
WiFi & Client Tags

ECE 476 Advanced Embedded Systems

Jake Glower - Lecture #35

Please visit [Bison Academy](#) for corresponding
lecture notes, homework sets, and solutions



Introduction:

The Pico can connect to an existing WiFi network as a client

- Last lecture

In this lecture, we look at having other clients on this network pass data to the Pico using *tags*.

- Text Inputs: Send a text string back to the Pico
- Number Inputs: Send an integer back to the Pico
- Check Boxes: Select any number of boxes
- Radio Input: Select one box.
- Hyperlink: Send a message using hyperlinks

Text Input	Number Input	Check Box	Radio	Hyperlink
V0 <input type="text" value="1.234"/>	N <input type="text" value="6"/>	<input checked="" type="checkbox"/> Cheese	<input type="radio"/> Vote A	<input type="text" value="ON"/>
V1 <input type="text" value="JohnDoe"/>	M <input type="text" value="157"/>	<input checked="" type="checkbox"/> Ketchup	<input checked="" type="radio"/> Vote B	<input type="text" value="OFF"/>
		<input type="checkbox"/> Mustard	<input type="radio"/> Vote C	<input type="text" value="Toggle"/>
		<input type="checkbox"/> Pickle	<input type="radio"/> Vote D	
		<input type="text" value="Submit"/>	<input type="text" value="Submit"/>	

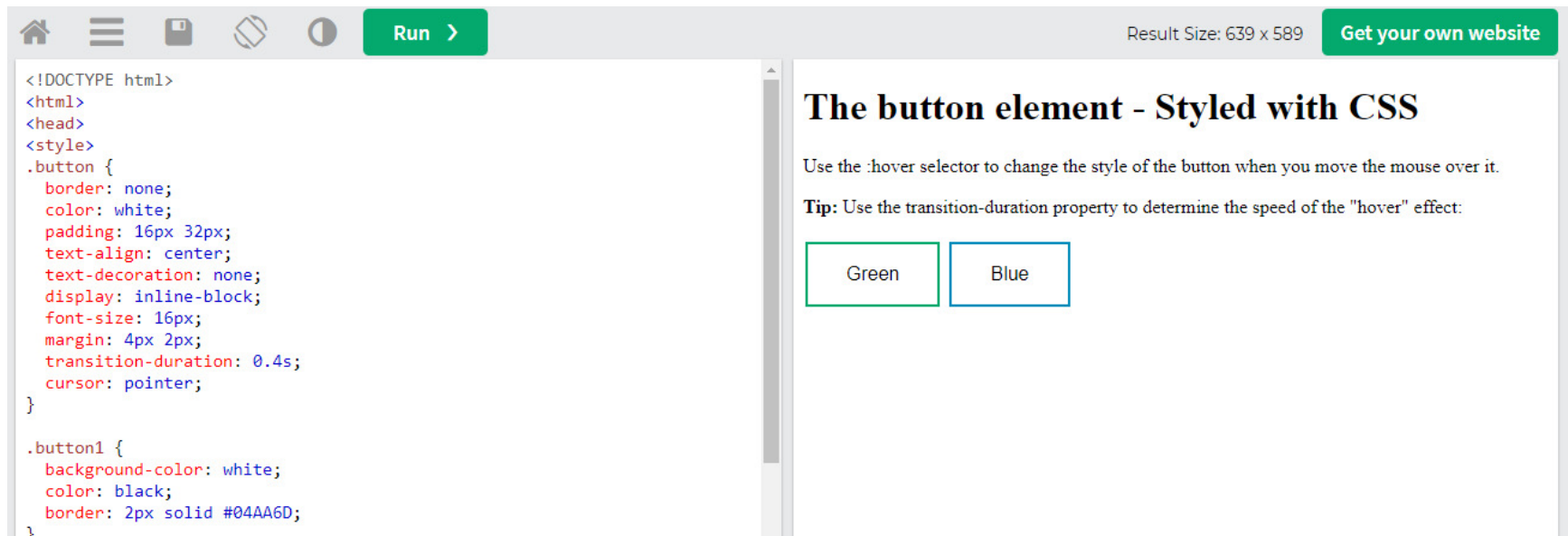
Trick to using Tags:

Write a short html program which send back data from the client.

