Quick answers to common problems

Intel Galileo Networking Cookbook

Over 45 recipes that will help you use the Intel Galileo board to build exciting network-connected projects

Marco Schwartz
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 7 'Cloud Data Monitoring'
- A synopsis of the book’s content
- More information on Intel Galileo Networking Cookbook
Marco Schwartz is an electrical engineer, entrepreneur, and blogger. He has a master's degree in electrical engineering and computer science from SUPELEC in France and a master's degree in micro engineering from the EPFL in Switzerland. He has more than 5 years of experience working in the domain of electrical engineering. His interests gravitate around electronics, home automation, the Arduino and the Raspberry Pi platforms, open source hardware projects, and 3D printing. He runs several websites related to Arduino, including the Open Home Automation website, which is dedicated to building home automation systems using open source hardware.

He has written another book on home automation and Arduino, named Arduino Home Automation Projects, Packt Publishing. He has also published a book on how to build Internet of Things projects with Arduino, which is called Internet of Things with the Arduino Yun, Packt Publishing.
Intel Galileo is a great development kit for all your Do-It-Yourself electronics projects. The board has an in-built powerful Intel processor, but is usable with the well-known Arduino software. It is also compatible with most Arduino shields.

This makes it the ideal board for your projects, especially in the fields where you need to use cloud-based services, for example, to store data online. It is therefore a great board for Internet of Things (IoT) applications.

In this book, we are going to start setting up your board and building simple applications with the board, such as running a web server on the board. After this, we'll dive into more complex topics, such as IoT applications. Finally, we'll sum up everything we learned in the book by building a simple home automation system based on the Galileo board.

What this book covers

Chapter 1, Installing and Configuring Linux, explains how to set up your Galileo board and the development environment, so you can start building projects. We will install the Galileo IoT image and install the required software on your computer.

Chapter 2, Connecting External Sensors, will discuss connecting sensors to the board, such as a temperature sensor. We will see how to read data from sensors and use it in our projects.

Chapter 3, Controlling Hardware Devices, covers how to control devices from the Galileo board, such as a relay that can be used to control electrical appliances.

Chapter 4, Creating a Web Server, is dedicated to running a simple web server on the Galileo board. We will see that a server can also be used to control the inputs and outputs of the board.

Chapter 5, Hosting Applications on the Galileo Board, covers how to host useful applications on the board, such as a simple file-sharing server.
Chapter 6, *Local Network Monitoring*, is dedicated to creating applications to monitor the activity of the Galileo board via the local network connection, for example, you will be able to monitor measurements done by the board in real time.

Chapter 7, *Cloud Data Monitoring*, is all about connecting your board to cloud services in order to build Internet of Things applications, such as remotely logging data on a cloud platform.

Chapter 8, *Building a Home Automation System*, sums up all that was done in the book with an application: building a home automation system based on Arduino, using the Galileo board as the "hub" of the system.
Cloud Data Monitoring

In this chapter, we will cover:

- Building and testing hardware for cloud monitoring
- Sending data to a cloud device
- Monitoring data remotely from a dashboard
- Logging data in an online datasheet
- Controlling your board from anywhere in the world
- Troubleshooting cloud monitoring issues

Introduction

In this chapter, we are going to continue to build Internet of Things (IoT) projects with the Galileo board. This time, we are going to see how to monitor data remotely, whether you are at home or on the other side of the world.

To do that, we are going to use different strategies. We are first going to see how to send data to the cloud to be stored online, and then use a dashboard to visualize this data. Then, in another recipe, we will see how to log that data in a spreadsheet. Finally, we will see how to use the MQTT protocol and a dashboard to control your board from anywhere in the world.

Building and testing hardware for cloud monitoring

In this first recipe, we are going to configure our hardware for cloud monitoring. We are also going to test the project so that we are sure the hardware is working correctly.
Getting ready

You will need a few components for this recipe. First, you will need a TMP36 temperature sensor to be used for temperature measurements. Then, you will need a photocell and a 10k ohm resistor for ambient light level measurements.

