Sistema de entrada/salida de un aparcamiento *Wireless*

Raúl Palencia Bastida

Resumen — Este proyecto se ha desarrollado para solventar la problemática encontrada en el parking de la EE. Uno de los objetivos de este proyecto ha sido el facilitar la toma de decisiones a los usuarios de un parking. Usando la metodología apropiada y la planificación de las tareas durante el timing establecido. A través de un par de sensores y una barrera, automatizaremos la entrada/salida al parking y conoceremos el estado de las plazas libres. Gracias a la motivación generada por el desarrollo de sistemas basados en microcontroladores hemos podido superar los diferentes tipos de adversidades que nos hemos ido encontrando en el camino alcanzando los diferentes objetivos que nos planteamos al inicio del proyecto.

Palabras clave—Arduino Yún, Librería bridge, Sensor ultrasonido, Servo motor.

Abstract—This project has been developed to solve the found problems in the EE parking. One of the objective of this project has been to facilitate the user's decisions making in a parking. Using an appropriate methodology and the task planning during the established timing. Through a couple of sensors and a barrier, we automated the entrance/exit of the parking and we will know the free places status. Thanks to the motivation by the system development based on microcontrollers, we have been able to overcome adversity that we have been encountering along the way reaching objectives that we set ourselves at the beginning of the project.

Index Terms—Arduino Yun, Bridge library, Servo engine, Ultrasonic sensor.

1 Introducción

ESTE proyecto, titulado Sistema de entrada/salida de un aparcamiento *Wireless,* nace de la problemática que existe en el parking de la *Escola d'Enginyeria* de la UAB. Concretamente en el parking interior, donde aparca el personal autorizado de la EE. Al ser un parking con una forma peculiar es difícil saber el número de plazas libres que hay en el parking, esto implica que el usuario debe recorrer todo el parking en búsqueda de una plaza libre y si no hay retroceder marcha atrás para salir del parking. En la siguiente Figura 1 mostramos la peculiar forma de parking:

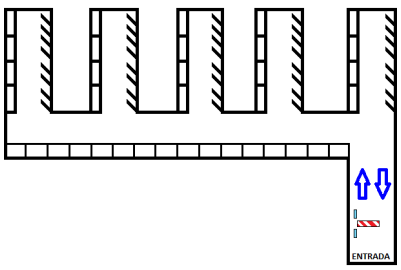

Figura 1 – Parking EE

Para resolver dicha problemática diseñamos un sistema formado por un módulo de hardware y un módulo de software. El sistema que hemos desarrollado es un prototipo el cual podrá ser escalado si se presentara la ocasión de desarrollar un proyecto real.

En este informe encontramos los aspectos más importantes que hemos desarrollado durante el trabajo final de grado de la mención de *Enginyeria del Software*.

El desarrollo del proyecto ha sido planteado usando la metodología en cascada, planificando las diferentes tareas a lo largo del timing establecido.

El desarrollo de este proyecto es motivado por el poco conocimiento y las ganas de trabajar con el concepto *Internet of Things*. Este concepto está basado en el uso de componentes que hace posible, a través de la comunicación con un servidor, conocer el estado en el que se encuentra el sistema.

Este documento consta de las siguientes partes: los objetivos definidos para el proyecto. Además, veremos el estado del arte, en la que explicaremos el estado de este tipo de sistemas, o parecidos, en el mercado inspirándonos en la elección del nuestro. Seguidamente, explicaremos las fases que hemos utilizado a lo largo del proyecto y la planificación realizada para llevar a cabo las diferentes tareas.

Finalmente, expondremos los diferentes resultados que hemos obtenido al finalizar el proyecto después de haber superado las diferentes adversidades que nos hemos encontrado. Explicaremos las diferentes conclusiones que hemos obtenido y, finalmente, plantearemos una serie de líneas futuras que nos permitirían escalar el proyecto y solucionar los problemas encontrados durante el desarrollo.

- *E-mail de contacto: raul.palencia@e-campus.uab.cat*
- *Mención realtzada: Enginyeria del Software.*
- *Proyecto tutorizado per: Marta Prim Sabrià (Dep. Microelectrònica i Sistemes electrònics)*
- *Curs 2016/17*

2 OBJETIVOS

En este apartado, expondremos los diferentes objetivos que se definieron al principio del proyecto. Los objetivos son:

1- Conocer el estado del aparcamiento sin estar en él.
2- Reducir el tiempo de espera al abrir la barrera.[1]
3- Ayudar al usuario en la toma de decisiones.[2]
4- Mejorar la capacidad de resolver los problemas que puedan aparecer en el desarrollo del proyecto.
5- Aumentar el conocimiento en el desarrollo de software con diferentes sensores a partir de la programación de Arduino.
6- Tener todo listo antes 27 de junio de 2017.

Los tres primeros objetivos están centrados más a la parte del usuario. Gracias al conocimiento de la programación podrán ser alcanzados.

Seguidamente, nos encontramos con los objetivos 4 y 5. Estos se centran más en el estudiante ya que, resolviendo los diferentes problemas que nos podamos encontrar obtendremos experiencia para un proyecto futuro.

Finalmente, nos encontramos con el último objetivo, centrado en la correcta planificación con el fin de marcarnos una fecha de entrega final.

