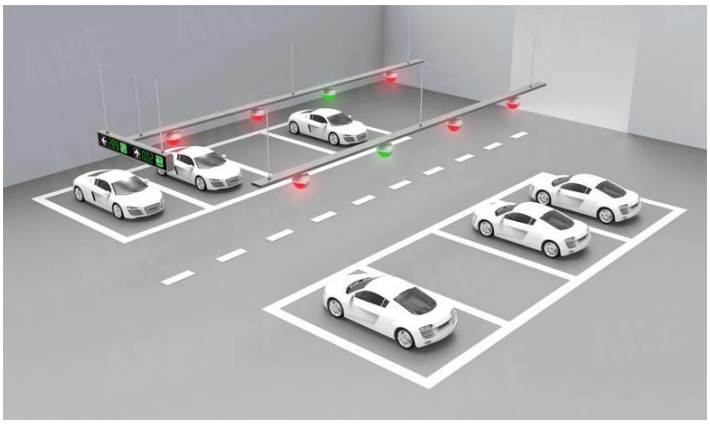


SMART PARKING SOLUTION

Technical Documentation

ABSTRACT

Includes the technical specifications and configurations related to the sensor nodes, gateways and AWS cloud. Purpose is to map with the current progress with the initial specifications.

REVISION

1.1

CREATED BY

Thusara Grero

On 3rd May 2018

Security Credentials

** Raspberry Pi Login **

username : pi

password : raspberry

** MySQL credentials **

username : root2

password : root2

database : parking

** AWS configurations with Node-RED **

Thing name : Gateway_1

Server : a2ktipjm632gcm.iot.us-east-2.amazonaws.com

port: 8883

Certificate : /home/pi/cert/a4ce6b7c13-certificate.pem.crt

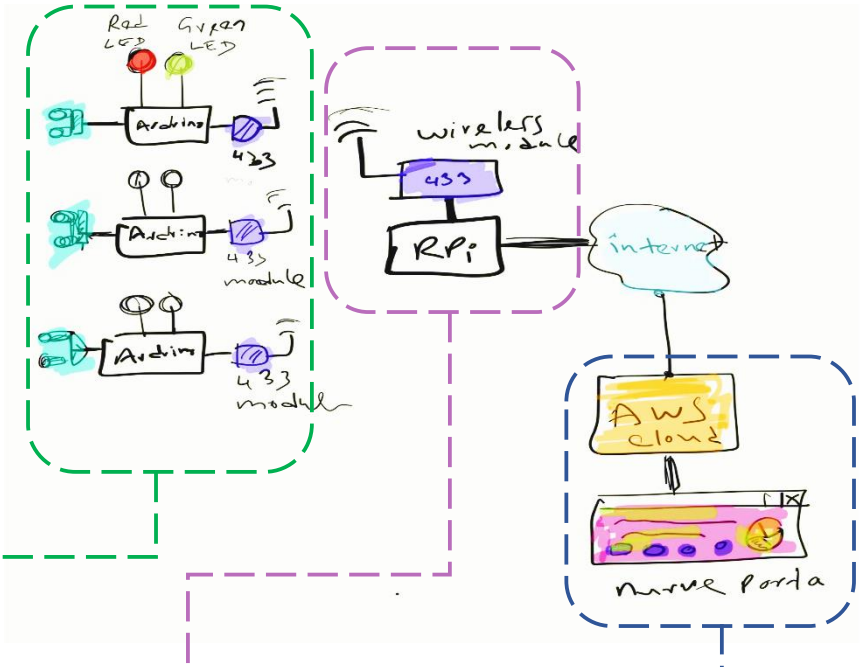
Private Key : /home/pi/cert/a4ce6b7c13-private.pem.key

CA Certificate : /home/pi/cert/rootCA.pem

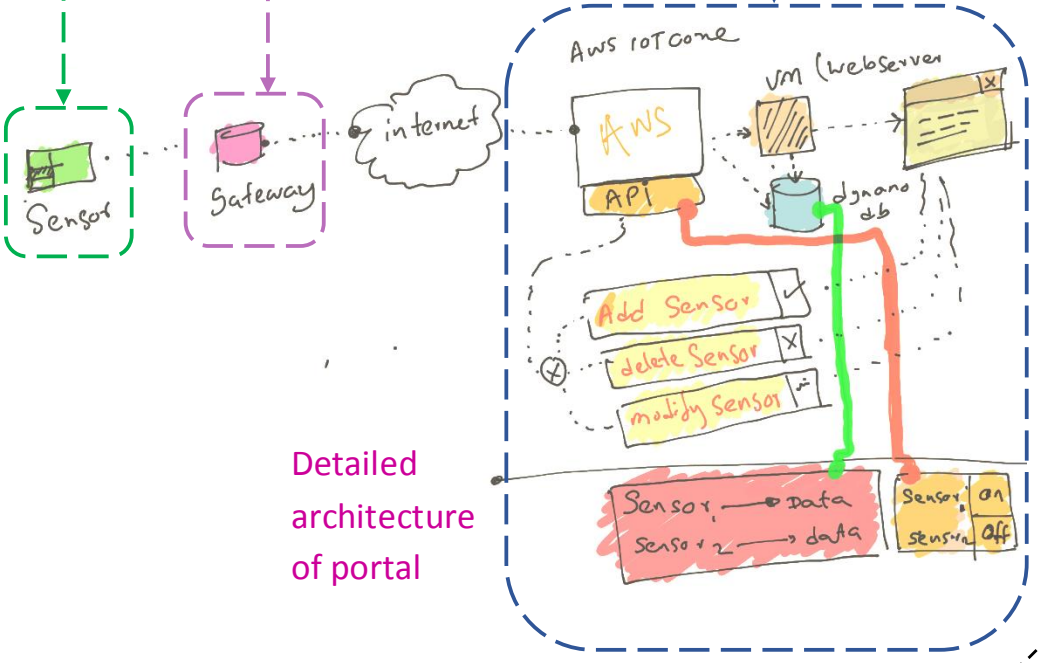
Certificate files : Click here.

