Week 9: Homework 2: Blinking LED three times every 5 seconds using Websocket server

https://hc.labnet.sfbu.edu/~henry/npu/classes/embed/raspberry_pi/slide/exercise_raspberry_pi.html

Q33 ==>Blinking LED three times every 5 seconds using Websocket server

1. Blinking LED three times every 5 seconds using Websocket server .
   - Please modify Blinking LED Websocket server to let a LED blink twice every 3 seconds
   - References
     - 2022 Fall

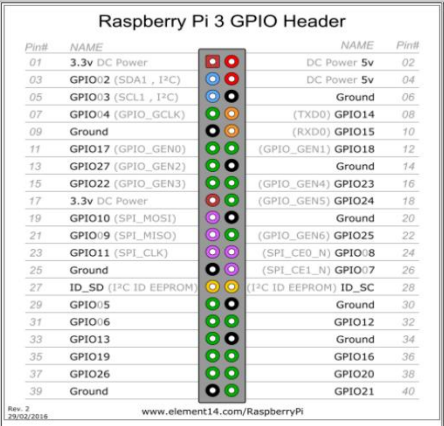
Step 1: Install Package

Install websocket

```
$ npm install websocket
$ npm install onoff
$ pigpiod -v
$ sudo apt-get update
$ sudo apt-get update
$ sudo apt-get install pigpio
$ npm install pigpio
```

Step 2: Connecting Raspberry Pi and LED light

Reference:

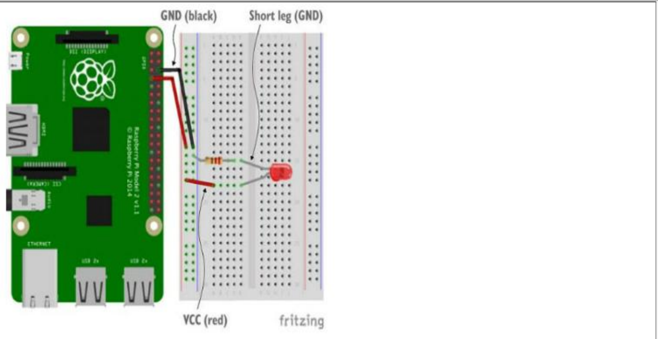


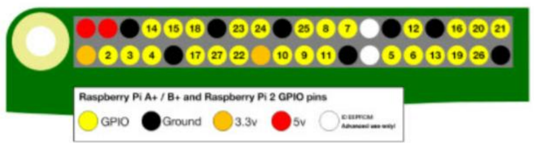
Raspberry Pi 3 GPIO Header

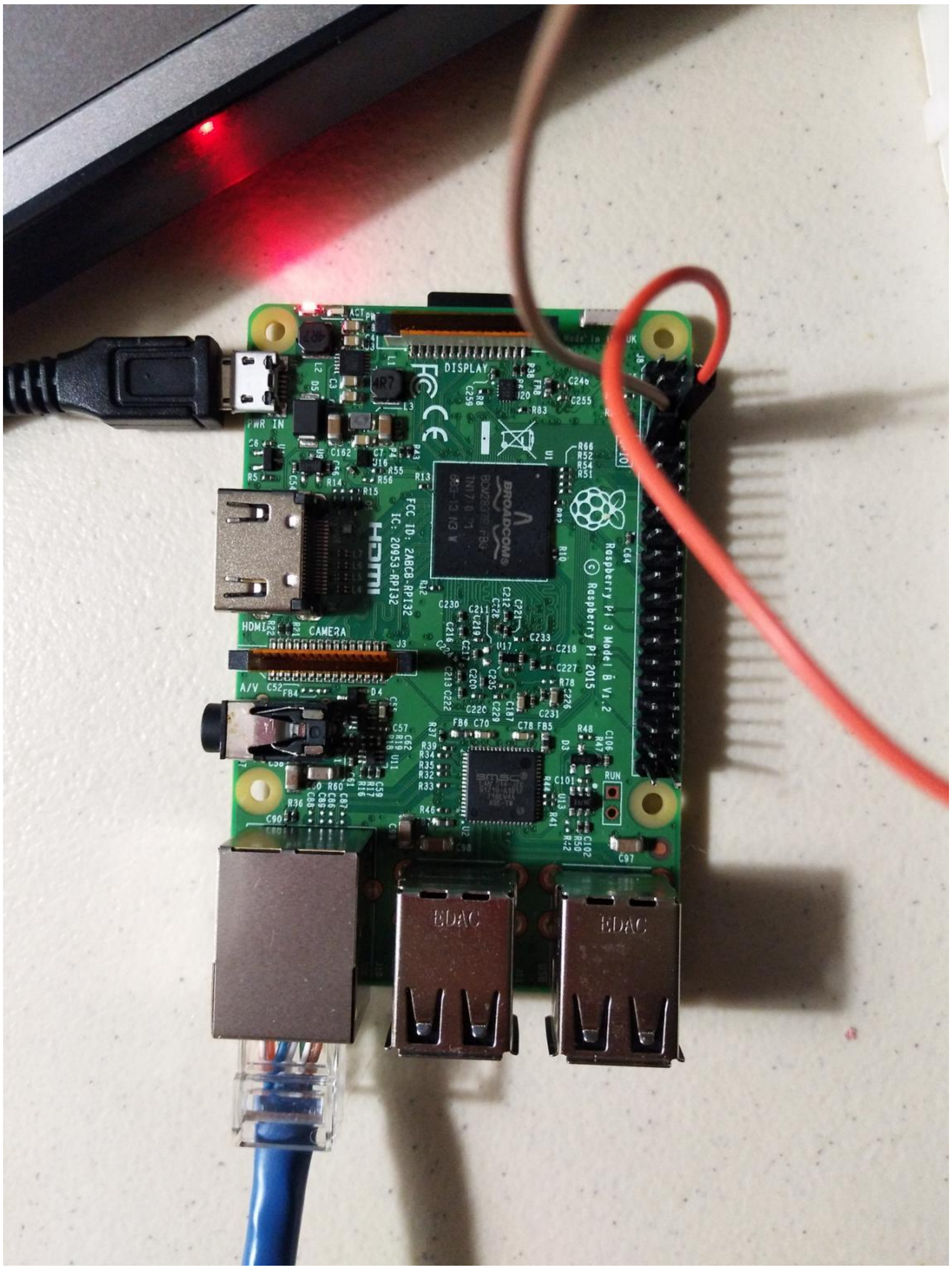
Pin#	NAME			NAME	Pin#
01	3.3v DC Power	🟥	🟥	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	🟦	🟥	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	🟦	⚫	Ground	06
07	GPIO04 (GPIO_GCLK)	🟩	🟧	(TXD0) GPIO14	08
09	Ground	⚫	🟧	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	🟩	🟩	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	🟩	⚫	Ground	14
15	GPIO22 (GPIO_GEN3)	🟩	🟩	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	🟥	🟩	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	🟪	⚫	Ground	20
21	GPIO09 (SPI_MISO)	🟪	🟩	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	🟪	🟪	(SPI_CE0_N) GPIO08	24
25	Ground	⚫	🟪	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	🟨	🟨	(I²C ID EEPROM) ID_SC	28
29	GPIO05	🟩	⚫	Ground	30
31	GPIO06	🟩	🟩	GPIO12	32
33	GPIO13	🟩	⚫	Ground	34
35	GPIO19	🟩	🟩	GPIO16	36
37	GPIO26	🟩	🟩	GPIO20	38
39	Ground	⚫	🟩	GPIO21	40

