

Universidade Católica de Pelotas  
(UCPel) Centro de Ciências Sociais e  
Tecnológicas Engenharia de  
Computação

# Projeto Esp32 Leitor de Dispositivos para Esp32

Aluno: Bruno Lopes Soares  
Professor: Adenauer

Pelotas  
2020  
Universidade Católica de Pelotas  
(UCPel) Centro de Ciências Sociais e  
Tecnológicas Engenharia de  
Computação

## Projeto Esp32 Leitor de Dispositivos para Esp32

Relatório do projeto apresentado ao Centro de Ciências Sociais e Tecnológicas do Curso de Engenharia de Computação da Universidade Católica de Pelotas (UCPel), como requisito para aproveitamento e conclusão da disciplina de Robótica.

Aluno: Bruno Lopes Soares

Professor: Adenaure

Pelotas

2020

## Introdução :

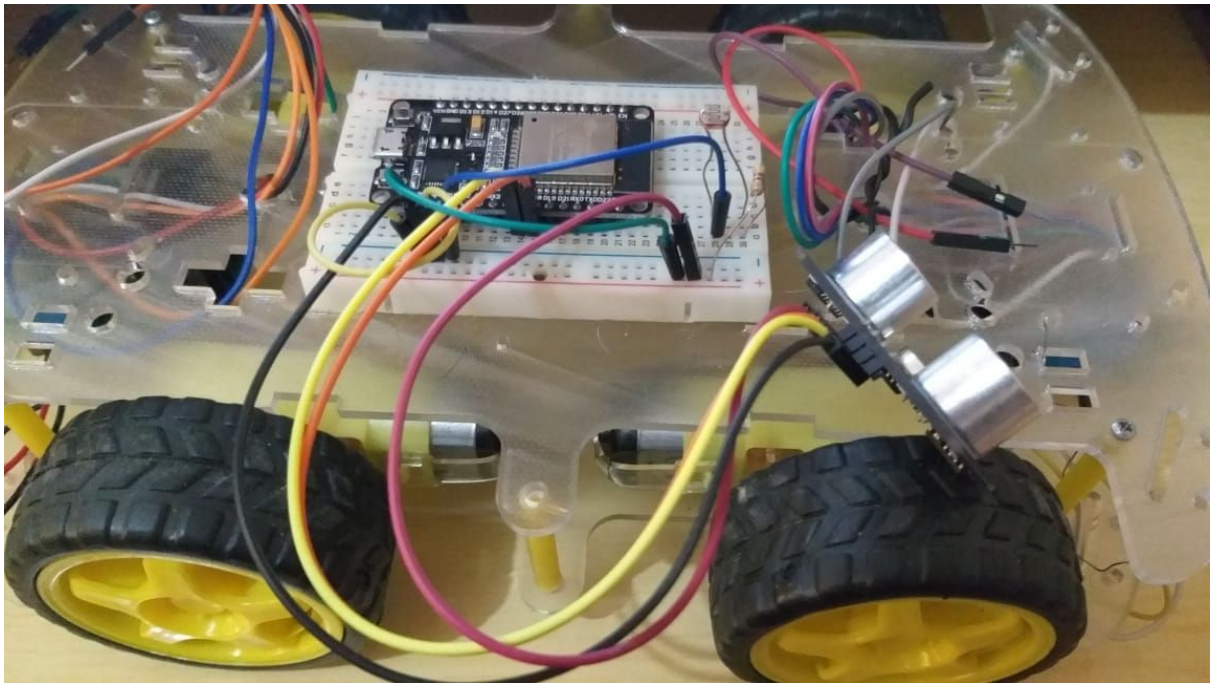
- **Motivação:**

A motivação do projeto seria bota em pratica os conhecimento de PHP para poder se comunicar com um embarcado

- **Objetivos**

Fazer a leitura de sensores e outros dispositivos da Esp32 com uma pagina Web, onde ela ira mostras os valores dos sensores capturados, acender e apagar dispositivos.

