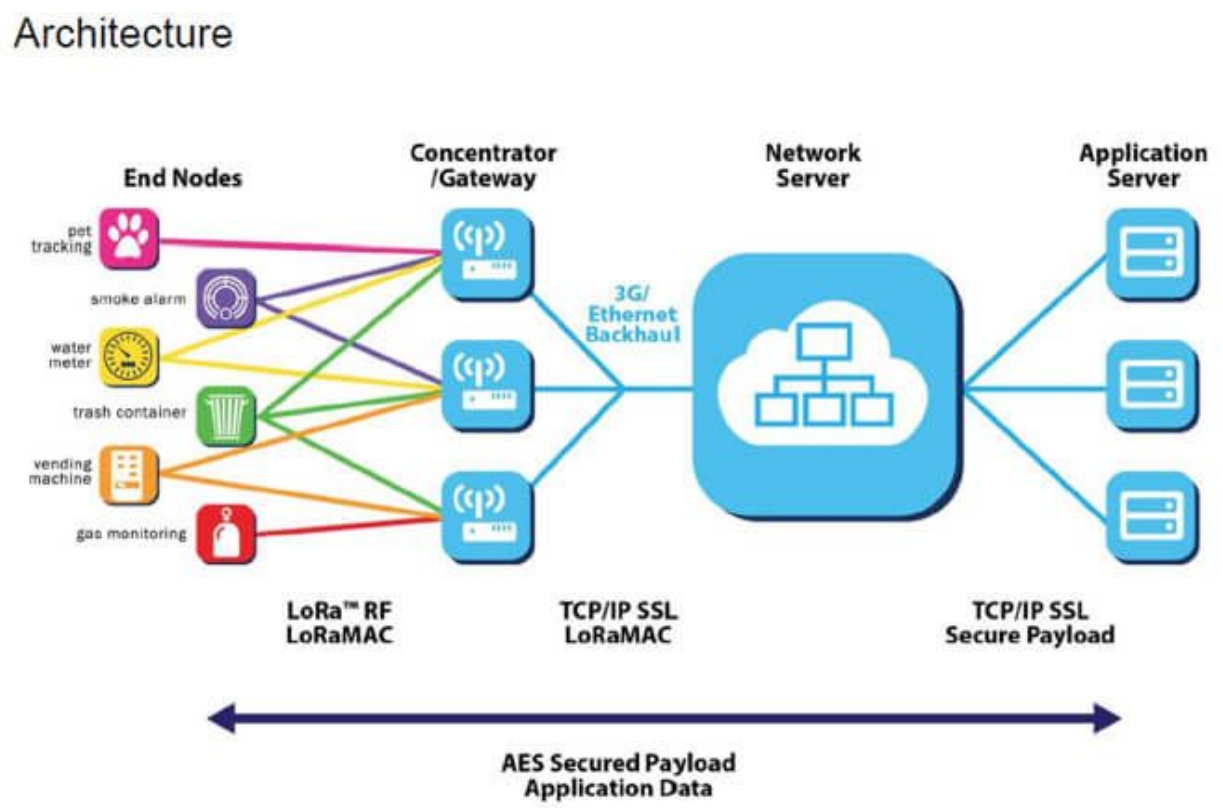
LoRaWAN Notes

LMIC Library for LoRaWAN

There are multiple forks supporting a variety of country band plans.



Forks and Subprojects

IBM LoraMAC in C Library (lmic)

Copyright (C) 2014-2016 IBM Corporation

MCCI Catena

https://github.com/mcci-catena/arduino-lmic

Copyright (c) 2016-2019 MCCI Corporation

Copyright (c) 2015 Thomas Telkamp and Matthijs Kooijman

Terry Moore, MCCI November 2018

https://github.com/matthijskooijman/arduino-lmic

https://github.com/things-nyc/arduino-lmic

Thomas Laurenson (modifications for AU915)

Blogpost: https://www.thethingsnetwork.org/forum/t/limitations-data-rate-packet-size-30-sec-up-and-10-messages-down-p-d-fair-access/1300

