



Morse Micro  
reaching farther™

# MM8108-EKH05

## USER GUIDE

Copyright © 2025 Morse Micro

# Table of Contents

<b>1 Introduction</b>	<b>4</b>
<b>2 Features</b>	<b>5</b>
<b>3 Development Environment</b>	<b>9</b>
3.1 System Environment	9
3.2 Development Options	9
3.2.1 CMSIS Pack	9
3.2.2 MM IoT SDK	10
3.2.3 PlatformIO + MM IoT SDK	10
<b>4 Getting Started</b>	<b>11</b>
4.1 Default Jumper Configuration	12
4.2 AP Device Settings	13
4.2.1 Changing Channel, Bandwidth, DTIM Period	15
4.3 Software Examples	16
4.4 Viewing MM8108-EKH05 Demo HTTP Server	17
<b>5 Software Development</b>	<b>18</b>
5.1 Installing CMSIS Package	18
5.2 Build and Run Example Applications	22
5.2.1 UART Output	30
5.2.1.1 Windows	30
5.2.1.2 Linux	31
5.2.1.3 Mac OS	31
5.3 Changing Example Application	32
5.3.1 rf-test Example Considerations	34
5.4 Changing Country Code	35
5.5 Changing Other Example Configurations	36
5.6 Changing Between SPI and SDIO	37
5.7 Changing Network Stacks	38
5.8 Updating BLUENRG-M2SP Module Firmware	40
<b>6 Hardware Layout and Configuration</b>	<b>43</b>
6.1 Power Selection	43
6.2 Using an External Debugger/Programmer	44
6.4 Switching Between SDIO and SPI	45
6.5 Switching Between SMA and U.FL Connector	46
6.6 Disconnecting Sensors	47
<b>7 Power Consumption Measurements</b>	<b>48</b>

<b>7.1 Power Consumption Measurement Points</b>	<b>48</b>
<b>7.1.1 General Structure</b>	<b>48</b>
<b>7.1.2 Whole System Power Consumption</b>	<b>48</b>
<b>7.2 Power Consumption Measurement Procedure</b>	<b>49</b>
<b>7.3 Optional Debug GPIO Connections</b>	<b>51</b>
<b>7.4 Measuring DTIM Power</b>	<b>53</b>
<b>7.5 Measuring WNM Sleep Power</b>	<b>56</b>
<b>7.6 Current Spikes When Waking Up</b>	<b>59</b>
<b>7.7 Negative Current Consumption</b>	<b>61</b>
<b>8 Troubleshooting</b>	<b>64</b>
<b>8.1 Device Is Not Programming</b>	<b>64</b>
<b>8.2 Poor HaLow Performance</b>	<b>66</b>
<b>8.3 High Power Consumption</b>	<b>67</b>
<b>9 Errata</b>	<b>68</b>
<b>10 Revision History</b>	<b>69</b>
<b>Appendix A: Schematics</b>	<b>70</b>
<b>Appendix B: Mechanical Drawing</b>	<b>78</b>

# 1 Introduction

The MM8108-EKH05 evaluation kit is a fully integrated Wi-Fi HaLow development platform, designed for a wide range of IoT applications—from smart home devices to industrial automation systems. It features the MM8108-MF15457 module, incorporating Morse Micro’s MM8108 Wi-Fi HaLow low-power SoC, alongside the STM32U585 low-power microcontroller (MCU) and the BlueNRG-M2 Bluetooth® SoC, the board provides robust wireless connectivity, low power consumption, and an extensive range of programmable interfaces and sensors. This versatile platform is ideal for software engineers developing energy-efficient IoT solutions.

This user guide provides a comprehensive overview of the MM8108-EKH05 evaluation kit, offering step-by-step instructions for setup, configuration, and programming. It covers all aspects of the board's capabilities, including the integration of Wi-Fi HaLow and Bluetooth® connectivity, utilization of the STM32U585 MCU, and leveraging the onboard sensors and interfaces. Detailed examples and practical use cases are provided to help developers quickly prototype and validate IoT applications, ensuring they can harness the full potential of this evaluation kit. The guide also includes troubleshooting tips, optimization techniques for low-power operation, and insights into best practices for IoT system design. Whether the user is new to IoT development or an experienced engineer, this guide is structured to help efficiently create innovative, connected solutions.

Note: At this time, a Wi-Fi HaLow access point is required in addition to the MM8108-EKH05. Refer to [Section 4](#) for setup details.

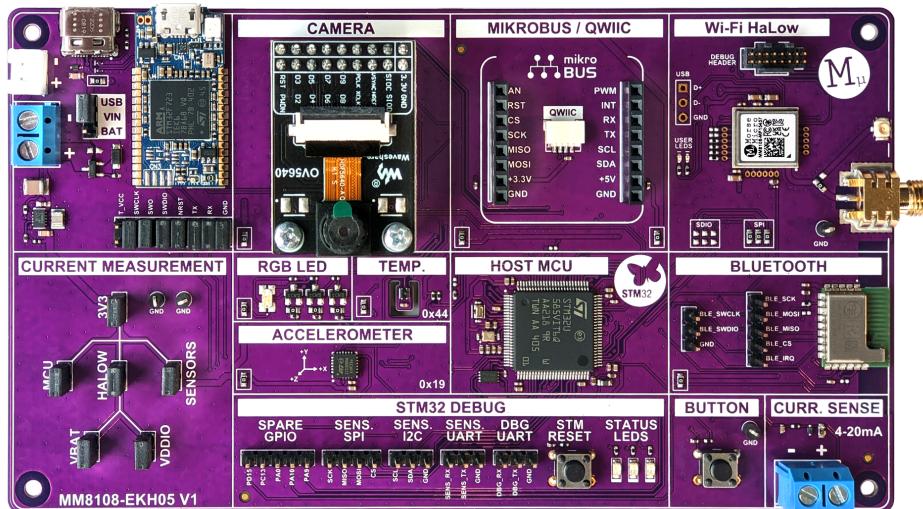


Figure 1. MM8108-EKH05 evaluation kit.

## 2 Features

This section highlights the key features of the MM8108-EKH05 Wi-Fi HaLow evaluation kit, showcasing the advanced capabilities that make it a versatile platform for IoT development.

Key features:

- Wi-Fi HaLow support for long-range, low-power wireless connectivity.
- STM32U585 microcontroller with energy-efficient architecture.
- Integrated programmable interface for flexible development.
- Integrated sensors for enhanced IoT applications.
- Multiple power input sources, including battery connectors.

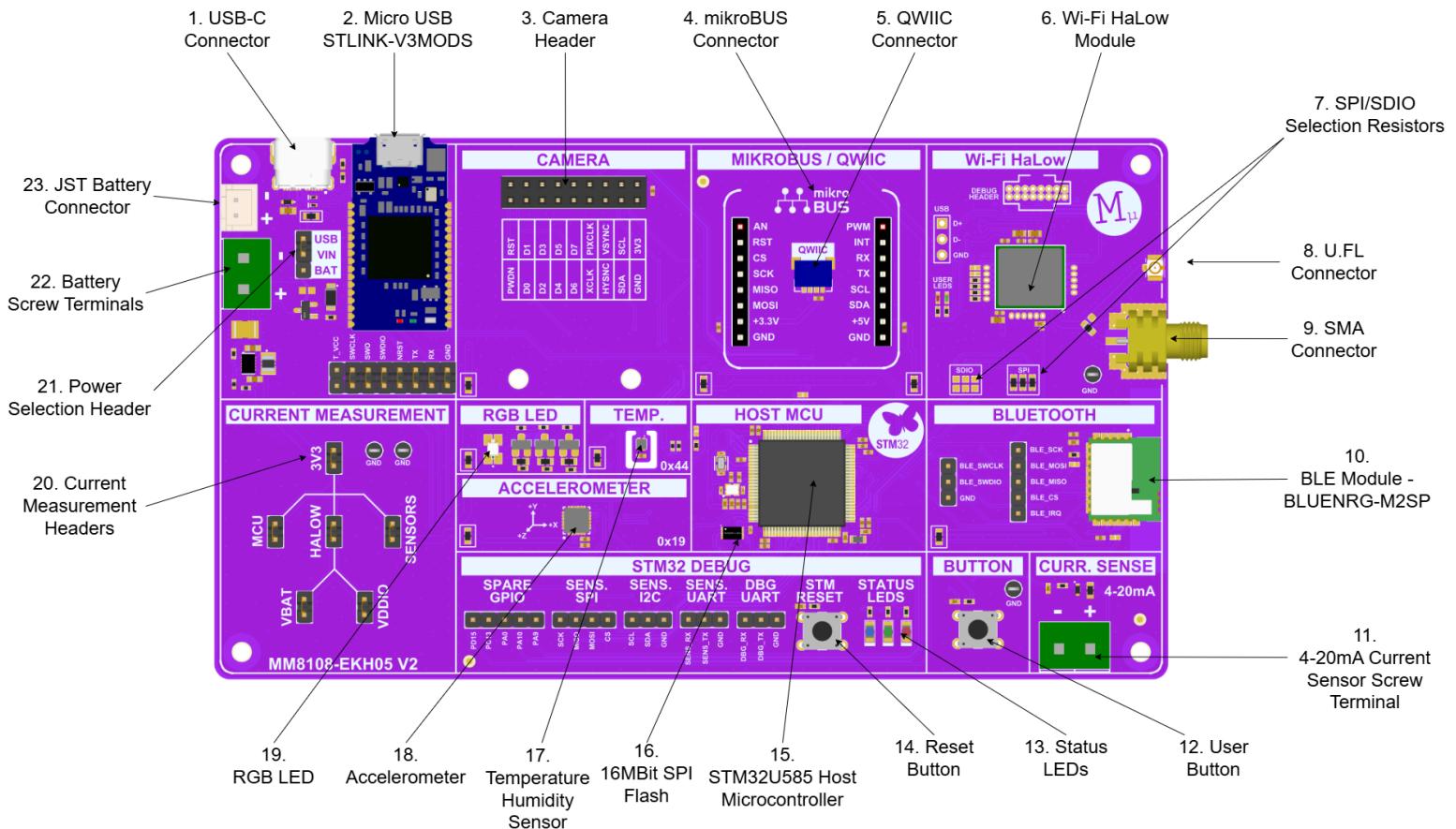


Figure 2. Features of the MM8108-EKH05.

#	Item	Description
1.	<b>USB-C Connector</b>	5V power supply. USB data lines connecting to STM32U585 host microcontroller.
2.	<b>Micro USB STLINK-V3MODS</b>	5V power supply. Programming and debugging access to STM32U585 host microcontroller. Access to STM32U585 host microcontroller UART.  Note: The header pins below the STLINK-V3MODS module allow for disconnection from the host microcontroller for low-power testing.
3.	<b>Camera Header</b>	Camera connector to suit a DCMI (Digital Camera Interface) camera module. Connects to DCMI interface on STM32U585 host microcontroller.  Note: A Camera module is provided in the MM8108-EKH05 kit.
4.	<b>mikroBUS Connector</b>	Standard connector for mikroBUS modules. Interfaces connect to STM32U585 host microcontroller.
5.,	<b>QWIIC Connector</b>	Standard connector for QWIIC modules. Interface connects to STM32U585 host microcontroller.
6.	<b>HaLow Module</b>	Morse Micro MM8108-MF15457 module with the MM8108 Morse Micro chipset. Connects to SPI interface (by default) on STM32U585 host microcontroller.
7.	<b>SPI/SDIO Selection Resistors</b>	Option to populate 0-ohm resistors for either SPI (default) or SDIO operation to STM32U585 host microcontroller.  Note: Changing this setting also requires software modifications.

8.	<b>U.FL Connector</b>	Antenna output for U.FL connections.  Note: SMA Connector is set by default. 0-ohm resistor change required to use U.FL connector. Instructions available in a chapter below.
9.	<b>SMA Connector</b>	Antenna output for SMA connections.  Note: SMA is the default antenna output.
10.	<b>BLUENRG-M2SP Module</b>	ST BLUENRG-M2SP BLE module for provisioning devices. Connects to SPI interface on STM32U585 host microcontroller.  Note: Access to SPI data lines and SWD data lines through header pins beside the BLUENRG module.
11.	<b>4-20mA Current Sensor Screw Terminal</b>	Connects to an ADC pin on the STM32U585 host microcontroller for 4-20mA operation.
12.	<b>User Button</b>	Programmable button for user applications. Connects to GPIO on STM32U585 host microcontroller.
13.	<b>Status LEDs</b>	Individual red, green, and blue LEDs for user applications. Connects to GPIO on STM32U585 host microcontroller.
14.	<b>Reset Button</b>	Resets the STM32U585 host microcontroller.
15.	<b>STM32U585 host microcontroller</b>	Ultra-low-power MCU with FPU ARM Cortex-M33 MCU with TrustZone, 160 MHz with 2 MBytes of Flash memory.
16.	<b>16MBit SPI Flash</b>	Winbond W25Q16JV SPI flash. Available as extra memory for user applications on STM32U585 host microcontroller

17.	<b>Temperature Humidity Sensor</b>	Sensirion SHT40 Ultra-Low-Power, 16-bit relative humidity and temperature sensor. Connects to I2C Interface on STM32U585 host microcontroller
18.	<b>Accelerometer</b>	ST IIS328DQ Ultra low-power 3-axis accelerometer with digital output. Connects to I2C Interface on STM32U585 host microcontroller.
19.	<b>RGB LED</b>	RGB LED connected to PWM interface on STM32U585 host microcontroller.
20.	<b>Current Measurement Headers</b>	Headers for power measurement. Follows a tree structure allowing measurement of either the entire system or individual blocks.
21.	<b>Power Selection Header</b>	Header to select system power source. Options: <ol style="list-style-type: none"><li>1. USB power from USB-C or STLINK</li><li>2. Battery JST Connector or Battery screw terminals</li></ol>
22.	<b>Battery Screw Terminals</b>	Option for connecting a battery to the system through screw terminals.
23.	<b>JST Battery Connector</b>	Option for connecting a battery to a system through a standard JST connector.

# 3 Development Environment

## 3.1 System Environment

All major operating systems are supported:

- Windows 10, Windows 11
- Linux 64-bit
- macOS (macOS 14 and below.)

The development options are described in the following section.

## 3.2 Development Options

Morse Micro provides three development options for the MM8108-EKH05. The CMSIS Pack with STM32CubeIDE fully supports all hardware and includes pre-loaded example applications for immediate, out-of-box use. The other development options are available but have limited support for the hardware peripherals of the MM8108-EKH05. The rest of this document assumes the CMSIS pack will be used and describes how to use it with the MM8108-EKH05 hardware.

### 3.2.1 CMSIS Pack

A CMSIS (Cortex Microcontroller Software Interface Standard) pack is a standardized software package format defined by ARM to facilitate the development of software for ARM Cortex-based microcontrollers. It bundles together all necessary components, such as device drivers, middleware, libraries, and configuration files, into a single package that can be easily distributed and integrated into development environments. The Morse Micro CMSIS pack includes everything needed to integrate Wi-Fi HaLow applications using this standardized package.

Morse Micro CMSIS pack releases can be found via the following link. The .pack file should be downloaded, although the source code is also available in each release.

<https://github.com/MorseMicro/mm-iot-cmsis/releases>

The instructions for using this package with STM32CubeIDE as the supported IDE are included in [section 5](#) of this document.

### 3.2.2 MM IoT SDK

The Morse Micro IoT SDK provides a framework for developing embedded applications that include connectivity provided by Morse Micro MM8108 chip. This SDK package contains the software components to build the application firmware for the STM32 processor. These components include the RTOS (FreeRTOS), network stack (LwIP or FreeRTOS + TCP), Morse Micro host software (morselib), and the platform-specific board support package (BSP).

The MM IoT SDK release packages can be found via:

<https://github.com/MorseMicro/mm-iot-sdk/releases>

NOTE: This development option does NOT support existing peripherals on the MM8108-EKH05. The rest of this user guide is tailored for the CMSIS pack development option.

### 3.2.3 PlatformIO + MM IoT SDK

PlatformIO is an open source multi-platform build system with integration available for various IDEs. It has powerful built-in features such as debugging, unit testing, and static code analysis and it supports a multitude of hardware platforms. PlatformIO also has a vast amount of libraries and example code available for various sensors and actuators used in IoT applications. The MM IoT SDK provides an integration for PlatformIO that supports multiple development boards from ST Microelectronics combined with Morse Micro extension board for developing Wi-Fi HaLow enabled applications.

The user guide for the PlatformIO can be found from the following link on the Morse Micro support portal:

[https://www.morsemicro.com/resources/user\\_guides/Platform-IO-MM-IoT-SDK-User-Guide.pdf](https://www.morsemicro.com/resources/user_guides/Platform-IO-MM-IoT-SDK-User-Guide.pdf).

NOTE: This development option does NOT support existing peripherals on the MM8108-EKH05. The rest of this user guide is tailored for the CMSIS pack development option.

## 4 Getting Started

The most straightforward setup involves connecting a host PC to both the MM8108-EKH05 as a station (STA), and a separate Wi-Fi HaLow access point (AP). The recommended AP is the HaLowLink 1, which will be referenced throughout this document.

⚠ Note: At this time, the MM6108-EKH05 **cannot** operate as an AP.

The MM8108-EKH05 connects through the STLINK Micro USB connector, and the AP connects through an Ethernet cable to the LAN port. The Host PC must set the Ethernet interface to a DHCP client - see [AP Device Settings](#). The HaLow connection can either be Over The Air (OTA) through antennas, or cabled. For cabled, it is recommended to use 40 dB of external attenuation as shown below. The device can be damaged if connected directly without attenuation.

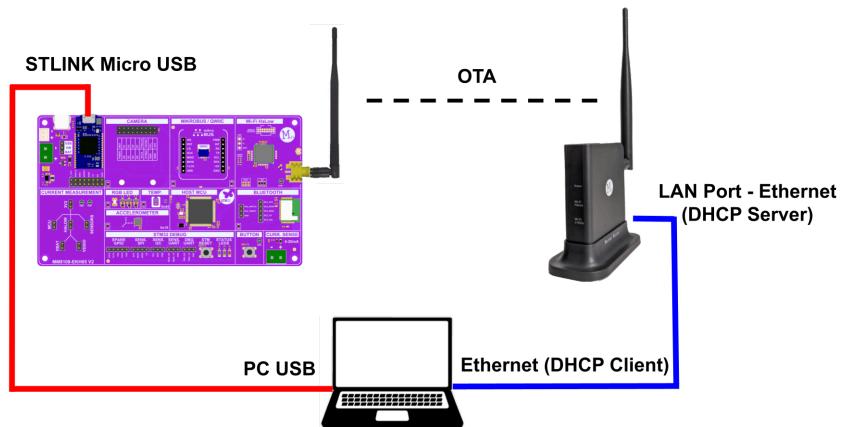


Figure 3. Hardware setup with antennas.

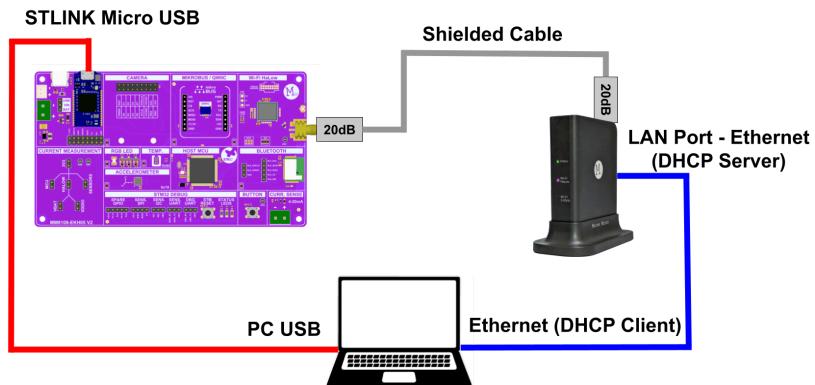


Figure 4. Hardware setup with shielded cable.

## 4.1 Default Jumper Configuration

By default, the following headers are populated:

1. USB power selection header
2. All STLINK headers (except for T\_VCC)
3. Current measurement headers

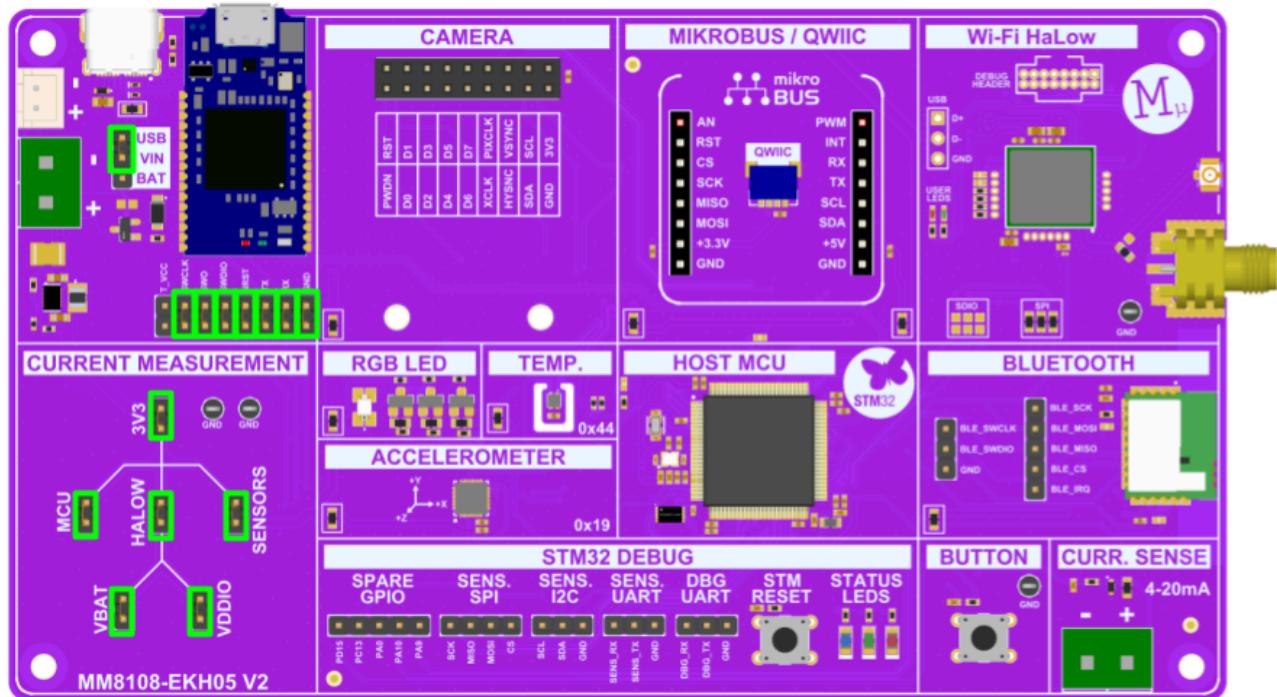


Figure 5. Default jumper placements on MM8108-EKH05.

## 4.2 AP Device Settings

This section describes how to set up the HaLowLink 1 access point (AP). These steps assume the device is being configured for the first time. For more information, the [User Guide](#) for the HaLowLink 1 can be found on the Customer Portal.

Ensure the PC used is set up as a DHCP client for its ethernet configuration. Navigate to **192.168.12.1** in a web browser. Enter the username and password on the HaLowLink 1 label. Click **Log in**.

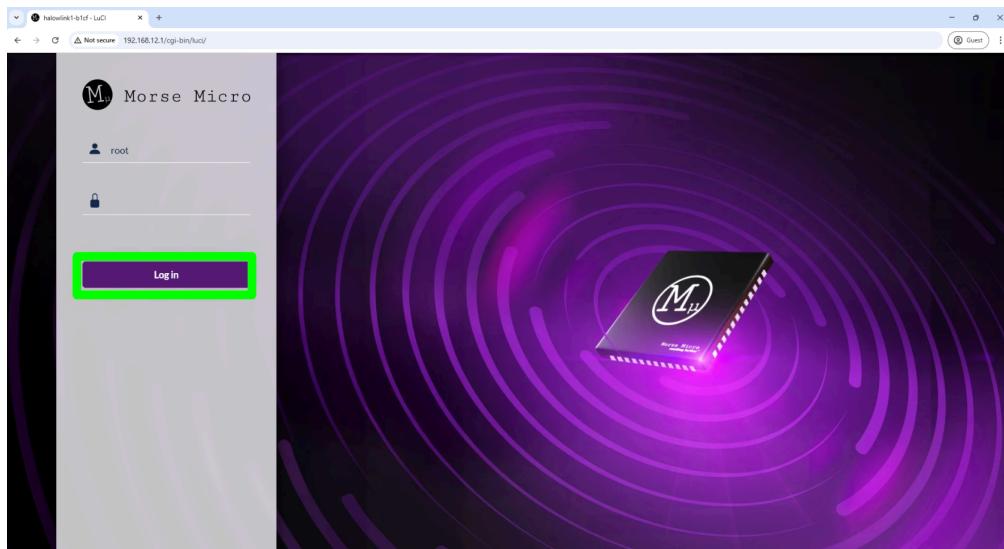


Figure 6. OpenWRT login screen.

Select **Quick Config**.

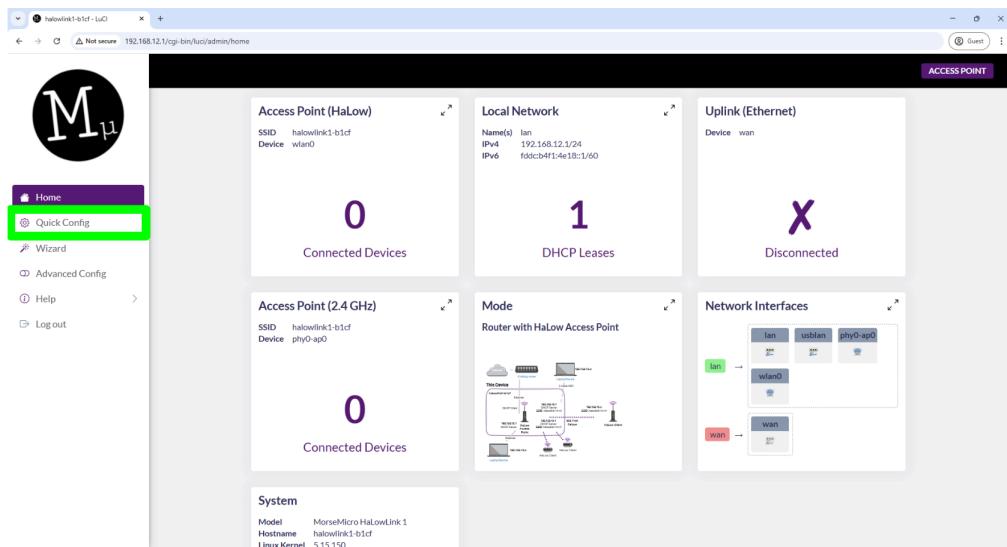


Figure 7. Highlighting the Quick Config menu.

Under the Wireless → Morse Micro HaLow Wi-Fi 802.11ah (radio1) section is where the **SSID**, **Passphrase**, **Bandwidth**, **Channel** and **Country Code** can be set.

All MM8108-EKH05 example applications expect the following SSID and Passphrase. Set the following to ensure the MM8108-EKH05 associates with the AP.

**SSID:** MorseMicro

**Passphrase:** 12345678

The default **Country Code** loaded on the HaLowLink 1 and the MM8108-EKH05 is **US**. If operating these devices in a different country, these will need to be changed. On the HaLowLink1, this can be changed from the same page. On the MM8108-EKH05, please see section [5.4](#).

After configuring, click **Save & Apply**. These parameters can be changed later if needed.

