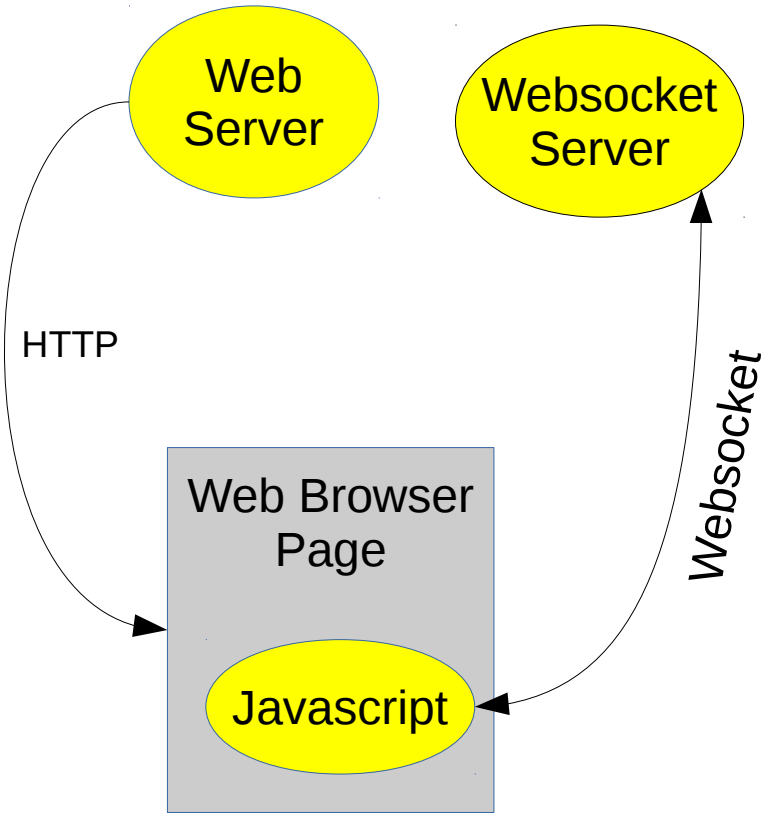
Websockets

What are Websockets

- A websocket is a separate network connection usually between a javascript running in a web browser and the web server. It provides a more efficient way to pass information between the browser and the web server. This allows for interactive and real time updating of information on a web browser.
- The simple web server lesson used static web pages and required the browser to reload web pages to update information.
- Most real time interaction is done with javascript embedded in the web page. Web sockets are also available for PHP.



Websocket Implementation

- In the SLATE processor, two servers need to operate, the web server and the websocket server.
- The web server will use a static web page using the **server.serveStatic()** function. A separate html file will be uploaded into the file system.
- The websocket server will call a function when ever a message is received from the web page javascript program.
- In this example, the SLATE will use a web page to provide a user interface to turn two LEDs on and off individually.
- Open the program websocketsample and follow along the explanation.
- A library is needed to use websockets. Open the **Library Manager** and search for **websockets**. Find the library called **Websockets by Markus Sattler**. Install the latest version.

Websocket Server Program

- The first three lines include the needed libraries. For static pages, the file system library is included and the websockets server library is installed.
- Skip to line 47 for the setup() function. The two digital ports used to control the LEDs are set to output. Connect pin 12 to the red LED and pin 13 to the yellow LED.
- The file system is initialized at line 51.
- Lines 52, 53 set up the WiFi to operate as an access point.
- Line 54 sets up the link to access the html file from the file system.
- Line 55 starts the web server.
- Line 56 sets up the websocket server to call the function **webSocketEvent()** when a websocket message is received.
- Line 57 starts the websocket server.

Websocket Server Program

- The loop() function checks the websocket and the web server.
- Each function checks for a packet or message being received and the processes it.
- The **webSocket.loop()** function will call **webSocketEvent()** when a packet is received from the javascript program.
- The **server.handleClient()** function will process the http packet received and send the requested web page.