Github: https://github.com/thomaslaurenson/arduino-lmic

Version - 1.6 Build 1468577746

Arduino LMIC Version: 2.3.2

People and Organisations

- IBM
- MCCI
- Thomas Telkamp
- Matthijs Kooijman
- Thomas Laurenson
    - Email: thomas@thomaslaurenson.com
    - GitHub: https://github.com/thomaslaurenson
    - Blog: https://www.thomaslaurenson.com/blog/

LMIC Library Notes

```
project_config/lmic_project_config.h
-     Set bandplan here.
lmic/lmic.h
-     oslmic.h
-     lorabase.h
lmic/lmic_bandplan.h
lmic/llmic_util.h
lmic/lmic_bandplan_us915.h
lmic/lmic_bandplan_au921.h
```

Adding a Region to LMIC

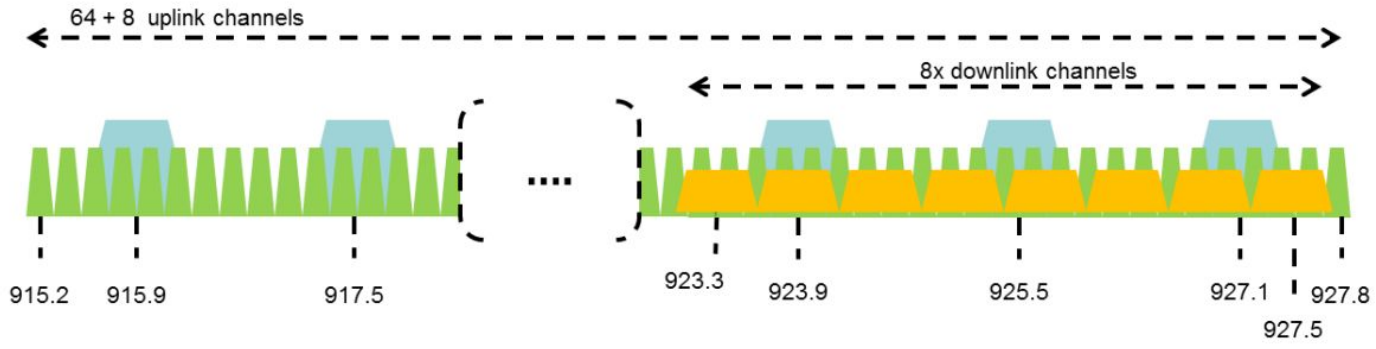
https://github.com/openwave-co-jp/arduino-lmic-master-for-LG01-JP/blob/master/HOWTO-ADD-REGION.md

https://lora-alliance.org/sites/default/files/2018-05/lorawan_regional_parameters_v1.0.2_final_1944_1.pdf

Frequency Allocation for Australia/New Zealand

762 2.5.2 AU915-928 Channel Frequencies

(From Lora Regional parameters document)
763 The AU ISM Band shall be divided into the following channel plans.
764 Upstream – 64 channels numbered 0 to 63 utilizing LoRa 125 kHz BW varying from
765 DR0 to DR5, using coding rate 4/5, starting at 915.2 MHz and incrementing linearly
766 by 200 kHz to 927.8 MHz
767 Upstream – 8 channels numbered 64 to 71 utilizing LoRa 500 kHz BW at DR6
768 starting at 915.9 MHz and incrementing linearly by 1.6 MHz to 927.1 MHz
769 Downstream – 8 channels numbered 0 to 7 utilizing LoRa 500 kHz BW at DR8 to
770 DR13) starting at 923.3 MHz and incrementing linearly by 600 kHz to 927.5 MHz
771
772
773 Figure 2: AU915-928 channel frequencies
774 AU ISM band end-devices may use a maximum EIRP of +30 dBm.
775 AU915-928 end-devices should be capable of operating in the 915 to 928 MHz frequency
776 band and should feature a channel data structure to store the parameters of 72 channels. A
777 channel data structure corresponds to a frequency and a set of data rates usable on this
778 frequency.
779 If using the over-the-air activation procedure, the end-device should broadcast the JoinReq
780 message alternatively on a random 125 kHz channel amongst the 64 channels defined using
781 DR0 and a random 500 kHz channel amongst the 8 channels defined using DR6. The end
782 device should change channel for every transmission.
783 Personalized devices shall have all 72 channels enabled following a reset.