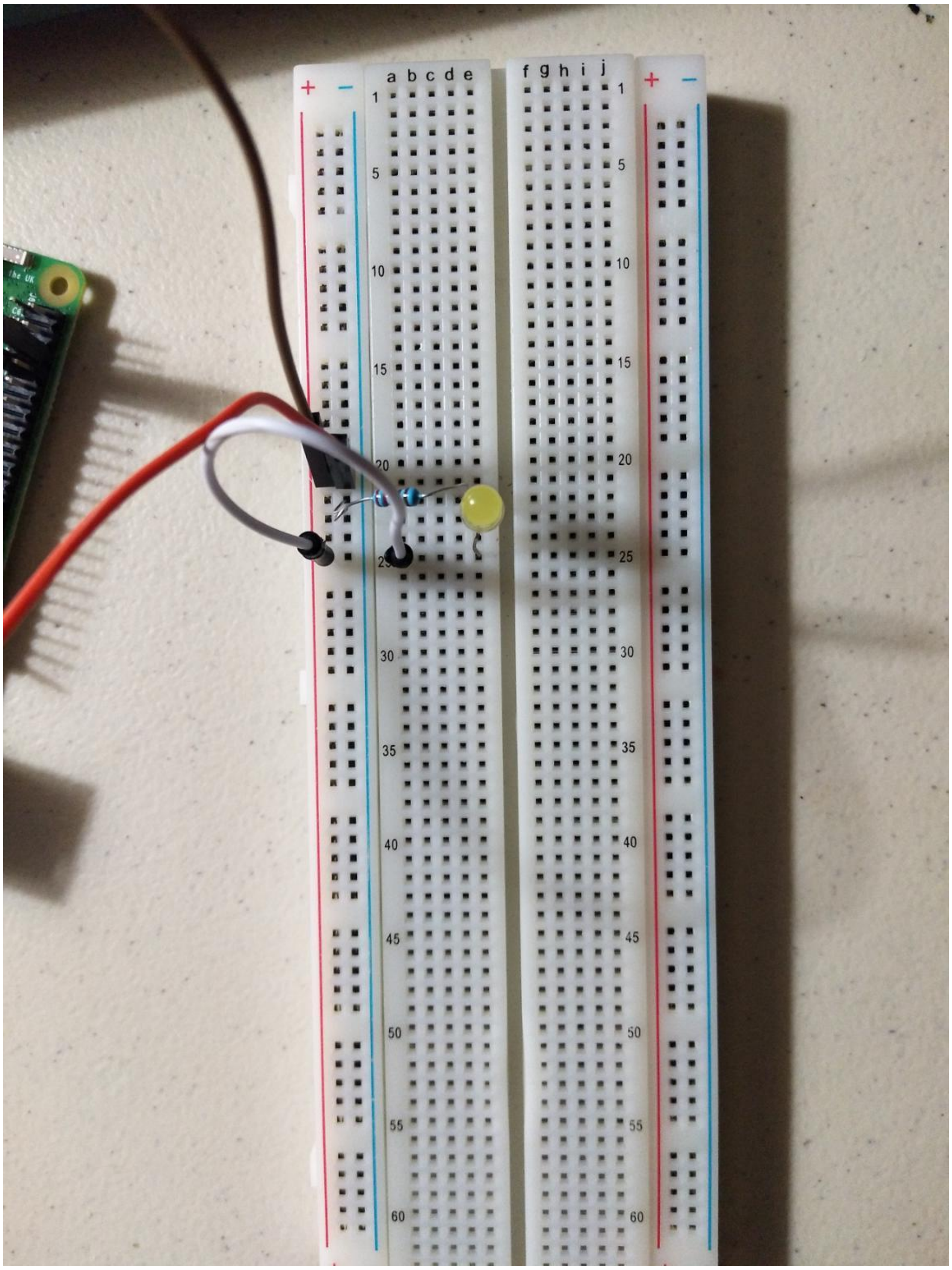
Rev. 2 29/02/2016

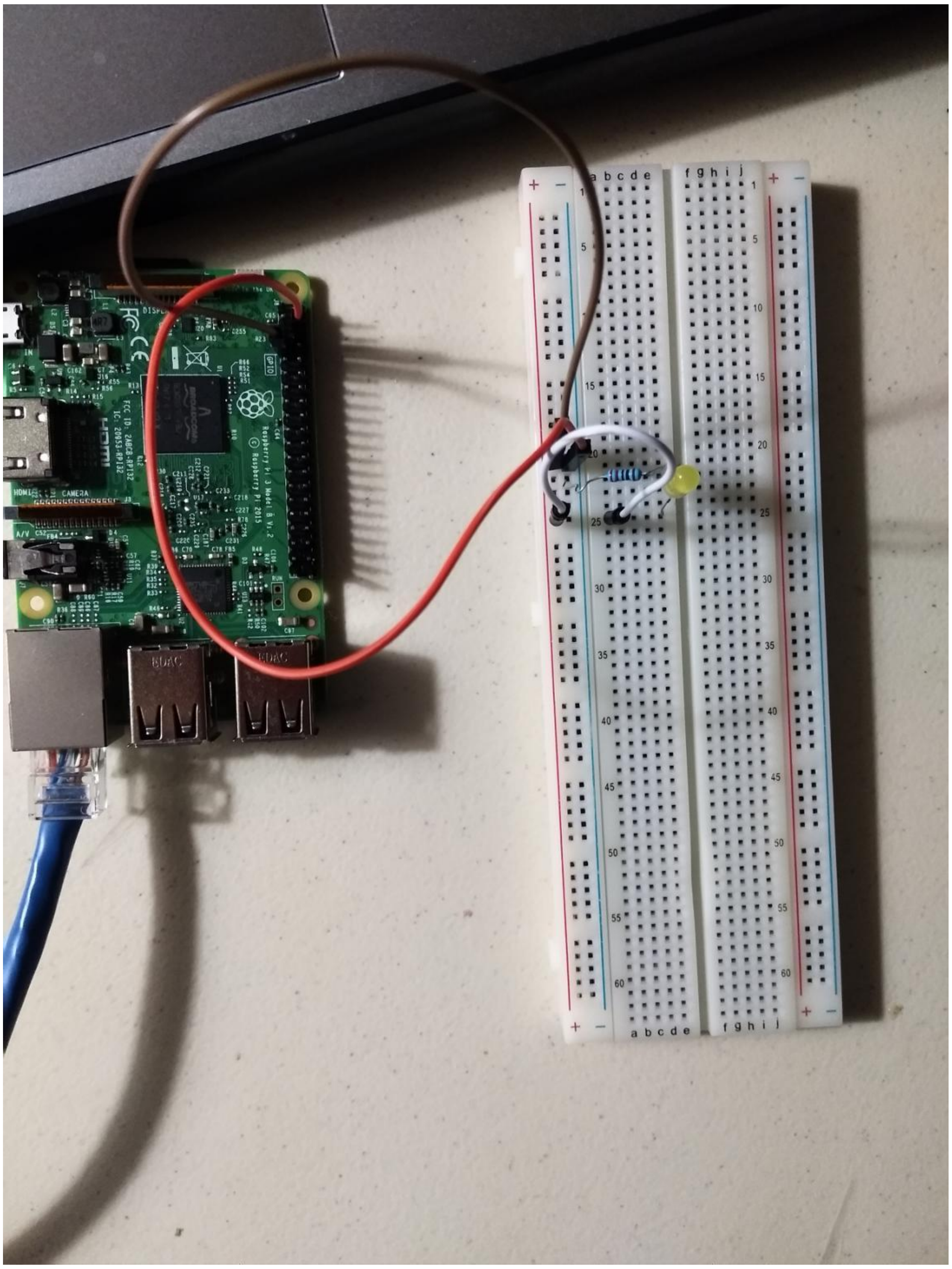
www.element14.com/RaspberryPi




14
15
18
23
24
25
8
7
12
16
20
21
2
3
4
17
27
22
10
9
11
5
6
13
19
26
Raspberry Pi A+ / B+ and Raspberry Pi 2 GPIO pins
GPIO
Ground
3.3v
5v
ID EEPROM Advanced use only!


