

Project #5: Mbed Emulator, Installation, & IoT Service Set-up to the Mobile Device via TTN Server

Instructor: Bongho Kim

Support: Try the installation steps and troubleshoot yourself, but **before you start frustrating, contact the instructor** via email or Piazza message. The instructor will work with you via Zoom screen sharing as much as time permits. Depending on your PC environment, some steps may not work as expected, but this is also expected. If the installation still doesn't work as intended, then there are pre-built VM images with all installations for you to download. The link will be shared with you after we try enough because it is also important to have hands-on experience, setting up the working environment from scratch.
Don't wait for the office hour. You should contact the instructor anytime when you need support.

The goal of this project is for you to implement and set up a simple end-to-end IoT monitoring service with LoRa client and LoRaWAN gateway devices using simulation in a software emulator. The temperature and humidity measurements generated by the emulator (not real) will be sent to a server and forward a real-time notification to the subscribed user devices using the MQTT protocol.

You will need to download the "**Project5_package.zip**" file from Canvas. This zip file includes the LoRa client and gateway emulator source code (**mbed-simulator-master.tar.gz**) and the installation script file (**prj5_bash.sh**).

NOTE: For this project, you must use the cloned VM (**cis549-VM-clone**) you had built in week1. Do not use the VM that NS3 is installed and being used so far.

What to Submit

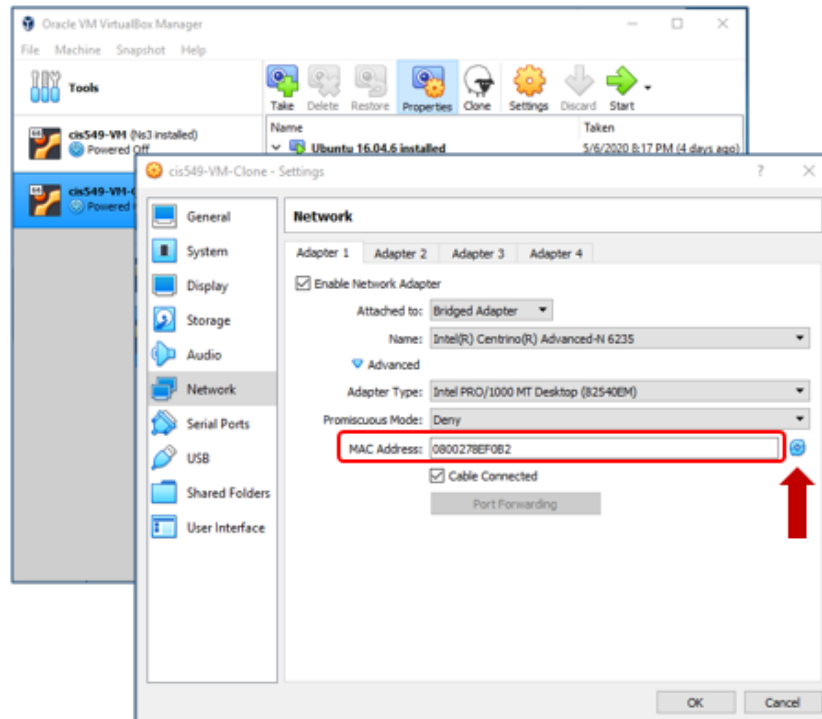
Create a folder named as "Prj5" Copy the following materials below into this folder, and submit the one zipped file: "Prj5.zip." Problems 1-3 are at the end of this document.

- A report in PDF format containing all other responses including the diagram (for problem 1)
- Screenshot and decoder code (for problem 2)
- MQTT client screenshot and the MQTT subscription information (for problem 3)

If you have not done this when you have cloned the VM, then before running this cloned VM, change the Network Interface MAC address first to ensure that this cloned VM and the original VM are using different MAC addresses.

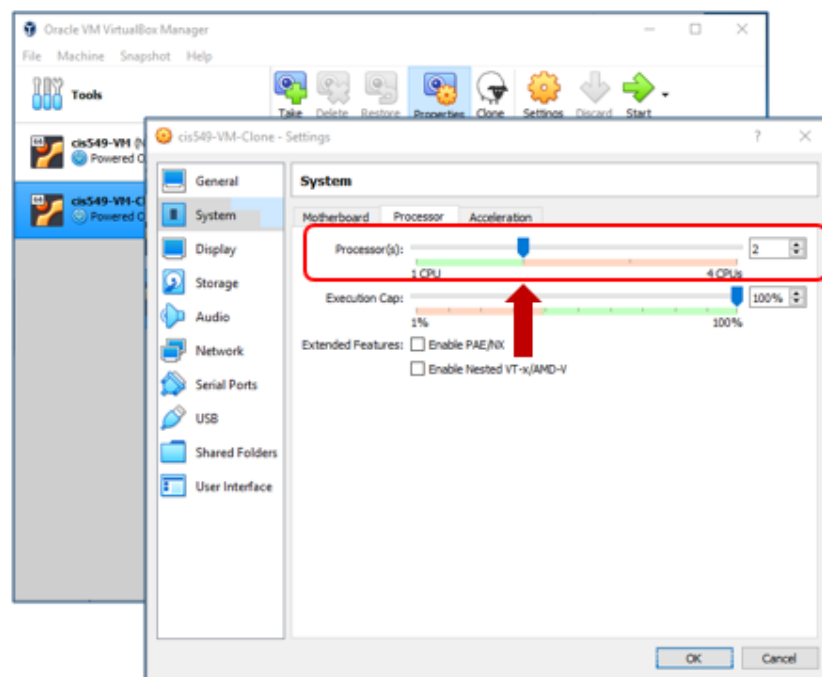
As shown in Figure 1, highlight the cloned VM, which we are referring to as "**cis549-VM-clone**", and select **Setting** → **Network** → **Adapter 1** → **Advanced**, and then **click the icon** that the arrow in Figure 1 is pointing to generate a random MAC address. You need to do this only once.

