Build Your Own IoT Infrastructure Using Low Cost Devices and the Public Cloud – A Practical Tutorial

ITPalooza 2016
IoT Session
2-2:45 PM (just before cafecito time!)

Introduction to IoT using the ESP8266

Mahesh Neelakanta
Director of IT, College of Engineering & Computer Science
Florida Atlantic University
mahesh@fau.edu

What to expect from this session…

- Understanding of IoT
- The ESP8266 Platform
- Cloud and IoT
- Code examples & Demo
- Questions?
  - http://s.fau.edu/itpalooza

Agenda

- How did I end up here?
- What is IoT?
- What is the ESP8266?
- Serverless computing and AWS Lambda (Cloud!)
- Code Examples (Node & Sketch & C)
- Demo Time!
- Q & A

What is the IT guy doing here in front of you?

* How did I end up here?

Two reasons….

What is my pool temperature?

How can I share what I learned with others?

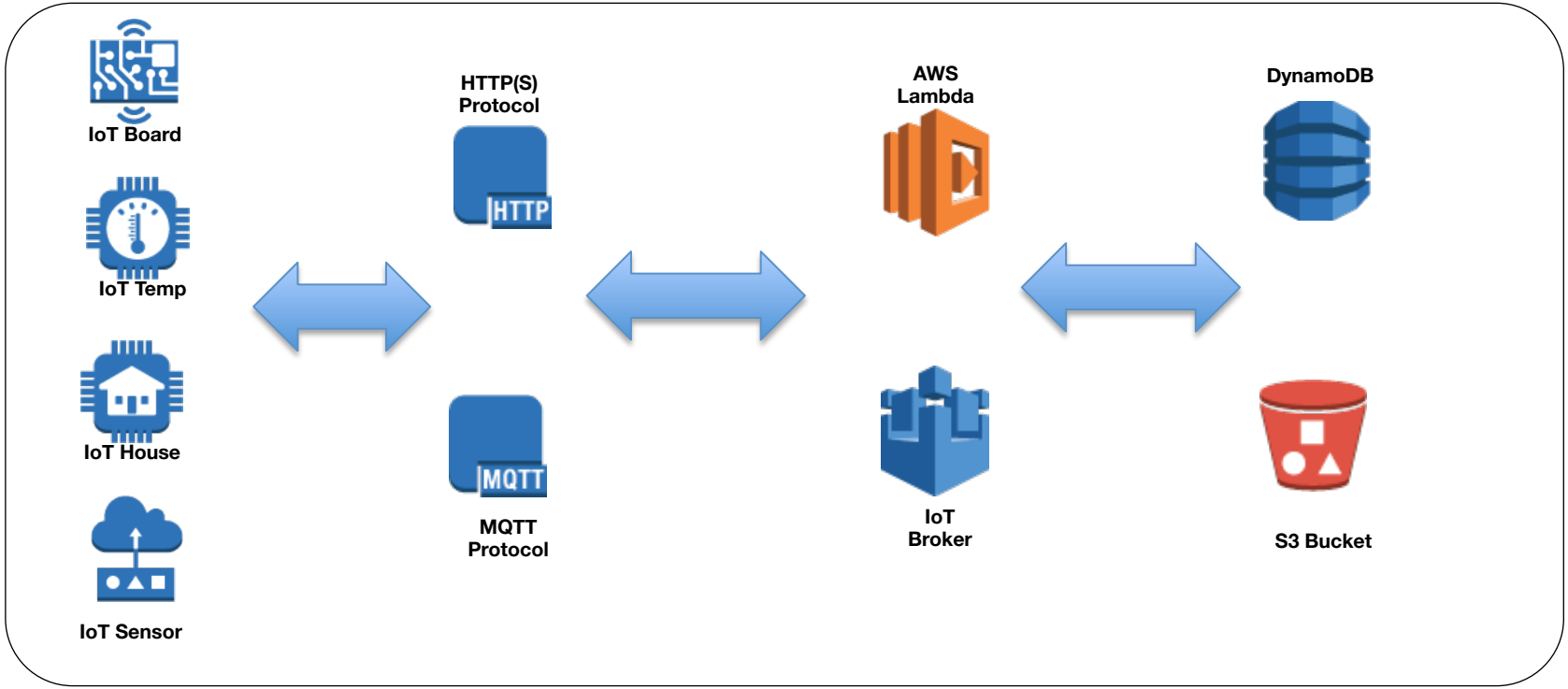
- IoT
- ESP8266
- Cloud Services
- …alexa..

