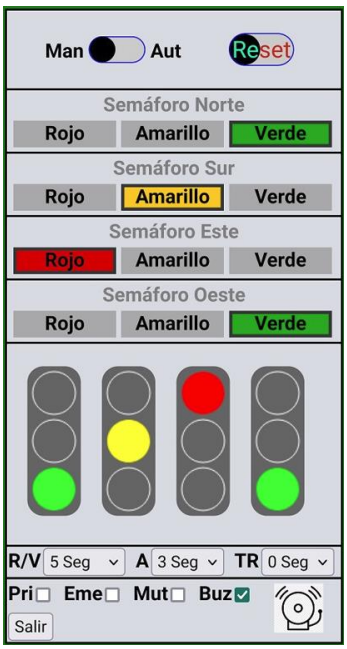
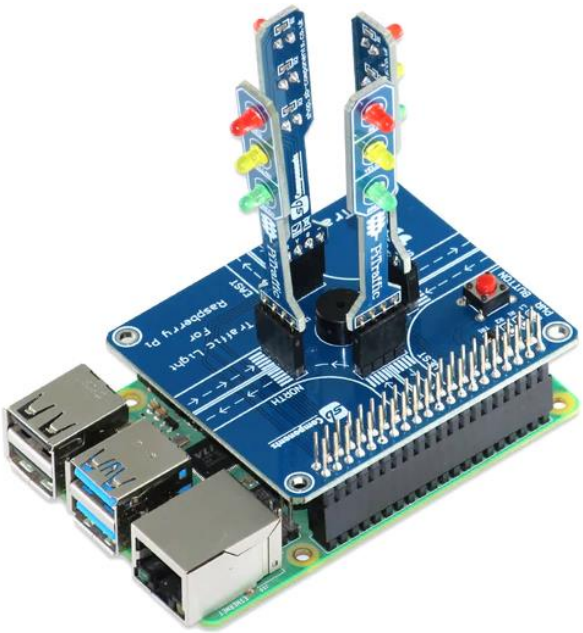
TFG

Grado de Ingeniería Informática





Control experimental de semáforos vía web con RaspBerry Pi y PiTraffic

Estudiante: **Marcos Tomás Leirós Otero**

Profesor responsable: **Santi Caballé Llobet**

Profesor consultor: **Gregorio Robles Martínez**

Fecha de entrega: 09/06/2022

FICHA DEL TRABAJO FINAL TFG

Título del trabajo:	**Control experimental de semáforos vía web con RaspBerry Pi y PiTraffic**
Nombre del autor:	Marcos Tomás Leirós Otero
Nombre del consultor:	Gregorio Robles Martínez
Fecha de entrega:	09/06/2022
Área del Trabajo Final:	Desarrollo WEB
Titulación:	Grado en Ingeniería Informática (Itinerario en Ingeniería de Computadores)

Resumen del Trabajo (máximo 250 palabras):

El presente trabajo final de Grado de Ingeniería en Informática, se desarrolla en el campo de desarrollo web, con el objetivo final de controlar experimentalmente mediante un navegador web, cuatro semáforos de tres luces cada uno (rojo, amarillo y verde), en un cruce de cuatro calles.

Como servidor web se ha escogido el famoso minicomputador Raspberry Pi 3 modelo B, debido a que se emplea su conector de expansión conocido como GPIO, para conectar físicamente los cuatro semáforos. Para facilitar el montaje eléctrico, se ha empleado el sombrero PiTraffic de SB Components, puesto que permite incorporar dichos semáforos simplemente enchufando esta tarjeta encima de la RPi, añadiendo además un zumbador y un pulsador. El zumbador se emplea para avisar de que los semáforos se han puesto en amarillo y van a pasar a rojo, y el pulsador se usa para que un peatón pueda pulsar y poner todos los semáforos en rojo unos instantes.

Este sistema posee tres modos, el manual, el automático y el modo emergencia. En modo manual podemos encender a voluntad las lámparas de todos los semáforos independientemente, el modo automático realiza sin parar un ciclo semafórico apropiado para la calle en estudio, y el modo emergencia hace parpadear sin fin todos los semáforos en amarillo, sonando el zumbador a cada encendido. Los tiempos de encendido cuando trabaja en el modo automático, son modificables por parte del usuario por medio de selects. Además, tiene un control para resetear el sistema, otro para poner la calle este-oeste en prioridad (dando a esta dirección el doble de tiempo que a la calle norte-sur), y la posibilidad de poner el zumbador en mute, para que no moleste.

Abstract (in English, 250 words or less):

The present final project of the Computer Engineering Degree is developed in the field of web development, with the final objective of experimentally controlling, through a web browser, four traffic lights with three lights each (red, yellow and green), at a crossroads of four streets.

The famous Raspberry Pi 3 model B minicomputer has been chosen as the web server, because its expansion connector known as GPIO is used to physically connect the four traffic lights. To facilitate the electrical assembly, the PiTraffic hat from SB Components has been used, since it allows these traffic lights to be incorporated simply by plugging this card on top of the RPi, also adding a buzzer and a button. The buzzer is used to warn that the traffic lights have turned yellow and are going to turn red, and the button is used so that a pedestrian can press and turn all the traffic lights red for a moment.

This system has three modes, manual, automatic and emergency mode. In manual mode we can turn on the lamps of all the traffic lights independently at will, the automatic mode non-stop performs a traffic light cycle appropriate for the street under study, and the emergency mode makes all the traffic lights flash yellow endlessly, sounding the buzzer every time. switched on. The ignition times when working in automatic mode, are modifiable by the user through selects. In addition, it has a control to reset the system, another to put the east-west street in priority (giving this address twice as much time as the north-south street), and the possibility of putting the buzzer on mute, so that do not bother.

Palabras clave (entre 4 y 8):

Web, Control, Cruce, Semáforos, RaspBerry Pi, PiTraffic

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Este TFG, trata de elaborar una página web desde cero, para el “control experimental” a distancia, de un cruce de semáforos mediante cualquier browser, empleando comunicación asíncrona. Para tal fin, se instalará un servidor web en una RaspBerry Pi Modelo 3, y se conectarán los semáforos a dicho servidor, de manera que, desde el navegador web de cualquier cliente, podamos controlar las lámparas de los semáforos a voluntad, sea en modo manual (encendiendo cada luz por separado una a una) o de manera automática (con temporizadores). Por seguridad, existirá una página web de acceso, donde se pedirá loguearse, una vez dentro, tendremos disponibles los controles para el sistema. El semáforo, se instalará físicamente en la RPi empleando el sombrero PiTraffic (de SB Components), el cual posee los semáforos de un cruce de cuatro calles, un zumbador o buzzer, y un button o pulsador. El zumbador lo emplearemos para que, cuando parpadeen las luces amarillas, lo indique de manera sonora, así las personas con movilidad reducida lo podrán tener en cuenta. El pulsador, lo podrán usar los peatones para poner todo el cruce en rojo durante un tiempo, y poder así pasar a pie.

En la pantalla del navegador web se verá en tiempo real como van funcionando los semáforos, es decir, si el semáforo norte tiene la luz roja encendida y el sur la luz verde encendida, se verá tanto en la realidad en el propio PiTraffic conectado al Raspberry Pi, como en la página web.

Las órdenes se darán vía AJAX, y la representación gráfica en pantalla del cruce de semáforos será empleando el CANVAS de HTML5. El encendido y apagado físico de las luces y el zumbador será empleando Python ejecutado en el servidor desde PHP.

Veamos un diagrama con los ítems de la arquitectura de este TFG

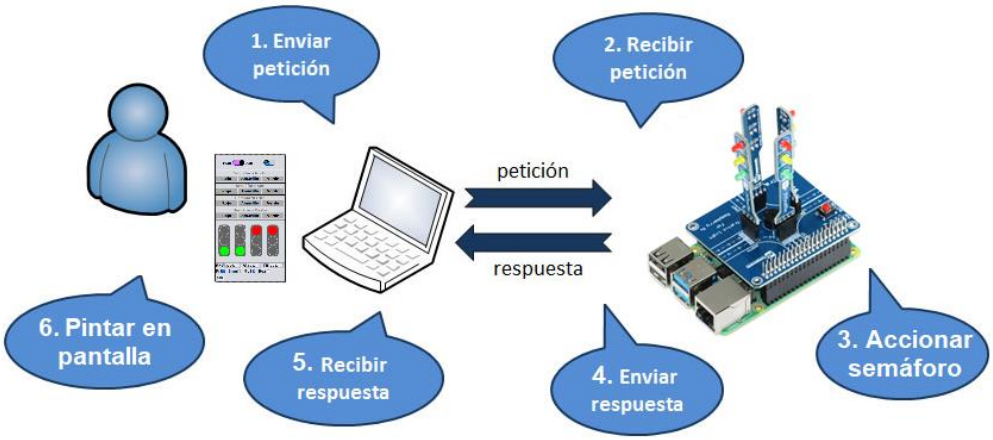


Ilustración 1: Diagrama de la arquitectura

1) El cliente, desde cualquier dispositivo con navegador web, envía un comando de manera asíncrona vía JavaScript AJAX, quedando a la espera de respuesta.

2) El servidor Apache, alojado en la RaspBerry Pi, recibe la petición y usando PHP la analiza en un switch.

3) Según el análisis anterior, el servidor acciona físicamente el pertinente semáforo del PiTraffic, vía Python.

4) El servidor contesta a la petición asíncrona AJAX, haciendo esta respuesta las veces de ACK.

5) El cliente recibe la respuesta a su petición inicial, y la analiza en un switch.

6) Según el análisis anterior, el cliente manda “pintar” el semáforo correspondiente en su pantalla, el cual se representa mediante el uso de canvas.

1.2 Objetivos del Trabajo

El objetivo principal, es diseñar e implementar desde cero una página web dinámica (tanto el lado cliente como el lado servidor), que nos permita acceder remotamente desde un ordenador cliente a un servidor instalado en un miniordenador RaspBerry Pi, de manera que podamos controlar vía web a voluntad las luces y zumbador de un cruce de semáforos, empleando como hardware el sombrero PiTraffic.

Otro de los objetivos, es aprender a integrar conjuntamente diversas tecnologías vistas durante la carrera de **Grado de Ingeniería Informática** de la UOC, en un proyecto único e integrador, las nombro en el punto 2.

Al mismo tiempo, será necesario investigar el uso y la programación del componente HTML5 conocido como CANVAS, muy útil para “dibujar” en tiempo real el estado físico de las lámparas de los semáforos en la pantalla de la computadora cliente.

Otro objetivo de cualquier proyecto, es prever posibles fallos del sistema a desarrollar, encontrando soluciones apropiadas a los mismos.

Por otro lado, terminar a tiempo el proyecto es un objetivo clave, solo hay que ver, que si fuéramos una empresa seguramente nos costaría dinero y tiempo no cumplir con los plazos.

En este TFG, no se requiere diseñar ni modificar hardware, para evitarlo, vamos a emplear el sombrero PiTraffic para la RPi, que nos permite visualizar físicamente un cruce de dos calles con cuatro semáforos.

1.3 Enfoque y método seguido

A la hora de elaborar un proyecto podemos desarrollarlo completamente nuevo o adaptar uno ya existente. En este proyecto se trabaja tanto con software como con hardware, y se han escogido estrategias diferentes.

En cuanto al hardware se ha escogido emplear tecnologías ya existentes, para facilitar el montaje físico y debido a que no estamos ante un grado de ingeniería electrónica. Para alojar el servidor web, se ha escogido un minicomputador conocido como Raspberry Pi, puesto que tiene la potencia y características requeridas, además, dispone de un conector conocido como GPIO, que permite conectar dispositivos externos, que es nuestro caso, son las lámparas de los semáforos. No obstante, por comodidad, finalmente se ha decidido emplear además el sombrero PiTraffic, que es una placa de circuito impreso que se conecta directamente en el GPIO, sin necesidad de cablear.

Por lo que a software se refiere, se ha escogido crearlo desde cero, teniendo en cuenta que, tras un estudio del “estado del arte”, no se ha encontrado nada similar, más allá de sistemas profesionales de altísimo coste y de implementación “secreta”. Por otro lado, se ha tomado una decisión muy importante de enfoque, teníamos la posibilidad de poner el control en el lado servidor o en el lado cliente. Desde mi punto de vista, para un sistema profesional, el control debería de residir en el servidor, de manera de que si cae un cliente el sistema funcione autónomamente, e incluso que el cliente valga para controlar múltiples cruces de calles, sin embargo, adoptar este enfoque dificultaría una funcionalidad muy importante para este proyecto, y es que se desea que en la pantalla del cliente se visualicen en tiempo real las lámparas que están encendidas o apagadas. Por ello, y teniendo en cuenta que este es un trabajo “experimental” y por ende no profesional, se ha decidido poner el control en el lado cliente, se manera que se “pinten” los semáforos en pantalla a medida que se van recibiendo respuestas a las llamadas de control iniciadas mediante AJAX.

1.4 Planificación del Trabajo

Para este trabajo, disponemos como máximo de 300 horas, oficialmente se puede comenzar a trabajar el 16/02/2022 y debe estar terminado y entregado el 09/06/2022, esto hacen 16 semanas más 2 días. Al anterior período hay que añadir otros días extra posteriores para la defensa ante un tribunal, puesto que será tras la fecha de entrega. Además, hay unos días anteriores usados para la negociación de la temática del TFG vía email con el profesor. Teniendo en cuenta a mi situación familiar, laboral, y académica, para esta planificación temporal preveo trabajar todos los días entresemana sean laborables o no. Como medida de contingencia, en el caso de no poder disponer de un día por cualquier causa, las horas de ese día perdido se trasladarán a cualquier otro día, especialmente en festivos o fin de semana.

En estas horas hay que incluir absolutamente todo lo implicado en el proyecto, tanto las tareas "académicas" de entrega de PEC’s como la parte de

“la defensa” (que implica una memoria final, un vídeo, unas transparencias tipo “PowerPoint" y un cuestionario de autoevaluación).

Haciendo un estudio preliminar de la planificación del trabajo, he identificado los siguientes ítems o subtareas del trabajo global. No obstante, esta planificación no está exenta de riesgos o de nuevas ideas surgidas durante su seguimiento, de manera que podría ser modificada durante su ejecución. Existen ítems generales y complementarios.

Ítems generales

- Estudio inicial y propuesta de TFG.
- PEC1 (Plan de trabajo).
- PEC2 (Análisis y diseño).
- PEC3 (Implementación).
- PEC4 (Memoria y presentación).
- Defensa ante el Tribunal virtual (Asíncrona).

Ítems complementarios

- Investigación sobre RPi y PiTraffic para ver su idoneidad.
- Contextualización y diseño (DCU) de la aplicación WEB.
- Compra del hardware y comprobación tras su llegada.
- Montaje de los refrigeradores en la RPi e inserción en su caja.
- Instalación del SO, actualización y configuración de la RPi.
- Instalación y configuración del servidor WEB Apache. Pruebas.
- Instalación y configuración de PHP. Pruebas.
- Instalación y configuración de MySQL. Pruebas.
- Montaje del PiTraffic sobre la RPi en el GPIO 40.
- Instalación de la librería en Python del PiTraffic. Pruebas.
- Diseño de interfaces
- Programación inicial de lado cliente con JS.
- Investigación de uso del CANVAS de HTML5
- Programación inicial de lado servidor con PHP.
- Programación Python.
- Programación de estilos CSS.
- Pruebas iniciales con el código base del proyecto.
- Mejoras en general del código inicial tras las pruebas.
- Testeo final.
- Elaboración de la memoria final del TFG y entrega.

Para expresar gráficamente la temporalización, empleo el Microsoft Project, elaborando con él un Diagrama de Gantt. Se muestran dos, el de ítems generales y el de ítems complementarios.

Diagramas de Gantt

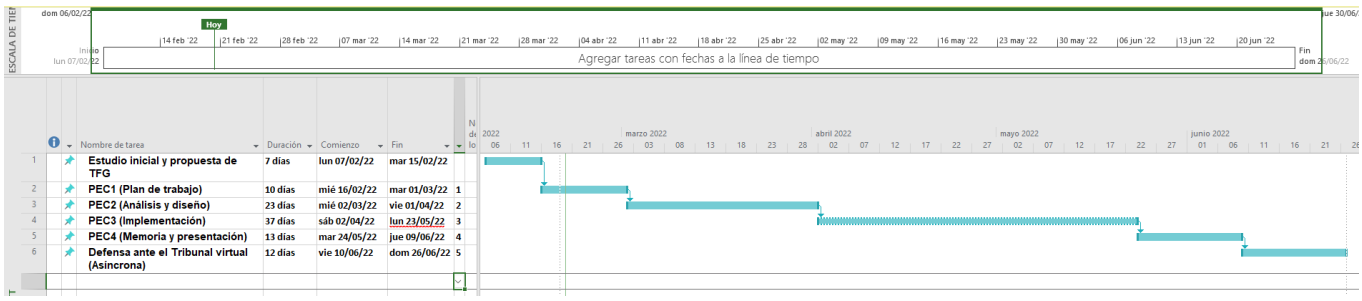
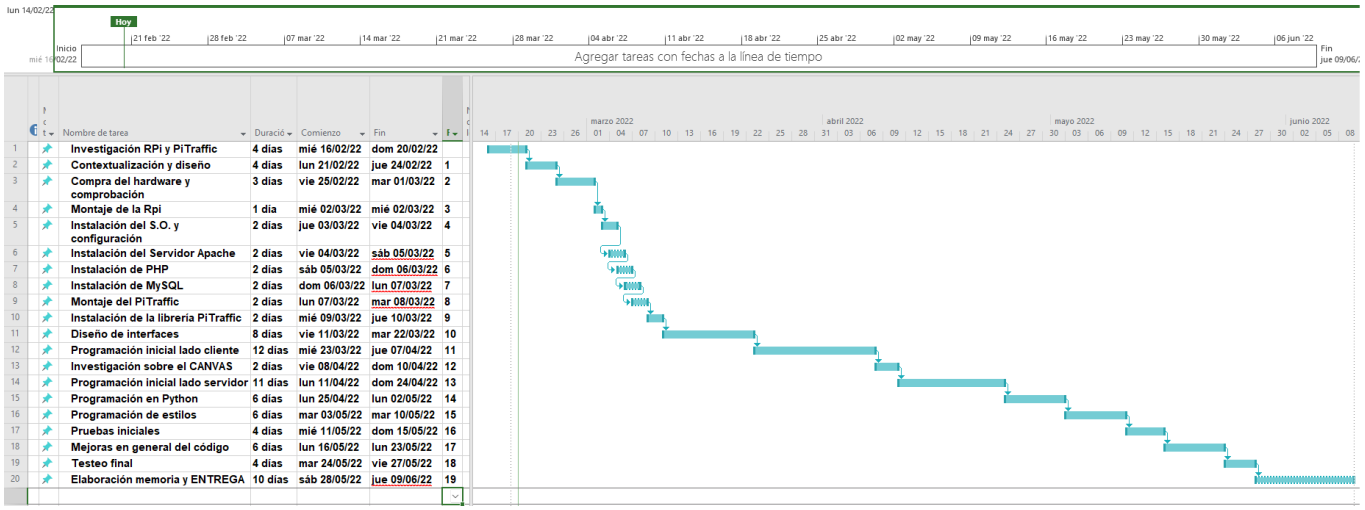


Ilustración 2: Gantt con los ítems generales



	Nombre de tarea	Duració	Comienzo	Fin	P
1	Investigación RPi y PiTraffic	4 días	mié 16/02/22	dom 20/02/22	
2	Contextualización y diseño	4 días	lun 21/02/22	jue 24/02/22	1
3	Compra del hardware y comprobación	3 días	vie 25/02/22	mar 01/03/22	2
4	Montaje de la Rpi	1 día	mié 02/03/22	mié 02/03/22	3
5	Instalación del S.O. y configuración	2 días	jue 03/03/22	vie 04/03/22	4
6	Instalación del Servidor Apache	2 días	vie 04/03/22	sáb 05/03/22	5
7	Instalación de PHP	2 días	sáb 05/03/22	dom 06/03/22	6
8	Instalación de MySQL	2 días	dom 06/03/22	lun 07/03/22	7
9	Montaje del PiTraffic	2 días	lun 07/03/22	mar 08/03/22	8
10	Instalación de la librería PiTraffic	2 días	mié 09/03/22	jue 10/03/22	9
11	Diseño de interfaces	8 días	vie 11/03/22	mar 22/03/22	10
12	Programación inicial lado cliente	12 días	mié 23/03/22	jue 07/04/22	11
13	Investigación sobre el CANVAS	2 días	vie 08/04/22	dom 10/04/22	12
14	Programación inicial lado servidor	11 días	lun 11/04/22	dom 24/04/22	13
15	Programación en Python	6 días	lun 25/04/22	lun 02/05/22	14
16	Programación de estilos	6 días	mar 03/05/22	mar 10/05/22	15
17	Pruebas iniciales	4 días	mié 11/05/22	dom 15/05/22	16
18	Mejoras en general del código	6 días	lun 16/05/22	lun 23/05/22	17
19	Testeo final	4 días	mar 24/05/22	vie 27/05/22	18
20	Elaboración memoria y ENTREGA	10 días	sáb 28/05/22	jue 09/06/22	19

Ilustración 3: Gantt con los ítems complementarios

1.5 Breve sumario de productos obtenidos

Al finalizar este proyecto final de carrera, se espera obtener los siguientes productos.

a) Hardware completamente montado: Incluye el montaje de la Raspberry Pi en una caja con todos los accesorios requeridos, y la instalación del cruce con los cuatro semáforos sobre la RPi.
b) Código del lado cliente: Será el encargado del control del sistema, mandando comandos vía AJAX, y de “pintar” en pantalla el estado real de los semáforos según la respuesta recibida.
c) Código de lado servidor: Controlará físicamente las lámparas de los semáforos, actuando según las órdenes recibidas del cliente, y proporcionando respuestas al cliente.

1.6 Breve descripción de los otros capítulos de la memoria

Se describen brevemente el resto de capítulos del presente TFG.

a) Tecnologías y materiales a emplear: Analiza tanto el hardware como el software requerido para este trabajo.
b) Análisis y diseño (DCU): Diseño centrado en el usuario
c) Implementación: Pormenoriza las decisiones tomadas durante la escritura del código
d) Revisión de la planificación: Control de la consecución de ítems perseguidos
e) Conclusiones: Valoraciones personales
f) Glosario: Aclaración de términos científico-técnicos
g) Bibliografía: Material consultado
h) Anexos: Código fuente completo (disponible en GitHub)

2. Tecnologías y materiales a emplear

En cuanto a las tecnologías vamos emplear tanto software como hardware específico:

Hardware

- RaspBerry Pi 3
- Sombrero PiTraffic para la RPi
- Alimentador (5,1VDC/2,5A)
- Tarjeta microSD (8GB mínimo)
- Disipadores
- Caja

Software

- Sistema Operativo RPi OS Legacy (Linux)
- Servidor Web APACHE2
- MariaDB/MySQL (BDD de los usuarios/contraseñas para el acceso seguro)
- phpMyAdmin
- HTML5 (Canvas, Audio y LocalStorage)
- PHP (Para el código de lado servidor)
- JS (Para el código de lado cliente)
- PYTHON (Para el control real, activado desde PHP)
- CSS3 (Para los estilos)
- AJAX (Para mandar los comandos al servidor y gestionar las respuestas)
- DOM (necesario para modificar la web sin refrescar)

Veamos seguidamente las características del RaspBerry Pi 3 y del PiTraffic.

Ilustración 4: RaspBerry Pi 3 Model B v1.2 de 2015

CARACTERÍSTICAS

- CPU Quad Core ARM Cortex A-53 1.2GHz 64bit ARMv8 (Broadcom BCM2837 Chipset)
- 1GB RAM LPDDR2
- WiFi wireless LAN IEEE 802.11b/g/n (BCM43143)
- Bluetooth 4.1 Low Energy BLE (BCM43143)
- 10/100 BaseT Ethernet
- 40-pin extended GPIO

- 4 USB 2.0 ports
- 4 Pole stereo Jack 3,5mm output and composite video port (for RCA's)
- GPU Broadcom Dual Core Video Core IV
- Full size HDMI v1.4 1080p H.264 Full HD width Audio
- CSI-2 camera port for connecting a Raspberry Pi camera 15-pin
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD slot port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A (5.1VDC)
- Expansion header GPIO de 2x20 pins
- Dimensions 85 x 56 x 17 mm
- Weight 45 grams (Without box)

PiTraffic (Raspberry Pi Traffic Light HAT)

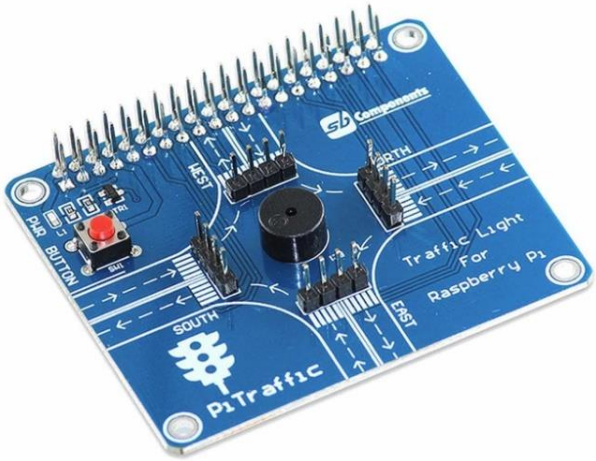


Ilustración 5: PiTraffic HAT (sin semáforos)

Ilustración 6: PiTraffic montado en la RPi (con semáforos)

CARACTERÍSTICAS

- Raspberry Pi compatibility, with 40 pins GPIO
- 4 directions traffic lights (East, North, South and West)
- Buzzer
- Push button
- 4-wire SPI communication
- 5VDC

DIMENSIONES

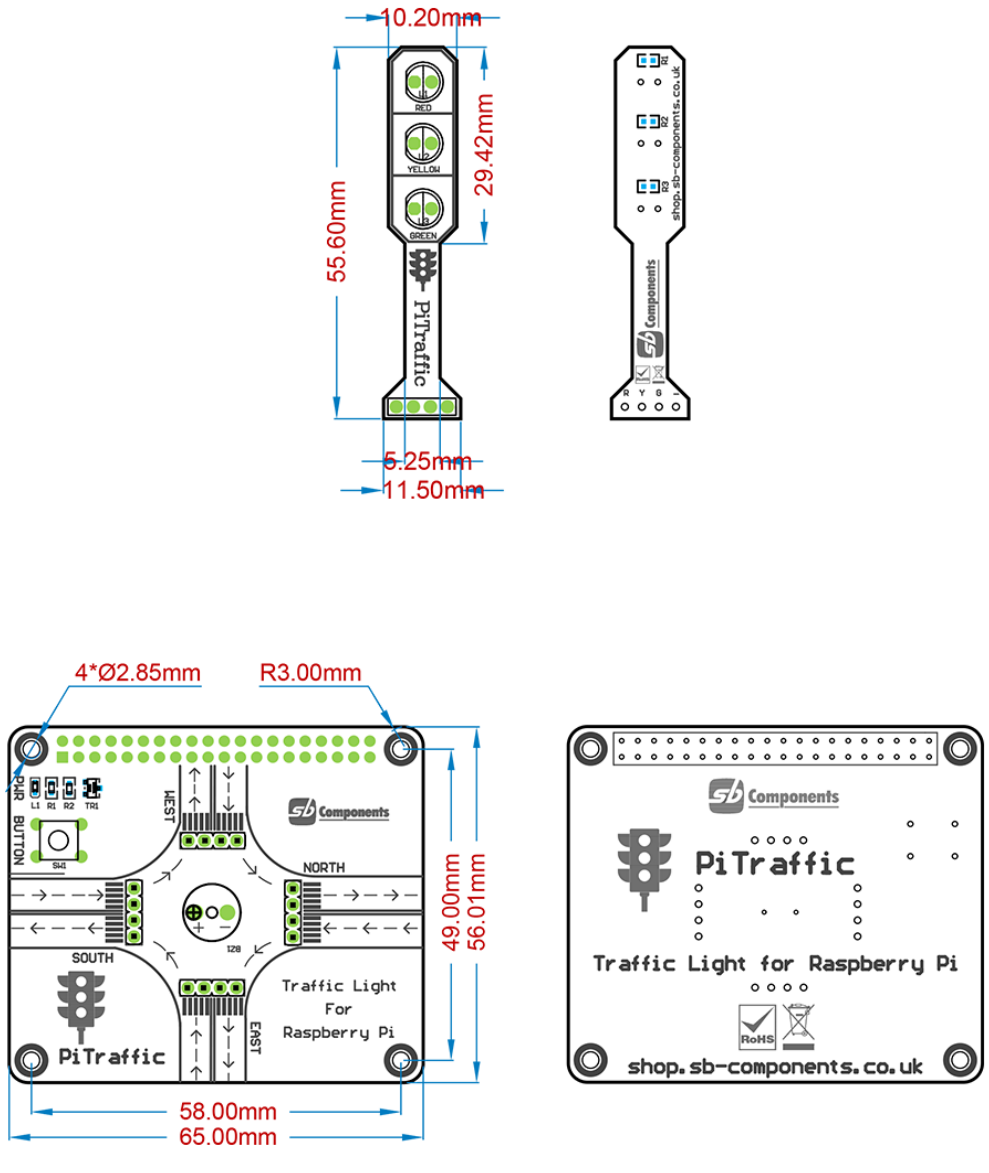


Ilustración 7: Dimensiones del PiTraffic

3. Análisis y diseño (DCU)

3.1 DCU

En este punto, se comienza con el diseño del sistema sobre el papel, pero sin entrar a escribir código, para ello, voy a aplicar la técnica del diseño DCU (Diseño Centrado en el Usuario) para decidir cómo será el aspecto de los interfaces según las funcionalidades. Se recomienda en cualquier diseño tener en cuenta “el estado del arte”, es decir, que diseños parecidos hay en el mercado, que funcionalidades tienen etc.