Figure 1:



The next step before starting the project is to allocate more processing power to this VM for running fast. As shown in Figure 2, highlight the cloned VM, which we are referring as **"cis549-VM-clone"**, and select **Setting → Systems → Processor**, and then **increase the number of the processor** by moving the bar as indicated with the red arrow. The number of the processor will be shared between the host machine and the VMs. You may allocate more processors to the VM until project 5 installation is completed. Once the installation is done, the processor can be redistributed following the same steps described here.

Figure 2:



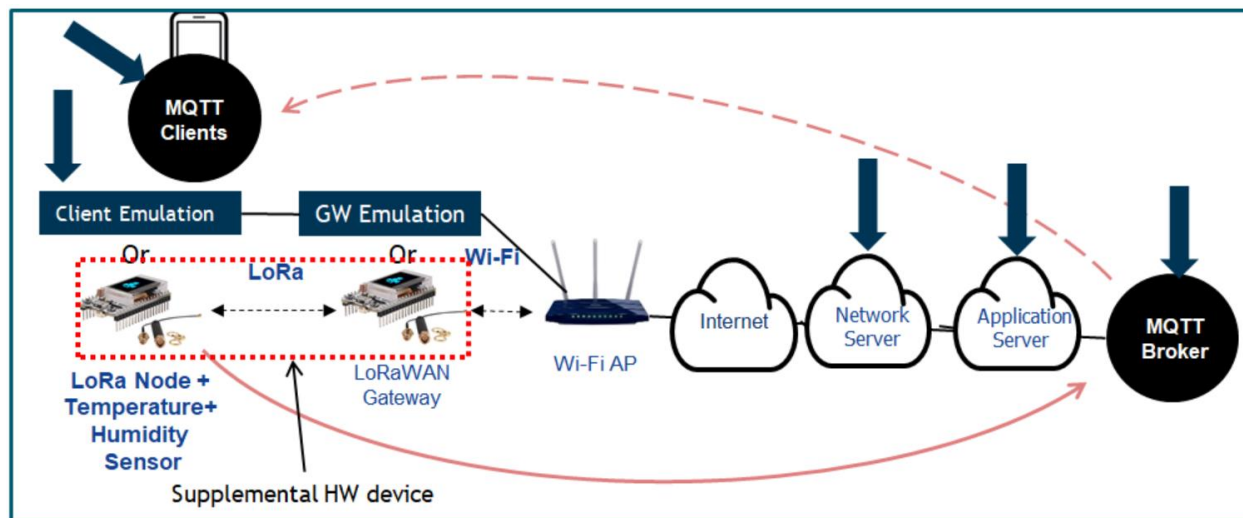
The list below shows the components that you need for building Project 5. The installation steps will guide you through:

- Mbed emulator for simulating the LoRa client, and Gateway hardware. The emulator source code is originally from [this link](#) and has been modified for this course. If you want to try more examples, then you can find it from [here](#).
- GNU ARM Embedded Toolchain software. The source location is [here](#).
- Emscripten: an Open Source LLVM to JavaScript compiler. Emscripten makes native code available on the Web. We are simply using it to execute the LoRa client device emulator, but if you are interested in, then you can find more information starting from this [link](#).
- An account on the Things Network: <https://www.thethingsnetwork.org/> or <https://nam1.cloud.thethings.network/console/>
 - An account on ThingSpeak: <https://thingspeak.com/> to set up the notification system.
 - MQTT client application: The example in this assignment is using the "IoT MQTT Dashboard" application from Google App Store. This application may not be available, but you can use any MQTT client suitable for Apple devices, desktops, or any other devices.
 - **Project5-package.zip** file in Canvas includes the source codes for the LoRa client as well as the gateway and necessary libraries but will require modifications as instructed below.
- Demo video is available on Canvas to help you install the configuration

Figure 3 shows the overview of the end-to-end architecture from the LoRa client, which generates sensor data to the MQTT client, which monitors the sensor data. It shows two types of LoRa client device and the gateway: emulator and hardware-based devices.

