RAK11310 WisBlock LPWAN Module Datasheet

Overview

Description

RAK1130 is a WisBlock Core module for the RAK hardware IoT platform WisBlock. It is powered by an RP2040 ARM microcontroller developed by the Raspberry Pi Foundation, combined with the SX1262 LoRa transceiver from Semtech. RAK11310 can be programmed via the Arduino™ IDE, PlatformIO or MicroPython.

The RAK11310 WisBlock Core module has a RAK11300 LoRa stamp module in it together with a high-quality connector that is compatible with WisBlock Base boards. It allows an easy way to access the pins of the RAK11310 module in order to simplify the development of IoT devices.

The module complies with LoRaWAN 1.0.2 protocols, and also supports LoRa point-to-point communication.

The RF communication characteristic of the Lora module makes it suitable for a variety of applications in the IoT field such as home automation, sensor networks, building automation, personal area networks applications (health/fitness sensors, and monitors, etc.).

Features

- Based on RAK11300
- Uses the RP2040 as the main processor
- Semtech SX1262 low power high range LoRa transceiver
- LoRaWAN 1.0.2 protocol stack
- I/O ports: UART/I2C/GPIO/USB
- Serial Wire Debug (SWD) interface
- Module Size: 20 x 30 mm
- Supply Voltage: 2.0 V – 3.6 V
- Temperature Range: -20 °C – 70 °C

Specifications

Overview

The overview covers the RAK11310 board overview and the mounting mechanics of the board into the baseboard.

Board Overview
Mounting Sketch

The RAK11310 module is designed to work with the RAK5005-O base board. Figure 2 shows how a RAK11310 module should be mounted on top of the RAK5005-O.

![RAK11310 Mounting Sketch](image)

Figure 2: RAK11310 Mounting Sketch

Hardware

The hardware specification is categorized into four parts. It covers the RF, electrical, and mechanical parameters, which include the tabular data of the functionalities and standard values. It also shows and discusses the diagram of the RAK11310 WisBlock LPWAN Module.

RF Characteristics

The RAK11310 module supports the LoRaWAN bands, as shown in the table below. There are two types of RAK11310 module: **RAK11310(L)** is for the low-frequency band regions and **RAK11310** (no extra suffix) is for the high-frequency band regions.

**NOTE**

Check the frequency and band regions, as shown in the table, when ordering your RAK11310 WisBlock Core.
### Module

<table>
<thead>
<tr>
<th>Module</th>
<th>Core Module</th>
<th>Region</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAK11310(L)</td>
<td>RAK11300(L)</td>
<td>Europe</td>
<td>EU433</td>
</tr>
<tr>
<td>RAK11300(L)</td>
<td>China</td>
<td>CN470</td>
<td></td>
</tr>
<tr>
<td>RAK11310</td>
<td>RAK11300</td>
<td>Europe</td>
<td>EU868</td>
</tr>
<tr>
<td>RAK11300</td>
<td>North America</td>
<td>US915</td>
<td></td>
</tr>
<tr>
<td>RAK11300</td>
<td>Australia</td>
<td>AU915</td>
<td></td>
</tr>
<tr>
<td>RAK11300</td>
<td>Korea</td>
<td>KR920</td>
<td></td>
</tr>
<tr>
<td>RAK11300</td>
<td>Asia</td>
<td>AS923</td>
<td></td>
</tr>
<tr>
<td>RAK11300</td>
<td>India</td>
<td>IN865</td>
<td></td>
</tr>
<tr>
<td>RAK11300</td>
<td>Russia</td>
<td>RU864</td>
<td></td>
</tr>
</tbody>
</table>

### Electrical Characteristics

#### Power Consumption

**NOTE**

The power consumption will be published after the hardware test results.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Condition</th>
<th>Minimum</th>
<th>Typical</th>
<th>Maximum</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Average Current</td>
<td>TX</td>
<td>-</td>
<td>24.6</td>
<td>-</td>
<td>mA</td>
</tr>
<tr>
<td>Sleep Current</td>
<td>-</td>
<td>3.8</td>
<td>-</td>
<td></td>
<td>mA</td>
</tr>
</tbody>
</table>

### Voltage Ratings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Minimum</th>
<th>Typical</th>
<th>Maximum</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>2.0</td>
<td>3.3</td>
<td>3.6</td>
<td>V</td>
</tr>
</tbody>
</table>

### Mechanical Characteristics

#### Board Dimensions
WisConnector PCB Layout

Schematic Diagram
WisConnector: The breakout module exposes the RAK11310 stamp module pinout to the board-to-board WisConnector.