Websocket Server Program

- Going back to line 9, This function is called when a websocket packet is received. The server passes four arguments. First is the connection number that is not used in the example. The second is the type of packet received. The third is the contents of the packet and the last argument is the size of the contents, the number of bytes.
- Line 10 is a switch() statement that is used to determine how to process the different types of packets.
- Line 11 is executed is the client closes the websocket connection.
- Line 14 is executed if the packet type is a connection being established. The web browser IP address is displayed in the serial terminal in this example. This packet type is useful if things need to be initialized when a connection is made.

Websocket Server Program

- Line 20 is executed when a text type packet is received. Strings can be sent and received this way.
- Line 21 is another **switch()** statement that determines which case is executed to turn the specified LED on or off.
- Line 38 is executed when a binary packet is received. This is not used in this example.

HTML File Description

- The html file creates the user interface. A canvas is created and four buttons below it are created.
- Open the index.html file in the data folder. Use a text editor to open it.
- Line 1 is the minimum needed to define the file is an html file.
- Line 2 creates the canvas that is 600 pixels wide by 400 pixels tall.
- Lines 4 -7 create buttons below the canvas.
  - **onclick=”send_command('1')”** tells the browser to call the javascript function send_command() to execute.
  - Each button calls the same function with a different argument.

HTML File Description

- Lines 10-48 is the javascript program. Line 9 is a tag that indicates the following text is to be interpreted as javascript. Line 49 </script> indicates the end of the javascript code.
- Line 10 links variable **c** to the canvas. Notice line 2 has **id=”can”**. This is used to link the canvas to the javascript code.
- Line 11 creates a graphic variable **ctx** that lets 2D graphics commands to be used in the canvas.
- Lines 12 and 13 create variables to hold the state of the LEDs and initializes them to 0.
- Line 14 calls the function **update_display()** that updates the contents in the canvas.

HTML File Description

- Line 15 makes the websocket connection to the server.
  - **ws://192.168.4.1:81/** is the address.
  - **ws** indicate a websocket type connection.
  - 192.168.4.1 is the IP address
  - 81 is the socket port number.
  - The rest is required as is.
- Line 16 is an event function that sends error messages to the browser console. The browser console is activated in the browser under the developer menu.
- Line 17 sets up an event function that executes when a websocket packet is received at the browser.

HTML File Description

- Line 18 copies the packet contents received to the **dat** variable.
- Line 19 breaks up the packet into 3 elements. The message sent from the server has three parts separated by a space.
  - First element should containt **LED**. This is checked to make sure the proper message is received.
  - The second element is the LED number.
  - The third element contains either **ON** or **OFF**
  - Lines 20 through 28 check the message and sets the **led1_state** and **led2_state** variables to the appropriate value based on the packet content.
  - Line 29 refreshes the canvas with the updated state of th LEDs.

HTML File Description

- Lines 31 - 33 are for the send_command() function. It's a simple function. All it does is send the argument to the websocket server. The argument contains the text from the button.
- Lines 34 – 48 updates the contents of the canvas.
- Line 35 clears the canvas to the background color which is white.
- Line 36 sets the font and size.
- Line 37 sets where the coordinates reside relative to the text to be displayed. It is set to align to the center of the text.
- Line 38 sets the color of the text.
- Line 39 renders the text in the canvas at the specified location.
- Lines 40-42 set the color for the red LED. If **led1_state** is zero, the color will be grey otherwise it will be bright red.
- Line 43 draws a rectangle in the specified color.
- Lines 44 – 47 do the same for the second LED.

Uploading

- First compile and upload the code.
- Once the code is uploaded, select the **Tools** menu and select **ESP8266 Sketch Data Upload**.
- It does take a while to upload the data folder. The duration depends on the size of the file system.
- After everything has uploaded, connect to the webserver and try out the web page.