WiFi & AP Tags

ECE 476 Advanced Embedded Systems

Jake Glower - Lecture #33

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

Introduction:

The last lecture looked at

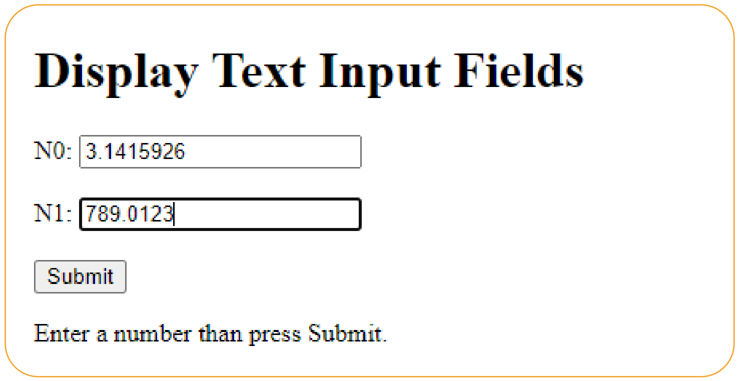
- Setting up a local area network and
- Displaying data to clients on that network.

In this lecture, we look having clients send data back to the Pico

- Using AP tags for
- Text, binary, and float data
- To control lights, voltages, speeds, etc.

AP Tags presented in clude

- Text Fields
- Number Fields
- Check Boxes
- Push Buttons

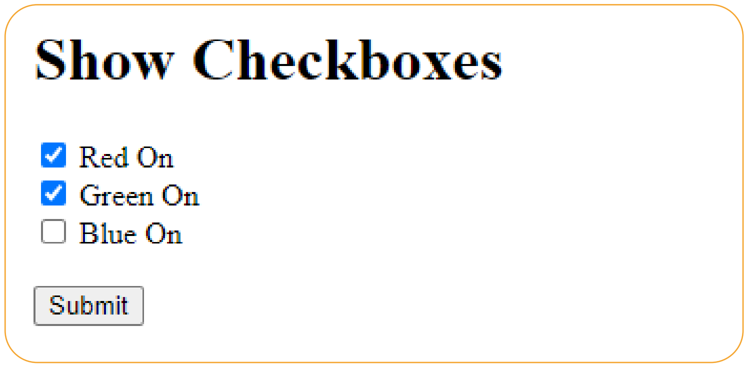


Display Text Input Fields

N0: 3.1415926

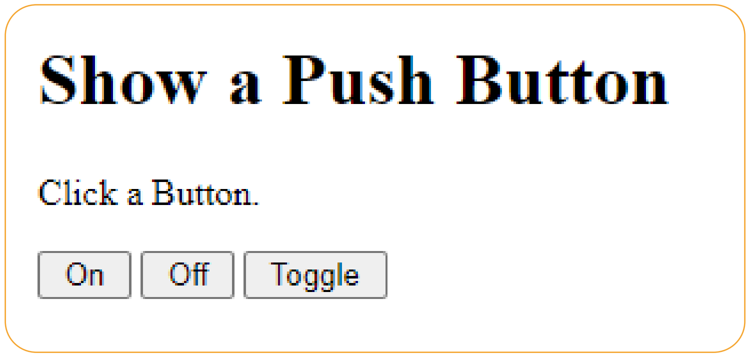
N1: 789.0123

Enter a number than press Submit.



Show Checkboxes

- [x] Red On
- [x] Green On
- [ ] Blue On



Show a Push Button

Click a Button.

