

# **ESP32 CHAT-COMMUNICATION**

## **Project Report.**

**EKLAVYA MENTORSHIP PROGRAMME**

**At**

**SOCIETY OF ROBOTICS AND  
AUTOMATION, VEERMATA JIJABAI  
TECHNOLOGICAL INSTITUTE MUMBAI.**

**SEPTEMBER-OCTOBER 2021.**

## **ACKNOWLEDGMENT.**

It was truly a great experience to work with the seniors. We are extremely grateful for the guidance the seniors provided without which reaching our goal was impossible. We thank SRA for giving us the opportunity to work on this project and would love to work to achieve future goals together.

Special thanks to:

- Gautam Agrawal
- Dhairya Shah
- Shreyas

Team Members

Rishikesh Donadkar

[rishidonadkar1470@gmail.com](mailto:rishidonadkar1470@gmail.com)

+919623920069

Viraj Jagadale

[virajjagadale123@gmail.com](mailto:virajjagadale123@gmail.com)

+919167309396

## TABLE OF CONTENTS.

<b>NO.</b>	<b>TITLE</b>	<b>Page No.</b>
1.	Project Overview	4
2.	Technology Used <ul style="list-style-type: none"><li>● TCP</li><li>● MQTT</li><li>● ESPNOW</li></ul>	4-7 4-5 5 5-7
3.	Project Idea	7
4.	Basic Project Domains	7
5.	Workflow <ul style="list-style-type: none"><li>● ESPNOW</li><li>● TCP protocol</li></ul>	7-12 7-10 10-12
6.	Challenges Faced	12
7.	Future Work	12
8.	Reference Links <ul style="list-style-type: none"><li>● Networking</li><li>● Embedded C</li></ul>	13-14 13-14 14

## **1] PROJECT OVERVIEW**

In this project we will establish connection between two ESP32 boards (mainly focusing on the ESP-NOW protocol). Each board will receive messages from the users, from the prompt, and will send the messages to the other board.

## **2] TECHNOLOGY USED**

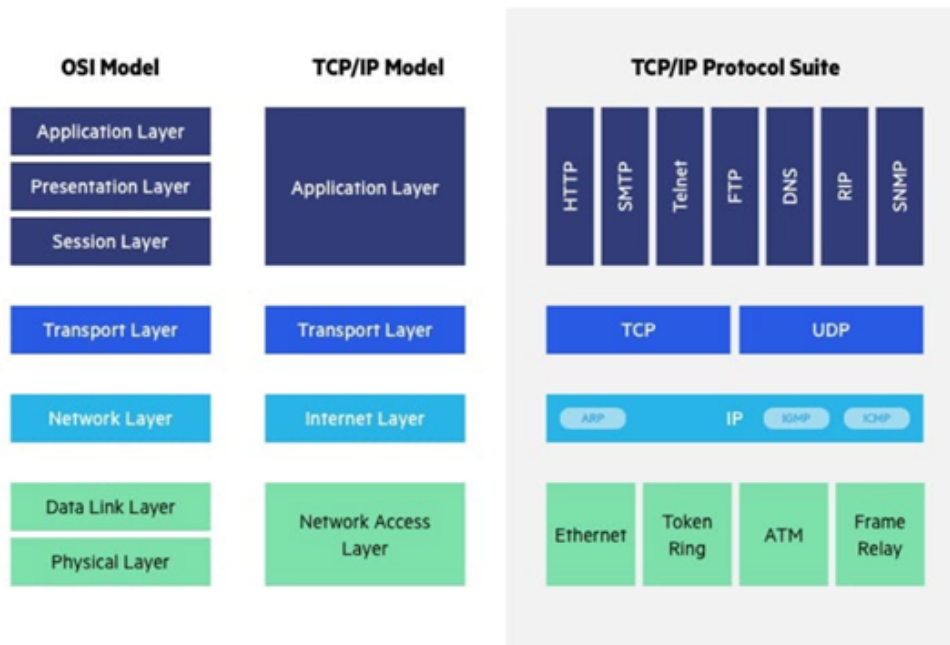
### **1] TCP protocol(Transmission control protocol.)**

It is a connection-oriented communication.

