Single Channel LoRaWAN Gateway for Raspberry Pi
Created by Brent Rubell
## Guide Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview</strong></td>
<td>3</td>
</tr>
<tr>
<td>What is a LoRaWAN Gateway?</td>
<td>3</td>
</tr>
<tr>
<td>About Single Channel Gateways</td>
<td>4</td>
</tr>
<tr>
<td>Parts</td>
<td>4</td>
</tr>
<tr>
<td><strong>Pi Wiring</strong></td>
<td>7</td>
</tr>
<tr>
<td>Antenna Wiring</td>
<td>7</td>
</tr>
<tr>
<td>Pi Wiring</td>
<td>7</td>
</tr>
<tr>
<td>Wiring the Push-Buttons</td>
<td>7</td>
</tr>
<tr>
<td>Wiring the OLED</td>
<td>8</td>
</tr>
<tr>
<td>Wiring the RFM Radio Module</td>
<td>8</td>
</tr>
<tr>
<td><strong>Things Network Raspberry Pi Setup</strong></td>
<td>10</td>
</tr>
<tr>
<td>Installing CircuitPython Libraries</td>
<td>10</td>
</tr>
<tr>
<td>Testing the Setup</td>
<td>10</td>
</tr>
<tr>
<td>Install the Single Channel Pi Gateway</td>
<td>13</td>
</tr>
<tr>
<td>Gateway Usage</td>
<td>14</td>
</tr>
<tr>
<td><strong>Configuring The Things Network</strong></td>
<td>15</td>
</tr>
<tr>
<td>Registering a Gateway with The Things Network</td>
<td>15</td>
</tr>
<tr>
<td>Usage with Pi Zero</td>
<td>19</td>
</tr>
<tr>
<td>Troubleshooting your Gateway</td>
<td>20</td>
</tr>
<tr>
<td>My gateway doesn't appear as “connected” in the console.</td>
<td>20</td>
</tr>
<tr>
<td><strong>Device Setup</strong></td>
<td>22</td>
</tr>
<tr>
<td>Building a CircuitPython LoRaWAN Device</td>
<td>22</td>
</tr>
<tr>
<td>Building an Arduino CircuitPython Device</td>
<td>23</td>
</tr>
<tr>
<td><strong>Gateway Usage</strong></td>
<td>24</td>
</tr>
<tr>
<td>Configuration between Device and Gateway</td>
<td>24</td>
</tr>
<tr>
<td>Gateway Usage</td>
<td>24</td>
</tr>
</tbody>
</table>
Overview

This guide is deprecated and is no longer possible! Single-channel packet forwarders no longer work after the Things Network migration to The Things Stack v3. For more information about this decision, visit: https://www.thethingsnetwork.org/forum/t/single-channel-packet-forwarders-scpf-are-deprecated-and-not-supported/31117

LoRaWAN is similar to a cellular network, our suggested provider is The Things Network (https://adafruit.it/BsB), a community-driven initiative with a large amount of public gateways to connect to (check out the map of publicly listed gateways here (https://adafruit.it/Des)).

We’re going to build a gateway (highlighted in red in the diagram below) to pass data from your LoRaWAN device to the internet.

What is a LoRaWAN Gateway?
If you have a Pi (or Feather) LoRaWAN device set up, you'll likely want to immediately start working with LoRaWAN. Our suggested network is [The Things Network](https://adafru.it/BsB), a community-driven initiative with a large amount of public gateways to connect to ([check out the map of publicly listed gateways here](https://adafru.it/Des)).

Gateways form a bridge a LoRaWAN device and the Network, receiving the device's data and forwarding them an application.

If you don't have a gateway near you, it's possible you could face a dead-end. While multi-channel gateways exist (we stock the [official The Things Gateway](https://adafru.it/DDQ)), they're often expensive - placing a cost barrier on entering the network (as a hobbyist/hacker/experimenter).

### About Single Channel Gateways

Single channel gateways are NOT LoRaWAN-compliant or officially supported by The Things Network.

Single channel gateways (like the one we'll be building) are a LoRa device which forwards data to a network. They're great for getting into LoRaWAN without purchasing a more expensive "full gateway" (like [The Things Network Gateway](https://adafru.it/DDQ) we stock).

