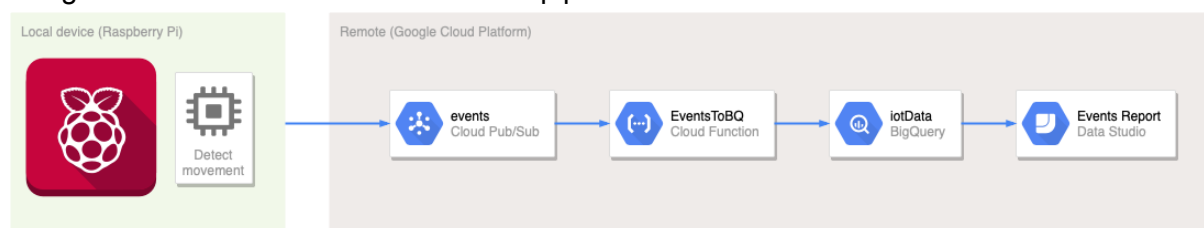


Lab4: Data Pipeline from Raspberry Pi to Cloud with local processing

This Lab is also based on Google Codelabs “Building a Serverless Data Pipeline: IoT to Analytics” <https://codelabs.developers.google.com/codelabs/iot-data-pipeline/index.html>

The basic setup is identical to [Lab 3](#), streaming data from accelerometer and magnetometer sensors to the cloud, but this time you should use the acceleration and magnetometer data to detect movement and send processed (event) data into the cloud. This way you will reduce the amount of data being transmitted between the IoT device and the cloud.

In this Lab, you are going to build a data pipeline that starts with an Internet of Things (IoT) device. Local data processing should detect movement of the device and generate and send the events to the cloud. At the cloud side, you will use a message queue to receive data, and a serverless function will move the data to a data warehouse. Analogous to Lab3, a dashboard that displays the information should be created using the Data Studio tool. A Raspberry Pi 3 Model B+ with a SenseHat sensor will be used for the IoT device and several components of the Google Cloud Platform will form the data pipeline.



Introduction

In this laboratory, you will stream data collected by the SenseHat sensors, installed in a Raspberry PI board running a Linux based system. You should develop a script to detect board movements, and when a change occurs your code should stream the event data to a Google Cloud Platform project.

Objectives

In this lab, you will practice how to perform the following tasks:

1. Setup your own data pipeline in Google Cloud Platform
2. Create Cloud Pub/Sub topics
3. Use IoT Core to create a registry
4. Authorize devices to publish data in your Pub/Sub topic
5. Create a cloud Function to export data from a Pub/Sub topic to a BigQuery table
6. Run the MQTT Application on Raspberry Pi
7. Stream changes regarding the position of the IoT Device (events)
8. Store the data using a BigQuery table from Google Cloud Platform
9. Visualize the collected data using the Data Studio Dashboard

Prerequisites

- A Google Cloud Platform account
- Raspberry Pi 3 Model B
- SenseHat for Raspberry Pi
- MicroSD card for Raspbian (8GB+ recommended)
- MicroSD card reader
- USB keyboard
- MicroUSB to USB-A cable
- Power adapter for Raspberry Pi 3
- Monitor with HDMI input
- HDMI cable

In addition, having access to a computer monitor or TV with HDMI input, HDMI cable, keyboard and a mouse is assumed. Also, it is assumed you have configured the Raspberry Pi to connect to the Internet and reach the Google Cloud Platform website.

Instructions

The steps to set up the data pipeline are akin to the steps detailed in Lab3. Therefore you can always check the instructions in the Lab3 document if you have some questions.

There are fewer steps compared to Lab3, and in this lab you will focus on the local data processing, as you should write a script to detect movements in the Raspberry Pi board using the accelerometer and magnetometer sensors.

