**RASPBERRY PI 3B**

This document is made for system configration: 64 bit Ubuntu 16.04 LTS

**Raspberry Pi OS(optional )**

* Can build a custom kernel image using Buildroot
* Ready image :Jessie image.

**Install Custom OS onto the SD card**

Having built the kernel, you need to copy it onto your Raspberry Pi and install the modules; this is best done directly using an SD card reader.

First, use lsblk before and after plugging in your SD card to identify it. You should end up with something like this:

sdb

sdb1

sdb2

with sdb1 being the FAT (boot) partition, and sdb2 being the ext4 filesystem (root) partition.

If it's a NOOBS card, you should see something like this:

sdb

sdb1

sdb2

sdb5

sdb6

sdb7

with sdb6 being the FAT (boot) partition, and sdb7 being the ext4 filesystem (root) partition.

Mount these first, adjusting the partition numbers for NOOBS cards:

mkdir mnt

mkdir mnt/fat32

mkdir mnt/ext4

sudo mount /dev/sdb1 mnt/fat32

sudo mount /dev/sdb2 mnt/ext4

Next, install the modules:

sudo make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- INSTALL\_MOD\_PATH=mnt/ext4 modules\_install

Finally, copy the kernel and Device Tree blobs onto the SD card, making sure to back up your old kernel:

sudo cp mnt/fat32/$KERNEL.img mnt/fat32/$KERNEL-backup.img

sudo cp arch/arm/boot/zImage mnt/fat32/$KERNEL.img

sudo cp arch/arm/boot/dts/\*.dtb mnt/fat32/

sudo cp arch/arm/boot/dts/overlays/\*.dtb\* mnt/fat32/overlays/

sudo cp arch/arm/boot/dts/overlays/README mnt/fat32/overlays/

sudo umount mnt/fat32

sudo umount mnt/ext4

Another option is to copy the kernel into the same place, but with a different filename - for instance, kernel-myconfig.img - rather than overwriting the kernel.img file. You can then edit the config.txt file to select the kernel that the Pi will boot into:

kernel=kernel-myconfig.img

This has the advantage of keeping your kernel separate from the kernel image managed by the system and any automatic update tools, and allowing you to easily revert to a stock kernel in the event that your kernel cannot boot.

Finally, plug the card into the Pi and boot it!

**FLASHING JESSIE IMAGE**

**1) Etcher Software.**

**TEST MODEL:**

Raspberry Pi Talking to ESP8266 over WiFi using MQTT

### MQTT

MQTT is a machine-to-machine (M2M) data transfer protocol . MQTT was created with the goal of collecting data from many devices and then transporting that data to the IT infrastructure. It is lightweight, and therefore ideal for remote monitoring, especially in M2M connections that require a small code footprint or where network bandwidth is limited.

How MQTT works

MQTT is a publish/subscribe protocol that allows edge-of-network devices to publish to a broker. Clients connect to this broker, which then mediates communication between the two devices. Each device can subscribe, or register, to particular topics. When another client publishes a message on a subscribed topic, the broker forwards the message to any client that has subscribed.

MQTT is bidirectional, and maintains stateful session awareness. If an edge-of-network device loses connectivity, all subscribed clients will be notified with the “Last Will and Testament” feature of the MQTT server so that any authorized client in the system can publish a new value back to the edge-of-network device, maintaining bidirectional connectivity.

The project is divided in 3 parts

First, we create MQTT server on RPi and install some libraries.

Second, we will install libraries in Arduino IDE for NodeMCU to work with MQTT, upload the code and check whether server is working or not.Then we interface MQ135 with one NodeMCU and DHT11 with another

Third, we create a script in Rpi ,upload the required code in NodeMCU and run the python script to control leds from both server and client side.

Here, server is RPi and clients are two NodeMCU.

**First, we install latest MQTT server and client in RPi**

1. To use the new repository you should first import the repository package signing key:

wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key

sudo apt-key add mosquitto-repo.gpg.key

**2.**Then make the repository available to apt:

cd /etc/apt/sources.list.d/

sudo wget http://repo.mosquitto.org/debian/mosquitto-jessie.list

sudo apt-get update

Then install Mosquitto server using .

sudo apt-get install mosquitto

Errors:

The following packages have unmet dependencies:

mosquitto : Depends: libssl1.0.0 (>= 1.0.1) but it is not installable

Depends: libwebsockets3 (>= 1.2) but it is not installable

E: Unable to correct problems, you have held broken packages.

