Lecture 16

WiFi and LEDC on ESP32

Agenda

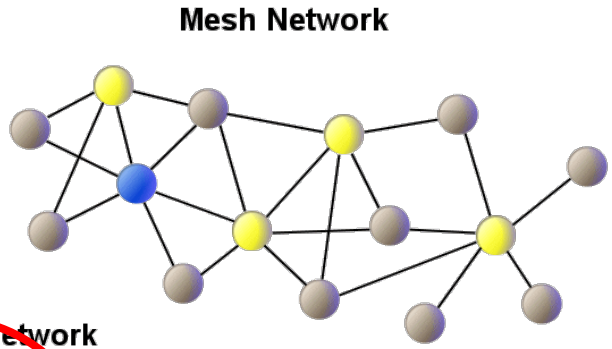
- Wireless protocols and communcations issues
- TCP/IP on ESP32
- UDP on ESP32
- LEDC on ESP32

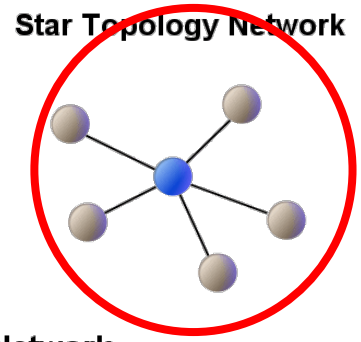
01

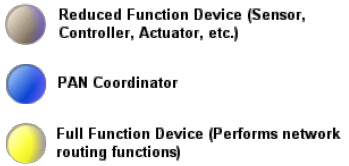
Wireless Communications

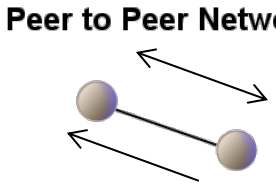
Serial Communications

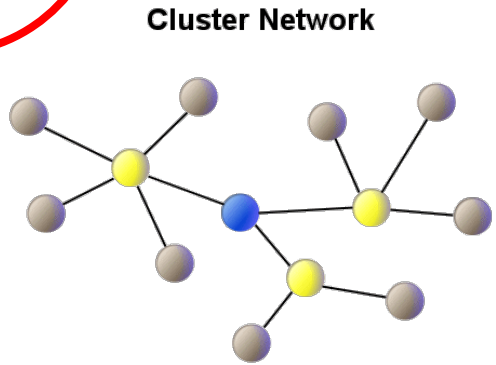
- Important Properties:
  - Data rate
  - Distance
  - Power consumption
  - Protocol Stack (complexity)
- Network architectures
  - Point to point vs multi-drop
  - Half vs full duplex
  - Broadcast (one master)
  - Mesh
  - Star
  - Cluster











Communication comparison (sorted by speed)

Method	net	Speed [kbs]	Range [m]	Power [mW]	Wirelss Freq	notes
Ethernet	bus	1000 to 1000000	100's	500	wired	Complex stack (TCP/IP etc.)
WiFi (802.11) (b,g,n,ac)	wlan	1000 to 150000	70 - 200	500	2.4Ghz, 5GHz	Complex stack (TCP/IP etc.)
Bluetooth	p-p	1000 to 3000	10	~1000	2.4Ghz	Complex stack, wake 3s (latency 0.1s)
BLE (BT 4.0)	scatternet	125-2000	10	10-500	2.4Ghz	low power std, fast startup (latency 0.006)
SPI	md	1 to 1000	~10	5	wired	Simple sync 5-wire.
UART	p-p	110	~20	5	wired	Simple async 3-wire
Zigbee (xbee)	Star pp	20-250	10-100	~30	2.4Ghz, 915MHz	Low power, low speed, cost~$2, wake 15ms
Toy RC	p-p hd	0.07	10		27MHz 49MHz	Low cost

