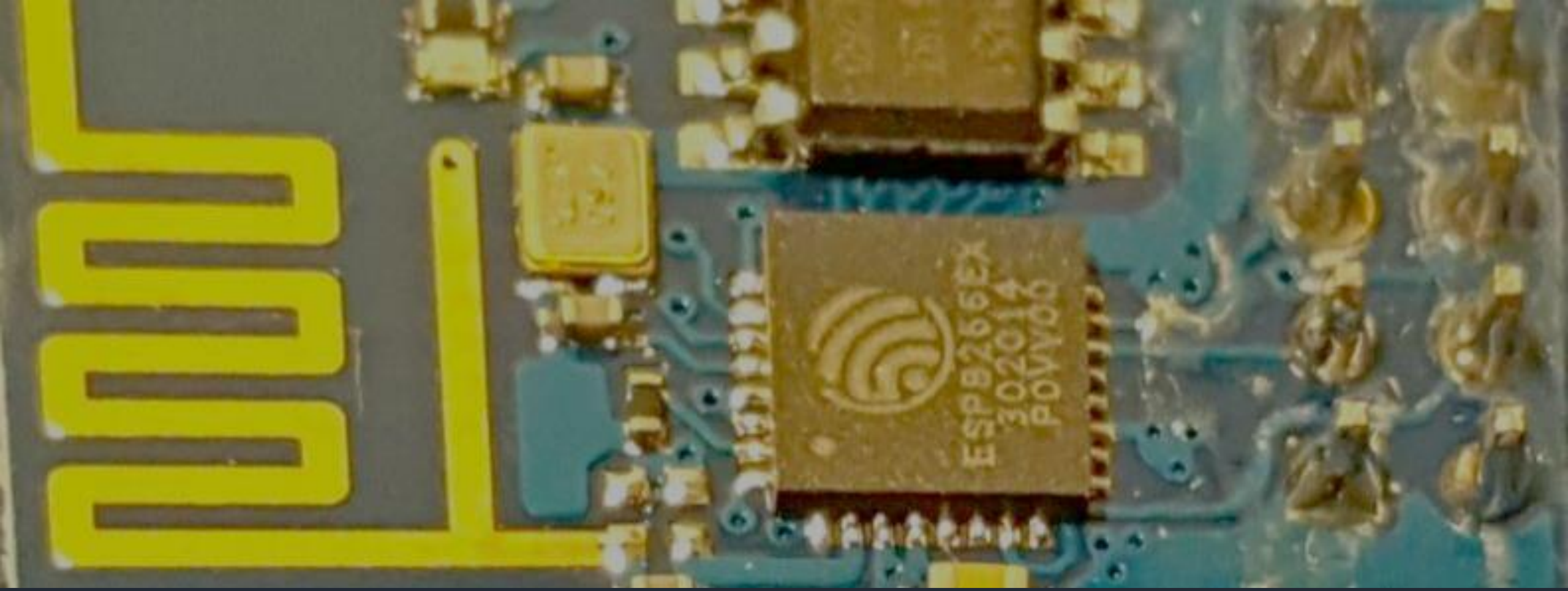
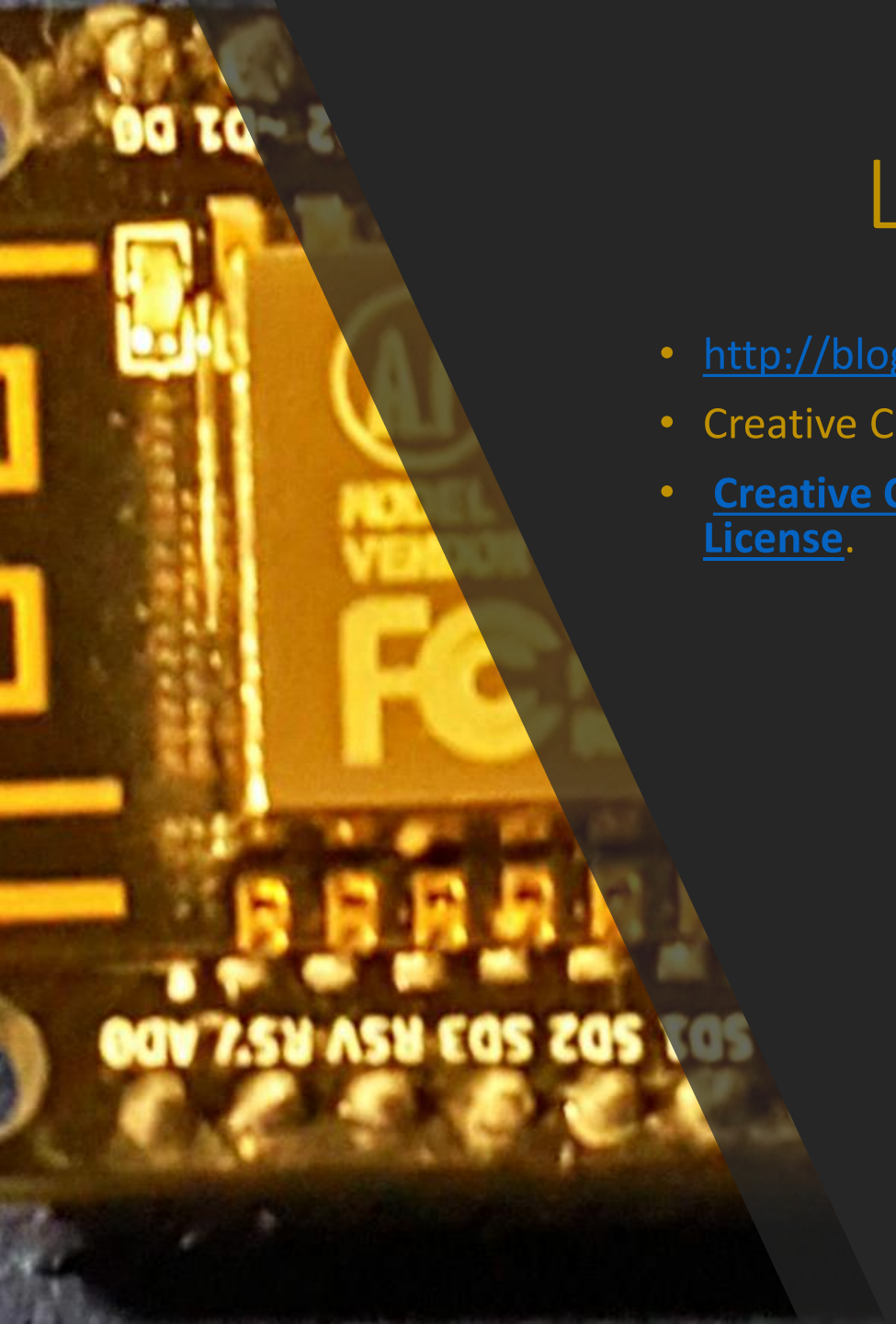