Note: LorasWAN network used by The Things Network in Australia/NewZealand uses the second 8 narrow uplink channels, the next wide uplink channel, and the downloin channel.

Dragino/Arduino Sketches - Examples

Add library

```
cd ~/Documents/git
git clone https://github.com/thomaslaurenson/arduino-lmic.git
cd ~/Arduino/libraries
ln -s ~/Documents/git/arduino-lmic
```

Sensor / Node / End device

Thomas Laurenson's modified library

Blog:
https://www.thomaslaurenson.com/blog/2018/07/21/dragino-lorashield-on-AU915-using-arduino-lmic-library/

Github: https://github.com/thomaslaurenson/arduino-lmic

Included examples:

```
header_test
raw
raw-feather
raw-halconfig
ttn-abp
ttn-abp-adafruit-feather-au915
ttn-abp-dragino-lorashield-au915
ttn-abp-feather-us915-dht22
ttn-otaa
ttn-otaa-adafruit-feather-au915
ttn-otaa-dragino-lorashield-au915
ttn-otaa-feather-us915
ttn-otaa-feather-us915-dht22
ttn-otaa-halconfig-us915
ttn-otaa-network-time
```

Dragino Gateway

Arduino sketch from Dragino - Single_pkt_fwd_v004

Github:
https://github.com/dragino/Arduino-Profile-Examples/tree/master/libraries/Dragino/examples/LoRa/LoRaWAN/single_pkt_fwd_v004

Uses and Applications

Applications

- Shopping trolly tracker
- Parcel tracker
- Child tracker

Demonstrations

- Parcel tracker across Australia and New Zealand
    - Posting a tracker

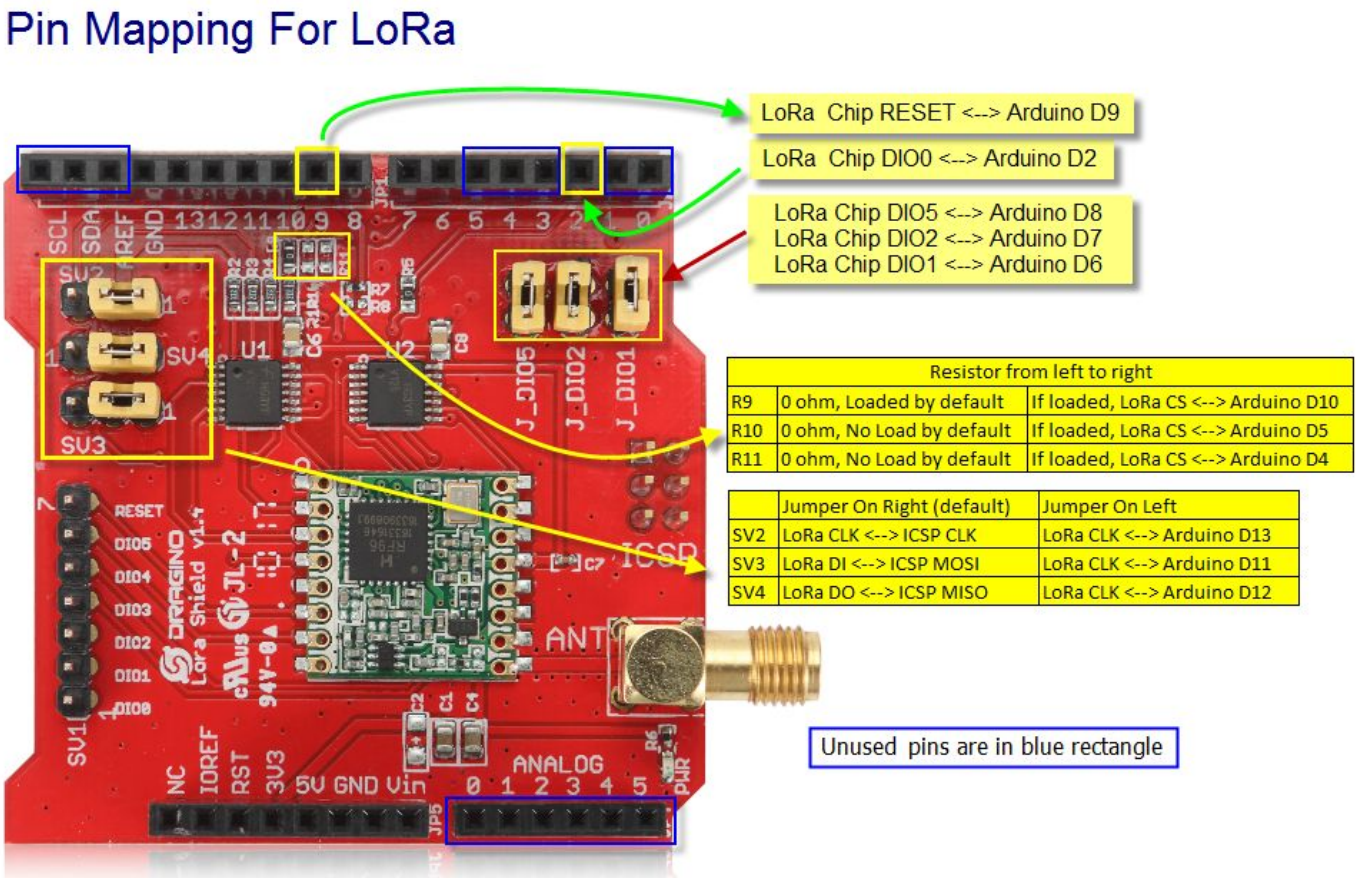
- Balloon telemetry / bidirectional traffic
    - Telemetry and Control
    - Payload support
- Cosmic Ray - IoT Device (See document)
- Locator services

