What you will find in this report

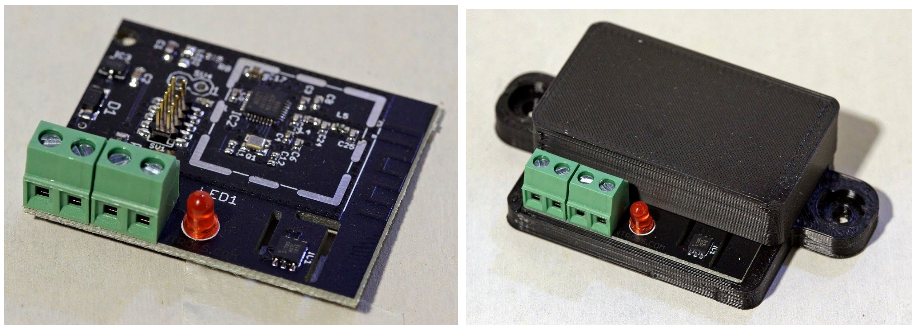
Building a complete and expandable IoT System

IoT system proposal

The initial idea is to integrate a temperature monitoring and light control to maintain the temperature for an indoor garden.

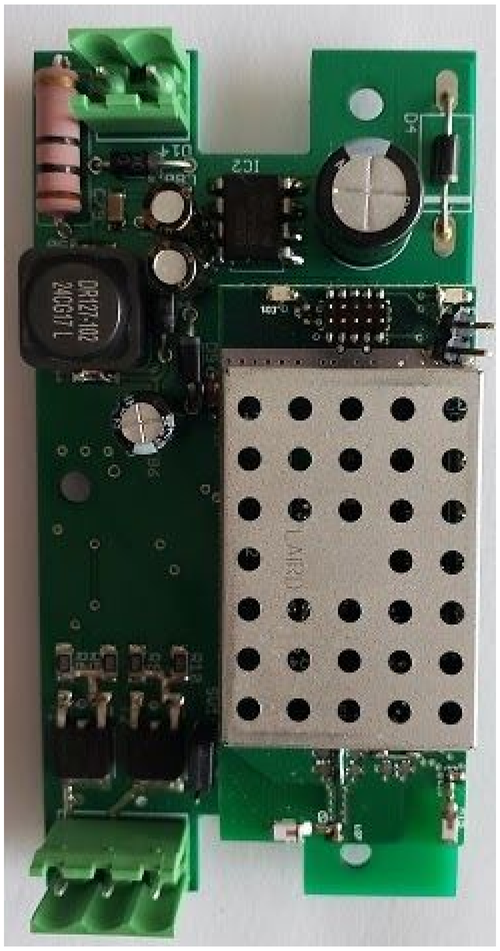
Required Hardware: Temperature Sensor, Digital I/O Light control module, IoT Gateway.
Note: Of course we need power supplies and some power electronics to control the light but its not central for the IoT design.

Wireless Temperature Sensor



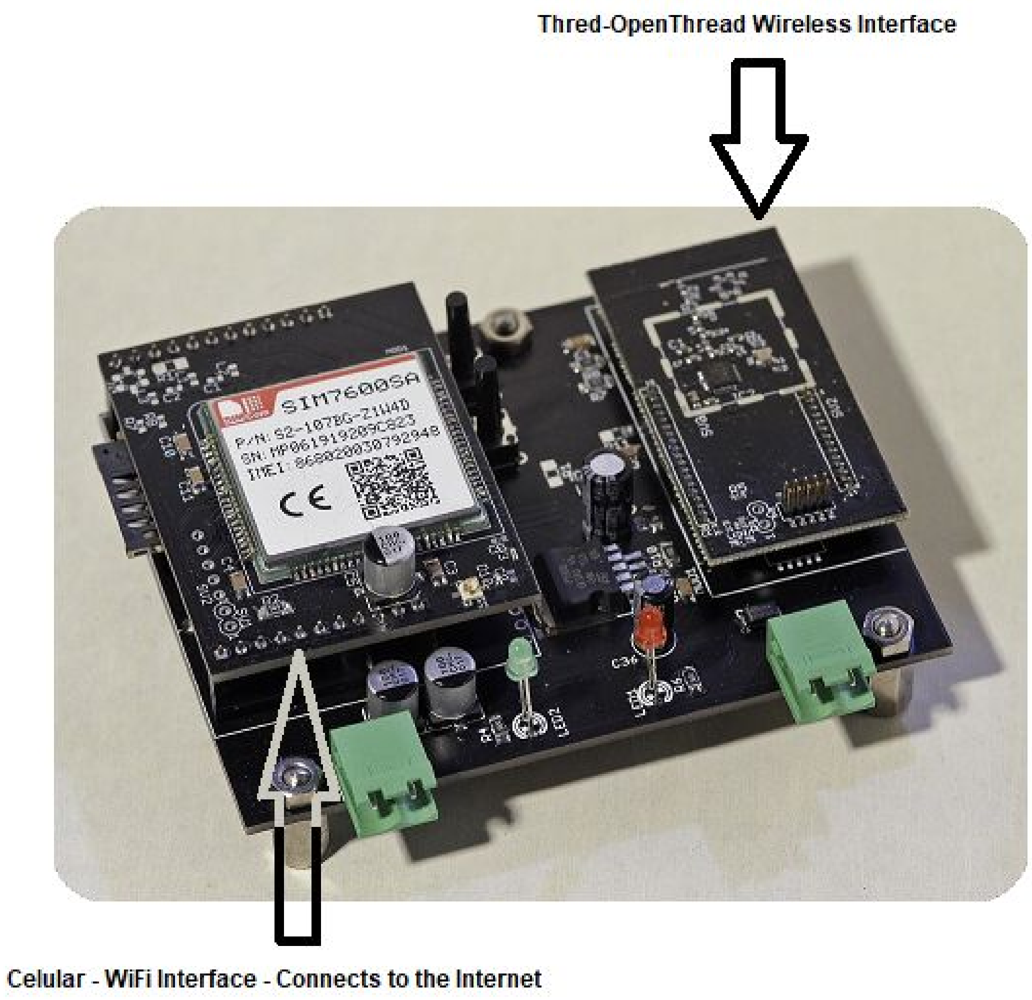
Thread-Openthread based Wireless Temperature Sensor

Wireless Light Control



Thread - OpenThread Based Wireless Light Control
With Integrated power.

