Cloud IoT With MCB1800 Starter Kit

Overview

In this document, you will be taught how to set up the hardware and software as well as using a way to connect to the MCB1857 board. Then, you will be shown step by step on how to build an end-to-end IoT solution using MCB1857 board with the Google Cloud Platform.

Audience

This document is intended for developers who will be working on connecting the MCB1857 board to the Google Cloud through the Internet using IoT solution.

Introduction to IoT

IoT is the network of physical devices, vehicles, cameras, alarms, home appliances and other items embedded with electronics, software, sensors, actuators, and network connectivity which enables these objects to connect and exchange data through the Internet.

Architectural diagram

IoT Solution with MCB1800 board

In this development, we use MDK v5 to build an image file and then flash it to the MCB1857 board. The image file is created using C language. Every time the board starts up, it will run this program from memory. The program basically send data to the Google Cloud Platform.

Components

1. MCB1800 Starter Kit
2. Google Cloud Platform

Software Requirements

Windows Operating System

Install one of the Windows OS below. The IDE currently only runs in these Windows Operating Systems:

- Microsoft Windows XP
- Microsoft Windows Vista
- Microsoft Windows 7
- Microsoft Windows 8

Tools

To compile, link and run applications on the MCB1800 or MCB4300 Evaluation Board, you must install an MDK-ARM IDE with the **MDK-Professional** edition. This IDE also debug scripts and upload scripts to the board. The link to download or purchase the IDE is available at http://www2.keil.com/mdk5/selector.

Hardware Requirements

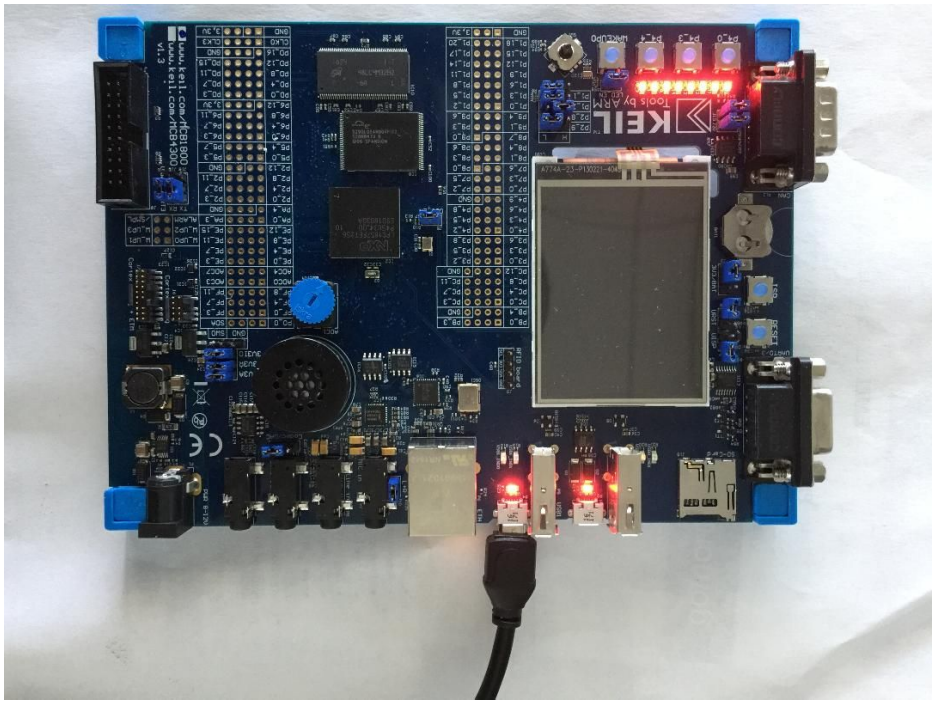
Need one of the following adapters to upload and debug scripts running on the MCB1800 board:

- A Keil ULINK2 USB-JTAG Adapter.
- A Keil ULINK-ME USB-JTAG Adapter.
- A Keil ULINKPro USB-JTAG Adapter.

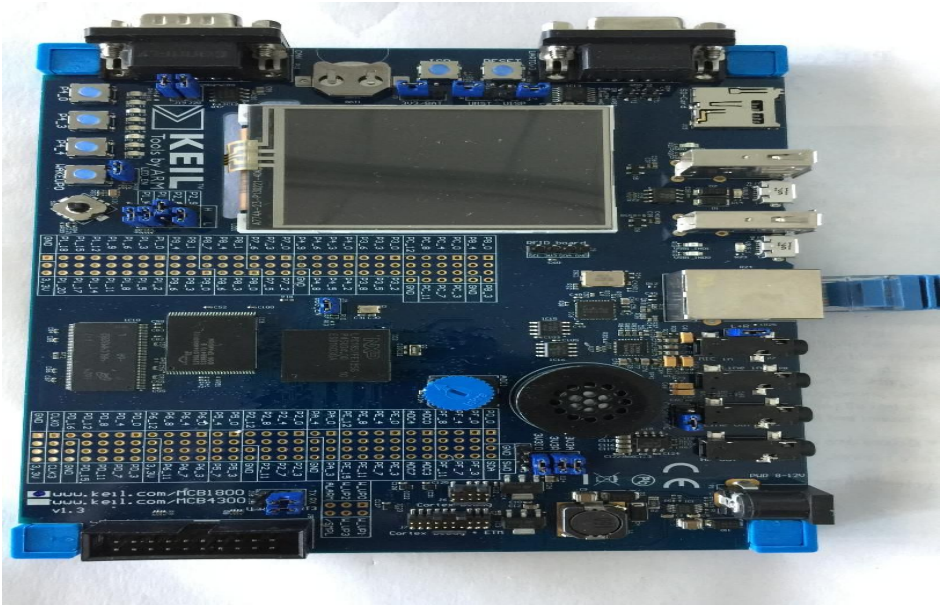
Set Up Hardware

1) Connect power to the MCB1800 board

To power up the board, connect your PC to the board using the included USB A to Micro B cable as shown below.