High Level Concept and features of Sensors, Gateways and portal

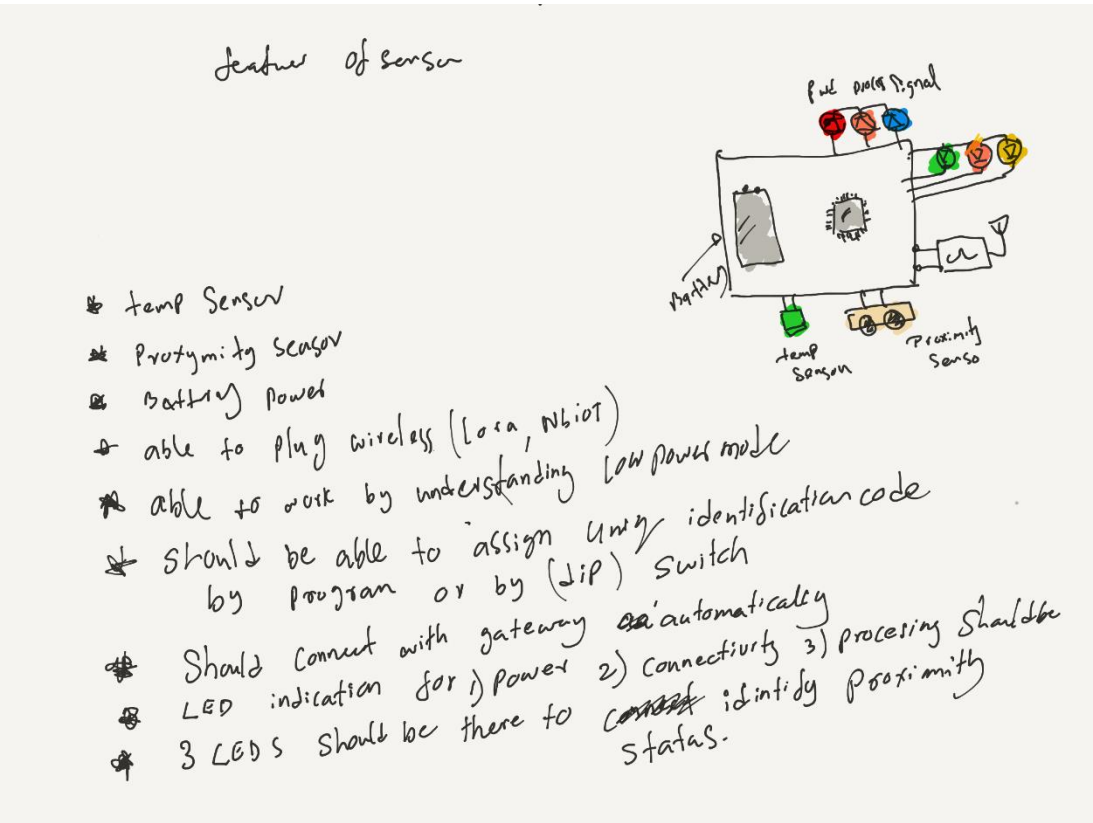
Detailed architecture of Sensors and Gateways



