A Beginner's Guide to IoT using ESP8266 and MicroPython

George Koodarappally
@spicecoaster

Winter Is Coming

So come down to Sunny Puerto Rico

Agenda

- Presentation
  - IoT
  - MQTT
  - ESP8266
  - MicroPython
- Workshop
  - Hands on with ESP8266 & MicroPython

IoT

IoT - What is it?

- Internet-of-Things (IoT)
- Inter-networking of devices, computers/micro-controllers, sensors, actuators and communications
- Use of sensors to measure/monitor the local environment
- Use of actuators to contor/modify the local environment
- Use of networking to deliver messages between devices and backends
- Use of micro-controllers to coordinate sensors, actuators and networking
- [More info on IoT](https://en.wikipedia.org/wiki/Internet_of_things)

IoT - Types

- Consumer-Residential
- Consumer-Personal
- Business
- Industrial

IoT Devices - Consumer

IoT Devices around the house

NEST Thermostat

NEST Indoor Camera

Samsung Refrigerator

Samsung Refrigerator

IoT Devices - Personal

Fitbit Surge

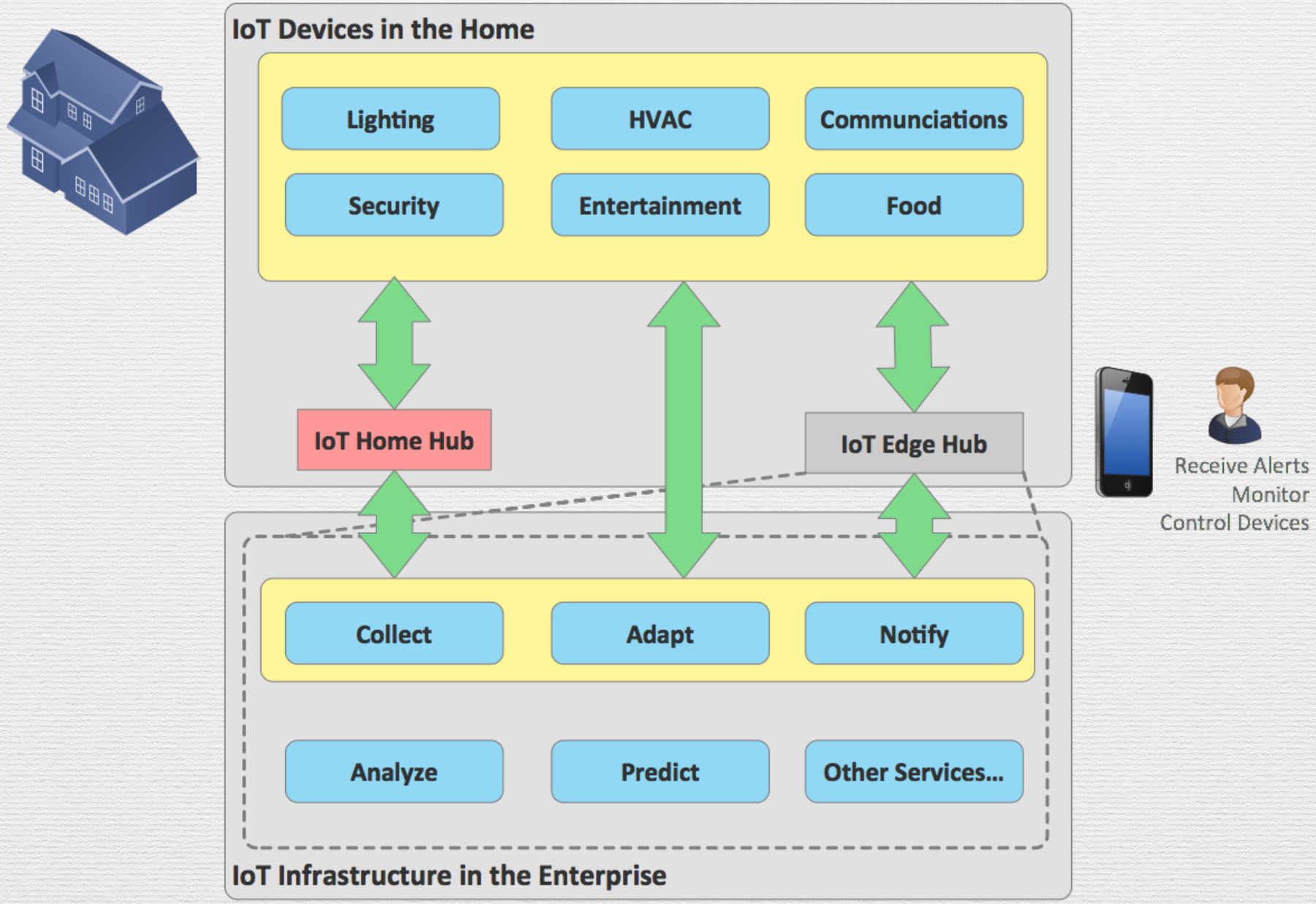
Qardioarm Blood Pressure Monitor

Images © Nest Inc., Samsung Inc., FitBit Inc., Cardio Inc.

IoT - Components

- Client Side Components
  - IoT Sensors
  - IoT Actuators
  - IoT Nodes
  - IoT Hubs
  - IoT Edge Devices
- Server Side Components
  - Gather Data
  - Analyze Data
  - Predict
  - Adapt
  - Machine Learning

The IoT Connected Home



MQTT

MQTT - What is it?

- Message Queue Telemetry Transport
- Publish/Subscribe Model
- Extremely simple and lightweight messaging protocol
- Designed for constrained devices
- Designed for low-bandwidth, high-latency or unreliable networks