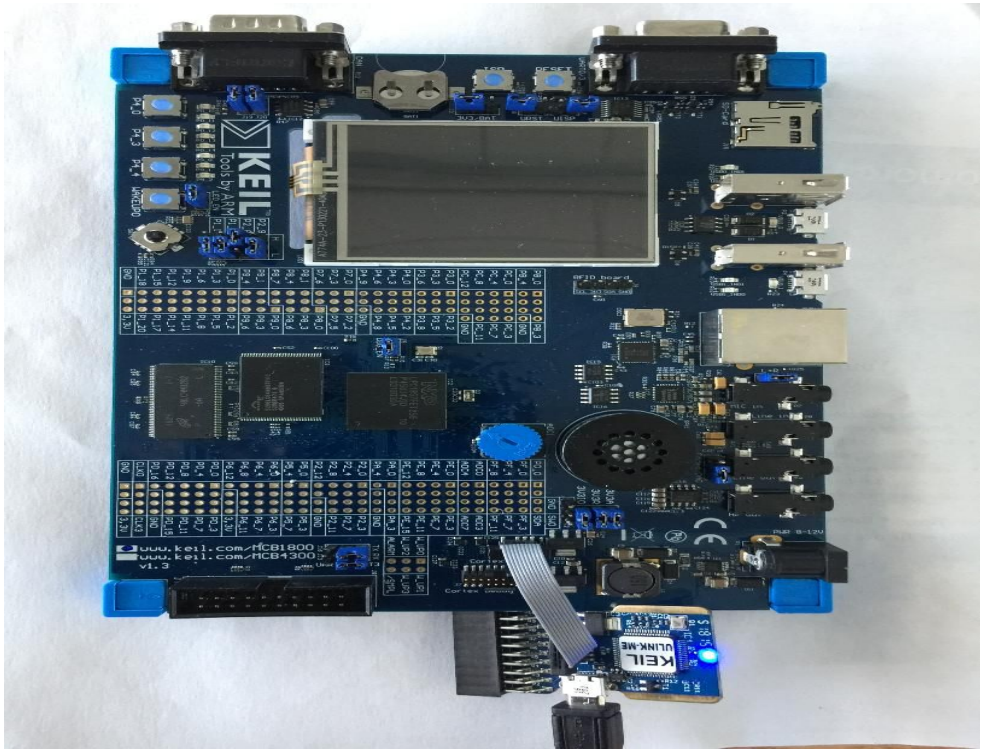


2) Connect Ethernet cable to the Board.



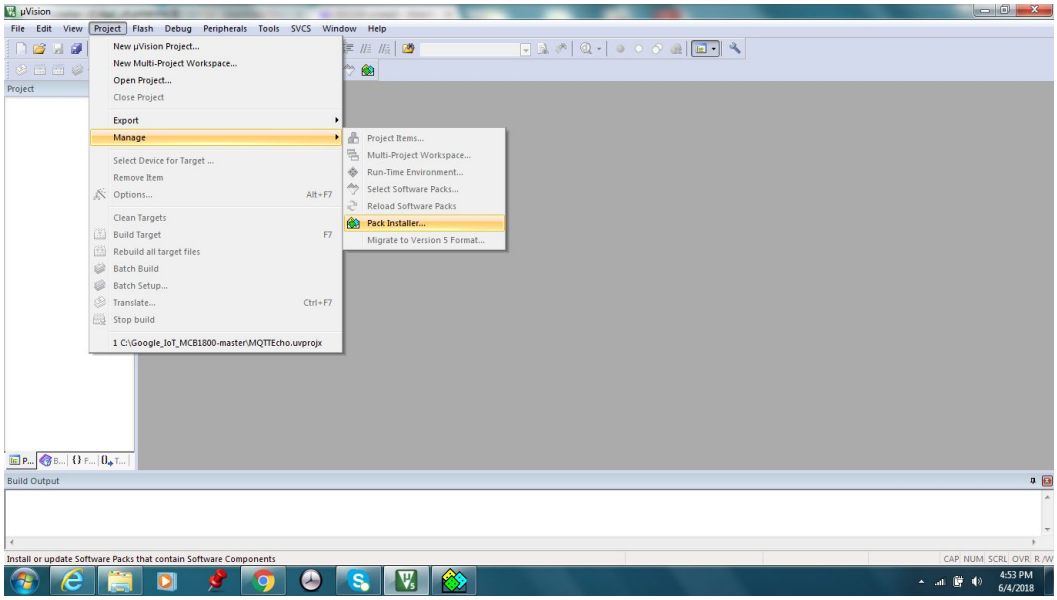
3) Connect ULINK-ME adapter to MCB1800 board for debug and program uploads.

1) Connect the ULINK-ME adapter to the Cortex Debug connector of the MCB1800 board as shown below.
2) Connect your PC to the ULINK-ME adapter using the included USB A to Micro B cable as shown below.

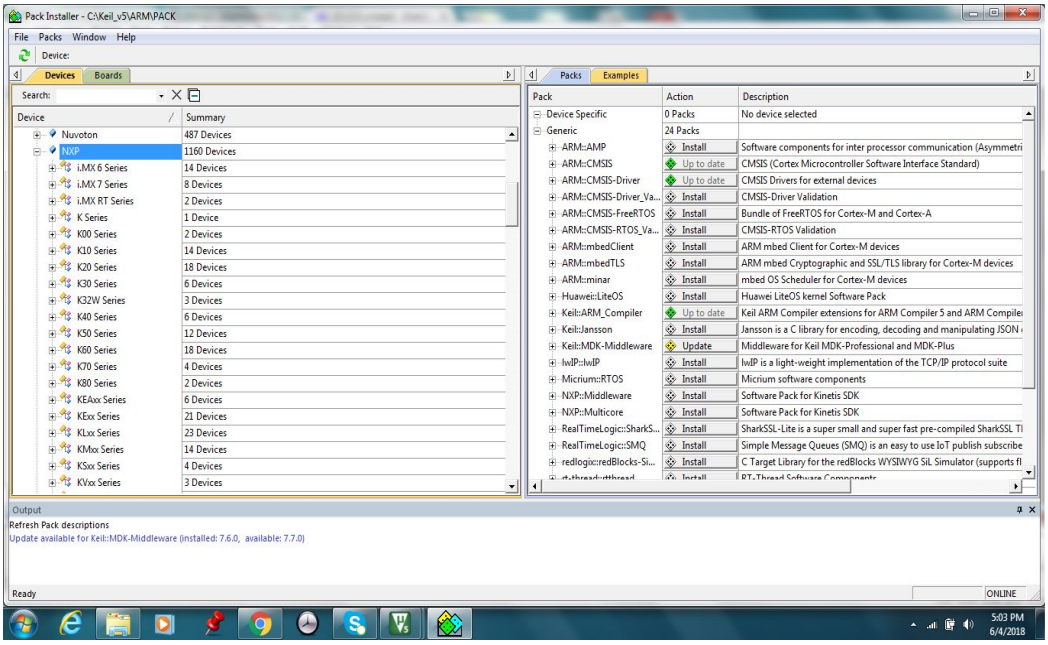


Install Device Driver

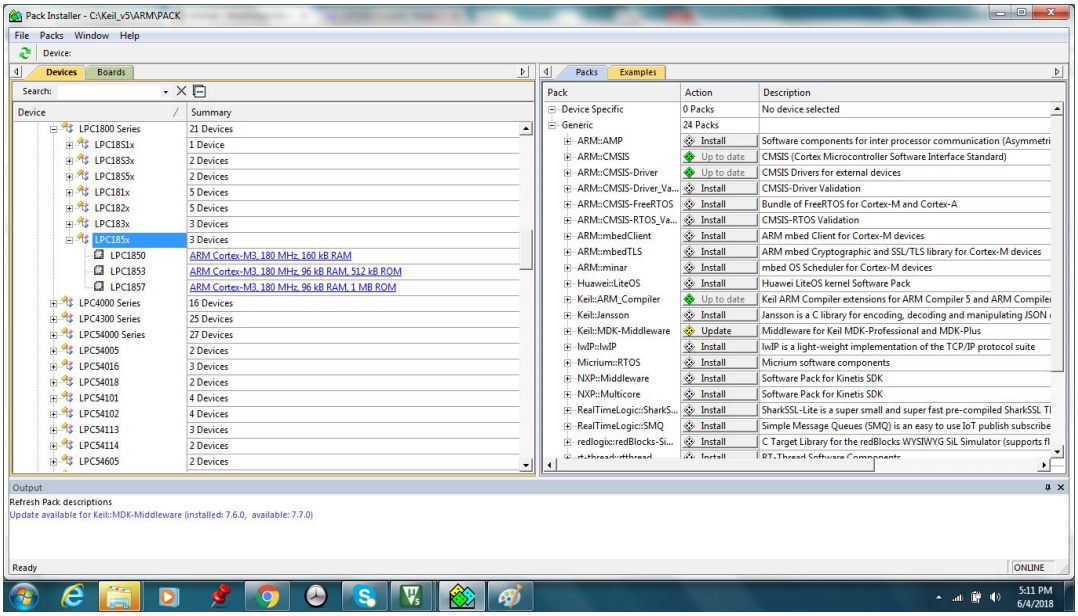
1) Open up the Keil uVision5 and click on “Project”. On the dropdown menu, click on “Manage” and then “Pack Installer”. A Pack Installer will open up new window.