What is IoT?

IoT provides a way to connect disconnected and semi-connected devices and sensors by adding metrics and telemetry that is sent to the cloud so that they can provide useful information analytics ; and send that information back so that those devices can act in a more meaningful way.

Chris McCurdy
Solutions Architect Specialist in Healthcare and Lifesciences, Amazon Web Services
(HLC304-R - Building IoT Applications with AWS and Amazon Alexa)
re:Invent 2016 - https://www.youtube.com/watch?v=7AVTyNlgNV4

a message queue



What?

- Connect any “thing” to/thru the internet (or intranet)
- Unidirectional or bi-directional traffic-flow (publish/subscribe)
- Small amounts of data from one device
- Large amounts of data across 1000s of devices
- Mostly machine-to-machine (M2M)
- Time-series based data

- Temperature, humidity, soil moisture, luminance, on/off, speed, position, altitude, pressure, salinity, CO2, CO, air quality…
- Counters, gauges, levels, booleans…
- Sensor networks, vehicular networks, smart building,…
- Crowd sourced data

How?

Layer 1-2

- Wi-Fi
- GSM / GPRS / LTE
- Bluetooth Low Energy (BLE)
- Serial / USB
- Zwave / ZigBee
- XBee 2.4 GHz or 900 Hz
- nRF2401A 2.4 GHz
- Browser (Javascript)
- ..or...you choose

Layer 3+

- MQTT Message Queue Telemetry Transport
- XMPP eXtensible Messaging and Presence Protocol
- CoAP Constrained Application Protocol
- HTTP(S) / REST
- UDP / TCP
- SMS
- Gateways and Proxies

Where?

Use a Platform

- AWS IoT
- Azure IoT
- IBM Bluemix IoT
- ATT IoT
- Telit IoT
- CloudMQTT.com
- thingspeak.com
- pubnub.com
- freeboard.io
- Sparkfun.com (phant)
- IOT-Playground.com
- Ubidots.com

Build your own

- Open Source MQTT Brokers (Mosquitto)
- Time Series Databases (DynamoDB, InfluxDB, Riak, Prometheus,…)
- Visualization tools (Grafana, Kibana, D3.js, Matlab, Freeboard,...)
- NodeRed (Visual Coding)

When?..or how often?

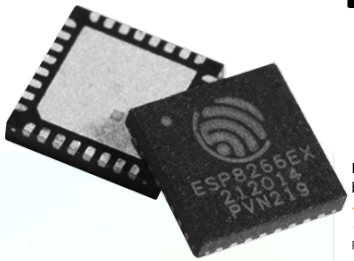
- Always “on” (but deep sleep to save power)
- Send data periodically (every 1 minute, 5 minutes, 15 minutes, hourly, etc.)
- Retain last known state (AWS IoT shadow) in case of loss of connectivity

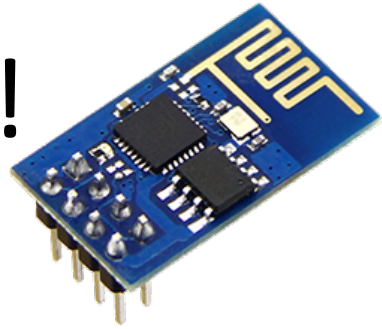
What is the ESP8266?

What can $4 USD get you these days?