Getting started with ESP8266 and IOT

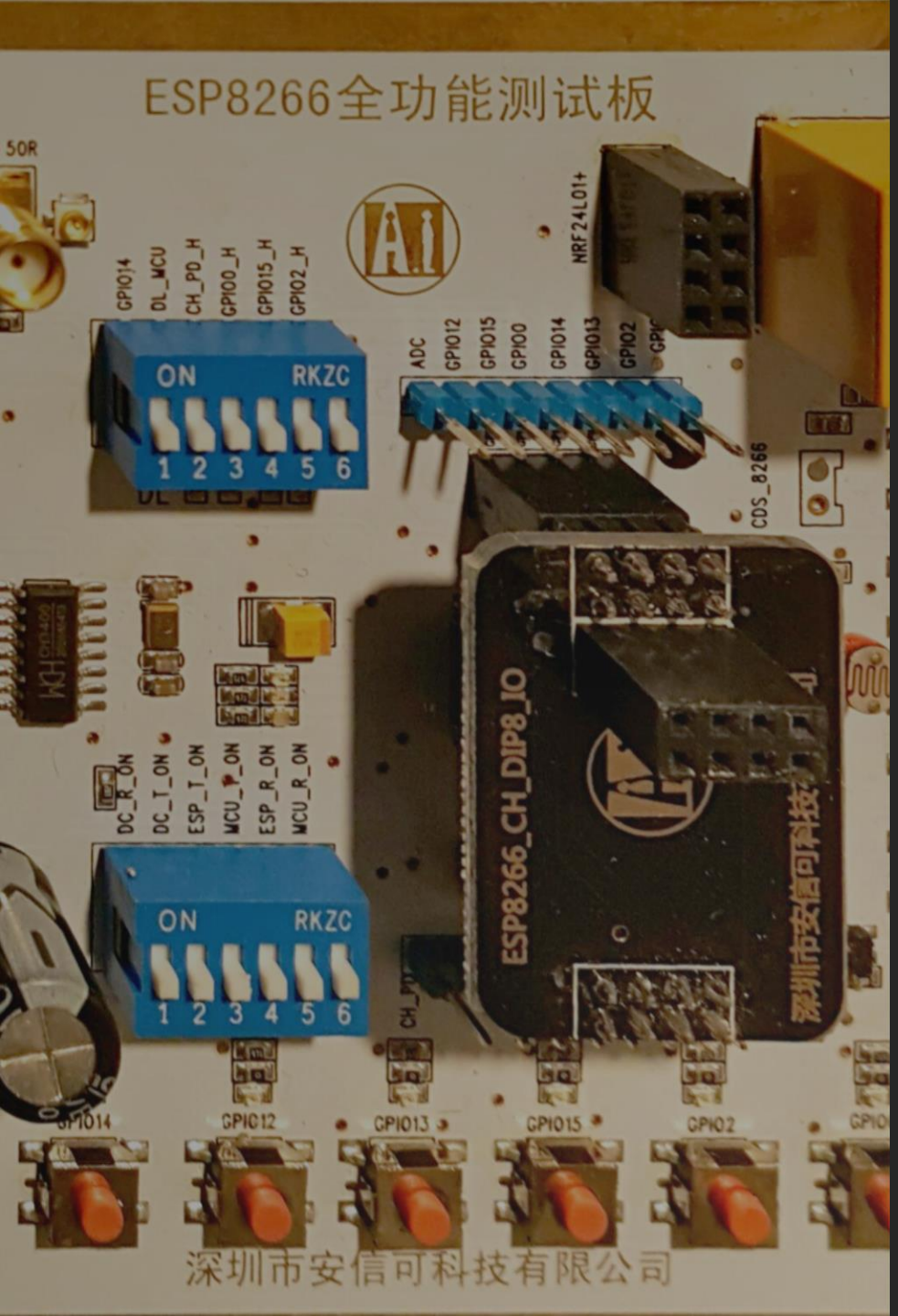
From a beginner Hobbyist to a beginner Hobbyist.

Ahmed Maklad



License

- http://blog.maksoft.ch/tutorials/
- Creative Commons with attribute to http://blog.Maksoft.ch/
- **Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License**.



Objectives

- From a beginner hobbyist to a beginner hobbyist.
- Quick start guide for programming ESP8266 and using it your Hobby projects.
- Share the ESP8266 experience.
- Focused on programming ESP8266 by Arduino IDE with Examples but also tell about other usage scenarios.
- What you need to start ? where to start ?
- Introduce terminology of Cloud and IOT.
- ESP8266 alternatives for Hobbyists.

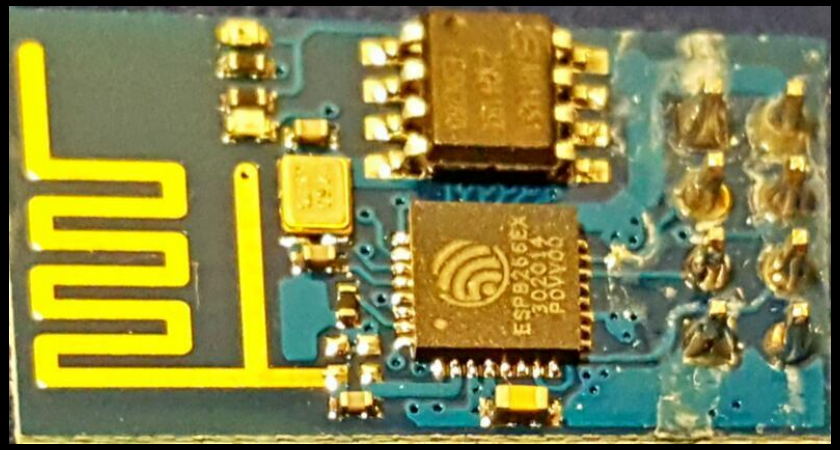
Agenda

- How hobbyists use the ESP8266 ?
  - As an Arduino WIFI Modem (original factory Firmware)
  - As a programmable interpreter (Lua, Basic, Python ..etc)
  - As and Arduino IDE programmable MCU.
- Hardware
  - What is it ?
  - Form Factors and NodeMCU
- Software
  - Cloud Products (thingsspeak , OpenHAB, …etc)
  - Arduino IDE Examples:
    - Wifi Station
    - Wifi AP
    - Web Client for NTP service.
    - WifiConfig library
- Other Options (competition)
- Other Web Resources.

What is an ESP8266 ?

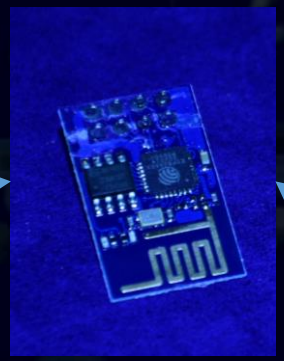
- A low cost WIFI chip with full TCP/IP stack and MCU.
- Produced by Shanghai-based Chinese manufacturer Espressif Systems.
- http://hackaday.com

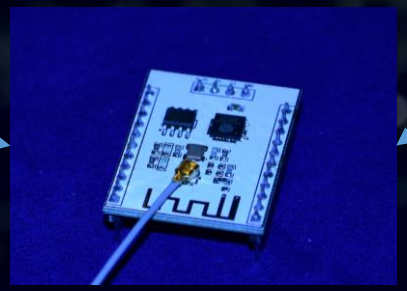
- AI Thinker Modules
- NodeMCU developer board.
- WiMos
- …

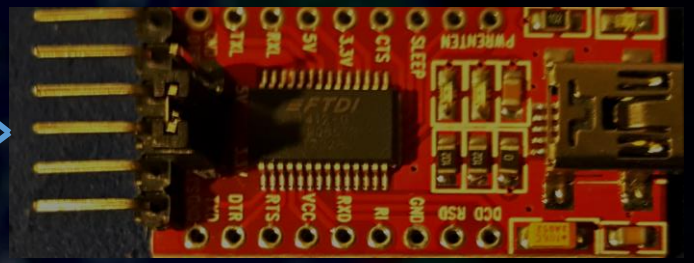


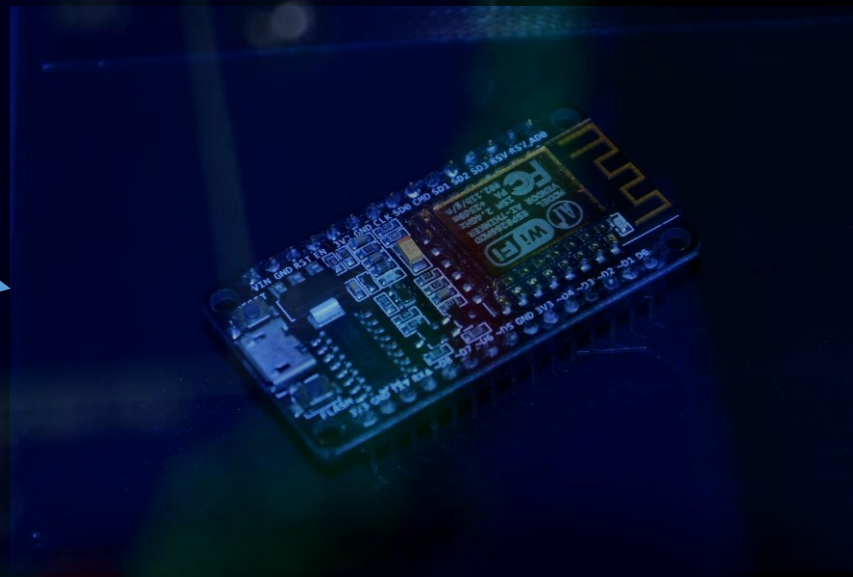
Form Factors

- ESP8266-01
- ESP8266-XX
- ESP8266-12
- ESP8266-12E
- NodeMCU
- Sonoff (commercial product)
  - A/C ~10Amps Relay
  - Programmable pins (almost) exposed



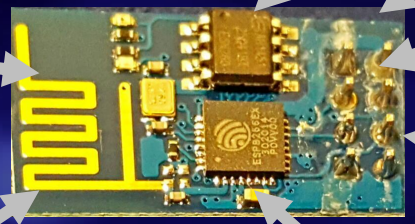






How to use the ESP8266 ? (use cases)

Arduino WIFI Modem Factory Firmware (AT Commands.)



