Lecture 17

Lab 4 and HTML

Agenda

01. Lab 4 WiFi Driven Vehicle

01a. Lab 4 Mobile Base Options

02. WiFi Modes

02a. HTML Basics

02b. HTML510 (MEAM510 support code)

Button and Slider examples for Lab 4

Stuff

- Pennsylvania voting, Nov 2 (tomorrow) Polls open 7AM-8PM.
- COVID Dashboard say 18 new student covid cases last week - a few more than previous week.
- MEAM516 / IPD516 Advanced Mechatronics in Reactive Spaces will be offered this Spring.
  - Will feature a live performance with Mendelssohn Chorus of Philadelphia in May at the Fillmore Theater
  - https://mcchorus.org/

01

Lab 4: WiFi driven Vehicle

Lab 4.1 (individual work)

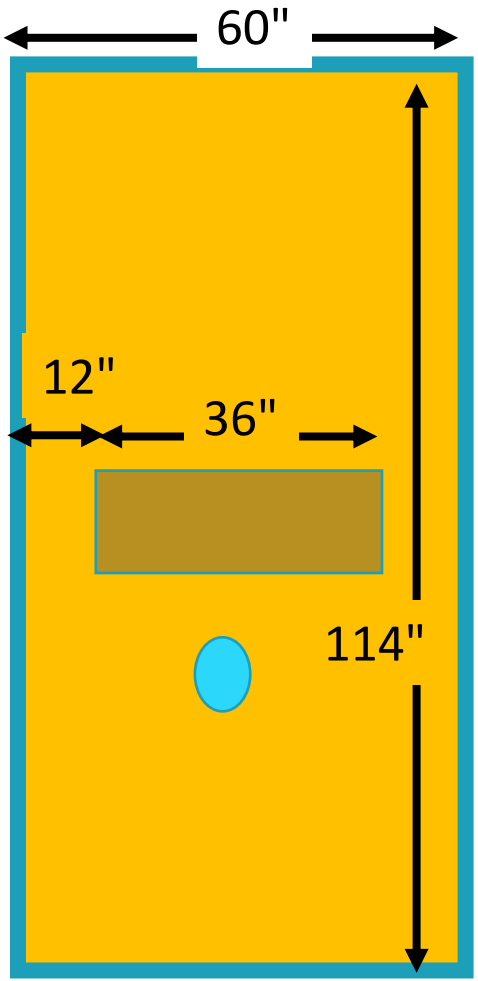
- ESP32 Digital and Analog IO
- Using a webpage to control an LED
- Using a webpage to control a motor with an h-bridge
- Forming a team so part of 4.1 – sign up on canvas
  - 4-member teams will have additional work over 3-member teams. What is extra credit for 3-member teams will be required (non-EC) for 4 member teams.
  - Limited 4-member slots
  - Individuals will be assigned to teams by Thursday.
- For the individual tasks (4.1) and you submit video, it must be your work and your video. Even if you work with someone else you must do your own work.

Lab 4.2 Team lab

- Build mobile base.
- Circle the center wall 3 times.

Constraints:

- You may use Teensy or ESP32 for controlling vehicle.
- You must use ESP32 for WiFi interface. (note using one processor is easier than two).
- Your vehicle can be up to 12" x 12" x 12"
- You may use any motor or servo you wish.
- You are encouraged to purchase items for this lab from your allocation



Purchasing Allocation

- Each student may purchase upto $50 worth of items from the following:

- Digikey.com: Electronics
- Mcmaster.com: Mechanical
- Adafruit: DIY electronics
- Sparkfun: DIY electronics
- Polulu: Motors, hubs
- Amazon.com: Everything

Tentative Lab 4 Schedule

November

Sun	Mon	Tues	Wed	Thurs	Fri	Sat
31	**Today** Lab 4 Intro				4.2.1 due Review in Rec.	
7						
14	Lab 4.due **Final Project Starts**					
21			(Fri Sched) Recitation	Thanksgiving	Thanksgiving	
28						

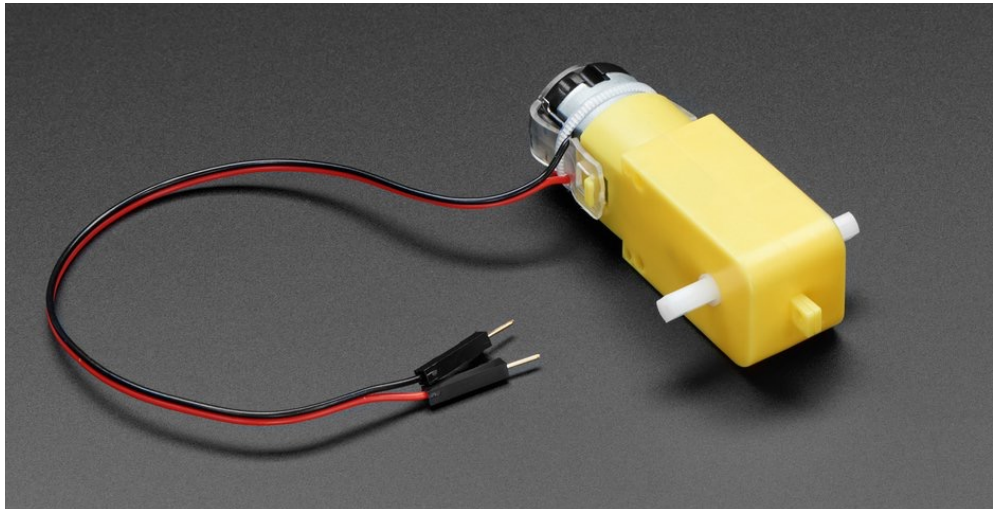
Purchasing Allocation

- Track the costs of things you purchase (you will submit your purchases in the final report)
  - Include cost of item
  - Ignore shipping costs in your cost estimation
- Details in *Purchasing for MEAM510.pdf* on Canvas files
  - Fill out purchase request form… Include you phone#, team# etc…
  - Send purchase request to yim@seas.upenn.edu
- Receiving parts may take 1 to 2 weeks (sometimes longer).
  - Purchase takes 3 business days to place order (currently)
  - Shipping takes 2-5 days to deliver.