on most phones and laptops

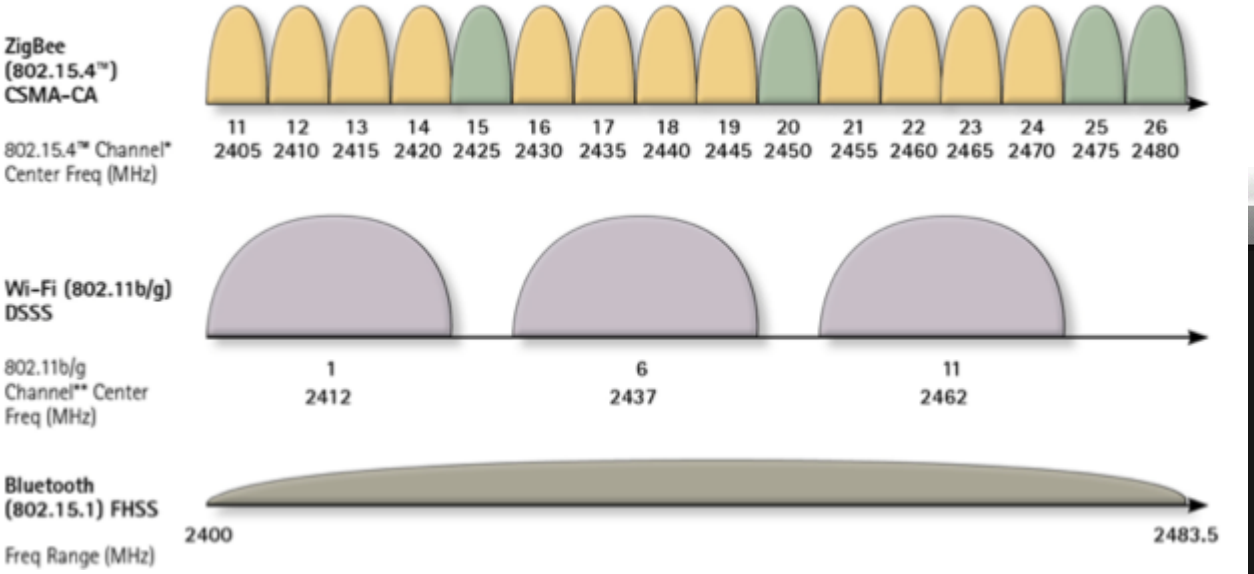
We'll cover later

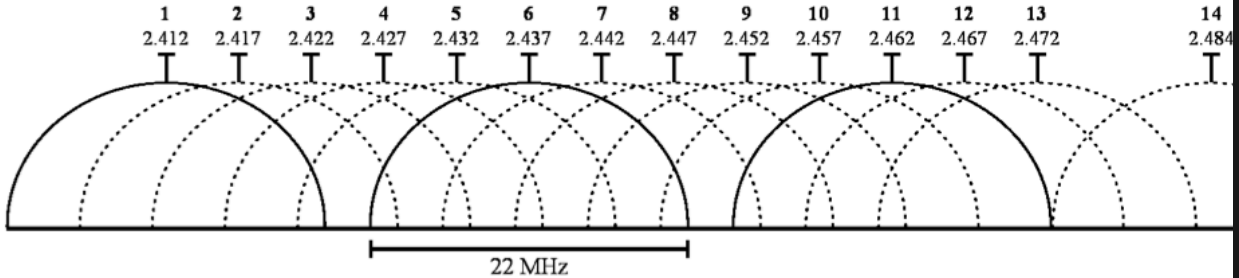
Monitor or usbprint

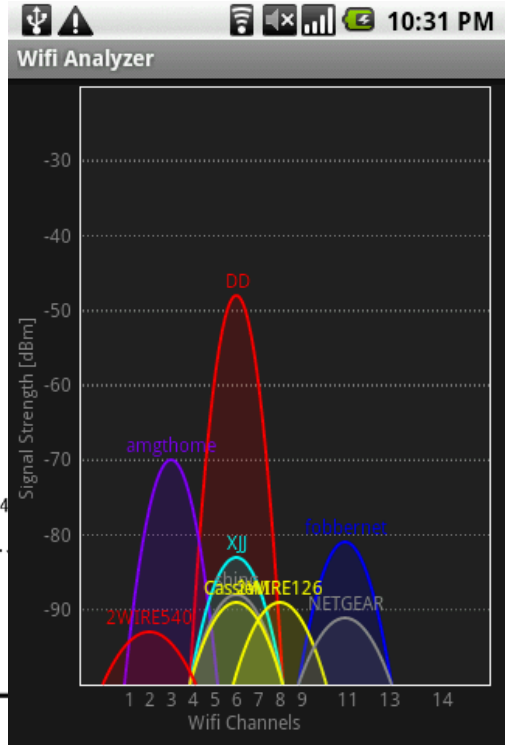
2.4GHz Freq Interference

- WiFi 802.11 2.4GHz and 5GHz
- Microwave Ovens run at 2.45GHz (but on 50% duty)

