

# Rede wireless

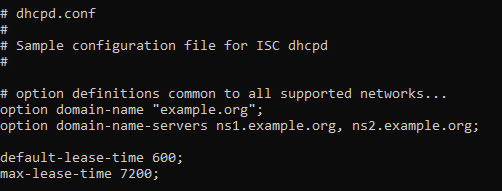
<https://cdn-learn.adafruit.com/downloads/pdf/setting-up-a-raspberry-pi-as-a-wifi-access-point.pdf>

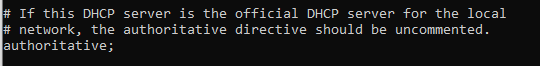
<https://raspberrypihq.com/how-to-turn-a-raspberry-pi-into-a-wifi-router/>

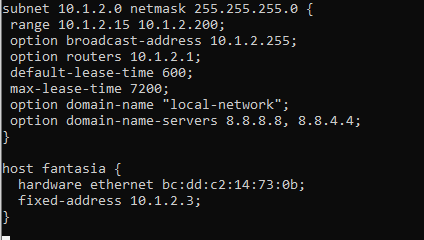
- Executar os passos dos tutorias (preferencialmente Adafruit) até ao ponto em que se faz o forwarding das comunicações para a porta LAN.

As configurações feitas são apresentadas nas imagens seguintes, de notar algumas diferenças relativamente ao tutorial:

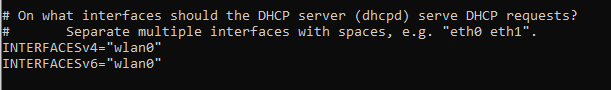
dhcpd.conf



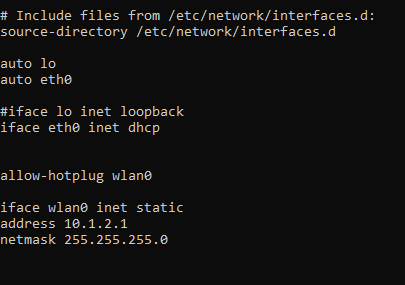




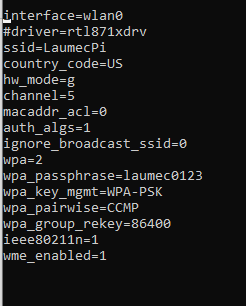
isc-dhcp-server



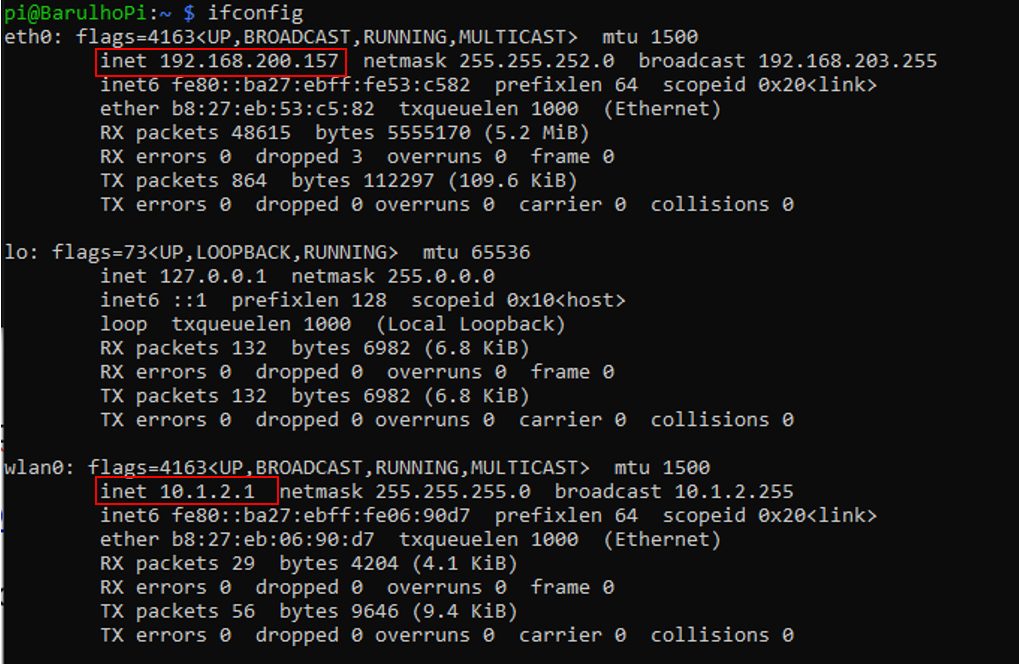
interfaces



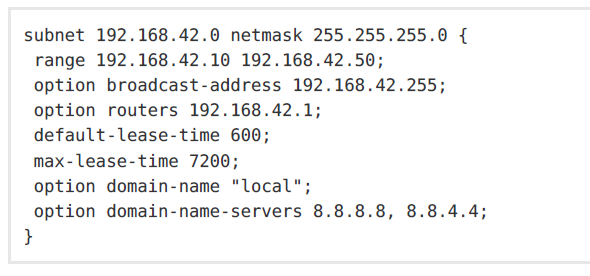
hostapd.conf



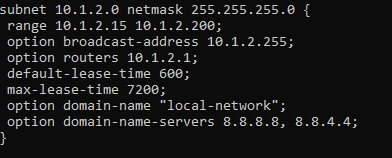
Ifconfig após todo o processo de configuração:



Atribuição dos IP’s



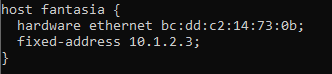
Modificar de modo a selecionar os IP da nova rede. No caso da rede criada os ip’s selecionados foram:



Ponto importante: Caso seja necessário atribuir ip’s fixos este não podem estar incluídos no range de ip’s disponíveis para atribuição pelo dhcp!

## IP’s Fixos

Para definir ip’s fixos deve-se incluir em baixo as linhas que contêm ip a atribuir e o mac adress do equipamento:



## Encontrar mac adress dos ESP

Código:

#include <ESP8266WiFi.h>

void setup(){

Serial.begin(115200);

delay(500);

Serial.println();

Serial.print("MAC:");

Serial.println(WiFi.macAddress());

}

void loop(){}

# MQTT

## Arduino IDE

As “libraries” a incluir serão:

#include "Adafruit\_MQTT.h"

#include "Adafruit\_MQTT\_Client.h"

Código:

//MQTT Variables

#define AIO\_SERVER "10.1.2.1" //"192.168.200.157" (IP eth0 do Rpi também funciona)

#define AIO\_SERVERPORT 1883 // use 8883 for SSL

#define AIO\_USERNAME ""

#define AIO\_KEY ""

const char\* topic = "TestNode"; //Topic where to write the information

void MQTT\_connect();

void setup(void)

{

// Start Serial

Serial.begin(115200);

while (!Serial) {

delay(10); // hang out until serial port opens

}

}

void loop() {

//MQTT

MQTT\_connect();

//MQTT publish

// Now we can publish stuff!

Serial.print(F("\nSending values "));

Serial.println(output);

Serial.print("...");

if (! Medido.publish(output)) {

Serial.println(F("Failed"));

} else {

Serial.println(F("OK!"));

}

}

void MQTT\_connect() {

int8\_t ret;

// Stop if already connected.

if (mqtt.connected()) {

return;

}

Serial.print("Connecting to MQTT... ");

uint8\_t retries = 3;

while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected

Serial.println(mqtt.connectErrorString(ret));

Serial.println("Retrying MQTT connection in 5 seconds...");

mqtt.disconnect();

delay(5000); // wait 5 seconds

retries--;

if (retries == 0) {

// basically die and wait for WDT to reset me

while (1);

}

}

Serial.println("MQTT Connected!");

}

## JSON

Ferramenta para criar json e determinar o numero de bytes a reservar em memoria. Gera também o código do json que for introduzido. Para usar com o ArduinoJson 5.13.5

<https://arduinojson.org/v5/assistant/>

IMPORTANTE: usar a versão <5.13.5 no ESP não estava a funcionar bem a nova versão.

## Broker (Raspberry Pi)

O mqtt broker utilizado é o Mosquitto:

<https://mosquitto.org/download/>

É apenas necessário instalar no *Raspberry* e correr o Broker tudo o resto é feito nos clientes.

