



Internet of Things Workshop



Wameedh Scientific Club

Warning

It is totally normal and okay to have moments where you understand nothing in this workshop :v

CONTENTS

01 Getting started with ESP32

02 Getting used to ESP-IDF

03 Connecting to WiFi

04 TCP/IP sockets

Session's content:

- Introduction to the ESP32 and its' variants
- Comparing ESP32 to Arduino UNO
- Getting started with IoT Development Framework
- Understand the “blink” example code



01

Getting Started with ESP32

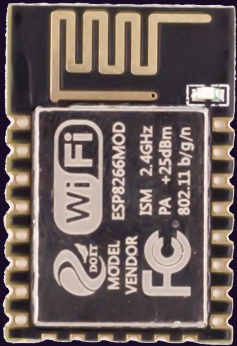
Introduction

ESP32 is a low-cost dual core microcontroller developed by Espressif Systems. It has both WiFi and Bluetooth integrated with various peripherals.

ESP32 vs Arduino UNO

Specs	ESP32	Arduino UNO
CPU	Dual Core 160 MHz	Single Core 16MHz
Flash Memory	4MB	32KB
SRAM	520KB	2KB
GPIOs	38	20
WiFi/Bluetooth	Yes	No
UART (Hardware Serial)	3	1
SPI	3	1
I ² C	2	1
Operating Voltage	3.3V	5V
Price	2500DA/ 3.5\$	2800DA/ 4\$

Other ESPs



ESP8266



ESP32 Wroom



ESP32 Wrover



ESP32-S2



ESP32 PICO D4



ESP32-S3

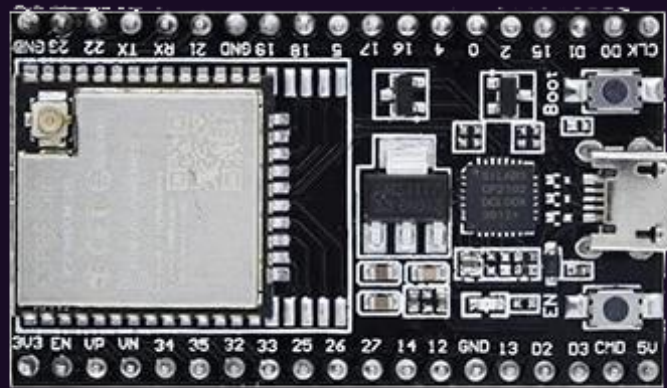
Development boards



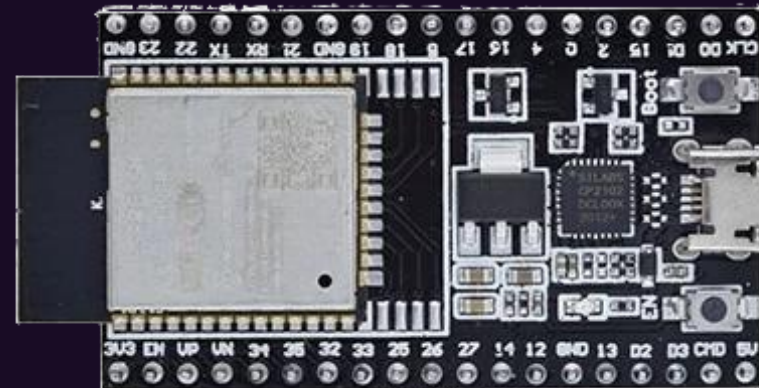
ESP-32 30PIN



ESP-32 38PIN



ESP32-WROOM-32U



ESP32-WROOM-32D

How to program the ESP32

Arduino Framework

Beginner friendly but thats it.
Very limited, lacks many features and
bad coding mindset (copy-paste)

IoT Dev Framework

Official framework supported by
Espressif. Extremely reliable and
configurable. Well documented with
growing community.

micropython, lua...

Just do not use this.
You do not want a high level language
when dealing with microcontrollers.
C/C++ is the way to go with
embedded systems.