Supplied DC Motor (x2)

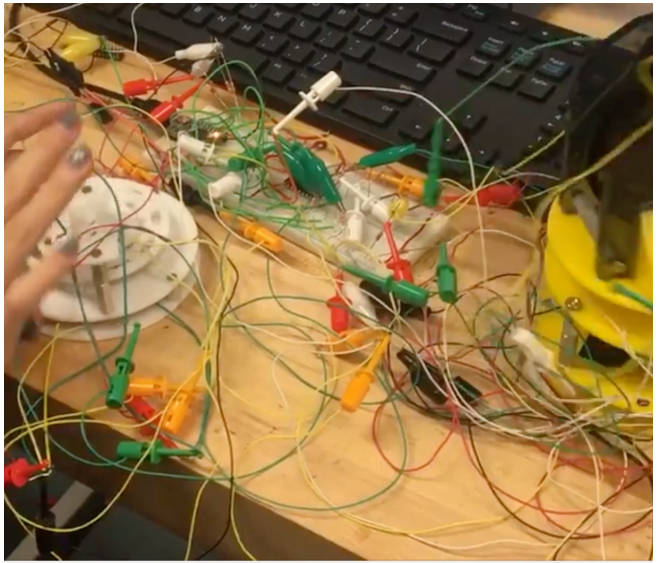
Motor specs

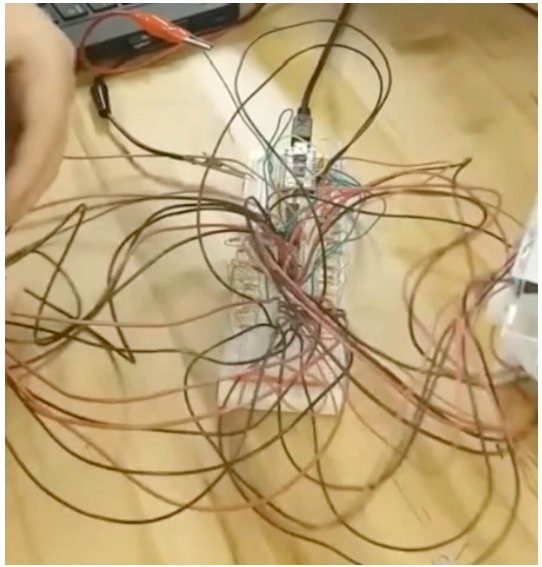
- Speed: 90+/- 10% RPM @ 3V, 200+/- 10% RPM @ 6V
- Continuous No-Load Current: 150mA +/- 10%
- Rated Voltage: 3~6V
- Torque: 0.15Nm ~0.60Nm
- Stall Torque: 0.8kg.cm @ 6V
- Gear Ratio: 1:48



Wiring robustness…

- Did you experience moving a device and have it stop working?
- What can we do about this?

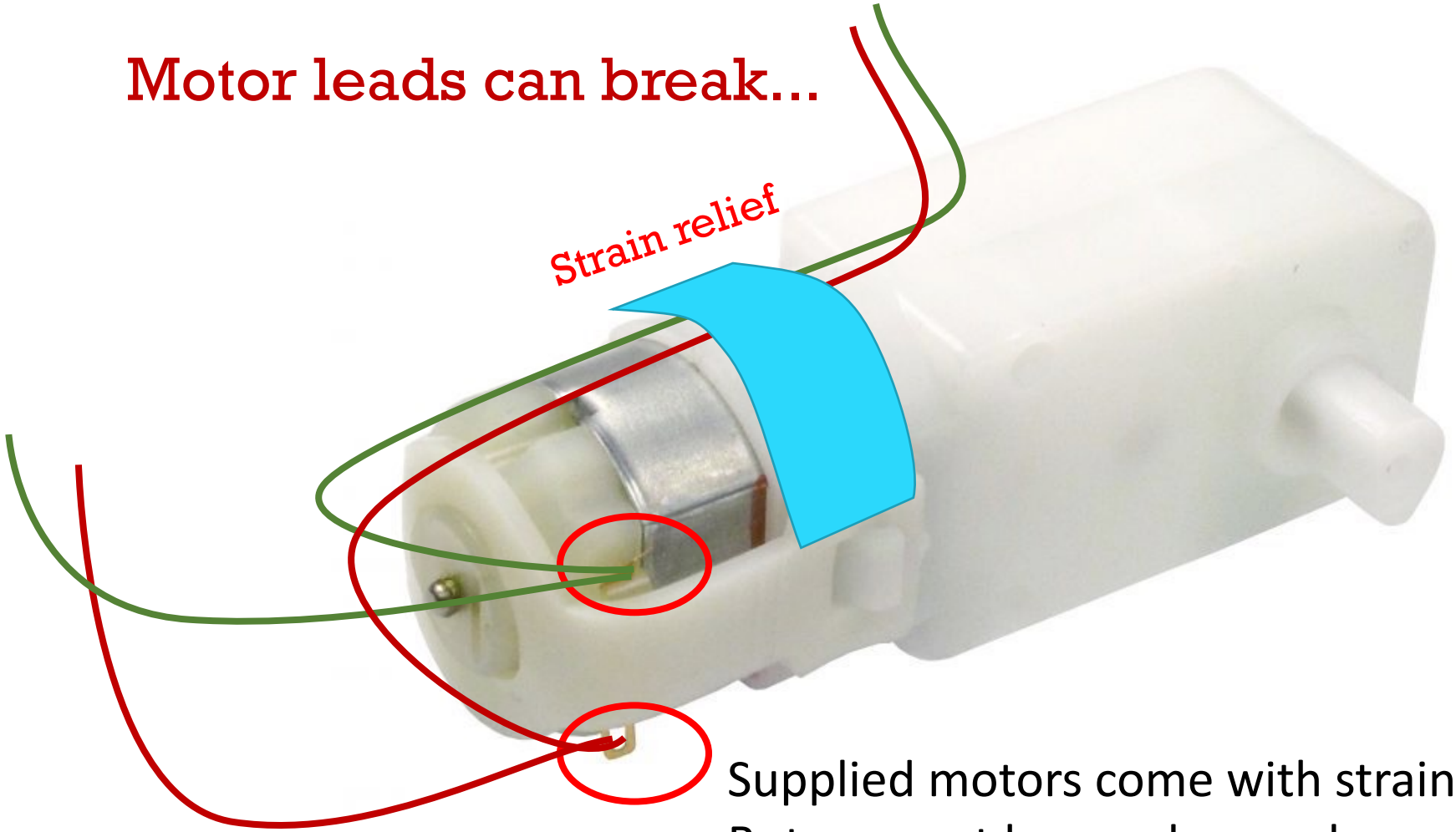




Connectors and cabling

- Most electromechanical failures occur at connections/connectors.
- If you need to have long wires, what can we do?