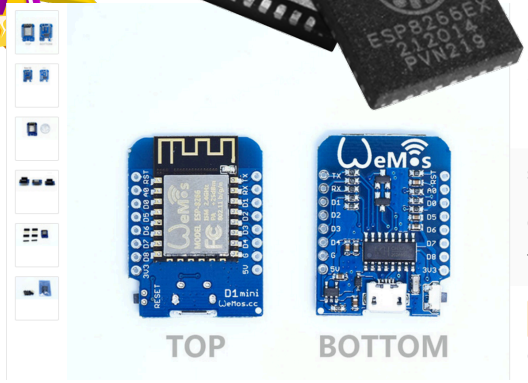
or

Or an Espressif ESP8266 !





www.aliexpress.com



D1 mini V2 - Mini NodeMcu 4M bytes Lua WIFI Internet of Things development board based ESP8266 by WeMos

4.9 (5390 votes) | 7689 orders

Price: **US $4.00** / piece

Find more deals on the app

Shipping: Free Shipping to United States via China Post Ordinary Small Packet Plus

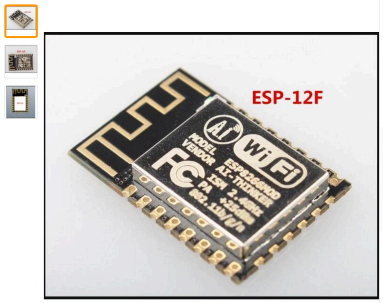
Estimated Delivery Time: 15-26 days (ships out within 7 business days)

Quantity: 1 piece (11637 pieces available)

Total Price: US $4.00

Buy Now | Add to Cart

Add to Wish List (2547 Adds)



2015 New version 1PCS ESP-12F (ESP-12E upgrade) ESP8266 remote serial Port WIFI wireless module

5.0 (136 votes) | 828 orders

Price: ~~US $1.80 / piece~~

Discount Price: **US $1.71** / piece 5% off 2 days left

Find more deals on the app

Shipping: Free Shipping to United States via China Post Ordinary Small Packet Plus

Estimated Delivery Time: 15-26 days (ships out within 7 business days)

Quantity: 1 piece (655 pieces at most per customer)