Raspberry Pi 3 Model B V1.2
© Raspberry Pi 2015
BROADCOM
HDMI
CAMERA
DISPLAY
PWR IN
A/V
RUN
EDAC
EDAC
FCC ID: 2ABCB-RPI32
IC: 20953-RPI32


a b c d e
f g h i j
+ −
+ −
1 5 10 15 20 25 30 35 40 45 50 55 60
1 5 10 15 20 25 30 35 40 45 50 55 60



Step 3: Prepare code:

websocket_blink1.js

```
#!/usr/bin/env node

// - Node.js installation

// - The module websocekt is installed by this command
//    npm install -g websocket

// - If you want to use a formal protocol, you need to replace

//      var WebSocketServer = require(

//         'C:/Users/Henry/AppData/Roaming/npm/node_modules/websocket').server;

//   with

//       var WebSocketServer = require("ws").Server

// - Please refer Zhomart Mukhamejanov's example if you want to deploy server.js on Heroku

// - Please refer Vidit Mody's use of ws protocol

//var WebSocketServer =
require('C:/Users/Henry/AppData/Roaming/npm/node_modules/websocket').server;

var WebSocketServer = require('websocket').server;

var http = require('http');


var server = http.createServer(function(request, response) {

    console.log((new Date()) + ' Received request for ' + request.url);

    response.writeHead(404);

    response.end();

});

server.listen(8080, function() {

    console.log((new Date()) + ' Server is listening on port 8080');

});


// Create Websocket Serve

wsServer = new WebSocketServer({

    httpServer: server,

    // You should not use autoAcceptConnections for production

    // applications, as it defeats all standard cross-origin protection

    // facilities built into the protocol and the browser.  You should

    // *always* verify the connection's origin and decide whether or not

    // to accept it.

    autoAcceptConnections: false
```

```
});


function originIsAllowed(origin) {
  // Put logic here to detect whether the
  // specified origin (i.e., client) is allowed.
  return true;
}


wsServer.on('request', function(request) {
    if (!originIsAllowed(request.origin)) {
      // Make sure we only accept requests from an allowed origin
      request.reject();
      console.log((new Date()) + ' Connection from origin '
                  + request.origin + ' rejected.');
      return;
    }

    var connection = request.accept('echo-protocol', request.origin);
    console.log((new Date()) + ' Connection accepted.');

    ////////////////////////////////////////////////////
    // Case 1: rerceive message from the client
    ////////////////////////////////////////////////////
    connection.on('message', function(message) {
        if (message.type === "utf8") {
            console.log('Received Message: ' + message.utf8Data);
            var onoff = require('onoff');
            var Gpio = onoff.Gpio;
            // Initialize GPIO 23 to be an output pin.
            var led = new Gpio(4, 'out');
            var interval;

            interval=setInterval(function () {
            var blinkInterval = setInterval(blinkLED, 250);
            function blinkLED() {
                if(led.readSync() === 0) {
                    led.writeSync(1);
                } else {
                led.writeSync(0);
```

```
               }
               }
           function endBlink() {
              clearInterval(blinkInterval);
              led.writeSync(0);
                  }
           setTimeout(endBlink, 1000);
                         }, 4000);
            process.on('SIGINT', function () {
            clearInterval(interval);
            // writeSync(value) write 0 or 1 to GPIO
            led.writeSync(0);
            // Cleanly close the GPIO pin befire existing.
            led.unexport();
            console.log('Bye, bye!');
            process.exit();
            });
            }
            });

    ////////////////////////////////////////////////////

    // Case 2: close the connection

    ////////////////////////////////////////////////////

    connection.on('close', function(reasonCode, description) {

        console.log((new Date()) + ' Peer '

                 + connection.remoteAddress + ' disconnected.');

    });
     });
```

