Lecture 17

HTML510

Agenda

- HTML interface
  - Structure
  - Buttons
  - Javascript
- HTML510
- Lab 4 example

Announcements

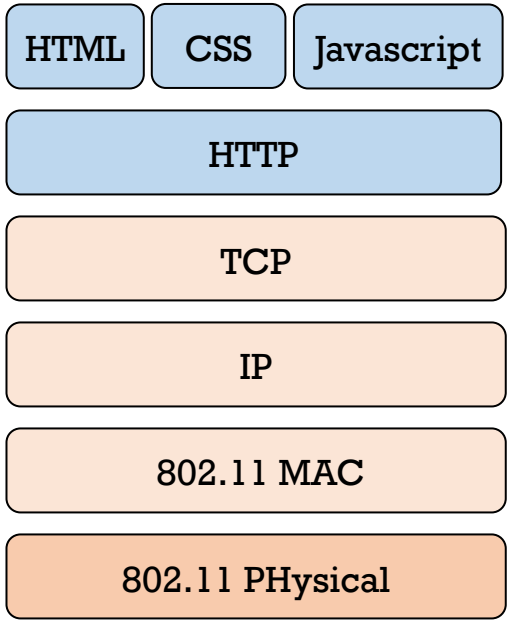
- If you have trouble qith 192.168.1.XXX your IP address with your router, try 192.168**.0.**XXX
- Becareful with calling `ledcSetup`() and `ledcWrite`() repeatedly. The `ledcWrite`() will occur after the first cycle has finished.
- Lab 4.2.6 (Races) will occur up to March 30. Arrange with your TA/coach.
- Lab 4.2.7 (Report) will be due March 31
- Concur reimbrusements
- Do not rely on canvas due dates alone. Modifications will be posted to Piazza and in lecture.

01

HTML bare bones for 510

Website construction

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



Website components

- **HTML**: Hypertext Markup Language defines the content of a Web page
  - 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 interactivity.
  - Ex:

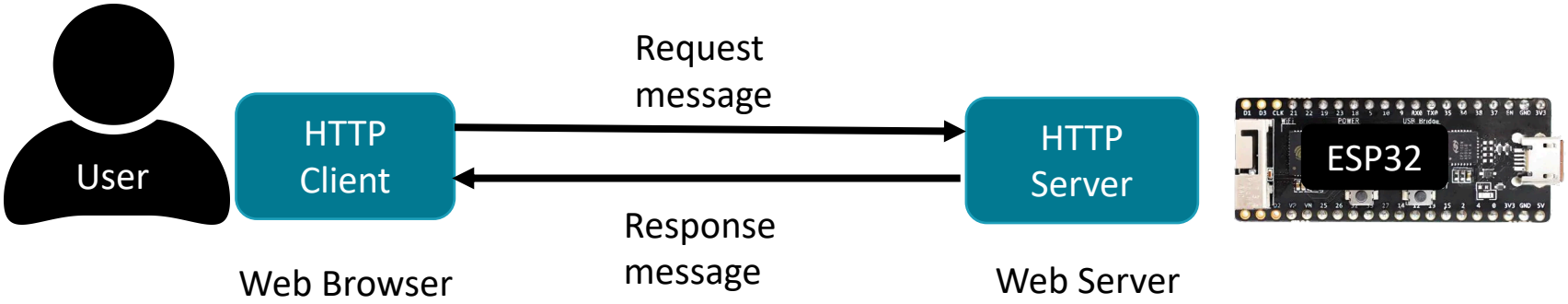
```
<script>
function changeLabel() {
    document.getElementById("label").innerHTML =  "sometext";
}
</script>
```

HTTP Client – Server Model

- Client intiates request.
- Server responds.