## Composição do Projeto:



- Figura acima com a arquitetura mostrando os "componentes" que formam o projeto;

- ## Esp32:

**PINOUT**

Pin	Function
1	3.3V
2	RESET
3	EN
4	ADC0
5	GPIO36
6	ADC3
7	GPIO39
8	ADC6
9	GPIO34
10	ADC7
11	GPIO35
12	TOUCH9
13	ADC4
14	GPIO32
15	TOUCH8
16	ADC5
17	GPIO33
18	DAC1
19	ADC18
20	GPIO25
21	DAC2
22	ADC19
23	GPIO26
24	TOUCH7
25	ADC17
26	GPIO27
27	SPI CLK
28	TOUCH6
29	ADC16
30	GPIO14
31	SPI MISO
32	TOUCH5
33	ADC15
34	GPIO12
35	GND
36	GPIO13
37	SPI MOSI
38	TOUCH4
39	ADC14
40	GPIO9
41	RX1
42	GPIO10
43	TX1
44	GPIO11
45	Vin
46	EN

**NOTE:**  
All pin supported PWM and I2C  
Pin current 6mA (Max. 12mA)

**ESP-WROOM-32**  
CE1313  
FCC ID 2AC7Z-ESPWROOM32

**Pinout (Right Side):**

Pin	Function
1	GND
2	GPIO23
3	GPIO22
4	TX0
5	GPIO1
6	RX0
7	GPIO3
8	GPIO21
9	GND
10	GPIO19
11	GPIO18
12	GPIO5
13	GPIO17
14	GPIO16
15	GPIO4
16	ADC10
17	TOUCH0
18	GPIO0
19	ADC11
20	TOUCH1
21	GPIO2
22	ADC12
23	TOUCH2
24	GPIO15
25	ADC13
26	TOUCH3
27	SPI SS
28	GPIO8
29	GPIO7
30	GPIO6

**Additional Labels:**

- VSPI MOSI
- I2C SCL
- I2C SDA
- VSPI MISO
- VSPI SCK
- VSPI SS
- TX2
- RX2
- Power
- Connected to GPIO2

## ■ Portas GPIO

- **Portas Touch**

Esses pinos funcionam como sensores capacitivos, ou seja, reagem quando há uma alteração em seu estado. Um simples exemplo é tocá-lo: isso iniciará um processo de troca de calor entre dois corpos distintos, onde o sensor perceberá e emitirá sinais para a ESP. Essa placa possui 10 pinos touch.

- ADC

Os ADC (*Analog Digital Converter*), são famosos na área da eletrônica. Eles convertem grandezas analógicas (velocímetro de ponteiro do carro, por exemplo) em grandezas digitais (um velocímetro digital, por exemplo). Segundo o diagrama do ESP, ele possui 16 pinos com essa capacidade.

- Demais pinos

O NodeMCU ESP32 conta com 3 portas GND, 1 porta de 3.3V e 1 porta de 5V (que no diagrama está identificada por Vin, visto que pode ser usada para alimentação da placa caso ela não esteja conectada pelo cabo USB). Há também outros pinos, como o RX e TX, CLK, MISO e outros, que serão explorados em outros artigos aqui no Blog. Note que alguns deles acumulam diversas funções em um pino só.

## Sensor Ultrassônico:



<https://blog.eletrogate.com/sensor-ultrassonico-hc-sr04-com-arduino/>

O princípio de funcionamento do HC-SR04 consiste na emissão de sinais ultrassônicos pelo sensor e na leitura do sinal de retorno (reflexo/eco) desse mesmo sinal. A distância entre o sensor e o objeto que refletiu o sinal é calculada com base no tempo entre o envio e leitura de retorno.

***” Sinais Ultrassônicos são ondas mecânicas com frequência acima de 40 KHz“***

Como ouvido humano só consegue identificar ondas mecânicas até a frequência de 20KHz, os sinais emitidos pelo sensor Ultrassônico não podem ser escutados por nós.

O **sensor** HC-SR04 é composto de três partes principais:

- **Transmissor Ultrassônico** – Emite as ondas ultrassônicas que serão refletidas pelos obstáculos;
- Um receptor – Identifica o eco do sinal emitido pelo transmissor;
- Circuito de controle – Controla o conjunto transmissor/receptor, calcula o tempo entre a emissão e recepção do sinal;

## Sensor LDR:

O LDR (Light Dependent Resistor) é um componente cuja resistência varia de acordo com a intensidade da luz. Quanto mais luz incidir sobre o componente, menor a resistência. Este sensor de luminosidade pode ser utilizado em projetos com arduino e outros microcontroladores para alarmes, automação residencial, sensores de presença e etc.