---

client.html

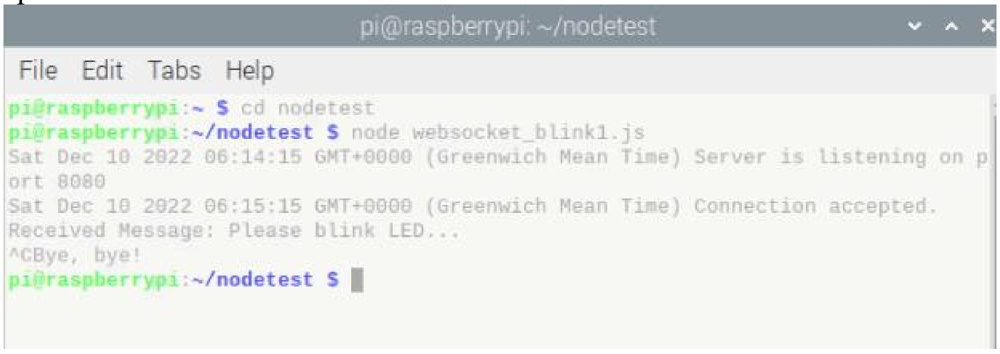
```
<!DOCTYPE HTML>

<html>

<head>

<script type="text/javascript">



function WebSocketTest()

{

  if ("WebSocket" in window)

  {

     alert("WebSocket is supported by your Browser!");

     // Let us open a web socket
     // - Error if use this line
     //     var ws = new WebSocket("ws://localhost:8080");
     // - Use this line if the browser would like to communicate with
     //   the server where client.html is downloaded.
     //     var ws = new WebSocket("ws://" + location.host, 'echo-protocol');
     //   Refer Zhomart Mukhamejanov's example
     // - Websocket allows connection from any source, but first
     //   connection should be http request, they call it
     //   "Websocket handshake". For example, you can access
     //      http://npu-socket.herokuapp.com/
     //   then it is possible to write like this
     //       var ws = new WebSocket("wss://npu-socket.herokuapp.com", 'echo-protocol');
     var ws = new WebSocket("ws://localhost:8080", 'echo-protocol');


     // - The readonly attribute readyState represents the state
     //   of the connection. It can have the following values:
     //   + A value of 0 indicates that the connection has
     //     not yet been established.
     //   + A value of 1 indicates that the connection is
     //     established and communication is possible.
     //   + A value of 2 indicates that the connection is going
     //     through the closing handshake.
     //   + A value of 3 indicates that the connection has been
     //     closed or could not be opened.
     // - The open event occurs when socket connection is established.
     ws.onopen = function()
     {
        // Web Socket is connected, send data using send()
        ws.send("Please blink LED...");
        alert("Message is sent...");
```

```
      };

      // The message event occurs when client receives data from server.
      ws.onmessage = function (evt)
      {
        var received_msg = evt.data;
        alert("Message is received...");
      };

      // The close event occurs when connection is closed.
      ws.onclose = function()
      {
        // websocket is closed.
        alert("Connection is closed...");
      };

      // The error event occurs when connection is closed.
      ws.onerror = function()
      {
        // There is erro
        alert("WebSocket error...");
      };
    }
    else
    {
      // The browser doesn't support WebSocket
      alert("WebSocket NOT supported by your Browser!");
    }
  }
</script>
```

```
</head>

<body>

<div id="sse">

   <a href="javascript:WebSocketTest()">Run WebSocket</a>

</div>

</body>

</html>
```

