IoT Lab - Bridges

IoT bridges

In this lab we are going to develop several IoT (UART) bridges to be used in different kinds of gateways. These bridges allow us to combine different communication technologies into complete IoT architecture.
We start by the presentation of the UART bridge already to send the data to ThingSpeak via a WiFi connection. There are two boards in our kit one (ESP32S1) for receiving different data flows from long range radio links (LoRa, ESP-NOW, Long Range WiFi, ..) and one (ESP32S2) for the transmission of the data to IoT servers (MQTT, ThingSpeak, ..)

Table of Contents

1.1 UART bridges for ESP32-S1 (Lolin D32) and ESP32-S2 (Lolin-S2 Mini)

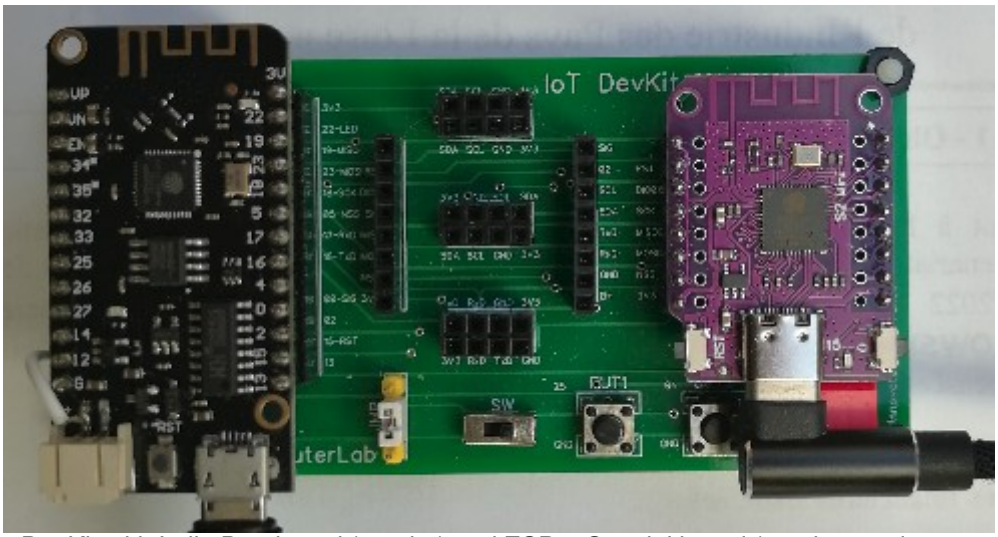


Fig 1 Simple DevKit with Lolin D32 board (sender) and ESP32S2 mini board (receiver and gateway) with **UART bridge**

1.1 UART senders

1.1.1 Sending into UART in main `loop()` function

The code for ESP32-S1 (Lolin D32)

```
#include <SoftwareSerial.h>
SoftwareSerial uart(17, 16);
```

```
void setup()
{
    Serial.begin(115200);
    delay(10);
    pinMode(17,INPUT);pinMode(16,OUTPUT);
    uart.begin(9600);
    Serial.println();
}

union
{
  uint8_t frame[16];
  float sensor[4];
  char  mess[16];
} sdp; // send data packet

void loop()
{
  sdp.sensor[0]=(float)random(0,1000)/13.0;
  sdp.sensor[1]=(float)random(0,1000)/17.0;
  sdp.sensor[2]=(float)random(0,1000)/19.0;
  sdp.sensor[3]=(float)random(0,1000)/23.0;
  for(int i=0;i<16;i++) uart.write(sdp.frame[i]);
  Serial.println("UART packet sent");
  Serial.println(sdp.sensor[0]);Serial.println(sdp.sensor[1]);
  delay(3000);
}
```

