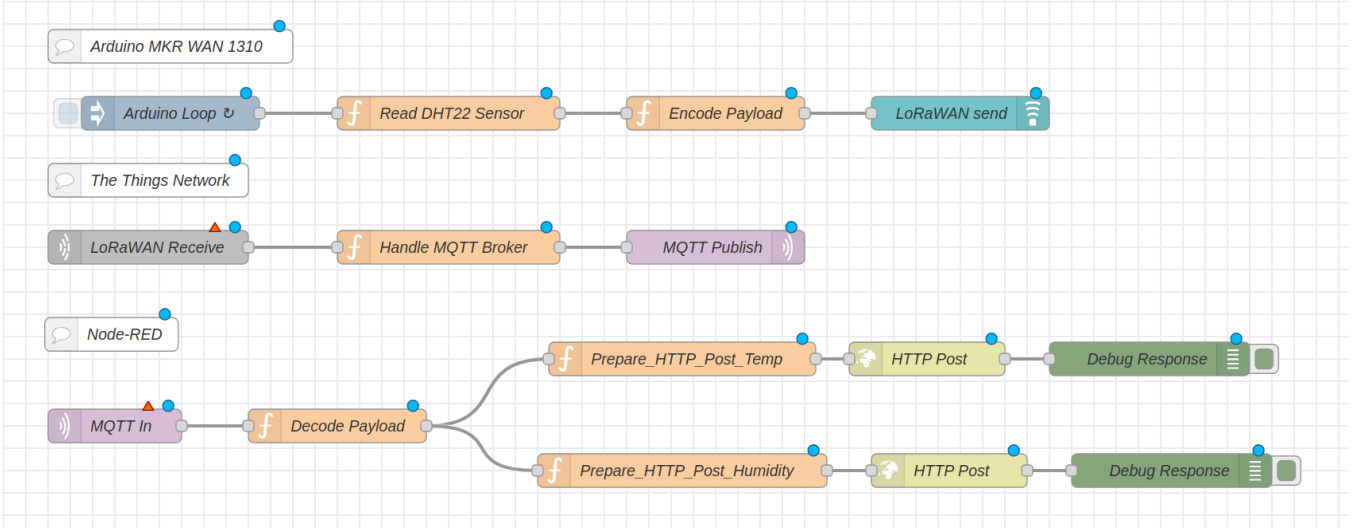
EQ2

> Use an Arduino MKR WAN 1310 to create a system that reads temperature and humidity data from a DHT22 sensor and sends this data wirelessly to ThingSpeak over LoRaWAN.
>
> Design a complete system block diagram (sketch in Node-Red) for the implementation

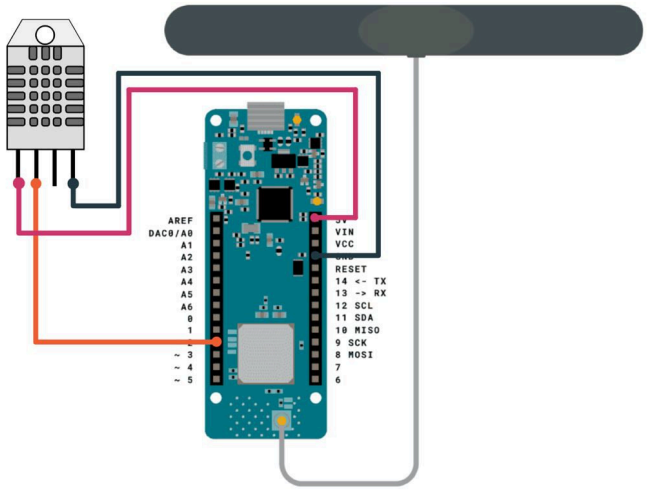
Main Idea

The main idea to achieve communication between the Arduino MKR WAN using LoRaWAN communication protocol and ThingSpeak is to pass through a gateway and an application server, such as the ones offered by *The Things Network*, in order to be able to send packets to a Node-RED local server. This can be done using the MQTT broker integrations available inside TTN.