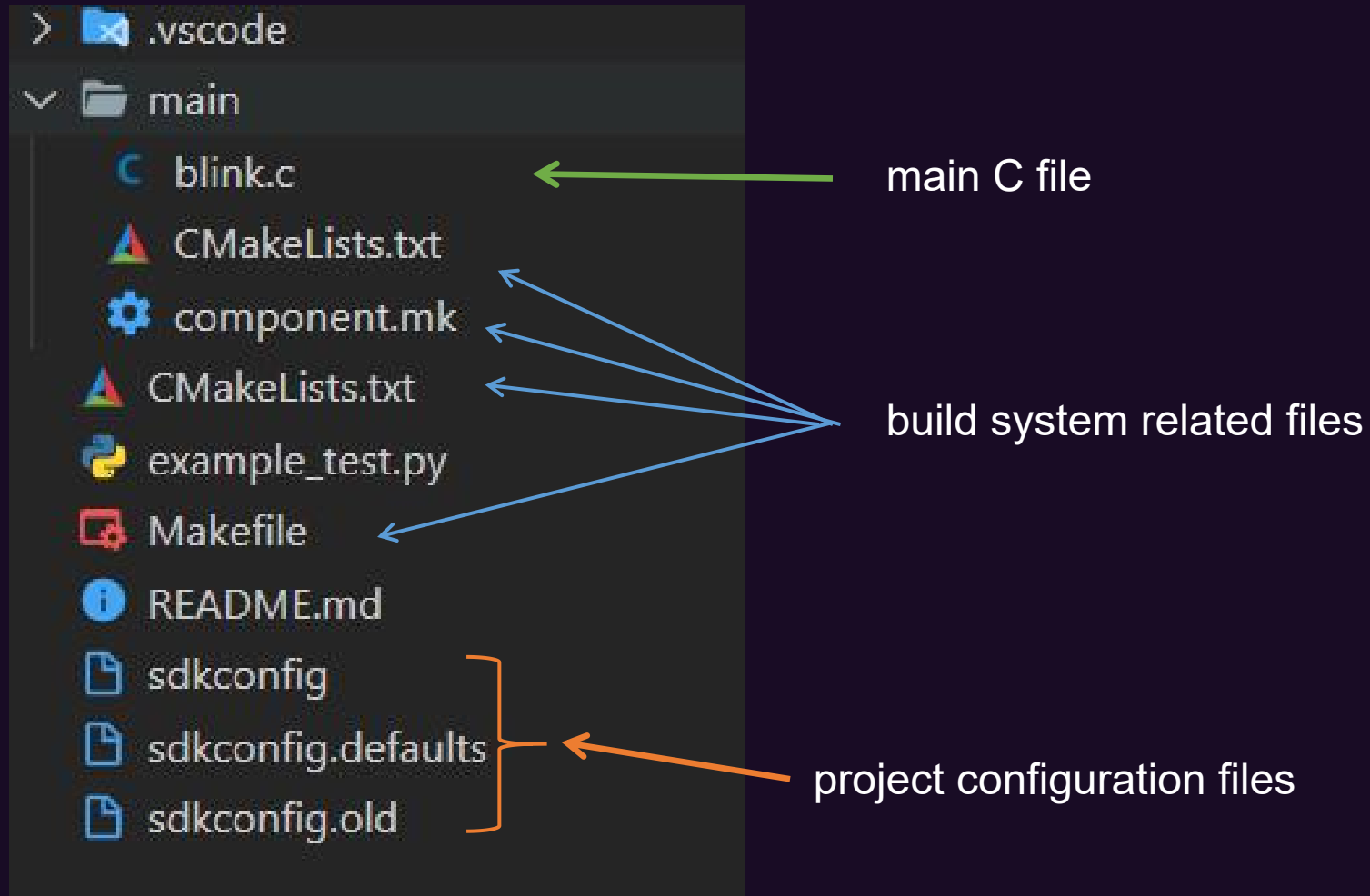
Setting up ESP-IDF

1. Install ESP-IDF with the esp-idf-tools-setup-offline-4.4.exe
2. Make sure the path you install esp-idf does not have space in it
3. Install VSCode
4. Install the Espressif IDF Extensions in VSCode
5. F1 or CTRL+Shift+P and search for “Configure ESP-IDF extension”
6. Choose “Use existing setup”

Your first ESP-IDF project

1. F1 or CTRL+Shift+P and search for “new project”
2. Set project name and directory then choose template
3. Pick the “blink” example as your project template

