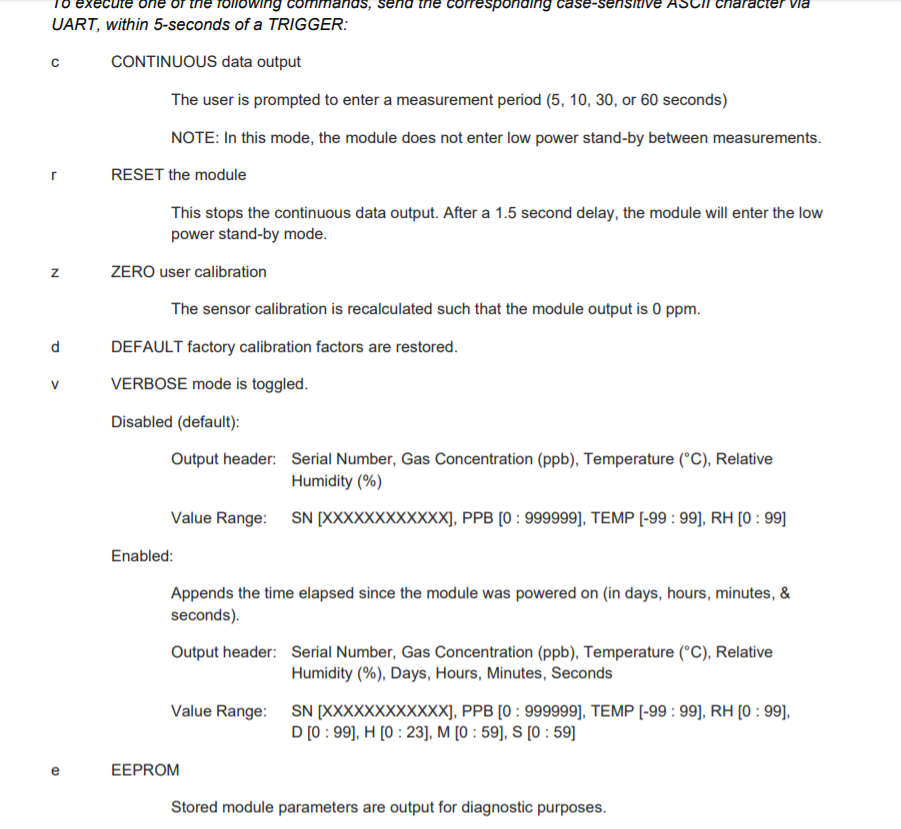
**Final Report-Step-By-Step to configure GPS and CO sensor to GCP**

**Module used:**

1. GPS module used in this project is <https://www.mikroe.com/gnss-4-click>. It works on UART communication and requires a GPS parsing algorithm to extract latitude and longitude.
2. CO Sensor is provided by Spec sensor(<https://www.spec-sensors.com/product/iot-co-1000-digital-co-sensor-module/>) which works on UART communication. We have various modes that can be configured to get the CO sensor data like Gas Concentration(PPB), humidity and temperature.
3. Below screenshot explains the characters to be sent over UART to get the sensor data as well as the output format (Reference datasheet:<https://www.spec-sensors.com/wp-content/uploads/2017/01/DG-SDK-968-045_9-6-17.pdf>)