<https://www.vidadesilicio.com.br/sensor-de-luminosidade-ldr-5mm>

## Desktop:

### Edição do Windows

---

Windows 10 Home Single Language

© 2019 Microsoft Corporation. Todos os direitos reservados.

### Sistema

---

Processador:	Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHz
Memória instalada (RAM):	8,00 GB (utilizável: 7,87 GB)
Tipo de sistema:	Sistema Operacional de 64 bits, processador com base em x64
Caneta e Toque:	Nenhuma Entrada à Caneta ou por Toque está disponível para este vídeo

- **Componentes de Software:**

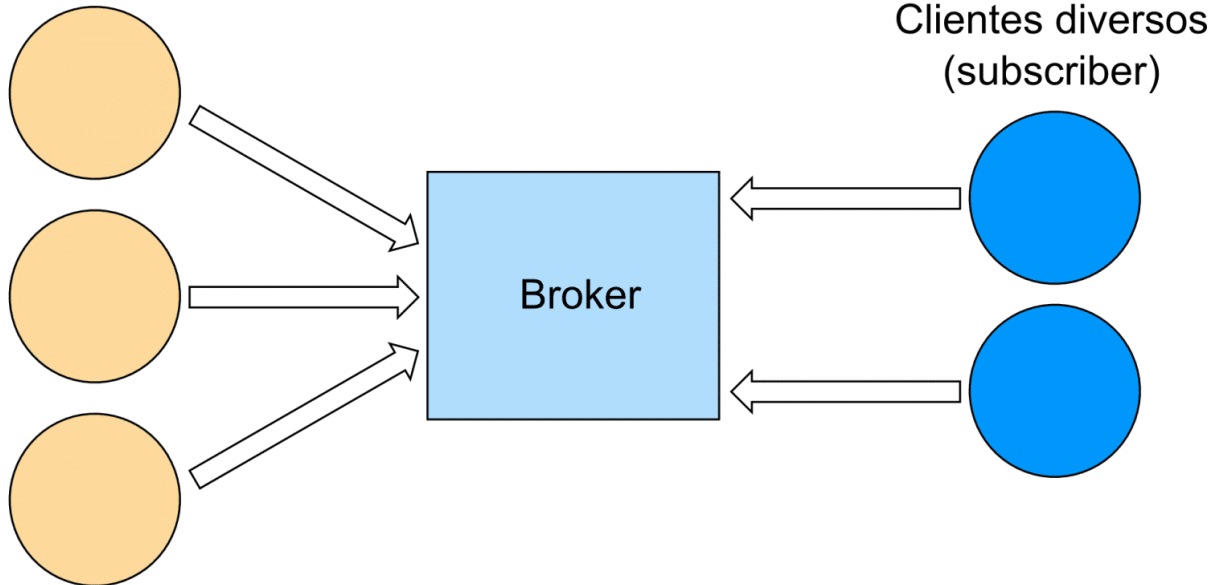
#### **MQTT:**

Um dos protocolos de troca de mensagens para IoT em uso recente é o **MQTT** (*Message Queue Telemetry Transport*). Criado pela **IBM** no final da década de 90, obviamente o protocolo carrega muito do cenário de uso original, mais voltado e adaptado para sistemas de supervisão e coleta de dados do tipo **SCADA** (*Supervisory Control and Data Acquisition* ou Sistemas de Supervisão e Aquisição de Dados, em português). Mas, mesmo assim, o MQTT encontrou seu espaço nesse amplo mercado de IoT.

O MQTT não é tão sofisticado quanto o **AMQP** (*Advanced Message Queuing Protocol*), que possui mais funcionalidades e cenários de uso, mas é simples o suficiente sem deixar de contemplar características como segurança, qualidade de serviço e facilidade de implementação. Essas características fazem do MQTT um bom candidato para implementações e usos em sistemas embarcados, embora a competição seja acirrada.



Sensores  
(publish)



<https://www.embarcados.com.br/mqtt-protocolos-para-iot/>

## Descrição do Projeto

- Banco de Dados

Nesse projeto foi utilizado o banco de dados do MyAdmin e nele foi criando o **banco bootstrap\_projeto** e nele foi criando **tabela ultrasonico** para guarda os dados do sensor ultrassônico e uma **tabela sensor\_ldr** para guarda o os dados do sensor LDR.



