Setup & Operation

Table of Contents

1. Introduction

The following document contains information about setting up a LoRa Iot network and connecting its gateway to the Takano platform via the internet, then reading the Lora end nodes sensor data on the Takano mobile app. A quick word about Lora: it is a long-range protocol that achieves long distance connectivity (+10Km) by trading off data rate and limiting duty cycles. It is suitable in practice for non-real time applications that can tolerate small transmission delays. It uses the unlicensed frequency bands of either 433Mhz, 868Mhz (Europe), or 915Mhz (North America). A typical Lora network consists of a gateway that connects to several end-nodes as in Fig 1. It also features a low-power consumption that is particularly adequate for battery-powered remote hardware that reads sensory data and relays it to the gateway, and supports two-way communication between gateway and end-nodes i.e. the gateway can send commands to an end node to activate a relay for example.

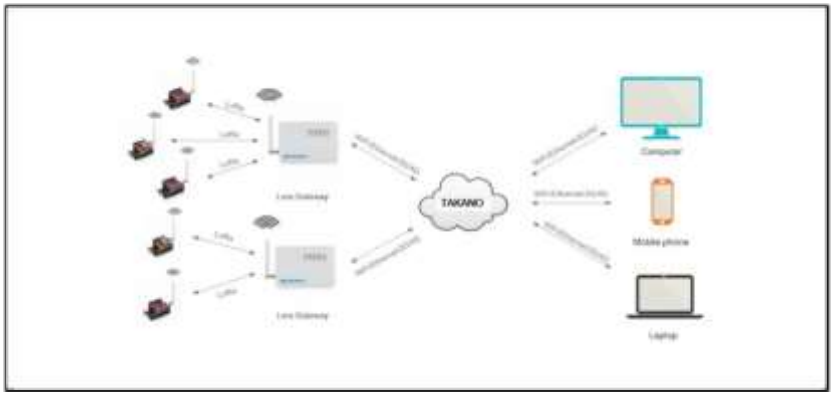


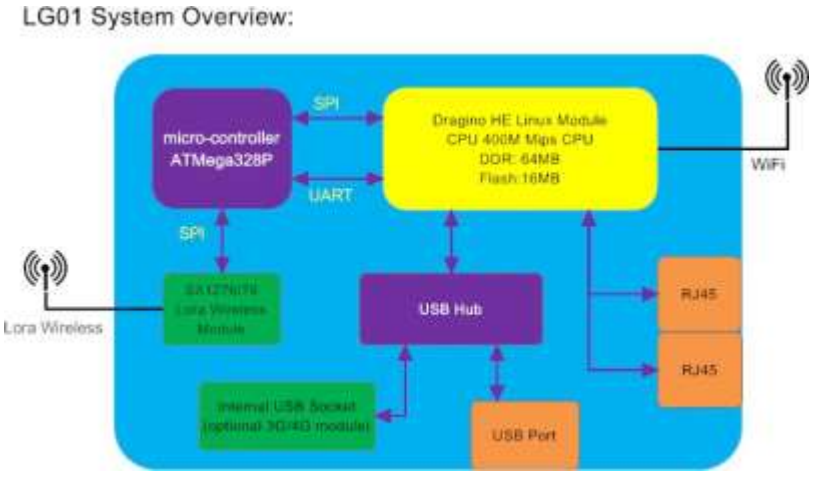
Fig 1.

The gateway we are going to use in our example is the open-source Dragino LG01-P due to its reasonable cost (few tens of dollars), its wide availability among major on-line retail suppliers (eBay, Amazon, Aliexpress) and its multi-connectivity offerings, that is Ethernet, Wi-Fi and even optional 3G/4G via sim card insertion (https://www.dragino.com/products/lora/item/117-lg01-p.html). It is suitable for a Lora network of up to 100 nodes. We will also use two open-souce Arduino Uno based end-node hardware LoRa shields also made by Dragino (http://www.dragino.com/products/lora/item/102-lora-shield.html), one that reads both temperature and humidity from a DHT11 sensor, and the other one a push button digital input and activates a led from a digital output pin. Before buying, just make sure you select the right hardware frequency band for the Dragino gateway and the Arduino shields depending on the country you live in. The Dragino and the Arduino Uno with its mounted Lora shield are pictured below.

Dragino gateway LG01-P

Arduino Uno and Lora shield

The LP01-P houses a MPU running under Linux that controls the Ethernet , USB and Wi-Fi ports, and an Arduino MCU that controls the Lora module as shown below. We will connect it to an internet router by an Ethernet cable to its WAN port, and to a laptop by an Ethernet cable to its LAN port. This way will we program its embedded Arduino directly over the Ethernet connection and use the laptop usb port to program the two Arduino Uno end nodes. All programming we be done using Arduino IDE.



2. LoRa Gateway Setup

At the first boot of LG01, it will auto generate an unsecure WiFi network call ***dragino2-xxxxxx.*** But since we will be using a wired solution, we will enable our laptop to connect directly to the Ethernet LAN port of LG-01P. The laptop will get an IP address 10.130.1.xxx and the LG01 has the default IP 10.130.1.1.