Motor leads can break…

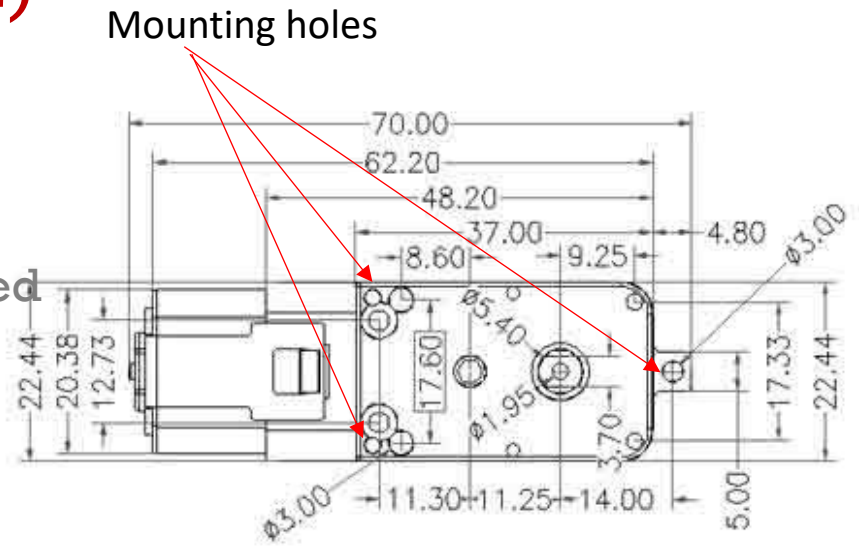


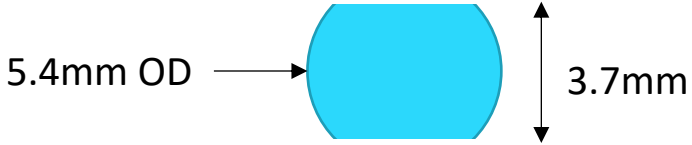
Supplied motors come with strain relief
But may not be good enough

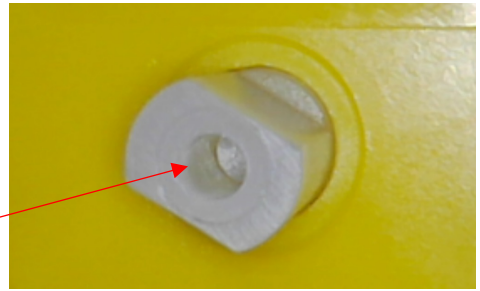
Supplied DC Motor (x2)

https://www.adafruit.com/product/3777

- Mounting
  - 3mm x3 mounting holes
  - Using these holes is encouraged
- Shaft mount – has 2 flats
  - Can laser cut this shape hole





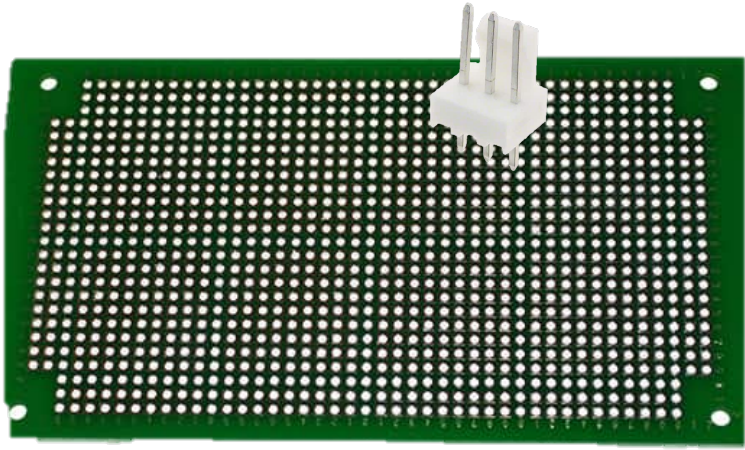


Smooth hole
Can use #2 or #3 wood screw

Lab 4 Implementation details

Reliability will be more important

- Cleaner wiring.
- Optional soldering and connectors:
  - Use chip sockets for ESP32 and motor-drivers (don't solder directly to pins).
  - Use connectors for wires
  - Soldering takes time!



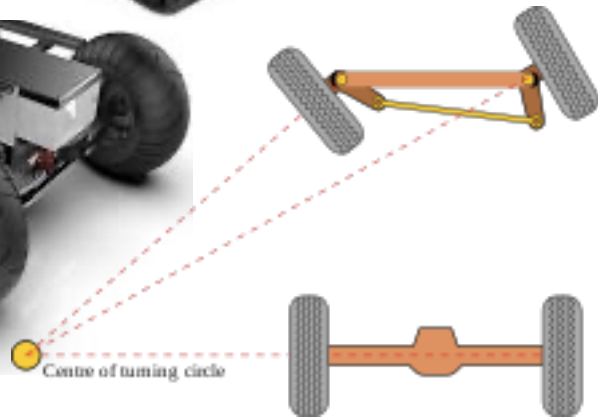
01a

Lab 4: Mobile Base Options

Mobile Bases

- Skid steer
  - Easy to make, hard to steer

- Ackerman steering
  - Hard to make, constrained motion
  - Not recommended for this class



- Holonomic
  - Expensive, high mobility

Mobile Bases

- Easiest Differential Drive
  - Castor or skid in front
  - Easy to make
  - Lower cost
  - Recommended for simplicity
  - Compared to skid steer:
    - Similar control for steering
    - Less motor power required to turn

02

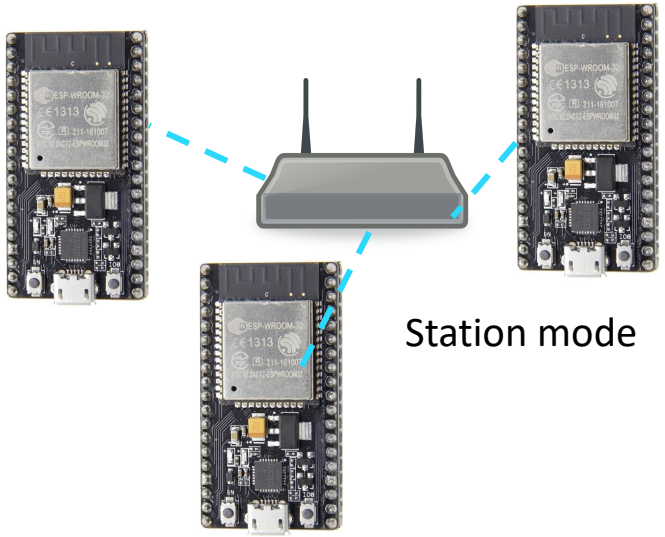
WiFi Modes

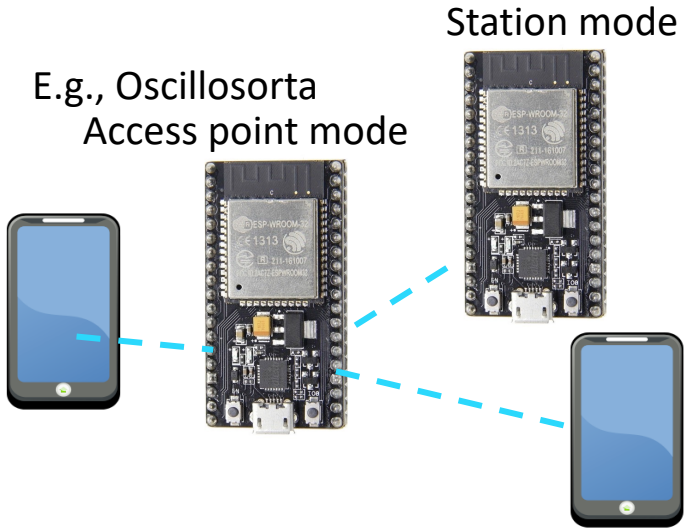
WiFi.modes()