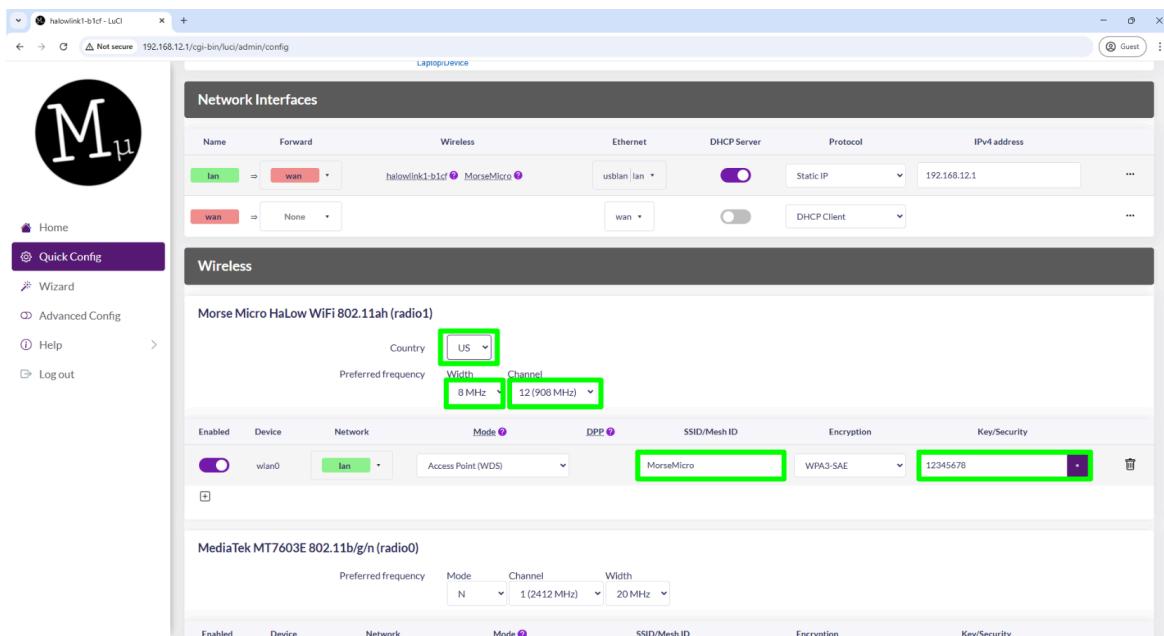


Figure 8. Quick config menu showing where to change country code, SSID, passphrase, bandwidth, and channel.

## 4.2.1 Changing Channel, Bandwidth, DTIM Period

After initial configuration, the channel, bandwidth, and DTIM period can be changed by navigating to the **Advanced Config → Network → Wireless** page, and then clicking **Edit** on the HaLow radio.

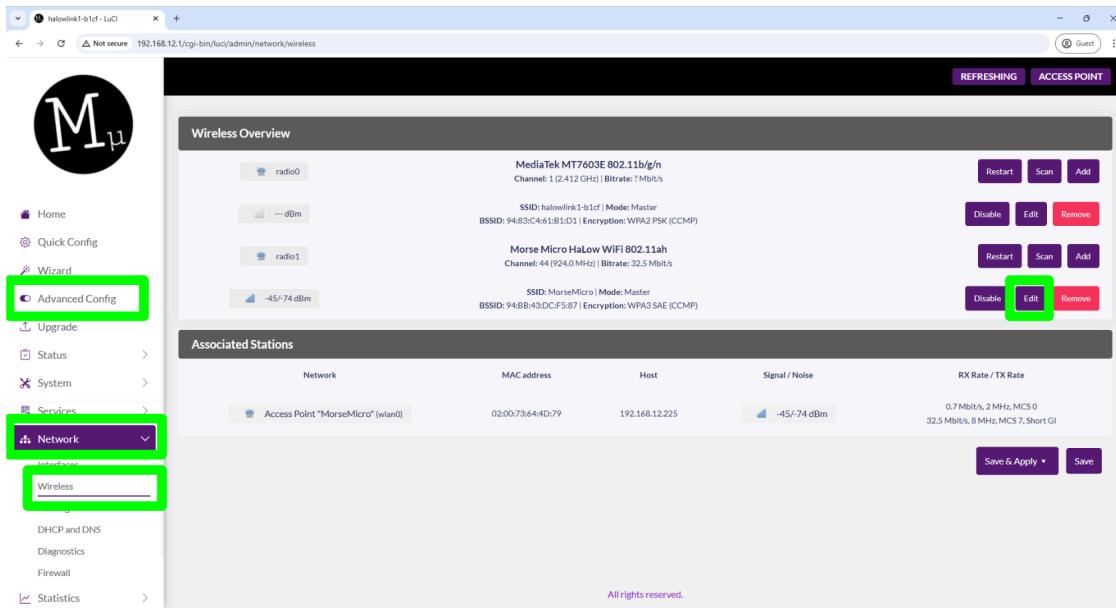


Figure 9. Wireless network configuration page on OpenWRT.

The **Bandwidth** and **Channel** can then be changed from the **Device Configuration** menu. The **DTIM Interval** can then be changed by selecting **Advanced Settings** from the **Interface Configuration** menu.

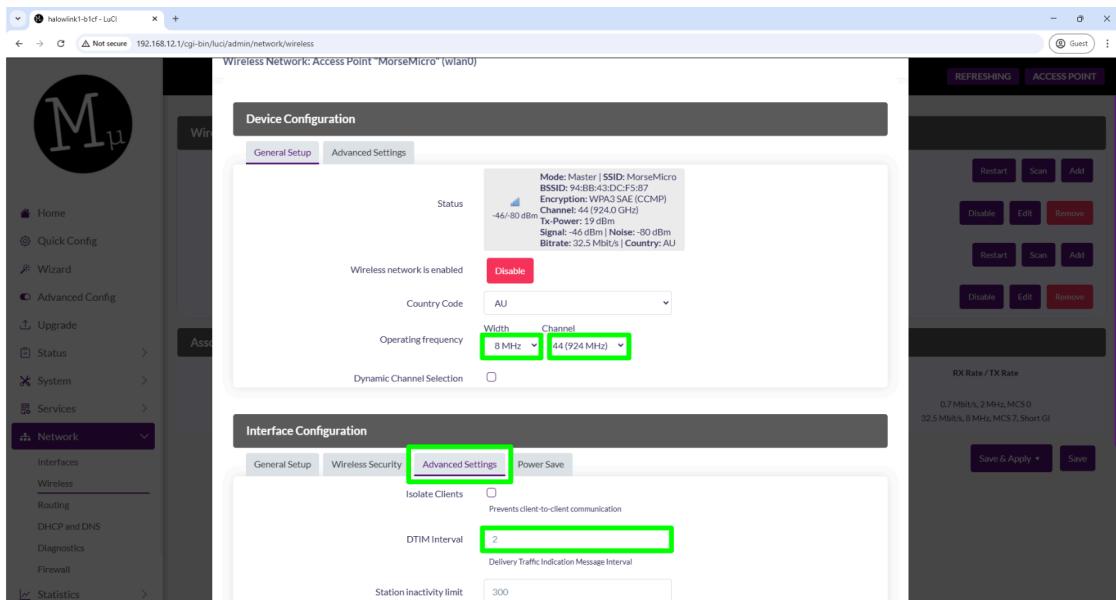


Figure 10. Advanced setup showing where to set DTIM interval, bandwidth, and channel.

## 4.3 Software Examples

The following table will describe the list of supported software examples that are included with the CMSIS pack for the MM8108-EKH05. The “EKH05 Demo” example is loaded on the device out of the box.

Software Example	Description
<b>EKH05 Demo</b>	As the default out-of-box example, the EKH05 Demo showcases the operation of all on-board peripherals of the MM8108-EKH05. A HTTP server runs on the device and displays images from the camera, temperature, humidity and accelerometer values. The BLE module is also activated and acts as a sensor.
<b>Ping</b>	Simple ping demonstration. Also used to stay connected to an AP while idling to measure DTIM sleep currents.
<b>WNM Sleep</b>	Example utilizing WNM sleep to conserve power in between periodic transmissions.
<b>iPerf</b>	Example showing how to perform throughput measurements using iPerf.
<b>AWS IoT</b>	AWS IoT example to demonstrate connecting to AWS Shadow service and demonstrate a simple light bulb example.
<b>Scan</b>	Example application to demonstrate scanning for Wi-Fi HaLow networks.

## 4.4 Viewing MM8108-EKH05 Demo HTTP Server

To view the HTTP server running on the MM8108-EKH05 device from a PC, the IP address of the MM8108-EKH05 needs to be found. To do this, navigate back to the **Home** section, and click on the **Connected Devices** link.

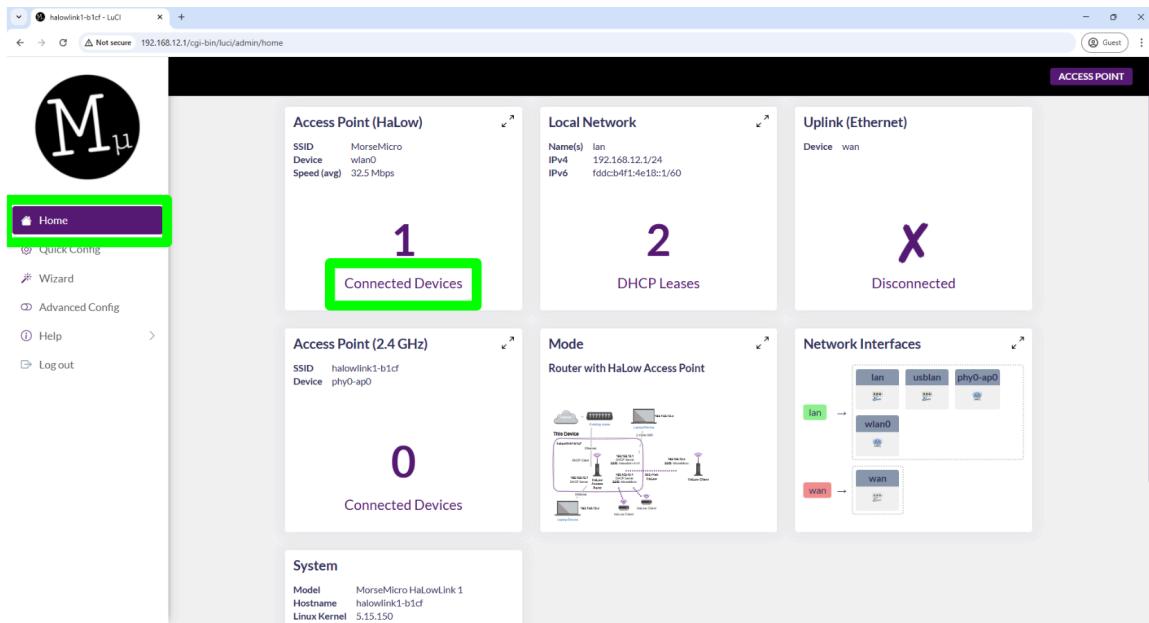


Figure 11. Home OpenWRT menu showing connected devices.

This will bring up a list of DHCP leases the AP has distributed to connected stations. In this case, the MM8108-EKH05 connected to the AP has an IP address of 192.168.12.225.

### Access Point (HaLow)

SSID	MorseMicro
Device	wlan0
IPv4	192.168.12.1
IPv6	fdccb4f1:4e18::1/60
Encryption	WPA3 SAE (CCMP)

### Associated devices

► Connect a new device

MAC Address	IPv4	IPv6	Time connected	Noise	Signal
02:00:73:64:4D:79	192.168.12.225	fe80::73ff:fe64:4d79	7 min(s)	-78 dBm	-45 dBm

Figure 12. Connected devices and their connection details.

Navigate to the IP address of the MM8108-EKH05 (192.168.12.225 in this case) in a web browser to view the contents of the HTTP server.

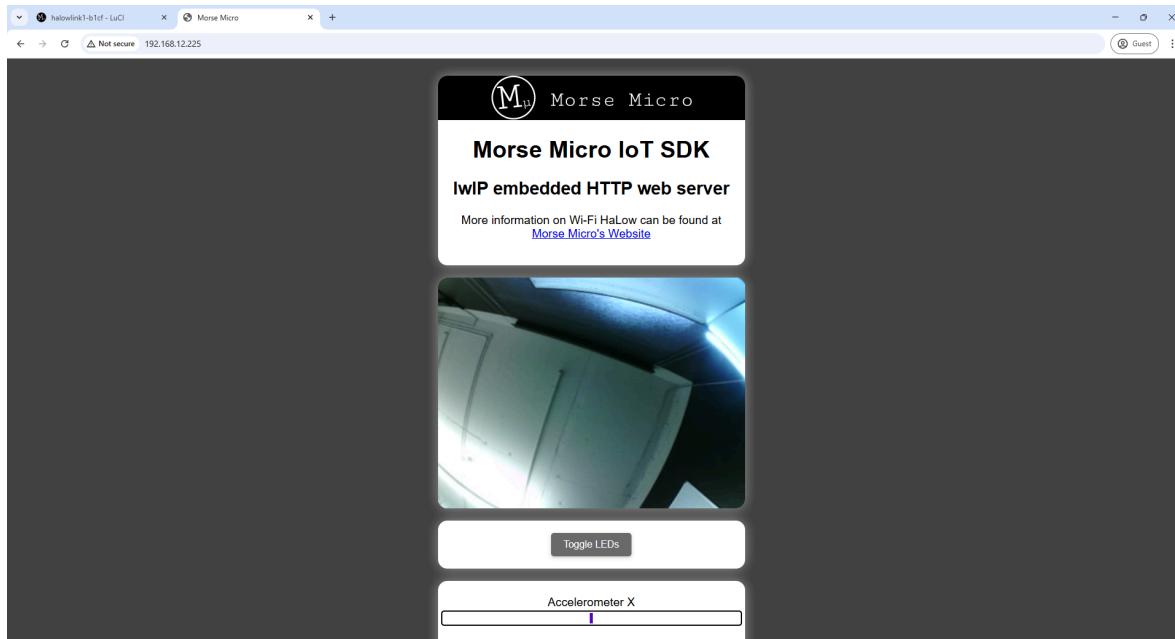


Figure 13. HTTP server running on MM6108-EKH05 demo application.

# 5 Software Development

This section will describe how to install the CMSIS package, how to build and run Morse Micro example applications within the STM32CubeIDE and how to [update the firmware on the BLUENRG-M2SP module](#).

STM32CubeIDE can be downloaded from the ST website -  
<https://www.st.com/en/development-tools/stm32cubeide.html>.

The CMSIS package can be downloaded from the Morse Micro GitHub page -  
<https://github.com/MorseMicro/mm-iot-cmsis/releases>.

## 5.1 Installing CMSIS Package

After Starting STM32CubeIDE, click on **Manage Embedded Software Packages** from the **Help** menu:

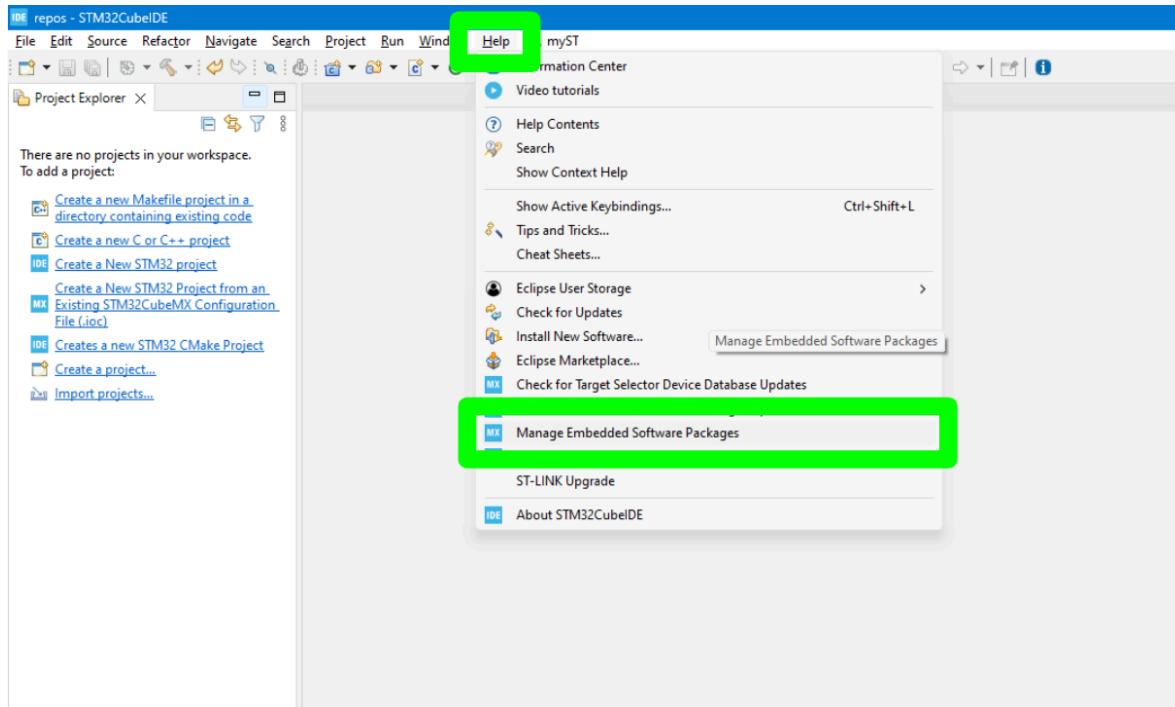


Figure 14. STM32CubeIDE help menu dropdown.

Click on **From Local ...**:

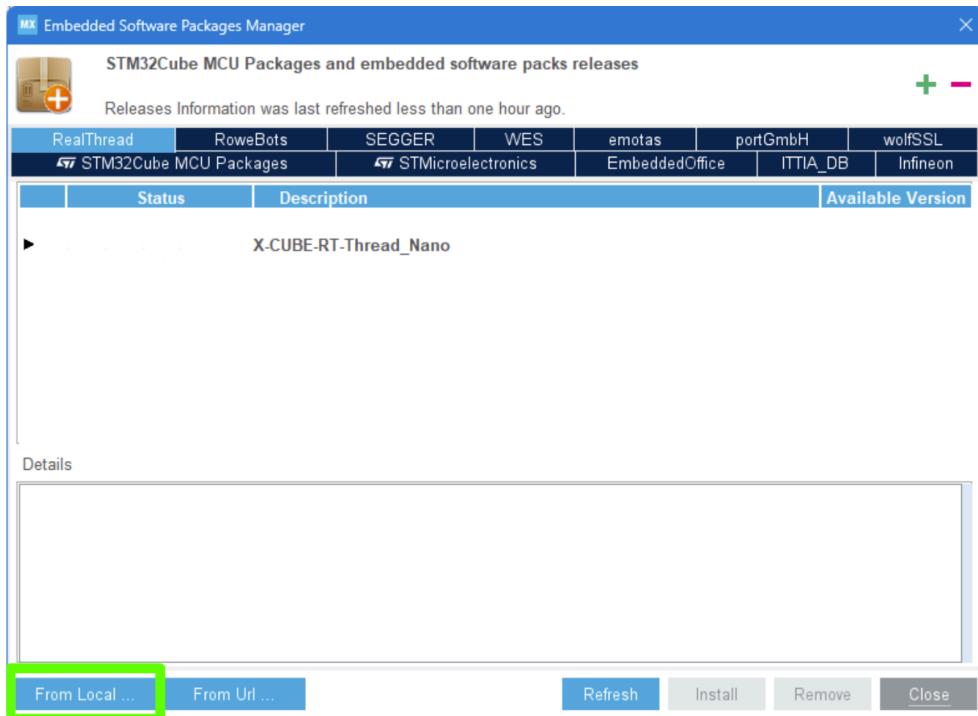


Figure 15. Embedded software packages manager menu.

Navigate to where the Morse Micro CMSIS package is downloaded, and click **Open**:

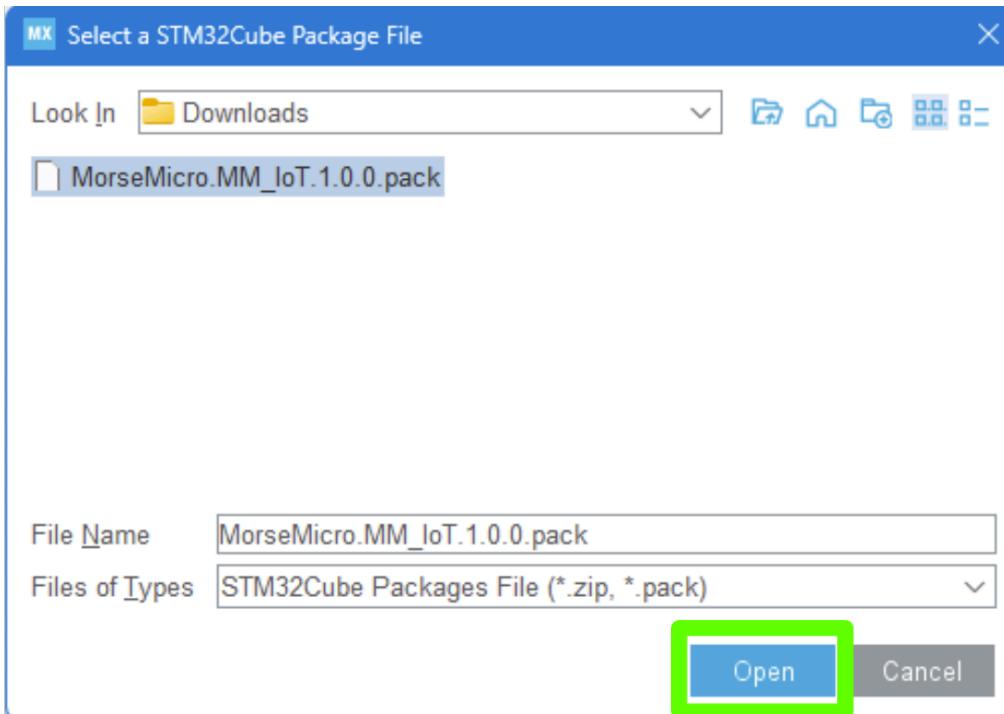


Figure 16. Opening CMSIS pack from downloaded location.

Accept the license agreement, and click **Finish**:

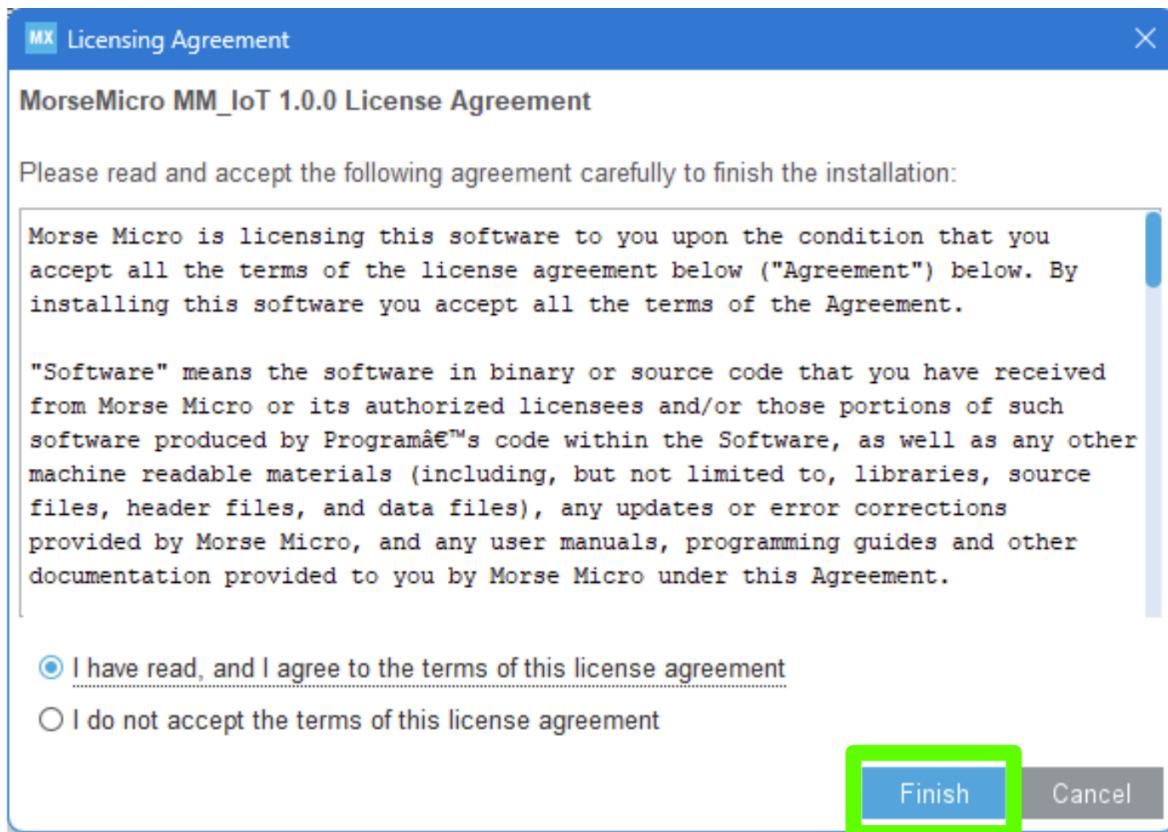


Figure 17. License agreement menu for installing CMSIS pack.

The package is now installed.

## 5.2 Build and Run Example Applications

After installing the Morse Micro CMSIS package, an example application can be run. As an example, this will run the “HaLow Example” application.

First, create and log into a myST account. This is needed to install other packages included within the HaLow example.

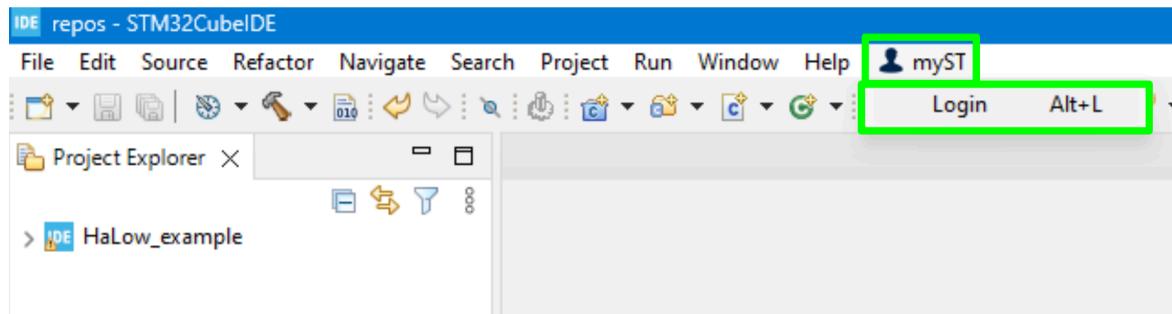


Figure 18. Where to login to ST account in STM32CubeIDE.

From **File**, click **Open Projects from File System**:

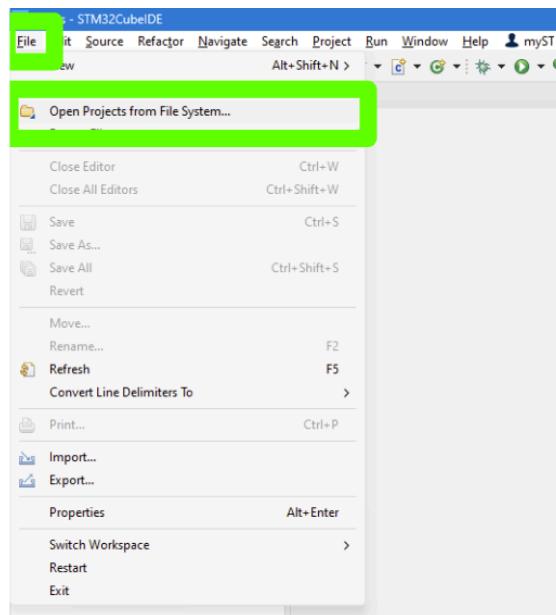


Figure 19. Where to open project in STM32CubeIDE.