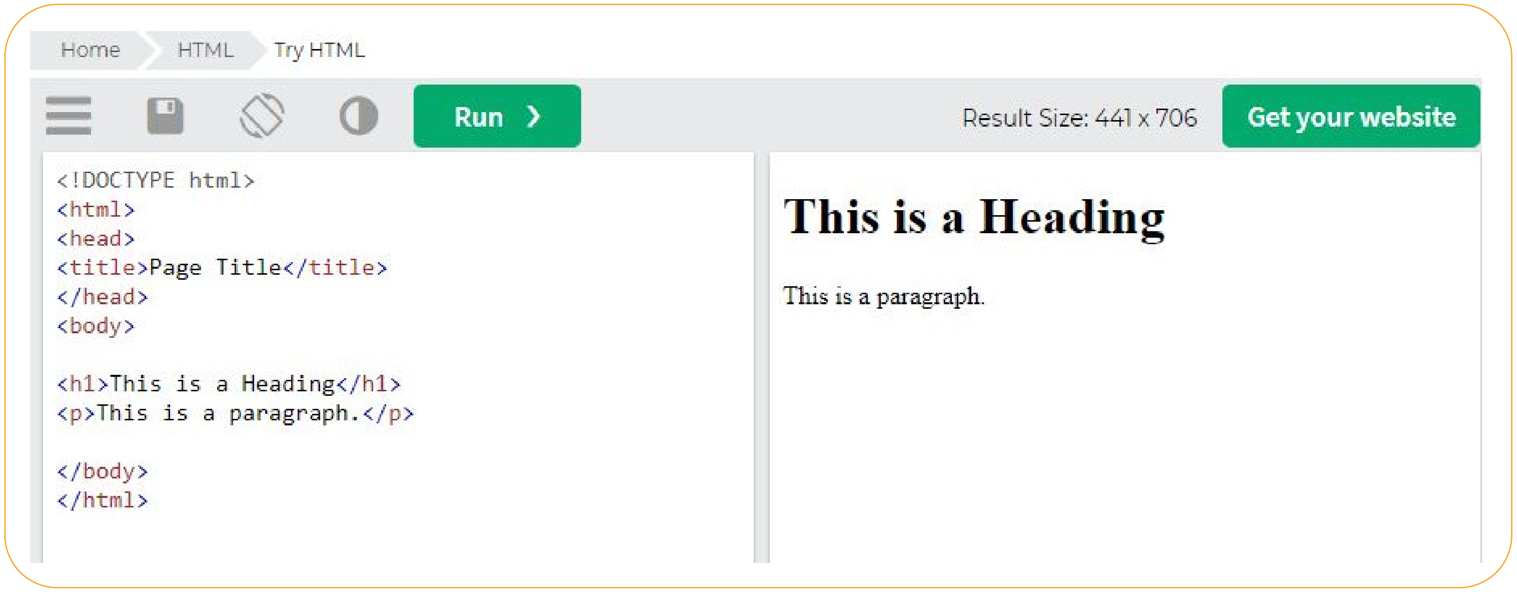
w3schools

The technique used here is to write a short html program which send back data from the client. The core of the code presented here is based upon code written at

https://www.w3schools/tags/

which is an excellent site for learning about tags. In addition, you can write and test out your html code in interactive windows at