- Minimizes network bandwidth and device resource requirements
- Offers various Quality-of-Service assurances
- Ideal for machine-to-machine(M2M)/IoT devices
- [More info on MQTT](http://mqtt.org/faq)

MQTT - Components

- MQTT Publishers
  - Devices can publish messages
- MQTT Subscribers
  - Devices can subscribe/listen to messages
- MQTT Topics
  - Clients publish messages to specific topics
  - Clients subscribe to messages on specific topics
- MQTT Brokers
  - Brokers provide Topic Queues for publishers to publish messages to and subscribers to receive messages on
  - Brokers coordinate receipt of published messages and distribution of subscribed messages

MQTT - Some Implementations

- Eclipse Paho (https://eclipse.org/paho/)
- Apache ActiveMQ/MQTT Adapter (http://activemq.apache.org/mqtt.html)
- RabbitMQ/MQTT Adapter (https://www.rabbitmq.com/mqtt.html)
- Mosquitto (https://mosquitto.org)
- HiveMQ ()

IoT Hardware

The IoT Hardware Options

Sensors

- Temp/Humidity/Pressure
- PIR Motion
- Light
- Accelerometers
- Liquid Level
- pH
- Magnetic

Actuators

- Relays
- DC Motor Controllers
- Servo/Stepper Controllers
- Solenoid/Pumps
- Etc...

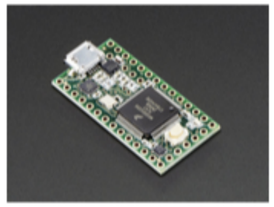
MicroControllers/SBC

Arduino

Raspberry Pi

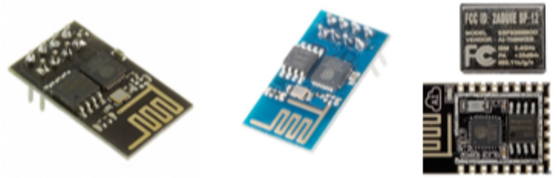
BeagleBone

Teensy



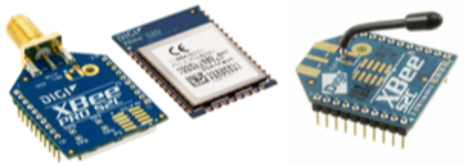
Communications

WiFi



Bluetooth Low Energy (BLE)

ZigBee



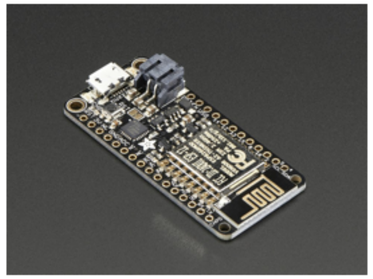
Long Range Radio (LoRa)

Images © Adafruit Industries, SparkFun Inc., Nest Inc., Samsung Inc., Espressif Inc. Digikey Inc.