- 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.
- ESP-NOW – Proprietary ESP32 to ESP32 mode
  - More on this in future lectures





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);
  }
}
```

Omit if you want no password

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

```
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));
}
```

Web server on a chip Access Point mode 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 Access Point mode 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)
  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();
}
```

Web server on a chip Station Mode **with a router** (using DHCP)

```
#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);
  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();
}
```

Wifi setup
Uses DHCP

Web server on a chip Station Mode **with a router** (setting IP)

```
#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);
  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);
    Serial.print(".");
  }
  server.begin();
}
```

SSID + passwd

Std setup

ESP-NOW (extra credit mode – more later)

- Simple setup using MAC address
- Espressif Proprietary (only works with ESP)
- Very fast to connect and low latency. Great for sending small messages (low overhead).
- Not guaranteed like UDP, but can receive ACK if message fails.
- Can network up to 20 ESPs (10 if using encrypted modes).
- No router, no DHCP, no HTTP overhead
- Maximum packet size is 250 bytes
- Likely still slower than laptop or phone wifi.

WiFi Setup Quiz

Q1. To use WiFI to connect two ESP32s in Station mode you must

A. have a router

B. have set IP address

C. set to DHCP

Q2. If you don't set explicitly set the IP Address in Station mode,

A.The ESP32 will choose a random one

B. The ESP32 will not connect

C. The system will default to 192.161.1.1

WiFi Mode Summary

To setup WiFi you must choose a WiFi mode and a method of choosing IPAddresses.

- There are two WiFi modes:

1. STATION
2. ACCESS POINT

- There are two methods of getting an IPAddress

1. Explicitly set one with wifi.config

```
WiFi.config(IPAddress(192, 168, 1, 2), IPAddress(192, 168, 1, 2),
                                       IPAddress(255, 255, 255, 0));
```

2. Use DHCP. Find out what the ESP chooses with

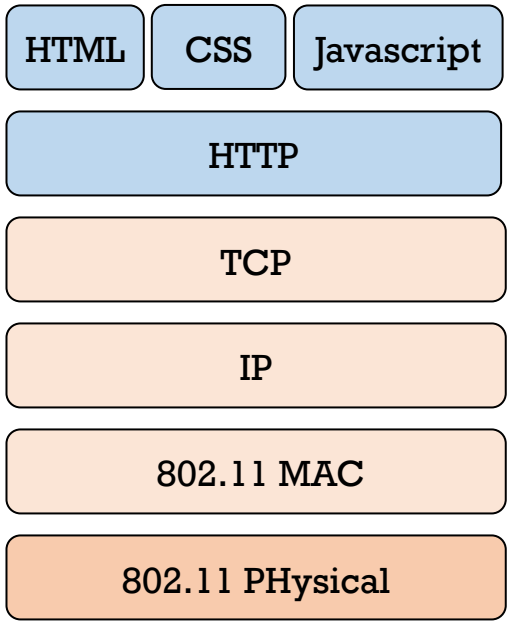
```
Serial.print(WiFi.localIP());
```

02a

HTML bare bones for 510

Website construction (HTML)

- HyperText Mark-up Language
- Webpages use different code. We will use
  - HTML (content)
  - Javascript and AJAX (interactivity)
  - CSS (style)
- All sit ontop of HTTP protocol for messaging
  - Human readable code.
- HTTP uses TCP/IP for transporting messages



Website components

- HTML: defines the content of a Web page. Elements have <tags>
  - Ex: <body> <H1> Title </H1> </body>
- CSS: Cascading Style Sheets, changes the appearance (things like color, fonts etc.)
  - Ex: <style> H1 { border: none; background-color: #eee; } </style>
- Javascript: adds programmability, interactivity.
  - Ex: <script>

```
function changeLabel() {
    document.getElementById("label").innerHTML =  "sometext";
}
```

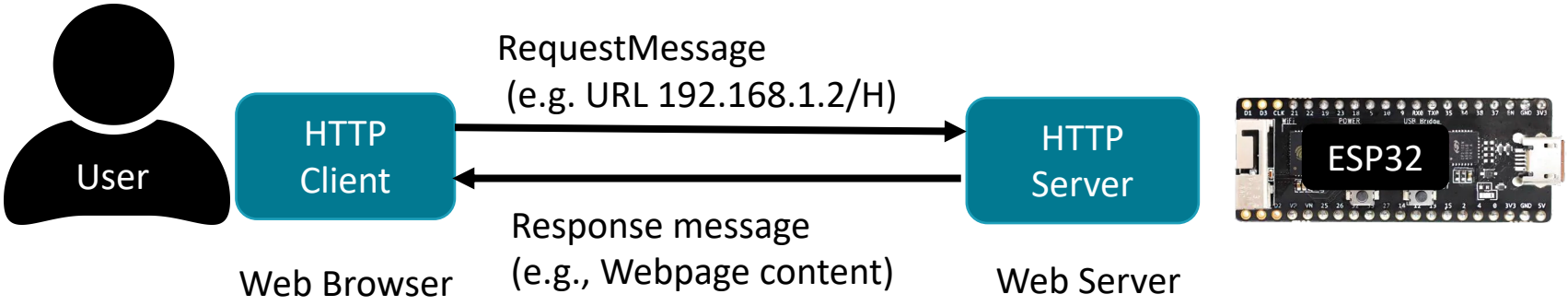
</script>

HTTP Client – Server Model

- Client (e.g., webbrowser) intiates request.
- Server (e.g., ESP32) responds.

The server cannot initiate messages.

- HTTP/1.1 connections open on request, respond, then close.