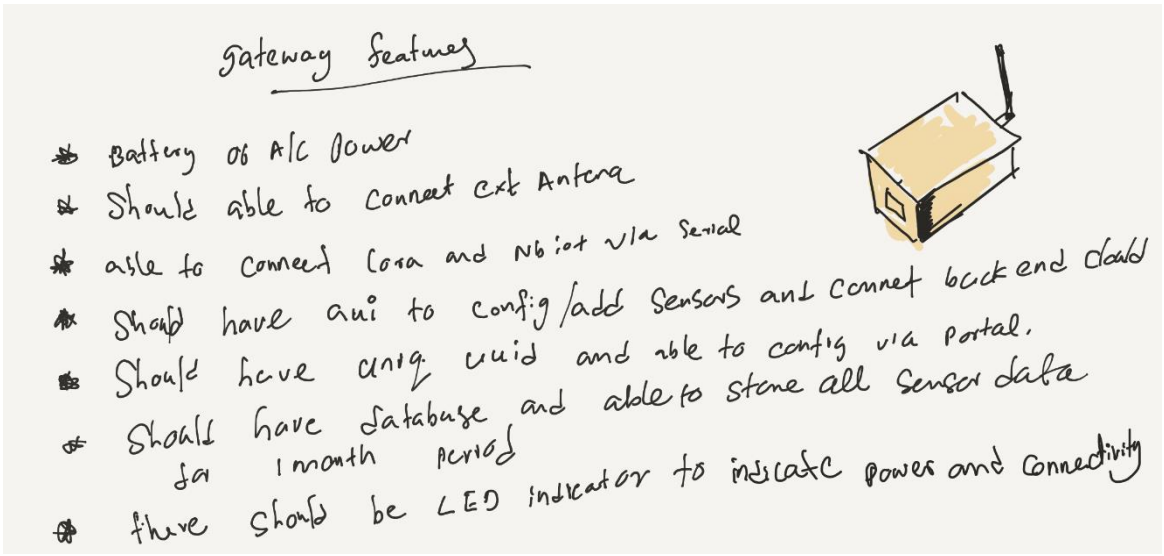
Detailed architecture of portal



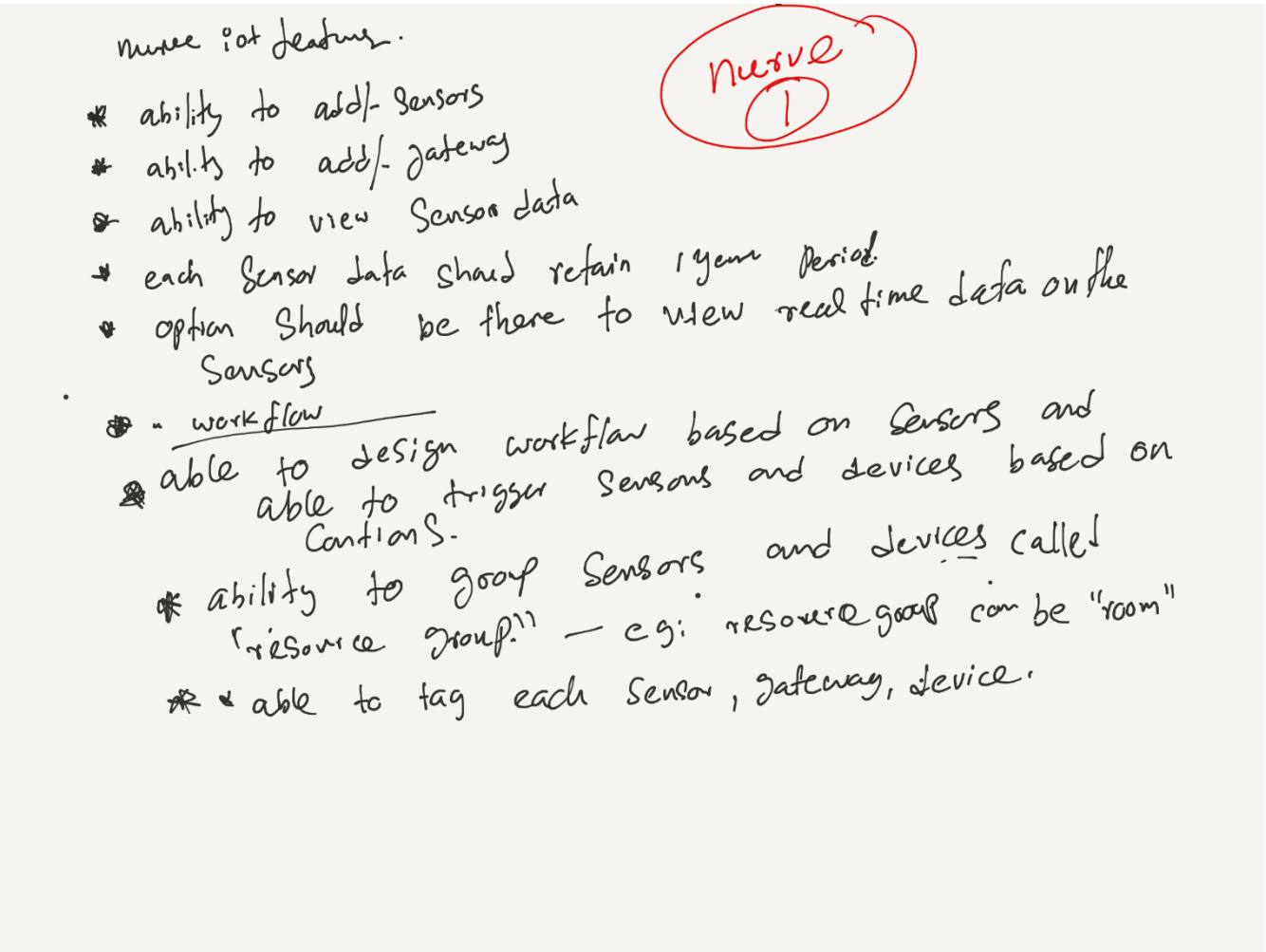
Features expected on the sensor



Features expected on the gateway



Features expected on the Nurve platform



*** Amendments possible for expected features

Project initialization

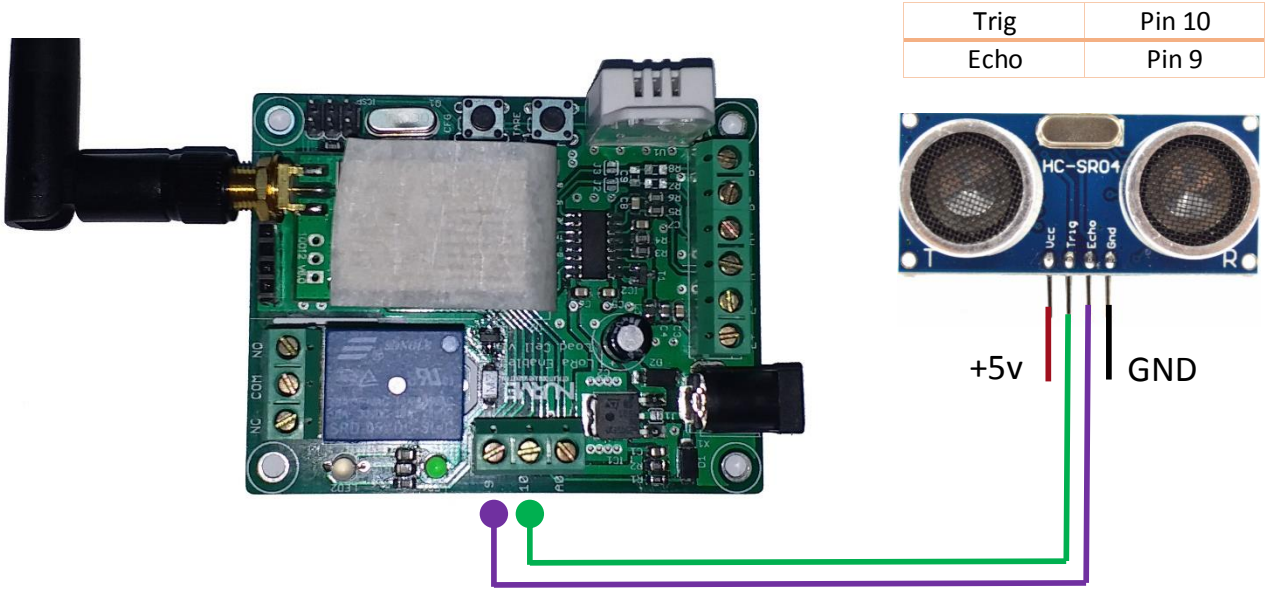
- The initial hardware prototype of the parking solution was built using a PCB designed for a previous project (Lora Enabled load cell).
- Certain features of the past project was also utilized (ie. Temperature, Battery Level, Humidity etc.)
- 9th and 10th pin which were exposed from the above PCB was used to mount the ultrasonic sensor (HC-SR06)
- In this revision (1.1) no modifications to the PCB was done and was utilized as it is.