2.1 Prioridad de los objetivos

Una vez definidos los objetivos del proyecto, les asignamos una prioridad con la intención de conocer la importancia que tenían cada uno de ellos. En la Tabla 1, se presentan los objetivos con su correspondiente estado de prioridad según la importancia de cada uno de ellos:

Tabla 1 – Prioridad objetivos

Objetivos	Prioridad
Conocer el estado del aparcamiento sin estar en él.	Crítico
Reducir el tiempo de espera al abrir la barrera	Baja
Ayudar al usuario en la toma de decisiones.	Crítico
Capacidad de resolver los problemas que puedan aparecer en el desarrollo del proyecto.	Alta
Aumentar el conocimiento en el desarrollo de software con diferentes sensores a partir de la programación de Arduino.	Alta
Realizar una interfaz web intuitiva y fácil de utilizar.	Baja
Antes del 28 de junio de 2017.	Crítico

[1] Actualmente se hace mediante la aproximación de una tarjeta, ésta es reconocida por el sistema y abre la barrera.

[2] Si sabemos que el *parking* no tiene plazas de aparcamiento, ayudaremos a que el usuario tome la decisión de aparcar en otro lugar.

3 ESTADO DEL ARTE

Actualmente, cuando nos desplazamos en coche a cualquier centro comercial no nos hemos de preocupar de donde vamos a aparcar, ya que éstos disponen de estacionamiento gratuito. Un claro ejemplo sería el centro comercial la Maquinista. La Maquinista cuenta con un sistema de aparcamiento el cual informa a los usuarios del número de plazas libres de cada una de las filas. Como se puede ver en la Figura 2 La Maquinista cuenta con 4 zonas de aparcamiento, un total de 5500 plazas repartidas en dos plantas.

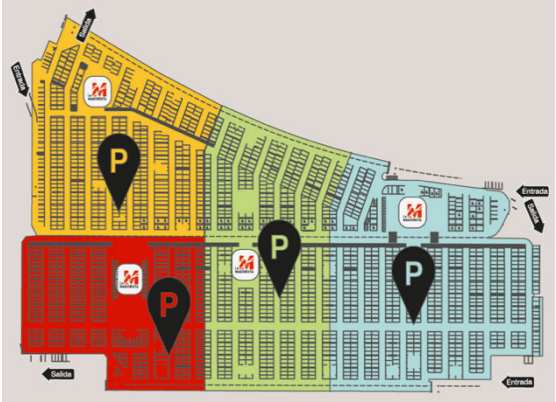

Figura 2 – Zonas de aparcamiento de la Maquinista

Dependiendo de la entrada que escojamos cada dos filas de aparcamiento, nos encontramos con una pantalla informándonos de las plazas disponibles para estacionar el coche. A demás, cada una de las plazas de aparcamiento posee un sensor el cual al detectar el coche cambia el color del led. Al cambiar de color a verde o a rojo actualizará el número de plazas de aparcamiento disponibles en la pantalla.

Podemos decir que el coste de construcción de un parking y añadir el sistema de cableado y de sensores necesario es mucho más costoso que el sistema que se propone en nuestro proyecto.

Por eso después de contactar con diferentes empresas hemos podido saber el coste que supondría añadir solamente el sistema de aparcamiento a un número de plazas inferior al que tiene el centro comercial anterior. Para un parking con 50 plazas de aparcamiento, añadir los sensores necesarios hace que el presupuesto sea de unos 8000€. Dependiendo de la empresa este precio variaba. Este presupuesto incluye el cableado de los sensores y las pantallas de información de las plazas libres del aparcamiento.

Por otro lado, sabiendo que nuestro sistema va a ser diferente, ya que va a controlar las plazas de parking según el número de coches que entren i salgan del parking, nuestro presupuesto lo podemos reducir. Siendo menos complejo y más fácil de instalar que los actuales ya que no necesitará ni tanto cableado ni tantos sensores.

4 METODOLOGÍA

La metodología de trabajo utilizada en el desarrollo de nuestro proyecto ha sido el modelo en Cascada. La razón por la que hemos elegido este modelo para nuestro proyecto es porque ha sido desarrollado de manera individual, es decir, no podremos empezar la siguiente tarea mientras que no terminemos con la tarea en la que estamos trabajando. Aunque sea una de las metodologías más sencillas, este modelo utiliza las fases adecuadas para nuestra planificación.

En un principio las fases [1] de la metodología en cascada son:

1- Análisis de Requisitos.
2- Diseño del Sistema.
3- Diseño del Programa.
4- Codificación.
5- Ejecución de Pruebas.
6- Verificación.
7- Mantenimiento.

A las diferentes fases anteriores se han añadido nuevas tareas. En el WBS[3] definimos estas tareas que hemos incluido en nuestro modelo en cascada. A continuación, veremos el WBS en la Figura 3:

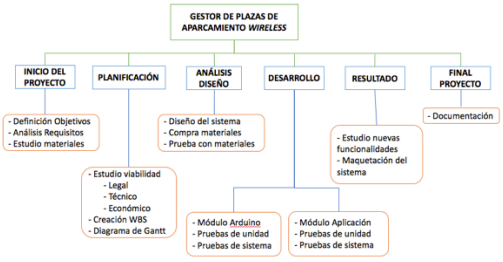

Figura 3 – Work Breakdown Structure

5 PLANIFICACIÓN

En la siguiente tabla veremos cómo se ha llevado a cabo la planificación con el objetivo de finalizar el trabajo de manera correcta y en la fecha prevista. Según la metodología elegida del modelo en cascada no podremos empezar otra tarea hasta que la tarea actual no sea terminada. La siguiente tabla 2 presenta la planificación que se ha ido siguiente mientras el proyecto se desarrollaba.