Click on **Directory...**:

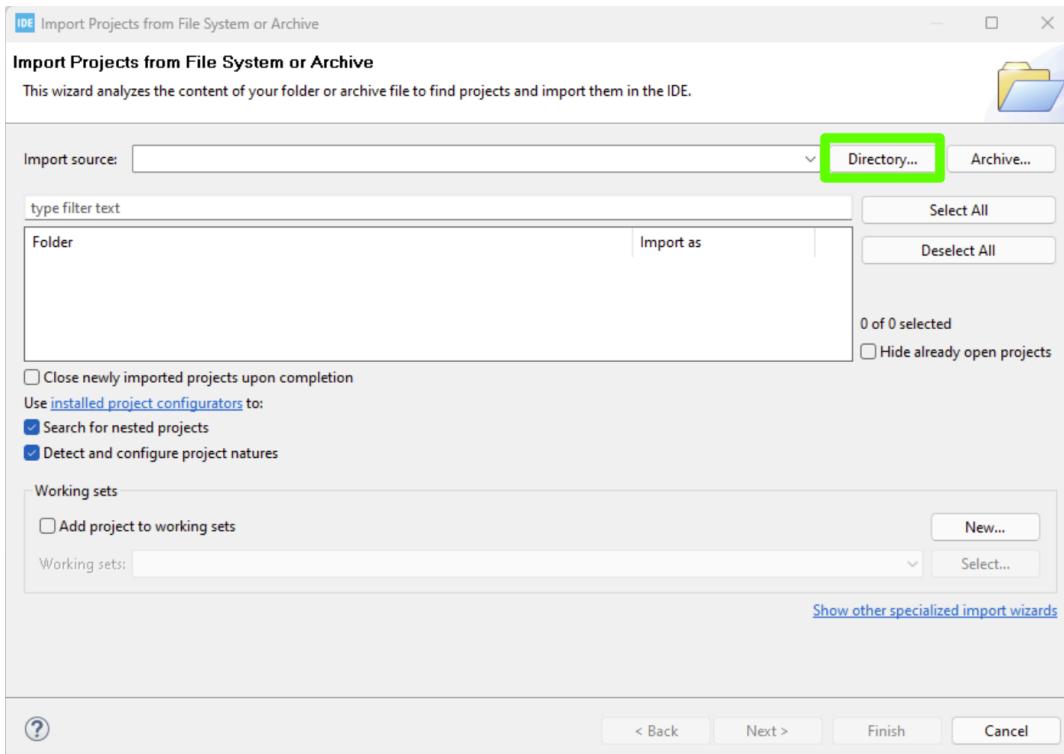


Figure 20. Highlighting directory button to search for project.

Navigate to the default STM32 package location, and then find the HaLow\_example application. Either HaLow\_example\_spi or HaLow\_example\_sdio can be chosen to use SPI or SDIO respectively. SPI will be chosen for the rest of this document.

On Windows, the default path is:

```
C:\Users\USER_NAME\STM32Cube\Repository\Packs\MorseMicro\MM_IoT\1.6.0\HaLow_example_spi
```

On Linux, the default path is:

```
/home/USER_NAME/STM32Cube/Repository/Packs/MorseMicro/MM_IoT/1.6.0/HaLow_example_spi
```

On Mac, the default path is:

```
/Users/USER_NAME/STM32Cube/Repository/Packs/MorseMicro/MM_IoT/1.6.0/HaLow_example_spi
```

Note that the version number may change depending on which version of the CMSIS pack is downloaded.

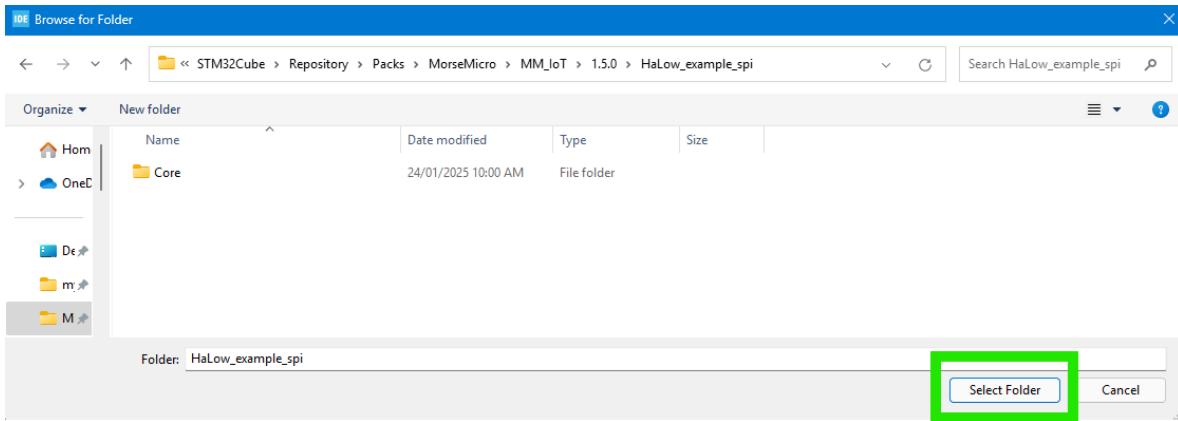


Figure 21. Example location of where to find the example application on a windows machine.

Click **Finish**:

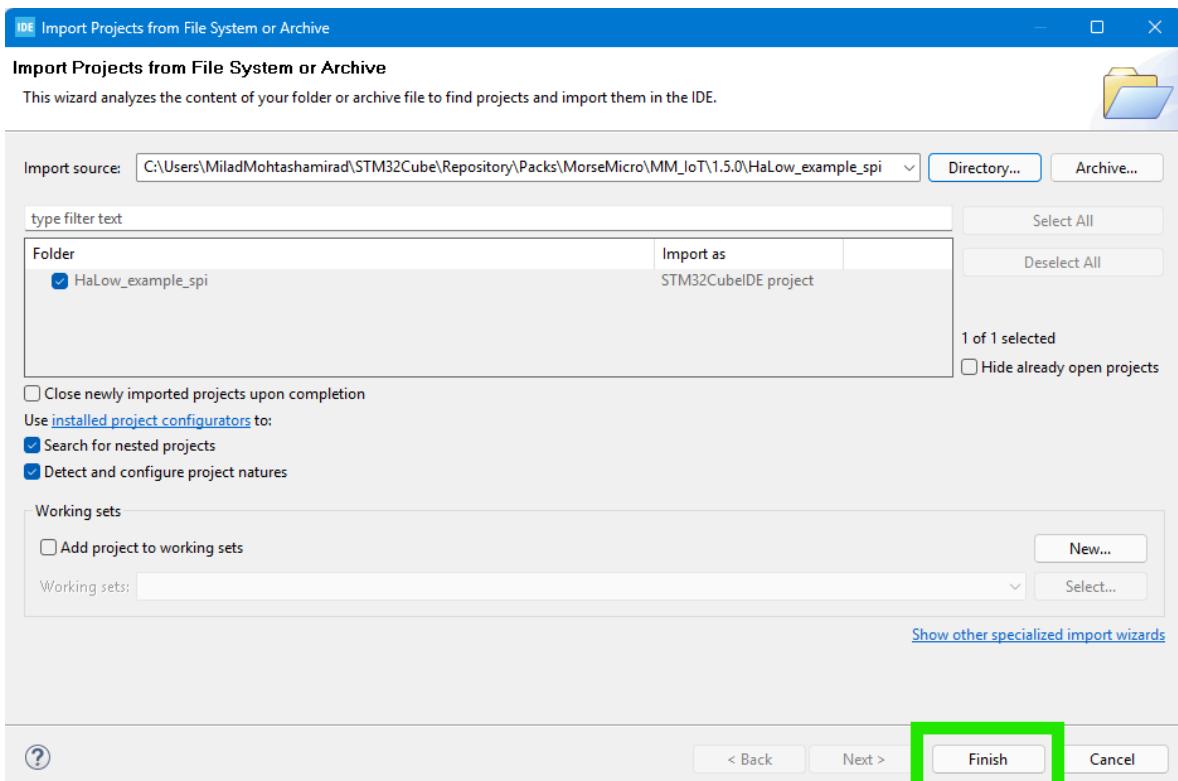


Figure 22. Finish menu importing the example application.

Expand the Halow\_example, and double click on the .ioc file:

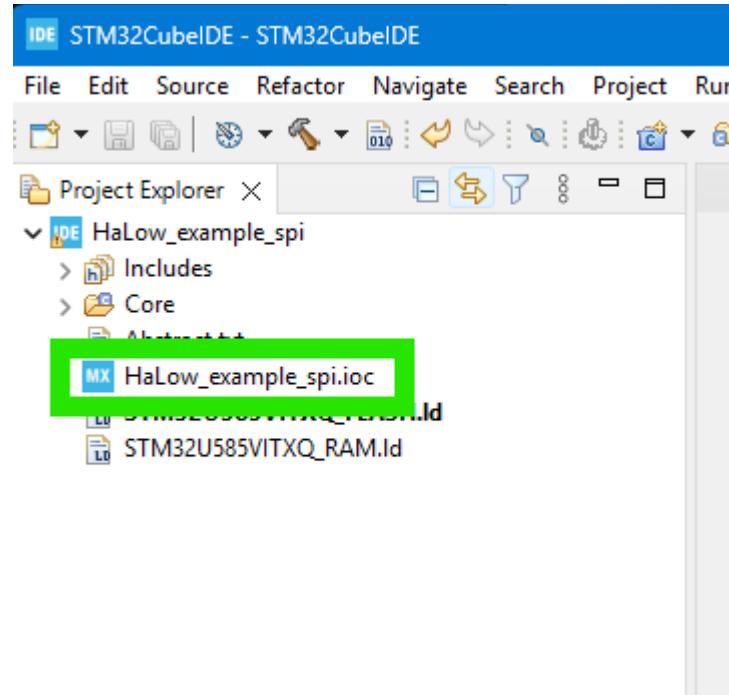


Figure 23. Highlighting .ioc file in project explorer.

Click **Continue**.

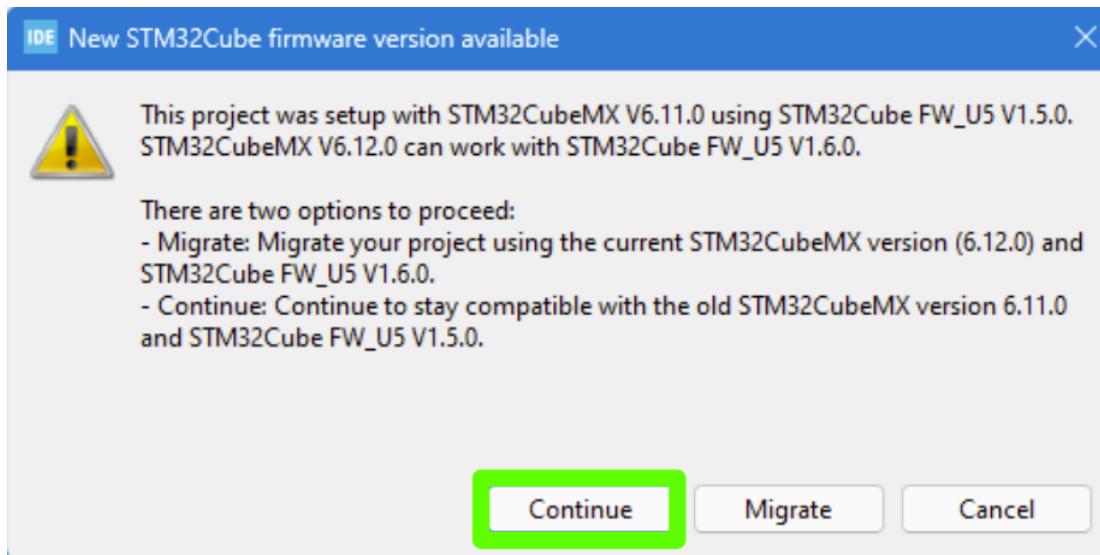


Figure 24. Pop-up menu after opening .ioc file.

There may be additional software packs that need to be installed including STMicroelectronics.X-CUBE-FREERTOS.1.2.0, and STMicroelectronics.X-CUBE-BLE2.3.3.0. Download these items in the subsequent pop-up windows.

Click **Select Components** from the **Software Packs** menu.

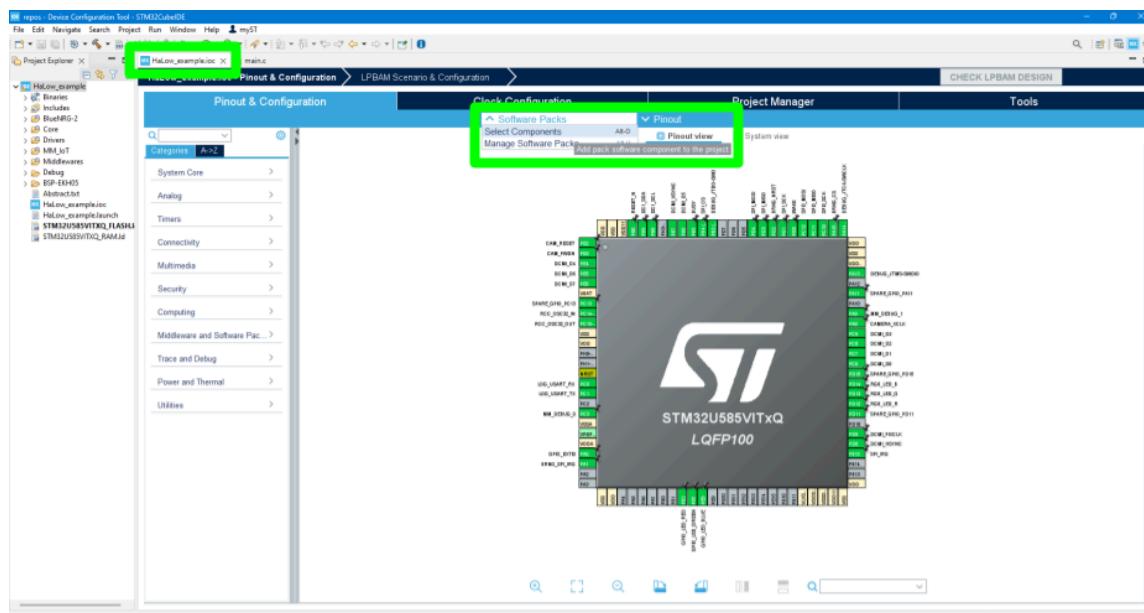


Figure 25. Select components menu in the .ioc file.

Expand **MorseMicro MM\_IoT**, and then **MM\_IoT**. From the **morselib** drop down box, select **MM8108**. Click **Ok**.

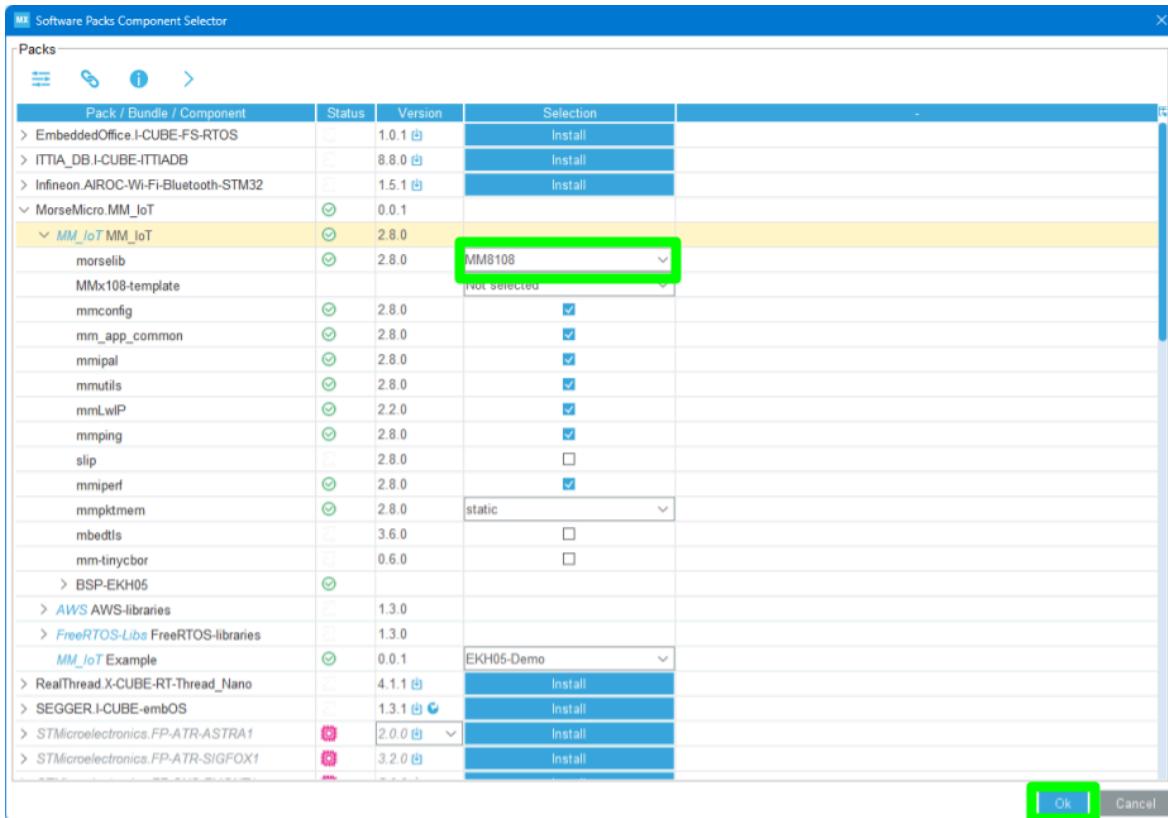


Figure 26. Select chip type in the components selector menu.

Open the **Core**→**Inc**→**main.h** file from the project explorer. Change the #define for **MM\_CHIP** from **MM6108** to **MM8108**, and save the file.

```

77 #define BUSY_IRQ_HANDLER      EXTI5_IRQHandler
78
79 #define LOG_USART           (LPUART1)
80 #define LOG_USART IRQn      (LPUART1_IRQn)
81 #define LOG_USART_IRQ_HANDLER LPUART1_IRQHandler
82
83 #define MM6108 1
84 #define MM6108_2
85 #define MM_CHIP MM8108
86
87 #if (MM_CHIP == MM6108)
88 #define BCF_VFEM_THRESHOLD 9500
89 #define BCF_DATA_3V3          bcf_aw_hm593
90 #define BCF_DATA_3V3_LEN      bcf_aw_hm593_len
91 #define BCF_DATA_4V3          bcf_aw_hm593_4v3
92 #define BCF_DATA_4V3_LEN      bcf_aw_hm593_4v3_len
93 #elif (MM_CHIP == MM8108)
94 #define BCF_DATA          bcf_mfl5457
95 #define BCF_DATA_LEN        bcf_mfl5457_len
96 #else
97 #error "Invalid MM_CHIP selection. Available options are MM6108 and MM8108"

```

Figure 27. Select chip type in the main.h file.

Click **Project, Generate Code**. This may also start a download of the STM32U5 firmware if this is not downloaded already.

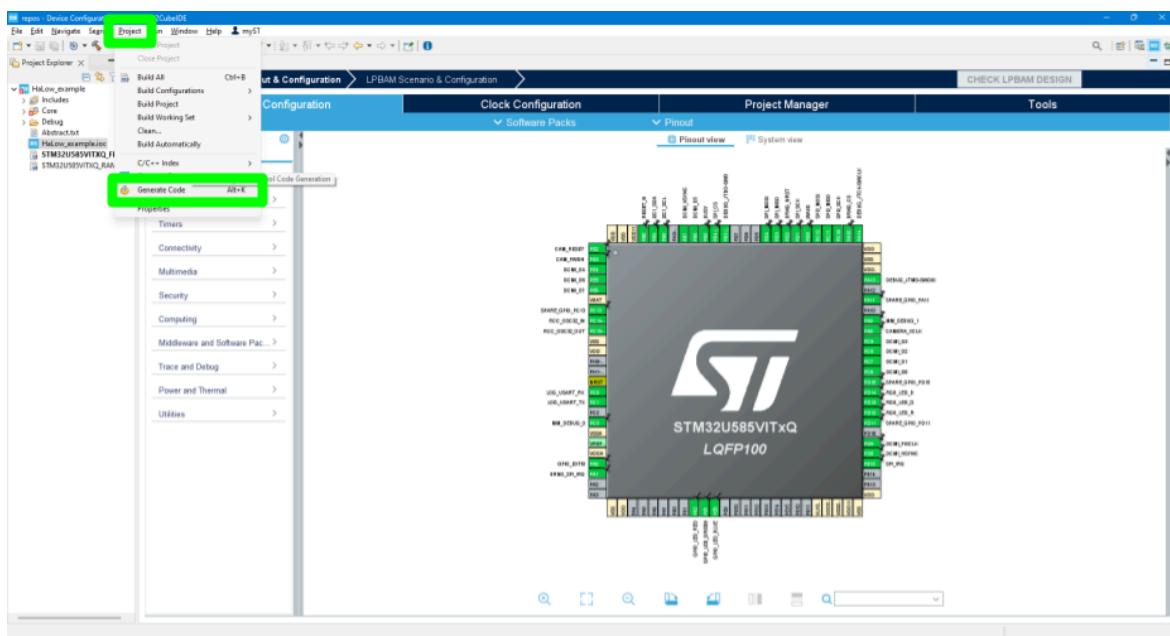


Figure 28. Highlighting where to generate code from after opening the .ioc file.

The example can then be built by clicking the hammer icon in the toolbar. If successful, there should be no errors in the Console window:

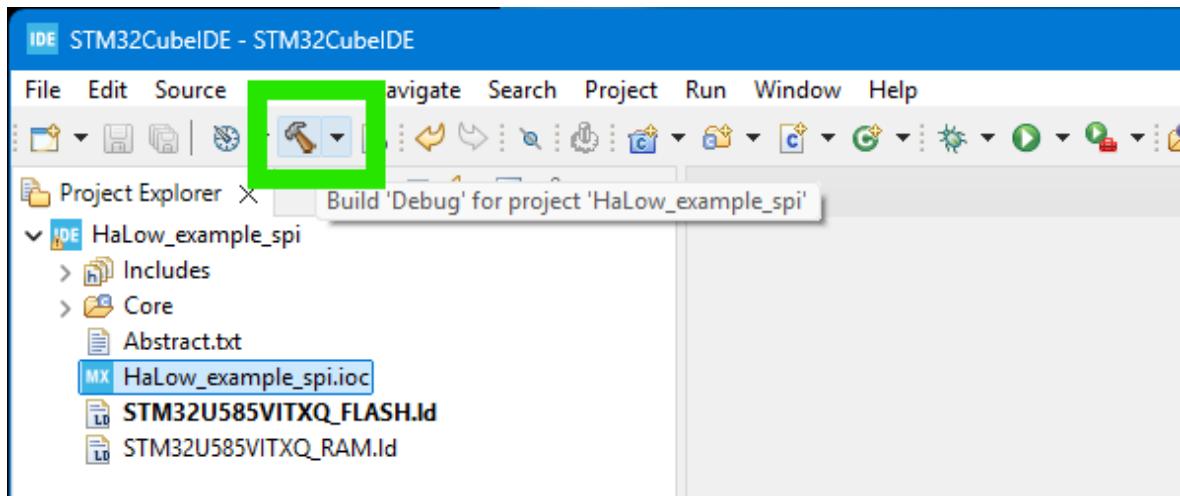


Figure 29. Icon in STM32CubeIDE for building application.

The program can now be run by clicking on the green Run icon:

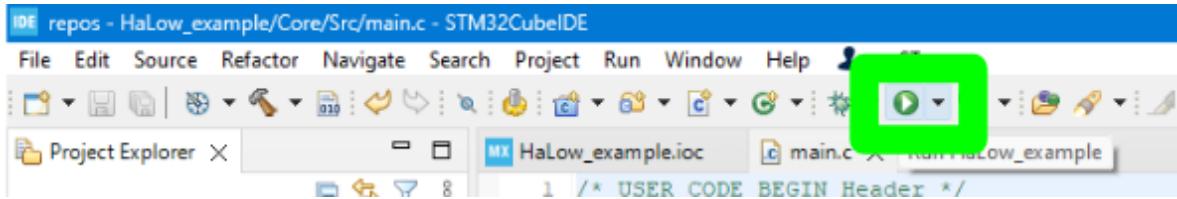


Figure 30. Icon in STM32CubeIDE for running application.

This will then bring up a window for the Run Configuration. All settings can be left as default. Click **Ok**:

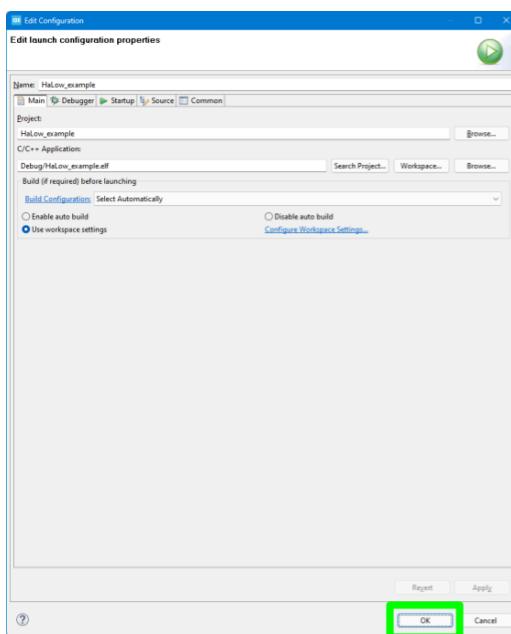


Figure 31. Run configuration menu.

The example program will now build again if not built already, and then be flashed to the device.

## 5.2.1 UART Output

The UART output can be monitored by opening a serial monitor with the following settings:

Item	Setting
Baud Rate	115200
Hardware Flow Control	None
Data Bits	8
Stop Bits	1
Parity	None

### 5.2.1.1 Windows

The STLINK-V3MODS will show up as a Virtual COM port on windows machines.

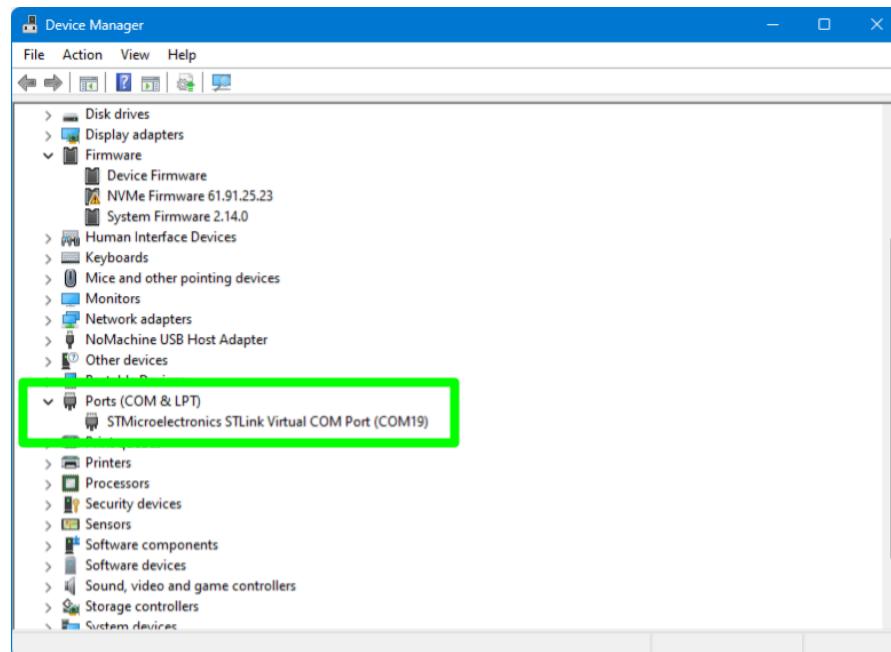


Figure 32. Device manager on windows machine showing where the STLINK-V3 COM port is located.