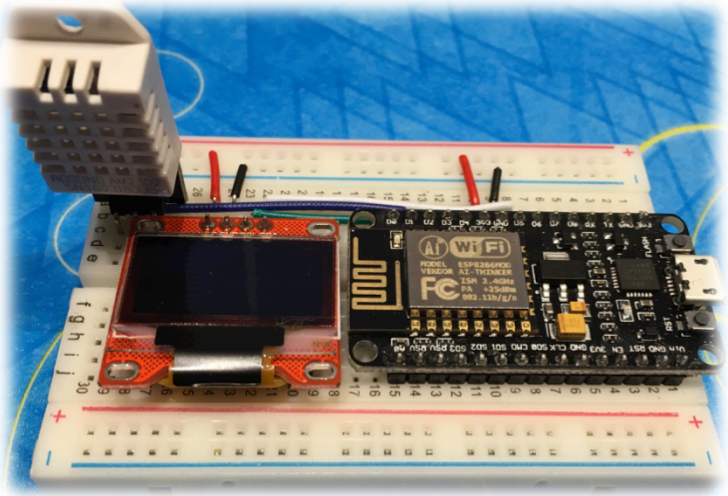
Total Price: US $1.71

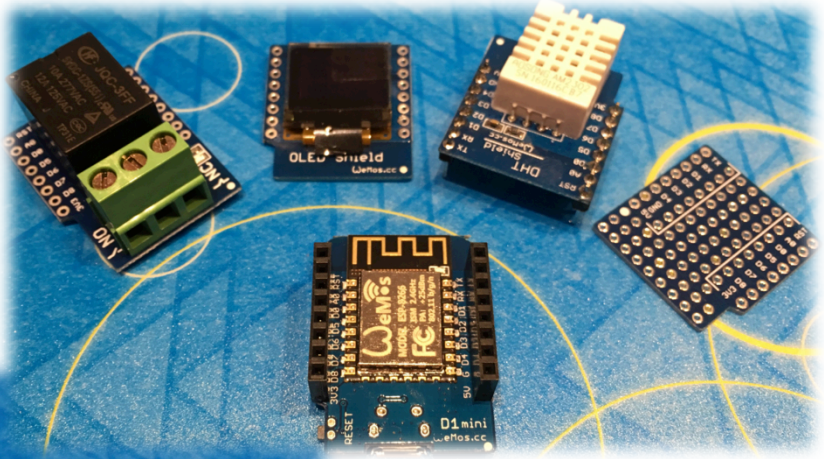
Buy Now | Add to Cart

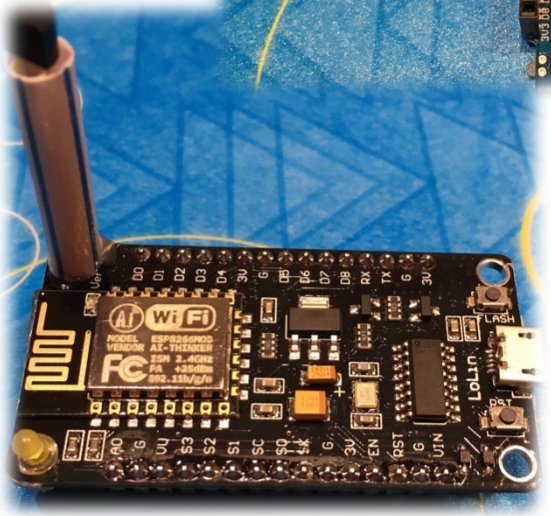
Add to Wish List (318 Adds)

* iotawards.postscapes.com

Build your own or buy pre-built shields







Espressif ESP8266 Specifications

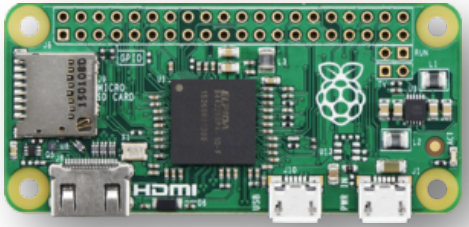
- CPU: 80/160 MHz Tensilica Xtensa LX106
- RAM: 64 KiB of instruction RAM, 96 KiB of data RAM
- Flash: 512M - 16M, depending on model
- WiFi embedded into MCU. 802.11 a/g/n
- Station or Access-Point Mode
- PCB Trace Antenna, Ceramic Antenna or UFL Antenna
- 16 x GPIO pins - SPI, I2C, I2S, UART capable
- 1 x 10-bit ADC

ESP8266 Strengths

- Easy to program:
  - LUA, MicroPython or Basic
  - Arduino Sketch (C Like)
  - FreeRTOS or C (SDK)
- Standalone or co-processor
- Built-in TCP/IP Stack and Wi-Fi radio
- Ultra-CHEAP!
- <10 g .. drones anyone?

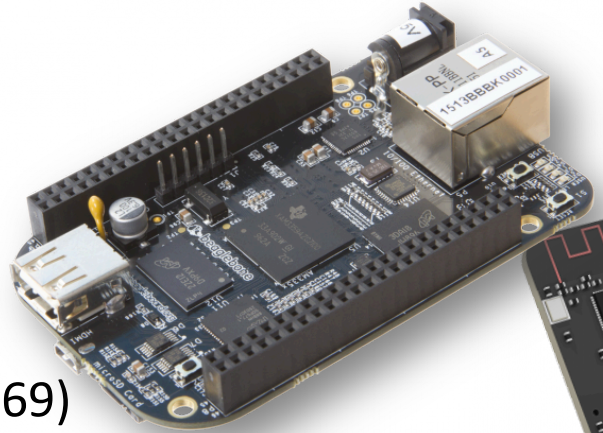
ESP8266 Shortfalls

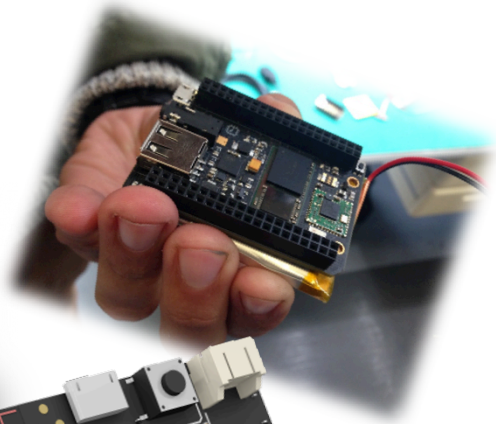
- Some libraries are not stable.
- C SDK needs improvements (2.0 is much better)
- TLS 1.2 now supported (as of SDK 2.0)
- Memory limitations vs Raspberry PI, Beaglebone, etc.
- No built-in digital-analog converter (DAC)
- But…
- It’s Ultra-CHEAP!