2.4GHz Freq Interference



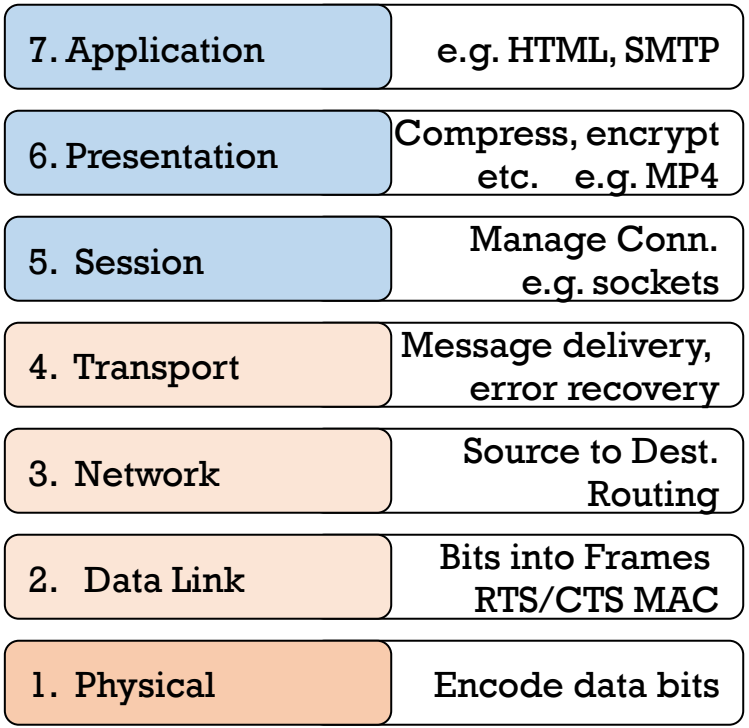




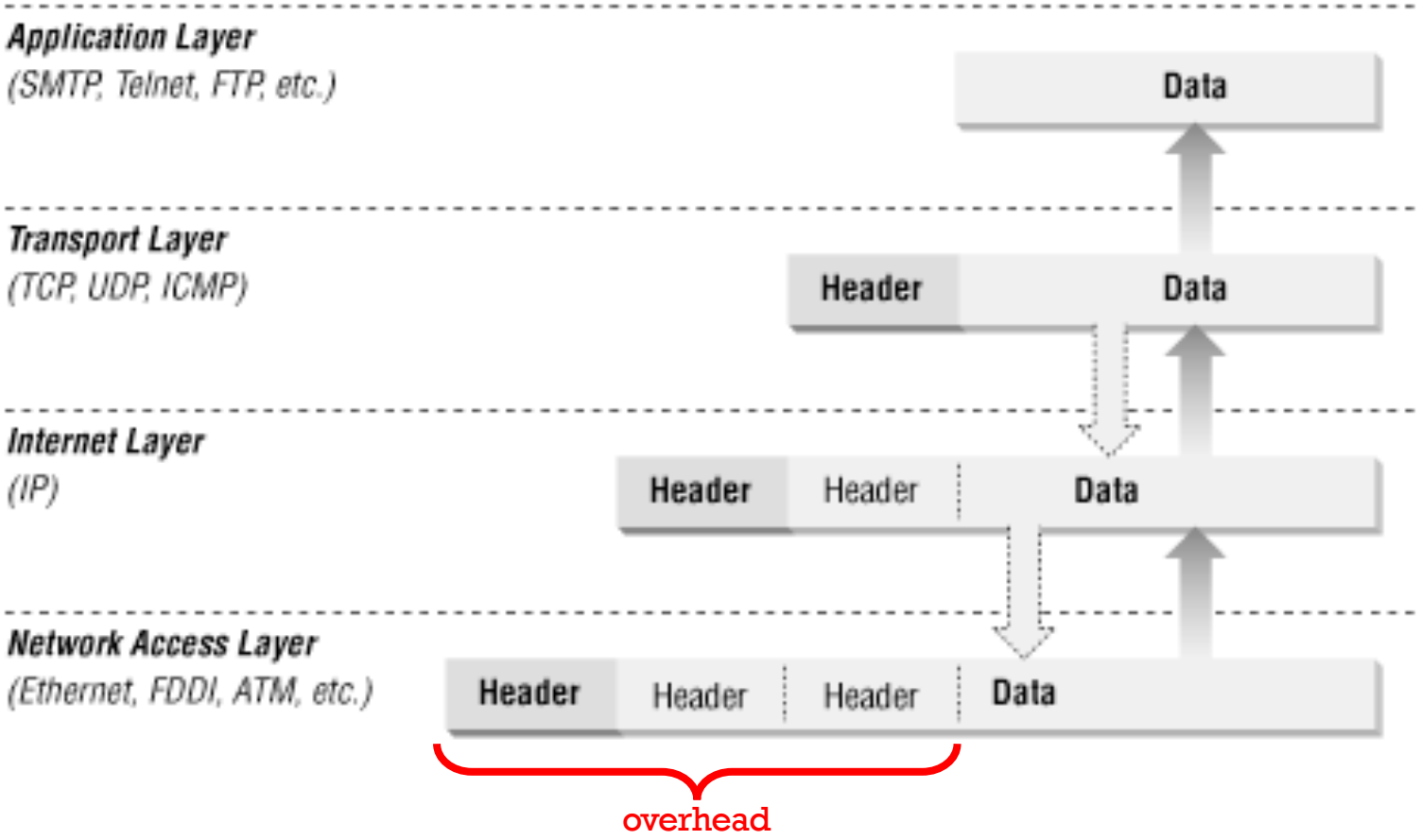
OSI – ISO 7 layer model

Open Systems Interconnection at the International Standards Organization

- Layer 1 and some 2 :
  - RS232 w/UART (async serial – old std)
  - SPI, I2C, I2S (sync serial)
  - RS485 w/UART (multi-drop)
- Layer 3
  - Internet Protocol (IP)
- Layer 4
  - Transmission Control protocol (TCP)
  - User Datagram Protocol (UDP)



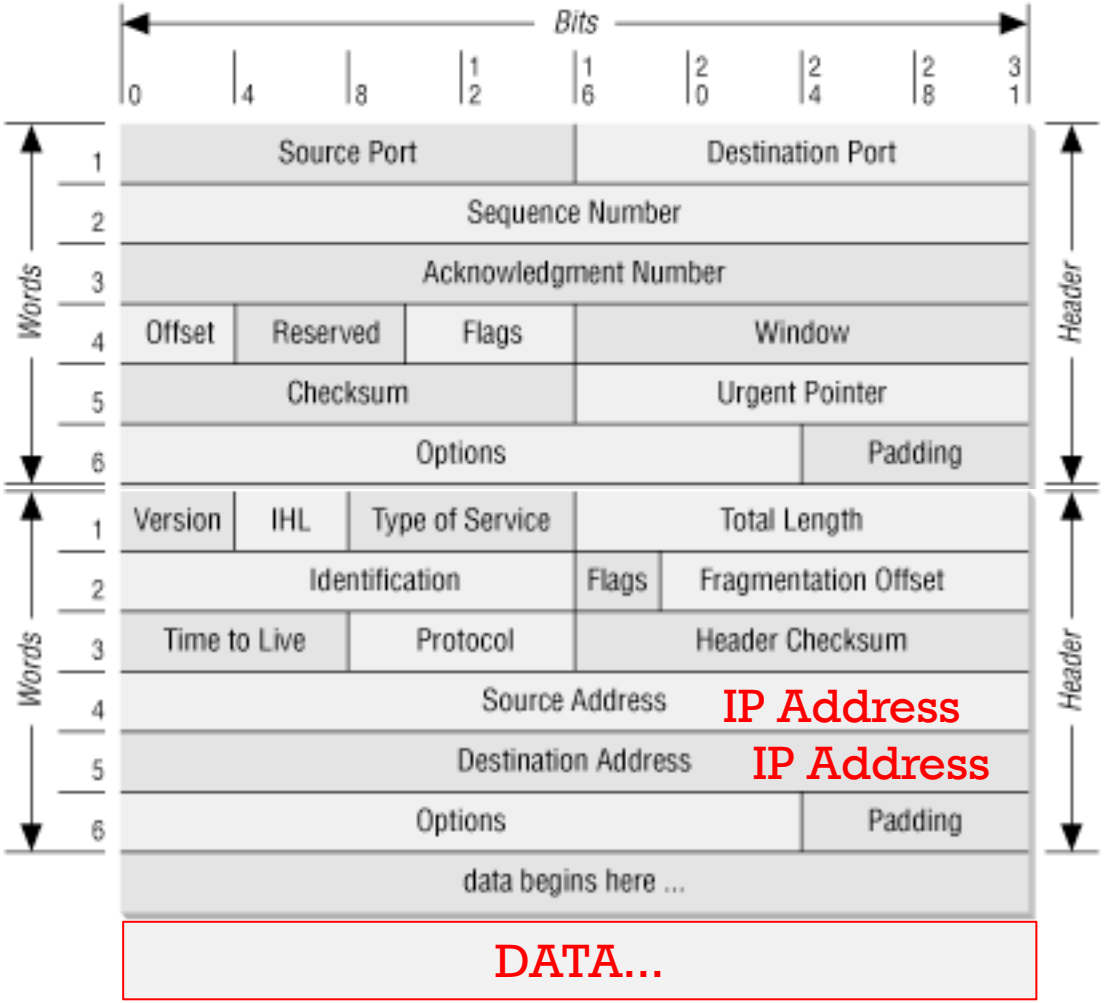
Data Encapsulation



Packets

TCP header
(20bytes + option bits)

IP header
(20bytes + option bits)



Latency vs bandwidth vs data rate

Case 1:

- Data is two bytes long
- We send 100 packets per sec.
- Assume TCP no options

Case 2:

- Data is 200 bytes long
- We send one packet per sec.
- Assume TCP no options

In chat:

Q1a: What is the data rate (amount of *data* we send per second)?

Q1b: What is the bandwidth required for each case?

Q1c: What is the latency (avg time before we receive data)?