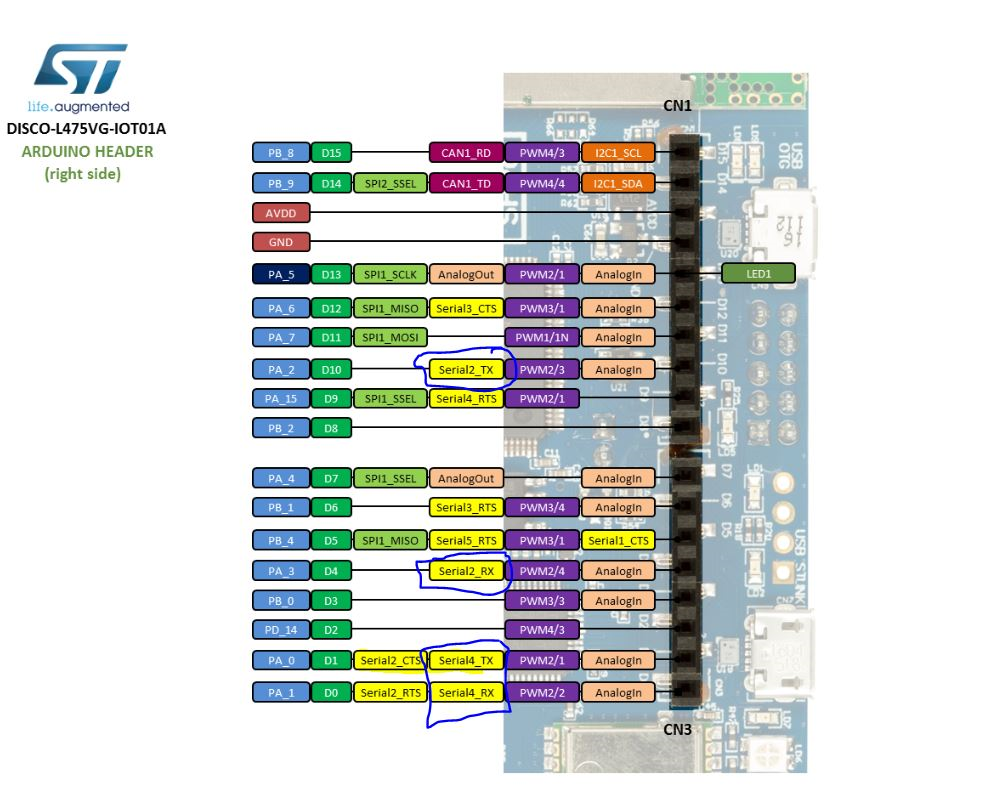


**About CO Sensor output:**

1. The gas concentration of CO Sensor ranges from **800PPB to 2000PPB** for normal environmental conditions, since the natural concentration of carbon monoxide in air is around **0.5 to 1 ppm**, and that amount is not harmful to humans.
2. CO sensors are normally very stable. They are cross sensitive to a couple of potential interfering gases, however.
3. **Usually negative readings** are the result of the CO sensor being fresh air adjusted while in the presence of a detectable interfering gas, or when the sensor is fresh air adjusted before it has completed recovering from a prior exposure to an interfering gas.
4. The CO sensor is always producing a signal as long as the instrument is turned on. In the presence of CO the signal goes up. In fresh air, you still get a signal, but the value is lower. When you fresh air zero the instrument you are telling the instrument to use the signal from the sensor at that moment as the point of comparison.