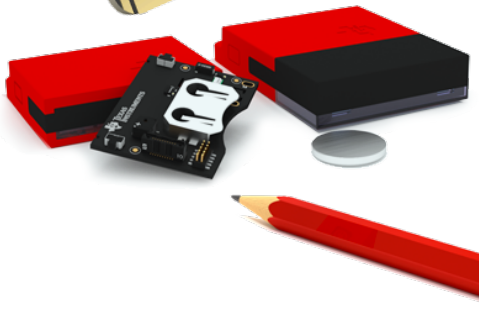


Other Options?

- C.H.I.P ($9)
- Raspberry PI ($5-25+)
- Beagle-bone ($50)
- Intel Edison ($50)
- BBC Micro Bit ($19)
- Particle Wi-Fi ($19) or GSM ($69)
- TI Sensortag ($30)
- ATT StarterKit (GSM) ($99)
- Much more (atmel, digispark, arduino)
  - https://www.hackster.io/platforms/microcontroller
- ESP32 ($10-$15)

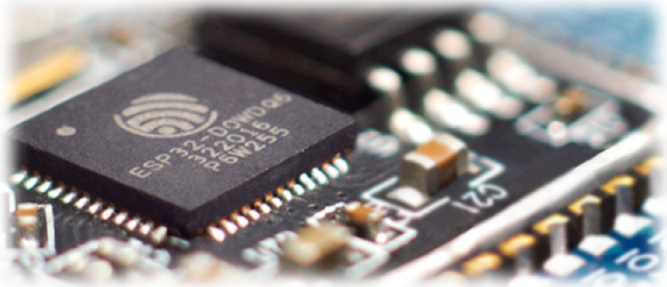






Espressif ESP32

- CPU: Xtensa Dual-Core 32-bit LX6 (160/240 MHz up to600 DMIPS)
- Memory: 520 KiB SRAM, 16 MB flash
- Wi-Fi: 802.11b/g/n , WFA, WPA/WPA2 and WAPI
- Bluetooth: v4.2 BR/EDR and BLE
- 12-bit ADC , 2 × 8-bit DAC
- 10 × capacitive touch sensors
- Temperature sensor



- 3 × SPI, 2 × I²S. 2 × I²C, 3 × UART
- IR (TX/RX), Motor PWM, LED PWM up to 16 channels, Hall effect sensor
- Ultra low power analog pre-amplifier
- Cryptographic HW acceleration: AES / SHA2 / Elliptical Curve Cryptography / RSA-4096
- https://espressif.com/en/products/hardware/esp32/overview

Serverless computing *

* Slides Courtesy of AWS

What is Serverless Computing?

- VMs
  - Machine as the unit of scale
  - Abstracts the hardware
- Containers
  - Application as the unit of scale
  - Abstracts the OS
- Serverless
  - Functions as the unit of scale
  - Abstracts the language runtime

Amazon EC2

Amazon ECS

AWS Lambda

How do I choose?

- VMs
  - "I want to configure machines, storage, networking, and my OS" **Amazon EC2**
- Containers
  - "I want to run servers, configure applications, and control scaling" **Amazon ECS**
- Serverless
  - "Run my code when it's needed" **AWS Lambda**

* Courtesy of AWS

AWS Lambda *

* Slides Courtesy of AWS

Microservices and AWS Lambda

AWS Lambda + Amazon API Gateway to create microservices

- **Event handlers** one function per event type
- **Serverless backends** one function per API / path
- **Data processing** one function per data type
- **Step Functions** Coordination + Visual Workflow