ESP-IDF Project Structure



ESP-IDF Extension status bar

Port

Target platform

Menu Config

Build

Flash

Monitor

New IDF
terminal



esp32



Project directory

Clean project

Build, Flash and Monitor

Arduino Framework vs ESP-IDF

Arduino	ESP-IDF
<code>pinMode(pin, mode)</code>	<code>gpio_set_direction(pin, mode)</code>
<code>digitalWrite(pin, state)</code>	<code>gpio_set_level(pin, state)</code>
<code>Serial.printf()</code>	<code>printf()</code>
<code>delay()</code>	<code>vTaskDelay()</code>
OUTPUT	<code>GPIO_MODE_OUTPUT</code>
INPUT	<code>GPIO_MODE_INPUT</code>



02

Getting used to ESP-IDF

Session's content:

- Understand the “Hello World” code example
- Learn how to use the ESP-IDF documentation and ESP32 datasheet
- Use “ESP_LOGx” instead of printf to make log statements
- A bit on freeRTOS and menuconfig

Resources you need

Example codes: Example codes are the best way to understand how to use any feature of the ESP32

Source code of esp-idf: You can open the source code by right-clicking and click on “go to definition”

Docs: docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/

Datasheet: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

Logging in general:

Log is outputting informations about a software or microcontroller to the console. This can be done using simple print statements. However, it gets really hard to debug big projects without organizing logs. One logging convention is to split your logs into 5 log levels: Verbose, Debug, Info, Warn, Error

	Verbose Statement	Debug Statement	Info Statement	Warn Statement	Error Statement
Verbose Level	Included	Included	Included	Included	Included
Debug Level	Excluded	Included	Included	Included	Included
Info Level	Excluded	Excluded	Included	Included	Included
Warn Level	Excluded	Excluded	Excluded	Included	Included
Error Level	Excluded	Excluded	Excluded	Excluded	Included
Off	Excluded	Excluded	Excluded	Excluded	Excluded

Logging in ESP-IDF:

Logs in ESP-IDF is achieved with esp_log.h library. The output has the following format:


Log Level (Time) TAG: Log Content

```
I (301) TAG: Information log: hello world
W (301) TAG: Warning log: hello world
E (311) TAG: Error log: hello world
```

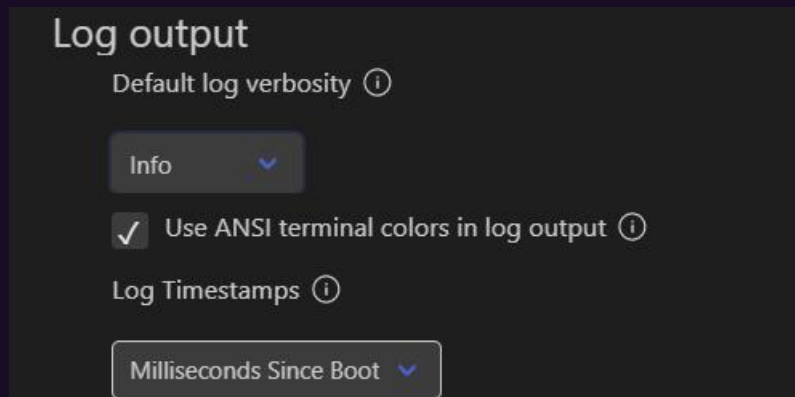
This is achieved with the following code

```
char data[] = "hello world";
ESP_LOGI("TAG", "Information log: %s", data);
ESP_LOGW("TAG", "Warning log: %s", data);
ESP_LOGE("TAG", "Error log: %s", data);
```

Introduction to MenuConfig:

MenuConfig is simply a configuration interface. You can find many options related to both the build system and esp-idf components. It can be accessed using the menuconfig icon in the status bar 

One of the things you can with menuconfig is change settings related to the serial monitor of the esp-idf. You can change log settings like log level (default is INFO) or the baudrate (default is 115200)



Introduction to freeRTOS:

freeRTOS is one of the most popular real-time operating systems for embedded systems. It is used in 35 different microcontrollers. An operating system is used to manage the execution of processes and provides the necessary resources to them (memory and cpu-time).

