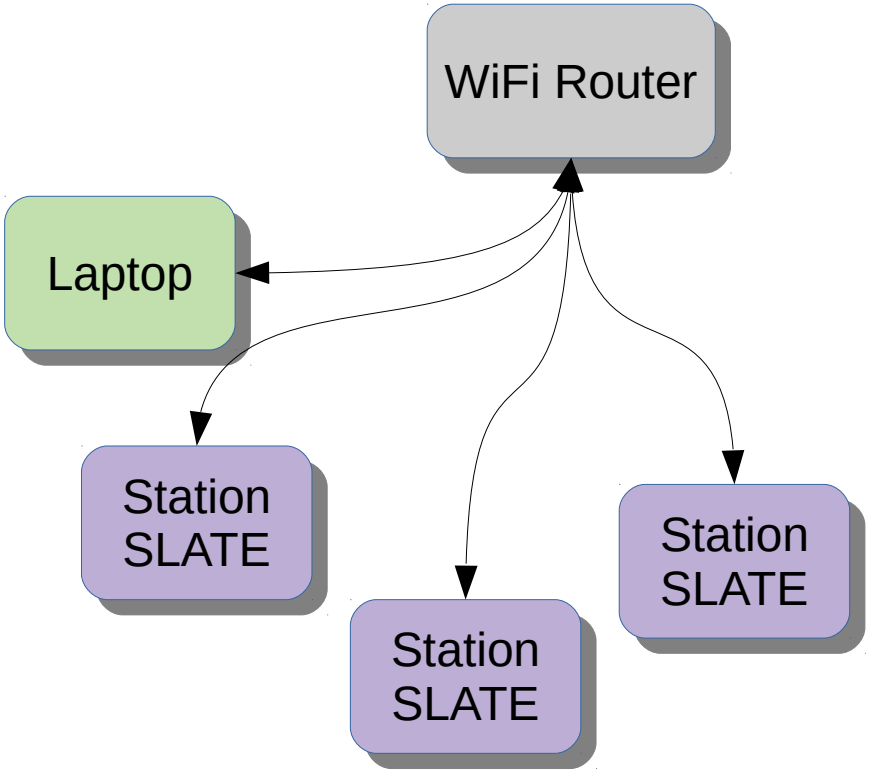
Intro to Internet of Things

Introduction

- Internet of Things also known as IoT is a term used for connecting any type of device to the internet to be controlled and/or monitored such as appliances, lighting, locks, etc.
- WiFi routers are everywhere and in many homes. This lesson will explain how multiple SLATEs can be connected to a WiFi router and how to access them through another computer.



Introduction

- WiFi routers give computers and devices an IP address when the devices connect to the router. The router runs a DHCP server program. DHCP is Dynamic Host Configuration Protocol. It is configured with a range of IP addresses it can send to devices that connect and request an IP address.
- Many WiFi routers are configured by default with an address range of 192.168.0.100 to 192.168.0.254 or 192.168.1.100 to 192.168.1.254. This leaves the addresses below 100 to be used as static IP addresses where the device can be manually configured with an IP address.

Connecting to Router

- This code shows how to connect to the WiFi router.
- The SLATE WiFi is configured to Station mode which means it connects to an access point which is the WiFi router.
- You need to insert the SSID and passphrase into the code in order to connect.
- The code will attempt to connect and display the IP address received.
- Try it out and see what IP address is received. Before compiling and uploading, open the serial monitor window. This way, you will see the IP address since it is displayed once.

