrama de blocos onde continha as funcionalidades gerais do sistema e a fim de facilitar a evolução do projeto foi necessário recorrer a dispositivos como sensor de temperatura, luminosidade, humidade, Mplab, Pic18F, Raspberry pi, Usb-uart, Pickit4, ESP32 e DH11.

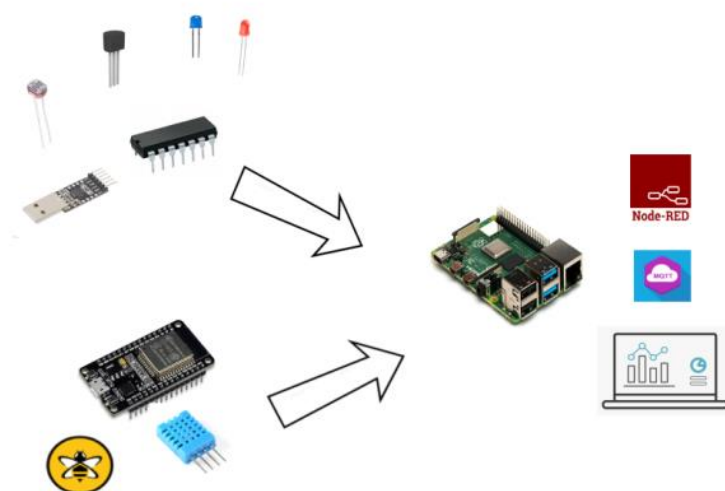


Figura 1 - Demonstração simplificada das tecnologias utilizadas

2 Funcionamento Geral

Neste capítulo iremos demonstrar o funcionamento geral do sistema sendo depois aprofundado nos capítulos seguintes.

