In applications the LoRa class should be subclassed while overriding one or more of the callback functions that are invoked on successful RX or TX operations, for example.

class MyLoRa(LoRa):

def \_\_init\_\_(self, verbose=False):

super(MyLoRa, self).\_\_init\_\_(verbose)

# setup registers etc.

def on\_rx\_done(self):

payload = self.read\_payload(nocheck=True)

# etc.

In the end the resources should be freed properly

BOARD.teardown()

**More details**

Most functions of SX127x.Lora are setter and getter functions. For example, the setter and getter for the coding rate are demonstrated here

print lora.get\_coding\_rate() # print the current coding rate

lora.set\_coding\_rate(CODING\_RATE.CR4\_6) # set it to CR4\_6

@todo

**Installation**

Make sure SPI is activated on you RaspberryPi: [SPI](https://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/README.md) **pySX127x** requires these two python packages:

* [RPi.GPIO](https://pypi.python.org/pypi/RPi.GPIO%22) for accessing the GPIOs, it should be already installed on a standard Raspian Linux image
* [spidev](https://pypi.python.org/pypi/spidev) for controlling SPI

In order to install spidev download the source code and run setup.py manually:

wget https://pypi.python.org/packages/source/s/spidev/spidev-3.1.tar.gz

tar xfvz spidev-3.1.tar.gz

cd spidev-3.1

sudo python setup.py install

At this point you may want to confirm that the unit tests pass. See the section [Tests](https://pypi.org/project/pyLoRa/#tests) below.

You can now run the scripts. For example dump the registers with lora\_util.py:

rasp$ sudo ./lora\_util.py

SX127x LoRa registers:

mode SLEEP

freq 434.000000 MHz

coding\_rate CR4\_5

bw BW125

spreading\_factor 128 chips/symb

implicit\_hdr\_mode OFF

... and so on ....

**Class Reference**

The interface to the SX127x LoRa modem is implemented in the class SX127x.LoRa.LoRa. The most important modem configuration parameters are:

| **Function** | **Description** |
| --- | --- |
| set\_mode | Change OpMode, use the constants.MODE class |
| set\_freq | Set the frequency |
| set\_bw | Set the bandwidth 7.8kHz ... 500kHz |
| set\_coding\_rate | Set the coding rate 4/5, 4/6, 4/7, 4/8 |
|  |  |
| @todo |  |

Most set\_\* functions have a mirror get\_\* function, but beware that the getter return types do not necessarily match the setter input types.

**Register naming convention**

The register addresses are defined in class SX127x.constants.REG and we use a specific naming convention which is best illustrated by a few examples:

| **Register** | **Modem** | **Semtech doc.** | **pySX127x** |
| --- | --- | --- | --- |
| 0x0E | LoRa | RegFifoTxBaseAddr | REG.LORA.FIFO\_TX\_BASE\_ADDR |
| 0x0E | FSK | RegRssiCOnfig | REG.FSK.RSSI\_CONFIG |
| 0x1D | LoRa | RegModemConfig1 | REG.LORA.MODEM\_CONFIG\_1 |
| etc. |  |  |  |

**Hardware**

Hardware related definition and initialisation are located in SX127x.board\_config.BOARD. If you use a SBC other than the Raspberry Pi you'll have to adapt the BOARD class.

**Script references**

**Continuous receiver rx\_cont.py**

The SX127x is put in RXCONT mode and continuously waits for transmissions. Upon a successful read the payload and the irq flags are printed to screen.

usage: rx\_cont.py [-h] [--ocp OCP] [--sf SF] [--freq FREQ] [--bw BW]

[--cr CODING\_RATE] [--preamble PREAMBLE]

Continous LoRa receiver

optional arguments:

-h, --help show this help message and exit

--ocp OCP, -c OCP Over current protection in mA (45 .. 240 mA)

--sf SF, -s SF Spreading factor (6...12). Default is 7.

--freq FREQ, -f FREQ Frequency

--bw BW, -b BW Bandwidth (one of BW7\_8 BW10\_4 BW15\_6 BW20\_8 BW31\_25

BW41\_7 BW62\_5 BW125 BW250 BW500). Default is BW125.

--cr CODING\_RATE, -r CODING\_RATE

Coding rate (one of CR4\_5 CR4\_6 CR4\_7 CR4\_8). Default

is CR4\_5.

--preamble PREAMBLE, -p PREAMBLE

Preamble length. Default is 8.

**Simple LoRa beacon tx\_beacon.py**

A small payload is transmitted in regular intervals.

usage: tx\_beacon.py [-h] [--ocp OCP] [--sf SF] [--freq FREQ] [--bw BW]

[--cr CODING\_RATE] [--preamble PREAMBLE] [--single]

[--wait WAIT]

A simple LoRa beacon

optional arguments:

-h, --help show this help message and exit

--ocp OCP, -c OCP Over current protection in mA (45 .. 240 mA)

--sf SF, -s SF Spreading factor (6...12). Default is 7.

--freq FREQ, -f FREQ Frequency

--bw BW, -b BW Bandwidth (one of BW7\_8 BW10\_4 BW15\_6 BW20\_8 BW31\_25

BW41\_7 BW62\_5 BW125 BW250 BW500). Default is BW125.

--cr CODING\_RATE, -r CODING\_RATE

Coding rate (one of CR4\_5 CR4\_6 CR4\_7 CR4\_8). Default

is CR4\_5.

--preamble PREAMBLE, -p PREAMBLE

Preamble length. Default is 8.

--single, -S Single transmission

--wait WAIT, -w WAIT Waiting time between transmissions (default is 0s)

**Tests**

Execute test\_lora.py to run a few unit tests.

**Contributors**

Please feel free to comment, report issues, or contribute!

Contact me via my company website [Mayer Analytics](http://mayeranalytics.com/) and my private blog [mcmayer.net](http://mcmayer.net/).

Follow me on twitter [@markuscmayer](https://twitter.com/markuscmayer) and [@mayeranalytics](https://twitter.com/mayeranalytics).

**Roadmap**

95% of functions for the Sx127x LoRa capabilities are implemented. Functions will be added when necessary. The test coverage is rather low but we intend to change that soon.

**Semtech SX1272/3 vs. SX1276/7/8/9**

**pySX127x** is not entirely compatible with the 1272. The 1276 and 1272 chips are different and the interfaces not 100% identical. For example registers 0x26/27. But the pySX127x library should get you pretty far if you use it with care. Here are the two datasheets:

* [Semtech - SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver](http://www.semtech.com/images/datasheet/sx1276_77_78_79.pdf)
* [Semtech SX1272/73 - 860 MHz to 1020 MHz Low Power Long Range Transceiver](http://www.semtech.com/images/datasheet/sx1272.pdf)

**HopeRF transceiver ICs**

HopeRF has a family of LoRa capable transceiver chips [RFM92/95/96/98](http://www.hoperf.com/) that have identical or almost identical SPI interface as the Semtech SX1276/7/8/9 family.

**Microchip transceiver IC**

Likewise Microchip has the chip [RN2483](http://ww1.microchip.com/downloads/en/DeviceDoc/50002346A.pdf)

The [pySX127x](https://github.com/mayeranalytics/pySX127x) project will therefore be renamed to pyLoRa at some point.

**LoRaWAN**

LoRaWAN is a LPWAN (low power WAN) and, and **pySX127x** has almost no relationship with LoRaWAN. Here we only deal with the interface into the chip(s) that enable the physical layer of LoRaWAN networks.