Lab4: Data Pipeline from Raspberry Pi to Cloud with local processing	1
Introduction	1
Objectives	2
Prerequisites	2
Instructions	3
Steps already taken in Lab3	5
Create a new project at Google Cloud Platform	6
Enable APIs (Optional)	7
Create a Cloud Pub/Sub topic	8
Create a BigQuery table	8
Create a Cloud Function	9
Create an IoT Core registry	10
Create a device in the IoT Core Registry	11
Add the device key to the IoT Core Registry	11
Start sensing and streaming events from Raspberry Pi to the Cloud	13
Time to code!	14
Start the data pipeline	14
Verify that data is arriving into the cloud	15
Examine the stored data	15
Outcomes from this lab	16
Cleaning up	17
References	18

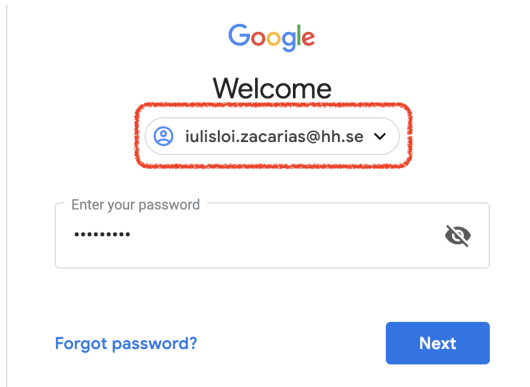
Steps already taken in Lab3

It is assumed that you have access to the google cloud and set up the Raspberry Pi already as part of [Lab 3](#). If not, please do that following the instructions for [Lab 3](#).

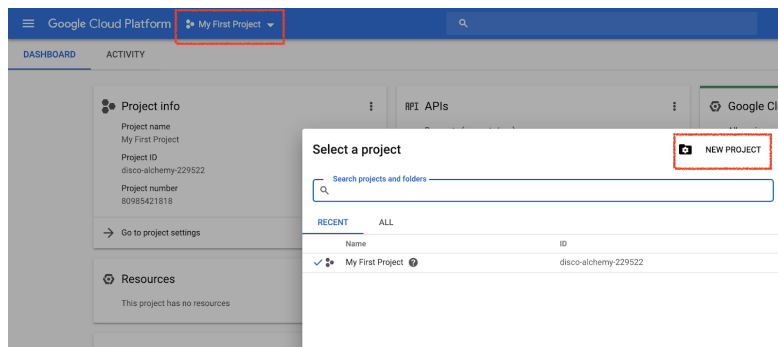
Create a new project at Google Cloud Platform

In this section, we will set up a project in Google Cloud Platform to store the data acquired by your Raspberry Pi board.

1. Sign-in to Google Cloud Platform console (console.cloud.google.com) using the same account informed in the section [Get free access to the Google cloud](#) (i.e., the account you have used to receive the Google Cloud Platform credits)



2. Once you have logged in, **create a new project** by clicking in the default project name, on the blue bar at the top of the page and by clicking in the New Project button




3. Fill in the project name field and take note of your project ID (i.e. Project ID: _____) as it will be needed later. The project ID needs to be a unique name across all GoogleCloud projects. You should name your project following the format “**proj-userID-X**”, replacing the “userID” part by your user id (login) at the Halmstad University, and replacing the X by a number, starting from 1. If you need to create a second project, increase the number used for the “X” part.

Example:

My student email at the Halmstad University is “iulzac2018@student.hh.se”. My first project at Google Cloud Platform should be “**proj-iulzac2018-1**”. The Project ID for the second project should be “**proj-iulzac2018-2**” and so on.


New Project

 You have 22 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project Name *
proj-iulzac2018-3

Project ID: proj-iulzac2018-3. It cannot be changed later. [EDIT](#)

Location *
 No organization [BROWSE](#)

Parent organization or folder

[CREATE](#) [CANCEL](#)

4. Click **Create**

5. Change to your new project by clicking on the Project Name at the top blue bar and select your new project

Google Cloud Platform **My First Project**

DASHBOARD ACTIVITY

Project info

Project name
My First Project

Project ID
disco-alchemy-229522

Project number
80985421818

[Go to project settings](#)

Resources

Select a project [NEW PROJECT](#)

Search projects and folders

RECENT ALL

Name	ID
proj-iulzac2018-3	proj-iulzac2018-3
My First Project	disco-alchemy-229522
dt8030-iulzac	dt8030-iulzac

6. If you forget your project ID, it is located in the Cloud Console in the Home area

Project info

Project name
proj-iulzac2018-3

Project ID
proj-iulzac2018-3

Project number
472709737388

[Go to project settings](#)

Enable APIs (Optional)

In this lab, we will be using Pub/Sub, Dataflow, and IoT Core APIs, so we'll need to enable those APIs.