2.1 Diagrama de blocos geral

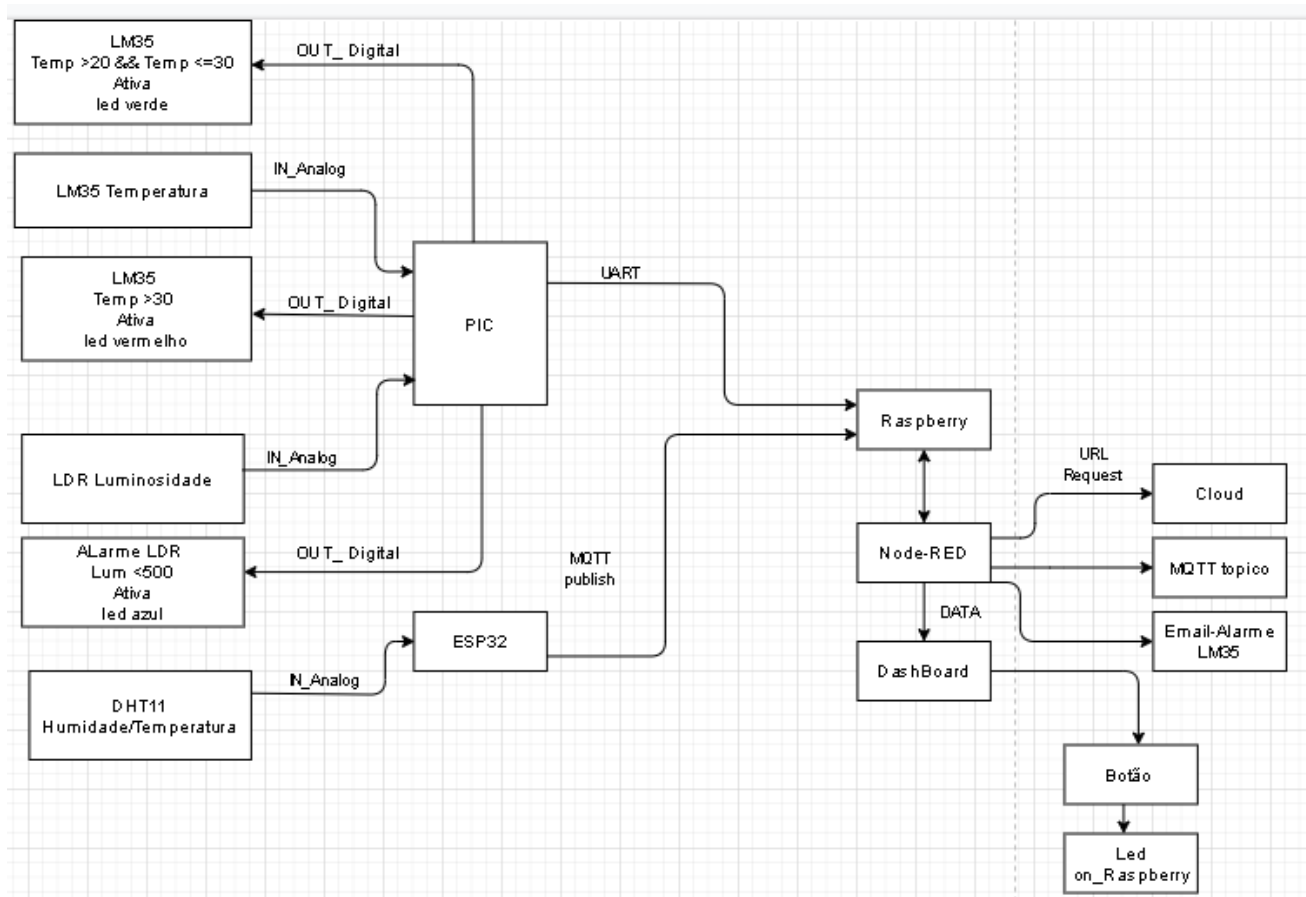


Figura 2 - Diagrama de blocos geral

3 Sistema de Temperatura

3.1 Sensor de temperatura

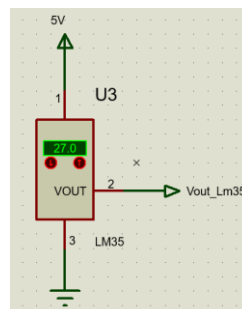


Figura 3 - Sensor de Temperatura

Para implementar o sistema de temperatura foi dimensionado um circuito utilizando o sensor LM35. Sendo este alimentado no pino1(5V), pino3(0V), de modo a gerar uma tensão de saída no pino2(Vout_Lm35) sendo esta uma tensão analógica e variável de 10.0 mV/°C.

Uma vez que na saída do lm35(Vout_Lm35) apresenta uma tensão relativamente baixa foi necessário recorrer a um amplificador de tensão não inversor de ganho 10 como está presente na Figura 4 de maneira a ter-se melhor resolução, sendo esta a amplificação de tensão (*Temp_Sensor*) dada na *Equação 1*. A mesma tensão será inserida e configurada no microcontrolador PIC18F26K40 como entrada analógica na porta RA1(*Temp_Sensor*) e convertida em sinal analógico para sinal digital, proveniente do ADCC de 10bits do microcontrolador.

Cálculos fundamentais:

O ganho do amplificador não inversor de tensão é dado na *Equação 1* por:

$$Temp_{Sensor} = 1 + \frac{R2}{R3} = 1 + \frac{90k}{10} = 10$$

Equação 1

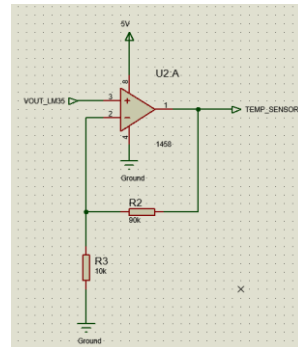


Figura 4 - Amplificador Não Inversor

3.2 Alarme de Temperatura

3.2.1 Temperatura padrão

Iremos controlar a temperatura apresentada pelo periférico de maneira que caso os valores estejam dentro da gama padrão [15°:24°] será caracterizado como temperatura recomendável/padrão do seu bom funcionamento que consequentemente a luz do led verde irá manter-se acesa.

Para tal configurou-se a porta RA0 (*Tem_LED*) do microcontrolador como saída digital, tendo assim por finalidade obter o valor convertido pelo ADC do sinal proveniente da porta RA1(*Temp_Sensor*), para tal foi dimensionado o seguinte circuito na *Equação 2*.

Para a condução do LED na porta RA0 é necessário que a resistência R4, onde o valor foi dimensionado na *Equação 2*, limite a corrente máxima de funcionamento impedindo que LED entre em saturação.

$$R4 = \frac{V_{cc} - V_{led}}{I_F} = \frac{5 - 1.7}{10 * 10^{-3}} = 330\Omega$$

Equação 2

Para aproveitamento de material disponível recorreu-se a uma resistência de 1kΩ em substituição a 330Ω calculada.

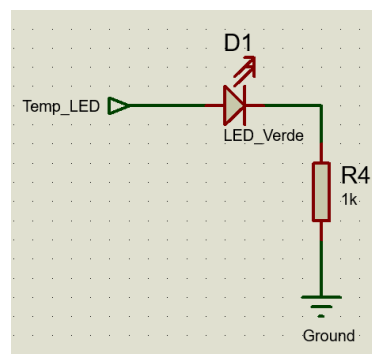


Figura 5 - Temperatura padrão

3.2.2 Temperatura de Alarme

Para este sistema de alarme foi necessário configurar a porta RC3(Temp_LED_Alarme) do microcontrolador como saída digital, de maneira a obter o valor convertido pelo ADC do sinal proveniente do pino3(Temp_Sensor).

E caso o sinal convertido Temp_Sensor ultrapasse os limites recomendáveis irá ser acionado um Led vermelho com uma frequência de 1Hz para sinalização da ultrapassagem do limite padrão quando se encontra superior a 25°.

Para o dimensionamento do circuito como mostra na Figura 6 recorreu-se ao cálculo da Equação 2.

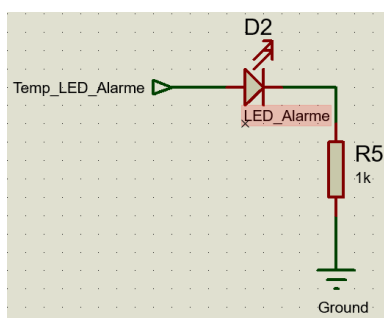


Figura 6 - Temperatura de Alarme

4 Sistema de Luminosidade

4.1 Sensor de luminosidade

Para implementar o sistema de luminosidade foi dimensionado um circuito utilizando o sensor LDR, recorrendo e calculando um divisor de tensão como mostra a Equação 3 na qual o LDR é alimentado a 5V em serie com uma resistência de 10k Ω como é mostrado na Figura 7. Na qual a saída do divisor de tensão foi configurada no microcontrolador como entrada analógica na porta RC5(Luz_Sensor).