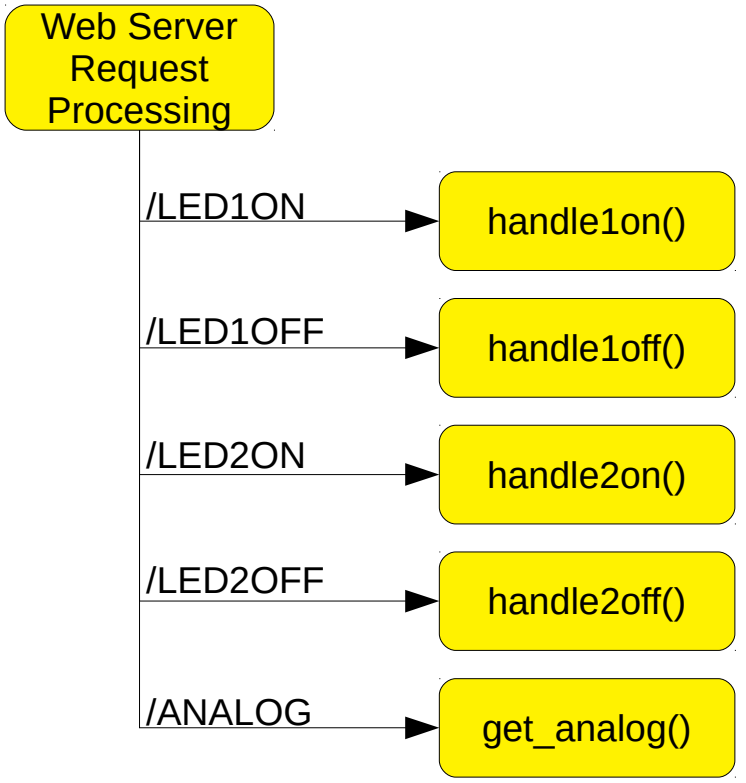
```
#include <ESP8266WiFi.h>

const char* ssid = "serveSLATE";
const char* pass = “passphrase”;

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid,pass);
  while(WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println(WiFi.localIP());
}

void loop() {
}
```

Creating a Web Server

- The client and server will use HTTP which is **H**yper **T**ext **T**ransfer **P**rotocol.
- HTTP is stateless. The client has to connect for each data transfer and then disconnect.
- The client sends a request to the server and the server responds.
- There are two request methods, GET and POST. GET will be used here.
- GET requests data from a specified resource. It is the simpler method of communicating with the server.
- Web browsers require a URL to identify a website. It is the website name or address.
- A GET request can be typed into the web browser URL location. Example:
- 192.168.x.x/LED1ON (insert SLATE IP address)
- The /LED1ON identifies a resource in the web server such as a directory or file.
- This web server will use it to execute a function.

Simple Web Server

- The web server will listen for a client to connect and send a request. If the server receives:
- 192.168.4.1/LED1ON
- The server will call function **handle1on()**.
- Other commands shown to the right will call the functions specified.
- The connection between the requests and the functions are configured in the **setup()** function.



Simple Webserver

- Connect the RED LED to digital pin 14.
- Connect the GREEN LED to digital pin 15.
- The webserver will control the LEDs.

Simple Webserver

- The include file is a bit different. This one provides functions to support http packet processing.
- A web server object is created in the second line.
- Next are the functions that execute the commands received. Each operation is a separate function. The first four are turning LEDs on and off. The last one gets the ADC value and sends it back to the client.
- Each function sends a response indicating its operation.
- This is is the first part of the SLATE code.

```
#include <ESP8266WebServer.h>

ESP8266WebServer server(80);

void handle1on() {
  digitalWrite(14,HIGH);
  server.send(200,"text/plain","LED 1 ON\n");
}

void handle1off() {
  digitalWrite(14,LOW);
  server.send(200,"text/plain","LED 1 OFF\n");
}

void handle2on() {
  digitalWrite(15,HIGH);
  server.send(200,"text/plain","LED 2 ON\n");
}

void handle2off() {
  digitalWrite(15,LOW);
  server.send(200,"text/plain","LED 2 OFF\n");
}

void get_analog() {
  int a = analogRead(0);
  String b = String("Result: ") + String(a) + "\n";
  server.send(200,"text/plain",b);
}
```