Siguiendo la técnica DCU, una persona que use el sistema puede ser focal, secundaria o excluida. Y las funcionalidades, aspecto gráfico etc. Tienen que diseñarse pensando en ellos.

►Persona focal: Se tratarían de usuarios que debido a su actividad laboral se dedican a gestionar el tráfico para pequeñas o medias poblaciones. La calidad del sistema para este tipo de personas debe ser muy exigente, tanto, que se podrían desarrollar versiones distintas del sistema para estos y para el grupo siguiente.

Ejemplo (ficha de una persona focal): Manolo López tiene 42 años está casado y tiene un hijo mayor, vive en Castellón de la Plana que es una ciudad de tamaño medio, y es bastante responsable. De joven estudió un CFGS en electrónica, y tiene 15 años de experiencia en la empresa privada en trabajos técnicos relacionados con la informática. Acaba de aprobar una oposición al ayuntamiento de la ciudad, y es destinado a la sala de control de tráfico.

►Persona secundaria: En este caso estamos, hablando de estudiantes (o forofos del mundo “maker”) que tengan que realizar prácticas o experimentos con objetivos educativos. Es evidente, que para este tipo de usuarios la calidad puede ser experimental, o media como mucho.

Ejemplo (ficha de una persona secundaria): Iván Otero tiene 19 años y es soltero, vive en Valencia que es una gran ciudad. Ha estudiado bachiller y no ha trabajado nunca. En la actualidad es alumno de la Universidad Politécnica, y en una de las asignaturas técnicas, un profesor ha decidido realizar como práctica una mejora de un sistema de control de un cruce de semáforos.

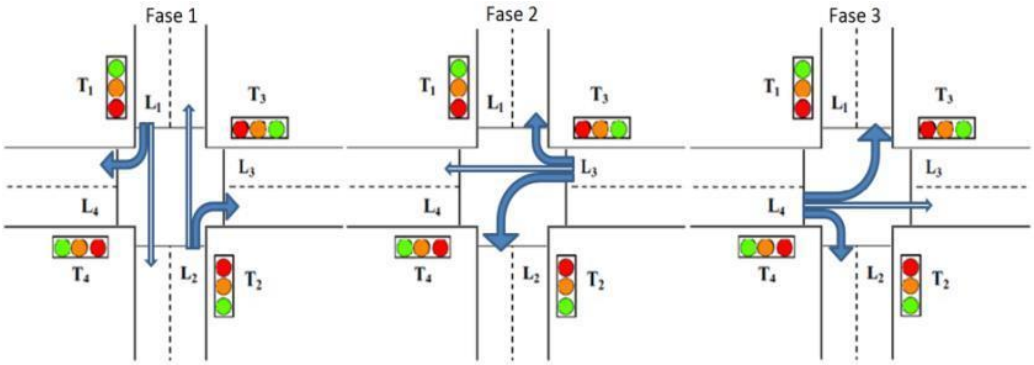
►Persona excluida: Todos los demás, debido a que no tiene sentido que usuarios que no gestionen el tráfico de una población o que estén realizando prácticas educativas empleen este sistema. Normalmente estamos hablando de personas con bajo perfil técnico o TIC, es decir, personas sin estudios o con estudios de letras y sin mucha devoción por las nuevas tecnologías.

3.2 Fases o ciclo semafórico

También hay que decidir cómo será el ciclo automático de semáforos, el PiTraffic lleva incorporado hardware para un cruce de cuatro semáforos con

tres luces cada uno, pero no lleva semáforos de peatones de dos luces, por ello, he decidido realizar un ciclo en tres fases principales sin tener en cuenta los peatones, este tipo de ciclo requiere que en una de las dos calles en ambas direcciones se tenga prohibido girar a la izquierda (fase 1 en el siguiente diagrama), lo cual se indicará a los conductores mediante señales de tráfico junto a los correspondientes semáforos.

Ilustración 8: Gráfico con las tres fases principales



Evidentemente, el funcionamiento de un cruce con estas tres fases, requieren como mínimo de tres fases más auxiliares, se trataría de intercalar fases intermedias para mostrar el amarillo en los semáforos que este en verde antes de pasar a rojo, estas tres fases intermedias activarán el buzzer sonoro como indicación. Por otro lado, por motivo de seguridad, en muchos lugares se ponen todos los semáforos en rojo durante un instante justo antes de que se ponga algún(os) semáforo(s) en verde, por ello he decidido intercalar tres fases adicionales intermedias para mostrar todo en rojo antes de pasar a verde, estas fases funcionarán o no de manera opcional, como el tiempo TR va a ser configurable si lo ponemos a 0 se desactivarán estas nuevas fases.

Veamos una tabla con el funcionamiento final en 9 fases (con los tres anteriores como base), que completan un ciclo (las fases 3/6/8 tendrán accionamiento opcional mediante un select):

Fase/Semáforo	**Norte**	**Sur**	**Este**	**Oeste**	**BUZZER**
1	Verde	Verde	Rojo	Rojo	OFF
2	Amarillo	Amarillo	Rojo	Rojo	ON 2seg
3 (opcional)	Rojo	Rojo	Rojo	Rojo	OFF
4	Rojo	Rojo	Verde	Rojo	OFF
5	Rojo	Rojo	Amarillo	Rojo	ON 2seg
6 (opcional)	Rojo	Rojo	Rojo	Rojo	OFF
7	Rojo	Rojo	Rojo	Verde	OFF
8	Rojo	Rojo	Rojo	Amarillo	ON 2seg
9 (opcional)	Rojo	Rojo	Rojo	Rojo	OFF

Ilustración 9: Tabla con las nueve fases

NOTA: Existirá una fase especial 0, que no formará parte del ciclo automático, se trata de un estado especial al cual se llegará cuando un peatón pulse un button o pulsador que lleva incorporado el PiTraffic, así conseguirá que todos los semáforos se pongan en rojo un tiempo para que dicho peatón pueda cruzar a pie.

3.3 Bases de datos

Para la pantalla de autenticación se requiere de una BBDD, la cual no requiere de una gran modelización teórica, puesto que es muy simple.

3.3.1 Diseño conceptual

El diagrama entidad-relación (E-R) sale muy sencillo, teniendo en cuenta que al disponer de una sola entidad E, va a ser una BDD de tabla única, por lo que no hay relación R entre entidades. La entidad *usuarios* tendrá los atributos *usuario* y *clave*. Aparece aquí una dependencia pues toda *clave* necesita la existencia previa de un *usuario*, y este último no se puede repetir, lo que hace que el campo clave sea el atributo *usuario*.

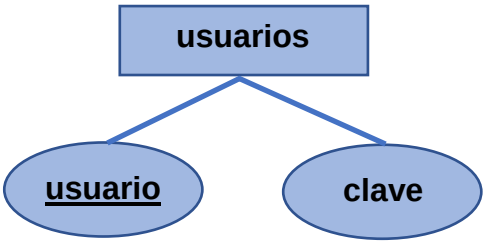


Ilustración 10: Diagrama entidad-relación de la BDD

3.3.2 Diseño lógico

La única tabla *usuarios* de la BBDD *pitraffic* tendría el siguiente aspecto:

Tabla usuarios		
usuario (campo clave)	**clave**	**privilegios**
pi	raspberry	normal
marcos	leiros	admin
gregorio	robles	admin
santi	caballe	admin

Ilustración 11: Tabla usuarios de la BDD pitraffic

NOTA: Se ha añadido el campo *privilegios* sin funcionalidad, es para posible uso futuro.

3.3.3 Diseño físico

Al ser un proyecto web se ha escogido como SGBD MySQL/MariaDB. Veamos la estructura de la BDD *pitraffic* y su tabla *usuarios* (con campos *usuario*/*clave*) desde el gestor web phpMyAdmin.

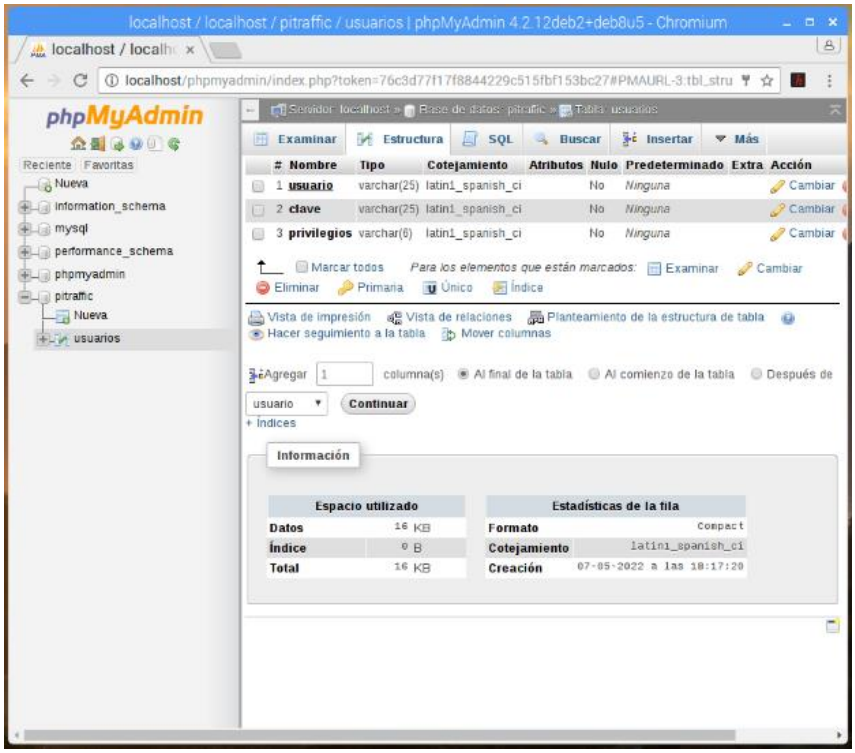


Ilustración 12: Estructura de la tabla usuarios

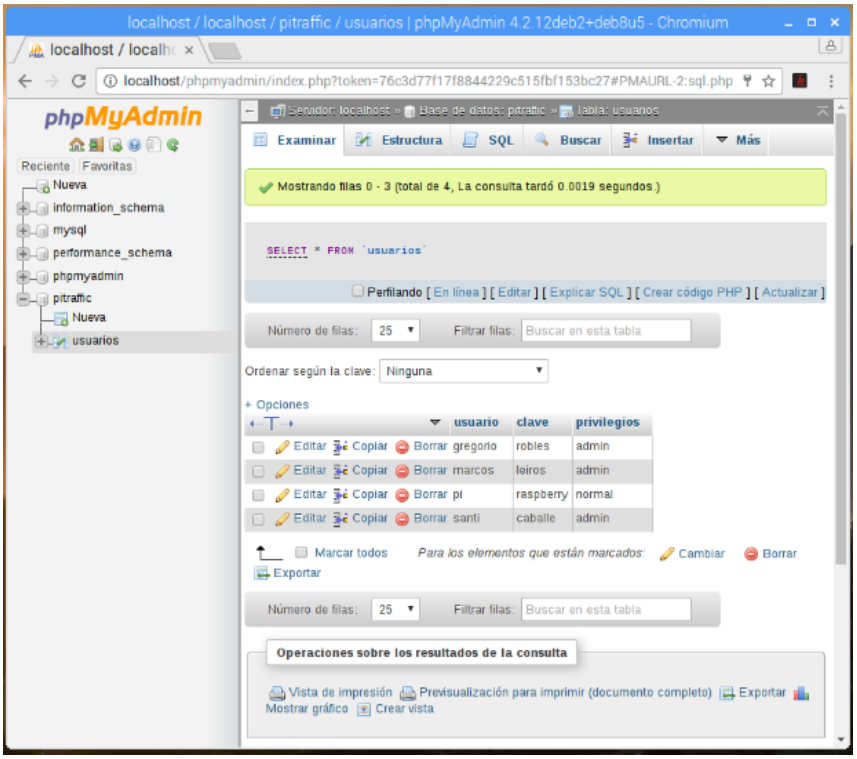


Ilustración 13: Examinando la tabla usuarios

NOTA: Por seguridad, en este TFG no se contempla la posibilidad de poder crear cuentas de usuario desde la web, puesto que los usuarios serán muy limitados y los creará un administrador desde phpMyAdmin.

3.4 Prototipos

La aplicación va a poseer dos pantallas, veamos un posible aspecto HTML/CSS tanto de la pantalla login como de la de control. Debe poseer RWD.

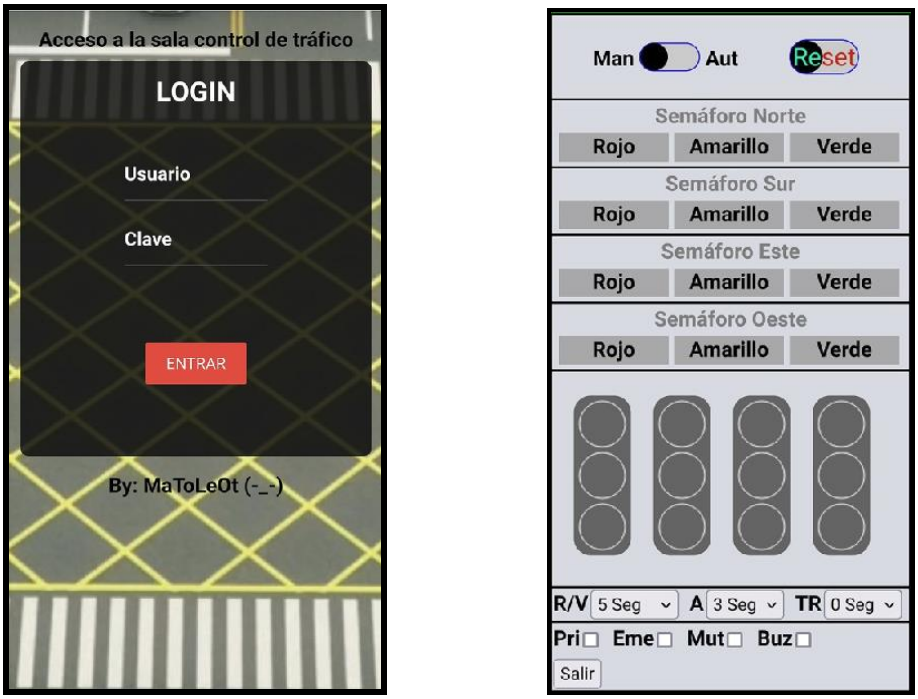


Ilustración 14: Prototipos de diseño

4. Preparación del hardware y software

4.1 Hardware

Se trata del montaje físico de la RPi en su caja junto con sus accesorios, incluido el sombrero PiTraffic. Al montar el PiTraffic, las cajas disponibles para la RPi ya no valen (al menos la parte superior). Como creo que por seguridad tanto mecánica como eléctrica es mejor poner una caja, escojo una de plástico impresa en impresora 3D y le corto con una sierra la parte superior, de manera que se “vea” el PiTraffic. Vemos la RPi en la caja y con la tapa sin cortar.

Ilustración 15: RPi en la caja y con la tapa sin cortar

Inserto el PiTraffic, conectándolo en el GPIO (sin los semáforos enchufados).

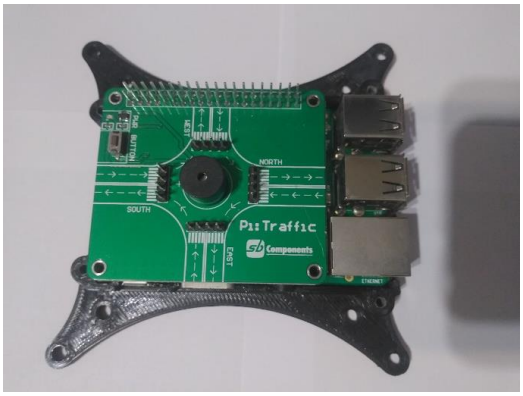


Ilustración 16: PiTraffic conectado en el GPIO

Corto la tapa con una sierra de marquetería.

Ilustración 17: Caja con parte superior cortada

PiTraffic dentro de la caja cortada, ya con los semáforos insertados.

Ilustración 18: PiTraffic totalmente montado en la RPi

4.2 Software

Una vez tenemos el hardware preparado, es el momento de instalar en la Raspberry Pi, todo el software necesario en su tarjeta microSD (que hace las veces de disco duro), incluido el Sistema Operativo.

4.2.1 Instalación del S.O.

Para instalar el S.O. en la SD lo más cómodo es emplear el “Raspberry Pi Imager” de la URL https://www.raspberrypi.com/software/, el archivo de llama imager_1.7.1.exe, lo ejecutamos en Windows con doble clic para instalar, y seguimos las indicaciones. Una vez instalado el RPi Imager el aspecto será el siguiente:

Ilustración 19: Aspecto del RPi Imager (I)

Podemos apreciar como disponemos de dos button, con el primero escogemos el S.O., y mediante el segundo el destino de instalación que será la microSD insertada en un lector de tarjetas. Como vamos a emplear una vieja **RaspBerry Pi 3 Model B v1.2** de 2015 instalamos la versión del S.O. Legacy (por problemas de rendimiento). Una vez seleccionado el SO y el destino de instalación aparecerá el button “Write” para comenzar la instalación.

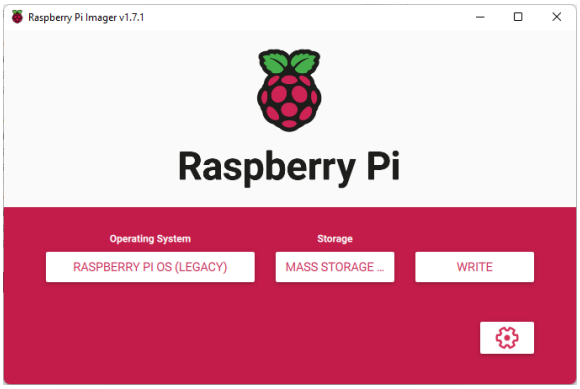


Ilustración 20: Aspecto del RPi Imager (II)

Una vez completada la instalación del S.O., insertamos la SD en la RPi y la arrancamos, seguiremos las indicaciones del asistente que aparece al ser la primera vez que se inicia, nos pedirá el idioma, contraseña, configuración wifi etc.

4.2.2 Configuración básica y actualización.

Una vez arrancado en modo gráfico, es ideal configurar la apariencia en el menú de inicio -> preferencias -> apariencia.

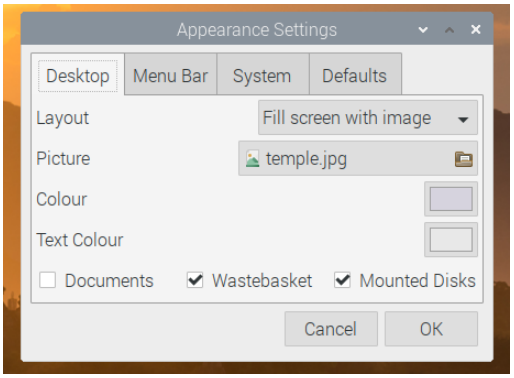


Ilustración 21: Aspecto del menú apariencia

Seguidamente, es recomendable repasar las nueve opciones que nos ofrece la herramienta de configuración de la RPi, modificando alguna si nos es preciso, esta herramienta de visualiza ejecutando en el terminal (Ctrl+t):

```
sudo raspi-config
```

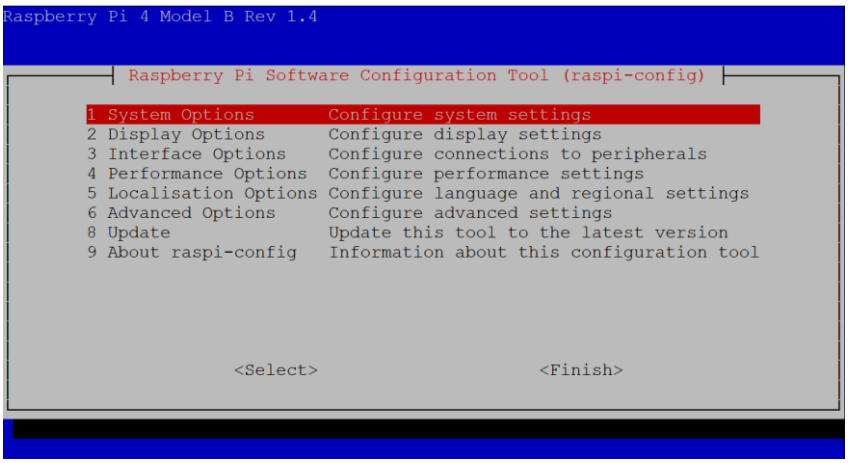


Ilustración 22: Aspecto de raspi-config

Seguidamente, se recomienda actualizar y reiniciar:

```
sudo apt-get update
sudo apt-get upgrade
sudo reboot
```

4.2.3 Desactivar el protector de pantalla.

El S.O. RPi Legacy, lleva activo un protector de pantalla muy molesto, que a fecha de hoy no se puede desactivar gráficamente desde los menús. Para desactivarlo, seguimos el siguiente proceso:

Editamos el siguiente archivo de configuración:

```
sudo nano /etc/xdg/lxsession/LXDE-pi/autostart
```

Anulamos la línea con #

```
#@xscreensaver -no-splash
```

Añadimos luego la siguiente línea

```
@xset dpms 0 0 0
@xset s off -dpms
```

Reiniciamos

```
sudo reboot
```

4.2.4 Configurar una IP estática

En un servidor web lo ideal es tener una IP estática, de esta manera es más fácil configurar el Router, Firewall y el DDNS (si fuera necesario, como lo es en nuestro caso) para ello editamos el archivo:

```
sudo nano /etc/dhcpcd.conf
```

Descomentamos y modificamos las siguientes líneas:

```
#estatic cable ip
interface eth0
static ip_address=192.168.1.10/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

Como se puede apreciar, hemos fijado para Ethernet vía cable la IP 192.168.1.10 siendo nuestro Router doméstico 192.168.1.1, en caso de querer hacerlo para el WIFI hay que sustituir el eth0 por wlan0. Es necesario reiniciar para que funcione.

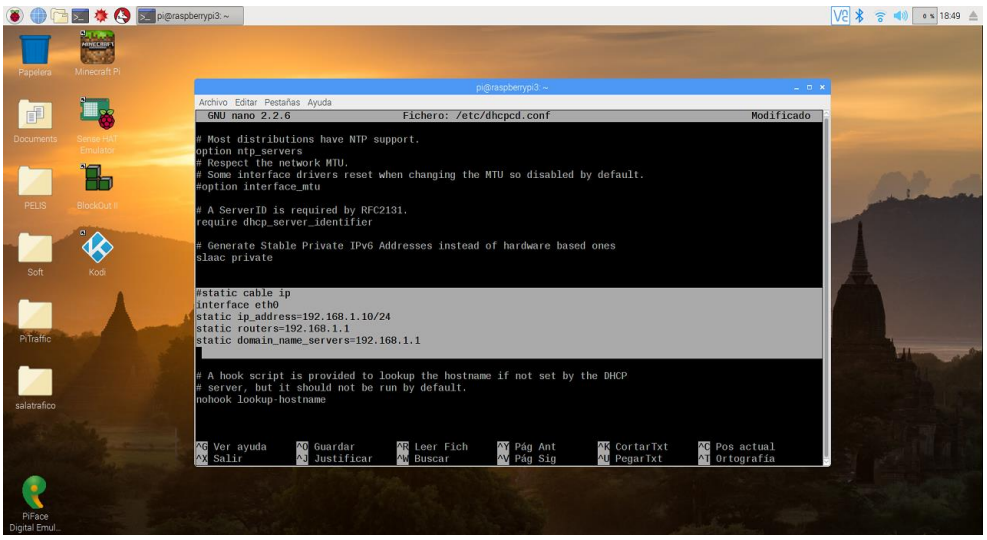


Ilustración 23: Fichero /etc/dhcpcd.conf editado

Para no tener problemas de acceso desde el exterior hemos configurado esa IP fija como un DMZ (zona desmilitarizada) en el Router doméstico.

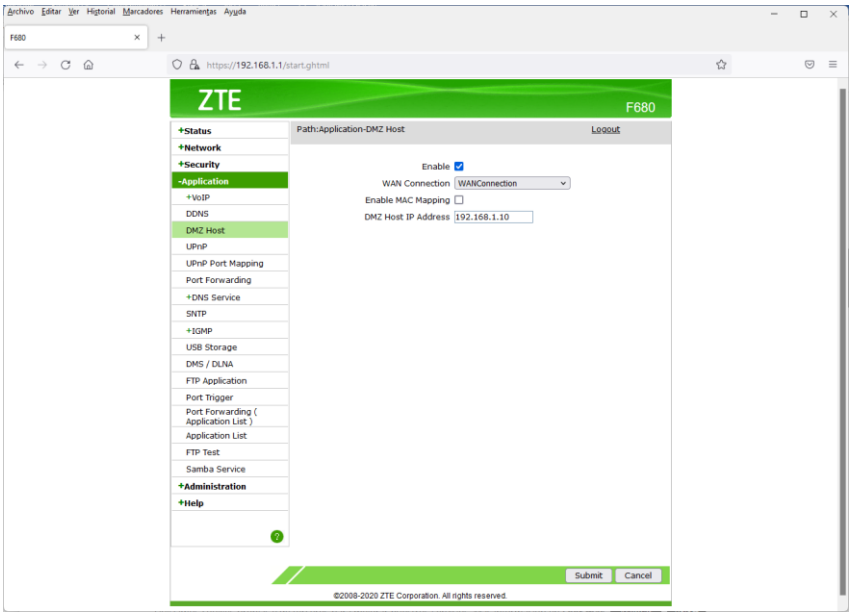


Ilustración 24: DMZ activado en el router

4.2.5 Instalación del servidor web

Para este proyecto necesitamos el siguiente entorno: Apache, PHP, mysql/MariaDB, phpMyAdmin y NO-IP. Aunque se puede instalar conjuntamente lo hacemos paso a paso.

Apache2

Ejecutamos:

```
sudo apt install apache2
```

Luego podemos ver si está en running mediante:

```
sudo systemctl status apache2
```

Cuando sea necesario podemos reiniciar el servicio mediante:

```
sudo /etc/init.d/apache2 restart
```

Para probar su funcionamiento, basta con poner sencillamente la URL **localhost** en el navegador Web Wrowser.

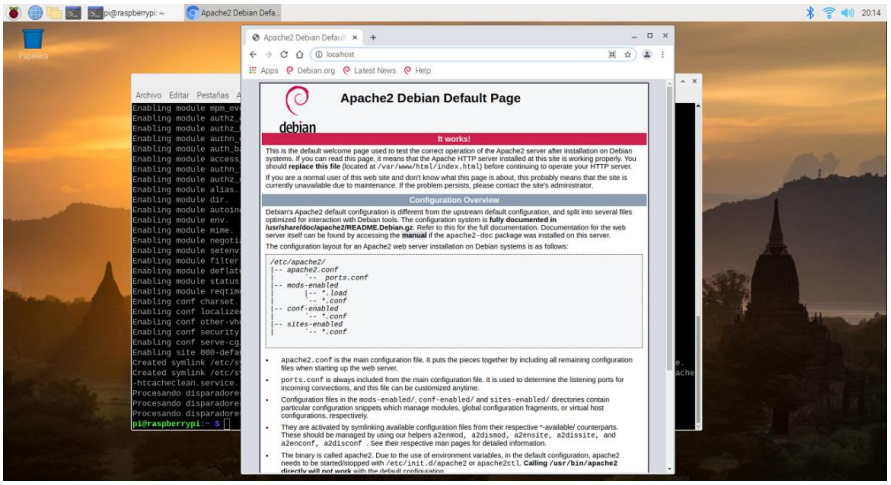


Ilustración 25: Sevidor web Apache2 en funcionamiento

Esta web de prueba se encuentra alojada en **/var/www/html** (ver mediante cat var/www/html/index.html) y es donde tendremos que poner nuestro proyecto.

Para evitar problemas añadimos el usuario pi (creado por defecto en las RPi) al grupo www-data que se creó automáticamente (ver antes y después /etc/passwd y/o /etc/group). Además, cambiamos la propiedad y permisos de /var/www/html/, así nos dejará trabajar con ella libremente.

