RAK LoRaWAN OpenMCU Development Guide

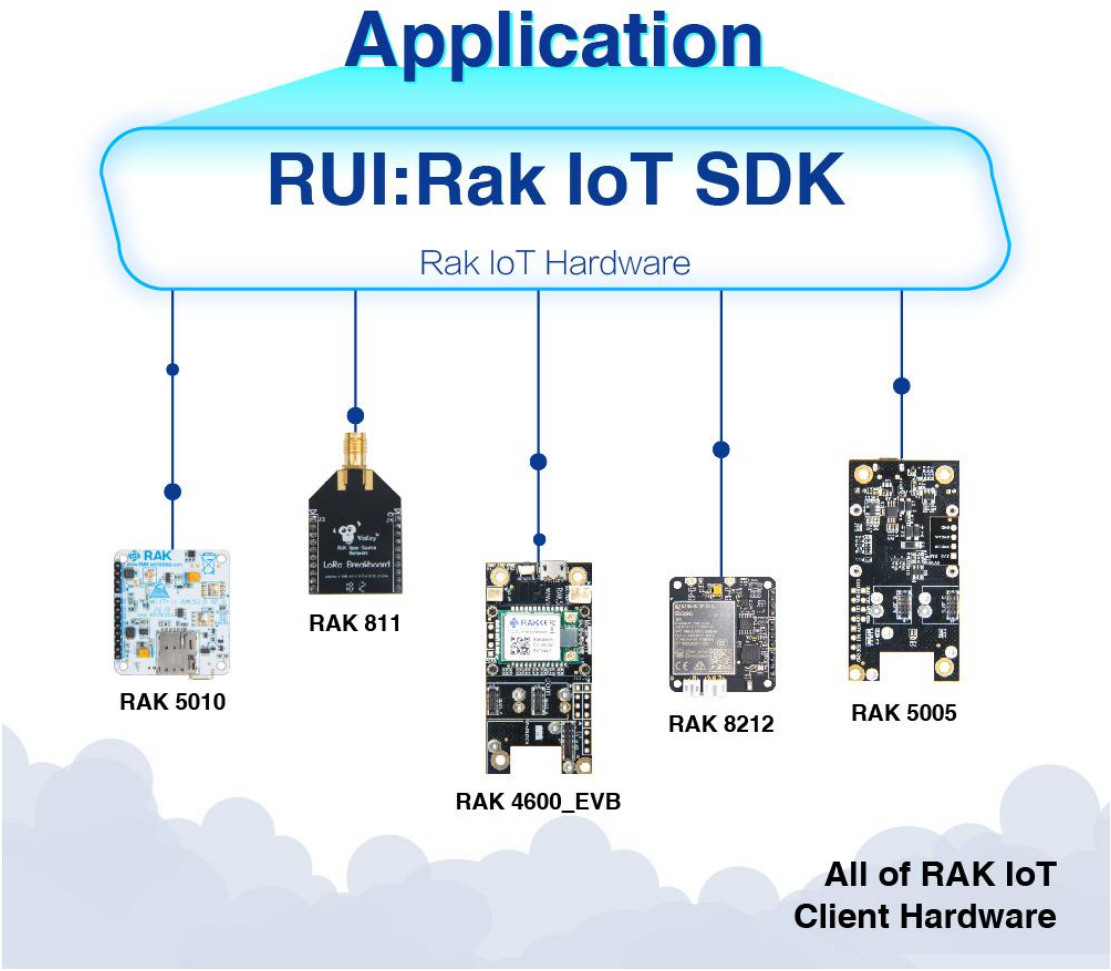


Figure 1: RAK RUI Application Diagram

Background: As shown in the Figure 1, The usual and most common way to integrate RAK LoRa node modules and products into your projects are throught the AT commands interface. In some cases, however, this is not sufficient and a tighter integration is required. Based on the OpenMCU approach, we introduce the RAK RUI framework. This framework is an innovative way that allows you to

customize the firmware of the RAK modules and products according to your business needs.

