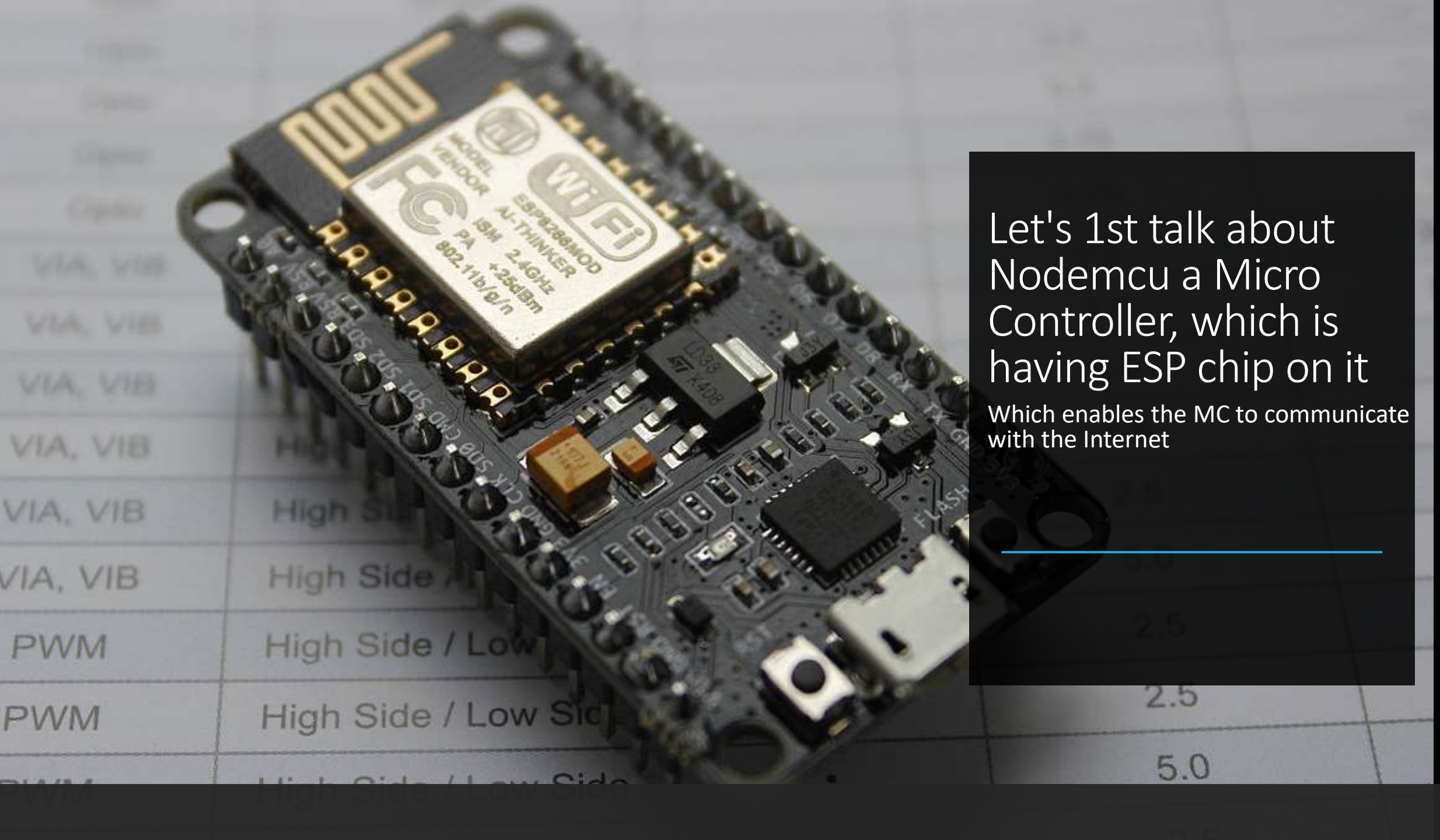
Communication protocols-II

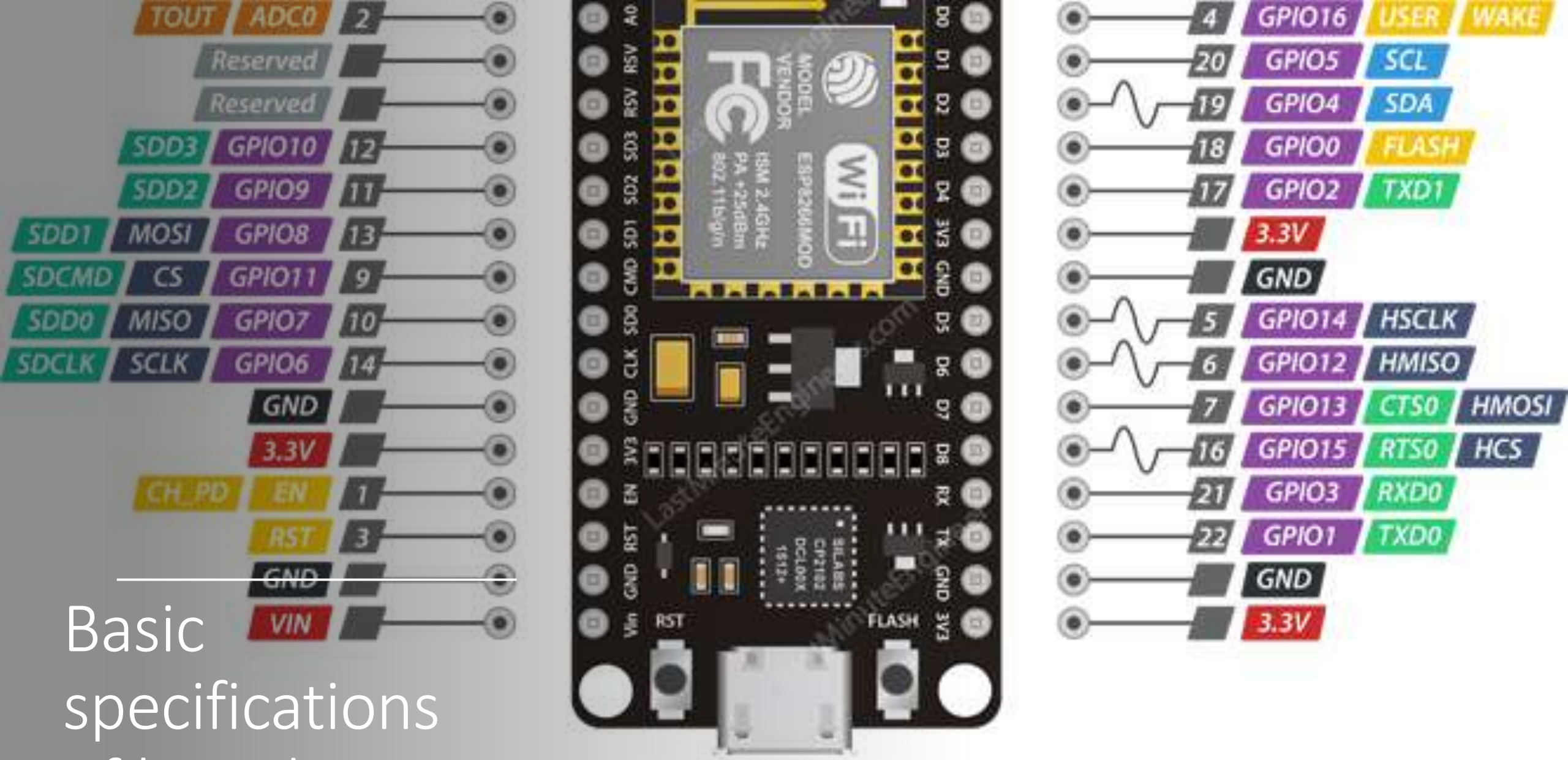
HOW DATA MOVES ON WEB

In robotics if we are talking about the IoT then we should know how Internet is accessible to the robots and 1st we should how internet works



Let's 1st talk about Nodemcu a Micro Controller, which is having ESP chip on it

Which enables the MC to communicate with the Internet



Basic specifications of board

- ADC channel – A 10-bit ADC channel.
- UART interface – UART interface is used to load code serially.
- PWM outputs – PWM pins for dimming LEDs or controlling motors.
- SPI, I2C & I2S interface – SPI and I2C interface to hook up all sorts of sensors and peripherals.
- I2S interface – I2S interface if you want to add sound to your project

GPIO Pins ESP8266 NodeMCU has 17 GPIO pins which can be assigned to various functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically

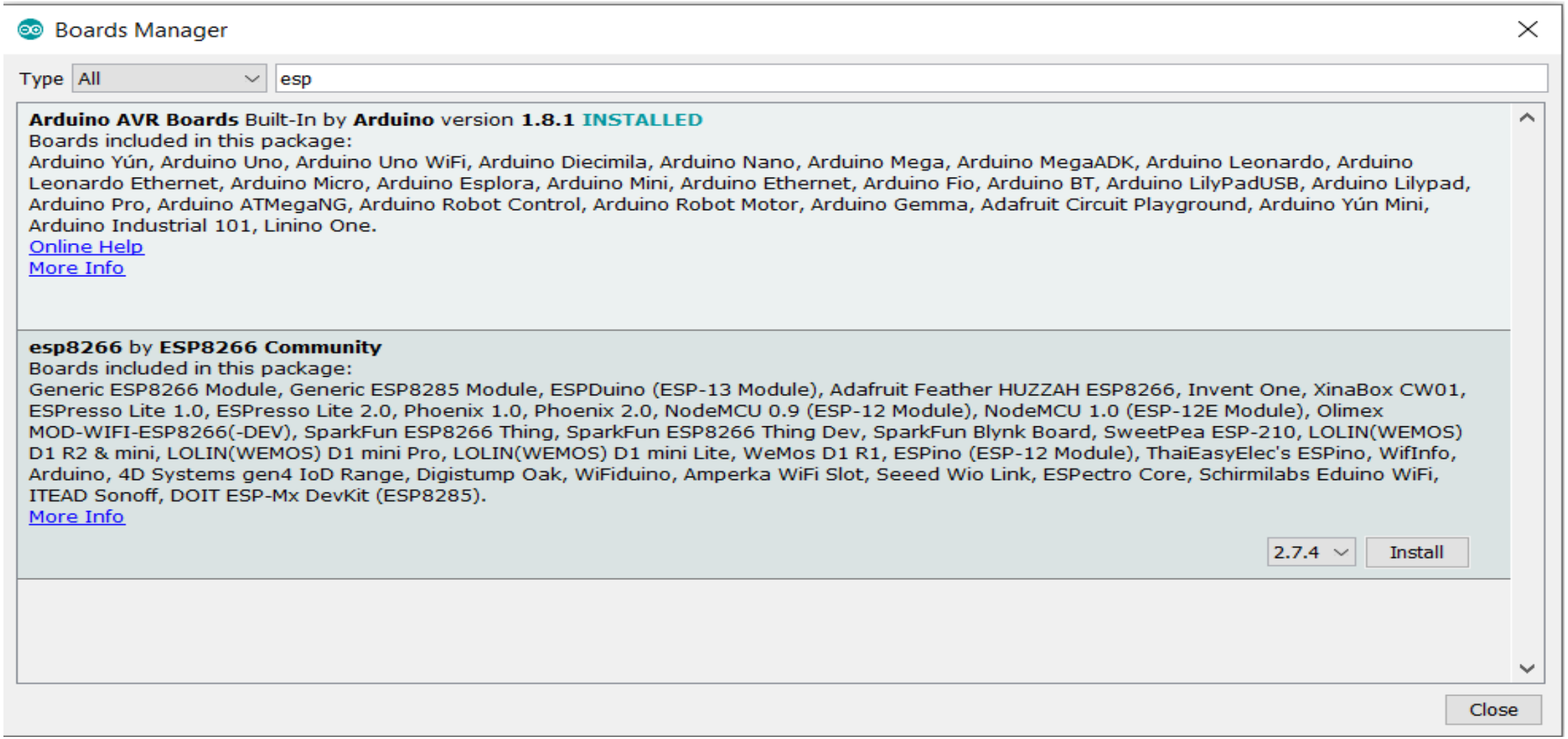
Power Pins There are four power pins viz. one VIN pin & three 3.3V pins. The VIN pin can be used to directly supply the ESP8266 and its peripherals, if you have a regulated 5V voltage source. The 3.3V pins are the output of an on-board voltage regulator. These pins can be used to supply power to external components.

I2C Pins are used to hook up all sorts of I2C sensors and peripherals in your project. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

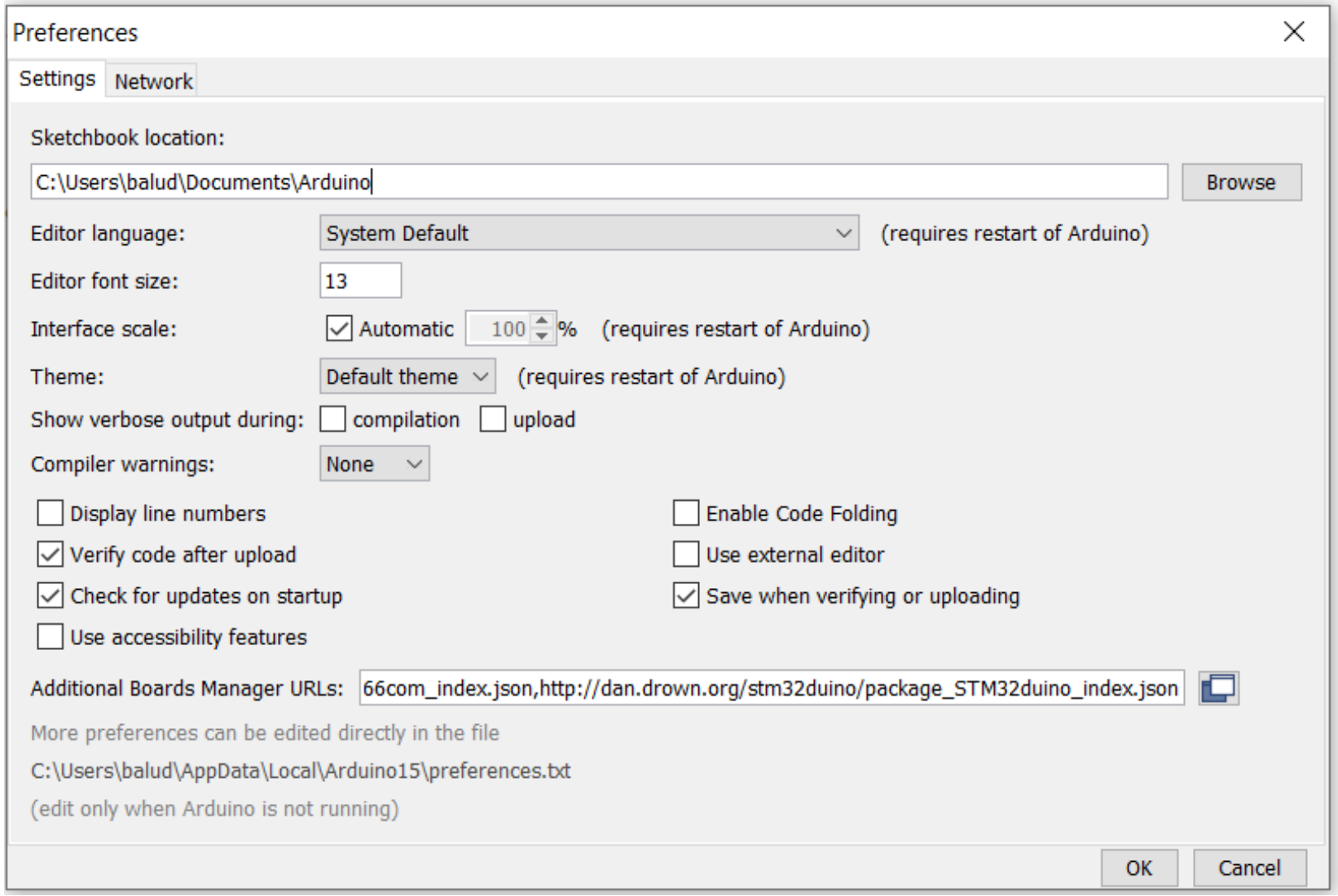
UART Pins ESP8266 NodeMCU has 2 UART interfaces, i.e. UART0 and UART1, which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. It supports fluid control. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.