1. **Select APIs & Services** from the **Navigation menu**.

2. Click on Enable APIs and Services
3. In the Search Bar, type in PubSub
4. Click on the resulting tile that says "Cloud Pub/Sub API" and, on the following page, "Enable API"
5. Repeat this process for Dataflow, Compute Engine and IoT Core

Create a Cloud Pub/Sub topic

To define a new Cloud Pub/Sub topic:

1. In the GCP Console, go to **Navigation menu > Pub/Sub > Topics**
2. If you see an Enable API prompt, click the **Enable API button**
3. Click **Create Topic**. The Create a topic dialog shows you a partial URL path, consisting of projects/ followed by your project name and a trailing slash, then topics/ . Confirm that the project name is the one you noted above
4. Give this string as your topic name: **"events"**

Property	Value (type or select)
Name	events

5. Then click Create

Create a BigQuery table

BigQuery is a serverless, highly scalable, low cost enterprise data warehouse and will be an ideal option to store data being streamed from IoT devices while also allowing an analytics dashboard to query the information.

Let's create a table that will hold the IoT events (board movements) that you will send from the script running in the IoT Device (Raspberry Pi).

1. Select **BigQuery** from the **Navigation menu**
2. On the left panel, select (click) your project name. After you select your project, the button Create Dataset will appear on the bottom right of your screen
3. Type **"iotData"** in the Dataset ID field and click **Create Dataset**
4. On the right panel, expand your project by clicking in the small arrow in front of your project name and click in the dataset you have create in the previous step (**iotData**)
5. On the right panel, click the **Create table**
6. In the Source Data section, select **Empty table** for the **Create table from** field
7. *Pay attention when entering the information. Any spelling errors for the field names or the table name can cause issues when Cloud Functions attempts to add data later. The field names are exactly the same field names that we are using in our code.*
8. In the **Destination** section, type **event_data** for the **Table name** field
9. In the **Schema** section, click **" + Add field "** for each field you want to create
10. Create the fields as specified in the next table:

Field name	Type	Mode
timestamp	TIMESTAMP	NULLABLE
device_id	STRING	NULLABLE
moved	INTEGER	NULLABLE
accel_x	FLOAT	NULLABLE
accel_y	FLOAT	NULLABLE
accel_z	FLOAT	NULLABLE
direction	FLOAT	NULLABLE

11. Leave the other defaults unmodified. Click **Create table**

Create a Cloud Function

For this lab, a Cloud Function will start each time a message is published to the **events** Pub/Sub topic7, will read the message and then store it in BigQuery.

1. From the **Navigation menu**, select **Cloud Functions**
2. If you see an API message, click on the **Enable API** button
3. Click on the **Create function** button
4. In the **Name** field, type **function-MovementToBQ**
5. Set the value for **Memory allocated** to 128MB
6. For **Trigger** select **Cloud Pub/Sub**
7. In the **Topic** dropdown select **"events"**
8. For **Source code**, select **Inline editor**
9. Select **"Node.js 10"** in the **"Runtime"** dropdown
10. In the **index.js** tab, paste the following code over what is there to start with. **Make sure to change the constants for projectId, datasetId and tableId to fit your environment.** If you need to find your project ID, it is at the Google Cloud Platform dashboard


```

/**
 * Triggered from a message on a Cloud Pub/Sub topic.
 *
 * @param {!Object} event Event payload and metadata.
 * @param {!Function} callback Callback function to signal completion.
 */
var BigQuery = require('@google-cloud/bigquery');
var projectId = 'YOUR-PROJECT-ID-HERE'; // IMPORTANT!! Enter your project ID here
var datasetName = 'iotData';
var tableName = 'position_data'; // Should be 'event_data'    -- Grp11

var bigquery = new BigQuery({
  projectId: projectId,
});

exports.subscribe = function (event, callback) {
  var msg = event.data;
  var incomingData = msg
    ? Buffer.from(msg, 'base64').toString()
    : '{"timestamp":"1970-01-01 00:00:00","device_id":"","moved": "0", "accel_x":"0",
"accel_y":"0", "accel_z":"0", "direction": "0" }';
  var data = JSON.parse(incomingData);

  bigquery
    .dataset(datasetName)
    .table(tableName)
    .insert(data)
    .then(function () {
      console.log('Inserted rows');
      callback(); // task done
    })
    .catch(function (err) {
      if (err && err.name === 'PartialFailureError') {
        if (err.errors && err.errors.length > 0) {
          console.log('Insert errors:');
          err.errors.forEach(function (err) {
            console.error(err);
          });
        }
      } else {
        console.error('ERROR:', err);
      }

      callback(); // task done
    });
};