```
sudo usermod -aG www-data pi
sudo chown -R pi:www-data /var/www/html/

sudo chmod -R 770 /var/www/html/
```

PHP (con las funciones para cadenas de texto multibyte mbstring)

Ejecutamos:

```
sudo apt-get install php php-mbstring
```

Luego para probarlo, creamos el archivo info.php en /var/www/html con el contenido:

```
<?php
      phpinfo();
?>
```

Si se prueba con el navegador, y sale el error 500, seguramente será debido a los permisos, por ejemplo, por ser del grupo pi, le cambiamos la propiedad a los archivos mediante (recuerda hacerlo siempre que mandes archivos si hiciera falta):

```
sudo chown -R pi:www-data /var/www/html/
```

Luego lo probamos sencillamente poniendo en el navegador la url **localhost/info.php**

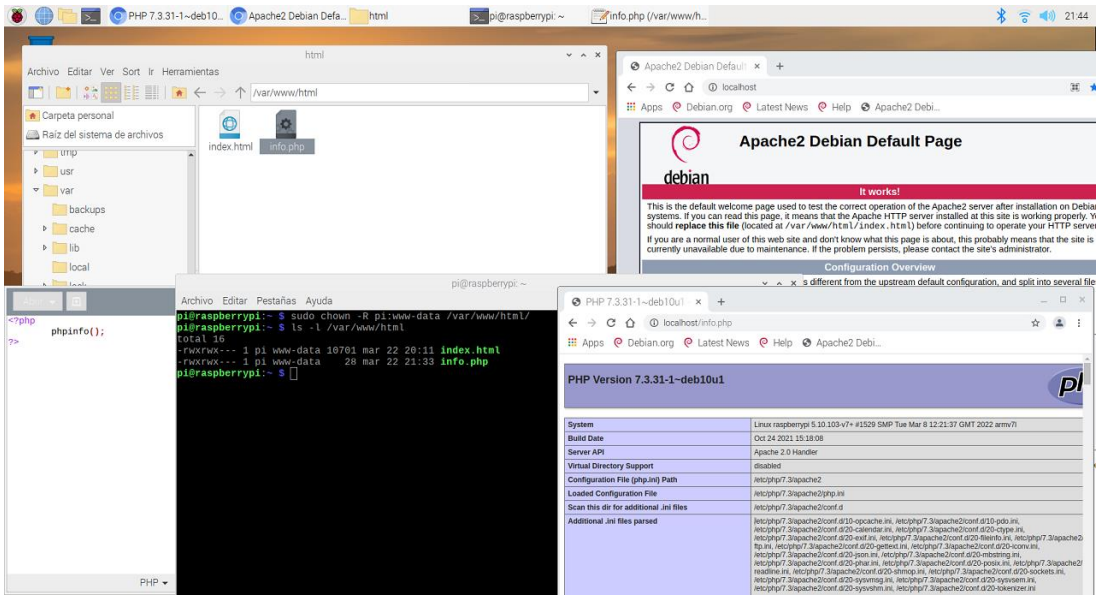


Ilustración 26: PHP en funcionamiento

BBDD (MariaDB/MySQL)

La base de datos la necesitamos para el proceso de login (usuarios/contraseñas). Ejecutamos:

```
sudo apt install mariadb-server
sudo apt install php-mysql
```

Luego la podemos proba sencillamente tecleando el comando **myqsl**, y saliendo con **quit** o **exit**.

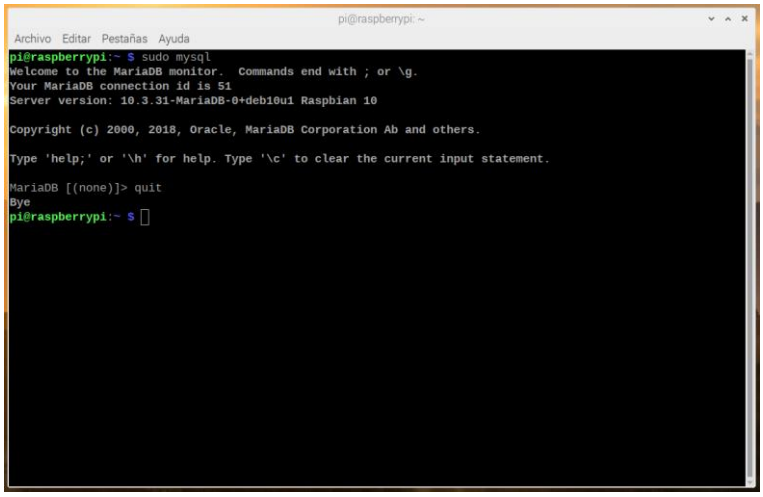


Ilustración 27: MySQL en funcionamiento

PHPMYADMIN

Lo ideal es gestionar nuestra BBDD mediante el navegador web, para ello usaremos phpmyadmin, vamos a instalarlo junto con gettext para facilitar la

traducción de programas (es obligatorio tener instalado previamente el servidor web Apache2, el intérprete de lenguaje de lado servidor PHP, y la base de datos mysql/MariaDB):

```
sudo apt install phpmyadmin php-gettext
```

Nos pedirá asociarlo a Apache2, aceptamos. Luego aceptamos configurar la asociación con la BBDD, y ponemos la contraseña para acceder a la misma (el usuario es **root** por defecto, como contraseña ponemos **raspberry**).

Al terminar crearemos dicho usuario para la BBDD, accedemos a la misma como root mediante:

```
sudo mariadb -u root -p
```

Nos pedirá la contraseña (ponemos raspberry).

Una vez dentro tecleamos (si todo va bien nos dirá Query OK):

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' IDENTIFIED BY 'raspberry';
```

Salimos de la BBDD mediante **quit**. Luego reiniciamos el Apache2.

```
sudo service apache2 restart
```

Luego probamos sencillamente poniendo en el navegador la url **localhost/phpmyadmin**.

Una vez lo tengas a la vista en el navegador, entra mediante el usuario **root** y la contraseña **raspberry**.

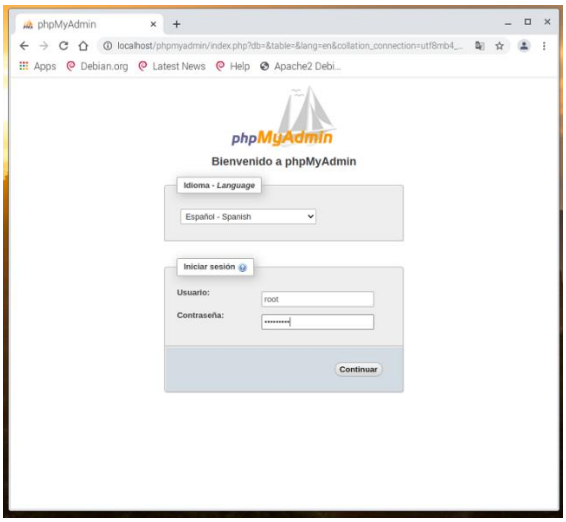


Ilustración 28: Login de phpMyAdmin

Una vez dentro, Este es el aspecto.

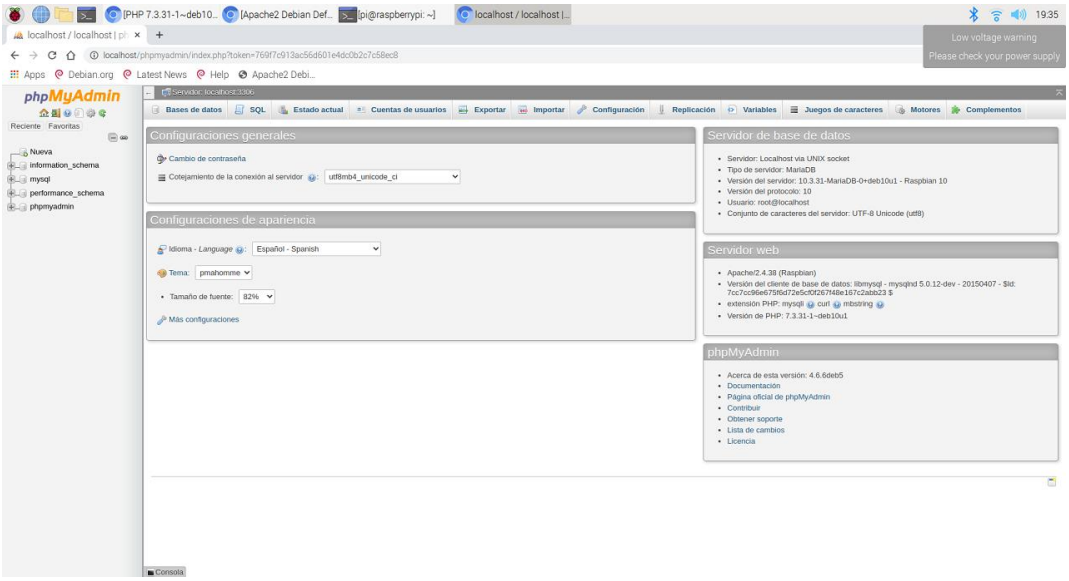


Ilustración 29: phpMyAdmin en funcionamiento

4.2.6 Configurar un DDNS (NO-IP)

Si deseamos usar el semáforo desde el exterior, tenemos el problema de que normalmente la IP pública exterior que nos proporciona nuestro ISP es dinámica. Por lo tanto, no podremos acceder al mismo de forma normal cuando nos la cambien, pues no sabremos cual es la nueva IP. Para solventar este problema, vamos a usar un DDNS (Dinamic Domain Name Server), concretamente el servicio NO-IP sito en https://www.noip.com

Para ello, hay que inscribirse en dicha web e instalar el cliente DUC en la RPi de manera que les mande la nueva IP cada vez que haya un cambio de la misma, de esta forma, podremos acceder desde el exterior con la URL http://xxx.ddns.net (siendo xxx un nombre elegido por nosotros) independientemente de la IP pública dinámica que nos asignen.

Antes que nada, nos inscribimos para usar el DDNS gratis, anotando los datos de acceso.

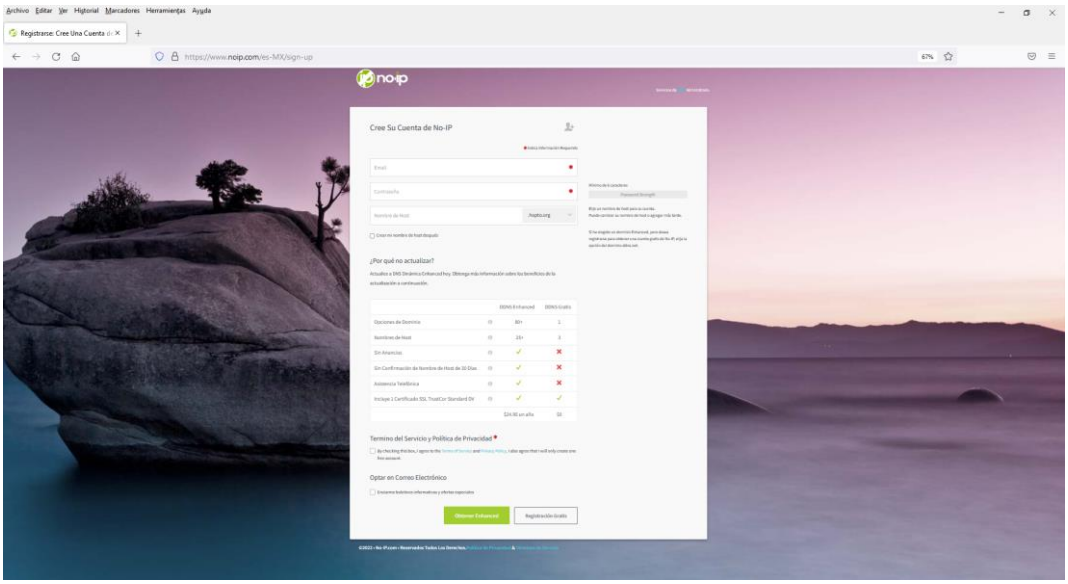


Ilustración 30: Registro free en NO-IP

Seguidamente entramos con esos datos del registro. Una vez dentro, creamos un hostname, esta URL será la que nos llevaría a nuestro index.html (Podemos saber nuestra IP pública actual entrando en https://www.cual-es-mi-ip.net/).

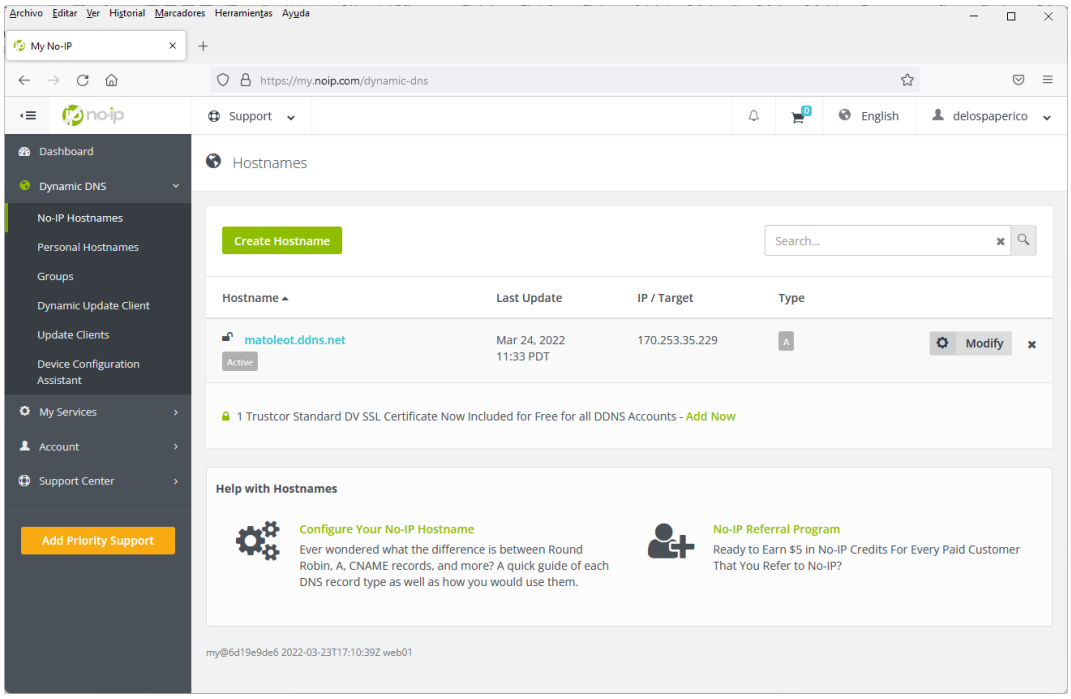


Ilustración 31: Hostname creado en NO-IP

NOTA: Al usar una cuenta gratuita, NO-IP nos mandará un email cada mes, el cual tendremos que confirmar su lectura con un simple clic para no perder el servicio.

Ahora instalamos el cliente DUC en la RPi, este software es un programa cuyo demonio manda al servidor de NO-IP nuestra nueva IP siempre que sea cambiada por nuestro ISP de manera dinámica.

```
cd /usr/local/src
sudo wget http://www.no-ip.com/client/linux/noip-duc-linux.tar.gz
sudo tar xzf noip-duc-linux.tar.gz
cd noip-2.1.9-1
sudo make
sudo make install
```

Durante la instalación se nos pedirá el correo usado como login y el password de la cuenta de NO-IP, además del tiempo de update (30 segundos por defecto) y si deseamos hace algo cuando sea exitosa a lo que responderemos que no. Al terminar habrá creado el archivo de configuración **/usr/local/etc/no-ip2.conf**

Seguidamente, hay que hacer que se ejecute cada vez que reiniciemos, para ellos creamos el archivo /etc/init.d/noip2 con la línea dentro sudo /usr/local/bin/noip2, para crearlo usamos el nano así:

```
sudo nano /etc/init.d/noip2
```

Añadimos con el nano la línea **sudo /usr/local/bin/noip2** y guardamos, seguidamente salimos y ejecutamos:

```
sudo chmod +x /etc/init.d/noip2
sudo update-rc.d noip2 defaults
```

NOTA: Para este proyecto, también requerimos tener instalado Python, en este punto no se ha efectuado tal instalación puesto que viene por defecto en sus tres versiones, podemos comprobarlo tecleando python, python2 o python3.

5. Implementación

►La implementación del código de este TFG está obligatoriamente ligado a la librería oficial del PiTraffic, disponible en (y reproducida en el anexo):

https://github.com/sbcshop/PiTraffic/blob/master/PiTraffic.py

Veamos el diagrama de clases de la librería pitraffic.py

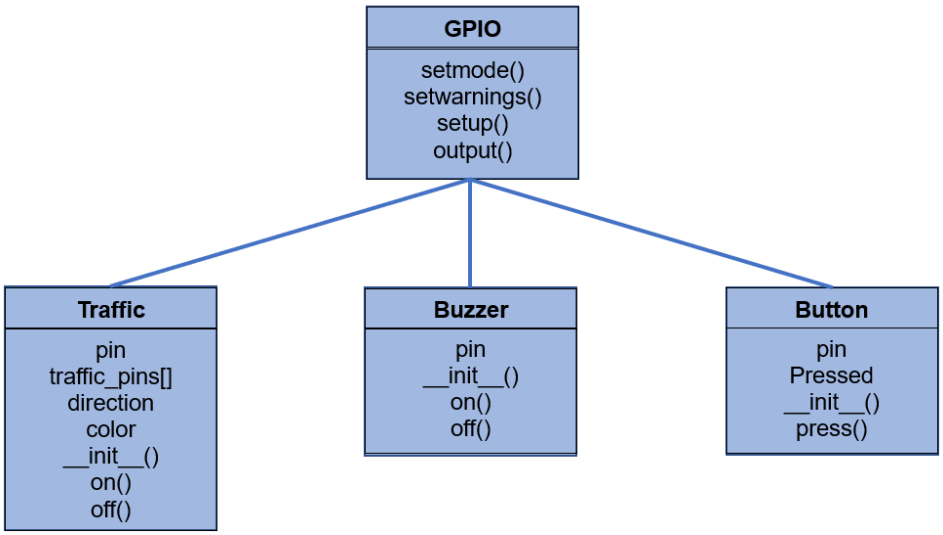


Ilustración 32: Diagrama de clases de PiTraffic.py

►La estructura de carpetas y ficheros del proyecto será la siguiente:

```
salatrafico
├── css
│   ├── index.css
│   ├── index.php
│   ├── login.css
│   └── salatrafico.css
├── img
│   ├── buzzer-icon.png
│   ├── direccion_prohibida.gif
│   ├── index.php
│   ├── pitraffic_amarillo.jpg
│   ├── pitraffic_fondo_1058-794.png
│   ├── pitraffic_gris.jpg
│   ├── pitraffic_logo.png
│   ├── pitraffic_rojo.jpg
│   ├── pitraffic_verde.jpg
│   ├── semaforo-interseccion_1920-1080.jpg
│   └── semaicon.png
├── index.html
├── js
│   ├── index.php
│   └── salatrafico.js
├── mp3
│   ├── index.php
│   └── zumbador_pitraffic.mp3
├── php
│   ├── button.dat
│   ├── cerrar_sesion.php
│   ├── control.php
│   ├── datos_conexion.inc.php
│   ├── index.php
│   ├── login.php
│   ├── presionado.dat
│   └── salatrafico.php
└── py
    ├── all_off.py
    ├── all_red_off.py
    ├── all_red_on.py
    ├── all_yellow_off.py
    ├── all_yellow_on.py
    ├── button_control_off.py
    ├── button_control_on.py
    ├── buzzer_10_on.py
    ├── buzzer_5_on.py
    ├── buzzer_off.py
    ├── buzzer_on.py
    ├── e_green_on.py
    ├── e_red_on.py
    ├── e_yellow_on.py
    ├── index.php
    ├── n_green_on.py
    ├── n_red_on.py
    ├── n_yellow_on.py
    ├── PiTraffic.py
    ├── PiTraffic.pyc
    ├── s_green_on.py
    ├── s_red_on.py
    ├── s_yellow_on.py
    ├── w_green_on.py
    ├── w_red_on.py
    └── w_yellow_on.py
```

Ilustración 33: Estructura arbórea del proyecto

►Veamos ahora el proceso seguido de escritura del código y las pruebas de funcionamiento. Por lo tanto, divido este punto nueve en dos subpuntos, la propia implementación (donde se explica los ítems más importantes que la afectan y las decisiones tomadas durante la misma) y las pruebas finales del proyecto, donde por lo tanto también se puede aprecia su aspecto en cuanto a estilos se refiere. Al final de esta PEC3 se presenta el código fuente completo en el punto 11, tanto el HTML, como los ficheros CSS/JS/PHP/PY.

5.1 Elaboración del código fuente

Este sistema "experimental" de control de un cruce semafórico de cuatro calles, posee la lógica en el lado cliente, aun siendo el servidor quien realmente enciende o apaga los semáforos usando Python, teniendo en cuenta que los mismos están conectados físicamente a él.

Para conseguir tal funcionalidad, el cliente manda órdenes vía AJAX al servidor (tanto en modo manual como en modo automático) usando JS, el servidor (la RPi) usando PHP recibe tales órdenes y las pasa por un switch para analizarlas y actuar en consecuencia, este PHP ejecuta mediante exec() un programa auxiliar escrito en Python que es quien realmente controla el semáforo real físicamente, teniendo en cuenta que como hardware de semáforo hemos usando el sombrero PiTraffic, cuya librería está escrita en dicho lenguaje, una vez efectuado el encendido/apagado del semáforo real deseado, el servidor contesta a esa orden asíncrona, de manera que el cliente al recibirla la pasa también por un switch para analizarla, y según su composición, enciende/apaga en la pantalla del mismo cliente lo mismo que se ha encendido/apagado en el servidor.

Para el control manual cada mando (sea radio o checkbox) manda directamente un comando vía AJAX, para el control automático se emplean los mismos comandos, pero se ha elaborado un carrusel de estados con un switch, el cual empleando un setTimeout va mandado comandos cada cierto tiempo.

Evidentemente el control de los semáforos debe realizarse de forma segura, no todo el mundo debería poder entrar al portal, por lo tanto, el acceso a la WEB de control se realiza por un proceso de autenticación usando una base de datos de usuarios MySQL.

5.1.1 Proceso login

La BDD *pitraffic.sql* empleada para la autenticación solamente necesita los campos *usuario* y *clave* (siendo usuario el campo *clave*), los cuales he puesto en la tabla *usuarios*, no obstante, he añadido el campo *privilegios* (que solo debería contener los valores *normal* o *administrador*) para posibles usos futuros. Ya hemos visto esta tabla en la página 11 de esta memoria.

El portal de acceso es **index.html**, el cual manda la información login a **login.php** usando un submit de formulario HTML como se ve seguidamente.

```
<form action="php/login.php" method="post">
        <span class="text-center">LOGIN</span>
```

```
		<div class="input-container">
			<input type="text" size="25" maxlenght="25" required="required" name="usuario" value="" />
			<label>Usuario</label>
		</div>
		<div class="input-container">
			<input type="password" size="25" maxlenght="25" required="required" name="clave" value="" />
			<label>Clave</label>
		</div>
		<button type="submit" class="btn">ENTRAR</button>
</form>
```

El **login.php** recibe estos datos y consulta la BDD, si las credenciales son correctas nos manda a la página de control **salatrafico.php**, mostrándonos un mensaje de error y retornándonos a **index.html** a los 7 segundos en caso contrario.

```
<?php
	include("datos_conexion.inc.php");

	$mysqlid = new mysqli($mysql_server, $mysql_login, $mysql_pass, 'pitraffic');
	$mysqlid -> set_charset("utf8");

	$sentencia = "SELECT usuario, clave FROM usuarios;";
	$registros = $mysqlid->query($sentencia);

	$permiso=false;
	if(isset($_POST['usuario']) and isset($_POST['clave']) and count($_POST)==2) {
		while($registro = $registros->fetch_object()){
			if($registro->usuario==$_POST['usuario'] && $registro->clave==$_POST['clave']){
				$permiso = true;
				$_SESSION['permiso']=true;
				$_SESSION['usuario']=$_POST['usuario'] ;

				header("Location: salatrafico.php");
				die();
			}
		}
	}

	if(!$permiso) {
		echo "<h1>ACCESO DENEGADO</h1>";
		echo "<div id='wrapperprohibido'>";
			echo "<img src='../img/direccion_prohibida.gif' id='prohibido' alt='Prohibido' />";
		echo "</div>";
		echo "<script language='javascript'>setTimeout(redirigir_index,7000);</script>";
	}
?>
```

Como es lógico, la sesión se propaga mediante session_start().

5.1.2 Protección de acceso con el navegador a las subcarpetas del proyecto

Si mediante el navegador intentamos entrar a una de las carpetas del proyecto dentro de la principal, se verían los archivos allí alojados pudiéndose abrir y/o descargar, lo que consideraría una falta de profesionalidad, por ello, he elaborado un archivo llamado **index.php** y he situado una copia del mismo en cada una de las subcarpetas menos en la principal, este PHP nos lleva a la WEB principal en caso de alguien intente entrar directamente con el navegador de dicho modo a esas carpetas. Veamos el código.

```
<html>
	<head>
		<title>Redirección</title>
	</head>
	<body>
		<?php
			header("Location: http://matoleot.ddns.net/salatrafico");
			die();
		?>
	</body>
</html>
```

5.1.3 Modo manual (Man)

Veamos el proceso completo de encendido de la luz roja del semáforo sur. Si el proceso login ha sido exitoso estaremos en **salatrafico.php** y en modo manual por defecto, el HTML para el control manual del semáforo sur es el siguiente (extracto de salatrafico.php).

```
<tr><td>
	<div>
		<span class='semaforonseo'>Sem&aacute;foro Sur</span>
	</div>
	<div id='sur'  class='container' >
		<form class='form' >
			<input type='radio' class='radiosem' name='sur' id='surr' value='s_red_on' /><label class='four col' for='surr'>Rojo</label>
			<input type='radio' class='radiosem' name='sur' id='sura' value='s_yellow_on' /><label class='four col' for='sura'>Amarillo</label>
			<input type='radio' class='radiosem' name='sur' id='surv' value='s_green_on' /><label class='four col' for='surv'>Verde</label>
		</form>
	</div>
</td></tr>
```

Al ser radios con el mismo nombre solamente uno puede estar activado (o ninguno), si observamos el correspondiente al sur, su atributo value es **s_red_on**, que no es más que el comando AJAX que se va a mandar, cuando se mande realmente mandará "**comando=s_red_on"**.

Si nos fijamos no hay eventos en el HTML, los doce radios de control manual de los cuatro semáforos pertenecen a la clase **radiosem**, y dicha clase se emplea en **salatrafico.js** (importado en salatrafico.php) para asignarles un listerner que llama a la función validar() en caso de ser accionados. Veamos dichos listeners.

```
radiosems=document.getElementsByClassName('radiosem');
for(var i=0; i<radiosems.length; i++){
	radiosems[i].addEventListener("click", validar);
}
```

Dicha función validar(), primero comprueba que el mando man/aut este situado en manual y si no es así da error vía un alert(), si es así manda el comando **s_red_on** vía AJAX usando el método llamadaAjax().

```
function validar(cmd){ //manda comando si manual
	if(document.getElementById('automan').checked || document.getElementById('echeck').checked){
		resetear(false);
		alert("No se puede usar este control estando en\nmodo automático o emergencia (Aut/Eme). \n");
	}else{
		if(cmd=="buzzer_on" || cmd=="buzzer_off"){
			llamadaAjax(cmd);
		}else{
			llamadaAjax(cmd.target.value);
		}
	}
}
```

El método llamadaAjax() hace una llamada AJAX normal mediante el método POST y mandado el comando al programa de control central de los semáforos **control.php**, cuando el AJAX reciba la respuesta llamará al callback analizaRespuestaAjax().

```
function llamadaAjax(cmd){ //servicio asíncrono para mandar comandos al servidor
	let xhr = new XMLHttpRequest();
	xhr.onreadystatechange = function() {
		if (this.readyState == 4 && this.status == 200) {
```

```
                        analizaRespuestaAjax(this.responseText);
                }
        };
        xhr.open('POST', 'control.php');
        xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
        xhr.send("comando=" + cmd);

        console.log("---> Mandado: comando=" + cmd);
}
```

Como se puede observar, se usa la consola para ver un log con el comando mandado, para identificar este tipo de log se usan los caracteres **--->**, este proyecto tiene numerosos log`s implementados, los cuales se recomienda observar cuando se pruebe.

El programa de control central de los semáforos **control.php**, recibe tal comando y tras pasarlo por un switch actúa en consecuencia, veamos un extracto de dicho código.

```
<?php
        switch($_REQUEST[comando]){
                # todo off
                case "all_off":         exec("sudo python ../py/all_off.py");
                                        echo "all_off_OK";
                                        exec("sudo python ../py/button_control_off.py");
                                        break;
                # N=NORTH (NORTE)
                case "n_red_on":        exec("sudo python ../py/n_red_on.py");          echo "n_red_on_OK";
                break;
                case "n_yellow_on":     exec("sudo python ../py/n_yellow_on.py");       echo "n_yellow_on_OK";
                        break;
                case "n_green_on":      exec("sudo python ../py/n_green_on.py");        echo "n_green_on_OK";
                        break;
                # S=SOUTH (SUR)
                case "s_red_on":        exec("sudo python ../py/s_red_on.py");          echo "s_red_on_OK";
                break;
                case "s_yellow_on":     exec("sudo python ../py/s_yellow_on.py");       echo "s_yellow_on_OK";
                        break;
                case "s_green_on":      exec("sudo python ../py/s_green_on.py");        echo "s_green_on_OK";
                        break;
                # buzzer
                case "buzzer_5_on":     exec("sudo python ../py/buzzer_5_on.py");       echo "buzzer_5_on_OK";
                        break;
                case "buzzer_on":       exec("sudo python ../py/buzzer_on.py");         echo "buzzer_on_OK";
                break;
                case "buzzer_off":      exec("sudo python ../py/buzzer_off.py");        echo "buzzer_off_OK";
                break;
                # button
                case "button_setup":    exec("sudo python ../py/button_control_on.py");   echo "button_setup_OK";
                        break;
                # contestación por defecto
                default:                print "ERROR_comando_NO_reconocido";
                                        break;
        }
?>
```

Como se puede apreciar, si recibe el comando **s_red_on** ejecuta el código Python **s_red_on.py** y contesta al AJAX mediante la respuesta **s_red_on_OK**.