Tabla 2 – Planificación Proyecto

Tareas	Fecha Inicio	Fecha Fin
Definición objetivos	07/02/2017	27/02/2017
Análisis de requisitos	28/02/2017	06/03/2017
Estudio de materiales	07/03/2017	13/03/2017
Diseño del sistema	07/03/2017	13/03/2017
Compra de materiales	14/03/2017	30/03/2017
Prueba con los materiales	31/03/2017	06/04/2017
Desarrollo del módulo Arduino	07/04/2017	27/04/17
Pruebas Unitary Test del módulo Arduino	07/04/2017	27/04/17
Pruebas de sistema módulo Arduino	10/04/2017	28/04/2017
Desarrollo App	01/05/2017	01/06/2017
Pruebas Unitary Test App	01/05/2017	01/06/2017
Pruebas de sistema App	08/05/2017	14/06/2017
Estudio nuevas funcionalidades	15/06/2017	19/06/2017
Documentación	07/02/2017	27/06/2017

6 DESCRIPCIÓN DEL PROYECTO

En el este apartado explicaremos las diferentes partes del proyecto. Nos centraremos en los dos módulos que forman nuestro sistema. Describiremos brevemente los componentes utilizados en el desarrollo de nuestro sistema y el coste que ha supuesto la creación de éste.

6.1 Módulo de hardware

En un principio, nuestro subsistema iba a tener dos barreras y un 7-segmentos con dos dígitos, pero, finalmente, reducimos a una barrera que hará la funcionalidad de entrada-salida, y un 7-segmentos con un solo dígito. A continuación, veremos, en la Figura 4, el módulo de hardware final:

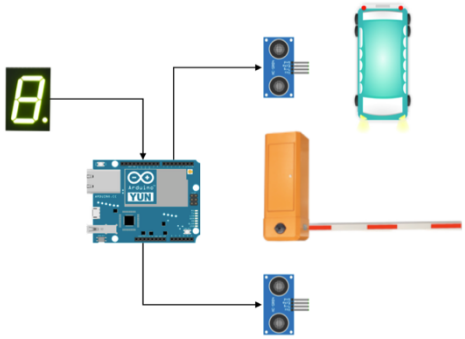

Figura 4 – Sistema de hardware

[3] Work Breakdown Structure

El funcionamiento será el mismo, lo único que cambiará será que pasaremos a tener una entrada/salida y un 7-segmentos con un solo dígito. Cuando uno de los dos sensores reconozca un coche, el Arduino Yún será el encargado de abrir la barrera y mostrar el número de plazas disponibles que hay en el parking.

Las razones para estos pequeños cambios han sido:

- Escasez de pines digitales para los componentes.
- Al ser un prototipo nos hemos decantado finalmente por solo un dígito.

En la Tabla 3 listamos los componentes que hemos usado para el desarrollo de nuestro proyecto:

Tabla 3 – Componentes del sistema

Componente	Cantidad
Arduino Yún	1
Servo Motor	1
Sensor ultrasonido HC-SR04	2
7-Segmentos simple	1

Sabiendo las diferentes cantidades de componentes que hemos utilizado para la implementación del módulo hardware, ahora nos vamos a centrar en el número de pines de cada uno de los componentes que hemos usado para el desarrollo de nuestro proyecto:

Tabla 4 – Pines usados por los componentes

Componente	Cantidad	Pines Totales	Pines Usados
Arduino Yún	1	-	-
Servo Motor	1	3 (Vcc, GND y PWM)	1
Sensor ultrasonido HC-SR04	2	4x2 (Vcc0, Trigg0, Echo0, GND0, Vcc1, Trigg1, Echo1 y GND1)	4 (Trigg0 y Echo0, Trigg1 y Echo1)[4]
7-Segmentos simple	1	10	7

La cantidad de pines digitales asignados en el Arduino Yún son 14, de estos vamos a usar del 2 al 13, sin tener en cuenta pines 0 y 1[5].

Para ahorrar pines, hemos puesto en común aquellos pines que utilizan los diferentes componentes y se pueden unificar, tales como Vcc y GND[6]. Este tipo de componentes solo usa pines digitales, no como otros componentes como, por ejemplo, el sensor de temperatura que usa los pines analógicos para la recogida de la temperatura. Por lo tanto, no podríamos usar los pines digitales para este sensor ya que solo pueden tener dos valores HIGH o LOW.

El otro problema fue al recibir los 7-Segmentos de doble dígito. Buscamos la información correspondiente por diferentes fuentes de información. Y vimos que, al usar dos dígitos, el número de pines a utilizar era más alto que el de solo un dígito. Finalmente, nos decidimos por comprar un nuevo 7-segmentos, pero esta vez, con un solo dígito con la ventaja de decrementar el número de pines, pero incrementando el gasto en materiales del proyecto. La Tabla 5 presenta el coste total del sistema:

Tabla 5 – Coste del sistema

Componente	Cantidad	Precio Unitario	Precio Total
Arduino Yún	1	58€	58€
Servo Motor	2	1,5€	3€
Sensor ultrasonido HC-SR04	2	1,8	3,6€
7-Segmentos simple	2	2,5€	5€
Total	**7**		**69,6€**

Aunque el módulo de hardware haya sufrido cambios en el número de componentes, los requisitos no se han visto influenciados. Seguidamente, veremos en la Tabla 6 los requisitos que están implementados en el módulo de hardware:

Tabla 6 – Requisitos hardware

ID	Requisito	Estado
MOD_HW_SYS_REQ_1	El sistema deberá encontrarse en la posición inicial con la barrera en horizontal cuando llegue un coche.	✓
MOD_HW_SYS_REQ_2	El sistema debe detectar un coche cuando se aproxime a la barrera correspondiente.	✓
MOD_HW_SYS_REQ_3	El sistema deberá subir la barrera cuando un coche sea detectado.	✓
MOD_HW_SYS_REQ_4	El sistema deberá descontar una plaza de aparcamiento cuando el sensor0 haya subido la barrera y ésta haya vuelto a su posición inicial.	✓
MOD_HW_SYS_REQ_5	El sistema deberá incrementar una plaza de aparcamiento cuando el sensor1 haya subido la barrera y ésta haya vuelto a su posición inicial.	✓
MOD_HW_SYS_REQ_6	El sistema deberá permanecer abierto siempre que un coche esté parado en frente de él.	✗

[4] Hay dos sensores, sensor0 y sensor1, se han definido como 0 y 1 para las señales de los diferentes sensores.
[5] Utilizados para recibir y transmitir a través del puerto serie ATmega32U4.
[6] Vcc es 5v y GND es tierra.

MOD_HW_SYS_REQ_7	El sistema tendrá un timeout de 30 segundos si el coche no entra.	✓
MOD_HW_SYS_REQ_8	El sistema deberá diferenciar entre un hombre y una persona.	✗
MOD_HW_SYS_REQ_9	El sistema no abrirá la barrera una vez que el parking esté lleno.	✓

Una vez definidos los diferentes requisitos, se desarrolló un diagrama de caso de uso (Figura 5), el cual nos detallara el funcionamiento del sistema.

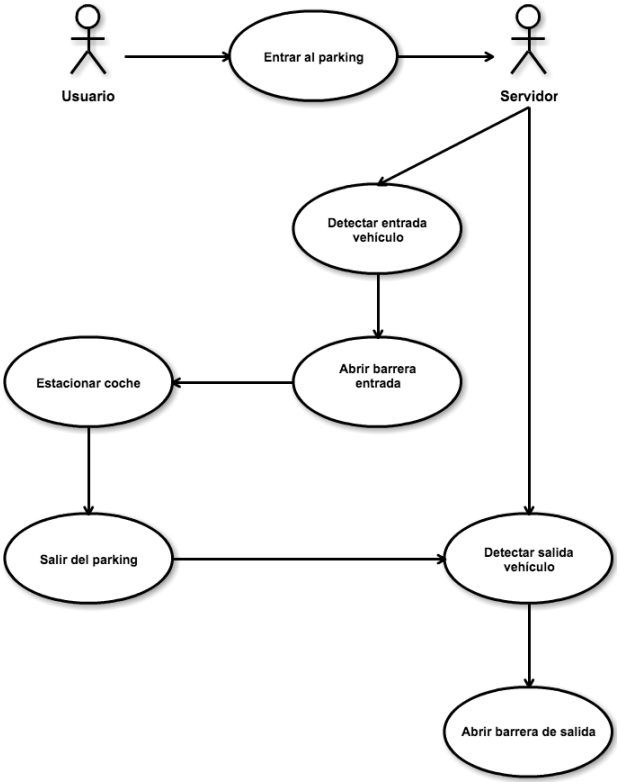

Figura 5 – Caso de uso módulo hardware

6.2 Módulo de software

La razón por la que hemos elegido el Arduino Yún es porque es un híbrido entre un Arduino Uno y una Raspberry Pi, teniendo como ventajas la sencillez a la hora de programar, siendo igual que su hermano pequeño Uno y además contamos con la potencia del Linux[7].

Este sistema operativo será el que albergará la web accesible para los usuarios que quieran saber el estado del parking. La web estará disponible conectándonos a nuestro Arduino Yún vía navegador móvil.

Sin embargo, también nos encontramos con alguna desventaja, como la capacidad de almacenamiento del Arduino Yún. Tiene una memoria ROM de 16 Megabytes, 9 de ellos son utilizados por el sistema operativo Linux.

Por lo tanto, nos encontramos que si queremos hacer una web que contenga alguna imagen no debe pesar más de 7 Megabytes. Sabiendo que las imágenes de hoy en día tienen una calidad bastante alta, significa que no podemos tener una web con bastantes imágenes, por lo cual seguimos el tutorial de Arduino Community [2] para determinar cómo obtener más espacio interno. Una vez seguido el tutorial anterior, podremos cargar los sketch e imágenes de mayor tamaño.

Seguidamente, la Figura 6 se muestra como está estructurado el módulo de software:

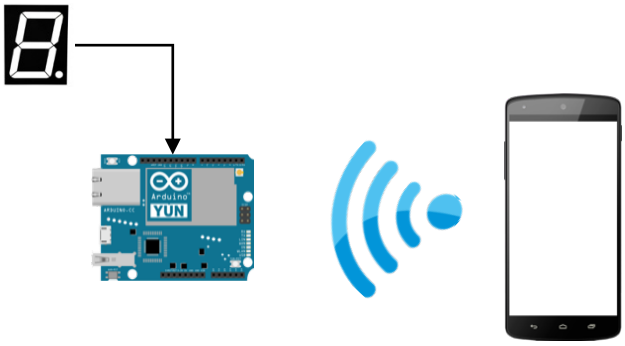

Figura 6 – Sistema software

A través de un dispositivo móvil nos podremos conectar a la página web la cual nos informará de las plazas disponibles.

A continuación, se muestra un mockup de la página web en la Figura 7:

Figura 7 – Mockup web

La página web será muy simple tal y como hemos visto en la figura anterior. Consta de: una imagen, un título y una tabla la cual nos mostrará la fecha, la hora y las plazas disponibles cumpliendo así con los requisitos definidos al principio del proyecto. La Tabla 7 se indica el estado de implementación de los requisitos del módulo software:

[7] El linux instalado en el Arduino Yún es Linino OS.