```

11. In the **package.json** tab, paste the following code over the placeholder code that is there

```
{
  "name": "cloud_function",
  "version": "0.0.1",
  "dependencies": {
    "@google-cloud/bigquery": "^1.0.0"
  }
}
```

12. In the **Function to execute** field, type "**subscribe**".

13. Click the Create button

Create an IoT Core registry

1. On the **Navigation menu**, click **IoT Core** menu. If the IoT Core API is disable, you should enable it before use, by clicking **Enable API**
2. Click **Create a device registry**
3. On the **Create a registry** page, specify the following, and leave the remaining settings as their defaults:

Property	Value (type or select)
Registry ID	raspberry-events
Region	europe-west1
Default telemetric topic	projects/<project_id>/topics/events

4. Click Create

Create a device in the IoT Core Registry

In this step you will add a device to your device registry and later we will authorize this device to publish data on our Pub/Sub topic using a digital key.

1. On the **Navigation menu**, click **IoT Core** menu
2. Click on the device registry name you have created on the previous step (**raspberry-events**)
3. On the left panel, click on **Devices** and then click **Create a device**
4. Fill the follow information and leave the other information as default. You will return to this page later, because you should authorize your device (the Raspberry Pi) to publish data to the Pub/Sub Topic. In the next step you will add a digital key to authenticate the device.

Property	Value (type or select)
Device ID	rasp1
Device communication	Allow
Authentication	Enter manually (PS: You will add a digital key file later)

5. Click **Create**

Add the device key to the IoT Core Registry

To add the public key for the Raspberry Pi into IoT Core Registry we need a way to get it from the Raspberry Pi. If you can access the Raspberry pi using ssh the key entry can be done as in the Qwicklab “Streaming IoT Data to Google Cloud Storage” you did in Lab 2.

But if you do not have ssh access, you can access the IoT Core directly from the Raspberry Pi system, as described below.

Note that the required software should be already installed in your Raspberry Pi. Please see section [Set up the Raspberry Pi and the SenseHat](#) if you have not set up your Raspberry Pi system yet.

In this step, you will add the generated keys in the Google Cloud Platform. You should perform these steps **from your Raspberry Pi** system to be able to upload the keys to the Google Cloud Platform console.

1. Open the web browser (called Chromium) in your Raspberry Pi
2. Go to the Google Cloud Platform website (<https://console.cloud.google.com>)
3. Login with your Google Cloud Platform account (the same account used in the first part of this laboratory: [Create a new project at Google Cloud Platform](#))
4. Click **IoT Core** menu on the Navigation menu
5. Click on the device register created previously (**raspberrry-events**)
6. On the left panel, click on Devices and then click on the device called **rasp1**
7. 2Click on the **Add public key** button
8. Under the **Input method**, select **Upload**
9. Select **RS256_X509** for **Public key format**
10. Click the **Browse** button, on the right of the field **Public key value**
11. A new window will open. Go to your home directory by clicking in the **Home** button, in the left panel of the new window

12. On the right panel, select the **demo.pub** file and click **Open**
13. Click **ADD** at the bottom right corner of the windows **Add authentication key**

If you cannot find the “demo.pub” file in the Home directory, it means that you have not created the digital keys yet. Check the section “**Add the digital keys to your device / Create the certificates**” from the document “**Lab3 -- Building a IoT Data Pipeline: Raspberry Pi to Analytics**” to get the set-by-step instructions to create the digital keys.

Congratulations! **Your Raspberry Pi device is now authorized** to communicate with your project on the Google Cloud Platform.