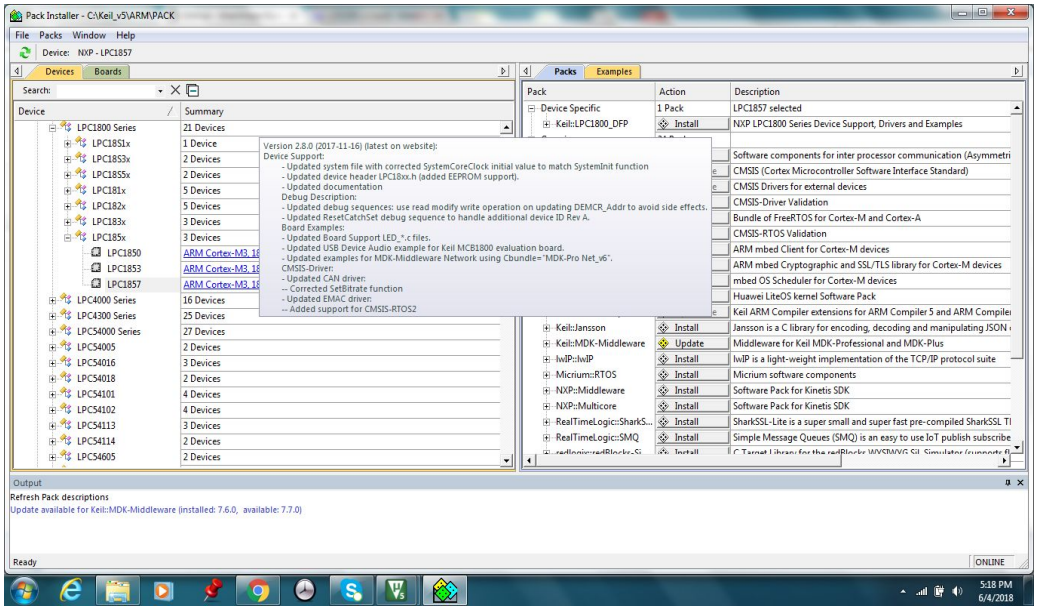
2) On the Pack Installer window, scroll down to NXP and expand it.



3) Scroll down to LPC1800 Series and expand it. Then scroll down to LPC185x and expand it.



4) Choose the LPC185 device. Once the device is selected, it is shown on the right hand side window under “Device Specific”. Now you can click on “Install” to install the device driver. Once the installation is complete, the “Install” text will be changed to “Up to date” text.



Cloud Side Setup

Before setting up the project, an account with Google is needed to log into the Google Cloud Platform. Please register an account with Google if you don’t have it. Once an account is created successfully, log into the Google Cloud Platform and create a public and private key pair, a Pub/Sub topic, a Pub/Sub subscription, a device registry, and a device.

Generate RSA Public and Private keys

Log into a box that can run shell scrips and use the commands below to create the private and self-signed public key and put the the keys in the folder “PemFiles”.

```
mkdir PemFiles
cd PemFiles
openssl req -x509 -newkey rsa:2048 -keyout rsa_private.pem -nodes -out rsa_cert.pem -subj "/CN=unused"
```

Create a Topic

1) Go to the Google Cloud Pub/Sub topics page in the GCP Console.
2) Click Create a topic.
3) Enter a unique Name for your topic.

Create a Subscription

1) Go to the Google Cloud Pub/Sub Subscriptions page in the GCP Console
2) Click New subscription.
3) Type a name for the subscription. Check the Pull box on the delivery type.
4) Click Create.

Create a Device Registry

1) Go to the Device registries page in GCP Console.
2) At the top of the page, click Create device registry.
3) Enter a Registry ID and select a cloud region.

Select both MQTT and HTTP protocols that devices in this registry will use to connect to Cloud IoT Core.

4) Select a Telemetry topic or create a new one. All device telemetry (the event data sent from devices to the cloud) will be published to the Cloud Pub/Sub topic you specify in this field.

Selecting the Device state topic or create a new one is optional. You can leave this one out.

Click Create to continue.

Create a Device

1) On the Registry Details page, click Add device.

2) Enter my-device for the Device ID.

3) Select Allow for Device communication.

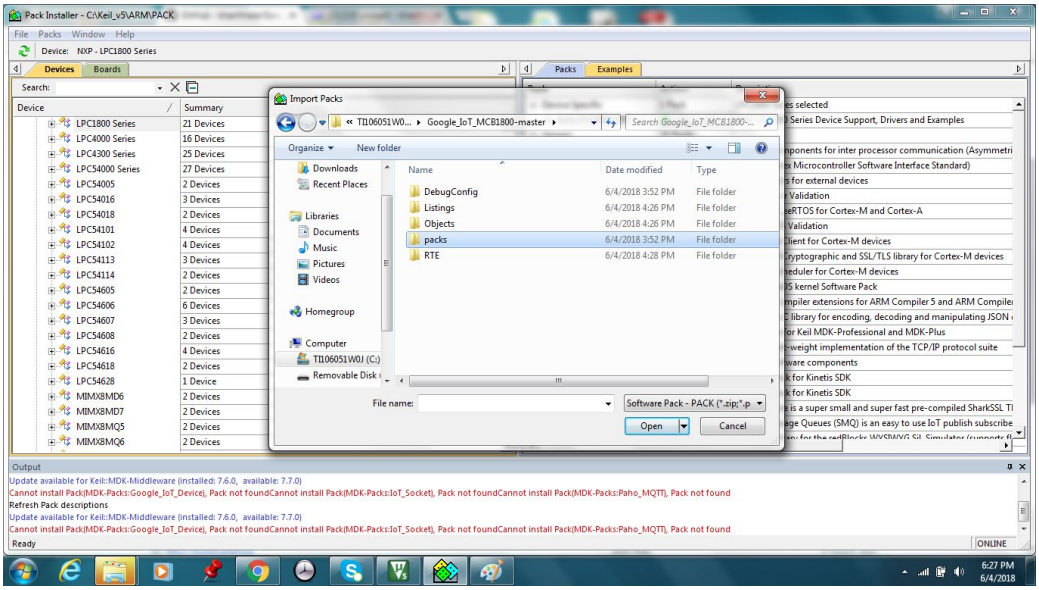
4) On the Authentication section, click Add and select public key format RS256_X509. Copy the data from the public key file rsa_cert.pem in the PemFiles folder and paste onto the **Public key value** box.

5) Click Add to complete.

