Week 6: Homework 3: Pushbutton LED Websocket server

https://hc.labnet.sfbu.edu/~henry/npu/classes/embed/raspberry_pi/slide/exercise_raspberry_pi.htmlLinks to an external site.

Q20 ==> Pushbutton LED Websocket server

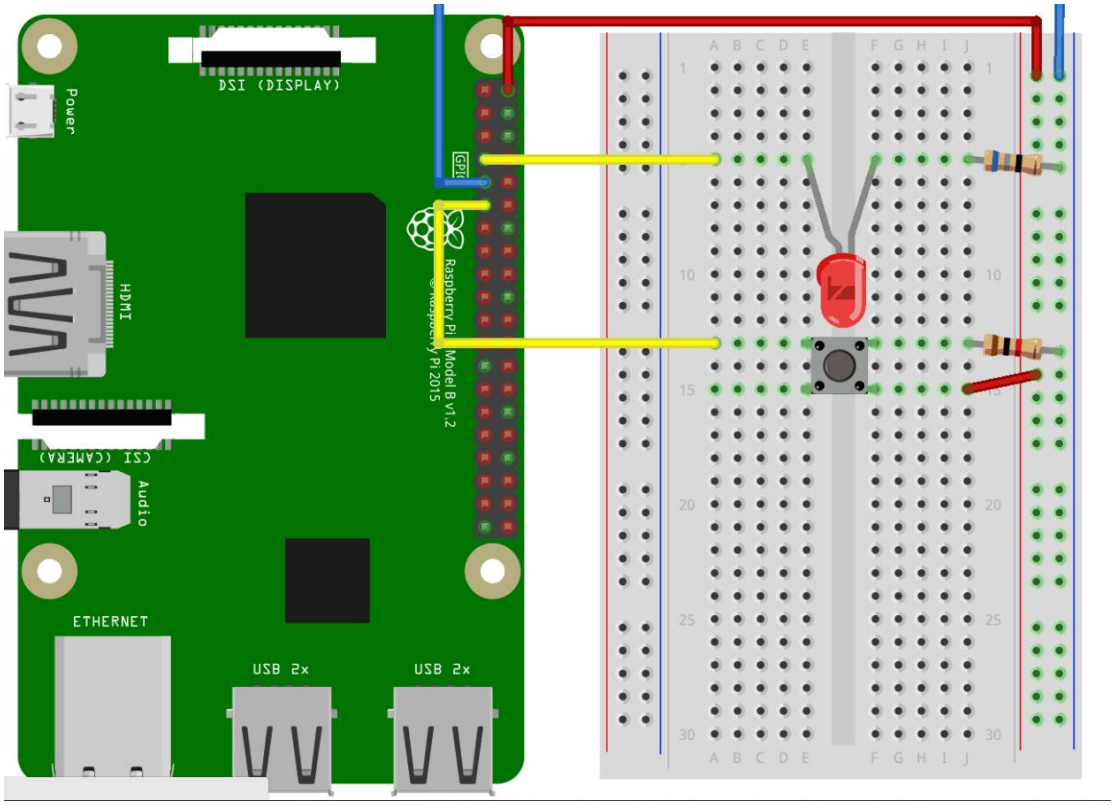
1. Pushbutton LED Websocket server
    - Please implement a Pushbutton LED websocket server on Raspberry Pi.
    - Reference
        - Quiz
        - RasPi LED & Pushbutton
        - Implement WebSockets server using Node.js
        - Gangbaolede Li - 2019 Spring
        - Mai La - 2019 Spring