Dicho código **s_red_on.py** tiene el siguiente aspecto.

```
import PiTraffic

# Bloque declaración constantes
SouthRed = PiTraffic.Traffic("SOUTH", "RED")
SouthYellow = PiTraffic.Traffic("SOUTH", "YELLOW")
SouthGreen = PiTraffic.Traffic("SOUTH", "GREEN")

# Bloque funciones
def South_Yellow_Off():
   SouthYellow.off()
def South_Green_Off():
   SouthGreen.off()
def South_Red_On():
   SouthRed.on()
```

```
# Bloque acción
South_Yellow_Off()
South_Green_Off()
South_Red_On()
```

Con el anterior código, ejecutado desde el PHP mediante exec(), se apagan físicamente del semáforo sur las lámparas amarilla y verde, encendiendo la roja. Para ello, importa y emplea la librería **PiTraffic.py** proporcionada por el fabricante (reproducida en página 55 de esta memoria).

Cuando el **salatrafico.js** recibe la respuesta a su llamada AJAX, en este ejemplo **s_red_on_OK** se pasa por un switch que actúa en consecuencia, veamos un extracto de dicho código.

```
function analizaRespuestaAjax(msg){ //analiza comandos recibidos y actúa en consecuencia
        console.log("<--- Recibido mensaje: " + msg);

        switch(msg){ //analiza respuesta y pinta semáforos canvas
        case "button_all_red_on_OK":    //todos rojo button on
                x_rojo_on(ctxN);
                x_rojo_on(ctxS);
                x_rojo_on(ctxE);
                x_rojo_on(ctxW);
                console.log("*** Button presionado, encendiendo semáforos en rojo para que pase el peatón ***");
                ciclo=0; // pone el ciclo especial de todo en rojo por button presionado
                break;
        case "n_red_on_OK":             //norte rojo on
                x_rojo_on(ctxN);
                console.log("Encendido semáforo norte rojo");
                break;
        case "n_yellow_on_OK":          //norte amarillo on
                x_amarillo_on(ctxN);
                console.log("Encendido semáforo norte amarillo");
                break;
        case "n_green_on_OK":           //norte verde on
                x_verde_on(ctxN);
                console.log("Encendido semáforo norte verde");
                break;
        case "s_red_on_OK":             //sur rojo on
                x_rojo_on(ctxS);
                console.log("Encendido semáforo sur rojo");
                break;
        case "s_yellow_on_OK":          //sur amarillo on
                x_amarillo_on(ctxS);
                console.log("Encendido semáforo sur amarillo");
                break;
        case "buzzer_on_OK":            //sonido manual on
                document.getElementById('ibuzz').style.visibility="visible";            //visualiza buzzer (suena en el servidor)
                new Audio("../mp3/zumbador_pitraffic.mp3").play();      //suena sonido en el cliente
                console.log("Buzzer manual sonando");
                break;
        case "buzzer_off_OK":           //sonido manual off
                document.getElementById('ibuzz').style.visibility="hidden";             //invisibiliza buzzer (ya no suena)
                console.log("Buzzer manual apagado");
                break;
        }
}
```

Se puede observar, como si recibe el comentado comando ejecuta **x_rojo_on(ctxS);**, este método -o su nombre- lo considero ‘curioso’, la “x” inicial significa que vale para cualquier semáforo sea norte/sur/este/oeste, lo que recibe como parámetro es el contexto del **canvas** del semáforo deseado a pintar en pantalla, en este caso se manda **ctxS**, este contexto del canvas es una variable global situada al principio del salatrafico.js y la “S” final significa que es el semáforo sur el que se desea controlar.

Además, apreciamos como se usa la consola para ver un log con el mensaje recibido, para identificar este tipo de log se usan los caracteres **<---**,

este proyecto tiene numerosos log`s implementados, los cuales se recomienda observar cuando se pruebe.

Veamos cómo se obtiene el canvas del semáforo sur (extracto de salatrafico.js).

```
canvasS = document.getElementById('semS');
ctxS= canvasS.getContext('2d');
```

Veamos el método **x_rojo_on()** (extracto de salatrafico.js).

```
function x_rojo_on(ctx){                //x rojo on
        ctx.fillStyle="rgb(100, 100,100)";
        ctx.fillRect(0, 0, 50, 140);
        //rojo on
        ctx.beginPath();
        ctx.fillStyle="rgb(255, 0, 0)";
        ctx.arc(25, 25, 20, 0, 2 * Math.PI);
        ctx.fill();
        //amarillo off
        ctx.beginPath();
        ctx.strokeStyle="rgb(255, 255, 255)";
        ctx.arc(25, 70, 20, 0, 2 * Math.PI);
        ctx.stroke();
        //verde off
        ctx.beginPath();
        ctx.strokeStyle="rgb(255, 255, 255)";
        ctx.arc(25, 115, 20, 0, 2 * Math.PI);
        ctx.stroke();
}
```

Con esto, ha quedado explicado el proceso completo de control manual, desde que se pide una orden presionando un radio, se manda un comando, se ejecuta el control deseado en el servidor haciendo el control real, se contesta a dicha petición, y para finalizar, el cliente 'pinta' en su pantalla el semáforo correspondiente al recibir tal respuesta. Todo esto significa que si en el cliente está encendida/apagada una lámpara en la pantalla lo estará también en la realidad físicamente en el servidor.

5.1.4 Modo automático (Aut)

En **salatrafico.php** el primer control es un checkbox llamado Aut/Man que nos permite poner los modos manual o automático, estando en manual por defecto al entrar, el cual ya se ha explicado en el punto anterior.

El modo manual funciona igual que el automático en cuanto a los comandos se refiere, incluso usan los mismos switch de los que ya hemos mostrado un extracto tanto del lado cliente (JS) como del servidor (PHP). En modo automático lo que cambia es que en este modo no se permite pulsar en los radio/checkbox destinados al control manual dando una alerta en tal caso, y se pone en marcha una "máquina de estados" creada con otro switch que va cambiado de estado a cada ejecución de un setTimeout(función, tiempo), es decir, se va llamando a una función cada cierto tiempo que lo que hace es cambiar de estado, y este, ejecuta las acciones correspondientes a dicho estado, hay un total 9 estados. Los estados implementados ya fueron comentados en la página 10 de la presente memoria.

Veamos el código de dicho control o checkbox llamado Aut/Man.

```
<div class='container1'>
        Man
```

```
		<label class='switch'>
		<input type='checkbox' id='automan' onchange='setTimeout(() => { validar_automan(this.checked); }, 200); ' />
		<div class='slider round'></div>
		</label>
		Aut
</div>
```

Si ‘desplazamos’ dicho control hacia la derecha ponemos el modo automático a los 0,2 segundos debido a que tras ese tiempo llamamos a validar_automan(), el retardo se ha puesto para que no de problemas si se intentaba pasar de manual a automático -o al contrario- rápidamente. Dicha función tiene el siguiente aspecto.

```
function validar_automan(estado){ //a true si automático
	if(estado){
		if(document.getElementById('echeck').checked){
			document.getElementById('automan').checked =false;
			alert("No puede usar esto estando en\nmodo emergencia (Eme), desáctivelo primero. \n");
		}else{
			llamadaAjax("auto_on");
		}
	}else{
		llamadaAjax("manual_on");
	}
}
```

Como se puede apreciar, tanto para poner el modo manual como el automático se manda un comando AJAX (**auto_on** para automático y **manual_on** para manual), en la práctica este comando lo único que hace en el servidor es borrar el semáforo para comenzar desde cero y contesta, luego el cliente hace lo mismo en la pantalla al recibir dicha respuesta para comenzar el modo deseado desde cero. Veamos un extracto de **salatrafico.js** para ver lo que hace el cliente al recibir dicha respuesta (en este caso se ha quitado del switch).

```
if(msg=="manual_on_OK"){resetear(false);}
if(msg=="all_off_OK" || msg=="manual_on_OK") {
	tauto_off();
	todo_off();
}
if(msg=="auto_on_OK") tauto_on();
```

Se puede apreciar como si hemos puesto el modo manual al recibir la respuesta de que en el servidor está el semáforo todo apagado se llama a resetear() que reinicia todas la variables globales y desactiva o pone por defecto todos los radio, checkbox y select del cliente, se llama a tauto_off() que quita el modo automático por medio de resetear el setTimeout mediante window.clearTimeout(id_automatico), y para terminar se llama también a todo_off() que resetea todos los canvas de la pantalla del cliente, es decir, apaga los semáforos en pantalla.

Y si hemos puesto el modo automático se llama a tauto_on(), que activa el modo automático por medio de poner el primer ciclo iniciando la variable ciclo=1 para que comience por primer ciclo y llamado a auto_roll(). Este último método es la máquina de estados, que es llamada tras cada tiempo por un setTimeout(auto_roll, mTiempo), esta máquina de estados tiene el siguiente aspecto (el ciclo 0 es especial como se verá posteriormente, el modo automático normalmente comienza por el 1 y termina en el 9, recomenzando otra vez en el 1, saltándose el estado 0 que está reservado para gestionar el Button o pulsador situado físicamente en el PiTraffic).

```
function auto_roll(){     //rueda automática
        console.log("Pasando por el ciclo: " + ciclo);
        switch(ciclo){
                case 0:     //Button presionado. Todo en rojo
                        mTiempo=15000;
                        ciclo=1;
                        break;
                case 1:     //norte y sur en verde
                        llamadaAjax("n_green_on");
                        llamadaAjax("s_green_on");
                        llamadaAjax("e_red_on");
                        llamadaAjax("w_red_on");
                        mTiempo=mTiempoRV;
                        ciclo=2;
                        break;
                case 2:     //norte y sur en amarillo
                        llamadaAjax("n_yellow_on");
                        llamadaAjax("s_yellow_on");
                        llamadaAjax("e_red_on");
                        llamadaAjax("w_red_on");
                        if(!mute){
                                sonido_temporal();                  //suena y se visualiza en el cliente
                        }
                        mTiempo=mTiempoA;
                        mTiempoTR != 0 ?  ciclo=3 : ciclo=4
                        break;
                case 3:     //todo rojo (opcional según TR)
                        llamadaAjax("all_red_on");
                        mTiempo=mTiempoTR;
                        ciclo=4;
                        break;
                case 4:     //este en verde
                        llamadaAjax("n_red_on");
                        llamadaAjax("s_red_on");
                        llamadaAjax("e_green_on");
                        llamadaAjax("w_red_on");
                        prioridad==true ? mTiempo=mTiempoRV*2 : mTiempo=mTiempoRV;
                        ciclo=5;
                        break;
                case 5:     //este amarillo
                        llamadaAjax("n_red_on");
                        llamadaAjax("s_red_on");
                        llamadaAjax("e_yellow_on");
                        llamadaAjax("w_red_on");
                        if(!mute){
                                sonido_temporal();                  //suena y se visualiza en el cliente
                        }
                        mTiempo=mTiempoA;
                        mTiempoTR != 0 ?  ciclo=6 : ciclo=7         //a 0 se salta todo en rojo (TR)
                        break;
                case 6:     //todo rojo (opcional según TR)
                        llamadaAjax("all_red_on");
                        mTiempo=mTiempoTR;
                        ciclo=7;
                        break;
                case 7:     //oeste en verde
                        llamadaAjax("n_red_on");
                        llamadaAjax("s_red_on");
                        llamadaAjax("e_red_on");
                        llamadaAjax("w_green_on");
                        prioridad==true ? mTiempo=mTiempoRV*2 : mTiempo=mTiempoRV;
                        ciclo=8;
                        break;
                case 8:     //oeste amarillo
                        llamadaAjax("n_red_on");
                        llamadaAjax("s_red_on");
                        llamadaAjax("e_red_on");
                        llamadaAjax("w_yellow_on");
                        if(!mute){
                                sonido_temporal();                  //suena y se visualiza en el cliente
                        }
                        mTiempo=mTiempoA;
                        mTiempoTR != 0 ?  ciclo=9 : ciclo=1         //a 0 se salta todo en rojo (TR)
                        break;
                case 9:     //todo rojo (opcional según TR)
                        llamadaAjax("all_red_on");
                        mTiempo=mTiempoTR;
                        ciclo=1;
                        break;
        }

        id_automatico = window.setTimeout(auto_roll, mTiempo);      //espera siguiente estado de automático
}
```

Como se puede apreciar, tras cada ejecución del setTimeot se activa un estado o ciclo en el switch entrando cíclicamente a sus ramas, y cada rama de dicho switch tiene una llamada o llamadas AJAX necesarias para el control de ese estado deseado.

Veamos también el código del ya anteriormente nombrado todo_off(), que reinicia los canvas del cliente para que se vean todos los semáforos apagados en su pantalla.

```
function todo_off(){      //todos semáforos apagados en pantalla cliente
            //inicio canvas semáforo norte
            canvasN.style.borderRadius = "15px";
            ctxN.fillStyle="rgb(100, 100,100)";
      ctxN.fillRect(0, 0, 50, 140);
            //norte rojo off
            ctxN.beginPath();
            ctxN.strokeStyle="rgb(255, 255, 255)";
            ctxN.arc(25, 25, 20, 0, 2 * Math.PI);
            ctxN.stroke();
            //norte amarillo off
            ctxN.beginPath();
            ctxN.strokeStyle="rgb(255, 255, 255)";
            ctxN.arc(25, 70, 20, 0, 2 * Math.PI);
            ctxN.stroke();
            //norte verde off
            ctxN.beginPath();
            ctxN.strokeStyle="rgb(255, 255, 255)";
            ctxN.arc(25, 115, 20, 0, 2 * Math.PI);
            ctxN.stroke();

            //inicio canvas semáforo sur
            canvasS.style.borderRadius = "15px";
            ctxS.fillStyle="rgb(100, 100,100)";
      ctxS.fillRect(0, 0, 50, 140);
            //sur rojo off
            ctxS.beginPath();
            ctxS.strokeStyle="rgb(255, 255, 255)";
            ctxS.arc(25, 25, 20, 0, 2 * Math.PI);
            ctxS.stroke();
            //sur amarillo off
            ctxS.beginPath();
            ctxS.strokeStyle="rgb(255, 255, 255)";
            ctxS.arc(25, 70, 20, 0, 2 * Math.PI);
            ctxS.stroke();
            //sur verde off
            ctxS.beginPath();
            ctxS.strokeStyle="rgb(255, 255, 255)";
            ctxS.arc(25, 115, 20, 0, 2 * Math.PI);
            ctxS.stroke();

            //inicio canvas semáforo este
            canvasE.style.borderRadius = "15px";
            ctxE.fillStyle="rgb(100, 100,100)";
      ctxE.fillRect(0, 0, 50, 140);
            //este rojo off
            ctxE.beginPath();
            ctxE.strokeStyle="rgb(255, 255, 255)";
            ctxE.arc(25, 25, 20, 0, 2 * Math.PI);
            ctxE.stroke();
            //este amarillo off
            ctxE.beginPath();
            ctxE.strokeStyle="rgb(255, 255, 255)";
            ctxE.arc(25, 70, 20, 0, 2 * Math.PI);
            ctxE.stroke();
            //este verde off
            ctxE.beginPath();
            ctxE.strokeStyle="rgb(255, 255, 255)";
            ctxE.arc(25, 115, 20, 0, 2 * Math.PI);
            ctxE.stroke();

            //inicio canvas semáforo oeste
            canvasW.style.borderRadius = "15px";
            ctxW.fillStyle="rgb(100, 100,100)";
      ctxW.fillRect(0, 0, 50, 140);
            //oeste rojo off
            ctxW.beginPath();
            ctxW.strokeStyle="rgb(255, 255, 255)";
            ctxW.arc(25, 25, 20, 0, 2 * Math.PI);
            ctxW.stroke();
            //oeste amarillo off
```

```
        ctxW.beginPath();
        ctxW.strokeStyle="rgb(255, 255, 255)";
        ctxW.arc(25, 70, 20, 0, 2 * Math.PI);
        ctxW.stroke();
        //oeste verde off
        ctxW.beginPath();
        ctxW.strokeStyle="rgb(255, 255, 255)";
        ctxW.arc(25, 115, 20, 0, 2 * Math.PI);
        ctxW.stroke();
}
```

Para el modo automático disponemos de tres select/option que nos permiten cambiar los tiempos. El select R/V nos permite cambiar el tiempo que está cada semáforo en Rojo o en Verde, este tiempo tiene que ser obligatoriamente el mismo para cada par de semáforos de una misma calle norte/sur o este/oeste, es una restricción que no se puede saltar pues cada calle depende de la otra, no obstante, existe un checkbox llamado Pri (Prioridad) que hace multiplicar por dos ese tiempo para la calle este/oeste dándole prioridad. El select A nos permite cambiar el tiempo que está cada semáforo en amarillo. El select TR nos permite cambiar el tiempo en el que están todos los semáforos en rojo antes de pasar alguno de ellos a Verde (por seguridad). Por defecto lo tiempos son 5seg, 3 seg y 0 seg (que significa TR apagado) respectivamente, son bajos para que se pueda apreciar o probar cómo funciona el proyecto sin que necesitemos mucho tiempo. El funcionamiento de estos select/option es muy sencillo, si accionamos cualquiera de ellos se modifica una variable global que luego tiene en cuenta la máquina de estados auto_roll() anterior, las variables son mTiempo, mTiempoRV, mTiempoA y mTiempoTR que podemos observar en dicho método ya reproducido más arriba.

5.1.5 Modo Emergencia (Eme)

Al pie de la página web existe el checkbox Eme (Emergencia), el cual solamente se puede activar si estamos en modo manual. Este control activa un modo que consiste en que parpadeen todas las luces amarillas, haciendo sonar el buzzer como veremos en el siguiente punto, este modo es ideal para casos especiales, como días u horas sin tráfico, mantenimiento de la calzada etc. El funcionamiento es una máquina de estados similar al modo automático, pero con solo dos estados. Aunque hay más código implicado en el modo Eme, veamos el aspecto del código de esta máquina de dos estados.

```
function emergencia_roll(){         //rueda emergencia
        switch(ciclo){
                case 1:     //todo amarillo
                            llamadaAjax("all_yellow_on");
                            if(!mute){
                                    sonido_temporal();      //suena y se visualiza en el cliente
                            }
                            mTiempo=2000;
                            ciclo=2;
                            break;
                case 2:     //todo apagado
                            llamadaAjax("all_yellow_off");
                            mTiempo=1000;
                            ciclo=1;
                            break;
        }

        id_emergencia = window.setTimeout(emergencia_roll, mTiempo);    //espera siguiente estado
}
```

Para apagar este modo la función emergencia_off() ejecuta window.clearTimeout(id_emergencia).

5.1.6 Buzzer (Buz) y mute (Mut)

El escudo PiTraffic dispone de Buzzer (zumbador), que en este TFG se emplea para avisar sonoramente tan solo un instante (y de manera opcional) de que algún semáforo está en amarillo, al mismo tiempo, si se manda al servidor que suene el buzzer, se muestra en la pantalla del cliente una imagen de una alarma para que sepamos que está sonando físicamente en el semáforo real, y encima suena también en los altavoces del cliente (si están conectados) empleando la etiqueta <audio>. El zumbador también se puede hacer sonar indefinidamente en manual si estamos en dicho modo (checkbox **Buz**). Si está en automático se puede poner el modo mute (**Mut**), en ese caso no suena ni en el servidor ni en el cliente, ni se muestra la imagen de la alarma, el modo mute funciona sencillamente poniendo la variable global mute a true o false.

Los comandos que recibe **control.php** para hacer sonar el buzzer un instante en automático o indefinidamente en manual son: buzzer_5_on, buzzer_on y buzzer_off.

Ya ha aparecido en códigos anteriores algunas líneas de código referente al buzzer. Por ejemplo, el comando buzzer_5_on activaría en el servidor buzzer_5_on.py, que tiene el siguiente aspecto.

```
import PiTraffic
import time

# Bloque declaración constantes
Buzz = PiTraffic.Buzzer()

# Bloque acción
Buzz.on()
time.sleep(0.2)
Buzz.off()
time.sleep(0.2)
Buzz.on()
time.sleep(0.2)
Buzz.off()
time.sleep(0.2)
Buzz.on()
time.sleep(0.2)
Buzz.off()
time.sleep(0.2)
Buzz.on()
time.sleep(0.2)
Buzz.off()
time.sleep(0.2)
Buzz.on()
time.sleep(0.2)
Buzz.off()
```

Para mandar al servidor sonar el buzzer temporalmente, y accionar en el cliente el zumbador para que se oiga en los altavoces, y además muestre la imagen de la sirena en pantalla, vamos a analizar el método sonido_temporal().

```
function sonido_temporal(){          //manda sonar buzzer temporal en servidor, visualiza y suena buzzer cliente
        llamadaAjax("buzzer_5_on");        //manda sonar zumbador en el servidor
        document.getElementById('ibuzz').style.visibility="visible";                //visualiza buzzer en pantalla ciente
        new Audio("../mp3/zumbador_pitraffic.mp3").play();          //suena sonido en el cliente
        setTimeout(() => {document.getElementById('ibuzz').style.visibility="hidden";}, 1750); //al tiempo invisibiliza
        console.log("Activando Buzzer temporalmente");
}
```

En este caso, al usar el Python sleep, NO esperamos a la respuesta del servidor para hacer sonar y mostrar en la pantalla del cliente la alarma, pues esa respuesta llegaría al terminar el Python, y por lo tanto, ya seria tarde (ya estaría el buzzer real sin sonar).

5.1.7 Veamos cómo se gestiona el button del PiTraffic

Para terminar, el escudo PiTraffic dispone de un Button (un pulsador físico) situado en la misma placa junto a los cuatro semáforos que es coherente darle funcionalidad. Teniendo en cuenta de que en muchos semáforos reales existe un pulsador, que pone todos los semáforos en rojo para que pasen peatones, para este proyecto he implementado la misma funcionalidad, haciendo que si estando en automático un peatón pulsa dicho pulsador se pondrán los cuatro semáforos en rojo 15 segundos.

No ha sido fácil en base a los siguiente (por eso el código es rebuscado). Este es un proyecto del “control experimental” de semáforos, donde el control se efectúa desde el cliente, es el cliente quien manda órdenes al servidor siendo este “pasivo”, confiando en que la comunicación será continua y con unas latencias muy bajas. Sin embargo, el pulsador está en el servidor, quien no puede mandar un mensaje al cliente a no ser que sea como respuesta a una llamada explícita AJAX, y además parece que no tenga ningún sistema de control por interrupción. Enseguida podemos pensar en mandar desde el cliente cada muy poco tiempo un comando al servidor que le obligue a chequear el pulsador y si está pulsado poner todo el rojo, pero tantas llamadas AJAX en tan poco tiempo hace que el sistema no funcione correctamente. La solución adoptada ha sido la siguiente:

Como solo puede funcionar el modo automático cuando pasamos a tal modo el servidor ejecuta **button_control_on.py**, este programa Python escribe un “1” en el archivo button.dat y entra en un buble infinito, configurando lo que se conoce como una entrada/salida programada (polling), que no es más que testear periódicamente si el pulsador está activado, cuando se da cuenta de que esta activado escribe otro “1” en presionado.dat que vale como chivato de tal evento. Este bucle o polling se puede romper si en el archivo button.dat escribimos un “0” (veremos seguidamente como hacerlo). Veamos este código **button_control_on.py**.

```
import PiTraffic
import time

# Bloque declaración constantes
Buzz = PiTraffic.Buzzer()
Btn=PiTraffic.Button()

# Bloque métodos
def sonar(tim):
        Buzz.on()
        time.sleep(tim)
        Buzz.off()

# Bloque acción
file1=open("presionado.dat", "w")
file1.write("0")
file1.close()
file2=open("button.dat", "w")
file2.write("1")
file2.close()
```

```
while(True):
        time.sleep(8)
        file3=open("button.dat", "r")
        dato=file3.read(1)
        file3.close()
        if(dato=='0'):
                break;
        Btn.press()
        if(Btn.Pressed==True):
                file4=open("presionado.dat", "w")
                file4.write("1")
                file4.close()
                sonar(3)
```

Luego, en **control.php** (el receptor de todos los comandos AJAX), antes del switch que discrimina el comando concreto recibido, chequeamos si el chivato de que se ha pulsado el pulsador está a “1”, leyendo presionado.dat, si está a “1” ejecuta el programa Python **all_red_on.py** para encender físicamente todos los semáforos en rojo y contesta al cliente con **button_all_red_on_OK** para que el mismo ponga en pantalla los semáforos también en rojo por medio de los canvas, además, el cliente al recibir tal mensaje pone el ciclo especial 0 (recordemos que el carrusel automático iba del ciclo 1 al 9, repasar dicho código más arriba), este ciclo tiene un tiempo especial de 15 segundos y tras él siempre continúa por el ciclo 1 (es decir estemos en el ciclo que estemos luego de gestionar el button siempre pasamos al ciclo 1). Veamos un extracto del principio de **control.php** donde se aprecia como si presionado.dat tiene un “1” hace las gestiones y lo pone en “0” de nuevo para resetearlo (como curiosidad se hace notar que este fichero se lee y escribe tanto desde el Python como desde el PHP, comunicando dichos lenguajes o archivos).

```
$fp=fopen("presionado.dat", "r");
$dato=fgets($fp);
fclose($fp);
if($dato=="1"){
        exec("sudo python ../py/all_red_on.py");        echo "button_all_red_on_OK";
        $fp=fopen("presionado.dat", "w");
        fwrite($fp, "0");
        fclose($fp);
        exit();
}
```

El programa **button_control_off.py** lo único que hacer es poner a “0” de nuevo ambos archivos *.dat, de manera que el pulsador ya no funciona, este programa se ejecuta al pasar al modo manual.

NOTA: Para que funcione el anterior código de la gestión del pulsador en el presente proyecto, he tenido que modificar primero la clase Button de la librería **PiTraffic.py** (ver completa en la página 55 de esta memoria) proporcionada por el fabricante. El motivo ha sido que el atributo Pressed, una vez activo ya no se reseteaba a False hasta reiniciar, es decir, que el pulsador solamente funcionaba una vez, vemos seguidamente en negrita la modificación o ampliación.

```
class Button:
   Pressed = False

   def __init__(self):
      self.pin = 7
      GPIO.setup(self.pin,GPIO.IN, pull_up_down=GPIO.PUD_UP)

   def press(self):
      input_state = GPIO.input(self.pin)
```

```
if input_state == False:
    print("Button - Pressed")
    self.Pressed = True
else: # Modificado por MaToLeOt
    print("Button - Unpressed")
    self.Pressed = False
```

5.2 Comandos AJAX, ejecución Python y respuestas

Se muestra a continuación la lista que comandos AJAX que puede mandar el cliente, junto con el código Python que ejecuta el servidor al recibirlos, y la respuesta al mismo. Se recomienda verlos en los log's de la consola mientras se prueba el proyecto.

TABLA DE COMANDOS AJAX		
COMANDO	**EJECUCIÓN**	**RESPUESTA**
all_off	all_off.py y button_control_off.py	all_off_OK
manual_on	all_off.py y button_control_off.py	manual_on_OK
auto_on	all_off.py	auto_on_OK
n_red_on	n_red_on.py	n_red_on_OK
n_yellow_on	n_yellow_on.py	n_yellow_on_OK
n_green_on	n_green_on.py	n_green_on_OK
s_red_on	s_red_on.py	s_red_on_OK
s_yellow_on	s_yellow_on.py	s_yellow_on_OK
s_green_on	s_green_on.py	s_green_on_OK
e_red_on	e_red_on.py	e_red_on_OK
e_yellow_on	e_yellow_on.py	e_yellow_on_OK
e_green_on	e_green_on.py	e_green_on_OK
w_red_on	w_red_on.py	w_red_on_OK
w_yellow_on	w_yellow_on.py	w_yellow_on_OK
w_green_on	w_green_on.py	w_green_on_OK
all_red_on	all_red_on.py	all_red_on_OK
all_red_off	all_red_off.py	all_red_off_OK
all_yellow_on	all_yellow_on.py	all_yellow_on_OK
all_yellow_off	all_yellow_off.py	all_yellow_off_OK
buzzer_5_on	buzzer_5_on.py	buzzer_5_on_OK
buzzer_on	buzzer_on.py	buzzer_on_OK
buzzer_off	buzzer_off.py	buzzer_off_OK
button_setup	button_control_on.py	button_setup_OK
otro	---	ERROR_comando_NO_reconocido

Ilustración 34: Tabla de comandos AJAX, ejecución y respuestas

NOTA: Los comandos all_off y manual_off ejecutan el mismo código Python (all_off.py y button_control_off.py), pero no contestan lo mismo, es debido a que en el cliente se desea que no hagan exactamente lo mismo en un caso u otro.

5.3 Pruebas de proyecto

5.3.1 Pantalla de acceso o login (index.html)

Aspecto en un PC de escritorio.