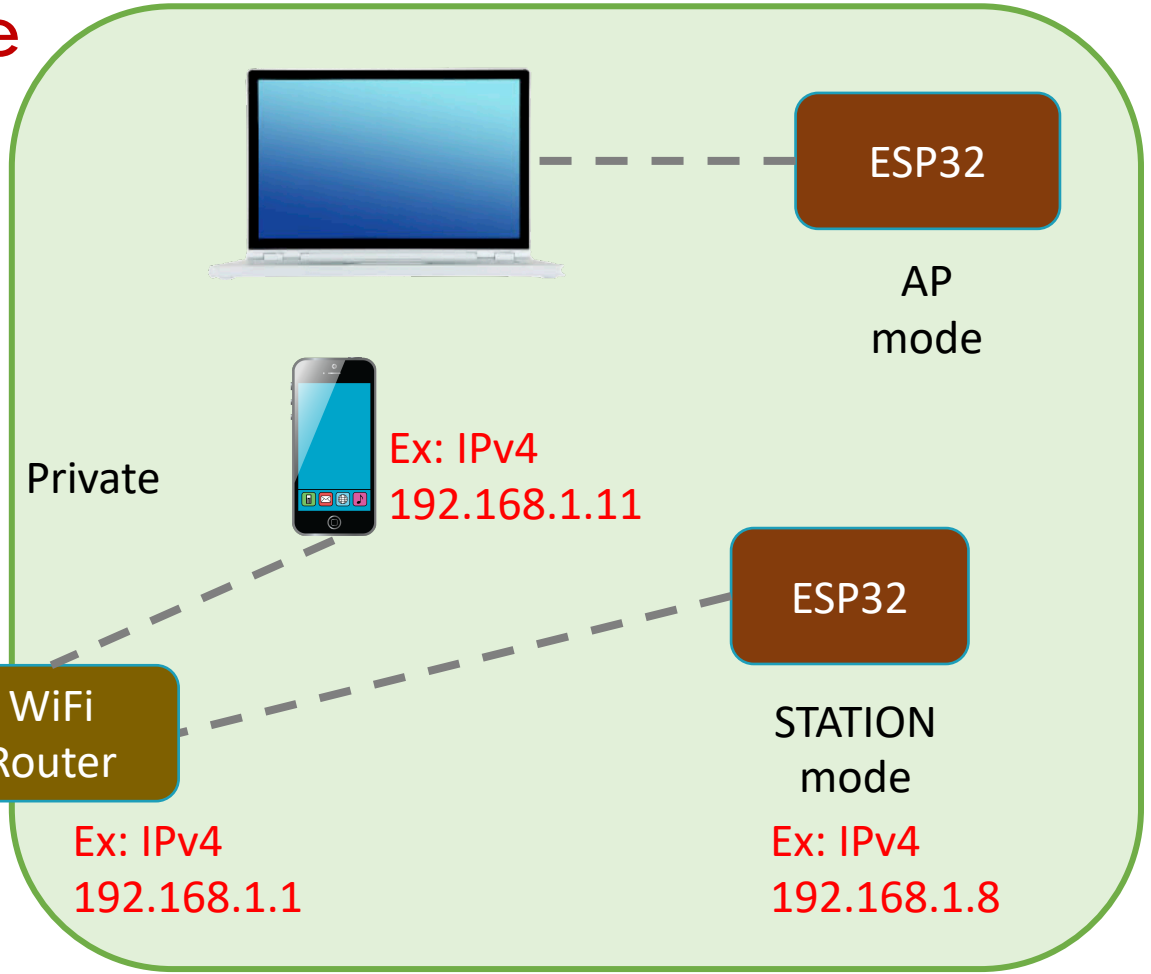
The server cannot initiate messages.