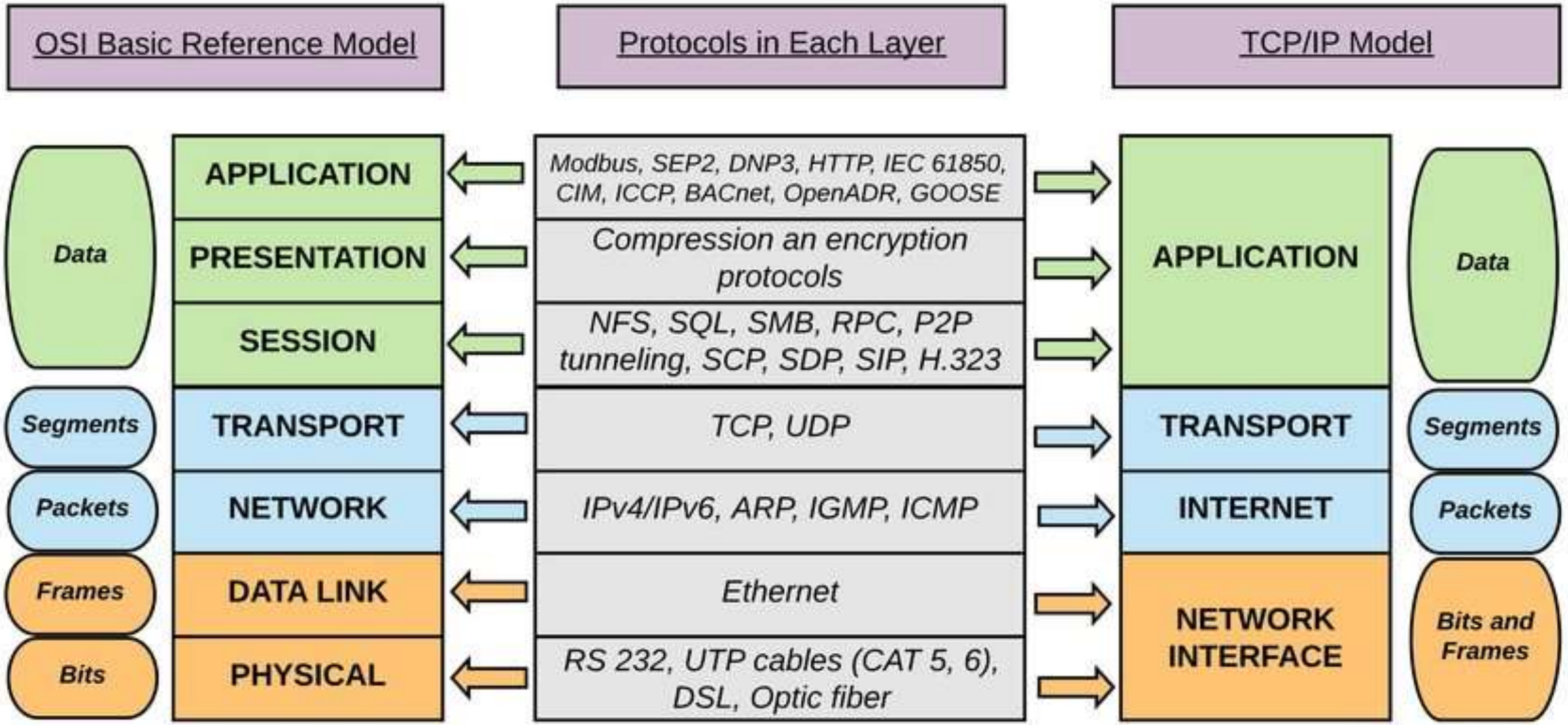
SPI Pins ESP8266 features two SPIs (SPI and HSPI) in slave and master modes.