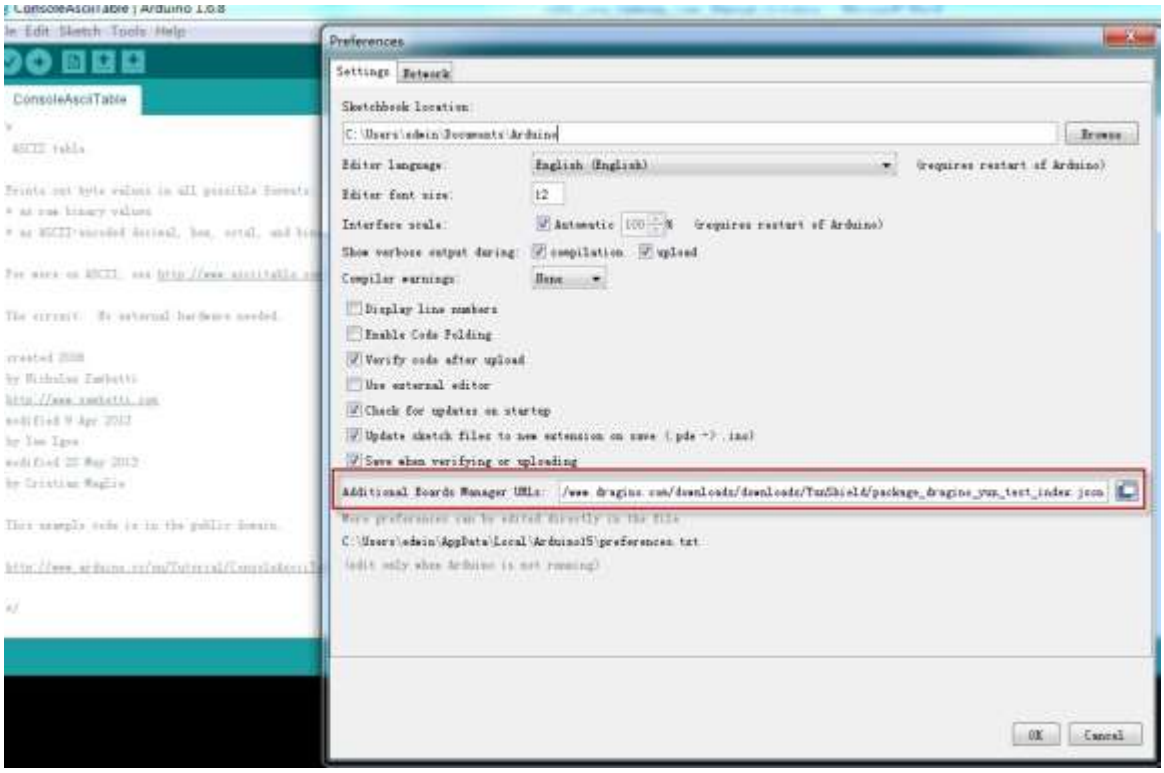
Open a browser in the laptop and type 10.130.1.1 and will see the login interface of LG01.

The account for Web Login is:

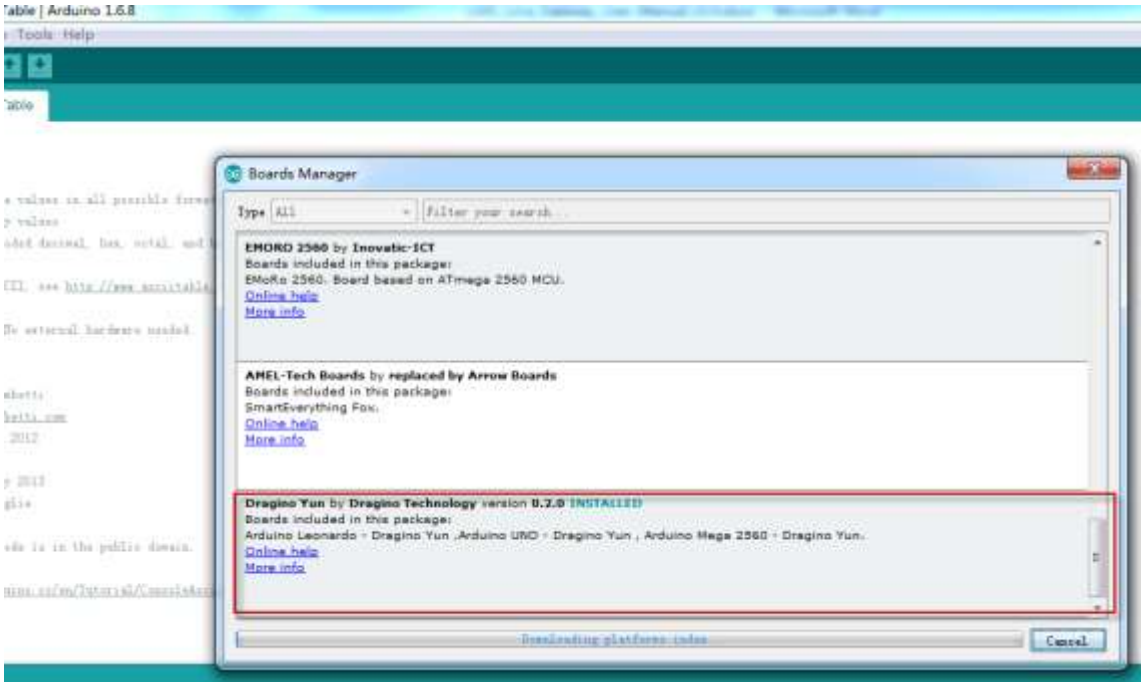
User Name: root

Password: dragino

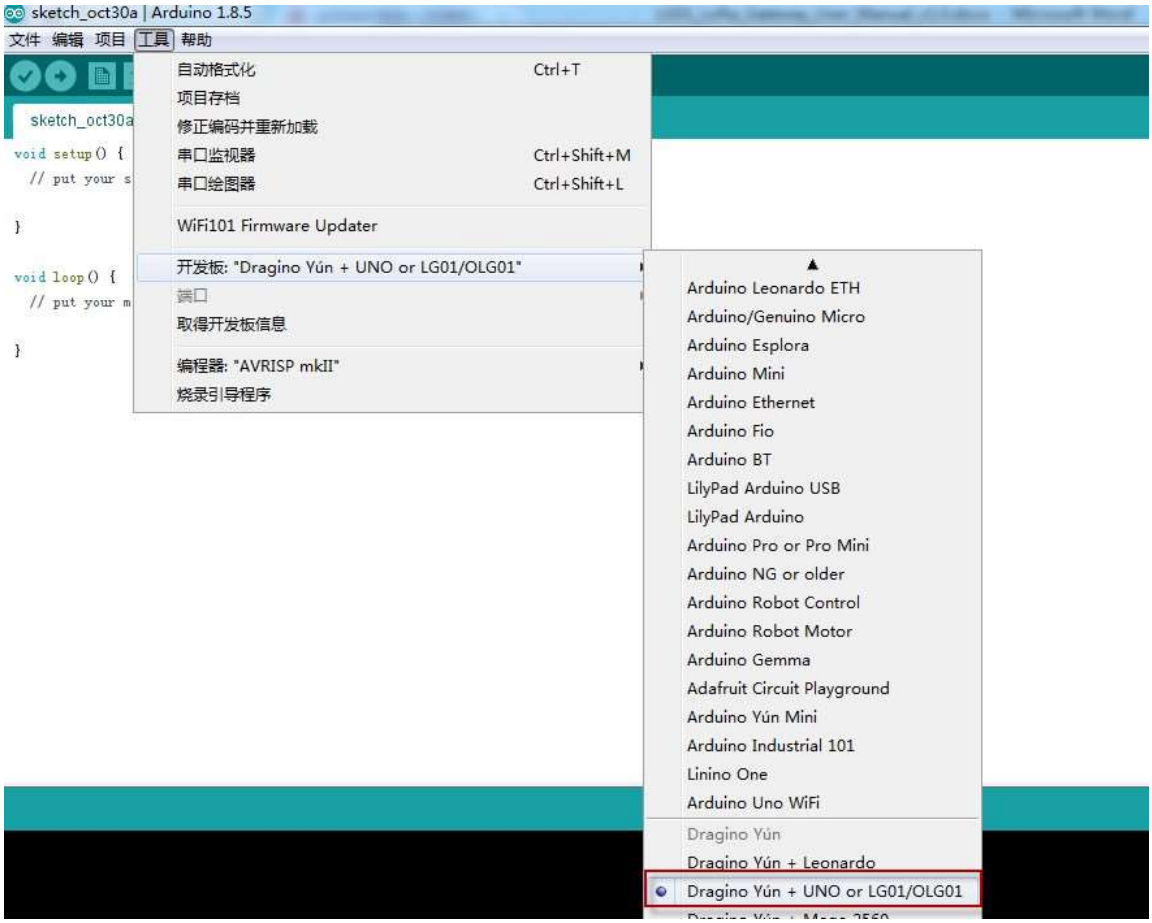
The first step is to Download the latest Arduino Software(IDE) from Arduino official site: https://www.arduino.cc/en/Main/Software. Install the IDE on the laptop, open it and click **File --> Preference**, add the below URL: http://www.dragino.com/downloads/downloads/YunShield/package_dragino_yun_test_index.json in the **Additional Boards Manager URLs**.



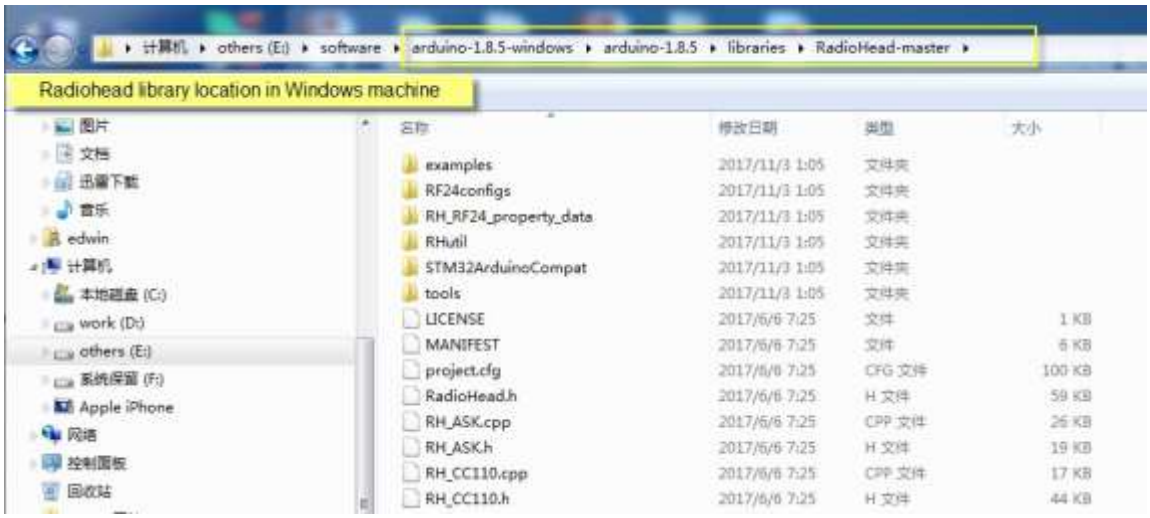
Next, Go to **tools --> Boards --> Boards Manager**, find Dragino boards info and install it.



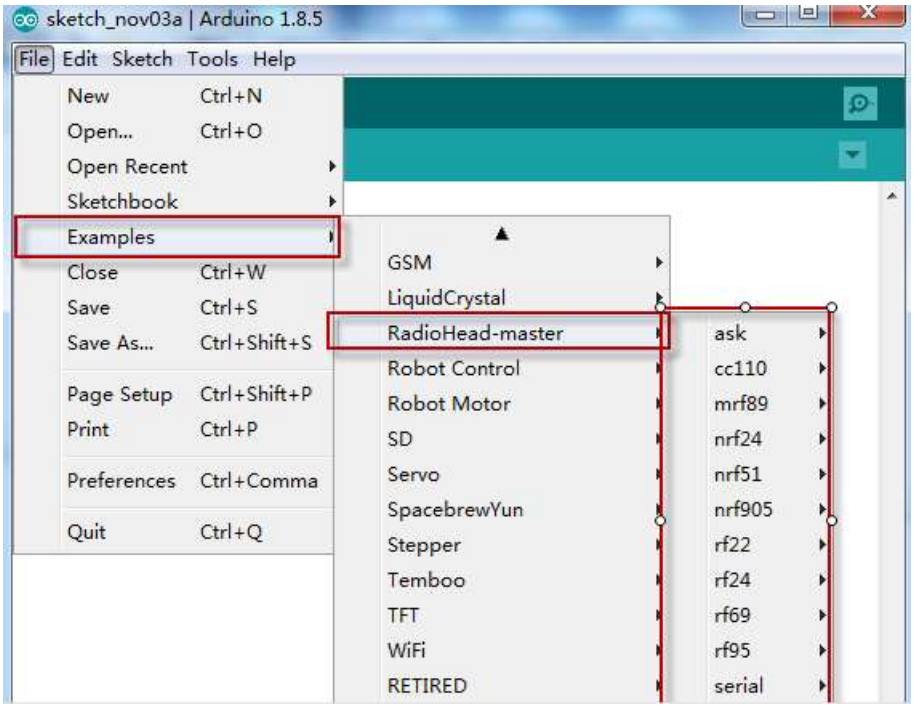
After install Dragino's board info in the IDE, we can see the boards info from the IDE, as below screenshot. For LG01, we should choose: **Dragino Yun-UNO or LG01/OLG01**.