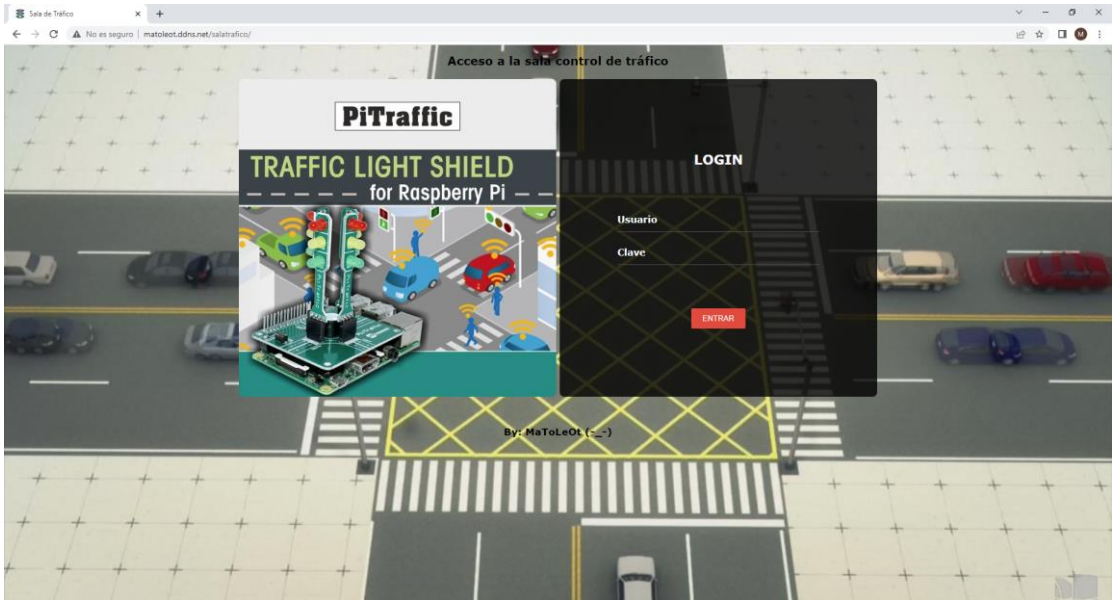


Ilustración 35: Aspecto pantalla acceso en un PC Desktop

Aspecto reduciendo pantalla (se reduce tamaño y posición).

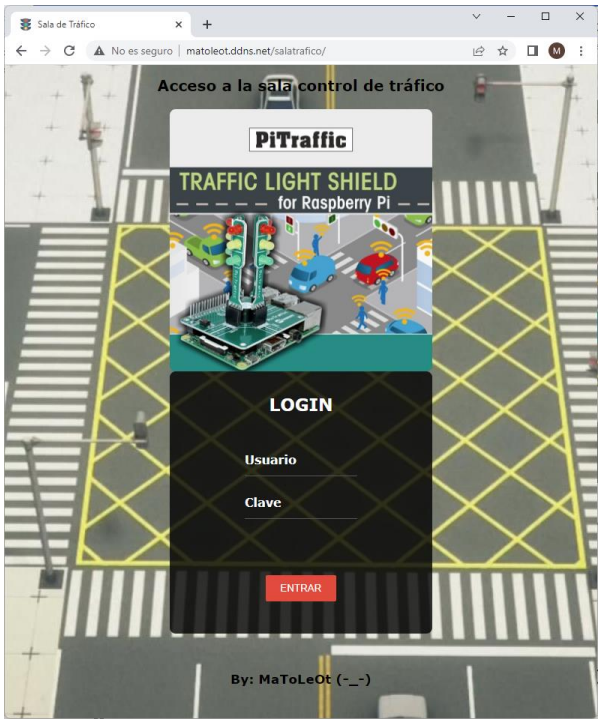


Ilustración 36: Aspecto pantalla acceso reduciendo pantalla

Aspecto en un Smartphone (se oculta la imagen).

Ilustración 37: Aspecto pantalla acceso en teléfono inteligente

5.3.2 Pantalla de error de credenciales (login.php)

Ponemos credenciales erróneas en un PC de escritorio.

Ilustración 38: Aspecto pantalla error en PC

Ponemos credenciales erróneas en smartphone.

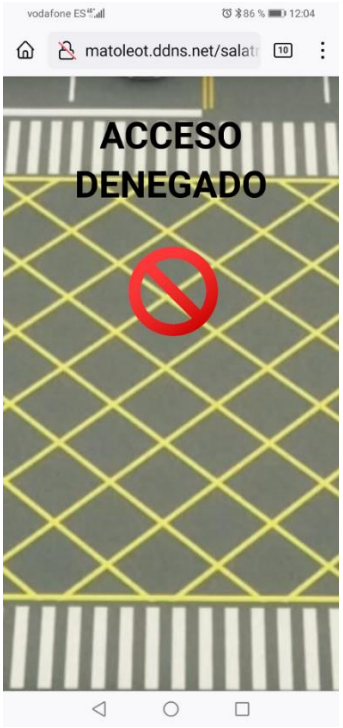


Ilustración 39: Aspecto pantalla error en teléfono inteligente

5.3.3 Pantalla de control (salatrafico.php)

Si el proceso login es correcto se nos lleva a salatrafico.php

Aspecto en un PC de escritorio (con todo apagado).

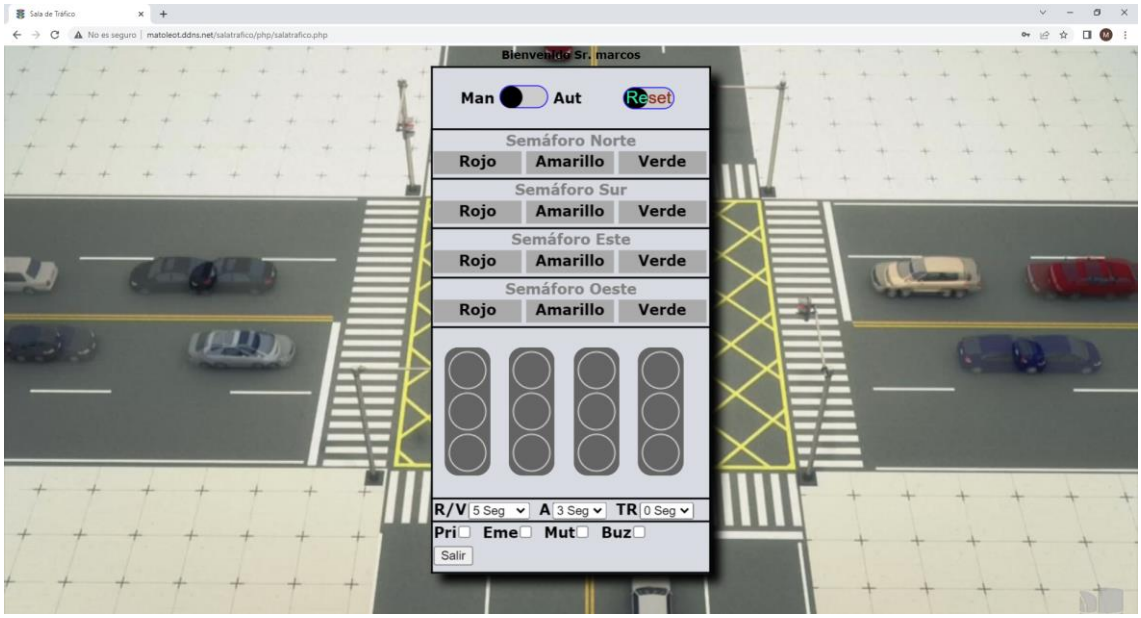


Ilustración 40: Aspecto pantalla de control en PC Desktop (apagado)

Aspecto en un Smartphone (se oculta imagen de fondo y se escala).

Ilustración 41: Aspecto pantalla de control en smartphone (apagado)

5.3.4 Modo manual (Man)

Encendidos varios.

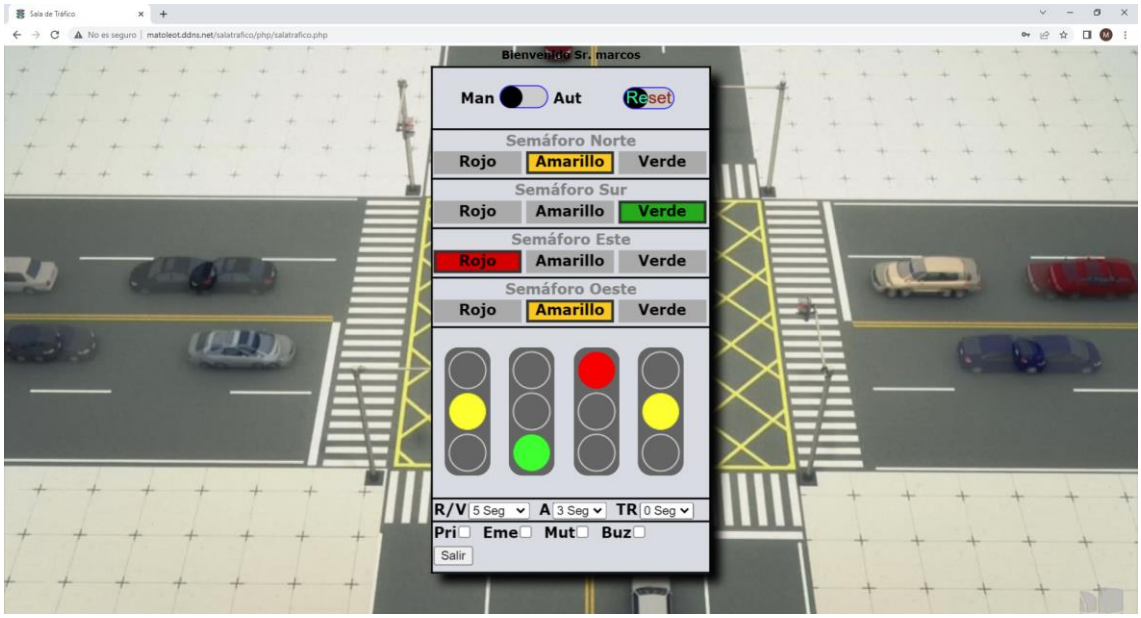


Ilustración 42: Aspecto pantalla de control en PC (Modo Man I)

Todo en rojo y con el buzzer sonando.

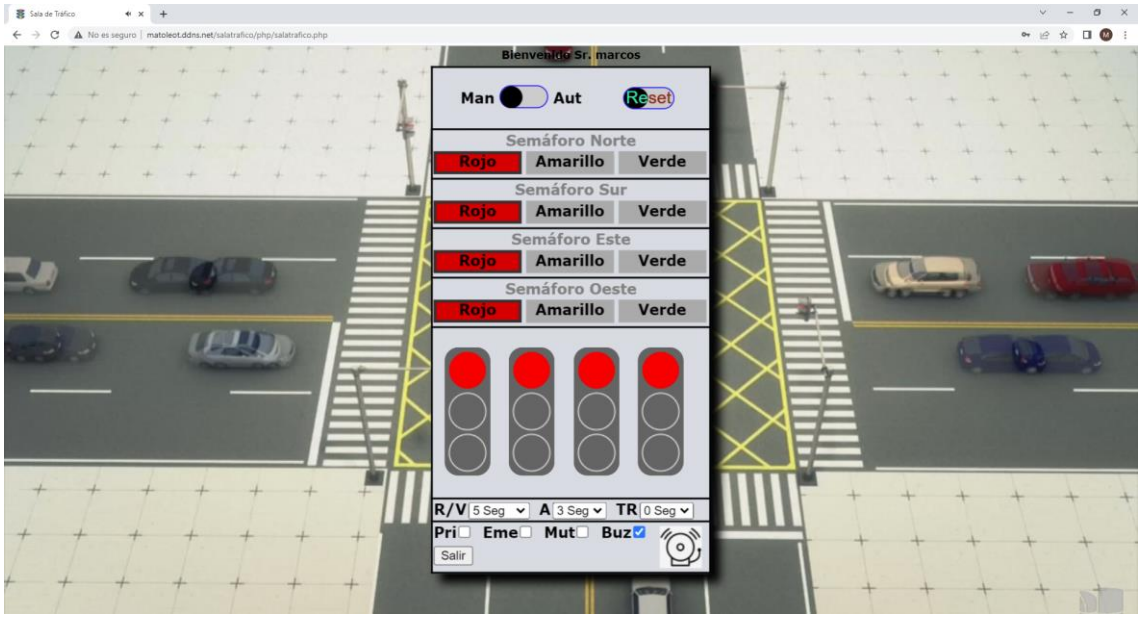


Ilustración 43: Aspecto pantalla de control en PC (Modo Man II)

En smartphone.

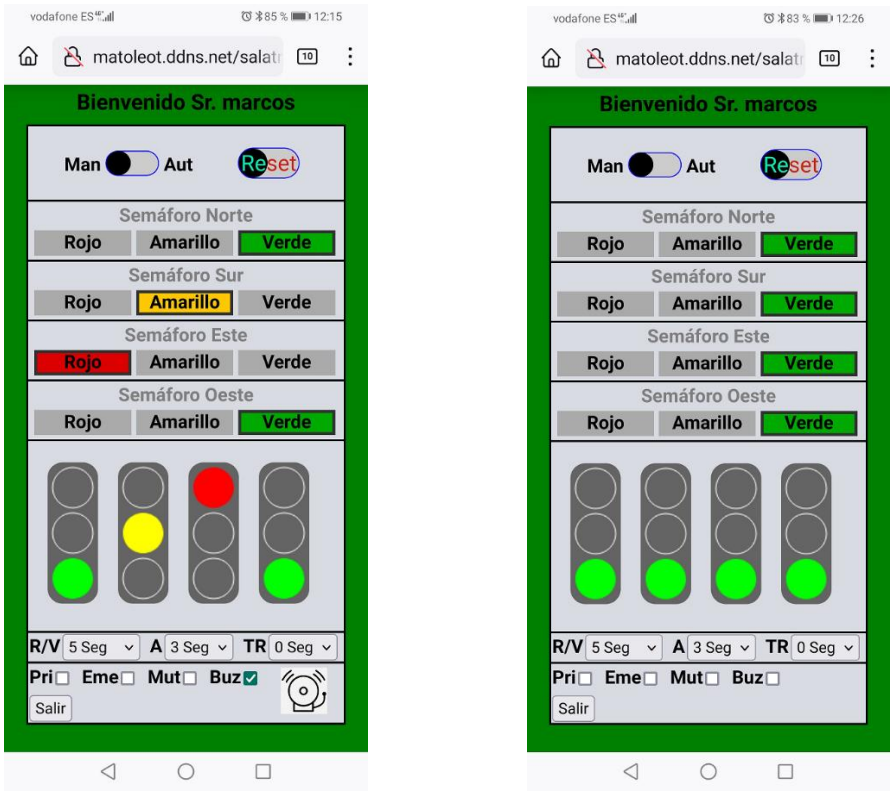


Ilustración 44: Aspecto pantalla de control en smartphone (Modo Man I y II)

5.3.5 Modo emergencia (Eme)

Parpadeante amarillo con buzzer.

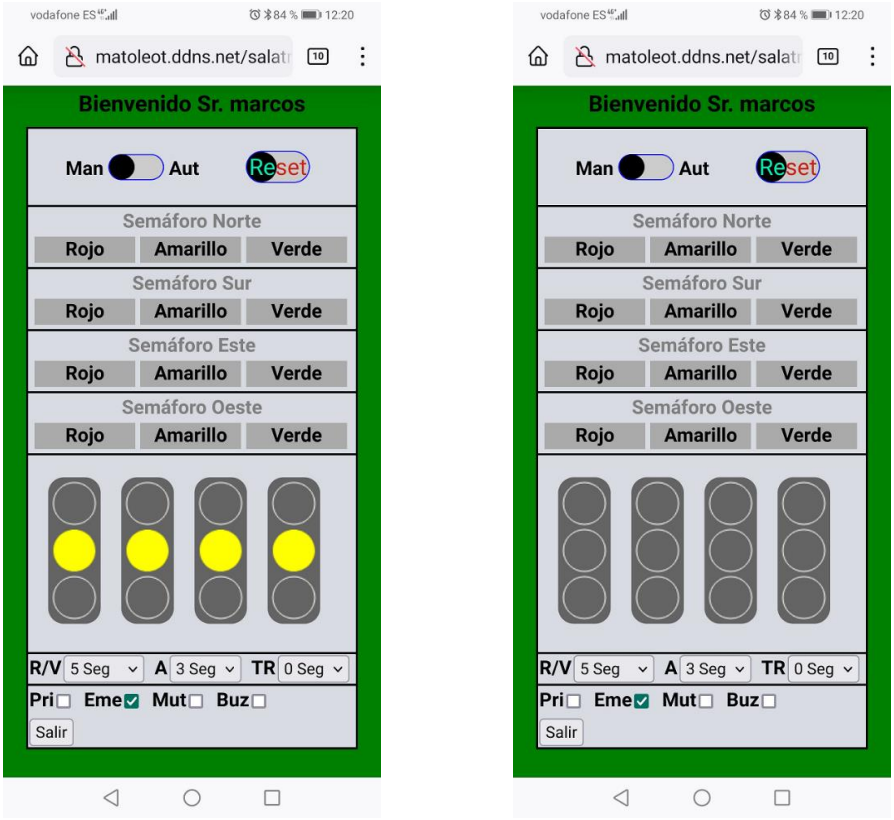


Ilustración 45: Aspecto pantalla de control en smartphone (Modo Eme I y II)

5.3.6 Modo automático (Aut)

Estado 1

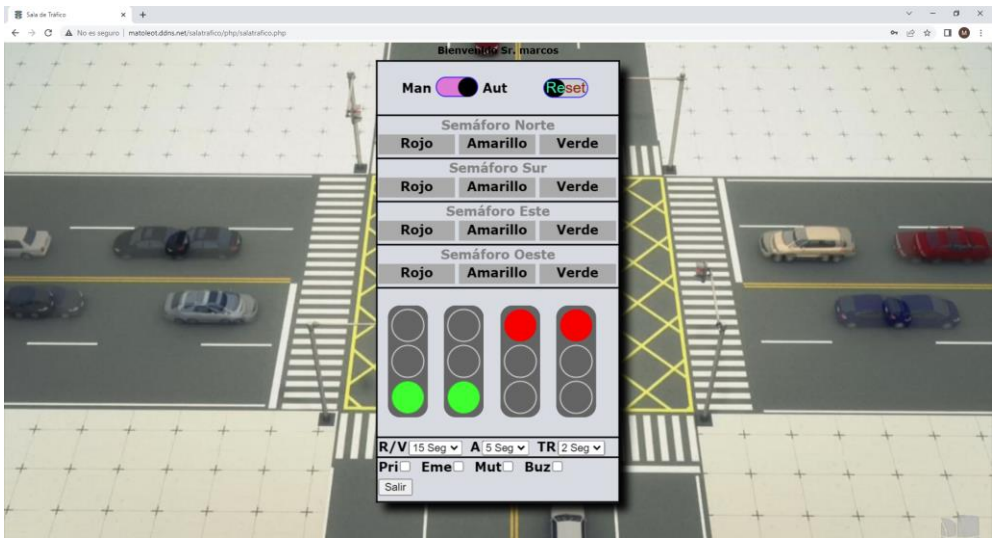


Ilustración 46: Aspecto pantalla de control en PC (Modo Aut, estado 1)

Estado 2

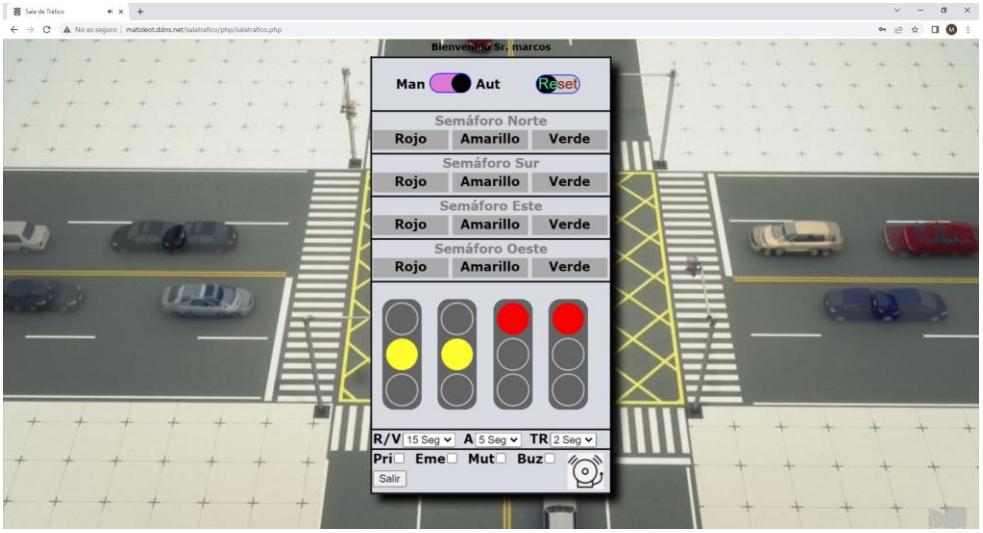


Ilustración 47: Aspecto pantalla de control en PC (Modo Aut, estado 2)

Estados 3, 6 y 9 (TR)

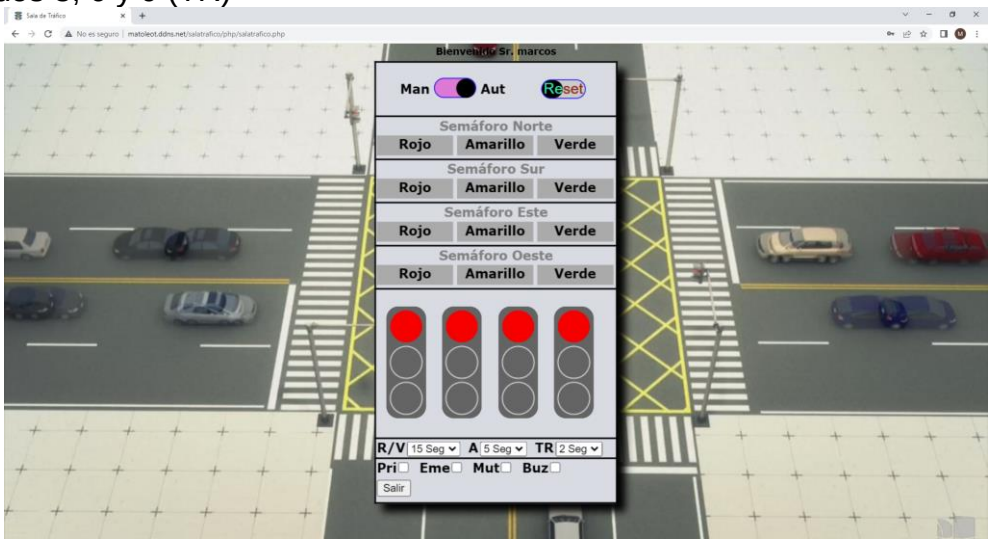


Ilustración 48: Aspecto pantalla de control en PC (Modo Aut, estados 3/6/9)

Estado 4

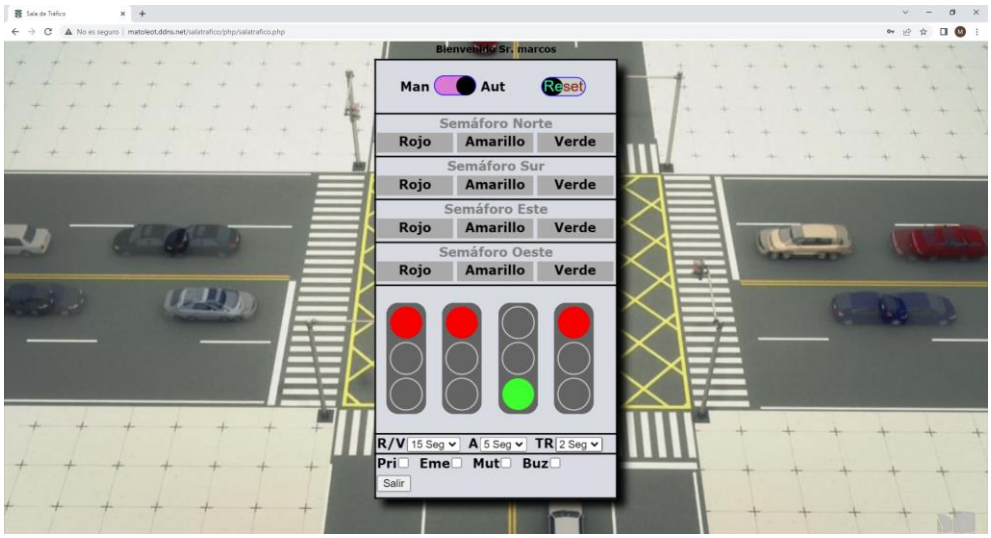


Ilustración 49: Aspecto pantalla de control en PC (Modo Aut, estado 4)

Estado 5

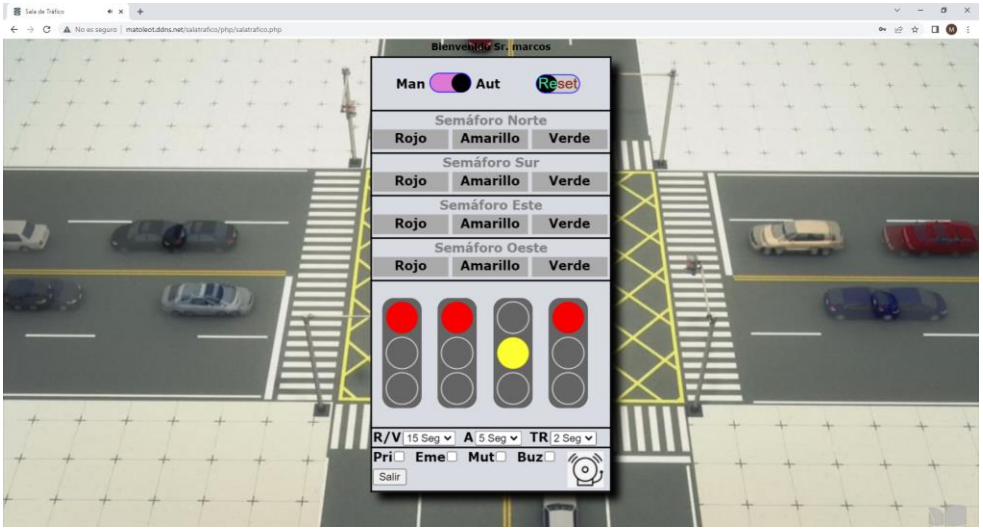


Ilustración 50: Aspecto pantalla de control en PC (Modo Aut, estado 5)

Estado 7

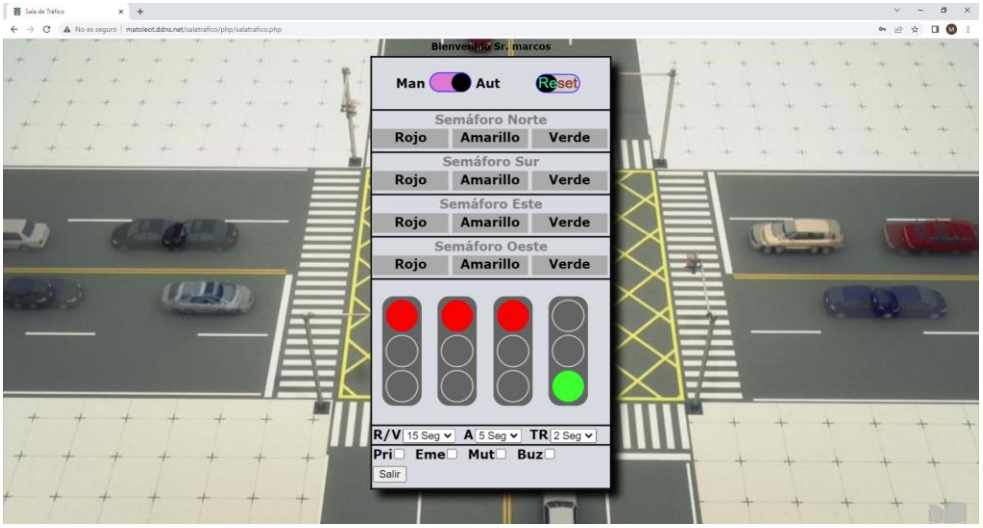


Ilustración 51: Aspecto pantalla de control en PC (Modo Aut, estado 7)

Estado 8

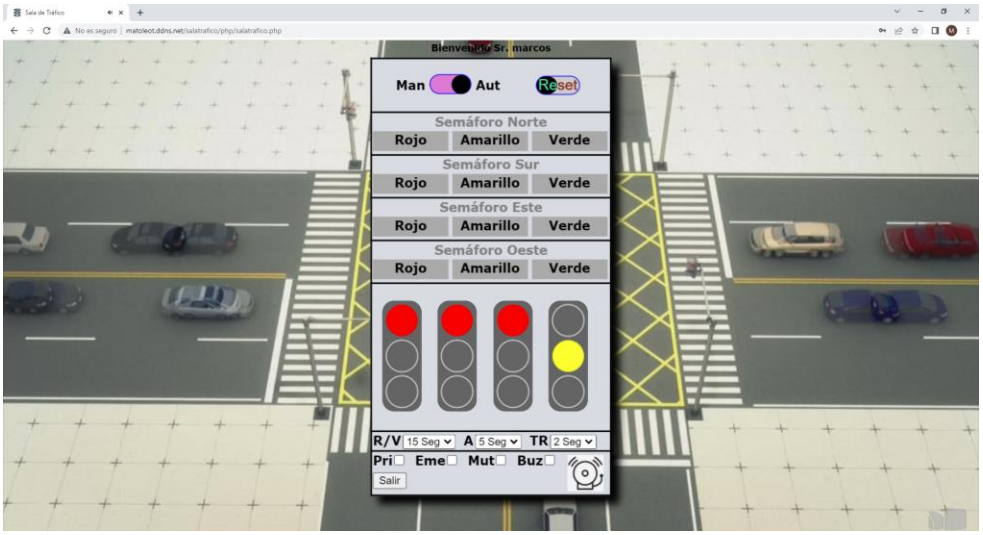


Ilustración 52: Aspecto pantalla de control en PC (Modo Aut, estado 8)

Ahora en smartphone y con Pri y Mut activados.