Start sensing and streaming events from Raspberry Pi to the Cloud

Using the code provided in this example you will stream an events to the cloud every time the SenseHat detects that the Raspberry Pi is moving.

The code will set up a connection to the google cloud (as in previous examples), and you should implement the script to detect the movement based in the example provided in the SenseHat documentation (<https://projects.raspberrypi.org/en/projects/getting-started-with-the-sense-hat/9>).

Once the script detects that the IoT Device is moving, the script should send the event to the cloud. As already defined in the Big Query table, the event data should include:

- 'timestamp': The date/hour in UTC format of the event (current time)
- 'device_id': A string with the ID of the device (see the device_id variable at the header of the template script)
- 'moved': Should be the integer value “1” to indicate that the board was moved
- 'accel_x': A Float type value, indicating the value read by the accelerometer on the X-axis
- 'accel_y': A Float type value, indicating the value read by the accelerometer on the Y-axis
- 'accel_z': A Float type value, indicating the value read by the accelerometer on the Z-axis
- 'direction': A Float type value, indicating the value read by the magnetometer sensor

The template code was written using the Python Language, and a connection to the Google Cloud Platform using the MQTT protocol has already been defined (see the client Object in the script). You should develop the movement detection, and send the data to the cloud using the predefined MQTT connection – `client.publish(...)`.

From your Raspberry board, either using an SSH connection (see [Extra...](#)) or accessing the Raspberry Pi directly (i.e., using a computer screen, keyboard and mouse), get the code template using the git tool:

1. Go to your “home” directory using the cd command:

```
cd ~
```

2. And grab the code running the git command:

```
git clone https://github.com/izacarias/dt8030-lab4.git
```

Time to code!

Now, open the file `dt8030-lab4.py` with your preferred editor tool (for example using the Geany Editor) in the Raspberry Pi and start developing the movement detection algorithm, using the values from the accelerometer and magnetometer sensors.

Hint: Take a look at this page to understand how we can use SenseHat and Raspberry Pi to detect when the board was moved.

Some questions that needs to be answered...

What do the acceleration values actually tell us about movement? What is the unit of acceleration values? (Maybe look it up in the data-sheets for the device!)

Note that the earth mass is giving an acceleration downwards! How large is this in accelerator values?

What is a reasonable trigger value for movement?

Start the data pipeline

1. Open a new **Terminal** and run the code with this command:

```
0
```

2. Move the Raspberry Pi board to check if your script is detecting movement

Additional examples of what one can do with this sensor board can be found at:

<https://www.deviceplus.com/how-tos/raspberrypi-guide/the-sense-hat-add-on-board-for-raspberrypi-6-types-of-sensors/>

Verify that data is arriving into the cloud

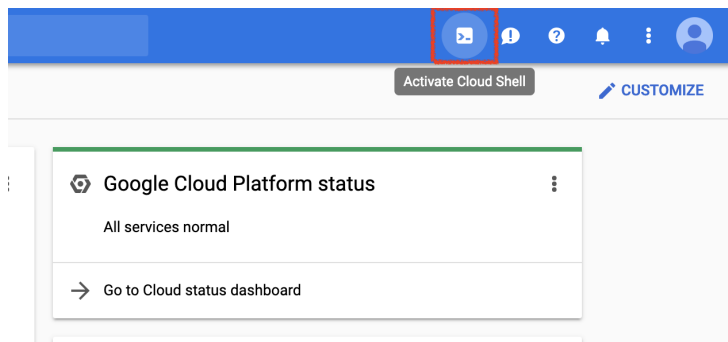
In this step, you will use the Google Cloud Shell to check if the data is arriving the the Google Cloud Platform. Google Cloud Shell provides command-line access to your GCP resources from the command line, and allows you to perform operations in Google Cloud Pub/Sub topics. You also need to create a Pub/Sub Subscription to access the Pub/Sub Topic.