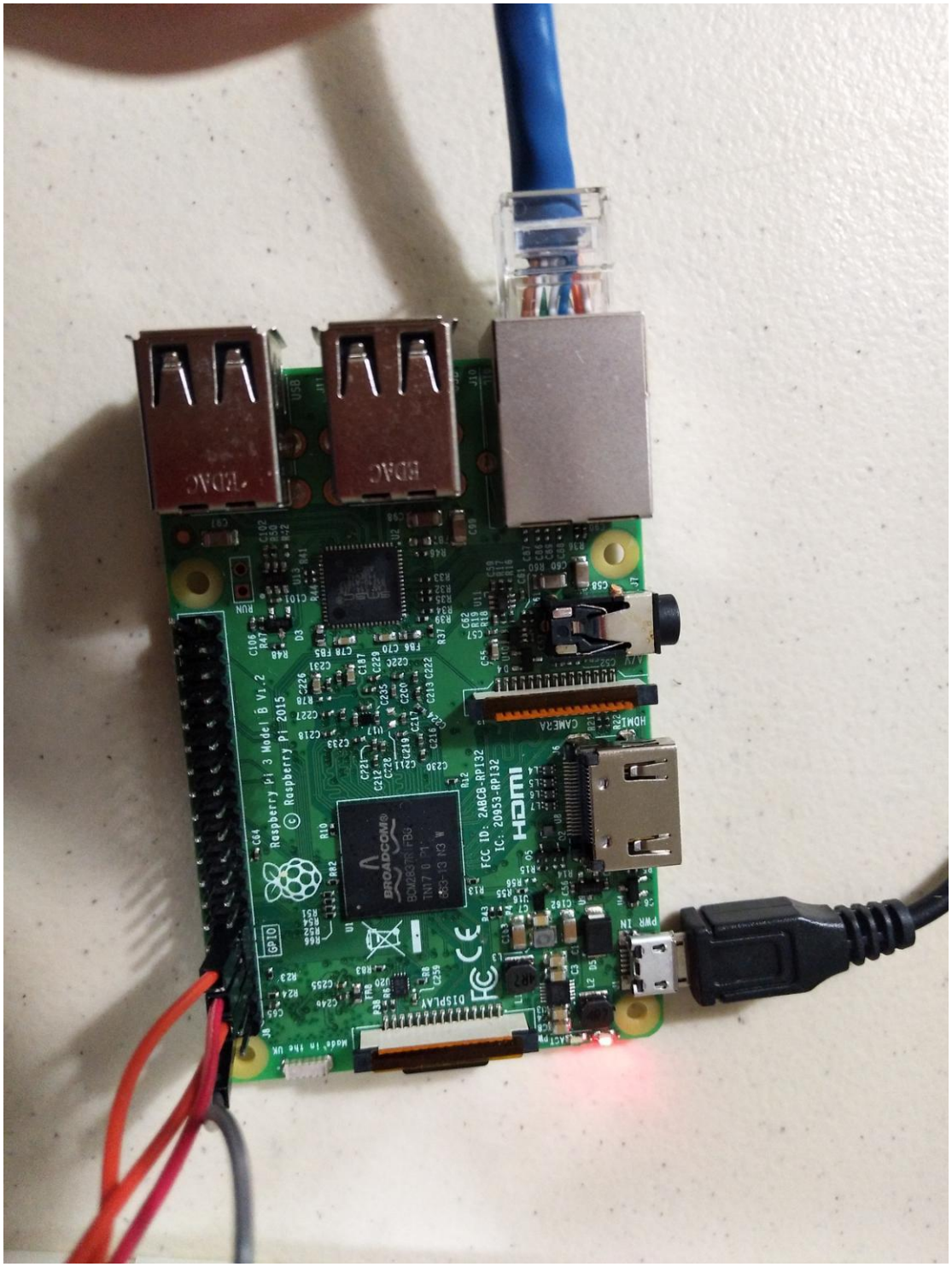
Step 1: Install Package

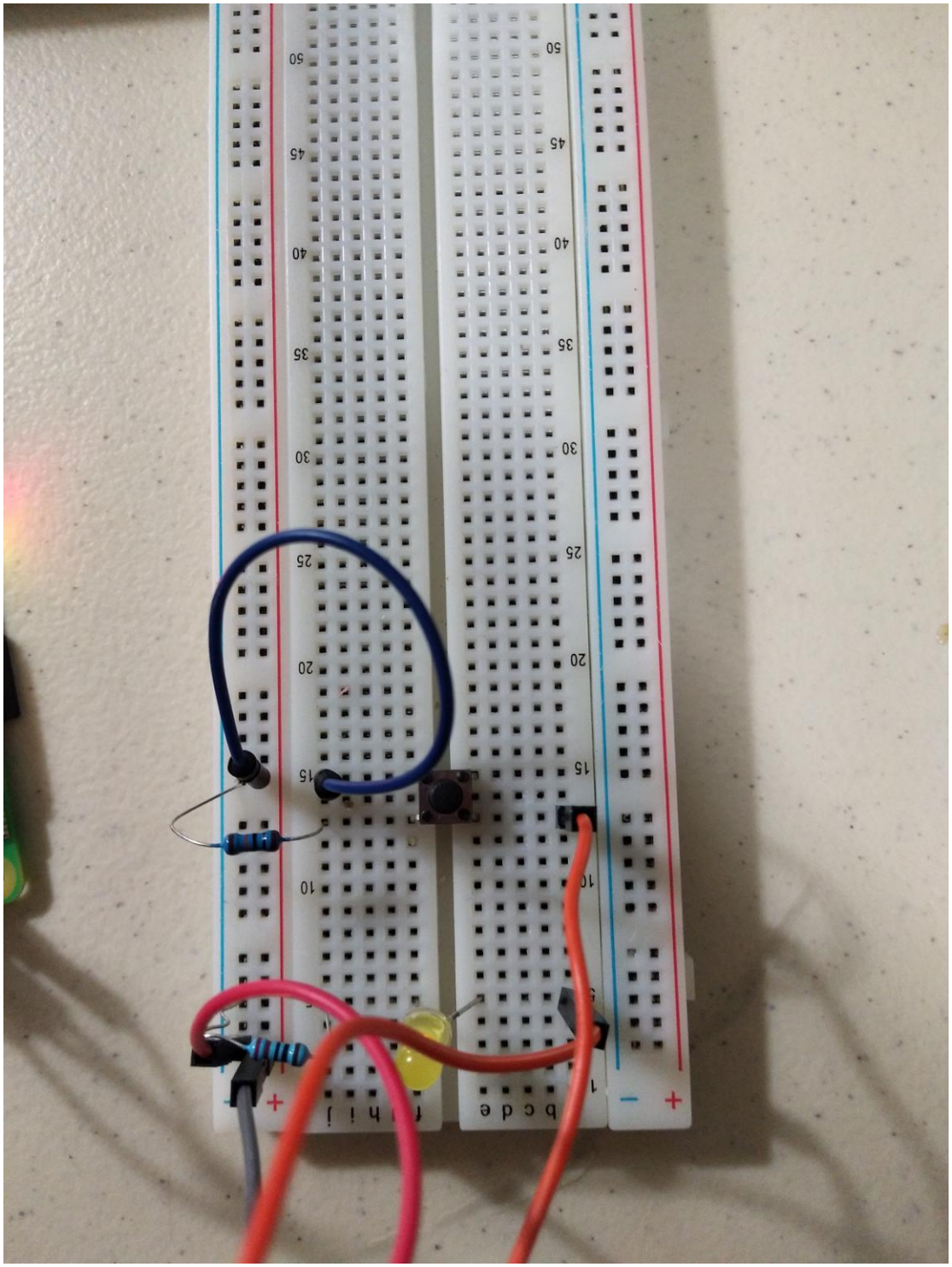
Install websocket
$ npm install websocket
$ npm install onoff
$ pigpiod -v
$ sudo apt-get update
$ sudo apt-get update
$ sudo apt-get install pigpio
$ npm install pigpio

