Golioth with PSoC 6 Wi-Fi BT Kit

Yadati Krishna & Darla Pratheek

CONTENTS

1 Introduction 1

2 Golioth-Platform Overview 1
- 2.1 Registration & Wizard 1
- 2.2 Manage Devices 2

3 Board Support Tiers 2
- 3.1 Continuously Verified Boards 2
- 3.2 Verified Boards 2
- 3.3 Unverified Boards 2

4 Golioth with Esp32 2
- 4.1 Setup Esp-IDF for Esp32 2
- 4.2 Setup Zephyr for Esp32 2

5 Golioth with PSoC 3
- 5.1 ModusToolBox 3
- 5.2 Install ModusToolBox 3
- 5.3 Golioth Example project 3
- 5.4 Compile and flash the Golioth App . 3

1. Introduction

Golioth is an IoT platform that provides cloud services for embedded devices. Colloquially, to solve the impedance mismatch between hardware and cloud engineering teams. Golioth is purpose-built for you and the hardware you develop. Enable everything that your devices need from the cloud, including device messaging, security, updates, analytics, and more.

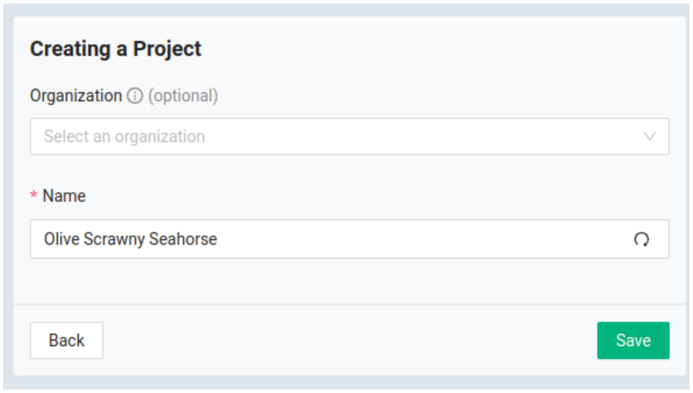
2. Golioth-Platform Overview

To prepare your Golioth account to communicate directly with your hardware devices.The following steps are involved:

A. Registration & Wizard

To begin using Golioth please register for an account at Golioth. Once registered, you can review the terms of service and continue to the Wizard that guides you through provisioning your first device.
Follow These steps:

1) **Project name:** Enter a Project Name of your choosing



Creating a Project

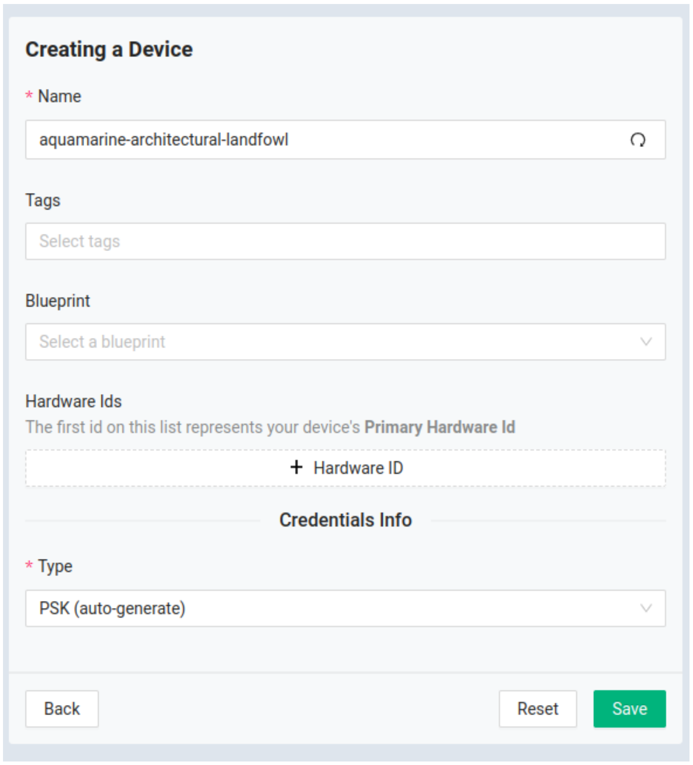
Organization (optional)

