Pycom Workshop 1

Micropython, LoRa, LoPy

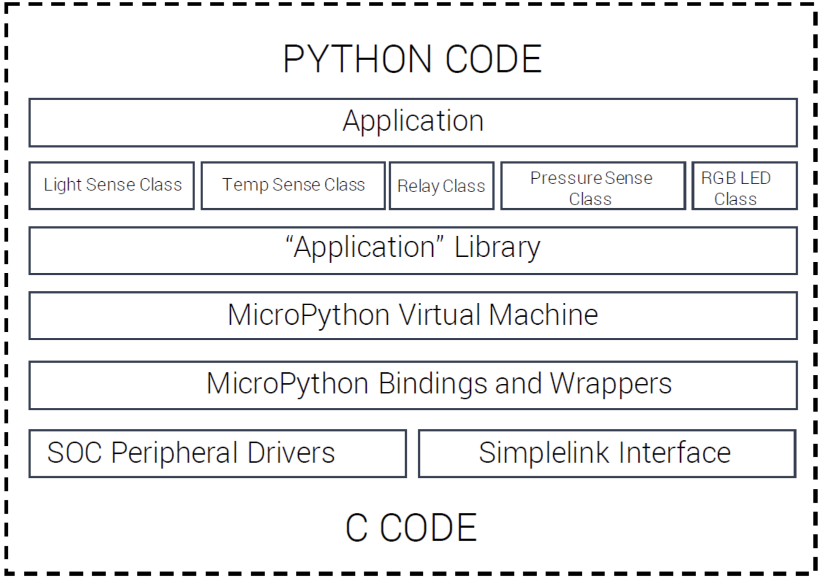
Rob Braggaar

02-05-2017

Micropython

- Lean and efficient implementation of Python 3
- Designed with IoT in mind
- 256k of code and 16k of RAM
- As compatible with Python as possible
- 10x faster development than C

Micropython



PYTHON CODE

Application

Light Sense Class | Temp Sense Class | Relay Class | Pressure Sense Class | RGB LED Class

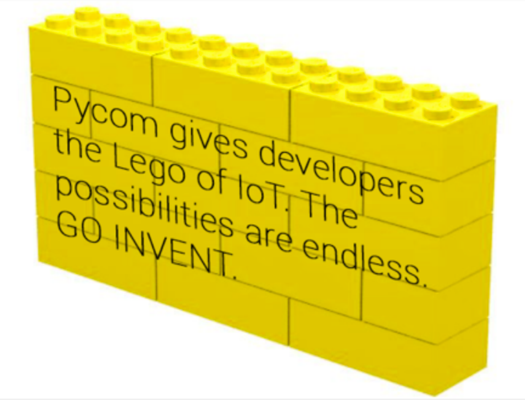
“Application” Library

MicroPython Virtual Machine

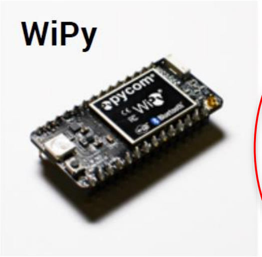
MicroPython Bindings and Wrappers

SOC Peripheral Drivers | Simplelink Interface

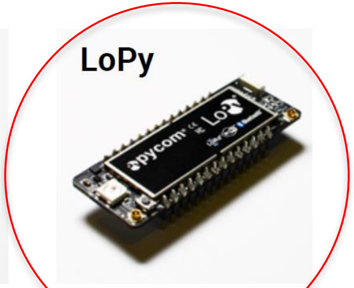
C CODE


Pycom gives developers the Lego of IoT. The possibilities are endless.
GO INVENT.

Pycom microcontrollers (e.g. Lopy)



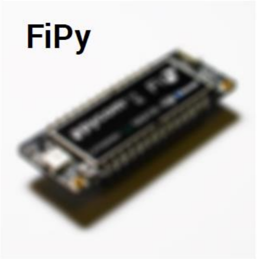
ESP32
WiFi
Bluetooth



ESP32
Semtech
LoRa
WiFi
Bluetooth

ESP32
Texas
Instruments
Sigfox
WiFi
Bluetooth

Multi-network
to be
announced



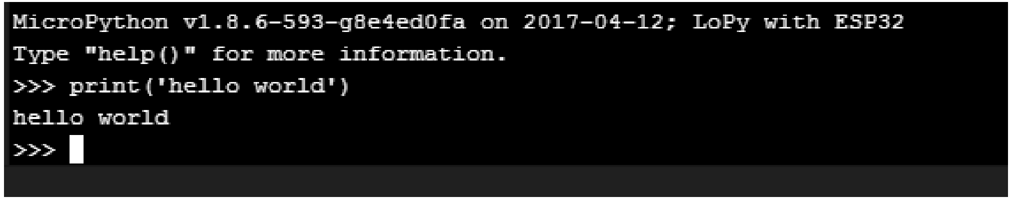
5-Network
Board
ESP32
LTE-M
LoRa
Sigfox
WiFi
Bluetooth

Pycom microcontrollers (e.g. Lopy)