Lua Programming

Basic Programming

JavaScrip Programming

MicroPython Programming

Arduino IDE

#include <ESP8266WiFi.h>

Arduino Libraries & Eco System

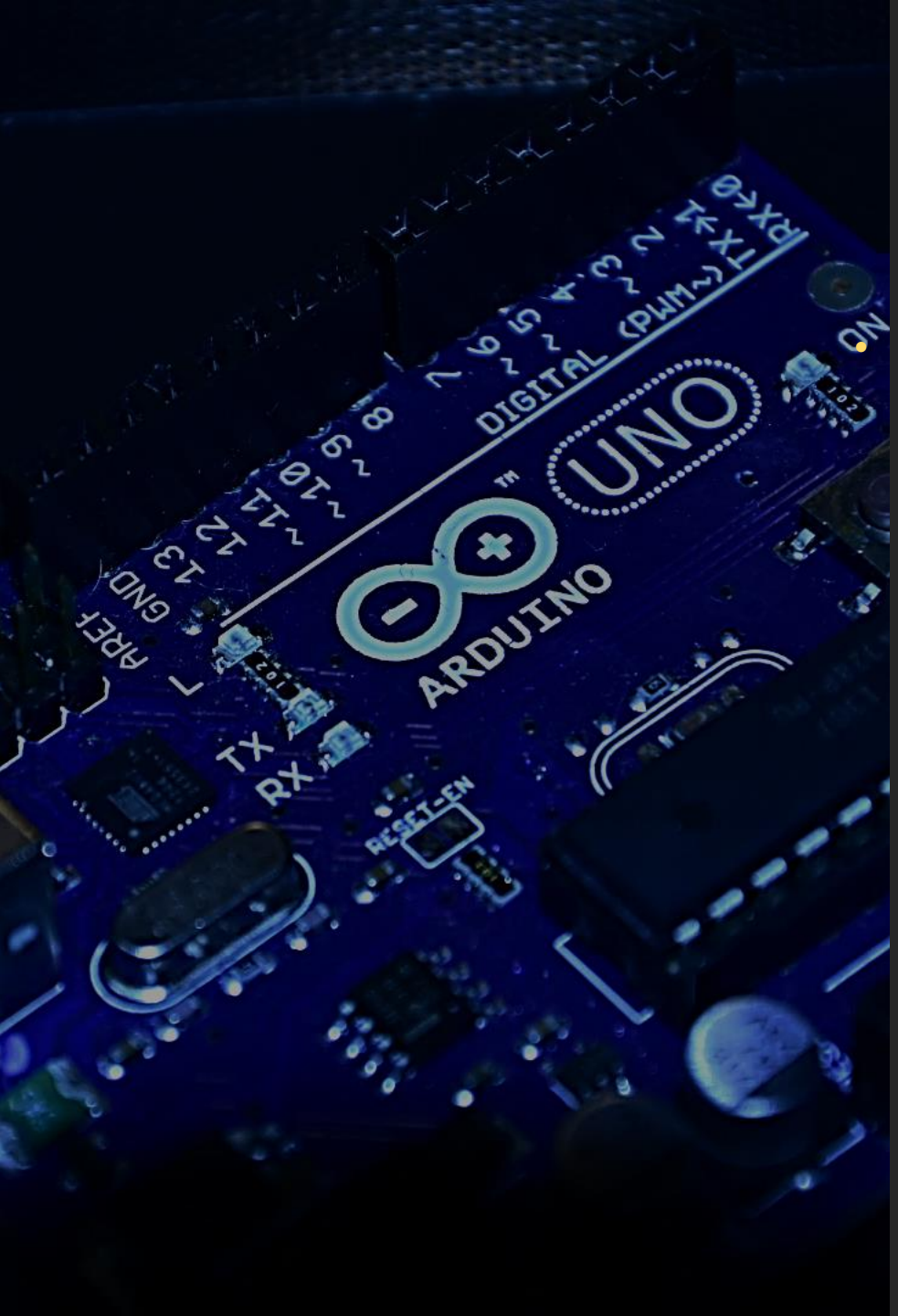
EasyESP Firmware (no programming)

Flashing the ESP8266 (The Firmware)

- The Firmware flashing tools:
  - ESP8266Flasher.exe, you could upload multiple Firmwares: https://github.com/nodemcu/nodemcu-flasher
  - Espressif Flash download tool: http://bbs.espressif.com/viewtopic.php?f=57&t=433
  - Esptool.py

    https://github.com/espressif/esptool

- The available Firmwares:
  - Flash the Lua Firmware https://github.com/nodemcu/nodemcu-firmware
  - Basic programming Firmare https://www.esp8266basic.com/
  - The microPython firmware https://docs.micropython.org/en/latest/esp8266/esp8266/tutorial/intro.html
  - The original Espressif Firmware : http://bbs.espressif.com/download/file.php?id=991



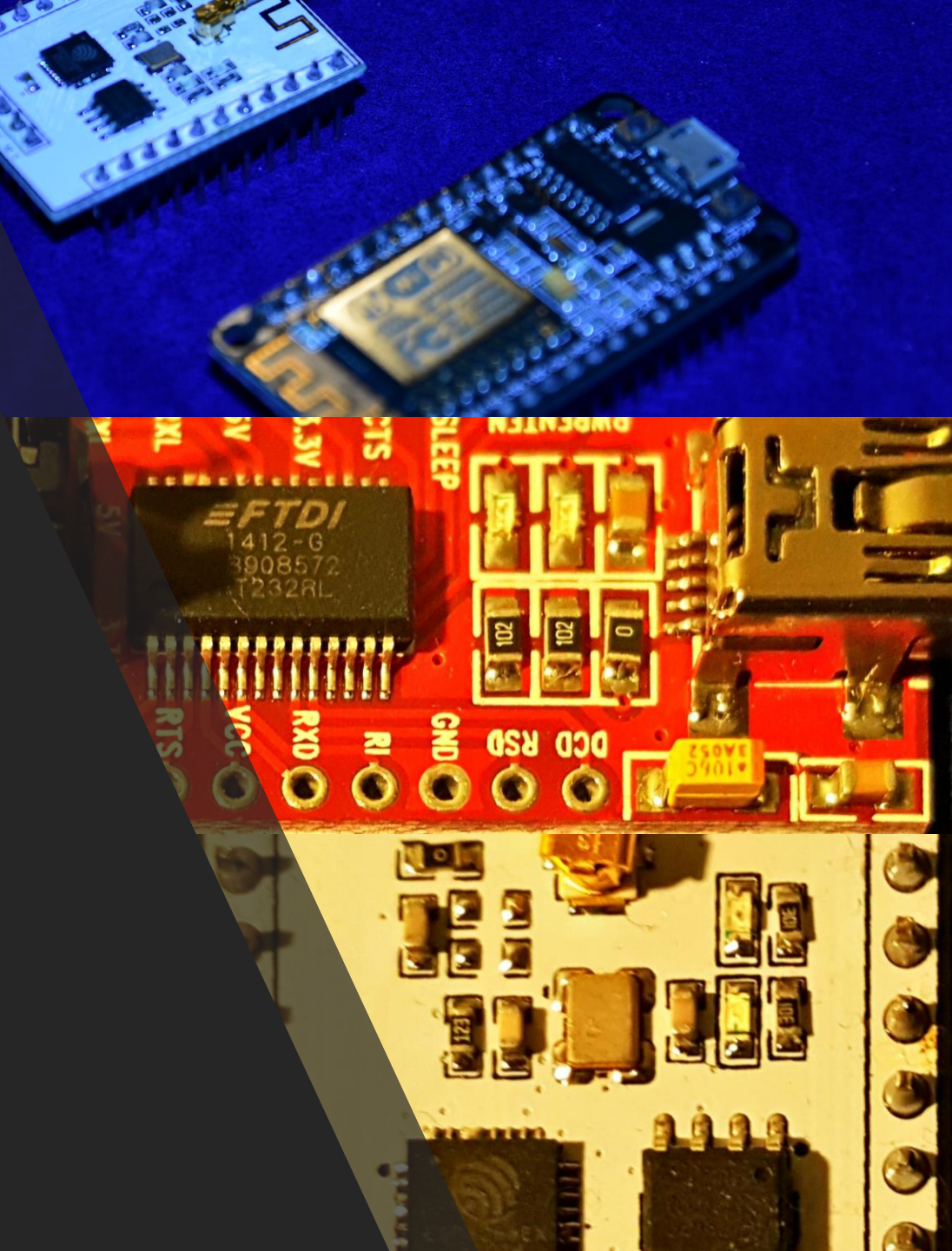
Uses of the ESP8266 (1)