! Single Analog pin present on the board !



This appears only after you include json file to the this to preferences:"http://arduino.esp8266.com/stable/package_esp8266com_index.json"


Preferences
Settings
Network
Sketchbook location:
C:\Users\balud\Documents\Arduino
Browse
Editor language:
System Default
(requires restart of Arduino)
Editor font size:
13
Interface scale:
Automatic
100
%
(requires restart of Arduino)
Theme:
Default theme
(requires restart of Arduino)
Show verbose output during:
compilation
upload
Compiler warnings:
None
Display line numbers
Enable Code Folding
Verify code after upload
Use external editor
Check for updates on startup
Save when verifying or uploading
Use accessibility features
Additional Boards Manager URLs:
66com_index.json,http://dan.drown.org/stm32duino/package_STM32duino_index.json
More preferences can be edited directly in the file
C:\Users\balud\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)
OK
Cancel


OSI Basic Reference Model
Protocols in Each Layer
TCP/IP Model
Data
Segments
Packets
Frames
Bits
APPLICATION
PRESENTATION
SESSION
TRANSPORT
NETWORK
DATA LINK
PHYSICAL
Modbus, SEP2, DNP3, HTTP, IEC 61850, CIM, ICCP, BACnet, OpenADR, GOOSE
Compression an encryption protocols
NFS, SQL, SMB, RPC, P2P tunneling, SCP, SDP, SIP, H.323
TCP, UDP
IPv4/IPv6, ARP, IGMP, ICMP
Ethernet
RS 232, UTP cables (CAT 5, 6), DSL, Optic fiber
APPLICATION
TRANSPORT
INTERNET
NETWORK INTERFACE
Data
Segments
Packets
Bits and Frames

HTTP(Hyper Text Transfer Protocol)

In this protocol, a client initiates communication by making a request for a specific web page using HTTP and the server responds with the content of that web page or an error message if unable to do so (like famous 404 Error). Pages delivered by a server are mostly HTML documents.

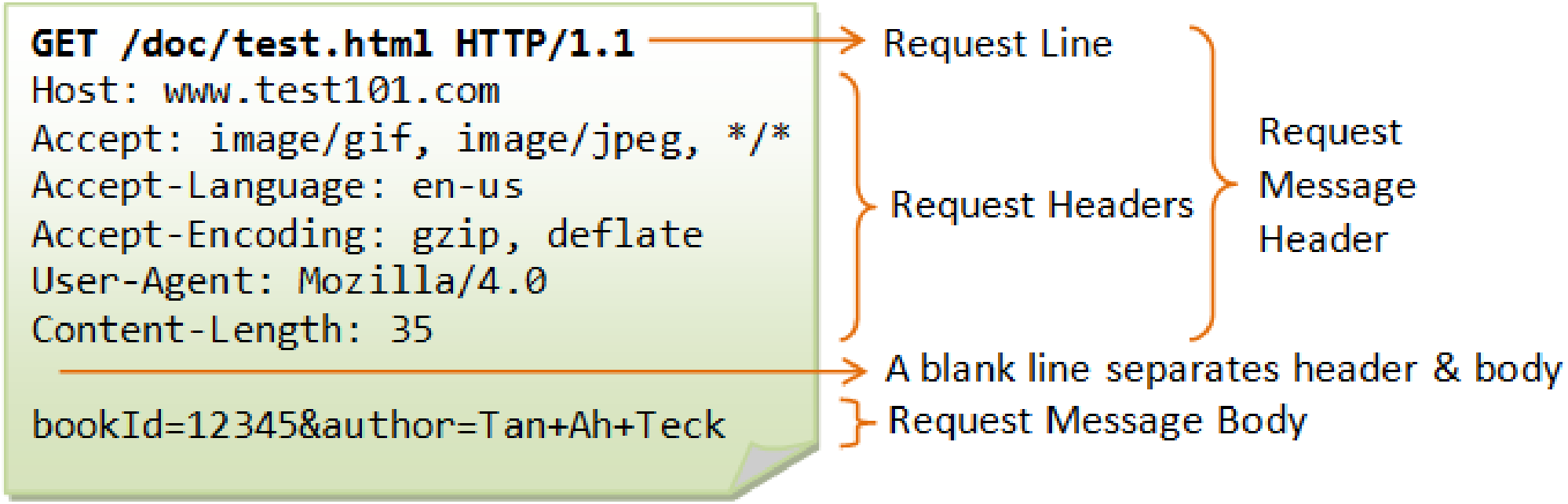

Web Server


The Globe
The Internet
Web Server
The user type
in the URL :
www.mywebsite.com/products/myproduct.html


Request (HTTP message)
Web Server
The Internet
The user type
in the URL :
www.mywebsite.com/products/myproduct.html


Web Server
Request (HTTP message)
The Internet
The user type
in the URL :
www.mywebsite.com/products/myproduct.html
Response(HTTP message)

This is how a http request looks like



https://www.w3.org/Protocols/rfc2616/rfc2616-sec4.html Refer this for entire knowledge on how request looks

Some of the http requests

GET

The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.