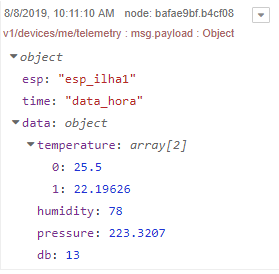
# NODE-RED

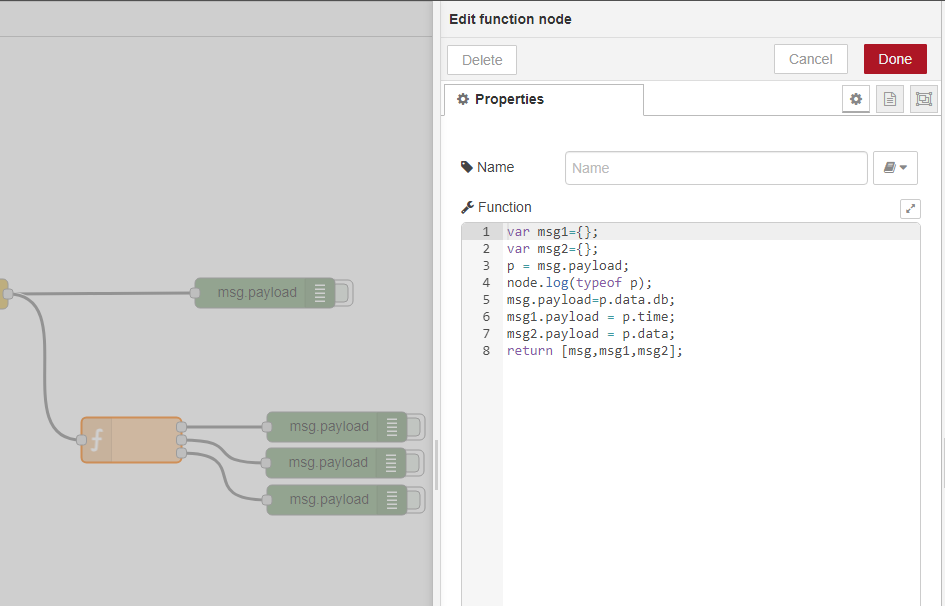
Instalar o Node-RED no servidor (ou no raspberry):

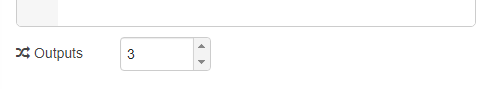
<https://nodered.org/>

Depois de ter o serviço a correr aceder ao localhost:1880 ou xxx.xxx.xxx.xxx:1880 para aceder à interface de programação do node red.

## Extrair dados do Json







## Mensagens

var msg1 = { payload:"first out of output 1" };

var msg2 = { payload:"second out of output 1" };

var msg3 = { payload:"third out of output 1" };

var msg4 = { payload:"only message from output 2" };

return [ [ msg1, msg2, msg3 ], msg4 ];

Output 1

Output 2

# MARIADB

## Criar Tabelas

COLUNAS

Nome Tipo Características

NOME DA TABELA

CREATE TABLE temp\_table(

Data DATETIME,

temp1 VARCHAR(5) NOT NULL,

temp2 VARCHAR(5) NOT NULL,

PRIMARY KEY ( temp1,temp2 )

);

Math.round(VALOR) – Arredonda ao numero inteiro.

VARIAVEL .toFixed (2) – Arredonda ao numero de casas decimais.

## Passar data e hora

var msg2={};

var date= msg.payload[0] ;

//time=msg.now;

p = msg.payload[1];

temp1=p.temperature[0];

temp2=p.temperature[1];

msg2.topic=("INSERT INTO temp\_table (Data,temp1, temp2) VALUES ('"+ date.toString() + "'," + temp1 + "," + temp2.toFixed(2)+ ")");

return msg2;

VER DADOS DA TABELA:

SELECT \* FROM temp\_table ORDER BY Data DESC LIMIT 20;

SELECT Data,temp1 FROM temp\_table WHERE Data >= “2019-09-05” LIMIT 20;

SELECT \* FROM temp\_table WHERE Data BETWEEN “2019-09-05” AND “2019-09-06”;

ALTERAR O NOME DOS DADOS PARA X,Y PARA USAR NOS GRAFICOS:

d = msg.payload;

var i;

var x;

var k = [];

var z = [];

for (i = 0; i < d.length; i++) {

w = d[i];

z[i] = {x: Date.parse(w.Data),y: Number(w.temp1)};

k[i]={x: Date.parse(w.Data),y: Number(w.temp2)};

}

msg.payload = [z,k];

return msg;

INSERIR NUM ARRAY PARA USAR NOS GRAFICOS

p=msg.payload

msg.topic='';

var q=[]

q=p

msg.payload = [

{

"series": [

"Temp",

"Temp Ada"

],

"data": q,

"labels": [

""

]

}

]

return msg;