Getting started with the Onethinx development kit

This document describes the steps to get you started with the Onethinx development kit. Helpful figures are at the end of this document, so don't forget to scroll through if something is not clear.

1. Prerequisites

We advise to use a MiniProg4 or KitProg3 Debug kit to program the PSOC63 that is on the Onethinx Module. Next, you will need to download and install the latest ModusToolbox IDE from Cypress. Finally, go to the Onethinx website to download the *hello world* example project.

2. LoRaWAN coverage

The Onethinx Core module is able to connect with any LoRaWAN provider using the LoRaWAN OTAA[1] join procedure. Onethinx advises to use The Things Network (TTN) in combination with our developers' kit. This guide focuses on setting up an OTAA connection with TTN.

3. Modus Toolbox

We advise to install ModusToolbox into C:\ if you are on windows. Keep in mind that ModusToolbox will not accept paths with spaces or dots etc. Open Modus toolbox example projects by pointing modus at startup to the directory that contains the <APP_NAME>_mainapp, <APP_NAME>_config and <APP_NAME>_mainapp_psoc6pdl directories. We have developed and tested our code only in the windows version of Modus Toolbox.

4. Hello World project

If you have installed your Modus Toolbox IDE and have your devkit hooked up to your programmer, you can try uploading your first project to thedevelopers kit. We have made a simple *Hello World* project available for first time users that sends a LoRaWAN package at regular intervals. This project is available on our website and forum. Whenever opening ModusToolbox with a copied project folder, the first thing to do is 'clean application'. In the project explorer (upper-left window pane), right-click the '.._mainapp' project and select 'clean application'. Next, select 'build application' from the same drop-down menu. To view the complete build log in the console, select the 'CDT global build console' from the menu-bar of the output window (lower-right window pane). To program or debug, go to the 'play' or 'bug' icon of the ModusToolbox toolbar. You may have to select a program/debug configuration first.

[1] OTAA, over the air activation.

5. Setup a TTN connection

The Things Network (TTN) is an open-source network of LoRaWAN gateways. Visit https://www.thethingsnetwork.org/ for coverage maps and description to build your own gateway. Here we shortly describe the steps to acquire keys that will connect your developers kit to TTN via LoRaWAN. These keys identify your LoRaWAN module and connect it to an (your) internet server, whilst also encrypting the data. Follow these steps to acquire TTN Keys (similar steps must be followed for your own gateway or another LoRaWAN provider):

1. If you haven't already, register for an Account at https://www.thethingsnetwork.org/
2. Go to the TTN Console page and select Applications (see the paragraph Registering your device at TTN)
3. Add new Application and select this newly created application
4. Go to the Devices page and register a new device
5. Note the Device EUI, Application EUI and App Key values - these are needed for OTAA
6. Implement these keys in the Hello World project, make sure you join with OTAA.
    a. You can extend the file OnethinxCore/LoRaWAN_keys.h for this.
    b. Let the coreConfig structure in main.c point to the correct TTN_OTAAkeys structure.

6. Build and Program

After implementing the LoRaWAN Keys in the project you can now program the project into the Developers Kit. Use debug mode to step through the code and check the return status of the join request. The 'LightSensor' and 'SendOnSwitch' examples provide more feedback - use a 3V3 FTDI on pins 10.0 and 10.1 - so be sure to check these out as well.

7. Send Your First Package

If you are using TTN you should now be able to see your package on the device overview data page on the TTN console (again, other providers or your own gateway software will likely provide a similar view). Once programmed, you can run the device on a CR2450 coin cell if you set the adjacent jumpers to 'battery'.

8. Have Fun!

Now that you have sent your first LoRaWAN Package with the Onethinx LoRaWAN Core Module you can extend the Hello World example to create all kinds of applications. Integration of all sorts of sensors and actuators are possible with the versatile PSoC6. We would also love to see your creative solutions posted on our forum. Check out our downloads section regularly for more code examples. Good luck and have fun!