Push button feedback and Remote Lighting Control

Hardware

Dragino Arduino Shield

Also: duinotech LoRa Shield (available from Jaycar)



Available Arduino pins

- SCL
- SDA
- AREF
- GND
- D3-5,D0-1 (5 digital lines)
- A0-5

Can really only use D3-5 (3). D0-1 are uses by serial io, and D11-13 have some other signals on them.

SparkFun Pro RF - LoRa, 915MHz (SAMD21, Black Board)

https://core-electronics.com.au/sparkfun-pro-rf-lora-915mhz-samd21-black-board.html

I-NUCLEO-LRWAN1 - Expansion Board, Arduino Compatible

https://au.element14.com/stmicroelectronics/i-nucleo-lrwan1/expansion-board-arm-cortex-m0/dp/2809319?st=LoRaWAN

Adafruit LoRa Radio Bonnet with OLED

https://core-electronics.com.au/adafruit-lora-radio-bonnet-with-oled-rfm95w-915mhz-radiofruit.html

Bonnet for RPi

RAK831 LoRa Gateway Kit (SX1301)

https://core-electronics.com.au/rak831-lora-gateway-kit-sx1301-au915-as923-compatible.html

IMST iC880A LoRaWAN "concentrator" board

https://www.rs-online.com/designspark/building-a-raspberry-pi-powered-lorawan-gateway

The front-end is provided by an IMST iC880A LoRaWAN "concentrator" board. This integrates two Semtech SX1257 transceiver ICs plus an SX1301 baseband processor. A combination that is able to emulate 49x LoRa demodulators, with 10 parallel demodulation paths, in order to receive up to 8 LoRa packets simultaneously sent with different spreading factors on different channels.

