

Google Cloud IoT Core and i.MX7D Development Platform for Android Things

Quick Start Guide

1. Overview

This tutorial helps developers get started with the NXP based development platform for Android Things – PICO-i.MX7D board, software support, and the Google Cloud IoT Core. Specifically, it walks through the hardware setup, Android Things image build, board booting process, and how to enable and publish a sensor hub demo on a Google Cloud IoT PubSub topic. Refer to page 13, section 12 for the Google Cloud IoT demo and setup.

Code development, build, and unit testing take place on the developer's host computer. The resulting image is flashed to the target hardware for further integration testing and debugging over USB or Ethernet. Just as Android Things is Android-based, the software development leverages Android development tools including ADB (Android Development Bridge) and FASTBOOT mode to interact with the target.

This development platform together with the Board Support Package software aim to enable faster development of IoT devices based on Android Things, and flexible hardware/software customization needed for the particular device.

2. Hardware Requirement

The development kit contains:

- PICO-i.MX7D-eMMC System-On-Module (SOM)
- PICO- carrier board (pre-assembled with the SOM)

Besides, other required materials include:

- Cables:
 - For ADB/FASTBOOT/MFGTool**
 - USB type-A to USB type-C cable
 - Serial console:**
 - USB type-A to micro USB
- WiFi antenna (IPEX interface)

3. Getting Familiar with the Development Platform

For more information on the platform, go to the following link:

https://www.technexion.com/support/download-center/?wpv-product=pico-imx7-emmc&wpv_aux_current_post_id=78&wpv_view_count=181-TCPID78

The key interfaces of the board are shown in Figure 1:

- USB to serial console convertor interface (Number 1 in figure 1)
- WiFi+Bluetooth antenna connector (Number 5 in figure 1)
- Microphone and headphone jack (Number 3 in figure 1)
- USB OTG (USB Type-C) and power supply interface (Number 4 in figure 1)

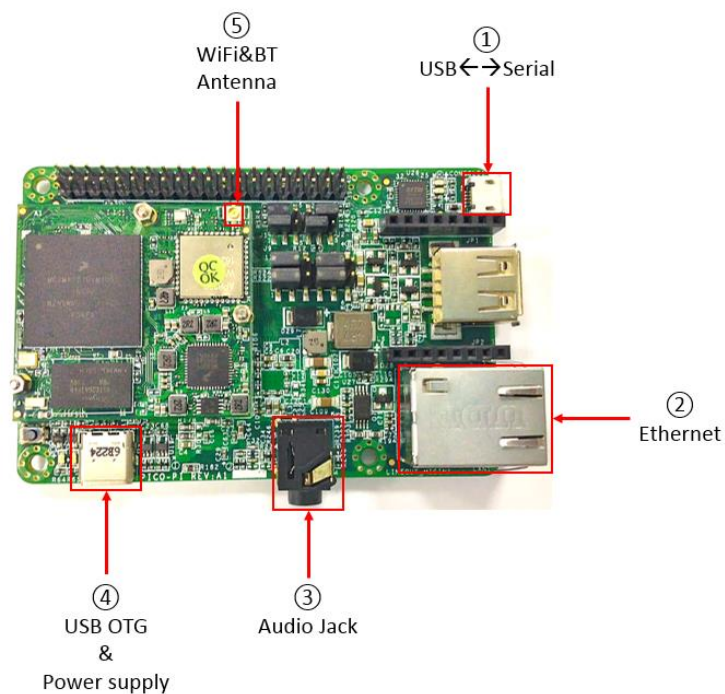


Figure 1. Top view of the PICO-i.MX7D board

- Take a close look at the jumpers on the top view of the board. There are two different setup for download mode and boot mode. Continue reading for further details.

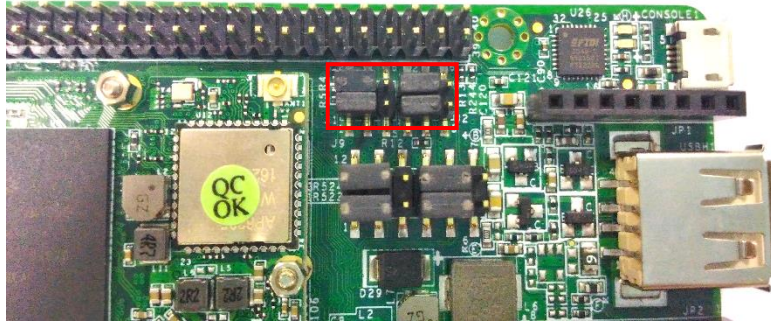


Figure 2a. Jumper Setup (Download Mode)

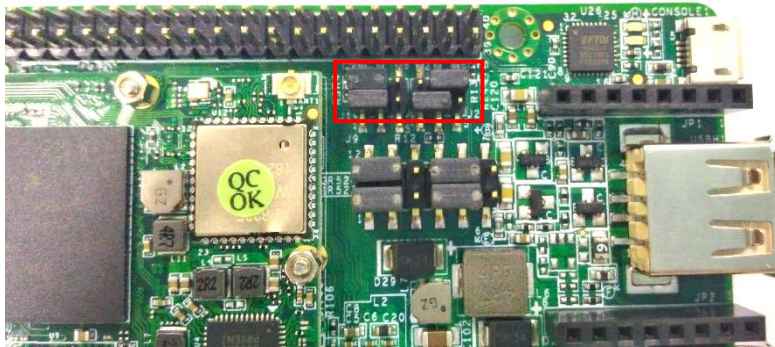


Figure 2b. Jumper Setup (Boot Mode)

4. Connect the board and host computer

1. Connect the USB type-A to micro USB's micro USB end to the micro USB interface (number 1 in figure 1).
2. Connect the WiFi antenna to connector (number 5 in figure 1).
3. Get a USB type A to USB type C cable. Plug the USB type C end to the USB OTG type C connector (number 4 in Figure 1) for ADB and FASTBOOT interface. Plug the other end of the USB cable to your computer. This interface also be the power supply for the board.

5. Instructions to set up the serial console terminal

1. Make sure the you connect to the UART serial console as shown in step 3 in "Connect the board and host computer" section
2. Start the serial communication software
3. Choose operating system of host computer– Window
 - a. Once the PC recognizes the virtual USB to UART device, it can be seen in your PC Device Manager list. You can determine the port number of the virtual COM port by looking under the "Ports" group.
 - b. With the serial port driver installed, run your favorite terminal application (putty, minicom, etc.) to view the serial output from i.MX7D microprocessor's UART.

Recommended settings for the serial connection:

Serial port configuration: 115200 baudrate, 8 data bits, 1 stop bit, no parity.

Note: The PC needs a driver to enable a virtual COM port through the PC USB port. Please consult www.ftdichip.com/Documents/InstallGuides.htm to download the correct driver.