9. References

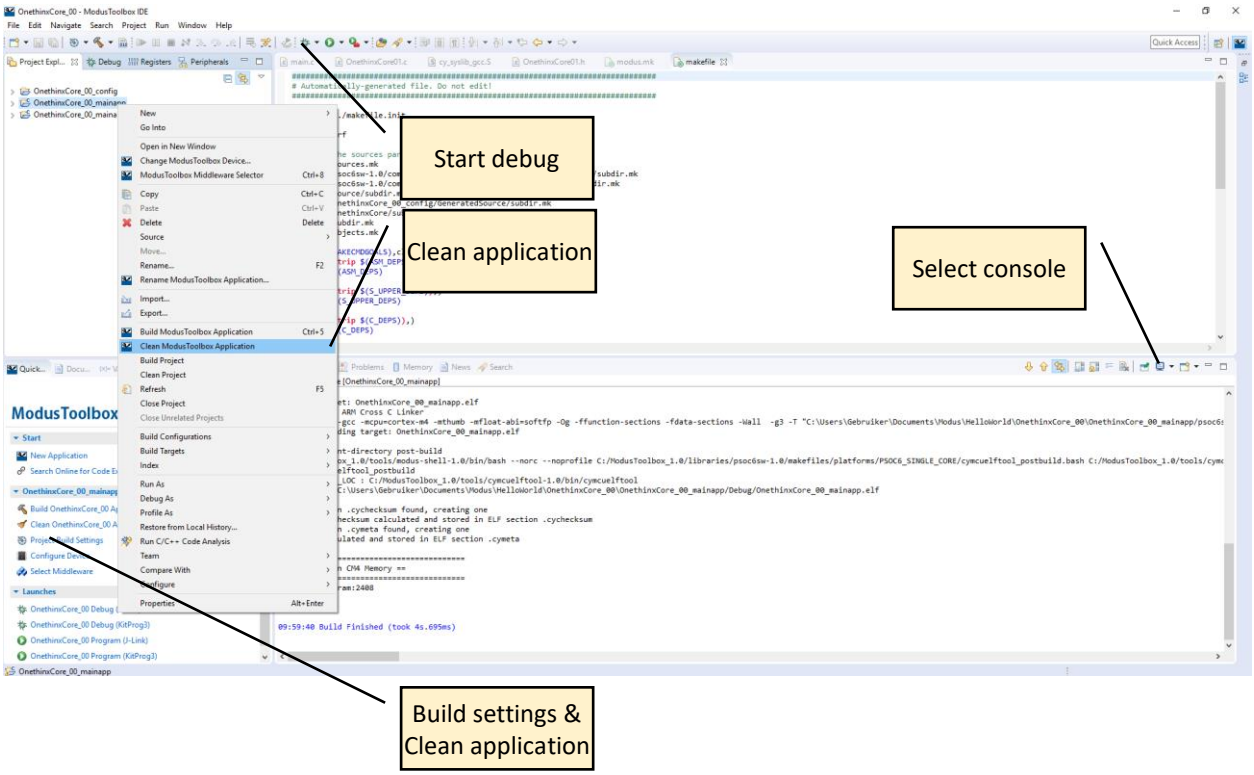
1. Order development kits: https://onethinx.com/request-module-samples
2. The Things Network: https://www.thethingsnetwork.org/
3. KitProg3 programmer: http://www.cypress.com/documentation/development-kitsboards/cy8ckit-147-psoc-4100ps-prototyping-kit
4. Modus Toolbox: http://www.cypress.com/products/modustoolbox-software-environment
5. Downloads section of Onethinx (Hello World code example): https://onethinx.nl/downloads/

6. Onethinx forum: https://forum.onethinx.com/

Appendices:

- Where's what in Modus Toolbox
- Registering a device at TTN
- Devkit pinout
- Known issues