One of the benefits of using freeRTOS is the ability of multi-threading in embedded systems. Multi-threading is essentially running two different piece of code (functions) in parallel. They are called tasks with each task have its own priority.



03

Connecting to WiFi

Wi-Fi Introduction

Wifi is a family of wireless network protocols, based on the IEEE 802.11 family of standards, allowing nearby digital devices to exchange data by radio waves.

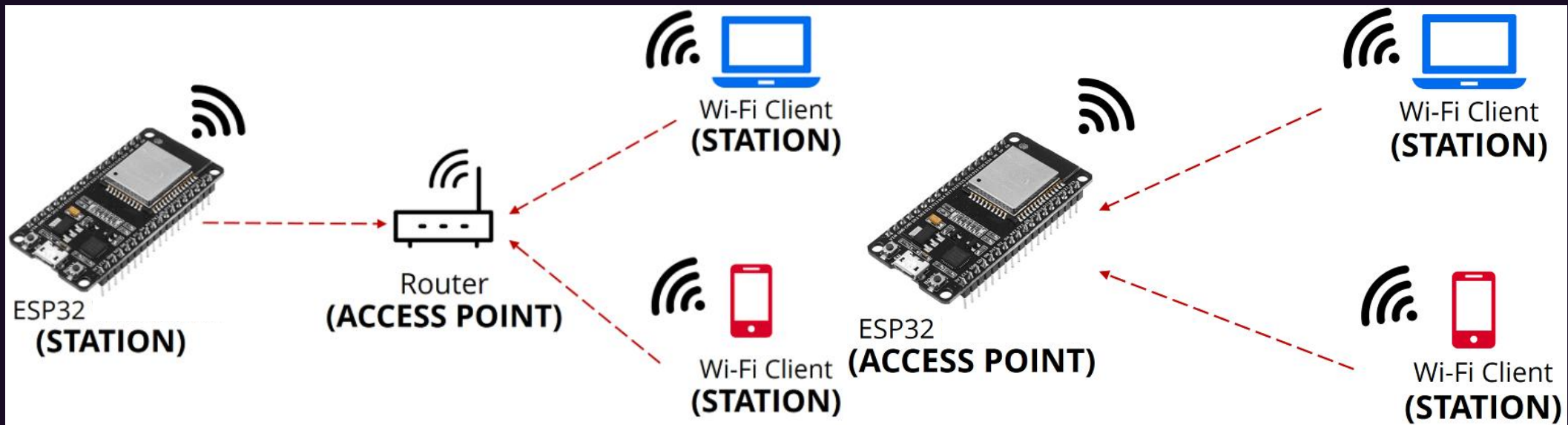
Wi-Fi Generations				
Generation	IEEE Standard	Maximum Linkrate (Mbit/s)	Adopted	Radio Frequency (GHz)
Wi-Fi 6E	802.11ax	600 to 9608	2020	6
Wi-Fi 6			2019	2.4/5
Wi-Fi 5	802.11ac	433 to 6933	2014	5
Wi-Fi 4	802.11n	72 to 600	2008	2.4/5
(Wi-Fi 3*)	802.11g	6 to 54	2003	2.4
(Wi-Fi 2*)	802.11a	6 to 54	1999	5
(Wi-Fi 1*)	802.11b	1 to 11	1999	2.4
(Wi-Fi 0*)	802.11	1 to 2	1997	2.4
*: (Wi-Fi 0, 1, 2, 3, are unbranded common usage. ^{[38][39]})				