Set up serial communication terminal in Putty as below:

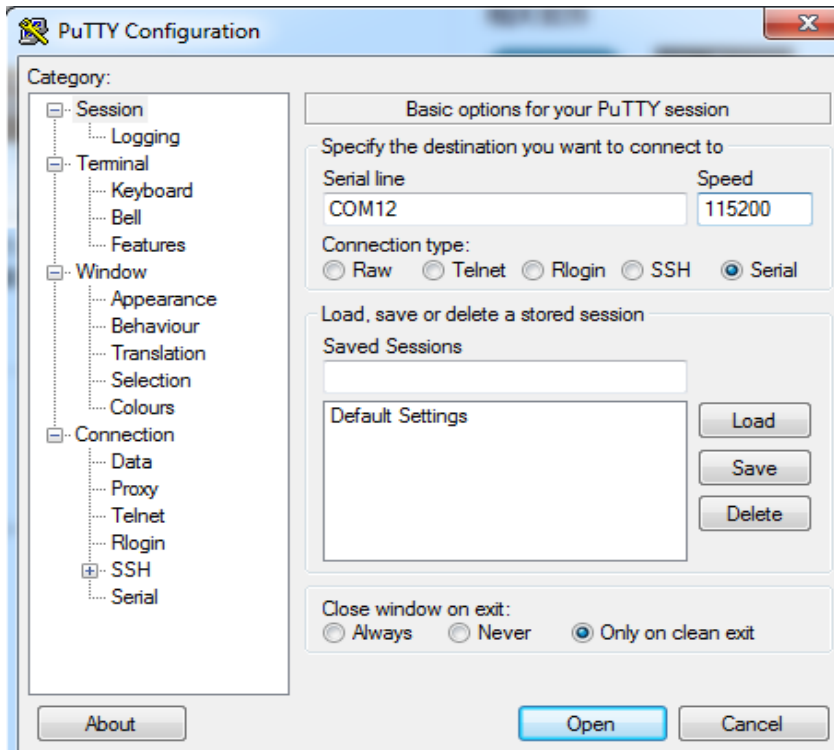


Figure 7a. Screenshot of Putty

4. Choose operating system of host computer – Ubuntu
Install Minicom on host computer as below commands:

```
$ sudo apt-get install minicom
```

Set up serial communication terminal in Minicom as below:

```
ad+-----+
ad| A - Serial Device      : /dev/ttyUSB0
ad| B - Lockfile Location  : /var/lock
nd| C - Callin Program     :
dr| D - Callout Program    :
dr| E - Bps/Par/Bits       : 115200 8N1
dr| F - Hardware Flow Control : No
dr| G - Software Flow Control : No
dr|
dr|   Change which setting? █
dr+-----+
droid work| Screen and keyboard |ONNECTED
```

Figure 7b. Screenshot of Minicom

6. Download Mode and Boot Mode

The board is designed as booting from the internal eMMC. There are two modes for the PICO-i.MX7D board. One is the download mode in which the board will receive the instructions from MFG Tools to flash images to boot storage such as eMMC. The other one is boot mode in which the board will load the image from the boot storage and boot from the image.

The board comes with a working image burned in eMMC. To boot the board from that image, you can boot the board directly with the power supply connected. Please make sure the board is in Boot Mode with the jumper setting as above.

7. Prepare Android Things Images

7.1 Prebuilt Android Things Images

Download the prebuilt Android Things images at:

<https://developer.android.com/things/preview/download.html>

7.1.1 Android Things Image Introduction

The table describes Android Things images and the targeted eMMC partition where the Android Things images to be flashed into:

Image Name	Image Description	Target Parition
u-boot.imx	The u-boot bootloader image, which is the first code run after the PICO-i.MX7D board hardware reset. It will load and jump to the boot.img either from Slot a's boot partition or Slot b's boot partition, based on the meta data stored in misc partition	The first boot partition of PICO-i.MX7D-emmc
partition-table.img	The GUID Partition Table image, which define the partitions in the PICO-i.MX7D-emmc	gpt partition(Slot a's boot partition) for PICO-i.MX7D-emmc

boot.img	The Android Things boot image which is composed by Linux kernel zImage, linux kernel dtb(Device Tree Binary) file, Android Things ramdisk image, and linux kernel boot arguments. The code in boot.img will mount the related system.img based on the meta data stored in misc partition.	boot_a partition(Slot a's boot partition) for PICO-i.MX7D-emmc boot_b(Slot b's boot partition) for PICO-i.MX7D-emmc
userdata.img	The Android Things user data image	userdata partition for PICO-i.MX7D-emmc
system.img	The Android Things system image which includes all Android Things related binaries, libraries, and system configuration files.	system_a partition(Slot a's system partition) for PICO-i.MX7D-emmc system_b partition(Slot b's system partition) for PICO-i.MX7D-emmc
gapps.img	The Google application image.	gapps_a partition(Slot a's system partition) for PICO-i.MX7D-emmc gapps_b partition(Slot b's system partition) for PICO-i.MX7D-emmc
oem.img	The oem image.	oem_a partition(Slot a's system partition) for PICO-i.MX7D-emmc oem_b partition(Slot b's system partition) for PICO-i.MX7D-emmc

7.2 The mfgtools

The mfgtools can be downloaded at

[http://www.nxp.com/products/software-and-tools/software-development-tools/i.mx-software-and-tools/iot-development-platforms-based-on-i.mx-6ul-processor-and-android-things-os:IOT-DEV-PLATFORMS-i.MX6UL?tab=Design Tools Tab](http://www.nxp.com/products/software-and-tools/software-development-tools/i.mx-software-and-tools/iot-development-platforms-based-on-i.mx-6ul-processor-and-android-things-os:IOT-DEV-PLATFORMS-i.MX6UL?tab=Design%20Tools%20Tab)

8. Testing and Debugging Tools

Unit tests run locally on the developer's host computer and integration tests interact with the target device via ADB.

Pico-imx7d board is Android Things compatible and provides full support for ADB and FASTBOOT over USB for debugging.

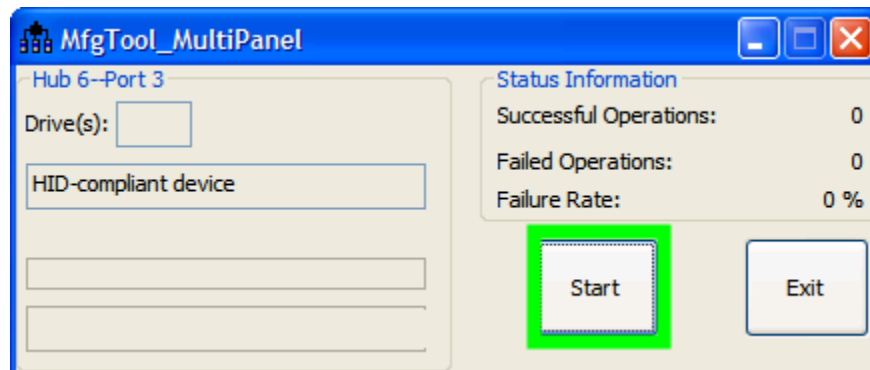
ADB and FASTBOOT are the tools in Android SDK. Please refer to the link <http://developer.android.com/sdk/index.html#Other> to download the latest version of Android SDK