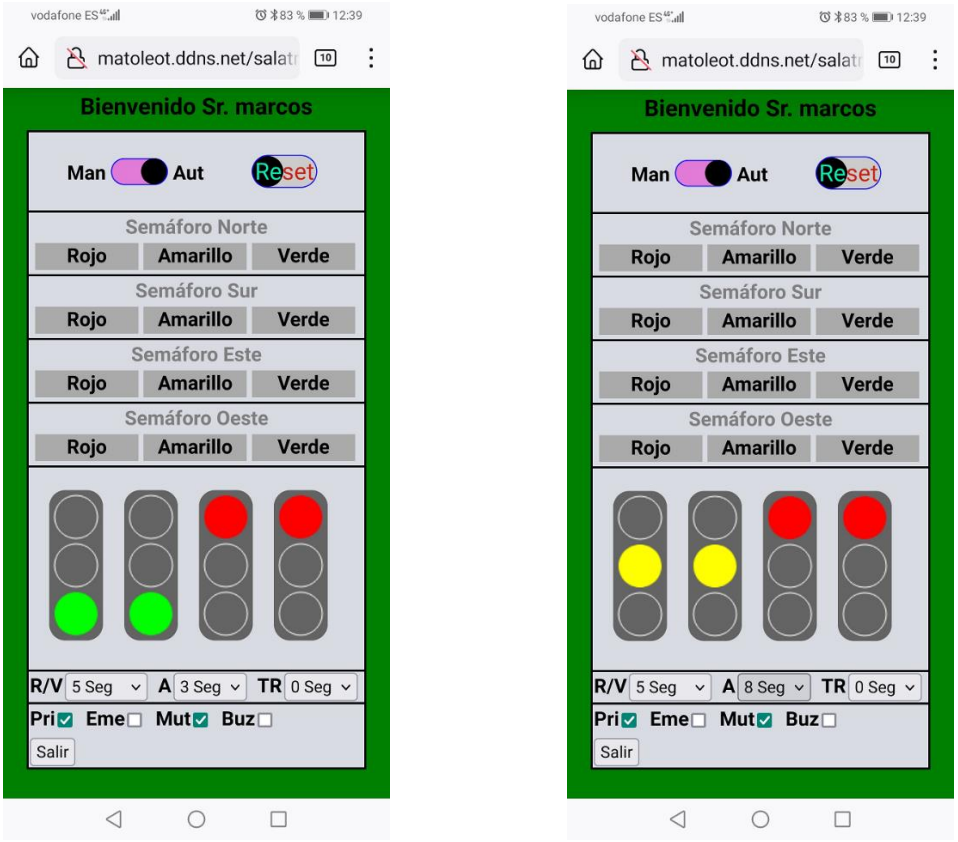


Ilustración 53: Aspecto pantalla de control en PC (Modo Aut, Pri y Mut)

5.3.7 Log's

En la consola del navegador podemos ver numerosos log's, incluido el envío de comandos y la recepción de respuestas o mensajes. Nada más entrar a la web, se puede observar, cómo se manda vía AJAX "**comando=manual_on**" y se recibe "**manual_on_OK**" como mensaje de respuesta.

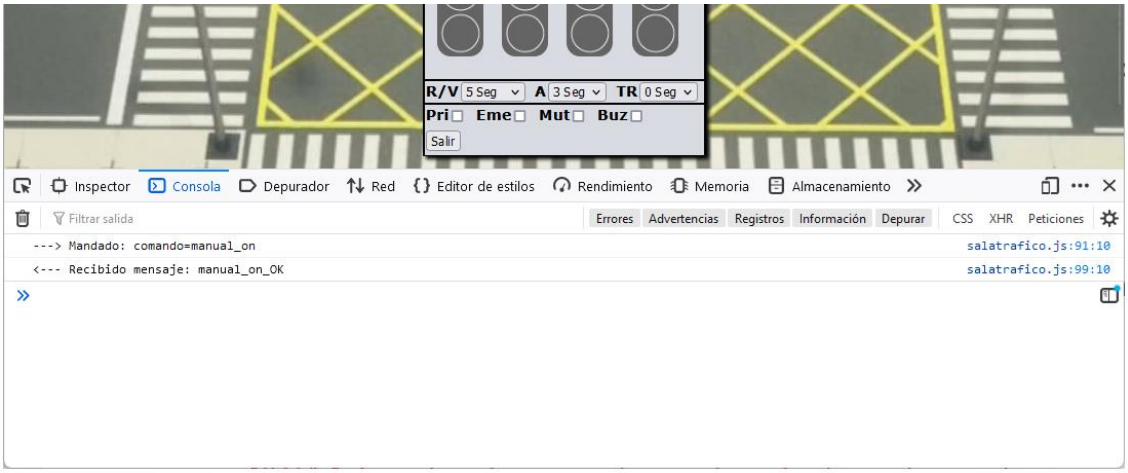


Ilustración 54: Aspecto log's (I)

Si hacemos algo en manual.

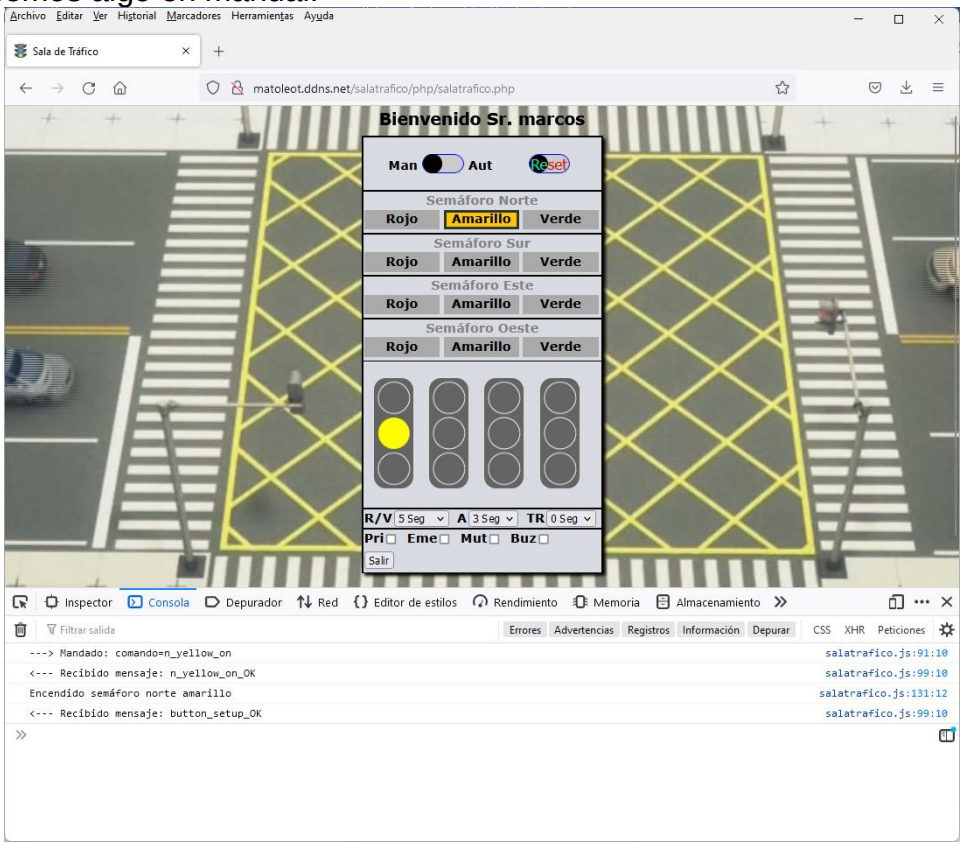


Ilustración 55: Aspecto log's (II)

Más acciones en manual.

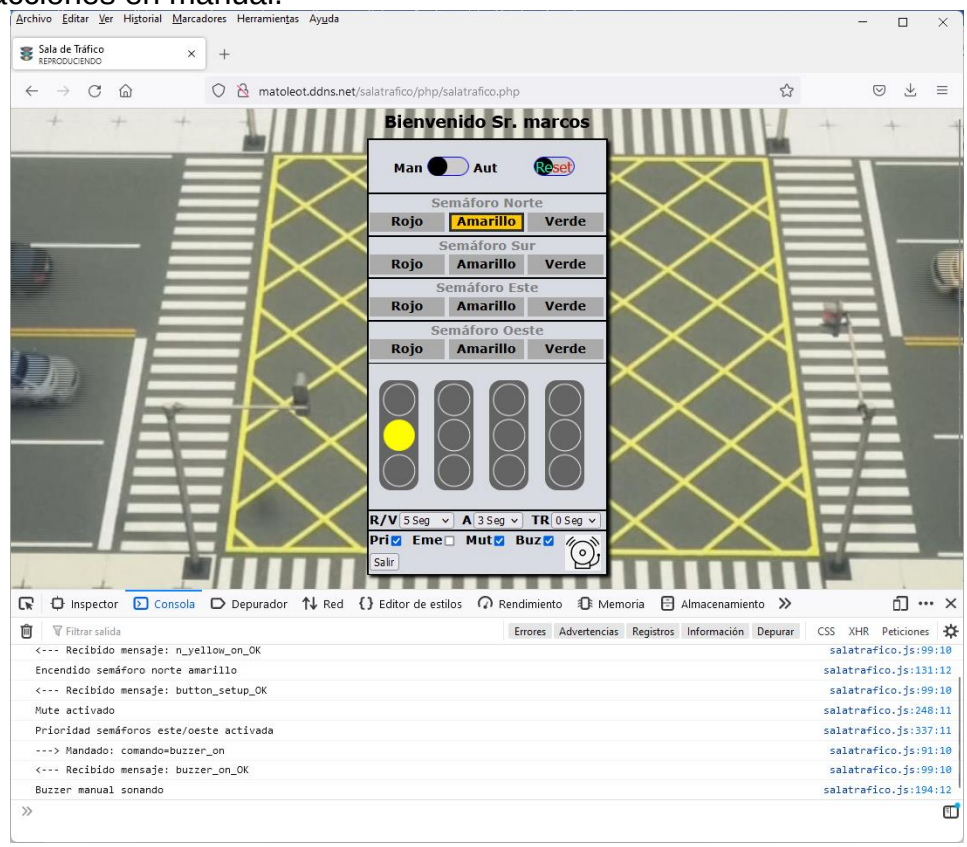


Ilustración 56: Aspecto log's (III)

Ponemos el automático.

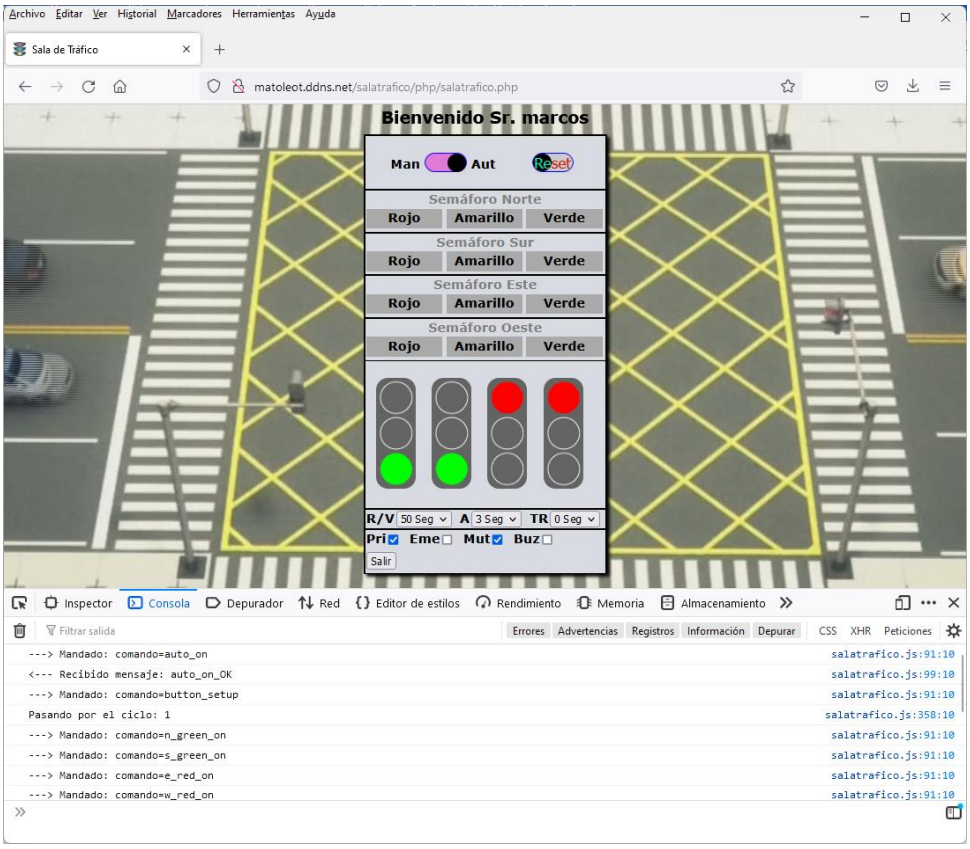


Ilustración 57: Aspecto log's (IV)

Tras pulsar el button.

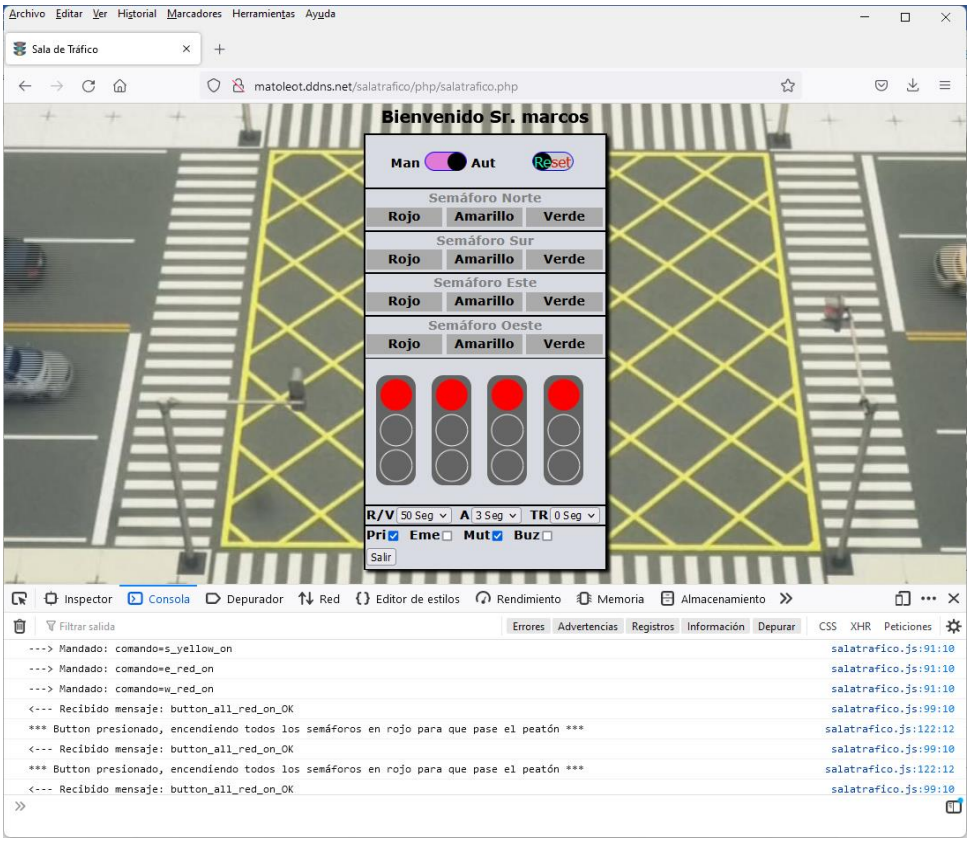


Ilustración 58: Aspecto log's (V)

5.3.8 Fotografías reales del servidor RPi con el PiTraffic en funcionamiento

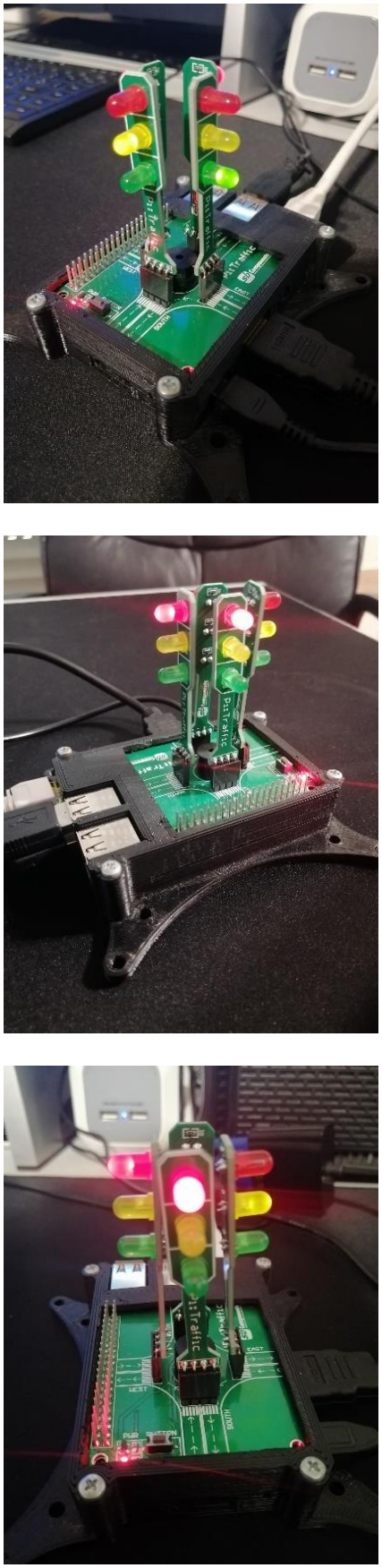


Ilustración 59: Imágenes reales del semáforo (I, II y III)

6. Revisión de la planificación

A fecha de entrega de este TFG (9/6/22), se puede afirmar, que la planificación ha transcurrido durante estas semanas prácticamente tal cual estaba planificada, pues el proyecto se ha terminado a tiempo poco antes de dicha fecha, sin que hayan cambios apreciables en relación con la planificación y el trabajo realmente efectuado.

Desde la fecha de comienzo del proyecto el 16/2/22, se ha conseguido entregar completado y a tiempo cada ítem del proyecto, el “Plan de Trabajo” (PEC1) el 1/3/22, el “Análisis y Diseño” (PEC2) el 1/4/22, la “Implementación” (PEC3) el 23/5/22, y, finalmente, la memoria y presentación (PEC4) el 9/6/22.

Podemos afirmar por lo tanto, que este proyecto ha sido muy equilibrado en cuanto a planificación y uso eficiente del tiempo se refiere, proponiendo unos objetivos coherentes y alcanzables en los tiempos propuestos.

7. Conclusiones

Mediante la definición previa de los objetivos, la planificación, la implementación y la redacción de la presente memoria de TFG, hemos aprendido a integrar en un único proyecto varias tecnologías, hardware, aplicaciones y lenguajes, para la consecución de un fin preestablecido, que ha consistido en poder controlar vía web y desde el lado cliente un cruce de dos calles con cuatro semáforos, sea en manual o automático, y viendo en tiempo real el estado de las lámparas de todos los semáforos en la pantalla del ordenador cliente.

Sobre las líneas de trabajo futuro se pueden establecer las siguiente:

►Este trabajo es experimental, se ha escogido que el control esté en el lado cliente vía AJAX para probar las latencias, aprovechando las respuestas asíncronas para “dibujar” en tiempo real los semáforos en la pantalla del cliente, pero si el cliente “cae” los semáforos se paran. Por ello, en un sistema real o profesional de control semafórico vía web, la lógica de control debería estar en el lado servidor, usando el AJAX solamente para cambiar de modo, cambiar tiempos o aspectos similares.

►Si se aplica el punto anterior, un avance importante sería que desde el cliente se pudieran controlar a voluntad muchos cruces, teniendo cada uno su propia RPi, y seleccionando de alguna manera el que se desee controlar en cada momento, por ejemplo, mediante su IP o URL, cada cruce tendría su verdadera lógica de control en el propio servidor. En contrapartida, esto dificultaría ver los semáforos en pantalla en tiempo real.

8. Glosario

Definimos a continuación, de los términos y acrónimos más relevantes utilizados dentro de la Memoria.

Raspberry Pi. Mini computador ideal para pequeñas aplicaciones, IoT o uso por los makers para DIY.

PiTraffic. Dispositivo enchufable en la RaspberryPi a modo de sombrero, que aporta físicamente un cruce de dos calles con cuatro de semáforos.

HTML. Lenguaje de marcas estándar (HyperText Markup Language), base de las páginas web, creando su estructura.

CSS. Lenguaje de estilos estándar (Cascading StyleSheets), usado junto con el HTML, para darle el aspecto deseado.

Javascript. Lenguaje web de lado cliente más empleado, por ser el estándar.

AJAX. Sistema de comunicación asíncrona (Asynchronous JavaScript and XML), muy empleada en páginas web.

PHP. Lenguaje web de lado servidor (Hypertext Preprocessor) más empleado.

Python. Lenguaje intérprete de alto nivel muy de moda hoy día.

Canvas. Etiqueta HTML5 que facilita un “lienzo” para dibujar en su interior.

Browser. Navegador web.

Switch. Estructura condicional muy usada en los lenguajes derivados del C, como Javascript o PHP.

Maker. Entusiasta de la informática y/o electrónica, muy activo en DIY tecnológico.

9. Bibliografía

Turiaci Gómez, Facundo Agustín. (6/2021).
Servidor WEB RPi para el control de un SenseHat.
[Proyecto final de CFGS (Desarrollo de aplicaciones web)].
IES J.R. Botet de Manises (Valencia).

Gómez Reche, Ismael. (6/2021).
Domótica web PiFace Digital 2.
[Proyecto final de CFGS (Desarrollo de aplicaciones web)].
IES J.R. Botet de Manises (Valencia).

SB Components. ***Video PiTraffic tutorial***. (17/2/2022).
URL: https://www.youtube.com/watch?v=6zsHnJoQaq0

SB Components. ***RPi PiTraffic traffic lights shield***. (17/2/2022).
URL: https://shop.sb-components.co.uk/products/pitraffic

SB Components. ***Librería oficial y ejemplos PiTraffic***. (30/3/2022).
URL: https://github.com/sbcshop/PiTraffic

Raspberry Pi Foundation. ***Web corporativa oficial RPi***. (17/2/2022).
URL: https://www.raspberrypi.org/

Raspberry Pi. ***Computing for everybody***. (17/2/2022).
URL: https://www.raspberrypi.com/

Mozilla foundation. ***Resources for Developers***. (HTML, CSS, JS)
URL: https://developer.mozilla.org/es/

W3C Schools. ***Learn to Code***. (HTML, CSS, JS, PHP, Python)
URL: https://www.w3schools.com/default.asp

Researchgate. ***Validar políticas de luces de semáforos.*** (18/2/2022).
URL:https://www.researchgate.net/publication/317264476_Herramienta_computacional_para_sugerir_y_validar_politicas_de_luces_de_semaforos

Wildwildweb. Como crear un interruptor con CSS. (23/04/2022).
URL: https://wildwildweb.es/es/blog/como-crear-un-interruptor-con-css

Codepen. ***JumpingInputText/CheckoutForm/UIButton.*** (24/04/2022).
URL: https://codepen.io/nikstech/pen/xmMxpr
URL: https://codepen.io/RRoberts/pen/ZOvwbM
URL: https://codepen.io/dan10gc/pen/EQbjgP

UOC. ***Formatos bibliográficos estilo APA***. (7/6/2022).
URL: https://biblioteca.uoc.edu/es/pagina/Estilo-APA/

10. Anexos

Se reproduce a continuación **el código fuente completo** de la aplicación web, no obstante, se puede encontrar en: https://github.com/matoleot/Control-experimental-de-semaforos-via-web-con-RaspBerry-Pi-y-PiTraffic

PiTraffic.py (librería oficial, modificada)

```
#!/usr/bin/python
# Library for PiTraffic
# Developed by: SB Components
# Author: Ankur
# Project: PiTraffic
# Python: 3.4.2

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

def closeGPIO():
    GPIO.cleanup()

class Buzzer:
    def __init__(self):
        self.pin = 12
        GPIO.setup(self.pin,GPIO.OUT)
        GPIO.output(self.pin, GPIO.LOW)

    def on(self):
        print("Buzzer - ON")
        GPIO.output(self.pin,GPIO.HIGH)

    def off(self):
        print("Buzzer - OFF")
        GPIO.output(self.pin,GPIO.LOW)

class Traffic:
    ''' Class to handle LED's
    Arguments:
    direction =  (i.e. "EAST","WEST","NORTH","SOUTH")
    color = Color of LED
    '''
    traffic_pins = {"SOUTH":{'RED':11,'YELLOW':13, "GREEN":15},
                    "WEST" :{'RED':16,'YELLOW':18, "GREEN":22},
                    "NORTH":{'RED':29,'YELLOW':31, "GREEN":33},
                    "EAST" :{'RED':36,'YELLOW':38, "GREEN":40}}

    def __init__(self, direction, color):
        self.pin = self.traffic_pins[direction][color]
        self.direction = direction
        self.color = color
        GPIO.setup(self.pin,GPIO.OUT)
        GPIO.output(self.pin, GPIO.LOW)

    def on(self):
        print(self.direction + " " + self.color + " - ON")
        GPIO.output(self.pin,GPIO.HIGH)

    def off(self):
        print(self.direction + " " + self.color + " - OFF")
        GPIO.output(self.pin,GPIO.LOW)

class Button:
    Pressed = False

    def __init__(self):
        self.pin = 7
        GPIO.setup(self.pin,GPIO.IN, pull_up_down=GPIO.PUD_UP)

    def press(self):
        input_state = GPIO.input(self.pin)
        if input_state == False:
            print("Button - Pressed")
            self.Pressed = True
        else: # Modificado por MaToLeOt
            print("Button - Unpressed")
            self.Pressed = False
```

index.html (web de acceso)

```
<!doctype html>
<!--		Sala de control de tráfico (By: MaToLeOt) index.html -->

<html lang="es-ES">

	<head>
		<title>Sala de Tr&aacute;fico</title>
		<meta charset="UTF-8" />
		<meta name="author" content="Marcos Leiros" />
		<meta name="description" content="PiTraffic Sala de Control de Tráfico" />
		<meta name="keywords" content="PiTraffic, Raspberry, Pi, Semáforo, Control" />
		<meta name="viewport" content="width=device-width, user-scalable=no">
		<meta http-equiv="Cache-Control" content="no-store" />
		<meta http-equiv="Cache-Control" content="no-cache" />
		<meta http-equiv="Pragma" content="no-cache" />
		<link rel="stylesheet" type="text/css" href="css/index.css" media="screen" />
		<link rel="icon" type="image/png" href="img/semaicon.png">
	</head>

	<body>
		<h3> Acceso a la sala control de tr&aacute;fico</h3>

		<div id="wrapper">

			<div id="primary"></div>

			<div id="secondary">
				<form action="php/login.php" method="post">
					<span class="text-center">LOGIN</span>
					<div class="input-container">
						<input type="text" size="25" maxlenght="25" required="required"
name="usuario" value="" />
						<label>Usuario</label>
					</div>
					<div class="input-container">
						<input type="password" size="25" maxlenght="25"
required="required" name="clave" value="" />
						<label>Clave</label>
					</div>
					<button type="submit" class="btn">ENTRAR</button>
				</form>
			</div>

		</div>
		<span class="varios">By: MaToLeOt (-_-)</span>
	</body>

</html>
```

login.php (proceso autenticación)

```
<!DOCTYPE html>
<html lang="es-ES">
	<head>
		<title>Sala de Tr&aacute;fico</title>
		<meta name="viewport" content="width=device-width, user-scalable=no">
		<meta charset="utf-8">
		<link rel="stylesheet" href="../css/login.css">
		<script language="javascript">function redirigir_index(){location.href="../index.html";}</script>
		<link rel="icon" type="image/png" href="../img/semaicon.png">
	</head>

	<body>
		<?php
			include("datos_conexion.inc.php");

			$mysqlid = new mysqli($mysql_server, $mysql_login, $mysql_pass, 'pitraffic');
			$mysqlid -> set_charset("utf8");

			$sentencia = "SELECT usuario, clave FROM usuarios;";
			$registros = $mysqlid->query($sentencia);

			$permiso=false;
			if(isset($_POST['usuario']) and isset($_POST['clave']) and count($_POST)==2) {
				while($registro = $registros->fetch_object()){
					if($registro->usuario==$_POST['usuario'] && $registro-
>clave==$_POST['clave']){
						$permiso = true;
						$_SESSION['permiso']=true;
						$_SESSION['usuario']=$_POST['usuario'] ;

						#header("Location:
salatrafico.php?".session_name()."=".session_id());
```

```
                                                header("Location: salatrafico.php");
                                                die();
                                        }
                                }
                        }
                        if(!$permiso) {
                                echo "<h1>ACCESO DENEGADO</h1>";
                                echo "<div id='wrapperprohibido'>";
                                echo "<img src='../img/direccion_prohibida.gif' id='prohibido' alt='Prohibido' />";
                                echo "</div>";
                                echo "<script language='javascript'>setTimeout(redirigir_index,7000);</script>";
                        }
                ?>
        </body>
</html>
```