- Tabela ultrasonico: possui coluna para guarda ID, nome, Valor, Data inicial, data Final e Estado;
- Onde:
  - nome = é uma descrição para identificar nome do projeto;
  - Valor = valor que está lendo do sensor;
  - Data inicial = data que ira iniciar leitura do sensor;
  - Data Final = data que ira Encerrar a leitura do sensor;
  - Estado: mostra se o sensor esta ligado ou desligado;

The screenshot shows the phpMyAdmin interface with the 'ultrasonico' table selected. The table structure is visible at the top, and the data is displayed in a table below. The table has columns: 'id', 'nome', 'Valor', 'data Inicio', 'data Final', and 'Estado'. The data is filtered to show 25 records.

	id	nome	Valor	data Inicio	data Final	Estado
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	41	cj	698	2020-12-07 12:00:00	2021-04-19 17:45:12	OFF
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	168	50	0.03436426	0000-00-00 00:00:00	0000-00-00 00:00:00	
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	169	50	0.03436426	0000-00-00 00:00:00	0000-00-00 00:00:00	
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	170	50	0.03436426	0000-00-00 00:00:00	0000-00-00 00:00:00	
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	171	50	0.05154639	0000-00-00 00:00:00	0000-00-00 00:00:00	
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	172	50	0.03436426	0000-00-00 00:00:00	0000-00-00 00:00:00	
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	173	50	0.06872852	0000-00-00 00:00:00	0000-00-00 00:00:00	
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	174	20	1.26385	0000-00-00 00:00:00	0000-00-00 00:00:00	
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	175	20	1.0143	0000-00-00 00:00:00	0000-00-00 00:00:00	
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	176	20	0.2737	0000-00-00 00:00:00	0000-00-00 00:00:00	

- Tabela sensor\_ldr: possui coluna para guarda ID, nome, Valor, Data inicial, data Final e Estado;
- Onde:
  - nome = é uma descrição para identificar nome do projeto;
  - Valor = valor que está lendo do sensor;
  - Data inicial = data que ira iniciar o sensor;
  - Data Final = data que ira Encerrar o sensor;
  - Estado: mostra se o sensor esta ligado ou desligado;

phpMyAdmin

Servidor: 127.0.0.1 » Base de Dados: bootstrap\_projeto » Tabela: sensor\_ldr

Procurar Estrutura SQL Pesquisar Inserir Exportar Importar

✓ A mostrar registos de 0 - 6 (7 total, A consulta demorou 0,0009 segundos.)

SELECT \* FROM `sensor\_ldr`

Mostrar tudo | Número de registos: 25 | Filtrar registos: Pesquisar esta tabela | Ordenar por

+ Opções

				id	Nome	Valor	data Inicio	data Final	Estado
<input type="checkbox"/>	Edita	Copiar	Apagar	23	cj	698	2020-12-07 12:00:00	2021-04-19 17:45:12	OFF
<input type="checkbox"/>	Edita	Copiar	Apagar	37	20	0.7889	0000-00-00 00:00:00	0000-00-00 00:00:00	
<input type="checkbox"/>	Edita	Copiar	Apagar	38	20	1.5295	0000-00-00 00:00:00	0000-00-00 00:00:00	
<input type="checkbox"/>	Edita	Copiar	Apagar	39	20	0.9016	0000-00-00 00:00:00	0000-00-00 00:00:00	
<input type="checkbox"/>	Edita	Copiar	Apagar	40	20	0.3381	0000-00-00 00:00:00	0000-00-00 00:00:00	
<input type="checkbox"/>	Edita	Copiar	Apagar	41	20	0.3864	0000-00-00 00:00:00	0000-00-00 00:00:00	
<input type="checkbox"/>	Edita	Copiar	Apagar	42	20	0.5313	0000-00-00 00:00:00	0000-00-00 00:00:00	

## • APACHE

Nesse site podemos escolher os tipos dispositivos que queremos ler da esp32 e nesse caso iremos trabalhar lendo sensores Ultrassônico e LDR.