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



WiFi Architecture

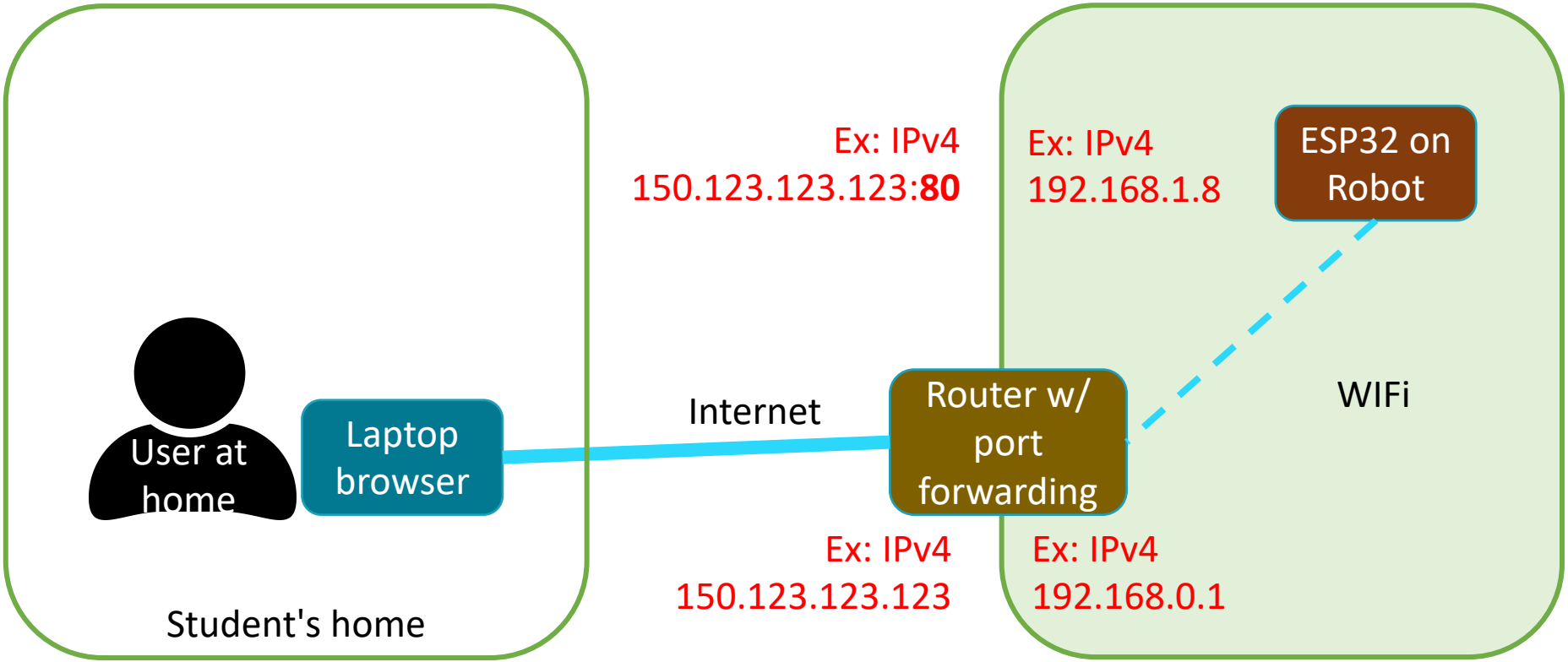
- 192.168.XXX.XXX is the local subnet.
- Routers may define the 1st XXX to be other than 1
- The 2nd XXX defines your host IP



02

ESP32 Internet access

Port forwarding from router to ESP32
(we won't do port forwarding in this class)