Wireless Gateway

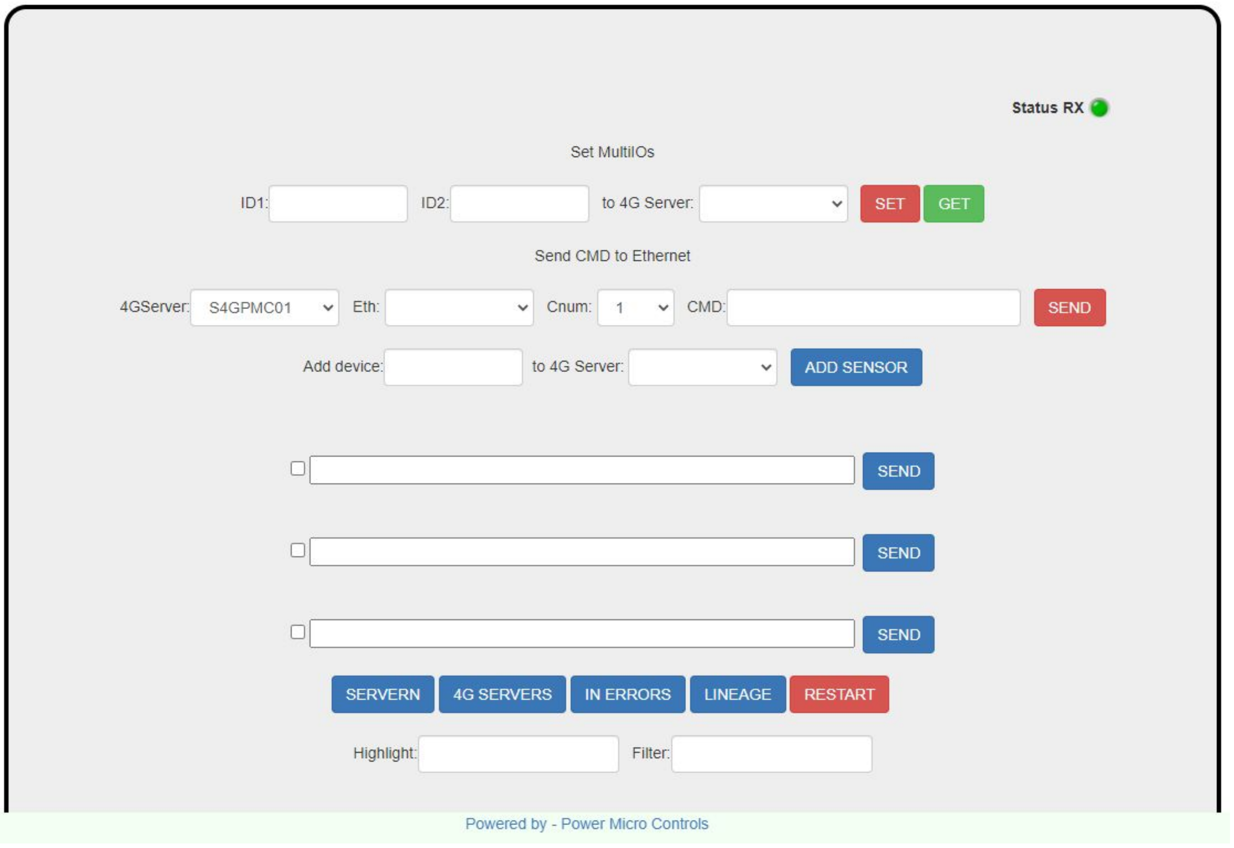

Thred-OpenThread Wireless Interface
Celular - WiFi Interface - Connects to the Internet

Configuring the system

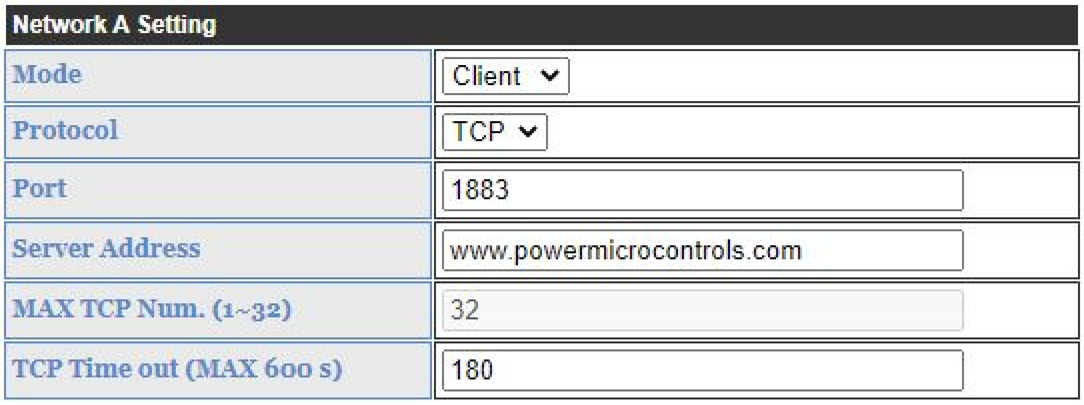
The temperature sensor has an ID which uniquely identifies it in the wireless Network 461NUM83. In a similar way, the Light control has its own ID UCS997UM and so the Gateway GMKUTT01. This IDs allows uniquely identifying each device on the network.

The System is based on the Thread - OpenThread Protocol and all communication between devices are wireless and encrypted.

The platform configuration page allows you to add a device to the wireless sensing network. In the future more devices can be added.



The Gateway configuration Involves at least the IoT platform TCP/IP parameters that interact with our IoT hardware.

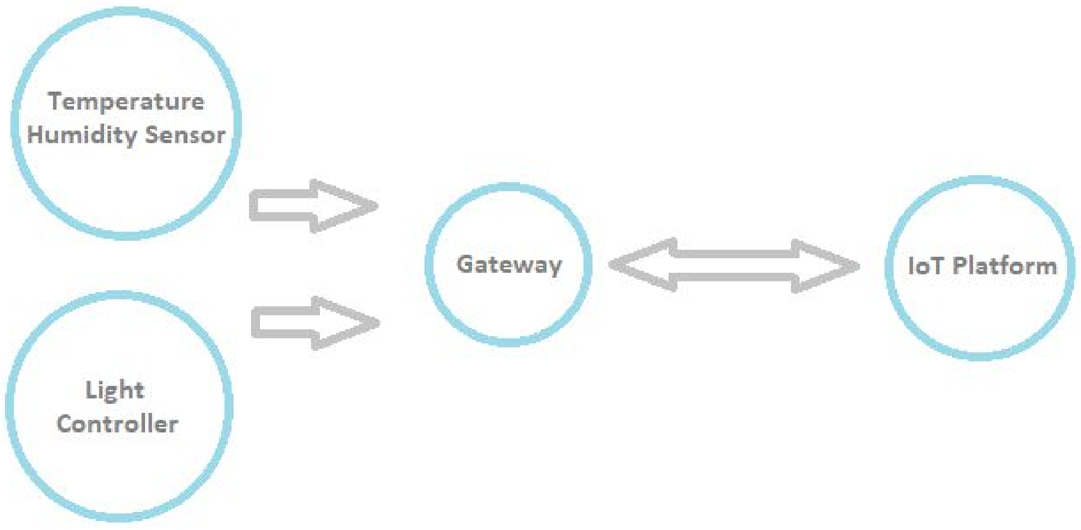


Network A Setting	
Mode	Client
Protocol	TCP
Port	1883
Server Address	www.powermicrocontrols.com
MAX TCP Num. (1~32)	32
TCP Time out (MAX 600 s)	180

The Gateway automatically connects-reconnects with the IoT Platform

What we have up to this point

We have configured the wireless sensors. The sensors are connected to the gateway and periodically they send packets-messages to it with the information of Temperature, Humidity and light status. The gateway is configured with the server - IoT platform address and port.



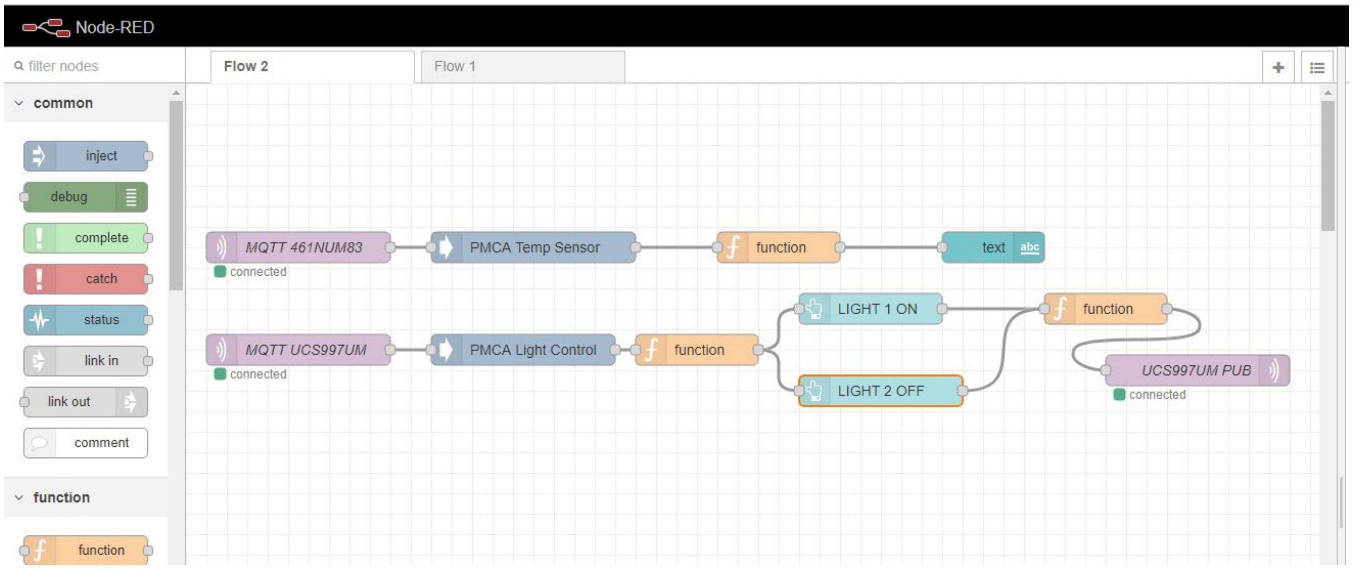
The IoT Platform

In this document we are showing three alternatives to control and monitor your IoT Platform.