**Botão ultrasonico:** quando é clicado ele aparece os dados do ultrasonico;

**Botão LDR 5mm:** quando é clicado ele aparece os dados do LDR;



Escolha aqui um dos sensores para ser lido :



ultrasonico



LDR 5mm

Teste aqui a sua comunicação:

Desliga Liga











Dados

Caso não clicou em nenhum botão tanto o ultrasonico ou LDR 5mm então os dados não aparecer. mas caso se tu clica do ULTRASONICO por exemplo, então os dados que ira aparecer sera esses abaixo:

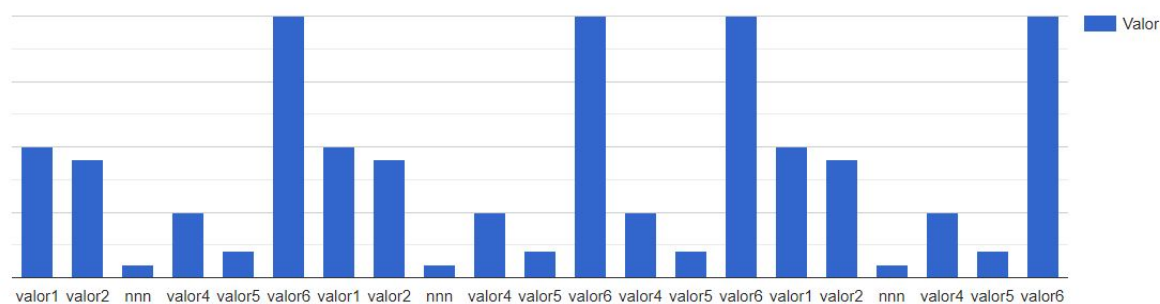
## Dados

Cadastrar Equipe:
<b>Nome do membro:</b>
<input type="text"/>
<b>Valor:</b>
<input type="text"/>
<b>Data Inicial:</b>
<input type="text" value="XXXX-XX-XX XX:XX:XX"/>
<b>Data Final:</b>
<input type="text" value="XXXX-XX-XX XX:XX:XX"/>
<b>Estado:</b>
<input type="text"/>
<input type="button" value="Submit"/>

O primeiro dado que ira aparecer sera um cadastro para cada Ultrasonico. Onde podemos trabalhar com varios sensores ultrasonicos e com isso podemos definir um horario de funcionamento para cada sensor e tambem podemos ter um estado para poder definir a situação do sensor, se ele esta desligado ou ligado .

Membros da equipe:						
ID:	Nome do membro	Valor	D_I	D_F	Estado	
41	cj	698	2020-12-07 12:00:00	2021-04-19 17:45:12	OFF	 Excluir
168	50	0.03436426	0000-00-00 00:00:00	0000-00-00 00:00:00		 Excluir
169	50	0.03436426	0000-00-00 00:00:00	0000-00-00 00:00:00		 Excluir
170	50	0.03436426	0000-00-00 00:00:00	0000-00-00 00:00:00		 Excluir
171	50	0.05154639	0000-00-00 00:00:00	0000-00-00 00:00:00		 Excluir
172	50	0.03436426	0000-00-00 00:00:00	0000-00-00 00:00:00		 Excluir
173	50	0.06872852	0000-00-00 00:00:00	0000-00-00 00:00:00		 Excluir
174	20	1.26385	0000-00-00 00:00:00	0000-00-00 00:00:00		 Excluir
175	20	1.0143	0000-00-00 00:00:00	0000-00-00 00:00:00		 Excluir
176	20	0.2737	0000-00-00 00:00:00	0000-00-00 00:00:00		 Excluir

E acima é uma tabela mostrando os sensores cadastrados e com seus dados ;



E também teria um gráfico mostrando um histórico da situação dos sensores;

OBS: para botão LDR 5mm teria os mesmos dados mas para o sensor LDR;

## **Funcionalidades do projeto:**

### **Funcionalidade:**

- tem que ter um mecanismo de teste para poder testar se a comunicação com a Esp32 esta ocorrendo;
- Deve definir o horário de funcionamento;
- Se horário do computador estiver dentro do horário definido, deve mostra uma aviso;
- O gráfico vai ser dinâmico de acordo com valores dos sensores;
- Caso não estive ocorrendo a comunicação com a Esp32, então um aviso de erro deve ocorrer;
- valores do sensores será gravada no banco de dados MySQL;

### **Entradas:**

- Definir o numero de colunas que terá o gráfico, e esse gráfico ira mostra o valor do sensores;
- Definir o horário de leitura do sensor que ira aparecer no gráfico;

### **Saídas:**

- Valor do sensor ira passar para o PHP;
- os Horarios de cada sensor sera visualizada no PHP;