Hardware Setup & Software Setup for the prototype

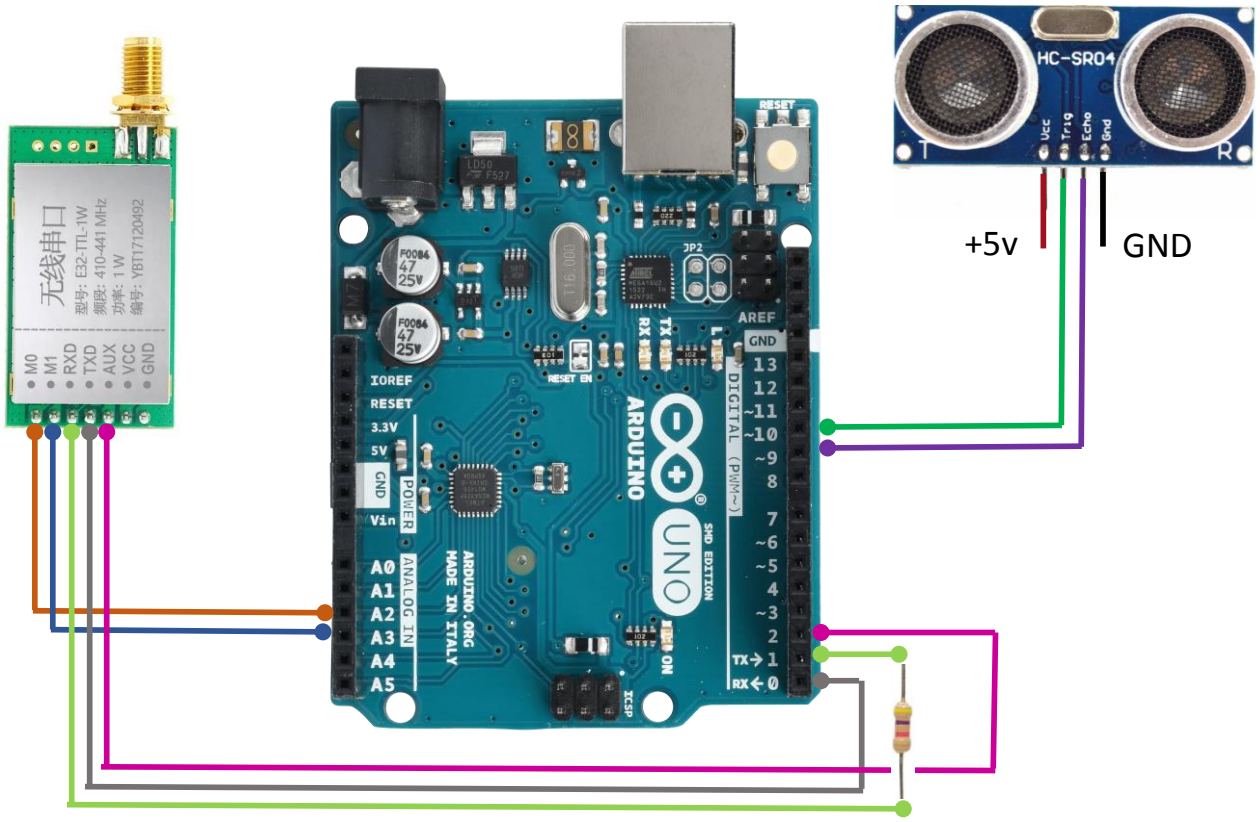
Sensor node setup

- Hardware setup was done in two forms.
    1. From load cell PCB
    2. From Arduino (because only two working PCBs were available and was required to be tested with multiple sensors)
- Refer the attached schematic for the PCB architecture. LoRa Enabled Load Cell ver1 (rev A) Schematic.pdf

Load cell PCB integration



Arduino UNO integration



HC-SR04	Arduino
Trig	Pin 10
Echo	Pin 9

Lora Module	Arduino
M0	Pin A2
M1	Pin A3
RxD	Pin 1 TX *(with 4-10k resistor)*
TxD	Pin 0 RX
AUX	Pin 2
VCC	+5v
GND	GND

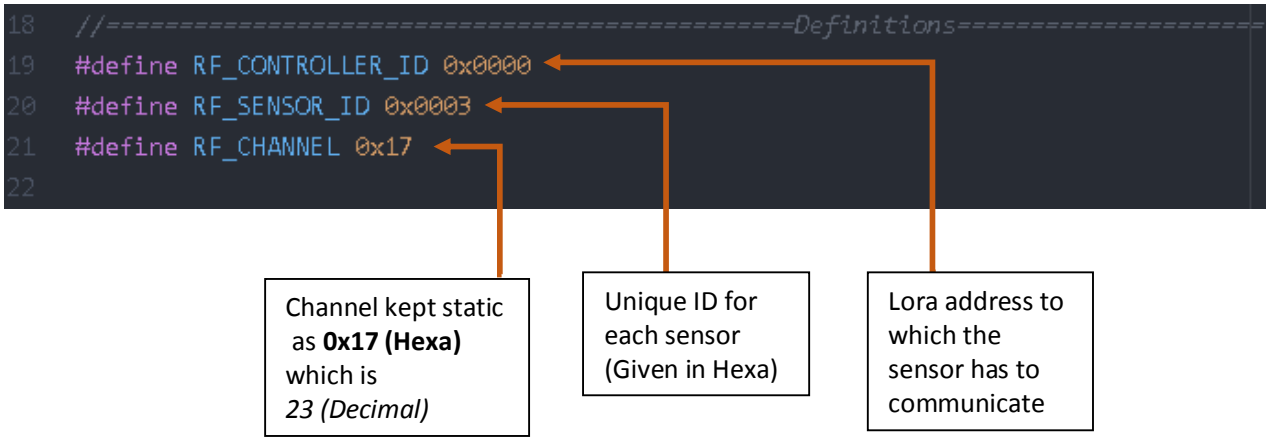
Note : *** 4-10 K Ohm resistor is necessary to connect the Rx(Lora) -> Tx (Uno) because of impedance issue as it will not transmit properly. This is properly discussed ***here***.

Latest modified source code of the sensor node

Source code file(s) : SENSOR NODE SOURCE CODE

Last updated : April 11 – 2018

Summary: The load cell code was modified and load cell component was replaced with ultrasonic component.



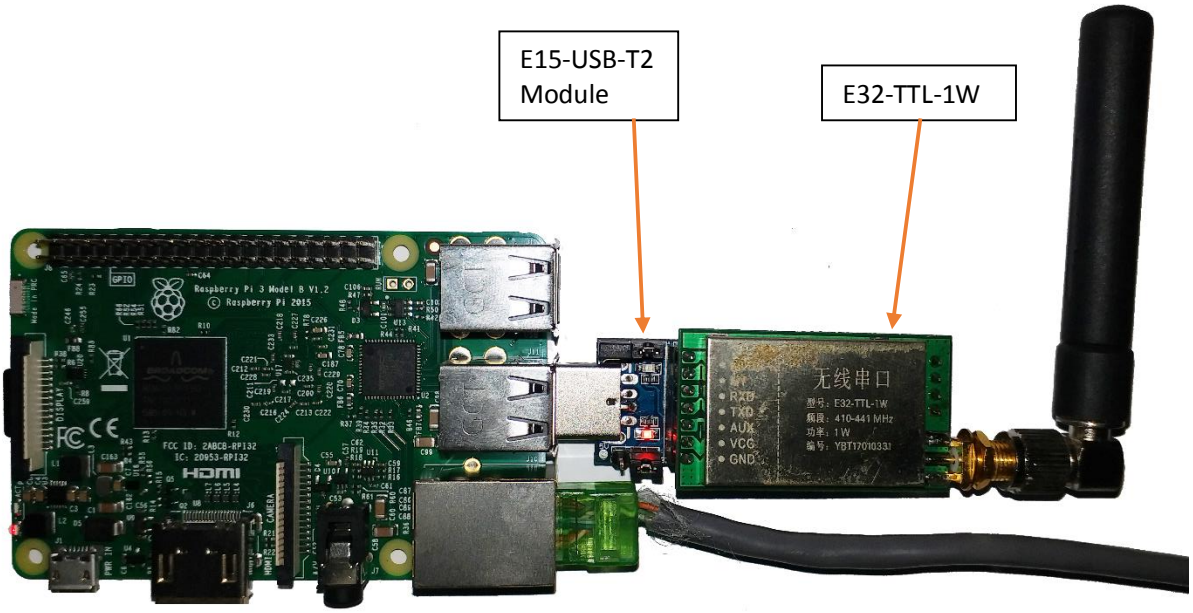
New variable definitions

```
91
92  // defines pins numbers
93  const int trigPin = 10;
94  const int echoPin = 9;
95
96  // defines variables
97  long duration;
98  int distance;
99  int Ultrasonic_Value;
100
```