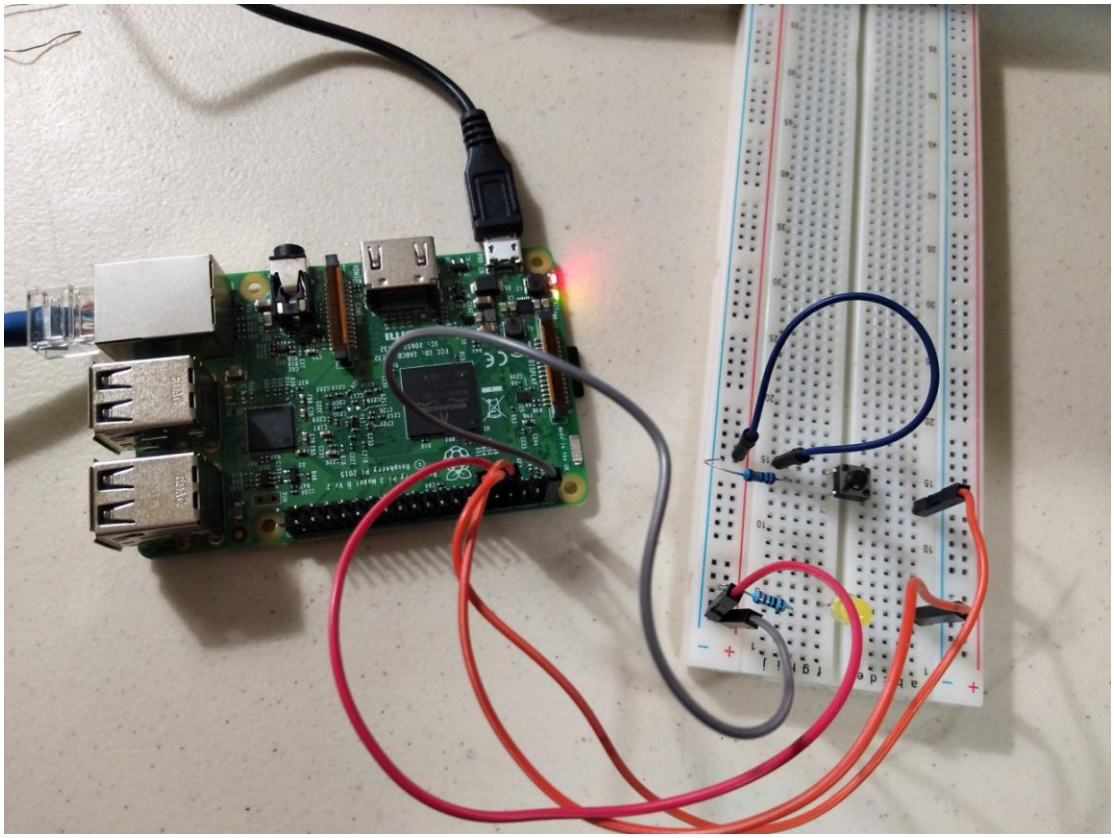
Step 2: Connecting Raspberry Pi and LED light, button

Reference:
Node.js Raspberry Pi LED and Pushbutton (w3schools.com)


DSI (DISPLAY)
Power
HDMI
CSI (CAMERA)
Audio
ETHERNET
USB 2x
USB 2x
GPIO
Raspberry Pi Model B v1.2
A B C D E F G H I J
1
10
15
20
25
30


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





Step 3: Prepare Codes:

websocket_ledbutton.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.');
    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);
          connection.sendUTF(message.utf8Data);
      }
      else if (message.type === 'binary') {
          console.log('Received Binary Message of '
                      + message.binaryData.length + ' bytes');
          connection.sendBytes(message.binaryData);
      }
  ////////////////////////////////////////////////////
      var Gpio = require('onoff').Gpio; //include onoff to interact with the GPIO
      var LED = new Gpio(4, 'out'); //use GPIO pin 4 as output
      var pushButton = new Gpio(17, 'in', 'both'); //use GPIO pin 17 as input, and 'both' button presses, and releases should be handled
```

```
    ////////////////////////////////////////////////////

        var Gpio = require('onoff').Gpio; //include onoff to interact with the GPIO
        var LED = new Gpio(4, 'out'); //use GPIO pin 4 as output
        var pushButton = new Gpio(17, 'in', 'both'); //use GPIO pin 17 as input, and 'both' button presses, and releases should be handled

        pushButton.watch(function (err, value) { //Watch for hardware interrupts on pushButton GPIO, specify callback function
            if (err) { //if an erro,
                console.error('There was an error', err); //output error message to console
            return;
            }
            LED.writeSync(value); //turn LED on or off depending on the button state (0 or 1)
            }
        )

        function unexportOnClose() { //function to run when exiting program
            LED.writeSync(0); // Turn LED off
            LED.unexport(); // Unexport LED GPIO to free resources
            pushButton.unexport(); // Unexport Button GPIO to free resources
        }

        process.on('SIGINT', unexportOnClose); //function to run when user closes using ctrl+c

});

