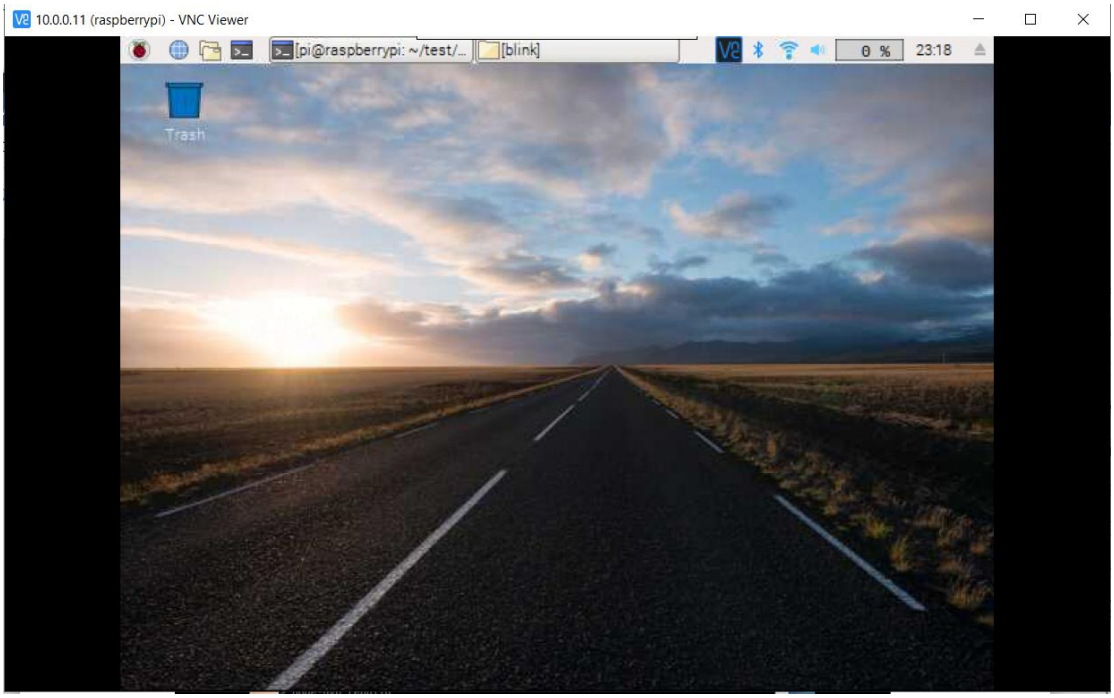
EE517 Internet of Things

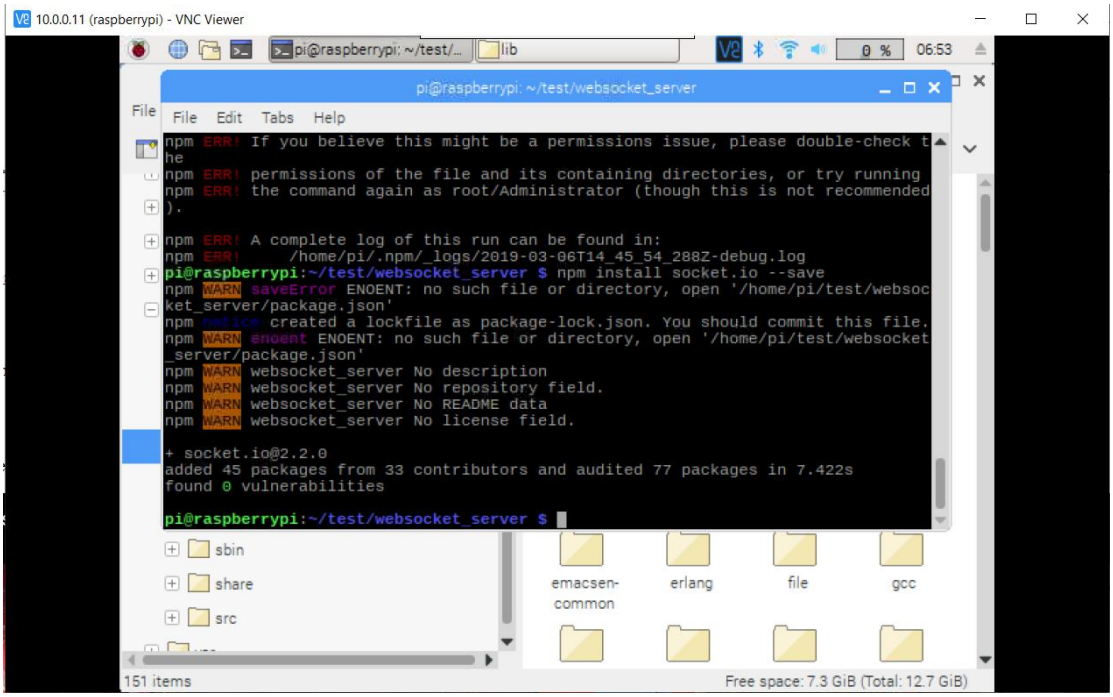
Project: Implement a Pushbutton LED websocket server on Raspberry Pi