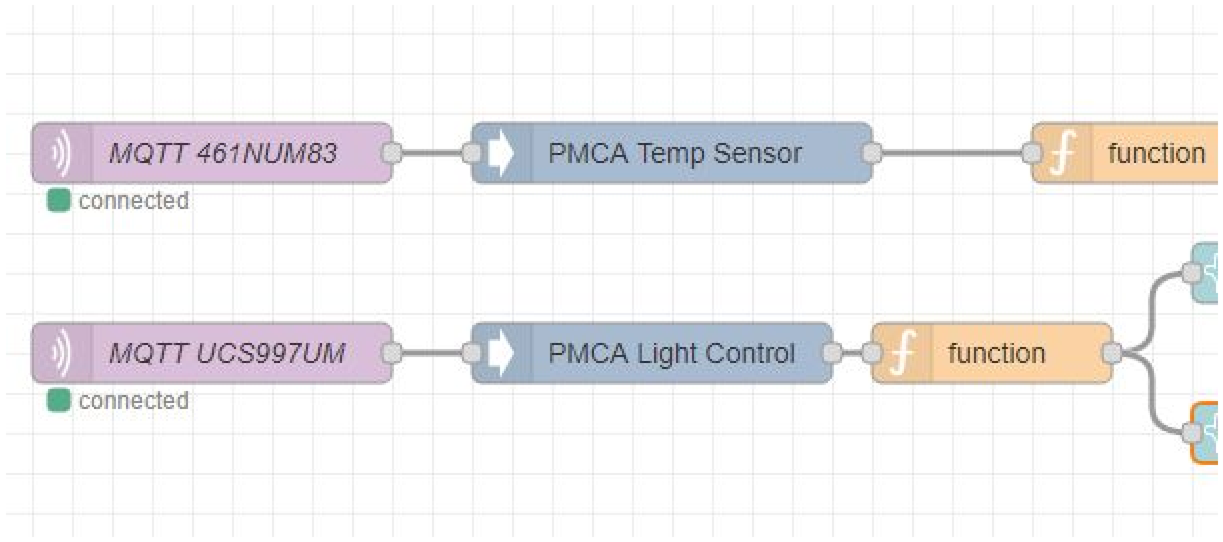
NODE-RED

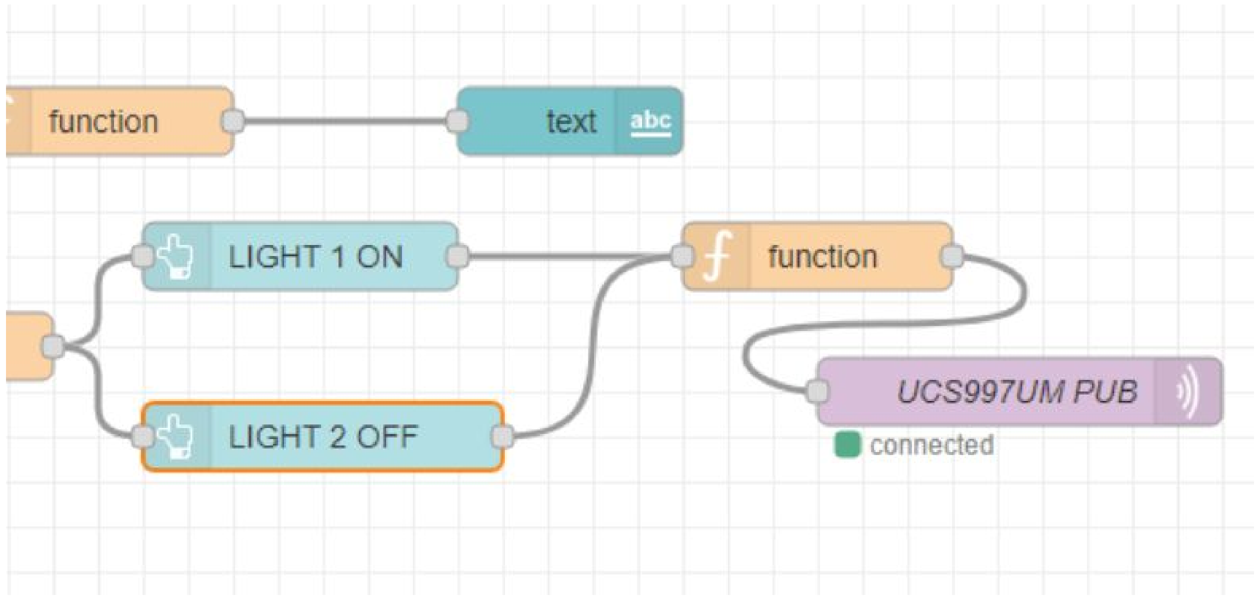
The first one is using **Node-Red** https://nodered.org/ . This is an easy way to put to work an IoT Platform. Based on a Flow Programming you can easily monitor and control your IoT device and network.

Below is an example of a flow programm and a simple dashboard showing basic functionality:

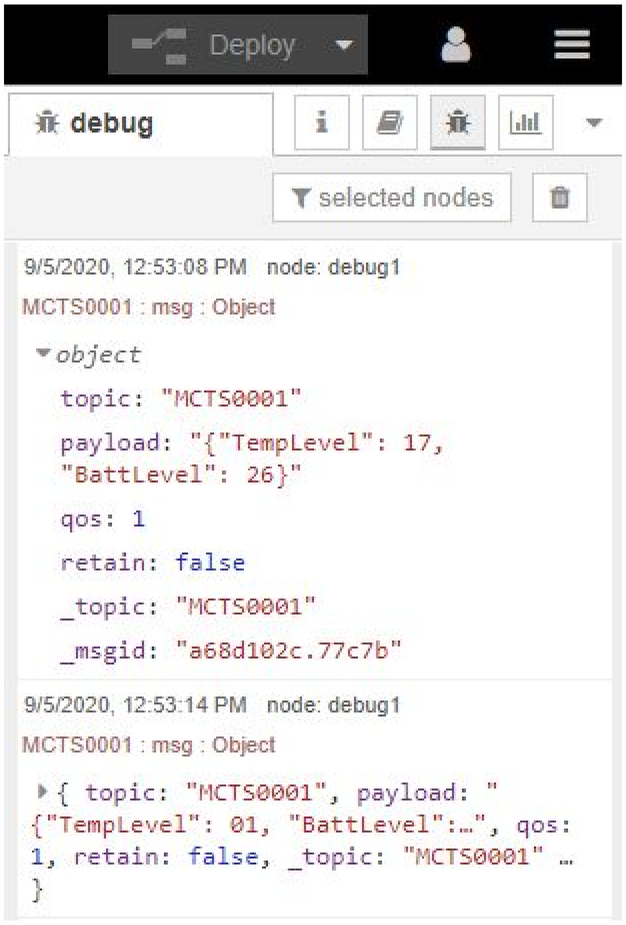


For better clarity here is a second image of the flow chart where you can visualize some of the building blocks of the program flow.