*However...*

Single-channel gateways are **not** LoRaWAN compliant and are not currently supported by The Things Network (due to poor coverage, among other reasons). For more information about the debate about single channel gateways, [check out The Things Network forum](https://adafru.it/DDR).

### Parts

The LoRa Radio Bonnet is recommended for this guide - you not only get a radio module, but also a 128x32 OLED display for status messages and three buttons you can use for creating a custom user interface or sending test messages.

Your browser does not support the video tag.

**Adafruit LoRa Radio Bonnet with OLED - RFM95W @ 915MHz**

The latest Raspberry Pi computers come with WiFi and Bluetooth, and now you can add even more radio options with the Adafruit Radio Bonnets! Upgrade your Raspberry Pi with a LoRa /...

$32.50

*In Stock*

[Add to Cart](https://learn.adafruit.com/raspberry-pi-single-channel-lorawan-gateway)
There’s also a 433MHz version which uses the amateur or license-free ISM band (ITU "Europe" license-free ISM or ITU "American" amateur with limitations).

**Adafruit LoRa Radio Bonnet RFM96W @ 433MHz**
The latest Raspberry Pi computers come with WiFi and Bluetooth, and now you can add even more radio options with the Adafruit Radio Bonnets! Upgrade your Raspberry Pi with a LoRa /…
$32.50
**In Stock**
Add to Cart

For a less-permanent gateway build, you can build a gateway using a RFM95W LoRa radio breakout and an OLED display, along with a few buttons.

**Adafruit RFM95W LoRa Radio Transceiver Breakout - 868 or 915 MHz**
"You see, wire telegraph is a kind of a very, very long cat. You pull his tail in New York and his head is meowing in Los Angeles. Do you understand this? And…
$19.95
**In Stock**
Add to Cart

There’s also a 433MHz version which uses the amateur or license-free ISM band (ITU "Europe" license-free ISM or ITU "American" amateur with limitations).

1 x **Adafruit RFM96W 433MHz**
RFM96W LoRa Radio Transceiver Breakout - 433 MHz
Add to Cart

**Raspberry Pi Zero W**
If you didn’t think that the Raspberry Pi Zero could possibly get any better, then boy do we have a pleasant surprise for you! The new Raspberry Pi Zero W…
Out of Stock

**Raspberry Pi 3 - Model B+ - 1.4GHz Cortex-A53 with 1GB RAM**
The Raspberry Pi 3 Model B is the most popular Raspberry Pi computer made, and the Pi Foundation knows you can always make a good thing better! And what could make the Pi 3…
Out of Stock

You’ll also want to pick up the following parts from the Adafruit Shop if you do not have them already:
1 x Pi T-Cobbler Plus
GPIO Breakout - Pi A+, B+, Pi 2, Pi 3, Zero
Add to Cart

1 x I2C OLED Display
Monochrome 128x32 I2C OLED graphic display
Out of Stock

1 x Breadboarding Wire Bundle
75 flexible stranded core wires with stiff ends.
Add to Cart

1 x Tactile Buttons
Tactile Switch Buttons (12mm square, 6mm tall) x 10 pack
Add to Cart

1 x Breadboard
Full Sized Breadboard
Add to Cart

1 x uFL Antenna
uFL SMT Antenna Connector
Add to Cart

1 x Raspberry Pi Power Supply
5V 2.5A Switching Power Supply with 20AWG MicroUSB Cable
Add to Cart

1 x 8GB SD Card with OS
8GB SD Card w/ Raspbian Stretch Lite
Add to Cart
**Pi Wiring**

**Antenna Wiring**

You *can* use a wire-based antenna, but this 1/2 wave whip antenna we stock works with the LoRa radios we have and extends the range of the gateway.

**900Mhz Antenna Kit - For LoPy, LoRa, etc**

This LoRa Antenna Kit is meant for use with the LoPy LoRa, WiFi and BLE board or the SiPy Sigfox,...

$12.75

In Stock

Add to Cart

---

If you have a LoRa Radio Bonnet, you can connect the antenna directly to the uFL antenna on the bonnet.