1. Access Raspberry Pi with your computer by using Putty or VNC. Here, I use VNC.



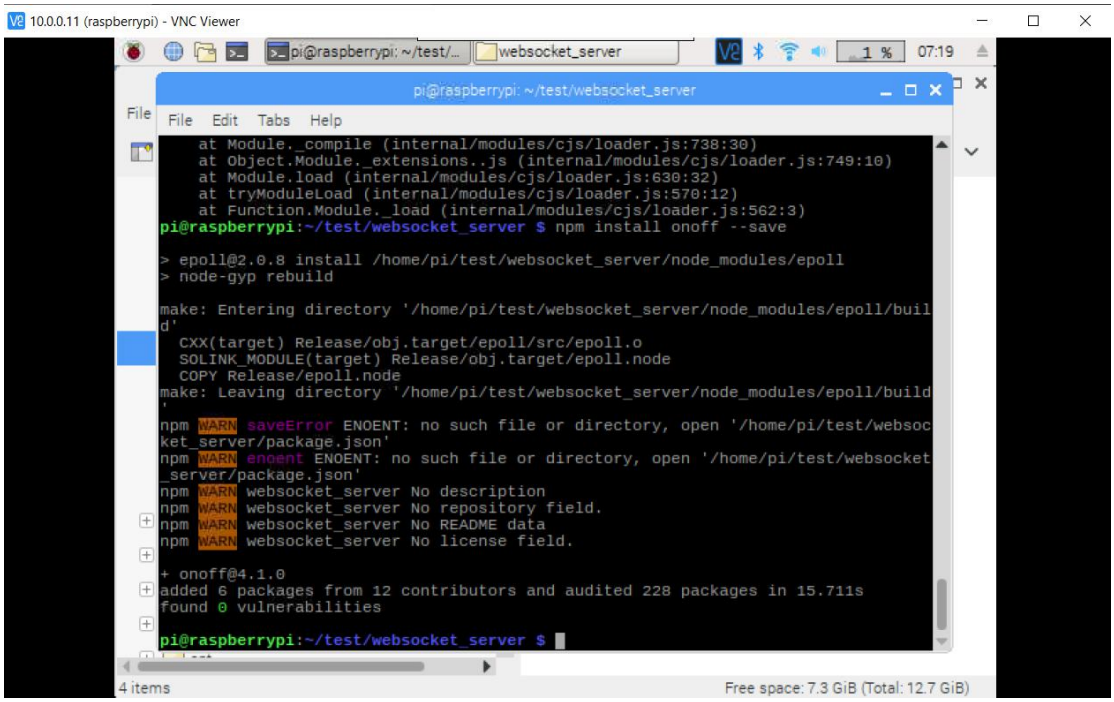
2. Install Websoket module under your direction

```
$ npm install socket.io --save
```