This is a list of all the components required for this recipe:

- TMP36 temperature sensor (https://www.sparkfun.com/products/10988)
- Photocell (https://www.sparkfun.com/products/9088)
- 10k ohm resistor (https://www.sparkfun.com/products/8374)
- Jumper wires (https://www.sparkfun.com/products/8431)
- Breadboard (https://www.sparkfun.com/products/12002)

How to do it...

We are now going to configure the hardware for this recipe. Follow these steps to complete the configuration process:

1. First, we connect the 5V and GND to the top or bottom horizontal rows of the breadboard.
2. Then, we can connect the TMP36 sensor’s left pin to the 5V horizontal slot, the middle pin to the analog pin A0, and the right pin to the GND horizontal slot on the breadboard. If you want to see the position of each pin, have a look at the following picture:
3. Then, position the photocell such that it is in series with the resistor. The node where the photocell and resistor meet should be connected to the board's analog pin A1.

4. Next, connect the free leg of the photocell to the 5V line. The free leg of the resistor should be pinned to the GND.

This is what the hardware configuration looks like when everything is connected:

Once this is done, we are ready to write some simple software to test the components.

**How it works...**

We will use two analog sensors in this recipe. Therefore, we are going to use analog `read()` functions to get data from these sensors and print the data in the Intel XDK software console, as a test.

This is the code:

```javascript
// Includes
var mraa = require("mraa");
var util = require('util');

// Sensor pins
var temp_sensor_pin = new mraa.Aio(0);
temp_sensor_pin.setBit(12);
var light_sensor_pin = new mraa.Aio(1);
```
light_sensor_pin.setBit(12);

function measure_data() {

    // Measure light
    var a = light_sensor_pin.read();
    var light_level = a/4096*100; // Analog input is on 4096 levels
    light_level = light_level.toPrecision(4);
    console.log("Light level: " + light_level + " %");

    // Measure temperature
    var b = temp_sensor_pin.read();
    var temperature = (b/4096*5000 - 500) / 10;
    temperature = temperature.toPrecision(4);
    console.log("Temperature: " + temperature + " C");

}

// Send data every 10 seconds
measure_data();
setInterval(measure_data, 10000);

This code reads data from both analog pins (A0 and A1), converts it into useful data, and then prints it in the console.

As usual, simply copy this code into the main.js file in a project in the Intel XDK software, build it, and then run it on the board. This is the result:

```
UPLOADING: Uploading project bundle to IoT device. 
[ Upload Complete ]
Intel XDK - Message Received: run
Light level: 77.64 %
Temperature: 25.00 C
```

See also

You can now move on to the following recipes to see how to send this measured data to the cloud so that it can be monitored remotely.

Sending data to a cloud device

We are now going to learn how to take measurements from the temperature sensor and the photocell, and send them to the cloud to be logged. They will then be accessible from anywhere, and you will also be able to use them in other applications.
Getting ready

There is nothing extra you need to do for this recipe. We are going to use dweet.io to store data in the cloud, and you can read more about this service at:

http://dweet.io/

Dweet.io is a very nice and easy-to-use cloud service for storing data. You make a request to the service with your data, and their servers store it.

How to do it...

In the project in this recipe, we will take measurements from the sensors, connect to dweet.io, and then send the data to the dweet.io server.