- ESP32 based
- Very low power usage
- 100% MicroPython (Python v3)
- Interpreted vs compiled (Arduino/Marvin)
- Multi-network capability
- Lots of GPIOs, interfaces and peripherals
- Small form factor
- Prototype and OEM versions

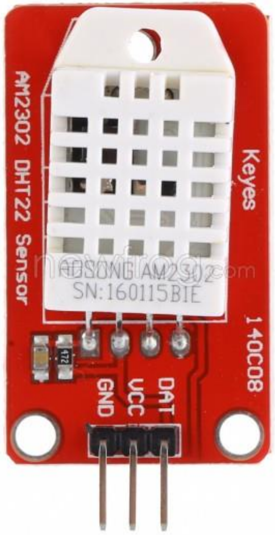
Difference with Arduino-like microcontrollers

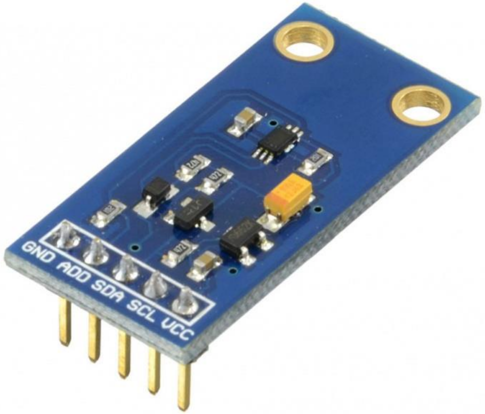
- Ability to directly/interactively run your code
- Small file system, works like standard python modules

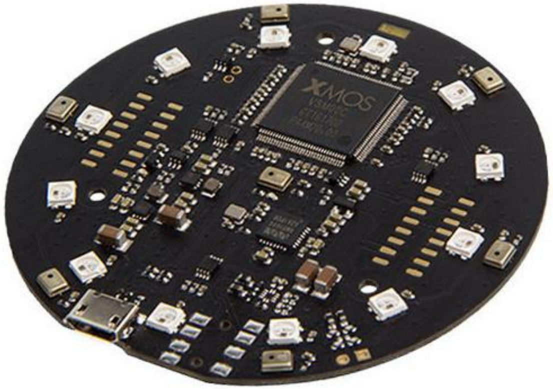


```
MicroPython v1.8.6-593-g8e4ed0fa on 2017-04-12; LoPy with ESP32
Type "help()" for more information.
>>> print('hello world')
hello world
>>> 
```

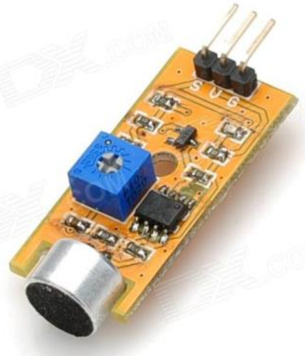
Sensors


AM2302 DHT22 Sensor
Keyes
AOSONG AM2302
SN:160115BIE
140C08
GND
VCC
DAT


GND ADD SDA SCL VCC


XMOS

GND RX TX VCC



Connections

- I2C (Inter-Integrated Circuit, "I squared C“)
- Analog
- UART (Universal Asynchronous Receiver Transmitter)
- SPI (Serial Peripheral Interface)
- Onewire