Tabla 7 – Requisitos Software

ID	Requisito	Estado
MOD_SW_SYS_REQ_1	El sistema estará conectado vía WIFI.	✓
MOD_SW_SYS_REQ_2	El sistema no requerirá ni usuario ni contraseña.	✓
MOD_SW_SYS_REQ_3	El sistema accederá a la URL del servidor.	✓
MOD_SW_SYS_REQ_4	El sistema se refrescará automáticamente sin necesidad de interacción.	✗
MOD_SW_SYS_REQ_5	El sistema estará comprendido por una tabla con los campos necesarios.	✓
MOD_SW_SYS_REQ_6	El sistema deberá dar la fecha en formato DD/MM/YYY.	✓
MOD_SW_SYS_REQ_7	El sistema deberá dar la hora en formato hh:mm:ss.	✓
MOD_SW_SYS_REQ_8	El sistema deberá informar del número de plazas disponibles.	✓
MOD_SW_SYS_REQ_9	El sistema deberá informar de las diferentes respuestas comunes de estado: -200 → respuesta correcta del servidor. -404 → página no encontrada. -500 → servicio no disponible.	✓
MOD_SW_SYS_REQ_10	El sistema deberá cambiará el color del número de plazas de parking según el estado.	✓

Después de haber definido los distintos requisitos para el sistema de software, se desarrolló un caso de uso para este módulo como veremos en la Figura 8:

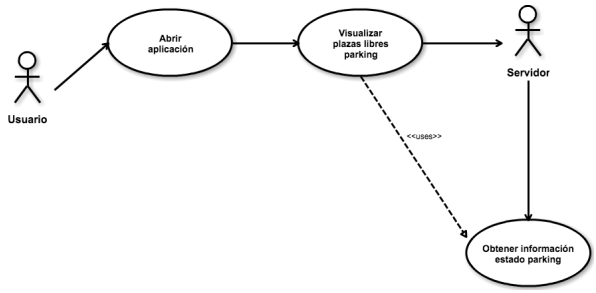

Figura 8 – Caso de uso software

La comunicación con el servidor se ha hecho a partir de la librería Bridge de Arduino.

El Arduino Yún tiene dos procesadores. Uno es el ATmega32U4 y el otro es el Atheros 9331, donde alberga el Linux. Esta combinación permite ejecutar programas o scripts en el sistema Linux, conectando el Arduino a varios servicios de internet.

La librería Bridge, como veremos en la Figura 9, simplifica la comunicación entre los dos procesadores. Los comandos de esta librería son interpretados por el lenguaje Python en el procesador AR933, permitiendo así una comunicación bidireccional, actuando como interfaz de la línea de comandos de Linux.

Gracias a la implementación de esta librería será posible conocer el estado del parking a través de la página web.

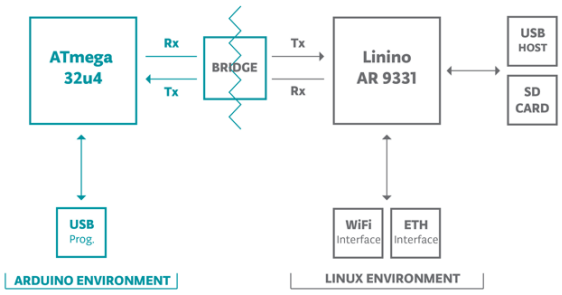

Figura 9 – Comunicación ATmega32u4 y Linino

Una vez explicado el funcionamiento de la librería bridge, mostraremos el proceso de funcionamiento de nuestro sistema.

En la siguiente Figura 10 se muestra el funcionamiento de nuestro sistema:

Figura 10 – Comunicación Arduino - web

7 HERRAMIENTAS

En este apartado explicaremos las diferentes herramientas que hemos utilizado durante el desarrollo del proyecto. Las diferentes herramientas que hemos utilizado han sido elegidas porque teníamos cierta experiencia de uso.

7.1 Módulo hardware

Este módulo está formado principalmente por el Arduino y sus componentes.

Las herramientas que hemos utilizado han sido:

1. **Arduino IDE:** es el entorno de programación oficial de Arduino, que facilita la creación del código y el poder cargarlo a la placa.

2. **Bitbucket:** repositorio en el que guardamos el código del Arduino. Nuestra manera de trabajar ha sido crear una rama master y para cada una de las funcionalidades hemos creado una rama. Por ejemplo, para la implementación del 7-segmentos creamos una rama con ese mismo nombre y una vez terminada la funcionalidad, hicimos las pruebas necesarias y finalmente, la fusionamos con la master.

7.2 Módulo de software

Básicamente, será la página web la cual integrará: html[8], css[9] y las imágenes que queramos cargar en la web. Las herramientas utilizadas son:

1. **Sublime Text 2:** editor de texto con el que se puede programar en una amplia cantidad de lenguajes.

2. **SFTP:** plugin de la herramienta anterior, la cual nos deja conectarnos al servidor y poder modificar el código.

3. Al usar dos tipos de sistemas operativos. Elegimos las siguientes aplicaciones para la conexión ssh con el Arduino Yún:

   a. **Terminal**: viene preinstalado en MAC OS.

   b. **Putty**: aplicación para Windows.

4. **CyberDuck**: herramienta para transferir archivos al Arduino, se puede utilizar tanto en MAC OS como en Windows.

Y, por último, comentar que para la gestión de las tareas hemos usado la siguiente aplicación **Trello.** La Figura 11 se puede observar cómo hemos estructurado el tablero que creamos para el proyecto:

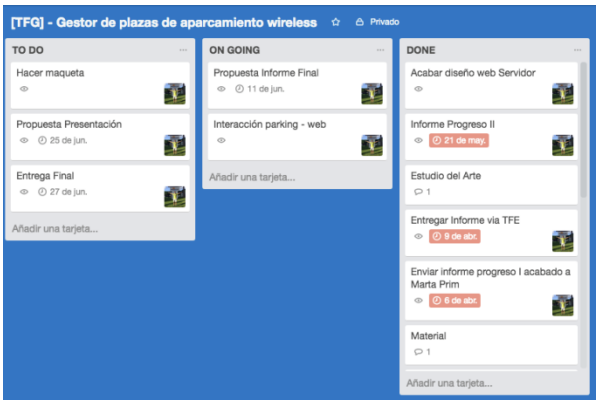

Figura 11 – Gestión de tareas

En la figura anterior vemos que tenemos tres columnas: TO DO, ON GOING y DONE; dependiendo del estado de las tareas, éstas cambian de una columna a otra. Otra de las ventajas de esta herramienta era la de establecer una fecha de finalización a la tarea.

8 TEST Y PRUEBAS

En este apartado veremos los diferentes test que hemos hecho para nuestro sistema.

Se han definido unos test para la validación del sistema. El proceso ha sido el siguiente: crear un Entry Criteria[10] para la versión del sistema. En la Figura 12 veremos los test se han diferenciado en dos tipos: pruebas unitarias y pruebas sistema.

Las pruebas unitarias han sido desarrolladas a la vez que íbamos implementando nuestro sistema, concretamente en la parte de hardware utilizando la librería ArduinoUnit. Por otro lado, las pruebas de sistema han sido desarrolladas con el objetivo de satisfacer los requisitos de sistema que hemos visto anteriormente.

DVP0003	**Prueba Unitaria**	Prueba unitaria no texto en la petición del servidor.	**OK**
DVP0004	**Prueba Unitaria**	Prueba unitaria de la función loop.	**OK**
DVP0005	**Prueba Sistema**	Conectividad de 10 dispositivos.	**OK**
DVP0006	**Prueba Sistema**	Levantar barrera cuando la distancia es más pequeña de 5 cm.	**OK**

Figura 12 – Pruebas realizadas

[8] Código de tags para la creación de las páginas webs.
[9] Hoja de estilo del código html.

[10] Listado de test que define el líder del equipo de test y validación y que software lo ejecuta cada vez que compilan una release de software con el objetivo de ofrecer un mínimo de calidad al equipo de validación.

A continuación, veremos cómo se realizó la prueba de sistema para saber si se enviaba correctamente el número de correcto de plazas de parking libres. En la Figura 13, vemos el estado inicial del parking:

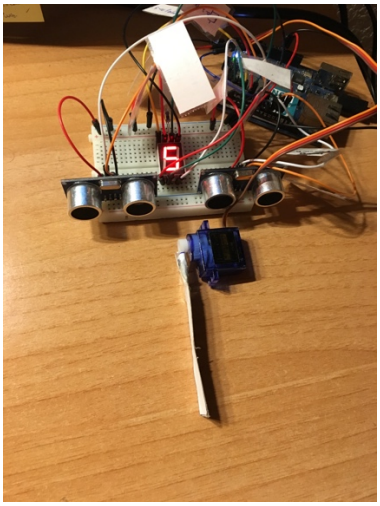

Figura 13 – Estado inicial

En la siguiente Figura 14, vemos la apertura de la barrera al haber un coche delante del sensor:

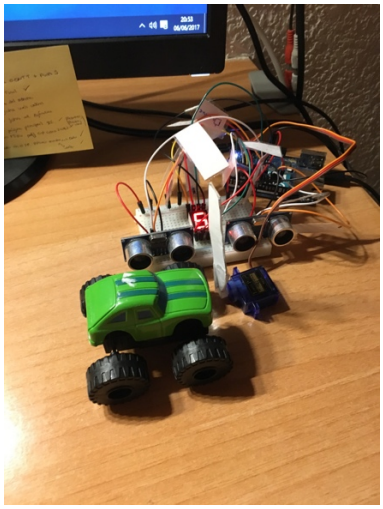

Figura 14 – Apertura barrera

Finalmente, vemos el estado final del test, donde vemos que el 7-segmentos muestra correctamente las plazas libres:

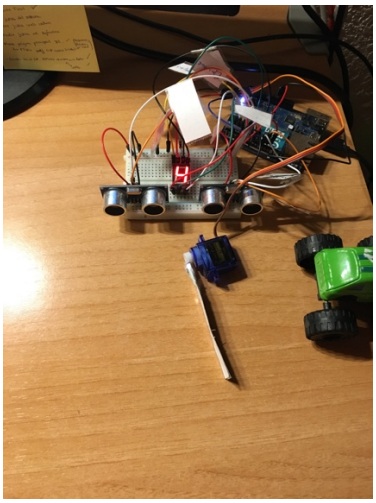

Figura 15 – Estado final

9 RESULTADOS

En el siguiente punto comentaremos los resultados adquiridos una vez finalizamos el proyecto. Seguidamente, se especifican los resultados:

- Controlamos la entrada/salida de vehículos del parking.
- Controlamos el número de plazas disponibles del parking.
- Controlamos la interacción con la barrera.
- Informamos al usuario de la fecha y la hora vía web.
- Comunicamos estado del parking a través del servidor web.
- Controlamos de las diferentes respuestas del servidor.
- Informamos a los usuarios el número de plazas disponibles vía wifi.
- Informamos a los usuarios, in situ, del estado del parking.
- Múltiple conexión vía wifi al servidor web.