9 Flash Android Things Images

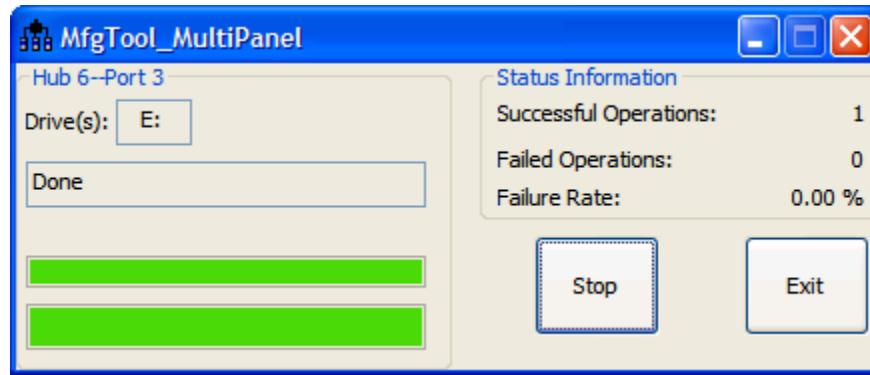
By default, a valid bootloader binary has been flashed into the PICO-i.MX7D board. It will make the board into FASTBOOT mode if Android Things Images are not been flashed yet. Please refer "11. Instructions to make board into FASTBOOT mode" to check whether your board is into FASTBOOT mode.

9.1 Flash Android Things bootloader binary with MFG Tools

1. Unzip the **mfgtools.tar.gz** file to a selected location. The directory is named MFGTool-Dir in this example.
2. Make your board into Serial download, as explained in Chapter 3, figure 2a.
3. Power on the board. Using USB cable on the Pico OTG port, connect your WINDOWS/LINUX PC with Pico-imx7d.
4. On WINDOWS, double click the file "**mfgtool2-brillo-mx7d-pico-emmc-firmware.vbs**" to flash only the uboot.imx of Android Things, or double click the file "**mfgtool2-brillo-mx7d-pico-emmc.vbs**" to flash all Android Things images. Then click "Start".



The image below shows what the tool will become once the download is complete.



For more information on the MFGTool, please check the “Manufacturing Tool V2 Quick Start Guide.docx” under the mfgtools\Documents path.

5. Program images in Linux OS:
 - a: In Linux, run below commands to flash the uboot.imx image of Android Things.
`sudo ./linux-runvbs.sh mfgtool2-brillo-mx7d-pico-emmc-firmware.vbs`
 - b: In Linux, run below commands to flash all the images of Android Things.
`sudo ./linux-runvbs.sh mfgtool2-brillo-mx7d-pico-emmc.vbs`

Note: If blocked, please plug out the USB OTG cable, then plug in.
6. Power off, set the board is in Boot Mode.

9.2 Provision Android Things images with FASTBOOT mode

1. Download Android Things images package for PICO i.MX7D from <https://developer.android.com/things/preview/download.html>, and unzip it.
2. Refer “11. Instructions to make board into FASTBOOT mode” to make the board into FASTBOOT mode.
3. Flash Android Things images with either of the two ways below:
 - 3.1 Flash all images with the shell script in Android Things images package
 - Execute the batch file *iot-flashall-imx7d.bat* On WINDOWS PC
 - Execute the shell script *iot-flashall-imx7d.sh* On LINUX PC
 - 3.2 Flash Android Things images with fastboot command

Execute below commands in Linux PC to flash the related images

Image File Name	Partition Name	Fastboot command
u-boot.imx	bootloader	<code>\$fastboot flash bootloader u-boot.imx</code>
partition-table.img	gpt	<code>\$fastboot flash gpt partition-table.img</code>
boot.img	boot_a/boot_b	<code>\$fastboot flash boot_a boot.img</code>

		<i>\$fastboot flash boot_b boot.img</i>
system.img	system_a/system_b	<i>\$fastboot flash system_a system.img \$fastboot flash system_b system.img</i>
userdata.img	userdata	<i>\$fastboot flash userdata userdata.img</i>
gapps.img	gapps_a/gapps_b	<i>\$fastboot flash gapps_a gapps.img \$fastboot flash gapps_b gapps.img</i>
oem.img	oem_a/oem_b	<i>\$fastboot flash oem_a oem.img \$fastboot flash oem_b oem.img</i>

Note: The partitions boot_a, boot_b, system_a, system_b and userdata are defined by the partition-table.img flashed in board's eMMC. The partition-table.img should be flashed into board's eMMC before flashing those partitions.

4. Run below commands in Linux PC to make the board in lock state, and reboot the board

\$fastboot flashing lock

\$fastboot reboot

10. Boot Android Things

After flashing the images, you can boot the board directly with the power supply connected. Please make sure the board is in Boot Mode

10.1 Change boot arguments

By default, the u-boot will take the boot arguments stored in Android Things' boot.img. Below is an example in case you need to change the default boot arguments used by u-boot.

```
U-Boot > setenv bootargs console=ttyMXC4,115200 init=/init
androidboot.console=ttyMXC4 androidboot.hardware=imx7d vt.global_cursor_default=0
rootwait ro
U-Boot > saveenv
U-Boot > boot
```

11. Instructions to make board into FASTBOOT mode

FASTBOOT mode is a state in which the board will respond the commands from host PC FASTBOOT commands to flash Android Things images or query board information. The board should connect with your host PC through USB type-A to USB type-C cable.

Turn on the board and stop the boot process to enter to uboot prompt. Then run the following uboot command:

```
$fastboot usb
```

11.1 Check whether the device is into FASTBOOT mode

You can check whether your board is into FASTBOOT mode through fastboot commands on your PC

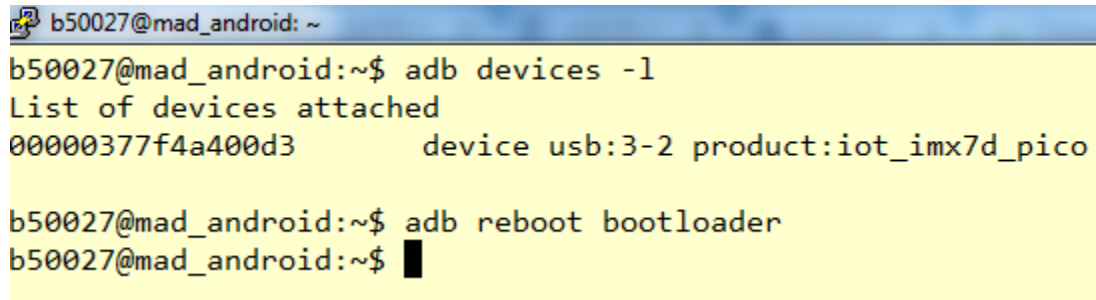
Commands with return string	Device in FASTBOOT mode
<pre>~\$ fastboot devices 000000f674a400d3 fastboot</pre>	Yes
<pre>~\$ fastboot devices</pre>	No

