Magnetometer Software Documentation

Daniel Grech, Kantert Severens, Zoraiz Ali Syed & Dimitris Tapeinos

As part of the HTSM Program at the University of Groningen, in collaboration with ASTRON.
July 19, 2024

Contents

Overview

This documentation details the software architecture that was implemented for the Magnetometer built by students as part of the High Tech Systems and Materials (HTSM) honours program in collaboration with ASTRON. The code referenced in this documentation can be found in the files provided, or the git repository: https://github.com/grechman303/ASTRON_Magnetometer

The software architecture consists of the following components:

- **Micro-controller Software:** the software that is run on the *Arduino MKR WAN 1310* micro-controller which reads and transmits measurements from the sensors.
- **Network Server:** the configuration of the *Dragino LPS8v2* LoRaWAN gateway including the *ChirpStack* server which receives and decodes packets sent by the Arduino using the LoRaWAN protocol.
- **Application Server:** the server; implemented in *Node-RED*, which processes the payload from *ChirpStack* and forwards it to the database.
- **Database:** *InfluxDB* is used as the database to store the sensor data.

Technical Specifications

This documentation provides steps to set up a complete LoRaWAN system using a Dragino Gateway, ChirpStack, Node-RED, and InfluxDB on Windows (which is how it was set up as part of the original HTSM project). This setup will enable you to collect data from LoRaWAN devices, process it with Node-RED and store it in InfluxDB which also allows you to have some basic visualisations.

Prerequisites

Hardware

- Windows PC
- Dragino LoRaWAN Gateway (ie. LPS8v2)
- LoRaWAN device (ie. Arduino MKR WAN 1310)

Software

- Windows 10/11
- Docker and Docker Compose

Step 1: Setting Up the Dragino Gateway

Note: refer to the Dragino documentation.

Connect the Gateway: Power up the Dragino Gateway and connect it to your network via Ethernet or Wi-Fi. If you connect through Wi-Fi, the gateway shows up as *dragino-xxxxxx* and the password is *dragino+dragino*.

Access the Gateway Configuration Interface: Open a web browser and go to http://10.130.1.1 (10.130.1.1 is the default IP assigned to the LPS8-V2 gateway). The PC will get an IP address *10.130.1.xxx* which can be found by opening a command line interface (access "Command Prompt" in Windows by typing `cmd` in the windows search bar), type `ipconfig` and search for the *IPv4 Address* starting with *10.130.1.* Login with the following credentials:

- *User Name:* root
- *Password:* dragino

Configure the Gateway:

1. In the top menu, navigate to *Lora* > *Lora* and ensure that the frequency plan is set to EU868.

2. Navigate to *LoRaWAN > LoRaWAN – Semtech UDP* and select *Custom / Private LoRaWAN* as the Service Provider under "Primary LoRaWAN Server". Set the server address as the IP address of the PC within the network (ie. *10.130.1.xxx*). Ensure both uplink and downlink ports are set to *1700*.

3. Navigate to *LoRaWAN > LoRaWAN – Basic Station* and select *ChirpStack* as the "Service Provider". Set the "CUPS Server URI" as *http://10.130.1.xxx:8000*. 8000 is the default port configured for ChirpStack in with the dragino gateway.

4. Navigate to *Server > Network Server* and ensure that ChirpStack is running.

Step 2: Set Up ChirpStack

1. Open a new web browser tab and go to http://10.130.1.xxx:8000.

2. Login with default credentials:

   - *User Name:* admin
   - *Password:* admin

3. From the side menu, navigate to *Gateways* and click on the "Add Gateway" button. Give the gateway a suitable name (ex: dragino), description (optional) and enter the Gateway ID. The Gateway ID can be found in the dragino gateway portal by navigating to *LoRaWAN > LoRaWAN – Semtech UDP*; copy the text next to "Gateway EUI". If the gateway is set up correctly, it should show up as "Online" in ChirpStack.

4. From the side menu, navigate to *Device Profiles* and click the button to add one. Choose a suitable name and optional description, the rest of the fields can be left with their default values. Navigate to the "Codec" tab and replace the default code with the following:

```
function decodeUplink(input){
    return{
        data: Decoder(input.bytes, input.fPort)
    };
}

function Decoder(bytes, port) {
    var result = "";
    for (var i = 0; i < bytes.length; i++) {
        result += String.fromCharCode(parseInt(bytes[i]));
   }
   return { payload: result, };
}

```

   Note: This code can also be found in the Git repository or drive in the document titled *basic-decode.js*.

5. From the side menu, navigate to *Applications* and add one, typing a suitable name and optional description. Once this is created, click on the button to add a device. Give it a suitable name (ex: *arduino*), optional description, set the correct "Device EUI" and set a

16-digit join EUI (this can be any 16-digit number sequence such as **0000000000000000**). Finally, select the device profile you created from the "Device Profile" dropdown. Probably the quickest way to find the Device EUI of your Arduino MKR WAN 1310 is to use the *LoraSendAndReceive* example sketch provided in the MKRWAN library within the *Arduino IDE*. This sketch reads the Device EUI and prints it to the Serial Monitor.