Function which reads the ultrasonic sensor

```
519  //=============================================================================
520  int readUltrasonic(){
521    // Clears the trigPin
522  digitalWrite(trigPin, LOW);
523  digitalWrite(COM_IND_LED_PIN,HIGH);
524  delayMicroseconds(2);
525  // Sets the trigPin on HIGH state for 10 micro seconds
526  digitalWrite(trigPin, HIGH);
527  //delayMicroseconds(10);
528  // Reads the echoPin, returns the sound wave travel time in microseconds
529  duration = pulseIn(echoPin, HIGH);
530  // Calculating the distance
531  distance= duration*0.034/2;
532  // Prints the distance on the Serial Monitor
533     //Serial.print("Status: ");
534  //Serial.println(distance);
535
536  if(distance>=10){
537    digitalWrite(COM_IND_LED_PIN, LOW); // LOW means HIGH here
538    //Serial.println("Available");
539    return 1;
540  }
541  else if(distance>=0 && distance<10){
542    digitalWrite(COM_IND_LED_PIN, HIGH); // HIGH means LOW here
543    //Serial.println("Occupied");
544    return 0;
545  }
```

Gateway Setup



Source file(s) : [GATEWAY SOURCE CODE]

Certificates for AWS : [CERTIFICATE FILES]

The script to be executed : main1.js (command: **node main1.js**)

Other important files : config/default.json

Local database architecture : MySQL (database -> parking)

```
mysql> use parking
Database changed
mysql> show tables;
+--------------------+
| Tables_in_parking |
+--------------------+
| data               |
| data_log           |
| gateway            |
| sensor             |
+--------------------+
4 rows in set (0.00 sec)
```

```
mysql> select * from sensor;
+----+------------+
| id | gateway_id |
+----+------------+
|  2 |          1 |
|  3 |          1 |
|  4 |          1 |
|  9 |          1 |
+----+------------+
4 rows in set (0.02 sec)

mysql> select * from gateway
    -> ;
+----+---------+--------------+
| id | channel | seria_port   |
+----+---------+--------------+
|  1 |      23 | /dev/ttyUSB0 |
+----+---------+--------------+
1 row in set (0.00 sec)
```

```
mysql> select * from sensor;
+----+------------+
| id | gateway_id |
+----+------------+
|  2 |          1 |
|  3 |          1 |
|  4 |          1 |
|  9 |          1 |
+----+------------+
4 rows in set (0.02 sec)

mysql> select * from gateway
    -> ;
+----+---------+--------------+
| id | channel | seria_port   |
+----+---------+--------------+
|  1 |      23 | /dev/ttyUSB0 |
+----+---------+--------------+
1 row in set (0.00 sec)

mysql> select * from data;
+----+---------------------+------------+-----------+--------------------+--------------------+--------------+--------------+------+
| id | created_at          | gateway_id | sensor_id | temperature        | humidity           | availability | battery      | flag |
+----+---------------------+------------+-----------+--------------------+--------------------+--------------+--------------+------+
|  5 | 2018-05-03 05:40:38 | 1          | 9         | 33.10000228881836  | 30.899999618530273 | 1            | 4.9951171875 | NULL |
|  6 | 2018-05-03 05:40:37 | 1          | 4         | 31.700000762939453 | 35                 | 1            | 2.4658203125 | NULL |
+----+---------------------+------------+-----------+--------------------+--------------------+--------------+--------------+------+
2 rows in set (0.00 sec)
```

Configurations :

1. Source files should be available in ***lora_mod*** in the home directory. Browse to it by

   cd ~/lora_mod

```
pi@raspberrypi:~ $ cd ~/lora_mod
pi@raspberrypi:~/lora_mod $
```