As a WIFI modem for Arduino

- Connect the ESP8266 RX and TX to the Arduino TX and RX Serial pins respectively , GND and VCC to 3.3v
- Send AT Commands and respond to them via the serial port.
- Original Firmware on the ESP8266-01
- Other "AT" enabled Firmware available to add more features like NTP (time Query)
- Some projects use the Esp8266 as replacement for RTC (by NTP).
- A perfect Cheap WIFI shield for Arduino.
- ESP8266 Adafruite Library.
- Example AT command:
  - AT+CWMODE=2 *sets the ESP8266 in Access Point Mode*
  - AT+CWMODE *queries the device which mode is it in now STA=1,AP=2,both=3*

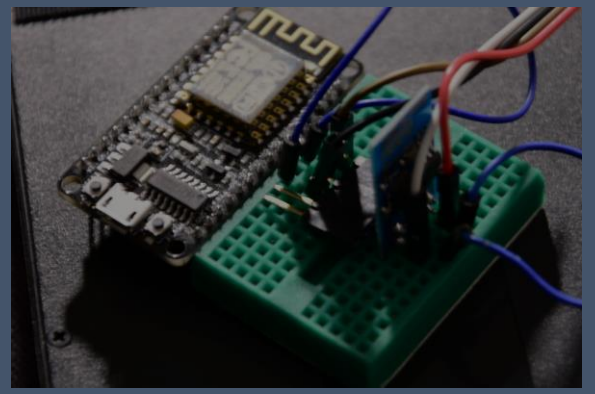
Uses of the ESP8266 (2)

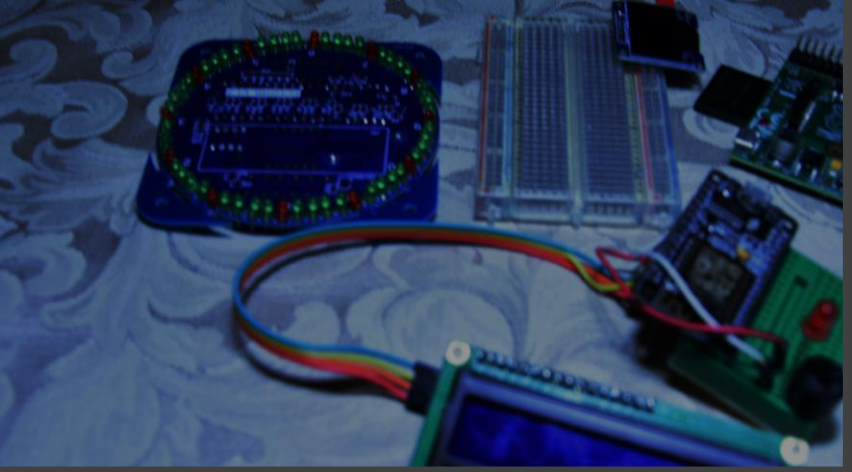
- Using the Basic , Lua or MicroPython firmware
- "Esplorer" and "Lua Uploader" tools to upload the LUA code
- Each implementation has its own eco-system of libraries.
- You Burn the Firmware which is some kind of an Embedded interpreter and the text source code is uploaded using the tools mentioned above.

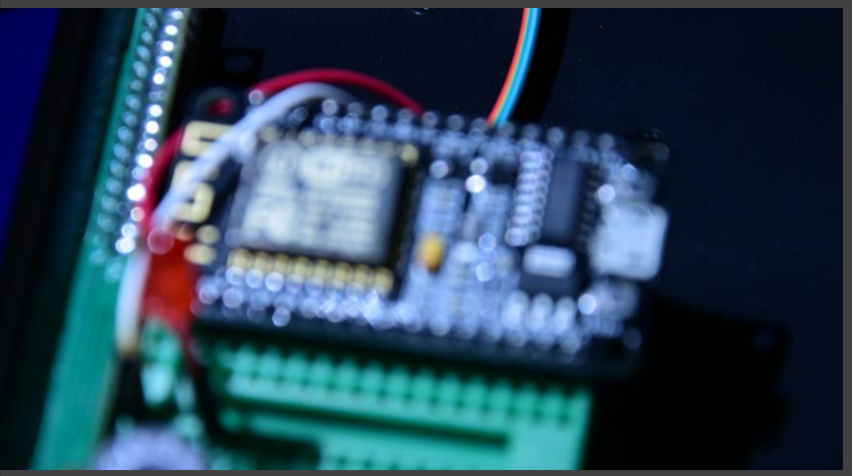


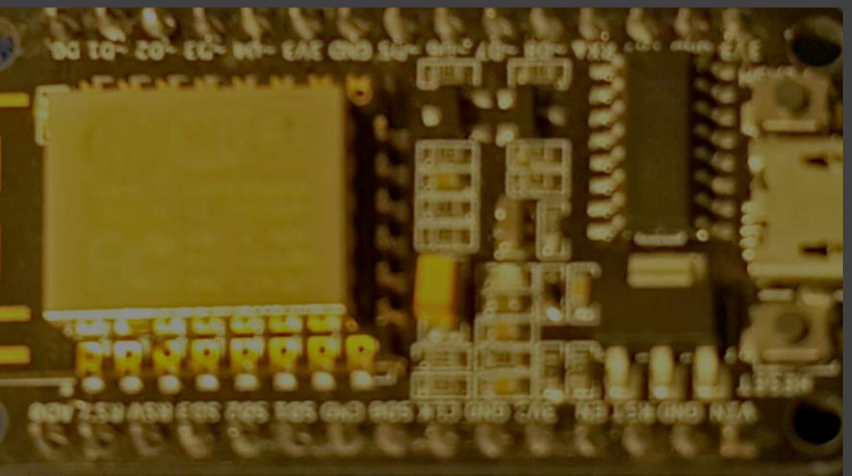
Uses of the ESP8266 (3)

- Programming by Arduino IDE & NodeMCU
  - Install CH340 driver
  - Install the Arduino IDE 1.6.8+
  - Add the ESP8266 repository (https://github.com/esp8266/Arduino)
  - Tools --> Board menu -> Board Manager
  - Library Manager
  - Explore the Library Examples









Best Setup for beginners (Your first shopping list)

- NodeMCU (including the USB interface) 12E with 4MB+ RAM (critical later for OTA) .
- Arduino IDE (use most the already existing Arduino Libraries).
- Breadboard and some M-F, F-F, M-M jumper wires.
- Some LED's and a buzzer .
- A couple of Sensors Like DHT11 or the more accurate DHT22. (best when they are I2C or SPI interfaced)
- An LCD 2004 or 1602 with I2C or SPI interface for saving PINs
- If you insist on having the ESP-01 , remember that you will need an FTDI USB-Serial port module also.
- Spoil yourself with NodeMCU motor shield if you want to build your own WiFi car later.