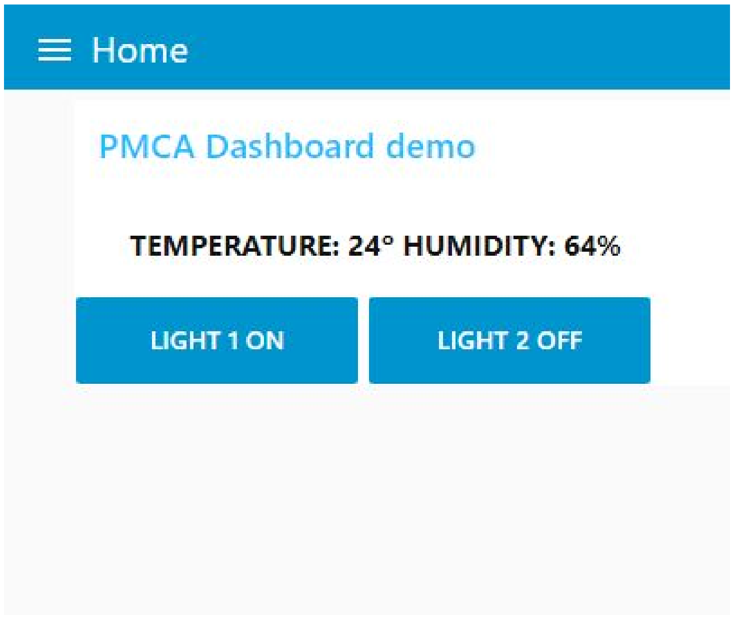


Below you can observe how data arrives to the IoT Platform for further visualization and control



```
9/5/2020, 12:53:08 PM   node: debug1
MCTS0001 : msg : Object
▾object
  topic: "MCTS0001"
  payload: "{"TempLevel": 17,
  "BattLevel": 26}"
  qos: 1
  retain: false
  _topic: "MCTS0001"
  _msgid: "a68d102c.77c7b"
9/5/2020, 12:53:14 PM   node: debug1
MCTS0001 : msg : Object
▸{ topic: "MCTS0001", payload: "
{"TempLevel": 01, "BattLevel":…", qos:
1, retain: false, _topic: "MCTS0001" …
}
```

Finally the Dashboard that will display and allows you to turn on and off the light manually and with some extra effort you can change the program flow and make the light turned on or off and thus regulate the temperature automatically.



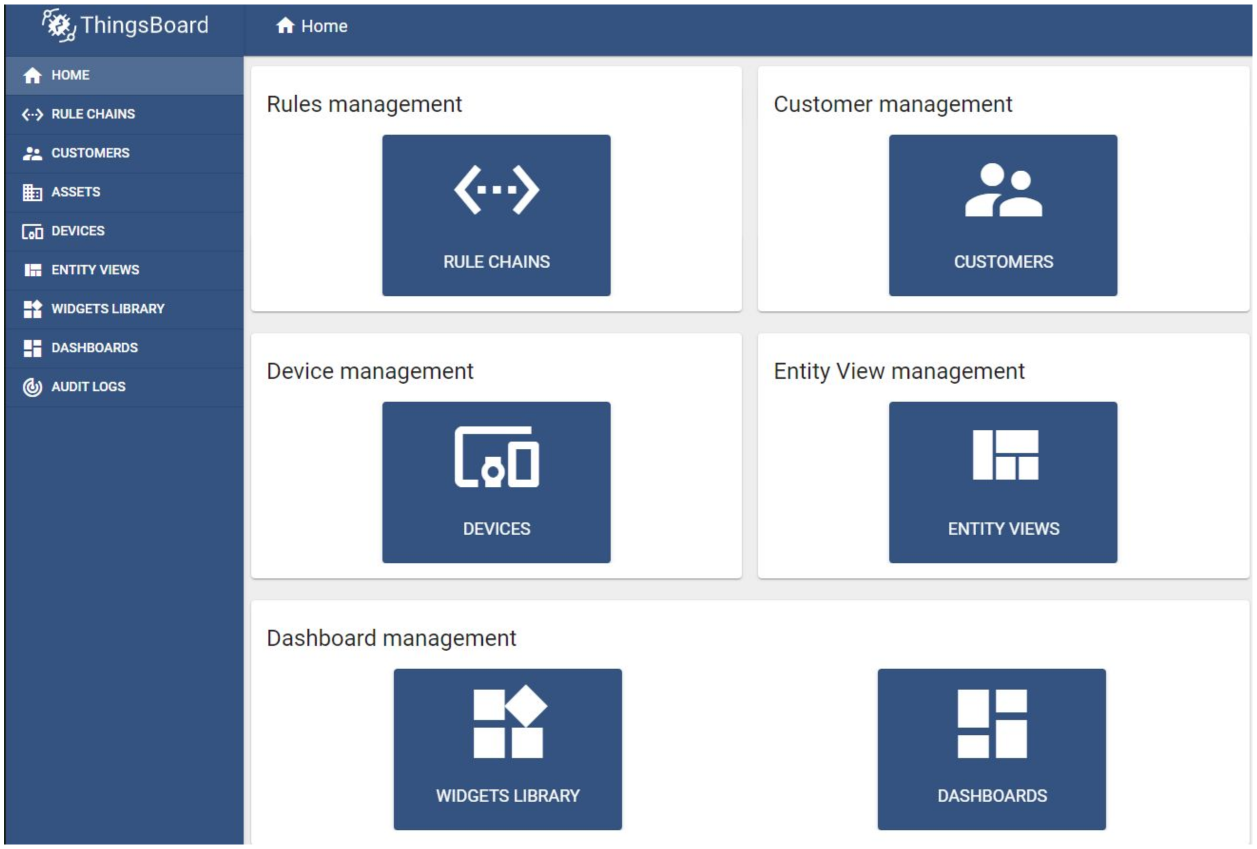
TEMPERATURE: 24° HUMIDITY: 64%

An interesting option NODE-RED can be deployed in a raspberry-pi and used as your IoT Cloud Server Hardware or you can deploy it in a VPS server.

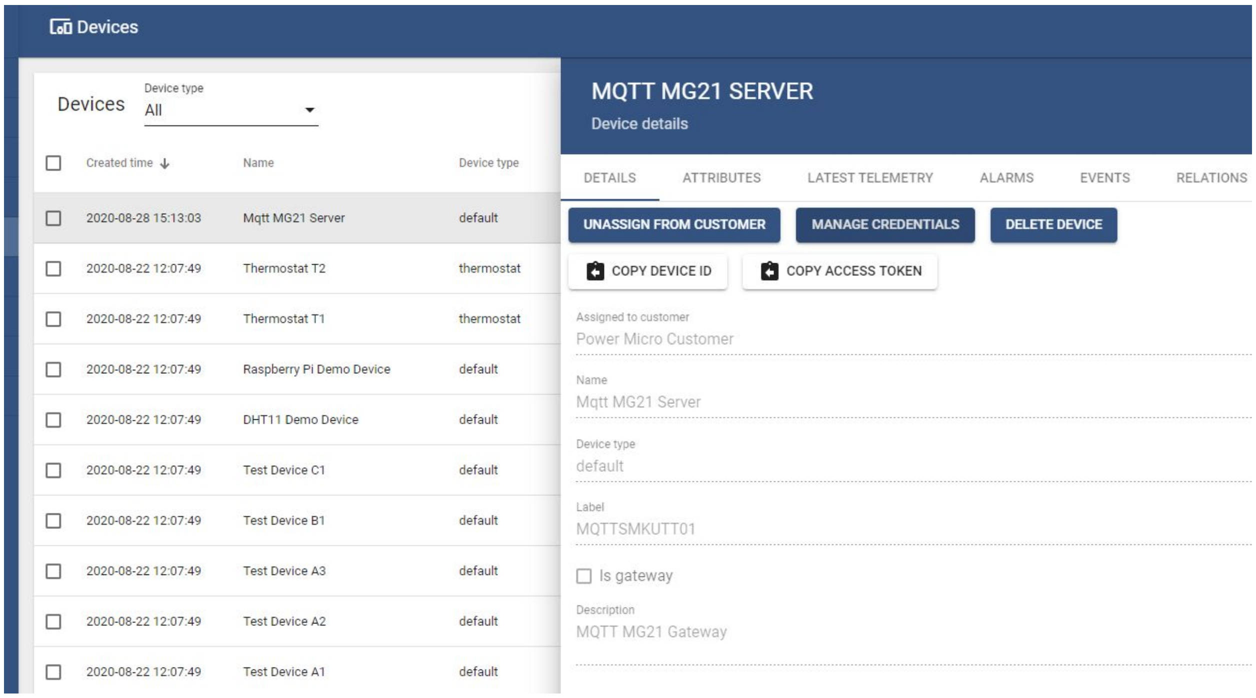
Raspberry based IoT Micro Server

Thingsboard

Another alternative is using **Thingsboard https://thingsboard.io/** that we found easy to use and with several alternatives. The community edition CE and the Professional Edition which includes a Cloud version with storage capacity and support.