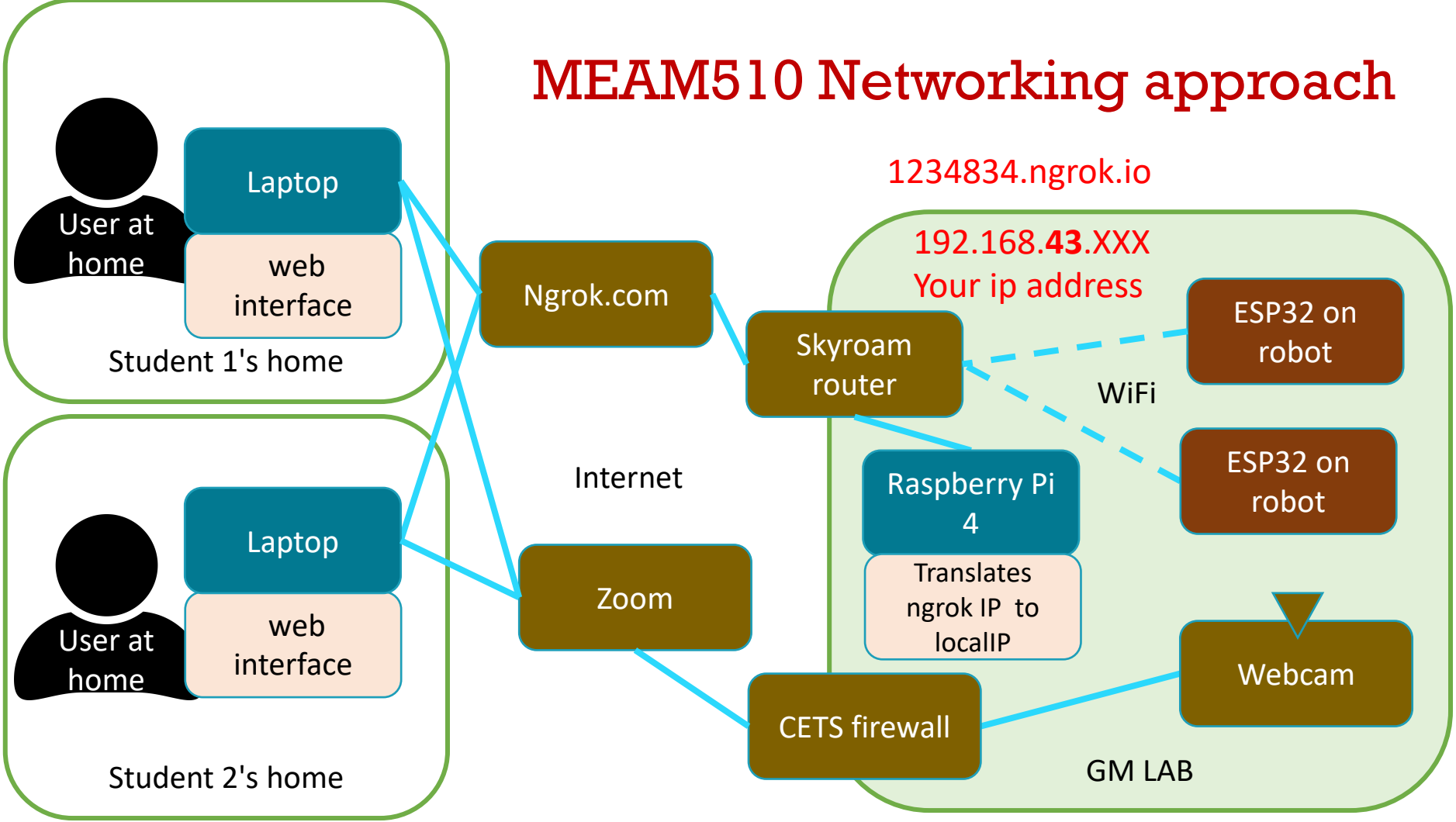
Port Forwarding Process Through your Router

- Find your router's public IP address (http://ip4.me/)
  - Example: 74.103.150.123
- Find your router's private IP address (usually 192.168.1.1 or similar)

https://www.howtogeek.com/236838/how-to-find-any-devices-ip-address-mac-address-and-other-network-connection-details/

- Set up router to forward your code's port to the routers IP address
  - Example ESP32 local address:
  - Example ESP32 internet address: 74.103.150.123:80
  - Standard website ports is 80, but you can set it to be up to 65535 (larger numbers have less chance of being reserved)
  - WiFiServer server(80);

MEAM510 Networking approach



03

HTML510

MEAM510 Web support code for ESP32

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();
}
```

Previous Sample 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 HTTP request messages

```
GET /L HTTP/1.1
Host: 777f4a18c8ef.ngrok.io
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.90
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Referer: http://777f4a18c8ef.ngrok.io/H
Upgrade-Insecure-Requests: 1
X-Forwarded-For: 172.58.207.29
X-Forwarded-Proto: http

GET /favicon.ico HTTP/1.1
Host: 777f4a18c8ef.ngrok.io
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.90
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Referer: http://777f4a18c8ef.ngrok.io/L
X-Forwarded-For: 172.58.207.29
X-Forwarded-Proto: http
```

New Version using HTML510

setup() and loop()

```
void setup() {
  Serial.begin(115200);
  pinMode(LEDPIN, OUTPUT);
  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);
    Serial.print(".");
  }
  Serial.println("Connected"); server.begin();

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

void loop(){
  serve(server, body);
  delay(1);
}
```

Q1. What's new here?

Routines from HTML510

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

Client Request Handlers