- Approach used in w3schools
- <https://www.w3schools.com/tags/>

You can test your html code at w3schools

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



The screenshot shows a web editor interface. On the left, there is a code editor with the following HTML code:

```
<!DOCTYPE html>
<html>
<head>
<style>
.button {
  border: none;
  color: white;
  padding: 16px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  margin: 4px 2px;
  transition-duration: 0.4s;
  cursor: pointer;
}

.button1 {
  background-color: white;
  color: black;
  border: 2px solid #04AA6D;
}
```

On the right, the rendered output is displayed. It features a title "The button element - Styled with CSS" and a paragraph: "Use the :hover selector to change the style of the button when you move the mouse over it." Below this is a tip: "Tip: Use the transition-duration property to determine the speed of the 'hover' effect:". At the bottom, there are two buttons: "Green" and "Blue". The "Green" button has a green border, and the "Blue" button has a blue border. The interface also includes a "Run" button and a "Get your own website" button.

Text Inputs

Type in anything you like in the box

- Words
- Numbers

When you press *select*, the data is sent to the Pico

Display Text Input Fields

N0:

N1:

Enter a number than press Submit.

html code

First, start with the html code to generate this display

- same as lecture #33

```
<!DOCTYPE html>
<html>
<body>

<h1>Display Text Input Fields</h1>

<form action="/action_page.php">
  <label for="fname">N0: </label>
  <input type="text" id="N0" name="N0"><br><br>
  <label for="lname">N1: </label>
  <input type="text" id="N1" name="N1"><br><br>
  <input type="submit" value="Submit">
</form>

<p>Enter a number than press Submit.</p>

</body>
</html>
```

web_page() routine

Next, write a subroutine to read this file

- Same as lecture #33
- Read as a text file
- Replace carriage return, line-feeds
- Return as a text string

```
def web_page():
    f = open("33_text.html")
    #f = open("33_CheckBox.html")
    #f = open("33_Number.html")
    #f = open("33_Radio.html")
    x = f.read()
    x = x.replace('\r\n', ' ')
    return(x)
```

Main Routine: Connect as a client

Connect to the WiFi network as a client and open a socket for the Pi Pico

- Same as lecture #34
- Slightly different from lecture #33
- Note: *netman.py* from Pepe80.com needs to be loaded onto your Pico board,
 - same as lecture #34

```
ssid = 'xxxx'
password = 'xxxx'
country = 'US'

wifi_connection = netman.connectWiFi(ssid,password,country)

# Open socket
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
wlan = socket.socket()
wlan.bind(addr)
wlan.listen(1)

print('listening on', addr)
```

Main Routine: Main Loop

- Wait for a ping from someone
- Keep checking until it is a *favicon* ping
 - Prevents double-reads
- Same as lecture #33

```
while(1):
    flag = 0
    while(flag == 0):
        conn, addr = wlan.accept()
        request = conn.recv(1024)
        request = request.decode('utf-8')
        if(request.find('favicon') > 0):
            print('-----')
            print('Got a connection from %s' % str(addr))
            flag = 1
        else:
            response = web_page()
            conn.send(response)
            conn.close()
```

Main Loop: Parse the Message

- Pull out the message
- Store the fields in array *msg[]*
- Update the web page
- Close the connection
 - same as lecture #33

```
n = request.find('php?')+4
request = request[n:]
n = request.find('\r\n')
request = request[0:n]
msg = []
while(request.find('&')>0):
    n = request.find('&')
    msg.append(request[0:n])
    request = request[n+1:]
msg.append(request)
for i in range(0,len(msg)):
    print('msg['+str(i)+'] = '+str(msg[i]))
response = web_page()
conn.send(response)
conn.close()
```

Shell Window

The data from the other client can then be seen in the shell window.