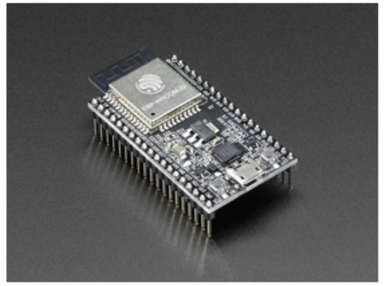
Example of IoT Development Boards

Adafruit

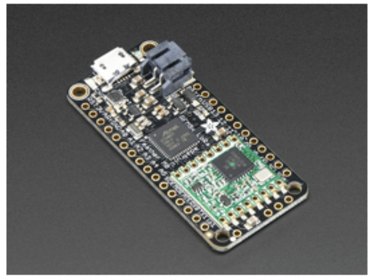
Feather HUZZAH ESP8266



Espressif ESP32 Dev Board



Feather LoRa



SparkFun

SparkFun ESP8266 Thing

SparkFun ESP32 Thing

PyCom

PyCom WiPy

PyCom LoPy

PyCom SiPy

Images © Adafruit Industries, SparkFun Inc., Pycom Inc.

ESP8266

IoT Using ESP8266

Sensors

- Temp/Humidity/Pressure
- PIR Motion
- Light
- Accelerometers
- Liquid Level
- pH
- Magnetic

ESP8266 MicroController
+ WiFi Communications

Actuators

- Relays
- DC Motor Controllers
- Servo/Stepper Controllers
- Solenoid/Pumps
- Etc...

Images © Adafruit Industries, SparkFun Inc., Espressif Inc.

ESP8266 - Functional Specs.

- System-on-Chip (SoC) from Espressif
- Highly integrated SoC with WiFi solution
- Integrated Tensilica L106 32-bit processor with on-chip SRAM
- Can perform as a standalone application or slave to host MCU
- WiFi can act as Access Point or Station mode.
- [More info on Espressif ESP8266 WiFi solution](https://espressif.com/en/products/hardware/esp8266ex/overview)

ESP8266 - Technical Specs.

- 32-bit RISC CPU: Tensilica Xtensa LX106 running at 80MHz
- 64 KB of instruction RAM, 96 KB of data RAM
- External QSPI flash - 512 KB to 4 MB* (up to 16 MB is supported)
- IEEE 802.11 b/g/n Wi-Fi
- WEP or WPA/WPA2 authentication, or open networks
- 16 GPIO pins
- SPI, I^2C,
- UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
- 1 10-bit ADC
- [More info on Espressif ESP8266](https://en.wikipedia.org/wiki/ESP8266)