1.1.1 Sending into UART with `task` function

```
#include <SoftwareSerial.h>
SoftwareSerial uart(17,16);

QueueHandle_t queue;

void uartTask( void * parameter )
{
  int i=0;char recv;
  Serial.println("uartTask created");
  union
  {
    uint8_t frame[16];
    float sensor[4];
    char  mess[16];
  } sdp; // receive data packet

  while(true)
  {
    xQueueReceive(queue,sdp.frame,portMAX_DELAY); // portMAX_DELAY);
    Serial.printf("%2.2f,%2.2f\n",sdp.sensor[0],sdp.sensor[1]);
    for(int i=0;i<16;i++) uart.write(sdp.frame[i]);
  }
  vTaskDelete( NULL );
}

void setup()
{
    Serial.begin(115200);
    delay(10);
    pinMode(17,INPUT);pinMode(16,OUTPUT);
    uart.begin(9600);
    Serial.println();
    queue = xQueueCreate(10,16);  // sizeof rdp union
    xTaskCreate(
                    uartTask,     /* Task function. */
                    "uartTask",       /* String with name of task. */
                    10000,            /* Stack size in words. */
                    NULL,           /* Parameter passed as input of the task */
                    1,                /* Priority of the task. */
                    NULL);            /* Task handle. */
}
```

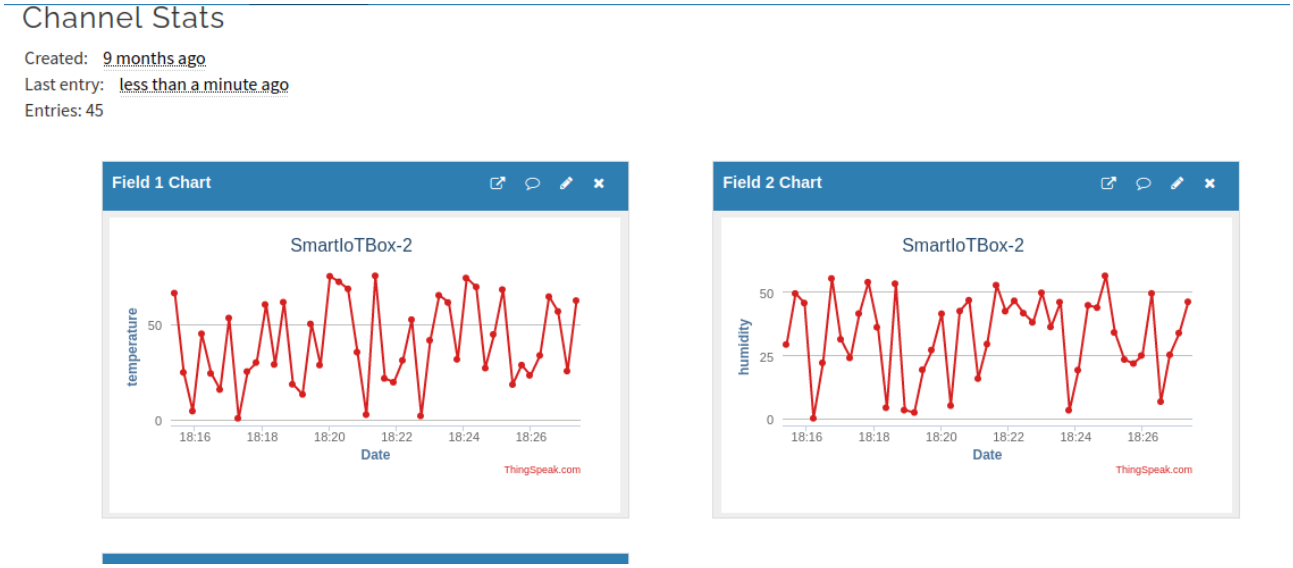
```
void loop()
{
  union
  {
    uint8_t frame[16];
    float sensor[4];
    char  mess[16];
  } sdp; // send data packet

  sdp.sensor[0]=(float)random(0,1000)/13.0;
  sdp.sensor[1]=(float)random(0,1000)/17.0;
  sdp.sensor[2]=(float)random(0,1000)/19.0;
  sdp.sensor[3]=(float)random(0,1000)/23.0;
  Serial.println(sdp.sensor[0]);Serial.println(sdp.sensor[1]);
  xQueueReset(queue);
  xQueueSend(queue,sdp.frame,0);
  Serial.println("Packet put in queue");
  delay(3000);

}
```