Select an organization

* Name

Olive Scrawny Seahorse

2) **Device name:** Enter a Device Name



Creating a Device

* Name

aquamarine-architectural-landfowl

Tags

Select tags

Blueprint

Select a blueprint

Hardware Ids

The first id on this list represents your device's **Primary Hardware Id**

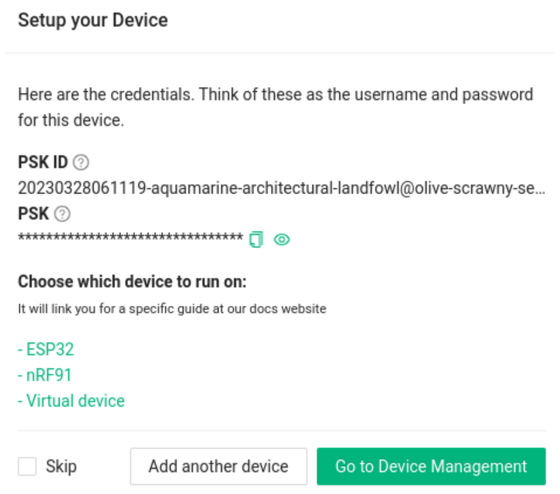
+ Hardware ID

Credentials Info

* Type

PSK (auto-generate)

3) **Device credentials:** The Identity of this device is automatically populated from the device name with -id and @project-name appended,A Pre-Shared Key (PSK) is automatically generated. This is a password that will authenticate this device to the Golioth Cloud.



Setup your Device

Here are the credentials. Think of these as the username and password for this device.

PSK ID
20230328061119-aquamarine-architectural-landfowl@olive-scrawny-se...

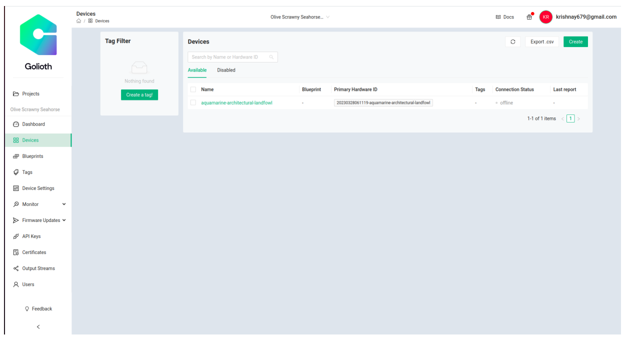
PSK
****

Choose which device to run on:
It will link you for a specific guide at our docs website

- ESP32
- nRF91
- Virtual device

B. *Manage Devices*

Use the Golioth Console to create a new device.



Name	Blueprint	Primary Hardware ID	Tags	Connection Status	Last report
aquamarine-architectural-landfowl	-	20230328061119-aquamarine-architectural-landfowl	-	offline	-

In the center at the top of the console widow the currently selected project is shown. On the left sidebar we can use the Devices option to list this project's devices. Here we see the device that was created by the quickstart wizard.

3. Board Support Tiers

Golioth has three levels of board support: Continuously Verified, Verified, and Unverified.

A. *Continuously Verified Boards*

A continuously verified board is highly recommended for new users that want to try out Golioth,is tested on every release of the Golioth Zephyr SDK,is regularly tested and used by the Golioth development team & has first class support and maintenance from Golioth. The set of boards in this category covers commonly used connectivity options including WiFi, cellular, and Thread. Additionally, these boards cover common MCUs, such as the ESP32 and nRF91.