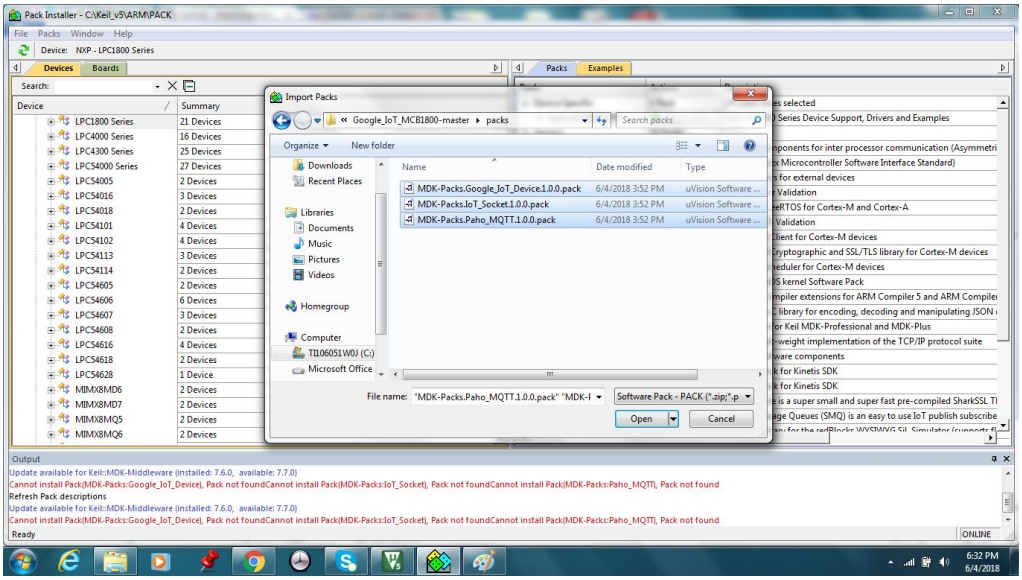
Set up Cloud IOT Project (Step by Step)

1) Go to https://github.com/khanhhale/Google_IoT_MCB1800 and download the project in a zip file. Open the zip file and extract the folder Google_IoT_MCB1800-master to C: drive.
2) Install additional drivers from projects.

    2.1 - Switch to Pack Installer window, and click on “File” and then Import. Navigate to Google_IoT_MCB1800-master folder and choose subfolder “packs”to open it.

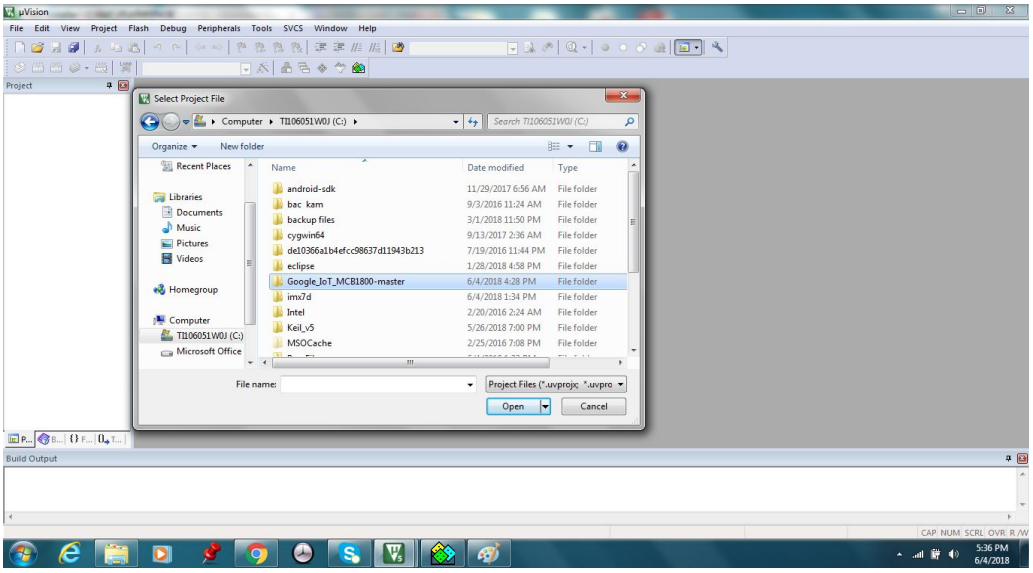


2.2 - While inside the sub folder packs, hold down the shift key and select all the files with the mouse click until all the files are highlighted as shown. Then release the shift key and hit open. This will import the drivers from the project.

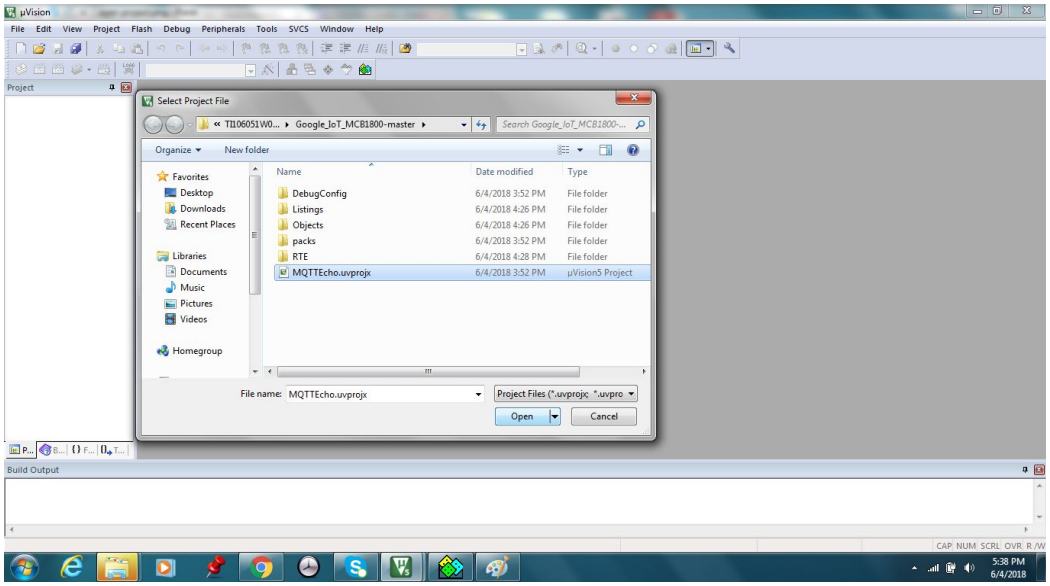


3) Open up the Keil uVision5 and click on “Project”. On the dropdown menu, click on “Open Project”. On the popup window, navigate to the Google_IoT_MCB1800-master folder and open it.