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

    // Case 2: close the connection

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

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

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

                    + connection.remoteAddress + ' disconnected.');

    });
       });
```

client.html

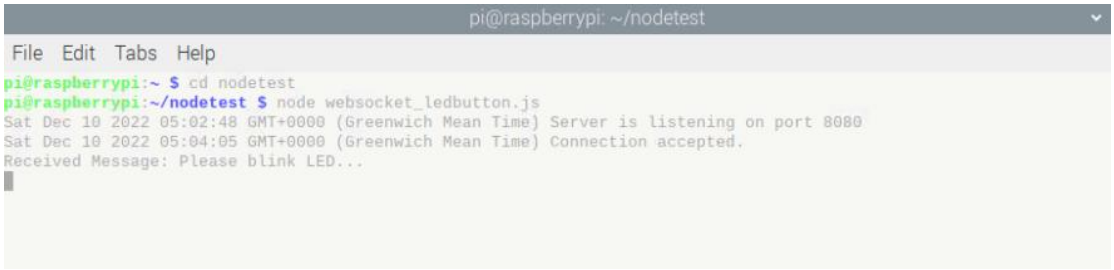
```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">


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');
```

```
//   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...");
};
```

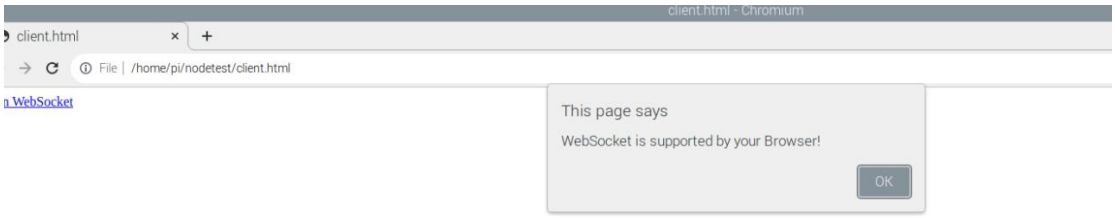
```
    // The close event occurs when connection is closed.
    ws.onclose = function()
    {
       // websocket is closed.
       alert("Connection is closed...");
    };