The RUI Framework comprises two main sections: the RUI Core layer, and the RUI Application layer. The RUI core exposes its functionalities through a set of APIs for the RUI Application. This architecture allows developers to focus on the requirement of their applications and the implementation of the business logic in the RUI Application layer. All this without the need of understanding the underlying technical details.

Once developers have customized the logic in the RUI Application layer, they will be able to upload the code to the RUI Online compiler and generate the customized firmware and flash the modules.

Objective: This guide is intended for developers who want to know how to get started with the RUI platform, and want to customize the official firmware of the RAK modules / products.

You can customize firmware for the following RAK modules / products:

- RAK811 LoRa module
- RAK4200 LoRa module
- RAK4600. LoRa+BLE module
- RAK5205/RAK7205
- RAK7204
- RAK5010 Cellular+BLE board

Learning Worflow:

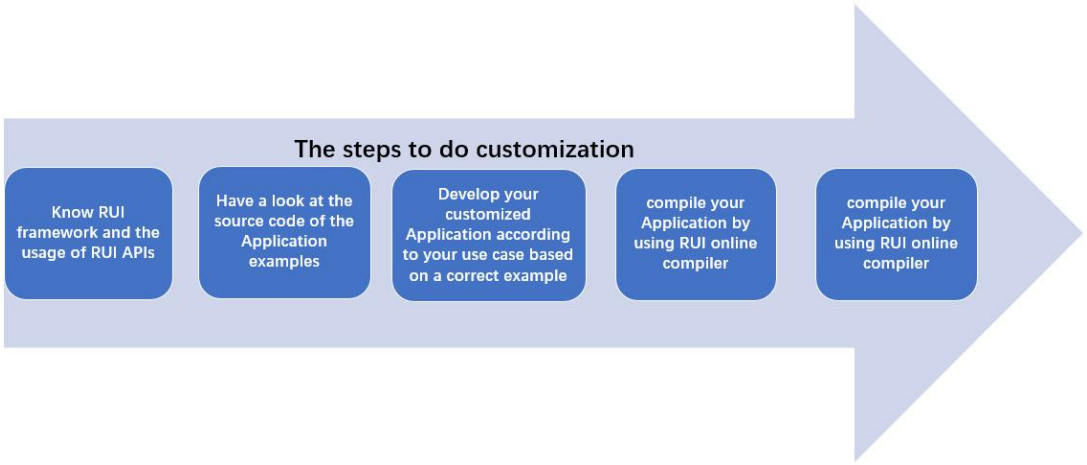


Figure 2: Workflow to learn RAK RUI framework

1. Know RUI framework and usage of RUI APIs

The firmware in a Wisblock module comprises two sections: a) the RUI Core, and b) the RUI Application.

The RUI core hides the complexity of the firmware development such as the interaction between the micro controller and the LoRa transceiver through the SPI interface, interaction to the sensors through the I2C interface, etc; and exposes only the high level functionalities through a set of APIs for the RUI Application. This architecture allows developers to focus on the requirements of their applications and the implementation of the business logic in the RUI Application layer. All without the need of spending time and effort to understand the associated technical details.

Once developers have customized the logic in the RUI Application layer, they will be able to upload the code to the RUI Online compiler and generate the customized firmware needed to upgrade the Wisblock modules.

The RUI API reference document can be found here:

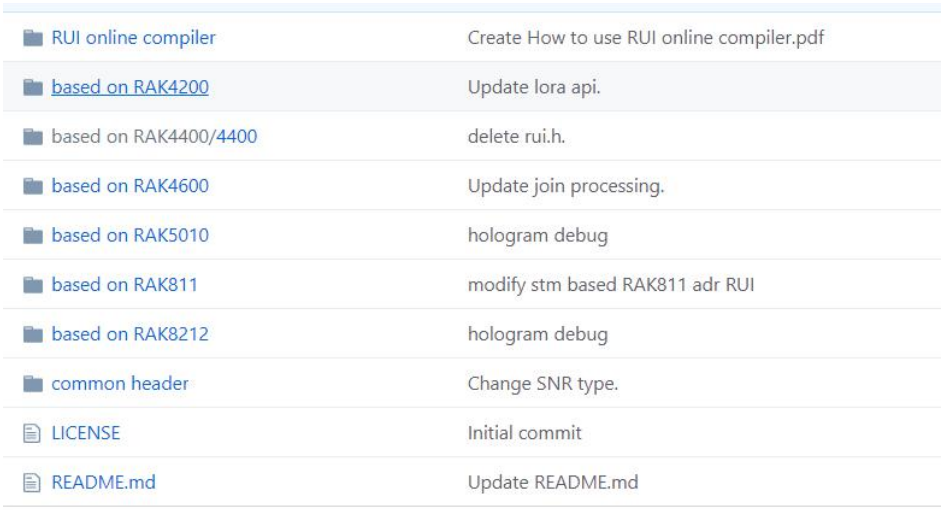
https://doc.rakwireless.com/developer-tools/developer-tools/-

The RUI Application layer is fully open source, and RAK has supplied a sample application code on Github as reference. However, the RUI Core is not open source. After completing your customized Application, you should compile it using the RUI online compiler. The RUI core will be compiled together automatically in order to generate the final customized firmware.

2. Have a look at the sample source code of the RUI Application

The best way to learn about the RUI framework is to review the sample code available **on Github**. This will allow developers to **get familiar with the RUI framework and the usage of RUI APIs. The URL of the repo is the following:**

https://github.com/RAKWireless/Products_practice_based_on_RUI



RUI online compiler	Create How to use RUI online compiler.pdf
based on RAK4200	Update lora api.
based on RAK4400/4400	delete rui.h.
based on RAK4600	Update join processing.
based on RAK5010	hologram debug
based on RAK811	modify stm based RAK811 adr RUI
based on RAK8212	hologram debug
common header	Change SNR type.
LICENSE	Initial commit
README.md	Update README.md

Figure 3: RAK RUI Application repository at Github

Actually, it's important to have in mind that the official firmware of all the RAK module are also developed on top of the RUI core and compiled using the RUI Online compiler.

In the Figure 3, you can find several folders such as: "based on RAK4600", "based on "RAK811", etc. These are the folders for each type of RAK's core modules, as the names show. Inside of each folder there are subfolders that contain the sample codes implemented for the Wisblock boards with the same core module.

For example, the RAK5202 board, RAK7204 board are based on the RAK811 core module. Therefore, under the folder "based on RAK811" you will find the folders listed in the Figure 4.

app_5205	modify stm based RAK811 adr RUI
app_7204	modify stm based RAK811 adr RUI
app_PT100_Digital_Pressure_Sensor	update
app_RAK811	modify stm based RAK811 adr RUI
README.md	The first version

Figure 4: Examples based on RAK811 at Github.

1) "app_5205" is the folder that contains the application source code for the RAK5205 / RAK7205 boards.

BME680-board.c	update based on RAK811
BME680-board.h	update based on RAK811
BME680.c	update based on RAK811
BME680.h	update based on RAK811
BME680_defs.h	update based on RAK811
app_5205.c	optimize flash store,fix 5205 sleep sensor bug
at_cmd.c	modify RAK811:add 1 channel PWM function at RAK811 pin_num 2(For RUI ...
at_cmd.h	update RAK5205&RAK7204
bsp.c	update app_5205
bsp.h	updata RAK811_20191104_1
gps.c	update RAK811
gps.h	update RAK5205&RAK7204
lis3dh_reg.c	update RAK811
lis3dh_reg.h	update based on RAK811
lora_config.c	modify stm based RAK811 adr RUI
lora_config.h	updata RAK811_20191104_1

Figure 5: Source code for the RAK5205 and RAK7205 boards at Github.