**Connection-oriented communication** is a network communication mode in [telecommunications](#) and computer networking, where a [communication session](#) or a semi-permanent connection is established before any useful data can be transferred, enabling the ability to ensure that data is delivered in the correct order to the upper communication layer. The alternative to connection-oriented transmission is [connectionless communication](#), for example, the [datagram](#) mode communication used by the IP and UDP protocols, where data may be delivered out of order, since different [network packets](#) are routed independently, and maybe delivered over different paths.

It is an intermediate layer of the application layer and internet protocol layer in the OSI model.

# OSI vs. TCP/IP Model



The communication over the network in the TCP/IP model takes place in the form of a client-server architecture. i.e., the client begins the communication and establishes a connection with a server.

Reliable connection --> TCP protocol.

Three-way handshake: First, the sender computer sends a message called SYN which is short for synchronize. Second, the receiver computer sends an ACK (acknowledgment) message in the form of SYN-ACK. Third, the sender's computer sends an ACK message.

Sequence numbers: Data is sent in the form of small segments; they are assigned some numbers. The receiver pc then reorders the segments according to their numbers.

## **2] MQTT-protocol.**

MQ Telemetry Transport.

Lightweight publish and subscribe system. Simple communication between multiple devices. Simple messaging protocol.

Application – Internet of Things.

MQTT Broker:

1. Receives the messages.
2. Filter the messages.
3. Publishes the messages to all the subscribed clients.

## **3] ESP-NOW**

ESP-NOW, is a protocol developed by Espressif. It is a fast communication protocol that can be used to exchange small messages (up to 250 bytes) between esp32 boards.

The protocol is similar to the low-power 2.4GHz wireless connectivity that is often deployed in wireless mice. So, the pairing between devices is needed prior to their communication. After the pairing is done, the connection is secure and peer-to-peer, with no handshake being required.

ESP-NOW is versatile and you can have one-way or two-way communication between two esp32 boards.

## **Why use ESP-NOW?**

- It allows ESP devices to communicate directly without connecting to a WiFi network
- The pairing between devices is needed prior to their communication. After the pairing is done, the connection is secure and peer-to-peer. If suddenly one of your boards loses power or resets, when it restarts, it will automatically connect to its peer to continue the communication.

- It does not require a router to connect two esp32 boards. Thus, it can be used at any remote places.
- It provides encrypted and unencrypted unicast communication.
- It does not need to connect to a Wifi access point which makes it a fast communication protocol .
- Maximum 20 nodes can be connected.

### **3] PROJECT IDEA**

The aim of the project is to establish a chat-system between two ESP32. We will create a project where we would have the user type a message in the console and send it to the other user via ESP-NOW. We will also learn about TCP and MQTT protocols.

### **4] BASIC PROJECT DOMAINS**

- ☐ Embedded C
- ☐ Networking

### **5] WORKFLOW**

**1] ESP-NOW** protocol:

<https://github.com/espressif/esp-idf/tree/master/examples/wifi/espnow>

While using the above espnow-example to send data, you will come across a lot of 'extras' on top of espnow data. These 'extras' are not required to send the use the ESPNOW protocol, however to make the data more safe and reliable it is recommended that we use them.

We will require two esp devices for this project. While sending data these two devices must be on the same channel, must have the same primary and local master key.

The maximum data that can be transferred is of 250 bytes which can be changed under “Example Configuration”.