```
void setup() {
  void handleRoot(){
    sendhtml(body);
  }

  void handleH(){
    digitalWrite(LEDPIN, HIGH);  // LED ON
    sendhtml(body);
  }
  void handleL(){
    digitalWrite(LEDPIN, LOW);  // LED ON
    sendhtml(body);
  }
                                          ,0));
  attachHandler("/H",handleH);
  attachHandler("/L",handleL);
  attachHandler("/ ",handleRoot);
}

void loop(){
  serve(server, body);
  delay(1);
}
```

Client Request Handlers

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

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

Add button

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

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

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

. . . Add this to body:

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

. . . Add this script:

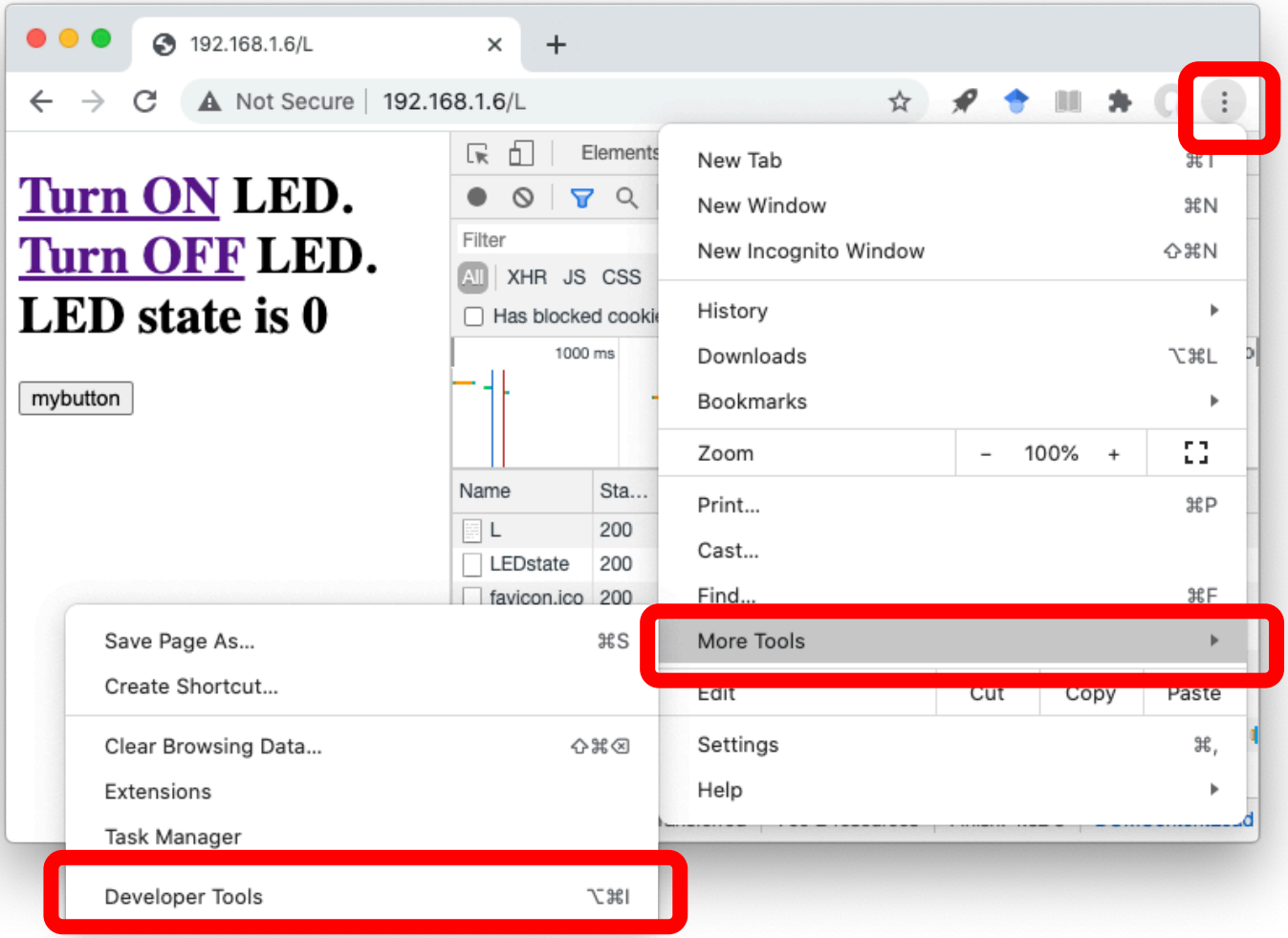
```
<script>
function hit() {
  var xhttp = new XMLHttpRequest();
  xhttp.open("GET", "hit", true);
  xhttp.send();
}
</script>
</html>
```

. . . Add this to setup():

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

Chrome Developer Tools

Very useful for debugging HTML and javascript code



Change 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
}

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

. . . 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();
}
```

. . . Add this to setup():

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

Q2 tup?