    // 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>
```

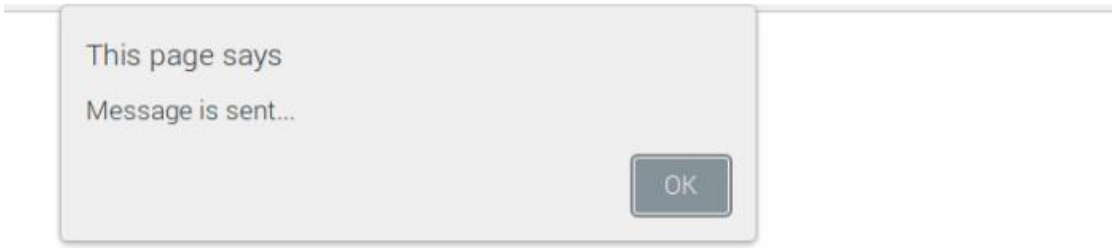
```
    // The error event occurs when connection is closed.
    ws.onerror = function()
    {
       // There is erro
       alert("WebSocket error...");
    };
  }
  else
  {
    // The browser doesn't support WebSocket
    alert("WebSocket NOT supported by your Browser!");
  }
}
</script>
</head>
<body>
<div id="sse">
    <a href="javascript:WebSocketTest()">Run WebSocket</a>
</div>
</body>
</html>
```

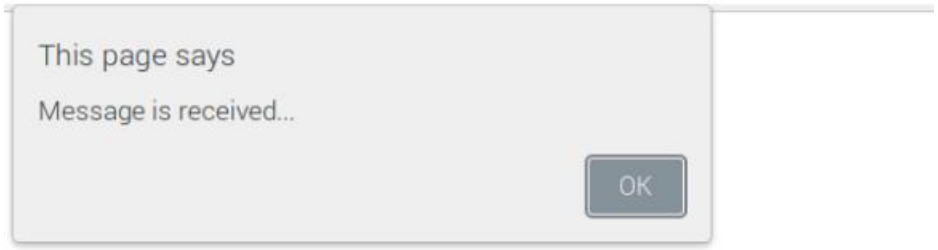
Run codes:
$ node websocket_ledbutton.js



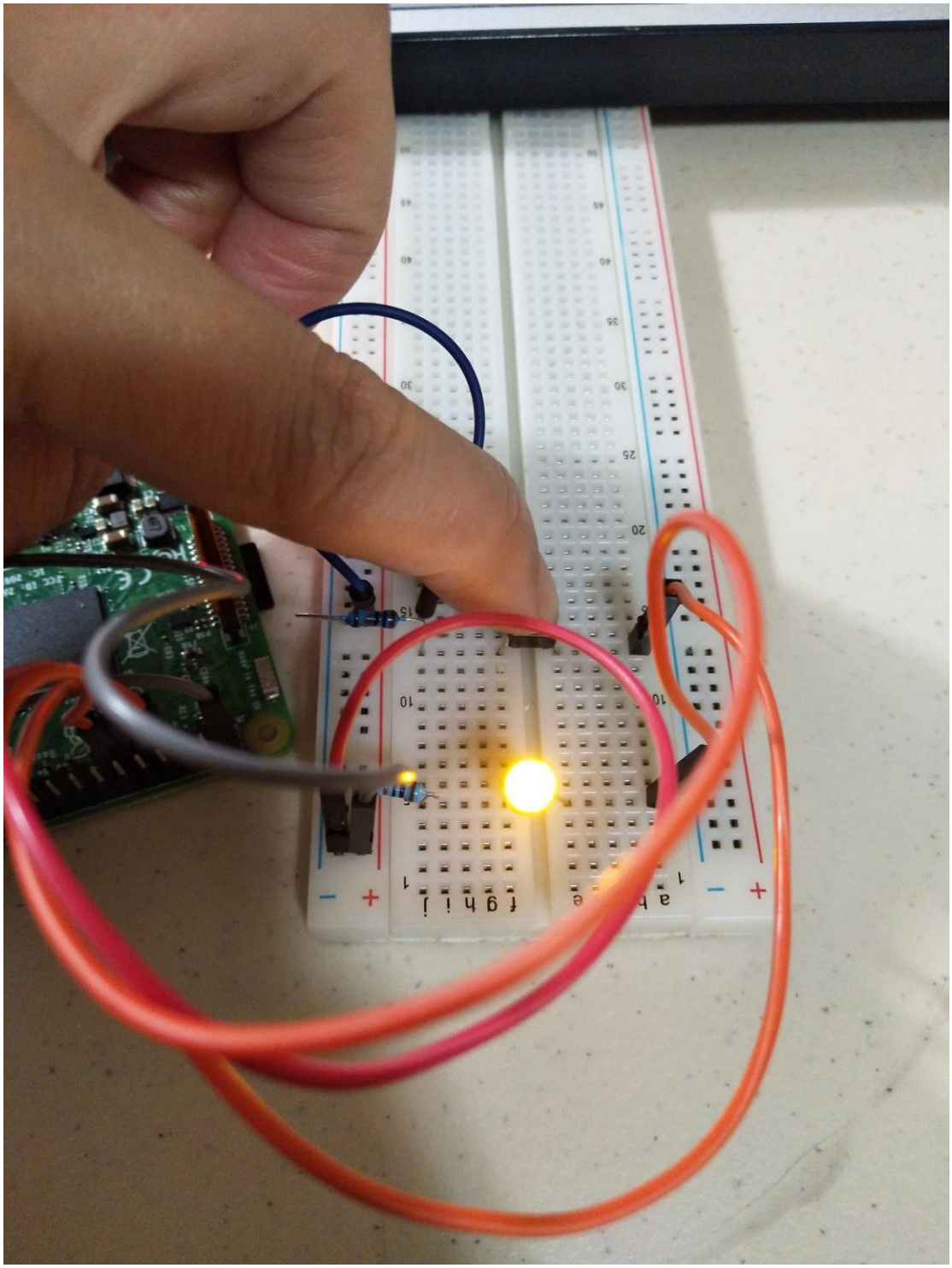
Open client.html, then press ok







Press the button:



End the program: Ctrl + C