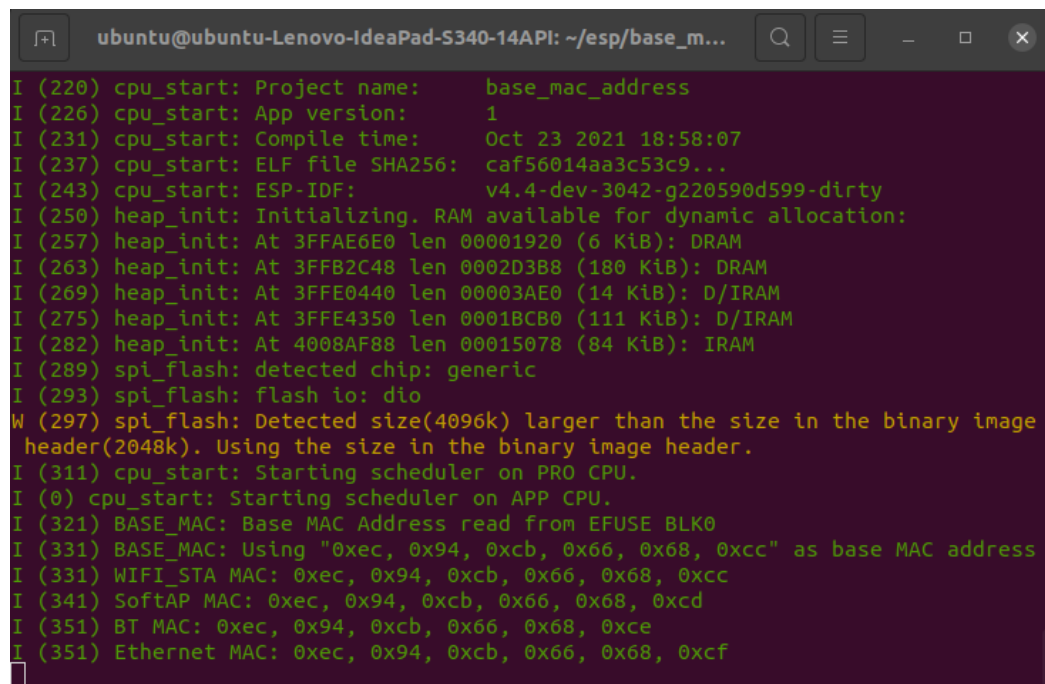
In our project we will be using the code snippets from the following ESPNOW code.

[https://github.com/Mair/esp32-course/tree/master/18\\_ESP\\_NOW/18\\_esp\\_now](https://github.com/Mair/esp32-course/tree/master/18_ESP_NOW/18_esp_now)

It is not necessary to know the mac address of the esp. To find out the mac address of the device we will flash this code on our esp device.

[https://github.com/espressif/esp-idf/tree/master/examples/system/base\\_mac\\_address](https://github.com/espressif/esp-idf/tree/master/examples/system/base_mac_address)

You will get the following output on your terminal.



```
ubuntu@ubuntu-Lenovo-IdeaPad-S340-14API: ~/esp/base_m...
I (220) cpu_start: Project name:      base_mac_address
I (226) cpu_start: App version:      1
I (231) cpu_start: Compile time:      Oct 23 2021 18:58:07
I (237) cpu_start: ELF file SHA256:   caf56014aa3c53c9...
I (243) cpu_start: ESP-IDF:           v4.4-dev-3042-g220590d599-dirty
I (250) heap_init: Initializing. RAM available for dynamic allocation:
I (257) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
I (263) heap_init: At 3FFB2C48 len 0002D3B8 (180 KiB): DRAM
I (269) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
I (275) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (282) heap_init: At 4008AF88 len 00015078 (84 KiB): IRAM
I (289) spi_flash: detected chip: generic
I (293) spi_flash: flash io: dio
W (297) spi_flash: Detected size(4096k) larger than the size in the binary image header(2048k). Using the size in the binary image header.
I (311) cpu_start: Starting scheduler on PRO CPU.
I (0) cpu_start: Starting scheduler on APP CPU.
I (321) BASE_MAC: Base MAC Address read from EFUSE BLK0
I (331) BASE_MAC: Using "0xec, 0x94, 0xcb, 0x66, 0x68, 0xcc" as base MAC address
I (331) WIFI_STA MAC: 0xec, 0x94, 0xcb, 0x66, 0x68, 0xcc
I (341) SoftAP MAC: 0xec, 0x94, 0xcb, 0x66, 0x68, 0xcd
I (351) BT MAC: 0xec, 0x94, 0xcb, 0x66, 0x68, 0xce
I (351) Ethernet MAC: 0xec, 0x94, 0xcb, 0x66, 0x68, 0xcf
```

