Hola Intercom Software Guide

# Revision History

* 09/01/2017: 1st draft
* 09/18/2017: Added buddy\_id to cloud api
* 10/02/2017: Added echo request and reply messages
* 10/16/2017: Added 3rd party components and updated messages
* 10/24/2017: Added manuf\_config.py. Updated message handler, messages and cloud\_api
* 10/25/2017: Added pin assignments. Added MAX17043 Particle Library
* 10/30/2017: Added restarted parameter to i\_am message
* 10/31/2017: Added battery\_lvl to cloud api. Cloud api now ticks.
* 11/21/2017: Added S1509 I/O Expander.
* 11/27/2017: Added multiple buddy support.
* 11/28/2017: Documented and applied coding convention
* 11/28/2017: Described object roles.
* 11/28/2017: Pin Assignment fixes
* 11/28/2017: Plf\_TickerBase class introduced
* 11/29/2017: Added Trace&Debug section
* 12/7/2017: Added volume control
* 12/15/2017: Added battery checker
* 1/4/2018: Added ratetune print group. Intercom\_Incoming is now a Ticker class.
* 1/9/2018: Added stuff() method to circularBuffer.
* 1/9/2018: Added COMM\_START/STOP+ACK messages
* 1/9/2018: Added recordRequest to Intercom\_Outgoing. Renamed transfer() method to run().
* 1/9/2018: Added codec patch version
* 1/11/2018: Added FSM diagrams
* 1/15/2018: Replaced BatteryChecker section with LevelChecker section. Added WifiChecker.
* 1/15/2018: Added Table of Contents
* 1/17/2018: Added getBatteryPct API
* 1/19/2018: Modified battery\_pct and wifi\_pct cloud API
* 1/21/2018: Added Intercom\_Buddy LED FSM
* 1/26/2018: Added DataDump module and expanded Registry section
* 1/26/2018: Expanded shell commands section
* 1/28/2018: Added comm state FSM to intercom\_buddy section
* 1/30/2018: Reversed relation between buddy and outgoing
* 1/30/2018: FSM updated for buddy and outgoing

Table of Contents

[1 Revision History 1](#_Toc504744293)

[3 Version Control 3](#_Toc504744294)

[4 Third Party Components 3](#_Toc504744295)

[5 Shell Commands 4](#_Toc504744296)

[6 Pin Assignments 4](#_Toc504744297)

[7 Firmware Coding Convention 5](#_Toc504744298)

[8 Firmware Design 7](#_Toc504744299)

[8.1 Ownership 7](#_Toc504744300)

[8.2 Intercom\_Root 9](#_Toc504744301)

[8.3 Intercom\_Incoming 11](#_Toc504744302)

[8.4 Intercom Outgoing 12](#_Toc504744303)

[8.5 Intercom\_Controller 13](#_Toc504744304)

[8.6 Intercom\_Buddy 14](#_Toc504744305)

[8.7 Volume Control 17](#_Toc504744306)

[8.8 Level Checkers 18](#_Toc504744307)

[8.9 Codec 20](#_Toc504744308)

[8.10 Intercom Message Handler 21](#_Toc504744309)

[8.11 Messages 22](#_Toc504744310)

[8.12 Intercom\_CloudAPI 24](#_Toc504744311)

[8.13 Intercom\_ButtonsAndLeds 25](#_Toc504744312)

[8.14 Trace & Debug 25](#_Toc504744313)

[8.15 Registry 25](#_Toc504744314)

[8.16 Data Dump 26](#_Toc504744315)

[9 Server Side 26](#_Toc504744316)

[10 Manuf\_config.py 28](#_Toc504744317)

# Version Control

<https://github.com/rlysens/particleIntercom/>

# Third Party Components

Draw.io:

<https://www.draw.io/>

VS1063 codec patch 2.01:

<http://www.vlsi.fi/en/support/software/vs10xxpatches.html>

SparkFun\_MAX17043\_Particle\_Library: <https://github.com/sparkfun/SparkFun_MAX17043_Particle_Library.git>

SparkFunSX1509 Library:

<https://github.com/sparkfun/SparkFun_SX1509_Arduino_Library>

LCM: <https://lcm-proj.github.io/> (in source tree)

Name-gen (in source tree)