The RAK5205 / RAK7205 boards are based on the RAK811 module. The boards have an environmental sensor (BME680), an accelerator sensor (LIS3DH), and a

GPS module (Ublox MAX 7Q). By reviewing these sample code, you will learn and understand how to create a customized version of firmware for "RAK811 + IIC sensor" and "RAK811 + UART GPS" using the RUI API.

2) "app_7204 is the folder that contains the application source code for the RAK7204 board.

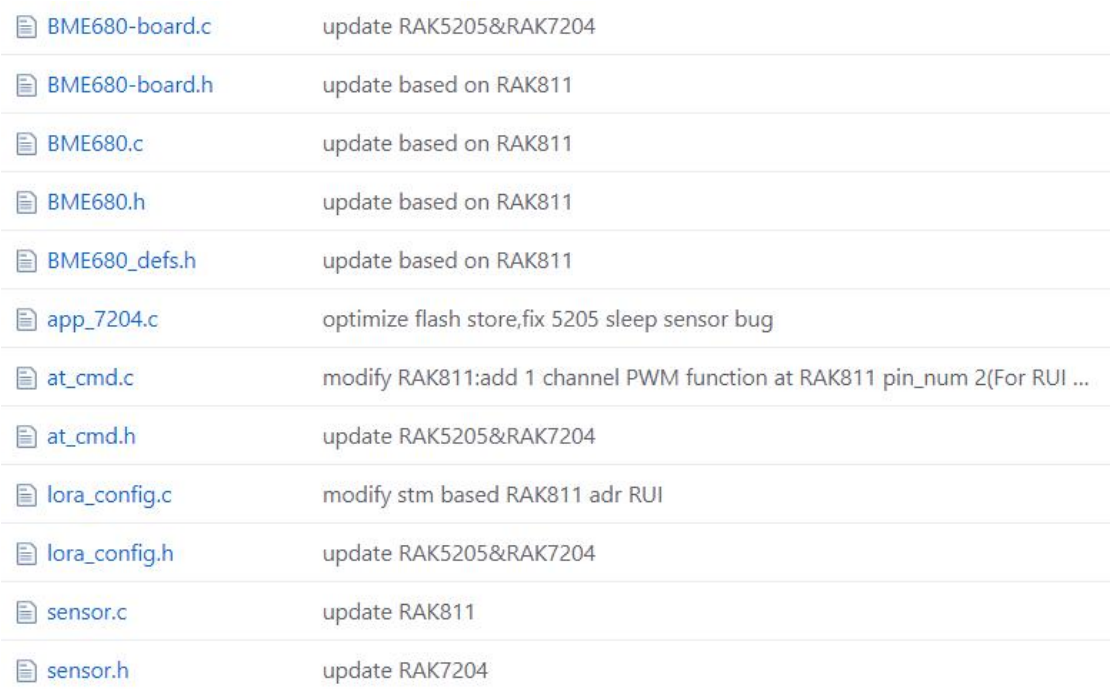
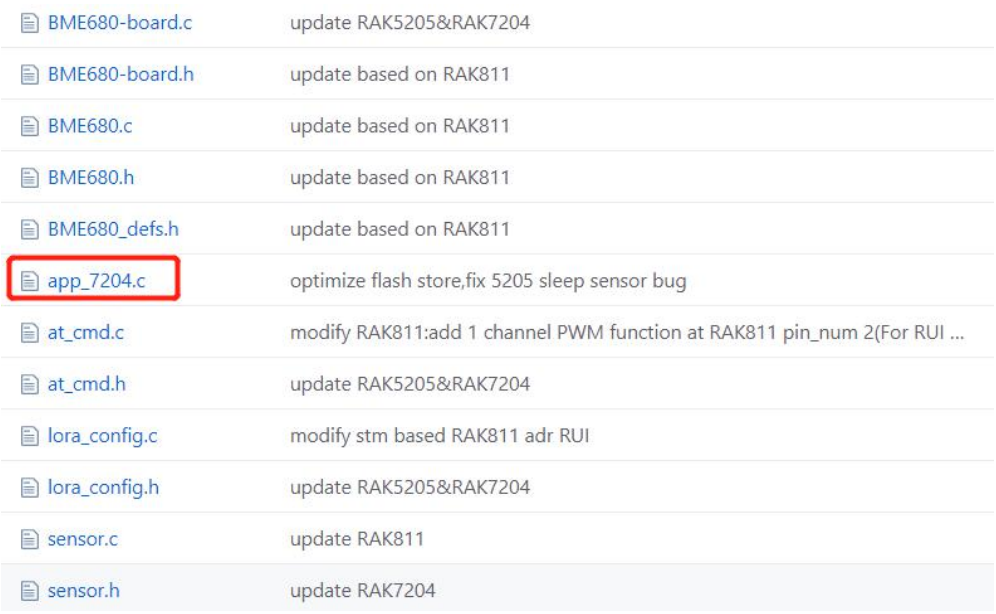