Now your esp devices are all set up to send and receive data.



ESP-IDF provides a console component to take messages from the terminal. This includes building blocks needed to develop an interactive console over serial port.

<https://github.com/espressif/esp-idf/tree/master/examples/system/console/advanced>

The above console component is configured such that it will only take commands from the prompt and act accordingly. These commands are predefined. We will make the required changes in the code to accept messages from the user.

Now the next challenge would be to pass the data from the console to espnow for it to be sent. We will pass the string with the help of 'Free RTOS Queue'. It is a type of data structure which follows the 'first in first out' approach.

In total we have created three tasks.

1. Console.
2. To Send Data.
3. To Receive Data.

The 'Console' task executes for the entire duration to take input from the user. When the user presses enter to send data, we create the 'Send' task to send data and suspend the console task. When the esp device receives data we will create a 'Receive' task with a higher priority than that of the console and will print the data on the terminal.

```
ubuntu@ubuntu-Lenovo-IdeaPad-S340-14API: ~/Desktop/ES...
I (533) wifi:Init management frame dynamic rx buffer num: 32
I (533) wifi:Init management short buffer num: 32
I (543) wifi:Init dynamic tx buffer num: 32
I (543) wifi:Init static rx buffer size: 1600
I (543) wifi:Init static rx buffer num: 10
I (553) wifi:Init dynamic rx buffer num: 32
I (553) wifi_init: rx ba win: 6
I (553) wifi_init: tcpip mbox: 32
I (563) wifi_init: udp mbox: 6
I (563) wifi_init: tcp mbox: 6
I (563) wifi_init: tcp tx win: 5744
I (583) wifi_init: tcp rx win: 5744
I (583) wifi_init: tcp mss: 1440
I (583) wifi_init: WiFi IRAM OP enabled
I (593) wifi_init: WiFi RX IRAM OP enabled
I (593) phy_init: phy_version 4670,719f9f6, Feb 18 2021,17:07:07
I (693) wifi:mode : sta (c4:dd:57:5b:f3:84)
I (693) wifi:enable tsf
I (693) ESPNOW: espnow [version: 1.0] init

Chat Communication
Sent: > omae wa mou shindeiru!
Received: nani?
```

```
ubuntu@ubuntu-Lenovo-IdeaPad-S340-14API: ~/Desktop/ES...
I (489) wifi:Init management frame dynamic rx buffer num: 32
I (499) wifi:Init management short buffer num: 32
I (499) wifi:Init dynamic tx buffer num: 32
I (499) wifi:Init static rx buffer size: 1600
I (509) wifi:Init static rx buffer num: 10
I (509) wifi:Init dynamic rx buffer num: 32
I (509) wifi_init: rx ba win: 6
I (509) wifi_init: tcpip mbox: 32
I (519) wifi_init: udp mbox: 6
I (519) wifi_init: tcp mbox: 6
I (519) wifi_init: tcp tx win: 5744
I (539) wifi_init: tcp rx win: 5744
I (539) wifi_init: tcp mss: 1440
I (539) wifi_init: WiFi IRAM OP enabled
I (549) wifi_init: WiFi RX IRAM OP enabled
I (549) phy_init: phy_version 4670,719f9f6, Feb 18 2021,17:07:07
I (659) wifi:mode : sta (ec:94:cb:66:68:cc)
I (659) wifi:enable tsf
I (669) ESPNOW: espnow [version: 1.0] init

Chat Communication
Received: omae wa mou shindeiru!
Sent: > nani?
Sent: >
```

Now while typing a message if your esp receives a message and your typed message is incomplete, your typed message will get stored and will get displayed once you start typing.