**Hardware connections:**

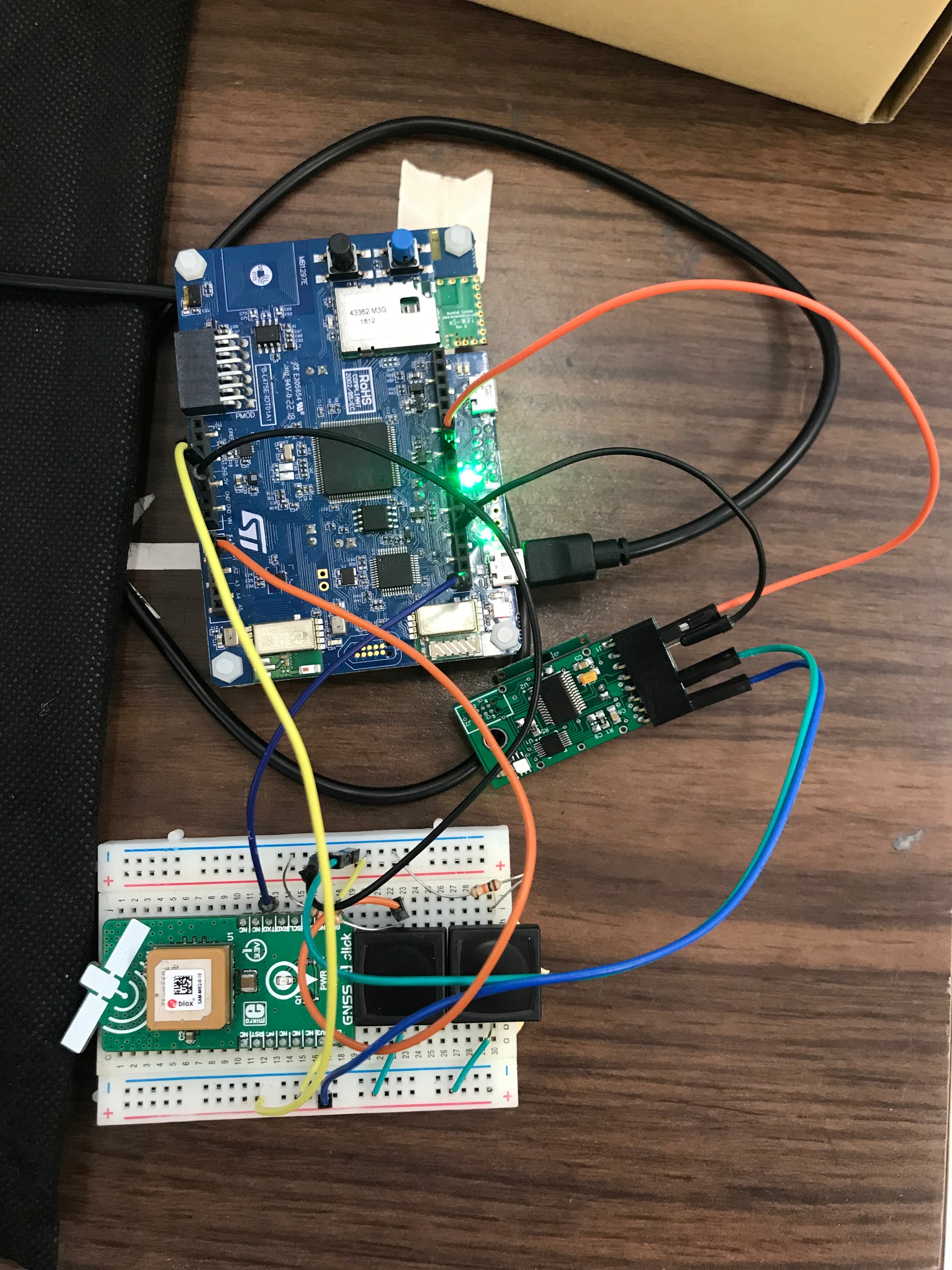
1. We use B-L475E-IOTA board to communicate with the sensors and publish the data over the google cloud. We will be using UART2 and UART4 pins of the board to connect to the sensors. Below are the pin connections:



1. The below table is the pin connection details from board to the sensors:

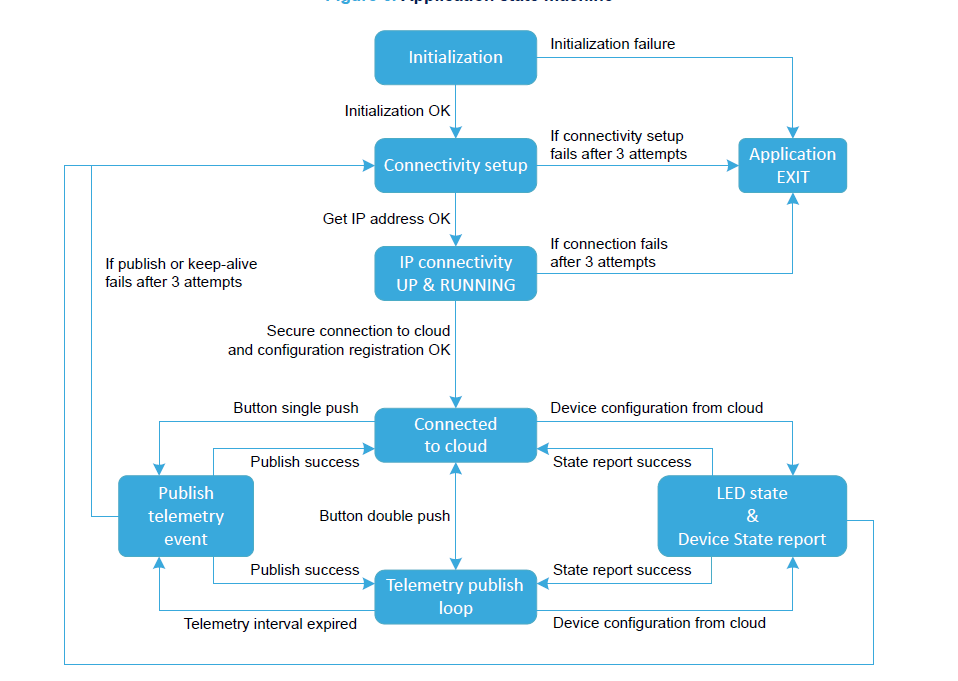
|  |  |
| --- | --- |
| **B-L475E-IOT01A pins** | **Sensor pins** |
| PA\_0 (D1) | ‘RXD’ pin of GPS Module |
| PA\_1 (D0) | ‘TXD’ pin of GPS Module |
| PA\_3(D4) | Pin 3(TX) of CO Sensor |
| PA\_2(D10) | Pin 2(RX) of CO Sensor |
| 3.3V | ‘3.3V’ pin of GPS module  Pin 8 of CO Sensor |
| GND | ‘GND’ pin of GPS Module  Pin 6 of CO Sensor |

1. The actual hardware connections look like this:



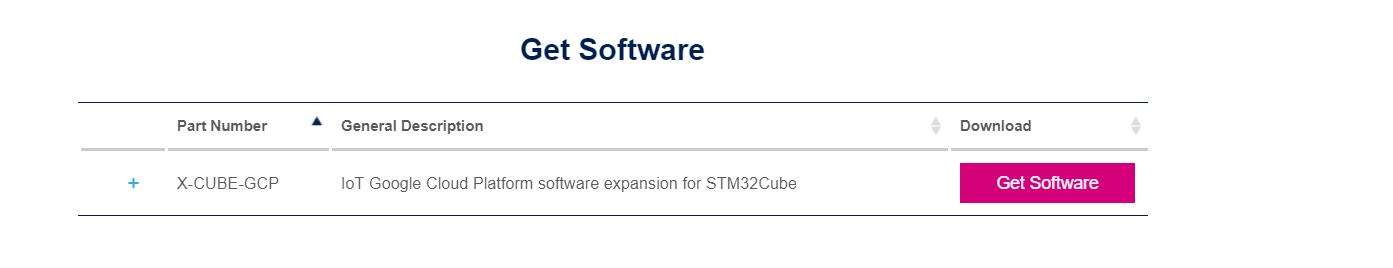
**Google Cloud Configuration:**

1. In order to publish the data to google cloud, we have to make a free google cloud account, register and enable the APIs. The steps are mentioned in section 5.2 of GCP User Manual (<https://www.st.com/content/ccc/resource/technical/document/user_manual/group1/e9/8a/9b/73/5c/ff/4d/10/DM00522079/files/DM00522079.pdf/jcr:content/translations/en.DM00522079.pdf>)
2. The Application flow for entire code shall be as follows:

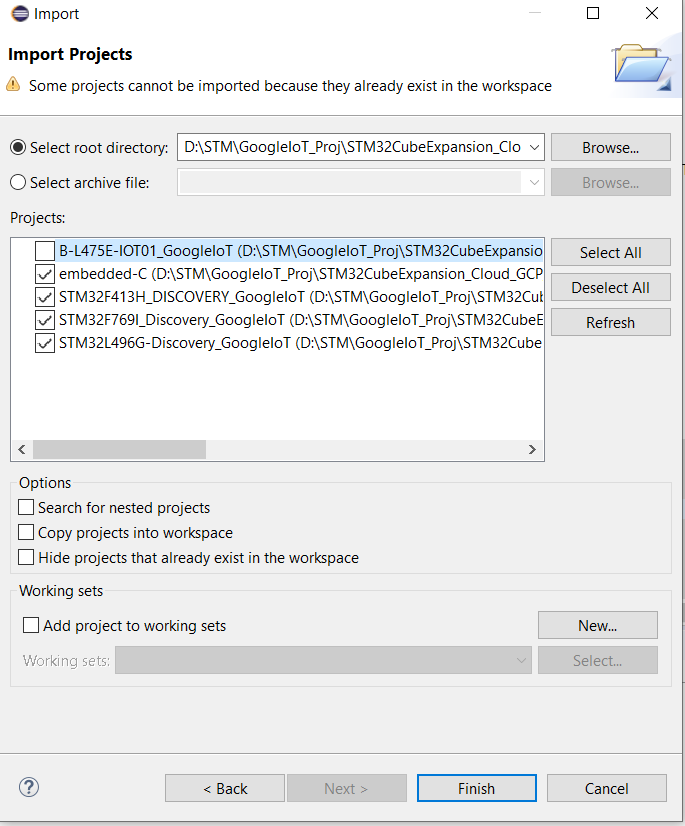


**X-CUBE-GCP integration with GPS and CO Sensor:**

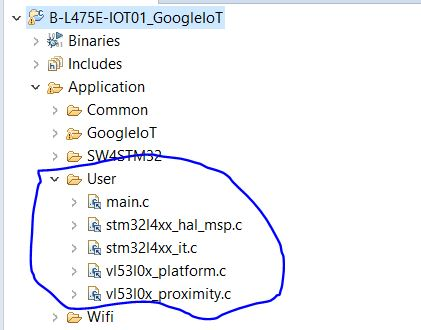
1. Download the X-CUBE-GCP package from ST website: <https://www.st.com/en/embedded-software/x-cube-gcp.html#get-software> and click on get software. (You need to register a new account in ST website if you don’t have a ST account to get the package)



1. Extract the project in the workspace. The folder name should be: STM32CubeExpansion\_Cloud\_GCP\_V1.0.0
2. Open the Ac6 System Workbench by STM. If not installed, you can get it here: <https://www.st.com/en/development-tools/sw4stm32.html>
3. Click on Import-->Existing project in workspace-->Select the folder: STM32CubeExpansion\_Cloud\_GCP\_V1.0.0 and click OK.
4. There will be multiple projects as follows. Deselect all and select: **B-L475E-IOT01\_GoogleIoT**



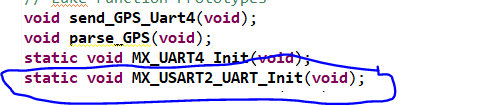
1. Go to Luke’s Github link: <https://github.com/lastjediluke/firefighterFinderGPS>. Go to Luke430Complete. Download the following files: main.c, stm32l4xx\_hal\_msp.c, stm32l4xx\_it.c. Copy these files and paste inside the Application-->User folder of the project in Workbench. Replace the existing files