1.2 UART receiver and WiFi – TS sender

1.2.1 Receiving the UART data and sending tit to ThingSpeak in main `loop()` function

```
#include <WiFi.h>
#include "ThingSpeak.h"
#include <SoftwareSerial.h>
const char* ssid     = "Livebox-08B0";
const char* password = "G79ji6dtEptVTPWmZP";

SoftwareSerial uart(18, 16);
WiFiClient  client;

unsigned long myChannelNumber = 1697980;
const char * myWriteAPIKey = "4K897XNNHTW7I4NO";

// Initialize our values
int number1 = 0;
int number2 = random(0,100);
int number3 = random(0,100);
int number4 = random(0,100);

void setup()
{
    Serial.begin(115200);
    delay(10);
    pinMode(18,INPUT);pinMode(16,OUTPUT);
    uart.begin(9600);
    Serial.println();
    Serial.print("[WiFi] Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while(WiFi.status() != WL_CONNECTED)
    {   Serial.print(".");  delay(500); }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    delay(500);
    ThingSpeak.begin(client);  // Initialize ThingSpeak
    delay(500);
}

float s1=0.1,s2=0.2,s3=0.3,s4=0.4;
int i=0;
union
{
  uint8_t frame[16];
  float sensor[4];
  char  mess[16];
} rdp; // receive data packet

char recv;
```

```
long lasttime=0;

void loop()
{
  if (uart.available())
  {
    recv=uart.read(); rdp.frame[i]=(uint8_t) recv; i++;
    if(i>15)
    {
      ThingSpeak.setField(1, rdp.sensor[0]);
      ThingSpeak.setField(2, rdp.sensor[1]);
      // write to the ThingSpeak channel
      int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
      if(x == 200) Serial.println("Channel update successful.");
      else Serial.println("Problem updating channel. HTTP error code " + String(x));
      i=0;
      delay(16000);
    }
  }
}
```



1.2.2 Receiving the UART data and sending it to ThingSpeak in `task` function

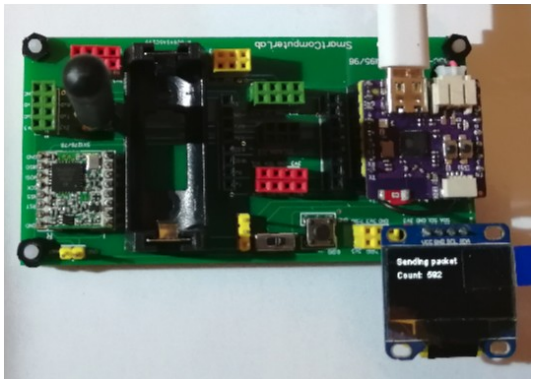
```
/* Wi-Fi STA Connect and Disconnect Example

   This example code is in the Public Domain (or CC0 licensed, at your option.)

   Unless required by applicable law or agreed to in writing, this
   software is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
   CONDITIONS OF ANY KIND, either express or implied.

*/
#include <WiFi.h>
#include "ThingSpeak.h"
#include <SoftwareSerial.h>

const char* ssid     = "Livebox-08B0";
const char* password = "G79ji6dtEptVTPWmZP";

SoftwareSerial uart(18, 16);

WiFiClient  client;

unsigned long myChannelNumber = 1697980;
const char * myWriteAPIKey = "4K897XNNHTW7I4NO";

// Initialize our values
```