---

**Pi Wiring**

If you're using an Adafruit LoRa Radio Bonnet ([https://adafruit.it/DDS](https://adafruit.it/DDS)), you do not need to wire anything - [proceed to setting up the Raspberry Pi as a gateway.](https://adafruit.it/DDT)

If you're using a breadboard, follow the wiring below

**Wiring the Push-Buttons**

We'll start by wiring three buttons. These will the physical interface between the Raspberry Pi and the Radio - we'll want to control the radio by pressing some buttons.
Make the following connections between the Pi and the three push-buttons:

- Pi GPIO #5 to Button A (orange wire)
- Pi GPIO #6 to Button B (blue wire)
- Pi GPIO #12 to Button C (green wire)
- Pi 3.3V to Power Rail (red wire)
- Pi GND to Ground Rail (black wire)

Wiring the OLED

- Display GND to Raspberry Pi GND (black wire)
- Display Vin to Raspberry Pi 3.3V (red wire)
- Display SCL to Raspberry Pi SCL (green wire)
- Display SDA to Raspberry Pi SDA (yellow wire)

Wiring the RFM Radio Module
Make the following connections between the Raspberry Pi and the RFM Radio Module:

- Vin to Raspberry Pi 3.3V
- GND to Raspberry Pi Ground
- RFM G0 to Raspberry Pi GPIO #5
- RFM RST to Raspberry Pi GPIO #25
- RFM CLK to Raspberry Pi SCK
- RFM MISO to Raspberry Pi MISO
- RFM MOSI to Raspberry Pi MOSI
- RFM CS to Raspberry Pi CE1

Next, let's move on to installing the necessary files for this project.
Things Network Raspberry Pi Setup

This guide assumes that you've gotten your Raspberry Pi up and running, and have CircuitPython installed.

- If you have not done this yet, visit the installation guide [here and come back when you’re set up](https://adafru.it/Deo)

Installing CircuitPython Libraries

We're running CircuitPython on the Raspberry Pi, installing the libraries for radio communication is simple.

To install the library for the display, enter the following into the terminal:

`pip3 install adafruit-circuitpython-ssd1306`

You'll also need to install the `framebuf` module in order to write to the display.

`sudo pip3 install adafruit-circuitpython-framebuf`

To install the library for the RFM9x Module, enter the following into the terminal:

`sudo pip3 install adafruit-circuitpython-rfm9x`

Testing the Setup

There have been a lot of steps so far - let's verify that everything is wired up and all the libraries are installed properly before moving on to sending and receiving data.

The following code is for checking if the RFM9x radio is set up for transmitting and receiving. Save the code on your Pi as `rfm9x_check.py`.

```python
# SPDX-FileCopyrightText: 2018 Brent Rubell for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
Wiring Check, Pi Radio w/RFM9x

Author: Brent Rubell for Adafruit Industries
"""
import time
import busio
from digitalio import DigitalInOut, Direction, Pull
import board
```
# Import the SSD1306 module.
import adafruit_ssd1306

# Import the RFM9x radio module.
import adafruit_rfm9x

# Button A
btnA = DigitalInOut(board.D5)
btnA.direction = Direction.INPUT
btnA.pull = Pull.UP

# Button B
btnB = DigitalInOut(board.D6)
btnB.direction = Direction.INPUT
btnB.pull = Pull.UP

# Button C
btnC = DigitalInOut(board.D12)
btnC.direction = Direction.INPUT
btnC.pull = Pull.UP

# Create the I2C interface.
i2c = busio.I2C(board.SCL, board.SDA)

# 128x32 OLED Display
reset_pin = DigitalInOut(board.D4)
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c, reset=reset_pin)

# Clear the display.
display.fill(0)
display.show()

width = display.width
height = display.height

# Configure RFM9x LoRa Radio
CS = DigitalInOut(board.CE1)
RESET = DigitalInOut(board.D25)
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)

while True:
    # Clear the image
    display.fill(0)