A flow diagram of the entire procedure is described below:



Circuit Design



Arduino and The Things Network

To be able to use TTN, it's necessary to create an account on the website, create a new application and **add a new device**. After selecting the right Arduino model, a **Extended Unique Identifier** is request. In order to retrieve this identifier, it's necessary to run the `FirstConfiguration` program which is available inside the **MKRWAN library**.
Once found the EUI, it's possible to complete the device registration on TTN and retrieve the AppKey which has to be used to authenticate to the service.

Once these initial configuration steps are done, it's possible to proceed by writing the Arduino sketch to read the temperature and send it *uplink* to the TTN gateways and servers:

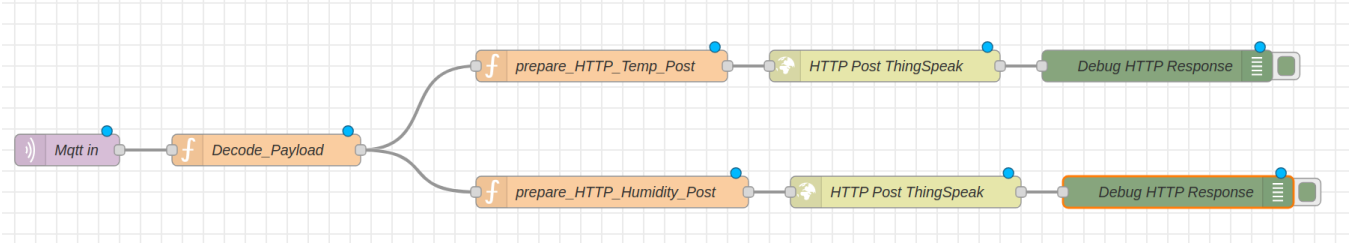
```
#include <MKRWAN.h>
#include <DHT.h>

// DHT22 Setup
DHT dht(2, DHT22);

// LoRaWAN Setup
LoRaModem modem;
String appEui = "XXXXXXXXXXXXXXXX";  // TTN AppEUI
String appKey = "XXXXXXXXXXXXXXXX";  // TTN AppKey

void setup() {
    Serial.begin(9600)

    dht.begin();

    // Try to connect using the European propietary bandwidth
    if (!modem.begin(EU868)) {
        Serial.println("Failed to start module");
    while (1){}
}

if (!modem.joinOTAA(appEui, appKey)) {
    Serial.println("Failed to connect");
    while (1) {}
  }
  Serial.println("Connected to network");
}

void loop() {
    //Read the DHT22 sensor using the appropiate library
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();

    // Encode the 2 floats so that they are easier to transmit
    byte payload[4];

    // Cast the floats into ints keepin a single decimal
    int tempInt = (int)(temperature * 10);
    int humInt = (int)(humidity * 10);

    // Build the payload by filling it with bytes composing:
    // Temperature
    payload[0] = highByte(tempInt);
    payload[1] = lowByte(tempInt);

    // Humidity
    payload[2] = highByte(humInt);
    payload[3] = lowByte(humInt);

    modem.beginPacket();
    modem.write(payload, 4);

  // Delay a minute before sending again data
  delay(60000);
}
```

Temperature and humidity are given by the `DHT` library as 2 floats with single decimal precision. Converting them into integers and splitting the results in 2 bytes each will make them easier to decode later, being the payload encoded in HEX format

The Things Network and Node-RED

It's possible to configure TTN in order to **expose a MQTT server** which can be used to **interact with Node-RED** using the `MQTT in` node, which must be configured to use TTN exposed address and port through the **add new MQTT broker** setting page, and to use the provided username and password inside the **security** tab.

It's now possible to use Node-RED to create a *flow* in order to parse received payloads and send an `HTTP Post` to ThingSpeak, such as:



where **Decode_Payload** converts back the HEX-formatted payload into two float values, which are sent through `msg` to two different subflows that will prepare the payload to send both temperature and humidity data to ThingSpeak.