Lambda function usage via cloudwatch logs

```
START RequestId: b4bc7494-2bf5-11e6-80d9-d7aeea5a302f Version: $LATEST
2016-06-06T14:48:18.157Z b4bc7494-2bf5-11e6-80d9-d7aeea5a302f [ { I: '3013276', V: '3', T: '1465224409' },
{ I: '3013277', V: '1', T: '1465224409' },
{ I: '3013278', V: '1', T: '1465224409' },
{ I: '3013279', V: '1', T: '1465224409' },
{ I: '3013284', V: '1', T: '1465224409' },
{ I: '3013285', V: '4', T: '1465224409' },
{ I: '3013300', V: '1', T: '1465224409' },
{ I: '3013302', V: '5', T: '1465224409' },
{ I: '3013303', V: '5', T: '1465224409' },
{ I: '3013304', V: '3', T: '1465224409' },
{ I: '3013306', V: '3', T: '1465224409' },
{ I: '3013307', V: '1', T: '1465224409' } ]
2016-06-06T14:48:18.158Z b4bc7494-2bf5-11e6-80d9-d7aeea5a302f Putting items into DynamoDB: fau_bocaraton_ee96
2016-06-06T14:48:18.158Z b4bc7494-2bf5-11e6-80d9-d7aeea5a302f{ "RequestItems": { "fau_bocaraton_ee96": [ { "PutRequest": { "Item": { "instance": { "N": "3013276" }
2016-06-06T14:48:18.231Z b4bc7494-2bf5-11e6-80d9-d7aeea5a302f Success
END RequestId: b4bc7494-2bf5-11e6-80d9-d7aeea5a302f
REPORT RequestId: b4bc7494-2bf5-11e6-80d9-d7aeea5a302f Duration: 73.96 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 47 MB
```

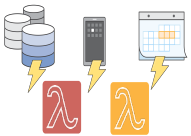
Using AWS Lambda

Bring your own code

- Node.js, Java, Python, C#
- Bring your own libraries

Simple resource model

- Select power rating from 128 MB to 1.5 GB
- CPU and network allocated proportionately
- Reports actual usage



Flexible use

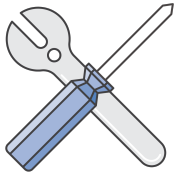
- Call or send events
- Integrated with other AWS services
- Build whole serverless ecosystems
- Step Functions (stateful)

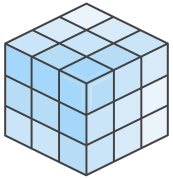
Flexible authorization

- Securely grant access to resources, including VPCs
- Fine-grained control over who can call your functions

Using AWS Lambda

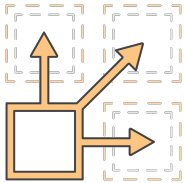
- **Programming model**
- AWS SDK built in (Python and Node.js …and now C#!)
- Eclipse plugin (Java)
- Lambda is the “webserver”





Stateless

- Persist data using Amazon DynamoDB, S3, or ElastiCache
- No affinity to infrastructure (can’t “log in to the box”)



Authoring functions

- Author directly using the console WYSIWYG editor
- Package code as a .zip and upload to Lambda or S3
- Plugins for Eclipse and Visual Studio
- Command line tools

Monitoring and logging

- Built-in metrics for requests, errors, latency, and throttles
- Built-in logs in Amazon CloudWatch Logs

How does it apply to IoT

- Securely process incoming data
- Redirect/Replicate/Archive
- Notify
- Feedback
- .
- .
- All without any server infrastructure using a pay-as-you-go infrastructure
- AT SCALE!

Lambda and IoT re:Invent 2016

- Lambda Step Functions
  - Coordination and Visual Workflow
- Lambda @Edge
  - Low latency at the CloudFront CDN endpoints
- AWS Greengrass
  - Run Lambda on your own devices

Less veggies, more ice-cream…

Code Examples *

* KISS – Keep It Simple, Silly!

