

## Aplicación Web Angular

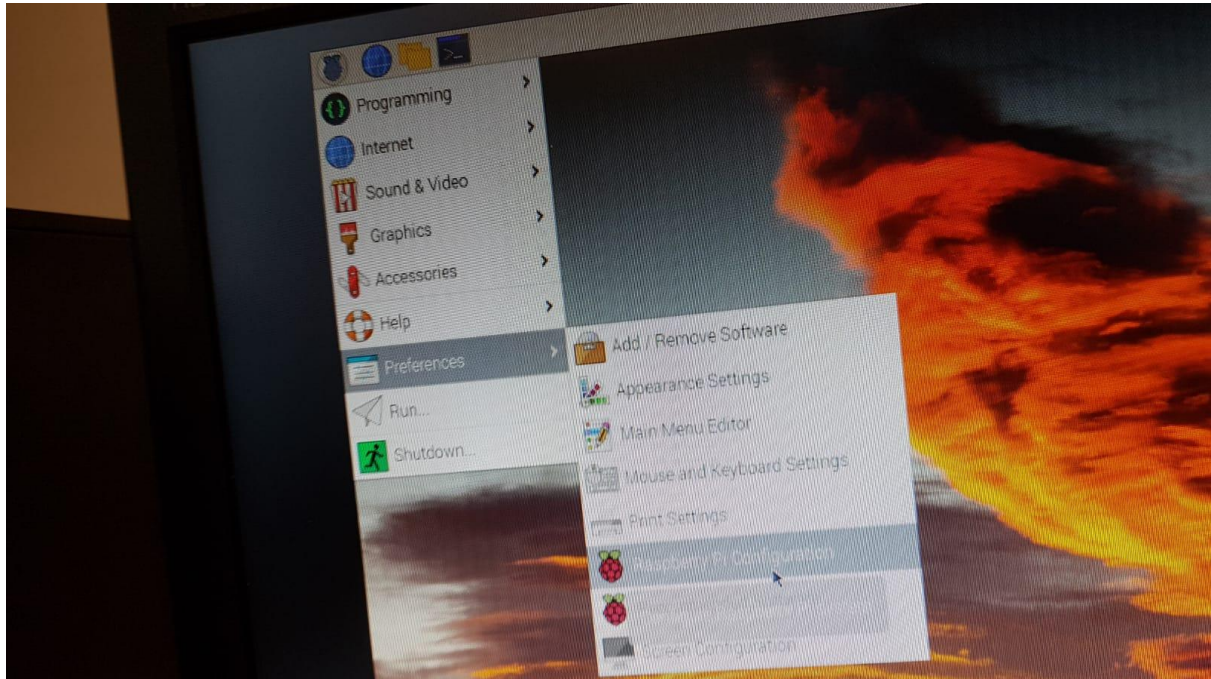
La idea es desarrollar una aplicación para un marcador electrónico con una Raspberry Pi. Tendrá un par de pulsadores físicos para los puntos de cada equipo y una luz que enciende para cada uno de ellos, por el lado del cliente se realizará una interfaz web en el servidor de Raspberry Pi para ir indicando la puntuación de cada equipo, un par de botones para puntos y otro para resetear.