For this project, you will use the emulator for the client device and the gateway. If you are interested in, however, you may obtain the hardware-based LoRa client and gateway listed in the supplemental information provided separately and connect to the same network side configuration that you build in this project. It merely replaces the emulator part, and they can be connected simultaneously.

Figure 3: Overall system architecture for Project 5



Installation

NOTE: Make sure you can access internet from the cloned VM. Try “ping 8.8.8.8” within the VM. If “ping” fails, then check the network interface configuration.

Step 1: Install and configure the mbed emulator through the following steps.

- Run the cloned VM (cis549-VM-cloned), and make sure that at least two processors are allocated to this VM. Otherwise, the installation will take much longer.
- NOTE: If you are using the provided VM-Clone, then SKIP this step b and c, and move to step d.** Download Project5_package.zip file from Canvas, unzip it and move **mbed-simulator.tar.gz** and **emsdk.tar.gz** file in the Project5_package folder **to the ~/Downloads folder**.
- NOTE: If you are using the provided VM-Clone, then SKIP this step c and move to step d.** Move **prj5_bash.sh** file in Project5_package folder **to the home directory**, and run following commands. This bash file contains a list of installation commands. If you want to see which tools and software are installed, then review the prj5_bash.sh file using any text editor. **mbed-simulator-master.tar.gz** and **emsdk.tar.gz** files must be in the **~/Downloads** directory before running **prj5_bash.sh** file, and **prj6_bash.sh** file must be in the home directory (~/)

```
cd
sudo chmod +x prj5_bash.sh
./prj5_bash.sh
```

Installation may take about 20 minutes.

Note: At the beginning of the installation, if you observe an error message similar to the following, then stop the installation (click Ctrl+C), reboot the VM, and start the installation again (“./prj5_bash.sh”).

```
E: Could not get lock /var/lib/dpkg/lock-frontent - open (11: Resource temporarily unavailable)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontent), is another process using it?
```

- Once compilation is complete, **reboot** the VM, and navigate to the mbed-simulator folder and run the server. As shown in the box below, the output text contains the 8 bytes “Gateway ID” information. Copy this 8 byte information and use it when you register gateway in your TTN account.

```
cd ~/mbed-simulator
node server.js

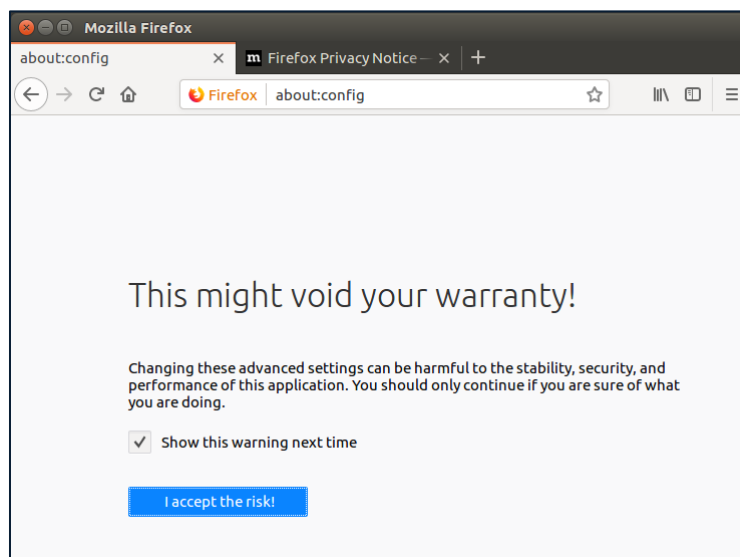
// Example output on the terminal

Mbed Simulator v1.9.7
Web server listening on port 7829!
LoRaWAN information:
  Gateway ID:          08:00:27:00:00:74:3f:17
  Packet forwarder host: router.eu.thethings.network
```

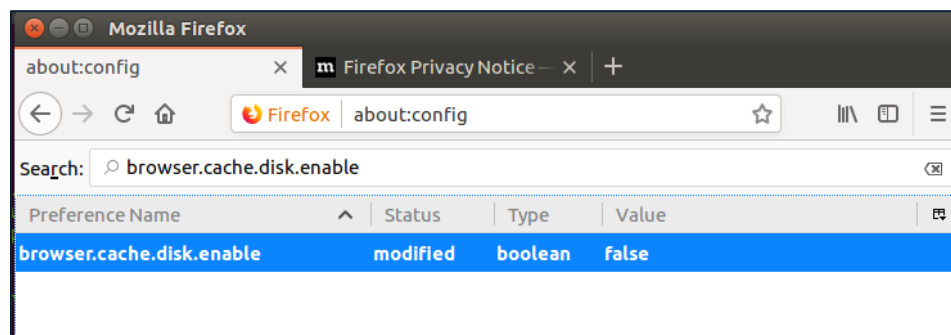
Packet forwarder port: 1700
Make sure the gateway registered in the network server running the *legacy packet forwarder*

Step 2: Open the Firefox Browser, and do the following steps to make the emulator load and display your source code correctly. The following steps disable Firefox caching (to not to display old information), and the step1-step4 need to be done only once.