XTEA: mbedtls-2.5.1

XTEA Python: <https://pypi.python.org/pypi/xtea/0.4.0> (in source tree)

particle.py (in source tree)

* Pytz.py
* Dateutil
* Requests
* pexpect

# Shell Commands

Enable/Disable certain print groups:

particle call Intercom1 en|dis\_prntgrp messages/ratetune/default

List available data dump modules:

particle call Intercom1 list\_ddump

Request a data dump for given module:

particle call Intercom ddump <modulename>

Serial port monitor:

particle serial monitor COM3/COM4

Build and Flash:

python build.py --device Intercom1/Intercom2/all [--flash]

Copy server-side code to Amazon EC2 server:

copypython2amazon.bat

Copy JSON files to server-side

copyjson2amazon.bat

Login to Amazon EC2 server:

ssh2amazonec3.bat

Run Manufacturing Configuration script:

python manuf\_config.py [-h,--help] [-c,--skip\_claim] [-f,--skip\_flash] [-i,--image\_filename <image\_filename>]

Finds and claims connected devices, sets up Wifi, flashes reference fw image and configures name, buddy name and secret key.

# Pin Assignments

Photon pin assignments:

A5 = SI to codec

A4 = SO to codec

A3 = SCK to codec

A2 = xDCS to codec

A1 = xRESET to codec

A0 = DREQ to codec

DAC = xCS to codec

D0 = SDA to MAX17043 fuel gauge IC and to sx1509 i/o expander

D1 = SCL to MAX17043 fuel gauge IC and to sx1509 i/o expander

D2 = xRST to sx1509 i/o expander

SX1509 I/O Expander pin assignments:

0 = Battery check button

1 = Buddy 0 button

2 = Buddy 1 button

3 = Buddy 2 button

4 = Buddy 0 LED

5 = Buddy 1 LED

6 = Buddy 2 LED

7 = Wifi check button

8 = Vol.Dec.

9 = Vol.Inc.

10 = LED Bar 0

11 = LED Bar 1

12 = LED Bar 2

13 = LED Bar 3

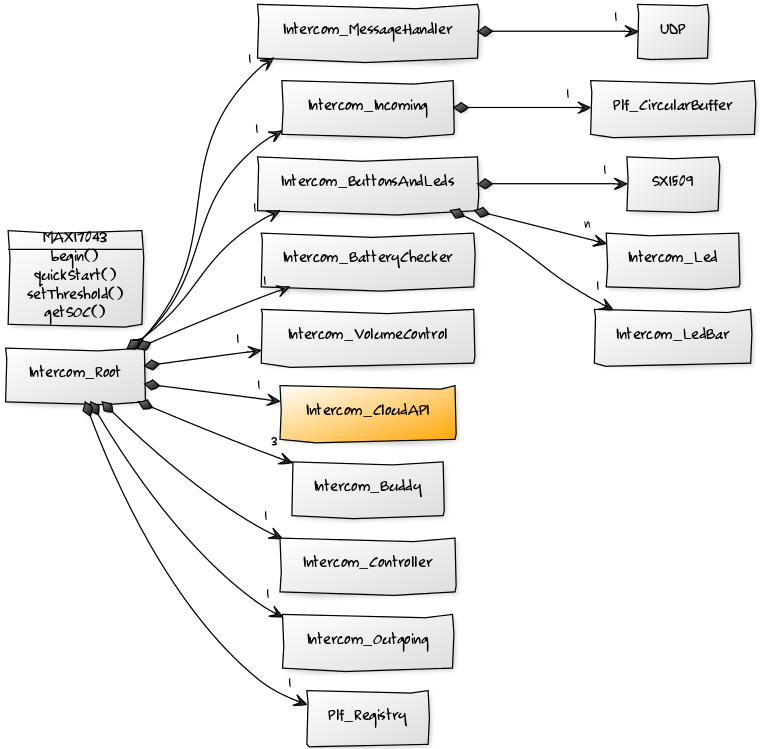
14 = LED Bar 4

# Firmware Coding Convention

* CamelCase
* Types start with capital
* Functions and variables start with lowercase, e.g. topIndex
* Typedefs end with \_t, e.g. RegisterEntry\_t
* Pointers end with p, e.g. registry
* Private members have leading underscore, e.g. \_registryp
* #defines are all caps and underscores, e.g. MAX\_KEY\_VAL
* ‘Namespace’-like prefixes are separated with an underscore, e.g. Intercom\_Outgoing, intercom\_outgoing
* Filenames are lowercase with underscores
* \_s may be used as string postfix
* When interacting with 3rd party code, the 3rd party coding convention may be used.