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);
```

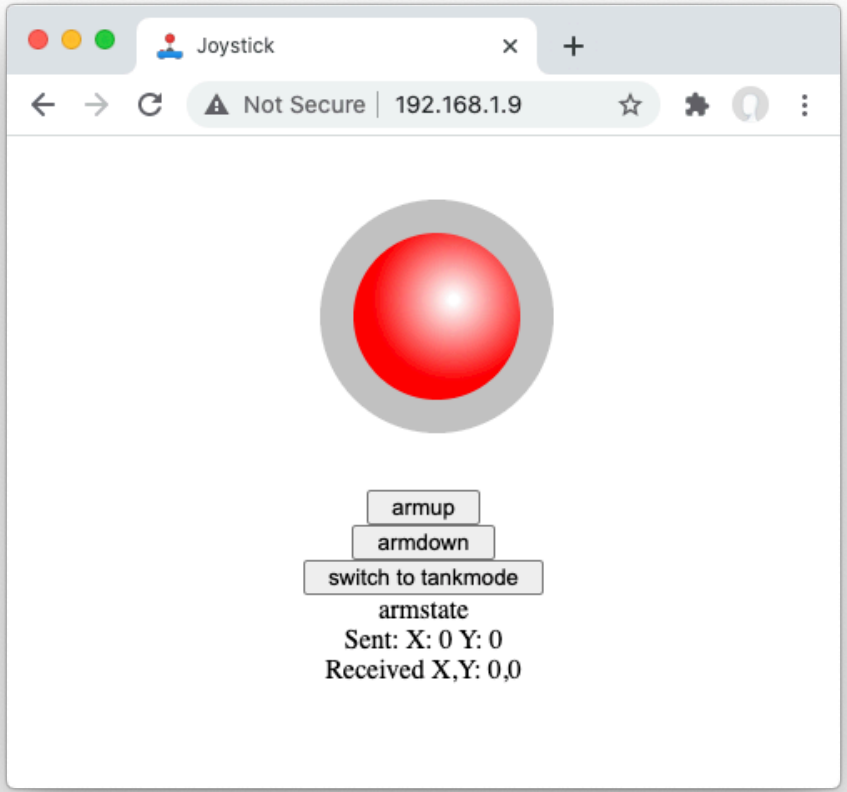
04

Lab4 Demo

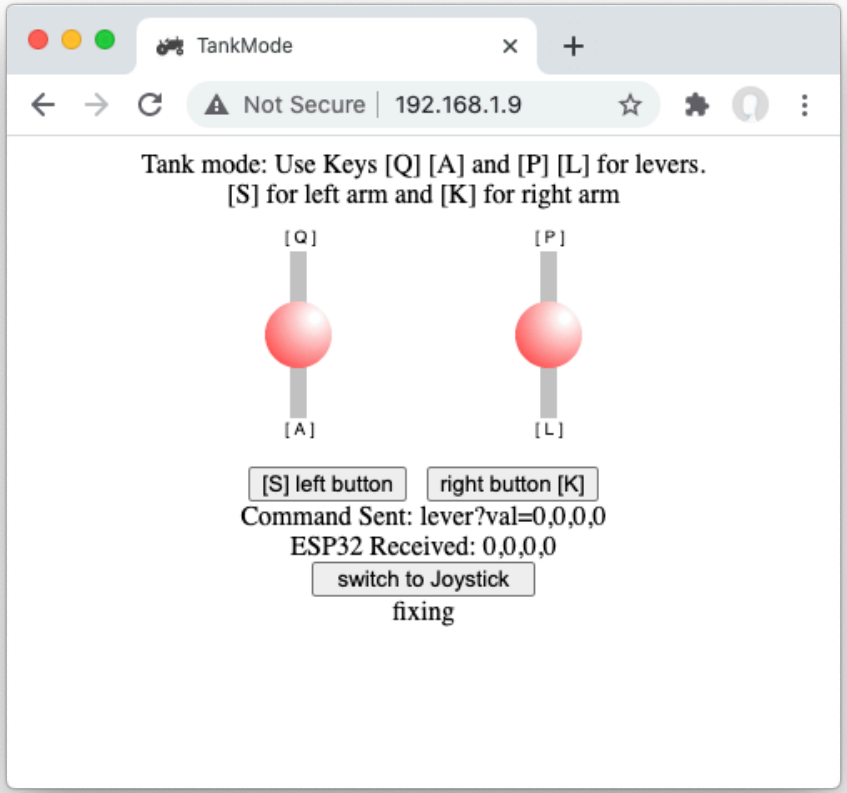
Lab 4 DemoCode structure

 lab4demo2.ino	• High level Arduino Modify this code for your robot
 tankJS.h	• Javascript code for tankmode interface
 joyJS.h	• Javascript code for joystick mode
 html510.cpp	• C++ code linking ESP32 and HTML
 html510.h	• C++ header support for ESP32 and HMTL

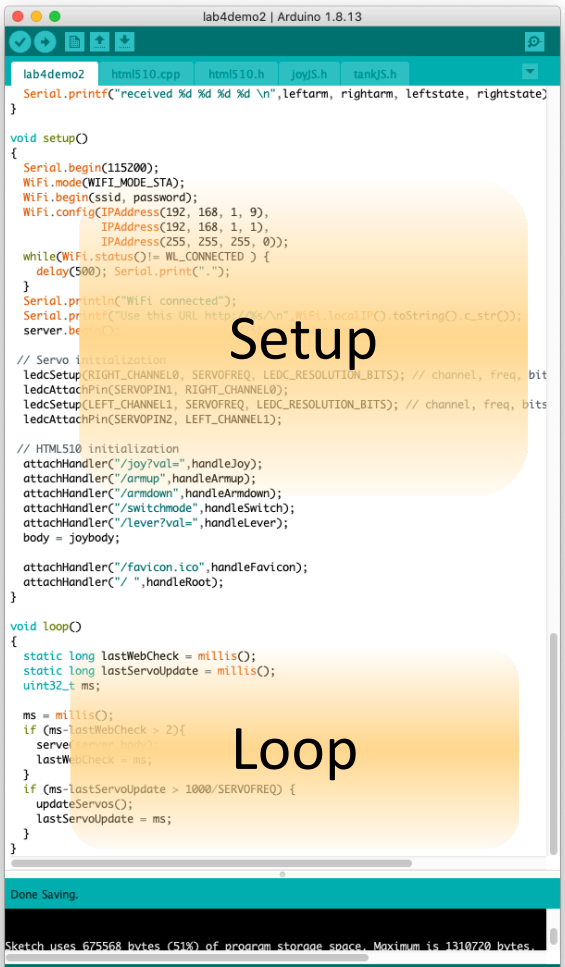
joyJS.h



tankJS.h



lab4demo2.ino



Lab4 Demo setup()