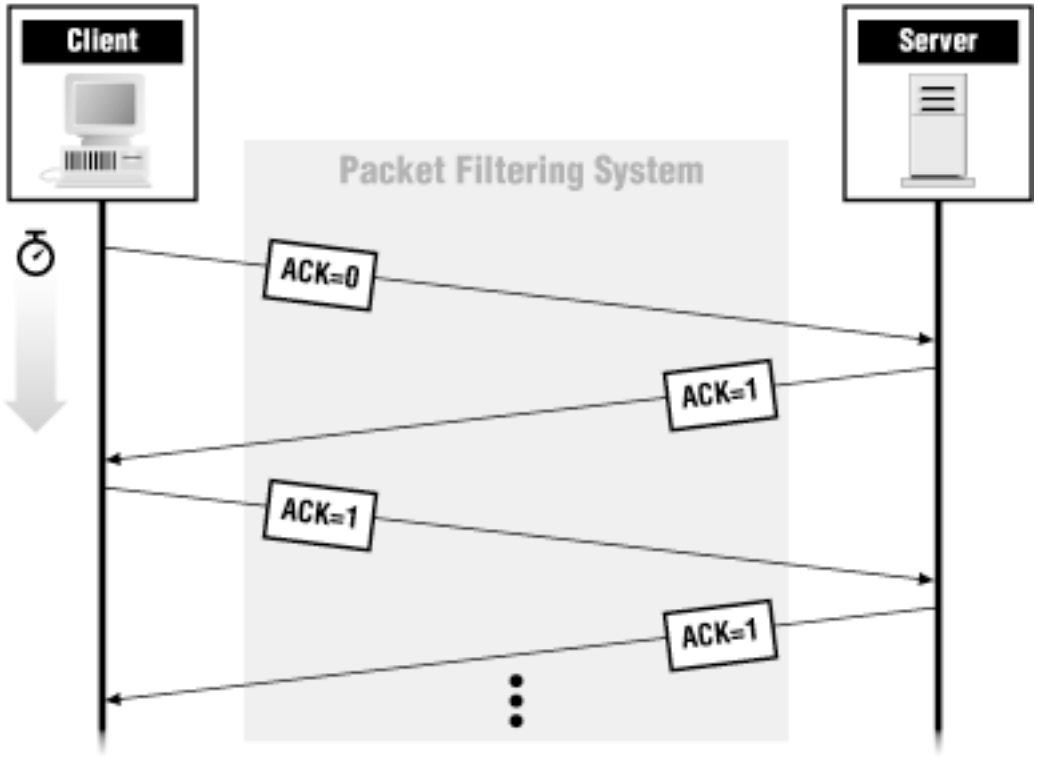
Q1d: What is the data efficiency (databits / total bits transmitted)

TCP (Transport Control Protocol)

- Receiver acknowledges packets – if not, sender resends after timout.

Reliable Transport:

- Receive in order
- Receive all data
- Receive no duplicates



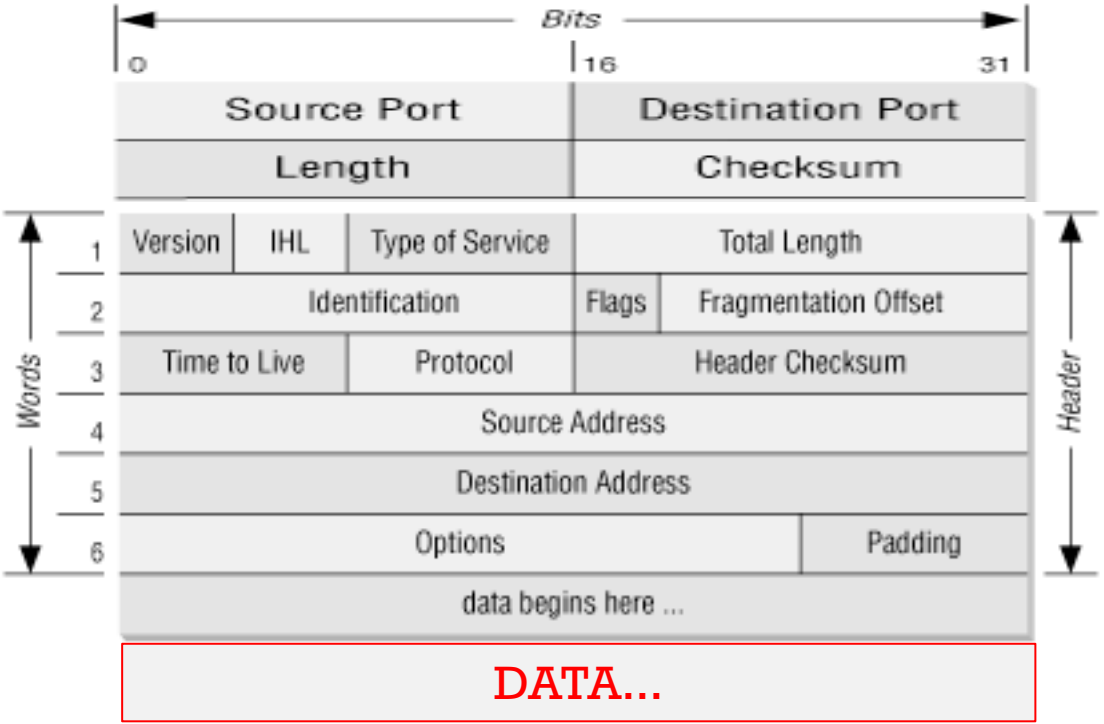
UDP Packets (~~User~~ Datagram Protocol)

Unreliable

- Much simpler (no verification, no resend)

UPD header
(8 bytes)

IP header
(20 bytes)



WiFi Quiz/summary

Q2: Which protocol would be better for streaming live video:

A) TCP or B) UDP ?

Q3: Which protocol would be better for bank transactions?

A) TCP or B) UDP ?

Q4: Which protocol do movies usually use?

A) TCP or B) UDP ?