This is the complete code for this recipe:

```javascript
// Includes
var mraa = require("mraa");
var util = require('util');
var request = require('request');

// Sensor pins
var temp_sensor_pin = new mraa.Aio(0);
temp_sensor_pin.setBit(12);
var light_sensor_pin = new mraa.Aio(1);
light_sensor_pin.setBit(12);

function send_data() {
  // Measure light
  var a = light_sensor_pin.read();
  var light_level = a/4096*100;
  light_level = light_level.toPrecision(4);
  console.log("Light level: " + light_level + " %");

  // Measure temperature
  var b = temp_sensor_pin.read();
  var temperature = (b/4096*5000 - 500) / 10;
  temperature = temperature.toPrecision(4);
  console.log("Temperature: " + temperature + " C");

  // Send request
  var device_name = 'galileo_5etr6b'; // Change with your own name here

  var options = {
    method: 'post',
    url: 'https://dweet.io/dweet/for/galileo_5etr6b',
    headers: {'content-type': 'application/json'},
    body: JSON.stringify({
      light: light_level,
      temperature: temperature
    })
  };

  request(options, function (err, response, body) { });
}
```

// Call the function
send_data();
Cloud Data Monitoring

```javascript
var dweet_url = 'https://dweet.io/dweet/for/' + device_name + '?temperature=' + temperature + '&light=' + light_level;
console.log(dweet_url);

var options = {
  url: dweet_url,
  json: true
};

request(options, function (error, response, body) {
  if (error) {console.log(error);}  
  console.log(body);
});

// Send data every 10 seconds
send_data();
setInterval(send_data, 10000);
```

You need to modify something in the code. Dweet.io works on the concept of things that identify your device on the cloud. It is labeled as `device_name` in the code sketch, and you should change it to your own name, so that you can identify your own device on the cloud.

You can now just copy this code and paste it in a new Intel XDK project. Refer to the previous chapters, for example, Chapter 4, Creating a Web Server, if you need more details on this procedure.

Don't forget to modify the `package.json` file, to specify that we are using the `request` Node.js module.

Upload the code to the board, build it, and then execute it. You should soon see the first set of measurements that were sent to dweet.io:

```
https://dweet.io/dweet/for/galileo_Setr6b?temperature=25&light=84.96
{ this: 'succeeded',
  by: 'dweeting',
  the: 'dweet',
  with:
    { thing: 'galileo_Setr6b',
      created: '2015-04-21T02:39:54Z',
      content: { temperature: 25, light: 84.96 } }
}
```

Intel XDK - Message Received: stop
How it works...

The code in this recipe works in a similar way to the code in the previous recipe for measurements. Next, following each measurement, the Galileo board uses the request Node.js module to send a GET request to the dweet.io server, along with the measured data encoded in the URL.

Then, the code reads the answer from dweet.io and prints it in the console.

The whole process is repeated every 10 seconds.

See also

You should now really see the next recipe to learn how to display this data graphically in an online dashboard.

You can, of course, also experiment with other cloud services that are similar to dweet.io, for example, Xively (https://xively.com/), or Enable IoT (https://dashboard.us.enableiot.com/ui/auth).

Monitoring data remotely from a dashboard

In this recipe, we are going to build a project on top of what we just created. We are going to use the data that was logged on dweet.io, and display this data in an online dashboard.

Getting ready

We are going to use a free online dashboard to display the data graphically; Freeboard.io. You can find this at the Freeboard website:

https://www.freeboard.io/
You can now create a new account, and, once this is done, you can create a new dashboard:

<table>
<thead>
<tr>
<th>My Freeboards</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Home</td>
<td>Public</td>
</tr>
<tr>
<td>Test</td>
<td>Public</td>
</tr>
<tr>
<td>CC3200</td>
<td>Public</td>
</tr>
<tr>
<td>Raspberry Pi</td>
<td>Public</td>
</tr>
<tr>
<td>Galileo</td>
<td>Public</td>
</tr>
<tr>
<td>ESP8266</td>
<td>Public</td>
</tr>
</tbody>
</table>

Make sure that the previous recipe, which sends data to dweet.io continuously, is still running.

**How to do it...**

We are now going to configure our newly created dashboard so that it can display the measurements from the Galileo board.

