Gigabits Device Connector Framework

We would like to build projects with a wide range of processors. This document describes what's required to attach the Gigabits IoT Platform to a processor. We assume that the user is familiar with the processor and its software.

Before enumerating the steps required to send and receive data, you should make sure that your device support SSL encryption for data moving between the device and the MQTT broker. SSL support is new enough that it's still a wonderful differentiator, not a boring commodity. The MRK1000 has special chips to hold certificates and encrypt/decrypt data. The ESP32 has a second core, but the new ESP32-S2 will have special chips. Ideally, once the mechanism has been set up, you can ignore it. The instructions for building ESSP32 and MKR1000 projects have detailed sections on encryption.

1. Connect your processor to the Internet via WiFi. The Arduino and ESP processors do this whenever the processor is reset. A Raspberry Pi does this only when the processor is initially configured.
2. If the Beta program is still running, use a web browser to open www.gigabits.io and request a Beta key. When you get one, go to www.gigabits.io/register and fill in the registration form.
3. Go to app.gigabits.io and sign in with the username and password you used when you registered your project. This takes you to the Project page.
4. If the project you'd like to use is not shown, press the blue "+" button to get a new project.
5. In either case, press the project's blue configuration icon, then select "Edit Project".
6. If the project doesn't have a Device, press the "Add Devices" button on the far right. If it's not visible, make your window wider.
7. Gigabits doesn't support your device yet, so pick any device and press its "Add" button. In the upper left corner of the window, press the "Edit" that's after your project's name and before "/ Add".
8. You should now see a device with a Device Key and Secret.
9. Pick an MQTT package that your device supports. For ESP32 and MKR1000, we used Light Weight MQTT (lwmqtt).
10. Compose the topic used to send data to the server as "device/<your device key>/records"
11. Compose the topic used to receive commands from the server as "server/<your device key>/command".
12. The device key and secret are also used to authenticate your device to the broker. The broker takes a username and password. Set these equal to the deviceKey and secret respectively. The lwmqtt package also has a ClientID. Set that to the deviceKey as well. The method for doing this will vary with the MQTT package chosen. It generally happens when the device tries to connect to the MQTT broker.
13. Once the device is connected to the broker, we can receive sensor data. Each sensor datum is a key/value pair. The key is one of the integers defined below:

    HUMIDITY_SENSOR_IDX 1
    TEMPERATURE_SENSOR_IDX 2
    OLED_INVERT_COMMAND_IDX 3

PRESSURE_SENSOR_IDX 4
GAS_SENSOR_IDX 5
SOIL_SENSOR_IDX 6
PROXY_SENSOR_IDX 7
VISIBLE_LIGHT_SENSOR_IDX 8
INFRARED_LIGHT_SENSOR_IDX 9

The value is a number, like the temperature or humidity.

In Software Step 16a of the MKR1000 or ESP32 instructions, there's a brief description of how we could have different kinds of sensors, or several of the same type of sensor.

14. The device wakes up every 5 seconds or so. It gathers the sensor key/value pairs into a dictionary. When all the sensor values have been placed in the dictionary, the dictionary is converted into a JSON object and sent to the server.