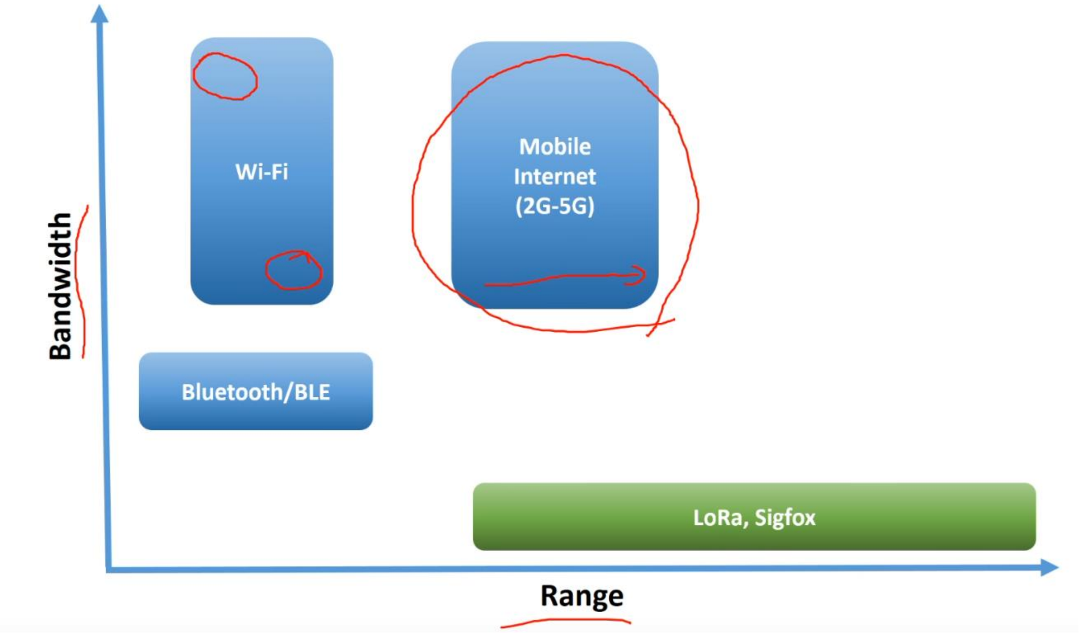
Network technologies

- LoRa
- WLAN
- BLE
- 3G / 4G

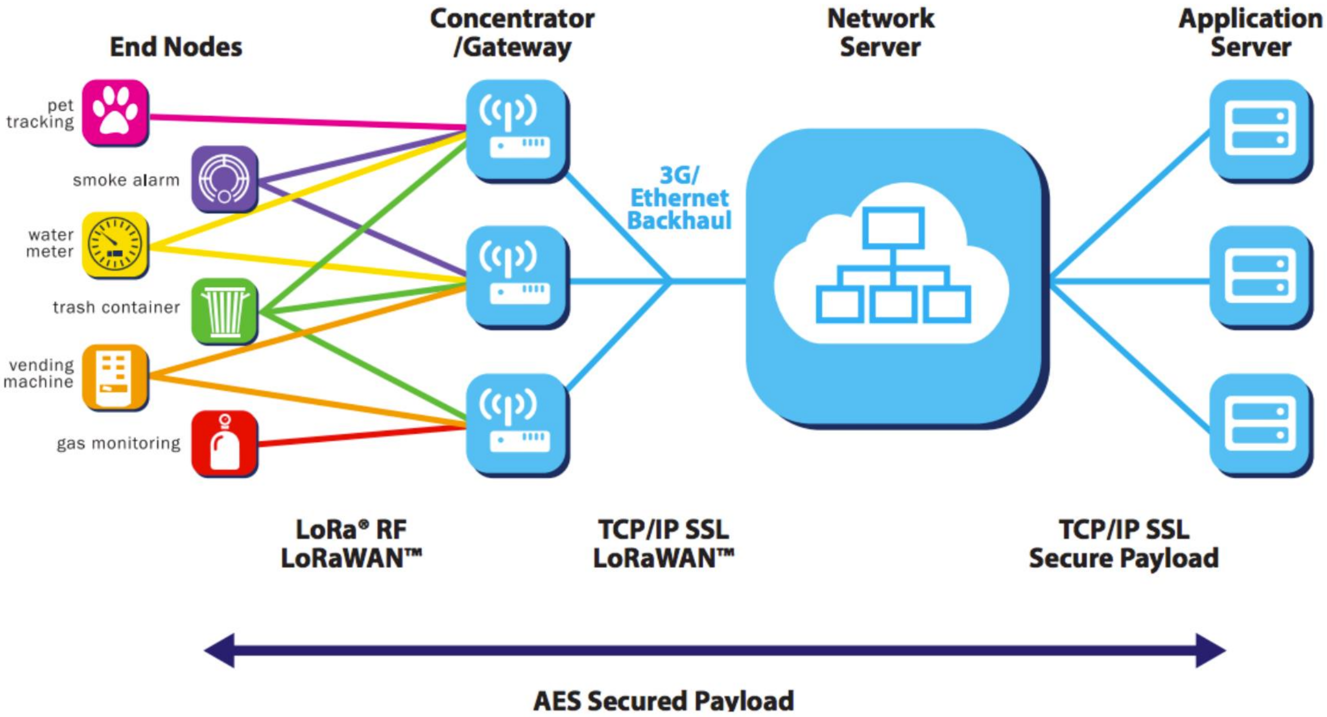
Comparison

Bandwidth

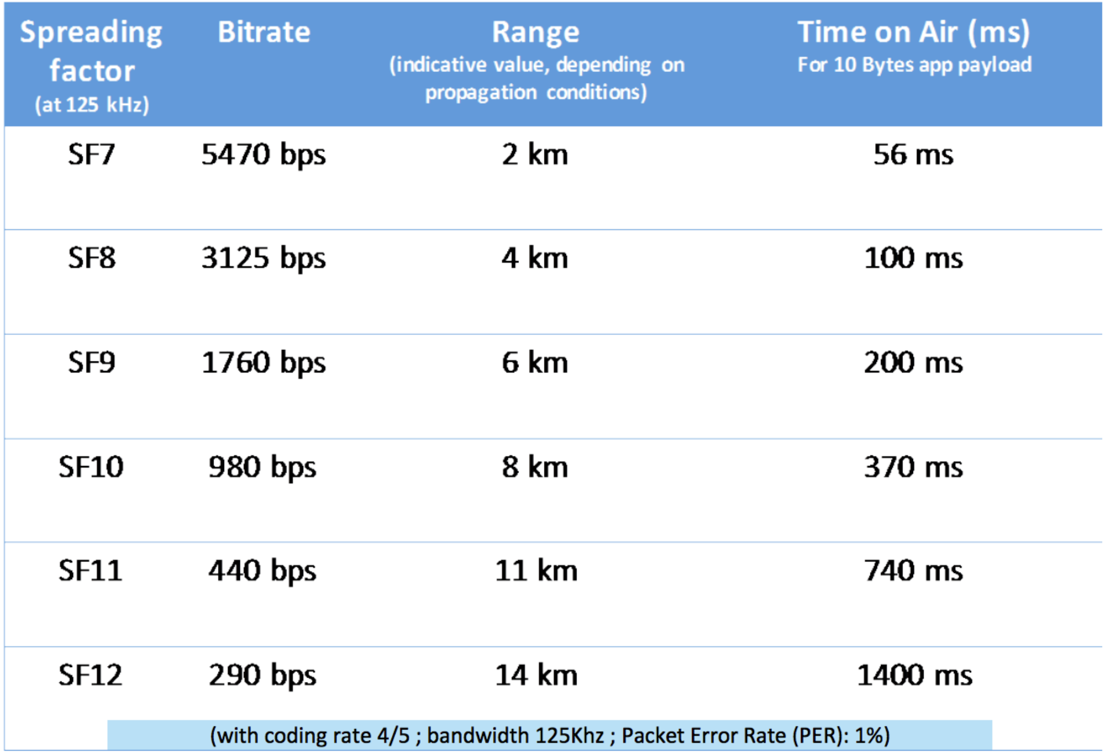
Wide Area ⟷ Low Power



LoRa architecture



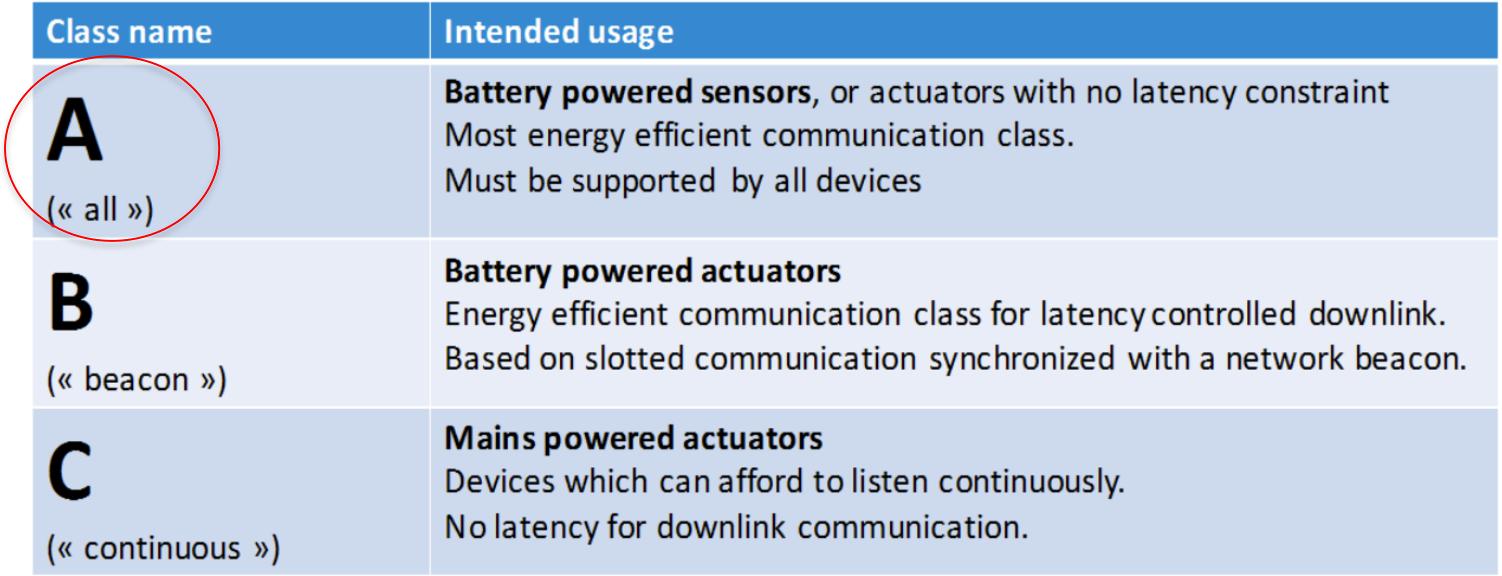
LoRa characteristics



Spreading factor (at 125 kHz)	Bitrate	Range (indicative value, depending on propagation conditions)	Time on Air (ms) For 10 Bytes app payload
SF7	5470 bps	2 km	56 ms
SF8	3125 bps	4 km	100 ms
SF9	1760 bps	6 km	200 ms
SF10	980 bps	8 km	370 ms
SF11	440 bps	11 km	740 ms
SF12	290 bps	14 km	1400 ms

(with coding rate 4/5 ; bandwidth 125Khz ; Packet Error Rate (PER): 1%)