02b

HTML510

MEAM510 Web support code for ESP32

Recommended MEAM510 ESP32 Web Structure

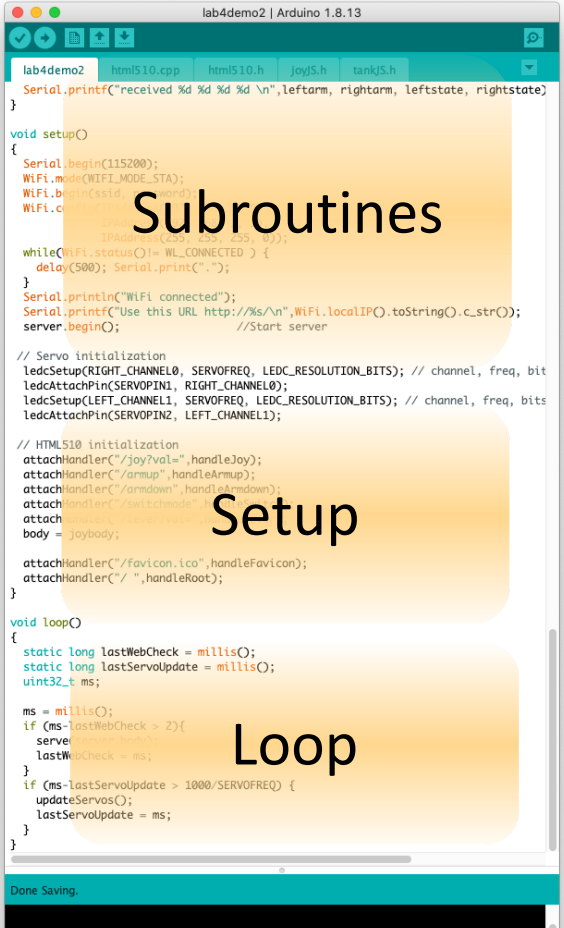
Your_Arduino.ino • Your Arduino code

html510.cpp • C++ code linking ESP32 and HTML

html510.h • C++ header support for ESP32 and HMTL

Your_webpage.h • Your HTML and Javascript code

Recommended MEAM510 ESP32 Web Structure



Your-html-code.h

```
const char body[] PROGMEM = R"===(
<!DOCTYPE html>
   <html>
   <body>

      HTML code here

   </body>
   <script>

      Javascript code here

   </script>
   </html>
)===";
```

Allows us to make a large literal String including white space

Web Page Code

Allows us to make a large literal String including white space

HTML510.cpp and HTML510.h

- Code that allows easy interface between ESP32 Arduino code and a webpage in the MEAM510 context (robot-control).

```
void attachHandler(String key, void (*handler)());
void serve(WiFiServer &server, const char *);
void sendhtml(String data) ;
void sendplain(String data);
int getVal();
```

- Code hastily written for this class.
- Be careful about re-using, re-sharing, don't post on CHEGG!

Previous 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();
}
```

loop() [part 1 of 2] checks for 3 things

```
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();
        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
  }
}
```

1) If "/" send the webpage code

2) If "/H" turn on LED and page

3) If "/L" turn off LED and page

```
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>
)===";
```

New Version using HTML510

setup() and loop()

```
#include "body.h"
#include "html510.h"
HTML510Server h(80);

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("/");

  h.begin();
  h.attachHandler("/H",handleH);
  h.attachHandler("/L",handleL);
  h.attachHandler("/ ",handleRoot);
}

void loop(){
  h.serve();
  delay(10);
}
```

Q3. Circle what's new here?

Routines from HTML510

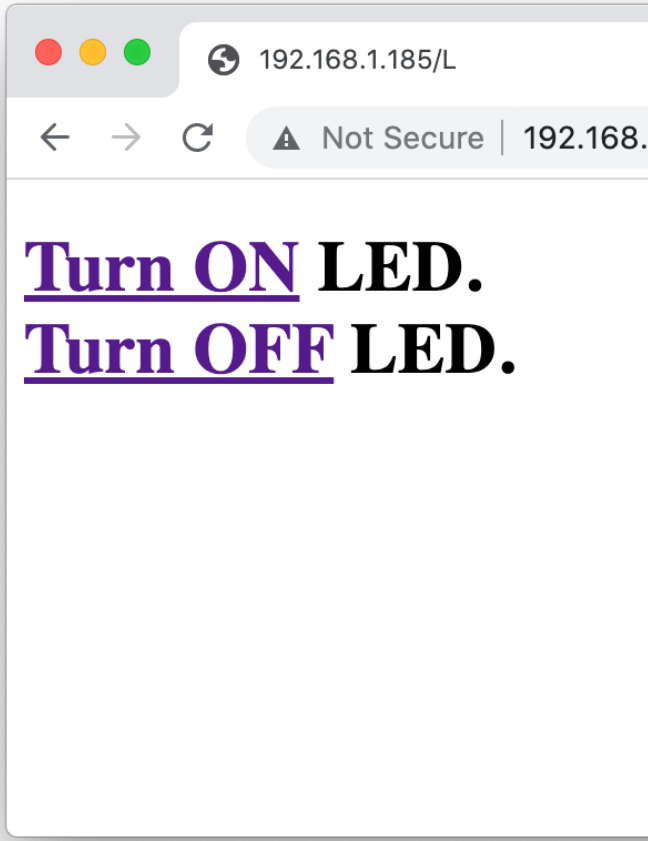
Previous version had client processing code here (the big while() loop)

Your-html-code.h

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

   </script>
   </html>
)===";
```

Web Page Code



Client Request Handler Subroutines

```
#include "html510.h"
HTM

vo

  void handleRoot(){
    h.sendhtml(body);
  }

  void handleH(){
    digitalWrite(LEDPIN, HIGH);  // LED ON
    h.sendhtml(body);
  }
  void handleL(){
    digitalWrite(LEDPIN, LOW);  // LED ON
    h.sendhtml(body);
  }

  h.
  h.attachHandler("/H",handleH);
  h.attachHandler("/L",handleL);
  h.attachHandler("/ ",handleRoot);
}

void loop(){
  h.serve();
  delay(10);
}
```

,168,1,1),

<a href> hyperlink to jump to a new webpage

```
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>
)===";
```

Client Request Handlers

```
void handleRoot(){
  h.sendhtml(body);
}

void handleH(){
  digitalWrite(LEDPIN, HIGH);  // LED ON
  h.sendhtml(body);
}
void handleL(){
  digitalWrite(LEDPIN, LOW);  // LED ON
  h.sendhtml(body);
}
```

/H and /L both load the same webpage (*body*)

Add a javascript button

```
void handleRoot(){
  h.sendhtml(body);
}

void handleH(){
  digitalWrite(LEDPIN, HIGH);  // LED ON
  h.sendhtml(body);
}
void handleL(){
  digitalWrite(LEDPIN, LOW);  // LED ON
  h.sendhtml(body);
}
```

. . . Add this to body:

```
<button type="button"
onclick="hit()"> mybutton </button>
```

. . . Add this to script:

```
function hit() {
  var xhttp = new XMLHttpRequest();
  xhttp.open("GET", "hit", true);
  xhttp.send();
}
```

. . . Add this to setup():

```
attachHandler("/hit ",handleHit);
```

```
void handleHit(){
  static int toggle;
  if (++toggle%2)  digitalWrite(LEDPIN,HIGH);
  else digitalWrite(LEDPIN,LOW);
  h.sendplain(""); // acknowledge
}
```