Los resultados satisfacen cada uno de los distintos objetivos que se definieron al principio del proyecto. Siendo el proyecto terminado tal y como lo especificaba el último objetivo:" Tener todo listo antes 27 de junio de 2017".

10 CONCLUSIONES

Finalmente, en este apartado mostramos las conclusiones que hemos obtenido una vez finalizado el desarrollo del proyecto.

Este proyecto se inició por que el estudiante tenía curiosidad por el desarrollo de un proyecto basado en sistemas de microcontroladores. Ya que, en la Mención del Software, escogida por el alumno, se ha enfocado más en el desarrollo de aplicaciones, test del software, recogida de requisitos, etc. Aceptado el trabajo por parte del coordinador de la mención de software, la tutora del TFG, Marta Prim, y el estudiante, yo mismo, hicimos una reunión inicial para ver como podíamos enfocar el proyecto a nivel de software sin olvidarnos del desarrollo de la parte del Arduino.

Aunque el proyecto es el sumatorio de dos menciones éste no ha sido fácil, dado que teníamos poca experiencia en el desarrollo de software en Arduino y sus sensores, aunque los problemas los hemos ido solucionando poco a poco.

Una vez resueltos los problemas que nos iban surgiendo, el estudiante ha ido obteniendo, poco a poco, experiencia como si se tratará de un proyecto para un cliente real, obteniendo como resultado la experiencia del uso de metodologías vistas en clase de manera teórica, experiencia planificando las diferentes fases del proyecto con la finalidad de cumplir con los objetivos y la planificación para entregarlo en la fecha especificada.

También comentar que los problemas y las soluciones que hemos detectado hacen que este proyecto pueda tener un amplío desarrollo en un futuro.

11 TRABAJO FUTURO

Por tal de dar una continuidad al proyecto actual, podríamos proponer una serie de soluciones a los diferentes problemas que hemos encontrado a lo largo del desarrollo del proyecto, consiguiendo mejorar la fiabilidad del sistema.

A continuación, veremos los problemas y las diferentes soluciones futuras que hemos pensado:

El sistema no diferencia entre coches y personas. Al usar dos sensores de tipo ultrasonido, estos no nos diferencian entre una persona o un coche. Aunque la implementación del sensor funciona por proximidad, cualquier objeto que se quede inmóvil abrirá la barrera del parking. Si una persona pasa y no se queda parada delante del sensor, al tener éste un retraso de unos pocos milisegundos la barrera no sé abrirá.

La manera para arreglar este problema puede ser solventada de dos maneras:

- **a.** Sensores de suelo: los cuales estarán definidos para pesos mayores a los de una persona.
- **b.** Sensor de matrícula: es una muy buena resolución a este problema, ya que podremos saber la cantidad de coches aparcados dentro del parking.

El uso de las soluciones nos incrementaría el coste del proyecto.

Pérdida de energía. Como pasa en muchos parkings de los centros comerciales, si se va la luz nos quedamos sin el funcionamiento del sistema.

La solución sería simple, añadiríamos un SAI[11] conectado a nuestro sistema. El SAI sería el encargado de controlar de controlar el nivel de tensión que alimenta al sistema. Si en cierto momento, el sistema detecta que no obtiene corriente alterna, cambiaría a la corriente generada por el SAI. En nuestro proyecto lo implementaríamos con una batería y la conectaríamos al Arduino.

Pérdida de información. Si se va la luz, perderíamos la información del número de aparcamientos libres.

Se implementaría la librería EEPROM de Arduino para guardar el estado en el que se encuentre el sistema al perder la energía.

Información de las plazas ocupadas. Nuestro sistema solo controla la entrada y salida de vehículos al parking.

Una opción para dar más funcionalidad a nuestro sistema, sería el de mejorar el módulo de software, añadiendo un sensor a cada plaza. Podríamos utilizar o un sensor de suelo o un sensor ultrasonido con led. Cuando el sensor detectara un coche dentro de la plaza, daría un aviso al sistema para actualizar la web. Mostrando así la plaza que ha ocupado el coche.

AGRADECIMIENTOS

Mis agradecimientos van destinados a diversa gente. Primeramente, dar las gracias a la profesora Marta Prim porque gracias a ella he podido realizar el proyecto que en un principio estaba definido para otra mención de la ingeniería. Además, agradecer a mis padres y a mi pareja el apoyo y el ánimo que me han dado creyendo más ellos en mí que yo en mi mismo. Y finalmente, agradecer a cada uno de los profesores que he tenido en los diferentes años de la universidad que han aportado su granito de arena.

BIBLIOGRAFÍA

[1] OK Hosting, "Metodologías del Desarrollo de Software". Enero 2016. [Online]. Disponible en: http://okhosting.com/blog/metodologias-del-desarrollo-de-software/ Consulta: 26 de febrero de 2017.

[2] Arduino comunity, "How to expand the Yún disk space". [Online]. Disponible en:
https://www.arduino.cc/en/Tutorial/ExpandingYunDiskSpace
Consulta: 20 de abril de 2017

Arduino IDE, "Download the Arduino IDE". Mayo 2011. [Online]. Disponible en:
https://www.arduino.cc/en/main/software
Consulta: 17 de febrero de 2017

Wikipedia, "La Maquinista", 23 de febrero de 2017. [Online]. Disponible en:
https://es.wikipedia.org/wiki/La_Maquinista
Consulta: 5 de abril de 2017.