Software: https://github.com/Lora-net

Gateway Hardware

LoRaWAN uGateway

https://core-electronics.com.au/tutorials/ugateway-wifi-setup.html

https://core-electronics.com.au/tutorials/ugateway-other-tips.html

Scripts network on desktop

```
Set LoRaWAN to 915 with script, requires internet access.
```

- Gett Gateway UID from top of script output
- Gateway will forward packets to TTN, but not support OTAA (not sure about this)

Projects

Custom/Miniaturized hardware

Multiple region support (multi frequency)

Other Networks

OpenHAB

Website - https://www.openhab.org

Uses MQTT for communication - https://www.openhab.org/v2.4/addons/bindings/mqtt/

The Things Network

Cheat Sheet

When setting up a ‘Things’ application in the The Things Network, the following parameters are required.

Details / Field	Setting / Value	Description / Comment
Create an Application		An application definition is required to allow data collected by The Things Network to be routed to its destination.
Application ID		Short name. Is used as username for MQTT login.
Application Description		A short description of what the application is used for. Useful as a quick reminder to why the application was created.
Handler	meshed-handler	This is the hander/server being used by the Australian Things Network. MTQQ access is then via thethings.meshed.com.au
Register Device		Devices are your end nodes which connect to the radio network.
Device ID		Application unique identity string for end device. It is used in MQTT channel name.
Authentication Type	OTAA (or APB)	What method does the device use to authenticate itself and it’s data to the network. The two options are ‘Over the Air Authentication’ (OTAA) or ‘Authentication by Personalisation’ (ABP)
Over the Air Authorisation (preferred) - OTAA		With OTAA, session keys are negotiated each time device connects to .the network. Requires bidirectional network connectivity.
Device EUI (DEVEUI)		The device EUI is the unique identifier for this device on the network. You can change the EUI later.
Application EUI (APPEUI)		Identifier for the application.
App Key (APPKEY)		Unique Data
Authentication By Personalisation - ABP		Uses fixed session keys. Does not require bidirectional network communication for authentication, but susceptible to replay attack if packet counters are not used. Packet counters at data endpoints allow duplicate packets to be ignored.
Device Address (DEVADDR)		Unique for every device in the application.
Network Session Key (NWKSKEY)	(Automatically generated)	Key is used to encrypt connection to server.
Application Session Key (APPSKEY)	(Automatically generated)	Key is used to authenticate data packets with the application.

Access Key (Application access key)	(Automatically generated)	Used as MQTT password/passphrase. Starts with `ttn-account-v2. Additional keys can be created`

Application Data

MQTT

Useful sites:
https://github.com/clojure-cookbook/clojure-cookbook/blob/master/05_network-io/5-07_communicating-with-mqtt.asciidoc

See the API reference: https://www.thethingsnetwork.org/docs/applications/mqtt/api.html

Server: thethings.meshed.com.au

Username: <application-id>

Password: <application-access-key>

Channels (examples)

- <application-id>/device/<device-id>/up
- <application-id>/device/<device-id>/up/<field>
- <application-id>/device/<device-id>/down
- <application-id>/device/<device-id>/events
- <application-id>/device/<device-id>/events/create
- <application-id>/device/<device-id>/events/update
- <application-id>/device/<device-id>/events/delete
- <application-id>/device/<device-id>/events/activations
- <application-id>/device/<device-id>/events/down/scheduled
- <application-id>/device/<device-id>/events/down/sent

Mosquitto (Ubuntu)

```
sudo apt update
sudo apt upgrade
sudo apt-install mosquitto-clients
man mosquitto_sub
```

```
mosquitto_sub {[-h hostname] [-p port number] [ [-u username] [-P password] ] -t message-topic...  |
               -L URL [-t message-topic...] } [-A bind_address] [-c] [-C msg count] [-d] [-i client_id]
               [-I client id prefix] [-k keepalive time] [-N] [-q message QoS] [-R | --retained-only] [-S]
               [-T filter-out...] [-U unsub-topic...] [-v] [-V protocol-version] [-W message processing timeout]
               [--proxy socks-url] [--quiet]
               [--will-topic topic [--will-payload payload] [--will-qos qos] [--will-retain]]
               [[{--cafile file | --capath dir} [--cert file] [--key file] [--tls-version version] [--insecure]]
               | [--psk hex-key --psk-identity identity [--tls-version version]]]
```

```
mosquitto_sub -h thethings.meshed.com.au -u enfieldlibrary_iot_trial \
  -P ttn-account-v2.-CAj7tWllLIQn1pCOZitrXI5ClM5j06uweHFubaqdPk \
  -t enfieldlibrary_iot_trial/devices/+/up
```