WisConnector Pin Order: The pin order of the WisConnector is located on the bottom layer of the module.

Core Module: The breakout module itself has a RAK11310 at its core, and it shows the core module pin and connection information. By default, the NFC function is disabled to preserve the low-power characteristic of this core.

SWD Interface: The breakout module exposes an SWD debug interface. Additionally, the RST pin is used to reset the RAK11310 core module.

Power Up Automatic Reset: The breakout module has a power-up automatic reset circuit, and the schematic shows the automatic reset mechanism. This module also can be reset through the WisBlock Base reset pin.
RAK11310 Quick Start Guide

This guide introduces the RAK11310 WisBlock Core LPWAN Module and how to use it. RAK11310 consists of a Raspberry Pi RP2040 chip and an SX1262 LoRa® transceiver, making it ideal for various IoT projects.

Prerequisites

What Do You Need?

Before going through each and every step on using RAK11310 WisBlock Core, make sure to prepare the necessary items listed below:

Hardware

- RAK11310 WisBlock Core LPWAN Module
- Your choice of WisBlock Base
- Your choice of WisBlock Modules
- USB Cable
- Li-Ion/LiPo battery (optional)
- Solar charger (optional)

Software

You can choose the Arduino IDE or Platform IO to code the RAK11310 WisBlock Core.

Programming RAK11310 via Arduino IDE:

- Download and install Arduino IDE.
- To add the WisBlock Core boards to your Arduino IDE, you need to install the RAKwireless Arduino BSP. You can follow this complete guide on adding the BSP in Arduino IDE. You can also have a look at the RAKwireless Arduino BSP github repository.

In the Arduino IDE, once you installed the BSP, some examples for RAK11310 will be automatically included in the list of examples when you select WisBlock Core RAK11310 Board in the Boards Manager.

Programming RAK11310 via Platform IO:

- Platform IO for RAK11310 complete setup guide

NOTE

Updated and complete WisBlock examples can be found in the WisBlock Examples repository, which contains source codes that you can copy-paste and upload.

Aside from that, each WisBlock Module has its own quick-start guide to help you in your WisBlock journey.

Product Configuration

Hardware Setup

RAK11310 to WisBlock Base
The RAK11310 will not work without a WisBlock Base board. The WisBlock Base provides a USB connection to program the RAK11310. It also provides a power source and various interfaces to RAK11310, so that it can be connected to other WisBlock Modules via different module slots.

RAKwireless offers many WisBlock Base Boards compatible with WisBlock Core. It is highly recommended to look at these WisBlock Base boards to see what matches your requirements in terms of available module slots, power supply options, and overall size.

To illustrate, RAK11310 can be connected to the RAK5005-O WisBlock Base, as shown in Figure 1.

![RAK11310 Connection to WisBlock Base RAK5005-O](image)

There are a few pins that are exposed on RAK5005-O, and you can easily use them via header pins. The labels are at the back, as shown in Figure 2.

![WisBlock Base exposed pins](image)

More information can be found on the official documentation of the specific WisBlock Base you used in your project.

For RAK5005-O WisBlock Base with RAK11310 WisBlock Core, the accessible GPIO pins are defined as follows in the Arduino IDE and Platform IO:

- `WB_IO1` for IO1 pin
- `WB_IO2` for IO2 pin (Also used to control the 3.3V supply of some WisBlock Modules to achieve low-power IoT devices.)
- `WB_A0` for AIN

There are two usable LEDs as well that can be controlled by the RAK11310 on the WisBlock Base board:

- `LED_GREEN`
- `LED_BLUE`

UART1 and I2C_1 are also exposed on the header of the WisBlock Base board.
RAK11310 has a native USB peripheral onboard (Serial), which is used for programming and Serial debugging, and two usable hardware UART1 and UART2 (Serial 1 and Serial 2). UART1 is accessible on WisBlock Slot A, WisBlock IO slot, and the exposed header pins. UART2 is accessible only on the WisBlock IO slot.

The I2C_1 header pins are shared on WisBlock Base Slots A to D as well. The second I2C_2 is available only on the WisBlock IO slot.

### RAK11310 to WisBlock Modules

RAK11310 WisBlock Core is designed to be interfaced with other WisBlock Modules like sensors, displays, and other interfaces. You need to connect these modules to compatible slots on the WisBlock Base.

Each WisBlock Module that will be used with RAK11310 WisBlock Core has a dedicated quick-start guide you can follow on how to connect to the WisBlock Base.