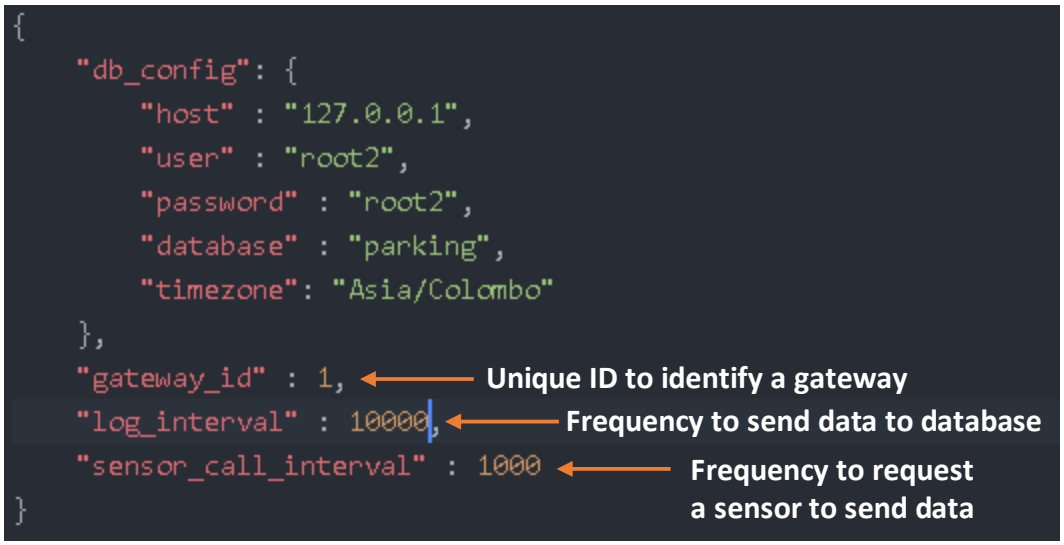
2. Certificate files of AWS should be located in **cert** directory in **home**. (MQTT node in node red will depend on this)

```
pi@raspberrypi:~ $ cd cert
pi@raspberrypi:~/cert $ ls
a4ce6b7c13-certificate.pem.crt  a4ce6b7c13-public.pem.key
a4ce6b7c13-private.pem.key      rootCA.pem
```

3. ***main1.js*** is the latest code to be executed. It will;
   a. *Read the data of the sensors by requesting each sensor recorded in the sensor table in the database.*
   b. *Display the values in the terminal*
   c. *Record the payload in the table **'data'** in the local database **'parking'**.*

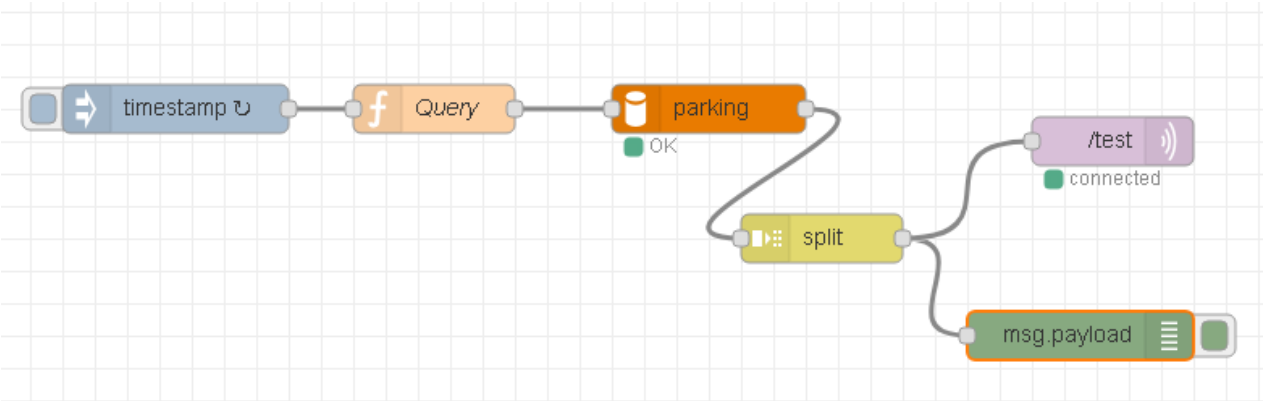
(Here if a record exist from that sensor it will be updated and if not a new record will be created)

4. Config/default.json has following configurations;



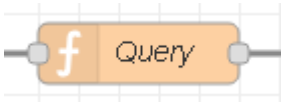
5. Node-red has to be installed and if the code fails to run after node-red installation follow;
   Serial Port Installation

6. Node red configurations



Repeat: interval

every 0.1 seconds



Function

```
1  var out = "SELECT * FROM data;"
2  msg.topic=out;
3  return msg;
```