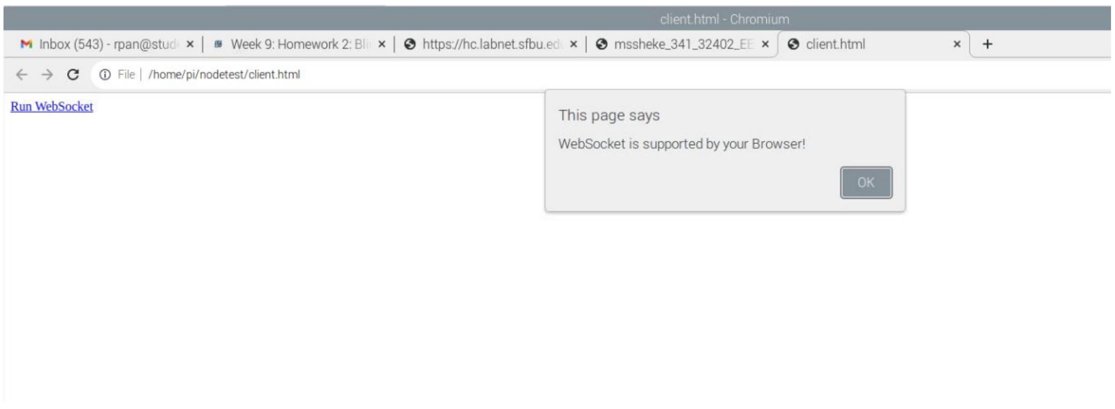
Step 4: Run the code

$ node websocket_blink1.js
Open the file client.html

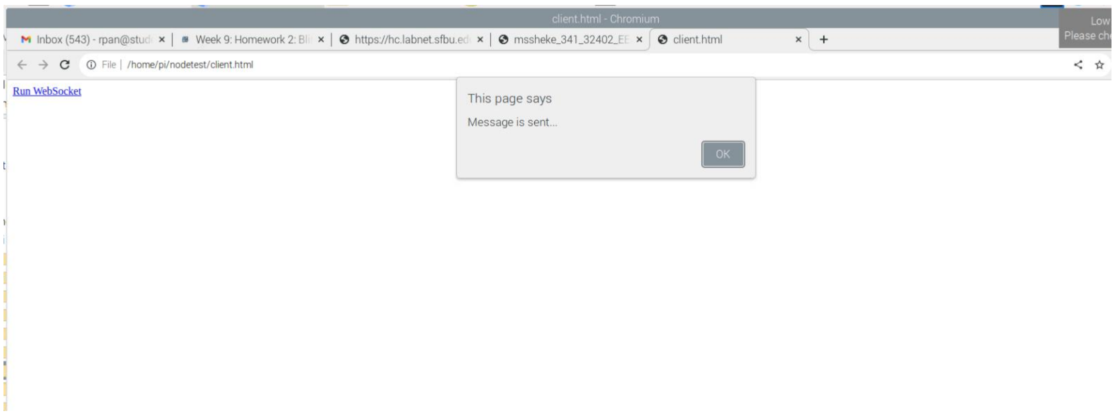


```
pi@raspberrypi:~ $ cd nodetest
pi@raspberrypi:~/nodetest $ node websocket_blink1.js
Sat Dec 10 2022 06:14:15 GMT+0000 (Greenwich Mean Time) Server is listening on p
ort 8080
Sat Dec 10 2022 06:15:15 GMT+0000 (Greenwich Mean Time) Connection accepted.
Received Message: Please blink LED...
^CBye, bye!
pi@raspberrypi:~/nodetest $
```

Result:
https://youtube.com/shorts/8Aa-uHX-Dw0?feature=share



This page says
WebSocket is supported by your Browser!


client.html - Chromium

Inbox (543) - rpan@stud | Week 9: Homework 2: Bli | https://hc.labnet.sfbu.ed | mssheke_341_32402_EE | client.html

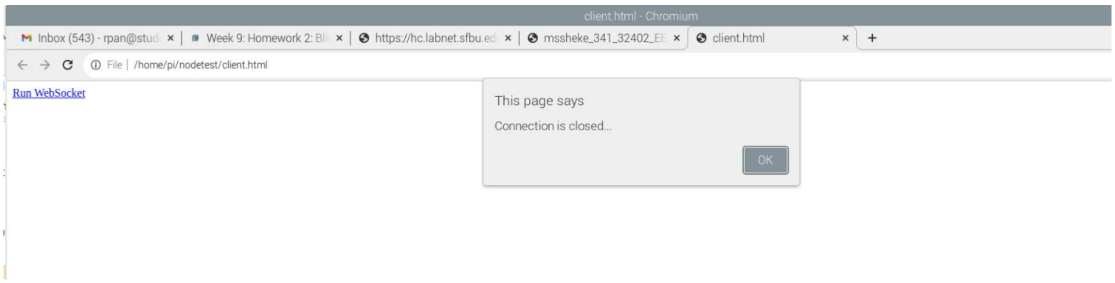
File | /home/pi/nodetest/client.html

Run WebSocket

This page says

Message is sent...

OK


client.html - Chromium

Inbox (543) - rpan@stud | Week 9: Homework 2: Bli | https://hc.labnet.sfbu.ed | mssheke_341_32402_EE | client.html

File | /home/pi/nodetest/client.html

Run WebSocket

This page says

Connection is closed...

OK