# loRaSwarmImplementation.py Reference

The LoRaSwarmImplementation class provides an implementation for use on a Pycom Pygate[1] in conjunction with the modified Pygate firmware, 'Pygate for P2P'[2]. This reference is a guide to the initialisation and usage of this implementation of LoRaSwarm.

## Initialisation

In order to initialise the LoRaSwarm protocol the WLAN Pycom MicroPython module of must be initialised first. The Pygate module which is provided as part of the 'Pygate for P2P'[2] repository must also be initialised. An example initialisation of the LoRaSwarmImplementation is provided below.

```
wlan = WLAN(mode=WLAN.AP, ssid='dummy-network')
pygate = Pygate()
rtc = RTC()

# Initialise LoRaSwarm for a node with ID 103
loraSwarm = LoRaSwarmImplementation(103, pygate, chrono)
```

An argument `memoryStats=True` may be passed into the LoRaSwarmImplementation constructor which will periodically print the available memory every 10 seconds.

For the LoRaSwarmImplementation to begin running, a secondary thread must be started which calls the 'run' function. The LoRaSwarmImplementation will then begin receiving/forwarding packets and transmit Heartbeat packets every 10 minutes.

An example of this is provided in Table 1.

*Table 1 | Example of starting loraSwarm implementation on a second thread.*

```
import _thread

wlan = WLAN(mode=WLAN.AP, ssid='dummy-network')
pygate = Pygate()
rtc = RTC()
chrono = machine.Timer.chrono()

# Initialise LoRaSwarm for a node with ID 103
loraSwarm = LoRaSwarmImplementation(103, pygate, chrono)

def run_thread():
      loraSwarm.run()
# Start loraSwarm run thread
_thread.start_new_thread(run_thread,())
```

[1] Pycom Pygate, https://pycom.io/product/pygate/
[2] Modified Pygate Firmware, https://github.com/RupertHSmith/Pygate-for-P2P

## Usage

### send

The 'send' function is used to transmit an Application Packet from the node. By default, Extended Application Packets are sent which also contain a Neighbour List.

```
send(payload, forwardHorizon=0, neighbourListEnabled=True, sf=0)
```

If bandwidth is a concern for the transmission, then the parameter `neighbourListEnabled=False` can be given to send the packet as a basic Application Packet.

By default, the packet will not be forwarded however packet forwarding can be enabled by specifying the forward horizon (e.g. `forwardHorizon=1`).

Normally the SF will be automatically chosen using the Dynamic Radio Parameter Selection algorithm however a user specified SF can instead be provided by including the argument `sf=`$x$ where $7 \leq x \leq 12$.

A 1% Duty Cycle is implemented and a send queue is used to regulate transmissions. Note if a transmission is not immediately emitted, it may be due to the recent transmission of a Heartbeat packet.

### sendControl

The 'sendControl' function is used to transmit a Control Packet.

```
sendControl(payload, sf=0)
```

By default, the Dynamic Radio Parameter Selection algorithm is enabled for these transmissions however may be disabled by providing the argument `sf=`$x$ where $7 \leq x \leq 12$.

### receive

The 'receive' function is non-blocking and will fetch the head of the received packets queue. If there are no packets to be received then `None` will be returned. The function returns a tuple in the format `(header, appPacket)`. 'header' is an object with fields described in Table 2. This may be useful for debugging or logging purposes. 'appPacket' is the Application Packet named tuple which is described in the LoRaSwarmProtocol document.

*Table 2 | Fields of Pygate Header JSON object.*

| ***Name*** | ***Description*** |
|---|---|
| *rssi* | RSSI of packet in dBm |
| *snr* | SNR of packet in dB |
| *chan* | Channel received on (0-7) |
| *rfch* | Radio module received on (0 or 1) |
| *freq* | Frequency received on (MHz) |
| *datr* | SF and Bandwidth combined as SF<SF>BW<Bandwidth in MHz> |
| *codr* | Code Rate (default: '4/5') |

### receiveControl

The ‘receiveControl’ function is non-blocking and will fetch the head of the received Control packets queue. If there are no Control packets to be received then `None` will be returned. The function returns a tuple in the format `(header, controlPacket).` ‘header’ is an object with fields described in Table 2. This may be useful for debugging or logging purposes. ‘controlPacket is the Control Packet named tuple which is described in the LoRaSwarmProtocol document.

### enableIdleMode/disableIdleMode

The ‘enableIdleMode’ and ‘disableIdleMode’ functions are used to enable/disable idle mode. This is a mode used to transmit Heartbeat packets at the maximum allowable rate. Useful if the node will be inactive for a long period of time.