The first step is to create a data source that will stream the data into our dashboard. You can do this by clicking on the **ADD** button:

This will take you to a panel where you can fill in the details about your datasource. Basically, you just need to select dweet.io and then enter the **NAME** of your **THING** on dweet.io:
Then, you need to link this datasource to a widget that will represent the data graphically. To do that, first add a new **Pane** to your project, and then add a new widget.

There are several widget types available but, for this project, I selected a **Gauge** widget. This is how I filled in the different sections of the **Temperature** widget:
I did the same thing for the light levels; I added another **Pane** and another **Gauge** widget. This is the final result in the dashboard:

![Dashboard Image](image)

Note that this dashboard can be accessed from anywhere in the world, so you can monitor your Galileo projects from any web browser!

**How it works...**

Freeboard.io works by interfacing different data sources (for example, dweet.io) to graphical widgets such as gauges or live graphs. Here, we simply linked the data on dweet.io, generated by the Galileo board, to two widgets on freeboard.io.

**See also**

You can now look at the following recipes to learn about other ways to log data online and to monitor the measurements of your Galileo board.

**Logging data in an online datasheet**

In this new recipe, we are going to use another way to log data on the cloud; we are going to use a Google Docs spreadsheet to store the data, and then use the built-in functionalities of Google Sheets to plot the data as it arrives in the sheet. We will also use Temboo to make the link between our project and Google Docs.

**Getting ready**

There are some things you will need to do before you can take on this recipe. First, create a Google account if you haven't got one. Once this is done, create a new Google Docs spreadsheet. Name it Galileo, for example. Also, give a name to the first three columns corresponding to the data that we are going to log—Date, Temperature, and Light Level.
Then, go over to Temboo to create an account there too. We used Temboo in the previous chapter; it allows us to use web APIs with no difficulties. You can create an account by going to:

https://temboo.com/

If you need information about the Temboo library, go to:

https://www.temboo.com/library/Library/Google/Spreadsheets/

**How to do it...**

We are now going to configure the different parts of the project so that we can log data in a Google Docs spreadsheet.

The first step is to create a new Google app, which we will use to log data in a spreadsheet. You can do this by going to:

https://console.developers.google.com

You can now create a new project:

![New Project Form](image)

New Project

Project name

Galileo App

Project ID

ecstatic-cosmos-92210

I agree that my use of any services and related APIs is subject to my compliance with the applicable Terms of Service.

Create Cancel
After that, you need to click on **APIs** and **Auth**, then on **APIs**, and activate the **Drive API**, as shown in this picture:

After that, click on **Credentials**. You will be taken to a page where you can find the **ClientID** that you will need later on Temboo.

Then, head over to Temboo at:

https://www.temboo.com/library/Library/Google/Spreadsheets/

You basically have to follow the instructions there to initialize the library. At some point, you will be asked to enter the **ClientID** you got before:

You will also be given a client secret key that you will also need in the rest of the recipe. Then, once the whole procedure is done, you will be given an **AccessToken**, which you need to enter here:
Note down this token; we will use it in a moment in our application.