B. *Verified Boards*

A verified board is tested and confirmed to work with Golioth,is tested less frequently than continuously verified boards. This means it was tested on an older version of the Golioth Zephyr SDK, but may not have been tested on the most recent commits & is supported and maintained by the Golioth development team.Boards in this category cover a wider range of MCUs and peripherals.

C. *Unverified Boards*

An unverified board has not yet been verified to work with Golioth.It's very possible that the board may work well with Golioth, but it has not yet been tested by the Golioth team.Check if the board is in List of supported boards. If it's in the list, there's a good chance it will work with Golioth with low development effort.

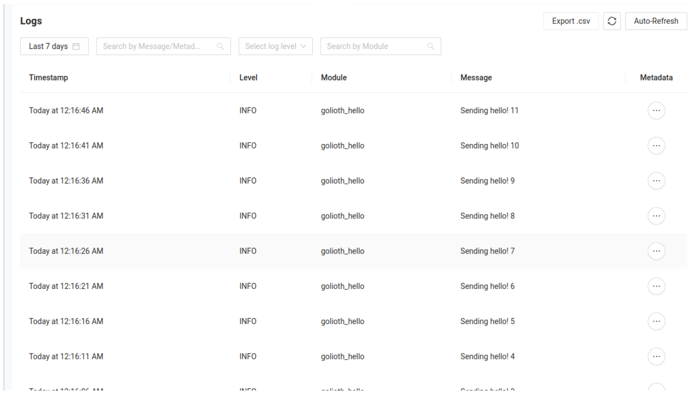
4. Golioth with Esp32

The ESP32 is a Wi-Fi and Bluetooth combination chip from Espressif Systems. There are multiple versions of the chip, with the most recent versions running in different configurations of core types, up to and including a RISC V core (on the ESP32-C3).The Espressif team has built-in support for Zephyr. This, in combination with the low cost and wide availability of the chips, has led to the ESP32 being one of the first platforms that Golioth has chosen to support.

A. *Setup Esp-IDF for Esp32*

The Required installations for Esp-IDF is given in Github.

1) Install all the required commands as given from the above link.
2) Build and Flash the code as given in the link above.
3) Then open monitor using monitor command "idf.py monitor".
4) Open your golioth account and notice logs you can see this



```
Timestamp              Level   Module         Message             Metadata
Today at 12:16:46 AM   INFO    golioth_hello  Sending hello! 11
Today at 12:16:41 AM   INFO    golioth_hello  Sending hello! 10
Today at 12:16:36 AM   INFO    golioth_hello  Sending hello! 9
Today at 12:16:31 AM   INFO    golioth_hello  Sending hello! 8
Today at 12:16:26 AM   INFO    golioth_hello  Sending hello! 7
Today at 12:16:21 AM   INFO    golioth_hello  Sending hello! 6
Today at 12:16:16 AM   INFO    golioth_hello  Sending hello! 5
Today at 12:16:11 AM   INFO    golioth_hello  Sending hello! 4
```

B. *Setup Zephyr for Esp32*

Golioth can be added to a device with Device SDKs which are based on different embedded Operating Systems. Currently Golioth targets the Zephyr Project and builds upon the APIs & tools of Zephyr.
The Required installations are given in Github.

1) Install all the required commands as given from the above link.
2) Build and Flash the code as given in the link above.
3) Then open monitor using screen command "screen /dev/ttyUSB0 115200".

4) Open your golioth account and notice logs you can see as in Esp32

5. Golioth with PSoC

Infineon's PSoC 6 chips are feature rich 32-bit Arm microcontrollers. Paired with the Infineon's 4343W, it a perfect platform for IoT device builders, and exactly the kind of constrained device that Golioth was built for.Golioth now supports Infineon parts via the ModusToolbox.

A. *ModusToolBox*

ModusToolbox (MTB) is a software support tool from Infineon Technologies. It includes partner SDKs alongside the company's officially supported IDEs, drivers, and examples. You can pull in the Golioth example and all dependencies using the Eclipse IDE that is included in MTB, or via the command line tool.