Figure 6: Source code for the RAK7204 at Github.

In this folder you will find the implementation of functionalities that allows to access the sensors, interface to the AT commands and application logic. All these are implemented on top of the RUI API. For example, if you just want to customize the application logic, then you will only need to modify the "app_7204.c"file.



BME680-board.c	update RAK5205&RAK7204
BME680-board.h	update based on RAK811
BME680.c	update based on RAK811
BME680.h	update based on RAK811
BME680_defs.h	update based on RAK811
app_7204.c	optimize flash store,fix 5205 sleep sensor bug
at_cmd.c	modify RAK811:add 1 channel PWM function at RAK811 pin_num 2(For RUI ...
at_cmd.h	update RAK5205&RAK7204
lora_config.c	modify stm based RAK811 adr RUI
lora_config.h	update RAK5205&RAK7204
sensor.c	update RAK811
sensor.h	update RAK7204

Figure 7: Source code for the RAK7204 at Github.

3) "app_PT100_Digital_Pressure_Sensor" is an example for "RAK811 + PT100 Digital Pressure sensor".

4) "app_RAK811"is the application source code of the RAK811 module.

3. Develop your customized application logic according to your use case and based on the apropiate example.

Every sample application on Github has been properly tested by an engineer of RAK. Taking it as the baseline of your target application will simplify the task at hand.

For example, to implement a new application based on the RAK7204 board:

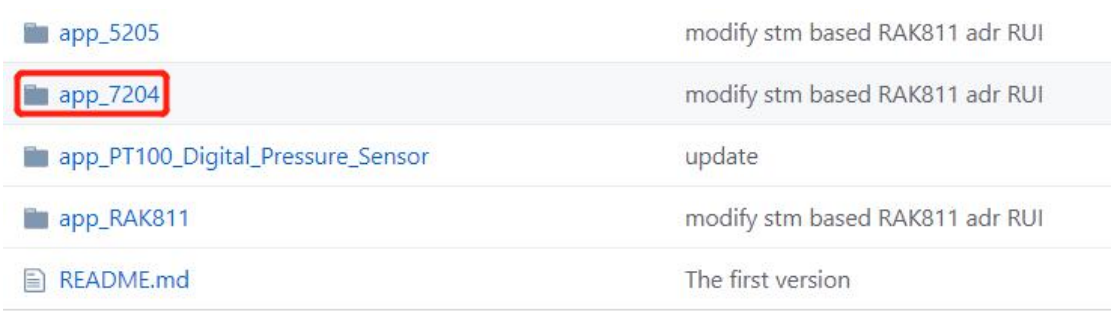


Figure 8: Source code for the RAK7204 at Github.