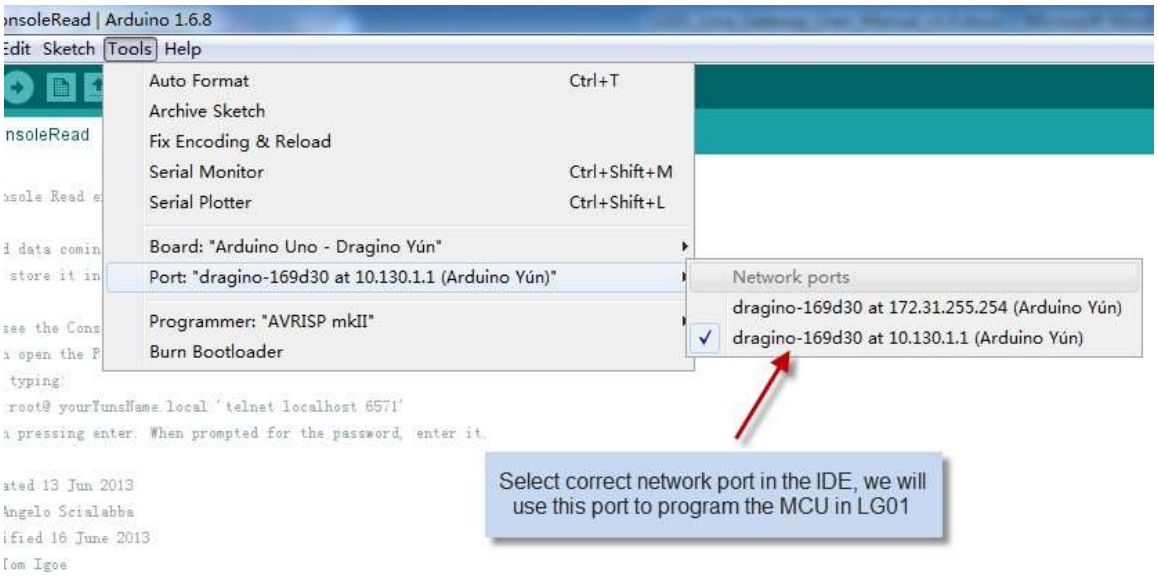
The LoRa communication library we will use in all of our sketches is called Radiohead, so we need to download it first from
https://github.com/dragino/RadioHead/archive/master.zip
Unzip and put it in the Arduino Library Folder as below:



To make sure the library is properly installed, we can restart the Arduino IDE and see if we can find it as shown below:



Since the PC and the LG01 are in the same network, in the Arduino IDE, select the correct port as in the below screenshot.