Devices are configured easily and the learning curve to manage the complete system is quiet fast.



Devices

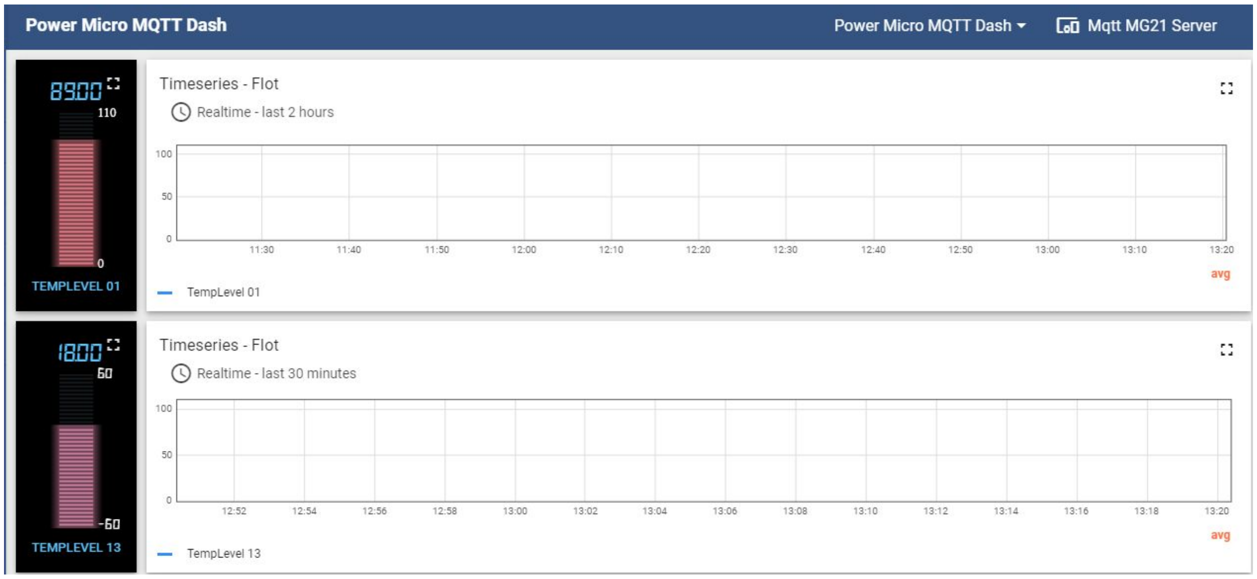
	Created time	Name	Device type
☐	2020-08-28 15:13:03	Mqtt MG21 Server	default
☐	2020-08-22 12:07:49	Thermostat T2	thermostat
☐	2020-08-22 12:07:49	Thermostat T1	thermostat
☐	2020-08-22 12:07:49	Raspberry Pi Demo Device	default
☐	2020-08-22 12:07:49	DHT11 Demo Device	default
☐	2020-08-22 12:07:49	Test Device C1	default
☐	2020-08-22 12:07:49	Test Device B1	default
☐	2020-08-22 12:07:49	Test Device A3	default
☐	2020-08-22 12:07:49	Test Device A2	default
☐	2020-08-22 12:07:49	Test Device A1	default

Device type: All

MQTT MG21 SERVER
Device details

Assigned to customer: Power Micro Customer
Name: Mqtt MG21 Server
Device type: default
Label: MQTTSMKUTT01
☐ Is gateway
Description: MQTT MG21 Gateway

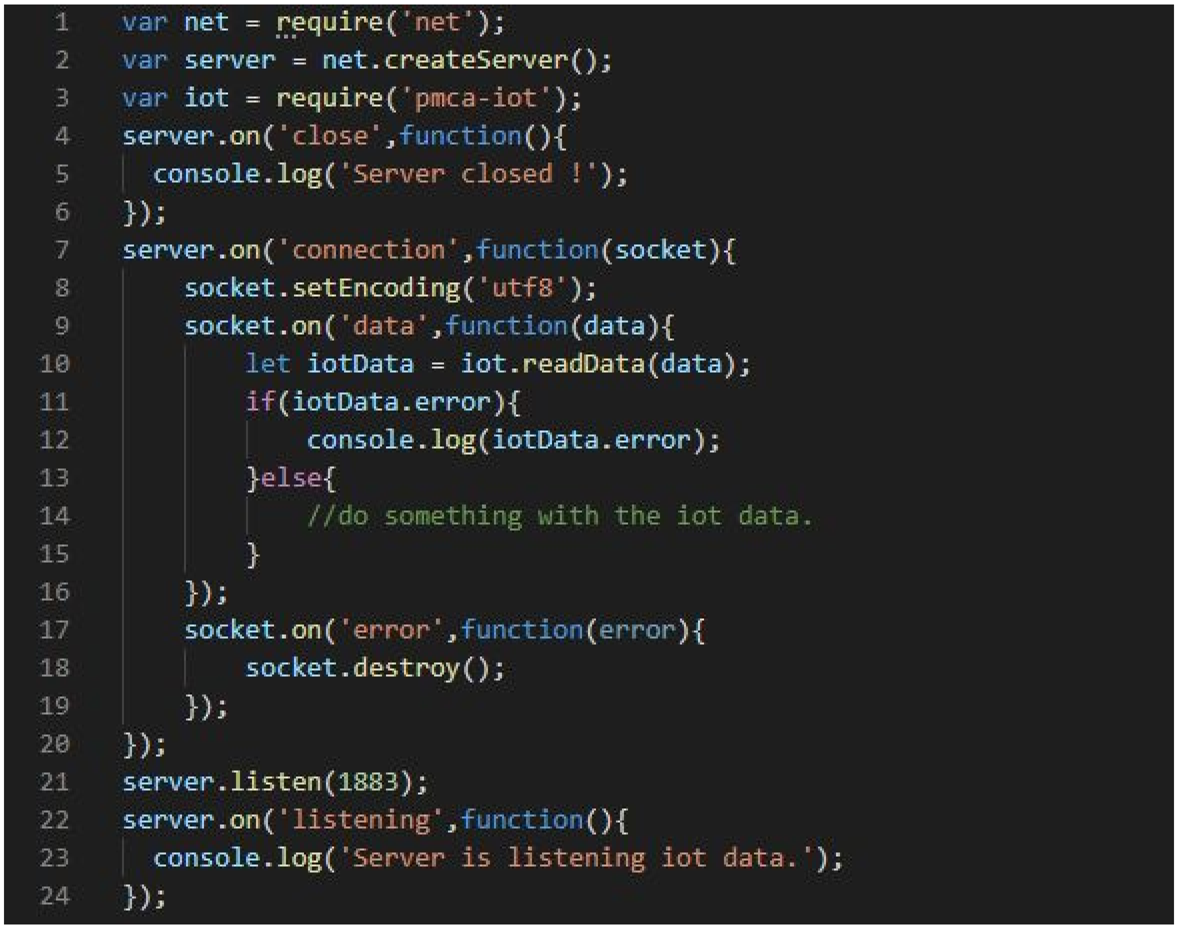
The dash boards are easily customized and a wide visual widgets and options are included



CUSTOM DEVELOPMENT

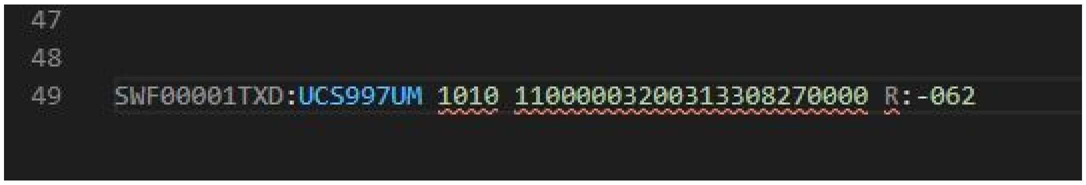
The last option we present here is based on custom development using JavaScript running in **Node.js** https://nodejs.org/en/ but that doesn't mean there aren't more, the fact is there are a lot of options for IoT platforms.