## 2] TCP protocol:

### TCP with esp

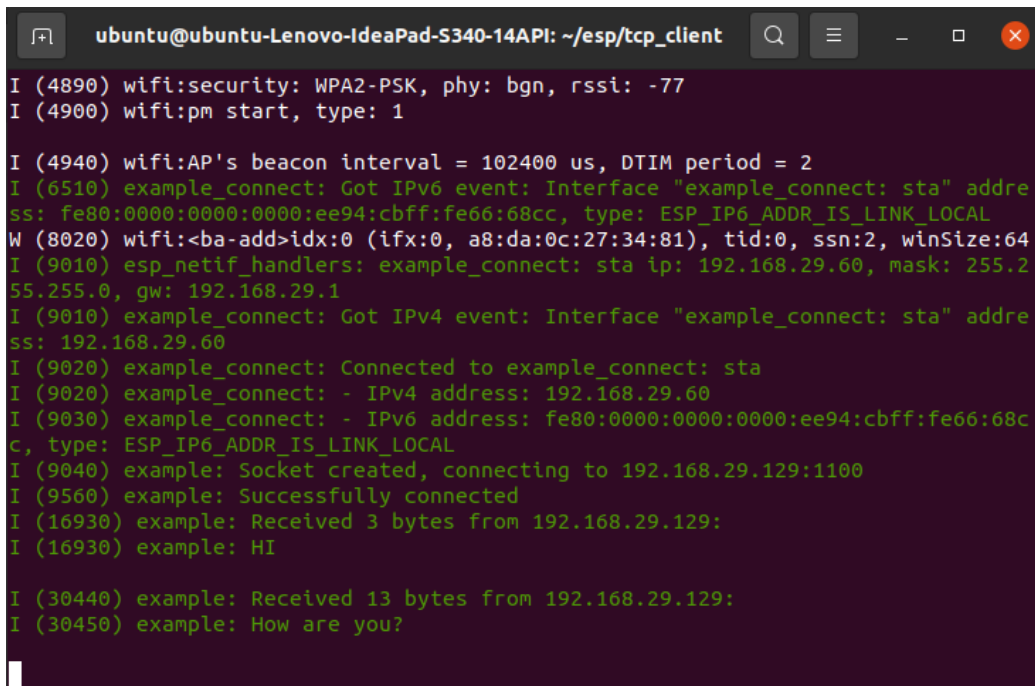
[https://github.com/espressif/esp-idf/tree/master/examples/protocols/sockets/tcp\\_client](https://github.com/espressif/esp-idf/tree/master/examples/protocols/sockets/tcp_client)

We will use this code to send data through tcp protocol.

We will use the command “netcat -nvlp 1100” to connect the terminal to port number 1100, to send messages to the esp device.

We will first need to make changes under the “Example Connection Configuration”. Change the ssid & password to connect the esp to the internet.