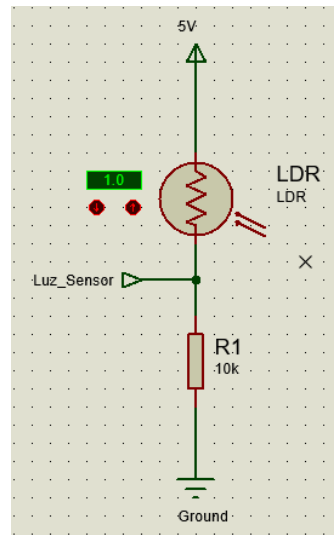


Figura 7 - Sensor de luminosidade

Cálculos fundamentais:

A saída do divisor de tensão (*Luz_Sensor*) foi calculado pela *Equação 3*.

$$Luz_{Sensor} = \frac{LDR}{LDR + 10k}$$

4.1.1 Alarme de luminosidade

Para este sistema de alarme foi necessário configurar a porta RC2(*Lum_Buzz_Alarme*) do microcontrolador como saída digital, de maneira a obter o valor convertido pelo ADC do sinal proveniente do RC5(*Luz_Sensor*).

Recorreu-se ao cálculo da *Equação 2* para dimensionamento do sistema de alarme como mostra a Figura 5.

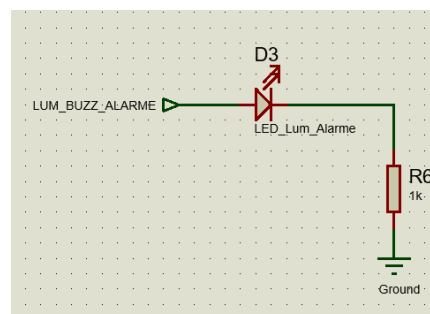


Figura 8 - Alarme de luminosidade

5 Logica Sensorial no Node-Red.

A comunicação entre a PIC e Raspberry é realizada através da comunicação por UART.

Os dados capturados pelos periféricos são enviados através da PIC, e em seguida através do Node-Red observamos estes dados e interagimos de maneira programática para os casos executados.

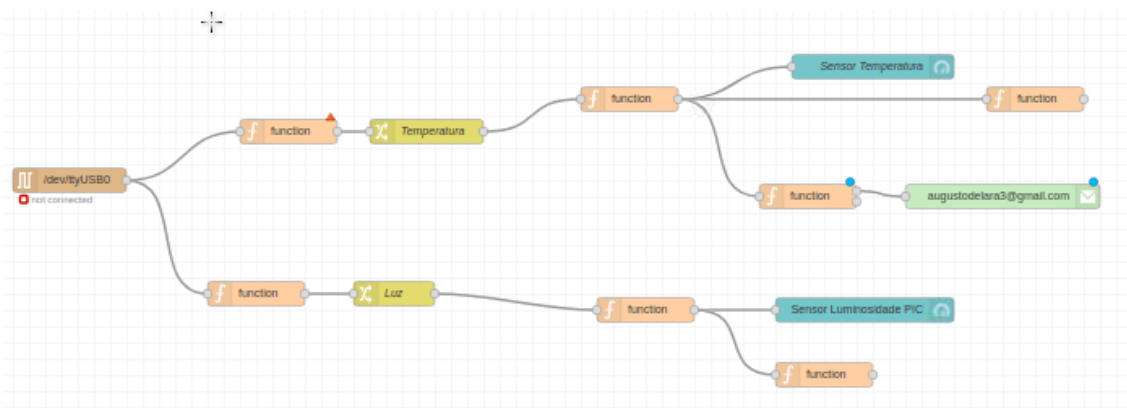


Figura 9 - Blocos desenvolvidos para o sensor de temperatura e luminosidade

Durante a inicialização é efetuada uma função que verifica o payload e direciona a temperatura e luminosidade para seus respetivos fluxos.

Quando estes dados passam para a segunda função, eles são salvos em variáveis globais.

Salvando a temperatura em uma variável global com o nome temperatura.

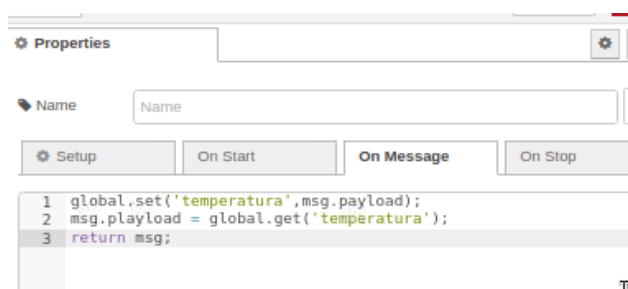


Figura 10 - Código da função que salva a temperatura em uma variável global

O mesmo ocorre para o fluxo de luminosidade.

Salvando a luminosidade em uma variável global com o nome luminosidade.

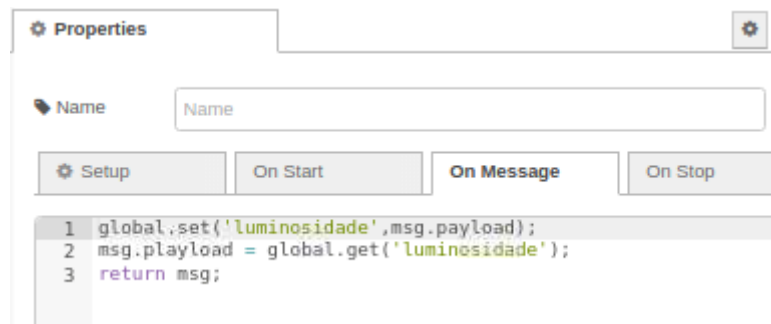


Figura 11 - Código da função que salva a luminosidade em uma variável global

A temperatura tem uma lógica extra que notifica através de um email quando a temperatura passa de 30 graus.