Save in a file

```
mosquitto_sub -h thethings.meshed.com.au \
  -u enfieldlibrary_iot_trial \
  -P ttn-account-v2.-CAj7tWllLIQn1pCOZitrXI5ClM5j06uweHFubaqdPk \
  -t enfieldlibrary_iot_trial/devices/+/up | tee ttndata.txt
```

JSON message can also be filtered by using ‘jq’ tool from the command line:

```
msub | jq
```

Downlink message

Topic: <AppID>/devices/<DevID>/down

Message format

```
{
  "port": 1,                 // LoRaWAN FPort
  "confirmed": false,        // Whether the downlink should be confirmed by the
device
  "payload_raw": "AQIDBA==", // Base64 encoded payload: [0x01, 0x02, 0x03, 0x04]
}
```

Usage (Mosquitto):

```
mosquitto_pub -h <Region>.thethings.network -d -t
'my-app-id/devices/my-dev-id/down' -m '{"port":1,"payload_raw":"AQIDBA=="}'
```

```
mosquitto_pub -h thethings.meshed.com.au \
 -u enfieldlibrary_iot_trial \
 -P ttn-account-v2.-CAj7tWllLIQn1pCOZitrXI5ClM5j06uweHFubaqdPk \
 -t 'enfieldlibrary_iot_trial/devices/roaming_mawsonlakes/down' \
 -m '{"port":1,"payload_raw":"AQIDBA=="}'
```

```
mosquitto_sub -h thethings.meshed.com.au \
  -u enfieldlibrary_iot_trial \
  -P ttn-account-v2.-CAj7tWllLIQn1pCOZitrXI5ClM5j06uweHFubaqdPk \
  -t 'enfieldlibrary_iot_trial/devices/roaming_mawsonlakes/down'
```

Tools

```
jq
base64
xxd
```

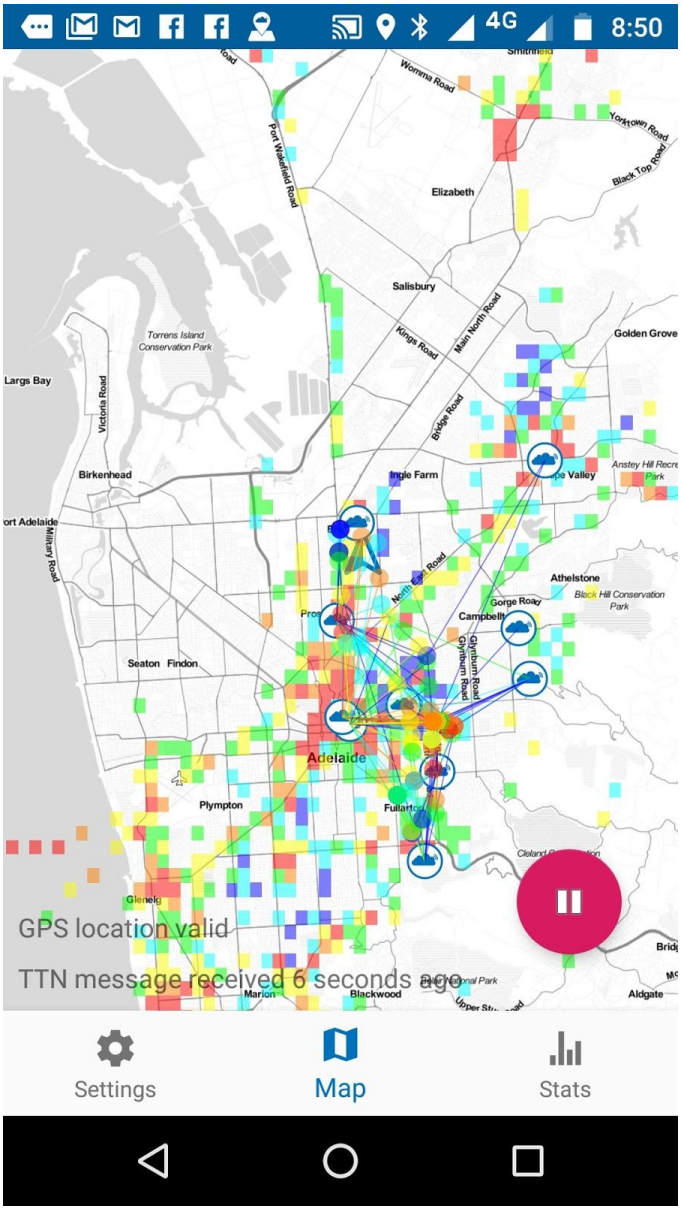
MQTTbox (Windows)