Simple Webserver

- The setup function sets up the web server and how all the functions will be called.
- First, the serial interface is configured.
- Next, the WiFi is set to be a station using the **WiFi.mode()** function.
- The SSID is set with the **Wifi.begin()** function.
- The next 5 lines set up the webserver. Each request is assigned a function.
- The server is then started and the digital pins are configured.
- In the loop function, the function **server.handleClient()** is repeatedly called to handle clients that connect.

```
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid,pass);
  while(WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println(WiFi.localIP());
  server.on("/LED1ON",handle1on);
  server.on("/LED1OFF",handle1off);
  server.on("/LED2ON",handle2on);
  server.on("/LED2OFF",handle2off);
  server.on("/ANALOG",get_analog);
  server.begin();
  pinMode(15,OUTPUT);
  pinMode(14,OUTPUT);
}

void loop() {
  server.handleClient();
}
```

Simple Webserver

- Open the serial monitor and run the SLATE code. You will need to open the serial monitor before compiling and uploading since the IP address is shown only once.
- Note the IP address the SLATE displays. That is needed below.
- A web browser is required to test out the web server. Connect the laptop WiFi to the WiFi router.
- Open a web browser.
- In the URL, enter **`192.168.x.x/LED1ON`** where x.x is the last part of the IP address the SLATE displayed.
- This should turn the LED on.
- Now enter **`192.168.x.x/LED1OFF`**
- The LED should turn off.
- Try **192.168.x.x/ANALOG** , the ADC value should be returned.
- There should be a response in the web browser for each command.
- You can always expand the response and add html code.

Auto-Refreshing Webpage

- You can make the web page automatically update. It requires a little bit of html code added to the string sent to the web browser. Go to the **get_analog()** function and add the highlighted line.
- The html code is added to the front of the measurement string and **</html>** is added to the end.
- The content=2 sets the refresh rate of the page. It can be changed to any integer value.
- Notice **\”**. You cannot have a quote inside a string which is requires quotes to identify the string. the backlash is an escape sequence that tells the compiler the next character is not to be interpreted as part of the language.

```
void get_analog()
{
  int a = analogRead(0);
  String b = String("Temperature: ") + String(a) + "\n";
  b = “<html><head><meta http-equiv=\”refresh\” content=\”2\”>” + b + “</html>”;
  server.send(200,"text/html",b);
}
```

IoT

- This simple webserver shows the basics of the Internet of Things. You can have multiple SLATES, each with their own IP address and use a web browser to control each one.

Better Web Server

- In this section, we will use the file system to contain a web page and set up the web server to use the file instead of having a String variable contain the html code.
- Review the File System lesson on how to set up to use part of the SLATE memory for file storage using the SPIFF library.
- Create a new program and name it **staticweb**.
- In the staticweb directory, create a directory called **data**.
- The directory **data** will hold the html file.

Static Web Pages

- You need to configure the processor to support a file system.
- Go into the **Tools** menu and select **Flash Size**
- If your processor is blue, select **1M (512K SPIFFS).** This will allocate 512 Kbytes for storing html and associated files.
- If your processor is black, select 4M (3M SPIFFS). This will allocate 3 MB for storing html and associated files.

Static Web Pages

- With a text editor, create the file to the right and save it as **index.html**.
- This will create a web page with a sentence in bold.
- Select **ESP8266 Sketch Data Upload** in the Tools menu. This will create a file system image and upload it to the SLATE processor. This will take a while depending on how large of a file system was selected.

```
<html><title>Simple HTML</title><body>
<h1>This is a simple page</h1>
</body></html>
```

A nice and popular text editor is called note++ for Windows and Mac. It can be found at https://notepad-plus-plus.org/