This COM port can then be opened with tools such as PuTTY.

```
putty -serial COM10 -sercfg 115200
```

### 5.2.1.2 Linux

On Linux machines, the device can be detected with the following command:

```
ls /dev/serial/by-id/usb-STMicroelectronics_STLINK-V3_*
```

And a serial terminal can be opened with tools such as minicom:

```
minicom -D /dev/serial/by-id/usb-STMicroelectronics_STLINK-V3_*
-b 115200
```

### 5.2.1.3 Mac OS

On Mac machines, the device can be detected with the following command:

```
ls /dev/tty.*
```

And similarly, a tool such as minicom can be used to open the serial terminal:

```
minicom -D /dev/tty<YOUR-DEVICE-HERE> -b 115200
```

## 5.3 Changing Example Application

To change the example application, open the HaLow\_example.ioc file and click **Select Components** from the **Software Packs** menu.

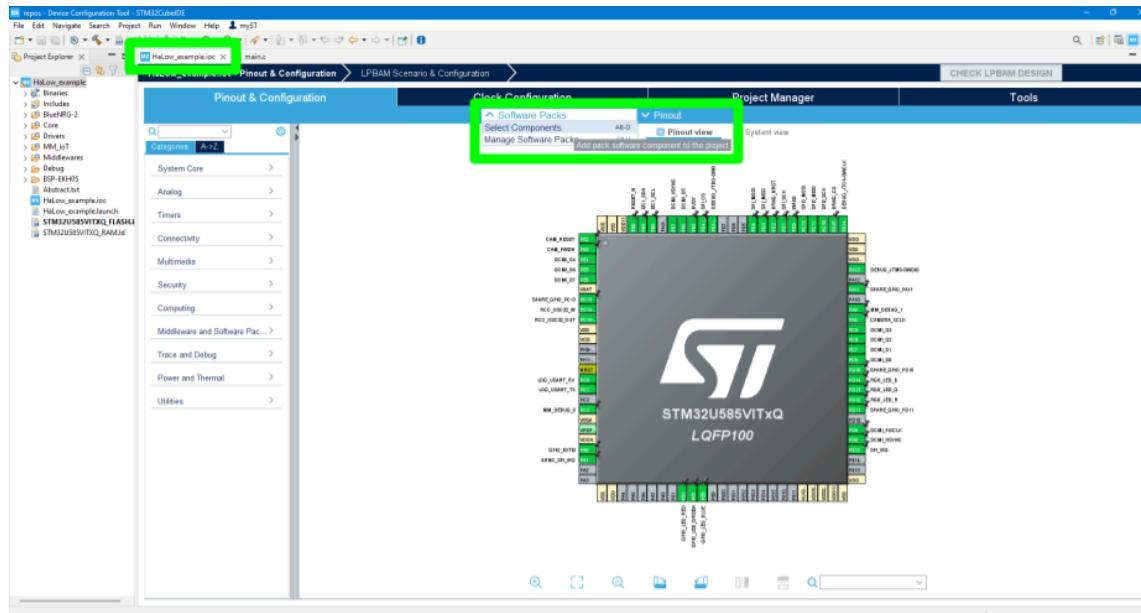


Figure 33. Select components menu in the .ioc file.

Click on the MorseMicro.MM\_IoT arrow, and then select the desired application from the drop down list. Please note that the HTTP server is only available on the EKH05-Demo application. Changing to a different application will mean the user will be unable to connect to the IP address of the MM8108-EKH05 in a web browser.

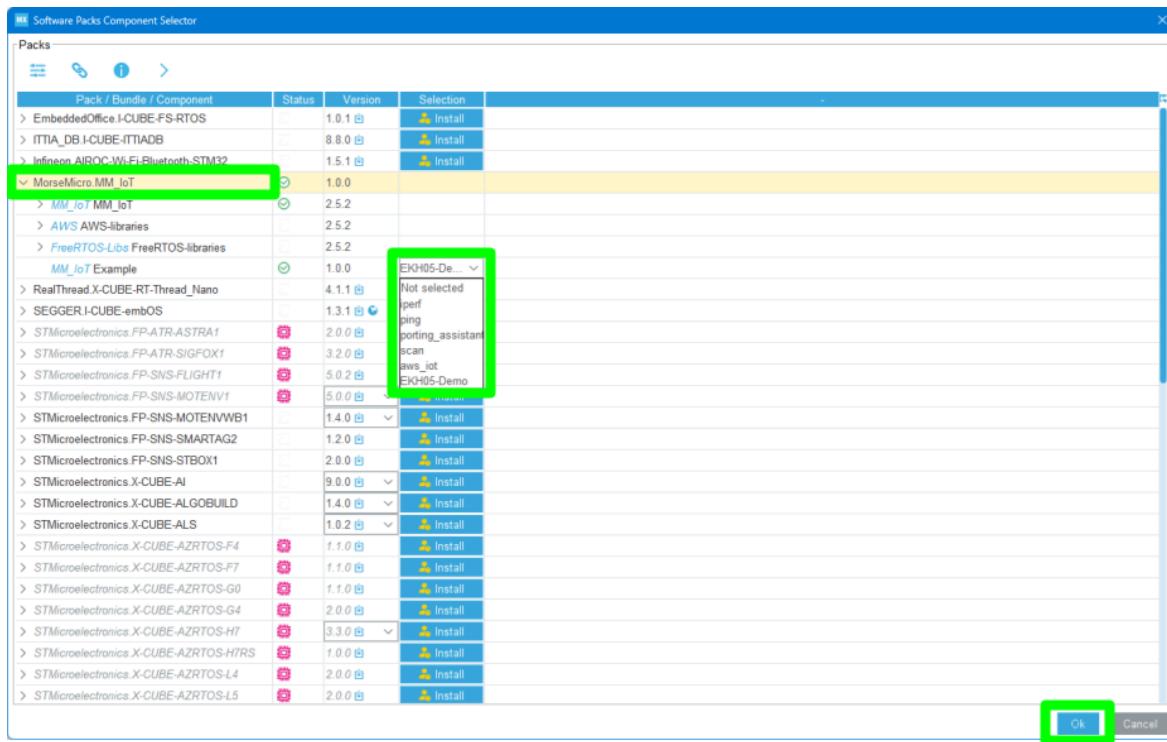


Figure 34. List of example applications within the CMSIS pack.

Click **Generate Code** from the **Project** menu again. The **Run** button can then be clicked again to build and run the new example.

### 5.3.1 rf-test Example Considerations

In case of selecting the rf-test application, a global define for DISABLE\_UART\_LOG is needed. To do so, open the project properties from **Project->Properties**:

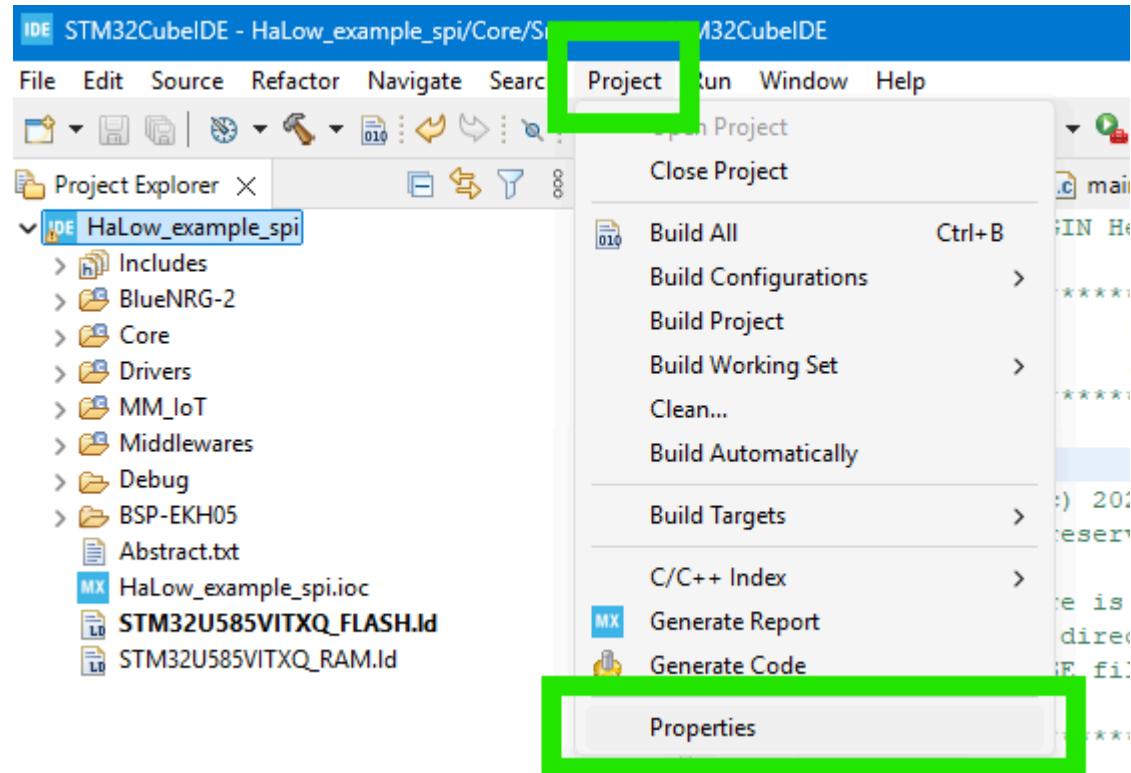


Figure 35. Project properties menu in STM32CubeIDE.

From **C/C++ Build -> Settings** select **Tools Settings** tab and from **MCU GCC Compiler -> Preprocessor** click on “**Add...**” button. In the text box add “**DISABLE\_UART\_LOG=1**” and click OK:

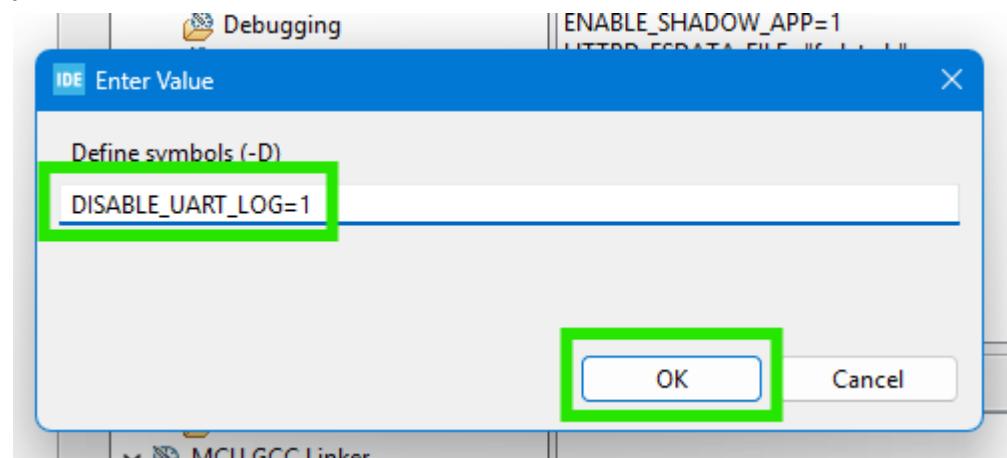


Figure 36. Adding the symbol necessary to build the rf-test application.

Click on **Apply and Close** and rebuild the project.

## 5.4 Changing Country Code

The country code parameter can be changed by navigating to the **Core → Src → configs.c** file. Search for “wlan.country\_code”. Replace the parameter and save the file. The changes will take effect upon the next build of the firmware.

```

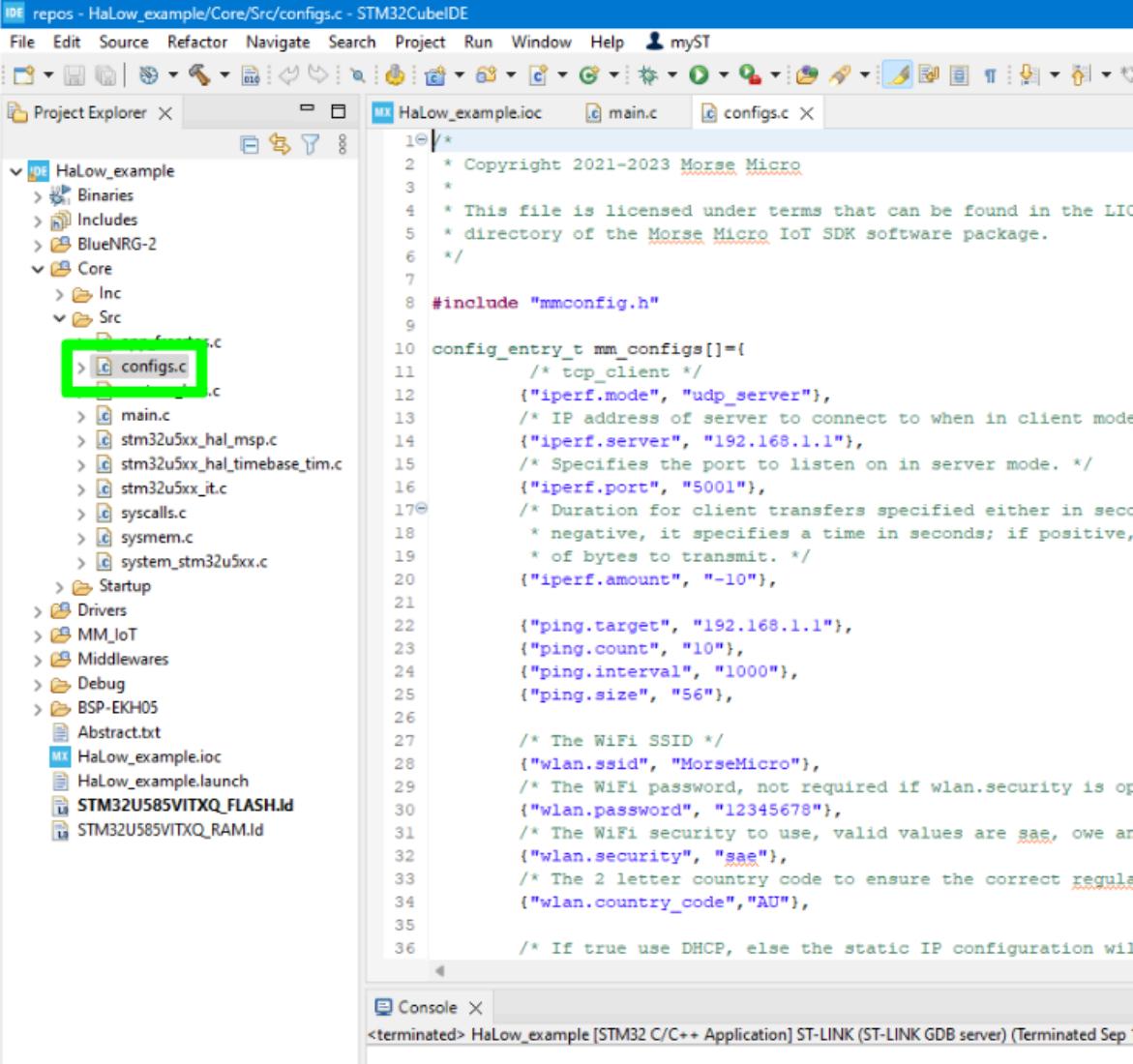
22     {"ping.target", "192.168.1.1"},  
23     {"ping.count", "10"},  
24     {"ping.interval", "1000"},  
25     {"ping.size", "56"},  
26     {"ping.idle_time", "60"},  
27  
28     /* The following setting is required only for the wnm_sleep example */  
29     {"wlan.wnm_sleep_duration_ms", "20000"},  
30  
31     /* The WiFi SSID */  
32     {"wlan.ssid", "MorseMicro"},  
33     /* The WiFi password, not required if wlan.security is open */  
34     {"wlan.password", "12345678"},  
35     /* The WiFi security to use, valid values are sae, owe and open */  
36     {"wlan.security", "sae"},  
37  
38     /* The 2 letter country code to ensure the correct regulatory domain is used */  
39     ["wlan.country_code","US"],  
40  
41     /* If true use DHCP, else the static IP configuration will be used */  
42     {"ip.dhcp_enabled", "true"},  
43  
44     /* These settings are ignored if DHCP is used, but mandatory if using static IP */  
45     {"ip.ip_addr", "192.168.1.2"},  
46     {"ip.netmask", "255.255.255.0"},  
47     {"ip.gateway", "192.168.1.1"},  
48  
49     /* These settings are for IPv6, ip6.ip_addr is only required if ip6.autoconfig is  
50     * false */  
51     {"ip6.autoconfig", "true"},  
52     // "ip6.ip_addr": "FE80::2"  
53  
54     /* Note: The following settings should be used only when explicitly required */

```

Figure 37. Source code location to change the country code.

## 5.5 Changing Other Example Configurations

Other parameters like SSID, passphrase, IP addresses, ping count etc can also be changed by navigating to the **Core → Src → configs.c** file. Replace the parameter and save the file. The changes will take effect upon the next build of the firmware.



The screenshot shows the STM32CubeIDE interface. The Project Explorer on the left lists the project structure under 'HaLow\_example'. The 'Configs' folder contains the 'configs.c' file, which is highlighted with a green box. The main editor window displays the code for 'configs.c', showing various configuration parameters like WiFi SSID, password, and port settings. The bottom console window shows the build status: '<terminated> HaLow\_example [STM32 C/C++ Application] ST-LINK (ST-LINK GDB server) (Terminated Sep 1)'.

```

1 /*
2 * Copyright 2021-2023 Morse Micro
3 *
4 * This file is licensed under terms that can be found in the LIC
5 * directory of the Morse Micro IoT SDK software package.
6 */
7
8 #include "mmconfig.h"
9
10 config_entry_t mm_configs[] = {
11     /* tcp_client */
12     {"iperf.mode", "udp_server"}, // Set to "tcp_client" for TCP client mode
13     /* IP address of server to connect to when in client mode */
14     {"iperf.server", "192.168.1.1"}, // Set to the target IP address
15     /* Specifies the port to listen on in server mode. */
16     {"iperf.port", "5001"}, // Set to the desired port number
17     /* Duration for client transfers specified either in seconds
18      * negative, it specifies a time in seconds; if positive,
19      * of bytes to transmit. */
20     {"iperf.amount", "-10"}, // Set to the desired amount of data to transmit
21
22     {"ping.target", "192.168.1.1"}, // Set to the target IP address
23     {"ping.count", "10"}, // Set to the number of ping requests
24     {"ping.interval", "1000"}, // Set to the interval between ping requests
25     {"ping.size", "56"}, // Set to the size of the ping packet
26
27     /* The WiFi SSID */
28     {"wlan.ssid", "MorseMicro"}, // Set to your WiFi SSID
29     /* The WiFi password, not required if wlan.security is open */
30     {"wlan.password", "12345678"}, // Set to your WiFi password
31     /* The WiFi security to use, valid values are sae, wpa2, or open */
32     {"wlan.security", "sae"}, // Set to the desired WiFi security
33     /* The 2 letter country code to ensure the correct regulations */
34     {"wlan.country_code", "AU"}, // Set to the correct country code
35
36     /* If true use DHCP, else the static IP configuration will be used */
37 }

```

Figure 38. Showing all configurations available.

## 5.6 Changing Between SPI and SDIO

In order to change between using SPI and SDIO as the communication protocol between the STM32U585 and the HaLow module, a different project is needed. Either the HaLow\_example\_spi project or HaLow\_example\_sdio project can be selected, as shown in section [5.2](#). Note that hardware changes are also required to function properly. These changes are described in section [6.4](#).

## 5.7 Changing Network Stacks

This section will document how to change between using LwIP and FreeRTOS+TCP as the network stack. For almost all situations, LwIP should be used. FreeRTOS+TCP should only be used when validating the best low-power performance. Not all examples are compatible with FreeRTOS+TCP.

Open the HaLow\_example.ioc file and click **Select Components** from the **Software Packs** menu.

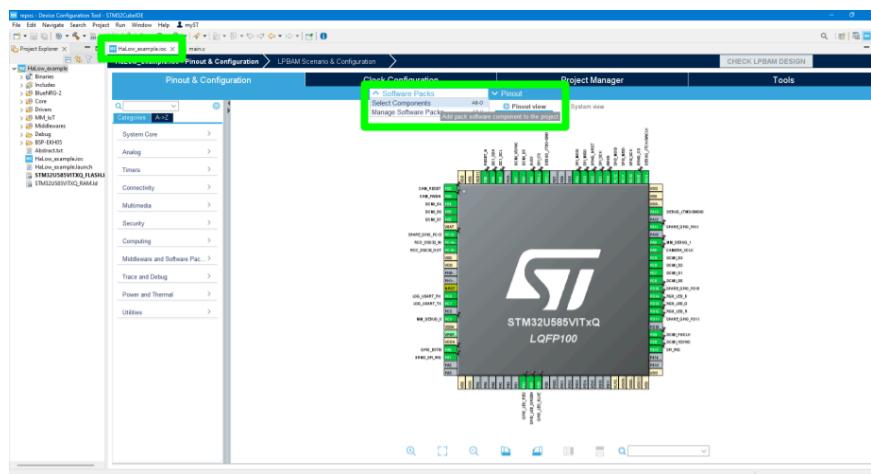


Figure 39. Select components menu in the .ioc file.

Select the **MM\_IoT** and **FreeRTOS-Libs** drop down menus. Select the desired network stack and ensure the other is deselected. There should be all green ticks if there are no issues.

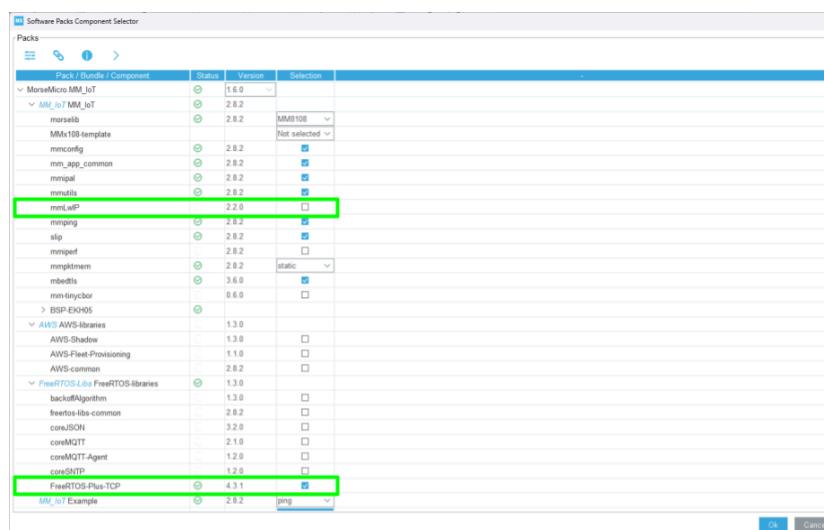


Figure 40. Unchecking the mmLwIP box, checking the FreeRTOS-Plus-TCP box.

Next, click the **Middleware and Software Packs** menu from the .ioc file. Select **MM\_IoT** and ensure the **FreeRTOS-Libs FreeRTOS-libraries** checkbox is ticked.

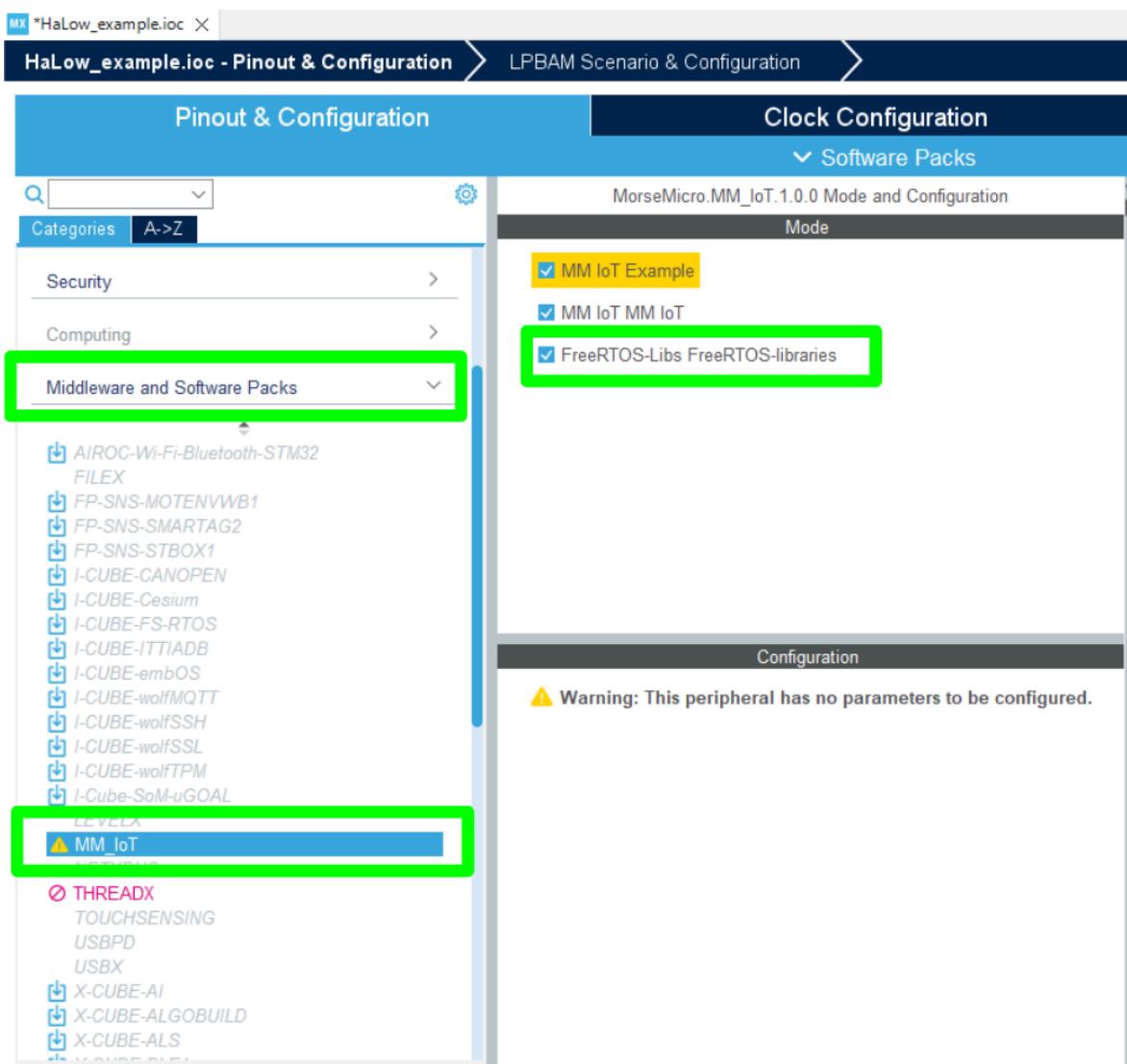


Figure 41. Middleware and software packs menu to add FreeRTOS libraries.