The <ESP8266WiFi.h> library

- #include <ESP8266WiFi.h>
- The following object is instantiated:
  - **WiFi (multiple Inheritance)**
    - .begin(ssid, password,…);
    - .status()
    - .localIP()
    - config(localIP,gateway,subnet,dns1,dns2)
    - isConnected()
    - RSSI()
    - beginWPSconfig();
    - ..

Has the following Classes:

WiFiServer

server(80)

.begin();

WiFiClient client = server.available()

.available()

.flush()

.readStringUntil('\r')

.println/print(String)

WiFiClient (connect to a web server)

.connect(server,80)

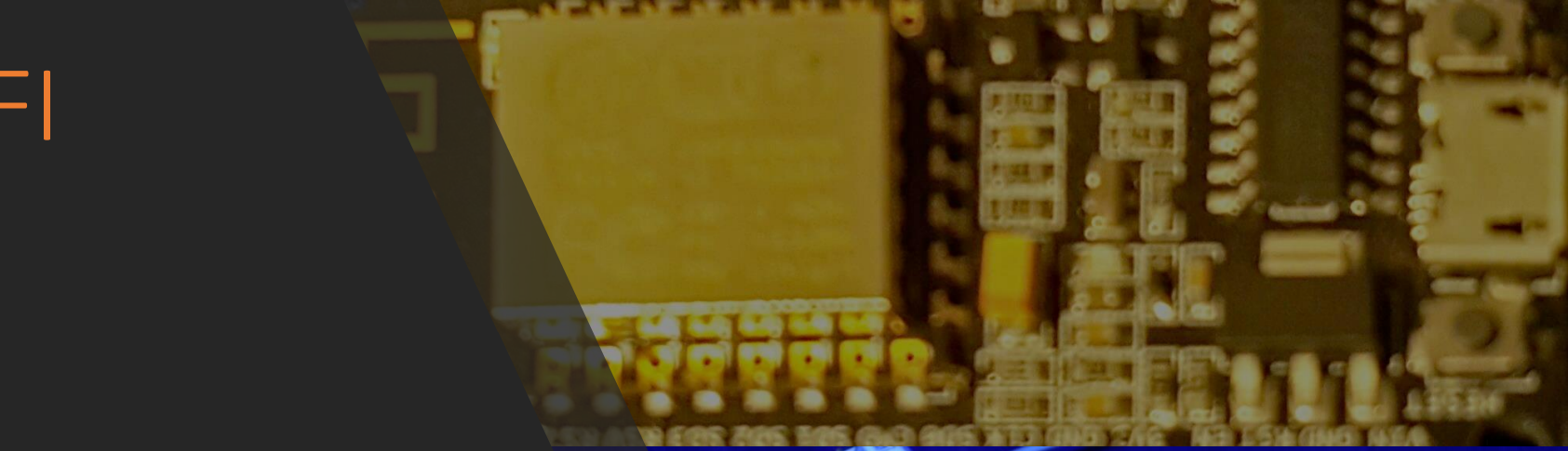
.print(String)

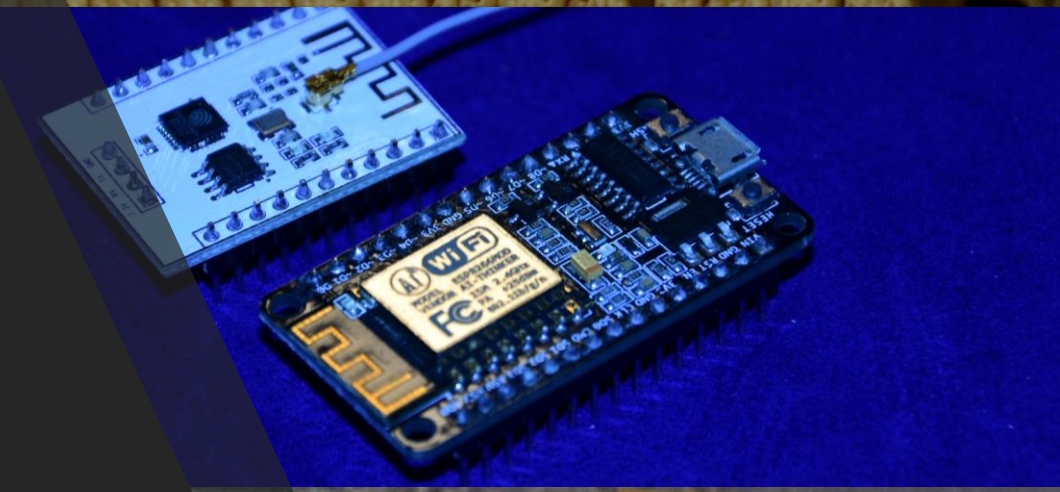
.stop()

.remoteIP() .remotePort()

Connect as a WIFI Station

```
#include <ESP8266WiFi.h>
    /*you could define “ssid” and “password” in .h file
    in your ArduinoIDE/libraires folder to avoid having
    to write them every time in your programs*/
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
}
Serial.println(WiFi.localIP());
```