Web510-button.h

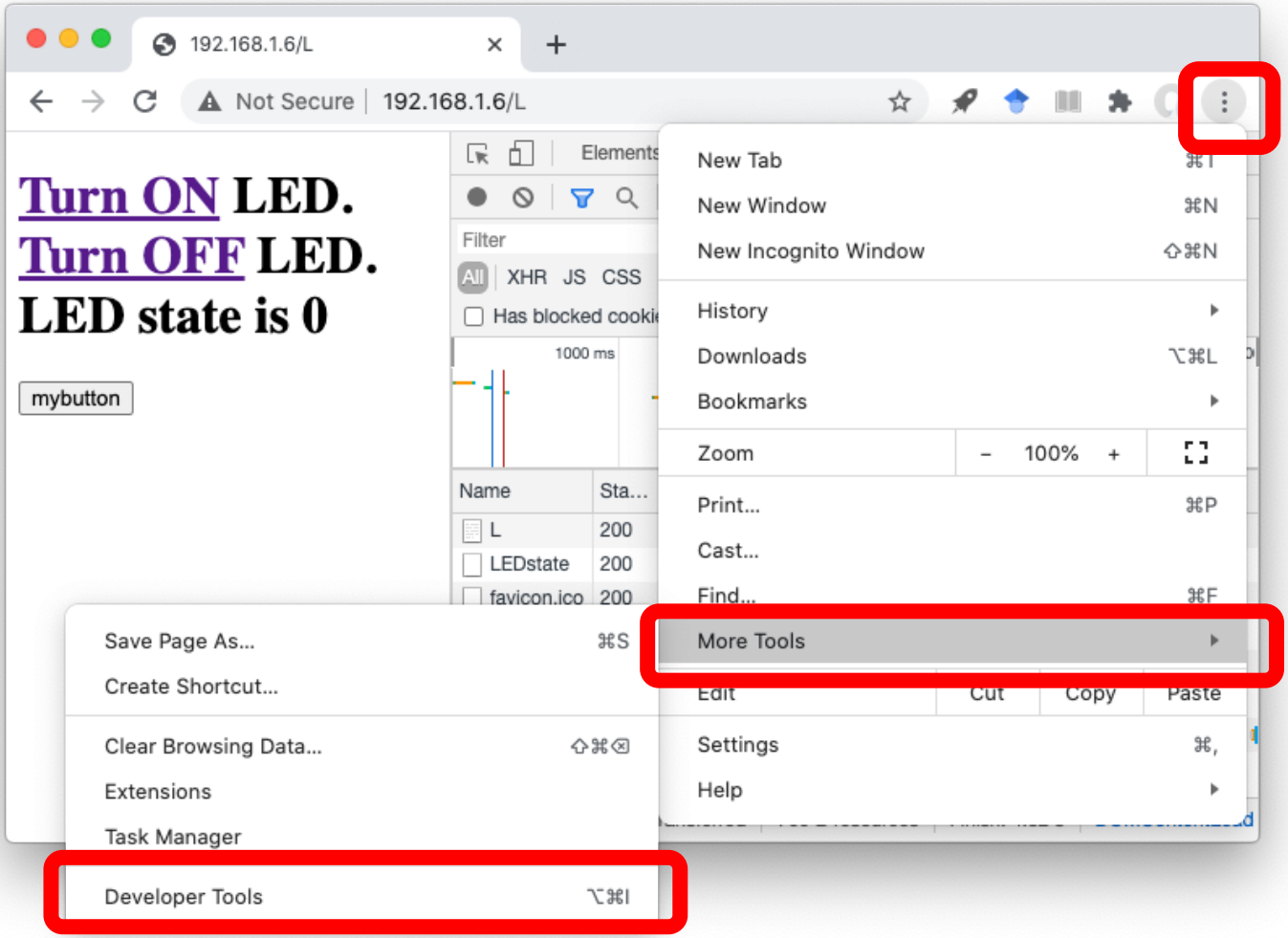
```
const char body[] PROGMEM = R"===(
<!DOCTYPE html>
<html>
  <body>
    <a href="/H">Turn ON</a> LED.<br>
    <a href="/L">Turn OFF</a> LED.<br>
    <button type="button" onclick="hit()"> mybutton </button>
  </body>

  <script>
    function hit() {
     var xhttp = new XMLHttpRequest();
     xhttp.open("GET", "hit", true);
     xhttp.send();
    }
  </script>

</html>
)===";
```

Chrome Developer Tools

Very useful for debugging HTML and javascript code



192.168.1.6/L

Not Secure | 192.168.1.6/L

Turn ON LED.
Turn OFF LED.
mybutton

Change webpage display

```
void handleH(){
  digitalWrite(LEDPIN, HIGH);
  sendhtml(body);
}

void handleL(){
  digitalWrite(LEDPIN, LOW);
  sendhtml(body);
}
void handleHit(){
  static int toggle;
  if (++toggle%2)    digitalWrite(LEDPIN,HIGH);
  else digitalWrite(LEDPIN,LOW);
  sendplain("");   // acknowledge
}
```

. . . Add this to <body>:

```
<span id="somelabel"> </span> <br>
```

. . . Add this to <script>:

```
updateLabel();
function updateLabel() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("somelabel").innerHTML =
            this.responseText;
    }
  };
  xhttp.open("GET", "LEDstate", true);
  xhttp.send();
}
```

Q4 W

. . . Add this to setup():

```
attachHandler("/LEDstate",handleLEDstate);
```

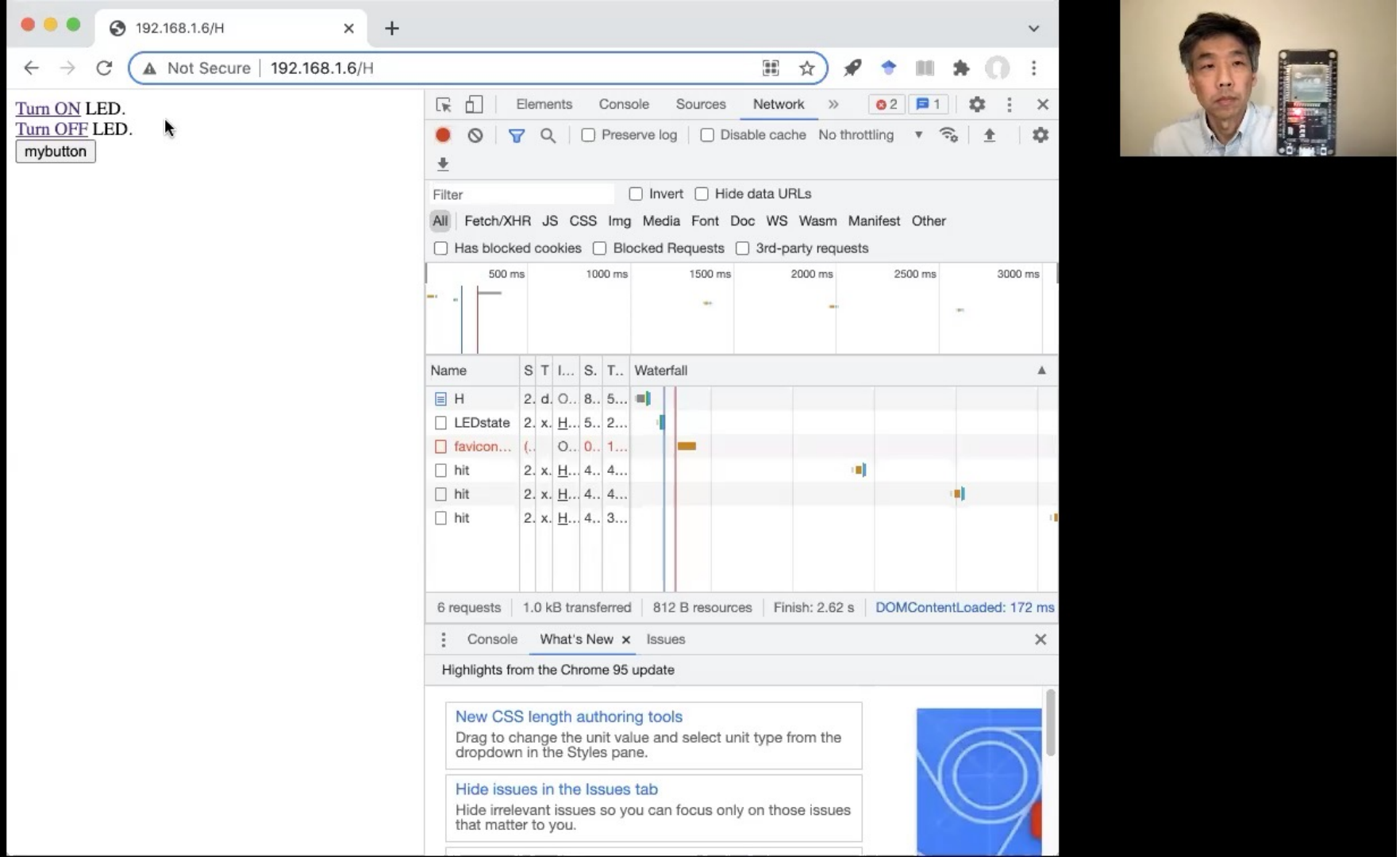
```
void handleLEDstate(){
  String s = "LED state is "+String(digitalRead(LEDPIN));
  sendplain(s);
}
```