## Implementação:

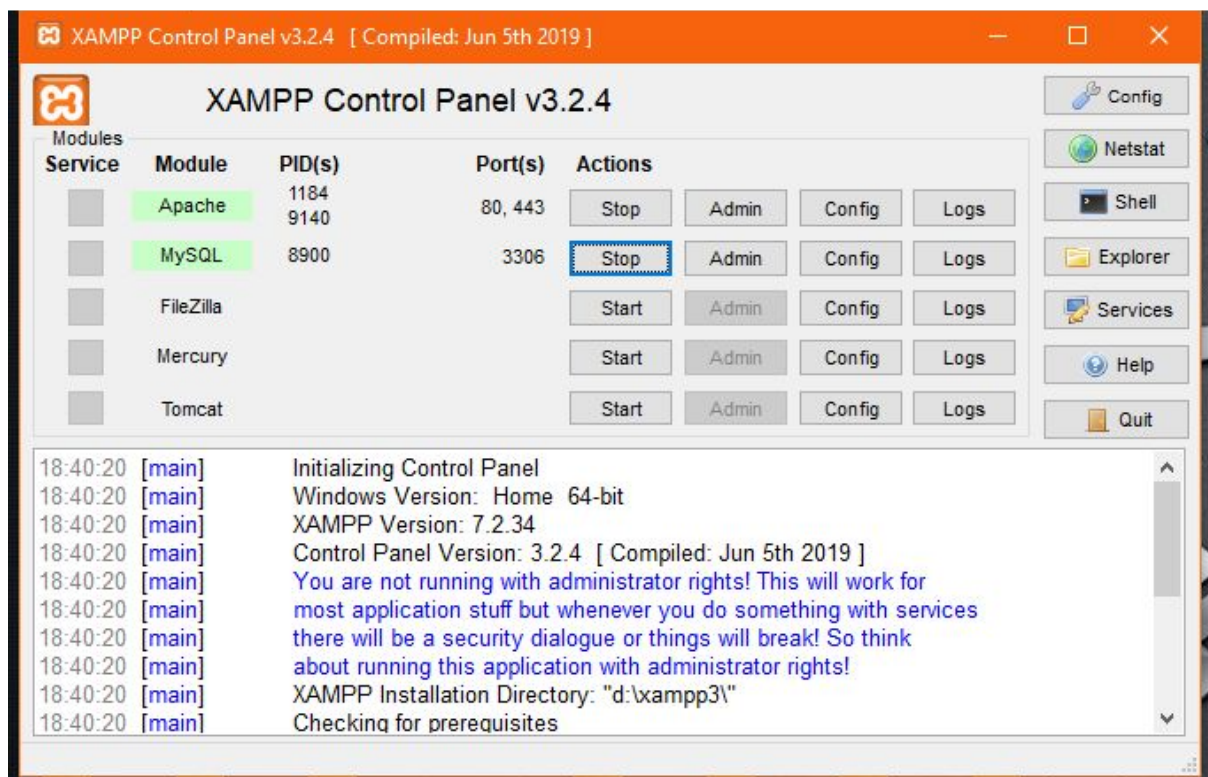
Foi utilizado PHP para poder mostra os dados do sensores para aparecer no APACHE, banco MyAdmin, Servidor XAMPP que faz roda o APACHE e também a linguagem Python e microPython, onde:

- **microPython:** Ler o valor do sensor e tranfere esse dado para Python e utilizando a IDE do uPyCraft para fazer a programação ;
- **Python:** pega o valor obtido pelo microPython e passa para o banco do Mysql e utilizando a IDE do Pycharm para fazer a programação ;
- **APACHE:** serve para definir o horário de funcionamento do sensores e mostra dados do banco.

OBS: para se comunicar o Micropython e Python foi utilizado o MQTT para fazer essa comunicação;