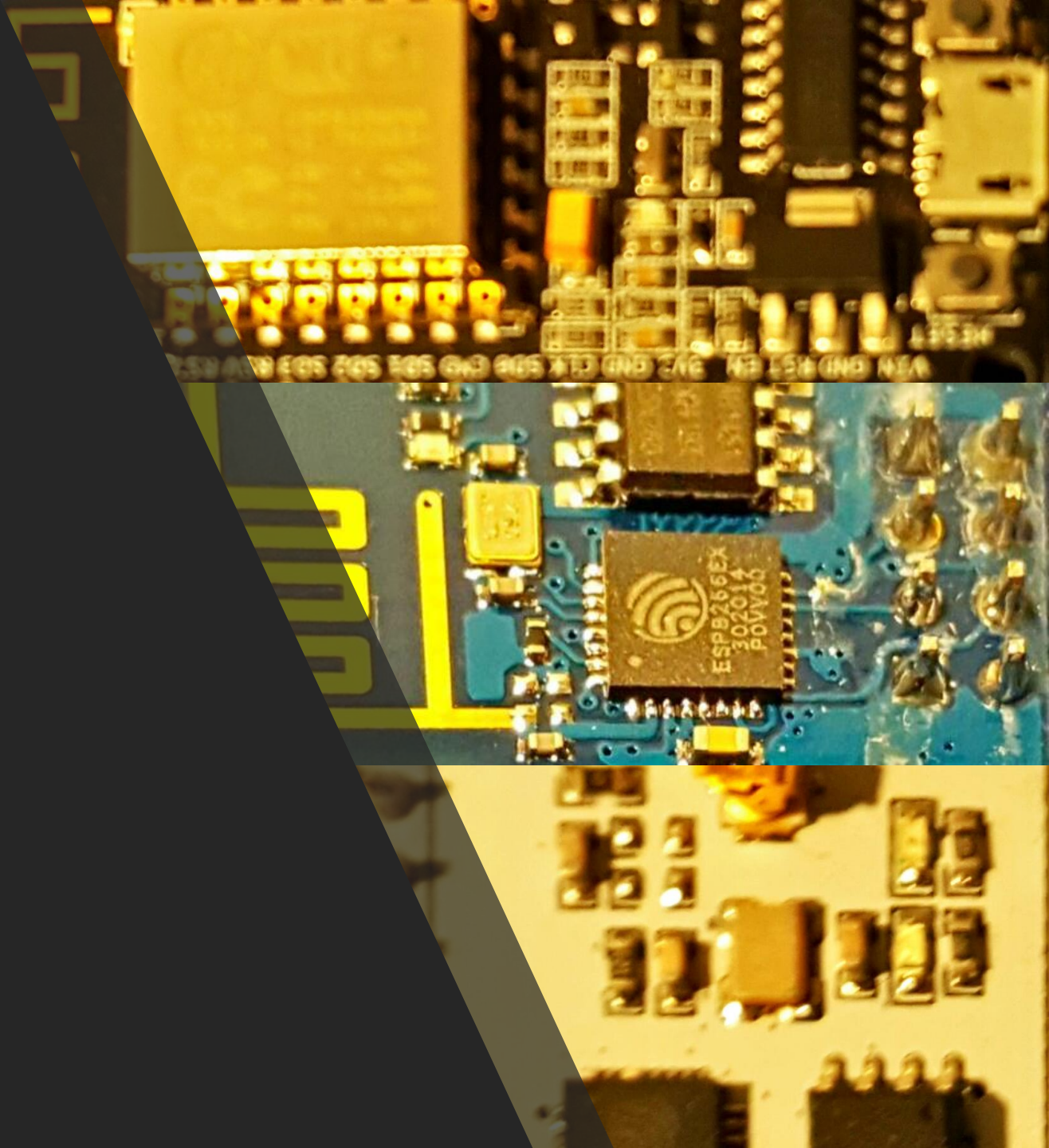
As an HTTP server

- setup()

```
server.begin();
```

- Loop()

```
// Check is some client connected
WiFiClient client = server.available();
if (!client) {
        return;
}
// wait till client sends in some data
Serial.println("new client");
while(!client.available()){
        delay(1);
}
// Read the GET/ POST request
String request = client.readStringUntil('\r');
Serial.println(request);
client.flush();
```



Create an Access Point (AP)

```
#include <ESP8266WiFi.h>

void setup()

WiFi.mode(WIFI_AP);
WiFi.softAP(ap_name,psk_key);
server.begin();
```

```
void loop()

// Check if a client has connected
  WiFiClient client = server.available();
  if (!client) {    return;  }
// Read the first line of the request
  String req = client.readStringUntil('\r');
  Serial.println(req);
  client.flush();
// process the “req” string
// Send the response to the client
  client.print(response_str);
```

Connect to the web

```
/*
https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/examples/NTPClient/NTPClient.ino
sketch -> examples --> ESP8266/NTPClient/NTPClient.ino
*/
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
unsigned int localPort = 2390;      // local port to listen for UDP packets
//IPAddress timeServer(129, 6, 15, 28); // time.nist.gov NTP server
IPAddress timeServerIP; // time.nist.gov NTP server address
const char* ntpServerName = "time.nist.gov";
// A UDP instance to let us send and receive packets over UDP
WiFiUDP udp;
setup()
                    wifi.begin and status checking loop
                    //Starting UDP
                      udp.begin(localPort);
```

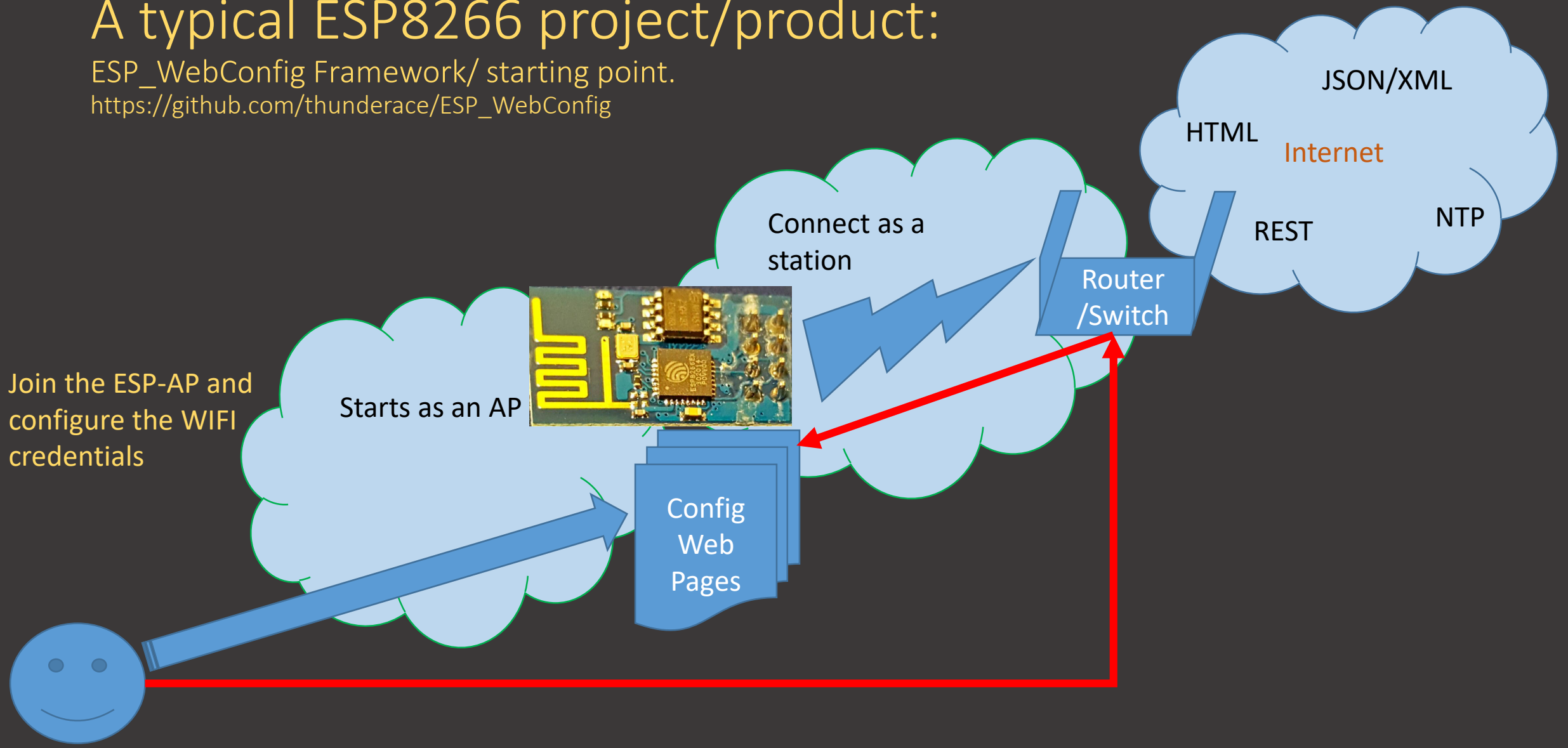
```
void loop()

WiFi.hostByName("time.nist.gov", timeServerIP);
sendNTPpacket(timeServerIP);  {
                udp.beginPacket(address, 123); //NTP requests are to port 123
                 udp.write(packetBuffer, NTP_PACKET_SIZE);
                 udp.endPacket();
 int cb = udp.parsePacket();
If (cb!=null)
     udp.read(packetBuffer, NTP_PACKET_SIZE); // read the packet into the buffer
     // calculations for epoc and UTC time
```

A typical ESP8266 project/product:

ESP_WebConfig Framework/ starting point.

https://github.com/thunderace/ESP_WebConfig

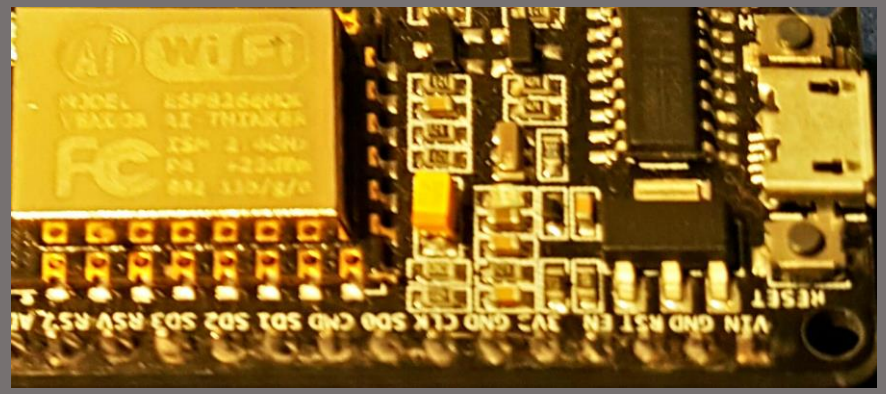


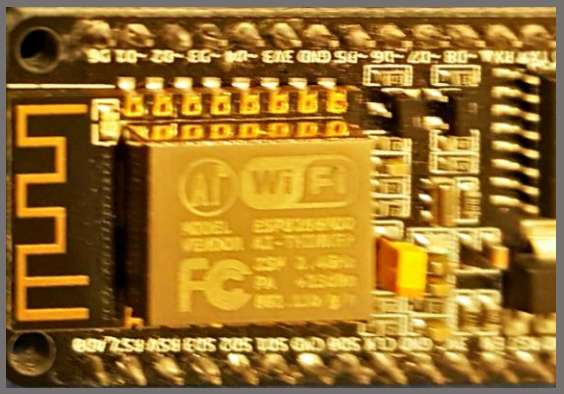
A peek into IOT

Just an introduction

IOT

- Internet of Things
- Things network through protocols (Arduino eco system).
  - Wifi (esp8266)
  - BT (hc-05, hc-06)
  - RF 2.4 GHz (nRF24L01)
  - ..etc
- Machine-Human
- Machine-Machine M2M
- Society 4.0
- Industry 4.0
- 4th Industrial Revolution (steam –Electric-Electronic- IOT)