There are other wireless protocols, we won't use in this class

02

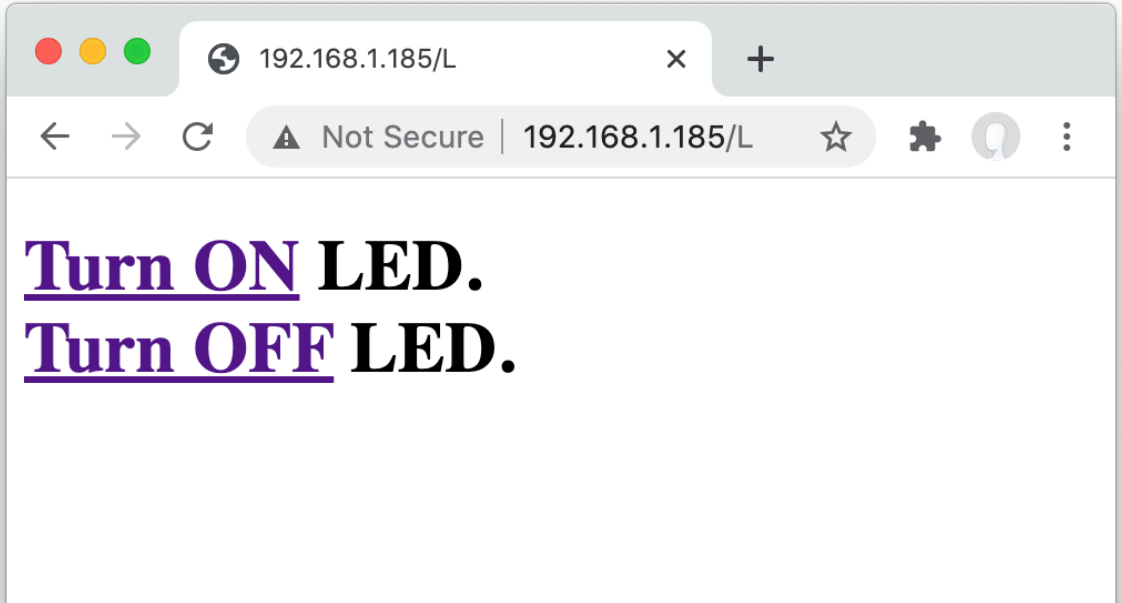
TCP-IP on ESP32

ESP32 Internet of Things (IOT)

- Control from your smart phone or laptop!
- 2.4GHz 802.11 (transmit b/g) (receive b/g/n)
- Bluetooth 4.2 (BLE) (we won't go into this in this class)
- Full ISO stack (HTML)
- Caveats:
  - Limited space and HTML code
  - Only from local net (port-forwarding)
  - http://www.wikihow.com/Set-Up-Port-Forwarding-on-a-Router
  - For Penn security reasons we will not connect to the internet.

Web server on a chip

- You can create a webpage that runs on the ESP32
  - The oscillosorta is one example.
- We'll go through a simple example with two arduino parts
  - `setup()`
  - `loop()`



Sample Web Code [part 1 of 2]: setup()

```
#include <WiFi.h>

const char* ssid = "yourhomerouterSSID";
const char* password = "yourhomepassword";

WiFiServer server(80);   // port 80 is standard for websites

void setup() {
  Serial.begin(115200);
  pinMode(21, OUTPUT); // use GPIO21
  WiFi.begin(ssid,password);
  while(WiFi.status()!= WL_CONNECTED ) {
    delay(500);
    Serial.print(".");
  }
  Serial.print("Use this URL to connect: http://");
  Serial.print(WiFi.localIP());   Serial.println("/");
  server.begin();
}
```

New include file

SSID + passwd

Std setup

Wifi setup
Uses DHCP

Sample Web Site Structure [part 2 of 2]: loop()

```
void loop(){
  WiFiClient client = server.available(); // loop until we have a client
  if (client) {
    while (client.connected()) {          // loop while connected
     if (client.available()) {            // if client has a request

      // Load the request into a String buffer one byte at a time
      // if the request is finished
      //   post the HTML code to be displayed
      //   interpret the string request and process

     }
    }
    client.stop();  // close the connection
  }
}
```

Sample Web Code: loop()

```
void loop(){
  WiFiClient client = server.available();
  if (client) {
    String currentLine = "";      // incoming data
    while (client.connected()) {
      if (client.available()) {   // if bytes to read
        char c = client.read();   // read a byte,
        if (c == '\n') {
          if (currentLine.length() == 0) {// if blank line
            client.print(body);
            break;    // exit while loop
          } else {
            if (currentLine.startsWith("GET /H "))
              digitalWrite(LEDPIN, HIGH);  // LED ON
            if (currentLine.startsWith("GET /L "))
              digitalWrite(LEDPIN, LOW);   // LED OFF
            currentLine = "";   // if new line but not blank, clear data
          }
        } else if (c != '\r') {  // if anything but a CR
          currentLine += c;      // add it to the end
        }
      }
    }
    client.stop();  // close the connection
  }
}
```

```
const char body[] PROGMEM = R"===(
 <!DOCTYPE html>
 <html><body>
 <h1>
 <a href="/H">Turn ON</a> LED.<br>
 <a href="/L">Turn OFF</a> LED.<br>
 </h1>
 </body></html>
)===";
```

Some HTML tags to play with

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>

<ul>
  <li>Bulleted list item </li>
  <li>Bulleted list item </li>
</ul>
```

```
// CSS to style the on/off buttons
<head><style>
html { font-family: Helvetica; display: inline-block;
        margin: 0px auto; text-align: center;
}
.button { background-color: #4CAF50; border: none;
        color: white; padding: 16px 40px;
        text-decoration: none; font-size: 30px;
        margin: 2px; cursor: pointer;
}
.button2 {background-color: #555555;
}
</style></head>
```

https://www.w3schools.com/tags/

ESP32 TCP Summary

- TCP can be used to make websites on ESP32
- You have c++ Arduino code on ESP32, you have web code (HTML, CSS, Javascript) that can run on websites, but are sourced on the ESP32.

03

UDP on ESP32

UDP packets

- Send from one IPaddress to another IPaddress (can broadcast)
- No error recovery, no packet retransmit, no guarantee
- Typically used for streaming applications
- Much faster than TCP (less latency, similar bandwidth)

- Use "PacketSender" to debug.
  https://en.wikipedia.org/wiki/Packet_Sender

IP Address for you to use on your ESP32 (also on piazza)

Adesoye	Konyin	192.168.1.100
Alexandrou	Andreas	192.168.1.101
Aufzien	Jacob	192.168.1.102
Banerjee	Anirban	192.168.1.103
Cao	Ruijie	192.168.1.104
Chen	Yuqi	192.168.1.105
Delattre	Andre	192.168.1.106
Du	Yi	192.168.1.107
Elms	Mason	192.168.1.108
Fedrick	Shaun	192.168.1.109
Fox	Christopher	192.168.1.110
Garza	Nicholas	192.168.1.111
Gehrke	Rafael	192.168.1.112
Gomes	Paedyn	192.168.1.113
Gong	Zachary	192.168.1.114
Grimaldi	Brian	192.168.1.115
Jurewicz	Ryan	192.168.1.116
Kim	Joah	192.168.1.117
Kulesza	Timothy	192.168.1.118
Li	Jason	192.168.1.119
Li	Peihan	192.168.1.120
Luo	Rui	192.168.1.121
Magee	Madison	192.168.1.122
Merczynski-Hait	Andrew	192.168.1.123
Nagarajan	Shravan	192.168.1.124
Ren	Joshua	192.168.1.125
Romanow	Bryan	192.168.1.126
Roth	Alexander	192.168.1.127
Sarda	Sheil	192.168.1.128
Saven	Celestina	192.168.1.129
Shi	Wentao	192.168.1.130
Shusharin	Pavel	192.168.1.131
Tamakloe	Ralph	192.168.1.132
Tippana	Sahachar	192.168.1.133
Wan	Shenshen	192.168.1.134
Wang	Joanna	192.168.1.135
Wang	Muxuan	192.168.1.136
Wang	Zhenyu	192.168.1.137
Woc	Michael	192.168.1.138
Zarin	Rachel	192.168.1.139
Zhao	Xinyuan	192.168.1.140

UDP receiver – sample code: setup() [part 1 of 2]