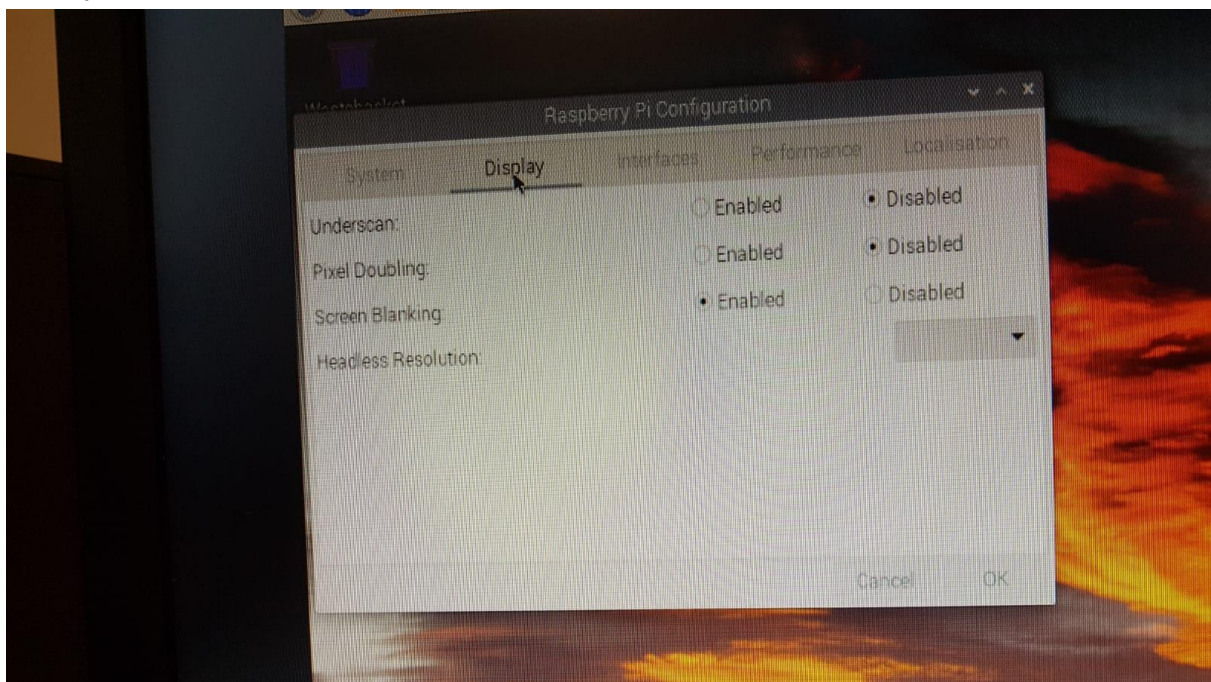
Conexión de la Raspberry Pi:



Una vez arrancada la Raspberry Pi hay que habilitar la conexión por SSH en menú => preferencias:

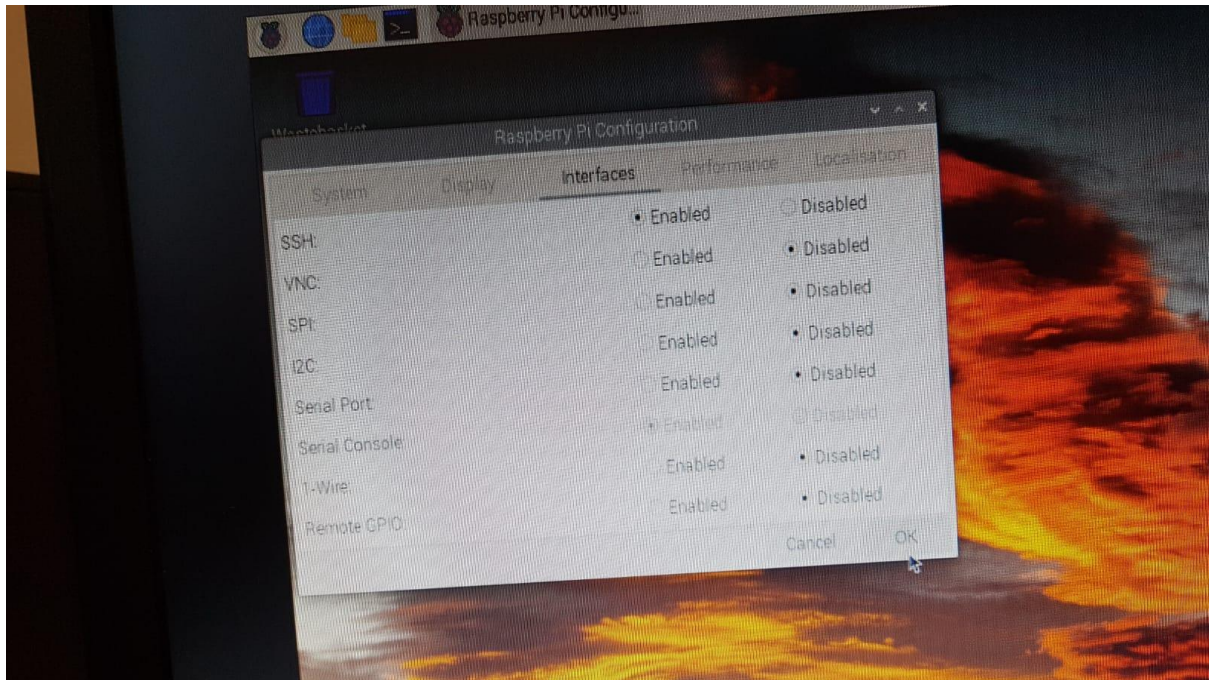


Display:

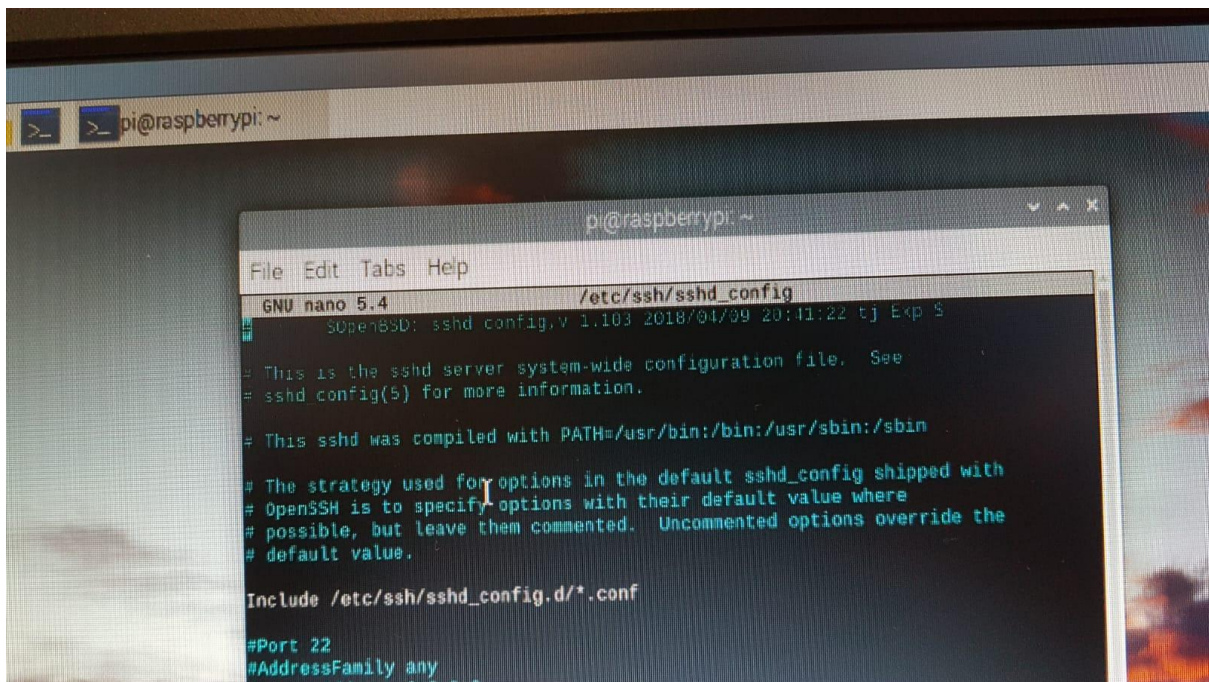




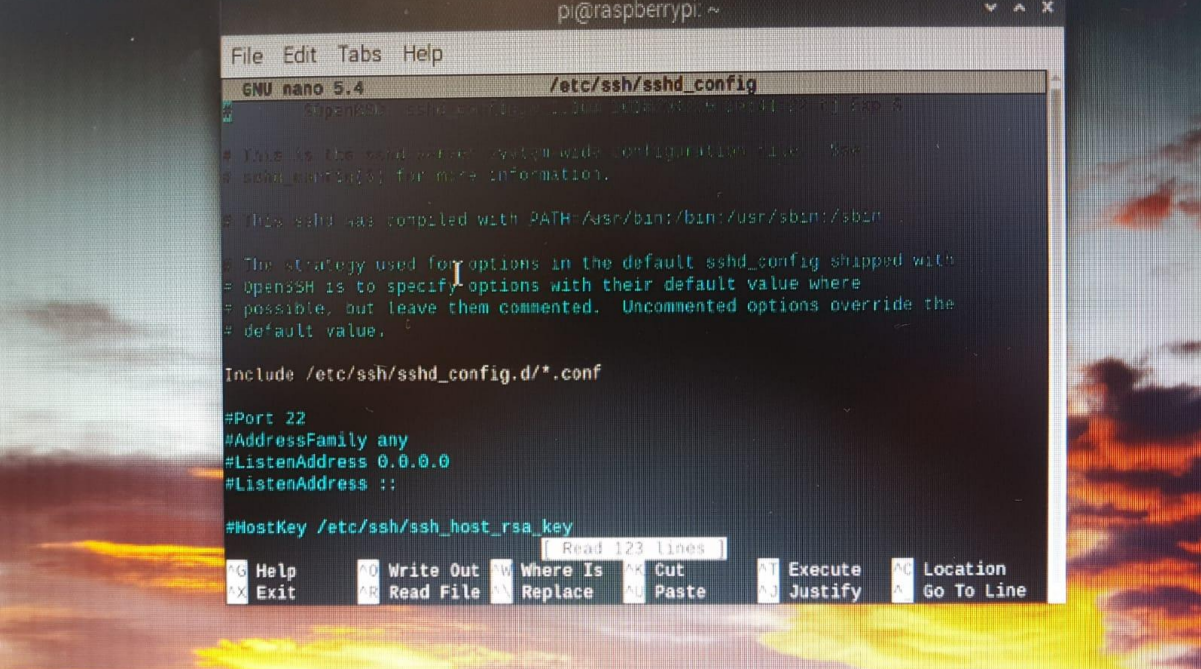
Habilitar SSH:



Después hay que modificar el archivo de configuración en /etc/ssh/ssh\_config:

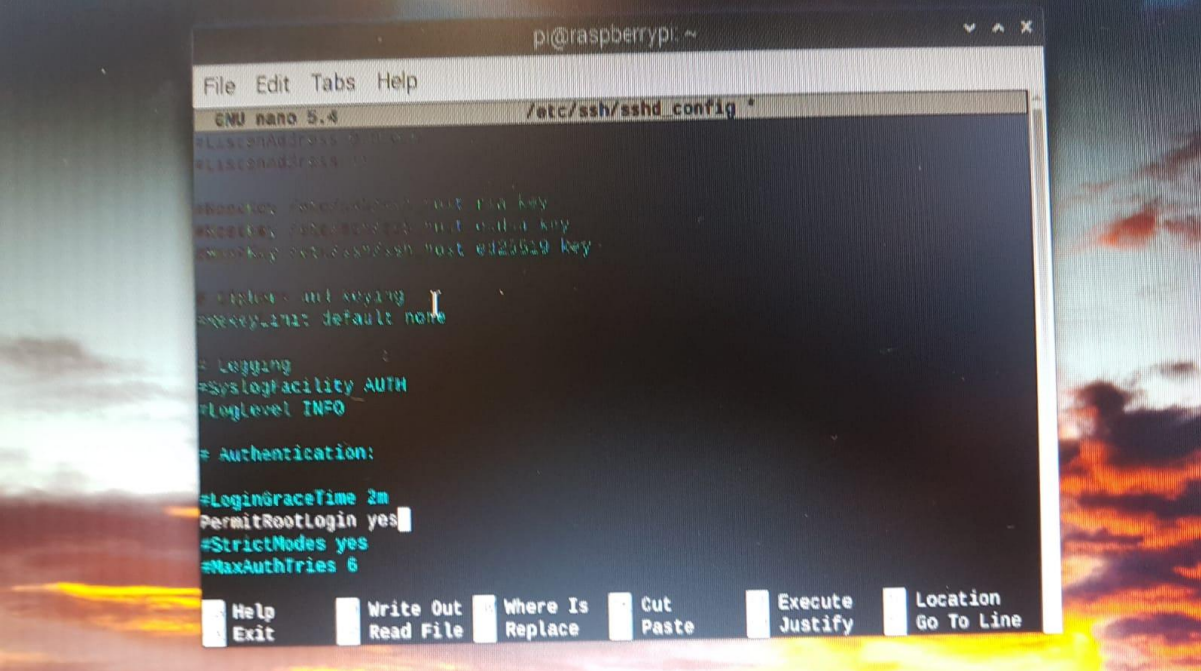


Desplazarse hasta la línea PermitRootLogin:



```
pi@raspberrypi: ~  
File Edit Tabs Help  
GNU nano 5.4 /etc/ssh/sshd_config  
# This is the system-wide configuration file. See  
# sshd(8) for more information.  
# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin  
# The strategy used for options in the default sshd_config shipped with  
# OpenSSH is to specify options with their default value where  
# possible, but leave them commented. Uncommented options override the  
# default value.  
  
Include /etc/ssh/sshd_config.d/*.conf  
  
#Port 22  
#AddressFamily any  
#ListenAddress 0.0.0.0  
#ListenAddress ::  
  
#HostKey /etc/ssh/ssh_host_rsa_key  
#HostKey /etc/ssh/ssh_host_ecdsa_key  
#HostKey /etc/ssh/ssh_host_ed25519_key  
  
# Authentication and logging  
#Subsystem sftp sftp-server  
  
# Logging  
#SyslogFacility AUTH  
#LogLevel INFO  
  
# Authentication:  
#LoginGraceTime 2m  
PermitRootLogin yes  
#StrictModes yes  
#MaxAuthTries 6
```