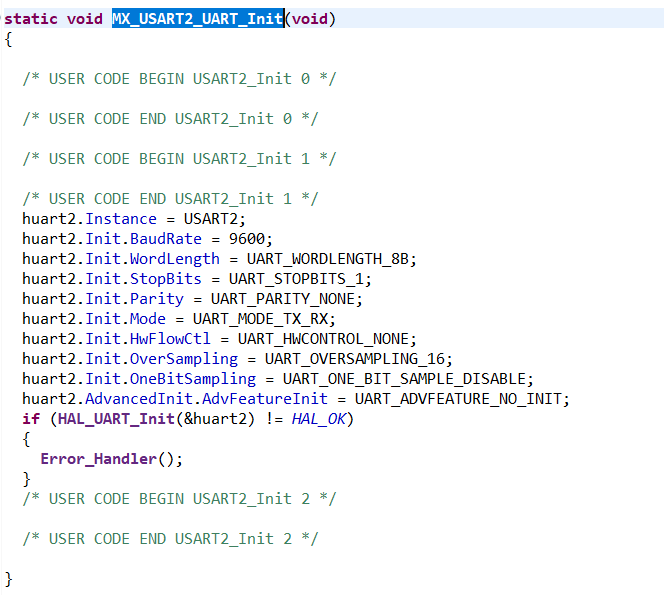


1. Download googleiot.c and replace inside the Application-->GoogleIot folder. Replace the existing files
2. Download the header files from the gitlink-->Luke430Complete-->main.h,stm32l4xx\_it.h and googleiot.h. Copy these files and paste inside the Inc folder (Path: ..\STM32CubeExpansion\_Cloud\_GCP\_V1.0.0\Projects\B-L475E-IOT01A\Applications\Cloud\GoogleIoT\Inc)
3. These files contains the UART4 configuration for GPS module.