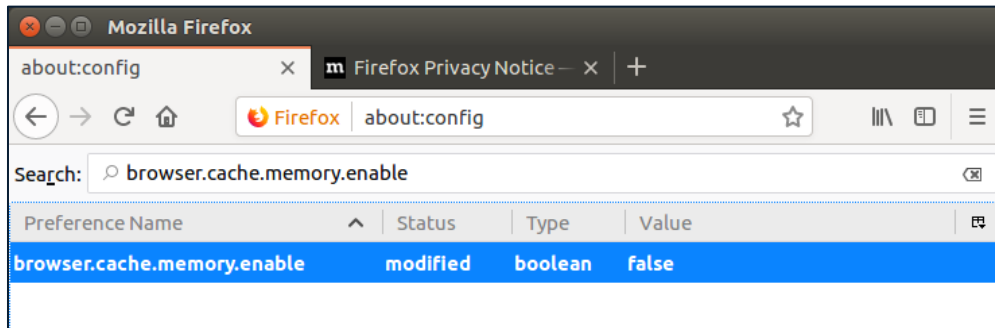
1. In your browser (Firefox) address bar type: **about:config**
2. Press on **"I accept the risk!"** button and proceed



3. In the search bar type **"browser.cache.disk.enable"** Double click on it, to make it false.

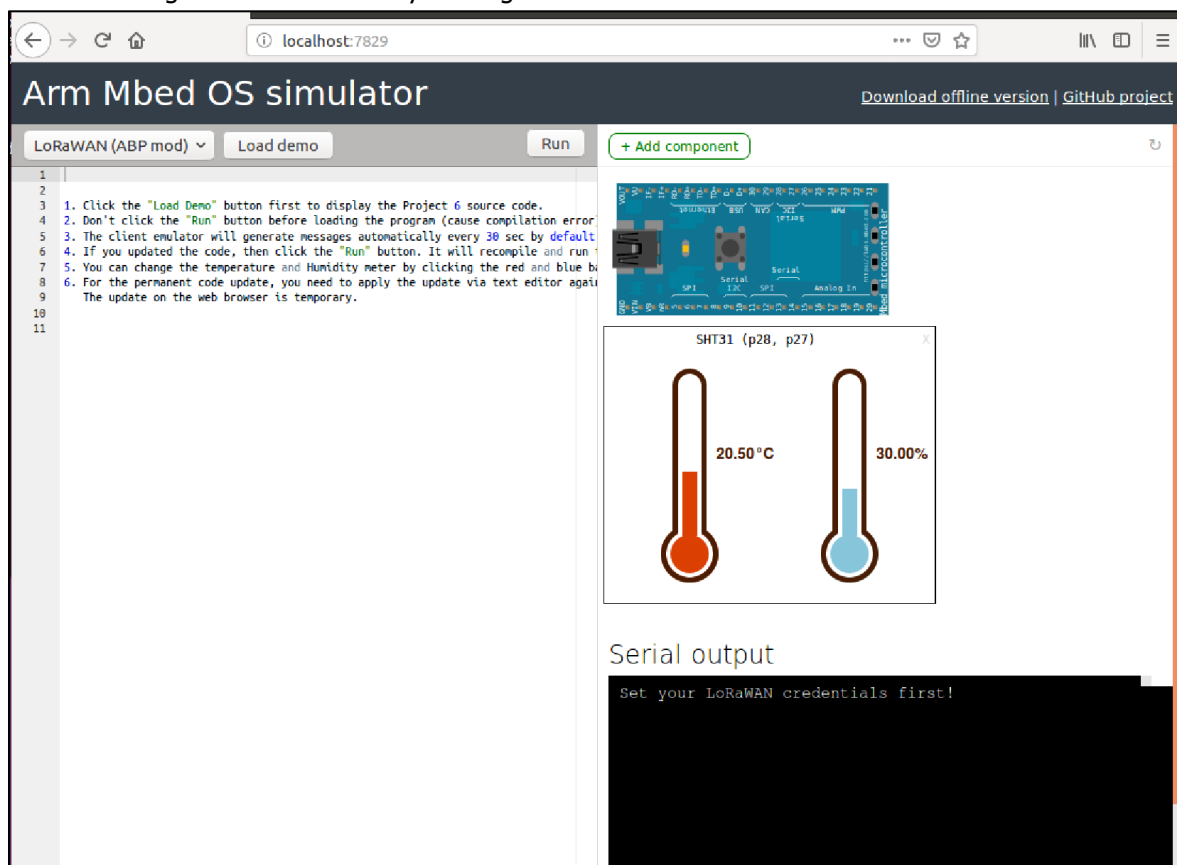


4. In the search bar type **"browser.cache.memory.enable"** Double click on it, to make it false.



- Exit Firefox and restart the browser.
- Put this as the URL: **http://localhost:7829**

Note: "node server.js" must be running in the background to open this page. You can change the sensor data by clicking the red and blue bars on the GUI.