11.2 Set the device into FASTBOOT mode

If your device isn't into FASTBOOT mode, you can set the device into FASTBOOT mode with either of the two ways below:

1. Via adb command line

- Once you confirm that you have access the device through adb on your PC, run command “adb reboot bootloader” as shown below:

A terminal window screenshot showing the execution of adb commands. The prompt is b50027@mad_android: ~. The first command is 'adb devices -l', which outputs 'List of devices attached' followed by '00000377f4a400d3 device usb:3-2 product:iot_imx7d_pico'. The second command is 'adb reboot bootloader', which is followed by a cursor on the next line.

```
b50027@mad_android: ~
b50027@mad_android:~$ adb devices -l
List of devices attached
00000377f4a400d3    device usb:3-2 product:iot_imx7d_pico

b50027@mad_android:~$ adb reboot bootloader
b50027@mad_android:~$ █
```

2. Via serial console

- Once the board completed booting-up, type the following commands in your serial console window:

```
$su
```

```
$reboot bootloader
```

Once the device is in FASTBOOT mode, your serial console will look similar to the screen shown below:

```
U-Boot 2015.04-00079-g2c14d01 (Feb 10 2017 - 16:52:42)

CPU:   Freescale i.MX7D rev1.2 at 792 MHz
CPU:   Temperature 37 C
Reset cause: POR
Board: i.MX7D PICOSOM
I2C:   ready
DRAM:  512 MiB
PMIC:  PFUZE300 DEV_ID=0x30 REV_ID=0x11
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
No panel detected: default to EJ050NA
Display: EJ050NA (800x480)
Video: 800x480x24
In:    serial
Out:   serial
Err:   serial
flash target is MMC:1
Net:   FEC0
Fastboot: Got bootloader commands!
```

CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.5

To get the device out of FASTBOOT mode, run command “fastboot reboot” from your PC.

12. Android Things Cloud IoT Demo

This demo shows how to implement a sensor hub on Android Things that collects sensor data from connected sensors and publish on a Google Cloud IoT PubSub topic.

12.1 Features

- Connection parameters are configurable via intent and configuration is saved in sharedpreferences
- Sensor robustness: you can remove and add sensors at runtime and the app will adapt accordingly
- Network robustness: device can loose connectivity. When connectivity is restored, it will auto-reconnect
- Power robustness: device can loose power. When is reboots, it will auto-reconnect
- Sensor data collected since the last publish is sent to pubsub every 20 seconds
- Sensor data is collected either as continuous mode or onchange mode. Continuous mode sensors (temperature and pressure) publishes only the most recent value. Onchange mode sensors (motion detection) stores up to 10 sensor changes in between pubsub publications.

12.2 Requirements

This section covers the Hardware and Software requirements needed to enable this sensor hub demo.

12.2.1 Hardware requirements

i.mx7d-pico-pi board.

USB A to USB Type C cable.

PC with internet connection and USB port.

Ethernet cable or Wifi connection with internet access.

Optional Hardware (at least one is required):

[Rainbow HAT board](#)

[PIR motion detector sensor](#), 3 female to female jumper wires.

Push button, breadboard, 2 female to male breadboard jumper wires.

12.2.2 Software requirements

Android Studio 2.2+

[Google Cloud Platform](#) project with Cloud IoT support enabled

12.3 Setup

This section talks about setting up the Hardware and Software for this demo.

NOTE:

This sections assumes the board has been flashed with Android Things previously. If not, go to the board flash section.

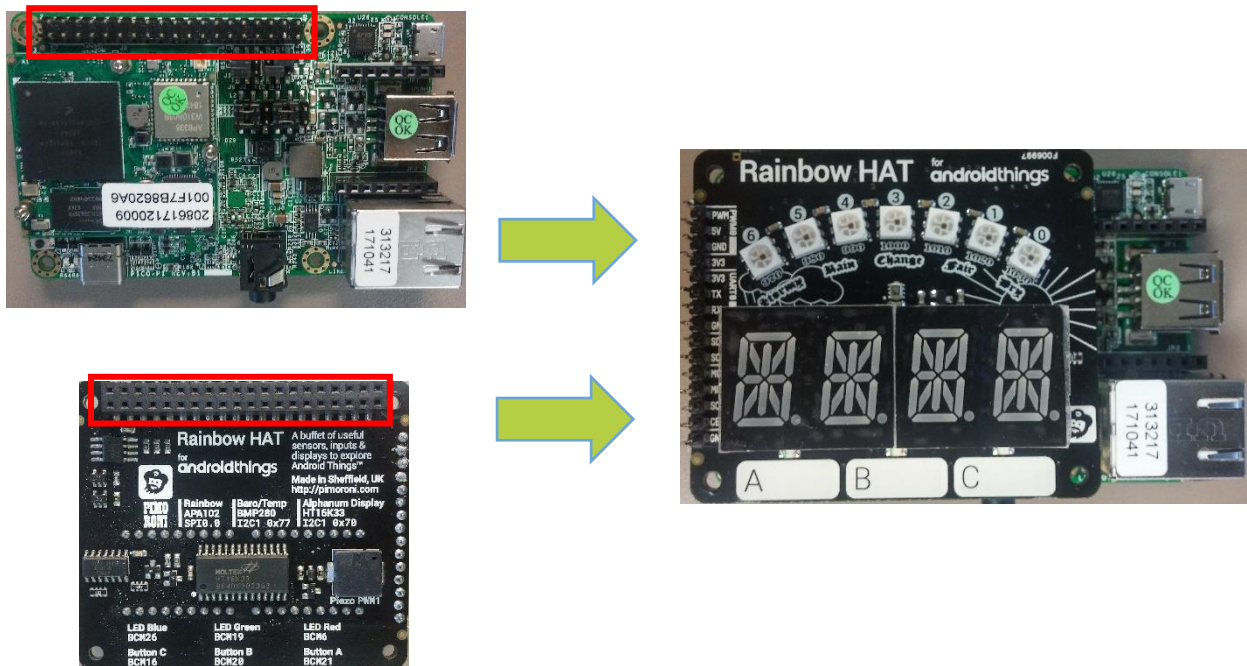
12.3.1 Hardware setup

Three different Hardware can be set up. First two need external boards. The third one uses a push button only.

12.3.1.1 Accessory connection

12.3.1.1.1 Rainbow HAT board

Connect the Rainbow HAT board to the Pico-Pi imx7d through the pin header. Check the picture below.