Paste the below Arduino sketch and click **Upload** to upload the sketch to LG01. The LG01 may ask you to enter its password in order to resume the upload, if so, type the password of LG01 which in our case is “dragino”.

```
/*
  Upload Data to IoT Server TAKANO (https://iotsprint.com):
  Support Devices: Dragino LG01P
  Require Library:
  https://github.com/dragino/RadioHead

  Example sketch showing how to get data from 2 remote LoRa nodes,
  node1 sends switch state (0 or 1) and node2 sends temperature & humidity
  data from both end nodes is sent to the Takano cloud platform, which will
  send back LED state (0 or 1) that is relayed to end node1 to turn on or off the led
  this LED state can be set/reset by a registered user through a mobile app
*/

#include <SPI.h>
#include <RH_RF95.h>
#include <Console.h>
#include <Process.h>
RH_RF95 rf95;

#define BAUDRATE 115200

String myRegKey = "witjmvmox0x9"; //hardware registration key from TAKANO platform
float frequency = 868.0; //868Mhz Lora frequency
String dataString = "";
char My_Array[100];
String led_data;
uint8_t LED;
int first;
```

```
int hh=0;
int hl=0;
int th=0;
int tl=0;
int count;
int i;
int char_count;
int TAKANO;
long timer1_counter;
char LED_READING[2] = {0,1};
int button_state=0;
int previous_button_state=0;
int send_push_notification=0;

void uploadData(); // Upload Data to Takano

int HEART_LED=A2; //this is the 'HEART' led on LP01P enclosure

void setup()
{
LED=0;
count= 0;
TAKANO= 0;
pinMode(HEART_LED, OUTPUT); //Heart Led on DRAGINO is digital output
////////////////////// SETUP TIMING INTERRUPT EVERY 1second
  noInterrupts(); // disable all interrupts
  TCCR1A = 0;
  TCCR1B = 0;

  // Set timer1_counter to the correct value for our interrupt interval
  timer1_counter = 3036;   // preload timer 65536-16MHz/256/1Hz

  TCNT1 = timer1_counter;   // preload timer
  TCCR1B |= (1 << CS12);    // 256 prescaler
  TIMSK1 |= (1 << TOIE1);   // enable timer overflow interrupt
  interrupts();             // enable all interrupts
////////////////////// END OF TIMING INTERRUPT SETUP

digitalWrite(HEART_LED, HIGH);   // turn the HEART_LED on (HIGH is the voltage level)
    Bridge.begin(BAUDRATE);
    Console.begin();
    // while(!Console);
    if (!rf95.init())
        Console.println("init failed");
    // Setup ISM frequency
    rf95.setFrequency(frequency);
    // Setup Power,dBm
    rf95.setTxPower(13);
    rf95.setSyncWord(0x34);

    Console.println("LoRa Gateway Example  --");
    Console.println("     Upload Single Data to Takano Platform");
}
```

```
ISR(TIMER1_OVF_vect) //timer interrupt service routine every 1 second
{
TCNT1 = timer1_counter;   // preload timer
digitalWrite(HEART_LED, !digitalRead(HEART_LED)); //toggle heart led on DRAGINO
count++;
if ((count > 9) && (TAKANO != 1))
{
  count=0;
  TAKANO=1;
}
}

void loop()
{
   if (TAKANO == 1){
    uploadData();  //Upload Data to TAKANO Platform & read its feedback
    count=0;
    TAKANO=0;
   }
    if (rf95.waitAvailableTimeout(2000))// Listen Data from LoRa Node
    {
        uint8_t buf[RH_RF95_MAX_MESSAGE_LEN]; //receive data buffer
        uint8_t len = sizeof(buf); //data buffer length

        if (rf95.recv(buf, &len))  //Check if there is incoming data
        {
            Console.print("Get LoRa Packet: ");
            for (int i = 0; i < len; i++)
            {
                Console.print(buf[i],HEX);
                Console.print(" ");
            }
            Console.println();
                if(buf[0] == 1 && buf[1] == 1 && buf[2] ==1) //Check if the ID match the LoRa Node ID
                {
                    uint8_t data[] = "    ACK1"; //Reply
                    data[0] = buf[0];
                    data[1] = buf[1];
                    data[2] = buf[2];
                    data[3] = LED; //0 or 1 is relay state
                    rf95.send(data, sizeof(data)); //Send Reply to LoRa Node
                    rf95.waitPacketSent();
                    int newData[4] = {0, 0, 0, 0}; //Store Sensor Data here
                    for (int i = 0; i < 4; i++)
                    {
                        newData[i] = buf[i + 3];
                    }
                    dataString = "T="; //temperature
                    dataString +=th;
                    dataString +=".";
                    dataString += tl;
                    dataString +=",H="; //humidity
                    dataString += hh;
                    dataString +=".";
```

```
            dataString += hl;
            dataString +=",BUTTON="; //toggle button state
            if (buf[4] == LED_READING[1])
            {
             button_state=1;
             if (previous_button_state == 0) send_push_notification=1;
             dataString += "1";
            }
            else
            {
             button_state=0;
             dataString += "0";
            }
            previous_button_state= button_state;
          }
          else if(buf[0] == 0 && buf[1] == 1 && buf[2] == 1) //Check if the ID match the LoRa
Node ID
          {
            uint8_t data[] = "    ACK2";//Reply
            data[0] = buf[0];
            data[1] = buf[1];
            data[2] = buf[2];
            rf95.send(data, sizeof(data)); // Send Reply to LoRa Node
            rf95.waitPacketSent();
            int newData[4] = {0, 0, 0, 0}; //Store Sensor Data here
            for (int i = 0; i < 4; i++)
            {
               newData[i] = buf[i + 3];
            }
            hh = newData[0];
            hl = newData[1];
            th = newData[2];
            tl = newData[3];
            Console.print("Get Temperature:");
            Console.print(th);
            Console.print(".");
            Console.println(tl);
            Console.print("Get Humidity:");
            Console.print(hh);
            Console.print(".");
            Console.println(hl);

            dataString ="T="; //temperature
            dataString += th;
            dataString +=".";
            dataString += tl;
            dataString +=",H="; //humidity
            dataString += hh;
            dataString +=".";
            dataString += hl;
            dataString +=",BUTTON="; //toggle button state
            if (button_state == 1)
             dataString += "1";
            else
```

```
                dataString += "0";
            }

        }
        else
        {
            //Console.println("recv failed");
            ;
        }
    }
}

void uploadData() //Upload Data to TAKANO Platform & read its feedback
{
  if (dataString != "")
  {
  String upload_url = "http://iotsprint.com/tk.php?r=";
  upload_url += myRegKey;
  upload_url += "&m=";
  if (send_push_notification == 1)
  {
    send_push_notification=0;
    upload_url += "Lora_Input_Alarm|p";
    Console.println("Sending Push Notification To Takano");
  }
  else upload_url += dataString;
  dataString="";

  Console.println("Call Linux Command to Send Data");
  Process p;    // Create a process and call it "p", this process will execute a Linux curl command
  p.begin("curl");
  p.addParameter("-k");
  p.addParameter(upload_url);
  p.run();    // Run the process and wait for its termination

  Console.print("Feedback from Linux: ");
  // If there's output from Linux, send it out the Console:
  char_count=0;
  while (p.available()>0)
  {
    char c = p.read();
    Console.write(c);
    My_Array[char_count++] = c;
  }
    //check LED status from user feedback
    led_data="";
for (i=0;i<96;i++)
 {
  if ((My_Array[i] == 'L') && (My_Array[i+1] == 'E') && (My_Array[i+2] == 'D') && (My_Array[i+3] ==
'=')) led_data=My_Array[i+4];
 }

  if (led_data == "1") LED=1;
  else LED=0;
```

```
  Console.println("");
  Console.println(led_data);
  Console.println("Call Finished");
  Console.println("########################################");
  Console.println("");
  }
  else
  {
  Console.println("Skipping Takano Call because no data to send");
  Console.println("");
  }
}
```

3. LoRa End Nodes Setup

The first Arduino end node is shown in Fig 2.