Listed are the quick-start guides of some WisBlock modules you can buy from our store:

- **RAK1901 Quick Start Guide**
- **RAK1902 Quick Start Guide**
- **RAK1903 Quick Start Guide**

**Figure 3** shows an illustration on how you can combine various WisBlock Modules with the RAK11310 WisBlock Core via the WisBlock Base board.

---

**Assembling and Disassembling of WisBlock Modules**

**Assembling**
Figure 4 shows how to mount the RAK11310 module on top of a WisBlock Base board (RAK5005-O). Follow carefully the procedure defined in WisBlock module assembly/disassembly instructions to secure the connection safely. Once attached, carefully fix the module with one or more pieces of M1.2 x 3 mm screws depending on the module.

![Figure 4: RAK11310 Mounting Sketch](image)

**Disassembling**

The procedure in disassembling any type of WisBlock module is the same.

1. First, remove the screws.

![Figure 5: Removing screws from the WisBlock module](image)

2. Once the screws are removed, check the silkscreen of the module to find the correct location where force can be applied.

![Figure 6: Detaching silkscreen on the WisBlock module](image)

3. Apply force to the module at the position of the connector, as shown in Figure 7, to detach the module from the baseboard.
LoRa Antenna

Another important component of RAK11310 is the LoRa antenna.

You need to ensure that the antenna is properly connected to have a good LoRa signal. Do not power the module without an antenna connected to the IPEX connector to avoid damage to the RF section of the chip.

RAK11310 has a label on its sticker showing where to connect the antennas, as shown in Figure 10.

NOTE

Detailed information about the RAK11310 LoRa antenna can be found on the antenna datasheet.

WARNING
When using LoRa, make sure that an antenna is always connected. Using these transceivers without an antenna can damage the system. Make sure to fix the module with the screws to ensure a proper function.

Battery and Solar Connection

RAK11310 can be powered via the USB cable or a Li-Ion/LiPo battery via the dedicated connectors, as shown in Figure 11. The matching connector for the battery wires is a JST PHR-2 2 mm pitch female.

This illustration uses RAK5005-O as WisBlock Base. There are other WisBlock Base boards available, and you need to check the datasheet of the specific WisBlock Base board for the right polarity and other parameters.

The battery can be recharged, as well, via a 5 V solar panel, as shown in Figure 13. The matching connector for the solar panel wires is a JST ZHR-2 1.5 mm pitch female.
You can find the specification of the battery and solar panel in the datasheet of the WisBlock Base.

⚠️ **WARNING**
- Only 3.7-4.2 V Rechargeable LiPo batteries are supported. Do not use other types of batteries with the system.
- Only 5 V solar panels are supported. Do not use 12 V solar panels. It will destroy the charging unit and eventually other electronic parts.
- Make sure the battery wires match the polarity on the RAK5005-O board. Not all batteries have the same wiring.

### Software Setup

RAK11310 WisBlock Core is designed to be interfaced with other WisBlock Modules like sensors, displays, and other interfaces. To make useful devices, you need to upload a source code to the RAK11310. Before you continue, you should have either the Arduino BSP or Platform IO already set up.

### RAK11310 Example Repository

To quickly build your IoT device effortlessly, examples for RAK11310 for all WisBlock Modules are provided.

You can access the code on the [WisBlock Example code repository](#). Examples compatible only with RAK11310 are in the `RAK11310` folder. Shared WisBlock Core examples are in the `common` folder.

To use these examples, you have two options: Arduino IDE or Platform IO.

### RAK11310 on Arduino IDE

Some examples for various WisBlock Modules like the RAK1901 and RAK1902 are available in the Arduino IDE once you install the [BSP for the Arduino IDE](#), as shown in [Figure 14](#). The updated and complete WisBlock examples are still in the [WisBlock Examples](#).
It is highly recommended to also check the dedicated quick-start guide that you can follow on various WisBlock Modules. Each quick-start guide contains the detailed steps on how to open the examples and upload them to the RAK11310.

Listed are the examples where you can check the Software Setup on the quick-start guide of the following WisBlock Modules:

- RAK1901 Quick Start Guide
- RAK1902 Quick Start Guide
- RAK1903 Quick Start Guide

**RAK11310 on Platform IO**

For the Platform IO, get the examples on the [WisBlock Example code repository](#) and add the necessary libraries individually. Then you can compile the example code.