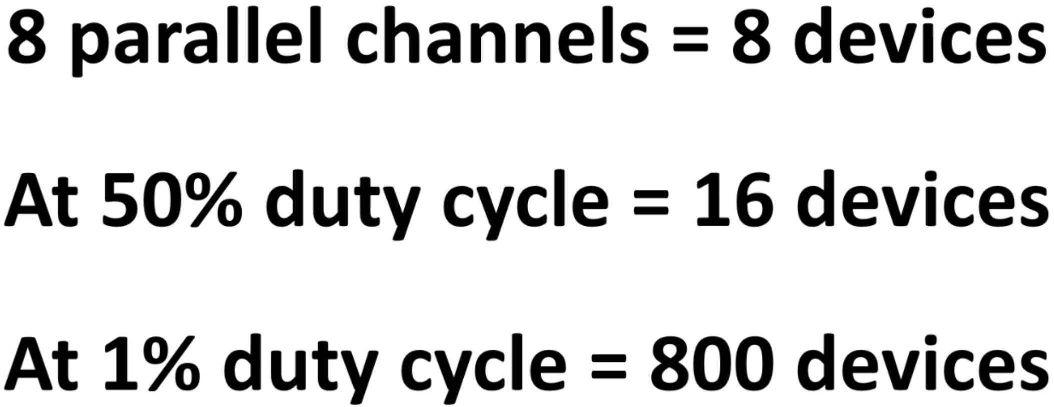
LoRa device classes



Class name	Intended usage
A (« all »)	**Battery powered sensors**, or actuators with no latency constraint Most energy efficient communication class. Must be supported by all devices
B (« beacon »)	**Battery powered actuators** Energy efficient communication class for latency controlled downlink. Based on slotted communication synchronized with a network beacon.
C (« continuous »)	**Mains powered actuators** Devices which can afford to listen continuously. No latency for downlink communication.

LoRa

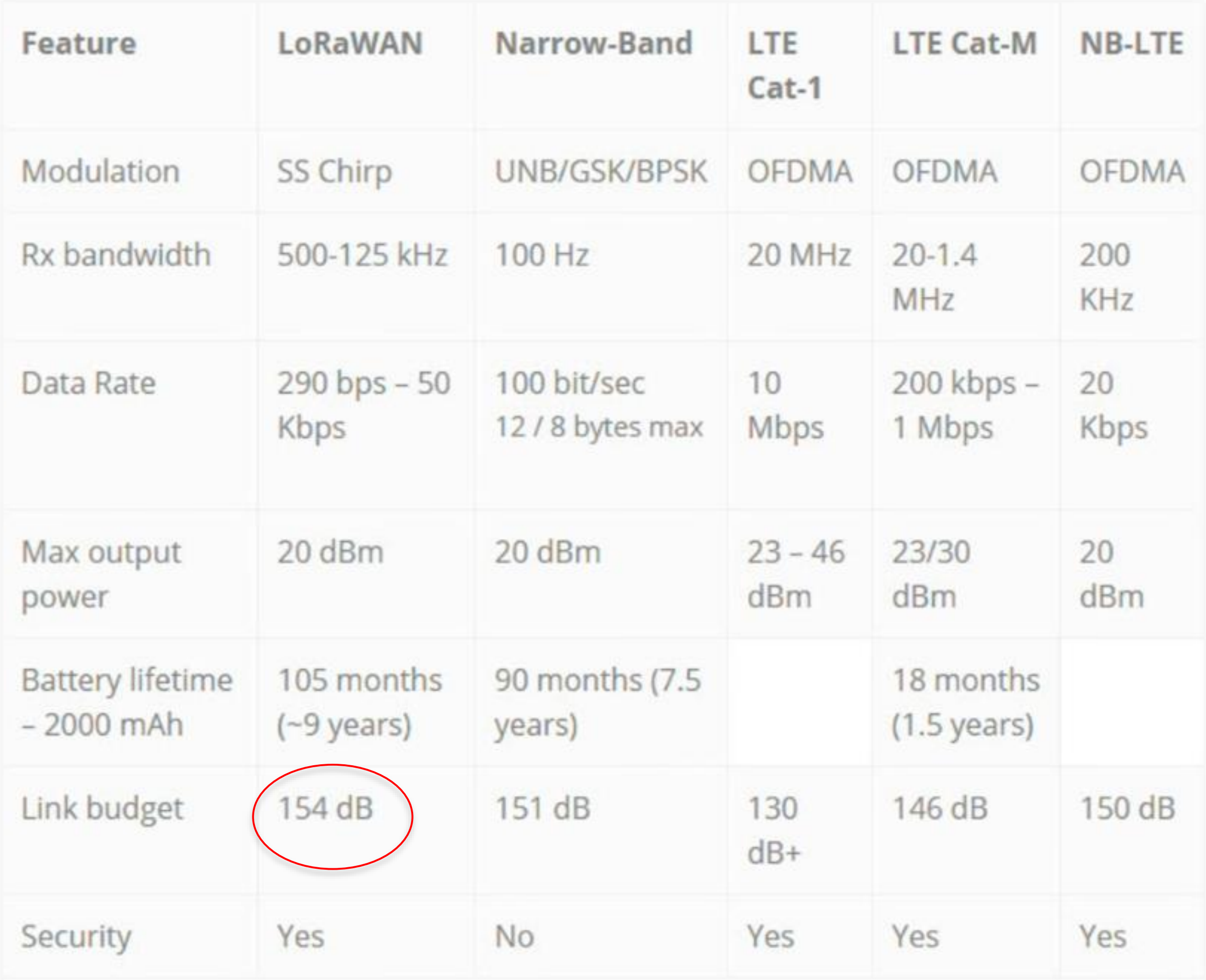
- LoRa nano gateway = 1 channel (e.g. Lopy)