We are now going to use Intel XDK again for the code for this recipe. This is the complete code:

```javascript
// Includes
var mraa = require("mraa");
var util = require('util');
var tsession = require("temboo/core/temboosession");
var session = new tsession.TembooSession("your_temboo_username", "your_app_name", "your_app_key");

// Sensor pins
var temp_sensor_pin = new mraa.Aio(0);
temp_sensor_pin.setBit(12);
var light_sensor_pin = new mraa.Aio(1);
light_sensor_pin.setBit(12);

var Google = require("temboo/Library/Google/Spreadsheets");
var appendRowChoreo = new Google.AppendRow(session);

// Instantiate and populate the input set for the choreo
var appendRowInputs = appendRowChoreo.newInputSet();

function send_data() {
  // Measure light
  var a = light_sensor_pin.read();
  var light_level = a/4096*100;
  light_level = light_level.toPrecision(4);
  console.log("Light level: "+ light_level + " %");

  // Measure temperature
  var b = temp_sensor_pin.read();
  var temperature = (b/4096*5000 - 500) / 10;
```

```
temperature = temperature.toPrecision(4);
console.log("Temperature: "+temperature+" C");

// Date
var d = new Date();
var date = d.toString();

// Set inputs (put your own Google app data here)
appendRowInputs.set_ClientSecret("google_client_secret");
appendRowInputs.set_RefreshToken("google_client_token");
appendRowInputs.set_RowData(date+","+temperature+","+light_level);
appendRowInputs.set_SpreadsheetTitle("spreadsheet_title");
appendRowInputs.set_ClientID("google_client_id");

// Run the choreo, specifying success and error callback handlers
appendRowChoreo.execute(
    appendRowInputs,
    function(results){console.log(results.get_Response());},
    function(error){console.log(error.type);
    console.log(error.message);}
);  

// Send data every 10 seconds
send_data();
setInterval(send_data, 10000);

There are several things you need to modify in this code. First, you need to enter your Temboo login, application name, and application key. This data can be found in your profile settings on Temboo.

Then, you need to enter data about your Google account—client ID, secret, and the access token you got before. Finally, you need to enter the name of the spreadsheet you created before.

You can now just copy this code and paste it into a new Intel XDK project. Refer to the previous chapter if you need more details on this procedure.

Don’t forget to modify the package.json file, to specify that we are using the Temboo Node.js module.

Upload the code to the board, build it, and then execute it. After that, go to the spreadsheet you created before. You should see that the data is being logged in the spreadsheet:
Chapter 7

Now, as you are in Google Sheets, you can also use the software's built-in functionalities to plot the data. To do that, simply select all the data (columns A to C), and click on Insert Chart. I chose a simple line plot for this project; here is the result:

![Line Plot]

Note that, as this is in a Google Docs document, it is accessible from any web browser, so you can monitor this data from anywhere in the world!

How it works...

This recipe uses Temboo to make the link between our Galileo board and the Google Drive API. After we got the AccessToken from Temboo, we were able to log live measured data from the Galileo board in the Google Docs spreadsheet. Then, we simply used the built-in plotting functionalities of Google Sheets to plot the incoming data in real time.

See also

I now recommend that you look at the next recipe to discover another way to monitor your Galileo board remotely, by using the MQTT protocol.
Controlling your board from anywhere in the world

In this last recipe, we are going to see another way to control your Galileo projects from anywhere. Here, I use the word control, because we are actually going to use this recipe not only to monitor data remotely, but also to send commands to your board from the cloud. Of course, for this application, your board has to be connected to the Internet, via the Ethernet, for example.

We are going to use the MQTT protocol and a service called Lelylan to control our projects from anywhere in the world.

Getting ready

The first step in this recipe is to create an account at Lelylan. This will allow us to control projects from anywhere in the world using a nice dashboard. To do this, go to:

http://lelylan.github.io/devices-dashboard-ng/#/

You will be redirected to your dashboard, where you can create a new device:

When asked about the device type, choose Basic Light. Choose MQTT for the protocol. Once the process is finished, you will be given a client or device ID and SECRET that you will need for the rest of this recipe:
It's also time to connect a test device to your Galileo board for control purposes. I used a simple LED, but you can also use a relay, for example. Simply connect the device you want to test to pin number 7 on the Galileo board. If you want more information on how to connect a device to your Galileo board, refer to Chapter 3, Controlling Hardware Devices.

How to do it...

We are now going to link the button in your dashboard to your Galileo board. This is the button in the dashboard:

We are going to use the Intel XDK software to write the code for this recipe. This is the complete code:

```javascript
// Device ID and secret
var device_id = "device_id";
var device_secret = "device_secret";
// Required modules
```
var m = require("mraa");
var util = require('util');
var mqtt = require('mqtt');

var client = mqtt.createClient(1883, "178.62.108.47", {
  username: device_id,
  password: device_secret,
  clientId: "7393956449"
});

// Set output on pin 7
var myDigitalPin = new m.Gpio(7);
myDigitalPin.dir(m.DIR_OUT);

// Topics
in_topic = 'devices/' + device_id + '/get';
out_topic = 'devices/' + device_id + '/set';

// Connect event
client.on('connect', function () {
  client.subscribe(in_topic);
});

// When message is received
client.on('message', function (topic, message) {
  // Message is Buffer
  console.log(message.toString());

  json_data = JSON.parse(message.toString());

  // Check the status property value
  var value = json_data['properties'][0]['value'];

  if (value == 'on') {
    myDigitalPin.write(1);
  }
  if (value == 'off') {
    myDigitalPin.write(0);
  }

  // Confirm to Lelylan
  client.publish(out_topic, message.toString());
});
Note that you need to change the device **SECRET** and the device **ID** in the code to the values you got from your dashboard.

You can now just copy this code and paste it into a new Intel XDK project. Refer to the previous chapter if you need more details on this procedure.

Don't forget to modify the package.json file, to specify that we are using the `mqtt` Node.js module.

Upload the code to the board, build it, and then execute it. After that, go to the dashboard again, and click on the button:

![Relay Control Settings](image)

If you have an LED connected to pin number 7 for example, you should see that it turns on immediately. You will also see the confirmation in the dashboard (**Status on**).

### How it works...

This whole project is based on the MQTT protocol. MQTT is a lightweight protocol that was made specifically for the Internet of Things. You can learn more about it here:

http://mqtt.org/

The Lelylan dashboard provides a nice way to control devices that run the MQTT protocol, as is the case with the Galileo board.

### See also

You now have several tools that you can use to monitor data and control your Galileo board from anywhere in the world!

Note that you can also use the Intel Enable IoT project for similar functionalities.
Troubleshooting cloud monitoring issues

The Internet of Things is a very interesting topic; however, there are always issues with the communication between your project and the cloud. This is why we are now going to look at the most common issues that can arise in the recipes found in this chapter.

Dweet.io is not responding

The most common cause of not being able to get an answer from dweet.io is that the Internet connection on your Galileo board is not functioning properly. This can happen when the board is inactive for a long time. To solve this, simply reboot the board by clicking on the REBOOT button on the board itself, and then try to execute the project again.

The dweet.io data does not appear in Freeboard.io

If no data appears on the dashboard, but you are sure that the data was logged correctly in dweet.io, there may be a problem with the datasource you defined. Make sure that you entered exactly the same Thing name in the Freeboard.io datasource. Then, make sure that you correctly linked this datasource to your dashboard widgets.

Data is not being logged in my Google spreadsheet

If the Temboo project is running correctly but you cannot see anything being logged in the Google Docs spreadsheet, there are several things you can check.

First, check that you followed the Temboo procedure correctly to get your Google access token. Without the correct access token, the project won't work.

Then, make sure you authorized your Google application to access the Google Drive API.

Finally, make sure that you entered the name of your Google spreadsheet in the code. Also make sure that you gave a name to your columns inside this spreadsheet.

You can also test the choreo directly on the Temboo website, to check whether the problem is with your Galileo board or with Temboo.
I cannot control my board from Lelylan

If you are running the code from the last recipe of this chapter, but nothing happens when you click the button, there are also several things you can check.

The first thing to check is that your Internet connection on the board is still active. Reboot the board if this is not the case.

Then, check if you get the confirmation message on the Lelylan dashboard. If so, you might simply have attached your test device (for example, the LED) to your board wrongly. If you think this is the case, check Chapter 3, Controlling Hardware Devices again.

Finally, also make sure that you correctly entered your Lelylan dashboard client ID and client SECRET in the JavaScript code running on your Galileo.
Where to buy this book

You can buy Intel Galileo Networking Cookbook from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.