Where's what in Modus Toolbox



Registering your device at TTN

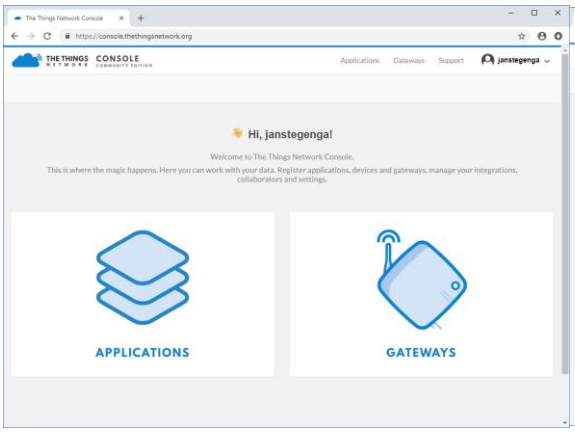


Figure 1: select *Applications*

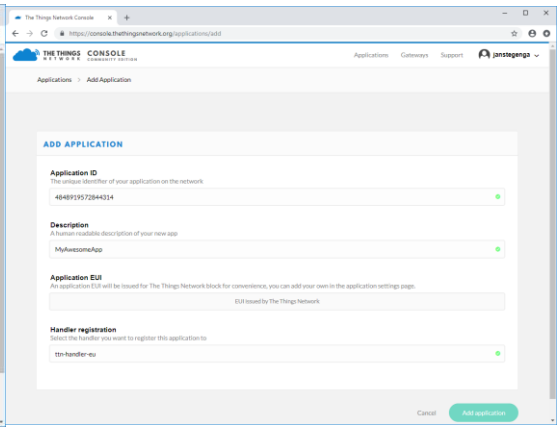


Figure 2: create new application