- Note that the data is a string: numbers and text are valid inputs.

```
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
waiting for connection...
waiting for connection...
waiting for connection...
waiting for connection...
waiting for connection...
connected
ip = 192.168.43.174
listening on ('0.0.0.0', 80)
-----
Got a connection from ('192.168.43.19', 49338)
msg[ 0 ] = N0=1234
msg[ 1 ] = N1=5678
-----
Got a connection from ('192.168.43.19', 49414)
msg[ 0 ] = N0=12345.678
msg[ 1 ] = N1=abcdefg
```

Number Fields

Input an integer

- Type in the number
- Use the arrows

Press *Submit* when ready

Display a Number Field

Red (between 0 and 255):

Green (between 0 and 255):

Blue (between 0 and 255):

Number Field: html Code

- Same as lecture #33

```
<!DOCTYPE html>
<html>
<body>

<h1>Display a Number Field</h1>

<form action="/action_page.php">
  <label for="red">Red (between 0 and 255):</label>
  <input type="number" id="red" name="r" min="0" max="255">
  <br>

  <label for="green">Green (between 0 and 255):</label>
  <input type="number" id="green" name="g" min="0" max="255">
  <br>

  <label for="blue">Blue (between 0 and 255):</label>
  <input type="number" id="blue" name="b" min="0" max="255">
  <br>
  <input type="submit">

</form>

</body>
</html>
```

Display a Number Field

Red (between 0 and 255):

Green (between 0 and 255):

Blue (between 0 and 255):

Shell Window

The main routine stays the same as before

- Just change which file is read in *web_page()*

The results show up the shell window

```
-----  
Got a connection from ('192.168.43.19', 50132)  
msg[ 0 ] = r=15  
msg[ 1 ] = g=20  
msg[ 2 ] = b=23
```

Check Box

A Check Box lets you select any number of items from a list and send those to the Pico.

- Same as lecture #33

Show Checkboxes

- Red On
- Green On
- Blue On

Submit

Check Box: html code

- Same as lecture #33

```
<!DOCTYPE html>
<html>
<body>

<h1>Show Checkboxes</h1>

<form action="/action_page.php">
  <input type="checkbox" id="r" name="color1" value="Red">
  <label for="color1"> Red On</label><br>
  <input type="checkbox" id="g" name="color2" value="Green">
  <label for="color2"> Green On</label><br>
  <input type="checkbox" id="b" name="color3" value="Blue">
  <label for="color3"> Blue On</label><br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

CheckBox: Shell Window

The main routine remains unchanged

- Just change which file is loaded in *web_page()*

When you press *Submit*, the checked boxes are returned

- If no buttons are checked, you get a null response (first entry)

```
-----  
Got a connection from ('192.168.43.19', 49911)  
msg[ 0 ] = color1=Red  
msg[ 1 ] = color3=Blue  
-----  
Got a connection from ('192.168.43.19', 49925)  
msg[ 0 ] = color2=Green  
-----  
Got a connection from ('192.168.43.19', 49942)  
msg[ 0 ] = color1=Red  
msg[ 1 ] = color2=Green  
msg[ 2 ] = color3=Blue
```

Radio Buttons

Select one item from a list

- If you select another, the previous selection is removed

Press *Select* when done

Example:

- Voting for a candidate
- (can only select one)

Display Radio Buttons

Favorite Pet:

- Cats
- Dogs
- Ferrets

Least Favorite Pet:

- Lions
- Tigers
- Bears

Submit

Radio Buttons: html code

```
<!DOCTYPE html>
<html>
<body>

<h1>Display Radio Buttons</h1>

<form action="/action_page.php">
  <p>Favorite Pet:</p>
  <input type="radio" id="cats" name="like" value="Cats">
  <label for="cats">Cats</label> <br>
  <input type="radio" id="dogs" name="like" value="Dogs">
  <label for="dogs">Dogs</label> <br>
  <input type="radio" id="ferret" name="like" value="Ferrets">
  <label for="ferret">Ferrets</label> <br>
  <p>Least Favorite Pet:</p>
  <input type="radio" id="lion" name="dislike" value="Lions">
  <label for="lion">Lions</label> <br>
  <input type="radio" id="tiger" name="dislike" value="Tigers">
  <label for="tiger">Tigers</label> <br>
  <input type="radio" id="bear" name="dislike" value="Bears">
  <label for="bear">Bears</label> <br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

Radio Buttons: Shell Window

The selected items are returned each time you press *Submit*

- Shows up in the Shell window
- Also available to the Python program

```
-----  
Got a connection from ('192.168.43.19', 50391)  
msg[ 0 ] = like=Ferrets  
msg[ 1 ] = dislike=Lions  
-----  
Got a connection from ('192.168.43.19', 50404)  
msg[ 0 ] = like=Cats  
msg[ 1 ] = dislike=Tigers
```

Hyperlink Buttons

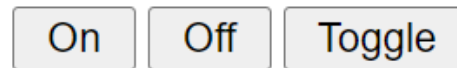
Hyperlinks normally take you to another web page

They can also be used to send a reply when clicked:

- `light_on`
- `light_off`
- `light_toggle`

Show a Push Button

Click a Button.