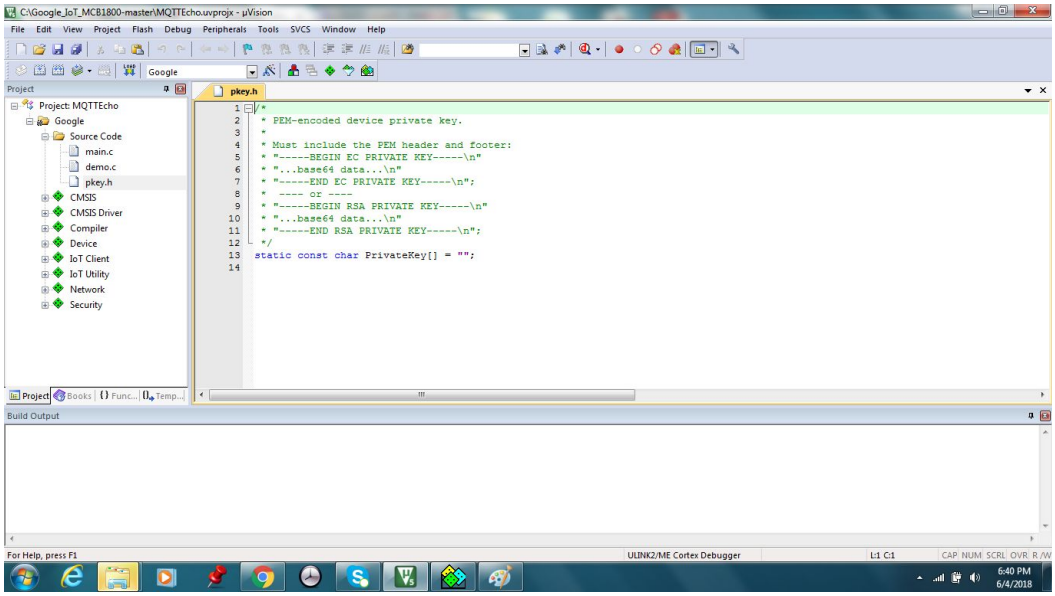
4) Once inside the folder, click on MQTTEcho.uvprojx and choose open to open the project.



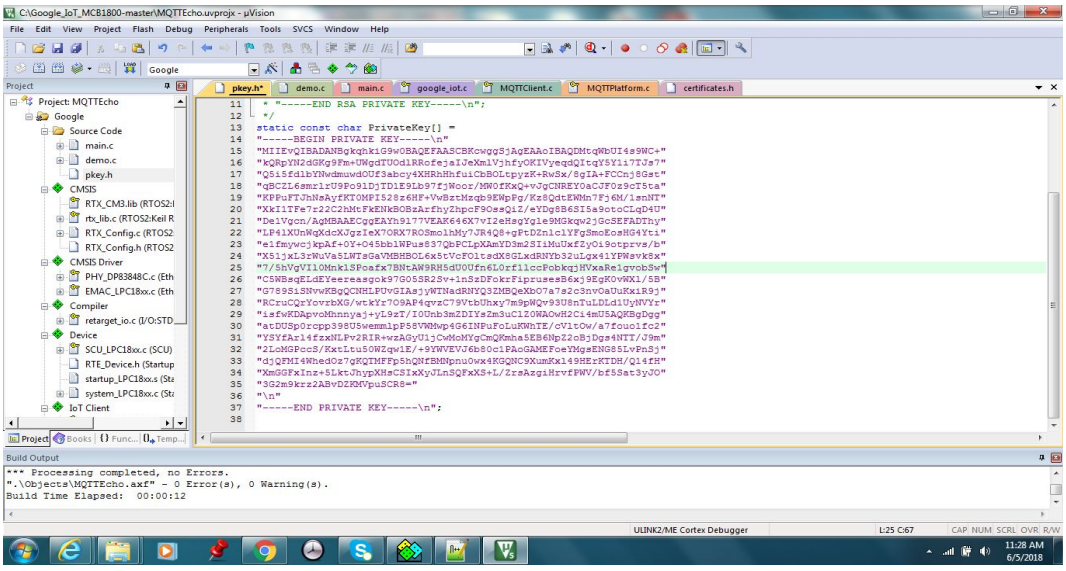
5) Once inside the folder, click on MQTTEcho.uvprojx and choose open to open the project.



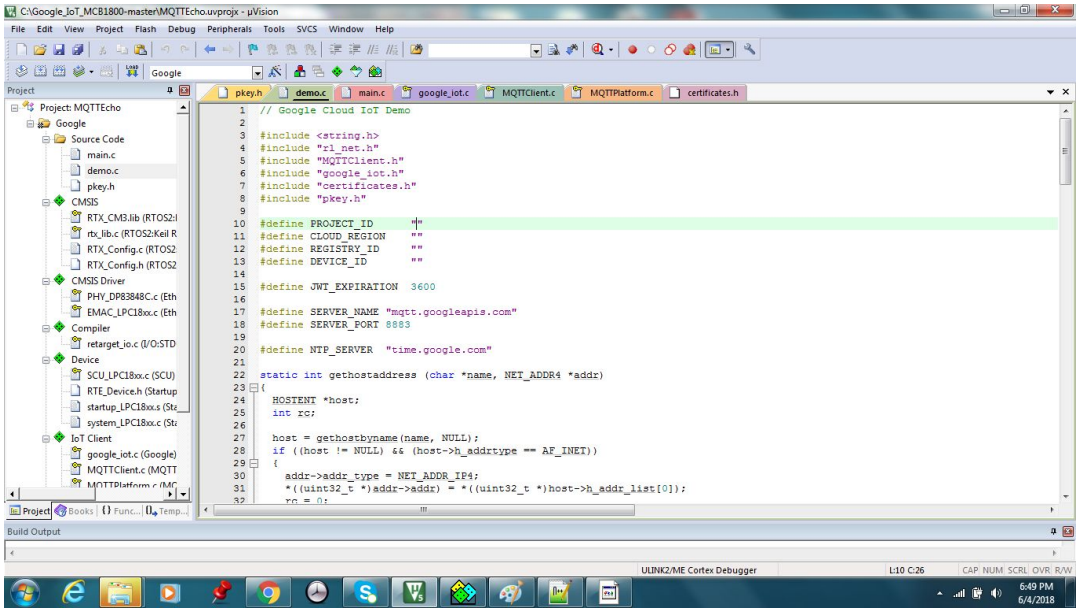
6) After the project is loaded into the IDE, expand the Google and then Source Code. Choose pkey.h and double click to open it for editing.



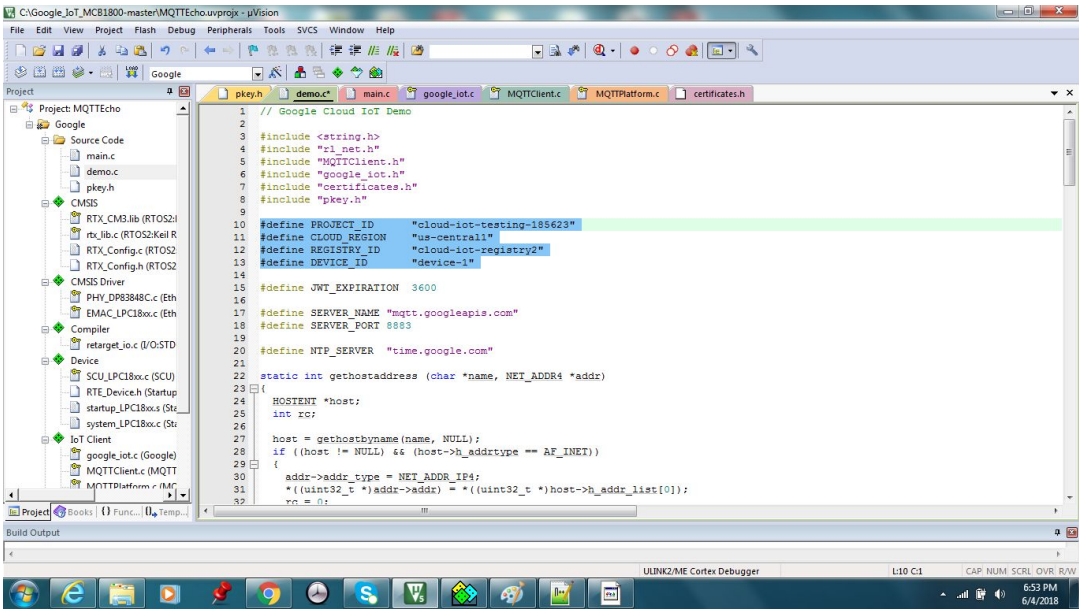
7) Take the content of the private key file rsa_private.pem in the PemFiles folder created in the "Cloud Side Setup" section and store it in the PrivateKey character array in the pkey.h file as shown below.