6. Navigate to the "OTAA Keys" tab. The application key should correspond to the `SECRET_APP_KEY` in *arduino_secrets.h* (file provided with code). In our case, this number is **7b47ba91b1f8c2152d38bc1d41db1a22**. You may generate another one in ChirpStack, however make sure to change the value of `SECRET_APP_KEY` in *arduino_secrets.h* accordingly. Click "Submit" to save these details.

Step 3: Set Up Node-RED

Installation: In order to install Node-RED, you must have Node.js installed on your machine; the steps to do this can be found here. You can then install Node-RED by entering `npm install -g --unsafe-perm node-red` in a terminal such as PowerShell or Command Prompt. Once it is installed, you can start up Node-RED by typing the command `node-red`.

Configuration:

1. Navigate to *http://localhost:1880* to access Node-RED's interface which allows you to set up a pipeline for processing data using *Flows*.
2. Navigate to the hamburger menu on the top right and click *Manage Palette* > *Install*. Search for `node-red-contrib-influxdb` and install this package. This includes nodes to interface with the database InfluxDB which is required to persist the data obtained from the magnetometer.
3. Navigate to the hamburger menu on the top right, select *Import* and press the button "select a file to import". Select the file *magnetometer-data-processing-flow.json* which imports the pipeline for processing the data coming from the network server, ie. ChirpStack.

Step 4: Set Up InfluxDB

Install InfluxDB: Install InfluxDB on Windows by following the appropriate steps here. Once InfluxDB is installed, you can start it by typing `influxd` in a terminal such as PowerShell or Command Prompt and go to *http://localhost:8086* on your browser.

Configuration: Click on "Get Started" and follow the steps outlined here. Ensure you are looking at the steps for "Set up with the UI". Enter the following for these particular fields (these mirror the setup in Node-RED):

- *Organization Name:* ASTRON
- *Bucket Name:* magnetometer-data

Copy the provided operator API token and store it for safe keeping.
Go back to the flow in Node-RED, select the *influx* node, select the pencil next to *mag-test* ("Server" field) and replace the data in "Token" with your generated API token. Then click on the "Update" and "Done" buttons.

Step 5: Running the Application

Once the software has been set up according to the above steps, the entire application can be run to receive data transmitted from the Arduino, decode it in ChirpStack, process and format it in Node-Red and persist it to the database in InfluxDB.

1. Ensure that your computer is connected to the gateway and that ChirpStack, Node-RED and InfluxDB are running. In Node-RED, click on the "Deploy" button to deploy the flow.
2. Open the file: *sensor_read_and_send.ino* in the Arduino IDE which is provided in the code. Ensure that the values in *arduino_secrets.h* match up with those in ChirpStack.
3. Connect the Arduino MKR WAN 1310 to your computer and upload the sketch.
4. The Arduino should start sending data that it reads from the sensors. Ensure that the data is received in:
    - **ChirpStack** - go to *Applications*, selecting the one you created, selecting the appropriate device and navigating to the *Events* tab. Here you can see the messages relaying sensor data which are labelled as *up*.
    - **Node-RED** - you can view messages received from ChirpStack (and their processing) in the debug console; click the button on the right with a bug on it to view this.
    - **InfluxDB** - you can make various queries on the bucket where the data is being saved. You can even create dashboards to visualise the data in many ways.

Conclusion

Limitations

It is important to note that the setup outlined in this documentation is not optimal and has several limitations. Specifically, it is tedious to set up and is tailored to Windows. Additionally, the secret keys and tokens are not properly secured. This documentation details how the system was implemented as part of the HTSM summer school project and should serve only as a rough guide for future work. Significant improvements are necessary before seriously implementing and scaling up this system.

Future Work

Given the need for continuous online availability, it is advisable to install this software on a more suitable device than a Windows computer, such as a Raspberry Pi. This would necessitate a different installation procedure for ChirpStack, Node-RED, and InfluxDB that is tailored for Debian. Additionally, containerizing this setup using Docker would facilitate easier deployment across multiple devices.

While InfluxDB provides basic tools for data visualization, it is primarily a database and has limitations in terms of advanced data visualization and analysis. Integrating Grafana, which can seamlessly handle data from InfluxDB, offers a more powerful solution for visualizing and analyzing data. Furthermore, additional pipelines can be established to build fit models on the magnetometer data and conduct more complex analyses.

This documentation focuses on setting up the system on a local machine, but the gateway is also capable of forwarding data to the cloud. Utilizing services like The Things Network makes the data more accessible and allows for a more versatile and scalable system. This approach enables better data accessibility and facilitates the development of a more robust and scalable solution.