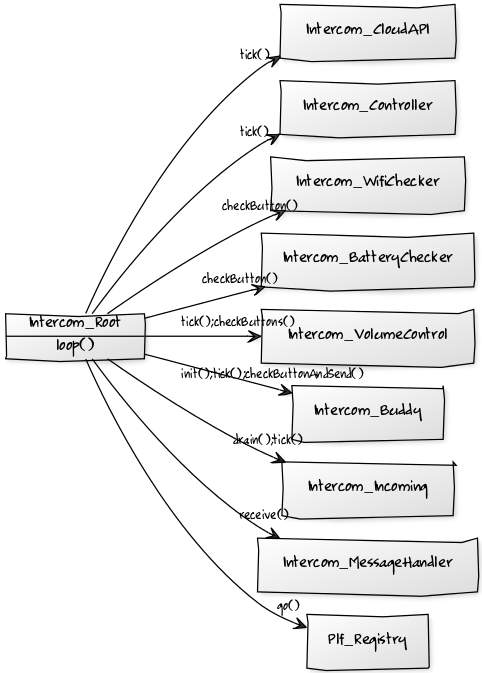
# Firmware Design

## Ownership



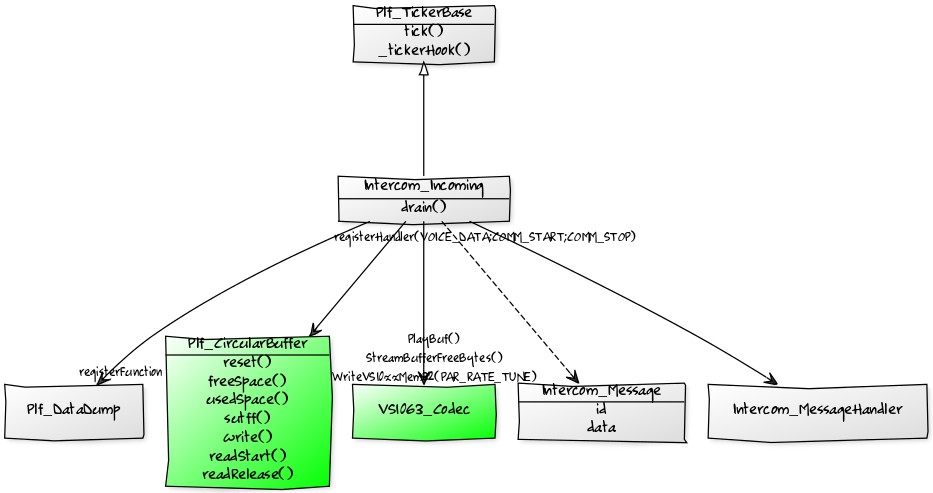
## Intercom\_Root

Intercom\_Root is the root of the Intercom object tree.



## Intercom\_Incoming

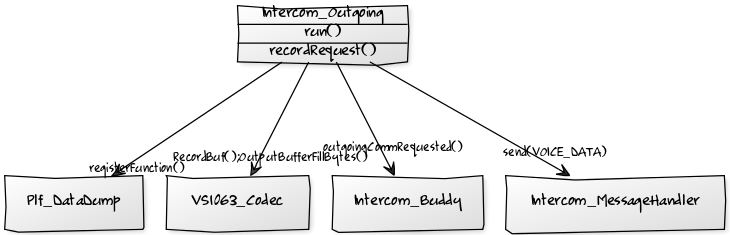
Intercom\_Incoming is in charge of the incoming datapath logic.