- Read the information displayed on the left side of the browser for the next steps and more information.

Step 3: You need to update a few lines of source code to generate the LoRa message correctly and deliver to your application server. For the next step you need to first create your account on "The Things Network"

- Go to "The Things Network": <https://www.thethingsnetwork.org/> or <https://nam1.cloud.thethings.network/console/>. Create your account, login to the account, and open the "Console" page. (**Watch the demo video for the application registration**).
 - Add application.

- b. Add a new device under the application you have just created. Copy and paste NWKSKEY, APPSKEY, and DEVADDR in a text editor for later use. When you copy and paste the values, first convert the format as you need (see the format in the code snippet #1). Figure 4 screen can be displayed on the TTN console Device Overview page.
 - "eye" icon: show or hide the code
 - "arrow" icon: toggle between the byte order ("lsb" and "msb"). Must use "msb: mode"
 - "<>" icon converts the hex format
 - "document" icon at the end: copy the code

The screenshot shows a form with three rows of input fields. Each row has a label on the left, a set of icons in the middle, and a text input field on the right. The first row is 'Device Address' with icons for hex/decimal, byte order, and a document icon; the input field contains '26 01 18 BE'. The second row is 'Network Session Key' with icons for hex/decimal, byte order, and a document icon; the input field contains a series of dots. The third row is 'App Session Key' with icons for hex/decimal, byte order, a document icon, and a 'msb' label; the input field contains a list of hex values: { 0xB1, 0x45, 0x66, 0x44, 0x7A, 0xE7, 0xE0, 0x9D, 0xDC, 0xED, 0xCB, ... }.

Figure 4. Device Information Screenshot

2. Register gateway in the TTN server using the Gateway ID displayed on the terminal after running "**node server.js**" above in the terminal. You must have recorded it. Please Watch: **Watch the demo video for the gateway registration**). The gateway registration page may show you to enter "Gateway ID", but once you check the **'I'm using the legacy packet forwarder'** line below the Gateway ID field, this field will be converted to "Gateway EUI" field. Then, simply copy and paste the 8 byte Gateway ID that you have recorded. Once it is registered, the final Gateway ID becomes "eui-" plus the 8 byte information you provided. The gateway might not be connected instantaneously, but the application message may still be received by the application server.
3. Update the Client side application by opening main.cpp in the **~/mbed-simulator/demos/lorawan-abp-mod/** folder using any text editor:
 - a. Open up the file **main.cpp** and read the comments and description ins the lines between "EDIT START" and "EDIT END", and "MESSAGE CREATION START" and "MESSAGE CREATION END." You may search for these texts in main.cpp file.
 - b. Change the STUDENT ID macro to your ID. If your student ID is 8 to 10 digits.
#define STUDENT_ID 12345678; // example
 - c. Replace the following default values where Code Snippet #1 is located in **main.cpp** with the values you recorded during application registration: **nwk_s_key** (Network session key) , **app_s_key** (application session key), and **devaddr** (device address). You can display these values in hex in your TTN account. It will be easy for you to copy and paste. See the step 1-b above.

Code Snippet #1:

```
// EDIT: Replace below 3 lines with your information
static uint32_t devaddr = 0x00000000;
```

```
static uint8_t nwk_s_key[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00 };
static uint8_t app_s_key[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00 };
```

Load the **lorawan-abp-mod** demo in the browser tab with URL: <http://localhost:7829>

NOTE: You can make changes to code in the browser but they will not be saved in your main.cpp when you exit. If you want to permanently change the main.cpp, then you need to use text editor outside of the browser.

4. Login to the TTN account and check the incoming data under the application data page. Messages will be displayed (see the figure below. The GUI look might be different.).

time	counter	port	dev id	payload
01:26:44	24	1	dev id: dev1	01ED8E4E001908980960
01:26:27	23	1	dev id: dev1	01ED8E4E001808980960
01:26:10	22	1	dev id: dev1	01ED8E4E001708980960

Figure 5. Application Data before message decoding

You will not find any meaningful data that the client device is sending (see payload field 10 bytes data above). You need to decode this payload to display data (Student ID, message count, temperature and, humidity).

Click the **Payload Formats** tab on this page, and update the decoder field.

To decode the message, you need to know the message format. For this information, read the code in between "MESSAGE CREATION START" and "MESSAGE CREATION END" marks in this file, `~/mbed-simulator/demos/lorawan-abp-mod/main.cpp`.

You can also search for information online to learn how to write a decoder with less than 20 lines of basic C code (example link: <https://www.thethingsnetwork.org/forum/t/decrypting-messages-for-dummies/4894>). For the decoder, **use port number 1** since the application on the emulator is sending the message using port 1.