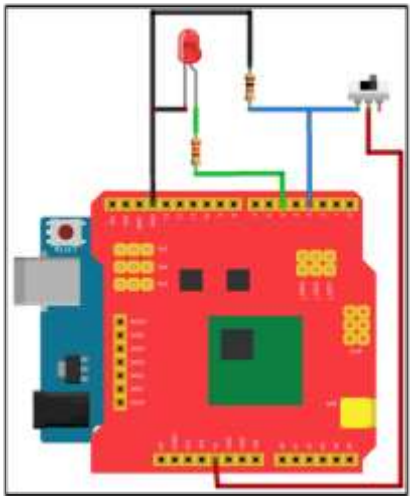


Fig 2.

It will send the on/off switch status to the LG01P gateway periodically every 10 seconds. The LG01P will in turn send this status to the Takano platform, and receive registered user(s) feedback in return from through the cloud, specifically whether to turn on or off the led. A user with a mobile app can read the switch state and in parallel send a command to turn on or off the led. When the switch is activated to 1, it will instruct the LP01P to immediately send a push notification to the user(s) without waiting for the end of the 10 seconds polling. Start by connecting it to a USB port on the laptop. Then make sure you select “Arduino/Genuino Uno” board and the right USB port from the Tools menu in the Arduino IDE before uploading the below sketch.

```
/*
  Require Library:
  https://github.com/dragino/RadioHead

  Upload Data to IoT Server TAKANO (https://iotsprint.com):
```

```
  Support Devices: Dragino Lora shield v1.4 + Arduino Uno

  Example sketch showing how to send on/off switch status to the LP01P LoRa Gateway every
10s
  The LoRa Gateway will in return send this value to the Takano server
  The Takano server will reply to the Lora Gateway the state of the led which in turn will
  send it to the Arduino running this sketch that will turn it on or off
*/

#include <SPI.h>
#include <RH_RF95.h>

RH_RF95 rf95;

char node_id[3] = {1,1,1}; //this sketch LoRa End Node ID
char LED[2] = {0,1}; //LED[0] is led off and LED[1] is led on
float frequency = 868.0; //868Mhz Lora frequency
unsigned int count = 1;
int dataLength;

int buttonState; //the switch reading from the input pin
int TAKANO;
long timer1_counter;

void setup()
{
count= 0;
TAKANO= 0;

////////////////////// SETUP TIMING INTERRUPT EVERY 1second
  noInterrupts(); // disable all interrupts
  TCCR1A = 0;
  TCCR1B = 0;

  // Set timer1_counter to the correct value for our interrupt interval
  timer1_counter = 3036;  // preload timer 65536-16MHz/256/1Hz

  TCNT1 = timer1_counter;  // preload timer
  TCCR1B |= (1 << CS12);    // 256 prescaler
  TIMSK1 |= (1 << TOIE1);  // enable timer overflow interrupt
  interrupts();             // enable all interrupts
////////////////////// END OF TIMING INTERRUPT SETUP

  pinMode(5, OUTPUT);  // sets the digital pin 5 (LED) as output
  digitalWrite(5, 0); //make sure led is off
  pinMode(3, INPUT);    // sets the digital pin 3 (BUTTON) as intput
    Serial.begin(9600);
    if (!rf95.init())
        Serial.println("init failed");
    // Setup ISM frequency
    rf95.setFrequency(frequency);
    // Setup Power,dBm
    rf95.setTxPower(13);
```

```
    rf95.setSyncWord(0x34);

    Serial.println("LoRa End Node1 Example --");
    Serial.println("LED and BUTTON");
    Serial.print("LoRa End Node ID: ");

    for(int i = 0;i < 3; i++)
    {
        Serial.print(node_id[i],HEX);
    }
    Serial.println("\n");
}

ISR(TIMER1_OVF_vect) //timer interrupt service routine every 1 second
{
TCNT1 = timer1_counter;   // preload timer
count++;
if ((count > 9) && (TAKANO != 1)) //enter this condition every 10s
{                                 //and don't enter it again till TAKANO==0
  count=0;
  TAKANO=1;
}
}

void refersgLP01()
{
  // read the state of the button into 'buttonState' variable:
  int buttonState = digitalRead(3);
    Serial.print("############     ");
    Serial.print("REFRESHING");
    Serial.println("     ############");
    char data[50] = {0} ;
    // Use data[0], data[1],data[2] as Node ID
    data[0] = node_id[0];
    data[1] = node_id[1];
    data[2] = node_id[2];
    data[3] = LED[0];
    if  (buttonState  ==  HIGH)  data[4]  =  LED[1];
    else if (buttonState == LOW) data[4] = LED[0];

    int i;
    dataLength=5; //data[0] data[1] data[2] data[3] data[4]
    for(i = 0;i < dataLength; i++)
    {
        Serial.print(data[i],HEX);
        Serial.print(" ");
    }
    Serial.println();

    unsigned char sendBuf[50]={0};

    for(i = 0;i < dataLength;i++)
    {
        sendBuf[i] = data[i] ;
```

```
    }

    rf95.send(sendBuf, dataLength+2);//Send LoRa Data

    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];//Reply data array
    uint8_t len = sizeof(buf);//reply data length

    if (rf95.waitAvailableTimeout(3000))// Check If there is reply in 3 seconds.
    {
        // Should be a reply message for us now
        if (rf95.recv(buf, &len))//check if reply message is correct
       {
            if (buf[0] == node_id[0] && buf[1] == node_id[2] && buf[2] == node_id[2]) //Check if reply
message has this end node ID
          {
              pinMode(4, OUTPUT);
              digitalWrite(4, HIGH);
              Serial.print("Got Reply from Gateway: "); //print reply
              Serial.println((char*)buf);
              if (buf[3] == LED[1]) digitalWrite(5, 1); //turn led on if user wants it
              else if (buf[3] == LED[0]) digitalWrite(5, 0); //turn led off

              delay(400);
              digitalWrite(4, LOW);
          }
        }
        else
        {
          Serial.println("recv failed");//
          rf95.send(sendBuf, strlen((char*)sendBuf)); //resend if no reply
        }
    }
    else
    {
        Serial.println("No reply, is LoRa gateway running?");//No signal reply
        rf95.send(sendBuf, strlen((char*)sendBuf)); //resend data
    }
    Serial.println("");
}

void loop()
{
   if (TAKANO == 1)
   {
    refersgLP01();
    count=0;
    TAKANO=0; //now you can enter the if condition again in the ISR(TIMER1_OVF_vect)
function..
   }
}
```

The second Arduino end node is shown in Fig 3.