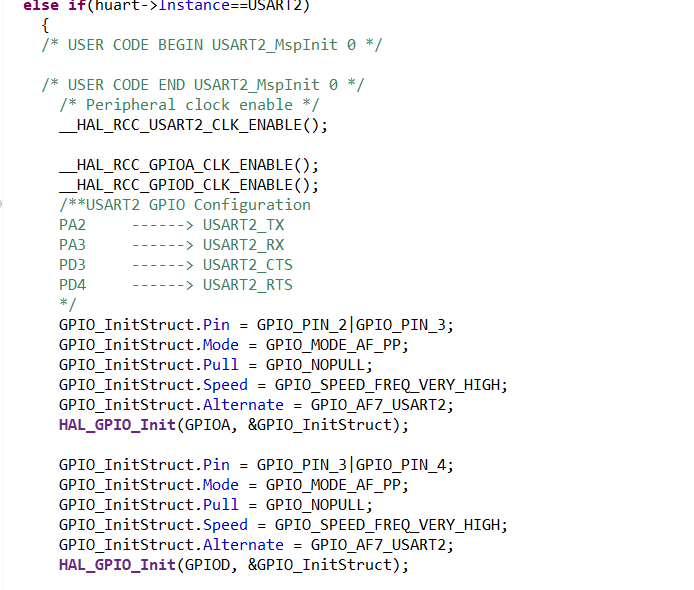
**Add the USART2 support in project:**

1. Add the init functions for USART2 in googleiot.c

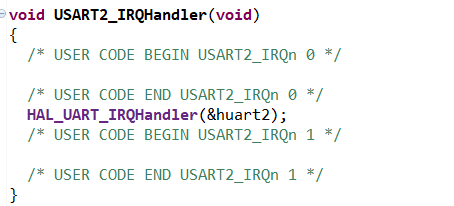




1. In stm32l4xx\_hal\_msp.c, add the USART2 pin configurations as follows:

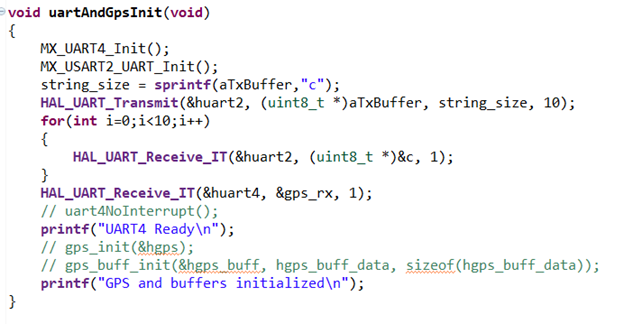


1. In stm32l4xx\_it.c, add the USART2 Interrupt handler function.

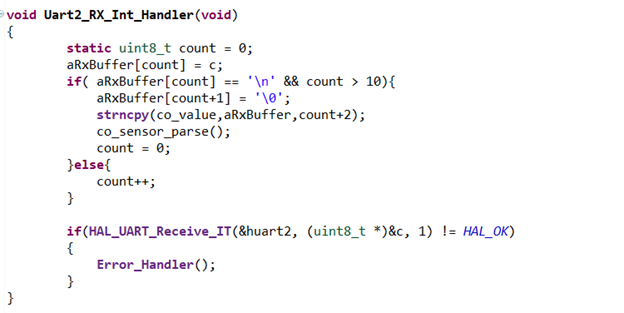


**X-CUBE-GCP Integration Contd….**

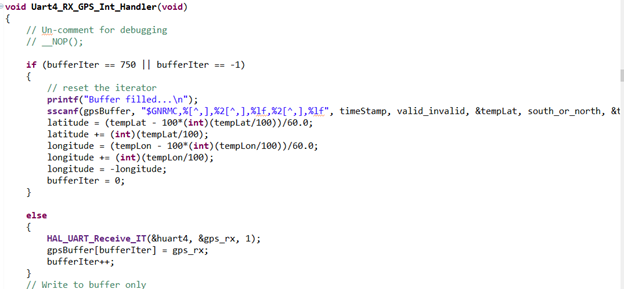
1. In googleiot.c, I added a init() which triggers UART2 for CO-Sensor and UART4 for GPS. I am calling the function before calling the gcp\_cloud function in main()



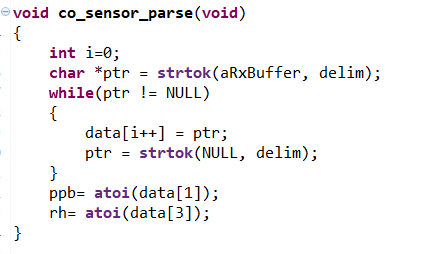
1. Once the gcp function starts, the UART2 and UART4 interrupts are triggered and UART\_RXCallback() was executed, where it started storing the sensor data as string.



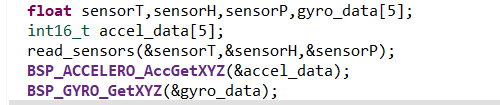
1. Since we are sending the data over JSON file, we had to parse GPS and CO-Sensor data. Worked with GCP code to parse GPS data. Commented parsing algorithm didn't work so I created my own parsing algorithm. Parsed lat and long successfully and appended the data in JSON file.
2. I used sscanf() to parse the data in form of string and convert to integer. Since the raw data contains latitude and longitude combined with time, I developed an algorithm to extract GPS latitude and longitude from GNRMC data format.



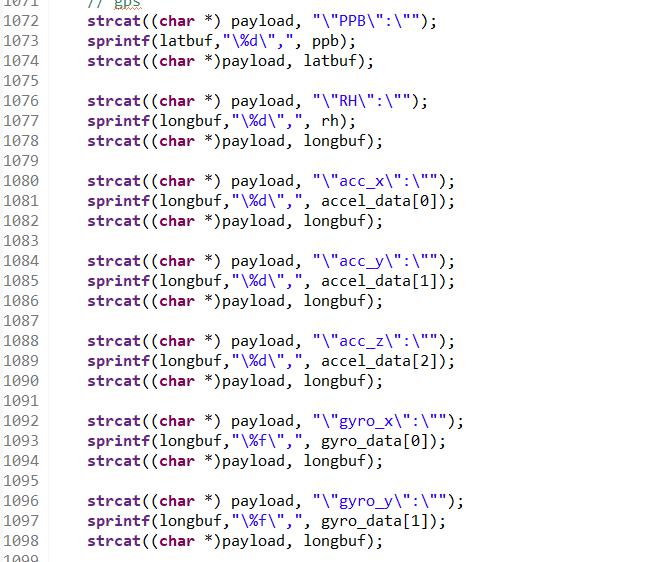
1. I used strtok() to parse the CO-Sensor Data(Parts per billion and Relative humidity). Since the data is separated by “,” , I used strtok() to scan the data and separate it from the main string. The algorithm is as follows:



1. In addition to sending the gps latitude and longitude and CO sensor’s gas concentration and relative humidity, I sent the temperature, pressure, accelerometer and gyroscope data to the cloud. GCP package has API built in to get the readings from onboard temperature sensor, accelerometer and gyroscope.
2. Below functions/API were used:



1. Once the data was present in the variable, next step was to send data over the google cloud. We stored the data format into a character buffer using snprintf() and strcat() and converted to the JSON format.