Hyperlink Button: html code

- Same as lecture #33
- aaaaa is a place holder for the IP-Address

```
<!DOCTYPE html>
<html>
<body>

<h1>Show a Push Button</h1>

<p>Click a Button.</p>
<form>
<p><a href="http://aaaaa/light_on"><input type="button" value=" On "></a>
<a href="http://aaaaa/light_off"><input type="button" value=" Off "></a>
<a href="http://aaaaa/light_toggle"><input type="button" value=" Toggle
"></a>
</p>
</form>

</body>
</html>
```

Hyperlink: *web_page()*

Uses some tricks from previous lectures:

- aaaaa is the IP-address of the host
- bbbbb is the status of the LED (ON or OFF)

Read the file

- Replace aaaaa with the IP-Address
- Replace bbbbb with the LED status

```
def web_page(ip_address, OnOff):
    f = open("33 Hyperlink.html")
    x = f.read()
    x = x.replace('\r\n', ' ')
    x = x.replace('aaaaa', ip_address)
    x = x.replace('bbbbb', OnOff)
    return(x)
```

Hyperlink: Main Routine

Save the IP-Address when connecting

- The *web_page()* routine needs this information

```
ssid = 'Galaxy S9+03ac'  
password = 'wnor7871'  
country = 'US'  
wifi_connection = netman.connectWiFi(ssid,password,country)  
IP_Address = wifi_connection[0]  
  
# Open socket  
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]  
wlan = socket.socket()  
wlan.bind(addr)  
wlan.listen(1)  
  
print('listening on', addr)
```

Down in the main loop,

- Keep looping until you get a ping starting with *favicon*

```
while(1):
    flag = 0
    while(flag == 0):
        conn, addr = s.accept()
        request = conn.recv(1024)
        request = request.decode('utf-8')
        if(request.find('favicon') > 0):
            flag = 1
        else:
            response = web_page(IP_Address, OnOff[LED.value()])
    )
    conn.send(response)
    conn.close()
```

Once you find this message, locate the string which starts with *Referer*:

```
Referer: //https://192.168.4.1/light_on
```

Keep the message past this point to the `<cr><lf>` (`/r/n`)

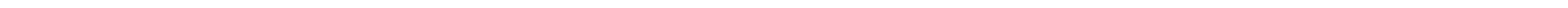
```
n = request.find('Referer:')+9
request = request[n:]
n = request.find('\r\n')
request = request[0:n]
```

At this point, the message will look something like

```
//https://192.168.4.1/light_on
```

Strip off everything to the left of the back-slash's

```
for i in range(0,10):
    n = request.find('/')+1
    if(n>0):
        request = request[n:]
print(request)
```



Now do something with the message

- light_on
- light_off
- light_toggle

```
if(request == 'light_on'):
    LED.value(1)
if(request == 'light_off'):
    LED.value(0)
if(request == 'light_toggle'):
    LED.toggle()
response = web_page(IP_Address, LED.value() )
conn.send(response)
conn.close()
```

The net results is

- Every time you click on Turn_ON, the LED turns on
- Every time you click on Turn_OFF, the LED turns off

Shell Window:

```
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
AP Mode Is Active, You can Now Connect
IP Address To Connect to:: 192.168.4.1
Channel 3

light_on
light_off
light_toggle
light_on
light_off
light_toggle
```

Summary:

In this lecture, techniques for having our Pico act as a client on a WiFi network was presented. Using tags allowss you to

- Connect to your Pico through an exiting WiFi router
- Input binary numbers, turning an LED on or off from your cell phone or browser,
- Input floating point numbers, allowing you to vary the brightness of an LED, speed of a motor, etc, and
- Select from several options using various tags.

Many more tags exist and are presented in w3schools. The ones presented here are the ones I was able to get to work with a Pi-Pico. With some effort, you can probably get the other ones to work as well.

References

- pepe80.com
- https://www.w3schools/tags/att_input_type.asp