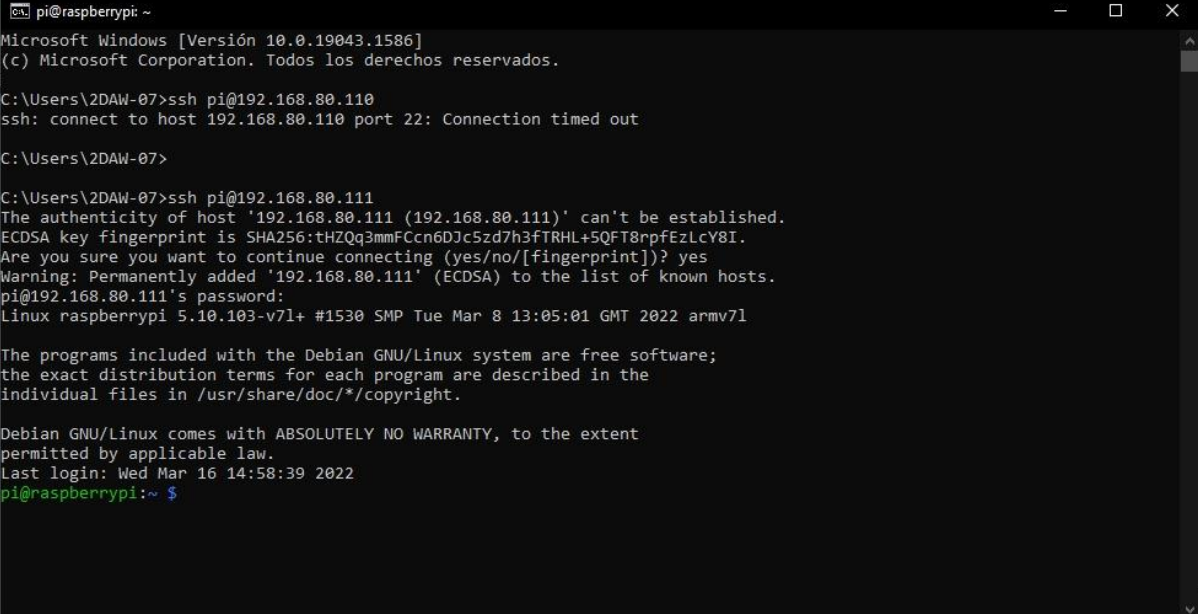
Descomentar la línea PermitRootLogin y cambiar a yes:



```
pi@raspberrypi: ~  
File Edit Tabs Help  
GNU nano 5.4 /etc/ssh/sshd_config  
# This is the system-wide configuration file. See  
# sshd(8) for more information.  
# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin  
# The strategy used for options in the default sshd_config shipped with  
# OpenSSH is to specify options with their default value where  
# possible, but leave them commented. Uncommented options override the  
# default value.  
  
Include /etc/ssh/sshd_config.d/*.conf  
  
#Port 22  
#AddressFamily any  
#ListenAddress 0.0.0.0  
#ListenAddress ::  
  
#HostKey /etc/ssh/ssh_host_rsa_key  
#HostKey /etc/ssh/ssh_host_ecdsa_key  
#HostKey /etc/ssh/ssh_host_ed25519_key  
  
# Authentication and logging  
#Subsystem sftp sftp-server  
  
# Logging  
#SyslogFacility AUTH  
#LogLevel INFO  
  
# Authentication:  
#LoginGraceTime 2m  
PermitRootLogin yes  
#StrictModes yes  
#MaxAuthTries 6
```



Una vez realizado estos pasos, desde el cliente nos podemos conectar a la Raspberry Pi por SSH y trabajar desde él.



```
pi@raspberrypi ~
Microsoft Windows [Versión 10.0.19043.1586]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\2DAW-07>ssh pi@192.168.80.110
ssh: connect to host 192.168.80.110 port 22: Connection timed out

C:\Users\2DAW-07>

C:\Users\2DAW-07>ssh pi@192.168.80.111
The authenticity of host '192.168.80.111 (192.168.80.111)' can't be established.
ECDSA key fingerprint is SHA256:tHZQq3mmFCcn6DJc5zd7h3fTRHL+5QFT8rpfEzLcY8I.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.80.111' (ECDSA) to the list of known hosts.
pi@192.168.80.111's password:
Linux raspberrypi 5.10.103-v7l+ #1530 SMP Tue Mar 8 13:05:01 GMT 2022 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 16 14:58:39 2022
pi@raspberrypi:~ $
```