    # Attempt to set up the RFM9x Module
    try:
        rfm9x = adafruit_rfm9x.RFM9x(spi, CS, RESET, 915.0)
display.text('RFM9x: Detected', 0, 0, 1)
    except RuntimeError as error:
        # Thrown on version mismatch
        display.text('RFM9x: ERROR', 0, 0, 1)
        print('RFM9x Error: ', error)

    # Check buttons
    if not btnA.value:
        # Button A Pressed
        display.text('Ada', width-85, height-7, 1)
display.show()
time.sleep(0.1)
if not btnB.value:
To use the code, enter the following in your terminal:

```python
python3 rfm9x_check.py
```

You'll also want to download the font file, `font5x8.bin`, and copy it into the same directory as the script:

```bash
https://adafruit.it/DvA
```

Then, let's check the setup:

If the RFM9x is detected, the OLED will display *Detected*.

You can test the buttons by pressing them and watching the display for changes.

If the wiring of the radio module is incorrect - the display will show *ERROR*. Check over your wiring on the Wiring Page and re-run the test. You may also need to ensure the correct CircuitPython library is installed for the module.

If the OLED does not turn on - first check that it is wired correctly. Then, make sure you enabled I2C from `raspi-config` and installed the required libraries (`adafruit-circuitpython-framebuf` and `adafruit-circuitpython-ssd1306`).
Install the Single Channel Pi Gateway

**WiringPi** ([https://adafruit.it/ely](https://adafruit.it/ely)) is **PRE-INSTALLED** with standard Raspbian systems. To check if it is pre-installed on your system, run:

```
gpio -v
```

If you get something, then you have it already installed. Otherwise, install WiringPi by running:

```
sudo apt-get install wiringpi
```

Then, **clone the packet forwarder repository**

```
git clone https://github.com/adafruit/single_chan_pkt_fwd.git
```

**Navigate** into the packet forwarder directory

```
cd single_chan_pkt_fwd/
```

and **compile** the code for the gateway

```
sudo make all
```

You'll also want to download the font file, **font5x8.bin**, and copy it into the same directory as the script:

[https://adafruit.it/DvA](https://adafruit.it/DvA)

If you are using a Raspberry Pi **ZERO**, you will need to change a value within the configuration file. Using a text editor, open the **global_conf.json** file. Replace "is_pi_zero": false with "is_pi_zero": true. Your configuration file should look like the following.
Gateway Usage

From within the [single_chan_pkt_fwd/](https://learn.adafruit.com/raspberry-pi-single-channel-lorawan-gateway) folder, run the Python program by entering the following into the terminal:

```bash
python3 lorawan_gateway.py
```

The display should show the Gateway EUI. Keep this screen open, we'll need it for the next step.
Configuring The Things Network

Registering a Gateway with The Things Network

Before a Gateway can be used with The Things Network (https://adafru.it/BsB) (TTN), it needs to be registered.

First, register an account with TTN.

- Navigate to their account registration page (https://adafru.it/CyW) to set up an account.

Once logged in, navigate to the The Things Network Console (https://adafru.it/CyX). This page is where you can register applications and add new devices or gateways.

Click *Gateways*

Click *register gateway*
Tick the *I'm using the legacy packet forwarder* checkbox.

Then, enter the Gateway EUI displayed on the gateway.

Fill out a description of what the gateway is.

Set the frequency plan to your region's frequency, we set ours to United States.

Set the Router to the location closest to you.
- Adafruit is located in New York City, so I set the router to *ttn-router-us-west*.
Click Register Gateway. TTN will register the gateway and redirect you to the Gateway Overview page.

The Things Network does not list Single Channel gateways on their map - these gateways are not great at handling multiple devices or accurately receiving a packets every time they're sent by a device. However, they're great for hacking on, experimenting, and using in a staging/lab setup.

We'll want to set the Gateway Privacy to Private and set the location to Unlisted.

From the Gateway Overview page, click Edit Info.

On the Privacy Page, uncheck all of the options and click Save Settings.
Before starting up the gateway, we'll optionally set the gateway information.

From the Gateway Overview page, click Edit Info

For brand, select *Single-Channel DIY Gateway*

For model, type *Raspberry Pi*

It's time to test! Run the Python program by entering the following into the terminal:

```
python3 lorawan_gateway.py
```

and press the middle button to launch the gateway.
You should see the **Status** of the gateway on your console switch from **Not Connected** to **Connected**. If you've had previously connected the gateway: the **Last Seen** label should update to **Now**.