datos_conexion.inc.php (credenciales mysql)

```
<?php
        # Sala de control de tráfico (By: MaToLeOt) datos_conexion.inc.php
        $mysql_server="localhost";
        $mysql_login="root";
        $mysql_pass="raspberry";
?>
```

salatrafico.php (web principal)

```
<?php
        # Sala de control de tráfico (By: MaToLeOt) salatrafico.php
        session_cache_limiter('nocache,private');
        session_start();

        $permiso="";
        if($_SESSION) {
                $permiso=$_SESSION['permiso'];
                $usuario=$_SESSION['usuario'];
        }else{
                header("Location: ../index.html");
                die();
        }
?>

<!DOCTYPE html>
<!--    Sala de control de tráfico (By: MaToLeOt) salatrafico.php -->
<html lang="es-ES">

        <head>
                <title>Sala de Tr&aacute;fico</title>
                <meta charset="UTF-8" />
                <meta name="author" content="Marcos Leiros" />
                <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=yes" />
                <meta http-equiv="Cache-Control" content="no-store" />
                <meta http-equiv="Cache-Control" content="no-cache" />
                <meta http-equiv="Pragma" content="no-cache" />
                <link rel="stylesheet" type="text/css" href="../css/salatrafico.css" media="screen" />
                <link rel="icon" type="image/png" href="../img/semaicon.png">
                <script type="text/javascript " src="../js/salatrafico.js" ></script >
        </head>

        <body>

                <?php
                        if(!$permiso) {
                                header("Location: ../index.html");
                                die();
                        }
                ?>

                <h3>
                        Bienvenido Sr. <?php echo $usuario; ?>
                </h3>

                <table>
                        <tr><td>
                                <div id='automanrst'>
                                        <div class='container1'>
                                                Man
                                                <label class='switch'>
                                                        <input type='checkbox' id='automan'
onchange='setTimeout(() => { validar_automan(this.checked); }, 200); ' />
```

```
                                                                <div class='slider round'></div>
                                                        </label>
                                                        Aut
                                                </div>
                                                <div class='container2'>
                                                        <button class='btn rounded' value='all_off'
onclick='llamadaAjax(this.value);'><span class='text-green'>Reset</span></button>
                                                </div>
                                        </div>
                                </td></tr>

                                <tr><td>
                                        <div>
                                        <span class='semaforonseo'>Sem&aacute;foro Norte</span>
                                        </div>
                                        <div id='norte'  class='container' >
                                                <form class='form' >
                                                        <input type='radio' class='radiosem' name='nor' id='norter'
value='n_red_on' /><label class='four col' for='norter'>Rojo</label>
                                                        <input type='radio' class='radiosem' name='nor' id='nortea'
value='n_yellow_on' /><label class='four col' for='nortea'>Amarillo</label>
                                                        <input type='radio' class='radiosem' name='nor' id='nortev'
value='n_green_on' /><label class='four col' for='nortev'>Verde</label>
                                                </form>
                                        </div>
                                </td></tr>

                                <tr><td>
                                        <div>
                                        <span class='semaforonseo'>Sem&aacute;foro Sur</span>
                                        </div>
                                        <div id='sur'  class='container' >
                                                <form class='form' >
                                                        <input type='radio' class='radiosem' name='sur' id='surr'
value='s_red_on' /><label class='four col' for='surr'>Rojo</label>
                                                        <input type='radio' class='radiosem' name='sur' id='sura'
value='s_yellow_on' /><label class='four col' for='sura'>Amarillo</label>
                                                        <input type='radio' class='radiosem' name='sur' id='surv'
value='s_green_on' /><label class='four col' for='surv'>Verde</label>
                                                </form>
                                        </div>
                                </td></tr>

                                <tr><td>
                                        <div>
                                        <span class='semaforonseo'>Sem&aacute;foro Este</span>
                                        </div>
                                        <div id='este'  class='container' >
                                                <form class='form' >
                                                        <input type='radio' class='radiosem' name='este' id='ester'
value='e_red_on' /><label class='four col' for='ester'>Rojo</label>
                                                        <input type='radio' class='radiosem' name='este' id='estea'
value='e_yellow_on' /><label class='four col' for='estea'>Amarillo</label>
                                                        <input type='radio' class='radiosem' name='este' id='estev'
value='e_green_on' /><label class='four col' for='estev'>Verde</label>
                                                </form>
                                        </div>
                                </td></tr>

                                <tr><td>
                                        <div>
                                        <span class='semaforonseo'>Sem&aacute;foro Oeste</span>
                                        </div>
                                        <div id='oeste'  class='container' >
                                                <form class='form' >
                                                        <input type='radio' class='radiosem' name='oeste' id='oester'
value='w_red_on' /><label class='four col' for='oester'>Rojo</label>
                                                        <input type='radio' class='radiosem' name='oeste' id='oestea'
value='w_yellow_on' /><label class='four col' for='oestea'>Amarillo</label>
                                                        <input type='radio' class='radiosem' name='oeste' id='oestev'
value='w_green_on' /><label class='four col' for='oestev'>Verde</label>
                                                </form>
                                        </div>
                                </td></tr>

                                <tr><td>
                                        <div id='semaforos'>
                                                <canvas id='semN' width='50' height='140'></canvas>
                                                <canvas id='semS' width='50' height='140'></canvas>
                                                <canvas id='semE' width='50' height='140'></canvas>
                                                <canvas id='semW' width='50' height='140'></canvas>
                                        </div>
                                </td></tr>

                                <tr><td>
                                        <div id='dselect'>
```

```
                                                <span>R/V</span><select class='stiempo' id='stiempoRV'
onchange='scambiarRV(this.options[this.selectedIndex].value);'>
                                                        <option value='5000' selected>5 Seg</option>
                                                        <option value='15000'>15 Seg</option>
                                                        <option value='30000'>30 Seg</option>
                                                        <option value='50000'>50 Seg</option></select>
                                                <span>A</span><select class='stiempo' id='stiempoA'
onchange='scambiarA(this.options[this.selectedIndex].value);'>
                                                        <option value='3000' selected>3 Seg</option>
                                                        <option value='5000'>5 Seg</option>
                                                        <option value='8000'>8 Seg</option></select>
                                                <span>TR</span><select class='stiempo' id='stiempoTR'
onchange='scambiarTR(this.options[this.selectedIndex].value);'>
                                                        <option value='0' selected>0 Seg</option>
                                                        <option value='1000'>1 Seg</option>
                                                        <option value='2000'>2 Seg</option>
                                                        <option value='3000'>3 Seg</option> </select>
                                        </div>
                                </td></tr>

                                <tr><td>
                                        <div id='total'>
                                                <div id='wrappercheks'>
                                                        <span>Pri</span><input type='checkbox' id='pcheck' class='pie'
onchange='ccambiarPRI(this.checked);' />
                                                        <span>Eme</span><input type='checkbox' id='echeck' class='pie'
onchange='cemergency(this.checked);' />
                                                        <span>Mut</span><input type='checkbox' id='mcheck' class='pie'
onchange='cmute(this.checked);' />
                                                        <span>Buz</span><input type='checkbox' id='bcheck' class='pie'
onchange='csonido(this.checked);' />
                                                </div>
                                                <br />
                                                <div id='salir'>
                                                        <form action="cerrar_sesion.php"><input type="submit"
value="Salir" /></form>
                                                </div>
                                        </div>

                                        <div id='imgbuzzer'>
                                                <img id='ibuzz' src='../img/buzzer-icon.png' alt='Sonido' />
                                        </div>
                                </td></tr>
                        </table>

                </body>
</html>
```

salatrafico.js (control lado cliente)

```
// Sala de control de tráfico (By: MaToLeOt) salatrafico.js

// bloque variables globales e iniciaciones --------------------------
window.onload = function(){
        //obtengo contexto canvas semáforo N
        canvasN = document.getElementById('semN');
        ctxN = canvasN.getContext('2d');
        //obtengo contexto canvas semáforo S
        canvasS = document.getElementById('semS');
        ctxS= canvasS.getContext('2d');
        //obtengo contexto canvas semáforo E
        canvasE = document.getElementById('semE');
        ctxE = canvasE.getContext('2d');
        //obtengo contexto canvas semáforo W
        canvasW = document.getElementById('semW');
        ctxW = canvasW.getContext('2d');

        //listeners varios
        //apaga todo si salimos
        window.addEventListener("unload", (event) => {
                resetear(true);
                todo_off();
                llamadaAjax("all_off");
                console.log("Evento unload en window salatrafico");
        });
        //obtiene los 12 radios y asocia evento
        radiosems=document.getElementsByClassName('radiosem');
        for(var i=0; i<radiosems.length; i++){
                radiosems[i].addEventListener("click", validar);
        }

        //variables para elmodo automático y emergencia
        id_timeout_ajax=null;   //id del setTimeout ajax
        id_automatico=null;     //id del setTimeout auto
```

```
	id_emergencia=null;	//id del setTimeout emer
	ciclo=1;					//ciclo inicial
	mTiempo=8000;				//variable auxiliar
	mTiempoRV=5000;		//tiempo de rojo y verde
	mTiempoA=3000;		//tiempo en amarillo
	mTiempoTR=0;				//tiempo todo en rojo
	prioridad=false;			//estado prioridad
	mute=false;					//estado mute

	//inicio siempre todo reseteado y en manual
	resetear(true);
	todo_off();
	llamadaAjax("manual_on");
}
// ---------------------------------------------------------------------------

// bloque validar radios y checkbox automan -------------------------
function validar_automan(estado){ //a true si automático
	if(estado){
		if(document.getElementById('echeck').checked){
			document.getElementById('automan').checked =false;
			alert("No se puede usar este control estando en\nmodo emergencia (Eme), desáctivelo primero. \n");
		}else{
			llamadaAjax("auto_on");
		}
	}else{
		llamadaAjax("manual_on");
	}
}

function validar(cmd){ //manda comando si manual
	if(document.getElementById('automan').checked || document.getElementById('echeck').checked){
		resetear(false);
		alert("No se puede usar este control estando en\nmodo automático o emergencia (Aut/Eme). \n");
	}else{
		if(cmd=="buzzer_on" || cmd=="buzzer_off"){
			llamadaAjax(cmd);
		}else{
			llamadaAjax(cmd.target.value);
		}
	}
}
// ---------------------------------------------------------------------------

// bloque de comunicación asíncrona ----------------------------------
function llamadaAjax(cmd){ //servicio asíncrono para mandar comandos al servidor
	let xhr = new XMLHttpRequest();
	xhr.onreadystatechange = function() {
		if (this.readyState == 4 && this.status == 200) {
			analizaRespuestaAjax(this.responseText);
		}
	};
	xhr.open('POST', 'control.php');
	xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
	xhr.send("comando=" + cmd);

	console.log("---> Mandado: comando=" + cmd);
}

function analizaRespuestaAjax(msg){ //analiza comandos recibidos y actúa en consecuencia
	//let msglen=msg.length;
	//console.log("Longitud del comando recibido: " + msglen);
	//msg=msg.substring(0, msglen-3);

	console.log("<--- Recibido mensaje: " + msg);

	if(msg=="all_off_OK") {//OJO recepcion de msgs
		resetear(true);		//true para que resetee checkbox automático y tiempos
		emergencia_off();
	}else if(msg=="manual_on_OK"){
		resetear(false);
	}

	if(msg=="all_off_OK" || msg=="manual_on_OK") {
		tauto_off();
		todo_off();
		setTimeout(() => { todo_off(); }, 250);		/* seguridad */
	}

	if(msg=="auto_on_OK") tauto_on();

	switch(msg){ //analiza respuesta y pinta semáforos canvas
		case "button_all_red_on_OK":	//todos rojo button on
			x_rojo_on(ctxN);
			x_rojo_on(ctxS);
			x_rojo_on(ctxE);
```

```
                        x_rojo_on(ctxW);
                        console.log("*** Button presionado, encendiendo los semáforos en rojo para que pase el peatón ***");
                        ciclo=0; // pone el ciclo especial de todo en rojo por button presionado
                        break;
                case "n_red_on_OK":             //norte rojo on
                        x_rojo_on(ctxN);
                        console.log("Encendido semáforo norte rojo");
                        break;
                case "n_yellow_on_OK":          //norte amarillo on
                        x_amarillo_on(ctxN);
                        console.log("Encendido semáforo norte amarillo");
                        break;
                case "n_green_on_OK":           //norte verde on
                        x_verde_on(ctxN);
                        console.log("Encendido semáforo norte verde");
                        break;
                case "s_red_on_OK":             //sur rojo on
                        x_rojo_on(ctxS);
                        console.log("Encendido semáforo sur rojo");
                        break;
                case "s_yellow_on_OK":          //sur amarillo on
                        x_amarillo_on(ctxS);
                        console.log("Encendido semáforo sur amarillo");
                        break;
                case "s_green_on_OK":                   //sur verde on
                        x_verde_on(ctxS);
                        console.log("Encendido semáforo sur verde");
                        break;
                case "e_red_on_OK":             //este rojo on
                        x_rojo_on(ctxE);
                        console.log("Encendido semáforo este rojo");
                        break;
                case "e_yellow_on_OK":          //este amarillo on
                        x_amarillo_on(ctxE);
                        console.log("Encendido semáforo este amarillo");
                        break;
                case "e_green_on_OK":           //este verde on
                        x_verde_on(ctxE);
                        console.log("Encendido semáforo este verde");
                        break;
                case "w_red_on_OK":             //oeste rojo on
                        x_rojo_on(ctxW);
                        console.log("Encendido semáforo oeste rojo");
                        break;
                case "w_yellow_on_OK":          //oeste  amarillo on
                        x_amarillo_on(ctxW);
                        console.log("Encendido semáforo oeste amarillo");
                        break;
                case "w_green_on_OK":           //oeste verde on
                        x_verde_on(ctxW);
                        console.log("Encendido semáforo oeste verde");
                        break;
                case "all_red_on_OK":           //todos rojo on
                        x_rojo_on(ctxN);
                        x_rojo_on(ctxS);
                        x_rojo_on(ctxE);
                        x_rojo_on(ctxW);
                        console.log("Encendidos todos los semáforos en rojo");
                        break;
                case "all_yellow_on_OK":        //todos amarillo on
                        x_amarillo_on(ctxN);
                        x_amarillo_on(ctxS);
                        x_amarillo_on(ctxE);
                        x_amarillo_on(ctxW);
                        console.log("Encendidos todos los semáforos en amarillo");
                        break;
                case "all_yellow_off_OK":       //todos amarillo off
                        todo_off();
                        console.log("Apagados todos los semáforos en amarillo");
                        break;
                case "buzzer_on_OK":            //sonido manual on
                        document.getElementById('ibuzz').style.visibility="visible"; //visualiza buzzer (suena en el servidor)
                        new Audio("../mp3/zumbador_pitraffic.mp3").play();      //suena sonido en el cliente
                        console.log("Buzzer manual sonando");
                        break;
                case "buzzer_off_OK":           //sonido manual off
                        document.getElementById('ibuzz').style.visibility="hidden"; //invisibiliza buzzer (ya no suena)
                        console.log("Buzzer manual apagado");
                        break;
                }
}
// ---------------------------------------------------------------------

// bloque resetear ----------------------------------------------------
function resetear(rst){  //desactiva los radio, desactiva checkbox buzzer, oculta buzzer y otros
        for(var i=0; i<radiosems.length; i++){
```

```
                        radiosems[i].checked=false;
                }

                document.getElementById('bcheck').checked=false;

        document.getElementById('imgbuzzer').style.visibility="hidden";

                if(rst){//a true se ha usado el radio de reset para mandar el comando all_off
                        //desactiva checkbox automan, prioridad, emergencia y mute
                        document.getElementById('automan').checked = false;
                        document.getElementById('pcheck').checked = false;
                        document.getElementById('echeck').checked = false;
                        document.getElementById('mcheck').checked = false;

                        //resetea variables tiempos, selects, prioridad y mute
                        ciclo=1;
                        mTiempo=8000;
                        mTiempoRV=5000;
                        mTiempoA=3000;
                        mTiempoTR=0;
                        document.getElementById('stiempoRV').value=mTiempoRV;
                        document.getElementById('stiempoA').value=mTiempoA;
                        document.getElementById('stiempoTR').value=mTiempoTR;
                        prioridad=false;
                        mute=false;
                }
}
// ---------------------------------------------------------------------------

// bloque sonido buzzer ------------------------------------------------
function csonido(estado){               //gestiona el checkbox del buzzer
                if(estado){
                        validar("buzzer_on");
                }else{
                        validar("buzzer_off");
                }
}

function cmute(estado){                 //cambia estado mute
                if(estado){
                        mute=true;
                        console.log("Mute activado");
                }
                else{
                        mute=false;
                        console.log("Mute desactivado");
                }
}

function sonido_temporal(){             //manda sonar buzzer temporal en servidor, visualiza y suena buzzer cliente
                llamadaAjax("buzzer_5_on");             //manda sonar zumbador en el servidor
                document.getElementById('ibuzz').style.visibility="visible";                    //visualiza buzzer en pantalla ciente
                new Audio("../mp3/zumbador_pitraffic.mp3").play();                              //suena sonido en el cliente
                setTimeout(() => {document.getElementById('ibuzz').style.visibility="hidden";}, 1750); //al tiempo invisibiliza
                console.log("Activando Buzzer temporalmente");
}
// ---------------------------------------------------------------------------

// bloque emergencia ---------------------------------------------------
function cemergency(estado){            //gestiona el checkbox de emergencia
                if(document.getElementById('automan').checked){
                        alert("No se puede usar este control, ponga antes el modo manual \n");
                        document.getElementById('echeck').checked=false;
                }else{
                        if(estado){
                                mTiempo=2100;
                                ciclo=1;
                                emergencia_on();
                                console.log("Modo emergencia activado");
                        }else{
                                mTiempo=8000;
                                ciclo=1;
                                emergencia_off();
                                console.log("Modo emergencia desactivado");
                        }
                }
}

function emergencia_on(){                       //enciende emergencia
                resetear(false);
                ciclo=1;
                emergencia_roll();
}

function emergencia_off(){                      //apaga emergencia
                window.clearTimeout(id_emergencia);
```

```
		document.getElementById('echeck').checked = false;
		llamadaAjax("all_yellow_off");
}

function emergencia_roll(){		//rueda emergencia
		switch(ciclo){
			case 1:	//todo amarillo
				llamadaAjax("all_yellow_on");
				if(!mute){
					sonido_temporal();	//suena y se visualiza en el cliente
				}
				mTiempo=2000;
				ciclo=2;
				break;
			case 2:	//todo apagado
				llamadaAjax("all_yellow_off");
				mTiempo=1000;
				ciclo=1;
				break;
		}

		id_emergencia = window.setTimeout(emergencia_roll, mTiempo);	//espera siguiente estado de emergencia
}
// ---------------------------------------------------------------------

// bloque automático ------------------------------------------------
function scambiarRV(tim){//cambia tiempo rojo y verde
		mTiempoRV=tim;
		console.log("Tiempo en Rojo y Verde cambiado a: " + tim/1000 + " segundos");
}

function scambiarA(tim){		//cambia tiempo amarillo
		mTiempoA=tim;
		console.log("Tiempo en Amarillo cambiado a: " + tim/1000 + " segundos");
}

function scambiarTR(tim){		//cambia todo en rojo
		mTiempoTR=tim;
		console.log("Tiempo especial de todos en Rojo cambiado a: " + tim/1000 + " segundos");
}

function ccambiarPRI(estado){	//cambia prioridad
		if(estado){
			prioridad=true;
			console.log("Prioridad semáforos este/oeste activada");
		}
		else{
			prioridad=false;
			console.log("Prioridad semáforos este/oeste desactivada");
		}
}

function tauto_on(){			//enciende automático
		resetear(false);
		ciclo=1;
		llamadaAjax("button_setup");
		auto_roll();
}

function tauto_off(){			//apaga automático
		window.clearTimeout(id_automatico);
		todo_off();
}

function auto_roll(){			//rueda automática
		console.log("Pasando por el ciclo: " + ciclo);
		switch(ciclo){
			case 0:	//Button presionado. Todo en rojo
				mTiempo=15000;
				ciclo=1;
				break;
			case 1:	//norte y sur en verde
				llamadaAjax("n_green_on");
				llamadaAjax("s_green_on");
				llamadaAjax("e_red_on");
				llamadaAjax("w_red_on");
				mTiempo=mTiempoRV;
				ciclo=2;
				break;
			case 2:	//norte y sur en amarillo
				llamadaAjax("n_yellow_on");
				llamadaAjax("s_yellow_on");
				llamadaAjax("e_red_on");
				llamadaAjax("w_red_on");
				if(!mute){
					sonido_temporal();			//suena y se visualiza en el cliente
```

```
                        }
                        mTiempo=mTiempoA;
                        mTiempoTR != 0 ?  ciclo=3 : ciclo=4
                        break;
                case 3:  //todo rojo (opcional según TR)
                        llamadaAjax("all_red_on");
                        mTiempo=mTiempoTR;
                        ciclo=4;
                        break;
                case 4:  //este en verde
                        llamadaAjax("n_red_on");
                        llamadaAjax("s_red_on");
                        llamadaAjax("e_green_on");
                        llamadaAjax("w_red_on");
                        prioridad==true ? mTiempo=mTiempoRV*2 : mTiempo=mTiempoRV; // tiempo según prioridad
                        ciclo=5;
                        break;
                case 5:  //este amarillo
                        llamadaAjax("n_red_on");
                        llamadaAjax("s_red_on");
                        llamadaAjax("e_yellow_on");
                        llamadaAjax("w_red_on");
                        if(!mute){
                                sonido_temporal();              //suena y se visualiza en el cliente
                        }
                        mTiempo=mTiempoA;
                        mTiempoTR != 0 ?  ciclo=6 : ciclo=7     //a 0 se salta todo en rojo (TR)
                        break;
                case 6:  //todo rojo (opcional según TR)
                        llamadaAjax("all_red_on");
                        mTiempo=mTiempoTR;
                        ciclo=7;
                        break;
                case 7:  //oeste en verde
                        llamadaAjax("n_red_on");
                        llamadaAjax("s_red_on");
                        llamadaAjax("e_red_on");
                        llamadaAjax("w_green_on");
                        prioridad==true ? mTiempo=mTiempoRV*2 : mTiempo=mTiempoRV; //tiempo según prioridad
                        ciclo=8;
                        break;
                case 8:  //oeste amarillo
                        llamadaAjax("n_red_on");
                        llamadaAjax("s_red_on");
                        llamadaAjax("e_red_on");
                        llamadaAjax("w_yellow_on");
                        if(!mute){
                                sonido_temporal();              //suena y se visualiza en el cliente
                        }
                        mTiempo=mTiempoA;
                        mTiempoTR != 0 ?  ciclo=9 : ciclo=1     //a 0 se salta todo en rojo (TR)
                        break;
                case 9:  //todo rojo (opcional según TR)
                        llamadaAjax("all_red_on");
                        mTiempo=mTiempoTR;
                        ciclo=1;
                        break;
        }

        id_automatico = window.setTimeout(auto_roll, mTiempo);          //espera siguiente estado de automático
}
// ---------------------------------------------------------------------

// bloque de dibujo semáforos en el canvas ----------------------------
function todo_off(){            //todos semáforos apagados
        //inicio canvas semáforo norte
        canvasN.style.borderRadius = "15px";
        ctxN.fillStyle="rgb(100, 100,100)";
    ctxN.fillRect(0, 0, 50, 140);
        //norte rojo off
        ctxN.beginPath();
        ctxN.strokeStyle="rgb(255, 255, 255)";
        ctxN.arc(25, 25, 20, 0, 2 * Math.PI);
        ctxN.stroke();
        //norte amarillo off
        ctxN.beginPath();
        ctxN.strokeStyle="rgb(255, 255, 255)";
        ctxN.arc(25, 70, 20, 0, 2 * Math.PI);
        ctxN.stroke();
        //norte verde off
        ctxN.beginPath();
        ctxN.strokeStyle="rgb(255, 255, 255)";
        ctxN.arc(25, 115, 20, 0, 2 * Math.PI);
        ctxN.stroke();

        //inicio canvas semáforo sur
```

```
            canvasS.style.borderRadius = "15px";
            ctxS.fillStyle="rgb(100, 100,100)";
        ctxS.fillRect(0, 0, 50, 140);
            //sur rojo off
            ctxS.beginPath();
            ctxS.strokeStyle="rgb(255, 255, 255)";
            ctxS.arc(25, 25, 20, 0, 2 * Math.PI);
            ctxS.stroke();
            //sur amarillo off
            ctxS.beginPath();
            ctxS.strokeStyle="rgb(255, 255, 255)";
            ctxS.arc(25, 70, 20, 0, 2 * Math.PI);
            ctxS.stroke();
            //sur verde off
            ctxS.beginPath();
            ctxS.strokeStyle="rgb(255, 255, 255)";
            ctxS.arc(25, 115, 20, 0, 2 * Math.PI);
            ctxS.stroke();

            //inicio canvas semáforo este
            canvasE.style.borderRadius = "15px";
            ctxE.fillStyle="rgb(100, 100,100)";
        ctxE.fillRect(0, 0, 50, 140);
            //este rojo off
            ctxE.beginPath();
            ctxE.strokeStyle="rgb(255, 255, 255)";
            ctxE.arc(25, 25, 20, 0, 2 * Math.PI);
            ctxE.stroke();
            //este amarillo off
            ctxE.beginPath();
            ctxE.strokeStyle="rgb(255, 255, 255)";
            ctxE.arc(25, 70, 20, 0, 2 * Math.PI);
            ctxE.stroke();
            //este verde off
            ctxE.beginPath();
            ctxE.strokeStyle="rgb(255, 255, 255)";
            ctxE.arc(25, 115, 20, 0, 2 * Math.PI);
            ctxE.stroke();

            //inicio canvas semáforo oeste
            canvasW.style.borderRadius = "15px";
            ctxW.fillStyle="rgb(100, 100,100)";
        ctxW.fillRect(0, 0, 50, 140);
            //oeste rojo off
            ctxW.beginPath();
            ctxW.strokeStyle="rgb(255, 255, 255)";
            ctxW.arc(25, 25, 20, 0, 2 * Math.PI);
            ctxW.stroke();
            //oeste amarillo off
            ctxW.beginPath();
            ctxW.strokeStyle="rgb(255, 255, 255)";
            ctxW.arc(25, 70, 20, 0, 2 * Math.PI);
            ctxW.stroke();
            //oeste verde off
            ctxW.beginPath();
            ctxW.strokeStyle="rgb(255, 255, 255)";
            ctxW.arc(25, 115, 20, 0, 2 * Math.PI);
            ctxW.stroke();
}

function x_rojo_on(ctx){                        //x rojo on
            ctx.fillStyle="rgb(100, 100,100)";
        ctx.fillRect(0, 0, 50, 140);
            //rojo on
            ctx.beginPath();
            ctx.fillStyle="rgb(255, 0, 0)";
            ctx.arc(25, 25, 20, 0, 2 * Math.PI);
            ctx.fill();
            //amarillo off
            ctx.beginPath();
            ctx.strokeStyle="rgb(255, 255, 255)";
            ctx.arc(25, 70, 20, 0, 2 * Math.PI);
            ctx.stroke();
            //verde off
            ctx.beginPath();
            ctx.strokeStyle="rgb(255, 255, 255)";
            ctx.arc(25, 115, 20, 0, 2 * Math.PI);
            ctx.stroke();
}

function x_amarillo_on(ctx){          //x amarillo on
            ctx.fillStyle="rgb(100, 100,100)";
        ctx.fillRect(0, 0, 50, 140);
            //rojo off
            ctx.beginPath();
            ctx.strokeStyle="rgb(255, 255, 255)";
```

```
		ctx.arc(25, 25, 20, 0, 2 * Math.PI);
		ctx.stroke();
		//amarillo on
		ctx.beginPath();
		ctx.fillStyle="rgb(255, 255, 0)";
		ctx.arc(25, 70, 20, 0, 2 * Math.PI);
		ctx.fill();
		//verde off
		ctx.beginPath();
		ctx.strokeStyle="rgb(255, 255, 255)";
		ctx.arc(25, 115, 20, 0, 2 * Math.PI);
		ctx.stroke();
}

function x_verde_on(ctx){				//x verde on
		ctx.fillStyle="rgb(100, 100,100)";
	ctx.fillRect(0, 0, 50, 140);
		//rojo off
		ctx.beginPath();
		ctx.strokeStyle="rgb(255, 255, 255)";
		ctx.arc(25, 25, 20, 0, 2 * Math.PI);
		ctx.stroke();
		//amarillo off
		ctx.beginPath();
		ctx.strokeStyle="rgb(255, 255, 255)";
		ctx.arc(25, 70, 20, 0, 2 * Math.PI);
		ctx.stroke();
		//verde on
		ctx.beginPath();
		ctx.fillStyle="rgb(0, 255, 0)";
		ctx.arc(25, 115, 20, 0, 2 * Math.PI);
		ctx.fill();
}
// ---------------------------------------------------------------------------

// THE END - By: MaToLeOt ------------------------------------------
```