```
#include <WiFi.h>
#include <WiFiUdp.h>

const char* ssid     = "yourhomessid";
const char* password = "yourhomepass";

WiFiUDP UDPTestServer;
IPAddress myIPaddress(192, 168, 1, 58);

const int UDP_PACKET_SIZE = 100;
// can be up to 65535
byte packetBuffer[UDP_PACKET_SIZE+1];
```

New include files

USE YOUR ASSIGNED IP

```
void setup() {
  Serial.begin(115200);
  Serial.println();
  Serial.print("Connecting to ");  Serial.println(ssid);

  WiFi.config(myIPaddress, IPAddress(192, 168, 1, 1),
              IPAddress(255, 255, 255, 0));

  WiFi.begin(ssid, password);
  UDPTestServer.begin(2808); // any UDP port# up to 65535
                             // but higher is safer > 1023
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
}
```

Std setup

Optional. If omitted will use DHCP

Wait until connected

UDP receiver – sample code: loop() [part 2 of 2]

```
void loop() {
  handleUDPServer();
  delay(1);
}

void handleUDPServer() {
  int cb = UDPTestServer.parsePacket();
  if (cb) {
    UDPTestServer.read(packetBuffer, UDP_PACKET_SIZE);
    Serial.printf("%s\n",packetBuffer);
  }
}
```

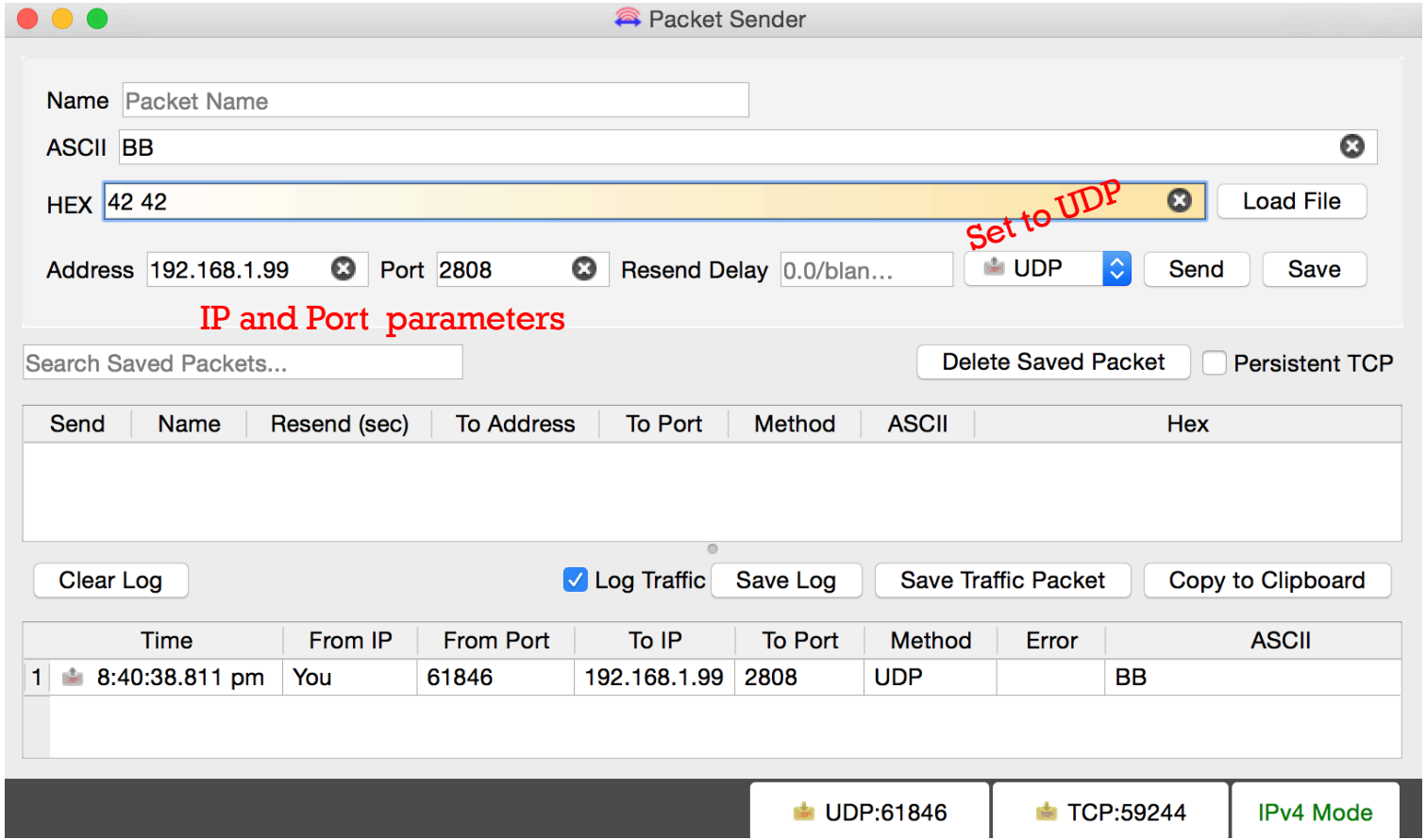
Loop checks for Packet events

Check for packet

Read packets into mydata

Do something with info in packet

PacketSender



UDP Sender: [part 1 of 2] setup()