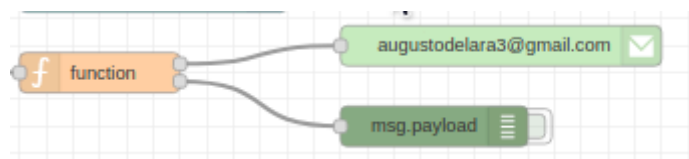


Figura 12 - Bloco node-red com a lógica para envio do email de alerta

6 Código para o envio da notificação.



Figura 13 - Função lógica para envio do email de alerta

7 Emails recebidos com as notificações.

☆ > eu	Message from Node-RED - Temperature too high: 30.4	14 de jun.
☆ > eu	Message from Node-RED - Temperature too high: 31.2	14 de jun.
☆ > eu	Message from Node-RED - Temperature too high: 30.4	14 de jun.

Figura 14 - E Mails de alerta recebidos.

8 Implementação Broker e Node-RED

O broker que utilizamos foi o Aedes Broker, ele é de simples instalação e possui integração com o Node-Red, a porta foi aberta em 1883.

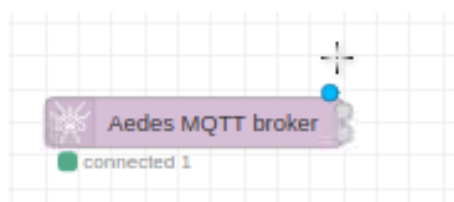


Figura 15 - Broker Aedes utilizado no Node-Red

Pisca LED raspberry node-red

Abaixo encontra-se outra implementação realizada, a de Piscar o LED 0(green) da Raspberry Pi através do Broker.

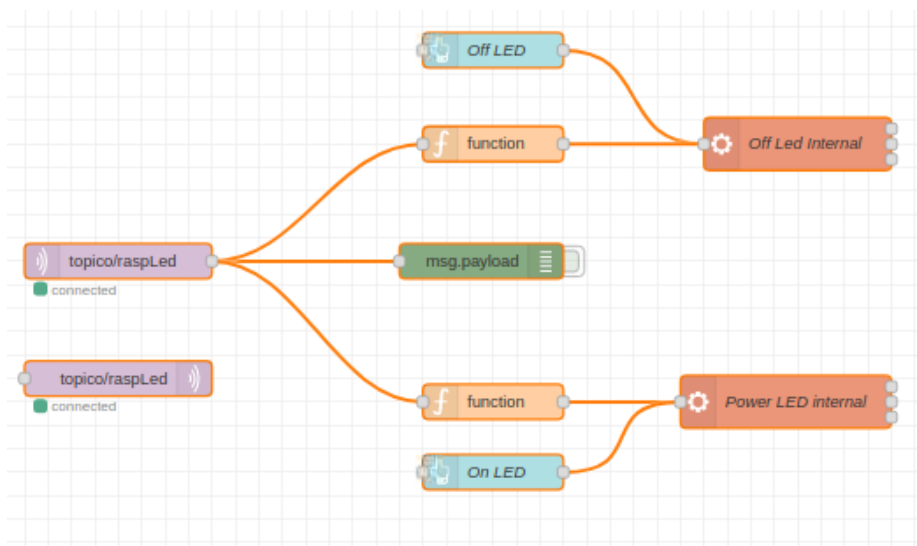


Figura 16 - Blocos desenvolvidos para ligar e desligar led na raspberry Pi.

O código funciona da seguinte maneira, sempre que o cliente envia uma mensagem com OFF ou ON para o tópico raspLed, é realizada uma ação.

```

1 = if(msg.payload == "OFF"){
2   return msg;
3 = }
4

```

Figura 17 - Código da função que verifica o payload enviado através do broker.

A execução é realizada por um comando Linux, que é executado diretamente na Raspberry, fazendo com o LED acenda quando 1 e apague quando 0.

Properties

Command: `sudo sh -c 'echo 0 > /sys/class/leds/led0/brightness'`

Append: ☒ msg. payload

extra input parameters:

Output: when the command is complete - exec mode

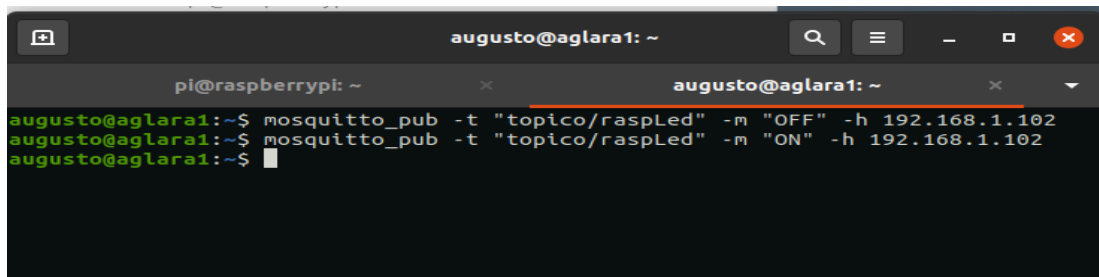
Timeout: optional seconds

Hide console: ☐

Name: Off Led Internal

Figura 18 - Execução do comando para desligar o Led.