In the source code of RAK7204's Application example, we will add a log upon receiving the downlink data over the UART interface. You are welcome to add more logic here.

```
void LoRaReceive_callback(RUI_RECEIVE_T* Receive_datapackage)
{
    char hex_str[3] = {0};
    RUI_LOG_PRINTF("at+recv=%d,%d,%d,%d", Receive_datapackage->Port, Receive_datapackage->Rssi, Receive_datapackage->Snr, Receive_datapacka

    if ((Receive_datapackage->Buffer != NULL) && Receive_datapackage->BufferSize) {
        RUI_LOG_PRINTF(":");
        for (int i = 0; i < Receive_datapackage->BufferSize; i++) {
            sprintf(hex_str, "%02x", Receive_datapackage->Buffer[i]);
            RUI_LOG_PRINTF("%s", hex_str);
        }
    }
    RUI_LOG_PRINTF("\r\n");
}
```

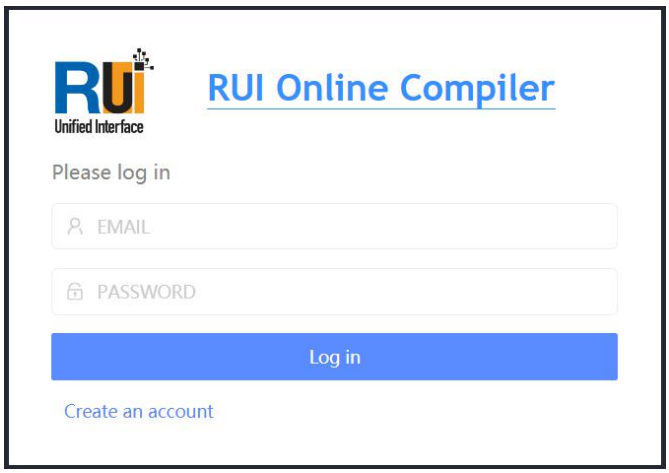
4. After completing the customized Application, you will be required to use the RUI Online Compiler to generate the customized firmware.

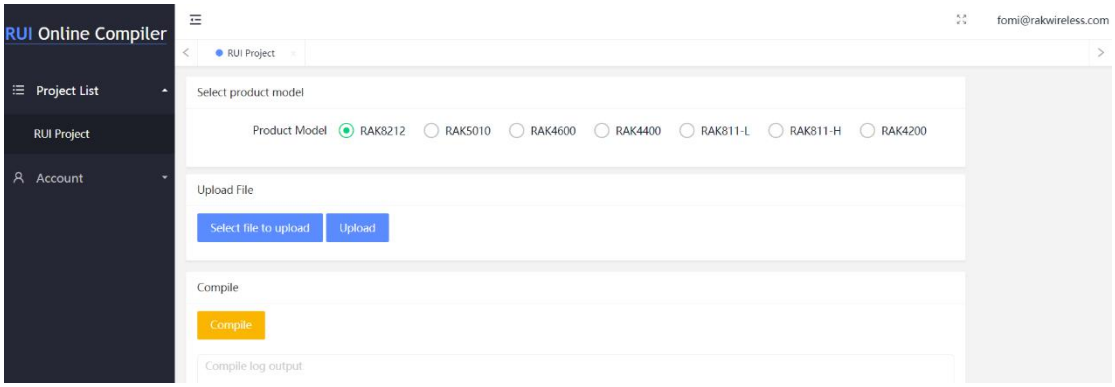
Now, that you have completed the development of your customized application, you must use the RUI Online Compiler to generate your customized firmware. The RUI Online Compiler can be access from this URL:

http://47.112.137.11:12090/#/user/login

You can read at the following document in order to know more about how to use RUI online compiler:

https://downloads.rakwireless.com.cn/cn/RUI/%E5%A6%82%E4%BD%95%E4%BD%BF%E7%94%A8RUI%E5%9C%A8%E7%BA%BF%E7%BC%96%E8%AF%91%E5%99%A8.pdf





5. Finally, flash the customized firmware into the RAK modules/products.

After compiling, you will have the final customized firmware. Now, you can flash the firmware into RAK modules/products. For the procedure on how to flash the firware, please refer to the user guide of each module at

https://doc.rakwireless.com/

About RAKwireless:

RAKwireless is the pioneer in providing innovative and diverse cellular and LoRa connectivity solutions for IoT edge devices. It's easy and modular design can be used in different IoT applications and accelerate time-to-market.
For more information, please visit RAKwireless website at www.rakwireless.com.