from time import sleep  
from SX127x.LoRa import \*  
from SX127x.board\_config import BOARD  
import paho.mqtt.client as mqtt

# for Cayenne MQTT

username = "20f70690-4976-11ea-84bb-8f71124cfdfb"  
password = "3d7eaaf9a7c9e28626fcab4ec5a61108cfbb8be0"  
clientid = "cccb41b0-4977-11ea-b73d-1be39589c6b2"

mqttc = mqtt.Client(client\_id=clientid)  
mqttc.username\_pw\_set(username, password=password)  
mqttc.connect("mqtt.mydevices.com", port=1883, keepalive=60)  
mqttc.loop\_start()

# Topics for MQTT

topic\_dht11\_temp = "v1/" + username + "/things/" + clientid + "/data/1"  
topic\_dht11\_humidity = "v1/" + username + "/things/" + clientid + "/data/2"

BOARD.setup()

# LoRa Class default 433MHz with 125 kHz bandwidth.

class LoRaRcvCont(LoRa):

    def \_\_init\_\_(self, verbose=False):  
        super(LoRaRcvCont, self).\_\_init\_\_(verbose)  
        self.set\_mode(MODE.SLEEP)  
        self.set\_dio\_mapping([0] \* 6)

    def start(self):  
        self.reset\_ptr\_rx()  
        self.set\_mode(MODE.RXCONT)  
        while True:  
            sleep(.5)  
            rssi\_value = self.get\_rssi\_value()  
            status = self.get\_modem\_status()  
            sys.stdout.flush()         

    def on\_rx\_done(self):  
        print ("\nReceived: ")  
        self.clear\_irq\_flags(RxDone=1)  
        payload = self.read\_payload(nocheck=True)

#print (bytes(payload).decode("utf-8",'ignore'))  
        data = bytes(payload).decode("utf-8",'ignore')  
        print (data)  
        temp = (data[0:4])  
        humidity = (data[4:6])  
        print ("Temperature:")  
        print (temp)  
        print ("Humidity:")  
        print (humidity)

        mqttc.publish(topic\_dht11\_temp, payload=temp, retain=True)  
        mqttc.publish(topic\_dht11\_humidity, payload=humidity, retain=True)

        self.set\_mode(MODE.SLEEP)  
        self.reset\_ptr\_rx()  
        self.set\_mode(MODE.RXCONT) 

lora = LoRaRcvCont(verbose=False)  
lora.set\_mode(MODE.STDBY)

print lora.get\_version() # this prints the sx127x chip version

print lora.get\_freq() # this prints the frequency setting

lora.set\_freq(433.0)

set\_bw(125) # bandwidth

#  Medium Range  **Defaults after init are 434.0MHz**, Bw = 125 kHz, Cr = 4/5, Sf = 128chips/symbol, CRC on 13 dBm  
lora.set\_pa\_config(pa\_select=1)

try:  
    lora.start()  
except KeyboardInterrupt:  
    sys.stdout.flush()  
    print ("")  
    sys.stderr.write("KeyboardInterrupt\n")  
finally:  
    sys.stdout.flush()  
    print ("")  
    lora.set\_mode(MODE.SLEEP)  
    BOARD.teardown()

**Arduino Code:**

#include <SPI.h>  
#include <RH\_RF95.h>  
#include "DHT.h"

#define DHTPIN A0     // what pin we're connected to  
#define DHTTYPE DHT11   // DHT type

DHT dht(DHTPIN, DHTTYPE);  
RH\_RF95 rf95;

void setup()   
{  
  Serial.begin(9600);  
  dht.begin();  
  if (!rf95.init())  
    Serial.println("init failed");  
  // Defaults after init are 434.0MHz, 13dBm, Bw = 125 kHz, Cr = 4/5, Sf = 128chips/symbol, CRC on

}

void loop()  
{

  int temp = dht.readTemperature();  
  int hum = dht.readHumidity();  
  String humidity = String(hum); //int to String  
  String temperature = String(temp);  
  String data = temperature + humidity;  
  Serial.print(data);

char d[5];  
  data.toCharArray(d, 5); //String to char array  
  Serial.println("Sending to rf95\_server");  
  rf95.send(d, sizeof(d));  
  rf95.waitPacketSent();  
    
  delay(400);  
}

from time import sleep

from SX127x.LoRa import \*

from SX127x.board\_config import BOARD

import paho.mqtt.client as mqtt

Now enter the required credentials for MQTT communication between Raspberry Pi and Cayenne.

username = "20f70690-4976-11ea-84bb-8f71124cfdfb"

password = "3d7eaaf9a7c9e28626fcab4ec5a61108cfbb8be0"

clientid = "cccb41b0-4977-11ea-b73d-1be39589c6b2"

After that, create two channels for temperature and humidity values.

topic\_dht11\_temp = "v1/" + username + "/things/" + clientid + "/data/1"

topic\_dht11\_humidity = "v1/" + username + "/things/" + clientid + "/data/2"

Now create a LoRa class with three functions i.e. *init*, *class* and *on\_rx\_done*. The *init* function is used to initialize the LoRa module in 433MHz with 125 kHz bandwidth.

def \_\_init\_\_(self, verbose=False):

super(LoRaRcvCont, self).\_\_init\_\_(verbose)

self.set\_mode(MODE.SLEEP)

self.set\_dio\_mapping([0] \* 6)

 The ***start*** function is used to configure the module as a receiver and to get RSSI value, operating frequency, etc.

def start(self):

self.reset\_ptr\_rx()

self.set\_mode(MODE.RXCONT)

while True:

sleep(.5)

rssi\_value = self.get\_rssi\_value()

status = self.get\_modem\_status()

sys.stdout.flush()

The ***on\_rx\_done*** function is used to read the incoming vales. Then received values are moved into a variable called ***payload*** and decoded with utf-8 and moved into a variable called *data*.

def on\_rx\_done(self):

self.clear\_irq\_flags(RxDone=1)

payload = self.read\_payload(nocheck=True)

data = bytes(payload).decode("utf-8",'ignore')

Now split the temperature and humidity values and publish them on Cayenne platform

temp = (data[0:4])

humidity = (data[4:6]

mqttc.publish(topic\_dht11\_temp, payload=temp, retain=True)

mqttc.publish(topic\_dht11\_humidity, payload=humidity, retain=True)

**Establishing Communication between Arduino & Raspberry Pi**

Once the hardware and program are ready, upload the Arduino code to the Arduino board and launch the python code in Pi. If everything is working properly, you should see the temperature and humidity values received in pi though the shell window, as shown below.