The code will need to be regenerated, and the firmware rebuilt and uploaded to the device for the changes to take effect.

## 5.8 Updating BLUENRG-M2SP Module Firmware

The most straightforward way to flash the firmware on the BLUENRG-M2SP module is to use the on board STLINK V3. The SWCLK and SWDIO jumpers can be removed, and wires connecting to the BLE SWD headers can be attached as shown below.

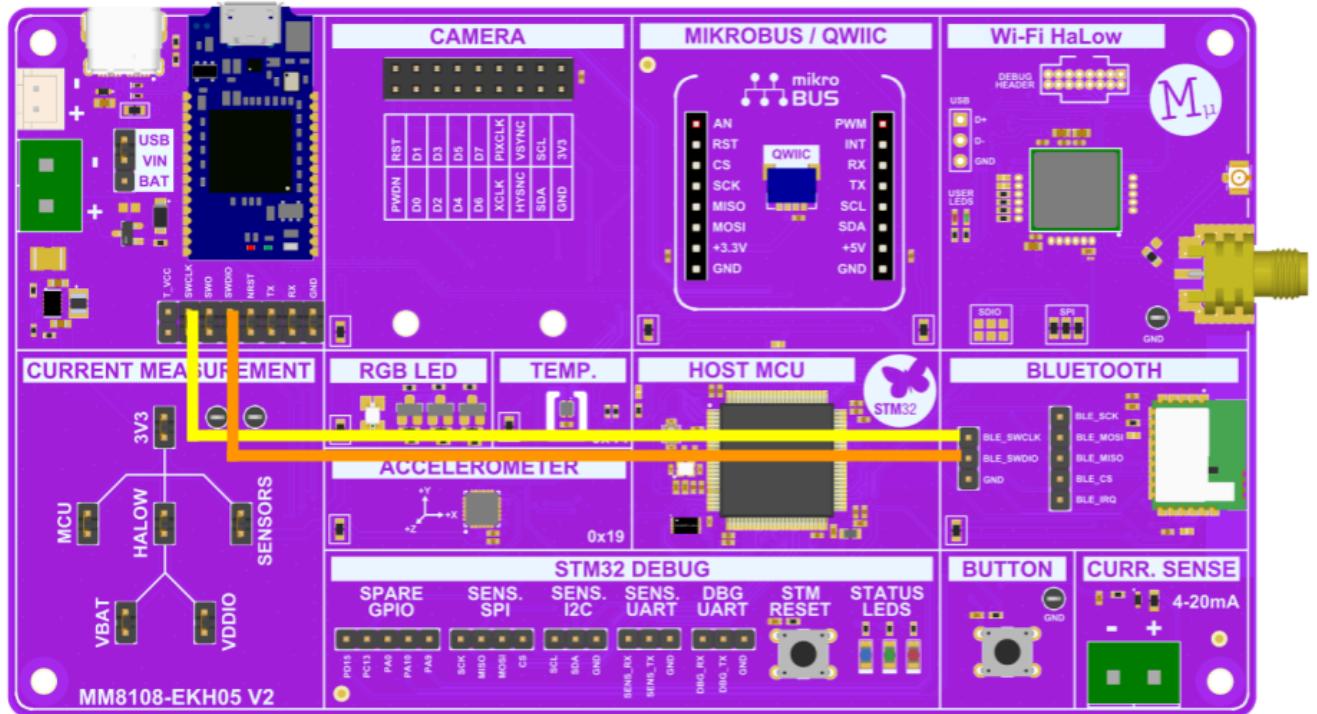


Figure 42. Connecting the STLINK V3 SWCLK and SWDIO pins to the BLUENRG-M2SP module.

Ensure the wires are attached to the top pins of the header as these are connected to the STLINK V3.

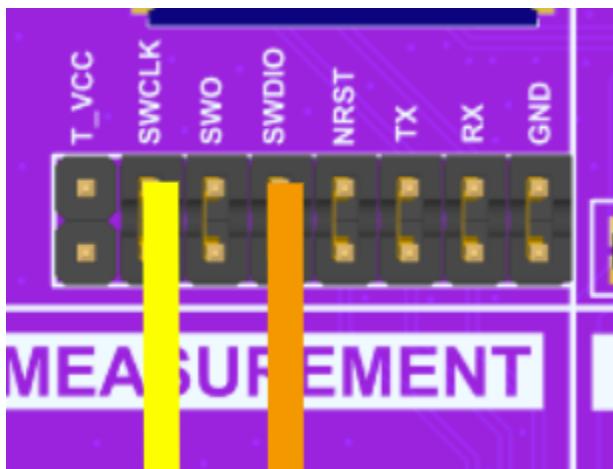


Figure 43. Close up view of connections, wires must be placed on top pins of the header.

Before doing this, the BLE RESET line must be held high by the STM32U585. Please ensure that pin PD2 is configured as an output, and is set to high. The BLUENRG-M2SP module will be held in reset if this is not done, and will not be able to be programmed.

The next step is to download the RF-Flasher Utility from ST. This can be found here - <https://www.st.com/en/embedded-software/stsw-bnrgflasher.html>

Once installed, open the program and do the following:

1. Click on the **SWD** tab.
2. Select your connected STLINK device
3. Click **Select Image File**. Select the firmware image file to load.
4. Click **Flash**.

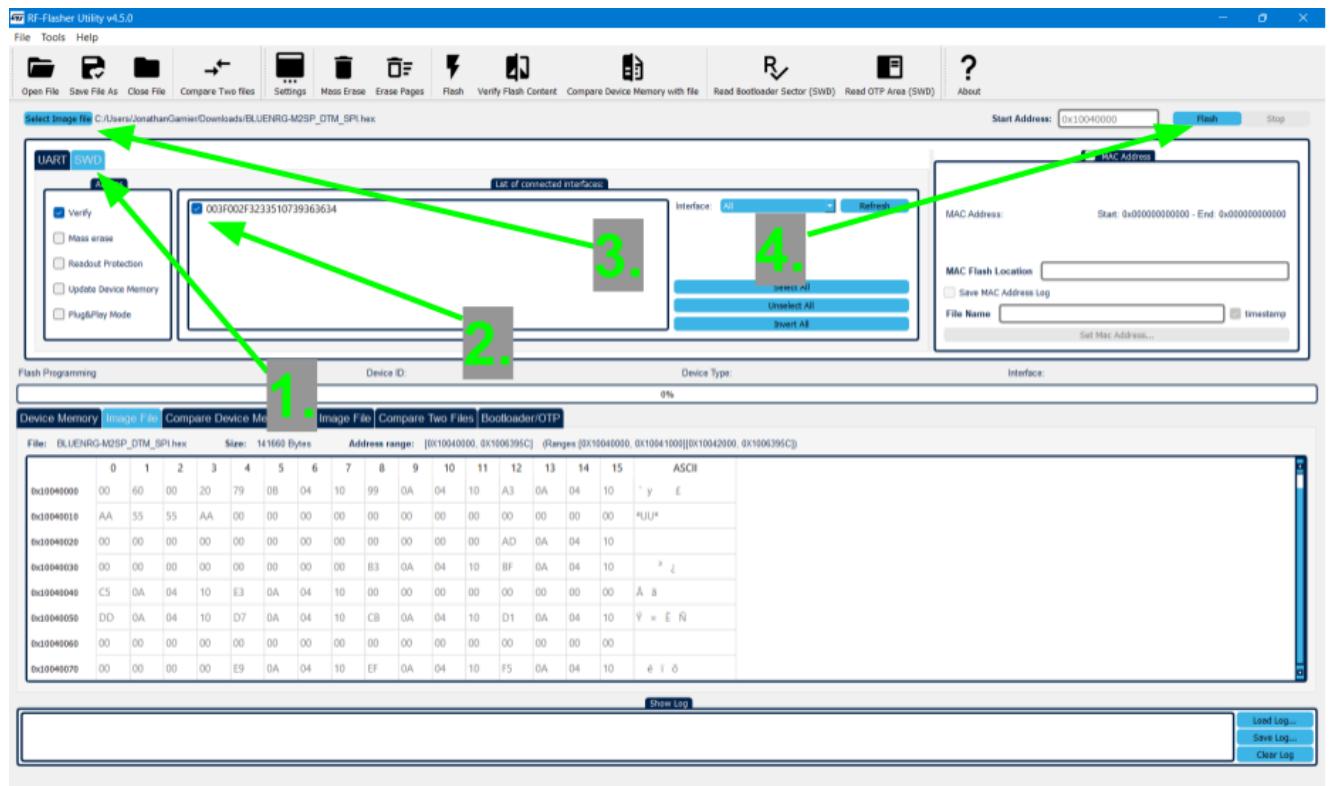


Figure 44. Steps in the RF-Flasher utility to program new firmware to the BLUENRG-M2SP module.

To verify this operation was successful, click the **Verify Flash Content** button. A popup window will appear notifying if the memory content matches the file or not.

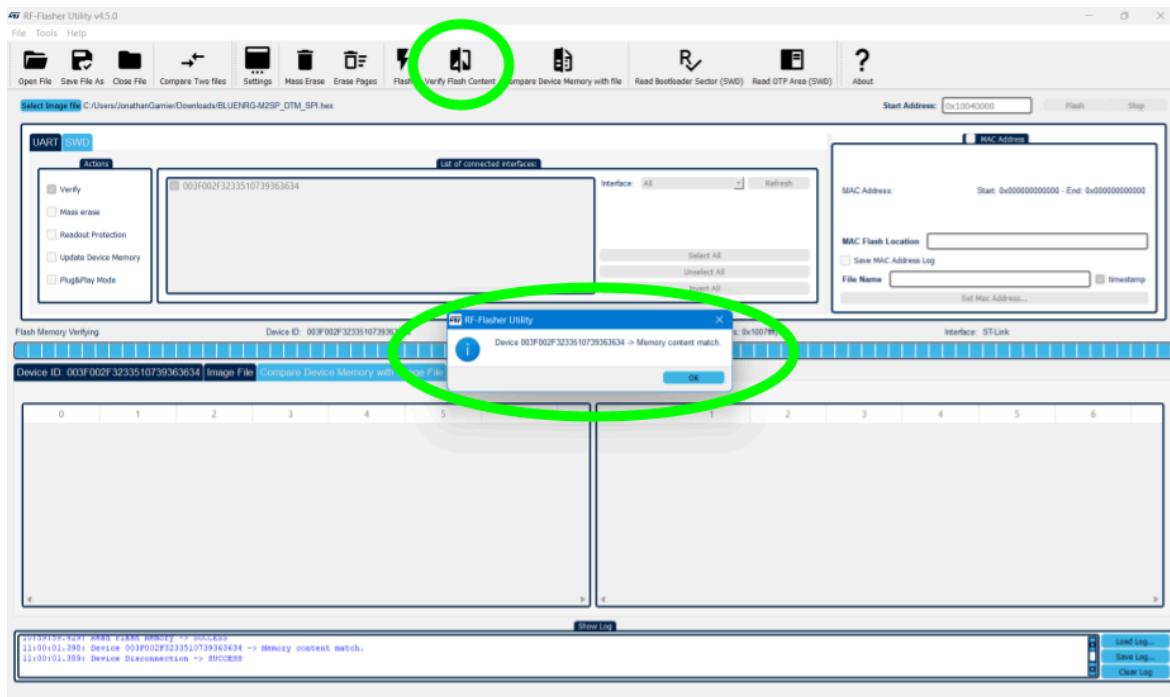


Figure 45. Verify firmware flash was successful.

# 6 Hardware Layout and Configuration

## 6.1 Power Selection

The voltage supply input to the buck-boost regulators can come from multiple sources. If the jumper is placed on the USB and VIN header, the supply can come from either the USB-C or STLINK Micro USB connectors. If the jumper is placed on the BAT and VIN header, the supply will come from the battery connections.

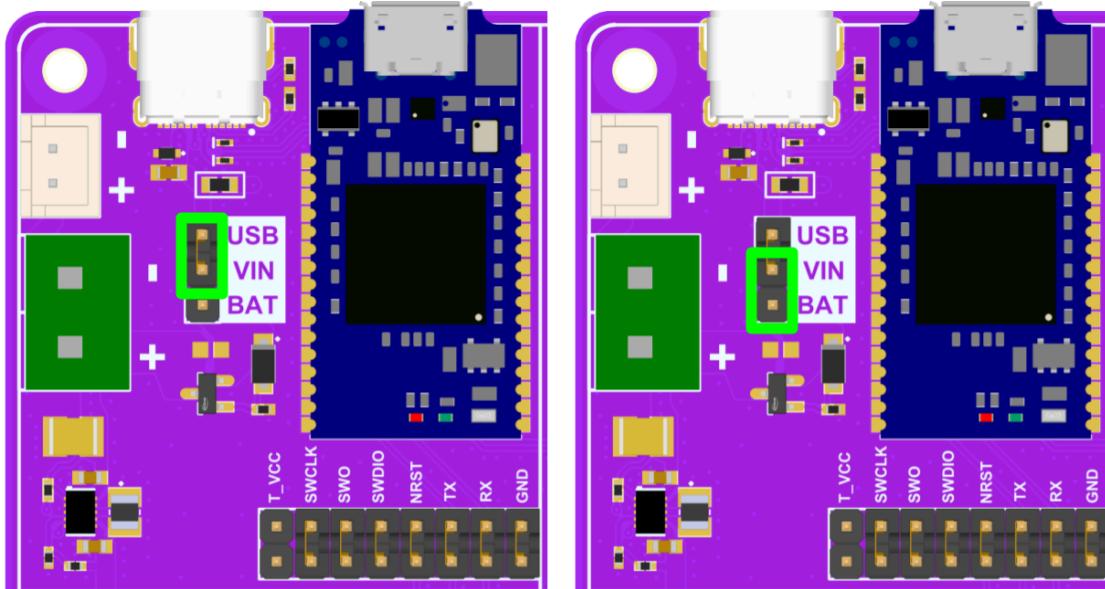


Figure 47. Changing power input between 5V USB on left (from USB-C or STLINK-V3) and battery inputs on right (JST connector or battery screw terminal).

## 6.2 Using an External Debugger/Programmer

The STLINK V3 MODS is a modular in-circuit debugger and programmer for STM32 and STM8 microcontrollers. It programs the device through a Single Wire Debug (SWD) interface, and connects to a UART interface on the STM32U585 microcontroller in the one cable.

Developers can opt to use their own external debug tool if desired. The jumpers below the module can be removed, and the bottom side of the headers (circled in green) can be attached to with an external debug tool to connect to the relevant pins on the STM32U585 microcontroller. The T\_VCC header (circled in red) connects to 3.3V. The target voltage pin of an external debug tool can be attached here.

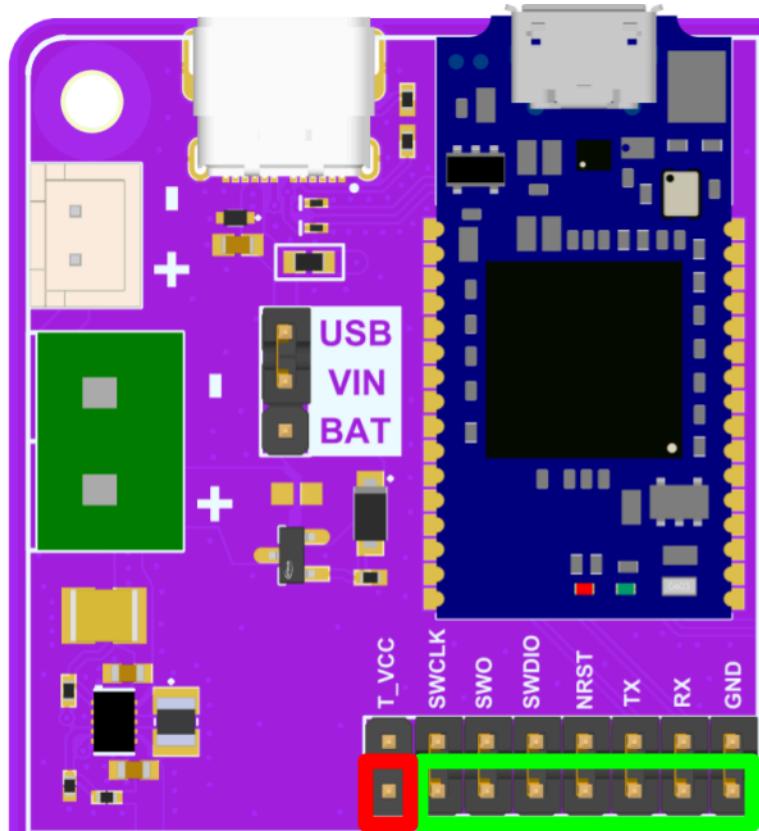


Figure 48. How to attach an external programmer.

## 6.4 Switching Between SDIO and SPI

The MM8108-EKH05 gives users the option to interface the STM32U585 Microcontroller with the HaLow module either through SDIO or SPI. SPI is the default

configuration. If using SDIO, 47 kOhm pull-up resistors must be populated as pointed out by the arrow below. For SPI, these pull up resistors are not needed for the protocol and will cause **extra leakage current** on the VDDIO rail if not removed. The pull up resistor on the SD\_D2 line **must be populated**.

A different pin configuration in the software is also needed when changing between the two. Section 5.6 details how to configure this.

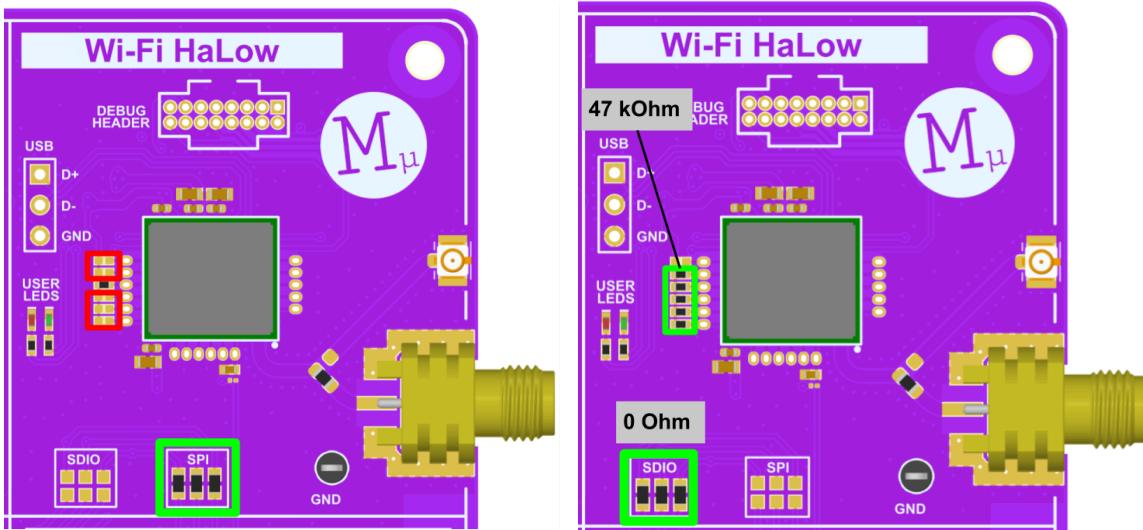


Figure 50. SPI configuration (left) and SDIO configuration (right).

## 6.5 Switching Between SMA and U.FL Connector

The user has the option to use either a SMA or U.FL connector for the HaLow antenna. The two can be switched between by changing the orientation of a 0 ohm resistor.

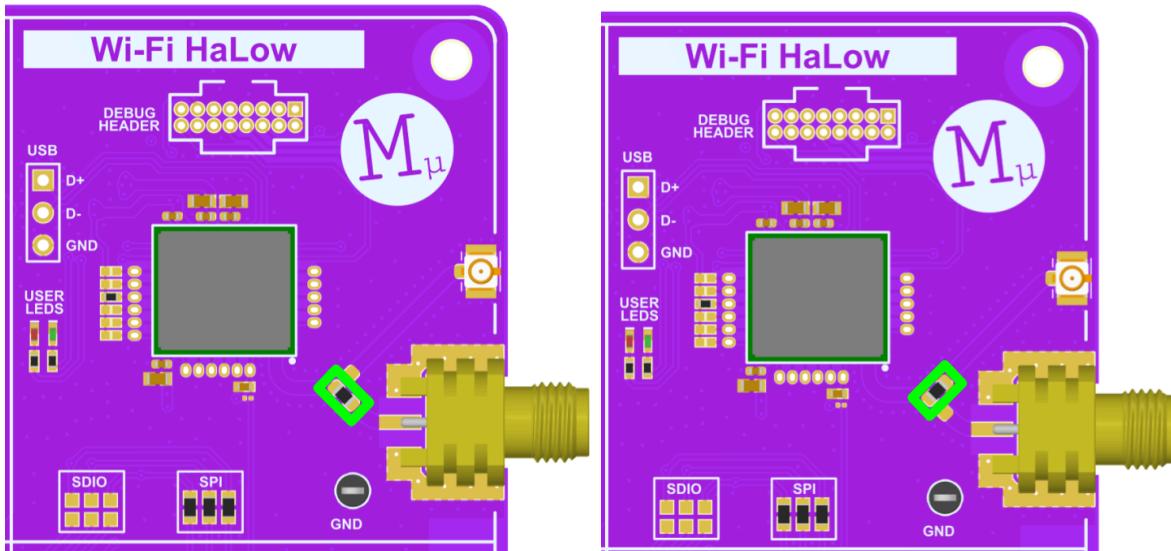


Figure 52. SMA connection (left) and U.FL connection (right).

## 6.6 Disconnecting Sensors

The sensor blocks on the MM8108-EKH05 feature a 0 ohm resistor that supplies power from the SENSOR header in the power tree to each individual sensor. These sensors include the temperature and humidity sensor, accelerometer, RGB LED, camera header, mikroBUS connector (3.3V on the left, 5V on the right), QWIIC connector, and the BLE module. By removing these 0 ohm resistors, the power can be cut off to the corresponding sensor individually. All these resistors are populated by default.

Please note to achieve optimal power consumption, it may be necessary to upload new firmware that ensures any I/Os connected to the STM32U585—which are now unpowered—are not actively driven.

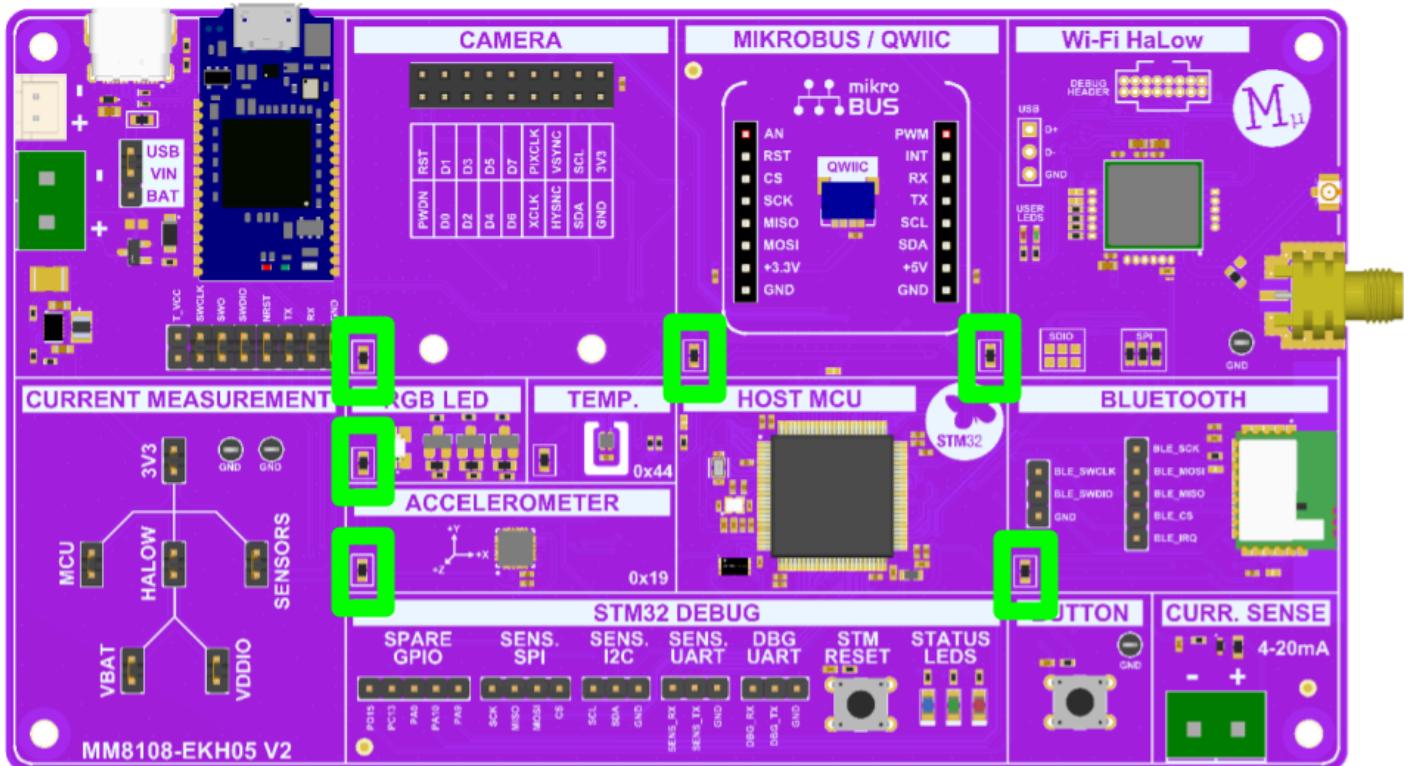


Figure 53. 0 ohm links which connect power to each sensor.

# 7 Power Consumption Measurements

## 7.1 Power Consumption Measurement Points

### 7.1.1 General Structure

The MM8108-EKH05 is designed to make key power consumption measurements straightforward. Its power distribution follows a tree structure, with each sub-power section branching from the main supply. For instance, to measure the total current draw on the 3.3V rail, an ammeter can be connected across the 3V3 header. If the user wishes to measure only the power consumption of the MCU for example, the ammeter can instead be placed across the MCU header.

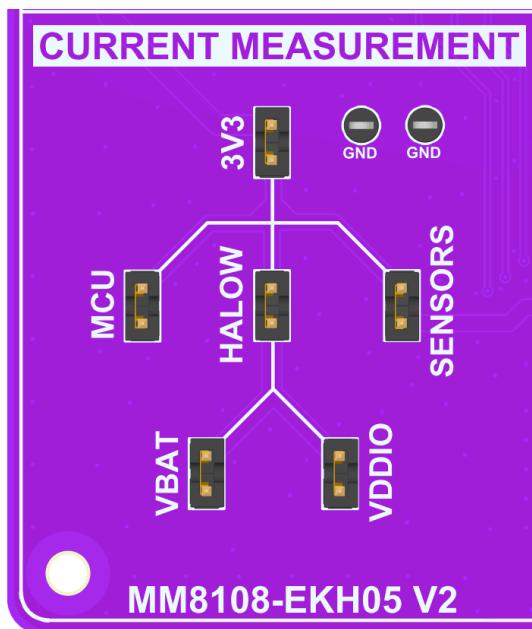


Figure 54. Current measurement tree on MM8108-EKH05.

### 7.1.2 Whole System Power Consumption

To measure the whole system power consumption, including efficiency losses of the 3.3V on-board buck-boost regulator, connect ammeter as per [Figure 55](#).