8 parallel channels = 8 devices

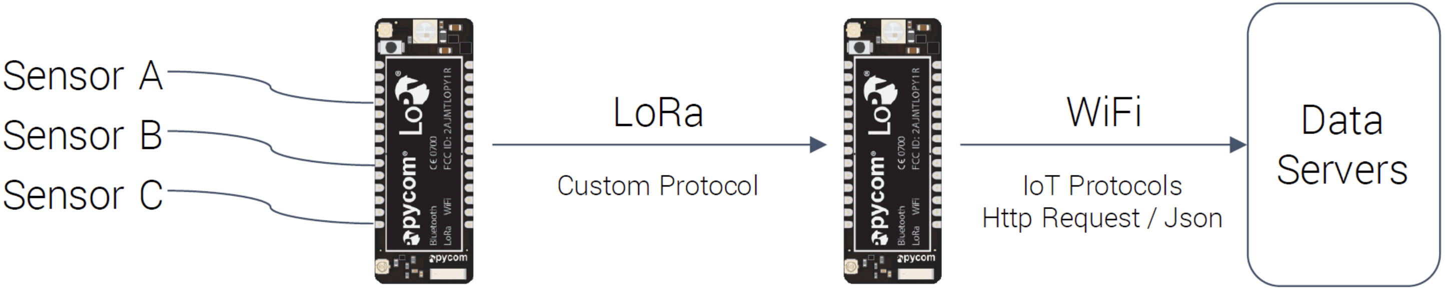
At 50% duty cycle = 16 devices

At 1% duty cycle = 800 devices



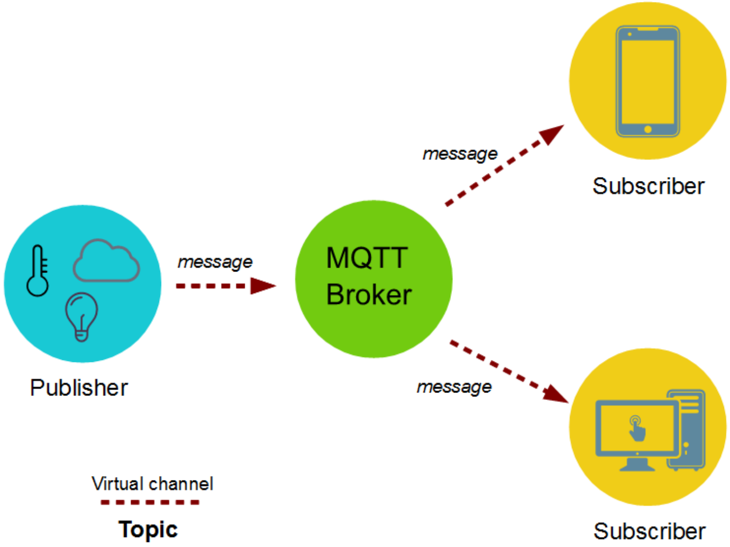
Feature	LoRaWAN	Narrow-Band	LTE Cat-1	LTE Cat-M	NB-LTE
Modulation	SS Chirp	UNB/GSK/BPSK	OFDMA	OFDMA	OFDMA
Rx bandwidth	500-125 kHz	100 Hz	20 MHz	20-1.4 MHz	200 KHz
Data Rate	290 bps – 50 Kbps	100 bit/sec 12 / 8 bytes max	10 Mbps	200 kbps – 1 Mbps	20 Kbps
Max output power	20 dBm	20 dBm	23 – 46 dBm	23/30 dBm	20 dBm
Battery lifetime – 2000 mAh	105 months (~9 years)	90 months (7.5 years)		18 months (1.5 years)	
Link budget	154 dB	151 dB	130 dB+	146 dB	150 dB
Security	Yes	No	Yes	Yes	Yes

LoRa


Sensor A
Sensor B
Sensor C
LoRa
Custom Protocol
WiFi
IoT Protocols
Http Request / Json
Data Servers