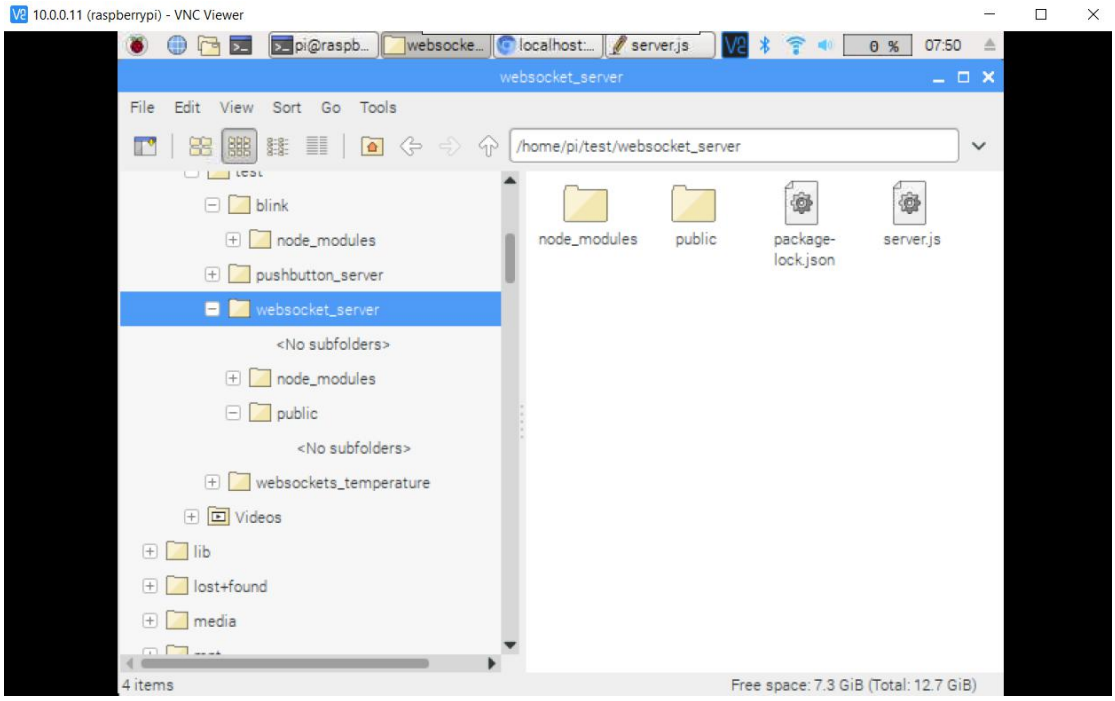


3. Install onoff packet under your direction

```
$ npm install onoff --save
```



4. Create a server.js file under your folder



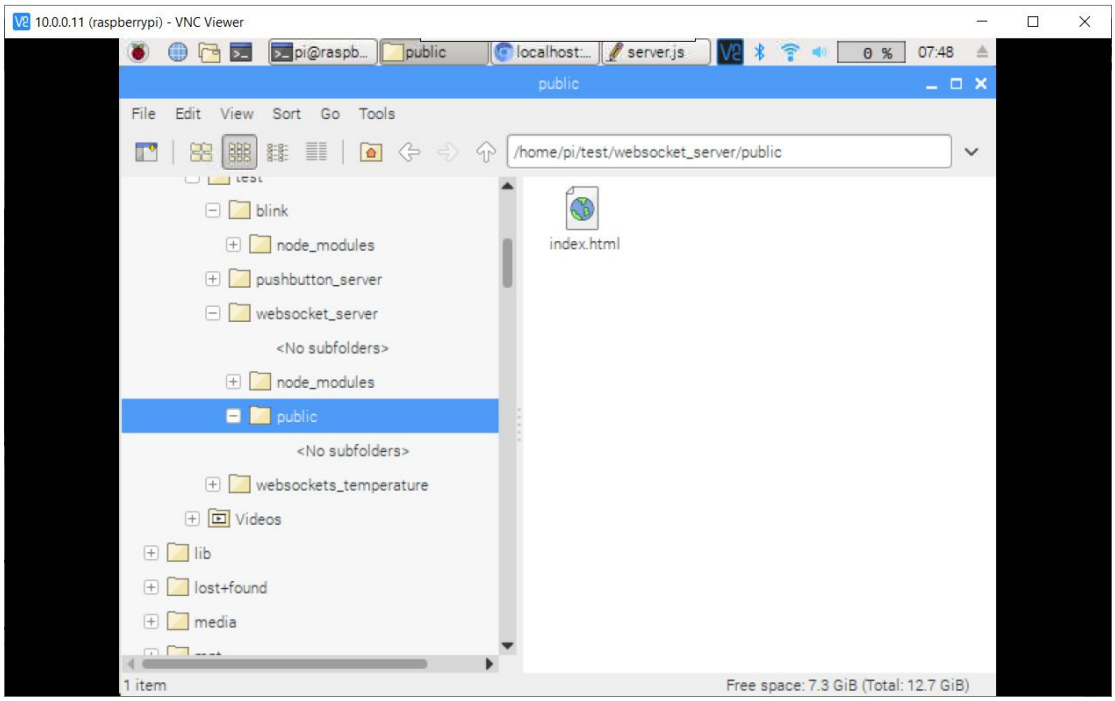
server.js:

```
//require http server, and create server with function handler()
var http = require('http').createServer(handler);
var fs = require('fs'); //require filesystem module
//require socket.io module and pass the http object (server)
```

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

http.listen(8080); //listen to port 8080

function handler (req, res) { //create server
   //read file index.html in public folder
   fs.readFile(__dirname + '/public/index.html', function(err, data) {
      if (err) {
         //display 404 on error
         res.writeHead(404, {'Content-Type': 'text/html'});
         return res.end("404 Not Found");
      }
      res.writeHead(200, {'Content-Type': 'text/html'}); //write HTML
      res.write(data); //write data from index.html
      return res.end();
   });
}

// WebSocket Connection
io.sockets.on('connection', function (socket) {
   // WebSocket Connection
   var lightvalue = 0; //static variable for current status
   //Watch for hardware interrupts on pushButton
   pushButton.watch(function (err, value) {
      if (err) { //if an error
         //output error message to console
         console.error('There was an error', err);
         return;
      }
      lightvalue = value;
      //send breadboard button status to client (i.e., the browser)
      socket.emit('light', lightvalue);