Check Wikipedia for more

WiFi Operating mode

Station

softAP



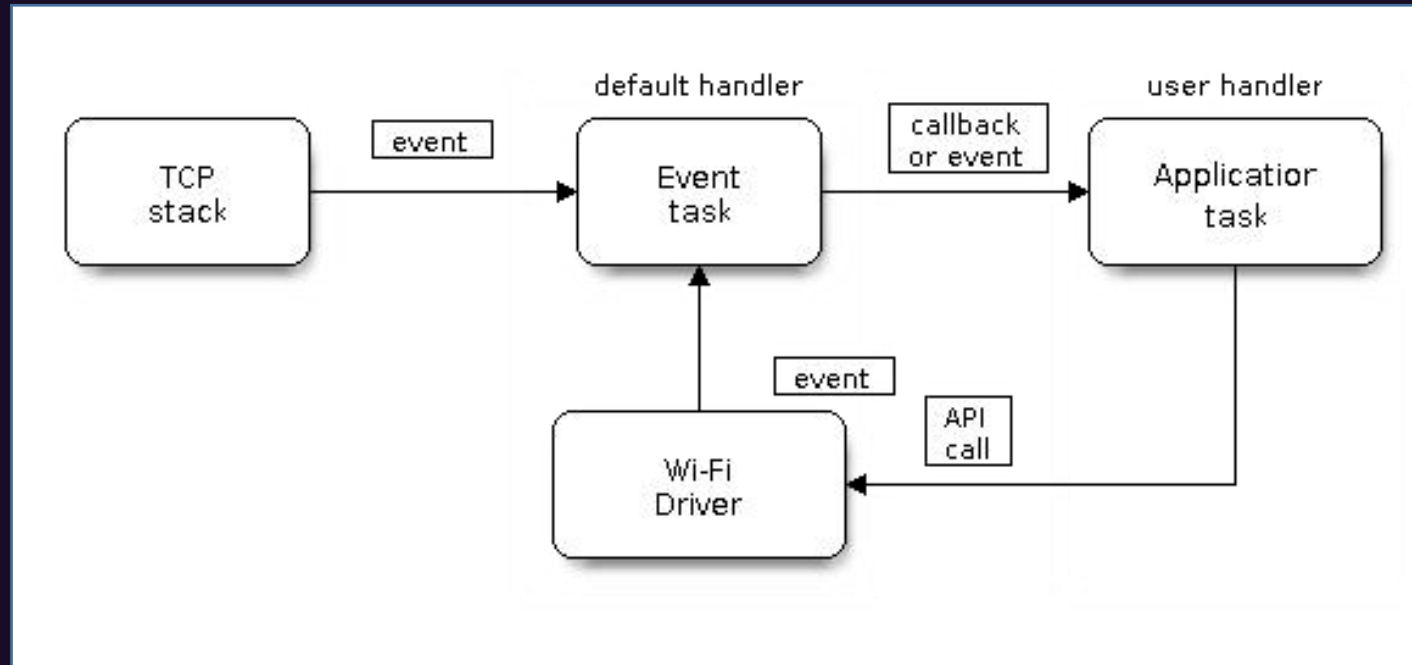
WiFi Station Example

1. F1 or CTRL+Shift+P and search for “new project”
2. Set project name and directory then choose template
3. Pick the “station” example as your project template

WiFi softAP Example

1. F1 or CTRL+Shift+P and search for “new project”
2. Set project name and directory then choose template
3. Pick the “softAP” example as your project template

WiFi Events





04

TCP/IP

TCP/IP

TCP/IP (Transmission Control Protocol/Internet Protocol) is one of the main protocols of the Internet protocol suite. The Internet protocol suite specifies how data should be packetized, addressed, transmitted, routed, and received. This functionality is organized into four abstraction layers. The IETF (Internet Engineering Task Force) is the organization that created and maintains the internet protocol suite.

Internet protocol suite

Application layer

BGP · DHCP(v6) · DNS · FTP · HTTP ·
HTTPS · IMAP · LDAP · MGCP · MQTT ·
NNTP · NTP · OSPF · POP · PTP · ONC/RPC
· RTP · RTSP · RIP · SIP · SMTP · SNMP ·
SSH · Telnet · TLS/SSL · XMPP · *more...*

Transport layer

TCP · UDP · DCCP · SCTP · RSVP · QUIC ·
more...