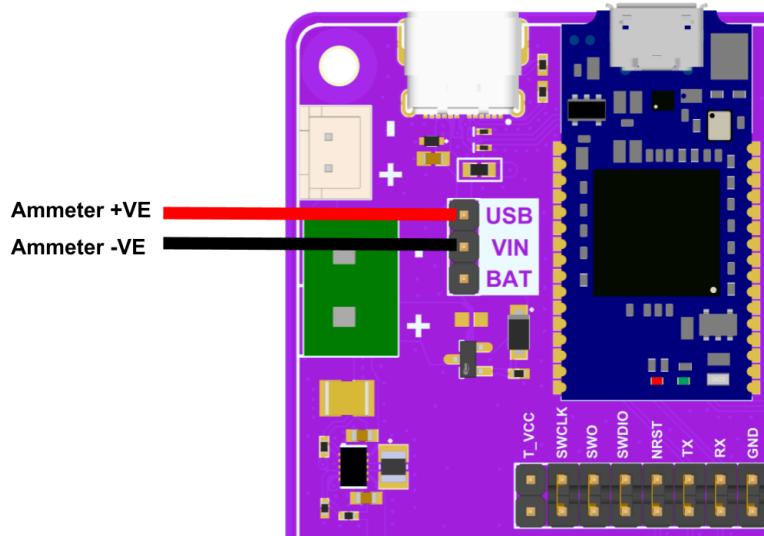


Figure 55. Ammeter positions to measure the entire board including the on-board regulators.

## 7.2 Power Consumption Measurement Procedure

A high-precision power analyzer, such as the Joulescope [JS220](#), is recommended for low power measurements. This device can measure both voltage and current, allowing for a precise calculation of overall power. The suggested configuration is shown below.

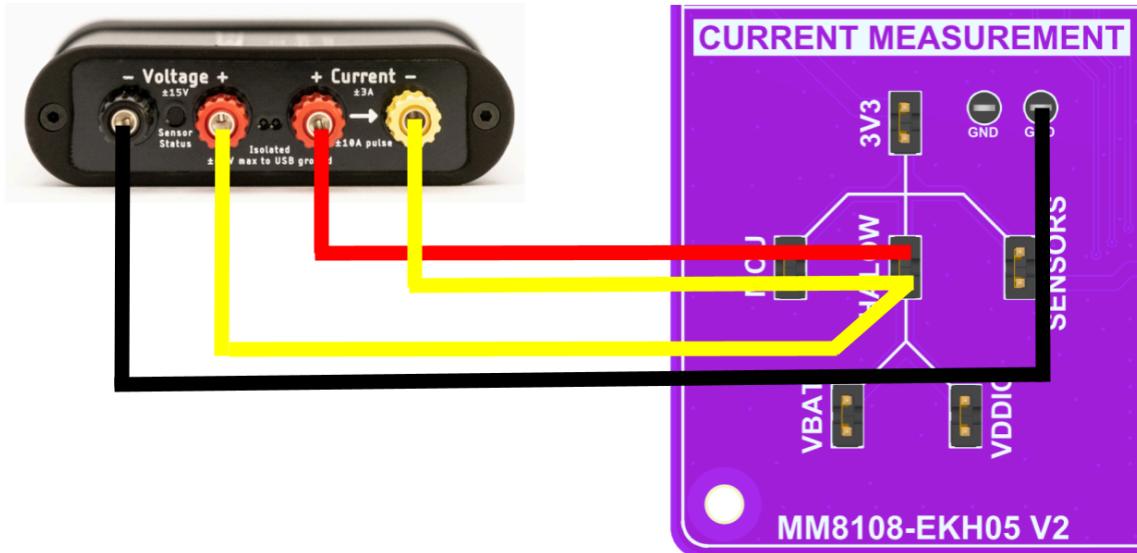


Figure 56. Example joulescope connections measuring HALOW header on MM8108-EKH05.

This configuration can be used for the 3V3, MCU, HALOW, SENSORS, VBAT and VDDIO headers, with the positive ammeter position at the top pin of the header, and the negative at the bottom.

## 7.3 Optional Debug GPIO Connections

Certain example applications utilise PA9 and PA10 of the STM32U585 as debug GPIOs. These debug GPIOs will toggle when the device is transitioning between different states and can be useful for measuring different parts of the example applications as they execute. These GPIOs are broken out to the SPARE GPIO header on the MM8108-EKH05 and can be connected to the Joulescope as follows.

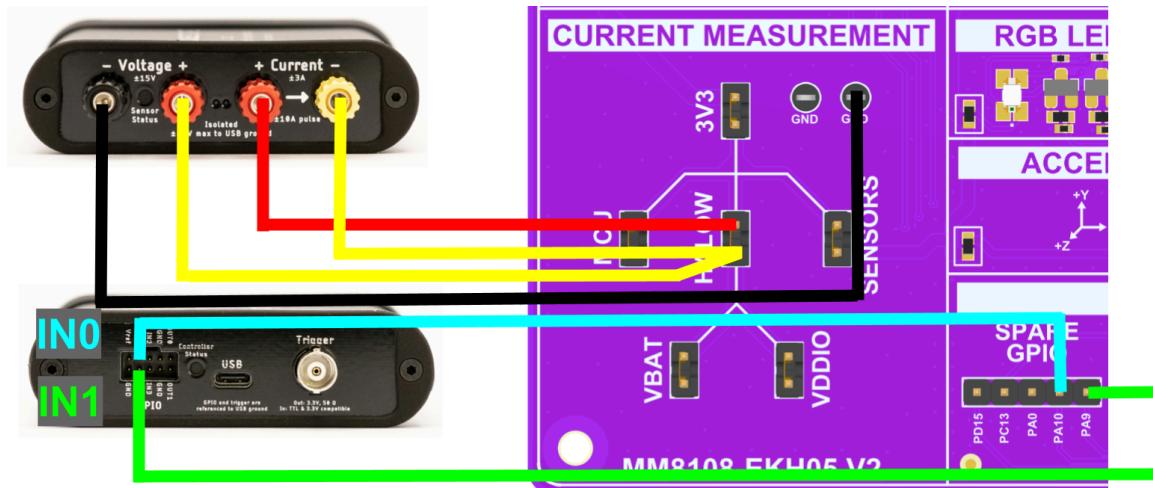


Figure 57. Showing where to connect debug GPIO's from MM6108-EKH05 to a joulescope.

On the Joulescope UI, the toggling debug GPIOs can be seen at the bottom and are annotated over to show the different states in a ping example application.

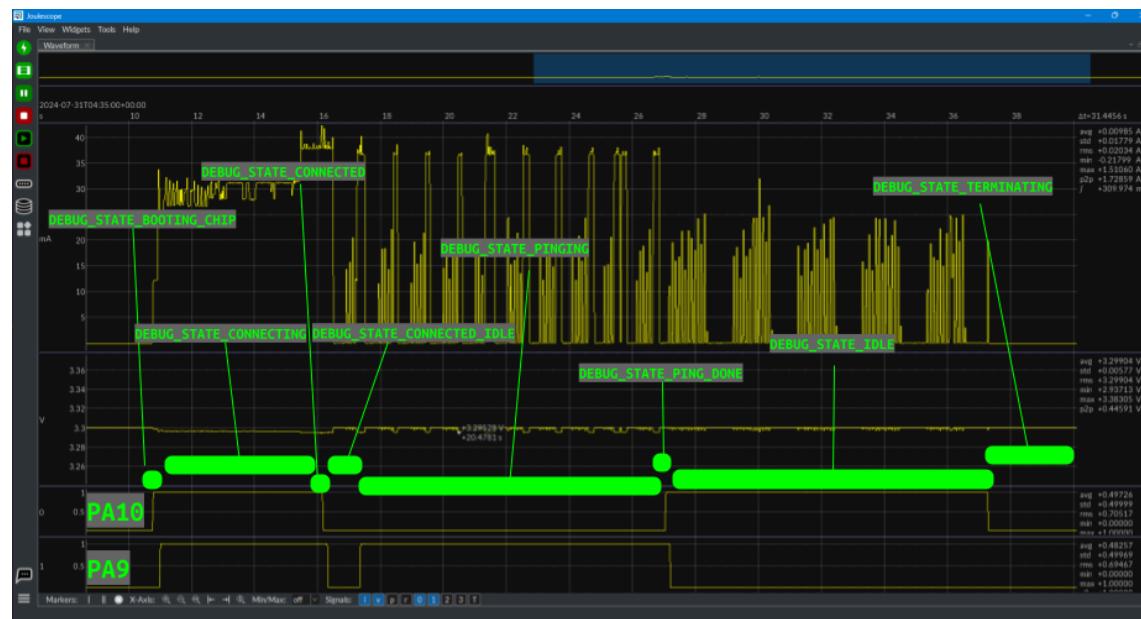


Figure 58. Power consumption profile of the ping app with the states from the debug GPIO's.

Definitions for each debug state can be found in the example folder under each application of interest. For example, the definitions for the ping application can be found in the following locations.

Windows:

```
C:\Users\USER_NAME\STM32Cube\Repository\Packs\MorseMicro\MM_IoT\1.6.0\Example\ping\ping.c
```

Linux:

```
/home/USER_NAME/STM32Cube/Repository/Packs/MorseMicro/MM_IoT/1.6.0/Example/ping/ping.c
```

Mac:

```
/Users/USER_NAME/STM32Cube/Repository/Packs/MorseMicro/MM_IoT/1.6.0/Example/ping/ping.c
```

A search for the **enum** of **debug\_state** will show the definitions. The function **set\_debug\_state** is used to set these throughout the application.

## 7.4 Measuring DTIM Power

To measure DTIM power consumption, use the ping application. Make sure to use the correct country code, BCF, SSID, and passphrase to connect the device to the AP. The DTIM period and bandwidth/channel can also be changed by following the steps in section [4.2.1](#).

The IP address that the MM8108-EKH05 will ping will also need to be set. See section [5.5](#) for how to change example parameters. The **ping.target** parameter needs to be set to 192.168.12.1 to match the default IP address of the HaLowLink 1.

For CMSIS Pack 1.5.0, the **ping.post\_ping\_delay\_ms** parameter should also be changed from 60 to 600000. This will set the device to idle while connected for 10 minutes after pinging to take DTIM measurements.

For the lowest power consumption, it is recommended to use FreeRTOS Plus TCP as the network stack, please see section [5.7](#) to change this.

The ping application will perform the following actions:

- Boot the MM8108 chip.
- Connect to the AP.
- Ping the AP 10 times.
- Remain connected in power save mode while idling for 600 seconds.
- Disconnect from the AP.

Before starting, ensure that the ping application is loaded onto the device and that the device is properly connected to the AP. This can be confirmed by monitoring the UART output. Note that the UART baud rate is set to 115200.

```
Morse Ping Demo (Built Sep 16 2024 13:44:00)

-----  

BCF API version:      6.7.0  

BCF build version:    fbed8cb 244ffd771b  

BCF board description: BAILEY_V12  

Morselib version:     998b9a497_NFP  

Morse firmware version: 1.13.0  

Morse chip ID:        0x0306  

-----  

Initialize IPv4 with static IP: 192.168.1.2...  

Initialize IPv6 using Autoconfig...  

Morse FreeRTOS+ TCP interface initialised. MAC address 94:bb:43:dc:f9:52  

Attempting to connect to MorseMicro with passphrase 12345678  

This may take some time (~30 seconds)  

WLAN STA connecting  

WLAN STA connected  

DNS server 0: 192.168.1.1  

Link is up. Time: 7068 ms, IP: 192.168.1.2, Netmask: 255.255.255.0, Gateway: 192.168.1.1, D1

Ping 192.168.1.1 56(64) bytes of data.  

(192.168.1.1) packets transmitted/received = 2/2, round-trip min/avg/max = 12/12/12 ms  

(192.168.1.1) packets transmitted/received = 3/3, round-trip min/avg/max = 11/11/12 ms  

(192.168.1.1) packets transmitted/received = 4/4, round-trip min/avg/max = 11/11/12 ms  

(192.168.1.1) packets transmitted/received = 5/5, round-trip min/avg/max = 11/11/12 ms  

(192.168.1.1) packets transmitted/received = 6/6, round-trip min/avg/max = 11/11/13 ms  

(192.168.1.1) packets transmitted/received = 7/7, round-trip min/avg/max = 11/11/13 ms  

(192.168.1.1) packets transmitted/received = 8/8, round-trip min/avg/max = 11/11/13 ms  

(192.168.1.1) packets transmitted/received = 9/9, round-trip min/avg/max = 11/11/13 ms  

(192.168.1.1) packets transmitted/received = 10/10, round-trip min/avg/max = 11/11/13 ms

--- 192.168.1.1 ping statistics ---
10 packets transmitted, 10 packets received, 0.000% packet loss
round-trip min/avg/max = 11/11/13 ms
WLAN STA disabled
Link is down. Time: 27868 ms

```

Figure 59. Example UART output from ping application.

The below screenshot shows the power consumption profile as the program executes.

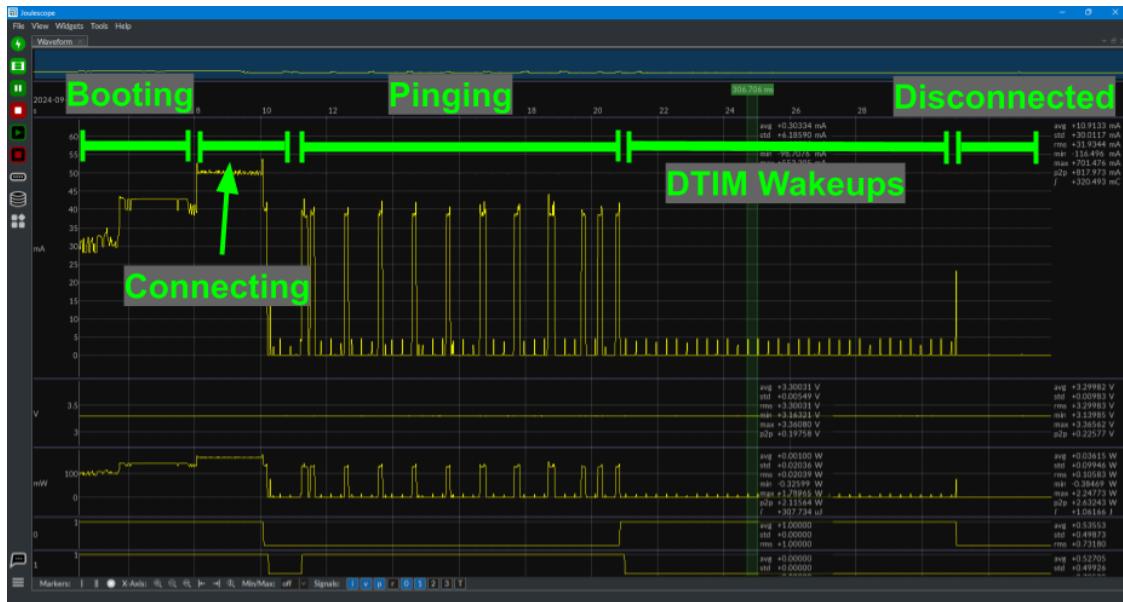


Figure 60. Example current consumption profile from ping application.

The green highlighted section below illustrates the measurement of a DTIM period, zoomed in from the DTIM Wakeups section displayed above. In this example, the measurement is for a DTIM3 with an 8 MHz channel and a 102.4 ms beacon interval. Adjusting these parameters will impact the measured power consumption.

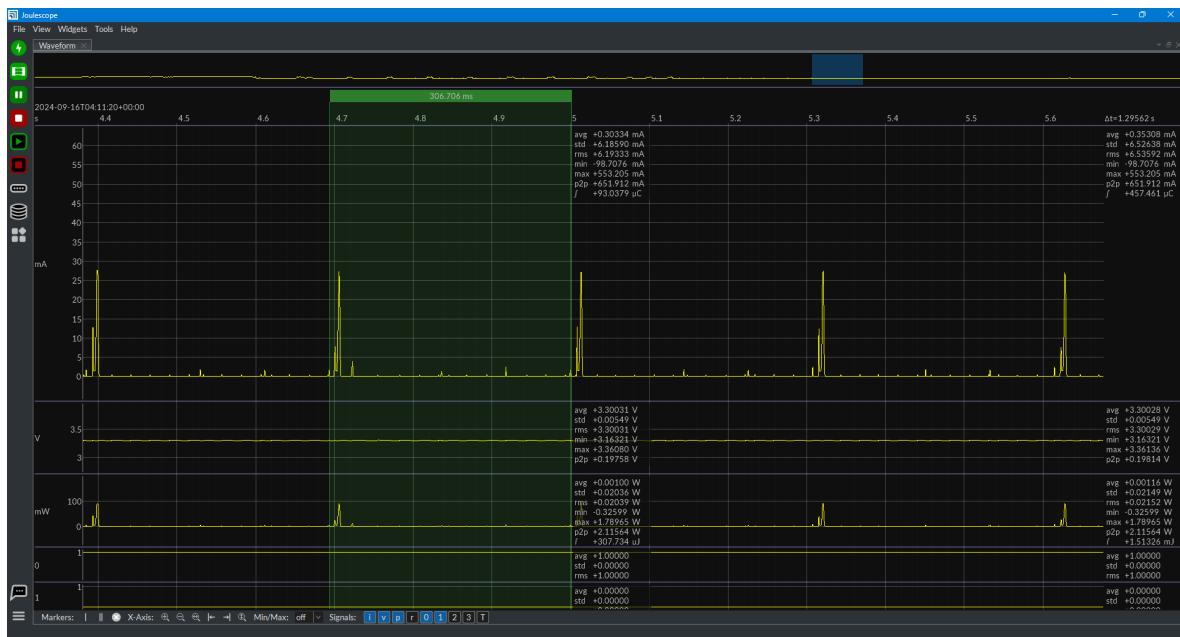


Figure 61. Zoomed in view of a DTIM3 measurement.

## 7.5 Measuring WNM Sleep Power

WNM Sleep provides a more sophisticated sleep mechanism, where the sleep schedule is explicitly negotiated between the station and the AP. This allows for longer sleep intervals and more efficient data retrieval when the client wakes up.

To measure WNM sleep power consumption, use the wnm\_sleep application. Ensure that the correct country code, BCF, SSID, and passphrases are configured so the device can connect to the AP.

Similar to the ping example mentioned in section [7.4](#), the IP address that the MM8108-EKH05 will ping in the WNM sleep example will also need to be set. See section [5.5](#) for how to change example parameters. The ping.target parameter needs to be set to 192.168.12.1 to match the default IP address of the HaLowLink 1.

The wnm\_sleep application will perform the following actions:

- Boot the MM8108 chip.
- Connect to the AP.
- Ping the AP 10 times.
- Enter WNM sleep for 20 seconds.
- Exit WNM sleep.
- Ping the AP another 10 times.
- Enter WNM sleep with chip in shutdown mode for 20 seconds.
- Exit WNM sleep.
- Disconnect from the AP.

As with DTIM measurements, start by verifying that the wnm\_sleep example is correctly loaded and that the device can successfully associate with the AP via the MM8108-EKH05's UART. Below are screenshots of the complete UART output from the program, along with an annotated power trace.

```
Morse WNM Sleep Demo (Built Sep 16 2024 14:15:05)

-----
BCF API version:      6.7.0
BCF build version:   fbed8cb 244ffd771b
BCF board description: BAILEY_V12
Morselib version:    998b9a497_NFP
Morse firmware version: 1.13.0
Morse chip ID:       0x0306

Initialize IPv4 with static IP: 192.168.1.2...
Initialize IPv6 using Autoconfig...
Morse FreeRTOS+ TCP interface initialised. MAC address 94:bb:43:dc:f9:52
Attempting to connect to MorseMicro with passphrase 12345678
This may take some time (~30 seconds)
WLAN STA connecting
WLAN STA connected
DNS server 0: 192.168.1.1
Link is up. Time: 8176 ms, IP: 192.168.1.2, Netmask: 255.255.255.0, Gateway: 192.168.1.1, D1

Ping 192.168.1.1 56(64) bytes of data.
(192.168.1.1) packets transmitted/received = 2/2, round-trip min/avg/max = 12/12/12 ms
(192.168.1.1) packets transmitted/received = 3/3, round-trip min/avg/max = 12/12/12 ms
(192.168.1.1) packets transmitted/received = 4/4, round-trip min/avg/max = 11/11/12 ms
(192.168.1.1) packets transmitted/received = 5/5, round-trip min/avg/max = 11/11/12 ms
(192.168.1.1) packets transmitted/received = 6/6, round-trip min/avg/max = 11/12/14 ms
(192.168.1.1) packets transmitted/received = 7/7, round-trip min/avg/max = 11/12/14 ms
(192.168.1.1) packets transmitted/received = 8/8, round-trip min/avg/max = 11/12/14 ms
(192.168.1.1) packets transmitted/received = 9/9, round-trip min/avg/max = 11/11/14 ms
(192.168.1.1) packets transmitted/received = 10/10, round-trip min/avg/max = 11/11/14 ms

--- 192.168.1.1 ping statistics ---
10 packets transmitted, 10 packets received, 0.000% packet loss
round-trip min/avg/max = 11/11/14 ms

Entering WNM sleep with chip power down disabled
Expected sleep time 20000ms.

Enter WNM sleep took 16 ms.

Exit WNM sleep took 17 ms.
```

Figure 62. Example UART output from wnm\_sleep application.

```

Ping 192.168.1.1 56(64) bytes of data.
(192.168.1.1) packets transmitted/received = 2/2, round-trip min/avg/max = 3/7/11 ms
(192.168.1.1) packets transmitted/received = 3/3, round-trip min/avg/max = 3/9/13 ms
(192.168.1.1) packets transmitted/received = 4/4, round-trip min/avg/max = 3/9/13 ms
(192.168.1.1) packets transmitted/received = 5/5, round-trip min/avg/max = 3/14/35 ms
(192.168.1.1) packets transmitted/received = 6/6, round-trip min/avg/max = 3/14/35 ms
(192.168.1.1) packets transmitted/received = 7/7, round-trip min/avg/max = 3/14/35 ms
(192.168.1.1) packets transmitted/received = 8/8, round-trip min/avg/max = 3/14/35 ms
(192.168.1.1) packets transmitted/received = 9/9, round-trip min/avg/max = 3/14/35 ms
(192.168.1.1) packets transmitted/received = 10/10, round-trip min/avg/max = 3/14/35 ms

--- 192.168.1.1 ping statistics ---
10 packets transmitted, 10 packets received, 0.000% packet loss
round-trip min/avg/max = 3/14/35 ms

Entering WNM sleep with chip power down enabled.
Expected sleep time 20000ms.

Enter WNM sleep took 13 ms.

Exit WNM sleep took 623 ms.
WLAN STA disabled
Link is down. Time: 69345 ms

```

Figure 63. Example UART output from wnm\_sleep application continued.

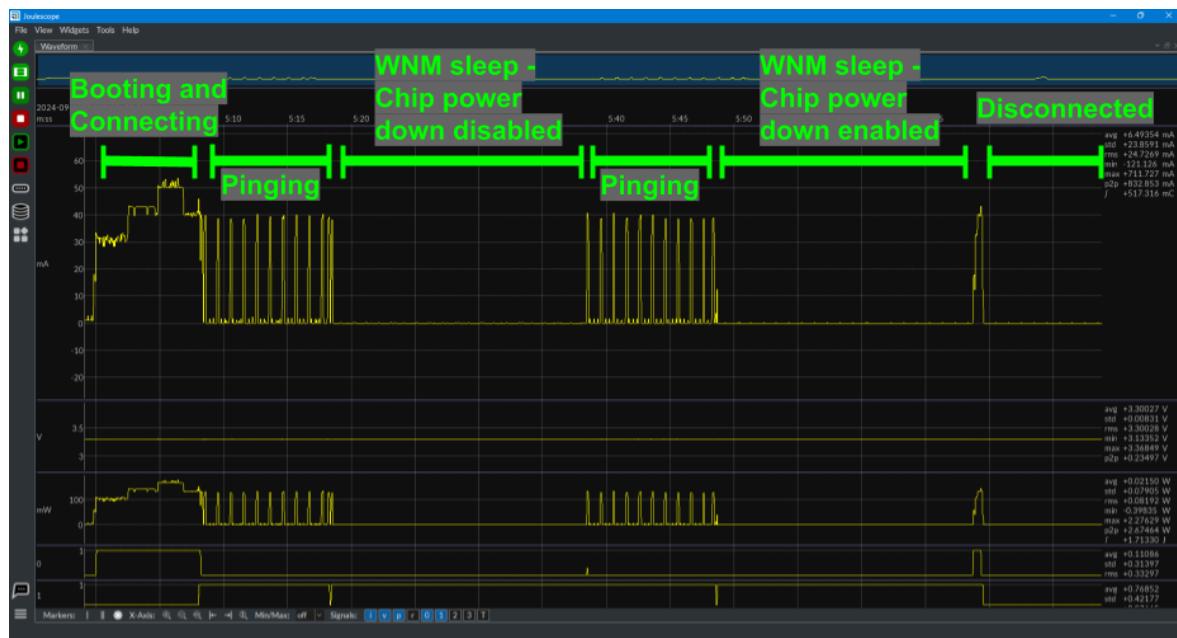


Figure 64. Example current consumption profile from wnm\_sleep application.

## 7.6 Current Spikes When Waking Up

Users may observe a momentary current spike when MM8108 based devices wake up from sleep modes. Typically, these spikes will be less than 200mA. The spikes are expected, and are caused from the operation of the internal buck converter. The buck converter generates ~1.4 V during active operation to supply the internal chip power rails. In power save mode, the chip enters a sleep mode where the Vbuck voltage is dropped to ~0.7 V. This voltage is used to keep the internal memory in a retention state; while the rest of the chip's power supplies are disabled to save power.

When the chip wakes up, the buck capacitor is charged from ~0.7 V to ~1.4 V. This rapid charging, occurring over tens of microseconds, generates the observed current spike.

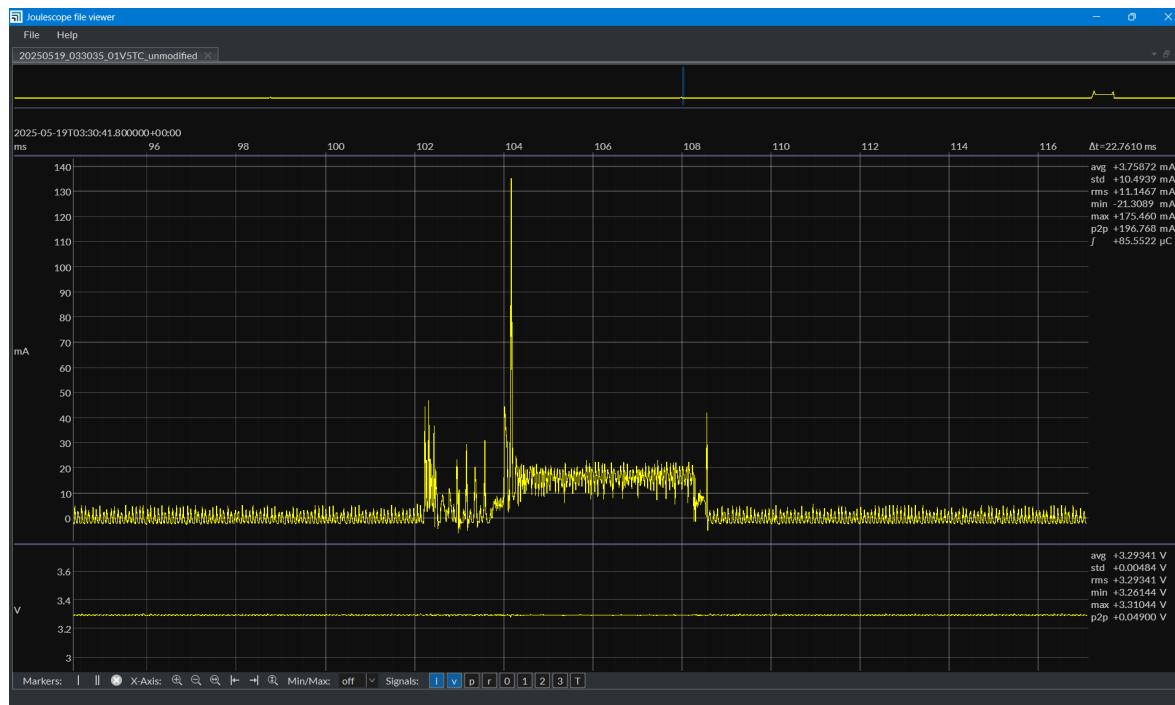


Figure 65. MM8108 DTIM wakeup zoomed out.