For Mac OSX:
https://www.sublimetext.com/

Static Web Pages

- Next is to create the web server. It will be mostly the same as before but not require a function to generate the web page.
- Besides starting the file system, the significant difference is the **server.serveStatic()** function. This replaces the **server.on()** function. The new function will automatically send the index.html file to the web browser when a browser connects.
- Upload the code and connect with the web browser by typing the URL **http://192.168.x.x/**

```
#include <ESP8266WebServer.h>
#include <LittleFS.h>

ESP8266WebServer server(80);

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid,pass);
  while(WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println(WiFi.localIP());
  LittleFS.begin();
  server.serveStatic("/",LittleFS,"index.html");
  server.begin();
}

void loop() {
  server.handleClient();
}
```

Static Web Page

- Multiple web pages can be included. Each web page needs to be linked using the **server.serveStatic()** function.
- The code to the right shows the second page added to the web server. A file **page2.html** needs to be added to the **data** directory.
- Copy the **index.html** to **page2.html**. Modify the text so it is different from **index.html**.
- Select **ESP8266 Sketch Data Upload** in the Tools menu to upload the new page.
- In the browser, go to 192.168.x.x/page2.
- The second page should come up.

```
#include <ESP8266WebServer.h>
#include <LittleFS.h>

ESP8266WebServer server(80);

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid,pass);
  while(WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println(WiFi.localIP());
  LittleFS.begin();
  server.serveStatic("/",SPIFFS,"index.html");
  server.serveStatic("/page2",LittleFS,"page2.html");
   server.begin();
}

void loop() {
  server.handleClient();
}
```

Generic Web Server

- This code will let you have as many web pages and images as can fit in the file system. The trick is to use the function **server.onNotFound()**. This function is called when no other **server.on()** functions detect a file. It is normally used to send an error message to the web browser.
- Don’t forget to replace ssid and pass with your WiFi router information.

```
#include <ESP8266WebServer.h>
#include <LittleFS.h>

ESP8266WebServer server(80);


void setup() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(“ssid”,”pass”);
  while(WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println(WiFi.localIP());
  LittleFS.begin();
  server.onNotFound(handleOther);
  server.begin();
}

void loop() {
  server.handleClient();
}
```

Generic Web Server

- **server.onNotFound()** calls the function **handleOther()**.
- **handleOther()** looks at the URL information received from the browser and determines what file to send.
- **server.uri()** gets the path of what is requested. The path is printed to the serial monitor. The function **path.c_str()** converts path to a printable string.

```
void handleOther()
{
  String path = server.uri();
  Serial.println(path.c_str());
  String dataType = "text/plain";
  if(path.endsWith("/")) path = "index.html";
  if(path.endsWith(".jpg")) dataType = "image/jpeg";
  else if(path.endsWith(".png")) dataType = "image/png";
  else if(path.endsWith(".html")) dataType = "text/html";
  else {
    server.send(404,dataType,"Error");
    return;
  }
  File datafile = LittleFS.open(path.c_str(),"r");
  if(!datafile) {
    server.send(404,"text/plain","file not found");
    return;
  }
  server.streamFile(datafile,dataType);
  datafile.close();
}
```

Generic Web Server

- Variable **dataType** is set to **“text/plain”**. This is required by the web browser to indicate the type of information being received. Each file type has a unique data type.
- The first check is to determine if the main page is to be accessed. This is usually indicated by */*. If it is, then the path variable is set to **/index.html**. This will cause the function to send the **index.html** page to the web browser.

```
void handleOther()
{
  String path = server.uri();
  Serial.println(path.c_str());
  String dataType = "text/plain";
  if(path.endsWith("/")) path = "index.html";
  if(path.endsWith(".jpg")) dataType = "image/jpeg";
  else if(path.endsWith(".png")) dataType = "image/png";
  else if(path.endsWith(".html")) dataType = "text/html";
  else {
    server.send(404,dataType,"Error");
    return;
  }
  File datafile = LittleFS.open(path.c_str(),"r");
  if(!datafile) {
    server.send(404,"text/plain","file not found");
    return;
  }
  server.streamFile(datafile,dataType);
  datafile.close();
}
```