Adafruit Feather HUZZAH ESP8266

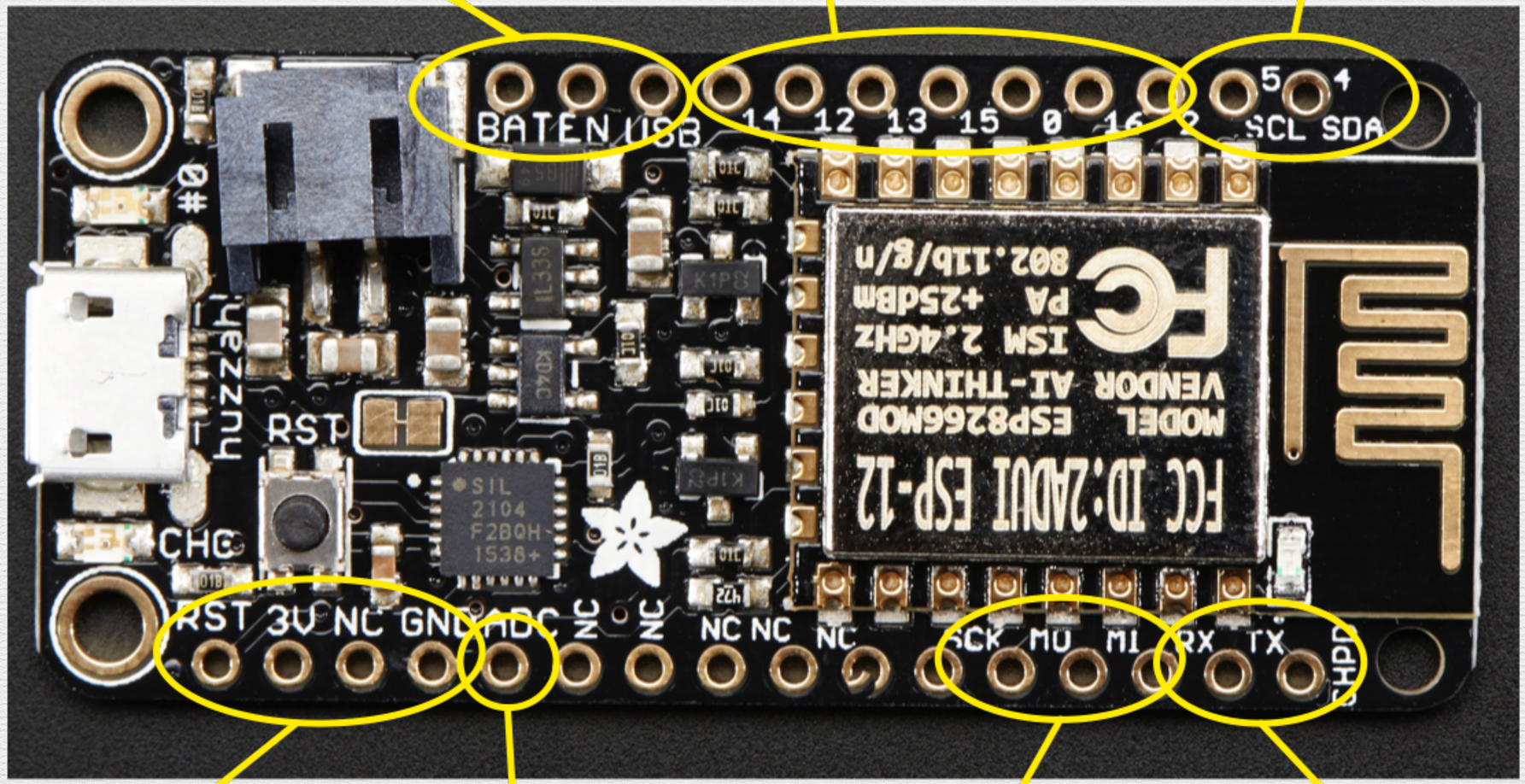
- A development board from Adafruit incorporating the ESP8266 WiFi module
- Incorporates the ESP8266 @ 80 MHz or 160 MHz
- 4MB of Flash
- 3.3V regulator with 500mA peak current output
- CP2104 USB-Serial convertor board with 921600 max baudrate
- 9 GPIO pins - can be used as I2C and SPI
- 1 analog input 1.0V Max
- 2 Pins connected to LEDs
- Power/Enable pin
- Reset button
- [More info on Adafruit Feather HUZZAH ESP8266](https://learn.adafruit.com/adafruit-feather-huzzah-esp8266/overview)

Adafruit Feather HUZZAH ESP8266

Power/Control Pins
BAT: Postitive Voltage from LiPoly Battery
USB: Positve Voltage for USB Micro
EN: 3.3V Regulator Enable

GPIO Pins (3.3V I/O)
Pins 0, 2, 4, 5, 12, 13, 14, 15, 16

I2C Pins
SCL: Serial Clock
SDA: Serial Data



Power/Control Pins
3V: 3.3V Regulator Output
GND: Ground
RST: Reset

A/D Pins
ADC: Analog-to-Digital Input

SPI Pins
SCK: Serial Clock
MOSI: Master Output Slave Input
MISO: Master Input Slave Output

Serial Pins
RX: Receive
TX: Transmit

Images © Adafruit Industries

IoT Device Firmware/Operating Systems

- RTOS (FreeRTOS) (http://www.freertos.org)
- Arduino (https://www.arduino.cc)
- NodeMCU Lua (http://www.nodemcu.com/index_en.html)
- MicroPython (https://micropython.org)

MicroPython

MicroPython - What is it?

