**Lab 6**

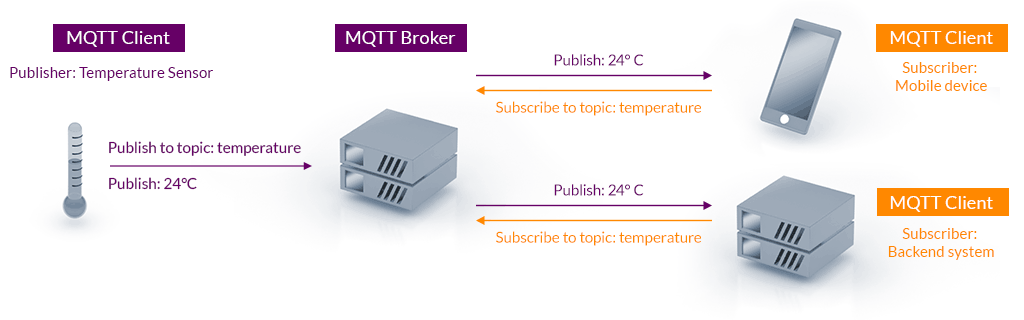
**Set up – Google Core IoT Big Architecture - OpenSSL**

Starting this week, you will be exploring many different topics and will experiment with different tools.

Two general activities:

1. Create a GitHub account, if you don’t have one already. Post your Python project from Reflection 1, and that initiations your activity history on GitHub. May employers investigate your GitHub activity history during the interview process. After you are done with your Arduino project, each of you will post your code and documentation on GitHub and that’s how you will submit the work to get credit for it.
2. As you all have different levels of experiences and use different operating systems, we’ll create a document to share findings. As a class, we’ll post notes on this document to keep track of issues and share solutions, [432-GoogleCoreIoTProject-DIY](https://docs.google.com/document/d/1JDT1MHQHD7TtXU9c4AfkbFcRq9VqjlWypHLVRUsbr3c/edit?usp=sharing), at the end of the project you can all download a copy of it and keep for future reference.
3. Visit Google Core IoT platform: <https://cloud.google.com/iot-core>
4. Create a Registry - for help, read the first part of this: <https://cloud.google.com/iot/docs/quickstart> or see the steps listed below:
   1. Click **Create registry.**
   2. Enter **SAD-IoT** for the **Registry ID**.
   3. If you're in the US, select **us-central1** for the **Region**. Or in Africa, select your preferred [region](https://cloud.google.com/iot/docs/requirements#cloud_regions).
   4. Select **MQTT** for the **Protocol**.
   5. In the **Default telemetry topic** dropdown list, select **Create a topic.**
   6. In the **Create a topic** dialog, enter **myDeviceEvents** in the **Name** field.
   7. Click **Create** in the **Create a topic** dialog.
   8. The **Device state topic** and **Certificate value** fields are optional, so leave them blank.
   9. Click **Create** on the Cloud IoT Core page.
5. Create device - for help, read the middle part of this: <https://cloud.google.com/iot/docs/quickstart> or see the steps listed below:
   1. On the **Registries** page, select **SAD-IoT** .
   2. Select the **Devices** tab and click **Create a device**.
   3. Enter myArduino (or myRaspberryPi) for the **Device ID**.
   4. Select **Allow** for **Device communication**.
   5. Add the public key information to the **Authentication** fields.
      1. Copy the contents of rsa\_cert.pem to the clipboard. Make sure to include the lines that say -----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----.
      2. Select **RS256\_X509** for the **Public key format**.
      3. Paste the public key in the **Public key value** box.
      4. Click **Add** to associate the RS256\_X509 key with the device.
   6. The **Device metadata** field is optional; leave it blank.
   7. Click **Create.**
6. **Note that we will need to have the private/public key to communicate with the cloud. You can use OpenSSL or Git Bash for this purpose, follow the following leads:**
7. Open SSL for Macbook: <https://www.openssl.org/source/>
8. Gift for Windows: <https://gitforwindows.org/>
   1. Try doing this for example: ssh-keygen -t rsa
9. Arduino SD Shield setup guide: <https://www.arduino.cc/en/Guide/ArduinoWirelessShield#configuring-the-xbee-module>
10. Arduino IDE setup: Tool 🡪Manage Libraries 🡪 search for Google Cloud IoT JWT

|  |  |
| --- | --- |
| **Terms** | **Description (taken from Google, MQTT.org, ssl.com)** |
| IoT | Internet of things |
| Device (thing) | A device is a "Thing" in "Internet of Things": a processing unit that is capable of connecting to the internet (directly or indirectly) and exchanging data with the cloud. |
| Registry | A device registry is a container of devices with shared properties |
| Gateways | A gateway is a device that connects its client devices to Cloud IoT Core and performs several tasks on their behalf, such as: ... authenticating to Cloud IoT Core when the device can't send its own credentials, or when you want to add a layer of security by using the credentials of both the device and the gateway. |
| MQTT | **Message Queuing Telemetry Transport**  MQTT is an OASIS standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth. See more info on [MQTT.org](https://mqtt.org/) |
| Events | Cloud IoT Core can optionally send device activity logs to Cloud Logging. Device activity logs include information such as device connections and errors. These are called events. |
| Pub/sub | See Figure 1 |
| Topic | A named resource to which messages are sent by publishers. |
| SSL | SSL Stands for secure sockets layer. Protocol for web browsers and servers that allows for the authentication, encryption and decryption of data sent over the Internet. The protocol uses a pair of keys – one private & one public, to authenticate, secure, and manage secure connections. |
| Private Key | The private key is used to digitally sign your Certificate Signing Request (CSR), and later to secure and verify connections to your server.  Your private key should be closely guarded, since anyone with access to it can readily break your encryption. (Note again that the private key is just a text file – however, it’s a really important text file and should be protected accordingly.) |
| Public Key | The public key, by contrast, is distributed as widely as possible – it’s included as part of your SSL certificate. Anyone with access to your public key can verify that your message is authentic without having to know your secret private key. |



**Figure 1:MQTT Publish / Subscribe Architecture**

|  |  |
| --- | --- |
| **Points** | **Requirement** |
| 10 | Setup on the cloud complete |
| 5 | Private/public key set is made |
| 5 | A simple project with Arduino has been completed |
| **20** |  |

**Important note about USB vs Micro switch**

You can configure the Xbee module from code running on the Arduino board or from software on the computer. To configure it from the Arduino board, you'll need to have the switch in the Micro position. To configure it from the computer, you'll need to have the switch in the USB position and have removed the microncontroller from your Arduino board.

Other pointers:

1. <http://nilhcem.com/iot/cloud-iot-core-with-the-esp32-and-arduino>
2. [https://ludovic-emo. medium.com/how-to-send-esp32-telemetry-to-google-cloud-iot-core-caf1a952020d](https://ludovic-emo.medium.com/how-to-send-esp32-telemetry-to-google-cloud-iot-core-caf1a952020d)
3. <https://www.arduino.cc/en/Guide/ArduinoWirelessShield>
4. <https://github.com/GoogleCloudPlatform/google-cloud-iot-arduino>