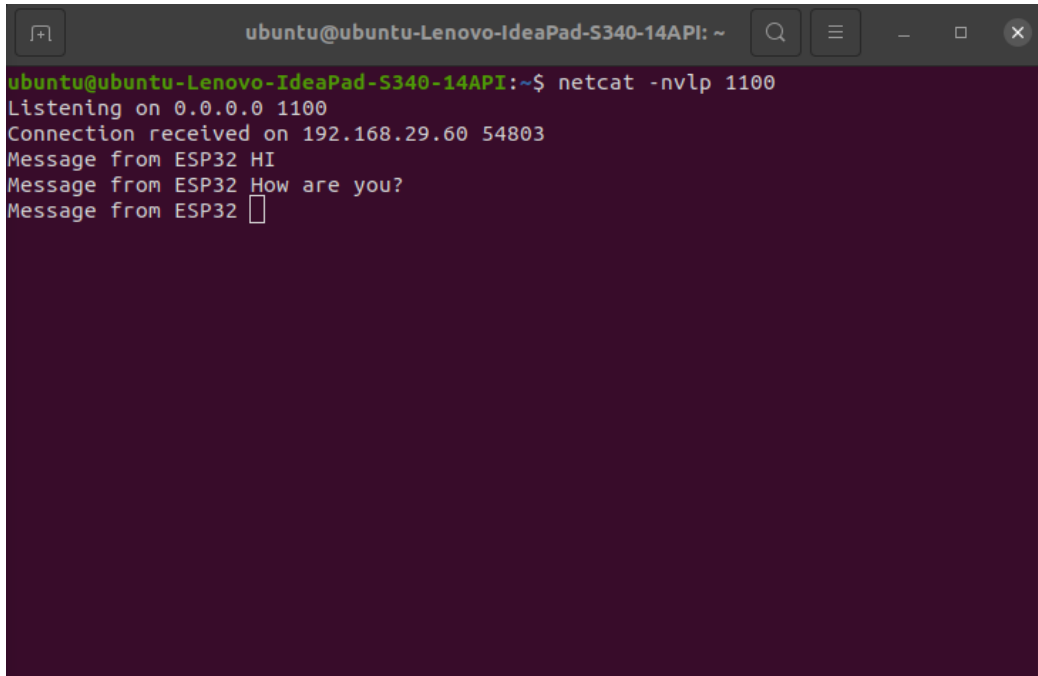
On flashing the code you will get the following output.



```
ubuntu@ubuntu-Lenovo-IdeaPad-S340-14API: ~/esp/tcp_client
I (4890) wifi:security: WPA2-PSK, phy: bgn, rssi: -77
I (4900) wifi:pm start, type: 1

I (4940) wifi:AP's beacon interval = 102400 us, DTIM period = 2
I (6510) example_connect: Got IPv6 event: Interface "example_connect: sta" address: fe80:0000:0000:0000:ee94:cbff:fe66:68cc, type: ESP_IP6_ADDR_IS_LINK_LOCAL
W (8020) wifi:<ba-add>idx:0 (ifx:0, a8:da:0c:27:34:81), tid:0, ssn:2, winSize:64
I (9010) esp_netif_handlers: example_connect: sta ip: 192.168.29.60, mask: 255.255.255.0, gw: 192.168.29.1
I (9010) example_connect: Got IPv4 event: Interface "example_connect: sta" address: 192.168.29.60
I (9020) example_connect: Connected to example_connect: sta
I (9020) example_connect: - IPv4 address: 192.168.29.60
I (9030) example_connect: - IPv6 address: fe80:0000:0000:0000:ee94:cbff:fe66:68cc, type: ESP_IP6_ADDR_IS_LINK_LOCAL
I (9040) example: Socket created, connecting to 192.168.29.129:1100
I (9560) example: Successfully connected
I (16930) example: Received 3 bytes from 192.168.29.129:
I (16930) example: HI

I (30440) example: Received 13 bytes from 192.168.29.129:
I (30450) example: How are you?
```

A terminal window titled 'ubuntu@ubuntu-Lenovo-IdeaPad-S340-14API: ~' with standard window controls. The terminal shows the command 'netcat -nvlp 1100' being executed. The output shows the listener on 0.0.0.0 1100 receiving a connection from 192.168.29.60 on port 54803. It then receives three messages from the ESP32: 'HI', 'How are you?', and an empty line.

```
ubuntu@ubuntu-Lenovo-IdeaPad-S340-14API: ~$ netcat -nvlp 1100
Listening on 0.0.0.0 1100
Connection received on 192.168.29.60 54803
Message from ESP32 HI
Message from ESP32 How are you?
Message from ESP32
```

## **6] CHALLENGES FACED**

- We were unable to use the MQTT protocol.
- As both of us have no experience in Web Development, we did not use the HTTP protocol.
- Faced challenges in designing the api of the project components.