Create a Pub/Sub Subscription

1. Click **Navigation menu > Pub/Sub > Topic**
2. Locate the topic you have created ("**events**"), click the **three dot icon > New subscription**
3. Type the name for the subscription: **testSub**
4. Set the **Delivery Type** to **Pull**, then click Create

Access the Pub/Sub Topic using the Cloud Shell

1. In GCP console, on the top right toolbar, click the **Open Cloud Shell** button



2. If a dialog box opens, click **START CLOUD SHELL**
3. To view the message you'll use the **testSub** subscription to **pull** the messages from the Pub/Sub topic (**events**). Enter the following command in the Google Cloud Shell command line:

```
gcloud pubsub subscriptions pull --auto-ack testSub
```

4. The messages appear in the DATA field of the command output

Examine the stored data

1. On the **Navigation menu**, click **BigQuery**
2. In the left-hand side of the browser window, click on **Compose New Query**
3. Enter the following query. Remember to replace **<project_id>** with the ID of your Google Cloud2 Platform project:

```
SELECT *  
FROM `<project_id>.iotData.event_data`  
LIMIT 1000
```

4. Browse the data using the previous-page and next-page links provided

Outcomes from this lab

Make a short report, answering the following questions:

- What do the acceleration values actually tell us about movement?
- What is the unit of acceleration values?
- How is earth mass acceleration showing up in the accelerator values?
- How large is earth mass acceleration in accelerator values?
- What is a reasonable trigger value/function for movement?
- How often do we need to sample to be sure to detect movement?
- If we want to do the detection in google cloud, what are the cloud tools one could use to achieve this (explain the tools needed and how they connect)?

Include the code that implements the movement trigger in the report.

You should also estimate the reduction in data transmission with your movement detector at the edge, compared with having the same detector in the cloud. This will of course depend on how often the device is moved, but let us assume a movement every 10 seconds.

The datasheets for the sensor components on the SenseHat can be found at:

<https://www.raspberrypi.org/blog/astro-pi-tech-specs/>

Inertial measurement sensor: ST LSM9DS1 ([data sheet](#)). This is a 3D accelerometer, 3D gyroscope and 3D magnetometer combined in one chip.

The report should be submitted to BlackBoard, under the assignment “Lab 4 Submission”.

Cleaning up

Once you are done experimenting with the data, you can remove the running resources.

If the python script still running in your Raspberry Pi, hit Ctrl-C in the terminal window to stop the script and then type the following to power down the Raspberry Pi

```
sudo shutdown -h now
```

Go to Cloud Functions (on Google Cloud Platform), click on the checkbox next to **function-MovementToBQ** item and then click on **Delete**

Go to **Pub/Sub** (on Google Cloud Platform), click on the checkboxes next to the **events** topic and then click on **Delete**

Go to **BigQuery**, click the down arrow next to your project name, click the down arrow to the right of the **iotData** dataset, then click on **Delete dataset**.

When prompted, type in the dataset ID (**iotData**) in order to finish deleting the data.

Go to **IoT Core** (on Google Cloud Platform), click on the Registry ID (**raspberry-events**) then click on “**Devices**” on the left panel. Click on the device name (**rasp1**), then click “**DELETE**”. Confirm deletion by typing the device ID (**rasp1**). Again, on the left panel click “**Registry details**” and click “**Delete Registry**”. When prompted, type in the Registry ID (**raspberry-events**) in order to finish deleting the registry.

References

“Setting up your Raspberry Pi”. Online:

<https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up>

“Getting started with the Sense HAT”. Online:

<https://projects.raspberrypi.org/en/projects/getting-started-with-the-sense-hat>

“Getting started with the Sense HAT - Detecting movement”. Online:

<https://projects.raspberrypi.org/en/projects/getting-started-with-the-sense-hat/9>

“Streaming IoT Data to Google Cloud Storage”. Online:

<https://www.qwiklabs.com/focuses/2765?parent=catalog>

“Cloud IoT step-by-step: Connecting Raspberry PI + Python”. Online:

<https://medium.com/google-cloud/cloud-iot-step-by-step-connecting-raspberry-pi-python-2f27a2893ab5>

“Using IoT Core to Stream Heart Rate Data”. Online:

<https://codelabs.developers.google.com/codelabs/iotcore-heartrate/index.html>

“Visualizing Data with Google Data Studio”. Online:

<https://www.qwiklabs.com/focuses/1158?parent=catalog>