```
#include <WiFi.h>
#include <WiFiUdp.h>

const char* ssid     = "yourhomessid";
const char* password = "yourhomepass";

WiFiUDP udp;
const int UDP_PACKET_SIZE = 100;
char udpBuffer[UDP_PACKET_SIZE];
IPAddress ipTarget(192, 168, 1, 58);
IPAddress myIP(192, 168, 1, 99);
```

```
void setup() {
  Serial.begin(115200);
  pinMode(LED_BUILTIN,OUTPUT);
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.config(myIP,
              IPAddress(192, 168, 1, 1),
              IPAddress(255, 255, 255, 0));
  WiFi.begin(ssid, password);
  udp.begin(2808);

   while(WiFi.status()!=WL_CONNECTED){
      delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected as");
  Serial.print(WiFi.localIP());
}
```

UDP Sender: [part 2 of 2] loop()

```
void fncUdpSend()
{
  // send what ever you want upto buffer size
  udp.beginPacket(ipTarget, 2808);  // send to UDPport 2808
    udp.printf("%s",udpBuffer);
  udp.endPacket();
  Serial.println(udpBuffer);
}


// send udp packet every 4 seconds
void loop() {
  delay (4000);
  strcpy(udpBuffer, "hello testing message");
  fncUdpSend();
}
```

Data Packets: String manipulations

- String is an array of bytes
- What if we want to send/receive an `int` (2 bytes) in binary?

```
void fncUdpSend(int i) // send int i
{
  udpBuffer[0] = i & 0xff; // send 1st (LSB) byte of i
  udpBuffer[1] = i>>8;     // send 2nd (MSB) byte of i
  udpBuffer[2] = 0;        // null terminate string
  udp.beginPacket(ipTarget, 2808);  // send to UDPport 2808
    udp.printf("%s",udpBuffer);
  udp.endPacket();
}

void handleUDPServer() { // receive and print int i
  int i, cb = UDPTestServer.parsePacket();
  if (cb) {
    UDPTestServer.read(packetBuffer, packetSize);
    i =
    Serial.println(i);
  }
}
```

Q5 how to reassemble int?

Difference between UPD and TCP on ESP32

- TCP has some latency on ESP32 (~60mS each packet)
- UDP has low latency (avg ~2mS)
- UDP delivery not guaranteed (~3% packet loss in clean env)

- TCP can use HTML to send (e.g. make a website, use a phone or laptop)
- UDP needs UDP sender program (custom program or 2nd ESP32)

ESP32 UDP Quiz/ Summary

- Q3 Can use use 1234 for a UDP port number?
- Q Can we send a bitmap image over UDP?

- *PacketSender* useful tool for debugging

04

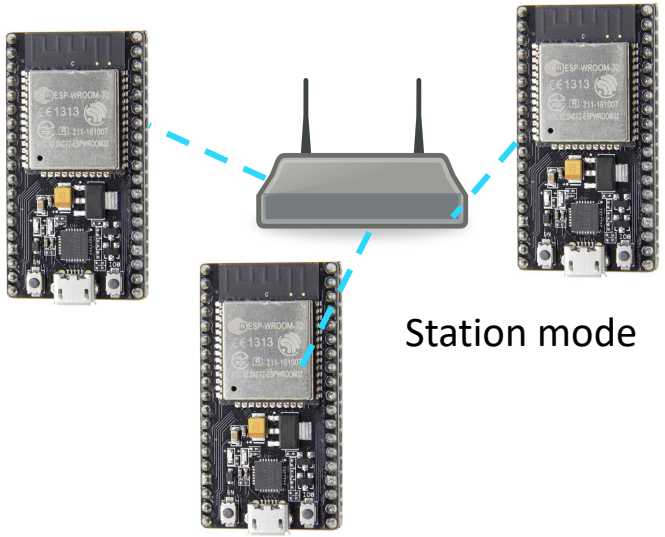
WiFi Setup options

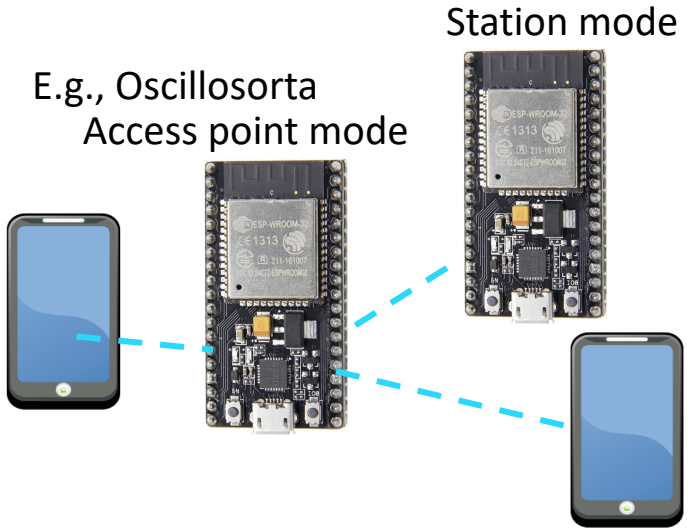
WiFi.modes()

- WiFi.mode(WIFI_STA) Station mode
  - ESP32 connects to router or other access point (input SSID etc.)

  [Need a router to connect to other things]
- WiFi.mode(WIFI_AP) Access Point Mode
  - ESP32 allows other devices to connect to it (supplies SSID etc.)

  [connect directly to ESP32 – no other router needed]
- WiFi.mode(WIFI_AP_STA) acts as both access point and station,
  - We haven't played with this – let us know if you learn about it.





IPaddresses

Private IPV4 Address Spaces

IP Address Range	number of addresses	Classful
10.0.0.0 – 10.255.255.255	16,777,216	Class A
172.16.0.0 – 172.31.255.255	1,048,576	Class B
192.168.0.0 – 192.168.255.255	65,536	Class C

- DHCP
  - IP address automatically assigned by router
  - use `WiFi.localIP()` to find out what it is
- Static IP Address in station mode.
  - use:

```
WiFi.config(IPAddress(192, 168, 1, 99),    // local IP
            IPAddress(192, 168, 1, 1),     // Gateway
            IPAddress(255, 255, 255, 0));  // subnet mask
```

  - If fixed IP address is not specified, defaults to DHCP

Access Point Mode vs Station mode

```
void setup() {   // for Station mode – need a router
  WiFi.config(IPAddress(192, 168, 1, 2), IPAddress(192, 168, 1, 2),
              IPAddress(255, 255, 255, 0));