   });
   // get light switch status from client (i.e., the browser)
   socket.on('light', function(data) {
      lightvalue = data;
      //only change LED if status has changed
```

```
        if (lightvalue != LED.readSync()) {
            LED.writeSync(lightvalue); //turn LED on or off
        }
    });
});

process.on('SIGINT', function () { //on ctrl+c
    LED.writeSync(0); // Turn LED off
    LED.unexport(); // Unexport LED GPIO to free resources
    pushButton.unexport(); // Unexport Button GPIO to free resources
    process.exit(); //exit completely
});
```

5. Create a /public/index.html file under your folder



index.html:

```
<!DOCTYPE html>
<html>
<body>
<h1>Control LED light</h1>
<p><input type="checkbox" id="light"></p>
<!-- include socket.io client side script -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.0.3/socket.io.js"></script>
<script>
//load socket.io-client and connect to the host that serves the page
var socket = io();
window.addEventListener("load", function(){ //when page loads
```

```
    var lightbox = document.getElementById("light");
    //add event listener for when the browser checkbox changes
    lightbox.addEventListener("change", function() {
        //send the browser button status to server (as 1 or 0)
        socket.emit("light", Number(this.checked));
    });
});
socket.on('light', function (data) { //get button status from the breadboard
    //change the browser checkbox according to push button on Raspberry Pi
    document.getElementById("light").checked = data;