Example LUA code for ESP8266

Connect to the wireless network

```
print(wifi.sta.getip())
--nil
wifi.setmode(wifi.STATION)
wifi.sta.config("SSID","password")
print(wifi.sta.getip())
--192.168.18.110
```

Arduino like IO access

```
pin = 1
gpio.mode(pin,gpio.OUTPUT)
gpio.write(pin,gpio.HIGH)
gpio.mode(pin,gpio.INPUT)
print(gpio.read(pin))
```

HTTP Client

```
-- A simple http client
conn=net.createConnection(net.TCP, false)
conn:on("receive", function(conn, pl) print(pl) end)
conn:connect(80,"121.41.33.127")
conn:send("GET / HTTP/1.1\r\nHost:
www.nodemcu.com\r\n"
  .."Connection: keep-alive\r\nAccept: */*\r\n\r\n")
```

HTTP Server

```
-- a simple http server
srv=net.createServer(net.TCP)
srv:listen(80,function(conn)
  conn:on("receive",function(conn,payload)
  print(payload)
  conn:send("<h1> Hello, NodeMCU.</h1>")
  end)
end)
```

Arduino Sketch

```
void setup() {
  // initialize serial communication at 115200 bits per second:
  Serial.begin(115200);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 3.2V):
  float voltage = sensorValue * (3.2 / 1023.0);
  // print out the value you read:
  Serial.println(voltage);
}
```

```
int inputPin = D3;  // pushbutton connected to digital pin D3
int val = 0;        // variable to store the read value

void setup() {
  pinMode(BUILTIN_LED, OUTPUT);  // set onboard LED as output
  pinMode(inputPin, INPUT);      // set pin as input
}

void loop() {
  val = digitalRead(inputPin);     // read the input pin
  digitalWrite(BUILTIN_LED, val);  // sets the LED to the button's value
}
```

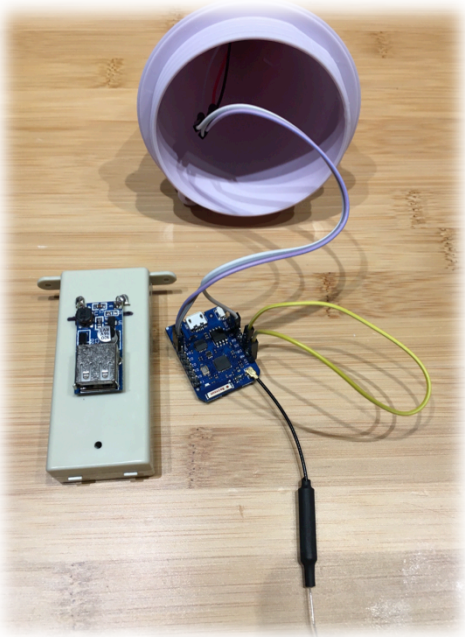
```
#include <ESP8266WiFi.h>

char wifi_ssid[] = "fausprint";
char wifi_pass[] = "owls2016";

void setup()
{
  Serial.begin(115200);
  Serial.println();
  Serial.print("Connecting to : ");
  Serial.println(wifi_ssid, wifi_pass);

  WiFi.mode(WIFI_STA);
  WiFi.begin(wifi_ssid, wifi_pass);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println();
  Serial.println("WiFi connected");
}


void print_ip_and_mac()
{
  uint8_t MA[6];
  char MC[18];

  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  WiFi.macAddress(MA);
  for (int i = 0; i < sizeof(MA); i++){
    sprintf(MC, "%02x:%02x:%02x:%02x:%02x:%02x",
      MA[0], MA[1], MA[2], MA[3], MA[4], MA[5]);
  }
  Serial.print("MAC Address : ");
  Serial.println(MC);
}

void loop ()
{
}
```

Demo Time!

1. Connecting to wifi (sketch & lua)
2. Scanning wifi networks (sketch & lua)
3. Turn a LED on/off (sketch & lua)
4. Temp & humidity using a DHT (sketch & lua)
5. OLED demo (lua)
6. DHT -> MQTT -> CloudMQTT & Freeboard.IO
7. Publish/Subscribe (bluemix + rickshaw)
8. Q/A webapp (javascript)
9. Surprise demo (based on re:Invent 2016) shortly..

But what about that Pool Thermometer !?