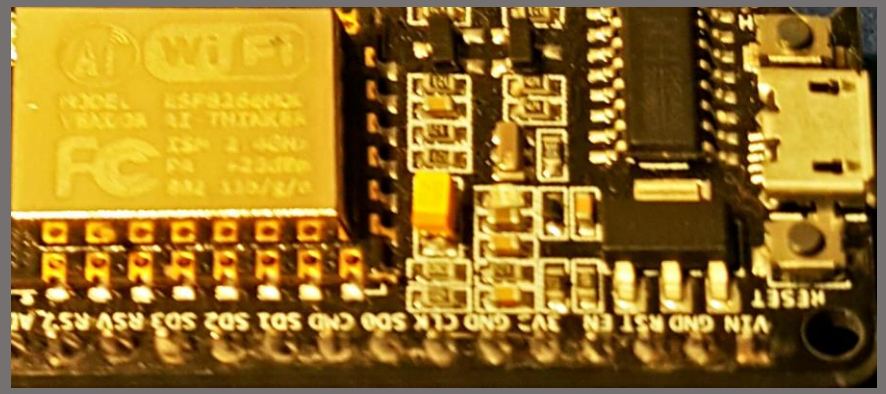


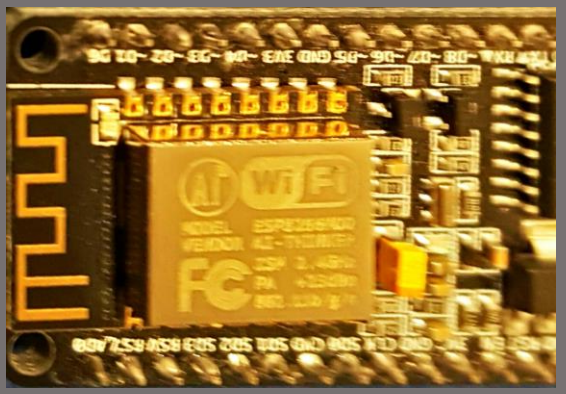


Eclipse Three stacks of IOT

- https://iot.eclipse.org/resources/white-papers/Eclipse%20IoT%20White%20Paper%20-%20The%20Three%20Software%20Stacks%20Required%20for%20IoT%20Architectures.pdf

- Constrained Device
  - MCU / CPU
  - Sensors
  - Focused devices
  - Limited Memory, Processing, communication ..etc
  - OS /RTOS
- Gateway
  - Aggregation of Sensors.
  - OS
  - Data management
  - Messaging
- IOT Cloud Platform
  - Data Management (DB)
  - Event management / Analytics
  - Security /device Registry
  - UI