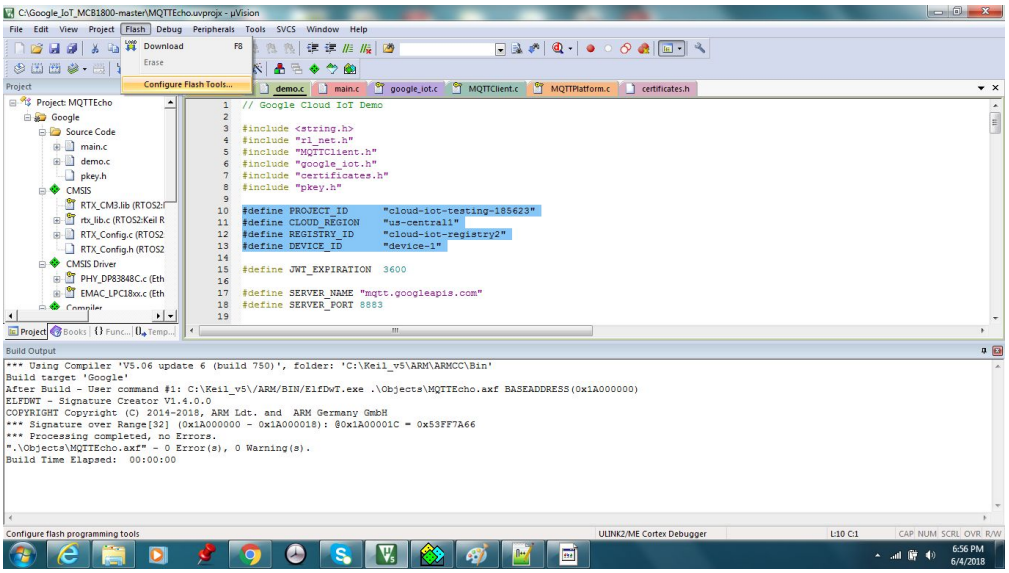
8) Under the Source Code folder, open the file demo.c for editing.



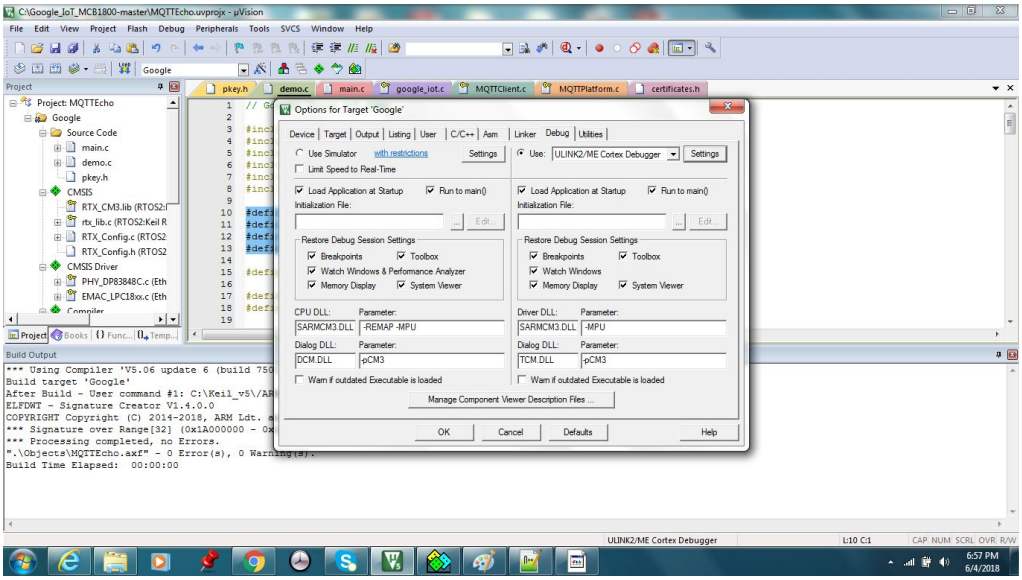
9) Fill in the value for constant variables PROJECT_ID, CLOUD_REGION, REGISTRY_ID and DEVICE_ID.



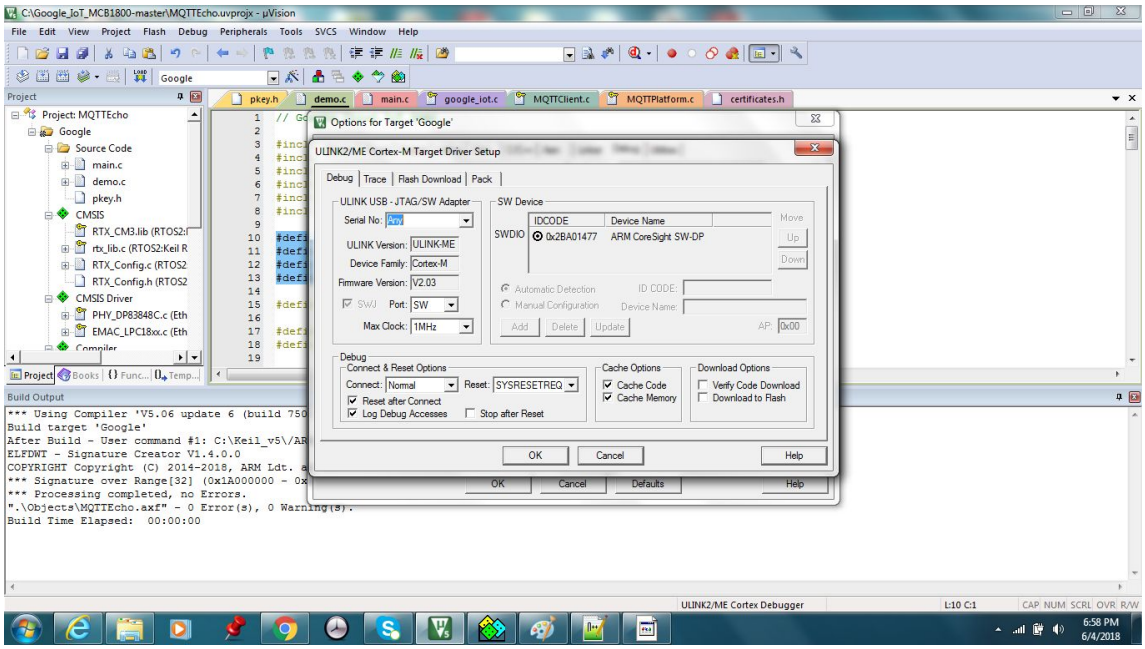
10) Before building the project and flash it to the board, we need to configure the flash tool. On the IDE, click on Flash. On the dropdown menu, click on “Configure Flash Tool”