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  server.begin();
}
```

Omit if you want no password

```
void setup() {   // for AP mode – don't need a router
  WiFi.mode(WIFI_AP);
  WiFi.softAP(ssid, password);
  // may need to add delay(100); to make this work
  WiFi.softAPConfig(IPAddress(192, 168, 1, 2), IPAddress(192, 168, 1, 2),
                    IPAddress(255, 255, 255, 0));
  server.begin();
}
```

Omit if you want to use DHCP, but remember to print `WiFi.localIP()`

Web server on a chip without router (DHCP)

```
#include <WiFi.h>
const char* ssid = "myuniqueSSID"; // come up with your own personal SSID
const char* password = "something";

WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  Serial.print("Access point "); Serial.print(ssid);
  WiFi.softAP( ssid, password) ; // omit password if you want it open.
  IPAddress IP = WiFi.softAPIP();
  Serial.print(" AP IP address"); Serial.println(IP);

  server.begin();
}
```

Web server on a chip without router (setting IP)

```
#include <WiFi.h>
const char* ssid = "myuniqueSSID"; // come up with your own personal SSID


WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  Serial.print("Access point "); Serial.print(ssid);
  WiFi.softAP( ssid) ;  // example of open access point (no password)
  delay(100); // hack for AP_START to finish
  IPAddress Ip(192, 168, 1, 2);
  IPAddress NMask(255, 255, 255, 0);
  WiFi.softAPIP(Ip, Ip, NMask);
  Serial.print(" AP IP address"); Serial.println(IP);

  server.begin();
}
```

Debugging WiFi on ESP32

- Using AP mode on laptop: don't forget to set the proper SSID on your network. (e.g. many laptops will automatically go back to AirPennNet after a network goes down)
- Using STA mode on laptop is more convenient because you don't lose access to the web for info.
- Debug websites with chrome and "*developer tools*"
- Debug a receiver with *Packetsender*
- Debug a sender with a debugged receiver… Maybe use *Wireshark* a comprehensive opensource free packet analyzer tool.

WiFi Setup Summary

- ESP32 can be a station with router or its own access point.
- Set up static with wifi.config. If you omit this line, DHCP will be used.

05

ESP32 LEDC

For ESP32 instead of `analogWrite()` use LEDC routines

- LEDC has 16 channels, not assigned to pins
  - Need to attach them to a pin that drives an LED or PWM output.
  - Prefer ledc channels 0-7 which use ESP32 high speed mode.
- Can set PWM frequency and resolution, within limits
  - Frequency * Resolution must be < 80 MHz
    - Example: for PWM resolution of 1000, max PWM freq is 80Khz
  - Frequency * Resolution must be > 1 KHz
    - Example: too fast for 1 Hz and resolution of 256
- For reference `analogWrite` on Atmega32 (i.e., Arduino Uno) has freq 490Hz and 8-bit resolution (freq*res = 3920)

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/ledc.html

LEDC functions

```
#define LEDC_CHANNEL       0 // use first of 16
#define LEDC_RESOLUTION_BITS  13
#define LEDC_RESOLUTION  ((1<<LEDC_RESOLUTION_BITS)-1)
#define LEDC_FREQ_HZ     5000
#define LED_PIN           2

// can set syntax to be like analogWrite() with input[ 0 - valueMax ]
void ledcAnalogWrite(uint8_t channel, uint32_t value, uint32_t valueMax = 255) {
  uint32_t duty =  LEDC_RESOLUTION * min(value, valueMax) / valueMax;
  ledcWrite(channel, duty);  // write duty to LEDC
}

void setup() {  // Setup timer and attach timer to a led pin
  ledcSetup(LEDC_CHANNEL, LEDC_FREQ_HZ, LEDC_RESOLUTION_BITS);
  ledcAttachPin(LED_PIN, LEDC_CHANNEL);
}

void loop() {
  static int brightness = 0;  // how bright the LED is
  static int fadeAmount = 5;  // how much to fade by

  ledcAnalogWrite(LEDC_CHANNEL, brightness);
  brightness = brightness + fadeAmount;
  if (brightness <= 0 || brightness >= 255)
    fadeAmount = -fadeAmount;  // reverse the direction
  delay(30);
}
```

Unsigned 32 bit integer (explicit size)

Q6a We need a 60Hz PWM frequency with a resolution of a million. Can we get that with LEDC?

Q6b What Arduino setup code will generate this ?

[use channel 0 and pin 21. Use 20 bits for one million]

Using Libraries with Arduino

- Using libraries with Arduino
- Libraries (not .h files) are the way to modularize your source code or use code that others have made without having to read the source code.
- You don't need to do this. But if you want to, there's a nice tutorial:
  - https://www.arduino.cc/en/Hacking/LibraryTutorial
  - It also explains some rudimentary aspects of c++ vs c.
- Using the arduino libraries in general
  - https://www.arduino.cc/en/Guide/Libraries
  - https://www.arduino.cc/en/Reference/Libraries

Link References

- Espressif IoT Development Framework (esp-idf)
  https://docs.espressif.com/projects/esp-idf/en/latest/index.html
- ESP32 tutorials (including wifi)
  https://www.instructables.com/id/IOT-Made-Simple-Playing-With-the-ESP32-on-Arduino-/
- Accesspoint
  https://randomnerdtutorials.com/esp32-access-point-ap-web-server/
- Arduino on ESP32 examples
  https://github.com/espressif/arduino-esp32/tree/master/libraries/ESP32/examples
- Tons of ESP32 links: http://esp32.net/

Other references

- TCP/IP and UDP more in depth reference
  - https://hpbn.co/building-blocks-of-udp/
  - https://hpbn.co/building-blocks-of-tcp/
- HTML, CSS, javascript and other web programming
  - https://www.w3schools.com/html/default.asp
- LEDC
  - https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/ledc.html C++ Espressif reference doc
  - https://randomnerdtutorials.com/esp32-pwm-arduino-ide/ Arduino tutorial
  - https://github.com/espressif/arduino-esp32/blob/master/cores/esp32/esp32-hal-ledc.c Arduino code

Answer in Chat

Answer how you feel about each topic below with:

1. I don't understand this topic at all
2. I don't know now, but know what to do to get by
3. I understand some, but expect to get the rest later
4. I understand completely already

A. TCP vs UDP

B. Using ESP32 for web applications

C. LEDC