Web510-button.h

```
<!DOCTYPE html>
<html>
  <body>
    <a href="/H">Turn ON</a> LED.<br>
    <a href="/L">Turn OFF</a> LED.<br>
    <button type="button" onclick="hit()"> mybutton </button>
    <span id="somelabel">  </span> LED is <br>
  </body>

  <script>
    function hit() {
      var xhttp = new XMLHttpRequest();
      xhttp.open("GET", "hit", true);
      xhttp.send();
    }

    updateLabel();      // call updateLabel when page is loaded
    function updateLabel() {
      var xhttp = new XMLHttpRequest();
      xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
          document.getElementById("somelabel").innerHTML = this.responseText;
        }
      };
      xhttp.open("GET", "LEDstate", true);
      xhttp.send();
    }
  </script>
```


192.168.1.6/H
Not Secure | 192.168.1.6/H
Turn ON LED.
Turn OFF LED.
mybutton
Elements
Console
Sources
Network
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
All
Fetch/XHR
JS
CSS
Img
Media
Font
Doc
WS
Wasm
Manifest
Other
Has blocked cookies
Blocked Requests
3rd-party requests
500 ms
1000 ms
1500 ms
2000 ms
2500 ms
3000 ms
Name
Waterfall
H
LEDstate
favicon...
hit
hit
hit
6 requests
1.0 kB transferred
812 B resources
Finish: 2.62 s
DOMContentLoaded: 172 ms
Console
What's New
Issues
Highlights from the Chrome 95 update
New CSS length authoring tools
Drag to change the unit value and select unit type from the dropdown in the Styles pane.
Hide issues in the Issues tab
Hide irrelevant issues so you can focus only on those issues that matter to you.

Timer Interval

```
void handleH(){
  digitalWrite(LEDPIN, HIGH);
  sendhtml(body);
}

void handleL(){
  digitalWrite(LEDPIN, LOW);
  sendhtml(body);
}

void handleHit(){
  static int toggle;
  if (++toggle%2)    digitalWrite(LEDPIN,HIGH);
  else digitalWrite(LEDPIN,LOW);
  sendplain("");   // acknowledge
}

void handleLEDstate(){
  String s = "LED state is "+String(LEDstate);
  sendplain(s);
}
```

. . . Add this to <script>:

```
setInterval(updateLabel, 1000);
```

. . . Add this to <script>:

```
updateLabel();
function updateLabel() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("somelabel").innerHTML =
              this.responseText;
    }
  };
  xhttp.open("GET", "LEDstate", true);
  xhttp.send();
}
```

. . . Add this to setup():

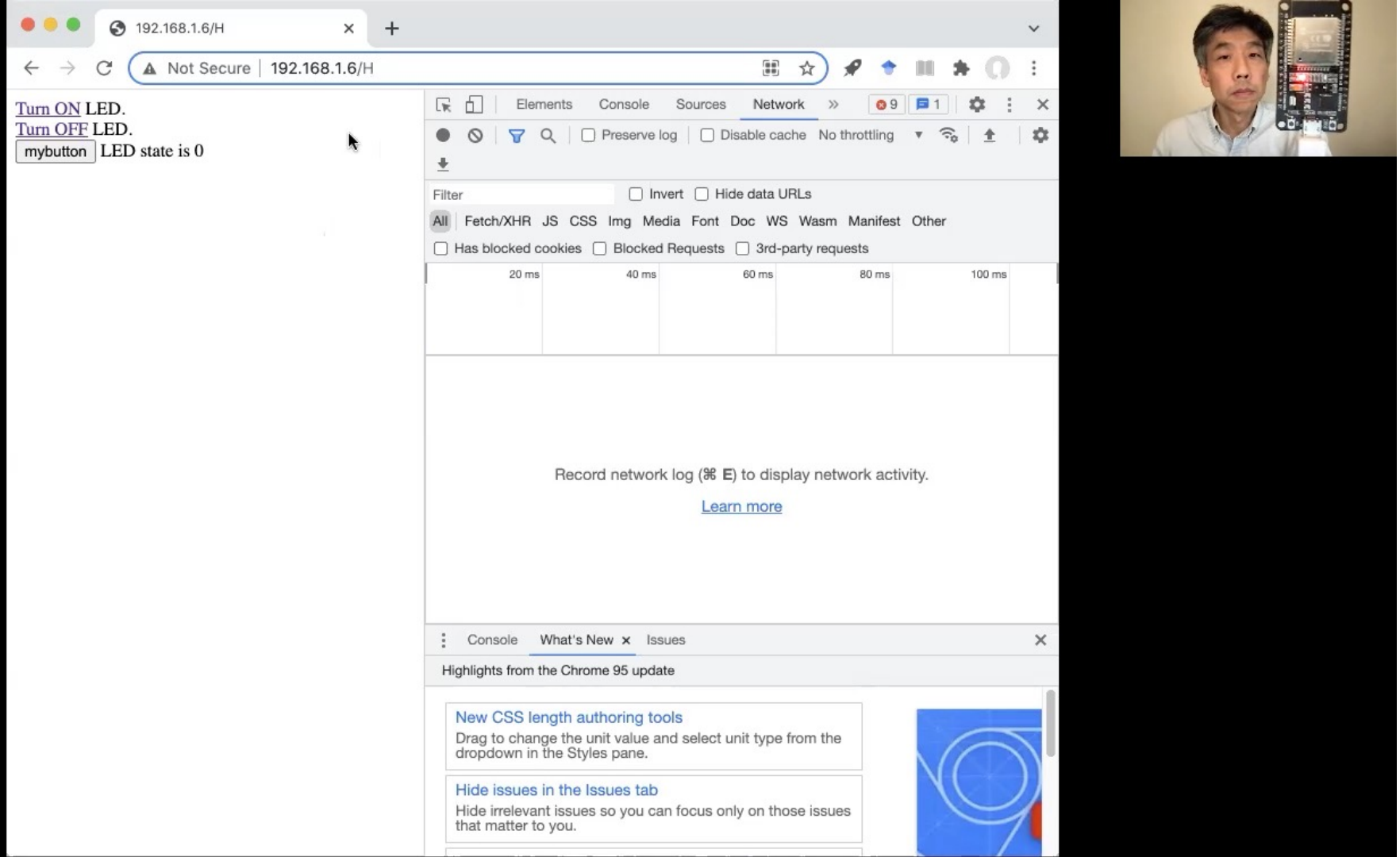
```
attachHandler("/LEDstate ",handleHit);
```