Solution:

sudo apt --fix-broken install

After, installing MQTT server install client using command.

sudo apt-get install mosquitto-clients

You can check services using command.

systemctl status mosquitto.service

MQTT SUB PUB MODELS:

For subscribe and publish commands:

<https://mosquitto.org/man/mosquitto_sub-1.html>

https://mosquitto.org/man/mosquitto\_pub-1.html

**Required library ==> paho-mqtt ,you can install using command.**

sudo apt-get update

sudo apt-get install python python-pip

sudo pip install RPi.GPIO paho-mqtt

**To setup static ip address**

Go to directory cd /etc and open file dhcpcd.conf using any editor. Add following lines:

interface eth0

static ip\_address=192.168.0.29 // ip you want to use

interface wlan0

static ip\_address=192.168.0.30

static routers=192.168.0.1 // your Default gateway

static domain\_name\_servers=192.168.0.1

**NODEMCU CONNECTED TO MQ135**

Requirements:

* Adafruit MQTT library
* Adafruit sleepydog library

CONNECTIONS:

GND --> G pin(NODEMCU)

VCC 🡪5V (NODEMCU)

Analog Pin 🡪 A0

CODE:

#include <ESP8266WiFi.h>

#include "Adafruit\_MQTT.h"

#include "Adafruit\_MQTT\_Client.h"

#include "MQ135.h"

#define ANALOGPIN A0 //

#define RZERO 79609.2 // Define RZERO Calibration Value

//Wifi Access Point

#define WLAN\_SSID "Tech\_D0017220"

#define WLAN\_PASS "YRJUUTJG"

#define MQTT\_SERVER "192.168.0.31" // give static address

#define MQTT\_PORT 1883

#define MQTT\_USERNAME ""

#define MQTT\_PASSWORD ""

MQ135 gasSensor = MQ135(ANALOGPIN);

// Create an esp8266 wificlient to connect to the MQTT server.

WiFiClient client;

// Setup the mqtt client class

Adafruit\_MQTT\_Client mqtt(&client, MQTT\_SERVER, MQTT\_PORT, MQTT\_USERNAME, MQTT\_PASSWORD);

// Setup a feed for publishing.

Adafruit\_MQTT\_Publish pi\_led = Adafruit\_MQTT\_Publish(&mqtt, MQTT\_USERNAME "/leds/pi");

// Setup a feed for subscribtion

Adafruit\_MQTT\_Subscribe esp8266\_led = Adafruit\_MQTT\_Subscribe(&mqtt, MQTT\_USERNAME "/leds/esp8266");

void MQTT\_connect();

void setup() {

Serial.begin(9600);

delay(10);

Serial.println(F("RPi-ESP-MQTT"));

// Connect to wifi access point.

Serial.println(); Serial.println();

Serial.print("Connecting to ");

Serial.println(WLAN\_SSID);

WiFi.begin(WLAN\_SSID, WLAN\_PASS);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println();

Serial.println("WiFi connected");

Serial.println("IP address: "); Serial.println(WiFi.localIP());

// Setup MQTT subscription for esp8266\_led feed.

mqtt.subscribe(&esp8266\_led);

float rzero = gasSensor.getRZero();

delay(3000);

}

int initValue= 0;

void loop() {

MQTT\_connect();

Adafruit\_MQTT\_Subscribe \*subscription;

while ((subscription = mqtt.readSubscription())) {

if (subscription == &esp8266\_led) {

Serial.print(F("Got: "));

Serial.println((char \*)esp8266\_led.lastread);

/\* \*/

}

}

float ppm = gasSensor.getPPM();

delay(1000);

Serial.print("CO2 ppm value : ");

Serial.println(ppm);

if (! pi\_led.publish(ppm)) {

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

} else {

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

}

}

// Function to connect and reconnect as necessary to the MQTT server.

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!");

}

**NODEMCU AND DHT**

#include "DHT.h"

#define DHTTYPE DHT11

#define dht\_dpin 0

DHT dht(dht\_dpin, DHTTYPE);

Void setup()

{

Serial.begin(9600);

Serial.println("Humidity and temperature\n\n");

delay(700);

}

Void loop

{

float h = dht.readHumidity();

float t = dht.readTemperature();

Serial.print("Current humidity = ");

Serial.print(h);

Serial.print("% ");

Serial.print("temperature = ");

Serial.print(t);

Serial.println("C ");

delay(800);

String temperature = String(t);

String humidity = String(h);

String payload = "";

payload += "\"temperature\":"; payload += temperature; payload += ",";

payload += "\"humidity\":"; payload += humidity;

payload += " ";

char attributes[100];

payload.toCharArray( attributes, 100 );

if (! (dht\_led.publish(attributes) ))

{

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

} else {

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

}

}