Arduino, "Arduino Yún". [Online]. Disponible en: http://www.arduino.org/products/boards/arduino-yun
Consulta: 5 de abril de 2017.

[11] Sistema de Alimentación Ininterrumpida.

Sublime SFTP, “Installation”. Año 2017. [Online]. Disponible en:
https://wbond.net/sublime_packages/sftp/installation
Consulta: 2 de mayo de 2017.

Educachip, “Como usar la memoria eeprom de arduino”. Noviembre 2014. [Online]. Disponible en:
http://www.educachip.com/como-usar-la-memoria-eeprom-de-arduino/
Consulta: 9 de mayo de 2017

TecnologiaArea, “Corrienta continua y Alterna”. Abril 2015 [Online]. Disponible en:
http://www.areatecnologia.com/corriente-continua-alterna.htm
Consulta: 9 de mayo de 2017

Arduino community, “Arduino Bridge Library”. Abril 2013. [Online]. Disponible en:
http://www.arduino.org/learning/reference/bridge
Consulta: 13 de mayo de 2017.

APÉNDICE

En el apartado del apéndice mostramos información suplementaria de la planificación y los diferentes estados en los que nos podemos encontrar el sistema.

A1. DIAGRAMA DE GANTT

En este apartado, vemos en la Figura A.1 la planificación que hemos hecho para nuestro proyecto. A través del Diagrama de Gantt vemos el orden de las tareas en el tiempo:

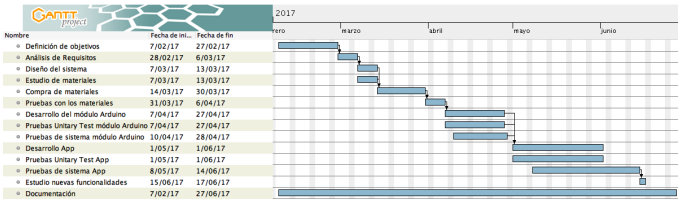

Figura A.1 – Diagrama de Gantt

A2. ESTADOS DEL SISTEMA

Por último, en el siguiente apartado veremos los diferentes estados que nos podemos encontrar el sistema. En primer lugar, veremos las fotos relacionadas con el módulo de hardware, Figuras de la A2 hasta A6.

El estado inicial del parking, todas las plazas libres, es el que veremos en la Figura A.2. Nuestro sistema estará listo para gestionar la entrada/salida de nuestro parking.

Figura A.2 – Estado inicial

Una vez el sensor0 detecta a un coche procederá a levantar la barrera, dejando paso a que el usuario pueda aparcar el coche dentro de la instalación. Como vemos en la Figura A.3, la barrera está levantada para que pase el usuario a estacionar el vehículo:

Figura A.3 – Apertura entrada

Una vez que el usuario ha pasado la zona de los sensores, la barrera se bajará descontando así una plaza libre del parking, Figura A.4.

Figura A.4 – Entrada coche

Cuando el usuario quiera abandonar el parking, al acercarse sensor1, este levantará la barrera, Figura A.5, dejando salir al usuario del parking.

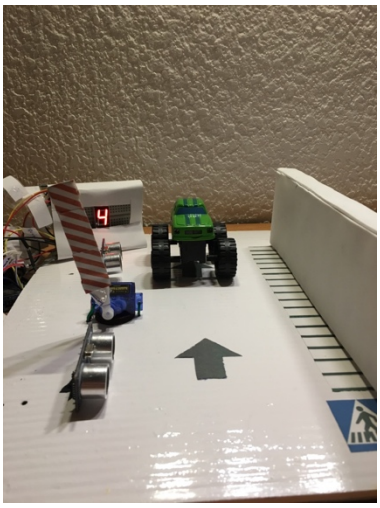

Figura A.5 – Apertura salida

Por último, cuando el parking no disponga de plazas disponibles la barrera no se levantará, además el 7-segmentos informará al usuario que no hay ninguna plaza libre, Figura A.6.

Figura A.6 – Parking lleno

Seguidamente, explicaremos que pasa en el módulo de SW, concretamente en la página web.

En la Figura A.7, vemos el estado en el que se encontrará la web cuando hayan entrado por primera vez y un usuario haya aparcado su vehículo en él.

Figura A.7 – Parking disponible

En la Figura A.8, veremos el estado en el que se encuentra la página web cuando el parking empieza a estar lleno. El número saldrá en naranja avisándonos de que quedan pocas plazas de aparcamiento.

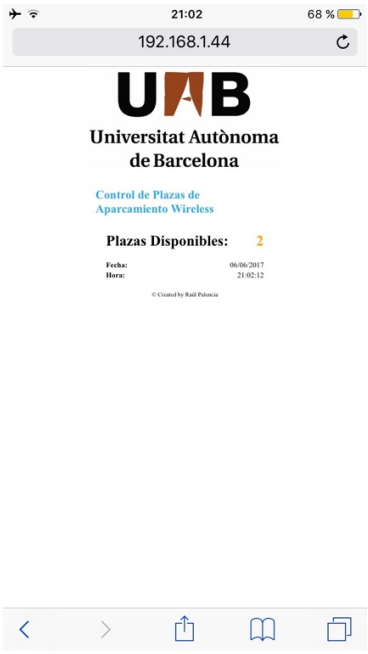

Figura A.8 – Parking medio vacío

Finalmente, cuando el parking no tenga ninguna plaza libre, se mostrará el número de plazas en color rojo como se puede apreciar en la Figura A.9.

Figura A.9 – Parking lleno