## **7] FUTURE WORK**

- Host an http web server on the esp32 along with espnow to provide a more attractive and dynamic chat interface that can be used through a browser on mobile phone or personal computer.
- Add encryption to the espnow interconnection.

## **8] REFERENCE LINKS**

### **A] Networking**

Basic of networks

<https://www.educative.io/blog/computer-networking-101>

TCP:

<https://embeddeddiaries.com/esp32-with-tcp-ip-protocol/>

TCP blogs-

<https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-internet/xcae6f4a7ff015e7d:transporting-packets/a/transmission-control-protocol--tcp>

OSI model:

<https://www.imperva.com/learn/application-security/osi-model/>

[https://www.educative.io/blog/osi-model-layers?aid=5082902844932096&utm\\_source=google&utm\\_medium=cpc&utm\\_campaign=blog-dynamic&utm\\_term=&utm\\_campaign=Dynamic+-+Blog&utm\\_source=adwords&utm\\_medium=ppc&hsa\\_acc=5451446008&hsa\\_cam=8090938743&hsa\\_grp=82569843726&hsa\\_ad=396819070286&hsa\\_src=g&hsa\\_tgt=aud-470210443636:dsa-837938538428&hsa\\_kw=&hsa\\_mt=b&hsa\\_net=adwords&hsa\\_ver=3&gclid=CjwKCAjwy7CKBhBM EiwA0Eb7auwIGJxf-PI5VuH7mXaNXZBB9TyWOOUKz1TgUoIGs76JqQ0sfG49ZBoCkr4QAvD BwE](https://www.educative.io/blog/osi-model-layers?aid=5082902844932096&utm_source=google&utm_medium=cpc&utm_campaign=blog-dynamic&utm_term=&utm_campaign=Dynamic+-+Blog&utm_source=adwords&utm_medium=ppc&hsa_acc=5451446008&hsa_cam=8090938743&hsa_grp=82569843726&hsa_ad=396819070286&hsa_src=g&hsa_tgt=aud-470210443636:dsa-837938538428&hsa_kw=&hsa_mt=b&hsa_net=adwords&hsa_ver=3&gclid=CjwKCAjwy7CKBhBM EiwA0Eb7auwIGJxf-PI5VuH7mXaNXZBB9TyWOOUKz1TgUoIGs76JqQ0sfG49ZBoCkr4QAvD BwE)

How does NETFLIX bring safer and faster streaming?

<https://netflixtechblog.com/how-netflix-brings-safer-and-faster-streaming-experience-to-the-living-room-on-crowded-networks-78b8de7f758c>

RSA (encryption technique)

<https://brilliant.org/wiki/rsa-key-exchange/>

SSL/TLS Handshake:

<https://www.ssl.com/article/ssl-tls-handshake-overview/>

HTTP:

[https://www.w3schools.com/tags/ref\\_httpmethods.asp](https://www.w3schools.com/tags/ref_httpmethods.asp)

Mqtt protocols-

Blogs -

<https://www.hivemq.com/mqtt-essentials/>

Api -

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/mqtt.html>

## B] Embedded C

freeRTOS Playlist:

<https://youtube.com/playlist?list=PLXyB2ILBXW5FLc7j2hLcX6sAGbmH0JxX8>

freeRTOS study-group

<https://sравјti.in/embedded-systems-study-group/week6/week6.html>

API reference:

<https://www.freertos.org/a00106.html>

API reference for espnow:

[https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp\\_now.html](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html)

Esp-idf project build system-

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/build-system.html>

Console frontend system-

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32s2/api-reference/system/console.html>