HEAD

The HEAD method asks for a response identical to that of a GET request, but without the response body.

POST

The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server.

PUT

The PUT method replaces all current representations of the target resource with the request payload.

DELETE

The DELETE method deletes the specified resource.

200 OK

The request has succeeded. The meaning of the success depends on the HTTP method:

- GET: The resource has been fetched and is transmitted in the message body.
- HEAD: The entity headers are in the message body.
- PUT or POST: The resource describing the result of the action is transmitted in the message body

300 Multiple Choice

The request has more than one possible response. The user-agent or user should choose one of them. (There is no standardized way of choosing one of the responses, but HTML links to the possibilities are recommended so the user can pick.)

401 Unauthorized

Although the HTTP standard specifies "unauthorized", semantically this response means "unauthenticated". That is, the client must authenticate itself to get the requested respons

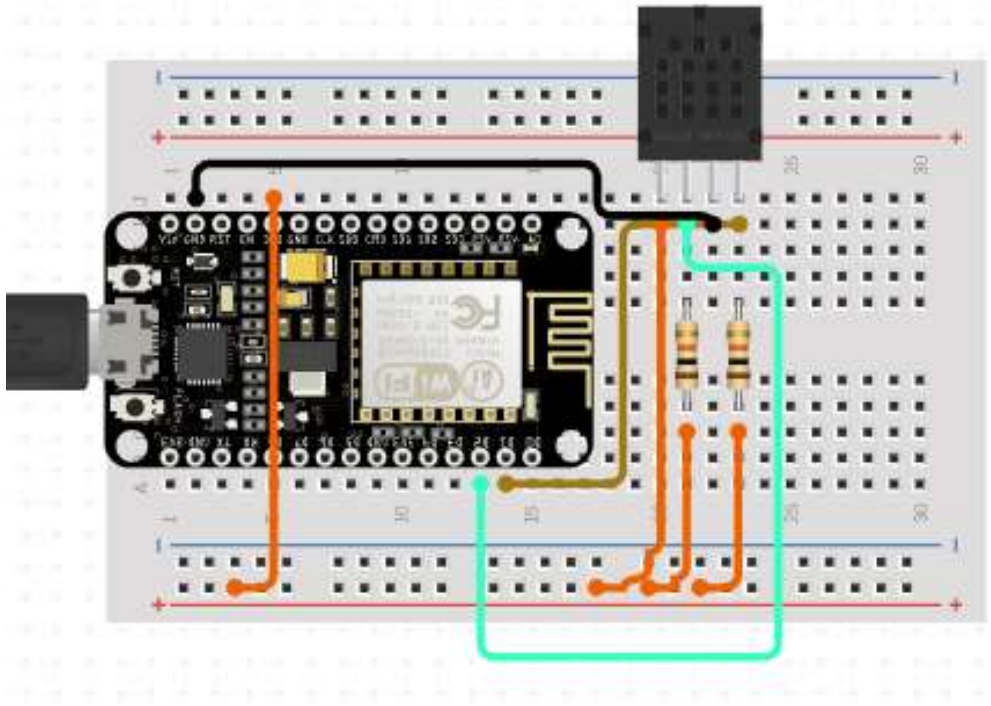
404 Not Found

The server can not find the requested resource. In the browser, this means the URL is not recognized. In an API, this can also mean that the endpoint is valid but the resource itself does not exist. Servers may also send this response instead of 403 to hide the existence of a resource from an unauthorized client. This response code is probably the most famous one due to its frequent occurrence on the web.

```
 #include <ESP8266WiFi.h>
// if a client is accessing server then his ip address and some other data shall be recorded and stored in request string.


WiFiServer server(80); //setting  up a server at port 80
String request = "";
void setup()
{
  Serial.begin(115200);

    boolean conn = WiFi.softAP("Wifi", "12345678"); // creating a wifi accesspoint
    server.begin();
}
void loop()
{
    WiFiClient client = server.available();// creating a client object
    if (!client)  {  return;  }
    request = client.readStringUntil('\r');// requesting data from client
    Serial.println(request);
    client.flush(); //after handling that request server is ready to accept request from other client
    delay(5);
}
}
```

A simple example to set up a server through nodemcu

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include "DHT.h"

// Uncomment one of the lines below for whatever DHT sensor type you're using!
//#define DHTTYPE DHT11   // DHT 11
//#define DHTTYPE DHT21   // DHT 21 (AM2301)
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321

/*Put your SSID & Password*/
const char* ssid = "YourNetworkName";  // Enter SSID here
const char* password = "YourPassword";  //Enter Password here

ESP8266WebServer server(80);

// DHT Sensor
uint8_t DHTPin = D8;