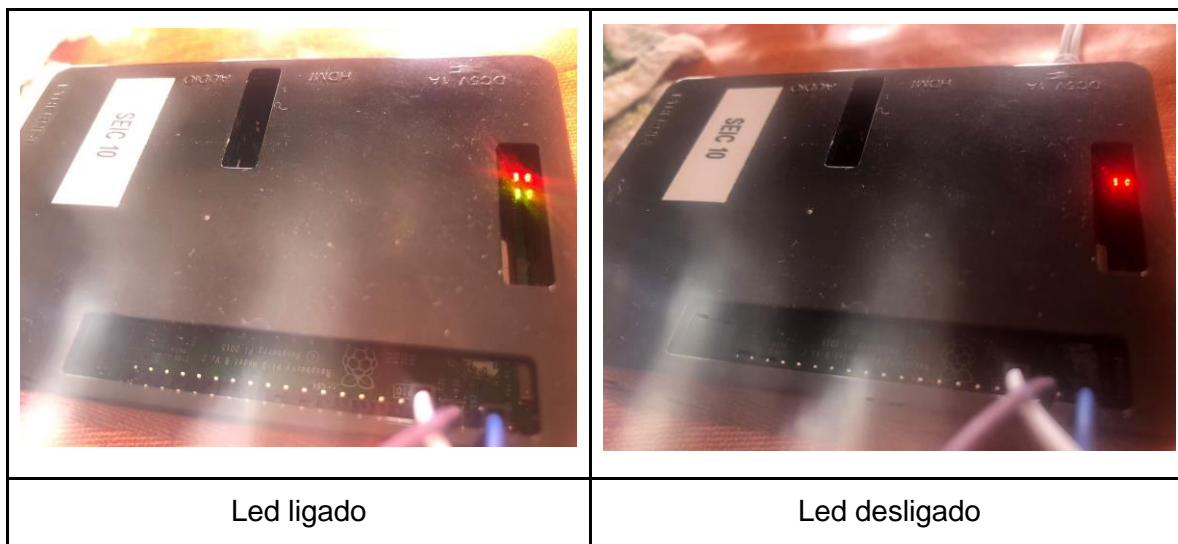
Controle do Led usando o protocolo mqtt através do mosquitto em computador local.



```

augusto@aglara1: ~
pi@raspberrypi: ~
augusto@aglara1:~$ mosquitto_pub -t "topico/raspLed" -m "OFF" -h 192.168.1.102
augusto@aglara1:~$ mosquitto_pub -t "topico/raspLed" -m "ON" -h 192.168.1.102
augusto@aglara1:~$
  
```

Figura 19 - Execução de pub através de outro computador na mesma rede local.



Outra implementação realizada foi o controle do Led pela UI do Node-Red através de botões.

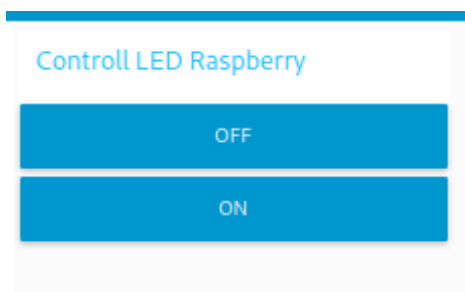


Figura 20 - Controle do led pela UI

9 SaveData

Para salvar os dados em tempo real, é necessário habilitar o saveData através do broker, se o broker não tiver o save de dados como ON, os mesmo não serão salvos

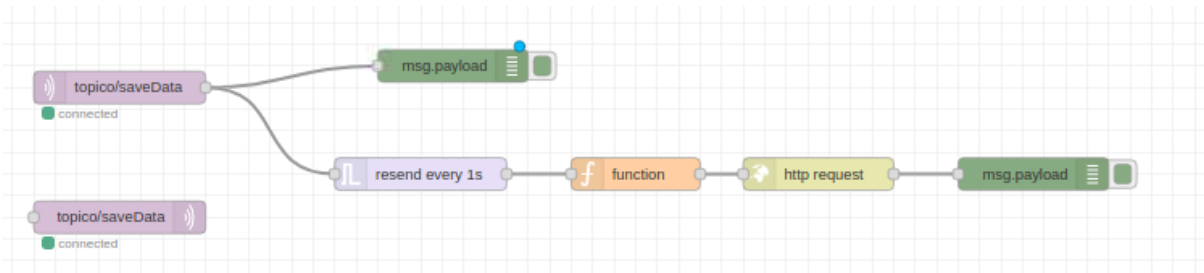


Figura 21 - Blocos desenvolvidos para salvar os dados na nuvem.

Antes da função ser executada, temos um trigger que recebe o payload do broker e replica a mensagem a cada 1 segundos, desta maneira controlamos a atividade das requisições.

Durante a função são requisitadas as variáveis globais, e as mesmas são repassadas na mensagem que será enviada para a planilha no google sheets.

```

1 = if(msg.payload == "ON"){
2   msg.payload = {};
3   var temp = global.get('temperatura');
4   var lum = global.get('luminosidade');
5   var hum = global.get('humidade');
6   var temp2 = global.get('tempEsp');
7
8   msg.payload.temperature = temp;
9   msg.payload.lum1 = lum;
10  msg.payload.hum = hum;
11  msg.payload.tempEsp = temp2;
12  return msg;
13 }
14

```

Figura 22 - Código da função que busca os dados e envia para a url request.