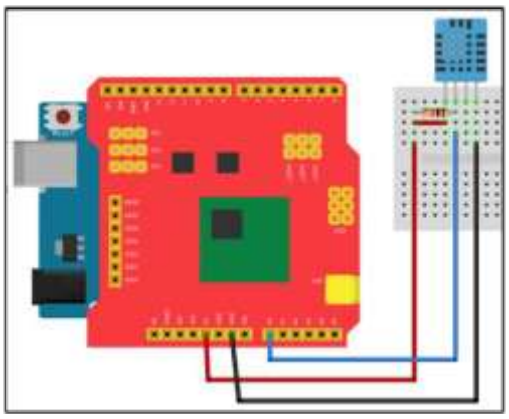


Fig 3.

It will send both the temperature and humidity from the 4-wire DHT11 sensor to the LG01P gateway every 15 seconds. The LG01P will in turn send these analogue values to the Takano platform. A user with a mobile app can read these statuses. Start by connecting it to a USB port on the laptop. If you are running short of USB ports on your laptop, you can remove the previously programmed Arduino Uno end node 1 and power it directly from an adaptor power supply, leaving the USB port for programming this end node 2. Then make sure you select "Arduino/Genuino Uno" board and the right USB port from the Tools menu in the Arduino IDE before uploading the below sketch. On the Takano platform, you can set limit values to both the temperature and humidity, that once breached, the platform can send either an email or push notification or even both to the registered user(s). You can read the instructions on how to set those limits in the Takano platform pdf (http://iotsprint.com/takano_setup/takano_app.pdf).

```
/*
  Require Library:
  https://github.com/dragino/RadioHead

  Upload Data to IoT Server TAKANO (https://iotsprint.com):
  Support Devices: Dragino Lora shield v1.4 + Arduino Uno

  Example sketch showing how to read Temperature and Humidity from DHT11 sensor,
  Then send both analogue values to LoRa Gateway every 15seconds
  which in turn will send them to the Takano server
*/

#include <SPI.h>
#include <RH_RF95.h>

RH_RF95 rf95;

#define dht_dpin A0 // Use A0 pin as Data pin for DHT11.
```

```
byte bGlobalErr;
char dht_dat[5]; // Store Sensor Data
char node_id[3] = {0,1,1}; //LoRa End Node ID
float frequency = 868.0;
unsigned int count = 1;

void setup()
{
   InitDHT();
   Serial.begin(9600);
   if (!rf95.init())
      Serial.println("init failed");
   // Setup ISM frequency
   rf95.setFrequency(frequency);
   // Setup Power,dBm
   rf95.setTxPower(13);
   rf95.setSyncWord(0x34);

   Serial.println("LoRa End Node2 Example --");
   Serial.println("DHT11 Temperature and Humidity Sensor");
   Serial.print("LoRa End Node ID: ");

   for(int i = 0;i < 3; i++)
   {
      Serial.print(node_id[i],HEX);
   }
   Serial.println("\n");
}

void InitDHT()
{
   pinMode(dht_dpin,OUTPUT);//Set A0 to output
   digitalWrite(dht_dpin,HIGH);//Pull high A0
}

//Get Sensor Data
void ReadDHT()
{
   bGlobalErr=0;
   byte dht_in;
   byte i;

   //pinMode(dht_dpin,OUTPUT);
   digitalWrite(dht_dpin,LOW);//Pull Low A0 and send signal
   delay(30);//Delay > 18ms so DHT11 can get the start signal

   digitalWrite(dht_dpin,HIGH);
   delayMicroseconds(40);//Check the high level time to see if the data is 0 or 1
   pinMode(dht_dpin,INPUT);
   // delayMicroseconds(40);
   dht_in=digitalRead(dht_dpin);//Get A0 Status
   //  Serial.println(dht_in,DEC);
   if(dht_in){
      bGlobalErr=1;
```

```
        return;
    }
    delayMicroseconds(80);//DHT11 send response, pull low A0 80us
    dht_in=digitalRead(dht_dpin);

    if(!dht_in){
        bGlobalErr=2;
        return;
    }
    delayMicroseconds(80);//DHT11 send response, pull low A0 80us
    for (i=0; i<5; i++)//Get sensor data
    dht_dat[i] = read_dht_dat();
    pinMode(dht_dpin,OUTPUT);
    digitalWrite(dht_dpin,HIGH);//release signal and wait for next signal
};

byte read_dht_dat(){
    byte i = 0;
    byte result=0;
    for(i=0; i< 8; i++)
    {
        while(digitalRead(dht_dpin)==LOW);//wait 50us
        delayMicroseconds(30);//Check the high level time to see if the data is 0 or 1
        if (digitalRead(dht_dpin)==HIGH)
        result |=(1<<(7-i));//
        while (digitalRead(dht_dpin)==HIGH);//Get High, Wait for next data sampleing.
    }
    return result;
}

void loop()
{
    Serial.print("############     ");
    Serial.print("COUNT=");
    Serial.print(count);
    Serial.println("     ############");
    count++;
    ReadDHT();
    char data[50] = {0} ;
    int dataLength = 7; // Payload Length
    // Use data[0], data[1],data[2] as Node ID
    data[0] = node_id[0] ;
    data[1] = node_id[1] ;
    data[2] = node_id[2] ;
    data[3] = dht_dat[0]; //Get Humidity Integer Part
    data[4] = dht_dat[1]; //Get Humidity Decimal Part
    data[5] = dht_dat[2]; //Get Temperature Integer Part
    data[6] = dht_dat[3]; //Get Temperature Decimal Part
    switch (bGlobalErr)
    {
      case 0:
        Serial.print("Current humidity = ");
        Serial.print(data[3], DEC);//Show humidity
        Serial.print(".");
```

```
        Serial.print(data[4], DEC);//Show humidity
        Serial.print("%  ");
        Serial.print("temperature = ");
        Serial.print(data[5], DEC);//Show temperature
        Serial.print(".");
        Serial.print(data[6], DEC);//Show temperature
        Serial.println("C  ");
        break;
      case 1:
        Serial.println("Error 1: DHT start condition 1 not met.");
        break;
      case 2:
        Serial.println("Error 2: DHT start condition 2 not met.");
        break;
      case 3:
        Serial.println("Error 3: DHT checksum error.");
        break;
      default:
        Serial.println("Error: Unrecognized code encountered.");
        break;
    }

    int i;
    for(i = 0;i < dataLength; i++)
    {
       Serial.print(data[i],HEX);
       Serial.print(" ");
    }
    Serial.println();

    unsigned char sendBuf[50]={0};

    for(i = 0;i < dataLength;i++)
    {
       sendBuf[i] = data[i] ;
    }

    rf95.send(sendBuf, dataLength+2);//Send LoRa Data

    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];//Reply data array
    uint8_t len = sizeof(buf);//reply data length

    if (rf95.waitAvailableTimeout(3000))// Check If there is reply in 3 seconds.
    {
       // Should be a reply message for us now
       if (rf95.recv(buf, &len))//check if reply message is correct
      {
          if(buf[0] == node_id[0] && buf[1] == node_id[2] && buf[2] == node_id[2] ) // Check if reply
message has the our node ID
         {
            pinMode(4, OUTPUT);
            digitalWrite(4, HIGH);
            Serial.print("Got Reply from Gateway: ");//print reply
            Serial.println((char*)buf);
```

```
            delay(400);
            digitalWrite(4, LOW);
          }
        }
        else
        {
          Serial.println("recv failed");//
          rf95.send(sendBuf, strlen((char*)sendBuf));//resend if no reply
        }
    }
    else
    {
        Serial.println("No reply, is LoRa gateway running?");//No signal reply
        rf95.send(sendBuf, strlen((char*)sendBuf));//resend data
    }
    delay(15000); // Send sensor data every 15 seconds
    Serial.println("");
}
```