Check the file `~/mbed-simulator/demos/lorawan-abp-mod/main.cpp` for the following line that configures the port number. If this number changes, then the decoder should use the same port number.

```
// The port we're sending and receiving on
#define MBED_CONF_LORA_APP_PORT 1
```


Once the decoder is correctly implemented, you will start seeing those four parameters as shown below Figure 6. You will find the exact payload format (i.e. message format) in problem #2 you will implement the decoder.

The screenshot shows the 'Data' tab of the 'cisapp1' application for device 'dev1'. It displays a table of application data with columns for time, counter, port, and decoded fields: payload, count, humidity, studentid, and temperature. The data is filtered by 'uplink' and shows four entries with timestamps from 01:22:27 to 01:23:19.

time	counter	port	payload	count	humidity	studentid	temperature
01:23:19	12	1	01 ED 8E 4E 00 0D 08 98 09 C4	13	25	32345678	22
01:23:01	11	1	01 ED 8E 4E 00 0C 08 98 0A 28	12	26	32345678	22
01:22:44	10	1	01 ED 8E 4E 00 0B 08 98 0A 8C	11	27	32345678	22
01:22:27	9	1	01 ED 8E 4E 00 0A 08 98 0A 28	10	26	32345678	22

Figure 6. Application Data after message decoding

5. Now, it is time to set up the notification system:
 - a. Create an account on <https://thingspeak.com/>
 - b. Login to the account.
 - c. Create a channel with fields, and record the "Channel ID", "Write API Key" and "Read API Key" while setting it up to publish and subscribe
(**Watch the channel_creation.avi video clip to do this step**)
 - d. Next step is to configure integration to **forward the message from TTN to TS (ThingSpeak)**. You will use "Write API Key" from your ThingSpeak account when you created the channel. (**Watch the demo video for the "integration" part**)
6. Login to the ThingSpeak account and check if messages are being forwarded. The graphs for the four fields should be updated with new messages from the TTN.

