Initial Boot

No Internet Connection:

Use a Soft AP Captive Portal webpage that allows a user to connect to a wireless network. Provide a list of SSID or enter SSID manually. Provide input for network password.

Save WiFi connect data locally. Save between device restarts.

Cellular Module Available:

If a Cellular Module is present connect via Cellular Network

Device Provisioning

After Initial Boot:

Internet is available (Cell/Wifi) go to a 'boot url' to get **initial configuration** info.

URL access will require a JWT (https://jwt.io/) signed by a private key (provided by me)

```
{
  "sub": "UniqueDeviceId",
  "exp": 1649657630,
  "iat": 1648657630
}
```

The boot URL will provide config for the device.

Example:


```
{
   "id": "2538333638445895",
   "aud": "",
   "dataFreq": "250",
   "saveMinFreq": "4000",
   "minAmps": "0.5",
   "minRpm": "1.0",
   "mqttHost": "",
   "mqttPort": "",
   "mqttClientId": "",
   "mqttPublish": "",
   "mqttHost": "mqtt.googleapis.com",
   "mqttPort": "8883",
```

```
    "mqttId": "",
    "endPoint": "",
    "publishURI": "",
    "setStateURI": "",
    "configURI": "",
    "cert": "private key"
}
```

The Configuration is saved locally and is persisted between power down / power on.

Device Operation

Note: IOT Back End is Google Cloud IOT

1) The device will use **MQTT** to communicate
   https://cloud.google.com/iot/docs/how-tos/mqtt-bridge

2) Communication requires JWT. The cert is provided in Boot Config (#4)
   See:
   https://cloud.google.com/iot/docs/how-tos/credentials/jwts?authuser=1#required_claims

3) The device config will include the private key required to sign JWT See:
   https://cloud.google.com/iot/docs/how-tos/mqtt-bridge#iot-core-mqtt-auth-run-cpp

4) The device will take inputs from 4 sensors
   a) Button Switch (Open, Close, Stop)
   b) RPM Encoder (e38s6g5 rotary encoder)
   c) Current Detector (SCT-013-030)
   d) Level Sensor (LCH-A-S : Low Cost Single Axis Inclinometer 0-10V Output)
5) (The device can record data on configurable interval “dataFreq” (milliseconds)
6) The device records data when
   a) Button is pressed
   b) RPM > **minRpm** OR Current > **minAmps**
7) Device sends data as JSON array encoded as Base64 string (URL safe)

Sample of data
ID - the id of the device - from the boot config
Date- The UTC date time including fractional seconds
RPM - the RPM from RPM encoder
REV - the abount of revolutions that occurred in the dataFreq window
SUM - total REV
DIR - the direction
BTN - the button that was pressed
CUR - the current

```
[{  "ID": "22",  "DATE": "2022-03-27T18:42:32.2",  "RPM": "15.6",  "REV":
"0.07",  "SUM": "0.07",  "DIR": "2",  "BTN": "2",  "CUR": "33.842"},
{  "ID": "22",  "DATE": "2022-03-27T18:42:32.3",  "RPM": "26",  "REV": "0.11",
"SUM": "0.17",  "DIR": "2",  "BTN": "2",  "CUR": "10.072"},
{  "ID": "22",  "DATE": "2022-03-27T18:42:32.4",  "RPM": "26",  "REV": "0.11",
"SUM": "0.28",  "DIR": "2",  "BTN": "2",  "CUR": "9.975"},
{  "ID": "22",  "DATE": "2022-03-27T18:42:33.1",  "RPM": "26",  "REV": "0.11",
"SUM": "0.39",  "DIR": "2",  "BTN": "2",  "CUR": "9.922"},
{  "ID": "22",  "DATE": "2022-03-27T18:42:33.2",  "RPM": "24.4",  "REV": "0.1",
"SUM": "0.49",  "DIR": "2",  "BTN": "2",  "CUR": "9.886"}]
```


Recording the data

Device Upgrade

Provide ability to upgrade devices over the air: download compiled binary and update software.