Below sample javascript code that opens a socket and listen to the incoming packet - message



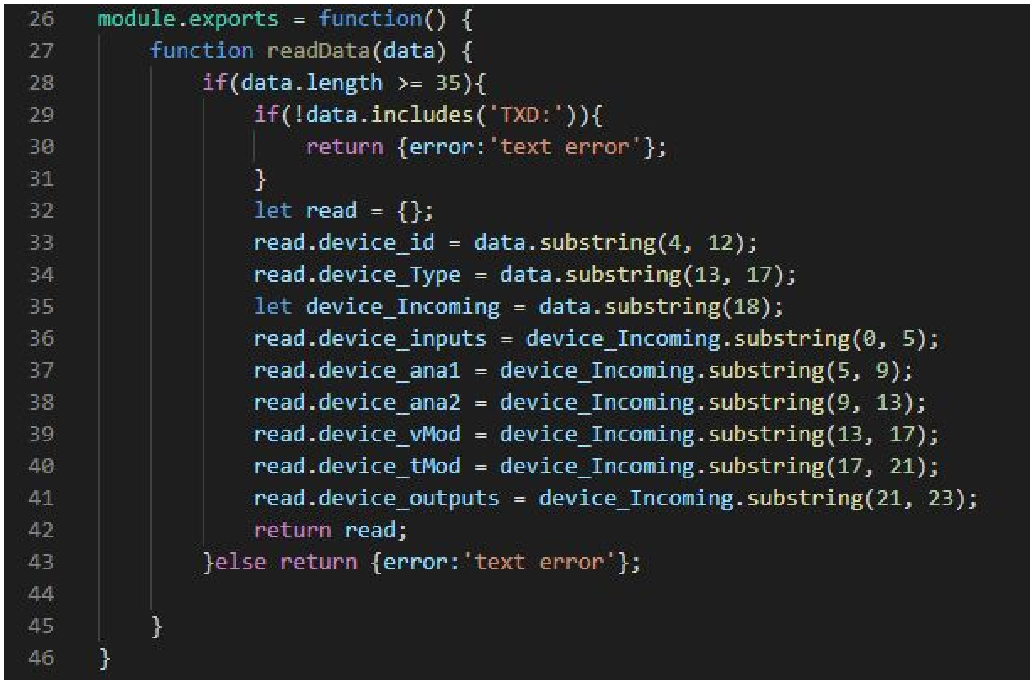
```
var net = require('net');
var server = net.createServer();
var iot = require('pmca-iot');
server.on('close',function(){
  console.log('Server closed !');
});
server.on('connection',function(socket){
    socket.setEncoding('utf8');
    socket.on('data',function(data){
        let iotData = iot.readData(data);
        if(iotData.error){
            console.log(iotData.error);
        }else{
            //do something with the iot data.
        }
    });
    socket.on('error',function(error){
        socket.destroy();
    });
});
server.listen(1883);
server.on('listening',function(){
  console.log('Server is listening iot data.');
});
```

This is a sample package in plain format where you can see for instance the Gateway ID, Sensor ID, battery voltage level and temperature and even the wireless signal strength.



```
SWF00001TXD:UCS997UM 1010 11000003200313308270000 R:-062
```

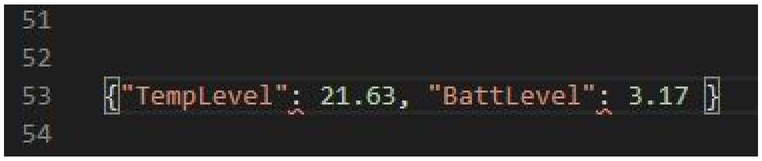
Finally a sample code which processes and extract the values from the above message.



```
module.exports = function() {
    function readData(data) {
        if(data.length >= 35){
            if(!data.includes('TXD:')){
                return {error:'text error'};
            }
            let read = {};
            read.device_id = data.substring(4, 12);
            read.device_Type = data.substring(13, 17);
            let device_Incoming = data.substring(18);
            read.device_inputs = device_Incoming.substring(0, 5);
            read.device_ana1 = device_Incoming.substring(5, 9);
            read.device_ana2 = device_Incoming.substring(9, 13);
            read.device_vMod = device_Incoming.substring(13, 17);
            read.device_tMod = device_Incoming.substring(17, 21);
            read.device_outputs = device_Incoming.substring(21, 23);
            return read;
        }else return {error:'text error'};

    }
}
```

Other widely used format of the packet / message is JSON



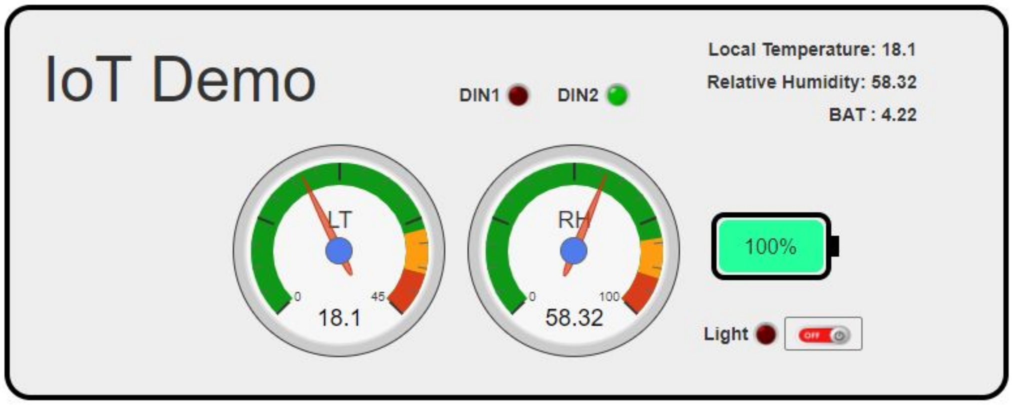
```
{"TempLevel": 21.63, "BattLevel": 3.17 }
```

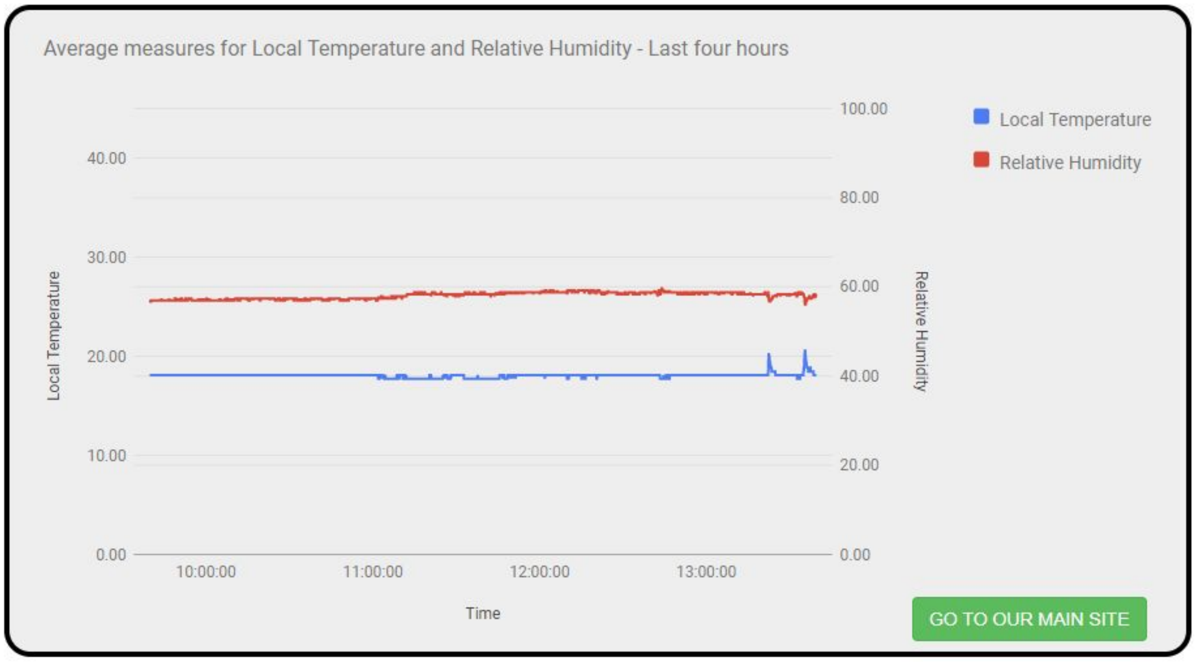
This format is widely used among platforms. It is not very efficient especially where a lot of parameters must be sent wirelessly in low power devices but it is easily processed by all IoT platforms and systems.