// Initialize DHT sensor.
DHT dht(DHTPin, DHTTYPE);

float Temperature;
float Humidity;
```



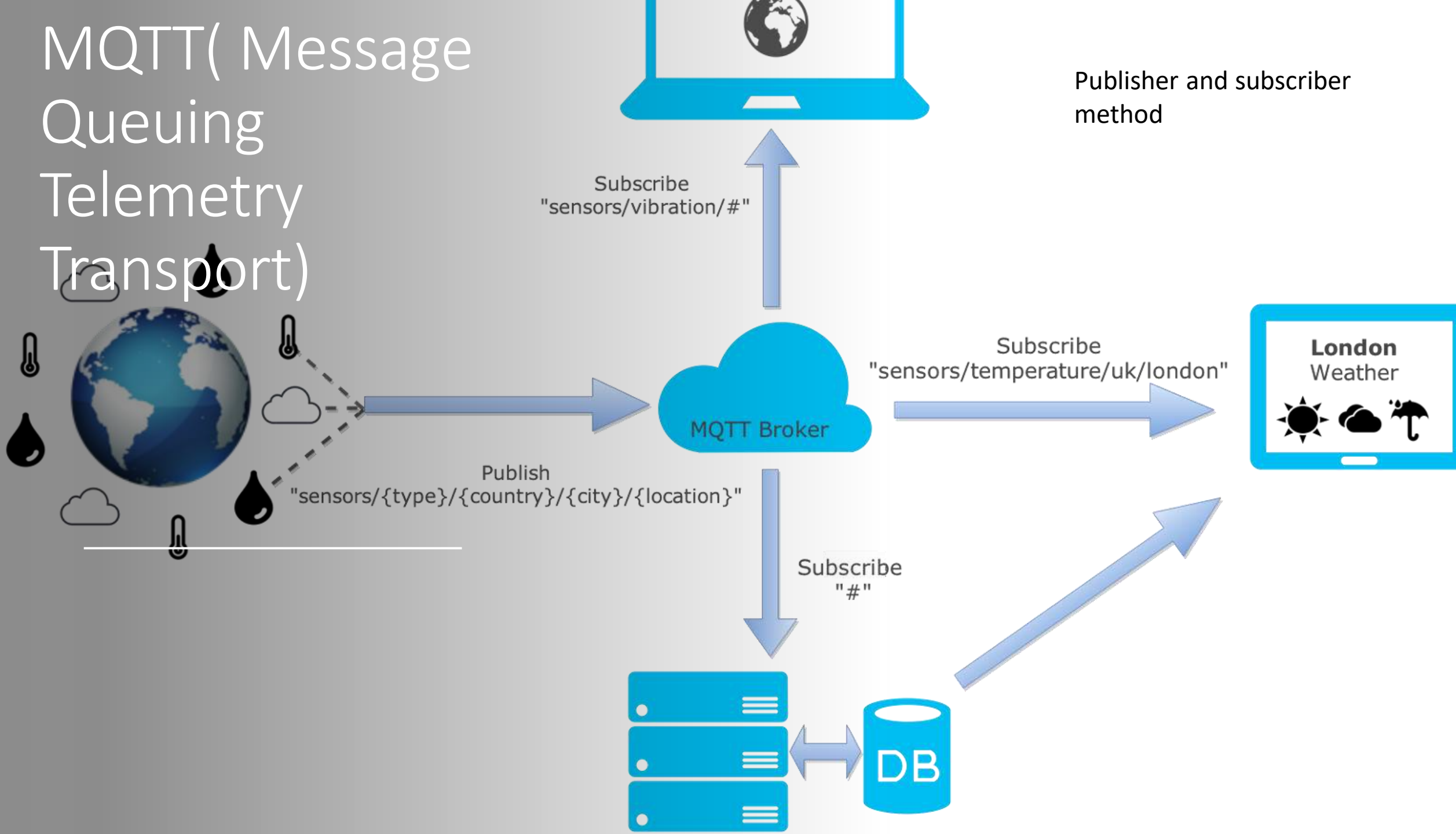
```
void setup() {
  Serial.begin(115200);
  delay(100);
  pinMode(DHTPin, INPUT);
  dht.begin();
  Serial.println("Connecting to ");
  Serial.println(ssid);

  //connect to your local wi-fi network
  WiFi.begin(ssid, password);

  //check wi-fi is connected to wi-fi network
  while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected..!");
  Serial.print("Got IP: ");  Serial.println(WiFi.localIP());
  server.on("/", handle_OnConnect);
  server.onNotFound(handle_NotFound);
  server.begin();
  Serial.println("HTTP server started");
}
```

```
void loop() {
  server.handleClient();
}
void handle_OnConnect() {
Temperature = dht.readTemperature(); // Gets the values of the temperature
  Humidity = dht.readHumidity(); // Gets the values of the humidity
  server.send(200, "text/html", SendHTML(Temperature,Humidity));
}
void handle_NotFound(){
  server.send(404, "text/plain", "Not found");
}
String SendHTML(float Temperaturestat,float Humiditystat){
  String ptr = "<!DOCTYPE html> <html>\n";
  ptr +="<head><meta name=\"viewport\" content=\"width=device-width, initial-
scale=1.0, user-scalable=no\">\n";
  ptr +="<title>ESP8266 Weather Report</title>\n";
  ptr +="</head>\n";
  ptr +="<body>\n";
```