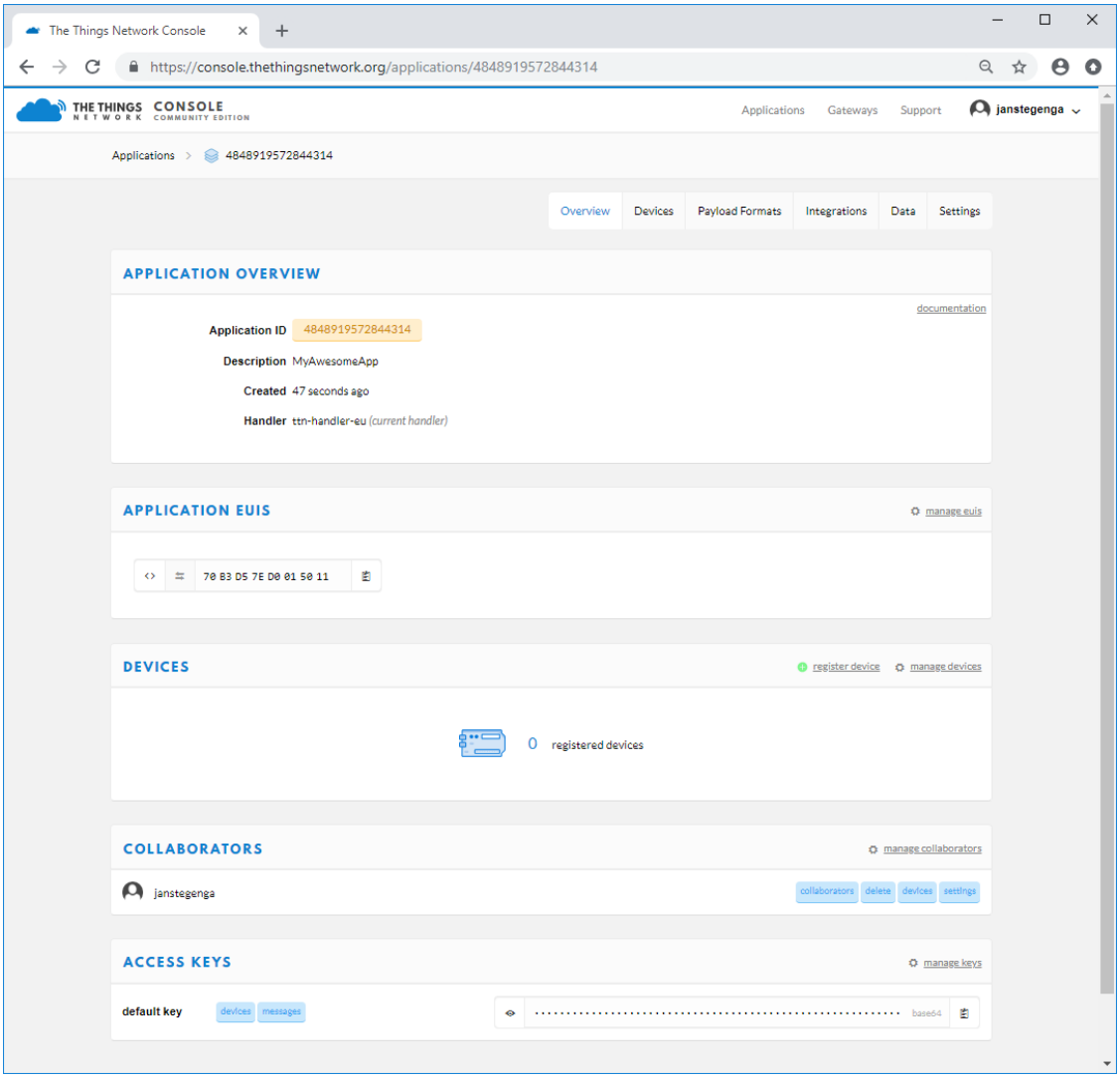


Figure 3: Application ID and EUI, select *register device*

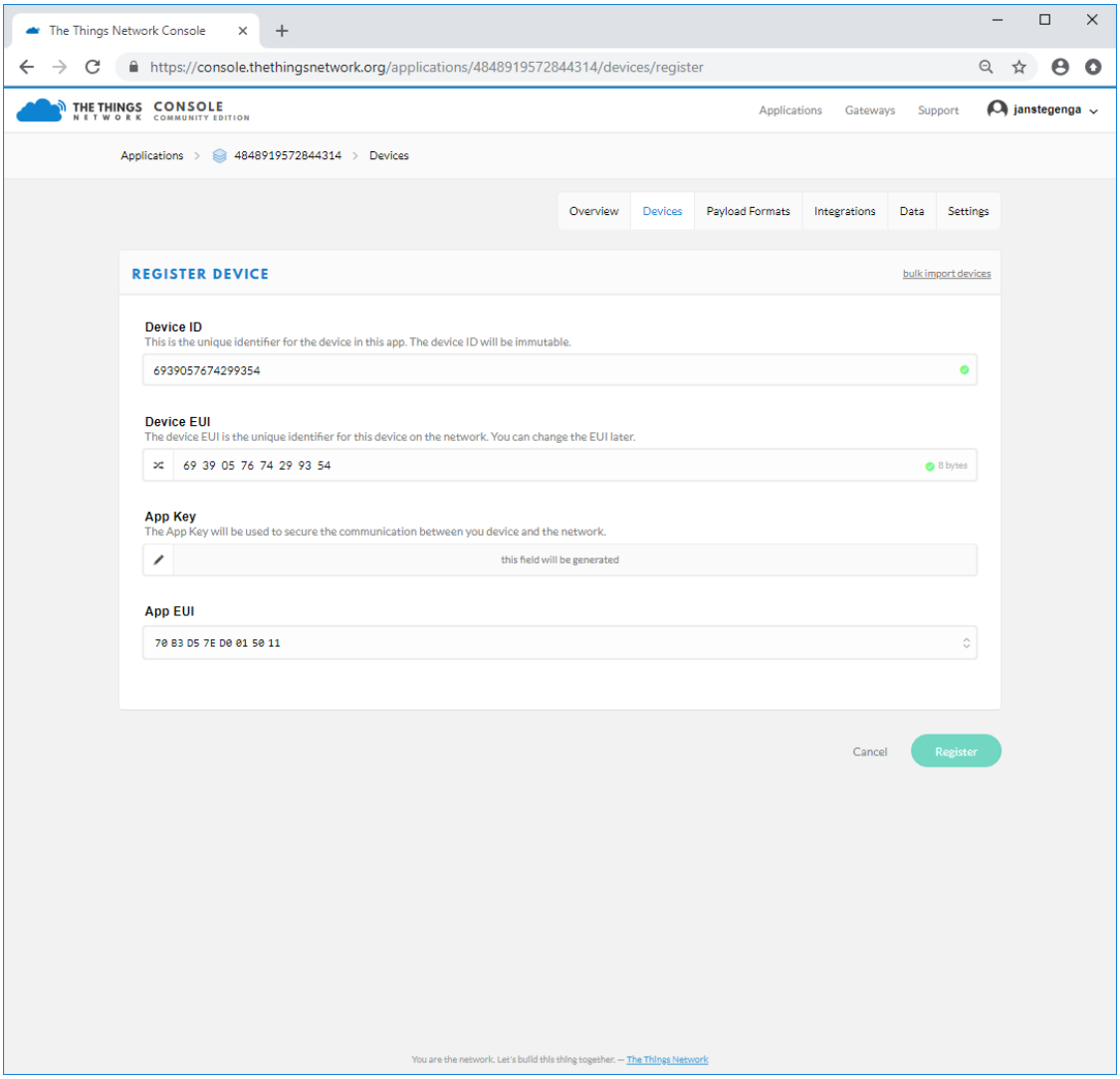


Figure 4: generate a device ID and device EUI.

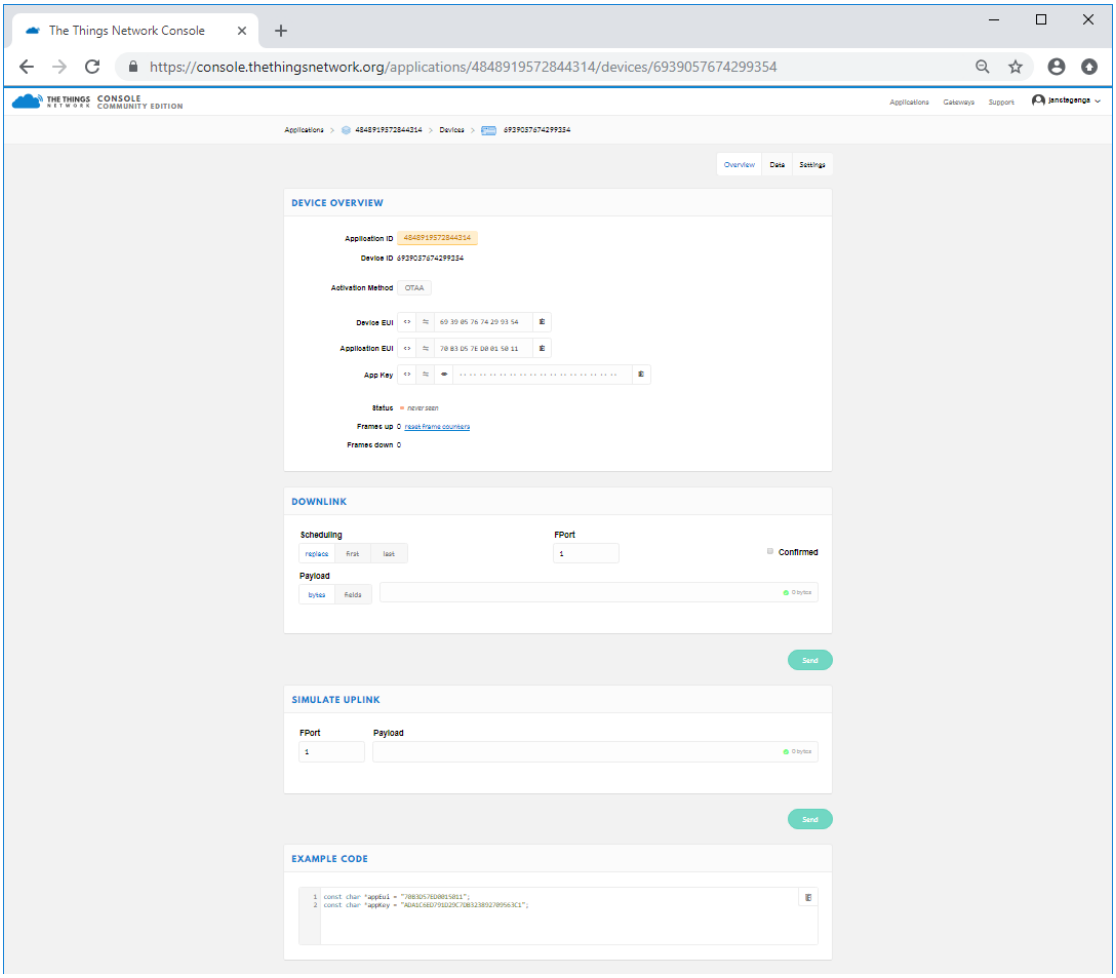


Figure 5: Device overview. Check out the *data* tab to see data coming in.

The key structure is defined as follows (LoRaWANKeys.h):

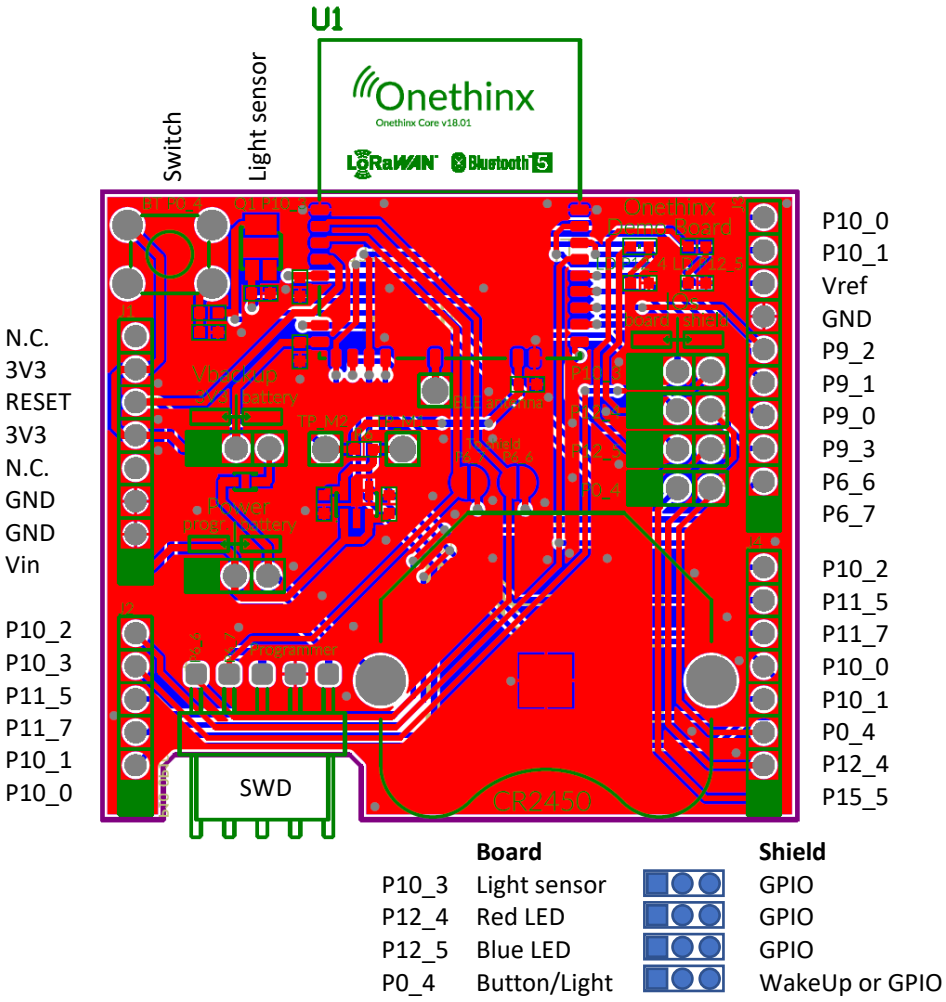
```
LoRaWAN_keys_t MyTTN_OTAAkeys = {
        .KeyType            = OTAA_10x_key,
        .PublicNetwork      = true,
        .OTAA_10x.DevEui    = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 },
        .OTAA_10x.AppEui    = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 },
        .OTAA_10x.AppKey    = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }
};
```

In main, the above key is referenced in the coreConfig structure:

```
coreConfiguration_t coreConfig = {
        .Join.KeysPtr       = &MyTTN_OTAAkeys,
        .Join.DataRate      = DR_0,
        .Join.Power         = PWR_MAX,
        .Join.MAXTries      = 10,
        .TX.Confirmed       = true,
        .TX.DataRate        = DR_0,
        .TX.Power           = PWR_MAX,
        .TX.FPort           = 1,
};
```

Devkit pinout

Since the module has less GPIOs than the Arduino standard, so several GPIO's connect to more than one header pin. The block of 4 jumpers select between devkit-functionality or general GPIOs available on the header(s).



	Board		Shield
P10_3	Light sensor		GPIO
P12_4	Red LED		GPIO
P12_5	Blue LED		GPIO
P0_4	Button/Light		WakeUp or GPIO

The test points TP_M1 and TP_M2 are connected by a 0-Ohm resistor. Remove the resistor and connect the test points with a current meter to measure the power consumption. Note that this will only give a valid reading when the device is battery-powered (and disconnected from the programmer).

In the examples, pins 10_0 and 10_1 are often used for serial communication (115200 baud, 8 data bits, no parity, 1 stop bits, no flow control).

Known issues

Version 00.

1. The module may send more than 1 join requests when debugging.
    a. This is due to a number of resets in the procedure and can be prevented by delaying the lorawan_join() call by a number of milliseconds.
2. Downlink messages will not be confirmed by the module.
3. ADR (automatic data rate) is not implemented.
4. Low power settings are not implemented.
5. Single channel gateways are not supported because they are not completely compatible with the LoRaWAN protocol.