## Intercom Outgoing

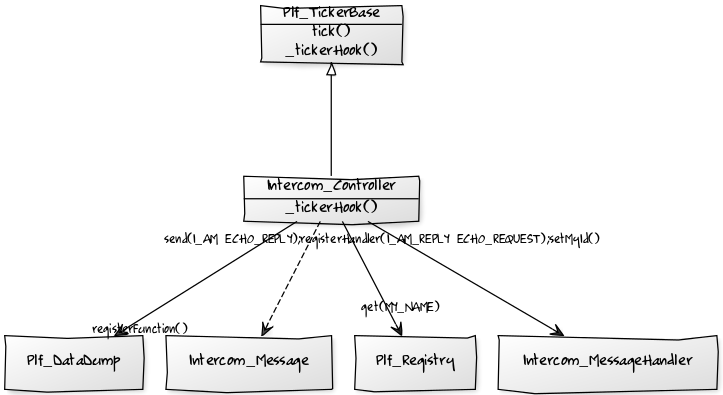
Intercom\_Outgoing is in charge of the outgoing datapath logic.





## Intercom\_Controller

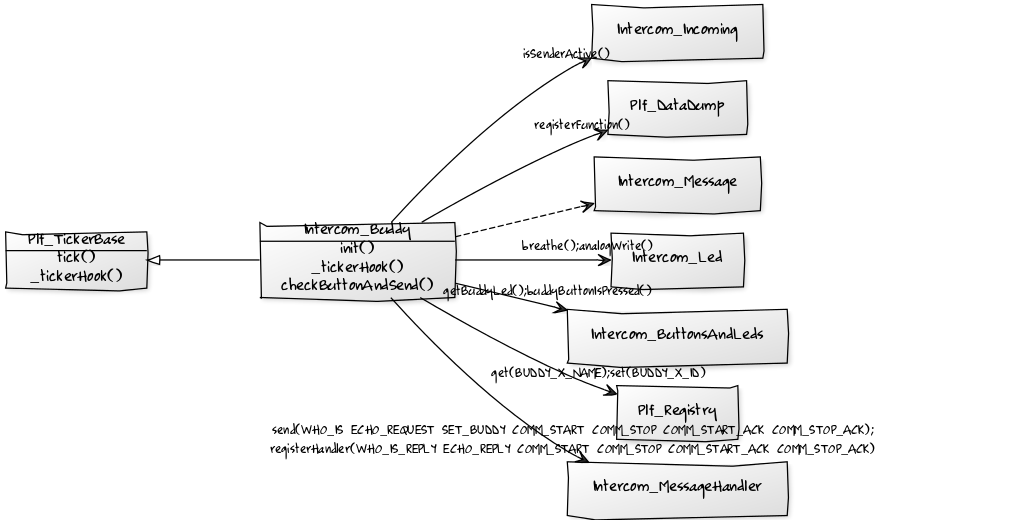
Intercom\_Controller handles the control plane logic that is not Buddy specific.





## Intercom\_Buddy

Intercom\_Buddy handles all buddy related control plane logic and user I/O.







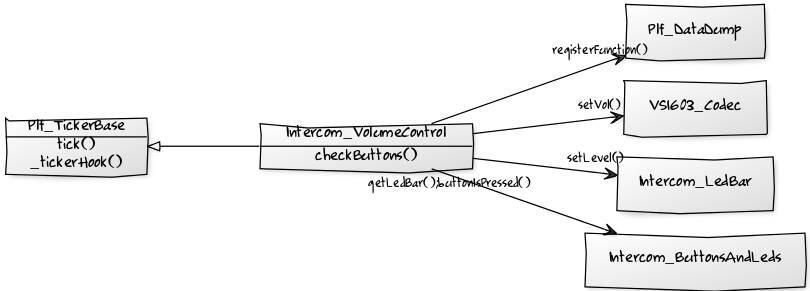




****

## Volume Control

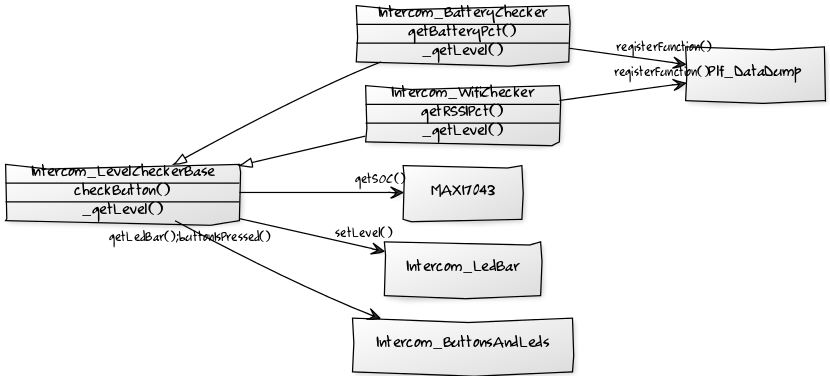
Manages volume control buttons and led bar.