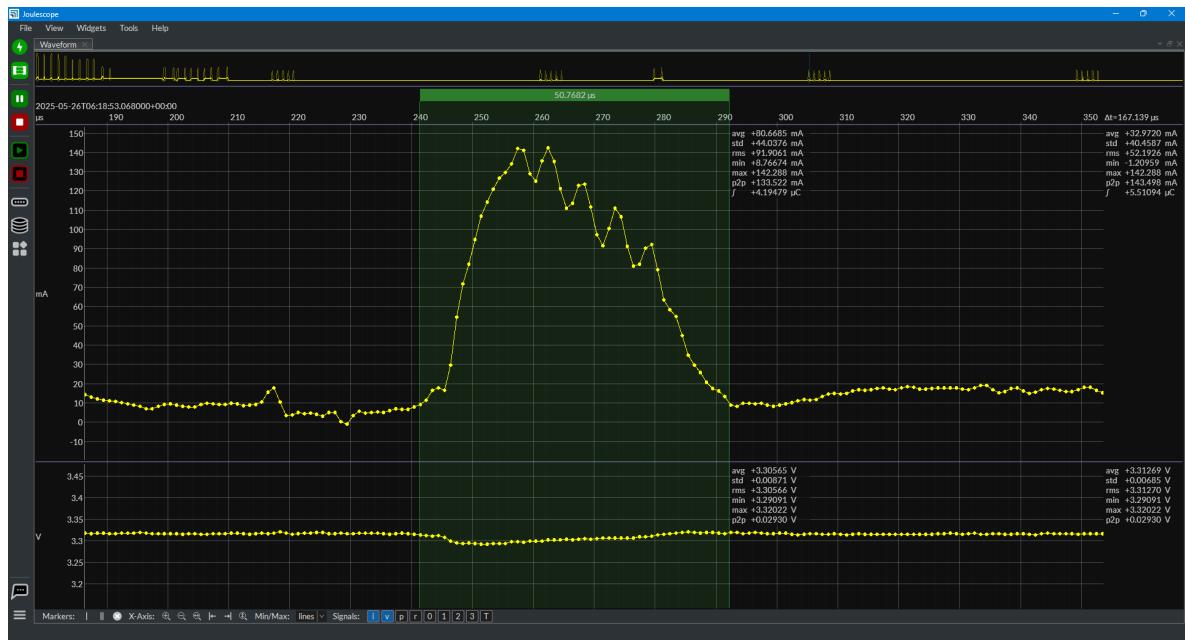


Figure 66. MM8108 wakeup current spike zoomed in.

Although the peak of the current is large, the duration is short, so the energy is small (in this instance less than 4  $\mu$ C) and the instantaneous current spike is sourced primarily from decoupling capacitors on the VBAT power rail. Batteries connected to the power supply are decoupled further by regulators and additional bulk capacitors upstream of the MM8108. All power consumption figures published in Morse Micro datasheets already include these transient spikes, ensuring there are no additional power costs beyond the specified values.

## 7.7 Negative Current Consumption

Users may observe brief negative current pulses when measuring the power consumption of the HaLow module, particularly during snooze periods. The screenshot below shows a zoomed-in view of the snooze current as measured on the HaLow header when running the ping application.

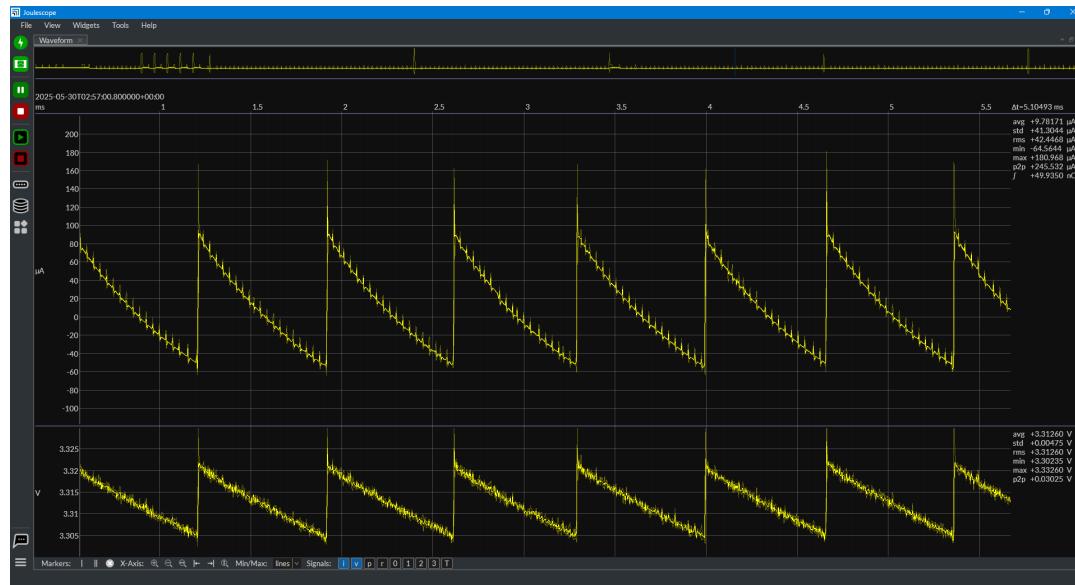


Figure 67. Zoomed in view of a snooze measurement showing negative current spikes.

This behavior is caused by the on-board buck-boost regulator, the [TPS63802](#). In the MM8108-EKH05 design, the MODE pin is tied low, which places the regulator in power-save mode (PFM) to optimize efficiency at light load:

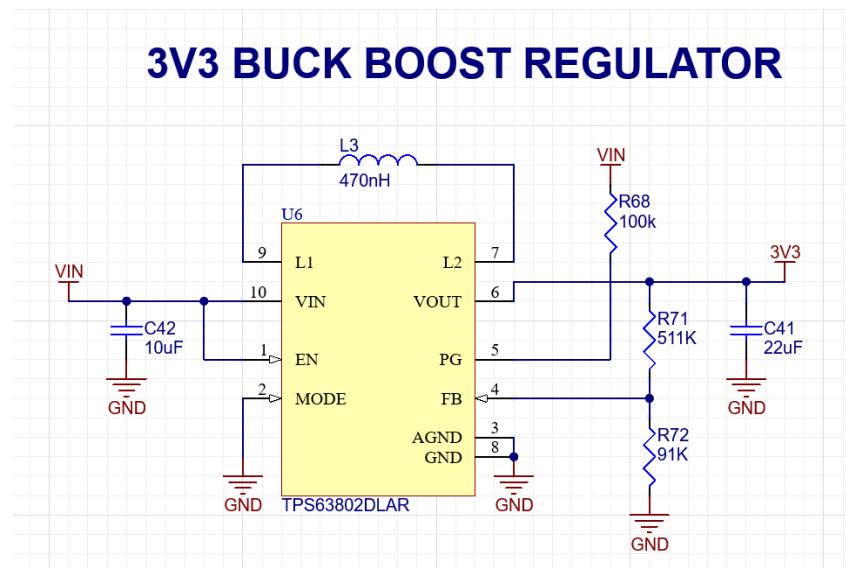
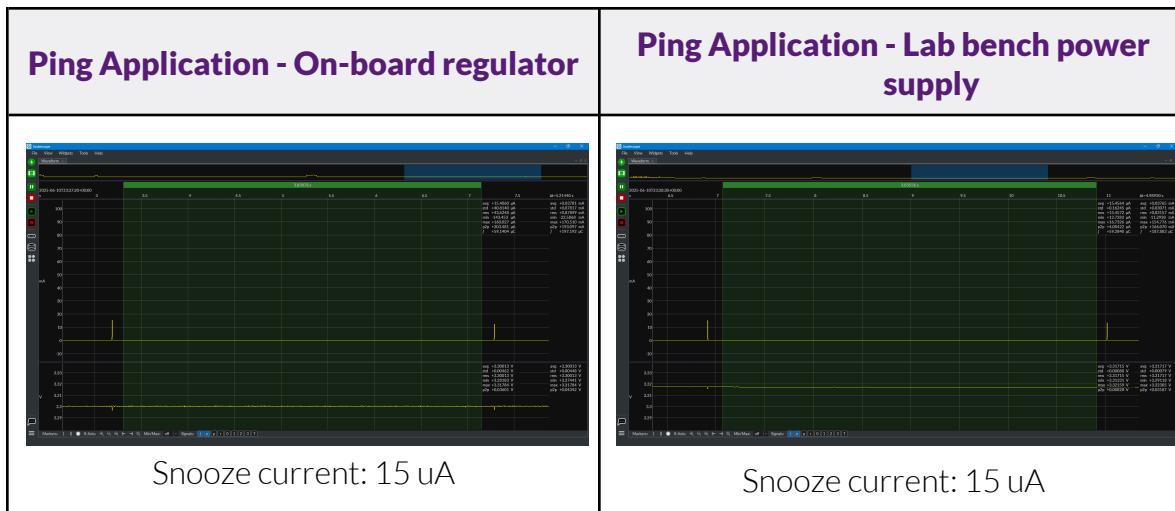


Figure 68. Schematic of TPS63802 in the MM8108-EKH05.

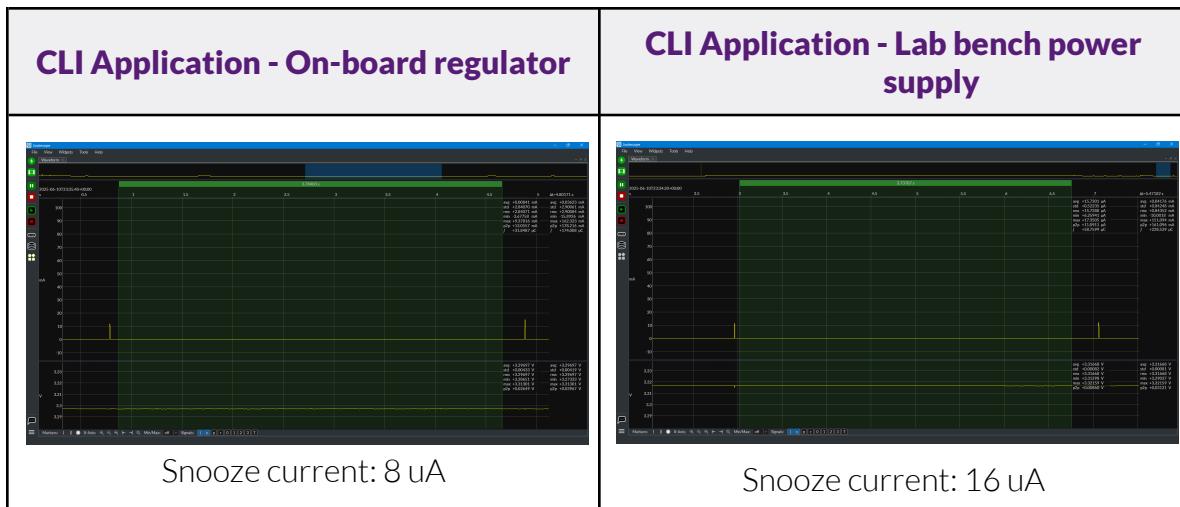
In PFM mode, the regulator delivers energy in short bursts and then enters an idle or “coast” phase. During these bursts, energy is stored in the output capacitor and inductor. If the system load is very low or momentarily inactive – such as during HaLow snooze – that energy may exceed what the load can absorb. This can lead to a temporary overshoot in output voltage and current to flow back into the regulator, through its internal switches or body diodes, or along parasitic paths.

This reverse current is captured by the Joulescope and appears as negative current, even though the HaLow module itself is not actively drawing current. These transients are a normal artifact of operating in PFM mode with a low or pulsed load and do not indicate an error or inefficiency. However, they may cause measured average power to appear lower or distorted unless properly interpreted.

The simplest way to eliminate the reverse current effect entirely is to bypass the on-board regulator and instead power the 3V3 rail using a lab bench power supply. However, under light load conditions, the difference in measured HaLow current between the on-board regulator and an external supply is negligible. For example, when running the ping application – which places the STM32U585 host microcontroller into a low-power sleep state – the difference in measured current is typically less than 0.1  $\mu$ A. In this case, the backflow effect is present but extremely minor.



In contrast, when using the CLI application, the STM32U585 host microcontroller remains active by default, drawing approximately 6 mA continuously. Under this heavier load, the reverse current pulses become more pronounced, with deeper negative current spikes observed on the Joulescope. This is due to larger energy bursts during power-save switching, which exacerbate transient effects.



# 8 Troubleshooting

## 8.1 Device Is Not Programming

If the STLINK V3-MODS is unable to connect to the STM32U585, please check the following steps.

1. Ensure the STLINK jumpers are populated. Without these, the programmer will be disconnected from the microcontroller.

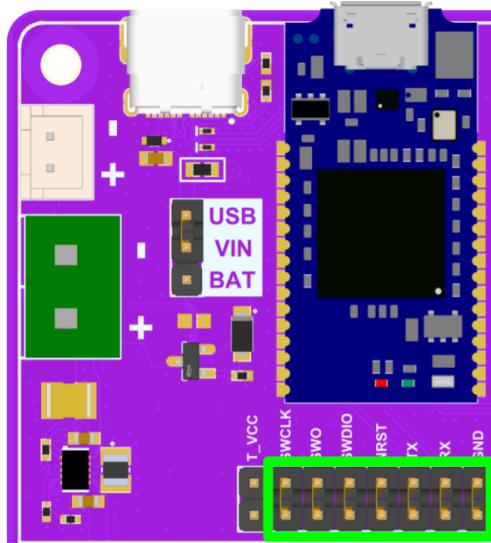


Figure 73. Troubleshooting step checking if the highlighted jumpers are connected.

2. Ensure the 3V3 and MCU jumpers are populated. The microcontroller will not be powered without these.

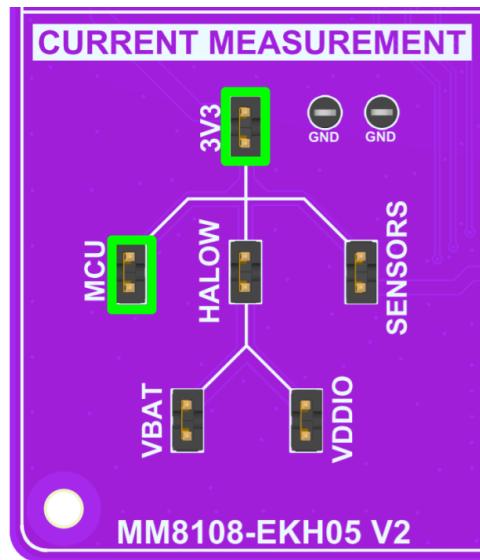


Figure 74. Troubleshooting step checking if the 3V3 and MCU jumpers are connected.

3. Ensure the power select jumper is on USB. If using batteries, check VIN voltage to ensure the battery is functioning normally.

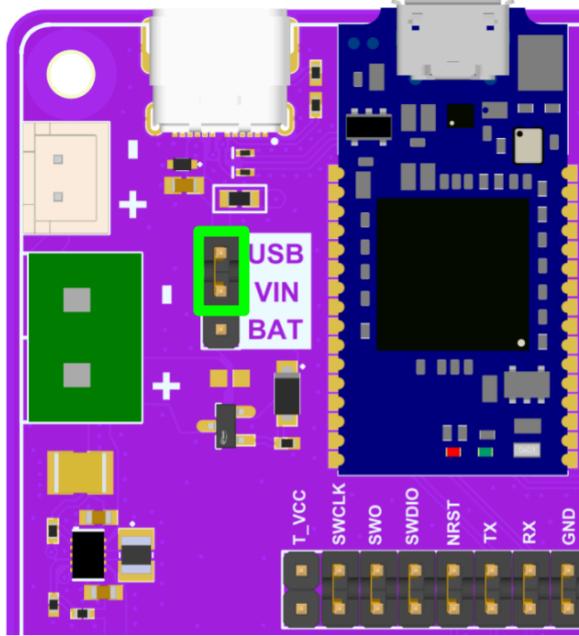


Figure 75. Troubleshooting step checking the input power supply.

4. Ensure the device is not in a low power state. If the microcontroller is in certain deep sleep states, the programmer will not be able to attach. To recover in this situation, hold the STM RESET button as the device is attempting to connect, and release. This will ensure the microcontroller can be attached to.

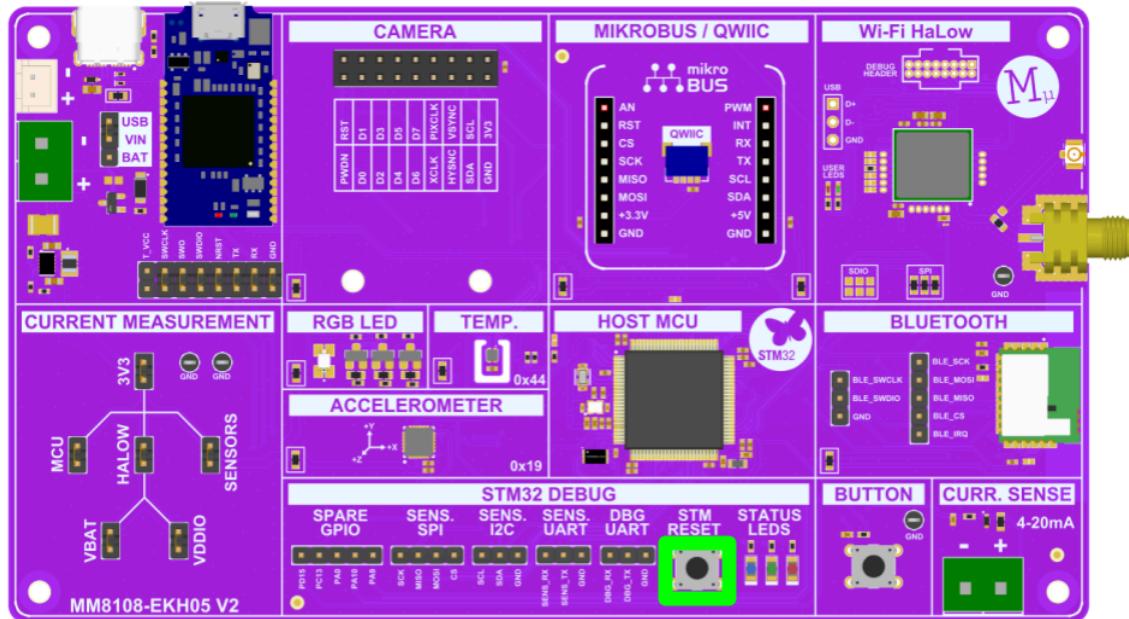


Figure 76. Reset button on the MM8108-EKH05. Resets the host MCU.

## 8.2 Poor HaLow Performance

If the HaLow performance is poor, the following items should be checked.

1. Ensure the AP configuration is expected. I.e. the correct channel and bandwidth are configured.
2. Ensure there are no other HaLow devices on the same channel causing in-band interference.
3. Ensure antenna connection is on securely.
4. If cabled, ensure there is sufficient attenuation (~40 dB) between the devices. If the power is too high, it will saturate the LNA.

## 8.3 High Power Consumption

There are many reasons that high power consumption may be measured. Please check the following:

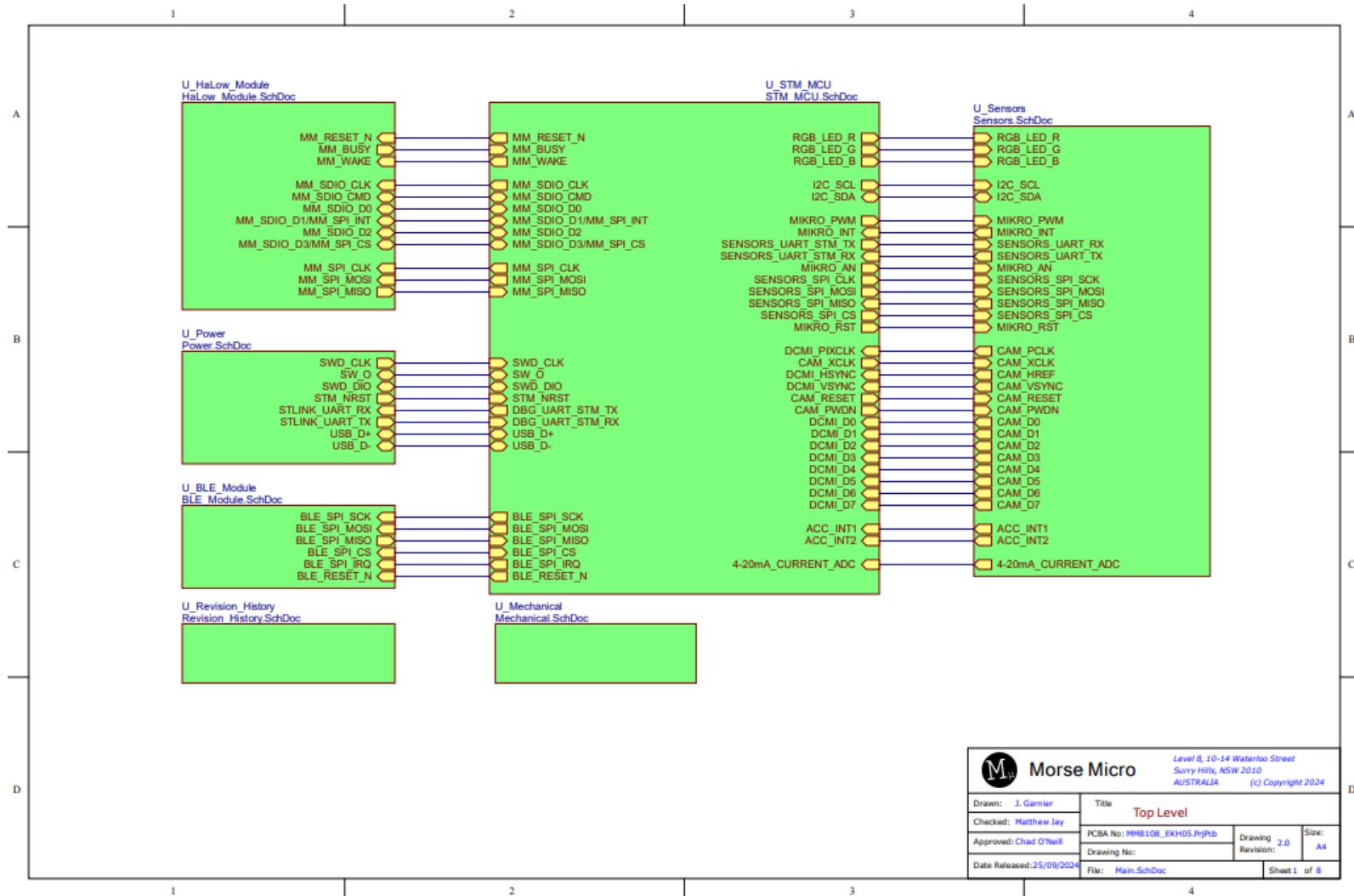
1. If the T\_VCC jumper is populated, this will add ~300uA to the 3V3 power consumption.
2. For lowest power consumption, ensure the network stack is set to FreeRTOS plus TCP instead of LwIP. See section [5.7](#) for details.
3. To replicate Morse Micro DTIM power consumption numbers, the devices should be placed inside a shielded box away from interference. If the device fails to receive the beacon from the AP, it will spend a longer time in active RX mode and consume more power resulting in misleading DTIM measurements.
4. The temperature plays a big factor in power consumption. All testing should be done at room temperature of ~77 degrees Fahrenheit (25 degrees Celsius) to match Morse Micro DTIM power consumption numbers.
5. Ensure the MM8108-EKH05 is associated with the AP. Double check SSID and passphrase match.
6. It is recommended to use as short leads as possible from the ammeter to the MM8108-EKH05. Longer leads will add equivalent series resistance (ESR) and impact the measurements.
7. Traffic from the network. If the AP is set to a mode where traffic gets bridged from the Ethernet interface to the HaLow interface, the MM8108-EKH05 will get sent any multicast traffic, and wake up to receive this. A good way to see this, is to set the security of both the AP and STA to an open network, and connect a sniffer to the setup to see what traffic is being sent while observing the power consumption. A setup script to launch the sniffer can be found on the Morse Micro customer portal after logging in - <https://www.morsemicro.com/downloads-dashboard/#/tools>. An easy way to remove all multicast traffic is to either set the AP to a mode where the interfaces are not bridged, or just simply disconnect the ethernet cable from the AP.