Generic Web Server

- Next, file types are checked. **path.endsWith()** will check if the argument matches the end of the string variable value. **dataType** is set to the type of file being accessed.
- If none of the file types are recognized, an error message is sent to the browser and the function exits.

```
void handleOther()
{
  String path = server.uri();
  Serial.println(path.c_str());
  String dataType = "text/plain";
  if(path.endsWith("/")) path = "index.html";
  if(path.endsWith(".jpg")) dataType = "image/jpeg";
  else if(path.endsWith(".png")) dataType = "image/png";
  else if(path.endsWith(".html")) dataType = "text/html";
  else {
    server.send(404,dataType,"Error");
    return;
  }
  File datafile = LittleFS.open(path.c_str(),"r");
  if(!datafile) {
    server.send(404,"text/plain","file not found");
    return;
  }
  server.streamFile(datafile,dataType);
  datafile.close();
}
```

Generic Web Server

- If there was no error, the requested data file is opened. **if(!datafile)** checks to see if the file exists. If not, an error message is sent. Notice in the second error message, the data type is spelled out. This is because dataType could have been changed to something else and the error message is plain text.

```
void handleOther()
{
  String path = server.uri();
  Serial.println(path.c_str());
  String dataType = "text/plain";
  if(path.endsWith("/")) path = "index.html";
  if(path.endsWith(".jpg")) dataType = "image/jpeg";
  else if(path.endsWith(".png")) dataType = "image/png";
  else if(path.endsWith(".html")) dataType = "text/html";
  else {
    server.send(404,dataType,"Error");
    return;
  }
  File datafile = LittleFS.open(path.c_str(),"r");
  if(!datafile) {
    server.send(404,"text/plain","file not found");
    return;
  }
  server.streamFile(datafile,dataType);
  datafile.close();
}
```

Generic Web Server

- Finally, if there were no errors, the selected file is sent and the file is then closed.
- This program will let you create a web server with multiple web pages and multiple images embedded in the web pages.
- Just remember all the web pages and images must fit within the file system space. Any image or file update requires the Sketch Data Upload. If all the files exceed the size of the file system, the upload will let you know.

```
void handleOther()
{
  String path = server.uri();
  Serial.println(path.c_str());
  String dataType = "text/plain";
  if(path.endsWith("/")) path = "index.html";
  if(path.endsWith(".jpg")) dataType = "image/jpeg";
  else if(path.endsWith(".png")) dataType = "image/png";
  else if(path.endsWith(".html")) dataType = "text/html";
  else {
    server.send(404,dataType,"Error");
    return;
  }
  File datafile = LittleFS.open(path.c_str(),"r");
  if(!datafile) {
    server.send(404,"text/plain","file not found");
    return;
  }
  server.streamFile(datafile,dataType);
  datafile.close();
}
```

Learn HTML

- If you want to learn how to create web pages with html, do a search for learn html. Videos will come up.
- Other possible sites:
  - https://www.codeacademy.com/learn/learn-html
  - https://www.w3schools.com/Html

Expanding Generic Server

- You can expand the web server by combining the simple web server shown at the start with the generic web server.
- You can add **server.on()** functions to call specific functions. The only requirement is that the **server.on()** function be inserted above **server.onNotFound()**.
- Complete the addition of the funtions from the first web server lesson to the generic server.

```
#include <ESP8266WebServer.h>
#include <LittleFS.h>

ESP8266WebServer server(80);


void setup() {
  WiFi.mode(WIFI_STA);
  WiFi.begin("ssid","pass");
  while(WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println(WiFi.localIP());
  LittleFS.begin();
  server.on("/LED1ON",handle1on);
  server.on("/LED1OFF",handle1off);
  server.onNotFound(handleOther);
  server.begin();
}

void loop() {
  server.handleClient();
}
```

End