Web510-button.h

```
<!DOCTYPE html>
<html>
  <body>
    <a href="/H">Turn ON</a> LED.<br>
    <a href="/L">Turn OFF</a> LED.<br>
    <button type="button" onclick="hit()"> mybutton </button>
    <span id="somelabel">  </span> LED is <br>
  </body>

  <script>
    function hit() {
      var xhttp = new XMLHttpRequest();
      xhttp.open("GET", "hit", true);
      xhttp.send();
    }

    setInterval(updateLabel, 1000);
    function updateLabel() {
      var xhttp = new XMLHttpRequest();
      xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
          document.getElementById("somelabel").innerHTML = this.responseText;
        }
      };
      xhttp.open("GET", "LEDstate", true);
      xhttp.send();
    }
  </script>
```


192.168.1.6/H
Not Secure | 192.168.1.6/H
Turn ON LED.
Turn OFF LED.
mybutton LED state is 0
Elements
Console
Sources
Network
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
All
Fetch/XHR
JS
CSS
Img
Media
Font
Doc
WS
Wasm
Manifest
Other
Has blocked cookies
Blocked Requests
3rd-party requests
20 ms
40 ms
60 ms
80 ms
100 ms
Record network log (⌘ E) to display network activity.
Learn more
Console
What's New
Issues
Highlights from the Chrome 95 update
New CSS length authoring tools
Drag to change the unit value and select unit type from the dropdown in the Styles pane.
Hide issues in the Issues tab
Hide irrelevant issues so you can focus only on those issues that matter to you.

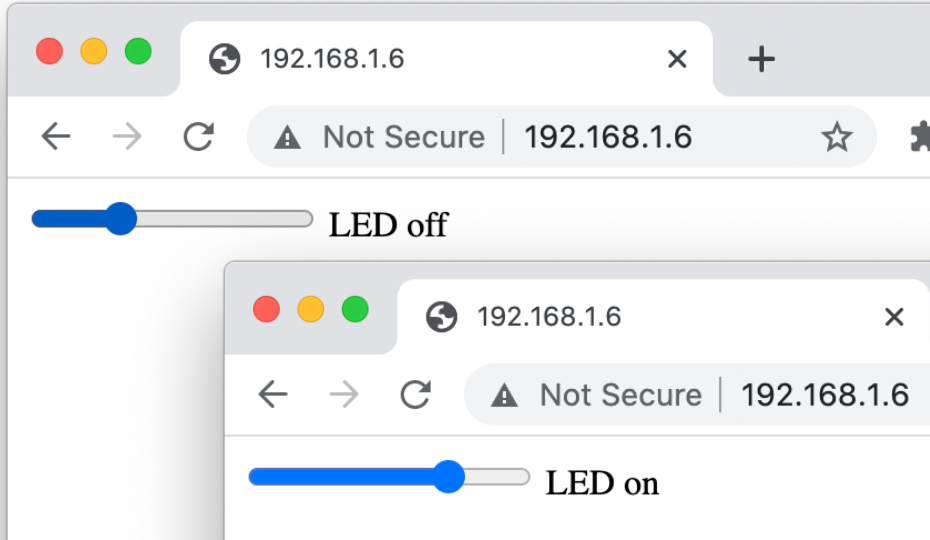
Slider

```
void handleSlider(){
  String s = "LED ";
  if (h.getVal()>50)  digitalWrite(LEDPIN,HIGH);
  else  digitalWrite(LEDPIN,LOW);
  if (digitalRead(LEDPIN)) s = s+ "on";
  else s = s+ "off";
  h.sendplain(s);
}
```

```
void setup() {
  WiFi.mode(WIFI_MODE_STA);
  WiFi.begin(ssid, password);
  WiFi.config(
    IPAddress(192, 168, 1, 6),IPAddress(192, 168, 1, 1),IPAddress(255, 255, 255, 0));
  while(WiFi.status()!= WL_CONNECTED ) {
    delay(500);
  }

  h.begin();
  h.attachHandler("/ ",handleRoot);
  h.attachHandler("/slider?val=",handleSlider);
  pinMode(LEDPIN, OUTPUT);
}

void loop() {
  h.serve();
  delay(10);
}
```



Slider webpage code

```
<!DOCTYPE html>
<html><body>
    <input type="range" min="1" max="100" value="50"  id="slider">
    <span id="outputlabel">  </span> <br>
</body>
<script>
    slider.onchange = function() {
      var xhttp = new XMLHttpRequest();
      xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
          document.getElementById("outputlabel").innerHTML = this.responseText;
        }
      };
      var str = "slider?val=";
      var res = str.concat(this.value);
      xhttp.open("GET", res, true);
      xhttp.send();
    }

</script>
```

HTML510 quiz

Q5. To send a command from a webpage button or slider to the ESP32 we use XML HTTP requests functions in javascript:

A. open();

B. send();

C. Both open(); and send();

Q6. To send a response from the ESP32 to a webpage (e.g. to change what the webpage displays) we can use XML HTTP request function onreadystatechange();

A. TRUE

B. FALSE

HTML510 Summary

- Button or slider are two input types you can use on a webpage
- Attach a handler routine to HTML text to send commands
- *xhttp*.open() and *xhttp*.send() will send HTML text to ESP32
- *xhttp*.onreadystatechange() can receive a response from the ESP32
- setInterval() will make the website periodically run a function (this is one way to get the server to act without the client asking.

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 internet.
- Debug HTML and 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.

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

- HTML, AJAX, 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. Lab 4 requirements and mobile base options

B. HTML structure and Javascript for use in MEAM510

C. Using HTML510 for webpages