4. Mobile App Setup

You first need to register with the Takano platform as specified in the following pdf (http://iotsprint.com/takano_setup/takano_platform.pdf), and download the Takano app from Google Play or Apple Store and familiarize yourself with it (http://iotsprint.com/takano_setup/takano_app.pdf). Once done, launch the mobile app, enter your email, the hardware registration key (that you got from the Takano Platform) and a name for the hardware as in Fig 4 for example. Next enter led (checkbox), button (ON/OFF Label), temperature (Gauge 180°) and humidity (Gauge 90°) as in Fig 5, 6, 7, and 8 respectively. And after you press on "SAVE ALL & GO TO RUN MODE" button you are presented with Fig 9.

Fig 4.

Fig 5.

Fig 6.

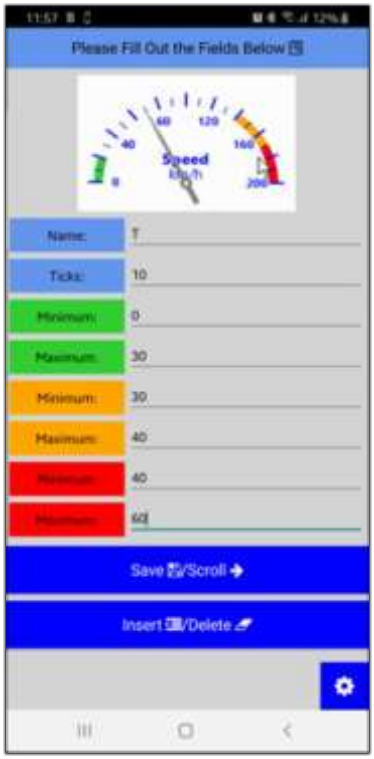


Fig 7.

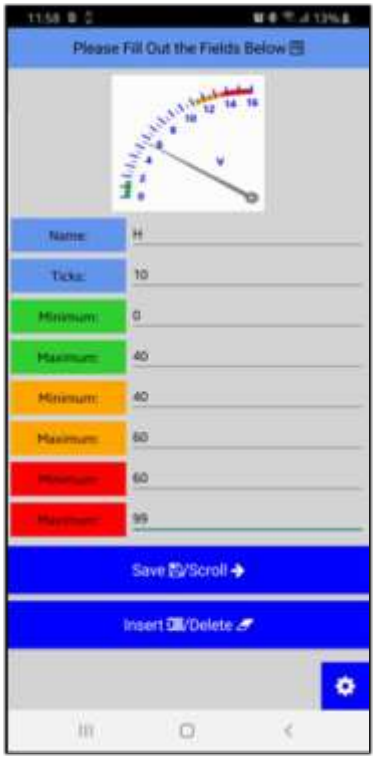


Fig 8.

Fig 9.

If the LP01 gateway is properly connected to the internet router, and the Lora end nodes are powered, you should read the analog temperature “T”, humidity “H”, button “BUTTON” and led “LED” values directly on the mobile app. You can toggle the led by tapping on the checkbox and receive an email when the button is pressed (switches from 0 to 1) on the Lora end node#2. Just remember that the LP01 normally sends the data to Takano every 20 seconds.

5. Contact information

For more information or comments, please do not hesitate to contact the Takano technical team by email at: info@iotsprint.com
We are available from 8:00AM (GMT+3) to 6:00PM (GMT+3).