Modified Pool Thermometer



ESP8266 Hardware

Chip Information

- Espressif (Manufacturer) - https://espressif.com/en/products/hardware/esp8266ex/resources
- Wikipedia - https://en.wikipedia.org/wiki/ESP8266
- Kolban's book on ESP8266 - http://neilkolban.com/tech/esp8266/

Development Platforms

- Sparkfun (*Thing* board and kits) - https://www.sparkfun.com/products/13799
- Adafruit (*Huzzah* board and kits) - https://www.adafruit.com/products/2680
- Wemos (Mini D1 board and shields) - http://www.wemos.cc/
- Seedstudio (*Wio* Link or Node and Kits) - https://www.seeedstudio.com/Wio-Link-p-2604.html

Where to Buy

- Sparkfun, Adafruit
- Aliexpress
- Amazon
- eBay

Development Platforms

- NodeMCU (LUA) http://nodemcu.com/index_en.html
- ESP8266 Basic http://www.esp8266basic.com/
- MicroPython https://micropython.org/
- ESP8266 Arduino https://github.com/esp8266/Arduino
- ESP Basic https://www.esp8266basic.com/
- FreeRTOS https://github.com/espressif/ESP8266_RTOS_SDK
    - C Based Development
    - Build instructions - https://github.com/pfalcon/esp-open-sdk

Local Resources..

- Makerspaces
  - http://myhacklab.org/
  - http://www.makerflorida.org/find-a-hackerspace-or-makerspace/
- Meetups
  - Embedded Entrepreneurship (E2)
- Maker faires
  - https://makerfairepalmbeach.com/
  - https://makerfairemiami.com/

More Resources

- MQTT Example: http://www.esp8266.com/viewtopic.php?f=29&t=8746
- LUA Example: http://www.foobarflies.io/a-simple-connected-object-with-nodemcu-and-mqtt/
- NodeMCU Examples: https://github.com/ckuehnel/NodeMCU-applications
- Neopixel Example : http://www.instructables.com/id/ESP8266-controlling-Neopixel-LEDs-using-Arduino-ID/
- IBM Bluemix MQTT: https://www.ibm.com/developerworks/cloud/library/cl-mqtt-bluemix-iot-node-red-app/
- IBM Bluemix Viz: https://console.ng.bluemix.net/docs/services/IoT/visualizingdata_sample.html
- IBM Bluemix ESP8266:https://tuts.codingo.me/connect-esp8266-to-ibm-bluemix
- ATT IOT: https://starterkit.att.com/
- ATT IOT Library : https://github.com/attm2x/m2x-arduino
- ESPlorer: https://esp8266.ru/esplorer/
- ESP8266 Forum: http://www.esp8266.com/
- ESP8266 + TLS 1.2: https://github.com/aws/aws-iot-device-sdk-embedded-C/issues/7#issuecomment-171749256
- DC Motor: https://learn.adafruit.com/adafruit-arduino-lesson-13-dc-motors/transistors
- Thermistor : https://learn.adafruit.com/thermistor/using-a-thermistor
- Photoreistor : http://www.childs.be/blog/post/how-to-connect-a-photoresistor-or-light-dependant-resistor-to-an-esp8266-12e
- DS18B20 : http://www.tweaking4all.com/hardware/arduino/arduino-ds18b20-temperature-sensor/
- Deep Sleep : https://learn.sparkfun.com/tutorials/esp8266-thing-hookup-guide/example-sketch-goodnight-thing-sleep-mode
- Wifi Scan + RSSI : https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/examples/WiFiScan/WiFiScan.ino
- ESP8266 Arduino: http://esp8266.github.io/Arduino/versions/2.0.0/doc/libraries.html
- Browser MQTT: http://www.hivemq.com/demos/websocket-client/
- NodeMCU main : http://nodemcu.com/index_en.html
- NodeMCU docs : http://nodemcu.readthedocs.io/en/dev/
- Node Red http://nodered.org/
- OSX Getting Started: https://github.com/nodemcu/nodemcu-devkit/wiki/Getting-Started-on-OSX

Future Ideas

- AWS Quicksight or Azure Power BI for business intelligence and Visualization
- AWS Redshift or Hadoop for Analytics
- Machine Learning / Neural Networks
- Alexa (Voice UI), Image recognition, etc..

IoT Pool Thermometer 2.0

- Improved battery life
- Solar Powered Charging
- Controls via MQTT subscribe
- GSM instead of WiFi
- Lambda Notification
  - “your pool is ready Mr. stark”

Thank You
&
Enjoy the rest of your day!