Our gateway is now listening for new packets, it'll refresh whenever it's received a new packet.

Next, we're going to set up a device to send data to the gateway.

**Usage with Pi Zero**

If you are using a Pi Zero, you will need to modify the **global_conf.json** file prior to running the command above.

**Change the following line** in the JSON configuration from:

```json
"is_pi_zero": false,
```

to

```json
"is_pi_zero": true,
```

And save the file.
Then, run the Python program by entering the following into the terminal:

```
python3 lorawan_gateway.py
```

Troubleshooting your Gateway

My gateway doesn't appear as "connected" in the console.

Double check the configuration steps on this page. Make sure the Pi is connected to the internet and that the **Gateway EUI matches the Gateway EUI displayed on the Pi's display**.
Device Setup

Next, you'll need a LoRaWAN device to send data to the single-channel gateway.

*Don't have a LoRaWAN device?*

We have a few options for getting one up and running - with support for CircuitPython and Arduino.

Building a CircuitPython LoRaWAN Device

If you'd like to use CircuitPython to send data to the gateway, we have two guides which use CircuitPython and the [CircuitPython_TinyLoRa library](https://adafruit.it/Dem).

This guide sets up weather-logging LoRaWAN device with an Adafruit Feather. ([https://adafruit.it/DEx](https://adafruit.it/DEx))

It works with any of the Feather Family boards which run CircuitPython and a RFM9x breakout, Feather M0 LoRa or a Radio FeatherWing.
Building an Arduino CircuitPython Device

Now that we have a LoRaWAN Device, let's use it to send data to our gateway.
Gateway Usage

Configuration between Device and Gateway

Before we broadcast to the single-channel radio, there are some things to keep in mind:

Since this gateway only listens on one channel - **frequency and spread factor should be the same between both the gateway and the device**. This means that if you're sending data over 904.1MHz (channel 1 in the US) with spread factor of 7, the gateway should be *listening* for transmissions on 904.1MHz with a spread factor of 7.

From the gateway, you can check and/or modify the gateway's configuration within `global_conf.json` under `freq` and `spread_factor`.

You'll want to modify your code to send over a single-channel (frequency). By default, TinyLoRa sends with a spread factor of 7.

- We have examples for sending data over a single-channel using either [TinyLoRa for CircuitPython](https://adafruit.it/DEz) or [TinyLoRa for Arduino](https://adafruit.it/DEA).

Gateway Usage

Run the Python program by entering the following into the terminal:

```python
python3 lorawan_gateway.py
```

Press the first button to display statistics about the Pi, such as its IP, CPU load, and available memory.
Press the third button to display the name of the gateway along with the frequency, spreading factor, and The Things Network router.

Press the second button to launch the gateway. It'll display the current status (if a packet is received or not) and update the timestamp every minute.

When a LoRa Packet is received by the gateway, the terminal will display that a packet has been received, and it'll print out useful data coming from the packet:

```
incoming pkt...
"rxpk":
[["tmst":4067850409,"freq":905.1,"chan":0,"rfch":0,"stat":1,"modu":"LORA","datr":"SF7BW125","codr":5,"rssi":-35,"lsnr":10.0,"size":24,"data":"QLERAiYAKAABE0NCX+3lkSnK09WVlk8N"]}
```

The display will also refresh to display that a packet has been received. In this example, we're sending data from a RFM9x FeatherWing to the Gateway:
We can't read the incoming data since it's encrypted. After all, device data shouldn't be readable by a gateway operator. We'll need to check our **device** on The Things Network Console to read the decrypted data.

From The Things Network Console, **navigate to the data tab for the device**. You should see a packet pass through this page when a new packet is received.

**Click on the packet** to display the packet's payload.

**Scrolling down** on the packet's information will show the gateways used to communicate (one device can communicate to more than one gateway) and the packet's information (**metadata**).

We've received data, but what about making it readable by humans - [we have instructions about using the Things Network's payload decoder here](https://adafru.it/DEB).