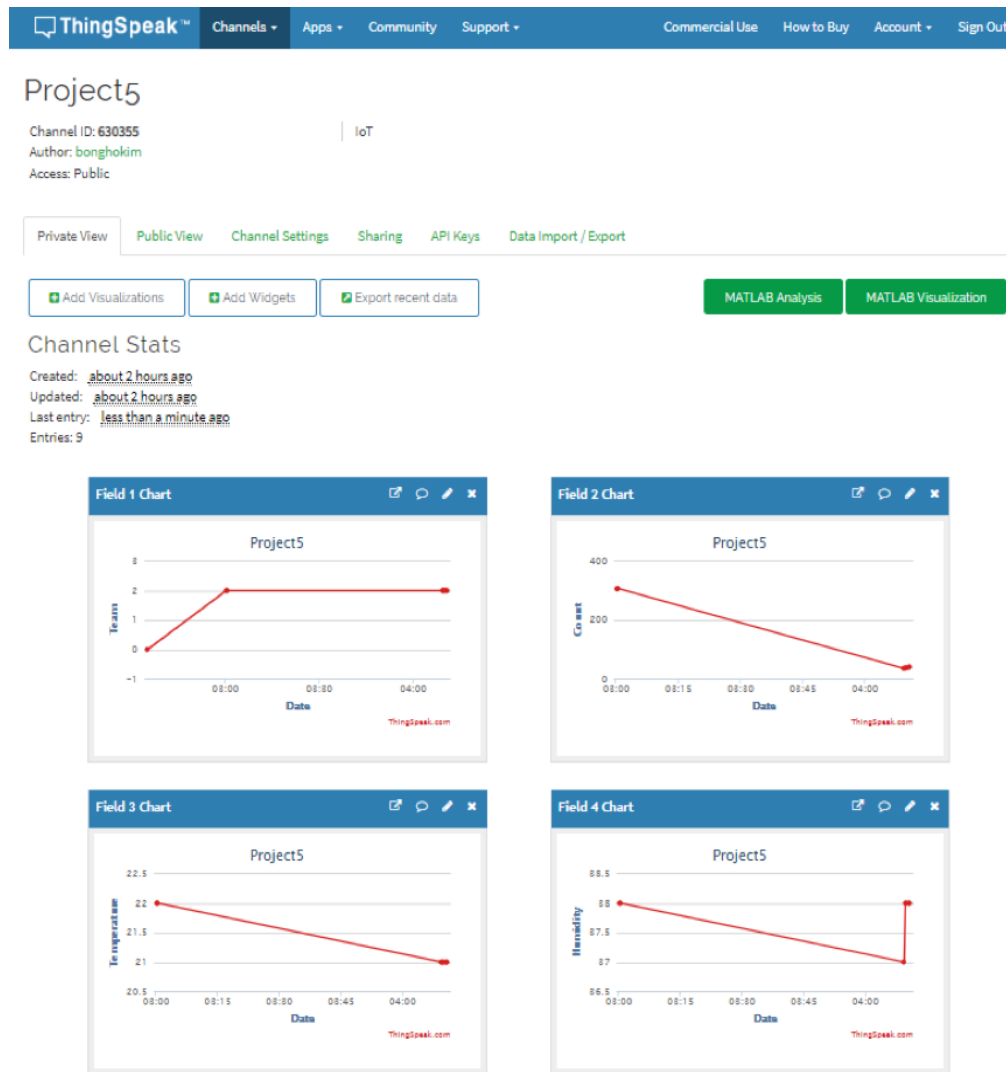
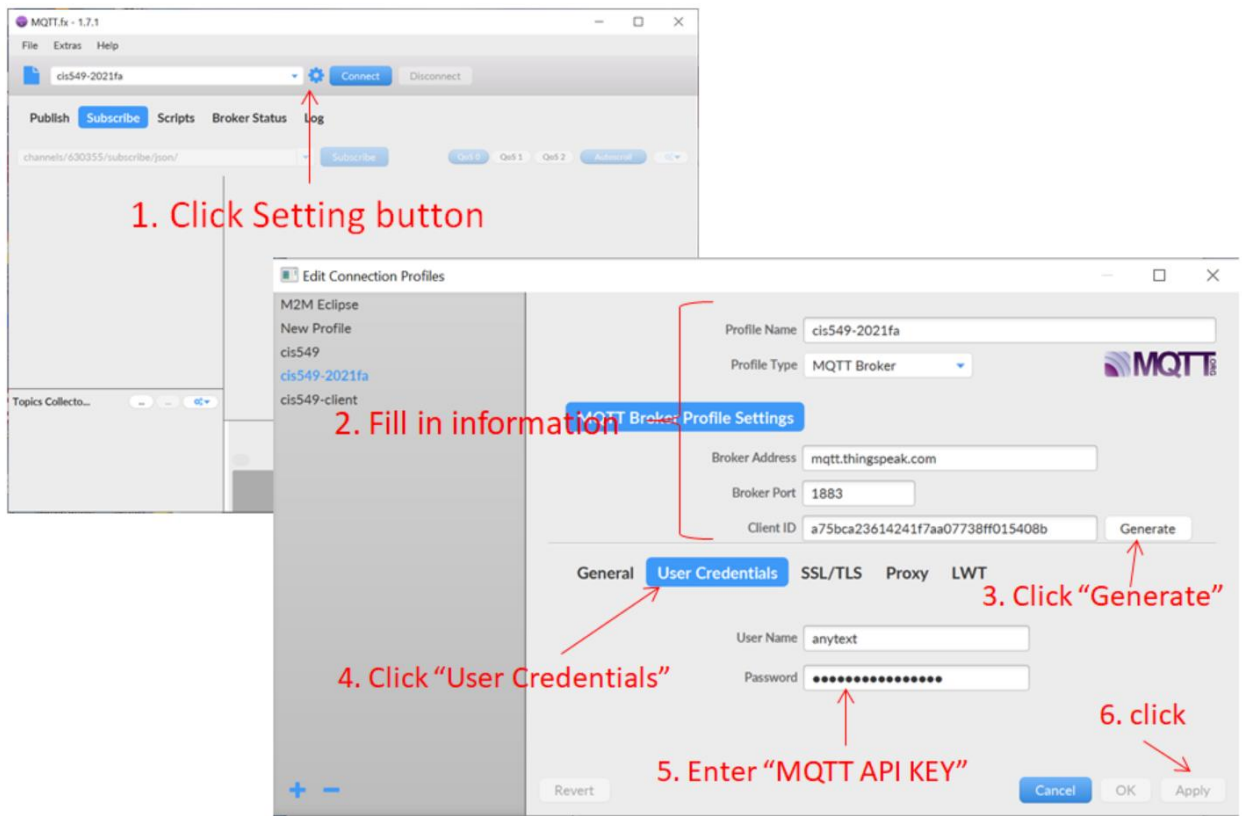
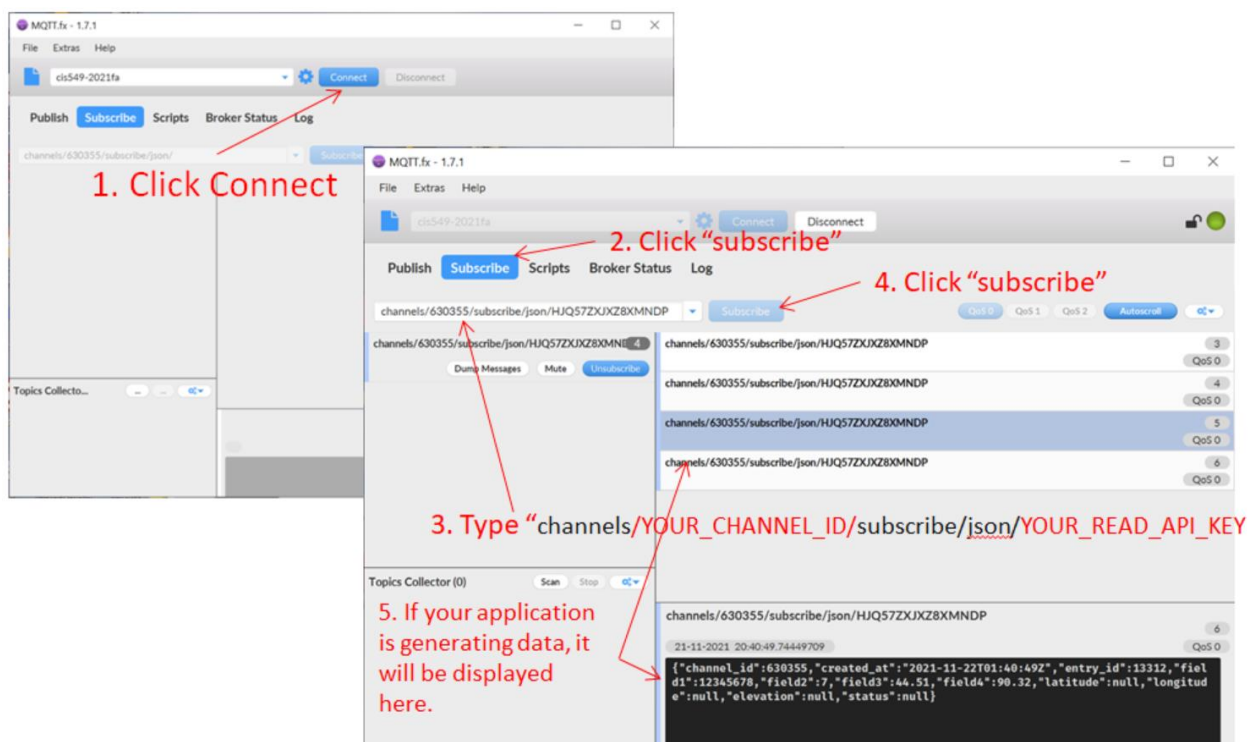