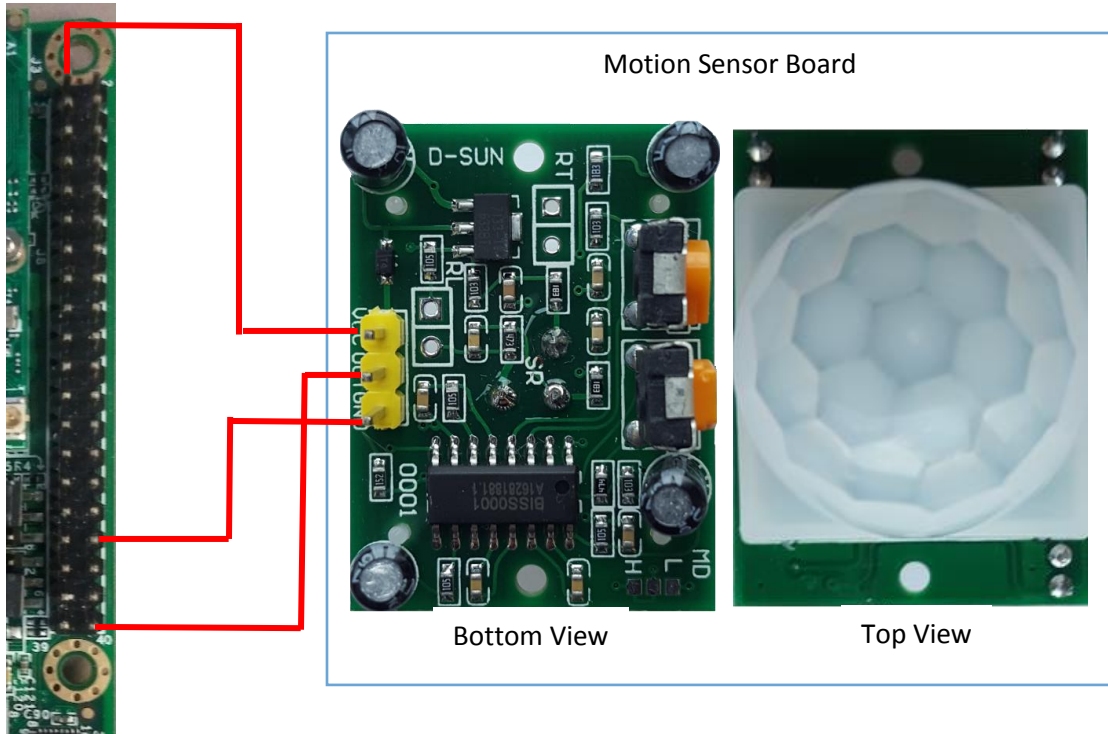
12.3.1.1.2 Motion Sensor

Connect the motion sensor board to the Pico-Pi imx7d through the pin header. Use the 3 female to female jumper wires.

Connection configuration

Pico Pi header pin	PicoPi board signal name	Motion sensor signal name
1	3.3V	VCC
40	GND	GND
34	GPIO_174	OUT

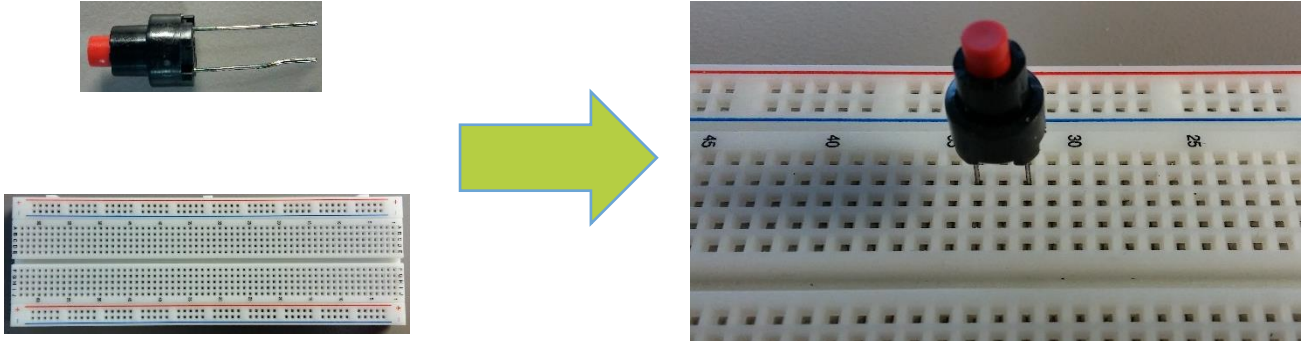
PicoPi Header



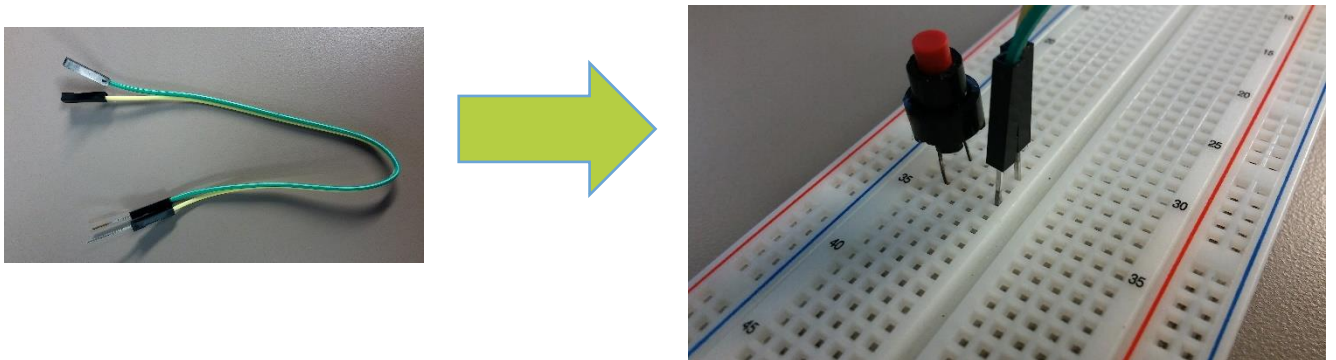
12.3.1.1.3 Push Button

In case you don't have a motion sensor, you can emulate it with a push button. Every time the button is pushed, a message will be published.

1. Plug the push button into the breadboard



2. Plug the male side from the jumper wires into the breadboard. On the same column as the push button pins are.



3. Plug the female side from the jumper wires into the pico-pi board header. Pins 1 and 40. It does not matter the orientation of the cables.



12.3.1.2 Internet access

Internet access can be enabled by Ethernet or Wifi on the Pico-Pi board.

12.3.1.2.1 Ethernet

Connect the RJ45 cable to the Ethernet port with Internet connection.