Sample Site

You can access an IoT sample running in PHP and Node.js. This shows temperature and humidity sensors. You can **turn on a light** which in turn heats the sensors making the temperature increase and the humidity decrease. In the same page you can observe the Temperature and Humidity graphics.

http://www.powermicrocontrol.com/index.php?file=iot&function=iot_demo

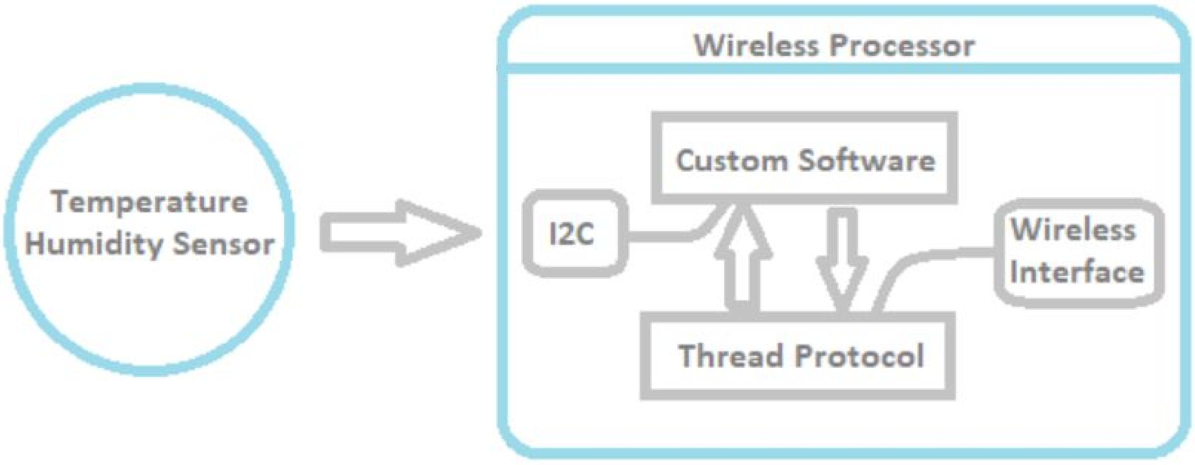




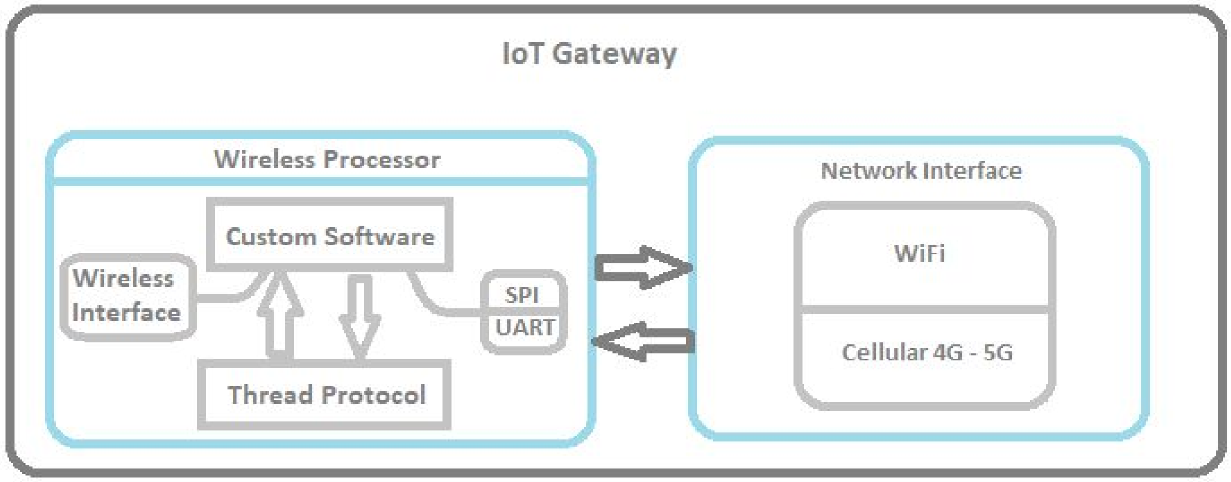
Some technicals insites

The low power network uses the **thread protocol** https://openthread.io/ which forms an IPV6 mesh network. Using **COAP** gateway and sensors communicate with each other. In general all messages go to the gateway however you can route messages between sensors with some extra software. The COAP protocol works in a very similar way to the UDP. The messages have different formats, some are plain text with each character representing status or a measurement, and others may be binary. All the messages between the gateway and the sensors are encrypted as per the thread protocol.

The sensors normally have general Digital Input / Outputs as well as low voltage analog inputs. The temperature used in the application presented in this document used a Digital temperature sensor connected thru I2C to the processor which in turn runs the **Thread Protocol** and all features integrated in it. The sensor automatically connects to the gateway and also reconnects in case of a lost connection. Also reports the wireless signal level to the gateway for link diagnostic purposes.



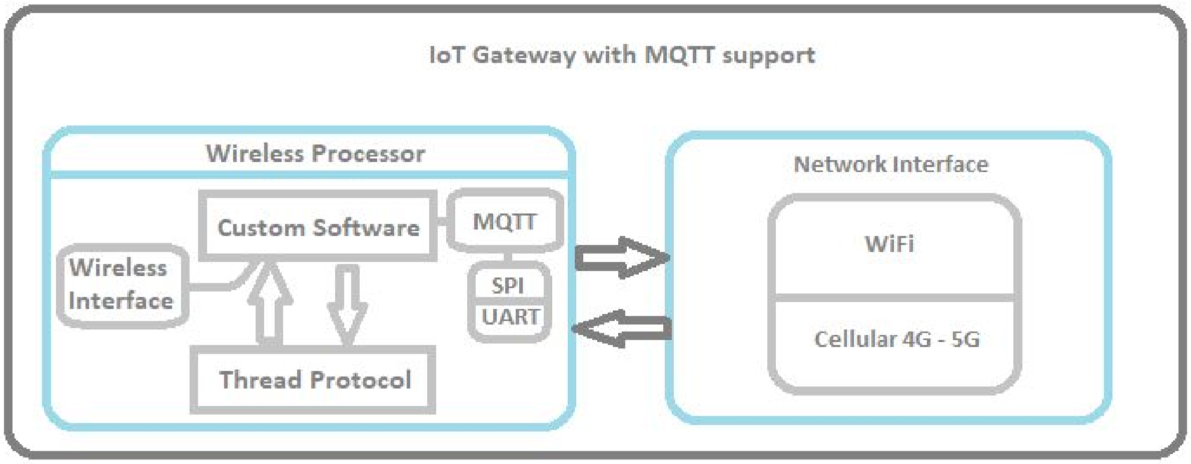
The gateway is built with two interfaces, one is the Thread Protocol Interface and the other is the network - Internet Interface that can be a Cellular Interface or a WiFi Interface. The communication between the two interfaces is done via UART or SPI. The Thread Interface communicates with the sensors while the Network Interface connects the sensors to the IoT Platform.



The Gateway transmits data to the cloud in different formats as mentioned before plain text binary or ASCII characters are sent to interact with the IoT platform. We mentioned that JSON is a common format for messages widely use to communicate with an IoT Platform or simply with a Web Page to present sensors data.

Another common protocol that is used in the IoT space is MQTT which is a twenty year old protocol that is having a revival as a lot of IoT applications are dedicated to sensors and this protocol seems to fit perfectly for the task.