**Terminal Output:**

1. Built the project (Clean and Build All) and run the project in debug mode (Debug as-->1 Ac6 STM32 C/C++ Application)
2. After it enters the debug mode, run the project using the green run icon.
3. After running in debug mode, we get the output over teraterm as explained in section 5.5 of STM32 GCP user manual (<https://www.st.com/content/ccc/resource/technical/document/user_manual/group1/e9/8a/9b/73/5c/ff/4d/10/DM00522079/files/DM00522079.pdf/jcr:content/translations/en.DM00522079.pdf>)
4. In teraterm, go to terminal->receive->AUTO, transmit->LF, Enable Local Echo. Go to serial port-->Baud rate: 115200
5. Perform the following steps as the output is shown in teraterm

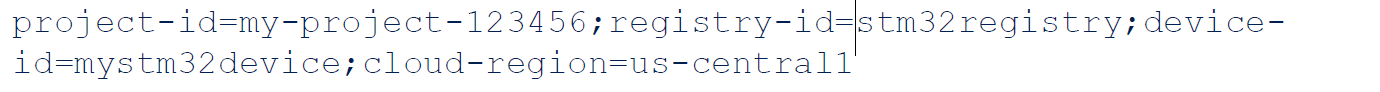
1) Enter Wifi SSID, Password and security type. **Make sure to use your Mobile Hotspot to establish data connection.**

2) Enter the TLS certifications by adding the public key and device key generated while configuring Google cloud in the terminal

3) The public key can be found inside “comodo\_google.pem” found here: **STM32CubeExpansion\_Cloud\_GCP\_V1.0.0\Projects\Common\GoogleIoT**

4) The private key is the one used to configure the google cloud topic.

3) Enter the Google cloud core connection server address (project id, device id, registry id, region). The following format is accepted in the terminal:



4) After socket connection is established, press the blue button twice to send data over the google cloud. Basically, it will subscribe to a pub/sub topic and publish the data over that topic.

1. Below is the output format going in the JSON format



**Common Problems and Solutions:**

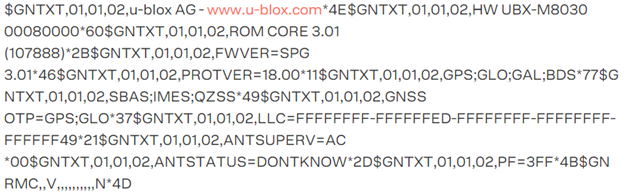
1. **During configuring TLS certificates in teraterm, authentication failed multiple times. You might get the error in terminal: “ #define MBEDTLS\_ERR\_X509\_INVALID\_FORMAT -0x2180”**

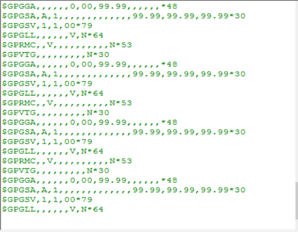
Solution: Generating a new device key with different key format like RS256, ES256 should solve the issue. Else check whether you are adding the public key from file “comodo\_google.pem” found here: **STM32CubeExpansion\_Cloud\_GCP\_V1.0.0\Projects\Common\GoogleIoT**

1. **In Spite of doing the configuration properly, it was not able to trigger the UART2 interrupt to communicate with the CO Sensor.**

Solution: During configuring for the first time, it is necessary to enable UART2 global interrupt by adding the IRQ handler in stm32l4xx\_it.c file. After including them, sensor should start sending data over UART

1. **Sometimes the GPS module won’t be working and giving the data. The raw data obtained from UART4 can be a GNTXT data or an invalid as below screenshots:**





Solution: This data format shows up because the **GPS is not locked.** Follow the link for reference: <https://portal.u-blox.com/s/question/0D52p00008HKDh0CAH/gps-not-locking-properly>. You might need to go outdoors to ensure the GPS is locked because this problem occurs indoors or in closed space.

**Other References:**

<https://www.mouser.com/applications/stm32l4-kit-iot-node-integrating-google-cloud/>