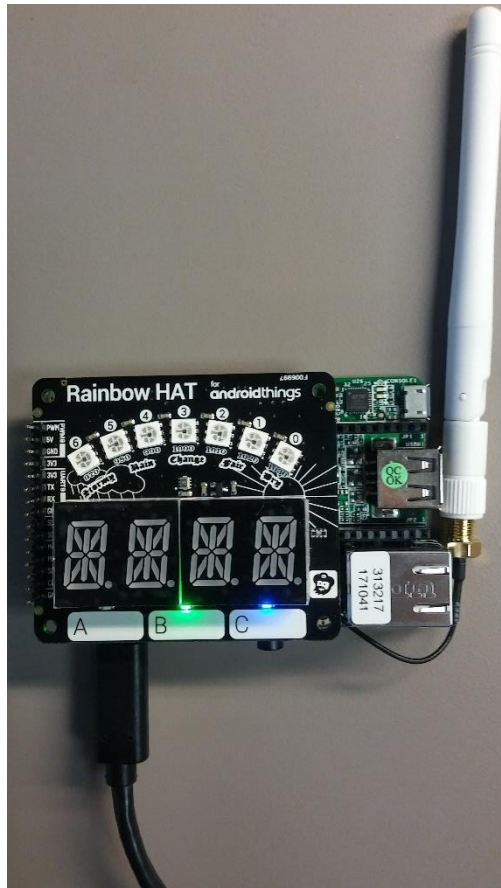
12.3.1.2.2 Wifi

Make sure the wifi antenna is connected on the top board. Check section 12.3.X to setup the Wifi connection from Android things.



12.3.1.3 USB power interface

Power on the board by connecting the USB type C cable. Connect the other side of the cable to your Windows 7 PC. Make sure you already have flashed the Android Things image in the board.



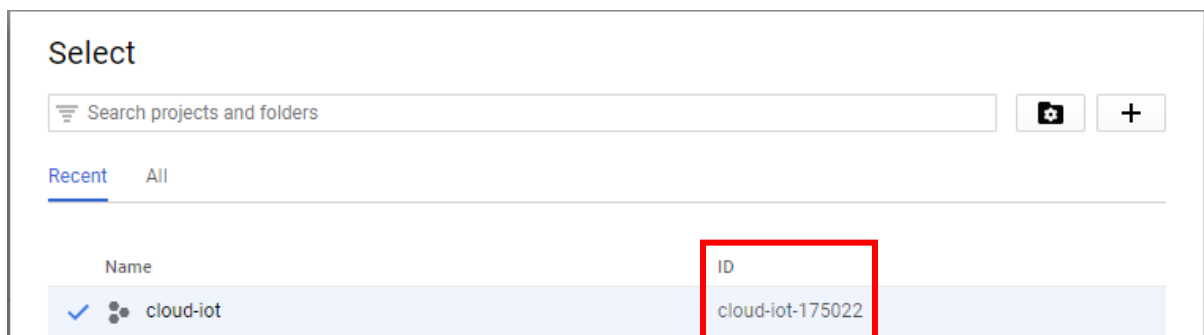
12.3.2 Cloud configuration and building process

12.3.2.1 Google Cloud IoT setup

1. Make sure you have Google account open.
2. Go to the Google Cloud Platform [GCP Console](#) and login.
3. Create a project named "cloud-iot".

NOTE:

The project name doesn't necessary have the same project id. To locate the project Id, click on the project name on your GCP and check what project id was assigned. In this example the project id is "cloud-iot-175022"



4. Enable Pub/Sub API

Google Cloud Platform cloud-iot

Home

STACKDRIVER

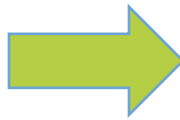
- Monitoring
- Debug
- Trace >
- Logging >
- Error Reporting

TOOLS

- Container Registry >
- Source Repositories >
- Deployment Manager
- Endpoints

BIG DATA

- BigQuery
- Pub/Sub >**

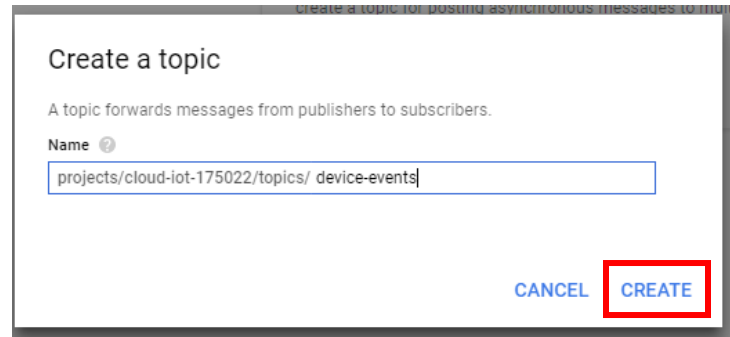
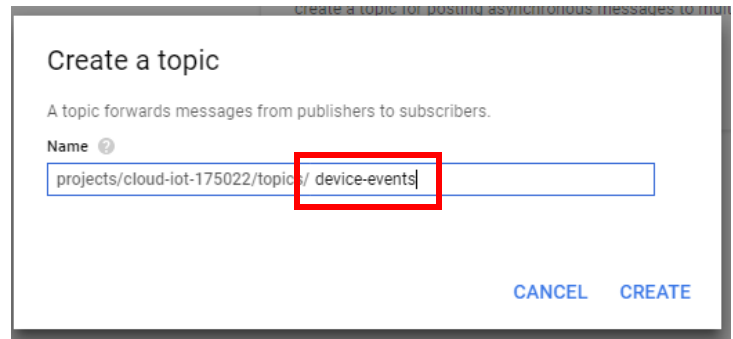
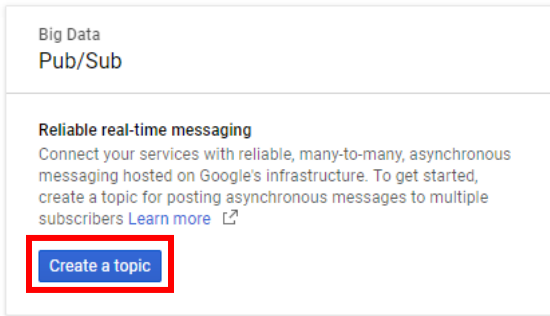


Big Data
Pub/Sub

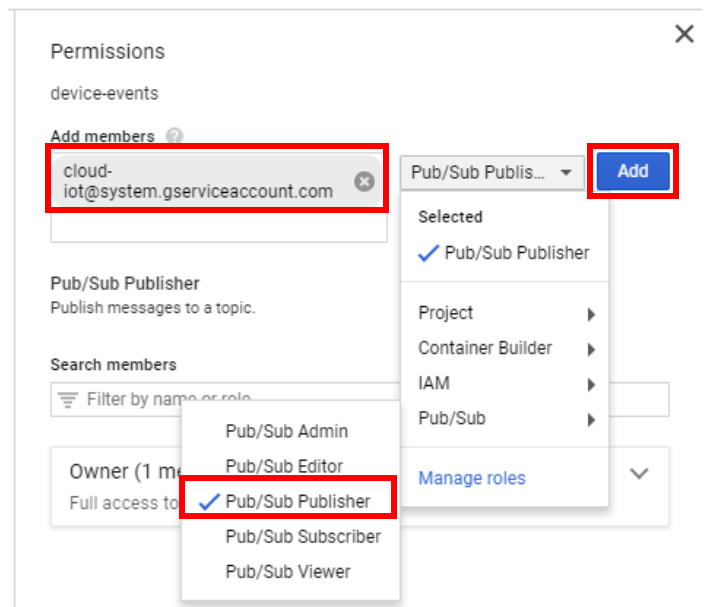
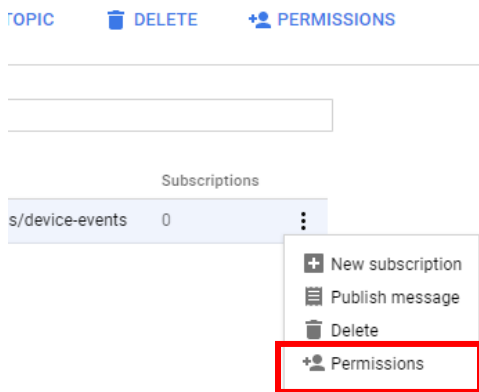
Reliable real-time messaging
Connect your services with reliable, many-to-many, asynchronous messaging hosted on Google's infrastructure. To get started, create a topic for posting asynchronous messages to multiple subscribers [Learn more](#)