Estas variáveis são repassadas na URL, contendo os parâmetros de temperatura e luminosidade, Pega os valores das variáveis globais e passam para o http request.

https://docs.google.com/forms/d/e/1FAIpQLSc9Jpbx3n-H8gOhj_ULpdvFXhv4cc0x4hPVauqocFG5G4Zw_A/formResponse?usp=pp_url&entry.1237479476={{payload.temperature}}&entry.900887157={{payload.lum}}&entry.1238241947={{payload.hum}}&entry.1520359710={{payload.tempEsp}}

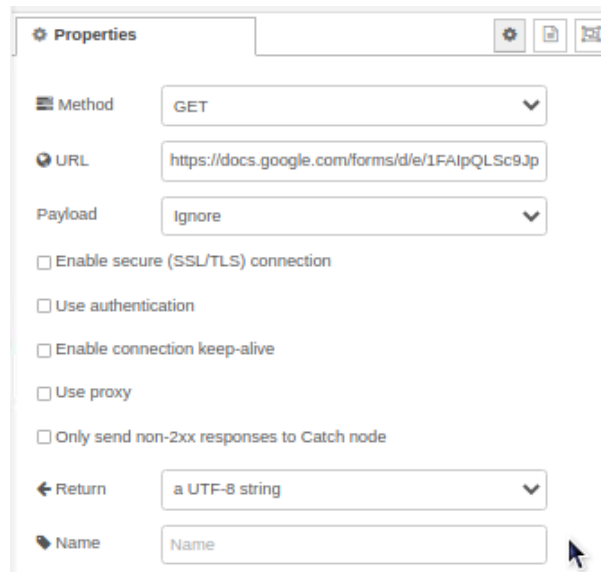


Figura 23 - Execução URL request.

Para isso foram necessárias uma autorização de segurança e a criação de uma senha de app.

← Senhas de app

Senhas de app permitem que você faça login na sua Conta do Google a partir de apps em dispositivos que não sejam compatíveis com a verificação em duas etapas. Como só será necessário informar a senha uma vez, você não precisa memorizá-la. [Saiba mais](#)


Suas senhas de app		
Nome	Criada	Usada pela última vez em
Linux	14 de jun.	14 de jun. 

Figura 24 - Senha de app na conta do Google.

Controlando quando serão salvos os dados através de um publish no tópico saveData.

```
augusto@aglara1: ~
augusto@aglara1: ~$ mosquitto_pub -t "topico/saveData" -m "ON" -h 192.168.1.102
augusto@aglara1: ~$ mosquitto_pub -t "topico/saveData" -m "OFF" -h 192.168.1.102
augusto@aglara1: ~$
```

Figura 25 - Ativando saveData através do mosquitto_pub

TemLog ☆ 📁 ☁

Arquivo Editar Ver Inserir Formatar Dados Ferramentas Extensões

100% R\$ % .0 .00 123 Padrão (Ari... 10

A1 \sum Carimbo de data/hora

	A	B	C	
1	Carimbo de data/hora	Temp	Lum	
137	16/06/2022 15:37:15	23.2	406.0	
138	16/06/2022 15:37:16	23.2	406.0	
139	16/06/2022 15:37:17	23.2	406.0	
140	16/06/2022 15:37:18	23.2	406.0	
141	16/06/2022 15:37:19	23.2	406.0	
142	16/06/2022 15:37:20	23.2	406.0	
143	16/06/2022 15:37:21	23.2	406.0	
144	16/06/2022 15:37:22	23.2	406.0	
145	16/06/2022 15:37:23	23.2	406.0	
146	16/06/2022 15:37:24	23.2	406.0	
147	16/06/2022 15:37:25	23.2	406.0	
148	16/06/2022 15:37:26	23.4	406.0	
149	16/06/2022 15:37:27	23.4	406.0	
150	16/06/2022 15:37:28	23.4	406.0	
151	16/06/2022 15:37:29	23.4	365.0	
152	16/06/2022 15:37:30	23.4	365.0	
153	16/06/2022 15:37:31	23.1	365.0	
154	16/06/2022 15:37:32	23.1	360.0	
155	16/06/2022 15:37:33	23.1	360.0	
156	16/06/2022 15:37:34	23.1	360.0	
157	16/06/2022 15:37:35	23.1	360.0	
158	16/06/2022 15:37:36	23.3	360.0	
159	16/06/2022 15:37:37	23.3	360.0	
160	16/06/2022 15:37:38	23.3	360.0	
161	16/06/2022 15:37:39	23.3	406.0	

Figura 26 - Dados sendo salvos na planilha online a cada 1 segundo.