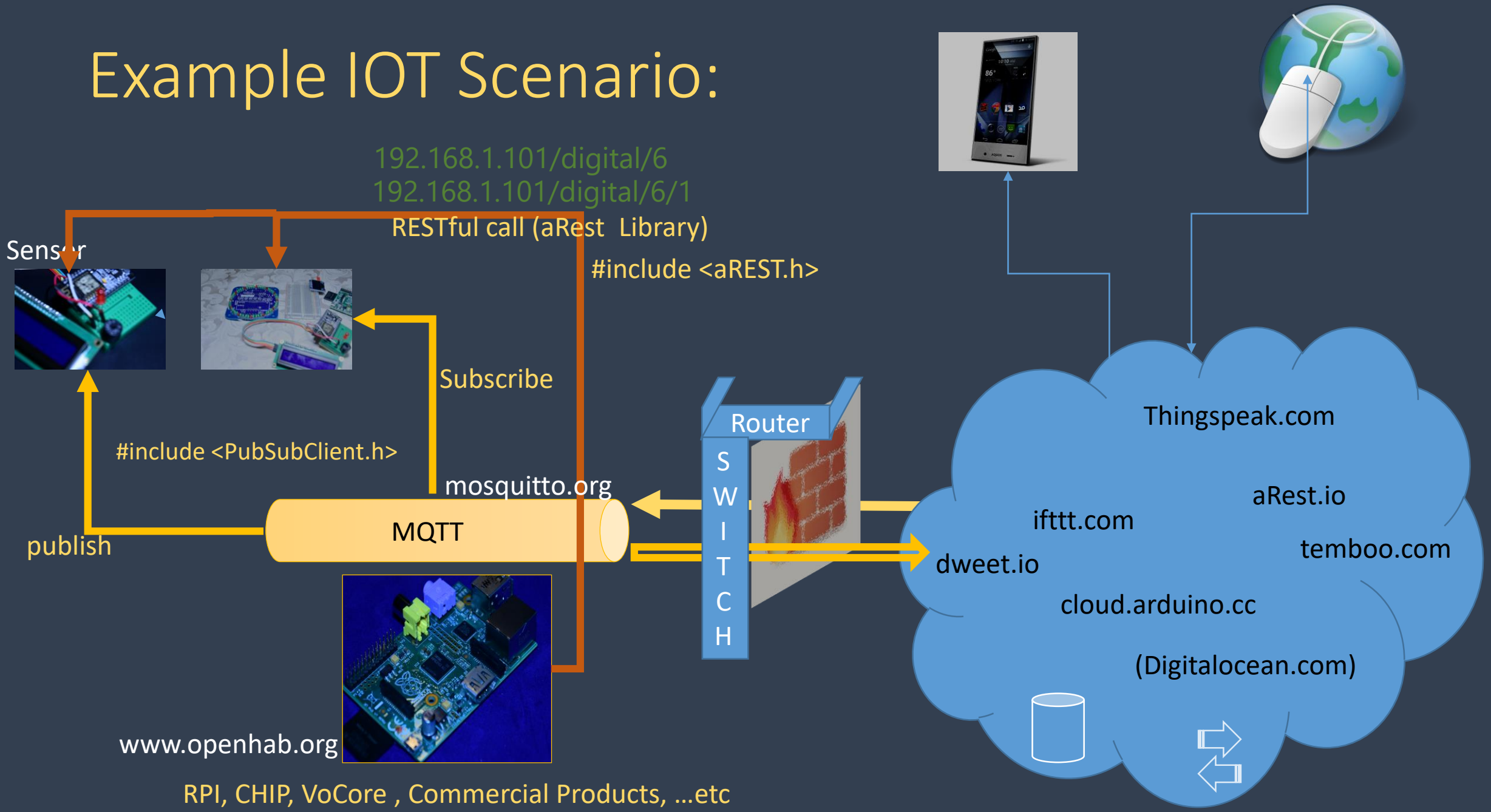


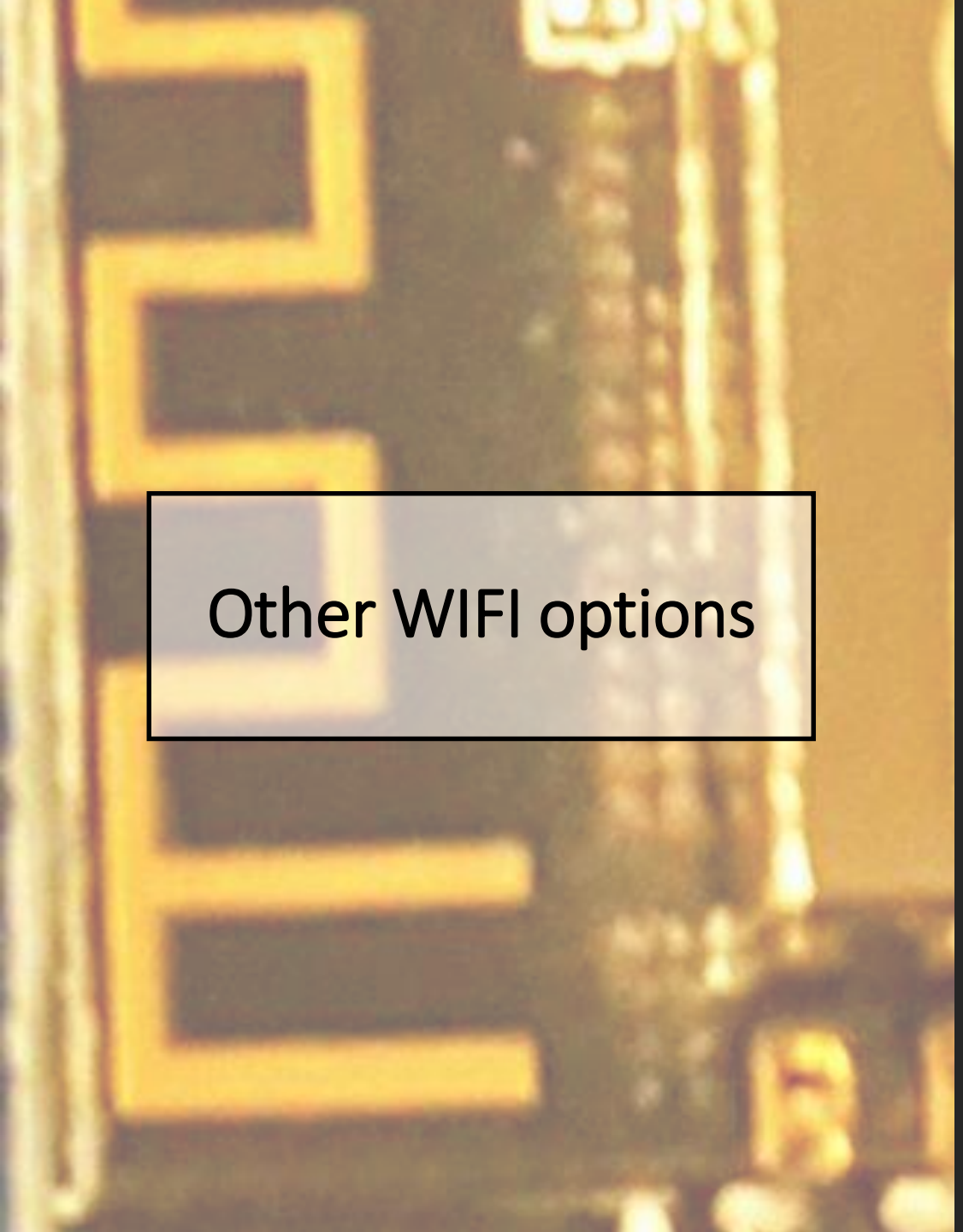


Why do we need the Cloud with IOT ?

- Heavy Lifting Processing
  - Image Processing
  - Speech Recognition
  - Knowledge database (example: FarmBot.org).
  - Machine Learning
  - Prediction , projection.
  - Large JSON/XML/Response parsing and reduction
- More Connectivity channels (M-M , M-H) like email, SMS, twitter, facebook, Encryption, app-notification ..etc
- Distributed (example: Car approaching, house automation, Garage door ..etc)

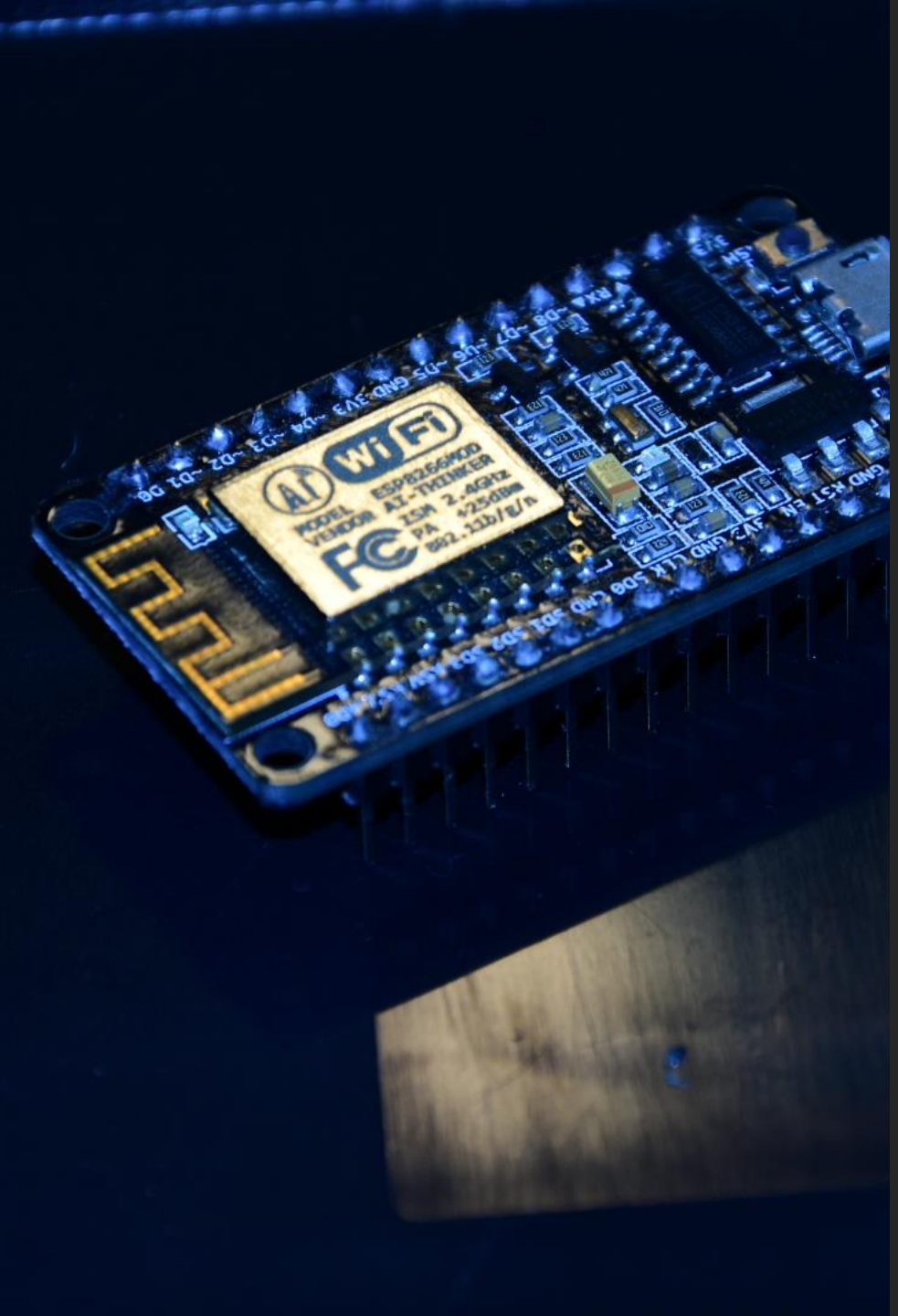
Example IOT Scenario:





Other WIFI options

- ESP8285 / ESP8255
- ESP32
- Realtek RTL8710
- VoCore
- Arduino Yun
- Chip



References and tools

- Andreas Spiess Youtube channel
- Internet of Things with ESP8266 by Marco Schwartz.
- ESP8266 Weather station by Daniel Eichhorn.
- NodeMCU dev kit using Arduino IDE by Magesh Jayakumar.
- http://www.arduinesp.com/ is a good place to look for examples.

- Office lens App@Android and Nikon DSLR to create the Background pictures.