```
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_MODE_STA);
  WiFi.begin(ssid, password);
  WiFi.config(IPAddress(192, 168, 1, 9),
        IPAddress(192, 168, 1, 1), IPAddress(255, 255, 255, 0));
  while(WiFi.status()!= WL_CONNECTED ) {
    delay(500); Serial.print(".");
  }
  Serial.println("WiFi connected");
  Serial.printf("Use this URL http://%s/\n",WiFi.localIP().toString().c_str());
  server.begin();

  ledcSetup(RIGHT_CHANNEL0, SERVOFREQ, LEDC_RESOLUTION_BITS);
  ledcAttachPin(SERVOPIN1, RIGHT_CHANNEL0);
  ledcSetup(LEFT_CHANNEL1, SERVOFREQ, LEDC_RESOLUTION_BITS);
  ledcAttachPin(SERVOPIN2, LEFT_CHANNEL1);

  attachHandler("/joy?val=",handleJoy);
  attachHandler("/armup",handleArmup);
  attachHandler("/armdown",handleArmdown);
  attachHandler("/switchmode",handleSwitch);
  attachHandler("/lever?val=",handleLever);
  body = joybody;
  attachHandler("/favicon.ico",handleFavicon);
  attachHandler("/ ",handleRoot);
}
```

The main joystick routine

The main tankmode routine

Lab4 Demo loop()