B. *Install ModusToolBox*

1) Install ModusToolBox Software as per your operating system.
2) Extract the files and do all the necessary installations(Which are given in docs folder of ModusToolBox) as per the OS you are using.
3) Then run modustoolbox-eclipse which is located in the ide3.0/eclipse/ subfolder.

C. *Golioth Example project*

1) With the Eclipse IDE open, click on File/New/ModusToolbox Application. This will launch the project creator window.
2) Select the CY8CPROTO-062-4343W from the list of PSoC 6 boards and then click next.
3) Choose the Golioth Example from the Wi-Fi list and click on the Create button. This will take a couple of minutes to clone the Golioth code and all dependencies.
4) Golioth uses MCUboot as the secure bootloader for our Over-the-Air updates. Before flashing the app to the board, we need to compile and install MCUboot. I did this using the IDE's built-in terminal.
5) First, we need to install the MCUboot dependencies.

```
cd /mtw/mtb_shared/mcuboot/v1.8.1−cypress/scripts/
python −m pip install −r requirements.txt
```

6) Now we can compile and flash MCUboot. Remember to plug a USB cable into the KITPROG3 connector on your PSoC 6 devboard before running the program command:

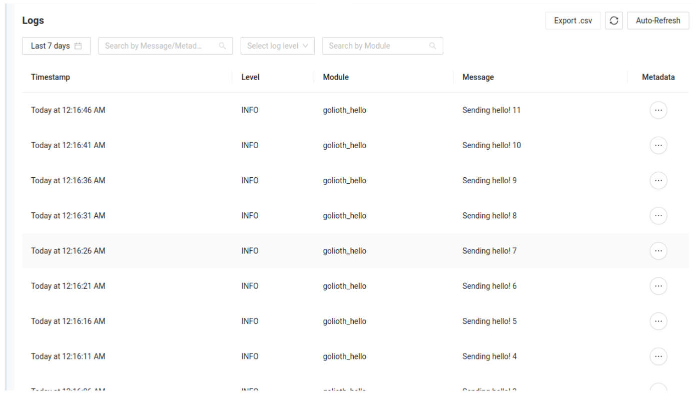
```
cd /mtw/Golioth_Example/bootloader_cm0p/
make build_proj −j8
make program_proj
```

D. *Compile and flash the Golioth App*

1) Before compiling the Golioth App we need to give it credentials to connect to Wi-Fi and also to authenticate with the Golioth server. These are set in the mtw/GoliothExample/goliothapp/source/goliothmain.h file.
2) Use the Wi-Fi credentials for your local access point. Get device credentials from the Golioth Console.
3) Once you have saved your changes to the goliothmain.h file, use the terminal to compile and flash the app to your PSoC 6 board:

```
cd /mtw/Golioth_Example/golioth_app
make build_proj −j8
make program_proj
```

4) By monitoring the serial output from the device (that's /dev/ttyACM0 on my system) we can see all the parts of the app at work. The board powers up and reports the firmware version before connecting to Wi-Fi. Once a Golioth connection is established it checks for firmware updates before it starts writing data to the cloud.
5) You can view device logs remotely through the Golioth console. Here we see Sending hello! messages arriving along with a counter.



Timestamp	Level	Module	Message	Metadata
Today at 12:16:46 AM	INFO	golioth_hello	Sending hello! 11	···
Today at 12:16:41 AM	INFO	golioth_hello	Sending hello! 10	···
Today at 12:16:36 AM	INFO	golioth_hello	Sending hello! 9	···
Today at 12:16:31 AM	INFO	golioth_hello	Sending hello! 8	···
Today at 12:16:26 AM	INFO	golioth_hello	Sending hello! 7	···
Today at 12:16:21 AM	INFO	golioth_hello	Sending hello! 6	···
Today at 12:16:16 AM	INFO	golioth_hello	Sending hello! 5	···
Today at 12:16:11 AM	INFO	golioth_hello	Sending hello! 4	···