https://www.w3schools/tags/att_input_type



```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

This is a Heading

This is a paragraph.

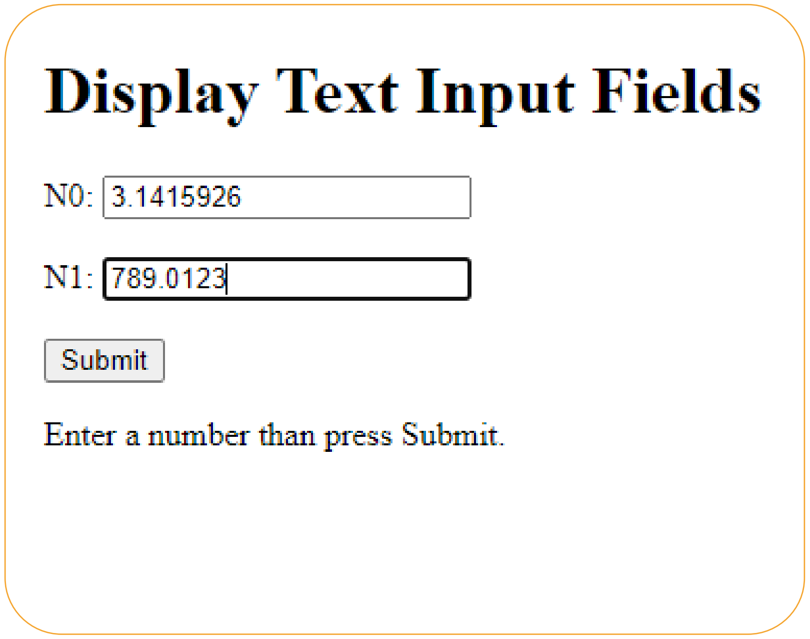
Text Inputs

Text inputs read the data as a string

- Can be words
- Can be integers
- Can be floating point numbers

When you hit *Submit*, the data is sent back to the host

Text strings can then be converted to numbers using a *int()* or *float()* command



Display Text Input Fields

N0: 3.1415926

N1: 789.0123

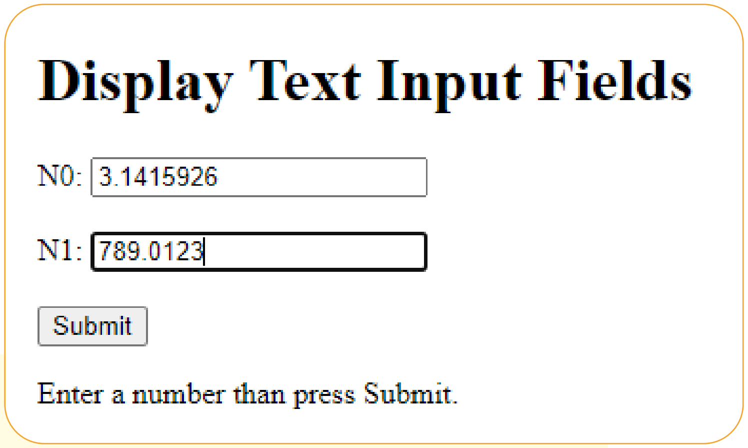
Submit

Enter a number than press Submit.

Text Tag: html code:

First, start with the html for this display

- Save this on your Pi-Pico
- File Name: *33_text.html*



Display Text Input Fields

N0: 3.1415926

N1: 789.0123

Enter a number than press Submit.

```
<!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>
```

Step 2: ***web_page()*** routine

The main routine which uses this file is as follows:

First, a subroutine *web_page()*

- reads the file *33_text.html* and
- converts it to a string.

The main routine sends this to the client each ping

- Same as *HelloWorld* example from before

```
import network
import time
import socket

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

Step 3: Open up an AP network

- same as previous lecture.

```
ssid = 'Pico-Network'
password = 'PASSWORD'

ap = network.WLAN(network.AP_IF)
ap.config(ssid=ssid, password=password)
ap.active(True)

while ap.active() == False:
    pass
print('AP Mode Is Active, You can Now Connect')
print('IP Address To Connect to:: ' + ap.ifconfig()[0])

print('Channel',  ap.config('channel'))

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)
```

Step 4: Main Loop

The bare minimum for this main loop

- Waits for a response from a client (*s.accept()*)
- Once received, the response is printed (saved in variable *request*),
- The web page is refreshed (*conn.send()*), and
- The connection is closed

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

Shell Window:

- Response with this minimal setup for the main loop
- *GET /favicon* is the start of the reply
- *Referer:* is the start of the message from the client to the Pico

```
Got connection from ('192.168.4.17', 59475)

GET /favicon.ico HTTP/1.1
Host: 192.168.4.1
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0
Safari/537.36
Accept:
image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://192.168.4.1/action_page.php?N0=123.456&N1=789.012
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

Improved Main Loop

```
Referer: http://192.168.4.1/action_page.php?N0=123.456&N1=789.012
```

Strip out stuff I don't care about

- First, look for the string *Referer:*
  - This marks the start of the response.
- Look for the string php?
  - This marks the start of the message
- Look for a carriage return and line feed (*/r/n*)
  - The end of the response.
- Next, look for *&* symbols
  - Denotes different fields

Pull out each field

Improved Main Loop

```
while(1):
  conn, addr = s.accept()
  print('Got a connection from %s' % str(addr))
  request = conn.recv(1024)
  request = request.decode('utf-8')
  n = request.find('Referer:')
  request = request[n:]
  n = request.find('php?')+4
  request = request[n:]
  n = request.find('\r\n')
  request = request[0:n]
  msg = []
  for i in range(0,10):
      if(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[',i,'] = ', msg[i])
  response = web_page()
  conn.send(response)
  conn.close()
```

Resulting Shell Window

The results each time you press *Submit* then show up in the Shell window

- Also available for the Pico to take actions
- Text and numbers are all OK to send

The shell window then shows the data each time you press *submit*

- note: spaces show up as + signs