Descargamos node en la Raspberry Pi (desde ahora servidor):

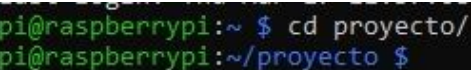
```
sudo apt install nodejs
```

Comprobamos las versiones

```
node --version
```

```
npm --version
```

Creamos directorio del proyecto:



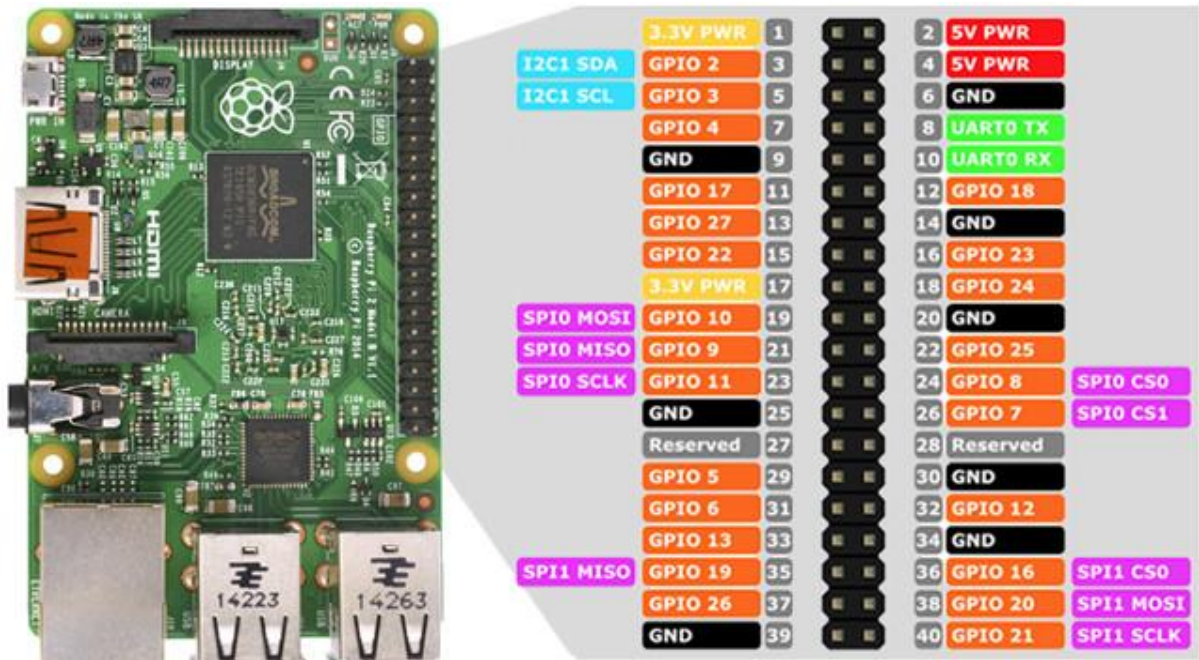
```
pi@raspberrypi:~ $ cd proyecto/
pi@raspberrypi:~/proyecto $
```

Creamos directorio public:



```
pi@raspberrypi:~/proyecto $ cd public/
pi@raspberrypi:~/proyecto/public $
```

Conectamos los pulsadores y los leds:



Pulsador rojo:

Pin Elemento Electrónico	Pin Raspberry Pi	Salida
s	2	GPIO 17
v	11	5V PWR
g	6	GND

Pulsador verde:

Pin Elemento Electrónico	Pin Raspberry Pi	Salida
s	12	GPIO 18
v	4	5V PWR
g	14	GND

Led semáforo:

Pin Elemento Electrónico	Pin Raspberry Pi	Salida
gnd	9	GND
g	13	GPIO 27
y	7	GPIO 4
r	-	-