In the present document we showed a couple of standard platforms, one is Node-Red, the other Thingsboard, both use MQTT as the communication protocol. In the example shown here, the gateway was modified to support the MQTT protocol to communicate with the platforms. All messages are encapsulated (serialized / deserialized) to communicate with the IoT Platform / IoT Cloud. The gateway also takes care of reconnecting the MQTT session in case of a communication failure.



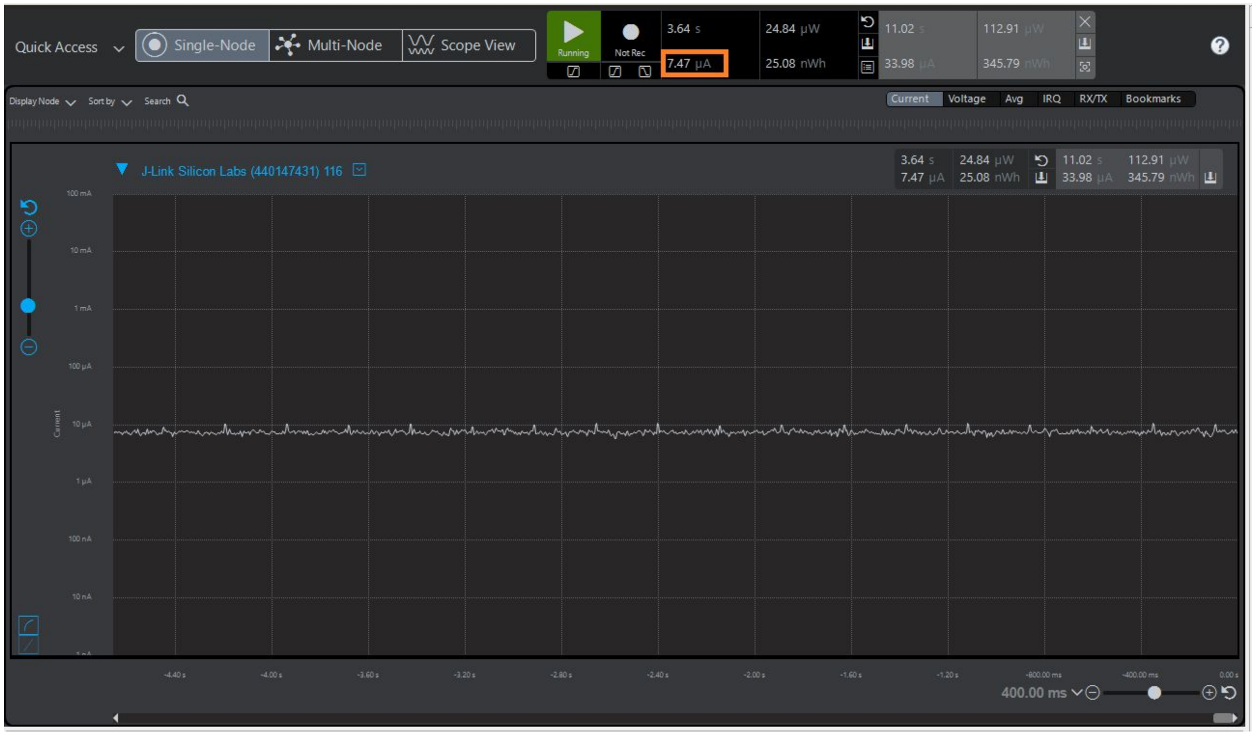
Special Mention - Low Power

One of the challenges in IoT is to minimize consumption. Battery operation is possible but several conditions must be evaluated: Transmission time, Transmission Power, Report cycle, battery size among others.

The transmission-reception and antenna circuit are critical. An optimal transmission-reception circuit will allow us to use lower transmission power and contribute to longer battery life.

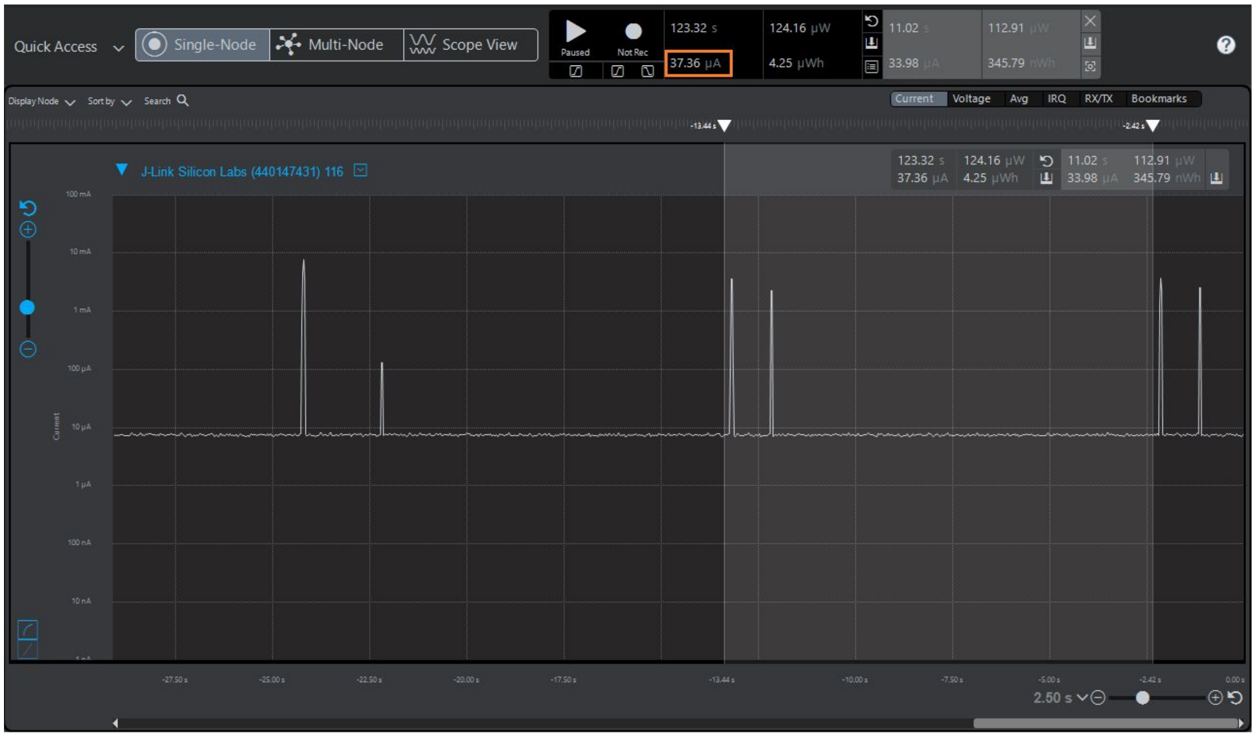
In our design we use the EFR32MG21A020 from silabs which has low power consumption in the sleep mode. The transmission consumption can go as high as 200ma when used with 20dbm power transmission output and all sensors / peripherals are activated. Minimizing the time to acquire the sensed magnitudes and time to communicate with the gateway will improve battery live.

Battery consumption In sleep Mode



7.47 micro ampere in sleep mode

Average battery consumption at 0 dbm transmission power
10 seconds transmission cycle



Average 37.4 micro ampere consumption

A CR2032 battery https://data.energizer.com/pdfs/cr2032.pdf has a 235 mah capacity. At an average of 37.4 micro ampere consumption will last ideally 6283 hours which is around 260 days. If we add more time between transmissions obviously the battery life will be extended. Also the peak consumption in this case is 0 dbm but as we go higher in transmission power consumption rises and a lot. It’s important to know the power consumption of sensors in advance and select the sensors carefully according to the application to optimize battery life.

Conclusion

We here presented how to build a basic IoT system using wireless sensors based on the Thread Protocol, a gateway to connect the IoT system which encapsulates all messages using the MQTT protocol and communicating with a standard or custom IoT platform. We also presented a couple of standard IoT platforms, Thingsboard , Node-Red and some hints on using Node.JS to develop custom Platform. At the final part we showed some information on hardware details and power consumption.

Sergio Rabin
R&D Manager
Power Micro Controls America SRL
www.powermicrocontrols.com
info@powermicrocontrols.com
Argentina