Internet layer

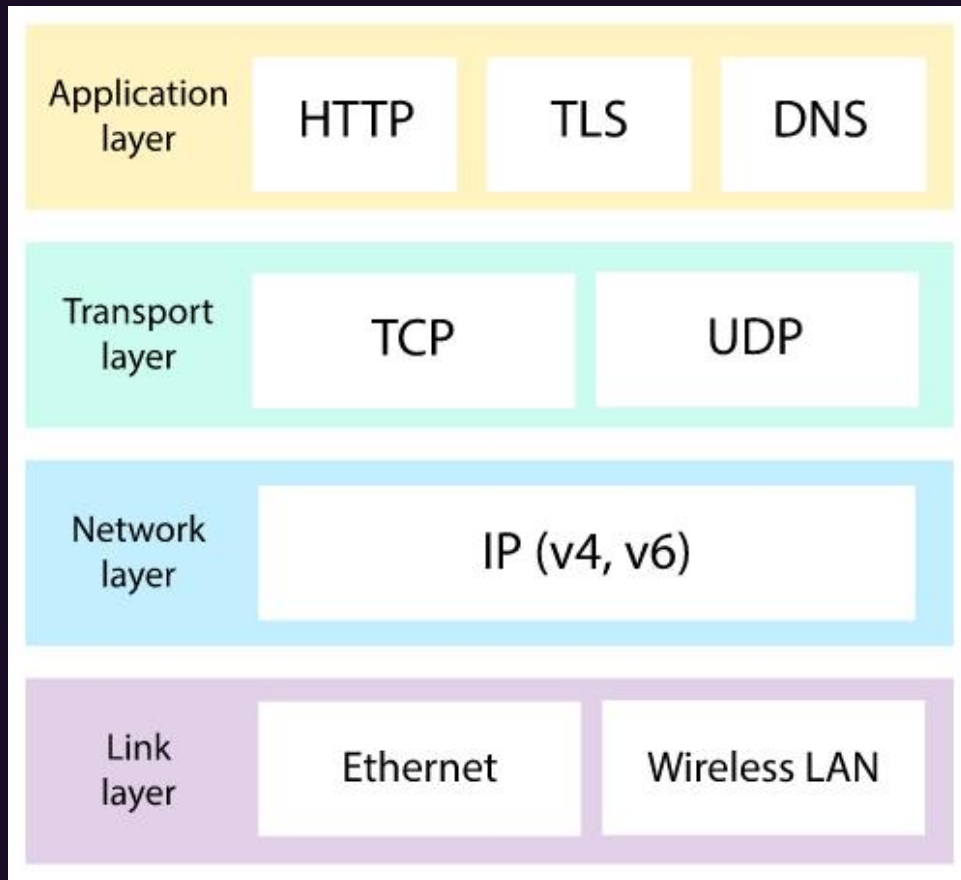
IP (IPv4 · IPv6) · ICMP(v6) · NDP · ECN ·
IGMP · IPsec · *more...*

Link layer

Tunnels · PPP · MAC
more...

Check Wikipedia for more

Layers of Internet protocol suite



A direct application of the transport layer protocols

Specifies how the communication between 2 devices should happen to ensure reliable transmission of packets

Specifies how packets are organized and makes sure they arrive at their destination.

Specifies how bits are sent between 2 devices

Khan Academy has good stuff on this

Why TCP/IP

TCP/IP allows reliable and simple communication between 2 devices. TCP provides accurate, ordered and error-checked stream of data. on the other hand, UDP is unordered and prone to packet losses but is faster compared to TCP.

In Internet of Things applications, we want to make sure that the data captured by our sensors is transmitted and stored correctly in databases.

If the data stream does not need to be reliable like video or audio streaming, UDP would be more suitable.

IP (Internet Protocol)

IP has the task of delivering packets from the source host to the destination host solely based on the IP addresses in the packet headers.

The most dominant version is IPv4 with the following packet structure

IPv4 header format																																	
Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification															Flags			Fragment Offset													
8	64	Time To Live								Protocol							Header Checksum																
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															
:	:																																
56	448																																

Check Wikipedia for more

TCP (Transmission Control Protocol)

TCP has the task of making sure that packets are error-free and has not been lost. Extra informations is added in the packet header on top of IP header.

TCP segment header																																	
Offsets	Octet	0								1								2								3							
Octet	Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0			N S	C W R	E C E	U R G	A C K	P S H	R S T	S Y N	F I N	Window Size															
16	128	Checksum																Urgent pointer (if URG set)															
20	160	Options (if <i>data offset</i> > 5. Padded at the end with "0" bits if necessary.)																															
⋮	⋮																																
60	480																																