## Level Checkers

Base class Intercom\_LevelCheckerBase manages check button and led bar. Intercom\_WifiChecker and Intercom\_BatteryChecker derive from this base class and implement respectively Wifi Level checking and Battery Level checking.





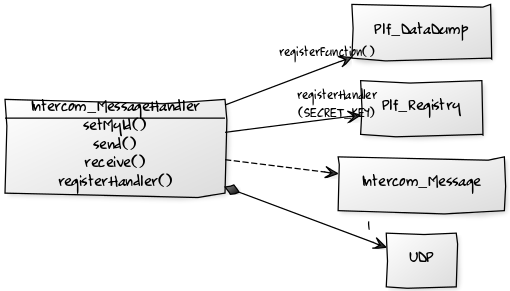
## Codec

Voice codec logic with back-end SPI interaction factored out into a separate module.



## Intercom Message Handler

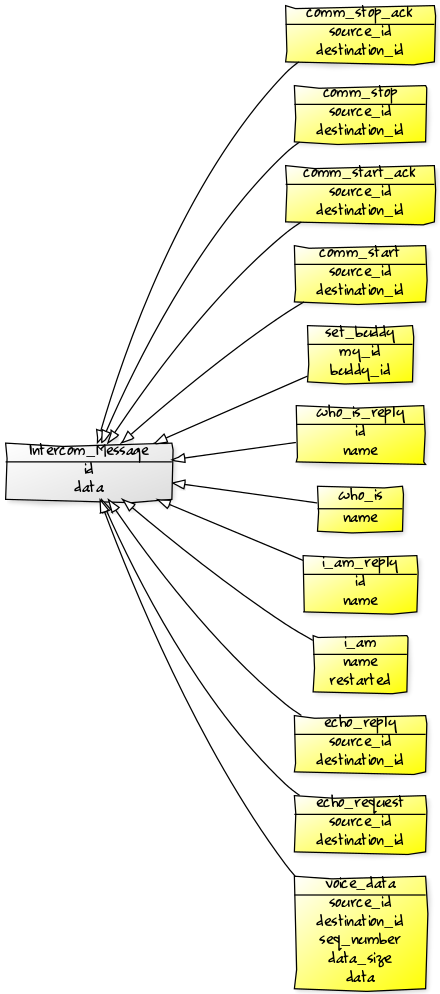
Sends, receives and dispatches Intercom messages.



## Messages

Note that the message channel is not considered reliable. Messages can get lost and the message protocols must be robust against occasional message loss. All messages except i\_am are encrypted using XTEA block cipher. The secret key is configured into the device at manufacturing time.

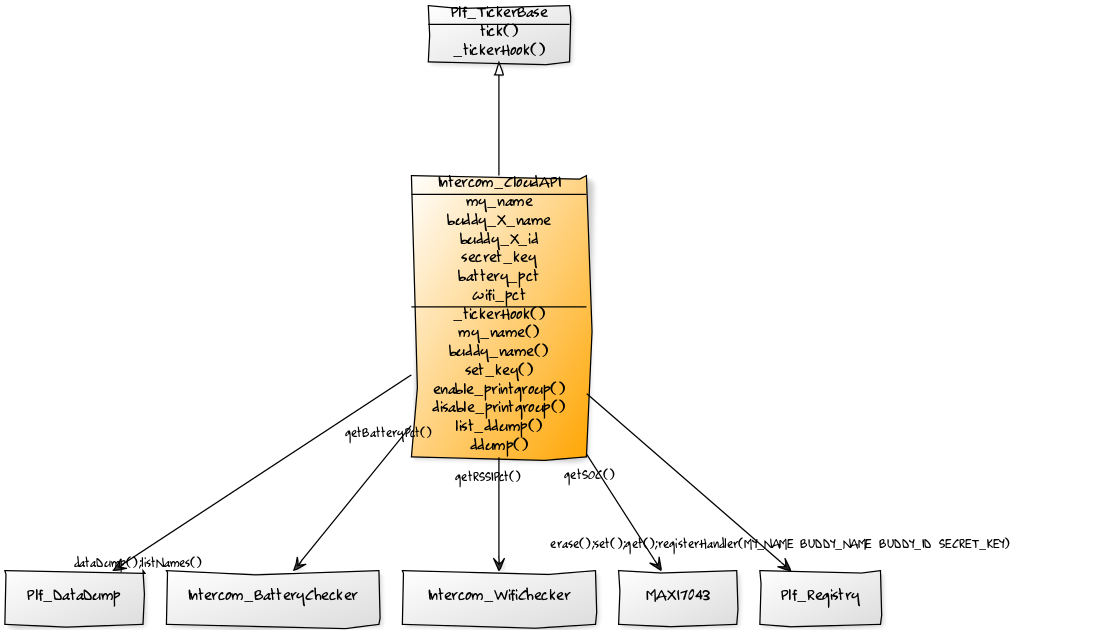
Encryption modes CBC and ECB are supported (selected at compile time).