Host 127.0.0.1

Port 3306

User root2

Password ••••••••

Database parking

Timezone t2

String / Buffer

Split using \n

Handle as a stream of messages

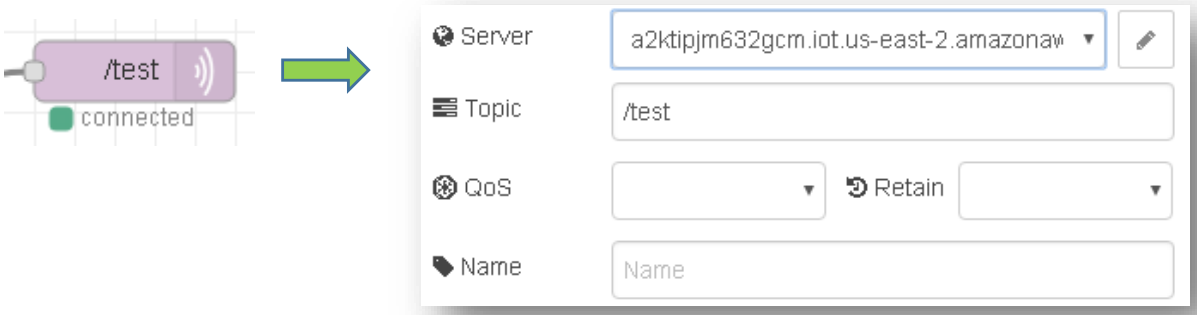
Array

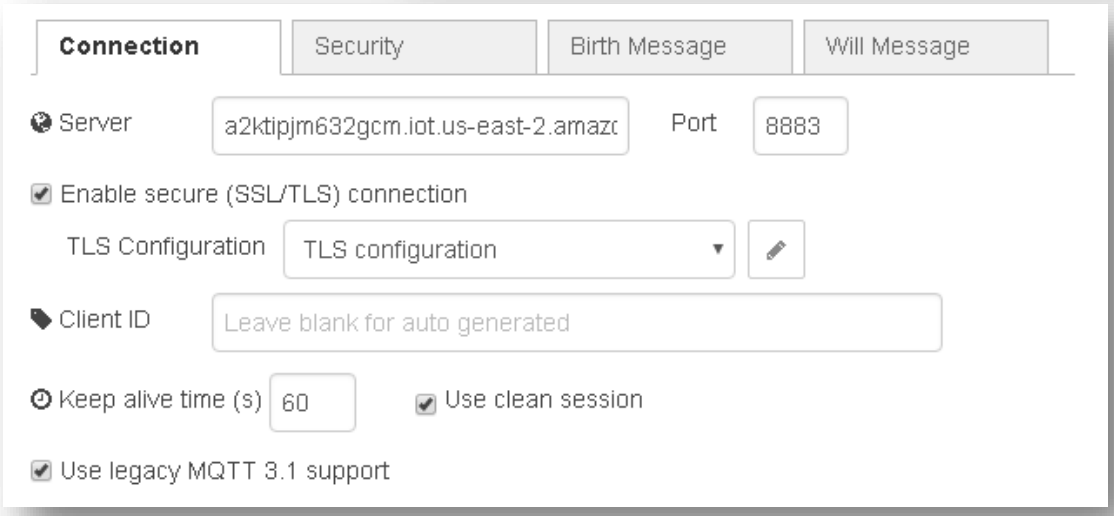
Split using Fixed length of 1

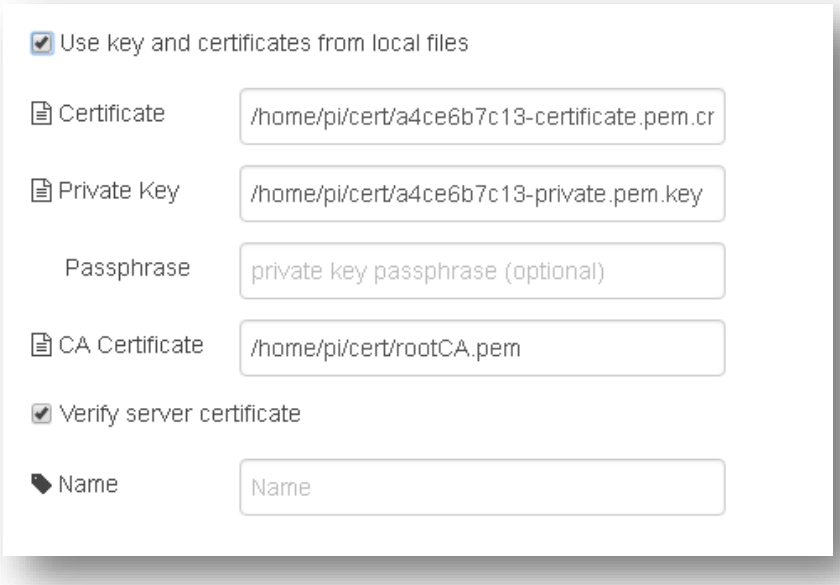
Object

Send a message for each key/value pa

Copy key to msg. topic


/test
connected
Server
a2ktipjm632gcm.iot.us-east-2.amazonaw
Topic
/test
QoS
Retain
Name
Name


Connection
Security
Birth Message
Will Message
Server
a2ktipjm632gcm.iot.us-east-2.amazc
Port
8883
Enable secure (SSL/TLS) connection
TLS Configuration
TLS configuration
Client ID
Leave blank for auto generated
Keep alive time (s)
60
Use clean session
Use legacy MQTT 3.1 support


Use key and certificates from local files
Certificate
/home/pi/cert/a4ce6b7c13-certificate.pem.cr
Private Key
/home/pi/cert/a4ce6b7c13-private.pem.key
Passphrase
private key passphrase (optional)
CA Certificate
/home/pi/cert/rootCA.pem
Verify server certificate
Name
Name

LORA Module configuration

Software to configure: RF_Setting_v3.45.exe

Address is the unique ID used to identify each Lora module. Screen capture shows the one on the gateway.

Channel is the path which we use to transmit and receive. In our scenario we have made it a static channel where all gateways and sensors uses this channel 23 (Hex 0x17).

Data Sheet of USB module: E15-USB-T2_Datasheet_EN_v1.0 .pdf