## Servidor XAMPP:

**1 passo:** Roda servidor XAMPP para poder roda APACHE e o banco MyAdmin;

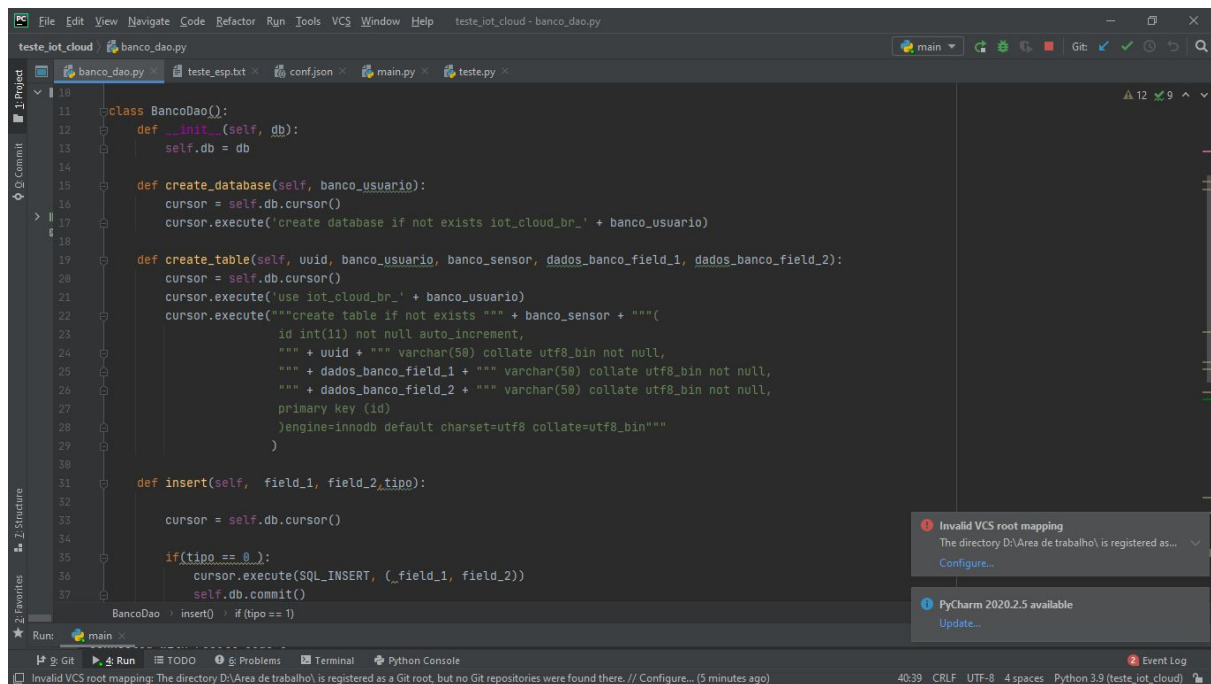




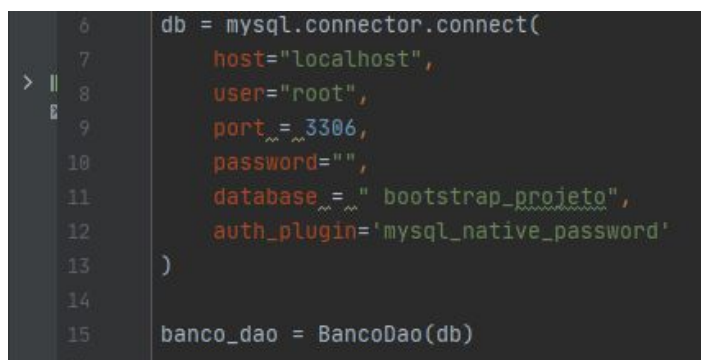
# Python:

**2 passo:** roda Python para poder ativar a transferência do microPython para banco.

O python possui um classe Banco\_dao para transferir os dados de criar banco, criar tabela, inserir dados no banco;



E possui um Main para fazer a chama da classe Banco\_dao e conectar com o banco:



A imagem acima mostra a conexão do banco;

```
18 def on_connect(client, userdata, flags, rc):
19     print("teste")
20     print("Connected with result code {}".format(str(rc)))
21     client.subscribe("iot_cloud_br/wellkamp") # título do TOPICO
22
```

A imagem a cima mostra que o MQTT está rodando Com o tópico

iot\_cloud\_br/wellkamp;

```
def on_message(client, userdata, msg): # 0 retorno de chamada para quando uma mensagem PUBLICAR é recebida do servidor.
    print("teste2")
    print("Message received-> " + msg.topic + " " + str(msg.payload)) # Print a received msg
    data_in = json.loads(msg.payload.decode('utf-8'))
    print(type(data_in))

    temperatura = data_in['nome']
    umidade = data_in['Valor']
    tipo = 1
    banco_dao.insert(temperatura, umidade, tipo)
```

e a imagem acima mostra as publicações neste tópico ;

```
Run: main x
D:\PYTHON\python.exe "D:/Area de trabalho/teste_iot_cloud/main.py"
teste
Connected with result code 0
```

E a imagem acima mostra que o Tópico com MQTT está rodando nesse momento

```
teste2
Message received-> iot_cloud_br/wellkamp b'{"nome": 20, "Valor": 4.3953}'
<class 'dict'>
teste2
Message received-> iot_cloud_br/wellkamp b'{"nome": 20, "Valor": 0.7406001}'
<class 'dict'>
```

E a imagem acima mostra que a transferência para o banco foi feita com sucesso se não essa mensagem da imagem não vai aparecer.