Available for free in app store.

TTN Mapper

Application useful for mapping TTN gateway coverage.

[Blog post for configuration settings]



Arduino Code Snippets

Twinkle LED

```
analogWrite(ledPin, ledState);
// set the pwm value of that pin
// determined previously
randNumber = random(-40, 41);
// generate new random number
// and add that to the current value
ledState += randNumber;
// that range can be tweaked
// to change the intensity of the flickering
if (ledState > 200) {
  // clamp the limits of the pwm values
  // so it remains within
  ledState = 200;
  // a pleasing range as well
  // as the pwm range
}
if (ledState < 10) {
  ledState = 10;
}
// delay(100);
// the delay between changes
// better if this is implemented
// without adding an explicit delay
```

Modes (m)

- Initial/start mode (0x00)
    - Red - connection status: flashing/solid/off
    - Orange - transmit
    - Yellow - ? Status Led
- Signal Report (0x01)
    - Red - connection pulse/slow twinkle
    - Orange - flash gateway number
    - Yellow - Signal report twinkle

Brightness/contrast? - strength

Twinkle Rate - signal to noise

```
int mode = 0
if (mode == 0){
}
```

References

Lora Alliance Documents - https://lora-alliance.org/lorawan-for-developers

Guidelines for using UIDs - https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/tutorials/eui.pdf

TTN Security (discussion)

https://mcuoneclipse.com/2019/05/26/reverse-engineering-of-a-not-so-secure-iot-device/

Resources

Accessing captive Wifi portals

See: Python script (on MacOS)
https://medium.freecodecamp.org/how-i-created-a-python-bot-to-automatically-log-into-a-captive-portal-3d4ba04dee9f

Hardware

Testing/Installation of Arduino LoRa 915Mhz Shield

- 10 shields purchased
- These shields were supplied with jumpers, but which weren't fitted. A couple only had 5 jumpers (rather then 6).
- Blue IC SP connector is tight. Care needs to be taken.

Test Procedure

- Unpackage
- Set jumpers
- Connect to arduino with TTN code
- Attach LED monitoring shield
- Apply power
- Wait for LoRAWAN connection status
- Look for TTN packet via MQTT
- (repeat) Clear MQTT message, press reset button Arduino an wait for second packet.

Common Faults

- If no LEDS lit on shield, check that jumpers have been fitted on LoRa board.

Training and Workshops

IoT Playground

Discussion

NodeRED Server: Having a NodeRED server on a separate server hardware allows:

- Multiple access and control of devices from multiple sources;
- Easy integration of additional devices
- Easy integration of additional controllers
- Avoid vendor dependance
- Avoid Internet access dependance, everything is local

NodeRED Development

- The NodeRED palette can be added via
    - Within NodeRED with Manage Palette
    - On the command line with 'cd ~/.node-red; npm install $PACKAGENAME'
    - Direct clone of GitHub repository into ~/.node-red, then link with 'sudo npm link $DIRECTORY

Restart

NodeRED Upgrade

- sudo apt update
- sudo apt upgrade
- sudo apt dist-upgrade
- sudo npm i -g npm
- npm outdated
- sudo npm update $PACKAGENAME

NodeRED Manually rebuild

- sudo update-nodejs-and-nodered

Securing NodeRED

- sudo npm install -g node-red-admin