    //send the browser push button status to back to server
    socket.emit("light", data);
});
</script>
</html>
</body>
</html>
```

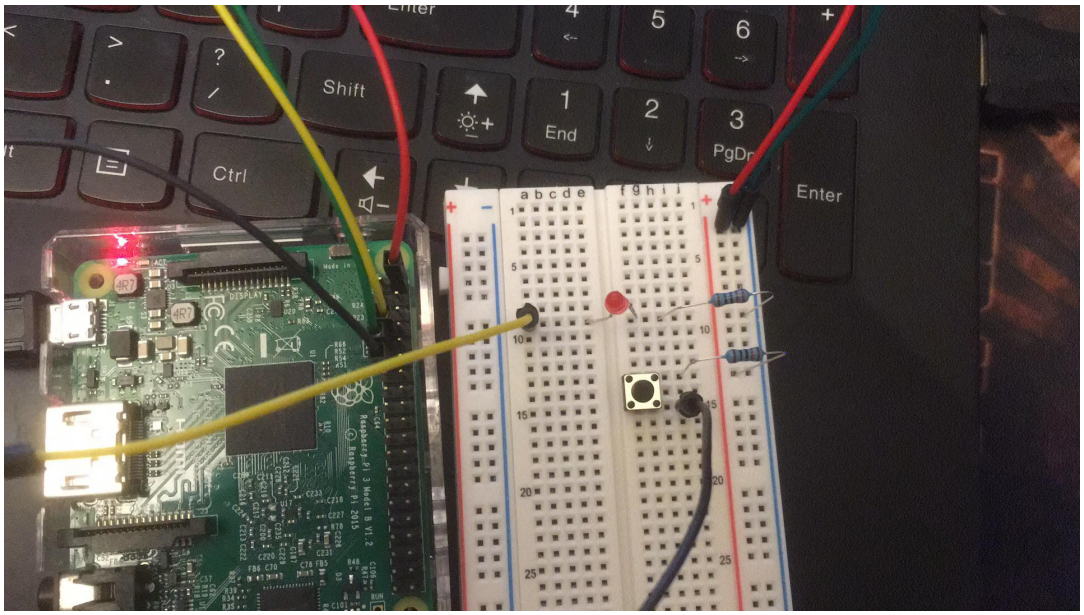
6. Hardware connection

Pin 2 (5V, which is the red wire) connects to power bus (the red line on right side of the breadboard).

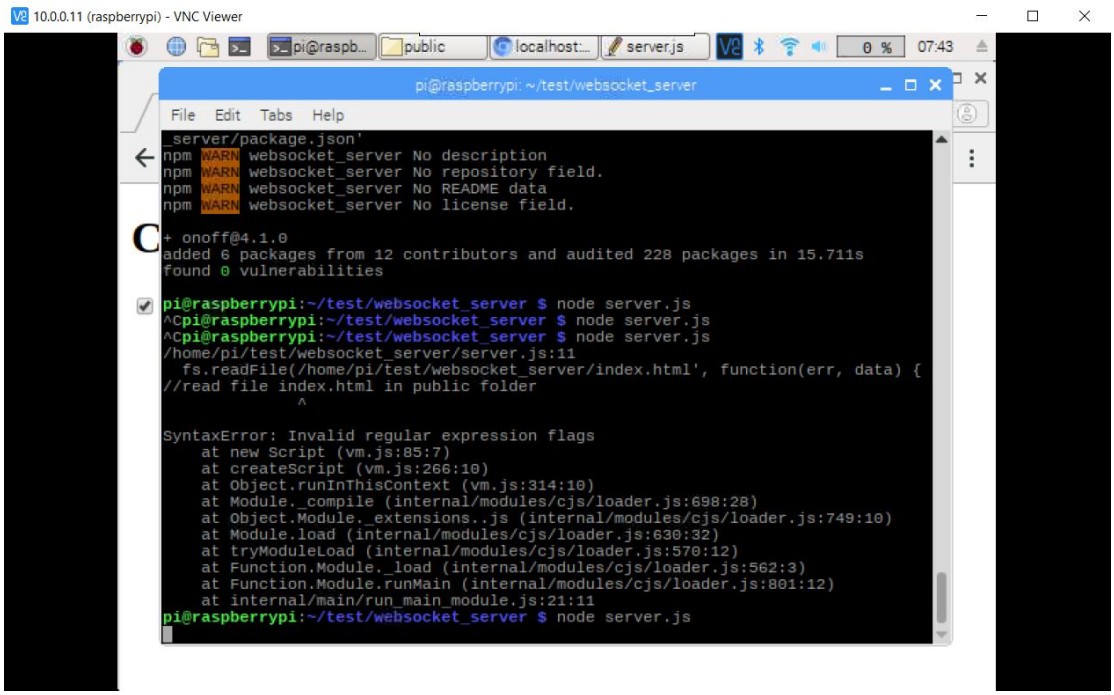
Pin 7 (GPIO4, which is the yellow wire) connects with the anode of LED (which is the long leg), and the cathode of LED (which is the short leg) connects with a 220Ω resistor, then connects to ground bus (which is blue line on right side of the breadboard).