## MicroPython:

**3 passo:** Roda Micropython para poder ler o sensor

E para isso a imagem abaixo mostra código para se conectar com wifi

```
16
17     import network
18
19     wlan = network.WLAN(network.STA_IF)
20
21 =   if not wlan.active() or not wlan.isconnected():
22       |
23       |
24       |     wlan.active(True)
25       |
26       |     print('connecting to:', ssid)
27       |
28       |     wlan.connect(ssid, password)
29       |
30 =   while not wlan.isconnected():
31       |       |
32       |       |     pass
33       |
34       |     print('network config:', wlan.ifconfig())
35       |
36       |     print('Conectado')
37
```

E a imagem abaixo mostra o comando para publica no tópico;

```
38
39 = def conf_banco():
40     c.publish(topic=user, msg=str_convert)
41     print('Configura變借未o enviada')
42     print(str_convert)
43
44
```

E a imagem abaixo mostra a função para ler o valor do sensor, quando a variável escolha for 0 então é gravado na tabela do ultrasonico e quando for 1 é gravado na tabela do LDR;

```
45
46 =def valores_dht11( escolha ):
47     ultrasonic = ultra.HCSR04(trigger_pin=18, echo_pin=5,echo_timeout_us=1000000)
48     distance = ultrasonic.distance_cm()
49
50     led = machine.Pin(2,machine.Pin.OUT)
51
52     adc = ADC(Pin(36))
53     adc.atten(ADC.ATTN_0DB)
54     adc.width(ADC.WIDTH_12BIT)
55     ldvalue=adc.read() * 0.00805
56
57 = if(escolha == 0):
58 =     valores = {
59         "nome": 50,
60         "Valor": distance,
61     }
62     file_json = json.dumps(valores)
63
64
65 = if(escolha == 1):
66 =     valores2 = {
67         "nome": 20,
68         "Valor": ldvalue,
69     }
70     file_json = json.dumps(valores2)
71
72     return file_json
73
```

Abaixo mostra a mensagem dos valores que estão sendo transferidos para o banco, caso não apareça essa mensagem então ocorreu algum erro, caso contrário a transferência para o Python aconteceu corretamente;

```
Ready to download this file, please wait!
.....
download ok
exec(open('esp.py').read().globals())
[0:32m] (12315) phy: phy_version: 4180, cb3948U9=51 (13045) wifi:new:<6,0>, old:<1,0>, ap:<255,255>, sta:<6,0>, prof:1
I (13895) wifi:state: init -> auth (b0)
I (13905) wifi:state: auth -> assoc (0)
I (13915) wifi:state: assoc -> run (10)
I (15035) wifi:connected with BRUNO, aid = 5, channel 6, BW CONNECTED [0m
I (15095) wifi:AP's beacon interval = 102400 us, DTIM period = 1
[0:32m] (15735) event: sta ip: 192.168.0.22, :rrr.JjConectado
mensagem enviada com sucesso
{"nome": 20, "Valor": 0.21735}
mensagem enviada com sucesso
{"nome": 20, "Valor": 0.8694}
mensagem enviada com sucesso
{"nome": 20, "Valor": 1.21555}
mensagem enviada com sucesso
{"nome": 20, "Valor": 1.9159}
mensagem enviada com sucesso
```

## Conclusões :

A captura dos valores do sensor para o banco e mostra esses dados no Apache foi realizada com sucesso. Mas a comunicação do PHP com o microPython não foi concluída.

Para poder fazer que na página WEB a gente ligue e desligue um LED ou defina data que irá rodar a leitura dos sensores, era essencial a comunicação do PHP com o microPython, mas como não consegui desenvolver o código, logo a página acabou servindo para ver dados dos sensores capturados pelo sensor.

Mas pelo menos foi aprendido como enviar dados do microPython para o Python.