cerrar_sesion.php (cierra la sesión)

```
<?php
		# Sala de control de tráfico (By: MaToLeOt) cerrar_sesion.php
		session_start();
		session_destroy();
		header("Location: ../index.html");
		die();
?>
```

control.php (control central y activador del python)

```
<?php
		# Sala de control de tráfico (By: MaToLeOt) control.php
		header('Content-Type: text/html; charset=utf-8');

		#si se ha presionado el button pone todo en rojo
		$fp=fopen("presionado.dat", "r");
		$dato=fgets($fp);
		fclose($fp);
		if($dato=="1"){
			exec("sudo python ../py/all_red_on.py");	echo "button_all_red_on_OK";
			$fp=fopen("presionado.dat", "w");
			fwrite($fp, "0");
			fclose($fp);
			exit();
		}

		switch($_REQUEST[comando]){
			# todo off
			case "all_off":		exec("sudo python ../py/all_off.py");
						echo "all_off_OK";
						exec("sudo python ../py/button_control_off.py");
						break;
			# manual
			case "manual_on":	exec("sudo python ../py/all_off.py");
						echo "manual_on_OK";
						exec("sudo python ../py/button_control_off.py");
						break;
			# auto
			case "auto_on":		exec("sudo python ../py/all_off.py");
						echo "auto_on_OK";
						break;
			# N=NORTH (NORTE)
			case "n_red_on":	exec("sudo python ../py/n_red_on.py");		echo "n_red_on_OK";			break;
```

```
                case "n_yellow_on":   exec("sudo python ../py/n_yellow_on.py");     echo "n_yellow_on_OK";          break;
                case "n_green_on":    exec("sudo python ../py/n_green_on.py");      echo "n_green_on_OK";           break;
                # S=SOUTH (SUR)
                case "s_red_on":      exec("sudo python ../py/s_red_on.py");        echo "s_red_on_OK";             break;
                case "s_yellow_on":   exec("sudo python ../py/s_yellow_on.py");     echo "s_yellow_on_OK";          break;
                case "s_green_on":    exec("sudo python ../py/s_green_on.py");      echo "s_green_on_OK";           break;
                # E=EAST (ESTE)
                case "e_red_on":      exec("sudo python ../py/e_red_on.py");        echo "e_red_on_OK";             break;
                case "e_yellow_on":   exec("sudo python ../py/e_yellow_on.py");     echo "e_yellow_on_OK";          break;
                case "e_green_on":    exec("sudo python ../py/e_green_on.py");      echo "e_green_on_OK";           break;
                # W=WEST (OESTE)
                case "w_red_on":      exec("sudo python ../py/w_red_on.py");        print "w_red_on_OK";             break;
                case "w_yellow_on":   exec("sudo python ../py/w_yellow_on.py");     echo "w_yellow_on_OK";          break;
                case "w_green_on":    exec("sudo python ../py/w_green_on.py");      echo "w_green_on_OK";           break;
                #N, S, E, y W Red
                case "all_red_on":    exec("sudo python ../py/all_red_on.py");      echo "all_red_on_OK";           break;
                case "all_red_off":   exec("sudo python ../py/all_red_off.py");     echo "all_red_off_OK";          break;
                #N, S, E, y W Yellow
                case "all_yellow_on":  exec("sudo python ../py/all_yellow_on.py");  echo "all_yellow_on_OK";        break;
                case "all_yellow_off": exec("sudo python ../py/all_yellow_off.py"); echo "all_yellow_off_OK";       break;
                # buzzer
                case "buzzer_5_on":   exec("sudo python ../py/buzzer_5_on.py");     echo "buzzer_5_on_OK";          break;
                case "buzzer_10_on": exec("sudo python ../py/buzzer_10_on.py");     echo "buzzer_10_on_OK";         break;
                case "buzzer_on":     exec("sudo python ../py/buzzer_on.py");       echo "buzzer_on_OK";            break;
                case "buzzer_off":    exec("sudo python ../py/buzzer_off.py");      echo "buzzer_off_OK";           break;
                # button
                case "button_setup":  exec("sudo python ../py/button_control_on.py");  echo "button_setup_OK";       break;
                # contestación por defecto
                default:      print "ERROR_comando_NO_reconocido";                                                    break;
        }
?>
```

all_off.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (all_off.py)

import PiTraffic

# Bloque declaración constantes
SouthRed = PiTraffic.Traffic("SOUTH", "RED")
SouthYellow = PiTraffic.Traffic("SOUTH", "YELLOW")
SouthGreen = PiTraffic.Traffic("SOUTH", "GREEN")
EastRed = PiTraffic.Traffic("EAST", "RED")
EastYellow = PiTraffic.Traffic("EAST", "YELLOW")
EastGreen = PiTraffic.Traffic("EAST", "GREEN")
NorthRed = PiTraffic.Traffic("NORTH", "RED")
NorthYellow = PiTraffic.Traffic("NORTH", "YELLOW")
NorthGreen = PiTraffic.Traffic("NORTH", "GREEN")
WestRed = PiTraffic.Traffic("WEST", "RED")
WestYellow = PiTraffic.Traffic("WEST", "YELLOW")
WestGreen = PiTraffic.Traffic("WEST", "GREEN")
Buzz = PiTraffic.Buzzer()

# Bloque acción
Buzz.off()
WestRed.off()
WestYellow.off()
WestGreen.off()
EastRed.off()
EastYellow.off()
EastGreen.off()
NorthRed.off()
NorthYellow.off()
NorthGreen.off()
SouthRed.off()
SouthYellow.off()
SouthGreen.off()
```

all_red_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (all_red_on.py)

import PiTraffic

# Bloque declaración constantes
SouthRed = PiTraffic.Traffic("SOUTH", "RED")
SouthYellow = PiTraffic.Traffic("SOUTH", "YELLOW")
SouthGreen = PiTraffic.Traffic("SOUTH", "GREEN")
EastRed = PiTraffic.Traffic("EAST", "RED")
EastYellow = PiTraffic.Traffic("EAST", "YELLOW")
```

```
EastGreen = PiTraffic.Traffic("EAST", "GREEN")
NorthRed = PiTraffic.Traffic("NORTH", "RED")
NorthYellow = PiTraffic.Traffic("NORTH", "YELLOW")
NorthGreen = PiTraffic.Traffic("NORTH", "GREEN")
WestRed = PiTraffic.Traffic("WEST", "RED")
WestYellow = PiTraffic.Traffic("WEST", "YELLOW")
WestGreen = PiTraffic.Traffic("WEST", "GREEN")
Buzz = PiTraffic.Buzzer()

# Bloque acción
WestYellow.off()
WestGreen.off()
EastYellow.off()
EastGreen.off()
NorthYellow.off()
NorthGreen.off()
SouthYellow.off()
SouthGreen.off()
WestRed.on()
EastRed.on()
NorthRed.on()
SouthRed.on()
```

all_yellow_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (all_yellow_on.py)

import PiTraffic

# Bloque declaración constantes
SouthRed = PiTraffic.Traffic("SOUTH", "RED")
SouthYellow = PiTraffic.Traffic("SOUTH", "YELLOW")
SouthGreen = PiTraffic.Traffic("SOUTH", "GREEN")
EastRed = PiTraffic.Traffic("EAST", "RED")
EastYellow = PiTraffic.Traffic("EAST", "YELLOW")
EastGreen = PiTraffic.Traffic("EAST", "GREEN")
NorthRed = PiTraffic.Traffic("NORTH", "RED")
NorthYellow = PiTraffic.Traffic("NORTH", "YELLOW")
NorthGreen = PiTraffic.Traffic("NORTH", "GREEN")
WestRed = PiTraffic.Traffic("WEST", "RED")
WestYellow = PiTraffic.Traffic("WEST", "YELLOW")
WestGreen = PiTraffic.Traffic("WEST", "GREEN")
Buzz = PiTraffic.Buzzer()

# Bloque acción
WestRed.off()
WestGreen.off()
EastRed.off()
EastGreen.off()
NorthRed.off()
NorthGreen.off()
SouthRed.off()
SouthGreen.off()
WestYellow.on()
EastYellow.on()
NorthYellow.on()
SouthYellow.on()
```

button_control_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (button_control_on.py)

import PiTraffic
import time

# Bloque declaración constantes
Buzz = PiTraffic.Buzzer()
Btn=PiTraffic.Button()

# Bloque métodos
def sonar(tim):
        Buzz.on()
        time.sleep(tim)
        Buzz.off()

# Bloque acción
file1=open("presionado.dat", "w")
file1.write("0")
```

```
file1.close()
file2=open("button.dat", "w")
file2.write("1")
file2.close()

while(True):
        time.sleep(8)
        file3=open("button.dat", "r")
        dato=file3.read(1)
        file3.close()
        if(dato=='0'):
                break;
        Btn.press()
        if(Btn.Pressed==True):
                file4=open("presionado.dat", "w")
                file4.write("1")
                file4.close()
                sonar(3)
```

button_control_off.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (button_control_off.py)

# Bloque acción
file1=open("button.dat", "w")
file1.write("0")
file1.close()
file2=open("presionado.dat", "w")
file2.write("0");
file2.close()
```

buzzer_5_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (buzzer_5_on.py)

import PiTraffic
import time

# Bloque declaración constantes
Buzz = PiTraffic.Buzzer()

# Bloque acción
Buzz.on()
time.sleep(0.2)
Buzz.off()
time.sleep(0.2)
Buzz.on()
time.sleep(0.2)
Buzz.off()
time.sleep(0.2)
Buzz.on()
time.sleep(0.2)
Buzz.off()
time.sleep(0.2)
Buzz.on()
time.sleep(0.2)
Buzz.off()
time.sleep(0.2)
Buzz.on()
time.sleep(0.2)
Buzz.off()
```

buzzer_off.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (buzzer_off.py)

import PiTraffic

# Bloque declaración constantes
Buzz = PiTraffic.Buzzer()

# Bloque acción
Buzz.off()
```

buzzer_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (buzzer_on.py)

import PiTraffic

# Bloque declaración constantes
Buzz = PiTraffic.Buzzer()

# Bloque acción
Buzz.on()
```

e_green_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (e_green_on.py)

import PiTraffic

# Bloque declaración constantes
EastRed = PiTraffic.Traffic("EAST", "RED")
EastYellow = PiTraffic.Traffic("EAST", "YELLOW")
EastGreen = PiTraffic.Traffic("EAST", "GREEN")

# Bloque funciones
def East_Red_Off():
    EastRed.off()
def East_Yellow_Off():
    EastYellow.off()
def East_Green_On():
    EastGreen.on()

# Bloque acción
East_Red_Off()
East_Yellow_Off()
East_Green_On()
```

e_red_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (e_red_on.py)

import PiTraffic

# Bloque declaración constantes
EastRed = PiTraffic.Traffic("EAST", "RED")
EastYellow = PiTraffic.Traffic("EAST", "YELLOW")
EastGreen = PiTraffic.Traffic("EAST", "GREEN")

# Bloque funciones
def East_Yellow_Off():
    EastYellow.off()
def East_Green_Off():
    EastGreen.off()
def East_Red_On():
    EastRed.on()

# Bloque acción
East_Yellow_Off()
East_Green_Off()
East_Red_On()
```

e_yellow_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (e_yellow_on.py)

import PiTraffic

# Bloque declaración constantes
EastRed = PiTraffic.Traffic("EAST", "RED")
EastYellow = PiTraffic.Traffic("EAST", "YELLOW")
EastGreen = PiTraffic.Traffic("EAST", "GREEN")
```

```
# Bloque funciones
def East_Red_Off():
    EastRed.off()
def East_Green_Off():
    EastGreen.off()
def East_Yellow_On():
    EastYellow.on()

# Bloque acción
East_Red_Off()
East_Green_Off()
East_Yellow_On()
```

n_green_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (n_green_on.py)

import PiTraffic

# Bloque declaración constantes
NorthRed = PiTraffic.Traffic("NORTH", "RED")
NorthYellow = PiTraffic.Traffic("NORTH", "YELLOW")
NorthGreen = PiTraffic.Traffic("NORTH", "GREEN")

# Bloque funciones
def North_Red_Off():
    NorthRed.off()
def North_Yellow_Off():
    NorthYellow.off()
def North_Green_On():
    NorthGreen.on()

# Bloque acción
North_Red_Off()
North_Yellow_Off()
North_Green_On()
```

n_red_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (n_red_on.py)

import PiTraffic

# Bloque declaración constantes
NorthRed = PiTraffic.Traffic("NORTH", "RED")
NorthYellow = PiTraffic.Traffic("NORTH", "YELLOW")
NorthGreen = PiTraffic.Traffic("NORTH", "GREEN")

# Bloque funciones
def North_Yellow_Off():
    NorthYellow.off()
def North_Green_Off():
    NorthGreen.off()
def North_Red_On():
    NorthRed.on()

# Bloque acción
North_Yellow_Off()
North_Green_Off()
North_Red_On()
```

n_yellow_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (n_yellow_on.py)

import PiTraffic

# Bloque declaración constantes
NorthRed = PiTraffic.Traffic("NORTH", "RED")
NorthYellow = PiTraffic.Traffic("NORTH", "YELLOW")
NorthGreen = PiTraffic.Traffic("NORTH", "GREEN")

# Bloque funciones
def North_Red_Off():
```

```
    NorthRed.off()
def North_Green_Off():
    NorthGreen.off()
def North_Yellow_On():
    NorthYellow.on()

# Bloque acción
North_Red_Off()
North_Green_Off()
North_Yellow_On()
```

s_green_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (s_green_on.py)

import PiTraffic

# Bloque declaración constantes
SouthRed = PiTraffic.Traffic("SOUTH", "RED")
SouthYellow = PiTraffic.Traffic("SOUTH", "YELLOW")
SouthGreen = PiTraffic.Traffic("SOUTH", "GREEN")

# Bloque funciones
def South_Red_Off():
    SouthRed.off()
def South_Yellow_Off():
    SouthYellow.off()
def South_Green_On():
    SouthGreen.on()

# Bloque acción
South_Red_Off()
South_Yellow_Off()
South_Green_On()
```

s_red_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (s_red_on.py)

import PiTraffic

# Bloque declaración constantes
SouthRed = PiTraffic.Traffic("SOUTH", "RED")
SouthYellow = PiTraffic.Traffic("SOUTH", "YELLOW")
SouthGreen = PiTraffic.Traffic("SOUTH", "GREEN")

# Bloque funciones
def South_Yellow_Off():
    SouthYellow.off()
def South_Green_Off():
    SouthGreen.off()
def South_Red_On():
    SouthRed.on()

# Bloque acción
South_Yellow_Off()
South_Green_Off()
South_Red_On()
```

s_yellow_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (s_yellow_on.py)

import PiTraffic

# Bloque declaración constantes
SouthRed = PiTraffic.Traffic("SOUTH", "RED")
SouthYellow = PiTraffic.Traffic("SOUTH", "YELLOW")
SouthGreen = PiTraffic.Traffic("SOUTH", "GREEN")

# Bloque funciones
def South_Red_Off():
    SouthRed.off()
def South_Green_Off():
```

```
    SouthGreen.off()
def South_Yellow_On():
    SouthYellow.on()

# Bloque acción
South_Red_Off()
South_Green_Off()
South_Yellow_On()
```

w_green_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (w_green_on.py)

import PiTraffic

# Bloque declaración constantes
WestRed = PiTraffic.Traffic("WEST", "RED")
WestYellow = PiTraffic.Traffic("WEST", "YELLOW")
WestGreen = PiTraffic.Traffic("WEST", "GREEN")

# Bloque funciones
def West_Red_Off():
    WestRed.off()
def West_Yellow_Off():
    WestYellow.off()
def West_Green_On():
    WestGreen.on()

# Bloque acción
West_Red_Off()
West_Yellow_Off()
West_Green_On()
```

w_red_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (w_red_on.py)

import PiTraffic

# Bloque declaración constantes
WestRed = PiTraffic.Traffic("WEST", "RED")
WestYellow = PiTraffic.Traffic("WEST", "YELLOW")
WestGreen = PiTraffic.Traffic("WEST", "GREEN")

# Bloque funciones
def West_Yellow_Off():
    WestYellow.off()
def West_Green_Off():
    WestGreen.off()
def West_Red_On():
    WestRed.on()

# Bloque acción
West_Yellow_Off()
West_Green_Off()
West_Red_On()
```

w_yellow_on.py

```
# -*- coding: utf-8 -*-

# Web Sala de Control de Tráfico (-_-) Marcos Leirós (w_yellow_on.py)

import PiTraffic

# Bloque declaración constantes
WestRed = PiTraffic.Traffic("WEST", "RED")
WestYellow = PiTraffic.Traffic("WEST", "YELLOW")
WestGreen = PiTraffic.Traffic("WEST", "GREEN")

# Bloque funciones
def West_Red_Off():
    WestRed.off()
def West_Green_Off():
    WestGreen.off()
def West_Yellow_On():
```

```
   WestYellow.on()

# Bloque acción
West_Red_Off()
West_Green_Off()
West_Yellow_On()
```

index.php (protector carpetas vía navegador WEB)

```
<!-- Redirección index.php", By: MaToLeOt -->
<html>
	<head>
		<title>Redirección</title>
	</head>
	<body>
		<?php
			header("Location: http://matoleot.ddns.net/salatrafico");
			die();
		?>
	</body>
</html>
```

index.css (estilo pantalla de acceso)

```
/* Sala de control de tráfico (By: MaToLeOt) index.css */

/* estilos globales ---------------------------------------------- */
 body{
	background-image: url("../img/semaforo-interseccion_1920-1080.jpg");
	background-position: center;
	background-repeat: repeat;
	background-attachment: fixed;
	padding: 3px 3px 3px 3px;
	background-color: #d8dbe2;
	font-family: Verdana, sans-serif;
	font-weight:bold;
	font-size: 13pt;
	text-align: center;
 }
h3{
	margin-top:4px;
	margin-bottom:4px;
}
div#wrapper{
	text-align: center;
 }
/* ------------------------------------------------------------------------ */

/* eslilo login ------------------------------------------------------- */
div#secondary{
	background-color: rgba(0, 0, 0, 0.8);
	border-radius:8px;
	padding:70px 100px;
}
div#secondary span.text-center{
	color:#fff;
	font-size: 23px;
	/*margin: 25px 0 80px 0;*/
	display: block;
	text-align: center;
}
div#secondary div.input-container{
	position:relative;
	margin-bottom:25px;
}
div#secondary div.input-container label{
	position:absolute;
	top:0px;
	left:0px;
	font-size:16px;
	color:#fff;
	pointer-event:none;
	transition: all 0.5s ease-in-out;
}
div#secondary div.input-container input{
	border:0;
	border-bottom:1px solid #555;
	background:transparent;
	width:100%;
	padding:8px 0 5px 0;
	font-size:16px;
	color:#fff;
```

```
}
div#secondary div.input-container input:focus{
	border:none;
	outline:none;
	border-bottom:1px solid #e74c3c;
}
div#secondary button.btn{
	color:#fff;
	background-color:#e74c3c;
	outline: none;
	border: 0;
	color: #fff;
	padding:10px 20px;
	margin-top:50px;
	border-radius:2px;
	cursor:pointer;
	position:relative;
}
div#secondary div.input-container input:focus ~ label,
div#secondary div.input-container input:valid ~ label{
	top:-14px;
	font-size:12px;
}
/* ------------------------------------------------------------------------ */

/* RWD generales para resoluciones PC ------------------*/
@media screen and (min-width: 621px) {
	div#wrapper{
		width:80%;
		margin:20px auto 20px auto;
	}
	div#primary, div#secondary{
		margin: 0px auto 0px auto;
	}
	div#primary{
		vertical-align:top;
		border-radius:8px;
		display:inline-block;
		background-repeat: no-repeat;
		background:url("../img/pitraffic_logo.png");
		background-size:cover;
	}
	div#secondary{
		display:inline-block;
	}
	span{
		display:block;
		padding-top:30px;
	}
}
/* ------------------------------------------------------------------------ */

/* RWD modo pc escritorio alta resolución ----------------*/
@media screen and (min-width: 1601px){
	div#primary{
		width:550px;
		height:550px;
	}
	div#secondary{
		width:350px;
		height:410px;
	}
	div#secondary span.text-center{
		margin: 25px 0 80px 0;
	}
 }
/* ------------------------------------------------------------------------ */

/* RWD modo pc escritorio resolución media -----------*/
@media screen and (min-width: 1367px) and (max-width: 1600px) {
	div#primary{
		width:450px;
		height:450px;
	}
	div#secondary{
		width:250px;
		height:310px;
	}
	div#secondary span.text-center{
		margin: 0px 0 50px 0;
	}
 }
/* ------------------------------------------------------------------------ */

/* RWD modo pc escritorio resolución baja ---------- */
 @media screen and (min-width: 621px) and (max-width: 1366px) {
```

```
	div#primary{
		width:350px;
		height:350px;
	}
	div#secondary{
		width:150px;
		height:210px;
	}
	div#secondary span.text-center{
		margin: -70px 0 50px 0;
	}
}
/* ------------------------------------------------------------------------ */

/* RWD modo smartphone ---------------------------------- */
@media screen and (max-width: 620px){
	div#wrapper{
		width:100%;
		margin:3px auto 3px auto;
	}
	div#primary{
		width:auto;
	}
	div#secondary{
		width:auto;
	}
	div#secondary span.text-center{
		margin: -70px 0 50px 0;
	}
	h3{
		font-size: 13pt;
		margin-bottom:8px;
	}
	span{
		display:block;
		padding-top:12px;
	margin-bottom:16px;
	}
}
/* ------------------------------------------------------------------------ */
```

login.css (estilos autenticación)

```
/* Sala de control de tráfico (By: MaToLeOt) login.css */

/* estilos generales ------------------------------------------------ */
body{
	background-image: url("../img/semaforo-interseccion_1920-1080.jpg");
	background-position: center;
	background-repeat: repeat;
	background-attachment: fixed;
	padding: 3px 3px 3px 3px;
	background-color: #d8dbe2;
	font-family: Verdana, sans-serif;
	font-weight:bold;
	font-size: 13pt;
	text-align: center;
}
img#prohibido{
	width:35%;
	height:35%;
}
/* ------------------------------------------------------------------------ */
```

salatrafico.css (estilo página control)

```
/* Sala de control de tráfico (By: MaToLeOt) salatrafico.css */

/* Bloque estilos general ------------------------------------------ */
body{
	padding: 3px 3px 3px 3px;
	font-family: Verdana, sans-serif;
	font-weight:bold;
	font-size: 12pt;
	text-align:center;
}
h3{
	margin-top: -8px;
	margin-bottom: 8px;
}
table, th, td{
	border: 2px solid black;
```

```
		border-collapse: collapse;
		margin-left:auto;
		margin-right:auto;
		background-color: #d8dbe2;
 }
span.semaforonseo{
		color:grey;
 }
/* ---------------------------------------------------------------------- */
 div#semaforos div#semN, div#semaforos div#semS, div#semaforos div#semE, div#semaforos div#semW{
		display:inline;
 }
 canvas#semN, canvas#semS, canvas#semE, canvas#semW{
	margin-right:15px;
		margin-top:20px;
		margin-bottom:20px;
 }
/* ---------------------------------------------------------------------- */
div#dselect{
		display: inline;
		float:left;
 }
select.stiempo{
		margin-left:2px;
		margin-right:5px;
 }
/* ---------------------------------------------------------------------- */
div#total{
		display: inline;
		float:left;
		text-align:left;
}
div#wrappercheks{
		display:inline;
		float:left;
}
input[type='checkbox'].pie{
		margin-left:2px;
		margin-right:8px;
 }
input[type='submit']{
		margin-top: 6px;
 }
div#imgbuzzer{
		display:inline;
		float:center;
 }
img#ibuzz{
		width:46px;
		height:46px;
		margin-top:3px;
		visibility:hidden;
 }
/* ---------------------------------------------------------------------- */
input[type='radio'], input[type='checkbox'], input[type='submit'], select.stiempo{
		cursor: pointer;
 }
/* ---------------------------------------------------------------------- */

/* RWD modo pc escritorio alta resolución ----------*/
@media screen and (min-width: 1601px){
		body{
			background-image: url("../img/semaforo-interseccion_1920-1080.jpg");
			background-position: center;
			background-repeat: repeat;
			background-attachment: fixed;
		}
		table{
			transform: scale(1.55, 1.55);
			transform-origin: center 0 0;
			box-shadow: 10px 10px 10px;
		}
 }
/* ---------------------------------------------------------------------- */

/* RWD  modo pc escritorio resolución media ----------*/
@media screen and (min-width: 1281px) and (max-width: 1600px) {
		body{
			background-image: url("../img/semaforo-interseccion_1920-1080.jpg");
			background-position: center;
			background-repeat: repeat;
			background-attachment: fixed;
		}
		table{
			transform: scale(1.04, 1.04);
			transform-origin: center 0 0;
```

```
                box-shadow: 6px 6px 6px;
        }
}
/* ---------------------------------------------------------------------- */

/* RWD modo pc escritorio resolución baja ------------ */
@media screen and (min-width: 621px) and (max-width: 1280px) {
        body{
                background-image: url("../img/semaforo-interseccion_1920-1080.jpg");
                background-position: center;
                background-repeat: repeat;
                background-attachment: fixed;
        }
        table{
                transform: scale(0.9, 0.9);
                transform-origin: center 0 0;
                box-shadow: 3px 3px 3px;
        }
}
/* ---------------------------------------------------------------------- */

/* RWD modo smartphone ------------------------------------ */
@media screen and (max-width: 620px){
        body{
                background-image: none;
                background-color:green;
        }
}
/* ---------------------------------------------------------------------- */

/* Bloque estilos checkbox automan ---------------------- */
/* Formateamos el label que servirá de contenedor */
.container1 {
        width: auto;
        display:inline;
        margin-left:-8px;
}
.switch {
        position: relative;
        display: inline-block;
        width: 53px;
        height: 25px;
        top:5px;
}
/* Ocultamos el checkbox html */
.switch input {
        display:none;
}
/* Formateamos la caja del interruptor sobre la cual se deslizará la perilla de control */
.slider {
        position: absolute;
        cursor: pointer;
         top: 0;
        left: 0;
        right: 0;
        bottom: 0;
        background-color: #ccc; /* color fondo */
        -webkit-transition: .4s;
        transition: .4s;
}
/* Pintamos la perilla de control o slider usando el selector before */
.slider:before {
        position: absolute;
        content: "";
        height: 24px;
        width: 24px;
        left: 0px;
        bottom: -0.5px;
        background-color: #000;
        -webkit-transition: .4s;
        transition: .4s;
}
/* Cambiamos el color de fondo cuando el checkbox esta activado */
input:checked + .slider {
        background-color: #E27AD8;
}
/* Deslizamos el slider a la derecha cuando el checkbox esta activado */
input:checked + .slider:before {
        -webkit-transform: translateX(27px);
        -ms-transform: translateX(27px);
        transform: translateX(27px);
}
/* Aplicamos efecto de bordes redondeados en slider y en el fondo del slider */
.slider.round {
        border-radius: 34px;
        border: 1px solid blue;
```

```
}
.slider.round:before {
	border-radius: 50%;
}
/* ---------------------------------------------------------------------- */

/* Bloque estilos reset  ---------------------------------------- */
.container2 {
	width: auto;
	display:inline;
	margin-left:40px;
}
.btn {
	cursor: pointer;
	margin: 20px auto;
	border: none;
	padding: 0px 2px;
	font-size: 19px;
	position: relative;
	background: #ccc;	/* color fondo */
	border: 1px solid blue;
}
.btn::before {
	transition: all 0.85s cubic-bezier(0.68, -0.55, 0.265, 1.20);
	content: "";
	width: 50%;
	height: 100%;
	background: black;
	position: absolute;
	top: 0;
	left: 0;
}
.btn:hover::before {
	background: black;
	width: 100%;
}
.btn.rounded {
	border-radius: 50px;
}
.btn.rounded .text-green {
	color:#00f0b5;		/* color texto interno con cursor encima */
	mix-blend-mode: difference;
}
.btn.rounded::before {
	border-radius: 50px;
	width: 48%;
	background: #000;			/* color círculo*/
}
.btn.rounded:hover::before {
	background: #E27AD8;		/* color rosado */
	width: 100%;
}
/* ---------------------------------------------------------------------- */

/* Bloque estilos radio RAV ---------------------------------- */
/* Contenedores */
.container{width: 300px; margin: 0 auto;}
.four{ width: 32%;}
/* Columnas */
.col{
	display: block;
	float:left;
	margin: 1% 0 1% 1.6%;
}
.col:first-of-type {margin-left: 0;}

/* Formulario */
.form input{
	display: none;
}
.form label{
	position: relative;
	color: #000;
	background-color: #aaa;
	font-size: 16px;
	text-align: center;
	height: 25px;
	line-height: 15px;
	display: block;
	cursor: pointer;
	border: 3px solid transparent;
	-webkit-box-sizing: border-box;
	-moz-box-sizing: border-box;
	box-sizing: border-box;
}
.radiosem:checked  + label{
```

```
		border: 3px solid #333;
}
#norter:checked  + label, #surr:checked  + label, #ester:checked  + label, #oester:checked  + label{
		background-color: #dc0000;
}
#nortea:checked + label, #sura:checked + label, #estea:checked + label, #oestea:checked + label{
		background-color: #ffc800;
}
#nortev:checked + label, #surv:checked + label, #estev:checked + label, #oestev:checked + label{
		background-color: #00aa00;
}
/* ------------------------------------------------------------------------------ */
```

Marcos Tomás Leirós Otero