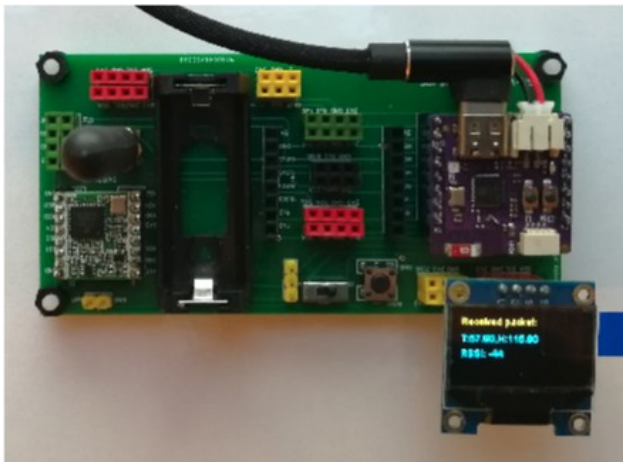
```
int number1 = 0;
int number2 = random(0,100);
int number3 = random(0,100);
int number4 = random(0,100);

QueueHandle_t queue;

void uartTask( void * parameter )
{
  int i=0;char recv;
  Serial.println("uartTask created");
  union
  {
    uint8_t frame[24];
    float sensor[4];
    char  mess[16];
  } rdp; // receive data packet
  while(true)
  {
    if (uart.available())
    {
      recv=uart.read(); rdp.frame[i]=(uint8_t) recv; i++;
      if(i==16)
        {
        delay(16000);
        Serial.println(rdp.sensor[0]); i=0;
        xQueueSend(queue, rdp.frame, 100);
        }
    }
  }
  vTaskDelete( NULL );
}

void setup()
{
    Serial.begin(115200);
    while(!Serial);
    pinMode(18,INPUT);pinMode(16,OUTPUT);
    uart.begin(19200);// SERIAL_8E2 to modify in SoftwareSerial
    Serial.println();
    Serial.print("[WiFi] Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while(WiFi.status() != WL_CONNECTED)
    {
      Serial.print(".");
      delay(500);
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    delay(500);
    ThingSpeak.begin(client);  // Initialize ThingSpeak
    delay(500);
    queue = xQueueCreate(100,24);  // sizeof rdp union
    xTaskCreate(
                    uartTask,     /* Task function. */
                    "uartTask",       /* String with name of task. */
                    10000,            /* Stack size in words. */
                    NULL,           /* Parameter passed as input of the task */
                    1,                /* Priority of the task. */
                    NULL);            /* Task handle. */
}

  union
  {
    uint8_t frame[24];
    float sensor[4];
    char  mess[16];
  } rdp; // receive data packet

void loop()
{
  xQueueReceive(queue,rdp.frame,20000); // portMAX_DELAY);
  delay(200);
```

```
  xQueueReset(queue);
  Serial.println("Queue received");
  Serial.printf("%2.2f,%2.2f\n",rdp.sensor[0],rdp.sensor[1]);
  if(100.0>rdp.sensor[0] && rdp.sensor[0]>0.0)
    {
    ThingSpeak.setField(1, rdp.sensor[0]);
    ThingSpeak.setField(2, rdp.sensor[1]);
    // write to the ThingSpeak channel
    int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
    if(x == 200) Serial.println("Channel update successful.");
    else Serial.println("Problem updating channel. HTTP error code " + String(x));
    }
}
```

1.3 Sending and receiving LoRa packets to ThingSpeak

The following example is based on 2 IoT DevKits with Lolin ESP32C3 (RISC-V) main board and integrated LoRa modem (RFM 95/96 module – SX1278/76) .





The IoT architecture consists of two boards , one the sender board that generates some data and sens them in LoRa packets.
The receiver board relays the received packets to ThingSpeak.com server via a WiFi connection.
The WiFi connection is created with **`SmartConfig`** that creates a simple WEB server with **`SoftAP`** and IP address **`192.168.4.1`**. The server is used to transfer the SSID and the corresponding password to the board in order to start WiFi connection.

The DevKit with Lolin ESP32C3 nano board exploits mini D1 interface for the connection of the board. This interface , similar to MicroBus is very popular and we have many MCU boards that can be used at the same place(ESP32-Wemos, ESP32S2 – Wemos-mini, ..)
Note that the SPI bus is connected to the LoRa module with the following lines/signals:

```
#define SS      5      // D0 – to NSS
#define RST     3      // D4  – RST
#define DI0     2      // D8 – INTR

#define SCK     1      // D5 – CLK
#define MISO    0      // D6 – MISO
#define MOSI    4      // D7 – MOSI
```

Attention:
The pins of the same signals my be different for different MCU boards.