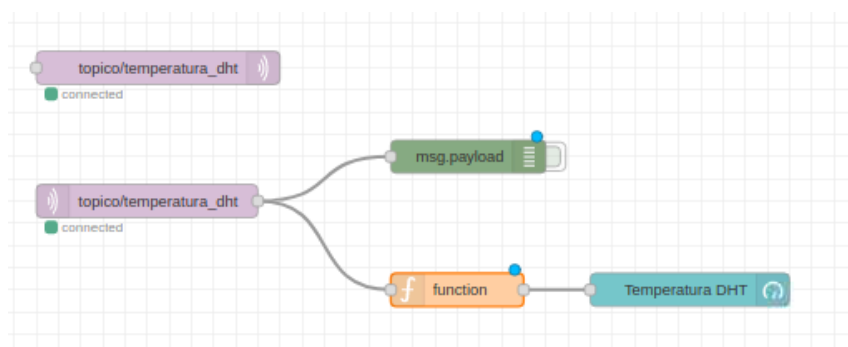


Figura 27 - Topico e publisher temperatura DHT11

10 Comunicação ESP32 BROKER

Elaboramos uma comunicação externa entre a ESP32 e o Broker presente na Raspberry Pi. Ligado a ESP32 temos um sensor DHT11, ao qual extraímos informações de humidade e temperatura.

Primeiramente desenvolvemos os tópicos no node-red, intitulados tópico/hum, e tópico/temperatura_dht.

Para cada um deles temos um função que captura o valor da msg.payload e salva em um variável global, está variável é utilizada mais tarde para salvar na planilha em que se encontram os valores no decorrer do tempo.

Os valores também são mostrados na dashboard do node-red.

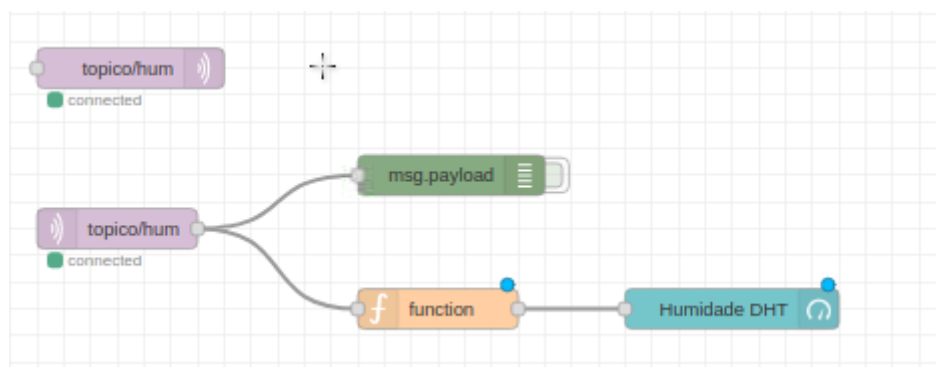


Figura 28 - Topico e Publisher Humidade DH11

Carimbo de data/hora	TemperaturaPIC	LuminosidadePIC	HumidadeESP	TemperaturaESP
20/07/2022 23:05:42	23.2	406.0	33	26
20/07/2022 23:05:44	23.4	406.0	33	26
20/07/2022 23:05:46	23.4	406.0	33	26
20/07/2022 23:05:48	23.4	406.0	33	26
20/07/2022 23:05:50	23.4	365.0	33	26
20/07/2022 23:05:52	23.4	365.0	33	26
20/07/2022 23:05:54	23.1	365.0	33	26
20/07/2022 23:05:56	23.1	360.0	33	26
20/07/2022 23:05:58	23.1	360.0	33	26
20/07/2022 23:06:00	23.1	360.0	33	26
20/07/2022 23:06:02	23.1	360.0	33	26
20/07/2022 23:06:04	23.3	360.0	33	26

Figura 29 - Dados extraídos salvos na Cloud em tempo real

Conforme mencionado, caso a o tópico saveData esteja habilitado os dados serão salvos em tempo real na planilha de dados.

```
void loop() {  
  //delay(5000);  
  Sensor_humidade();  
  client.loop();  
}  
  
void Sensor_humidade() {  
  hum = dht.readHumidity();  
  tem = dht.readTemperature();  
  
  while (isnan(hum)) {  
    Serial.println("Falha de leitura");  
    delay(2000);  
    hum = dht.readHumidity();  
    tem = dht.readTemperature();  
  }  
  
  Serial.print("Humidade:");  
  Serial.print(hum);  
  Serial.println("%.");  
  
  Serial.print("Temperatura:");  
  Serial.print(tem);  
  
  Serial.println("-----");  
  
  delay(5000);  
  hum_str = String(hum); //converting Humidity (the float variable above) to a string  
  hum_str.toCharArray(hum_char, hum_str.length() + 1); //packaging up the data to publish to mqtt whoa...  
  client.publish("topico/hum",hum_char);  
  
  tem_str = String(tem); //converting Humidity (the float variable above) to a string  
  tem_str.toCharArray(tem_char, tem_str.length() + 1); //packaging up the data to publish to mqtt whoa...  
  client.publish("topico/temperatura_dht",tem_char);  
}
```

Figura 30 - Parte de código desenvolvido para a ESP32

Para a implementação na ESP32 utilizamos o Arduino IDE, nele através de exemplos elaboramos a captura das informações no sensor DHT11, e a partir da extração dos dados enviamos através do client.publish para o respetivos tópicos, tópico/hum e tópico/temperatura_dht.

11 Conclusão

Durante a realização deste trabalho tivemos a oportunidade de aprender novas tecnologias de comunicação (UART, MQTT, SSH) entre diferentes periféricos, podendo assim interagir com o microcontrolador PIC, Raspberry e ESP32. Grande parte dos ensinamentos adquiridos em aula foram utilizados para a realização do trabalho, além de pesquisas complementares e auxílios durante as aulas.