- Based on Python 3.x
- Implements a small subset of the Python standard library
- Optimized to run on microcontrollers in constrained environments
- Fits and runs within 256K of code space and 16K of RAM
- Developed by Dr. Damien P. George (Cambridge University)
- [More info on MicroPython](https://micropython.org)
- [More info on MicroPython for ESP8266](http://docs.micropython.org/en/latest/esp8266/)

MicroPython Standard/Micro Libraries

- Builtin Functions
- array, gc, math, sys
- ubinascii, ucollections, uhashlib, uheapq
- uio, ujson, uos, ure
- usocket, ussl
- ustruct, utime, uzlib
- [More info on MicroPython Standard Libraries](http://docs.micropython.org/en/latest/esp8266/library/index.html)

MicroPython Specific Libraries

- machine
- micropython
- network
- uctypes
- [More info on MicroPython Specific Libraries](http://docs.micropython.org/en/latest/esp8266/library/index.html)

MicroPython ESP8266 Port Libraries

- esp
- [More info on MicroPython ESP8266 Port Libraries](http://docs.micropython.org/en/latest/esp8266/library/esp.html)

MicroPython External Libraries

- micropython-lib GitHub project
- umqtt
- [More info on MicroPython-Lib Project](https://github.com/micropython/micropython-lib)

MicroPython - Firmware - Pre-Compiled Binaries

- Precompiled Binaries are available for various boards
- Available for download from the MicroPython Downloads page
- Easiest way to get started with MicroPython

MicroPython - Firmware - Compile From Source

- Source code repository is available for each supported board
- Platform specific compiler tool chain needs to be downloaded
- Uses the Tensilica compiler tool chain from Espressif for ESP8266
- Uses a VM launched via Vagrant for compiling
- Allows selective inclusion of necessary modules
- Allows inclusion of external libraries
- User modules can be 'frozen' into the binary
- Frozen modules store the Python source with the firmware
- Frozen bytecode stores the compiled bytecode with the firmware
- Modules need to be placed in a specific directory
- Compilation process is long but binaries are more space efficient
- Flash the resulting binary to the device

MicroPython - Firmware - Boot Sequence

- Two files are used to boot up the device
- boot.py is the first file that is loaded and executed
- main.py if available is then loaded
- main.py is optional and contains the user application program
- main.py can either the full code or can call applicaton code in other application modules
- main.py is usually edited on the host machine and then pushed to the device
- If there is no main.py then REPL automatically launches

MicroPython - Firmware Installation

- Download the latest pre-compiled firmware from the MicroPython Downloads page (http://micropython.org/download#esp8266)
- Download and install the firmware deployment tool, esptool.py, from the following page (https://github.com/espressif/esptool)
- Connect the device (Adafruit Feather HUZZAH ESP8266) to the computer using the USB Micro cable
- Identify the serial port that is connected to the device
- Install the firmware onto the device using the esptool.py deployment tool
- Reset the device so that it boots into the MicroPython REPL

MicroPython - Firmware Deployment Tool (esptool.py)

esptool.py -h

```
$ esptool.py
usage: esptool [-h] [--port PORT] [--baud BAUD]

{load_ram,dump_mem,read_mem,write_mem,write_flash,run,image
_info,make_image,elf2image,read_mac,chip_id,flash_id,read_f
lash,verify_flash,erase_flash,version}
    ...
esptool: error: too few arguments
```

MicroPython - Firmware Deployment Tool (esptool.py)

```
$ esptool.py --port /dev/tty.SLAB_USBtoUART erase_flash

$ esptool.py --port /dev/tty.SLAB_USBtoUART --baud 460800
write_flash --flash_size=detect 0 esp8266-20170108-
v1.8.7.bin
```

MicroPython - Making the Connection

- Through the serial port REPL
- Through the WiFi WebREPL

Making the Connection - SerialREPL

- Connect the device using the USB cable

```
$ screen /dev/tty.SLAB_USBtoUART 115200


>>>
```

Making the Connection - WebREPL

```
$ screen /dev/tty.SLAB_USBtoUART 115200
>>> import webrepl_setup
WebREPL daemon auto-start status: disabled
Would you like to (E)nable or (D)isable it running on boot?
(Empty line to quit)
> E
To enable WebREPL, you must set password for it
New password: password
Confirm password: password
Changes will be activated after reboot
Would you like to reboot now? (y/n) y
>>>
...
 WebREPL daemon started on ws://192.168.4.1:8266
 Started webrepl in normal mode
 could not open file 'main.py' for reading
 MicroPython v1.8.7-7-gb5a1a20a3 on 2017-01-09; ESP module with ESP8266
 Type "help()" for more information.
 >>>
```

MicroPython - machine Module

- The machine module contains functions and classes that control the hardware on the board
- Some of the functions available:
  - reset(), freq(), idle(), sleep(), deepsleep()
- Some of the classes available:
  - Pin - controls the I/O pins on the board
  - I2C - a two-wire serial protocol for communications with devices
  - SPI - a synchronous serial protocol driven by a master
  - Timer - hardware timers
  - UART - implements the standard UART/USART duplex serial protocol
  - WDT - watchdog timer used to restart the system when the application crashes.