1.3.1 Sender code

```
#include <LoRa.h>
#include <Wire.h>               // Only needed for Arduino 1.6.5 and earlier
#include "SSD1306Wire.h"        // legacy: #include "SSD1306.h"

// Initialize the OLED display using Arduino Wire:
SSD1306Wire display(0x3c, 8, 10);   // ADDRESS, SDA, SCL

String receivedText;
String receivedRssi;
// with LoRa modem RFM95 and green RFM board - int and RST to solder

#define SS      5 // 26     // D0 - to NSS
#define RST     3  //16     // D4  - RST
#define DI0     2      // D8 - INTR

#define SCK     1      // D5 - CLK
#define MISO    0      // D6 - MISO
#define MOSI    4      // D7 - MOSI
#define BAND    868E6

int sf=7;
long sbw=125E3;

char dbuff[24];

union
{
  uint8_t frame[64];
  char mess[64];
  float sensor[8];
} sdp;

void disp(char *t1, char *t2, char *t3)
{
  display.clear();
  display.setFont(ArialMT_Plain_10);
  display.setTextAlignment(TEXT_ALIGN_LEFT);
  display.drawString(0, 0, t1);
  display.drawString(0, 16, t2);
  display.drawString(0, 32, t3);
  display.display();
}

void setup() {
  Serial.begin(9600);
  SPI.begin(SCK, MISO, MOSI, SS);  // SCK, MISO, MOSI, SS
  LoRa.setPins(SS, RST, DI0);

  Serial.begin(9600);
  delay(1000);
  Serial.println();Serial.println();
  Serial.println("Starting LoRa Sender");
  // Initialising the UI will init the display too.
  display.init();
  display.flipScreenVertically();
  if (!LoRa.begin(BAND)) {
    disp("start","LoRa failed","end");
    Serial.println("Starting LoRa failed!");
    while (1);
  }
  else
   {
   Serial.println("Starting LoRa ok!");
   disp("start","LoRa OK","end");
   LoRa.setSpreadingFactor(sf);
   LoRa.setSignalBandwidth(sbw);
   }
   sdp.sensor[0]=0.0;
   sdp.sensor[1]=0.0;
   sdp.sensor[2]=0.0;
   sdp.sensor[3]=0.0;
}
```

```
int counter=0;

void loop() {
char buff[32];
  Serial.print("Sending packet: ");
  Serial.println(counter);
  sprintf(buff,"Count:  %d",counter);
  disp("Sending packet",buff,"   ");
  // send packet
  LoRa.beginPacket();
  LoRa.write(sdp.frame,64);
//  LoRa.print("hello ");
//  LoRa.print(counter);
  LoRa.endPacket();
  sdp.sensor[0]+=0.1;
  sdp.sensor[1]+=0.2;
  sdp.sensor[2]+=0.3;
  sdp.sensor[3]+=0.4;
  counter++;
  delay(5000);
}
```

1.3.2 Receiver – gateway code

```
#include <LoRa.h>
#include <WiFiManager.h>
#include "ThingSpeak.h"
unsigned long myChannelNumber = 1697981;
const char *myWriteAPIKey="A1G48I8FNSKGLRUB" ;  //SmartIoTBox-3
WiFiClient  client;
#include <Wire.h>               // Only needed for Arduino 1.6.5 and earlier
#include "SSD1306Wire.h"        // legacy: #include "SSD1306.h"

// Initialize the OLED display using Arduino Wire:
SSD1306Wire display(0x3c, 8, 10);   // ADDRESS, SDA, SCL

String receivedText;
String receivedRssi;
// with LoRa modem RFM95 and green RFM board - int and RST to solder

#define SS      5 // 26     // D0 - to NSS
#define RST     3  //16     // D4  - RST
#define DI0     2      // D8 - INTR

#define SCK     1      // D5 - CLK
#define MISO    0      // D6 - MISO
#define MOSI    4      // D7 - MOSI
#define BAND    868E6
int sf=7;
long sbw=125E3;

char dbuff[24];

void disp(char *t1, char *t2, char *t3)
{
  display.clear();
  display.setFont(ArialMT_Plain_10);
  display.setTextAlignment(TEXT_ALIGN_LEFT);
  display.drawString(0, 0, t1);
  display.drawString(0, 16, t2);
  display.drawString(0, 32, t3);
  display.display();
}

void setup()
{
  Serial.begin(9600);
  delay(1000);
  display.init();
```

```
  display.flipScreenVertically();
  display.setFont(ArialMT_Plain_10);
  display.setTextAlignment(TEXT_ALIGN_LEFT);
  Serial.println();Serial.println();
  WiFi.mode(WIFI_STA);
  WiFiManager wm;
    // wm.resetSettings();
    disp("Starting","WEB portal at","192.168.4.1");
    bool res;
    res = wm.autoConnect("ESP32AP",NULL); // password protected ap
    if(!res) {
        Serial.println("Failed to connect");
        // ESP.restart();
    }
    else {
        //if you get here you have connected to the WiFi
        Serial.println("connected...yeey :)");
    }
  ThingSpeak.begin(client); // connexion (TCP) du client au serveur
  delay(1000);
  Serial.println("ThingSpeak begin");

  SPI.begin(SCK, MISO, MOSI, SS);  // SCK, MISO, MOSI, SS
  LoRa.setPins(SS, RST, DI0);

  Serial.println("Starting LoRa Receiver");
  // Initialising the UI will init the display too.

  if (!LoRa.begin(BAND)) {
    Serial.println("Starting LoRa failed!");
    disp("Connect","LoRa modem","failed");
    while (1);
  }
  else
   {
   Serial.println("Starting LoRa ok!");

   disp("Connect","LoRa modem","OK");
   LoRa.setSpreadingFactor(sf);
   LoRa.setSignalBandwidth(sbw);
   }
}

union
{
  uint8_t frame[64];
  char mess[64];
  float sensor[8];
} rdp;



float temperature=0.0,humidity=0.0;

void loop() {
char buff[32],brssi[32];
int i=0,rssi=0;
  // try to parse packet
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // received a packet
    Serial.print("Received packet '");
    // read packet
    while (LoRa.available()) {
      rdp.frame[i] = LoRa.read();i++;
    }
  if(packetSize==64)
     {
```

```
        sprintf(buff,"T:%2.2f,H:%2.2f",rdp.sensor[0],rdp.sensor[1]);
        // print RSSI of packet
        Serial.print("' with RSSI ");
        Serial.println(LoRa.packetRssi());
        rssi = LoRa.packetRssi();
        sprintf(brssi,"RSSI: %d",rssi);
        disp("Received packet:",buff,brssi);
        Serial.println("Fields update");
        ThingSpeak.setField(1, rdp.sensor[0]);
        ThingSpeak.setField(2, rdp.sensor[1]);
        int x=ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
        if(x == 200) Serial.println("Channel update successful.");
        else Serial.println("Problem updating channel. HTTP error code " + String(x));
        delay(16000);
        }
  }
}
```