Enable API

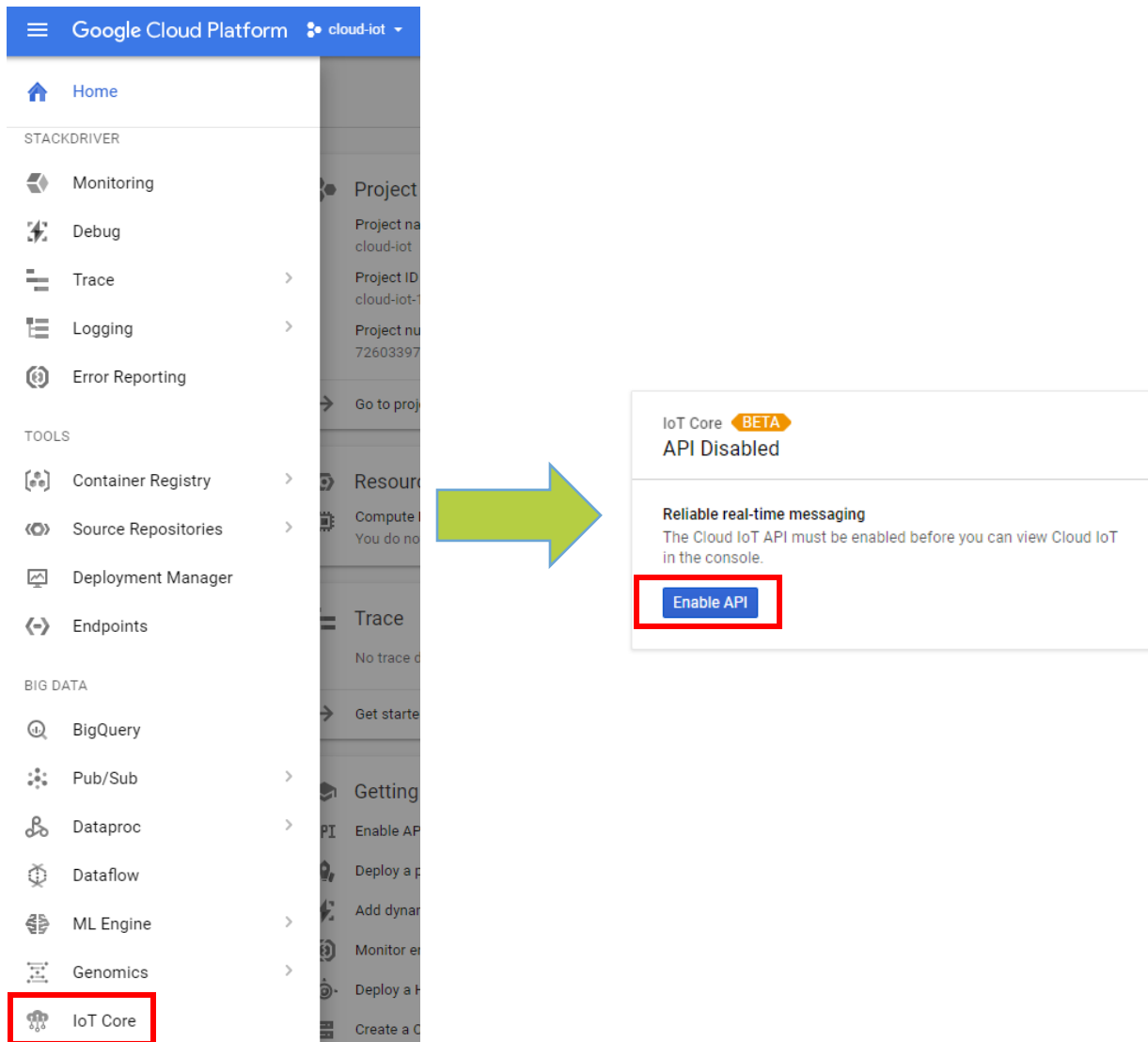
5. Under the Pub/Sub topics window, create a new topic called “device-events”.



6. In the Cloud Platform Console, select the topic and click the PERMISSIONS button at the top of the page. This will open the IAM permissions editor in the right-side panel. Add the member cloud-iot@system.gserviceaccount.com with the role Pub/Sub Publisher.



7. Enable Cloud Vision API.



The image shows a screenshot of the Google Cloud Platform console. On the left, the navigation menu is open, and the 'IoT Core' option is highlighted with a red box. A green arrow points from the 'IoT Core' option to a notification box on the right. The notification box has a white background and a thin border. At the top, it says 'IoT Core BETA API Disabled'. Below this, there is a section titled 'Reliable real-time messaging' with the text 'The Cloud IoT API must be enabled before you can view Cloud IoT in the console.' At the bottom of the notification box, there is a blue button with the text 'Enable API', which is also highlighted with a red box.

Google Cloud Platform cloud-iot

Home

STACKDRIVER

- Monitoring
- Debug
- Trace
- Logging
- Error Reporting

TOOLS

- Container Registry
- Source Repositories
- Deployment Manager
- Endpoints

BIG DATA

- BigQuery
- Pub/Sub
- Dataproc
- Dataflow
- ML Engine
- Genomics
- IoT Core**

Project

Project na
cloud-iot

Project ID
cloud-iot-

Project nu
72603397

Go to proj

Resource

Compute I
You do no

Trace

No trace d

Get starte

Getting

PI Enable AP

Deploy a p

Add dynar

Monitor e

Deploy a F

Create a C

IoT Core **BETA**
API Disabled

Reliable real-time messaging
The Cloud IoT API must be enabled before you can view Cloud IoT in the console.

Enable API

8. Under IoT Core window, create a new device registry called “registry-iot”. Select “us-central1” on the Cloud region. Select “device-events” on pub/sub topic.

The screenshot shows the IoT Core console interface. On the left, the 'Device registries' page is visible with a 'Create device registry' button highlighted in a red box. A green arrow points from this button to the 'Create device registry' form on the right. The form has the following fields:

- Registry ID**: registry-iot
- Cloud region**: us-central1
- Protocol**: MQTT
- Pub/Sub topic**: projects/cloud-iot-175022/topics/device-events

Below the form, there is a 'Pub/Sub permissions' section with a note about granting the 'Pub/Sub Publisher' role. At the bottom of the form, a 'Create' button is highlighted in a red box, with a 'Cancel' button next to it. A green arrow points from the 'Create' button down to the resulting registry list.