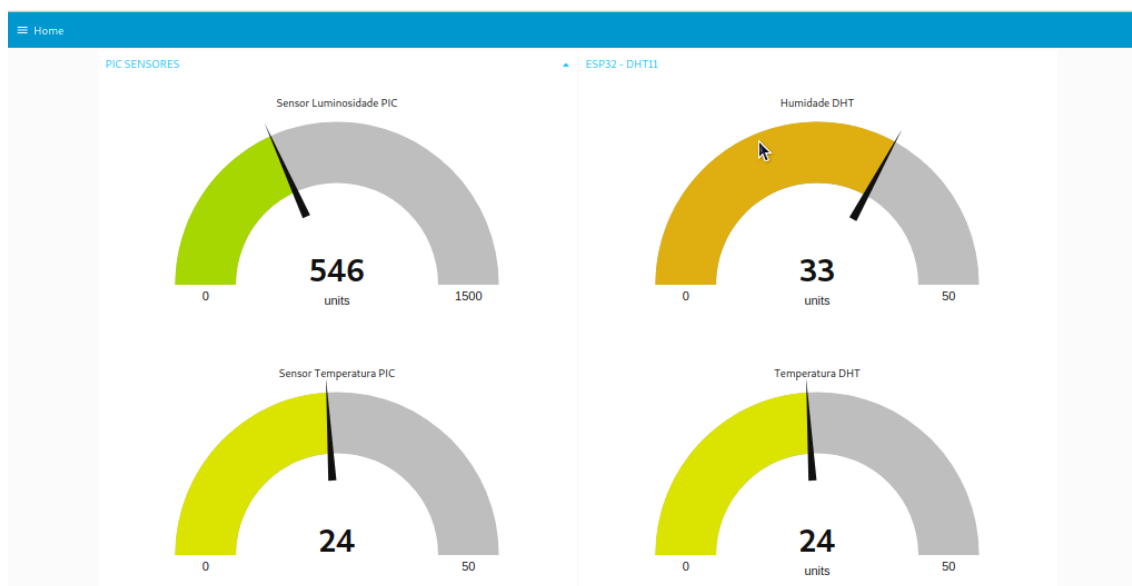


Figura 31 - Gráficos gerados na dashboard

De tal modo pode se verificar a leitura dos três sensores na qual conseguimos extrair quatro informações distintas, através de protocolos distintos UART e MQTT.

Pode-se observar na imagem acima, as quatro informações, temperatura LM35, luminosidade LDR e DHT11(Humidade e Temperatura).

12 Bibliografia

Pi4Iot, Raspberry Pi with Node-RED Tutorial #3 – unix command with exec node. 1 vídeo(6:01m). Disponível em:

https://www.youtube.com/watch?v=TIPOQHjNgwk&ab_channel=Pi4IoT.

Acesso em: 10 de junho de 2022.

Steve Cope, How to Run System Commands In Node-Red. 1 vídeo(10:34m). Disponível em:

https://www.youtube.com/watch?v=xcRkOH_0uek&ab_channel=SteveCope.

Acesso em: 10 de junho de 2022.

Raspberrypi Stackexchange , “How do I control the system LEDs using my software?”,

Disponível em: <https://raspberrypi.stackexchange.com/questions/697/how-do-i-control-the-system-leds-using-my-software> , 2022.

Raspberrypi Stackexchange , “Turning off LEDs on Raspberry Pi 3”, Disponível em:

<https://raspberrypi.stackexchange.com/questions/70593/turning-off-leds-on-raspberry-pi-3> , 2022.

Community Element14 , “New Raspberry-Pi 3, Green-LED keeps blinking.”, Disponível em:

<https://community.element14.com/products/raspberry-pi/f/forum/30811/new-raspberry-pi-3-green-led-keeps-blinking> , 2022.

Hardik Rathod, HPIR Motion Sensor with Raspberry Pi 2. 1 vídeo(10:34m). Disponível em:

https://www.youtube.com/watch?v=_ACe6z-Hp2E&ab_channel=HardikRathod.

Acesso em: 10 de junho de 2022.

Opto22 , “Sending an email attachment with Node-RED.”, Disponível em:

<https://forums.opto22.com/t/sending-an-email-attachment-with-node-red/3360> , 2022.

Nodered , “node-red-contrib-email-out.”, Disponível em:

<https://flows.nodered.org/node/node-red-contrib-email-out> , 2022.

I AM Z Studio, Insert Data to Google Sheet Autoatic with Node Red. 1 vídeo(10:11m).

Disponível em: https://www.youtube.com/watch?v=GKZ9Ro_3-ik.

Acesso em: 10 de junho de 2022.

thesharanmohanblog , “IoT: Sending Email Notifications using Node-RED”, Disponível em:

<https://thesharanmohanblog.wordpress.com/2020/04/27/iot-sending-email-notifications-using-node-red/> , 2022.

AnotherMaker, Node Red and MQTT will change your life (1 of 2). 1 vídeo(13:49m). Disponível

em: https://www.youtube.com/watch?v=B_4dvclsDRo&ab_channel=AnotherMaker.

Acesso em: 10 de junho de 2022.

pimylifeup , “Raspberry Pi Humidity Sensor using the DHT22.”, Disponível em:

<https://pimylifeup.com/raspberry-pi-humidity-sensor-dht22/> , 2022.

circuitbasics , “HOW TO SET UP THE DHT11 HUMIDITY SENSOR ON THE RASPBERRY PI.”,

Disponível em:

<https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-the-raspberry-pi/> , 2022.