Check Wikipedia for more

Network Addresses

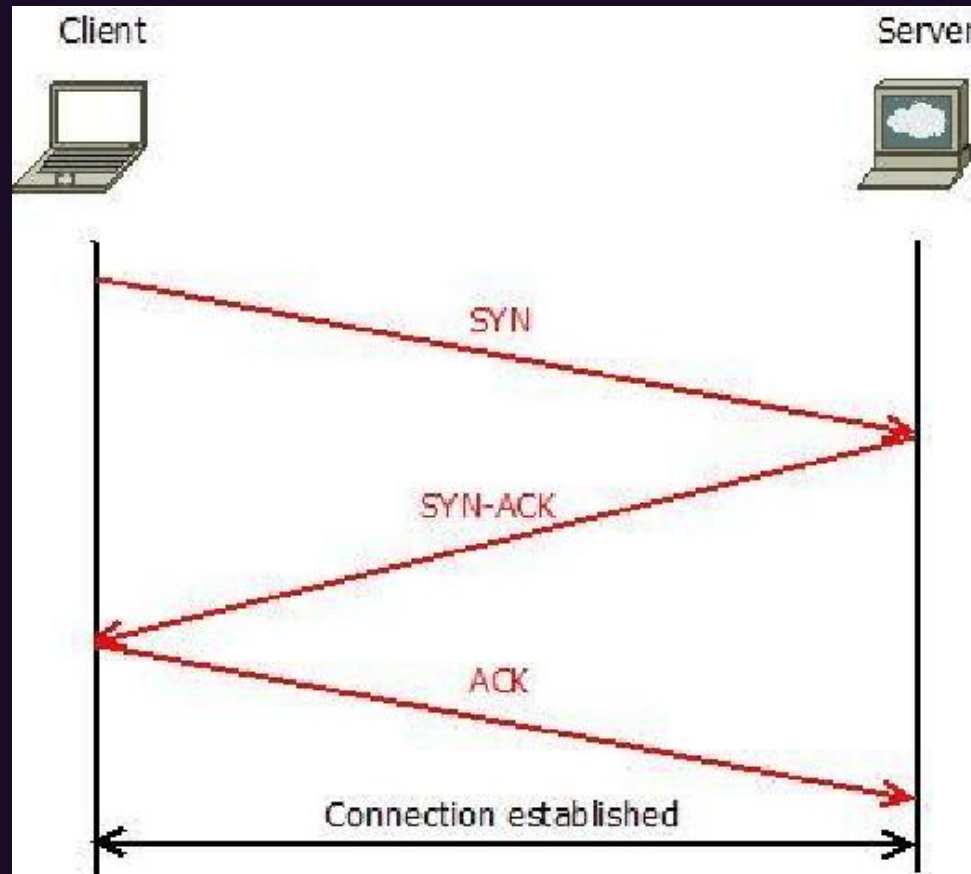
Each device connected to a network (LAN, Internet...etc) will get assigned an IP Address like the following example: 192.168.1.120

Each Device have to select a port from 0 to 65535 to set as a communication end-point. Ports can have a well-known function like port 80 for HTTP, 23 for SSH, 443 for HTTPS...etc.

Once the IP Address and Port are set for each device, connection can be established with

Connection 3-Way Handshake

Once the IP Address and Port are set for each device, connection can be established with the 3 way handshake



Socket Programming

Socket programming is basically the implementation of the internet protocol suite. A socket is an end-point of communication that has both IP address and port number. Sockets can be both clients and servers, their main functions are:

Client (initiator):

- `connect(socket_address)`
- `send(data)`
- `data = recv()`
- `close()`

Server (acceptor):

- `bind(socket_address)`
- `listen()`
- `accept()`
- `send(data)`
- `data = recv()`
- `close()`

Socket Programming

Sockets are universal across all platforms and programming languages.

The same sockets functions can be found in python, C/C++, Java, Javascript...etc

Sockets are supported even on small embedded systems like the esp32.

Because of the limited resources, lwIP (light-weight Internet Protocol) is used for embedded systems. Most of the functions of lwIP Sockets are identical to standard sockets found on PCs running windows or linux

TCP Client/Server Examples in ESP-IDF

The tcp_client and tcp_server examples on esp-idf can be used but if you find them complicated, you can find simplified version of them together with python client and server codes to run on your PC in the following github link:

https://github.com/dahmadjid/esp32-idf-codes/tree/main/tcp_sockets

What is Internet of Things



Thank you



Wameedh Scientific Club