```
void loop()
{
  static long lastWebCheck = millis();
  static long lastServoUpdate = millis();
  uint32_t ms;

  ms = millis();
  if (ms-lastWebCheck > 2){
    serve(server,body);
    lastWebCheck = ms;
  }
  if (ms-lastServoUpdate > 1000/SERVOFREQ) {
    updateServos();
    lastServoUpdate = ms;
  }
}
```

Q3: How many times per sec is serve() called?

Update servos 60 times per sec

Lab4 Demo Servo code

```
#define RIGHT_CHANNEL0      0 // use first channel of 16
#define LEFT_CHANNEL1       1
#define SERVOPIN1    33
#define SERVOPIN2    32
#define SERVOFREQ    60
#define LEDC_RESOLUTION_BITS  12
#define LEDC_RESOLUTION  ((1<<LEDC_RESOLUTION_BITS)-1)
#define FULLBACK LEDC_RESOLUTION*10*60/10000
#define SERVOOFF  LEDC_RESOLUTION*15*60/10000
#define FULLFRONT  LEDC_RESOLUTION*20*60/10000
int leftservo, rightservo;

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 updateServos() {
  ledcAnalogWrite(LEFT_CHANNEL1, leftservo, LEDC_RESOLUTION);
  ledcAnalogWrite(RIGHT_CHANNEL0, rightservo, LEDC_RESOLUTION);
}
```

Lab4 Demo Top Matter

```
#include <WiFi.h>
#include "html510.h"
#include "joyJS.h"
#include "tankJS.h"

WiFiServer server(80);

const char* ssid     = "#Skyroam_1t9";
const char* password = "55687127";
const char *body;

void handleFavicon(){
  sendplain(""); // acknowledge
}

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

This is the router in the GMlab
For testing at home, replace with your
Home router SSID and PASS

```
void handleSwtch() {
  String s="";
  static int toggle=0;
  if (toggle) body = joybody;
  else body = tankbody;
  toggle = !toggle;
  sendplain(s); //acknowledge
}
```

Lab4 Demo
Joystick mode

```
void handleJoy() {
  int left, right;
  x = getVal(); // from -50 to +50
  y = getVal();
  String s = String(x) + "," + String(y);

  left = x - y;
  right = x + y;

  leftservo = map(left, -50, 50, FULLBACK, FULLFRONT);
  rightservo = map(right, -50, 50, FULLBACK, FULLFRONT);

  sendplain(s);
  Serial.printf("received X,Y:=%d,%d\n",x,y);
}
```

Lab4 Demo
Tank mode

```
int leftstate, rightstate;
long lastLeverMs;

void handleLever() {
  leftarm = getVal();
  rightarm = getVal();
  leftstate = getVal();
  rightstate = getVal();
  String s = String(leftarm) + "," + String(rightarm) + "," +
             String(leftstate) + "," + String(rightstate);

  if (leftstate>0)      leftservo =  FULLBACK;
  else if (leftstate<0) leftservo =  FULLFRONT;
  else                  leftservo =  SERVOOFF;

  if (rightstate>0)      rightservo =  FULLFRONT;
  else if (rightstate<0) rightservo =  FULLBACK;
  else                   rightservo =  SERVOOFF;

  lastLeverMs = millis(); //timestamp command
  sendplain(s);
  Serial.printf("rec'd %d %d %d %d \n",leftarm, rightarm, leftstate, rightstate);
}
```

Lab 4 Demo

- Would be good to see what it is like to use the joystick or tank mode if you want to use those interfaces.
- Also to see what lag is like for control.
- During any TA OH you can request to try the robot.

Summary

- HTML510 is like webserver.h in Arduino that gives fast easy interface to web page interaction.
- The intent is not for you to learn javascript and HTML.
- You can use the joystick and tankmode interface for your Lab4 and build on it for the final project.
- We can add/change other simple javascript functionality if you need it.

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. HTML510 for use with ESP32

B. Accessing Internet with ESP32

C. Succeeding with Lab 4