**Connecting RAK11310 to LoRaWAN**

RAK11310 is the WisBlock Core capable of LoRaWAN connectivity.
There is an example on how to start with LoRaWAN in the **RAK WisBlock examples** in the Arduino IDE named **LoRaWAN_OTAA_ABP**. This is also available in the **WisBlock Repository**.

Configuration of LoRaWAN Example Code

There are configurations you need to set up to ensure that the device can join a LoRaWAN Network server.

The guide below will explain the default settings and how to configure them.

1. Set up the region.

   Default is EU868.

   ```c
   LoRaMacRegion_t g_CurrentRegion = LORAMAC_REGION_EU868;
   ```

   You can change this to the region applicable to you like `LORAMAC_REGION_US915`, `LORAMAC_REGION_AU915`, etc.

2. Set up the activation method.

   Default is **OTAA**.

   ```c
   bool doOTAA = true;
   ```

   To configure the device to ABP, you need to change this boolean variable to `false`.

3. Set up the message type to confirmed or unconfirmed.

   Default is **confirmed message**.

   ```c
   lmh_confirm g_CurrentConfirm = LMH_CONFIRMED_MSG;
   ```
You can change to unconfirmed message by changing the value to `LMH_UNCONFIRMED_MSG`.

4. Set up device class.

Default is **Class A**.

```c
DeviceClass_t g_CurrentClass = CLASS_A;
```

You can change this to `CLASS_B` (still under development) or `CLASS_C`.

5. Set up the keys.

If configured for OTAA activation:

```c
uint8_t nodeDeviceEUI[8] = {0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x33, 0x33};
uint8_t nodeAppEUI[8] = {0xB8, 0x27, 0xEB, 0xFF, 0xFE, 0x39, 0x00, 0x00};
uint8_t nodeAppKey[16] = {0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88,
                          0x88, 0x88, 0x88, 0x88, 0x88};
```

If configured for OTAA activation:

```c
uint32_t nodeDevAddr = 0x260116F8;
uint8_t nodeNwsKey[16] = {0x7E, 0xAC, 0xE2, 0x55, 0xB8, 0xA5, 0xE2, 0x69, 0x91, 0x51, 0x96, 0x06,
                          0x16, 0x02, 0x41, 0x1B, 0xFF};
uint8_t nodeAppsKey[16] = {0xFB, 0xAC, 0xB6, 0x47, 0xF3, 0x58, 0x45, 0xC7, 0x50, 0x7D, 0xBF, 0x16};
```

6. Set up uplink period.

The default is **20000 mS**.

```c
#define LORAWAN_APP_INTERVAL 20000
```

You can make the transmission interval shorter or longer by changing this value.

### LoRaWAN Payload

This default example will send a string `Hello!` to the LoRaWAN server. This is set up in the `send_lora_frame(void)` function.

```c
memset(m_lora_app_data.buffer, 0, LORAWAN_APP_DATA_BUFF_SIZE);
 m_lora_app_data.port = gAppPort;
m_lora_app_data.buffer[i++] = 'H';
m_lora_app_data.buffer[i++] = 'e';
m_lora_app_data.buffer[i++] = 'l';
m_lora_app_data.buffer[i++] = 'l';
m_lora_app_data.buffer[i++] = 'o';
m_lora_app_data.buffer[i++] = '!',
m_lora_app_data.buffer[i++] = 'l';
m_lora_app_data.buffer[i++] = 'l';
m_lora_app_data.buffer[i++] = 'o';
m_lora_app_data.buffer[i++] = '!',
m_lora_app_data.buffer[i++] = 'l';
m_lora_app_data.buffer[i++] = 'l';
m_lora_app_data.buffer[i++] = 'o';
m_lora_app_data.buffer[i++] = '!',
m_lora_app_data.buffer[i++] = 'l';
m_lora_app_data.buffer[i++] = 'l';
m_lora_app_data.buffer[i++] = 'o';
m_lora_app_data.buffer[i++] = '!',
m_lora_app_data.buffer[i++] = 'l';
m_lora_app_data.buffsize = i;
```

The LoRaWAN payload is packaged in this function. You can change this section to format your payload.
Formatting the Payload

For illustration, you can check how the 32-bit latitude value (named \texttt{ilat}) from a GNSS sensor is formatted to a 4-byte size array.