MQTT: IoT friendly protocol

- Ideal for sending small messages on unreliable networks

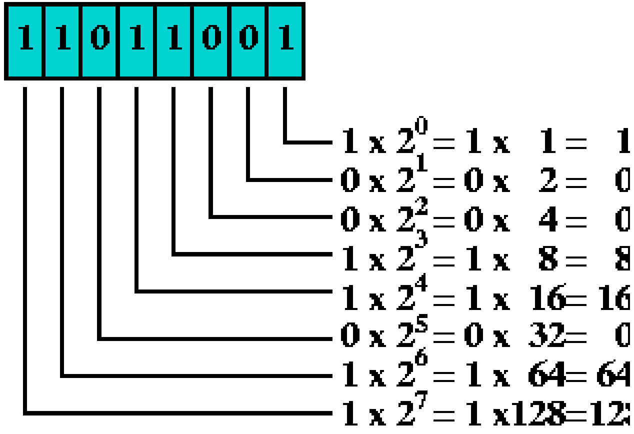


MQTT

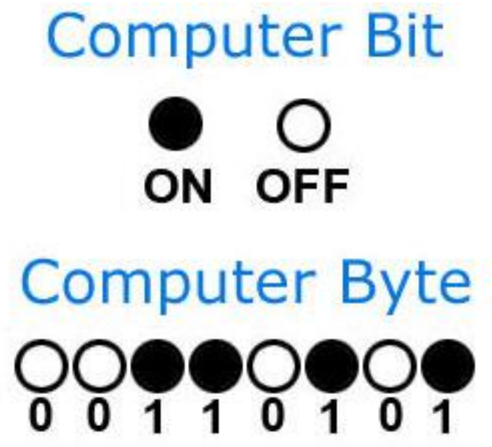
MQTT (MQ Telemetry Transport) is a machine-to-machine or “Internet of Things” connectivity protocol on top of TCP/IP. It allows extremely lightweight publish/subscribe messaging transport.

- Topic: address for messages
- /root_topic/subtopic
- QoS: 3 different levels
- Messages

Bits and bytes



$1 + 8 + 16 + 64 + 128 = 217$



http://www.computerhope.com

What you need today

- Development board (Lopy)
- 5V power supply (micro-usb or breadboard)
- Expansion board

Software preparation

Download the following programs:

- Atom
    - Ctrl + shift + p -> install packages
    - Install PyMakr plugin
- Putty (Windows only)
- FileZilla ftp client
    - Setup as a passive, plain connection, limit 1 conn.
- Github Desktop

Handle with care

- Expansion board usb connector
- Electrostatic shocks can damage the device
- Pins and other connectors are fragile

Before we get started

Update the firmware

- Latest version V1.6.12.b1

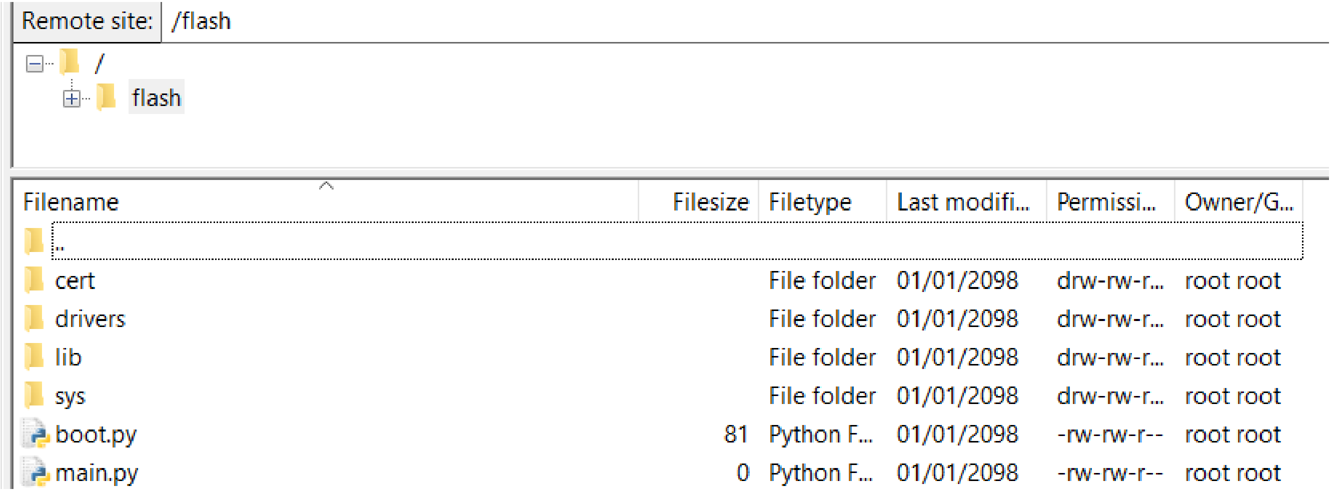
Github examples:

https://github.com/rbraggaar/sensor-city-delft

The filesystem

- boot.py (required)
- main.py (required)
- 512 Kb internal memory available for storage
- Other folders and files can be created, like drivers below:

  You need to add those folders to the module search path: e.g. sys.path.append(‘/flash/drivers’)