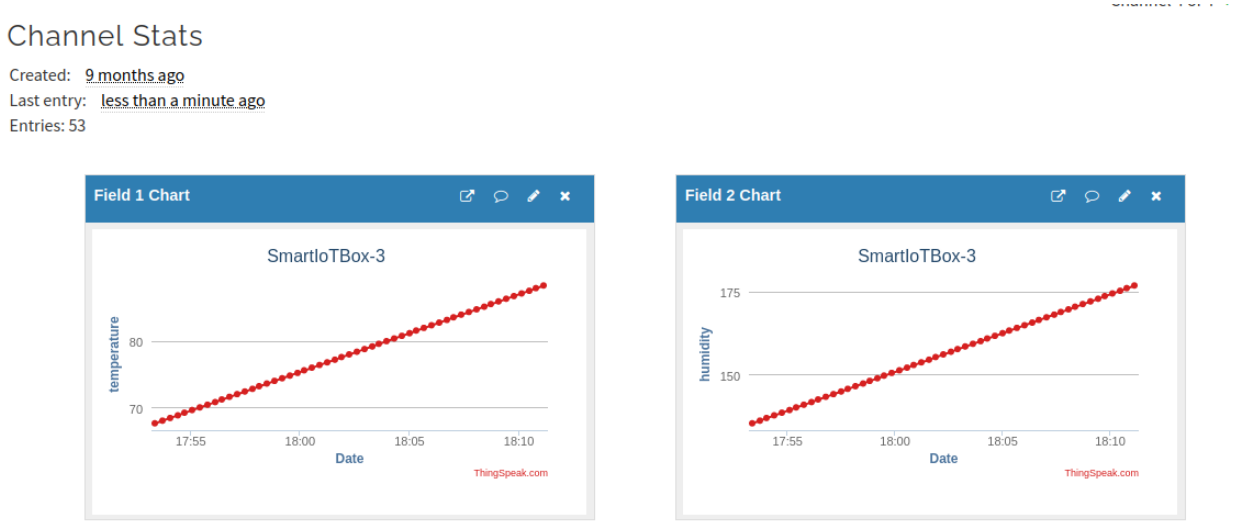


Table of Contents