This is the exact code snippet to format the Latitude data of the GPS Tracker example.

```c
gps.f_get_position(&flat, &flon, &age);
flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat;
ilat = flat * 1000000;

// longitude section...

m_lora_app_data.port = gAppPort;
m_lora_app_data.buffer[0] = 0x09;
// lat data
m_lora_app_data.buffer[1] = (ilat & 0xFF000000) >> 24;
m_lora_app_data.buffer[2] = (ilat & 0x00FF0000) >> 16;
m_lora_app_data.buffer[3] = (ilat & 0x0000FF00) >> 8;
m_lora_app_data.buffer[4] = ilat & 0x000000FF;
```

Another important part of the code is configuring the size of the payload.

This is done with the \texttt{m\_lora\_app\_data\_buffsize} variable. For illustration, with only \texttt{ilat} above as the value to send, you need to set the buffer size to five (5), because the array starts at zero (0) and ends at four (4).

```c
m_lora_app_data.buffsize = 5;
```

Decoding the Payload on the LoRaWAN Network Server

On the LoRaWAN network server side like TTN, Chirpstack, Helium, the value transmitted can be retrieved by decoding the payload.

```c
decoded.latitude = latitude_int / 100000;
```

Important Function in the LoRaWAN Example.

LoRa periodic transmission function should be very short, and all reading and processing of the data must be done in the main loop.

```c
void tx_lora_periodic_handler(void)
```

Same with the transmission function, the receiving event handler should be short as well. All processing of the received data should be done in the main loop.

```c
void lorawan_rx_handler(lmh_app_data_t *app_data)
```
Uploading the LoRaWAN Code

After all the configuration is done and the payload is formatted properly, you can now compile and upload the code. If you get errors compiling the LoRaWAN example, ensure that you have an updated BSP by checking the version in the boards manager.

You also need to ensure that you have the updated SX126x library.
NOTE
For Windows users, you need to run the post_install.bat script to be able to compile sketches for RAK11310. You can execute the script by double-clicking it or via command line.

![Executing post_install.bat script directly](image1)

![Running post_install.bat via commandline](image2)

LoRaWAN Configuration using AT Commands

Aside from the LoRaWAN example code in the previous section, you also have the option to configure the RAK11310 LoRaWAN parameters via AT commands.

The RAK11310 WisBlock Core is based on the RAK11300 WisDuo module, and it supports AT commands set that will let you configure LoRaWAN-related parameters. This is very useful in scenarios where you need to deploy many devices and you want to quickly onboard devices just by sending AT commands. The AT command set is already enabled in your RAK11310 WisBlock Core, and you can use it at any time. It will only be overwritten once you upload new firmware to it.

To utilize the AT command feature in your own custom firmware, you need to include the AT command parser functions that can be found in the RAK11300 AT command firmware repository.

Miscellaneous

Upgrading the Firmware

If you want to upgrade to the latest version firmware of the module, you can follow this section. RAK11310 uses the WisDuo RAK11300 module, and it shares the same firmware. The latest firmware can be found in the software section of RAK11300 Datasheet.

Firmware Upgrade Through USB

Minimum Hardware and Software Requirements
Refer to this table for the minimum hardware and software required to perform the firmware upgrade via USB.

<table>
<thead>
<tr>
<th>Hardware/Software</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer</td>
<td>A Windows/Ubuntu/Mac computer</td>
</tr>
<tr>
<td>USB Connection</td>
<td>RAK11310 attached to WisBlock Base Board</td>
</tr>
<tr>
<td>Firmware File</td>
<td>.uf2 firmware file downloaded from the website</td>
</tr>
</tbody>
</table>

**Firmware Upgrade Procedure**

Execute the following procedure to upgrade the firmware via USB using the .uf2 firmware file.

1. Download the latest application firmware of the RAK11300.
   - [RAK11300 Firmware](#)

2. Connect the WisBlock Base where RAK11310 is attached to the computer via USB.

3. Activate RAK11300 USB bootloader mode. There are two possible ways to do this:
   - Click the reset button of the WisBlock Base Board twice to enable the USB Bootloader mode of the RAK11300 module.
   - Hold the BootSel button in the RAK11300 module, reset the module (press the reset button of WisBlock Base Board for at least 2 seconds) then wait for the USB Mass Storage [RPI-RP2](#) to appear before releasing the BootSel button.

![RAK11310 Bootsel Button](#)

**Figure 19: RAK11310 Bootsel Button**

You need to do either of the two options until the USB Mass Storage [RPI-RP2](#) shows on your computer.

4. Copy the firmware.uf2 file to the [RPI-RP2](#) Mass Storage device.
Figure 20: Drag the firmware.uf2 file to RPI-RP2 USB Mass Storage

5. You need to reset the device after updating the .uf2 firmware file.