MicroPython - network Module

- The network module provides network drivers and configurations
- Contains the WLAN class which contains the following methods:
  - wlan.active(), wlan.connect(), wlan.disconnect(), wlan.scan(), wlan.status(), wlan.isconnected(), …

MicroPython - esp Module

- The esp module contains functions related to the ESP8266 module
- Examples of methods are:
  - esp.deepsleep()
  - esp.flash_id()
  - esp.flash_write()
  - esp.flash_read()
  - esp.flash_erase()

MicroPython-Lib - umqtt.simple Module

- The umqtt.simple module contains functions related to connecting to an MQTT Broker
- Examples of methods are:
  - connect(), disconnect()
  - publish(), subscribe()
  - set_callback()
  - set_last_will()
  - wait_msg()
  - check_msg()

Questions???

Workshop

Downloads Need

- micropython (http://micropython.org/download#esp8266)
- USB SLAB Driver for OS X (https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers)
- esptool.py (https://github.com/espressif/esptool)
- mpfshell (https://github.com/wendlers/mpfshell)
- mosquito (https://mosquitto.org/download/)
- sample code at github (https://github.com/spicecoaster/pycaribbean2017)
  - Workshop subdirectory

Topics

- Flash the ESP8266 with MicroPython
- Connect to REPL on ESP8266
- Enable WebREPL and connect to ESP8266
- Hello World of MicroPython - Blinky.py
- Connecting the ESP8266 to WiFi as a Station
- Sample MQTT publish from ESP8266
- Sample MQTT subscribe from ESP8266

Important Commands

- esptool
- mpfshell
  - mpfshell -c ‘open tty.SLAB_USBtoUART’
  - mpfshell/ > put main.py
- screen
  - screen /dev/tty.SLAB_USBtoUART 115200
  - Press Enter if the screen is blank
  - You should see ‘>>> ‘ prompt

WiFi Details

- For WebREPL
  - Find the WiFi AccessPoint that has the form ‘MicroPython-aaaaa’ where ‘aaaaa’ is based on your boards MAC Address. Look behind the board for your boards Access Point
  - Password: micropythoN
- For other labs:
  - SSID: PYCARIB_IOT_WORKSHOP
  - Password: WELCOMETOIOT

Connecting to REPL

```
$ screen /dev/tty.SLAB_USBtoUART 115200

$
```

Connecting to WebREPL

- Connect to the Micropython-aaaaa AP with password: micropythoN
- Open a browser tab and go to ‘http://micropython.org/webrepl'
  - This should open a web page with a text box at the to right that will say
    - ws://192.168.4.1:8266
    - and a button that says ‘Connect’.
  - Press ‘Connect’ and the screen below should show the MicroPython REPL prompt ‘>>> ‘
  - type help() to see options to paste code in…

Blinky - Hello World

- Download the sample_blinky_red.py from the github page
- Copy sample code to main.py
- use mpfshell to place the main.py onto the board
- Reset the board

Extra Credit: Try the blue blinky also

Access WiFi code

- Download the sample_wifi_station.py from the github page
- Copy sample code to main.py
- use mpfshell to place the main.py onto the board
- Reset the board

MQTT Publish

- Download the sample_mqtt_pub.py from the github page
- Copy sample code to main.py
- use mpfshell to place the main.py onto the board
- Reset the board
- Launch ‘mosquitto’
- Launch ‘mosquitto_sub’
- Watch for messages

MQTT Subscribe

- Download the sample_mqtt_sub.py from the github page
- Copy sample code to main.py
- use mpfshell to place the main.py onto the board
- Reset the board
- Launch ‘mosquitto’
- Launch ‘mosquitto_pub’
- Watch for RED led to turn on or off