## 9 Errata

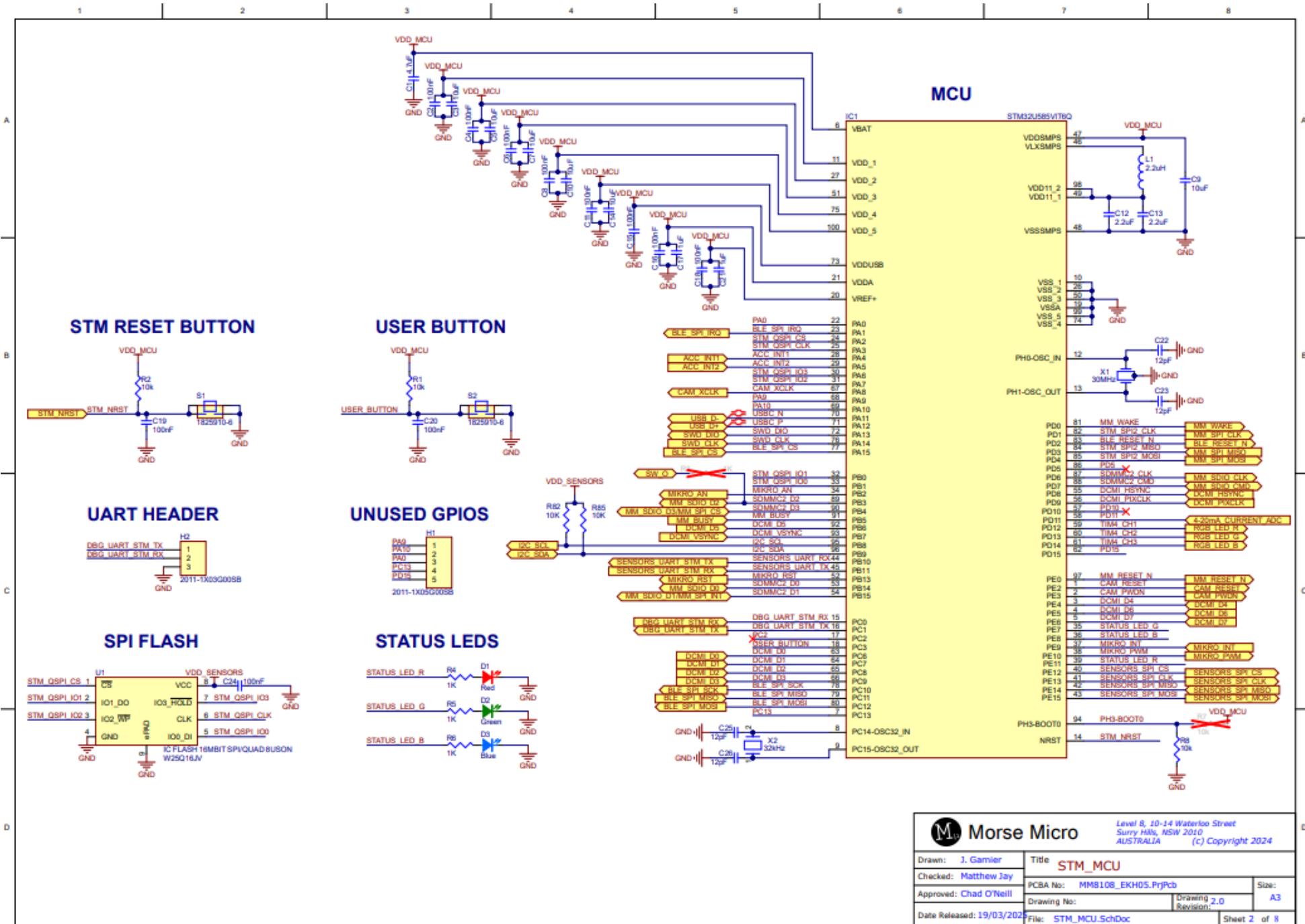
# 10 Revision History

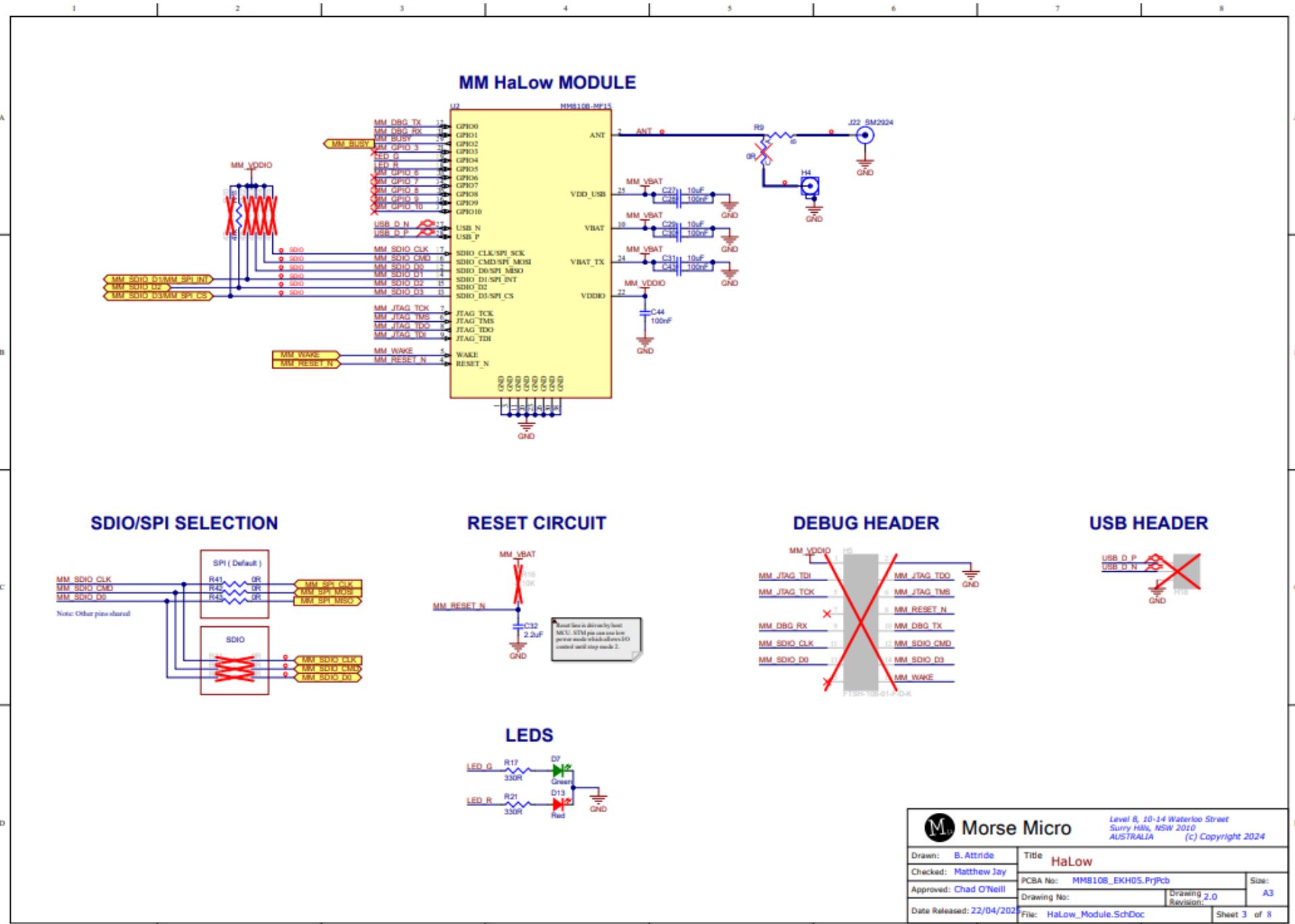
Release Number	Release Date	Release Notes
01	17/04/2025	<ul style="list-style-type: none"><li>Initial release.</li></ul>

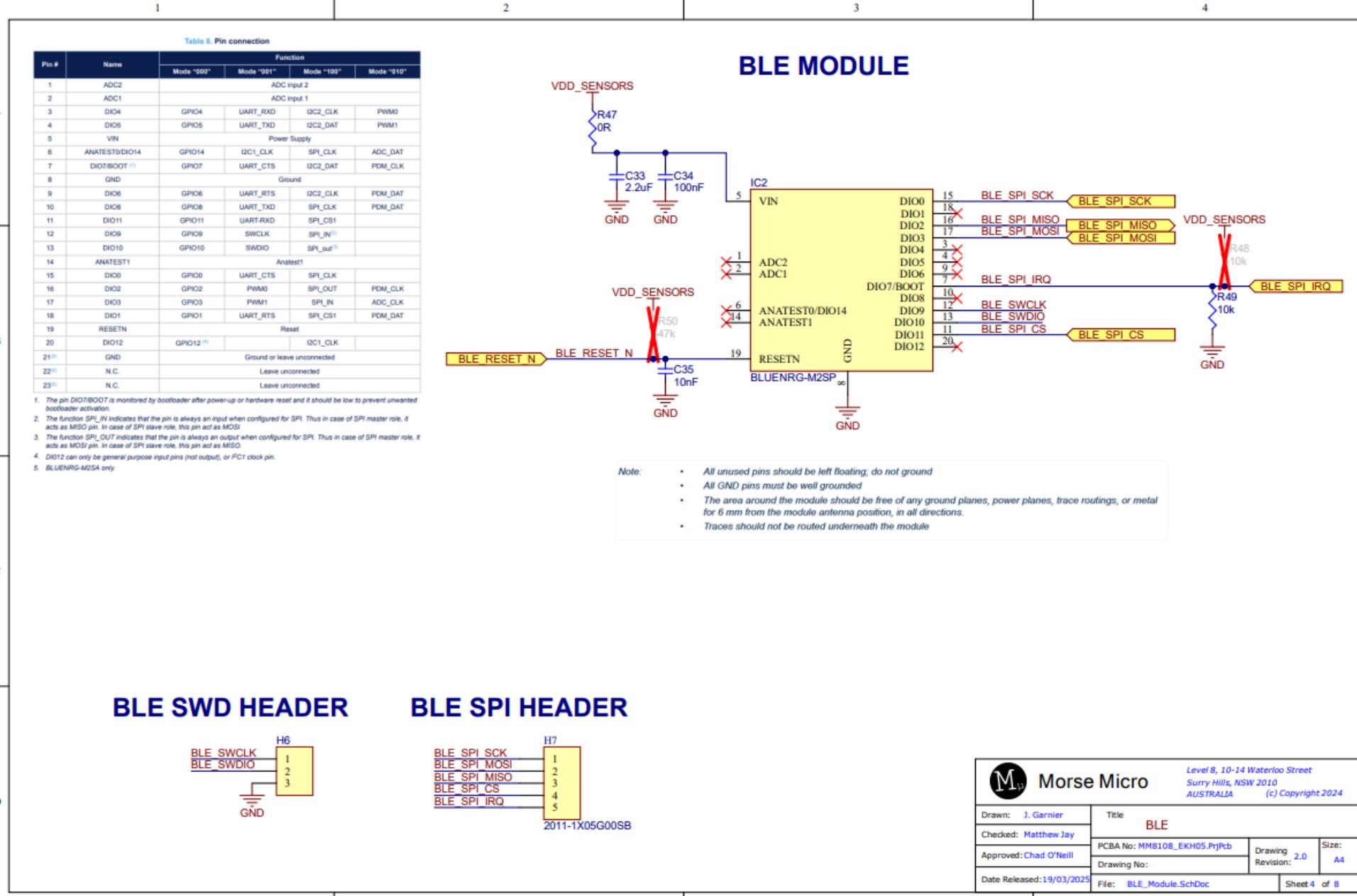
# Appendix A: Schematics

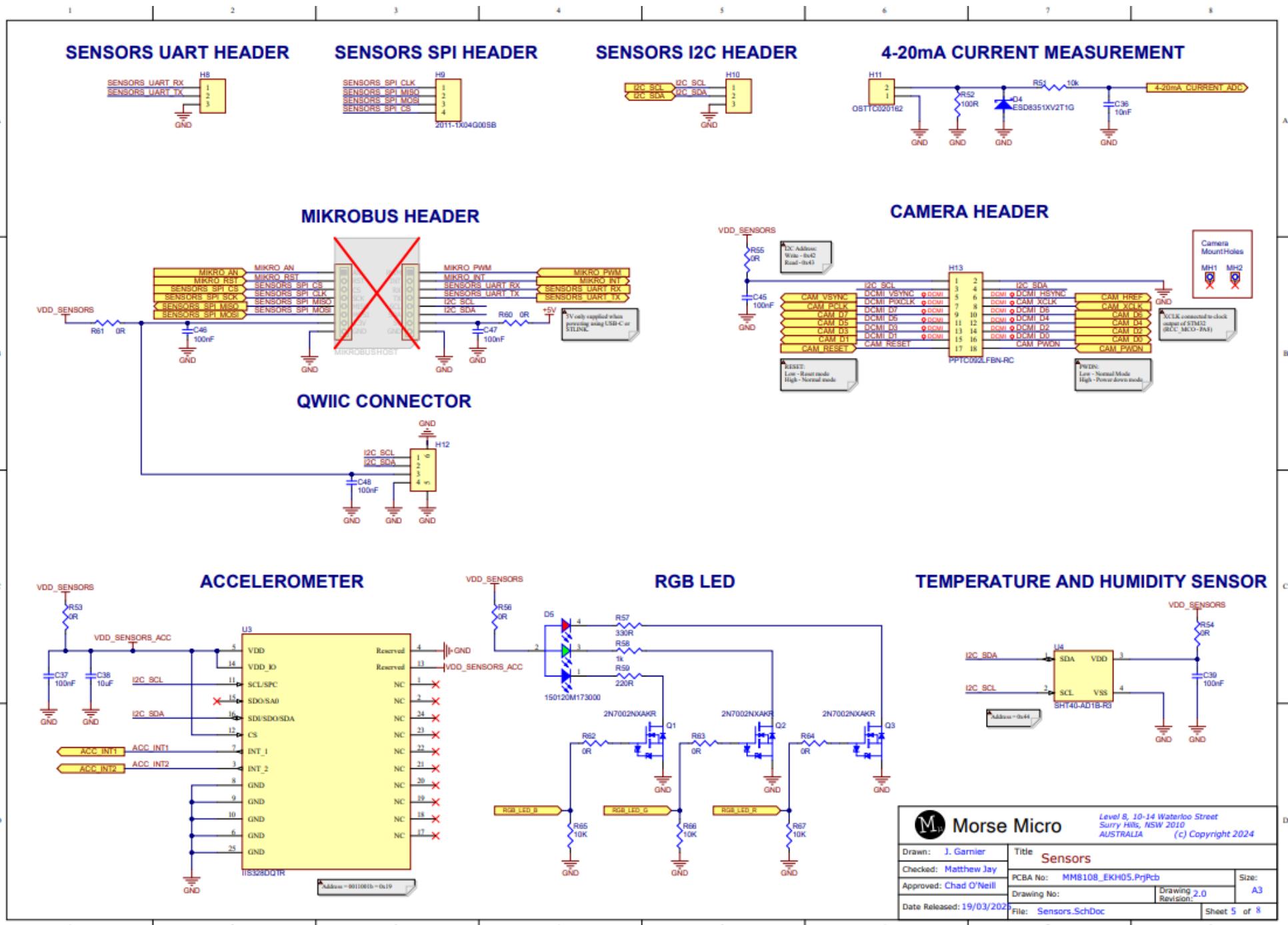


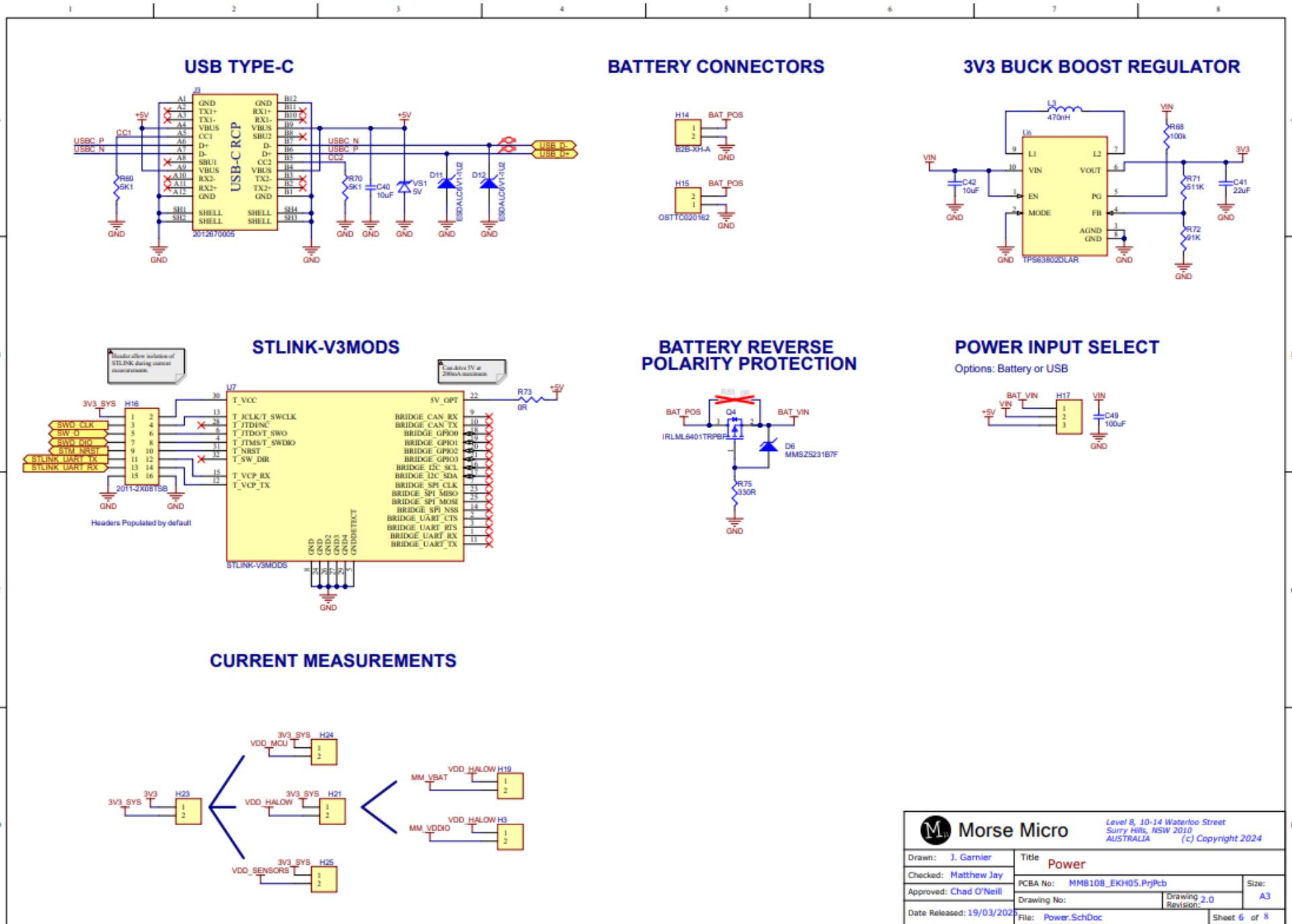
		Morse Micro	Level 8, 10-14 Waterloo Street Surry Hills, NSW 2010 AUSTRALIA	(c) Copyright 2024
Drawn:	J. Gamier	Title:	Top Level	
Checked:	Matthew Jay		PCBA No: MM8108_EKH05.PjPcb	Drawing 2.0 Size: A4
Approved:	Chad O'Neill		Drawing No:	Revision: A4
Data Released:	25/09/2024		File: Main.SchDoc	Sheet 1 of 8









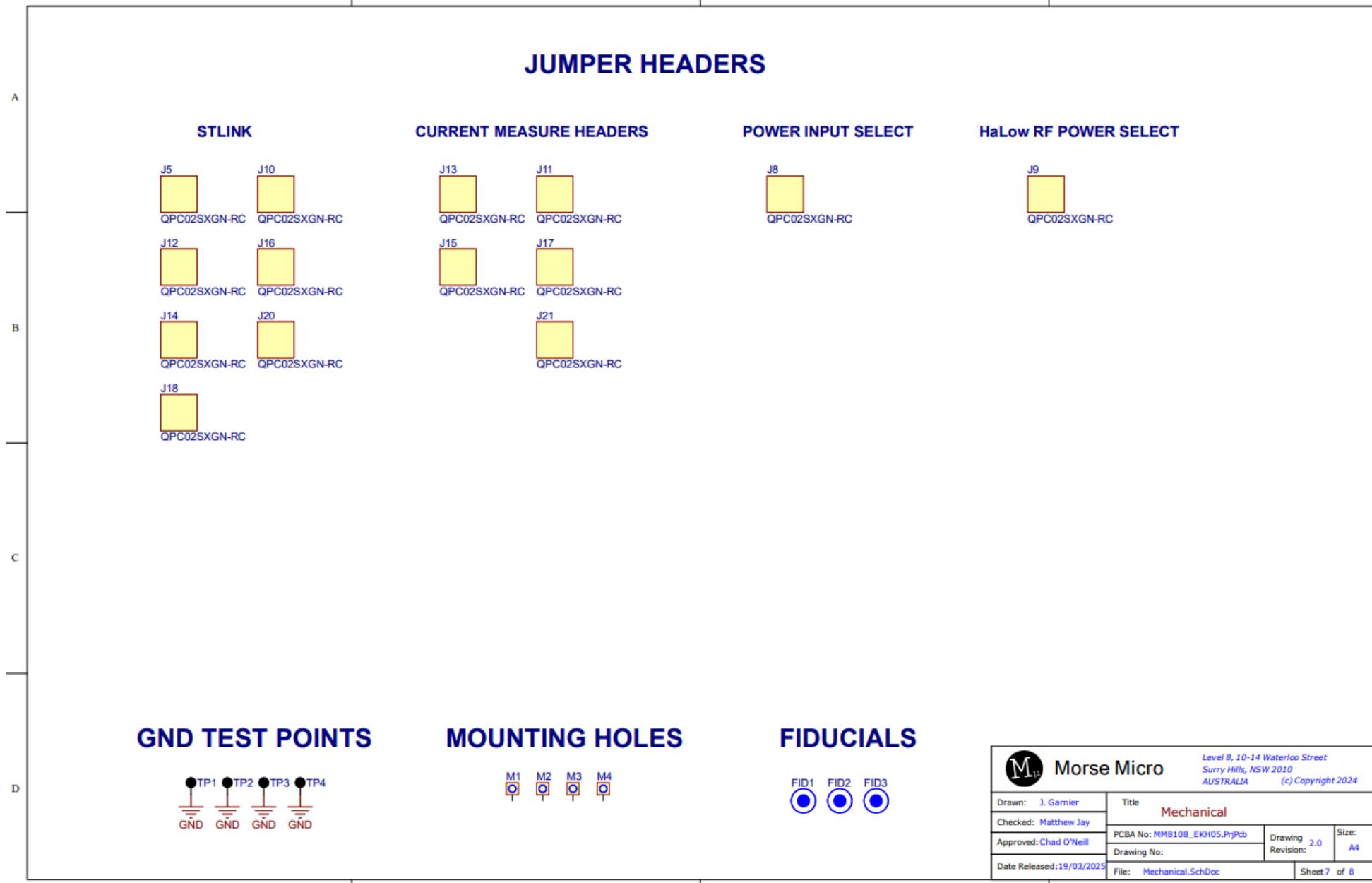


1

2

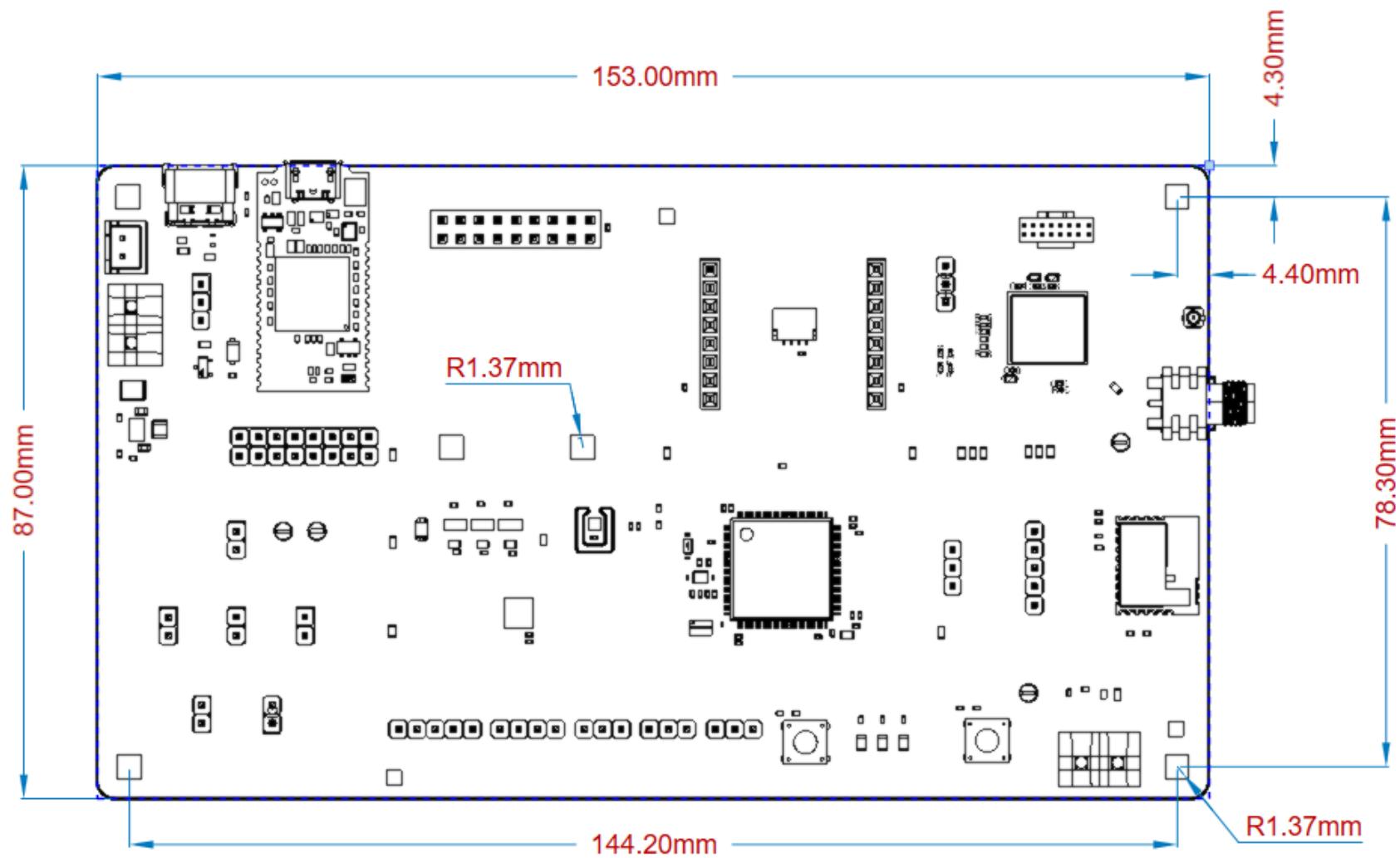
3

4



	1		2		3		4																
A							A																
B							B																
C							C																
D							D																
	1		2		3		4																
<table border="1"><tr><td colspan="2">Revision History</td></tr><tr><td>1.0</td><td>- Initial release.</td></tr><tr><td colspan="2"></td></tr></table>								Revision History		1.0	- Initial release.												
Revision History																							
1.0	- Initial release.																						
 <b>Morse Micro</b> Level 8, 10-14 Waterloo Street Surry Hills, NSW 2010 AUSTRALIA (c) Copyright 2024																							
<table border="1"><tr><td>Drawn: J. Garner</td><td>Title</td><td colspan="2">Revision History</td></tr><tr><td>Checked: Matthew Jay</td><td></td><td>PCBA No: MMB108_EKH05.PnjPcb</td><td>Drawing Size: A4</td></tr><tr><td>Approved: Chad O'Neill</td><td></td><td>Drawing No:</td><td>Revision: 2.0</td></tr><tr><td>Date Released: 25/09/2024</td><td></td><td>File: Revision_History.SchDoc</td><td>Sheet 8 of 8</td></tr></table>								Drawn: J. Garner	Title	Revision History		Checked: Matthew Jay		PCBA No: MMB108_EKH05.PnjPcb	Drawing Size: A4	Approved: Chad O'Neill		Drawing No:	Revision: 2.0	Date Released: 25/09/2024		File: Revision_History.SchDoc	Sheet 8 of 8
Drawn: J. Garner	Title	Revision History																					
Checked: Matthew Jay		PCBA No: MMB108_EKH05.PnjPcb	Drawing Size: A4																				
Approved: Chad O'Neill		Drawing No:	Revision: 2.0																				
Date Released: 25/09/2024		File: Revision_History.SchDoc	Sheet 8 of 8																				

## Appendix B: Mechanical Drawing





**Morse Micro Pty. Ltd. Corporate Headquarters**  
Level 8, 10-14 Waterloo Street, Surry Hills, NSW 2010,  
AUSTRALIA

**Morse Micro Inc. – USA**  
40 Waterworks Way  
Irvine, CA 92618, UNITED STATES

**Morse Micro – China**  
Floor 7, Room 08-210, Wework office 292 Yan'an Rd,  
Shangcheng District, Hangzhou 310003

**USA:** +1.949.501.7080  
**Australia:** +61.02.905.45922  
**China:** +86.571.229.30277  
**Email:** sales@morsemicro.com  
[www.morsemicro.com](http://www.morsemicro.com)

© 2025 Morse Micro Pty. Ltd.  
All rights reserved. Morse Micro and the Morse Micro logo are trademarks of Morse Micro Ltd. All other trademarks and service marks are the property of their respective owners.

### About Morse Micro

Morse Micro is producing IEEE 802.11ah / Wi-Fi HaLow solutions for Internet of Things (IOT) - based on a newly certified Wi-Fi standard called HaLow. Morse Micro is a VC-backed Startup headquartered in Sydney, Australia. Learn more at [www.morsemicro.com](http://www.morsemicro.com)