```
  ptr +="<div id=\"webpage\">\n";
  ptr +="<h1>ESP8266 NodeMCU Weather
Report</h1>\n";

  ptr +="<p>Temperature: ";
  ptr +=(int)Temperaturestat;
  ptr +="°C</p>";
  ptr +="<p>Humidity: ";
  ptr +=(int)Humiditystat;
  ptr +="%</p>";

  ptr +="</div>\n";
  ptr +="</body>\n";
  ptr +="</html>\n";
  return ptr;
}
```

MQTT(Message Queuing Telemetry Transport)

Publisher and subscriber method



The protocol uses a publish/subscribe architecture in contrast to HTTP with its request/response paradigm.

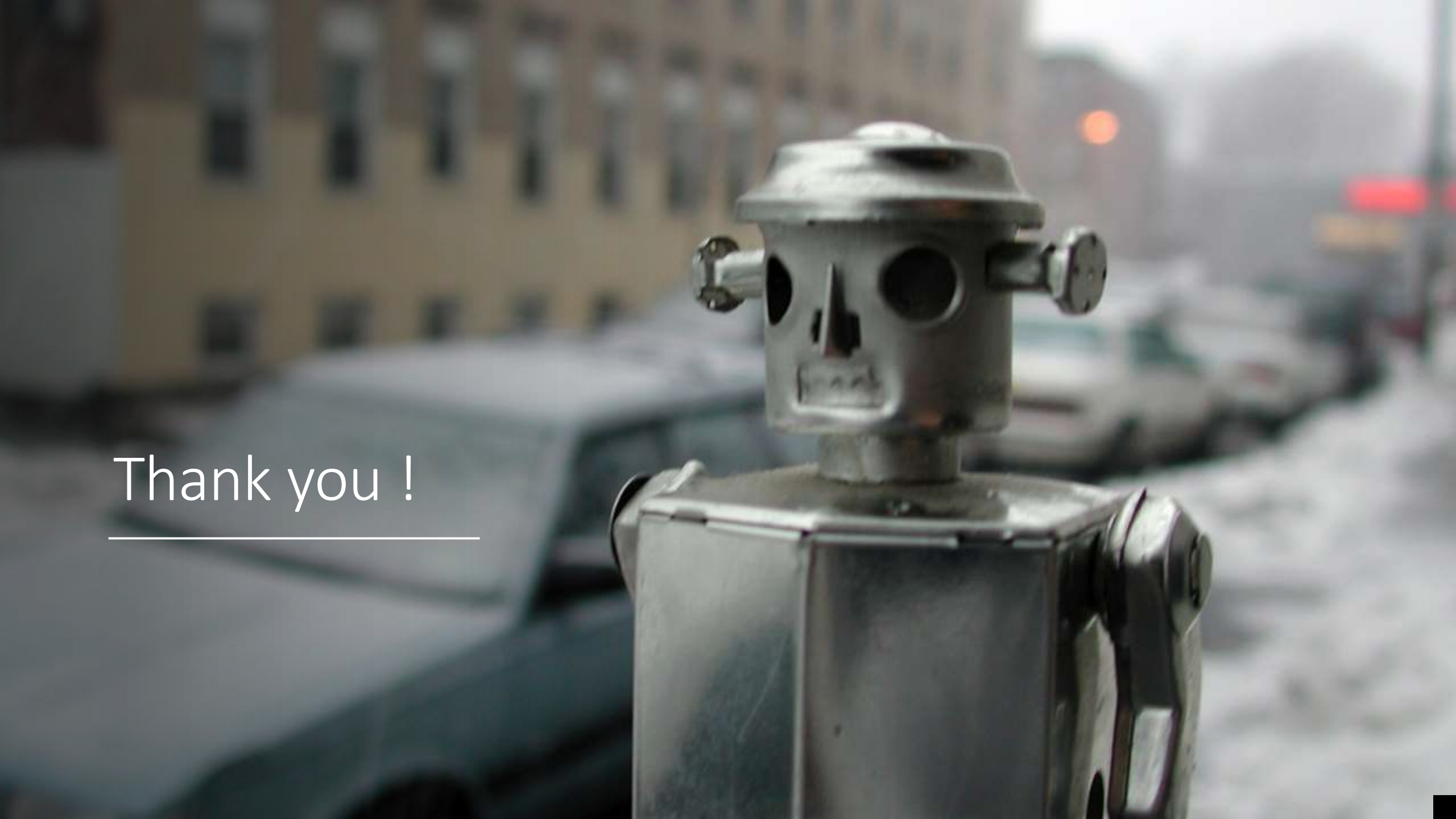
Mqtt

Http

MQTT Bokers

- Google Firebase
- Mosquitto
- Thingspeak

```
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>

#define FIREBASE_HOST "klen-d13b9.firebaseio.com"// topic we are subscribing to

const char* ssid     = "your wifi username";
const char* password = "your wifi password";
int p=12;
void setup() {
  Serial.begin(115200);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {//make sure you are connected to wifi
   delay(100);
  }
  Firebase.begin(FIREBASE_HOST);
}
void loop() {
  String x= Firebase.getString("KlenData/app"); //get data from the subscribed topic
  while(!Firebase.success());
  Serial.println(x);
  delay(1000);
}
```


Thank you !