Figure 7. Graphs on ThingSpeak

7. The next step is to subscribe to the ThingSpeak MQTT broker from an MQTT client application. We will use MQTT.fx for both Windows and Mac PC.
 - a. Download MQTT.fx from internet and install it. MQTT.fx 1.7.1 is used for this instruction. Before the configuration, check the following information from your ThingSpeak account
 - i. "MQTT API KEY": Login to your ThingSpeak account. Click **Account > My Profile**. At the bottom of the page, you will find this information.
 - ii. If you don't see any key next to the "MQTT API KEY," then click the "Generate New MQTT API Key" button.
 - b. Channel ID: Login to your ThingSpeak account. Click **Channels > YOUR_CHANNEL**. It is on the top part of the page.
 - c. Read API Key: On the same page at the bottom.
 - d. Run "MQTT.fx" on your PC.
 - i. Follow the steps in the figure below and use the "**MQTT API KEY**" for the "password"



- ii. Now, you need to subscribe to a topic to receive the data. Follow the steps in the figure below. Topic is "channels/YOUR_CHANNEL_ID/subscribe/json/YOUR_READ_API_KEY." Simply replace "YOUR_CHANNEL_ID" and "YOUR_CHANNEL_ID" with your real information.





Now, the MQTT client will start receiving messages from ThingSpeak when receiving from the TTN (TheThingsNetwork).

Completing the Project

1. [Architecture design, 20%]

Write your configuration information for the devices indicated with the arrows in Figure 3 diagram in the report with the following elements:

- Key parameters configured per network element, including **Device address, App session key, Network session key, application message format and data fields, subscriber topic**, etc.

Submit the diagram with you information on it.

2. [Implement decoder, 40%]

Implement a decoder (see step 3-4 above) to separate the data with following four variables:

- "field1" for studentid
- "field2" for count
- "field3" for temperature
- "field4" for humidity

The variables above must be "field1, field2, field3 and field4"

Submit the decoder source code (copy and paste the codes in the report) and the **TTN console data page screenshot** of the received message with the parameters separated by the decoder similar to Figure 6 in the report document.

3. [Build system, 40%]

Implement the end-to-end IoT sensor monitoring system with real-time notification. The bullet items below are just a summary of the steps above.

- Build LoRaWAN gateway (emulator) and register it on the TTN account with the correct device ID
- Register application on the TTN account
- Build LoRa client (emulator) with the temperature and humidity sensor and register it on TTN account
- The client device should send a message with StudentID, message count, temperature data, and humidity measurement data every 30 sec to the TTN server
- Transfer all data in a message from TTN to the ThingSpeak server
- Install MQTT (mobile) client application on your device (smartphone, tablet, or PC) and subscribe to a topic that delivers all data in a message.



If you have followed all the steps above and the system is working then building part is completed.

After the completion, your MQTT client should be receiving a message soon after your device emulator generates an IoT message. **Make a screenshot** of the MQTT client that displays your message, and include it in the report for the submission.

Also, **submit the following information used to receive your sensor data**. This will be used in an MQTT client device to receive your message. Include the following information in the report.

- Channel ID
- MQTT API KEY
- READ API KEY
- Topic string (full string)