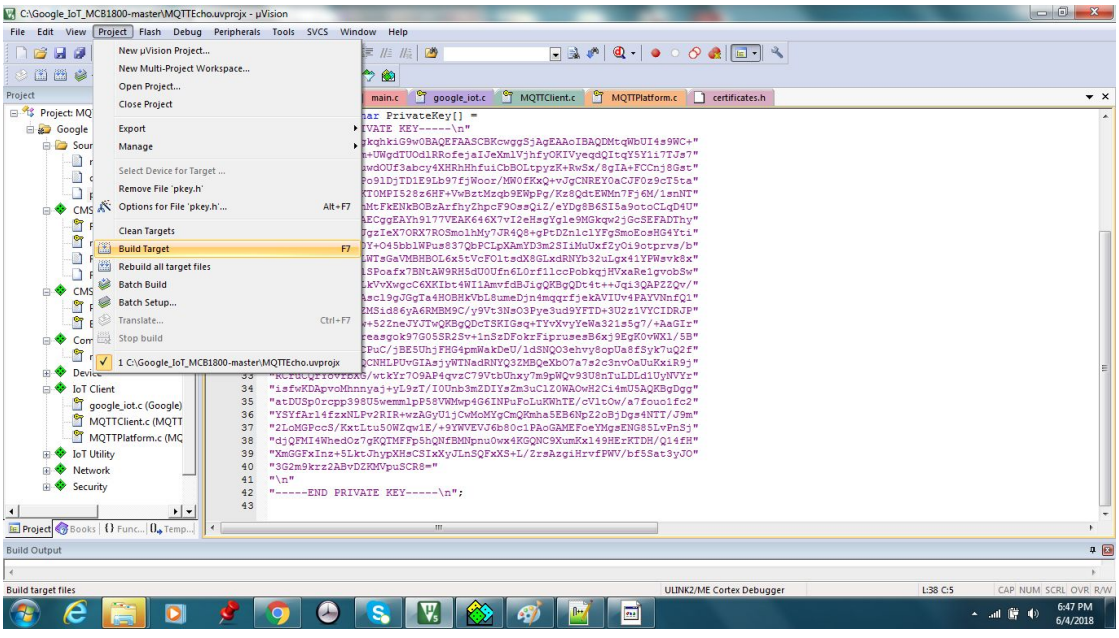
11) On the floating window, click on “Debug” tab to make sure the ULINK2/ME Cortex Debugger is set and the other options are set as shown below. Click on “Settings” button sitting right next to the ULINK2/ME Cortex Debugger text.



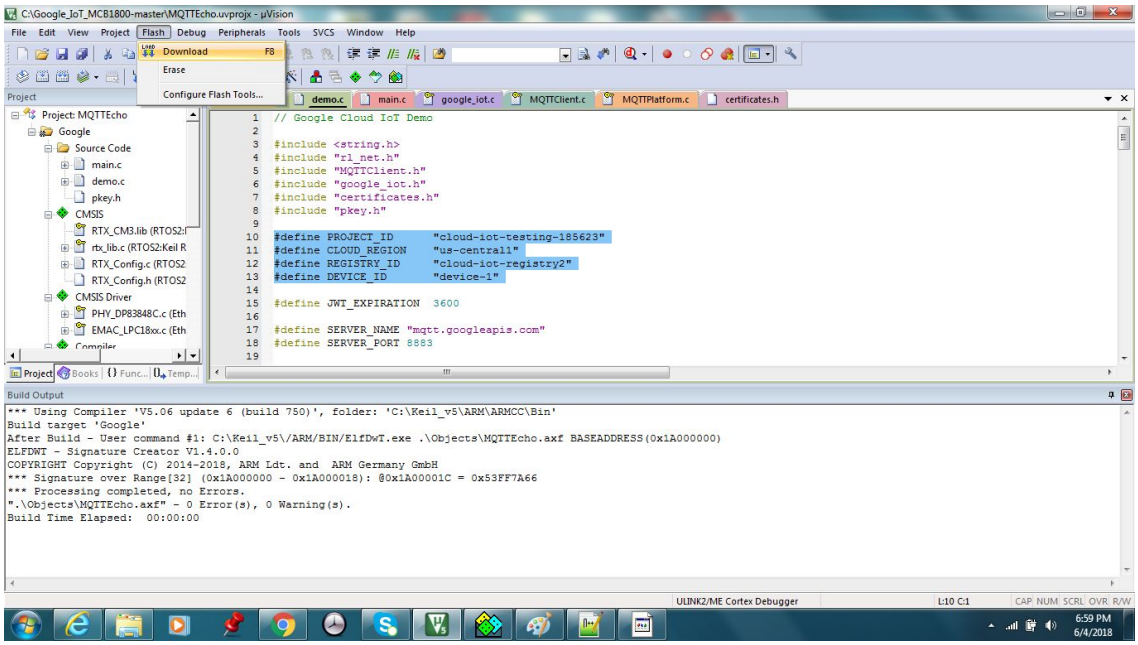
12) On the next window, checkmark “Log Debug Accesses” to show debugging information on bottom of the IDE. Also set the value of Max Clock to 1MHz as shown below.



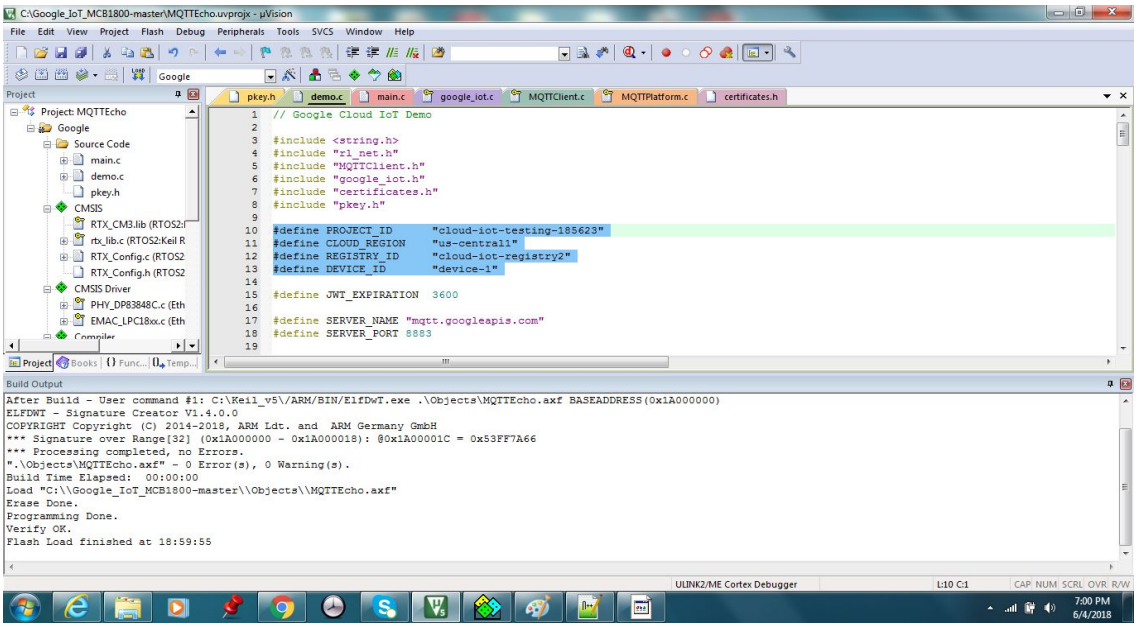
13) Next, click on “Project” and scroll down to choose “Build Target” to build the project. When building is complete, make sure there is no error shown.