Pin 9 (GND, which is the green wire) connects to ground bus of breadboard.

Pin 11 (GPIO17, which is the black wire) connects with the button and a 220Ω resistor, then connects to ground bus.



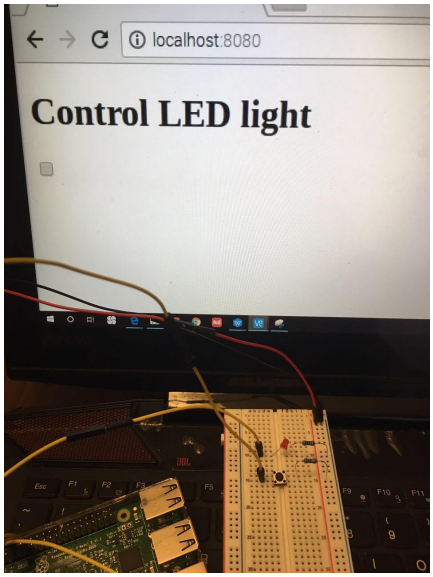
7. On your terminal, run: $ node server.js



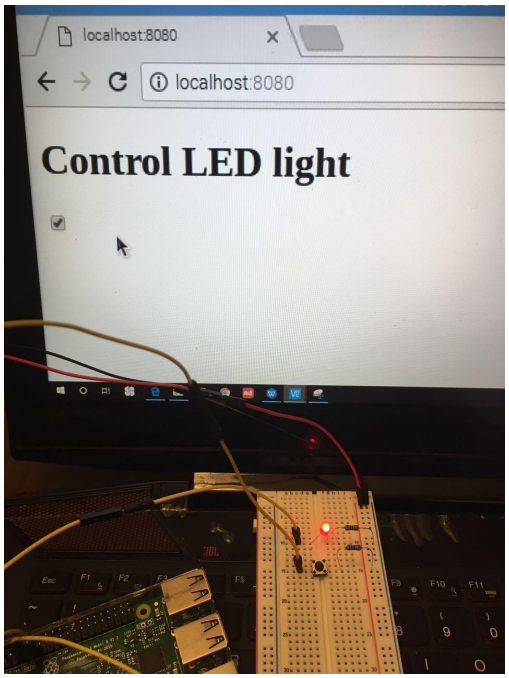
Demo:

on the Raspberry Pi, open a browser and run:

http://localhost:8080/



If the button is pressed, Led is on, and there is a check mark in the box



If the check mark in the box is selected in the browser, LED is on without pressing button