<https://pypi.org/project/raspi-lora/>

sudo apt-get install python-pip python3-pip

pip install RPi.GPIO

pip install spidev

#pip install pyLoRa

pip install raspi-lora

from raspi\_lora import LoRa, ModemConfig

# This is our callback function that runs when a message is received

def **on\_recv**(payload):

print("From:", payload.header\_from)

print("Received:", payload.message)

print("RSSI: {}; SNR: {}".format(payload.rssi, payload.snr))

#LoRa(channel, interrupt, this\_address, freq=915, tx\_power=14,

# modem\_config=ModemConfig.Bw125Cr45Sf128, acks=False, crypto=None)

# Use chip select 0.

# GPIO pin 17 will be used for interrupts

# The address of this device will be set to 2

# By default freq=915

lora = LoRa(0, 17, 2, modem\_config=ModemConfig.Bw125Cr45Sf128, tx\_power=14, acks=True)

lora.on\_recv = on\_recv

lora.set\_mode\_rx()

# Send a message to a recipient device with address 10

message = "Hello there!"

status = lora.send\_to\_wait(message, 10, retries=2)

if status is True:

print("Message sent!")

else:

print("No acknowledgment from recipient")

# Retry sending the message if we don't get an acknowledgment

status = lora.send\_to\_wait(message, 10, retries=2)

# And remember to call this as your program exits...

lora.close()

|  |  |
| --- | --- |
| set\_mode\_rx() | Set to RX continuous mode |
| set\_mode\_tx() | Set to TX continuous mode |
| set\_mode\_idle() | Disable radio |

**INITIALIZATION**

LoRa(channel, interrupt, this\_address, freq=915, tx\_power=14,

modem\_config=ModemConfig.Bw125Cr45Sf128, acks=False, crypto=None)

lora = LoRa(0, 17, 2, modem\_config=ModemConfig.Bw125Cr45Sf128, tx\_power=14, acks=True)

**channel** SPI channel to use (either 0 or 1, if your LoRa radio is connected to CE0 or CE1, respectively)

**interrupt** GPIO pin (BCM-style numbering) to use for the interrupt

**this\_address** The address number (0-254) your device will use when sending and receiving packets.

**freq** Frequency used by your LoRa radio. Defaults to 915Mhz

**tx\_power** Transmission power level from 5 to 23. Keep this as low as possible. Defaults to 14

**model\_config** Modem configuration. See [RadioHead docs](http://www.airspayce.com/mikem/arduino/RadioHead/classRH__RF95.html" \l "ab9605810c11c025758ea91b2813666e3). Default to Bw125Cr45Sf128.

**receive\_all** Receive messages regardless of the destination address

**acks** If True, send an acknowledgment packet when a message is received and wait for an acknowledgment when transmitting a message. This is equivalent to using RadioHead's RHReliableDatagram

**crypto** An instance of PyCryptodome Cipher.AES (see above example)

**OTHER OPTIONS:**

A LoRa instance also has the following attributes that can be changed:

* **cad\_timeout** Timeout for channel activity detection. Default is 0
* **retry\_timeout** Time to wait for an acknowledgment before attempting a retry. Defaults to 0.2 seconds
* **wait\_packet\_sent\_timeout** Timeout for waiting for a packet to transmit. Default is 0.2 seconds

**METHODS**

**send\_to\_wait(data, header\_to, header\_flags=0)**

#### Callbacks

on\_recv(payload) Callback function that runs when a message is received

 payload has the following attributes: header\_from, header\_to, header\_id, header\_flags, message, rssi, snr

### Encryption

If you're working with devices running RadioHead with RHEncryptedDriver, I recommend using the AES cipher.

pip install pycryptodome

and in your code:

from Crypto.Cipher import AES

crypto = AES.new(b"my-secret-encryption-key", AES.MODE\_EAX)

then pass in crypto when instantiating the LoRa object:

lora = LoRa(0, 17, 2, crypto=crypto)