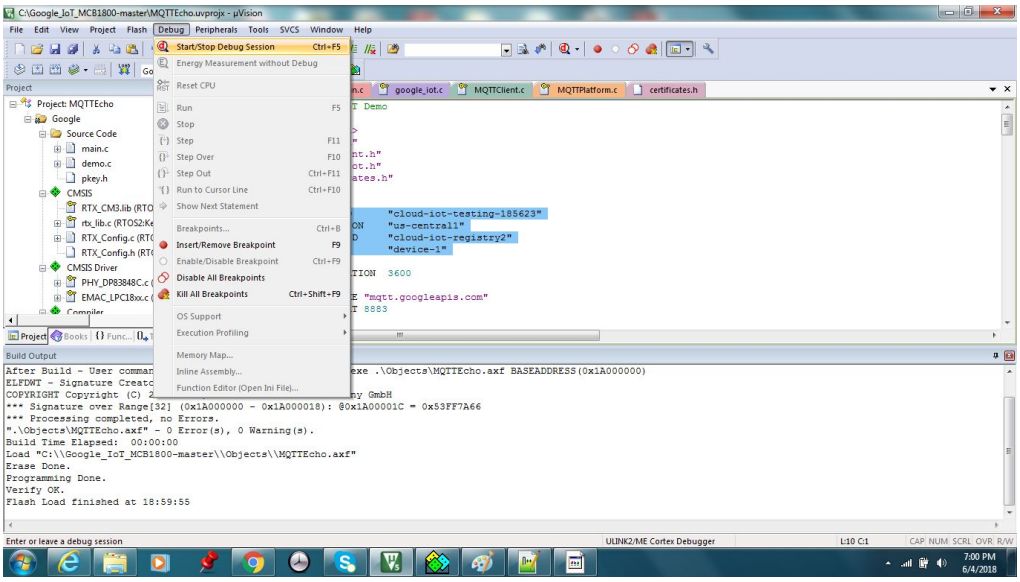
14) After the project is built successfully, we can flash the program to the MCB1800 board. To do this, go to “Flash” and click on Download.



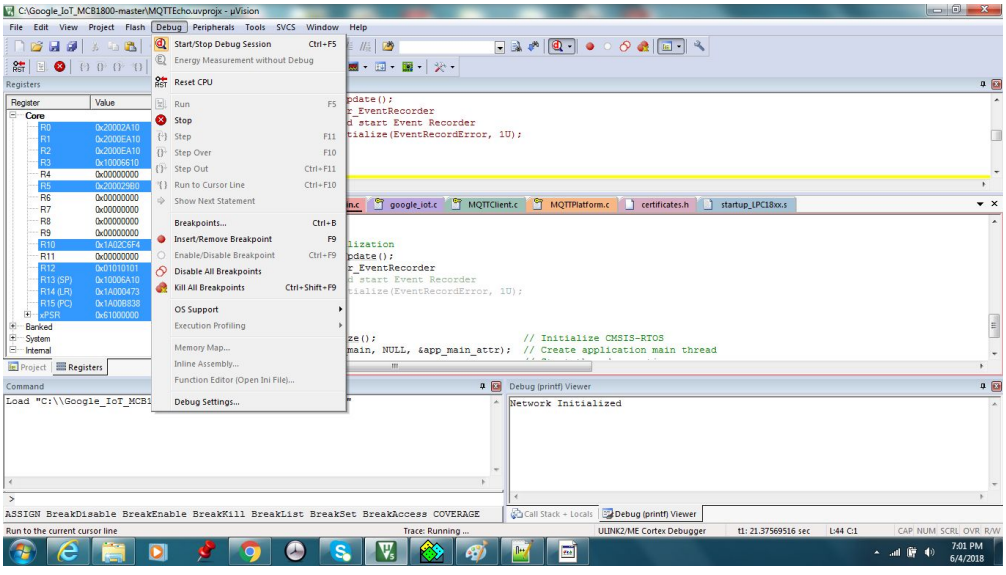
15) When flashing is done successfully, the screen below should show “Programming Done”



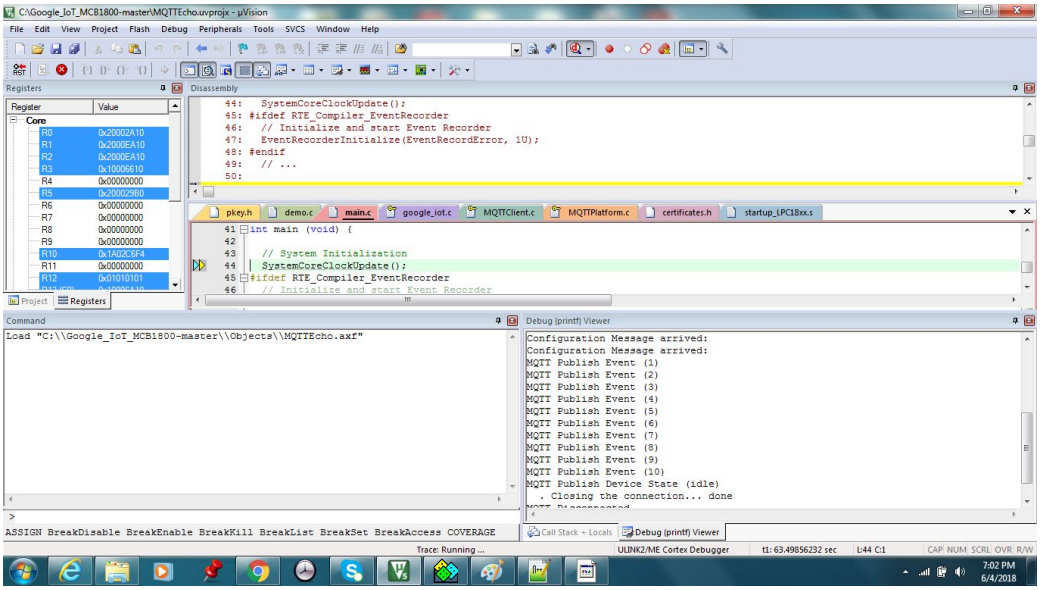
16) Next we need to start up debug so that we can see the outputs of the running program. For this to happen, click on “Debug” and choose “Start/Stop Debug Session” on the dropdown menu.



17) Once in debug mode, you can click “Run” to let the program finish.



18) The program outputs will then be shown as below. Successful message publishing should be shown as below.



19) Now run the command line below to verify data messages that were sent to the pub/sub.

```
gcloud beta pubsub subscriptions pull --limit=100 --auto-ack
subscription name
```