The resulting registry list is shown at the bottom of the screenshot:

Registry ID	Region	Protocol	Topic
registry-iot	us-central1	MQTT	projects/cloud-iot-175022/topics/device-events

9. At the end of last step, you should have the following information. This will be needed at the upcoming steps.
- Project ID → cloud-iot-175022
 - Registry ID → registry-iot
 - Cloud region → us-central1

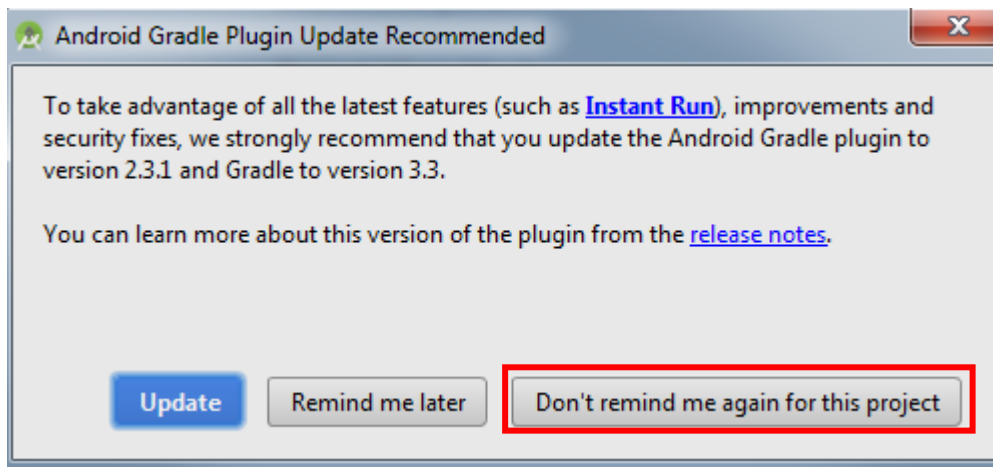
12.3.2.2 Building process

1. On your PC, clone the sensorhub-cloud-iot.

```
$git clone https://github.com/androidthings/sensorhub-cloud-iot.git
```

2. Open Android Studio, import the sensorhub-cloud-iot project.

NOTE: If a pop window is showed about to update the Android gradle plugin, click on “Don’t remind me again for this project”.



3. Make sure the Pico-Pi is connected to your PC through the USB type C connector. At this point Android Things has booted up on the board.
4. Click on the “Run” button to install the application.
5. After the application has been installed, use the adb tool to reboot the board. This will grant write permission to the application to write on disk. The application will restart automatically after the reboot.

```
$adb reboot
```

6. Once the system has been rebooted, use ADB tool to set the date on the system. This is the date format required MMDDhhmm[YY]

```
$adb root
$adb shell date 071816322017
```

NOTE:

Every time the board is powered or rebooted the date must be set. Use the GMT current time, otherwise the system will not publish the information.

- Once the system has been rebooted, use the ADB tool to start the mqtt service on the application. Pass the parameters created on the Google Cloud IoT setup section. At this point, any device hasn't been registered yet on the GCP (Google Cloud Platform). First, we need to create it on the board (my-device-1) and then we will add it to the GCP.

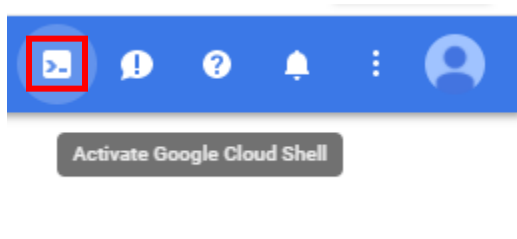
```
$adb shell am startservice -a
com.example.androidthings.sensorhub.mqtt.CONFIGURE -e project_id
cloud-iot-175022 -e cloud_region us-centrall -e registry_id
registry-iot -e device_id my-device-1
com.example.androidthings.sensorhub/.cloud.CloudPublisherService
```

- Once the service has started, the application will create two keys on the device. One private and the other public. The private key will be stored at the Android Keystore. The public one will be printed to logcat and will be available as a file on your external storage location. This file will be needed to register this device on the GCP. Use ADB tool to retrieve this file.

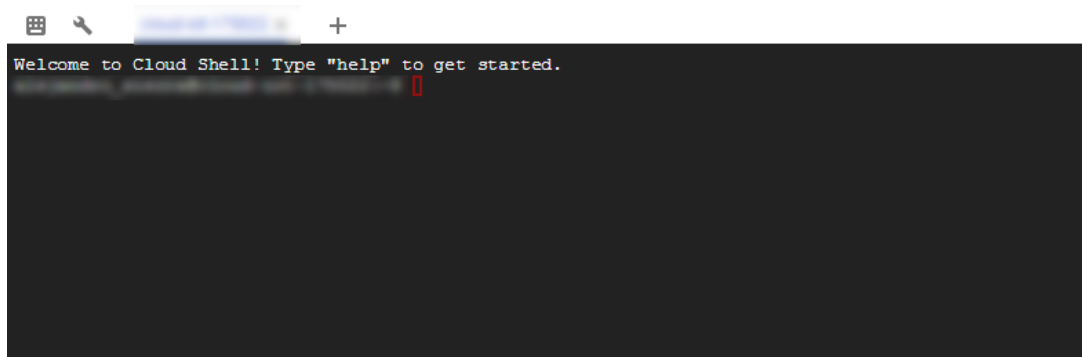
```
$adb pull sdcard/cloud_iot_auth_certificate.pem .
```

12.3.2.3 Adding device to GCP

- On your PC, go to the GCP console (at your browser)
- On the right up corner, click on the "Activate Google Cloud Shell" to open the cloud shell.



- Make sure the GCS (Google Cloud Shell) has opened on your browser. At the bottom of the screen.



4. Upload the file "cloud_iot_auth_certificate.pem" to the GCS. This file was created by the sensor_hub application that runs on the device.



5. On your GCS, use the following command to create the device on the GCP side. This command needs the "cloud_iot_auth_certificate.pem" file, which has the public key generated by the device.

You will need the following information:
Project_id, region, registry_id, public-key file.

```
$gcloud beta iot devices create my-device-1 \
  --project=cloud-iot-175022 \
  --region=us-central1 \
  --registry=registry-iot \
  --public-key path=cloud_iot_auth_certificate.pem,type=rs256
```

6. You should be able to check the device added inside the registry-iot.

registry-iot
Region: **us-central1** Protocol: **MQTT** Pub/Sub topic: [projects/cloud-iot-175022/topics/device-events](#)
[View in Stackdriver](#)

Registered devices

[Add device](#)

Device ID	State
my-device-1	Enabled

13. Useful Links

<http://developer.android.com/tools/help/adb.html>