## Intercom\_CloudAPI

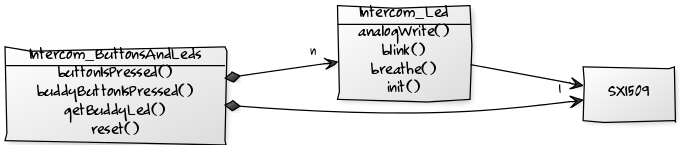
Intercom\_CloudAPI represents the API toward the cloud/Hola App.

X=0, 1 or 2



## Intercom\_ButtonsAndLeds

Intercom\_ButtonsAndLeds encapsulates interaction with buttons and leds. Uses the SX1509 I/O extender library.



## Trace & Debug

* Tracing: plf\_utils.h has a Trace print API (PLF\_PRINT) with printgroups that can be enabled/disabled via the cloudAPI.
* Asserts: plf\_assert in plf\_utils.h
* Error return codes: Each module has a unique MODULE\_ID to be used as a base for error return codes (e.g. return –(MODULE\_ID+1)).
* Event Counting: See plf\_even\_counter.h. Counters can be displayed by datadumping the stats module.
* Data Dumping: See Data Dump section below.

## Registry

The Registry keeps track of a number of Registry elements (strings) identified by a registry key. Registry elements can be set, retrieved and removed (made invalid). Certain registry elements are persistent (i.e stored in non-volatile memory), others are volatile. See list below.

Keys:

REG\_KEY\_MY\_NAME persistent

REG\_KEY\_BUDDY\_X\_NAME persistent

REG\_KEY\_BUDDY\_X\_ID volatile

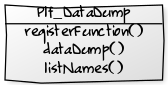
REG\_KEY\_SECRET\_KEY persistent

X=0, 1 or 2



## Data Dump

Other objects can register a data dump function with the Plf\_DataDump object. A data dump function implementation prints out the object’s state. Plf\_DataDump hooks into the CloudAPI so that the user can request a data dump of any registered object.



# Server Side

Message\_Handler is the central object. Intercom\_name\_to\_id table and interom\_id\_to\_intercom table are used for Intercom name to id mapping and looking up corresponding Intercom objects.

**MSG\_TABLE**

msg\_id -> msg\_handler\_fun

**Intercom\_name\_to\_id\_table**

Intercom\_name -> intercom\_id

**Intercom\_id\_to\_intercom\_table**

Intercom\_id -> intercom

**Message\_Handler**

**Send()**

**Receive()**

socket

**Intercom**

**sendTo()**

**setBuddy()**

**getEncoderCryptoCodec()**

**getDecoderCryptoCodec**

Msg\_voice\_data\_handler

Msg\_i\_am\_  
handler

Msg\_who\_is\_  
handler

Set\_buddy\_  
handler

1..1

n..1

1..1

Msg\_echo\_request\_data\_handler

Msg\_echo\_reply\_data\_handler

# Manuf\_config.py

**Credentials.json**

**Name\_key.json**

**Manuf\_config.py**

Particle.py

Name\_key\_gen.py

Namegen.py