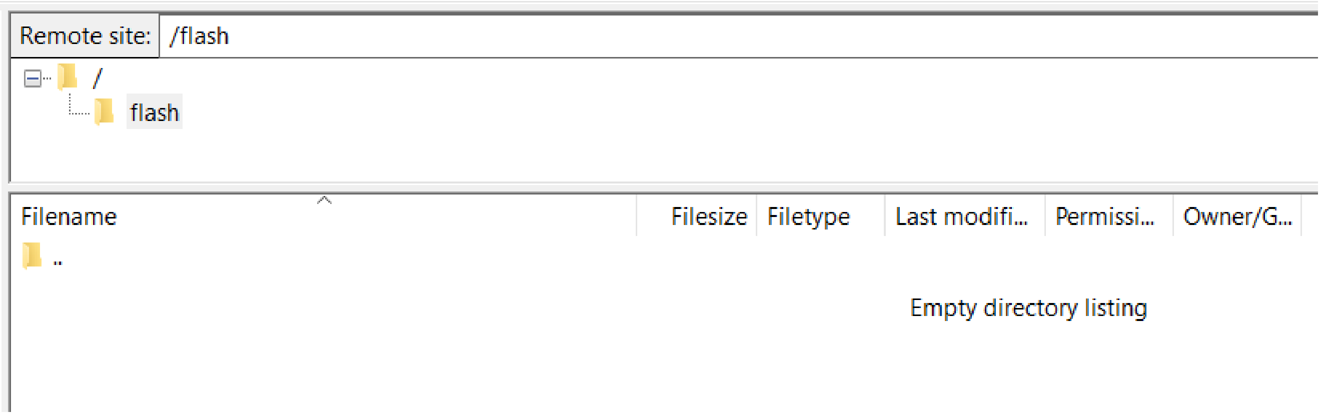
Remote site: /flash

- /
  - flash

Filename	Filesize	Filetype	Last modifi...	Permissi...	Owner/G...
..					
cert		File folder	01/01/2098	drw-rw-r...	root root
drivers		File folder	01/01/2098	drw-rw-r...	root root
lib		File folder	01/01/2098	drw-rw-r...	root root
sys		File folder	01/01/2098	drw-rw-r...	root root
boot.py	81	Python F...	01/01/2098	-rw-rw-r--	root root
main.py	0	Python F...	01/01/2098	-rw-rw-r--	root root

Reset filesystem

```
import os
os.mkfs(‘/flash’)
```



Remote site: /flash

Filename	Filesize	Filetype	Last modifi...	Permissi...	Owner/G...
..					

Empty directory listing

Example 1: onboard led

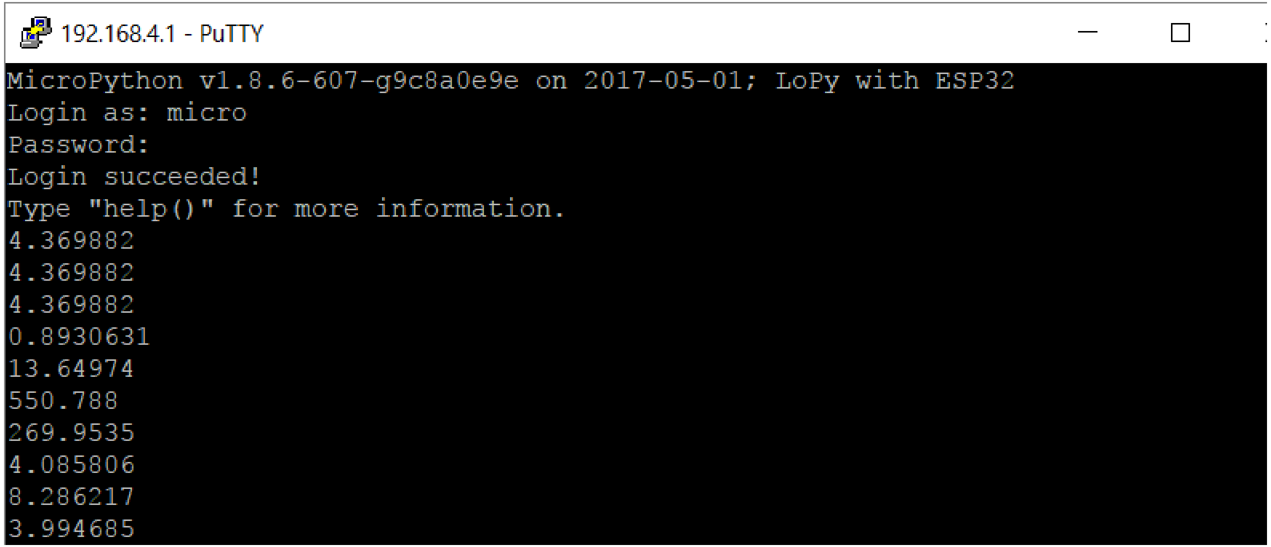
```
import pycom



pycom.heartbeat(False)
pycm.rgbled(0xff0000)
```

Example 2: light sensor

https://github.com/rbraggaar/sensor-city-delft/tree/master/light_sensor



```
MicroPython v1.8.6-607-g9c8a0e9e on 2017-05-01; LoPy with ESP32
Login as: micro
Password:
Login succeeded!
Type "help()" for more information.
4.369882
4.369882
4.369882
0.8930631
13.64974
550.788
269.9535
4.085806
8.286217
3.994685
```

For now

Start experimenting yourself and ask questions

Next time:

- electronic components and diagrams
- how to validate sensor data
- parsing GPS data