Creamos el archivo index.html para la parte visual en el cliente:

```
<!doctype html>
<html lang="es">

<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1">
  <link rel="stylesheet"
href="https://www.w3schools.com/w3css/4/w3.css">

  <!-- Bootstrap CSS -->
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.m
in.css" rel="stylesheet">
```

```

integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoq
yl2QvZ6jIW3" crossorigin="anonymous">
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.8.1/font/bootstrap
-icons.css">

  <title>Proyecto Marcador</title>
</head>

<body>
  <div class="container">
    <header>
      <div class="card">
        <!-- Titulos y subtítulos -->
        <div class="card-header">
          <h2 class="display-7">Aplicación Web Angular</h2>
        </div>
        <div class="card-body">
          <p class="lead">Tema 15 - Curso 21/22</p>
          <p class="lead">Desarrollo Web Entorno Cliente</p>
        </div>
      </div>
    </header>

    <div class="card">
      <!-- Formulario -->
      <div class="card-header">
        <h3 class="text-center">Proyecto Marcador</h3>
      </div>
      <div class="card-body">
        <table class="table">
          <thead>
            <tr>
              <th scope="col">#</th>
              <th scope="col">Equipo Rojo</th>
              <th scope="col">Equipo Verde</th>
            </tr>
          </thead>
          <tbody>
            <tr>
              <th scope="row">Puntos</th>
              <td id="puntosRojo">0</td>

```



```

        <td id="puntosVerde">0</td>
    </tr>
</tbody>
</table>
<!-- Botones de acción
----->
    <div class="card-footer text-muted text-center">
        <button type="button" id="rojo" class="btn
btn-danger w3-margin-top">Equipo Rojo</button>
        <button type="button" id="verde" class="btn
btn-success w3-margin-top">Equipo Verde</button>

    </div>
</div>
</div>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.0.3/socket.io.js"></script>
<script
src="https://www.w3schools.com/lib/w3color.js"></script>
<script>
    var socket = io(); //load socket.io-client and connect to
the host that serves the page
    window.addEventListener("load", function () { //when page
loads

        var equipoRojo = document.getElementById("rojo");
        var equipoVerde = document.getElementById("verde");

        equipoRojo.addEventListener("click", function () {
//add event listener for when checkbox changes
            socket.emit("boton", 'rojo'); //send button status
to server (as 1 or 0)
            console.log('botonRojo');
        });
        equipoVerde.addEventListener("click", function () {
//add event listener for when checkbox changes
            socket.emit("boton", 'verde'); //send button status
to server (as 1 or 0)
            console.log('botonVerde');
        });
    }

```

```

    });
    socket.on('actualiza', function (verde, rojo) {

        document.getElementById("puntosRojo").innerHTML = rojo;
        document.getElementById("puntosVerde").innerHTML =
verde;

        console.log('entra actualiza');
    });

</script>
<footer class="footer mt-auto py-3 fixed-bottom bg-light">
    <div class="container">
        <span class="text-muted">© 2022
            Antonio Fornell y Diego Sánchez - DWEC - 2º DAW -
Curso 21/22</span>
    </div>
</footer>
</div>
</body>

</html>

```

Creamos el archivo webserver.js para el control de la app:

```

// Añadir router NodeJS Express
const express = require('express');
const app = express();

// Crear servidor
const server = require('http').createServer(app);
const io = require('socket.io')(server);
const port = process.env.PORT || 8080;

// Ruta inicial del servidor
app.get('/', function(req, res) {
    console.log("Cargando HOME");
    res.sendFile(__dirname + '/index.html');
});

// Abrir puertos de escucha del servidor
server.listen(port, function() {
    console.log(`Listening on port ${port}`);
});

```



```

// Directorio de archivos estáticos como JS, Imágenes, etc.
// ==> Por ahora carpeta completa
app.use(express.static(__dirname));

var Gpio = require('onoff').Gpio; //include onoff to interact with the
GPIO
var LEDRojo = new Gpio(4, 'out'); //use GPIO pin 4 as output
var LEDVerde = new Gpio(27, 'out'); //use GPIO pin 4 as output
var pushButtonRojo = new Gpio(17, 'in', 'falling', { debounceTimeout:
350 }); //use GPIO pin 17 as input, and 'both' button presses, and
releases should be handled
var pushButtonVerde = new Gpio(18, 'in', 'falling', { debounceTimeout:
350 });

var contadorRojo = 0;
var contadorVerde = 0;

io.on('connection', function(socket) { // Web Socket Connection

    pushButtonVerde.watch(function(err, value) { //Watch for hardware
interrupts on pushButton GPIO, specify callback function
        if (err) { //if an error
            console.error('There was an error', err); //output error
message to console
            return;
        }
        // get the current state of the LED
        var state = LEDVerde.readSync();
        // write the opposite of the current state to the LED pin
        console.log(Number(!state));
        LEDVerde.writeSync(Number(!state));
        setTimeout(function() {
            LEDVerde.writeSync(0);
        }, 1000);
        contadorVerde++;
        socket.emit('actualiza', contadorVerde, contadorRojo);
        console.log(value);

    });

    pushButtonRojo.watch(function(err, value) { //Watch for hardware
interrupts on pushButton GPIO, specify callback function