```
Got connection from ('192.168.4.17', 59482)
msg[ 0 ] =  N0=123.456
msg[ 1 ] =  N1=789.012

Got connection from ('192.168.4.17', 59482)
msg[ 0 ] =  N0=3.14159
msg[ 1 ] =  N1=23.456

Got connection from ('192.168.4.17', 59482)
msg[ 0 ] =  N0=The+quick+red+fox+jumped
msg[ 1 ] =  N1=over+the+lazy+dof%27s+head
```

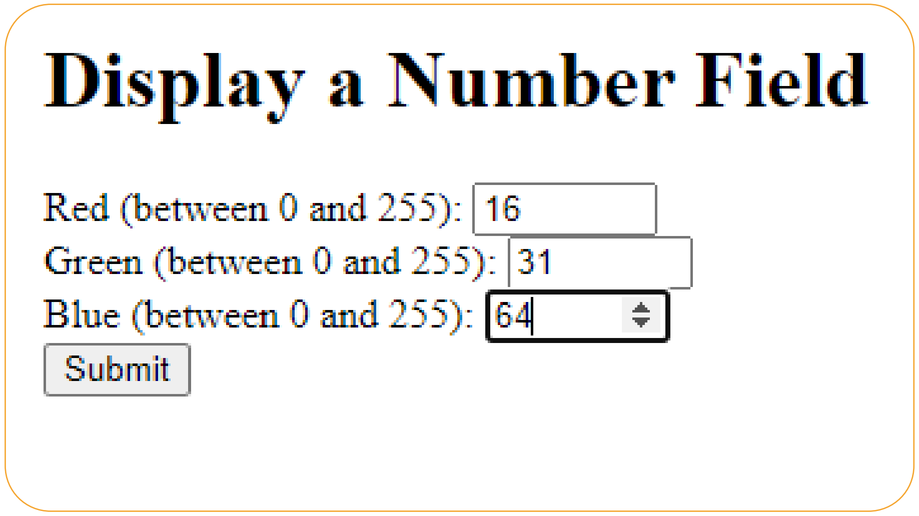
Number Fields

Send an integer back to the host

- Can type in the integer
- Can use arrow up/down
- Submit sends the data

Note:

- Previous solution works
- Just change the html file

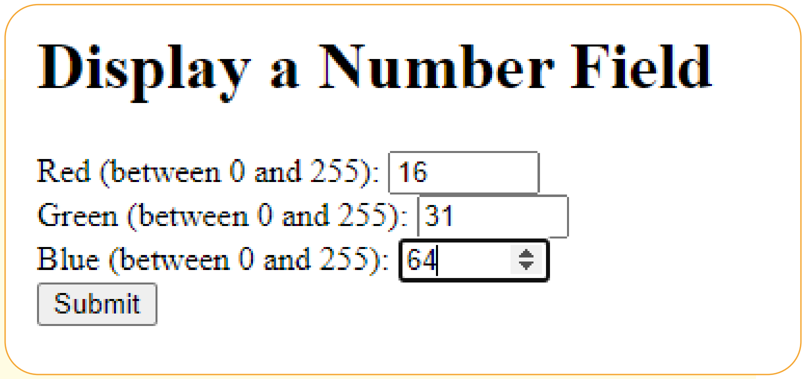


Display a Number Field

Red (between 0 and 255): 16
Green (between 0 and 255): 31
Blue (between 0 and 255): 64

Number Field: html code

```
<!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): 16
Green (between 0 and 255): 31
Blue (between 0 and 255): 64

Number Field: Shell Window

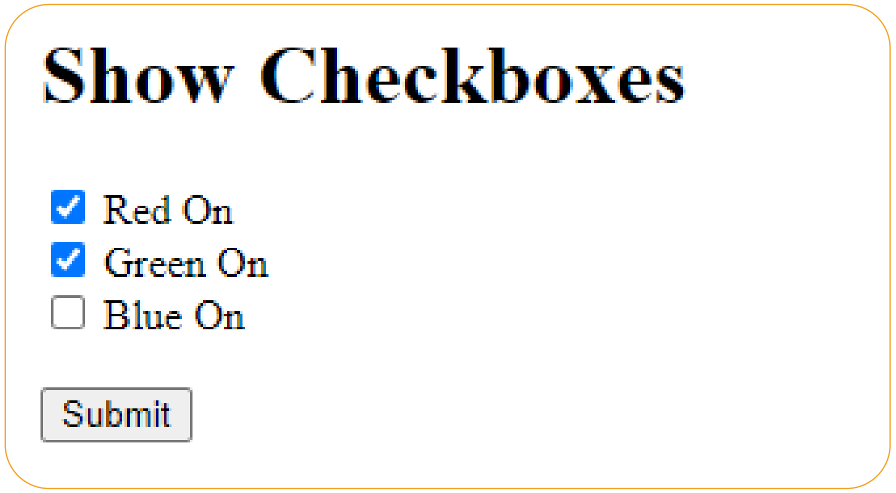
- Data is sent back each time you press *Submit*

```
--------------------
Got a connection from ('192.168.4.17', 57050)
msg[ 0 ] =  r=16
msg[ 1 ] =  g=31
msg[ 2 ] =  b=64
--------------------
Got a connection from ('192.168.4.17', 57051)
msg[ 0 ] =  r=35
msg[ 1 ] =  g=255
msg[ 2 ] =  b=88
--------------------
Got a connection from ('192.168.4.17', 64852)
msg[ 0 ] =  r=35
msg[ 1 ] =  g=255
msg[ 2 ] =  b=88
--------------------
Got a connection from ('192.168.4.17', 57053)
msg[ 0 ] =  r=0
msg[ 1 ] =  g=0
msg[ 2 ] =  b=0
```

Check Box

Check Box

- Gives you a list
- You can check any number of boxes
- Then hit *Submit*



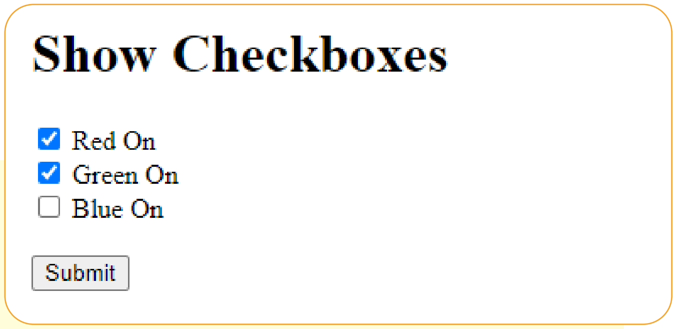
Show Checkboxes

[✓] Red On
[✓] Green On
[ ] Blue On

The boxes checked are sentto the host

CheckBox: html Code

```
<!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>
```



Show Checkboxes

[✓] Red On
[✓] Green On
[ ] Blue On

Check Box: Returned Data

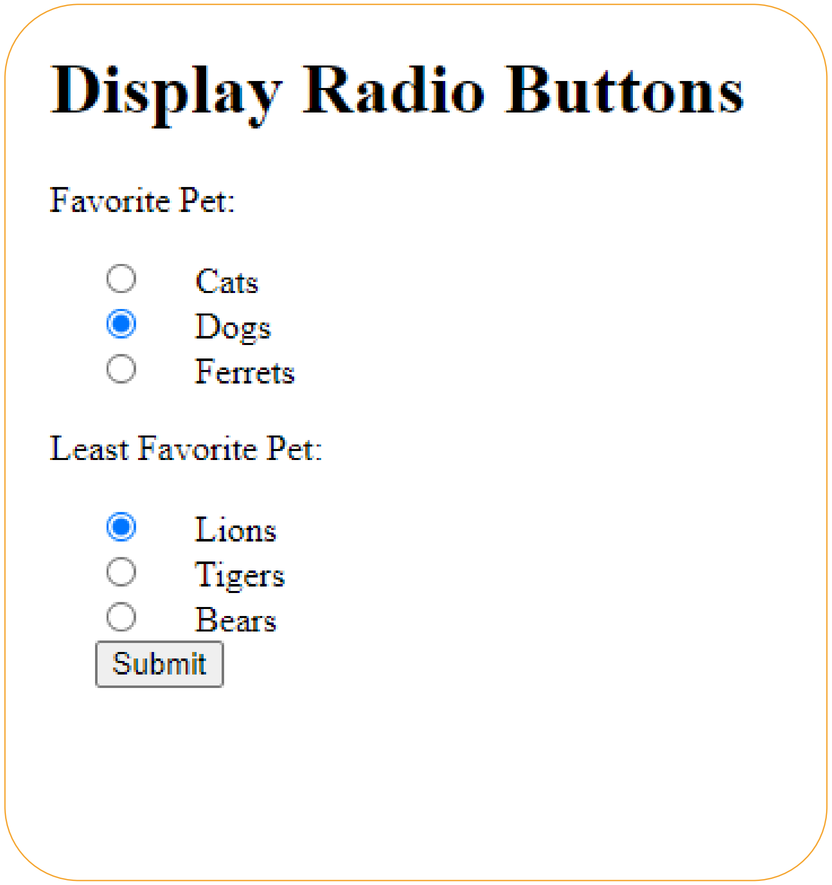
- Shell Window
- No buttons gives an empty message
- Multiple boxes give multiple results

```
Got a connection from ('192.168.4.17', 59069)
msg[ 0 ] =
-----
Got a connection from ('192.168.4.17', 59071)
msg[ 0 ] =  color1=Red
-----
Got a connection from ('192.168.4.17', 59073)
msg[ 0 ] =  color2=Green
-----
Got a connection from ('192.168.4.17', 57509)
msg[ 0 ] =  color1=Red
msg[ 1 ] =  color2=Green
msg[ 2 ] =  color3=Blue
```

Radio Buttons

Give a list

- Only one item from the list can be selected
- If you select another, the previous one is deselected
- Checked items are returned on *Submit*



Display Radio Buttons

Favorite Pet:

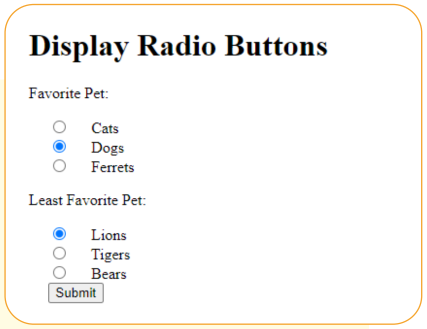
○ Cats
● Dogs
○ Ferrets

Least Favorite Pet:

● Lions
○ Tigers
○ Bears

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>
```



Display Radio Buttons

Favorite Pet:

○ Cats
● Dogs
○ Ferrets

Least Favorite Pet:

● Lions
○ Tigers
○ Bears

Radio Buttons: Returned Data

- Shell Window
- No buttons gives an empty message

```
>> %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
-----
Got a /favicon response from ('192.168.4.17', 56946)
msg[ 0 ] =  like=Cats
msg[ 1 ] =  dislike=Lions
-----
Got a /favicon response from ('192.168.4.17', 56951)
msg[ 0 ] =  like=Cats
msg[ 1 ] =  dislike=Tigers
-----
Got a /favicon response from ('192.168.4.17', 56954)
msg[ 0 ] =  like=Ferrets
msg[ 1 ] =  dislike=Lions
```

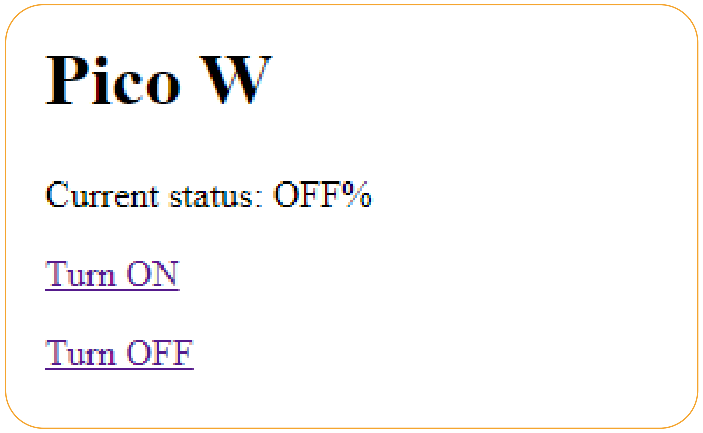
Hyperlink

Normally used to return a web address

- Can also be used to return data

Click on the hyper links

- Each click sends data back to the Pico



Pico W

Current status: OFF%

Turn ON

Turn OFF

Hyperlink: html code

- Need to know the IP address
  - Use a dummy variable: *aaaaa*
- Also display the LED status
  - Another dummy variable: *bbbbb*

```
<!DOCTYPE html>
<html>

<head> <title>Pico W</title> </head>
<body> <h1>Pico W</h1>
<p>Current status: bbbbb /p>
<p><a href="http://aaaaa/light_on">Turn ON</a></p>
<p><a href="http://aaaaa/light_off">Turn OFF</a></p>
</body>

</html>
```

Hyperlink: ***web_page()***

Pass data to insert into the web page:

- aaaaa = IP address
- bbbbb = LED status

Replace dummy variables

```
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 Loop

Only respond once each click

- Only acknowledge messages starting with *favicon*
- Echo back and close the connection if not found
- Should probably be placed in previous code too

```
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()
```

HyperLink: Main Loop

Message looks like:

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

Look for *Referer:* and */r/n* to get the message

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

Strip off everything prior to the back-slash's

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

At this point, you have the message

- *light_on* or *light_off*

Turning on / off the LED

Based upon what you receive, you can turn on and off the LED:

```
    if(request == 'light_on'):
        LED.value(1)
    if(request == 'light_off'):
        LED.value(0)
    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

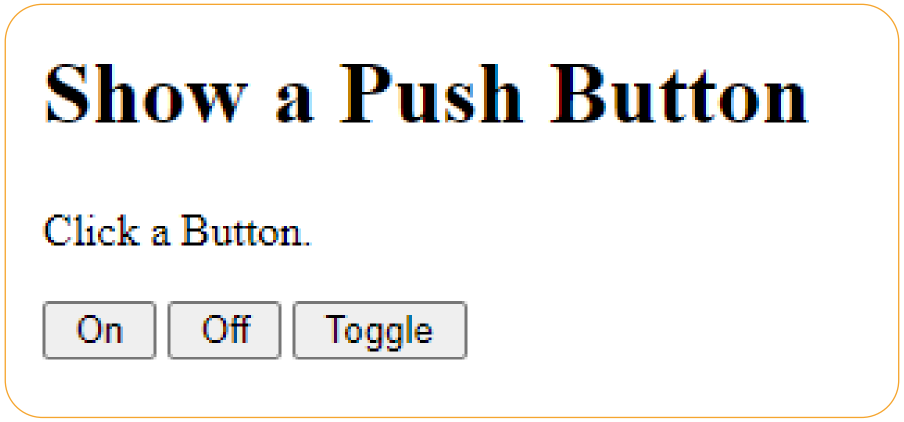
Note: The LED status is always one click in the past

- Not sure how to fix this

Hyperlink with Buttons

Same as before

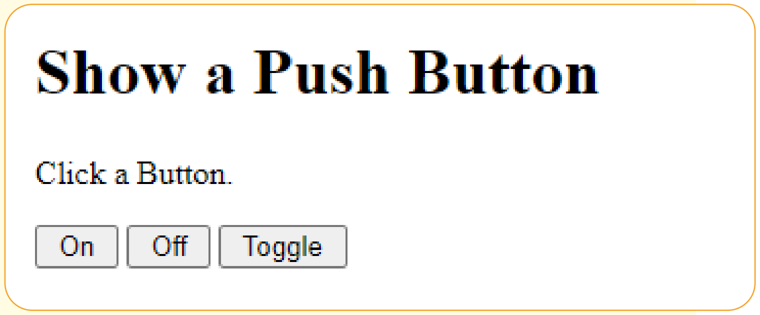
- Prettier display
- Button rather than text


Show a Push Button

Click a Button.

Push Buttons: html code

```
<!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>
```


Show a Push Button

Click a Button.

Push Button: Returned Data

- Shows up in *shell* window
- Also available for main routine
- Same code to turn on and off the LED

```
>>> %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 a client in AP mode send data back to the host were presented. This allows you to

- Connect to your Pico even if there is no internet present
- 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:

- https://www.w3schools/tags/att_input_type