```

```

        if (err) { //if an error
            console.error('There was an error', err); //output error
message to console
            return;
        }
        // get the current state of the LED
        var state = LEDRojo.readSync();
        // write the opposite of the current state to the LED pin
        LEDRojo.writeSync(Number(!state));
        setTimeout(function() {
            LEDRojo.writeSync(0);
        }, 1000);

        contadorRojo++;
        socket.emit('actualiza', contadorVerde, contadorRojo);
        console.log(value);

    });

    socket.on('boton', function(data) {
        if (data == 'rojo') {
            contadorRojo++;

        } else {
            contadorVerde++;
        }
        socket.emit('actualiza', contadorVerde, contadorRojo);
    });

    socket.on('botonReset', function(data) {
        if (data == 'reset') {
            contadorRojo = 0;
            contadorVerde = 0;
        }
        socket.emit('actualiza', contadorVerde, contadorRojo);
    });

});

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

```



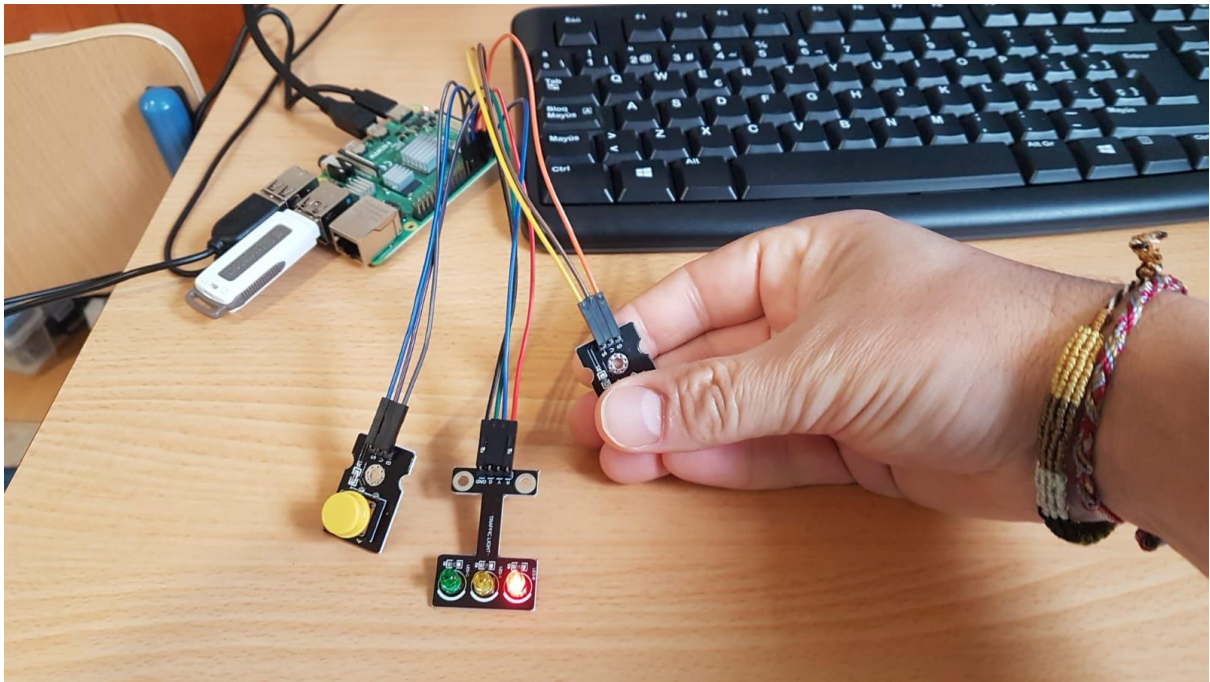
```

pushButtonVerde.unexport(); // Unexport Button GPIO to free
resources
pushButtonRojo.unexport();
contadorRojo = 0;
contadorVerde = 0;
};

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

```

Hacemos varias pruebas:



192.168.80.111:8080

Carta Foodisiac

## Aplicación Web Angular

Tema 15 - Curso 21/22

Desarrollo Web Entorno Cliente

### Proyecto Marcador

#	Equipo Rojo	Equipo Verde
Puntos	15	20

Equipo Rojo

Equipo Verde

Reset

© 2022 Antonio Fornell y Diego Sánchez - DWEC - 2º DAW - Curso 21/22

Creamos proyecto nuevo en angular

Captura de la interfaz del usuario:

