

UNIVERSIDAD CARLOS III DE MADRID

TRABAJO FIN DE GRADO



**DESARROLLO DE UNA AYUDA TÉCNICA PARA
ALUMNOS DEL COLEGIO SAN RAFAEL (15):
TRANSICIÓN A LA VIDA ADULTA – JUEGO DE
CARRERAS**

*GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL
Y AUTOMÁTICA*

Autor: Sergio Aguilera Sevilla

Tutor: Ricardo Vergaz Benito

Leganés, 4 de octubre de 2016

Título: Desarrollo de una ayuda técnica para alumnos del colegio San Rafael (15):
Transición a la Vida Adulta – Juego de Carreras

Autor: Sergio aguilera Sevilla

Tutor: Ricardo Vergaz Benito

EL TRIBUNAL

Presidente: Francisco Jose Rodriguez Urbano

Secretario: Elisa Cabana Garceran del Vall

Vocal: Pedro Martin Mateos

Suplente: Jorge Pleite Guerra

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 4 de Octubre de 2016 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

VOCAL

SECRETARIO

PRESIDENTE

AGRADECIMIENTOS/DEDICATORIA

Se acerca el momento en el cual pueda de decir que acabé el proyecto, tanto la parte física como la memoria. Debo decir que no ha sido un camino fácil, sino duro y trabajado, tengo que dar las gracias a muchas de las personas que me han ayudado y acompañado durante estos 5 años que he pasado en la carrera, porque con esto no solo a cabo un proyecto sino a cabo una etapa importante de mi vida tanto a nivel educativo como personal, que me ayudará muchísimo tanto en los próximos años de mi vida laboral como para toda la vida personal que me queda por recorrer.

En primer lugar quiero dar las gracias al Grupo de Displays y Aplicaciones Fotónicas de la UC3M, en especial a mi tutor Ricardo, por hacer posible este Proyecto de Fin de Grado para el Colegio de Educación Especial del Hospital San Rafael, ya que para mí es un honor y un privilegio poder hacer algo con una finalidad tan bonita, la de ayudar a personas.

No me podría olvidar nunca de mi compañero de proyecto Sergio Castillo, ya que gracias a él pudimos llevar a cabo este proyecto, aunque en momentos de su desarrollo lo veíamos casi imposible, debido a sus ideas, ya que algunas nos complicaban demasiado el trabajo y por ello tuvimos algunas discusiones sobre el rumbo que debíamos tomar. Aunque he de decir que las ideas que se llevaron a cabo han mejorado sustancialmente este proyecto.

A la empresa SAMTEC, que nos proporcionó los terminales de conexión entre el microcontroladores y las placas de circuito impreso, como muestra y de forma gratuita.

También debemos agradecer la ayuda que nos proporcionó la empresa de fabricación de circuitos electrónicos 2CISA, quienes nos mostraron su ayuda para la fabricación de las dos placas que necesitábamos para el juego de forma gratuita desde un primer momento, siendo estas unas placas con una calidad profesional, que no podíamos permitirnoslas si ellos.

Agradecerles la ayuda y el apoyo que nos han dado a los compañeros del laboratorio 2.1.C.12, Cristian, Edu, Fernando, Rafa, Almudena, que siempre han estado para echar una mano y apoyarnos con el proyecto, sobre todo en la parte anímica, ya que veíamos como ellos avanzaban en sus proyectos y nosotros teníamos problemas iniciales.

Por ultimo no podía olvidarme nunca de mis amigos de toda la vida que aunque ellos no podían ayudar con los problemas que surgían en el TFG siempre me apoyaban y me animaban cuando me veían desanimado con los problemas. En concreto a Iris por su ayuda y apoyo en los últimos días, que no veía el momento de acabar esta memoria.

A mi familia que me han aguantado todos estos meses de trabajo, sobre todo mis padres a los que tuve medio salón y parte de la casa como mi taller personal, en el que realice la construcción de las estructuras de madera, llenando todo de polvo y serrín.

También dedicárselo a mi abuelo que por desgracia no pudo llegar a ver la conclusión final del juego por unos días, y que en verano siempre me decía “Como sigues yendo a la universidad, que se tienen que ir de vacaciones”, y aun estando él mal, siempre me preguntaba como llevaba el proyecto y me daba ánimos para que siguiera y lo pudiera acabar. Aunque por desgracia no lo conseguí a tiempo para que lo viera.

RESUMEN

Este trabajo pertenece a un proyecto de colaboración que mantienen el Colegio de Educación Especial del Hospital San Rafael de Madrid para niños con discapacidad motora, intelectual y otros trastornos asociados y el Grupo de Displays y Aplicaciones Fotónicas de la Universidad Carlos III de Madrid (GDAF-UC3M).

El objetivo de este proyecto es desarrollar un dispositivo de ocio, basado en un juego de carreras típico de las ferias, pero adaptado para los alumnos del colegio. En él se distinguen tres grupos diferentes de alumnos y el presente trabajo está pensado para aquellos que se encuentran en la etapa de la Transición a la Vida Adulta, basándose siempre en el Diseño para Todos, ya que lo pueden usar tanto alumnos de menor edad como personas sin las limitaciones de estos alumnos.

Para poder tener una visión completa del proyecto “Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael (15): Transición a la Vida Adulta-Juego de Carreras” es necesario apoyarse en la memoria del proyecto de mi compañero Sergio Castillo Mohedano. Este proyecto está dividido en dos bloques, los cuales serán explicados entre las dos memorias, siendo en esta donde se detallan los sistemas que conforman el Bloque Rampa y la construcción de las dos estructuras de los bloques del juego, mientras que en la memoria de mi compañero Sergio Castillo Mohedano se explica detalladamente los sistemas del Bloque Carriles y todo el diseño referente a las PCB

Como se ha comentado, el sistema está compuesto por dos bloques independientes, en los cuales se implementa una comunicación unidireccional mediante radiofrecuencia, en el que el Bloque Rampa proporciona la información al Bloque Carriles.

En el Bloque Rampa se genera toda la información que se emite al otro bloque. Puede ser el número de jugadores y el personaje de cada uno, lo que le sirve al otro bloque para saber cuántos motores debe activar y qué sonidos. También se obtiene una puntuación que corresponde con los centímetros que avanza el motor en cuestión. Y una vez finalizada la partida se lo comunica al otro bloque para que mande a los motores a su posición inicial.

Toda esta información, aparte de ser enviada al Bloque Carriles, también se muestra por la pantalla LCD que tiene el Bloque Rampa.

El juego se ha probado en el colegio con resultados satisfactorios y actualmente permanece en funcionamiento en el colegio.

ABSTRACT

This project is part of a collaborative project between the School of Special Education of the San Rafael Hospital in Madrid for children with motor and intellectual disabilities as well as other associated disorders and the Displays and Photonic Applications Group of the Universidad Carlos III in Madrid (known in Spanish as GDAF-UC3M).

The aim of this project is to develop an entertainment device adapted to the students of this school based on a racing game that is typical in funfairs. Three different groups of students can be distinguished in it, and this project is intended for those who belong to the stage of Transition to Adulthood. It is based on the notion of Design for All, since also younger students as well as people without the limitations of these students can use it.

In order to see the project “Development of a technical aid for the students of the San Rafael School (15): Transition to Adulthood-Racing game” as a whole, it is necessary to rely on the project report of my peer Sergio Castillo Mohedano. The entire project is divided into two parts which are explained separately in each report. The present report shows the systems of the Ramp Section and the construction of the structures of both the game sections, whereas my peer’s report shows the systems of the Rails Section and the design related to the PCBs.

As mentioned, the system consists of two separate sections, in which there is a one-way communication by radio frequency. Thus the Ramp Section provides the information to the Rails Section.

All the information delivered to the other section is generated in the Ramp Section. This information could be the number of players and their characters, so the other section would know how many motors and which sounds should be activated. A score corresponding to the centimeters advanced by each motor is also obtained. And once the game is over it is communicated to the other section to send the motors to their initial position.

All this information, apart from being sent to the Rails Section, is also displayed by the LCD screen located in the Ramp Section.

The game has been tested in the school and has had a satisfactory outcome. Consequently, it now remains operative in the school.

INDICE

Contenido

AGRADECIMIENTOS/DEDICATORIA	4
RESUMEN	5
ABSTRACT	6
INDICE.....	7
Índice de Figuras	9
CAPITULO 1. INTRODUCCION Y OBJETIVOS.....	13
1.1 INTRODUCCION.....	13
1.2 MOTIVACION	13
1.2.1 Diseño para Todos	13
1.2.2 Colegio San Rafael	14
1.3 ESTADO DEL ARTE	15
1.4 OBJETIVOS.....	20
1.5 ESPECIFICACIONES DEL SISTEMA	21
1.5.1 Explicación del bloque completo	21
1.5.2 Explicación bloque Rampa.....	22
1.5.3 Explicación bloque Carriles	23
1.5.4 Requisitos del sistema	24
1.6 METODOLOGIA.....	25
1.7 FASES DEL PROYECTO	26
1.8 METODOS UTILIZADOS	27
1.9 DESCRIPCION DE LA MEMORIA	29
CAPITULO 2. DISEÑO DEL SISTEMA.....	30
2.1 Diseño del Sistema Completo.....	30
2.1.1 Bloque Rampa	30
2.1.2 Bloque Carriles	32
2.2 Bloque Rampa	34
2.2.1 Botón pulsador.....	34
2.2.2 Rampa.....	38
2.2.3 Zona de detección.....	42
2.2.4 Panel de Control	46
2.2.5 Sistema de transmisión	49

2.2.6 Alimentación	52
2.2.7 Microcontrolador	53
CAPITULO 3. IMPLEMENTACION DEL SISTEMA.....	57
3.1 Descripción del firmware	57
3.2 Implementación de la Electrónica	67
3.3 Construcción Mecánica del Sistema.....	71
3.3.1 Material.....	71
3.3.2 Impresora 3D	73
3.3.3 Bloque Rampa	76
3.3.4 Bloque Carriles	84
CAPITULO 4. PRUEBAS Y RESULTADOS EXPERIMENTALES	92
4.1 Pruebas iniciales	92
4.2 Pruebas experimentales de Campo	95
CAPITULO 5. CONCLUSIONES Y POSIBLES LINEAS FUTURAS	96
5.1 Conclusiones.....	96
5.2 Líneas Futuras	96
5.3 Presupuesto	97
Bibliografía.....	105
Índice de Acrónimos.....	106
Anexos	107
ANEXO 1. Data Sheet.....	107
ANEXO 2. Manual de Instrucciones	109
ANEXO 3. Programa Bloque rampa	114
ANEXO 4. Planos	139

Índice de Figuras

Ilustración 1. Colegio de Educación Especial San Rafael.....	14
Ilustración 2. Gráfico accesibilidad de juguetes para personas con discapacidad motora	16
Ilustración 3. Histograma del número de juegos accesibles en el mercado para niños con deficiencia motora, por edad en años del niño	16
Ilustración 4. Gráfico accesibilidad de juguetes para personas con las tres discapacidades	16
Ilustración 5. Pulsador adaptado para alumnos del colegio San Rafael	17
Ilustración 6. Jugete adaptado	17
Ilustración 7. Pulsador mediante soplo.....	17
Ilustración 8. Juego de carreras de caballos	18
Ilustración 9. Juego de carreras de 6 caballos	18
Ilustración 10. Mensaje de un jugador que ha conseguido más de 70 puntos.....	21
Ilustración 11. Selección del número de jugadores	22
Ilustración 12. Mensaje para seguir jugando, entre turno y turno	23
Ilustración 13. Laboratorio 1.2.C.12	27
Ilustración 14. Detalle de fuentes de alimentación y osciloscopio, equipo portátil y placa de Bloque Carriles	27
Ilustración 15. Diagrama de bloques del Bloque Rampa	30
Ilustración 16. Diagrama de bloques del Bloque Carriles.....	32
Ilustración 17. Boceto del Pulsador.....	34
Ilustración 18. Botón Rojo	35
Ilustración 19. Microswitch interno del botón	35
Ilustración 20. Tapa de la caja del botón	36
Ilustración 21. Caja del Botón	36
Ilustración 22. Jack aereo	37
Ilustración 23. Conexión eléctrica del sistema del pulsador.....	37
Ilustración 24. Servomotor SG90.....	38
Ilustración 25. Conexión para el control del servo.....	38
Ilustración 26. Señal PWM para el control del servo.....	38
Ilustración 27. Comparador que amplifica la señal PWM.....	39
Ilustración 28. Conexión eléctrica en el servo.....	39
Ilustración 29. Puerta con los servos colocados	40
Ilustración 30. Colocación de pivotes en una máquina de Galton.....	40

Ilustración 31. Distribución de la caída en la Máquina de Galton	41
Ilustración 32. Rampa con la Máquina de Galton y las palas.....	41
Ilustración 33. Sensor de peso FSR	42
Ilustración 34. Led infrarrojo.....	43
Ilustración 35. Gráfica de direccionalidad del led.....	43
Ilustración 36. Gráfica de longitud de onda de trabajo del led.....	43
Ilustración 37. Fototransistor infrarrojo lateral.....	43
Ilustración 38. Gráfica de potencia recibida en función de la distancia	44
Ilustración 39. Gráfica de Responsabilidad espectral.....	44
Ilustración 40. Colocación de la los led para la detección.....	44
Ilustración 41. Conexión eléctrica del sistema de detección	45
Ilustración 42. Amplificador no inversor	45
Ilustración 43. Botón para controlar el sistema	46
Ilustración 44. Conexión eléctrica de los botones de control	46
Ilustración 45. Pantalla LCD	47
Ilustración 46. Descripción de los pines.....	47
Ilustración 47. Configuración interna de la pantalla.....	48
Ilustración 48. Conexión eléctrico de la pantalla LCD.....	48
Ilustración 49. Bloque de 8x2 socket con agujero pasante de 2.54mm.....	49
Ilustración 50. Circuito Integrador RF600E.....	50
Ilustración 51. Conexión de RF600E.....	50
Ilustración 52. Esquemático AM-RT4-433	51
Ilustración 53. Fuente de alimentación TXM 025-105.....	52
Ilustración 54. Conexión eléctrico del sistema de alimentación.....	52
Ilustración 55. Placa de desarrollo NUCLEO-F411RE.....	53
Ilustración 56. Vista de los pines usados en el chip, desde STM32CubeMX	54
Ilustración 57. Imagen de la configuración de reloj del sistema	54
Ilustración 58. Switch para seleccionar número de jugadores.....	60
Ilustración 59. Función para enviar los datos por RF.....	61
Ilustración 60. Comandos para enviar la información y luego limpiarla	61
Ilustración 61. For para seleccionar los personajes	62
Ilustración 62. Inicialización de los servomotores	62
Ilustración 63. Función para mover los motores los grados que uno quiera	63
Ilustración 64. Declaración del bucle que hace nuestro juego hasta que acaba	63

Ilustración 65. Parte del programa de la apertura de la puerta hasta que cae la pelota en una casilla	64
Ilustración 66. Detección de cada casilla.....	64
Ilustración 67. Bucle de las tiradas	66
Ilustración 68. Función para parar todos los servomotores.	66
Ilustración 69. Prueba inicial del sistema de detección	67
Ilustración 70. Amplificador No Inversor con un Lm234	68
Ilustración 71. Prueba con varios leds y fototransistores	68
Ilustración 72. Señal después de amplificarla	69
Ilustración 73. Circuito integrado para controlar los servos.....	69
Ilustración 74. Colocación de los cables en el pulsador	70
Ilustración 75. Madera de calabo.....	71
Ilustración 76. Rollo de filamento PLA	72
Ilustración 77. Impresora Witbox 2.....	73
Ilustración 78. Pieza completa a la izquierda y otras dos que se quedaron a medias.....	74
Ilustración 79. Pieza con fallos en el extrusor	74
Ilustración 80. Esqueleto de listones de la estructura.....	76
Ilustración 81. Estructura con los lados colocados.....	77
Ilustración 82. Colocados las casillas	77
Ilustración 83. Caja donde se coloca la pelota.....	78
Ilustración 84. Colocación de los obstaculos de la rampa.....	78
Ilustración 85. Bloque Rampa completo	79
Ilustración 86. Caja del Pulsador.....	79
Ilustración 87. Tapa de la caja del pulsador	80
Ilustración 88. Interruptor de alimentación y entrada Jack	80
Ilustración 89. Colocación de los botones de control.....	81
Ilustración 90. Hueco de la pantalla LCD	¡Error! Marcador no definido.
Ilustración 91. Colocación de un de los leds	82
Ilustración 92. Estructura de listones de los Carriles	84
Ilustración 93. Estructura con los laterales.....	85
Ilustración 94. Colocación de los paneles frontales	85
Ilustración 95. Tapa de cada carril.....	86
Ilustración 96. Bloque Carriles completo	86
Ilustración 97. Colocación de un motor paso a paso	87
Ilustración 98. Varilla de la polea esclava.....	87

Ilustración 99. Polea esclava con el rodamiento interno.	88
Ilustración 100. Polea esclava colocada en la estructura.....	88
Ilustración 101. Polea completa.....	89
Ilustración 102. Correa de los carriles	89
Ilustración 103. Diseño 3D de apoyo para correa de los carriles	90
Ilustración 104. Colocación de los apoyos en la estructura.....	90
Ilustración 105. Hueco para los altavoces	91
Ilustración 106. Lateral de la estructura para ver los huecos necesarios.....	91

CAPITULO 1. INTRODUCCION Y OBJETIVOS

1.1 INTRODUCCION

Antes de comenzar esta memoria del Trabajo Fin de Grado (TFG), hay que recalcar que este proyecto ha sido realizado en conjunto con el de Sergio Castillo Mohedano, y por tanto el lector podrá encontrar partes con descripciones similares en ambas memorias de trabajo. Como por ejemplo, este primer capítulo de introducción y motivación del proyecto, en cuyo final se explicará cómo se organiza la presente memoria, y en qué parte del proyecto se va a centrar.

1.2 MOTIVACION

La principal motivación que se encontró a la hora de realizar este proyecto fue el hecho de que iba a ser utilizado en la vida real teniendo una importante utilidad ya que era para los niños del Colegio de Educación Especial del Hospital San Rafael. Este proyecto se pudo llevar a cabo gracias a la estrecha colaboración que existe entre este colegio y el GDAF-UC3M, Grupo de Displays y Aplicaciones Fotónicas del Departamento de Tecnología Electrónica de la Universidad Carlos III de Madrid, en especial con el profesor Ricardo Vergaz, y la cual conocimos gracias a la casualidad en una asignatura que en la que dicho profesor nos impartía clase.

Pero para entender la relevancia de este proyecto se debe explicar primero el concepto de diseño para todos y el Colegio San Rafael.

1.2.1 Diseño para Todos

El concepto de diseño para todos se basa en acercar los productos y servicios al mayor número de personas posible.

El Diseño para Todos se define, en España según la ley 51/2003, de la siguiente manera:

“Diseño para todos: la actividad por la que se concibe o proyecta, desde el origen, y siempre que ello sea posible, entornos, procesos, bienes, productos, servicios, objetos, instrumentos, dispositivos o herramientas, de tal forma que puedan ser utilizados por todas las personas, en la mayor extensión posible.” (12)

Son Principios del Diseño para Todos:

Aunque no se tiene un grupo de criterios de diseño establecidos, la filosofía del diseño universal, según el Centro para el Diseño Universal, se basa en 7 principios:

1. Uso equiparable: el diseño debe ser fácil de usar y adecuado para todas las personas independientemente de sus capacidades y habilidades.
2. Uso flexible: el diseño debe poder adecuarse a un amplio rango de preferencias y habilidades individuales.
3. Simple e intuitivo: el diseño debe ser fácil de entender independientemente de la experiencia, los conocimientos, las habilidades o el nivel de concentración del usuario.

4. Información perceptible: el diseño debe ser capaz de intercambiar información con usuario, independientemente de las condiciones ambientales o las capacidades sensoriales del mismo.

5. Con tolerancia al error: el diseño debe minimizar las acciones accidentales o fortuitas que puedan tener consecuencias fatales o no deseadas.

6. Que exija poco esfuerzo físico: el diseño debe poder ser usado eficazmente y con el mínimo esfuerzo posible.

7. Tamaño y espacio para su acceso y uso: los tamaños y espacios deben ser apropiados para el alcance, manipulación y uso atendiendo a al tamaño del cuerpo, su postura o la movilidad que tenga el usuario. (1) (2)

1.2.2 Colegio San Rafael

Este proyecto está centrado en proporcionar ayuda y entretenimiento a los alumnos de mayor edad del Colegio de Educación Especial del Hospital San Rafael, en Madrid.



Ilustración 1. Colegio de Educación Especial San Rafael

Este edificio como a utilizarse como colegio de educación especial a partir de 1976, pertenece a la Orden de Hermanos de San Juan de Dios, y se encuentra situado dentro del recinto de Hospital San Rafael. Este centro acoge a alumnos con discapacidad motora, intelectual y otros trastornos asociados, de edades comprendidas entre los 3 y los 21 años, a los que organiza en tres grupos según las edades, para así abarcar mejor las distintas etapas educativas.

- La Educación Infantil para alumnos entre los 3 y 6 años.
- La Educación Básica Obligatoria entre los 6 y 16 años.
- El Programa de Transición a la Vida Adulta entre los 16 y 21 años.

Para poder atender y tratar lo mejor posible a estos alumnos el colegio consta de un grupo de profesionales en sus distintos campos para conseguirlo. Estos van desde la parte

pedagógica, motriz y comunicativa con el exterior hasta la salud, alimentación y la higiene. Para ello el colegio tiene el siguiente equipo:

- Departamento Docente.
- Departamento de Logopedia.
- Departamento de Fisioterapia.
- Departamento de Orientación.
- Departamento de Autonomía y Accesibilidad.
- Departamento de Auxiliares Técnicos Educativos.
- Servicio de enfermería. (3)

El colegio se encuentra dividido en diferentes salas, dependiendo de las actividades que vayan hacer en cada momento y las etapas educativas a las que pertenezcan, como se comentó anteriormente. En el caso de nuestro proyecto está dedicado a los alumnos pertenecientes al programa de Transición a la Vida Adulta.

Por lo cual este TFG tiene que estar orientado a adolescentes pero adaptado a las necesidades de los alumnos del colegio, porque por desgracia la mayor parte de los juegos comerciales para este rango de edad no están hechos para las personas con discapacidad físico psíquico sensorial como es el caso de los alumnos, con lo que estos chicos no pueden utilizar juegos comerciales que vayan acorde a su edad. Como todo el mundo puede comprender, ellos van creciendo y sus inquietudes y estimulaciones van cambiando y evolucionando con el paso de los años, por lo que no pueden seguir jugando y divirtiéndose con los mismos dispositivos de cuando eran pequeños. Y es un derecho y necesidad de los niños/as con discapacidad el jugar y tener acceso a los distintos tipos y recursos de juego.

Por eso con nuestro TFG se quiere ayudarles para que puedan pasar sus ratos de ocio con un juego acorde a su edad y sus necesidades, además ayudarles en su estimulación.

1.3 ESTADO DEL ARTE

En el mercado existen productos similares al que nosotros vamos a diseñar y desarrollar, pero no están adaptados para personas discapacitadas, con lo cual no podrían ser utilizados por los alumnos. Además, este tipo de juegos son muy caros.

Para encontrar algún tipo de juego que esté adaptado a las necesidades que demandan los alumnos hay empresas que se dedican a adaptar los aparatos con las especificaciones que se le manden, pero evidentemente este hecho hace que el precio se eleve de forma considerable, también hay que tener en cuenta que no todo se puede adaptar por lo que hay una gama de productos muy reducida para las personas con este tipo de discapacidad.

Según un estudio realizado por AIJU sobre juguetes que existen en el mercado tan solo el 61% han sido valorados como accesibles para personas con discapacidad motora, pero solo una quinta parte de ellos son adecuados sin ayuda ni adaptación. (Ilustración 2)

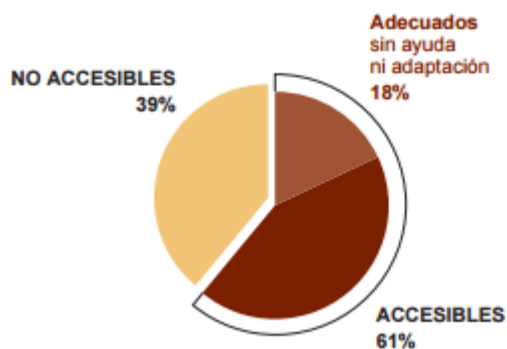


Ilustración 2. Gráfico accesibilidad de juguetes para personas con discapacidad motora

Pero esto va a peor según aumentamos la edad a la que van dirigidos los juegos, porque para los niños con menor edad encontramos más juegos que a los de mayor edad como se puede ver en el gráfico de la Ilustración 3.

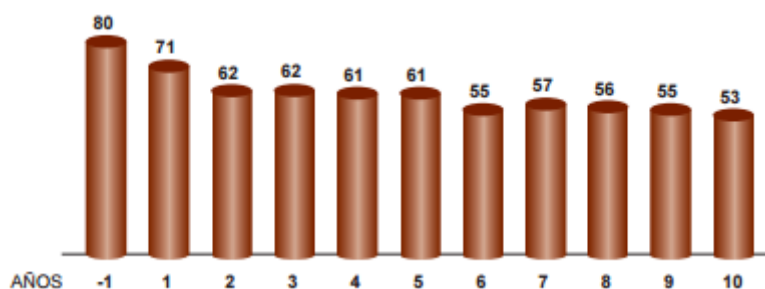


Ilustración 3. Histograma del número de juegos accesibles en el mercado para niños con deficiencia motora, por edad en años del niño

Aunque según nos comentaron los profesores del colegio algunos alumnos pueden llegar a tener más discapacidades asociadas a la principal, por lo que se reduce aún más el número de productos adaptados. Véase en la Ilustración 4 en el supuesto de que tuvieran tres discapacidades (visual, auditiva y motora). Que son algunos de los casos que tienen en el colegio, aunque no llegan a tener discapacidad completa en las discapacidades.

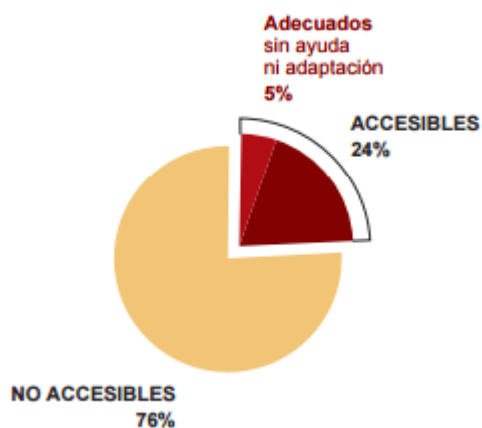


Ilustración 4. Gráfico accesibilidad de juguetes para personas con las tres discapacidades

Debido a todo esto los profesores del centro intentan modificar y adaptar productos comerciales que se encontrarían en cualquier sitio a las necesidades de los alumnos. En el colegio nos mostraron ejemplos de ello (Ilustración 5), ponían una entrada Jack hembra en los dispositivos, y de esta forma con un Jack macho colocado en un pulsador adaptado y personalizado a cada alumno, ellos son capaces de usar dicho dispositivo sin problema. (4)



Ilustración 5. Pulsador adaptado para alumnos del colegio San Rafael

Este tipo de pulsador está adaptado a las necesidades de cada alumno ya que cada uno puede tener limitaciones distintas. Algunos ejemplos de pulsadores son: un interruptor colocado para pulsar con la cabeza de lado ya que solo puede girar el cuello; un tubo por el que sopla el alumno (Ilustración 7), debido a que ésta es su única forma de interactuar con el exterior de forma física.



Ilustración 6. Juguete adaptado



Ilustración 7. Pulsador mediante soplo

Con la vista puesta en ofrecer un juego atractivo para alumnos del ciclo superior del Colegio, se ofertó por parte del equipo del San Rafael el desarrollo de un juego de carreras. Por su descripción, es inmediato encontrar un equivalente en el mercado: un juego que se puede encontrar en los parques recreativos y en cualquier feria de unas fiestas populares. Es un juego que no está enfocado para ninguna edad en concreto sino para que juegue toda la familia, ya que es un juego de carreras, en el que el muñeco puede variar: se puede encontrar camellos, caballos, frutas o cualquier otra cosa (ver Ilustración 8).



Ilustración 8. Juego de carreras de caballos

Consiste en un juego de destreza y habilidad, en el cual se trata de ir sumando el mayor número de puntos para que el personaje vinculado a tu tablero de juego sea el primero que llegue a la meta. Este tablero consiste en una rampa con una serie de huecos en los que hay que conseguir introducir una bola que se lanza deslizando por ella, donde cada uno tiene distinta puntuación. El personaje avanza según la puntuación asociada al hueco por donde entre la bola.



Ilustración 9. Juego de carreras de 6 caballos

Y claro este tipo de juegos son muy grandes y excesivamente caros: el juego de la ilustración 9, cuesta 34.200€, un precio muy elevado y que además no cumple las necesidades que ellos demandan: se necesita una amplia movilidad en los brazos para realizar el lanzamiento de la bola. (5)

1.4 OBJETIVOS

El objetivo principal de este proyecto es poder proporcionar a los alumnos del Colegio que se encuentran en la transición a la vida adulta un dispositivo de ocio que les sea también útil para desarrollarse de forma social y emocional.

Según numerosos estudios realizados el juego permite desarrollar la propia capacidad física y mental, y para los niños discapacitados es algo muy importante ya que es una de sus principales fuentes de autoafirmación, satisfacción y placer. (4)

Además de los beneficios que tiene el juego para los niños es un derecho para ellos y los adultos deben velar por su cumplimiento en todo momento, como queda reflejado en la Asamblea General de las Naciones Unidas de 1959 y que ratificó el Parlamento Español en 1990. (1)

Este juego que se va a desarrollar, se basa en un diseño accesible para todos, lo que debería ser algo en lo que se basasen todos los productos: en primer lugar, beneficiaría a las personas que tienen mayores problemas de accesibilidad, como las personas con discapacidad. En segundo lugar, sería bueno para la sociedad ya que facilitaría que las personas con o sin discapacidad puedan compartir recursos y los momentos de ocio, siendo esto una buena forma de integración en la sociedad.

Para ello el proyecto se ha dividido en dos bloques, uno en el que los alumnos pueden interactuar de forma física con el juego, ya sea pulsando el botón simplemente o, para aquellos que tengan mayor movilidad, colocando la bola en la caja para iniciar la caída, y de esta forma potenciar su desarrollo motriz.

Y a través del segundo bloque, se le intenta estimular con un sistema multisensorial, mediante el sistema de audio para la parte sonora, y con el movimiento de los muñecos en los carriles para la parte visual.

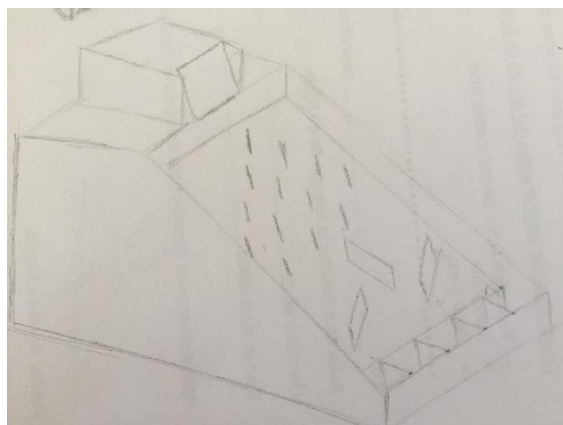
Este juego además fomenta la interacción con los compañeros y con los monitores, de esta forma hace que los alumnos no se repriman en sí mismos y les obliga a actuar con el exterior. Todas las discapacidades que tanto psicológicas como motoras les puede provocar este retraimiento, con este juego se quieren evitar.

Este juego pretende además servir de ejemplo para las empresas de juguetes, ya que no les supondría un gran aumento en el coste el adecuarlos para un diseño para todos, ya que sería algo de lo que nos beneficiáramos toda la sociedad.

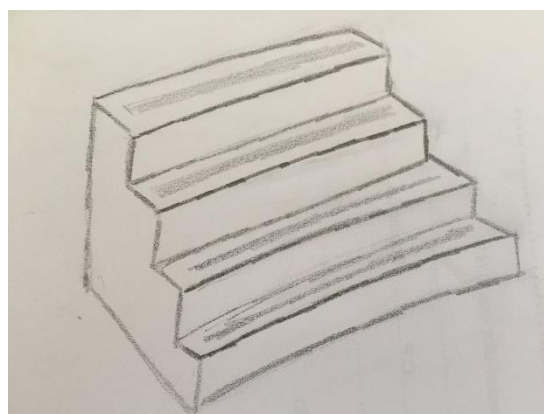
1.5 ESPECIFICACIONES DEL SISTEMA

El proyecto consta de dos partes claramente diferenciadas una de otra, pero dependientes entre sí, ya que entre los dos bloques existe una comunicación hecha mediante radiofrecuencia para que así no haya cables entre medias que dificulten el paso o molesten en la habitación donde se coloquen los bloques. Las especificaciones del funcionamiento y de las dimensiones vienen dadas por el personal del Colegio San Rafael, y en este apartado se describen tal como se propusieron, pero con ayuda de fotografías del resultado final mostradas en avance, a fin de facilitar la comprensión de las mismas.

1.5.1 Explicación del bloque completo



Boceto 2. Bloque Rampa



Boceto 1. Bloque Carriles

El juego consiste en el típico juego de carreras que se puede encontrar en las ferias, pero adaptado a los alumnos del Colegio. En nuestro caso tenemos dos bloques como se ha comentado anteriormente: el primero (Boceto 1) sería una rampa que es por donde caerá la pelota que otorgará la puntuación a cada jugador. Esta pelota se encuentra en la parte superior de la rampa dentro de una caja, la cual abrirá la puerta cuando se accione el pulsador, ya sea el que hemos colocado nosotros o el adaptado para el alumno. La pelota cae por la rampa a través de los obstáculos y luego cae en unas casillas, cada una con su puntuación.

Y esta puntuación se envía al otro bloque que es el de los carriles (Boceto 2), que con dicha puntuación, mueve el muñeco correspondiente al jugador la distancia correspondiente a los puntos obtenidos en el lanzamiento. Este proceso se repetirá hasta que un jugador obtenga 70 puntos como mínimo, ya que se puede dar el caso de conseguir más puntos de los necesarios para ganar en la última tirada, como se puede ver en la Ilustración 10.



Ilustración 10. Mensaje de un jugador que ha conseguido más de 70 puntos

Hay una opción que se puede llegar a dar, que dos jugadores obtengan 70 puntos en el mismo turno, por lo que ganará aquel que tenga más puntos, o en caso de empate a puntos será el último que haya pulsado.

Y para comenzar una nueva partida debemos pulsar un botón de RESET.

1.5.2 Explicación bloque Rampa

Este es el primer bloque del juego ya que con él comienza la partida.

En este bloque tenemos una zona donde el profesor interactúa con el juego, a través de unos botones colocados en el lateral del bloque, con los que seleccionará los jugadores que van a participar, que son de 1 a 4, y esto lo verá en una pantalla LCD (Ilustración 11), que se encuentra al lado de dichos botones.



Ilustración 11. Selección del número de jugadores

Una vez seleccionado esto, se debe seleccionar también el personaje que quiere utilizar cada jugador, en los que hay cuatro opciones: coche, moto, caballo y dinosaurio.

Cuando se han seleccionado los jugadores y los personajes se comenzará a jugar con un pulsador que se ha diseñado, aunque también se pueden usar los propios de los alumnos ya que hay una conexión Jack para ello.

Al accionar el botón se abre la compuerta que sujeta la pelota, activando los dos servomotores que la controlan, y la pelota cae por la rampa, a través de unos pivotes colocados específicamente formando una máquina de Galton que comentaremos más adelante en el capítulo 2, y unos servos que accionan unas palas que dificultarán la caída de la pelota y que se mueven de forma aleatoria.

Cuando la pelota pase la rampa anterior caerá en unas casillas, cada una con una puntuación distinta, y que será reconocida mediante unos detectores ópticos, y le dará la puntuación correspondiente. Siendo la casilla central la máxima puntuación con 20 puntos, las siguientes van decreciendo en puntuación hacia los lados de la siguiente forma: las dos colindantes a ésta de 15 puntos, y las dos de los extremos de 10 puntos.

Esta información será traspasada al otro bloque a través de un sistema de radiofrecuencia.

Antes de la siguiente tirada debemos pulsar el botón de OK en el panel de control, como se muestra en la Ilustración 12. Esto está hecho para que si se debe cambiar el pulsador no haya problemas y no se abra la compuerta de la pelota, al provocar un pico de tensión cuando se conecta el nuevo pulsador.



Ilustración 12. Mensaje para seguir jugando, entre turno y turno

1.5.3 Explicación bloque Carriles

Este segundo bloque del juego es la continuación del mismo y donde se ve el resultado de lo ocurrido en el bloque Rampa.

Una vez se recibe la información mediante la radiofrecuencia y una vez decodificada llega al microcontrolador. Esto hace que el sistema se ponga en marcha.

Primero se activa el sistema de audio, poniendo en marcha el tono correspondiente a cada personaje y al movimiento que le corresponda, ya que si es un movimiento ganador, sonará un todo de victoria que es distinto a los otros. Para el sonido hay colocados dos altavoces uno a cada lado del bloque.

Después de activar el sonido y mientras suena, comienza a moverse el carril correspondiente al jugador, mediante un motor paso a paso, colocado cada uno al comienzo de cada carril, y aunque estos motores se pueden controlar con alta precisión también hay unos finales de carrera colocados en los extremos de cada carril para mayor seguridad, a la hora de mover los motores.

El movimiento corresponde con la puntuación que se ha obtenido en el bloque anterior, cada punto corresponde a un centímetro, siendo 70 puntos los necesarios para obtener la victoria. Pero el jugador puede conseguir más de 70 puntos por ese motivo se tienen los finales de carrera para que paren los motores cuando se llegue al final.

1.5.4 Requisitos del sistema

Cuando se fue a hablar con el personal del colegio sobre el juego que se iba a desarrollar nos dejaron claras una serie de especificaciones que tenían en mente.

Nos dijeron que pensáramos en todo momento para quién iba dirigido el proyecto, porque los alumnos a los que va dirigido tienen ciertas limitaciones, por lo que no deberíamos poner detalles que dificultaran su uso, porque aunque a nosotros nos pudiera parecer sencillo deberíamos de atenernos al Diseño Universal, ya que de esta forma sería perfecto para las necesidades de ellos.

Los detalles luminosos también nos dijeron que no eran especialmente importantes ya que había muchos alumnos que tenían dificultades visuales, por lo que lo más importante para ellos y que sí que querían era el sistema de audio a la hora de moverse los personajes y que con él se intentara motivar a los alumnos.

Otro detalle fue que los bloques tuvieran conexión a la red eléctrica, ya que en el pasado tuvieron sistemas autónomos con batería y se acababa muy rápido la carga y por tanto su uso, esto era algo que también se tenía en mente y se quería hacer.

El juego por tanto tenía que estar adaptado a los niños del colegio con sus correspondientes discapacidades, pero querían que fuera multijugador, en el que se pudiera jugar hasta 4 personas y de esta forma fomentar su relación con otros compañeros.

También debía tener una opción para conectar un pulsador de Jack 3.5mm al bloque para activar el sistema, por lo cual se decidió que el propio mando tuviera esta conexión también para que así funcionase de la misma forma.

Debido al uso que va a tener el juego, el proyecto debe ser bastante robusto y resistente, debido a los cambios de sitio que va a tener, ya que no se va a quedar en un sitio fijo, y además el pulsador debe ser bastante resistente a los golpes, debido a que puede sufrir caída, y sobre todo al uso de los alumnos del colegio.

Como último requisito se marcaron las dimensiones una vez se les presentó un diseño previo del sistema completo para que se hicieran una idea de cómo iba a quedar. Con lo que los bloques quedaron establecidos con las siguientes dimensiones.

- La rampa tendrá 50x50x80 cm (siendo ancho, alto y fondo visto de frente).
- Los carriles tendrán 90x60x60 cm.

Debido a las dimensiones de los bloques y por tanto del sistema completo se pensó que entre los bloques no hubiera cables conectados entre sí ya que de esta forma se facilitaría su traslado y colocación en las habitaciones del colegio, por eso se comunican mediante radiofrecuencia.

1.6 METODOLOGIA

Para comenzar el presente TFG, lo primero que se hizo fue ir a hablar con el colegio para saber los requisitos necesarios y más importantes que ellos habían decidido y conocer el lugar y las personas a las que iba a ir el proyecto.

Tras esto nosotros planteamos un diseño superficial de los bloques, en el cual aparecieron dudas referentes a las dimensiones de estos con lo que nos pusimos en contacto con ellos para poder hacer el diseño general y lo que necesitábamos para el desarrollo del proyecto.

Lo siguiente fue pensar el sistema de control que iba a tener, se barajaron diferentes opciones, como usar Arduino que sería una forma a priori más sencilla, pero con el inconveniente de la limitación de pines. Por lo cual se decidió el uso de una placa de desarrollo con un microcontrolador, en principio se usó un STM8. Y se pasó al montaje de la electrónica en las placas protoboard para hacer las pruebas de pequeños subsistemas, con su programación necesaria.

Pero el proceso de programación se nos complicó bastante y no podíamos unir la parte electrónica con la programación, y se cambió a un Nucleo-F411RE lo que nos hizo poder avanzar bastante. A la par se empezó a realizar la construcción de la estructura de los dos bloques, tanto la rampa como los carriles, y la parte mecánica del sistema, en la cual debíamos pensar en soluciones a los múltiples problemas que se planteaban según se avanzaba, siendo algunas de las soluciones gracias al diseño de piezas 3D.

Una vez hecha toda la electrónica y una primera versión de la programación por partes se empezó el diseño de la PCBs para que nos las pudieran fabricar en una empresa externa, ya que eran multicapa.

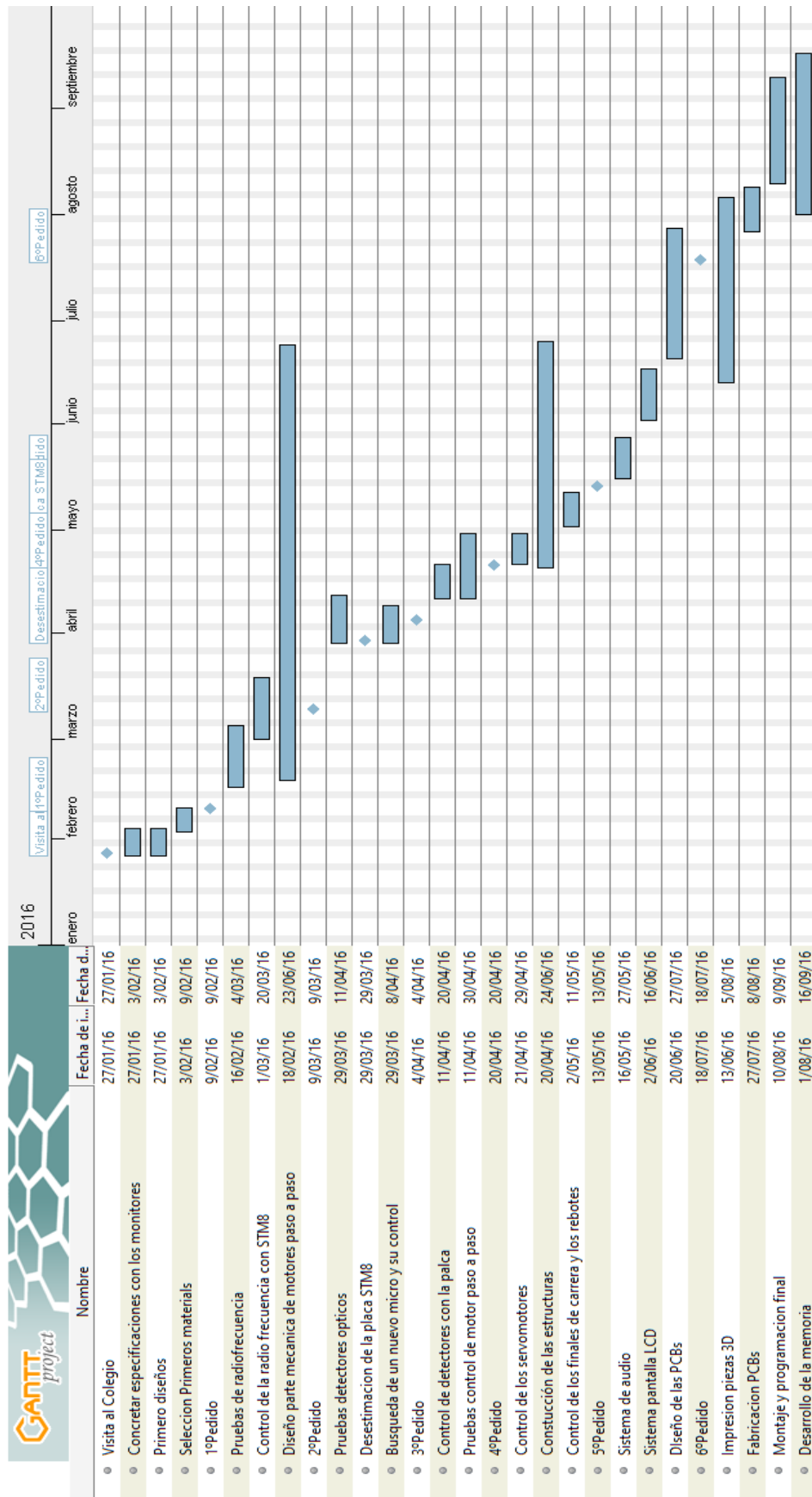
Cuando se recibieron las placas lo primero que se hizo fue soldar y colocar todos los componentes de las PCBs para poder comenzar con la programación del sistema para cada uno de los bloques, desde la selección de jugadores y personajes en el bloque rampa hasta la transmisión de la información por radiofrecuencia al bloque carriles, donde se debe interpretar correctamente esos datos para mover los motores paso a paso correctamente.

Una vez conseguida esta programación se pasó a la integración del sistema mecánico con la parte electrónica, conectando todo mediante el cableado correspondiente y fijando todos los componentes correctamente.

Una vez hecho todo esto el proyecto podía parecer completo, aunque aún le faltaba la fase de depuración del programa, a base de pruebas de ensayo/error, con lo que se realizaron las modificaciones necesarias para que funcionase el sistema lo mejor posible. Finalmente, el sistema se llevó al Colegio, donde fue probado, ajustado y ensayado con éxito, encontrándose operativo ya allí en el momento de escribir esta memoria.

Todo este desarrollo se resume de manera cronológica en el Diagrama de Gantt que se muestra en el siguiente apartado.

1.7 FASES DEL PROYECTO



1.8 METODOS UTILIZADOS

Para la realización del proyecto se dispuso en todo momento de todas las herramientas disponibles en el laboratorio 1.2.C12 del Departamento de Tecnología Electrónica (ilustración 13):

- Osciloscopios.
- Impresora 3D bq Witbox.
- Fuentes de Alimentación.
- Polímetro
- Componentes electrónicos: resistencias, condensadores, diodos, etc.
- Placas protoboard
- Otros materiales: cable, herramientas de corte, destornilladores, taladradora, pistola termoplástica...
- Conversor TTL-USB.



Ilustración 13. Laboratorio 1.2.C.12

Por otro lado, desde casa de mi compañero también se dispuso de diversos materiales que permitieron avanzar, en el mes de Agosto en especial (ilustración 14):

- Polímetro.
- Fuente de alimentación y osciloscopio.
- Puesto de soldadura: soldador JBC de 25W, desoldador, malla de cobre, flux...

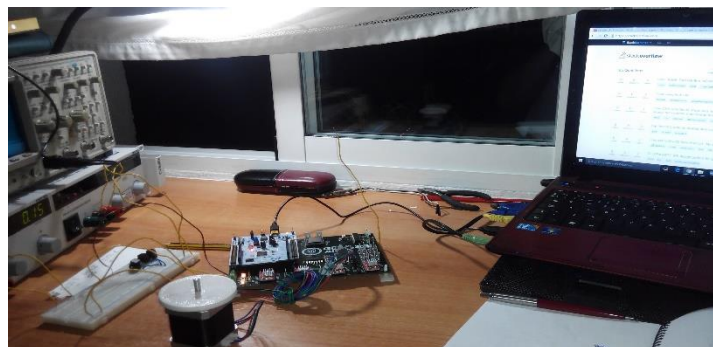


Ilustración 14. Detalle de fuentes de alimentación y osciloscopio, equipo portátil y placa de Bloque Carriles

En cuanto al software, utilizamos varios programas:

- Altium Designer 2016 – Evaluation License: con este software diseñamos los esquemas eléctricos de los dos bloques, así como las placas de circuito impreso.
- Autocad 2015 Student Version: para los diseños de las piezas 3D.
- STM32CubeMX: herramienta de configuración gráfica que permite generar el código de inicialización en C de cualquier microcontrolador o placa de desarrollo de la familia STM.
- Atollic TrueStudio 5.4.2 Lite Version: IDE (*Integrated Development Tool*) especialmente diseñada para microcontroladores y microprocesadores con arquitectura ARM (*Advanced RISC Machine*).
- HERCULES Setup Utility: terminal cuya principal ventaja por la que fue elegido es que es capaz de interpretar los bits en hexadecimal y mostrártelos de esta forma en tiempo real según van llegando.

1.9 DESCRIPCION DE LA MEMORIA

En la presente memoria podemos encontrar partes comunes con el trabajo de Sergio Castillo Mohedano, ya que ambos forman parte del mismo proyecto, aunque se encuentran partes propias que se deben al desarrollo del sub-proyecto del Bloque Rampa y la Construcción Mecánica de todo el proyecto.

El capítulo 1. Introducción y Objetivos. Este primer bloque es en su mayoría común a los dos trabajos. En él se puede leer una presentación del colegio y la necesidad que lleva a la construcción del proyecto, junto a una breve descripción de él, las fases por las que pasó durante su desarrollo y los medios que se utilizaron para llevarlo a cabo. Además de una explicación de las dificultades y requisitos que demandan los dispositivos que necesitan este tipo personas.

El capítulo 2. Diseño del Sistema. En este se presenta nuevamente el proyecto completo aunque centrándonos más en el Bloque Rampa, y a partir del apartado 2.2 Bloque Rampa en exclusividad ya de esta memoria se describen detalladamente los componentes electrónicos que lo conforman y las alternativas de diseño que se plantearon.

El capítulo 3. Implementación del Sistema. Se tiene una descripción del firmware, con la ayuda de unos flujogramas. También tenemos las primeras pruebas de la parte electrónica del bloque de la rampa que hemos empleado con las placas protoboard. Y llegaremos después a la construcción mecánica de todo el proyecto, tanto a la construcción de la estructura con madera como todo lo relativo a las piezas 3D necesarias para el juego.

El capítulo 4. Pruebas y Resultados Experimentales. Aquí se comentará la integración del firmware sobre las placas PCBs que ya estarán listas y montadas, esto será común a los dos trabajos. Y también las pruebas finales de campo que se realizarán una vez ya montado todo el proyecto.

El capítulo 5. Conclusiones y Posibles Líneas Futuras. En esta parte se hará un repaso a los objetivos marcados y las conclusiones personales que nos deja este proyecto. También se hablará de posibles mejoras que se le pueden realizar. Y se finalizará con el presupuesto, que será común y dará una idea del coste total que tiene el proyecto.

CAPITULO 2. DISEÑO DEL SISTEMA

2.1 Diseño del Sistema Completo

2.1.1 Bloque Rampa

Se muestra en La Ilustración 15 un diagrama genérico del Bloque Rampa. Son sus partes:

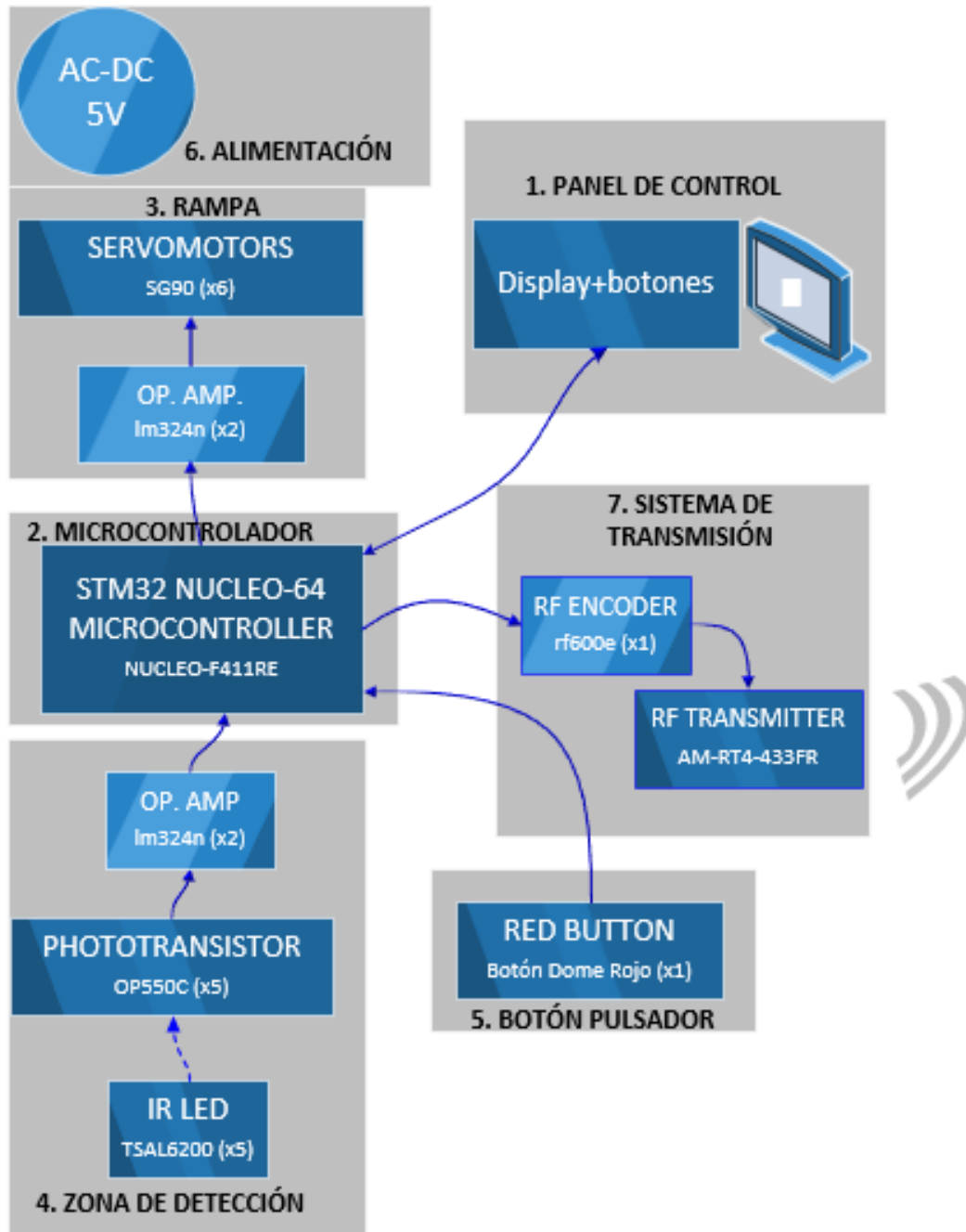


Ilustración 15. Diagrama de bloques del Bloque Rampa

1. Panel de Control:

Esta parte del sistema está formada por la pantalla LCD y tres botones que controlan la elección de los modos de juego (tanto el número de jugadores como el personaje elegido para cada uno) y a través de la pantalla se pueden ver todas las elecciones que se van a haciendo, y además se informa del desarrollo del juego. Esta parte del bloque mantiene una relación bidireccional con el microcontrolador del bloque rampa.

2. Microcontrolador:

Este sub-bloque es el encargado de recoger toda la información que se genera en el bloque rampa y de controlar todos los componentes y sistemas que lo conforman.

3. Rampa:

Esta parte está compuesta por una parte mecánica que define la caída de la pelota mediante una máquina de Galton y una parte electrónica que está formada por los servomotores y sus amplificadores correspondientes. Dos de estos servomotores sirven para controlar la apertura de la puerta de la caída de la pelota y los otros tres giran aleatoriamente sobre la rampa, provocando más aleatoriedad a la caída de la pelota.

4. Zona de detección:

Está formada por 5 huecos (casillas donde caerá la pelota), en los cuales hay 1 led infrarrojo y 1 fototransistor por cada hueco. Cuando la pelota cae en un hueco interrumpe el haz de luz que une el led y el detector, y esta interrupción provoca una señal que el microcontrolador detecta, otorgando una puntuación diferente según el hueco en el que cae.

5. Botón pulsador:

Es el botón que se ha puesto por defecto al juego, la función que realiza es la de activar los servos que controlan la puerta. Pero este pulsador puede ser cambiado por cualquier otro que tenga una conexión Jack 3.5mm.

6. Sistema de transmisión:

Es la encargada de transmitir la información de los puntos que ha conseguido cada jugador al bloque de los carriles por RF, este sub-bloque está formado por un encoder y circuito integrado encargado de la transmisión.

7. Alimentación:

Es una fuente de alimentación de 5V, que alimenta a todo el bloque.

2.1.2 Bloque Carriles

De la misma manera que para el Bloque Rampa, se describe en la Ilustración 16 un diagrama genérico del Bloque Carriles, siendo sus partes:

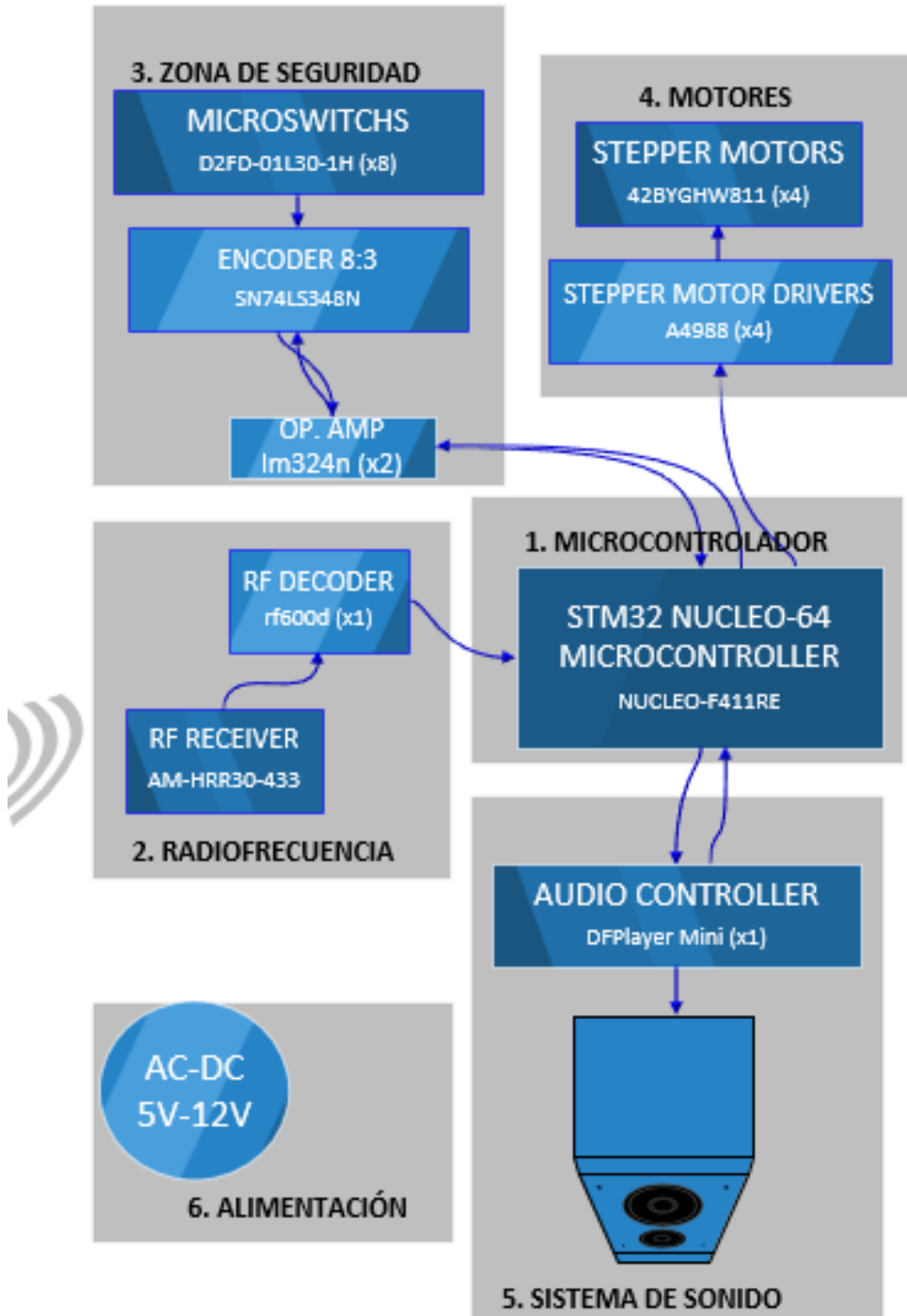


Ilustración 16. Diagrama de bloques del Bloque Carriles

1. Microcontrolador:
Este sub-bloque es el encargado de recoger toda la información que procede de la radiofrecuencia y la que se genera en el bloque de los carriles, y de controlar todos los componentes y sistemas que lo conforman.
2. Radiofrecuencia:
Es el encargado de recibir la información del otro bloque, y está formado por un circuito integrado receptor de radiofrecuencia y un decoder, que traduce la información que fue codificada en el otro bloque, permitiendo al microcontrolador entender la información y utilizarla.
3. Zona de seguridad:
Esta parte está compuesta por 8 microrruptores (2 por carril, uno a cada extremo del mismo), un encoder 8:3 y 2 amplificadores. Los microrruptores hacen la función de finales de carrera, por lo que cuando un personaje toca uno de ellos se para el motor, para que no esté funcionando una vez llegue a su final.
4. Motores:
Son cuatro motores paso a paso (con sus respectivos drivers de control), que son los encargados de mover los personajes sobre los carriles.
5. Sistema de sonido:
Es el encargado de aportar el sonido al juego, este sub-bloque tiene un circuito integrado que reproduce los sonidos que se guarden en una tarjeta microSD y estos sonidos se reproducen a través de dos altavoces colocados uno a cada lado del bloque de los carriles.
6. Alimentación:
Es una fuente de alimentación de dos canales, uno de 5V para la electrónica de este bloque y otro de 12V que alimentará los motores paso a paso.

2.2 Bloque Rampa

2.2.1 Botón pulsador

Este bloque consta del botón rojo, el cable Jack 3.5 mm, y la caja.

Este bloque consta principalmente de un gran botón rojo, el cual usamos como pulsador de nuestro juego, y es una parte clave, ya que con él arranca todo el proceso.

Basándonos en los principios del Diseño Para Todos, se tuvo que pensar y diseñar un pulsador que tuviera un diseño universal y de esta forma no fuese excluyente, para así fomentar el desarrollo y la mejora de los niños del centro a través del juego.

Por eso en primer lugar pensamos en diseñar un pulsador y fabricarlo mediante la impresora 3D del GDAF, para ello teníamos un primer boceto que se puede ver en la Ilustración 17.

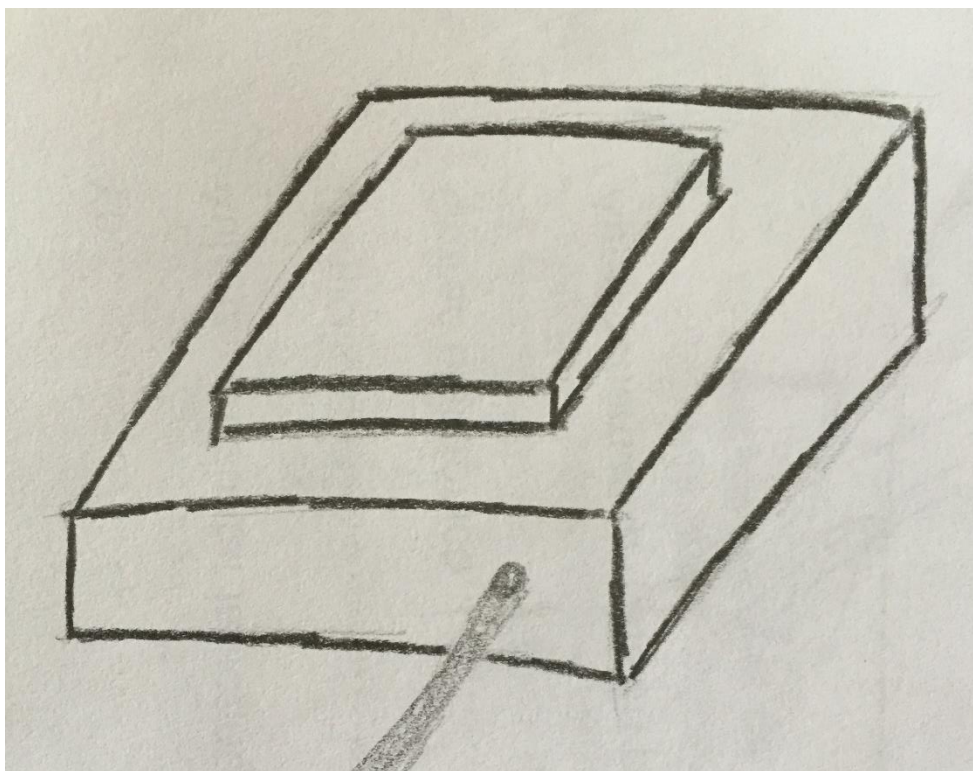


Ilustración 17. Boceto del Pulsador

Pero al ser un pulsador que va a tener un gran uso y debe tener una resistencia considerable, se decidió comprar un pulsador comercial, el cual tuviera una gran resistencia, que fuera grande y un gran ciclo de vida.

Por ello se eligió el “Botón Dome Rojo” (Ilustración 18).



Ilustración 18. Botón Rojo

Este botón consta de una serie de características que se adaptaban a la perfección a nuestras necesidades. Tiene un diámetro de 60mm y un ciclo de vida del interruptor probado de 10.000.000 ciclos, así como una bombilla incandescente que se enciende al pulsarlo.

Con lo que teníamos el botón ideal para el juego: grande, robusto y con una relación clara causa-efecto.

Este botón consta internamente del microswitch de la Ilustración 19, que tiene tres terminales para su conexión.



Ilustración 19. Microswitch interno del botón

En el terminal COM es al que va el cable Ring del Jack.

En el NO se coloca el Tip.

En el NC se le coloca el Sleeve.

Para este pulsador se diseñó una caja para encapsular y proteger las conexiones realizadas ya que este pulsador va conectado a la rampa mediante un cable Jack de 3.5mm conectado como se acaba de explicar en el microswitch.

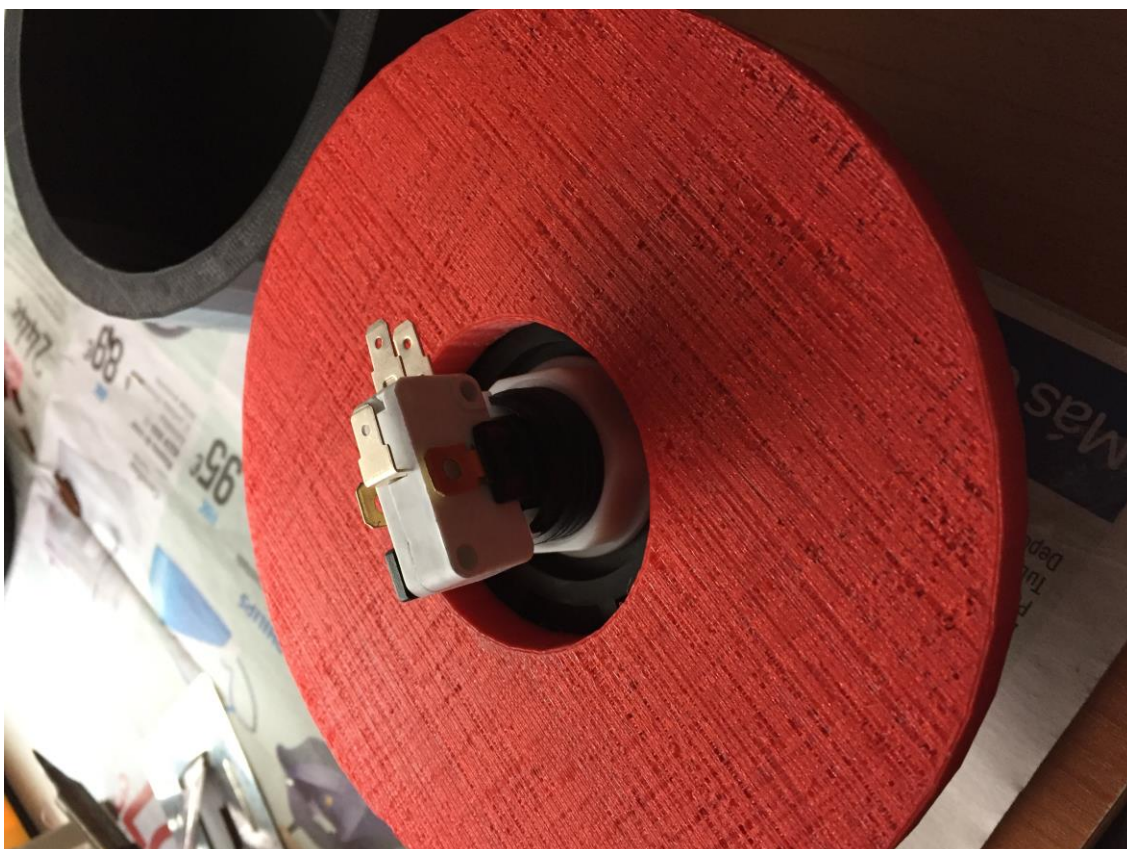


Ilustración 20. Tapa de la caja del botón

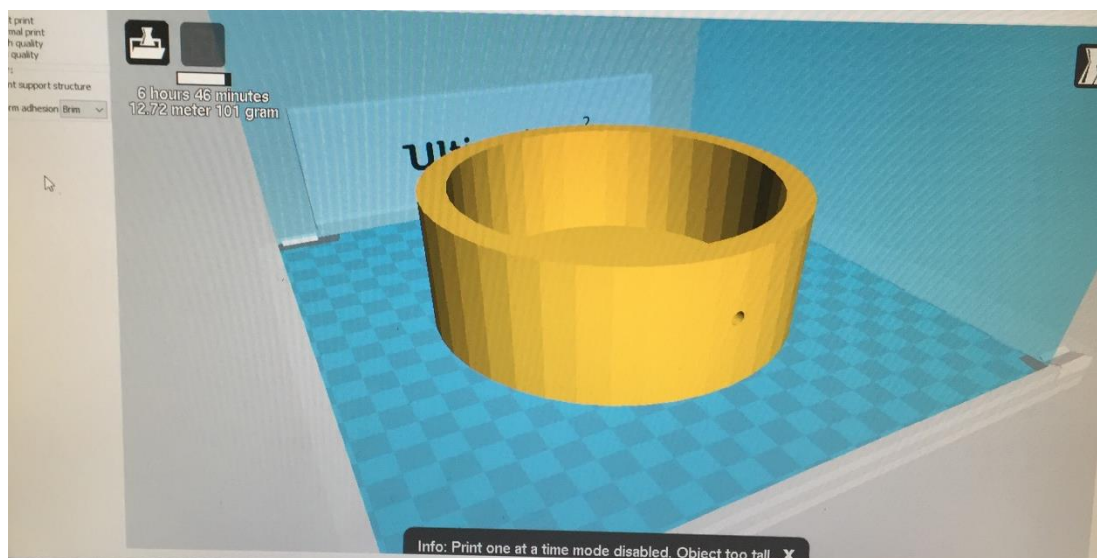


Ilustración 21. Caja del Botón

Las piezas que se ven en las Ilustraciones 20 y 21 son los dibujos 3D que ha diseñado el autor de esta memoria para proteger y colocar el pulsador.

El cable Jack se usa para conectar el mando con el bloque de la rampa, por lo que se debe colocar en ella un conector Jack hembra de 3.5mm aéreo como el de la Ilustración 22, y de esta forma cumplir con una de las máximas del diseño para todos de Igualdad de Uso y de Flexibilidad, con lo que los alumnos que tienen mayores problemas de movilidad puedan también usar el juego mediante sus propios pulsadores adaptados a sus propias limitaciones.



Ilustración 22. Jack aéreo

En el Jack hembra también se debe colocar una serie de cables en sus terminales: el Tip uno se conecta a la alimentación, el Ring es el que se dirige al micro y el Sleeve se conecta a la tierra.

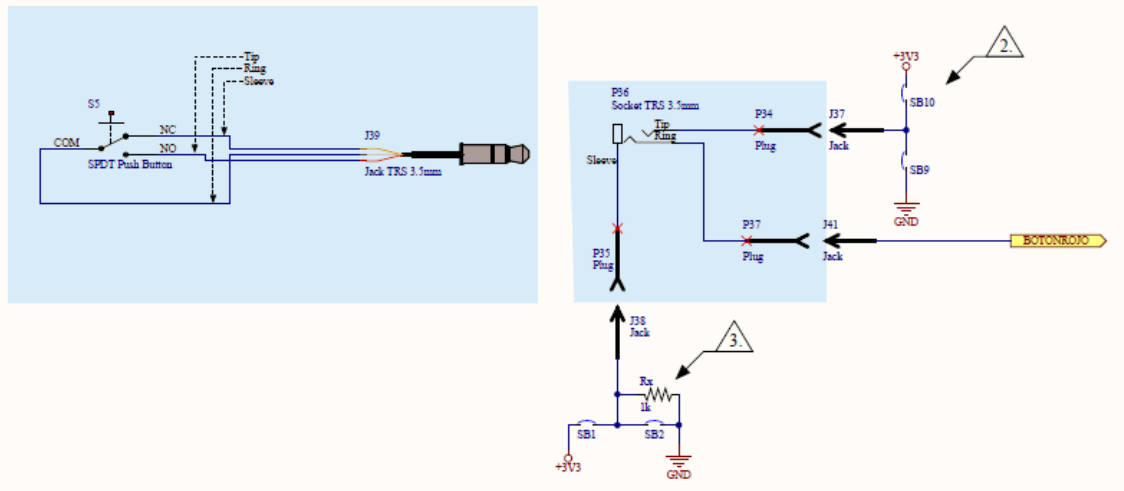


Ilustración 23. Conexión eléctrica del sistema del pulsador

La Ilustración 23 muestra el esquemático de la parte del pulsador, y los componentes necesarios para llevar la información hasta el microcontrolador. Una primera parte que es la conexión del pulsador con el cable Jack 3.5 con la conexión que explicamos anteriormente. Una segunda parte la del Jack hembra, en la que las conexiones que comentamos no son suficientes, sino que debemos colocar unos pequeños sistemas de componentes para conseguir que funcione correctamente.

El subcircuito 2 es para seleccionar el que se active por flanco de subida, por lo que se ha de soldar SB10 (puente que une el circuito con alimentación).

El subcircuito 3 es una resistencia con la que queremos evitar que se produzca un corto en caso de utilizar un pulsador con una conexión Jack 3.5 mm TS (conector con dos conductores Sleeve y Tip) en lugar de uno TRS (3 conductores Tip, Ring y Sleeve).

2.2.2 Rampa

Este parte es la parte más visual del juego aunque en la parte electrónica puede ser la que menos componentes tenga, ya que solo tenemos los servomotores y todo lo demás está construido en madera. Se explicará más adelante en el apartado de la construcción mecánica del bloque Rampa, [3.3.3 Bloque Rampa](#).

Los servomotores que se han empleado para el sistema son unos microservos SG90 de 9g (Ilustración 24).



Ilustración 24. Servomotor SG90

Es un servomotor pequeño, ligero y de alta potencia para su tamaño, 1.8Kgf·cm, con un barrido de 180°, que trabaja a 5V.

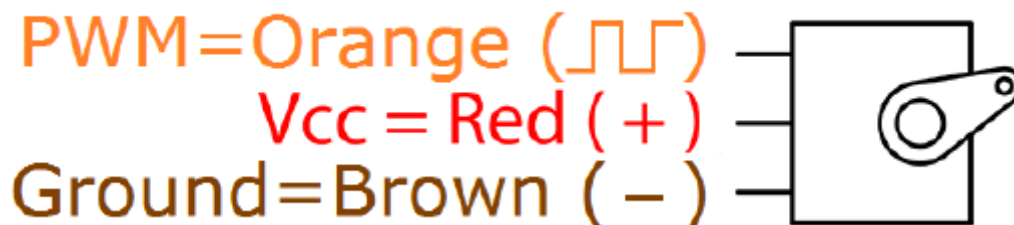


Ilustración 25. Conexión para el control del servo

Su conexión eléctrica es muy sencilla como se puede ver en la Ilustración 25. Debido a esta conexión el control de servo se debe a la señal PWM que tenemos que generar con el microcontrolador.

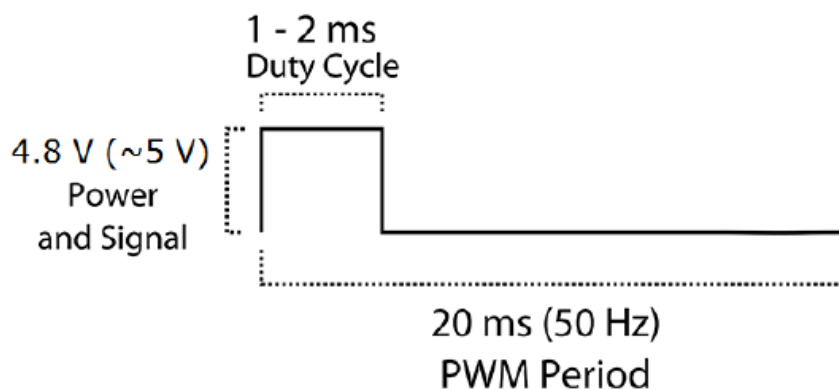


Ilustración 26. Señal PWM para el control del servo

La Ilustración 26 es un ejemplo de señal PWM con la que se controla el servo, para controlarlo solo debemos variar el ciclo de trabajo, para que el ángulo del servo esté a 0° el ciclo tiene que ser de 1ms y para que se vaya a los 180° se debe ser de 2ms. Por lo que para obtener el ángulo exacto que cada uno quiera se debe variar el ciclo de trabajo entre estos valores para conseguirlo, ya que varía de forma proporcional, siendo 90° con 1.5ms de ciclo de trabajo.

Para controlar el servo desde el microcontrolador, no es factible usar el servo, ya que necesita ser alimentado a 5V como se ha comentado anteriormente, y el micro no da tanto voltaje, por lo que se ha optado como solución usar un amplificador operacional como comparador, para así elevar la tensión que se va a mandar al servo. Para ello se ha utilizado un LM234n, el cual tiene las conexiones de la Ilustración 27.

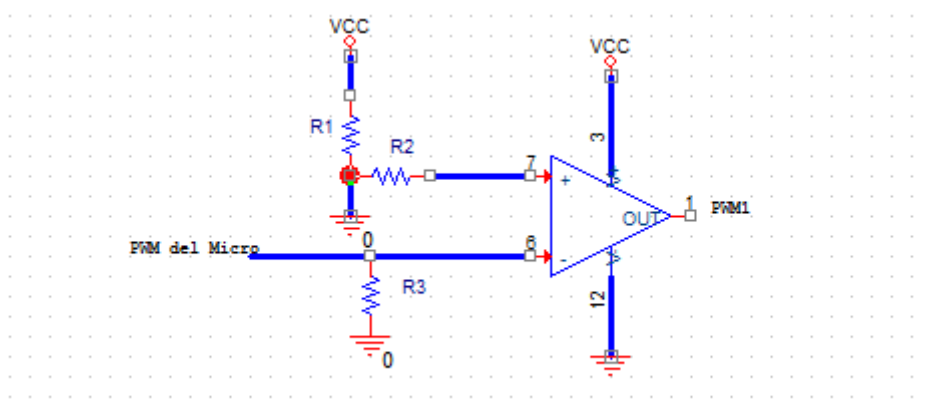


Ilustración 27. Comparador que amplifica la señal PWM

En la entrada negativa tenemos un divisor de tensión que reduce los 3,3V a 1V colocando las resistencias de R1 22k y R2 15k.

La entrada positiva es la que viene la señal de PWM proveniente del microcontrolador, con la tensión insuficiente para alimentar al servomotor, con una R3 de 1k.

El comparador se alimenta a 5V para +Vcc y a GND para -Vcc.

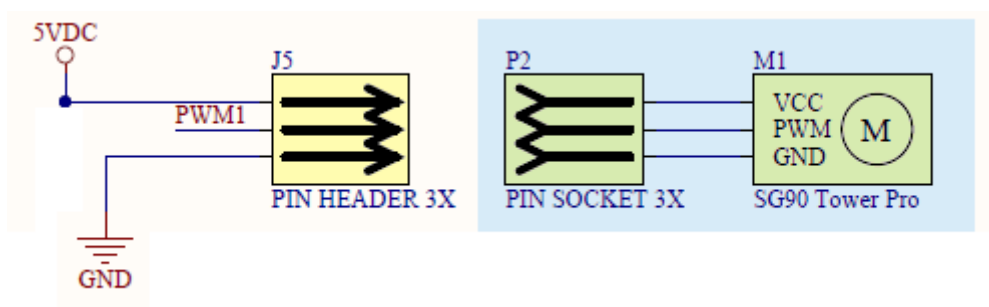


Ilustración 28. Conexión eléctrica en el servo

La salida del comparador PWM1 es la que llega al servo directamente, como podemos ver en la Ilustración 28. Las otras dos conexiones son la alimentación a 5V y la tierra.

Se han utilizado dos servos de este tipo para la apertura de la puerta que deja caer la pelota por la rampa, estos están colocados como en la Ilustración 29.

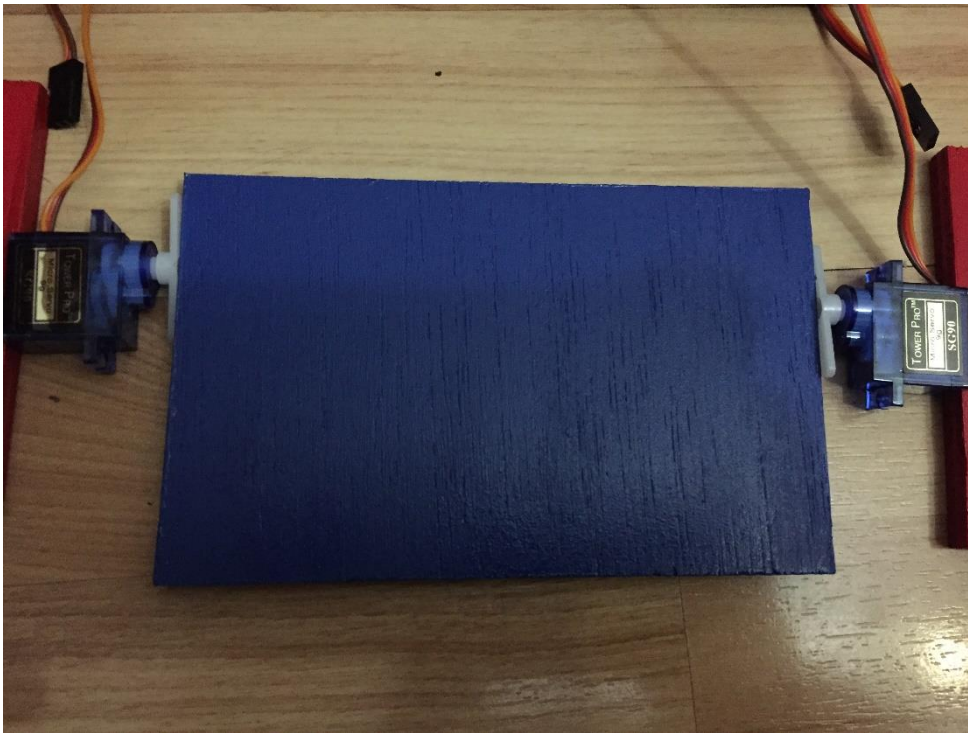


Ilustración 29. Puerta con los servos colocados

Estos servos se activarán una vez se apriete el pulsador, y de esta forma se dejará caer la pelota. Pero estos dos servos no serán los únicos que habrá, sino que tendremos otros tres dispuestos por la rampa para dar aleatoriedad a la caída de la pelota sobre las distintas casillas.

Aunque para ello también se tiene diseñada sobre la rampa una Máquina de Galton, con la cual ya se tendría una cierta aleatoriedad.

Este dispositivo fue creado por Francis Galton con la intención de demostrar el teorema del límite central, con lo que demostraba que la distribución normal es una aproximación bastante acertada de la distribución binomial.

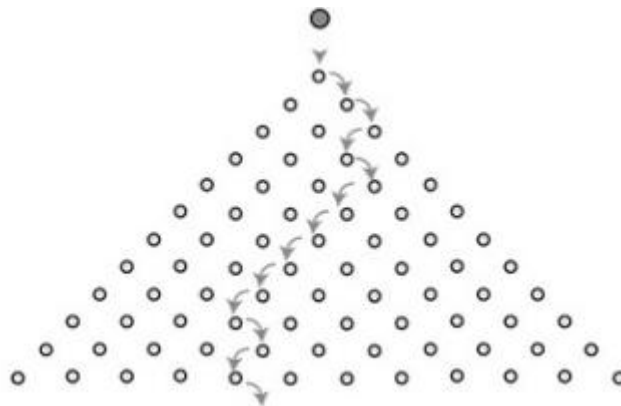


Ilustración 30. Colocación de pivotes en una máquina de Galton (7)

En la Ilustración 30 se puede ver un ejemplo de la máquina de Galton: como se puede apreciar la pelota va rebotando de un punto a otro y como están colocados a la misma distancia unos de otros la probabilidad de que la pelota caiga hacia la izquierda o hacia la derecha es de un 50% para cada lado, por lo que a medida que va cayendo a nivel siguiente tiene más posibilidades de que la pelota se desvíe.

Al cabo de muchas pruebas se puede apreciar que hay mayores probabilidades de que la pelota caiga en las zonas centrales en lugar de las exteriores, formando una superficie de campana tras sus caídas, conociéndose este tipo de distribución como distribución binomial, como se ve en la Ilustración 31. (6) (7)

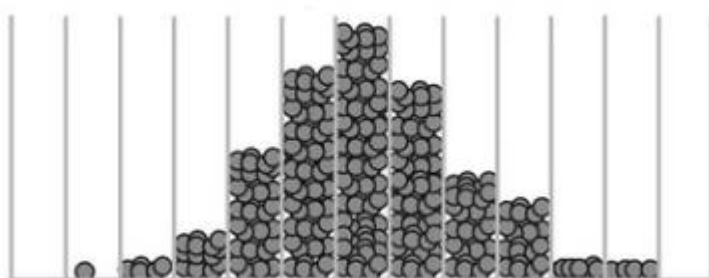


Ilustración 31. Distribución de la caída en la Máquina de Galton (7)

Como se comentó anteriormente había tres servos puestos en la rampa, estos irán puestos en la parte baja de la rampa dispuestos en forma de triángulo, se ha pensado en elegir esta colocación puesto que existe una gran probabilidad de que la pelota caiga en el centro, según lo visto anteriormente. Como los servos están activos habrá veces que la pelota pueda pasar sin chocar con ninguno, pero habrá otras en que impactará y será rebotada hacia otro lugar, cambiando así drásticamente su dirección.

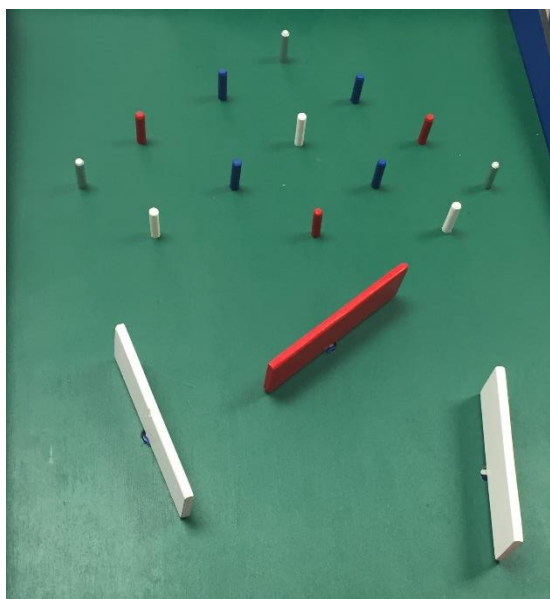


Ilustración 32. Rampa con la Máquina de Galton y las palas

2.2.3 Zona de detección

Esta parte del juego es la encargada de dictaminar el número de puntos que ha obtenido cada jugador, ya que en ella se detecta en qué casilla ha caído la pelota y de esa forma se le asigna una puntuación distinta.

ALTERNATIVA BARAJADA

Se valoraron distintas formas para detectar en qué casilla había caído la pelota: se valoró en un principio la opción de utilizar galgas extensiométricas colocadas en la base de cada casilla, de esta forma cada vez que la pelota entrara en la casilla el peso haría variar el valor nominal de la resistencia, y de esta forma saber que dicha casilla es donde se encuentra nuestra pelota.

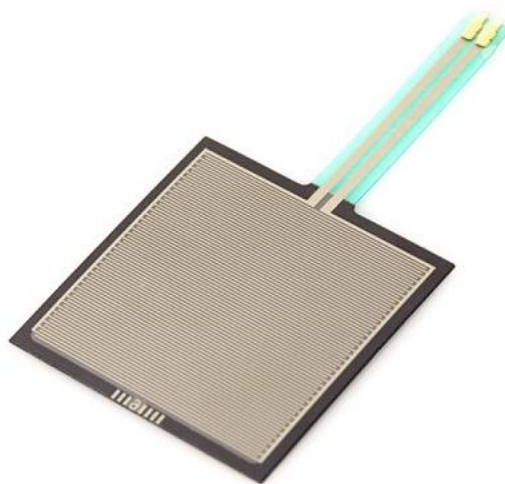


Ilustración 33. Sensor de peso FSR

Pero esta forma nos pareció más compleja y que nos podría acarrear mayor dificultad, además del hecho de que la pelota tendría poco peso (se escogerá una pelota de ping pong para facilitar los rebotes en la etapa anterior) lo que podría provocar problemas futuros.

La opción que se utilizó finalmente fue usar unos detectores ópticos.

Para este tipo de detección se han utilizado unos leds infrarrojos y unos fototransistores, que se van a describir a continuación.

Los leds son unos diodos infrarrojos que trabajan a 940nm de la marca Vishay.



Ilustración 34. Led infrarrojo

Forward Current 10µA para Reverse Voltage de 5V.

Fall Time de 15ns.

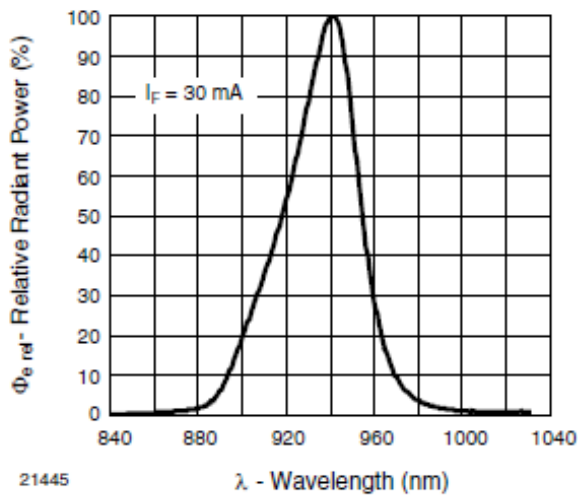


Ilustración 36. Gráfica de longitud de onda de trabajo del led

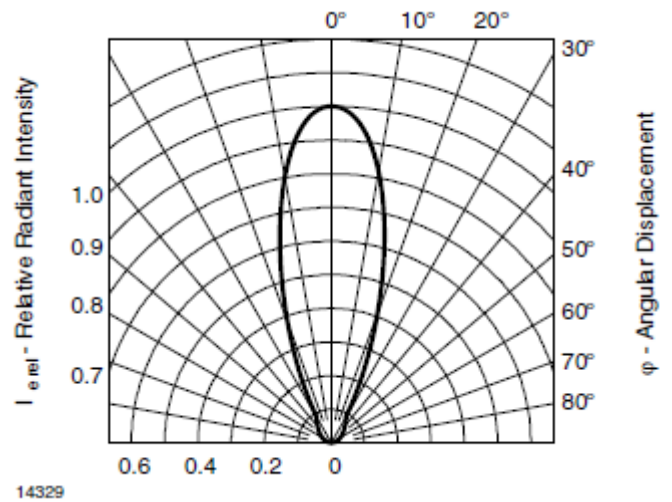


Ilustración 35. Gráfica de direccionalidad del led

Los fototransistores son del tipo NPN.



Ilustración 37. Fototransistor infrarrojo lateral

Este tipo de fototransistor es de tipo lateral, esta fue una de las características por las que se eligió, ya que sería muy útil para su colocación en el bloque.

Trabaja con una tensión de emisor-colector de 5V, que será la alimentación del circuito.

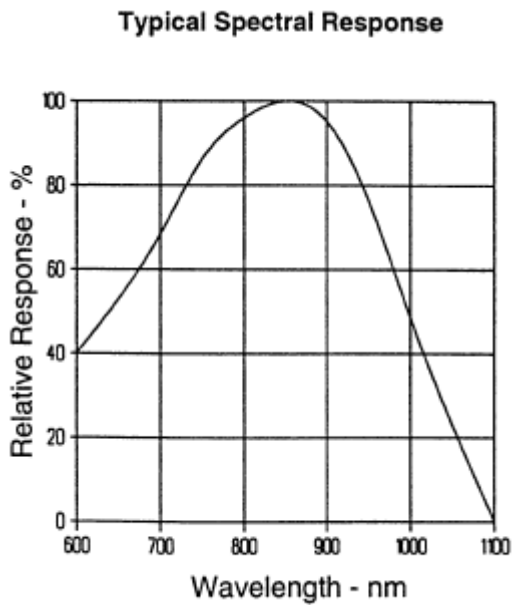


Ilustración 39. Gráfica de Responsabilidad espectral

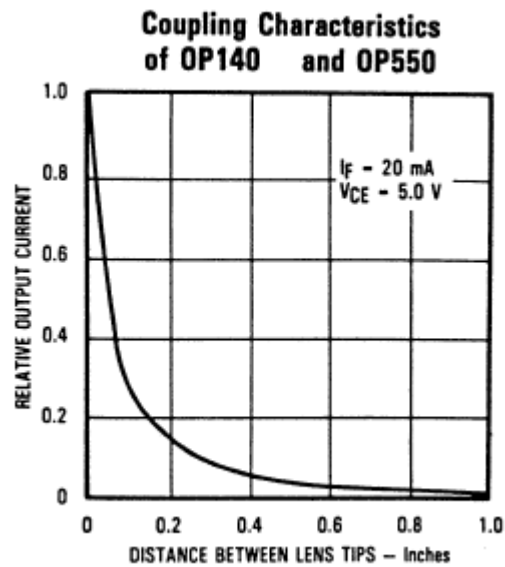


Ilustración 38. Gráfica de potencia recibida en función de la distancia

Los leds y los fotodetectores están enfrentados unos a otros de la forma más direccional posible, porque como ya hemos visto son dos componentes muy direccionales, por lo que sobre las casillas están colocados como en la Ilustración 40.



Ilustración 40. Colocación de la los led para la detección

Se puede ver en esta Ilustración 41 como van colocados los sistemas de detección, en las cinco casillas, aunque en la Ilustración solo vemos 1 casilla.

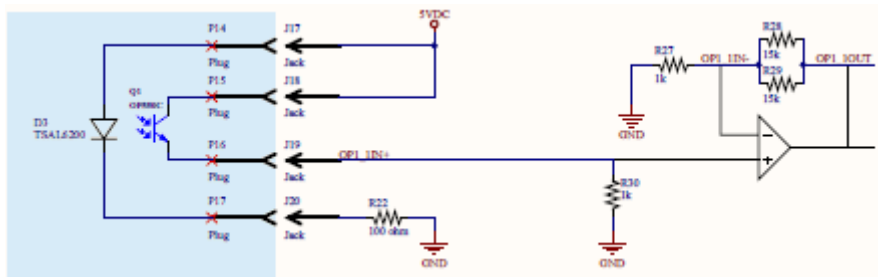


Ilustración 41. Conexión eléctrica del sistema de detección

El sistema de acondicionamiento que tiene es para el led una simple resistencia de 100 Ω conectada entre tierra y el cátodo y el ánodo puesto a la alimentación.

Para el fototransistor colocamos el colector a alimentación y el emisor a un amplificador operacional no inversor (Ilustración 42), en el que la R1 es de 1k, al igual que la R3 y en el que la R2 es de 7,5k pero para conseguirlo se colocan dos resistencias de 15k en paralelo.

Estos valores los hemos elegido para aumentar la tensión que nos llega que es de 400mV para los 10cm alimentando el led a 5V. Gracias a estos valores que se han elegido tenemos la siguiente ganancia.

$$G = 1 + \left(\frac{R2}{R1}\right) = 1 + \left(\frac{7,5}{1}\right) = 8.5$$

$$V_0 = 0.4 \cdot 8.5 = 3.4V$$

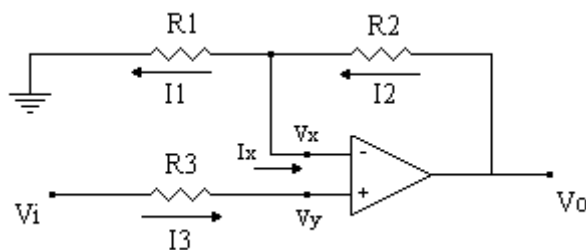


Ilustración 42. Amplificador no inversor

2.2.4 Panel de Control

Esta nueva parte es la interfaz con la que interactúa el profesor con el sistema, con ella es con la que el profesor selecciona el número de jugadores que van a participar, selección que realizará mediante tres botones. Y para poder saber qué está seleccionando en cada momento tendrá una pantalla LCD que se lo irá mostrando.

Los botones que se han empleado son de la marca RAFI y el modelo 1.10.001.011/0301 (Ilustración 43).



Ilustración 43. Botón para controlar el sistema

De este modelo de botón se usarán 3 ejemplares, con las siguientes funciones.

- El botón colocado sobre los otros es el de MOVER, con el que va cambiando el número de jugadores y de personaje.
- El botón de en medio es el de OK, y el encargado de confirmar las opciones que se han elegido y para poder jugar entre turno y turno.
- El botón de abajo es el de RESET, con este podemos parar la partida a mitad del juego y llevar los motores al puesto inicial, o bien hacerlo una vez acabada la partida para comenzar otra nuevamente.

Los botones llevan la siguiente conexión eléctrica.

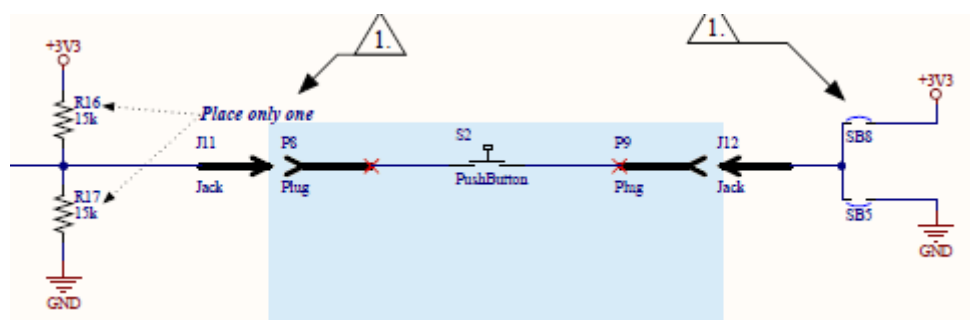


Ilustración 44. Conexión eléctrica de los botones de control

Los tres botones llevan la misma conexión eléctrica, y estos se accionan por flanco de subida, por lo que se deben soldar al pin SB8 y colocar solo la resistencia R17. Para la placa PCB se barajaron las dos opciones en la fase de diseño a la espera de recibir los botones adecuados en la fase de implementación; flanco de subida o bajada, por eso aparecen en el esquemático de la Ilustración 44.

La pantalla LCD es de la marca MIDAS y el modelo MC21605B6WR-BNMLW (Ilustración 45).

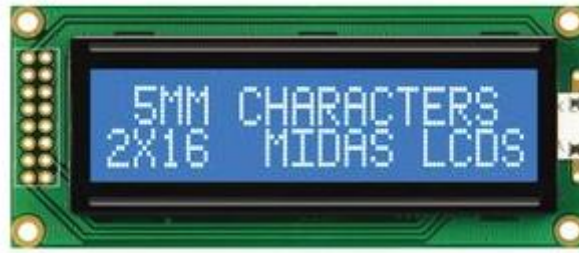


Ilustración 45. Pantalla LCD

Como se puede ver en la ilustración anterior, esta es una pantalla LCD de 16x2, con el fondo azul y los caracteres de 5 mm y en blanco, dispuestos en dos filas.

Principalmente ha sido elegida por:

- Ser alimenta a 5V.
- Bajo consumo de potencia.
- Ser compacta y ligera, ya que va a estar colocada en un lateral del bloque.

Los pines que tiene la pantalla se corresponde con la siguiente tabla. (10)

PIN	SYMBOL	FUNCTION
1	Vss	GND
2	Vdd	Power supply for LCM (+5.0V)
3	V0	Contrast Adjust
4	RS	Register Select Signal
5	R/W	Data Read / Write
6	E	Enable Signal
7-14	DB0 - DB7	Data bus line
15	LED+	Power supply for BKL (+5.0V)
16	LED-	Power supply for BKL (0V)

Ilustración 46. Descripción de los pines

Con estos pines controlamos la pantalla LCD que lleva internamente una serie de bloques con los que traduce la información que recibe y la muestra por la pantalla, como se indica en el datasheet, tiene una ROM donde se generan todos los caracteres a mostrar y una RAM donde se generan los caracteres más usados

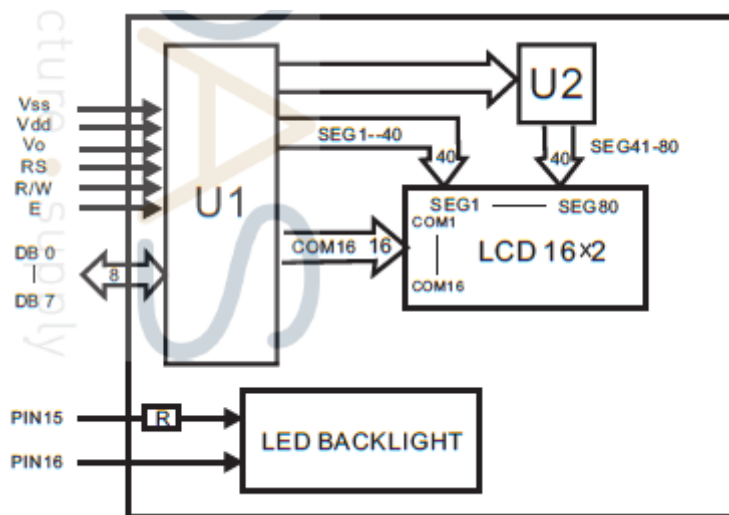


Ilustración 47. Configuración interna de la pantalla

Como vemos esta pantalla se controla con la comunicación en serie, en el datasheet del LCD que se encuentra en los anexos encontramos los comandos necesarios para controlarla.

El conexionado eléctrico que tiene esta pantalla es el siguiente

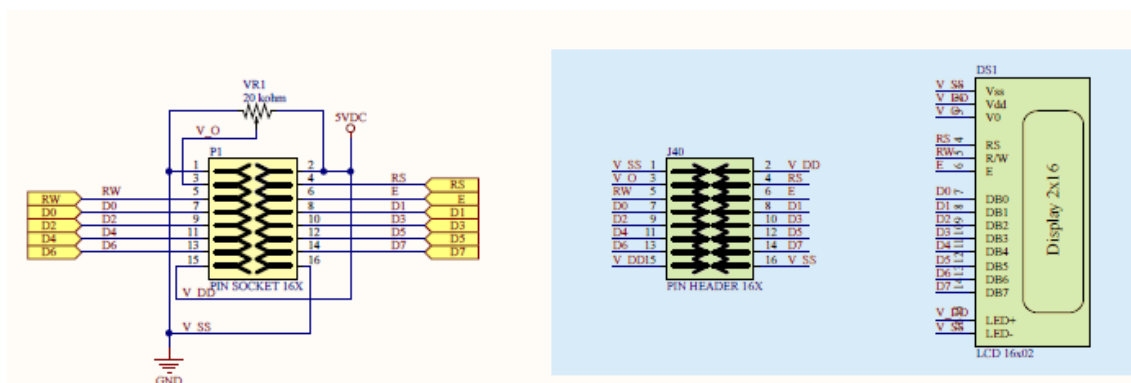


Ilustración 48. Conexionado eléctrico de la pantalla LCD

En la Ilustración 48 tenemos a la izquierda el bloque de pines hembra que hace de puente entre el microcontrolador y la pantalla LCD y a la derecha tenemos las conexiones de la pantalla LCD sobre este bloque. Este bloque que hace de puente entre los dos sistemas lo podemos ver en la Ilustración 48.

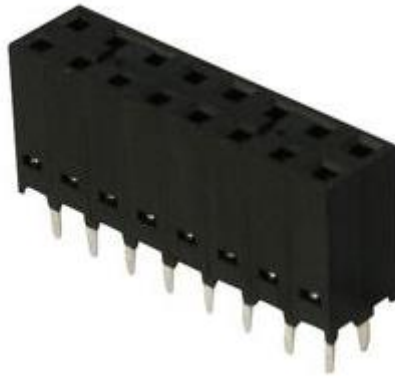


Ilustración 49. Bloque de 8x2 socket con agujero pasante de 2.54mm

Como se puede ver en este esquemático tenemos un potenciómetro de 20K con el cual podemos controlar y ajustar el brillo de la pantalla.

2.2.5 Sistema de transmisión

El sistema de transmisión que se ha decidido emplear para nuestro sistema es la radiofrecuencia, para ello se necesitan dos partes, una en cada bloque: para el bloque de la rampa es necesario la parte emisora y para el bloque de los carriles la receptora. En nuestro caso se ha optado por una solución de RFSolutions, debido a su uso en anteriores proyectos, de esta forma se tenía la certeza de su correcto funcionamiento.

El Bloque Rampa:

- Encoder: RF600E
- Transmisor: AM-RT4-433

El Bloque Carriles:

- Receptor: AM-HRR30-433
- Decoder: RF600D

Con este sistema de transmisión se va a trabajar a 433MHz de frecuencia, que es una frecuencia libre, la cual no requiere licencia para trabajar. El método de encriptación que se emplea para la transmisión es la codificación Manchester, que se basa en la codificación eléctrica de una señal binaria en la que cada tiempo de bit hay una transmisión entre los dos niveles de señal, en la cual cada bit puede obtenerse de la señal de reloj, lo que permite una sincronización del flujo de los datos: esto se conoce como codificación autosincronizada (8). Junto a este sistema durante la transmisión se manda también un CRC, que nos informará de que la señal llega correctamente al otro bloque. (9)

En este trabajo se explicará la parte emisora del sistema RF y la parte receptora se explicará en la de mi compañero Sergio Castillo.

2.2.5.1 Encoder

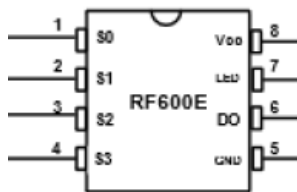


Ilustración 50. Circuito Integrador RF600E

El encoder es el circuito integrado RF600E, Ilustración 50. Este sistema es de fácil uso, tiene 4 entradas con las que se suministra la información a transmitir y los otros cuatro pines son la alimentación y la señal que va al módulo transmisor. Este sistema tiene un tiempo de retardo de 6.5ms como mecanismo de anti rebote.

Tabla 1. Tabla de codificación de los comandos

```

/***** TABLA DE COMANDOS RF *****/
*
*          S3  S2  S1  S0
* 0x00 Default  0  0  0  0
* 0x01 Player1  0  0  0  1
* 0x02 Player2  0  0  1  0
* 0x03 Player3  0  0  1  1
* 0x04 Player4  0  1  0  0
* 0x05 Character1 0  1  0  1
* 0x06 Character2 0  1  1  0
* 0x07 Character3 0  1  1  1
* 0x08 Character4 1  0  0  0
* 0x09 Points10  1  0  0  1
* 0x0A Points15  1  0  1  0
* 0x0B Points20  1  0  1  1
* 0x0C Restart   1  1  0  0
* 0x0D EndGame   1  1  0  1
* 0x0E InitialPos 1  1  1  0
* 0x0F NotUsed   1  1  1  1
*****/
    
```

Este componente es el encargado de codificar la información que se desea transmitir al otro bloque, ya sea el número de jugadores, el personaje que selecciona cada jugador o la puntuación adquirida por cada uno, como se puede ver en la tabla 1.

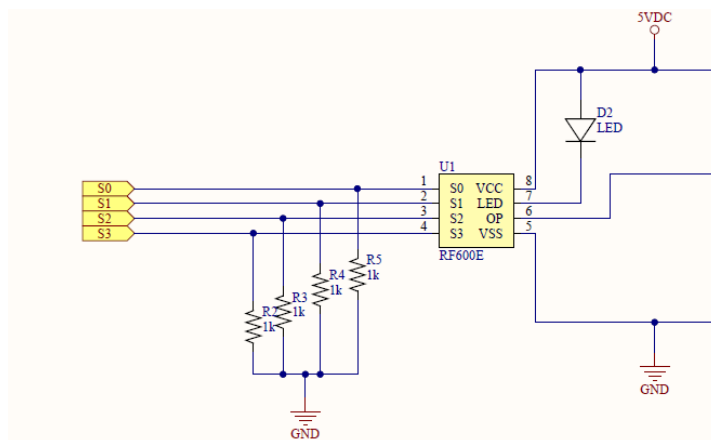


Ilustración 51. Conexión de RF600E

En la Ilustración 51, se puede ver cómo está conectado el sistema.

Primero se puede ver las conexiones entre las salidas del micro y las entradas del encoder, en las cuales tenemos entremedias unas resistencias de 1k conectadas a tierra.

La salida 8 va conectada a alimentación 5V.

La salida 5 es la conectada a tierra.

La salida 7 es la del led, a la que se conecta la parte del cátodo del led, y la parte del ánodo va conectada a la alimentación: este led nos informa cuándo está transmitiendo el sistema.

La salida 6 es la que lleva la información codificada mediante el sistema Manchester que se comentó anteriormente y el sistema CRC de comprobación, que se conecta al transmisor AM-RT4-433.

2.2.5.2 Transmisor

Se utiliza el circuito integrado AM-RT4-433 (Ilustración XX)

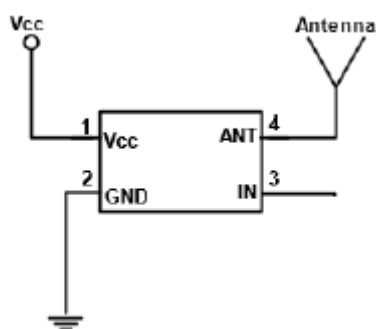


Ilustración 52. Esquemático AM-RT4-433

Es un sistema de transmisión que utiliza la modulación en amplitud (AM) para transmitir la información hasta los 4kHz de cualquier estándar CMOS o TTL.

Este circuito integrado modula la señal que le llega a través del encoder anterior con el sistema serie, generando una onda electromagnética en el espectro de radiofrecuencia. De ahí el uso de componentes capacitivos e inductivos que tiene el circuito integrado para no interferir en la sintonización del dispositivo.

Este sistema puede incorporar una antena para mejorar la transmisión, pero para nuestro sistema debido a toda la electrónica que contiene, ésta se hace indispensable para el correcto funcionamiento, ya que sin ello no transmite correctamente y no llegan los datos al sistema de recepción.

La longitud de la antena viene determinada por la frecuencia de trabajo, y la obtenemos con las siguientes fórmulas:

$$\lambda = \frac{c}{f} = \frac{300 \cdot 10^6 [m/s]}{433MHz} = 0.693m$$

Una vez que tenemos la longitud de onda obtendremos la longitud de la antena.

$$l = \frac{\lambda}{4} = 17.3 \text{ cm}$$

2.2.6 Alimentación

El sistema de alimentación que se ha empleado para este bloque del proyecto está formado por una fuente de alimentación de 5 voltios y un botón de encendido/apagado.

La fuente que se ha usado es una Traco Power TXM 025-105 (Ilustración 53).



Ilustración 53. Fuente de alimentación TXM 025-105

Todo nuestro sistema estará alimentado a 5 voltios y aporta 5 amperios de máxima corriente para el sistema. A priori, esto podría parecer suficiente, ya que los servos consumen unos 200 mA cada uno, y como hay cinco, quedarían 4 A para el resto del bloque. Pero aunque parezca que la fuente está sobredimensionada para los requisitos del sistema, pueden aparecer sobre picos de corriente cuando se activen todos los servomotores a la vez o cuando se cambie de dirección a los servomotores, y de ahí la salvaguarda.

Para el encendido/apagado de la fuente se usa un botón colocado en el lateral de la rampa.

Para ello se utiliza el botón MC34231-091-72 de Multicomp, este botón es retroiluminado, soporta hasta 250V de alterna y es compatible con los conectores hembra aéreo de la serie FASTON 250 de TE Connectivity, que son los que usaremos para todo nuestro sistema.

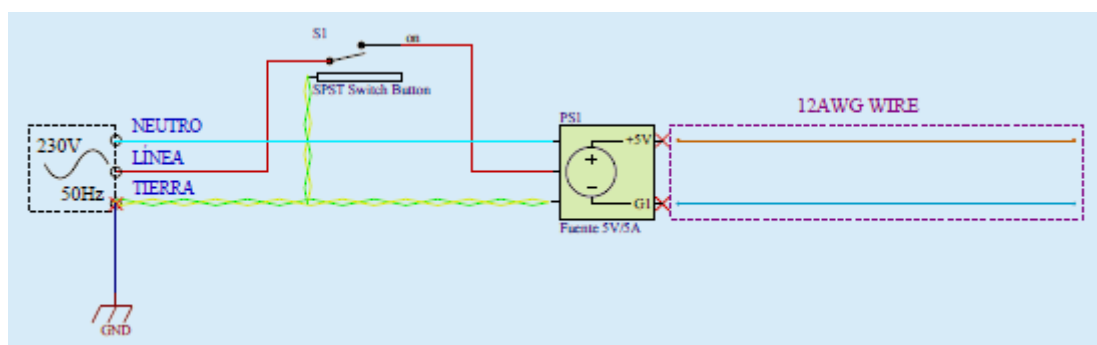


Ilustración 54. Conexión eléctrica del sistema de alimentación

2.2.7 Microcontrolador

El microcontrolador que se ha utilizado para controlar el bloque de la Rampa es el mismo que se utilizará para controlar el de los carriles, y es una placa de desarrollo de STMicroelectronics, el modelo es NUCLEO-F411RE (Ilustración 55).

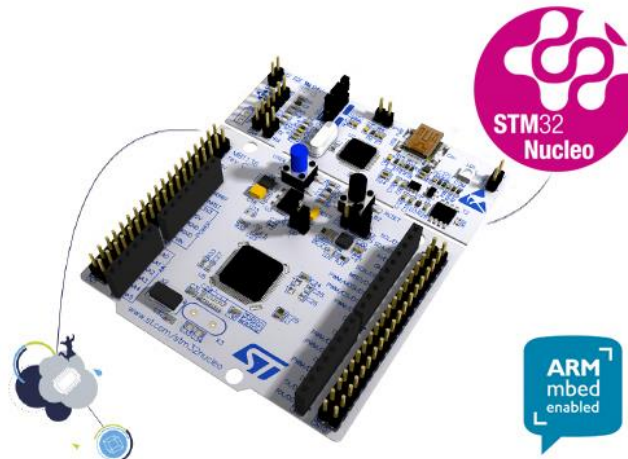


Ilustración 55. Placa de desarrollo NUCLEO-F411RE

Las principales características de esta placa son:

- Microcontrolador embebido STM32F411RE, de 32 bits y 64 pines.
- Posibilidad de depurar programa a través del ST-LINK, con conexión usb tipo mini-b.
- Dos pulsadores (“reset” y “user”).
- Posibilidad de alimentar a 3.3V, a 5V o a 12V.

El microcontrolador que tiene embebido posee una memoria flash de 512Kbytes, lo cual es más que de sobra para poder programar sin preocuparse del espacio que pueda ocupar. Capaz de trabajar a una frecuencia de hasta 26 MHz, tiene un consumo en funcionamiento normal de 100uA y de 2.4uA en modo Standby. Posee 11 timers, hasta 81 GPIO's (de los cuales la mayoría toleran hasta 5V), interfaz de comunicación i2c, spi, usart...

Muchas de estas características hacen que sea un microcontrolador ideal para trabajar con él. Así todo, uno de los motivos principales por el que se escogió como solución definitiva fue su compatibilidad con el software STM32CubeMX. Este programa permite, previa selección del microcontrolador que se desee (debe ser un STM de 32 bits), la modificación de parámetros tales como configuración de los pines (como I/O -pullup, pulldown-, timers, USART...), reloj que gobierna el micro y cada uno de los buses de cada periférico.

Una vez modificados los parámetros, el programa genera el código de inicialización en C necesario cargar en el microcontrolador para configurar todos los periféricos según las necesidades del proyecto, lo cual permite la liberación de una carga importante de trabajo por parte del programador.

Tabla 2. Configuración de los Pines

IP	Pin	Signal	GPIO mode	GPIO pull/up pull down	Max Speed	User Label
TIM3	PA6	TIM3_CH1	Alternate Function Push Pull	No pull-up and no pull-down	Medium *	SERVO1
	PA7	TIM3_CH2	Alternate Function Push Pull	No pull-up and no pull-down	Medium *	SERVO2
	PB0	TIM3_CH3	Alternate Function Push Pull	No pull-up and no pull-down	Medium *	SERVO3
TIM4	PB6	TIM4_CH1	Alternate Function Push Pull	No pull-up and no pull-down	Medium *	SERVO4
	PB7	TIM4_CH2	Alternate Function Push Pull	No pull-up and no pull-down	Medium *	SERVO5
	PB8	TIM4_CH3	Alternate Function Push Pull	No pull-up and no pull-down	Medium *	SERVO6
Single Mapped Signals	PC14-OSC32_IN	RCC_OSC32_IN	n/a	n/a	n/a	
	PC15-OSC32_OUT	RCC_OSC32_OUT	n/a	n/a	n/a	
	PH0 - OSC_IN	RCC_OSC_IN	n/a	n/a	n/a	
	PH1 - OSC_OUT	RCC_OSC_OUT	n/a	n/a	n/a	
	PA2	USART2_TX	Alternate Function Push Pull	No pull-up and no pull-down	Very High *	USART_TX
	PA3	USART2_RX	Alternate Function Push Pull	No pull-up and no pull-down	Very High *	USART_RX
	PA13	SYS_JTMS-SWDIO	n/a	n/a	n/a	TMS
	PA14	SYS_JTCK-SWCLK	n/a	n/a	n/a	TCK
	PB3	SYS_JTDO-SWO	n/a	n/a	n/a	SWO
GPIO	PC13-ANTI_TAMP	GPIO_EXTI13	External Event Mode with Rising edge trigger detection *	No pull-up and no pull-down	n/a	B1 [Blue PushButton]
	PC0	GPIO_EXTI0	External Interrupt Mode with Falling edge trigger detection	Pull-down *	n/a	IR1
	PC1	GPIO_EXTI1	External Interrupt Mode with Falling	Pull-down *	n/a	IR2

Tabla 3. Configuración de los pines

IP	Pin	Signal	GPIO mode	GPIO pull/up pull down	Max Speed	User Label
			edge trigger detection			
	PC2	GPIO_EXTI2	External Interrupt Mode with Falling edge trigger detection	Pull-down *	n/a	IR3
	PC3	GPIO_EXTI3	External Interrupt Mode with Falling edge trigger detection	Pull-down *	n/a	IR4
	PA4	GPIO_EXTI4	External Interrupt Mode with Falling edge trigger detection	Pull-down *	n/a	IR5
	PA5	GPIO_Output	Output Push Pull	No pull-up and no pull-down	Low	LD2 [Green Led]
	PC5	GPIO_EXTI5	External Interrupt Mode with Rising edge trigger detection	No pull-up and no pull-down	n/a	BOTONROJO
	PB1	GPIO_Output	Output Push Pull	Pull-down *	Medium *	S0
	PB10	GPIO_Output	Output Push Pull	No pull-up and no pull-down	Medium *	RS
	PB12	GPIO_Output	Output Push Pull	No pull-up and no pull-down	Medium *	RW
	PB13	GPIO_Output	Output Push Pull	Pull-down *	Medium *	S3
	PB14	GPIO_Output	Output Push Pull	Pull-down *	Medium *	S2
	PB15	GPIO_Output	Output Push Pull	Pull-down *	Medium *	S1
	PC6	GPIO_Output	Output Push Pull	No pull-up and no pull-down	Medium *	E
	PC7	GPIO_Output	Output Push Pull	Pull-down *	Medium *	D0
	PC8	GPIO_Output	Output Push Pull	Pull-down *	Medium *	D1
	PC9	GPIO_Output	Output Push Pull	Pull-down *	Medium *	D2
	PA8	GPIO_Output	Output Push Pull	Pull-down *	Medium *	D3
	PA9	GPIO_Output	Output Push Pull	Pull-down *	Medium *	D4
	PA10	GPIO_Output	Output Push Pull	Pull-down *	Medium *	D5
	PA11	GPIO_Output	Output Push Pull	Pull-down *	Medium *	D6
	PA12	GPIO_Output	Output Push Pull	Pull-down *	Medium *	D7
	PC10	GPIO_EXTI10	External Interrupt Mode with Rising edge trigger detection	No pull-up and no pull-down	n/a	BOTON1
	PC11	GPIO_EXTI11	External Interrupt Mode with Rising edge trigger detection	No pull-up and no pull-down	n/a	BOTON2
	PC12	GPIO_EXTI12	External Interrupt Mode with Rising edge trigger detection	No pull-up and no pull-down	n/a	BOTON3

CAPITULO 3. IMPLEMENTACION DEL SISTEMA

3.1 Descripción del firmware

El sistema del juego consta de dos programas distintos pero conectados entre sí, ya que el programa del bloque de los carriles depende en su totalidad de los comandos que son emitidos por radiofrecuencia por parte del programa de la rampa.

En esta memoria se va a explicar las principales partes del programa del bloque rampa.

Para ello primero se muestra un diagrama de bloques del main.

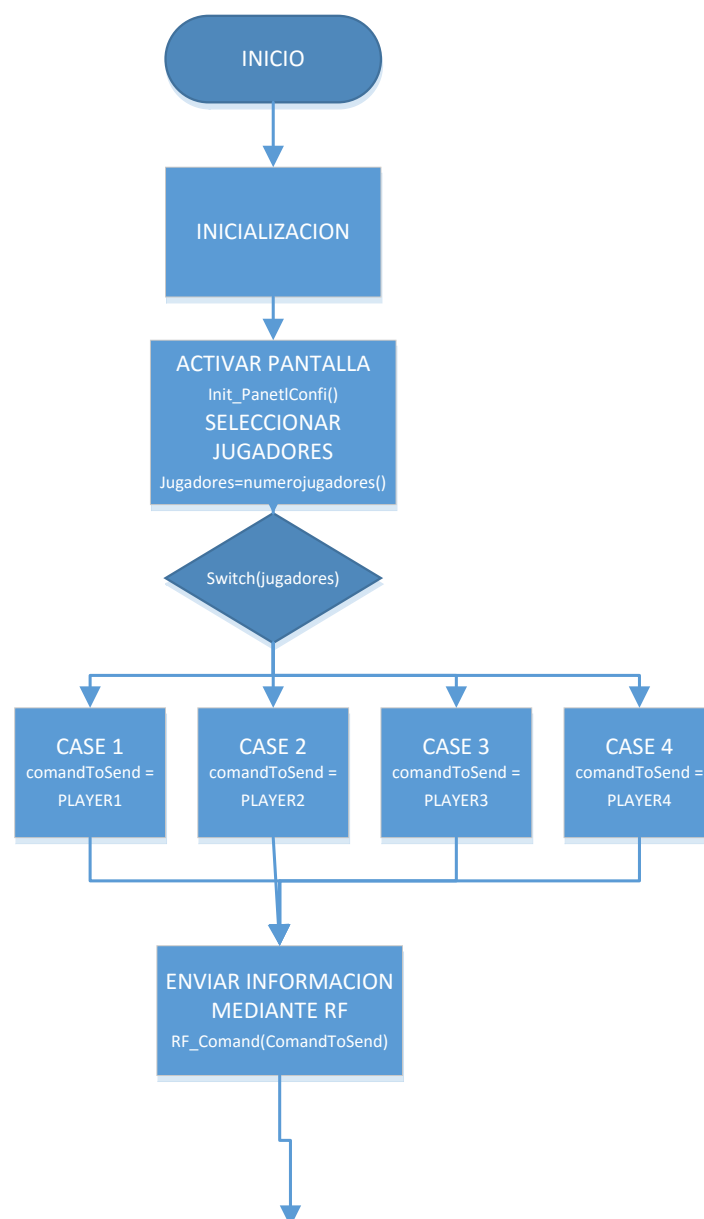


Ilustración 58. Diagrama de Bloque Rampa (1)

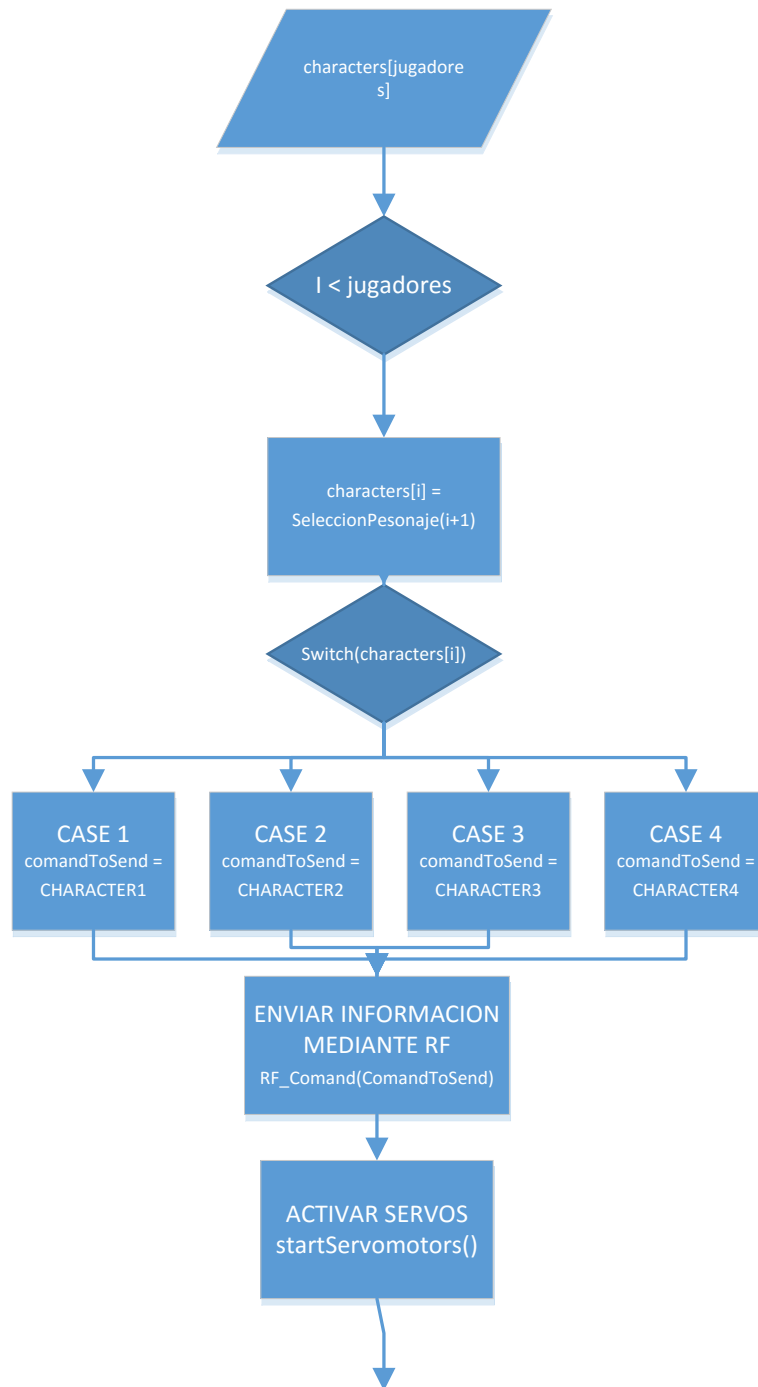


Ilustración 59. Diagrama de Bloque Rampa (2)

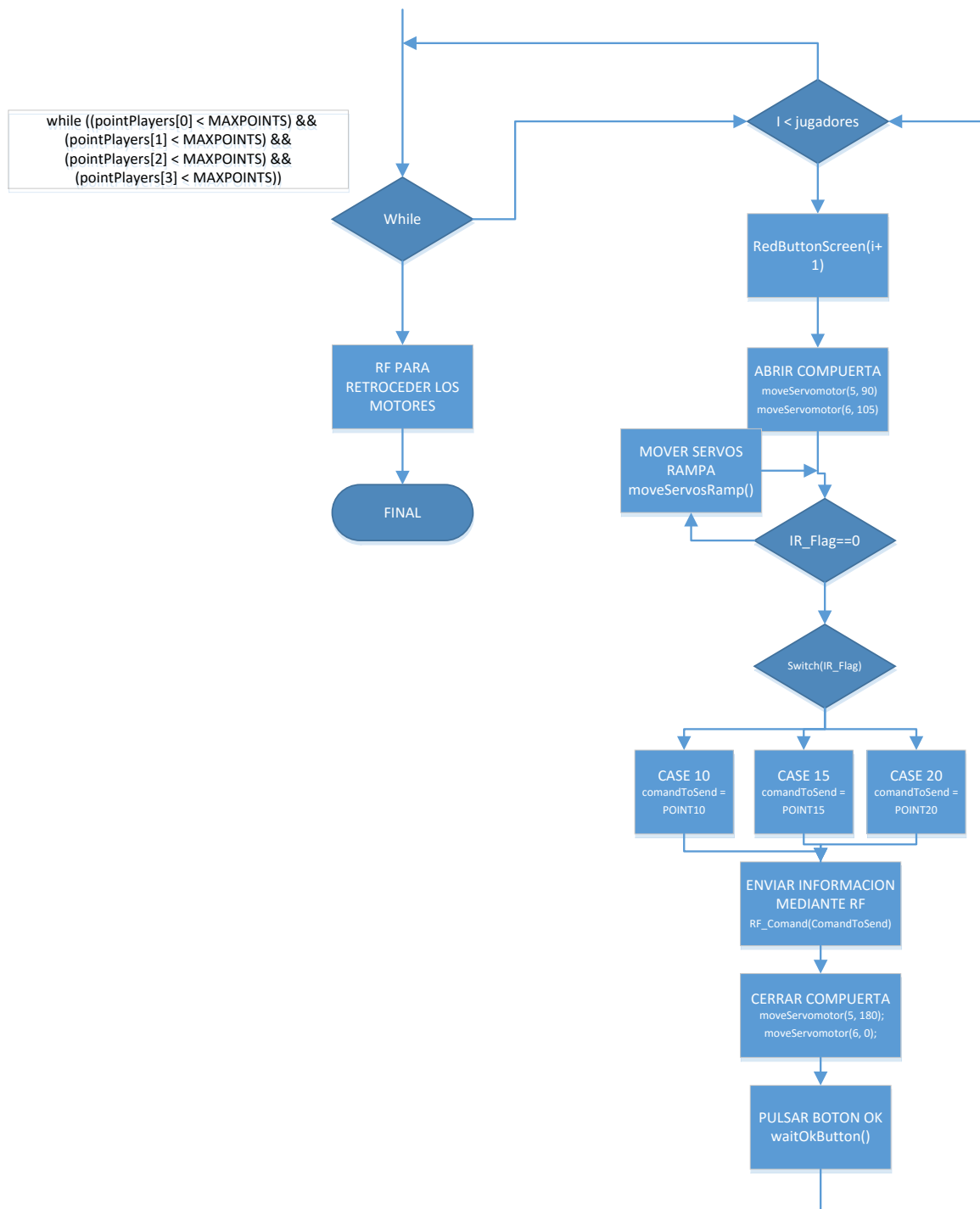


Ilustración 60. Diagrama de Bloque Rampa (3)

El programa comienza poniendo todas las variables a su estado inicial, por si se paró el programa de forma inesperada la última vez, e inicializando todos sus periféricos.

Después de esta parte comienza el menú del programa, que se mostrará por la pantalla LCD, gracias a la función numeroJugadores(), que se encuentra en controlpanel.c, donde se detallan las acciones que hay que hacer para mostrar por pantalla un mensaje u otro según el número de jugadores en el que se esté.

La selección se hace en el main con un menú que es un switch, con el que indicamos el número de jugadores que queremos para esta partida (Ilustración61).

```
switch (jugadores)
{
case 4:
    commandToSend = PLAYER4;
    break;
case 3:
    commandToSend = PLAYER3;
    break;
case 2:
    commandToSend = PLAYER2;
    break;
case 1:
    commandToSend = PLAYER1;
    break;
}
```

Ilustración 61. Switch para seleccionar número de jugadores

Como se ve según el número que se elija se le asigna el PLAYER correspondiente a la variable commandToSend, ya que esta es la variable que usamos para enviar la información por radiofrecuencia.

Para mandar esta información usamos la función RF_Command (commandToSend). Esta función se encuentra en el fichero RF.c, desarrollada.

Esta función lo que hace es leer y comprobar el valor que tenga la variable commandToSend, esta función se puede ver claramente en la Ilustración 62.

```
void RF_Command (uint8_t RF_Command)
{
    int i;
    uint8_t singleBit = 0;
    for(i = 3; 0 <= i; i --)    //extrae los 4 bits menos significativos del Byte
    {
        singleBit = (RF_Command >> i) & 0x01;
        switch (i)
        {
            case 3:    //lectura bit mas significativo
                if (singleBit == 0)
                    {HAL_GPIO_WritePin(S3_GPIO_Port, S3_Pin, GPIO_PIN_RESET);}
                else
                    {HAL_GPIO_WritePin(S3_GPIO_Port, S3_Pin, GPIO_PIN_SET);}
                break;
            case 2:    //lectura segundo bit mas significativo
                if (singleBit == 0)
                    {HAL_GPIO_WritePin(S2_GPIO_Port, S2_Pin, GPIO_PIN_RESET);}
                else
                    {HAL_GPIO_WritePin(S2_GPIO_Port, S2_Pin, GPIO_PIN_SET);}
                break;
            case 1:    //lectura segundo bit menos significativo
                if (singleBit == 0)
                    {HAL_GPIO_WritePin(S1_GPIO_Port, S1_Pin, GPIO_PIN_RESET);}
                else
                    {HAL_GPIO_WritePin(S1_GPIO_Port, S1_Pin, GPIO_PIN_SET);}
                break;
            case 0:    //lectura bit menos significativo
                if (singleBit == 0)
                    {HAL_GPIO_WritePin(S0_GPIO_Port, S0_Pin, GPIO_PIN_RESET);}
                else
                    {HAL_GPIO_WritePin(S0_GPIO_Port, S0_Pin, GPIO_PIN_SET);}
                break;
            default:
                break;
        }
    }
}
} //RF Command
```

Ilustración 62. Función para enviar los datos por RF

Después de enviar la información, se espera un tiempo que marcamos con un Delay, volvemos a cambiar la variable para ponerla a 0 y la mandamos para decir al sistema que pasamos a otra tarea porque si no seguiría mandando la información anterior todo el rato, estas sentencias se ven en la Ilustración 63.

```
RF_Command(commandToSend);
HAL_Delay(160);
commandToSend = 0;
RF_Command(commandToSend);
```

Ilustración 63. Comandos para enviar la información y luego limpiarla

Ahora creamos un array. El tamaño viene definido por el número de jugadores que se indicó, este se ha creado para guardar el personaje que va a elegir cada jugador a continuación.

Con la función selecciónpersonaje() mostramos por pantalla la opciones que tiene el jugador en cuestión para elegir un personaje, dentro de esta función se llama a otra función que es checkpersonaje(), que es con la que se comprueba qué personajes se pueden mostrar por la pantalla, ya que si el personaje ya ha sido elegido por otro jugador ya no se muestra para su elección. Estas dos funciones se encuentran en panelcontrol.c.

Mediante un bucle for realizamos la selección del personaje tantas veces como jugadores se hayan elegido, Ilustración 64, y como ya se hizo anteriormente mandamos la información al otro bloque.

```
for ( i=0 ; i<jugadores ; i++)
{
    characters[i] = seleccionPersonaje(i+1);

    switch(characters[i])
    {
        case 1:
            commandToSend = CHARACTER1;
            break;
        case 2:
            commandToSend = CHARACTER2;
            break;
        case 3:
            commandToSend = CHARACTER3;
            break;
        case 4:
            commandToSend = CHARACTER4;
            break;
    }
    RF_Command(commandToSend);
    commandToSend = 0;
    HAL_Delay(160);
    RF_Command(commandToSend);
}
```

Ilustración 64. Bucle For para seleccionar los personajes

Una vez hecho todo esto ya se habría acabado con la selección de los jugadores y personajes, esta es la parte que debe hacer el monitor para preparar el juego, antes de que los alumnos del colegio puedan utilizarlo.

Se inicializan los servomotores y se colocan los servos de la puerta en posición por si se hubieran movido manualmente, mientras el juego estuviera apagado, con la función startServomotors () (Ilustración 65).

```
void startServomotors (void)
{
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2);
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_3);
    HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_1);
    HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_2);
    HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_3);

    HAL_TIM_Base_Start_IT(&htim3);
} //startServomotors
```

Ilustración 65. Inicialización de los servomotores

Y con la función moveServomotor (), se muestra en la Ilustración 66, se mueven los motores que se quieran, la primera variable que se coloca es la que indica qué servomotor se quiere mover, y con la segunda variable se le indica el número de grados que se quiere mover. Como se le va a indicar entre 0 y 180, se ha necesitado crear una función que transforma el valor de esos grados en el ciclo de trabajo que queremos.

Debido a que nuestra señal PWM genera el pulso a los 2650, según la configuración de reloj con la que se ha programado el micro, con periodo de 20000 y pre escalado 84.

$$CCR\text{X}0 = (m \cdot \text{degrees}) + n$$

$$\text{Siendo } n=2650 \text{ y } m=-\frac{1700-750}{210-105} = -9.05$$

```
void moveServomotor(uint8_t servoMotor, double degrees)
{ //y=mx+n      ->  n=2650, m=-((1700-750)/(210-105))=-9.05, y=ccrx, x= degrees
  uint32_t CCRX = 0;
  double m = -9.05;
  double n = 2650;
  CCRX = round((m*degrees) + n);
  if (CCRX > 2650) {CCRX = 2650;}
  if (CCRX < 750) {CCRX = 750;}

  switch(servoMotor)
  {
  case 1:      TIM3->CCR1=CCRX;
               break;
  case 2:      TIM3->CCR2=CCRX;
               break;
  case 3:      TIM3->CCR3=CCRX;
               break;
  case 4:      TIM4->CCR1=CCRX;
               break;
  case 5:      TIM4->CCR2=CCRX;
               break;
  case 6:      TIM4->CCR3=CCRX;
               break;
  }
} //moveServomotor
```

Ilustración 66. Función para mover los motores los grados que uno quiera

Ahora se procede a continuar con el programa mediante un While que mantiene el juego activo hasta que uno de los jugadores consiga la puntuación máxima (Ilustración 67).

```
while ((pointPlayers[0] < MAXPOINTS) && (pointPlayers[1] < MAXPOINTS) && (pointPlayers[2] < MAXPOINTS) && (pointPlayers[3] < MAXPOINTS))
```

Ilustración 67. Declaración del bucle que hace nuestro juego hasta que acaba

Dentro de este while tenemos un for que nos va marcando los turnos, ya que éste evitará que el programa acabe cuando uno consiga la puntuación máxima y esperará a que acabe el turno, porque puede que varios jugadores consigan la puntuación máxima durante el mismo turno, todo esto se puede ver en la Ilustración 69.

En el for encontramos la función redButtonScreen() al principio, y es la encargada de mostrar por pantalla LCD que está esperando a que se pulse el pulsador en cuestión. Una vez se pulsa, la variable botón_Flag se cambia a 4 y por pantalla se muestra que está cayendo la pelota.

Se mueven los motores que abren la puerta para dejar caer la pelota por la rampa.

La variable servos_Cnt es igual a 24, para que los servos de la rampa no tarden en exceso en entrar en acción y de esta forma se activan casi de forma simultánea, ya que se activan cuando llega la cuenta del reloj a 25, esta aumenta 1 cada vez que se genera el tim_it_update, y esto se produce cada 20ms, ya que nosotros queremos que los servos se muevan cada medio segundo. En la siguiente ecuación vemos por qué es 25 y no otro número.

$$\frac{0.5s}{20^{-3}s} = 25$$

```
void moveServosRamp (void)
{
    uint8_t random = 0;
    do {random = rand();} while (random > 180);
    moveServomotor(1, random);
    do {random = rand();} while (random > 180);
    moveServomotor(2, random);
    do {random = rand();} while (random > 180);
    moveServomotor(3, random);
    do {random = rand();} while (random > 180);
    moveServomotor(4, random);
}
```

Ilustración 68. Función moveServosRamp ()

La función moveServosRamp(), ver Ilustración 68, da valores aleatorios de la variable degrees a cada uno de los servos de la rampa con la función rand(), y se llama a la función moveServomotor().

```
for (i=0 ; i<jugadores ; i++)
{
    redButtonScreen(i+1);

    moveServomotor(5, 90);
    moveServomotor(6, 105); /
    servos_Cnt = 24; /
    IR_Flag = 0;

    while(IR_Flag == 0)
    {
        if (servos_Cnt == 25)
        {
            moveServosRamp();
            servos_Cnt = 0;
        }
    }
}
```

Ilustración 69. Parte del programa de la apertura de la puerta hasta que cae la pelota en una casilla

Estos seguirán moviéndose mientras siga cayendo la pelota, la forma de detectar que la pelota ya ha caído es mediante los led y fotodetectores, que provocan una interrupción que cambia el valor del IR_Flag por la puntuación que obtiene cada personaje (Ilustración 70).

```
switch(IR_Flag)
{
    case 10:
        pointPlayers[i] = pointPlayers[i] + 10;
        pointPlayersScreen(10, i+1);
        commandToSend = POINTS10;
        break;
    case 15:
        pointPlayers[i] = pointPlayers[i] + 15;
        pointPlayersScreen(15, i+1);
        commandToSend = POINTS15;
        break;
    case 20:
        pointPlayers[i] = pointPlayers[i] + 20;
        pointPlayersScreen(20, i+1);
        commandToSend = POINTS20;
        break;
}
```

Ilustración 70. Detección de cada casilla

Según el case en el que se entre se le suma la puntuación al elemento correspondiente en el array de cada jugador para que se lleve una cuenta de ello.

La siguiente línea activa la pantalla LCD para mostrar la puntuación que se ha conseguido, esta es una función en panelcontrol.c.

La tercera línea asigna a la variable `commandToSend` la puntuación que se ha obtenido en este turno, para que mueva el motor paso a paso del otro bloque correspondiente al jugador que le toca esta tirada.

Se vuelve a mandar la información con la función `RF_Command(commandToSend)`, limpiar la variable, esperar con el `Delay` y volver a mandar la información para decir que se acabó la orden, como se vio anteriormente en la Ilustración 63.

Se coloca la puerta en la posición correcta para colocar de nuevo la pelota y que no se caiga, y se muestra por pantalla el mensaje de “Pulsa OK para continuar”, y no se puede continuar con el juego hasta que no se pulse OK debido a la función `waitOkButton()`. Esta función se colocó por seguridad, para que no hubiera problemas si al cambiar el pulsador se entrase en la función `redButtonScreen()` por accidente, debido a que se produzca un pico de tensión y se activara la interrupción en el programa de que se ha pulsado el pulsador.

Una vez que los monitores hayan cambiado el pulsador Jack por otro, y el alumno esté preparado, se pulsa el OK, para que el sistema ya reconozca que se ha pulsado el botón y reaccione, porque si no se pulsa OK, cualquier pulsación que se realice pasará desapercibida para el sistema.

La función `waitOkButton()` está desarrollada en `panelcontrol.c`, y esta el programa parado hasta que se pulsa el botón OK. Si se pulsa el de `MOVER` no hace nada, sigue en pausa. Cuando se ha pulsado el programa continúa con el `for` para que tiren todos los jugadores del turno, y en caso de que ya hayan tirado todos los jugadores, se vuelve al principio del `while`. Si ninguno ha conseguido la puntuación máxima sigue y entra de nuevo en el `for`.

En la Ilustración 71 se puede ver el bucle `while` al completo, que engloba lo que se ha comentado anteriormente.

```
while ((pointPlayers[0] < MAXPOINTS) && (pointPlayers[1] < MAXPOINTS) && (pointPlayers[2] < MAXPOINTS) && (pointPlayers[3] < MAXPOINTS))
{
    for (i=0 ; i<jugadores ; i++)
    {
        redButtonScreen(i+1);

        moveServomotor(5, 90);
        moveServomotor(6, 105); //para "igualar" con respecto al otro servo se pone 105 en vez de 90
        servos_Cnt = 24; //para evitar delay que se produce entre apertura y mov de servos rampa
        IR_Flag = 0;

        while(IR_Flag == 0)
        {
            if (servos_Cnt == 25) //cada vez que tim_it_update se genera se aumenta el contador
            { //tim update se genera cada 20ms, y queremos mover los servos de la
            }
        }
        switch(IR_Flag)
        {
            RF_Command(commandToSend);
            commandToSend = 0;
            HAL_Delay(160);
            RF_Command(commandToSend);

            moveServomotor(5, 180);
            moveServomotor(6, 0);

            lcd_clear();
            lcd_on();
            lcd_goto(0,0);
            lcd_puts("PULSE OK PARA");
            lcd_goto(0,1);
            lcd_puts("CONTINUAR.");

            waitOkButton();
        }
    }
}
```

Ilustración 71. Bucle de las tiradas

Una vez consiga salir de este while, se activa la función stopServomotors(), Ilustración 72, que lo que hace es modificar el ciclo de trabajo de las funciones PWM de los Timers 1 y 3, para que se pongan a cero, con la variable CCRX=2650. Y luego parará los Timers.

```
void stopServomotors (void)
{
    uint32_t CCRX = 2650;

    TIM3->CCR1=CCRX;
    TIM3->CCR2=CCRX;
    TIM3->CCR3=CCRX;
    TIM4->CCR1=CCRX;
    TIM4->CCR2=CCRX;
    TIM4->CCR3=CCRX;

    HAL_TIM_Base_Stop_IT(&htim3);

    HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_1);
    HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_2);
    HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_3);
    HAL_TIM_PWM_Stop(&htim4, TIM_CHANNEL_1);
    HAL_TIM_PWM_Stop(&htim4, TIM_CHANNEL_2);
    HAL_TIM_PWM_Stop(&htim4, TIM_CHANNEL_3);
} //stopServomotors
```

Ilustración 72. Función para parar todos los servomotores.

3.2 Implementación de la Electrónica

Para poder realizar y desarrollar este proyecto, se tuvieron que hacer pruebas iniciales de los distintos componentes electrónicos antes de poder ponerlos en la placa final y de su diseño previo, por lo que se tuvo que estudiar y comprobar los componentes que se van a emplear.

Se hicieron pruebas para la zona de detección.

Estas pruebas que se realizaron no solo eran para comprobar el funcionamiento de los leds y los fototransistores, ya que había que comprobar el que se detectaran correctamente entre sí con sus longitudes de onda de trabajo. Esta detección tenía que realizarse con una distancia concreta, porque tiene que cubrir el hueco de las casillas, y hay que conseguir un compromiso con la distancia y el voltaje y que estén acordes para poder consumir una potencia mínima.

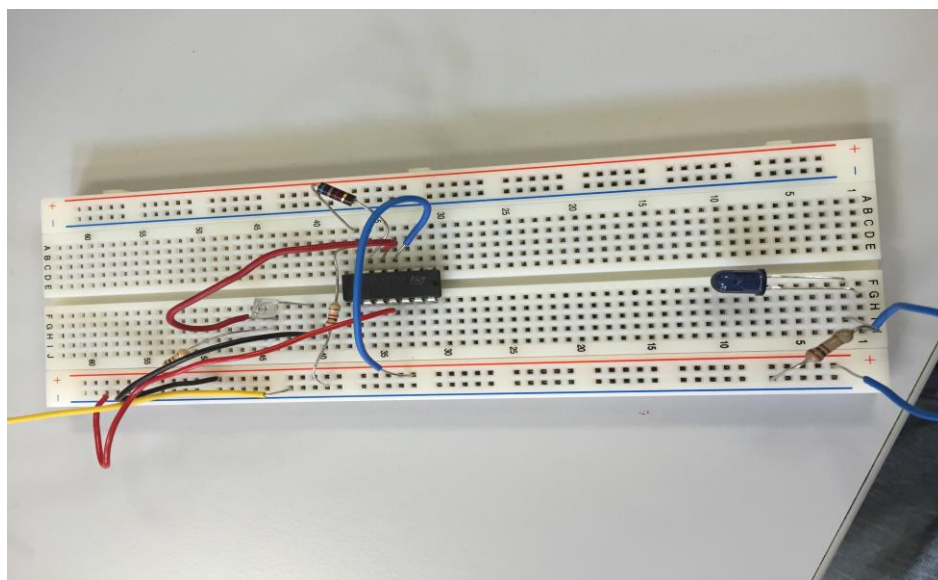


Ilustración 73. Prueba inicial del sistema de detección

En la Ilustración 73 se ve la primera prueba realizada con un solo led y detector.

En ella se comprobaba la distancia a la que queríamos que estuviera el led del detector, y con la alimentación que debíamos utilizar. Las características de los componentes que se emplearon son las siguientes.

La resistencia del led 100Ω .

La resistencia del fototransistor $1k\Omega$.

Pero con estos componentes nos llega una tensión de tan solo $400mV$ para una distancia de $8cm$. Esta tensión no nos vale por lo que se tiene que amplificar la señal para que llegue correctamente al micro. El amplificador que se usa es el que se puede ver en la Ilustración 74.

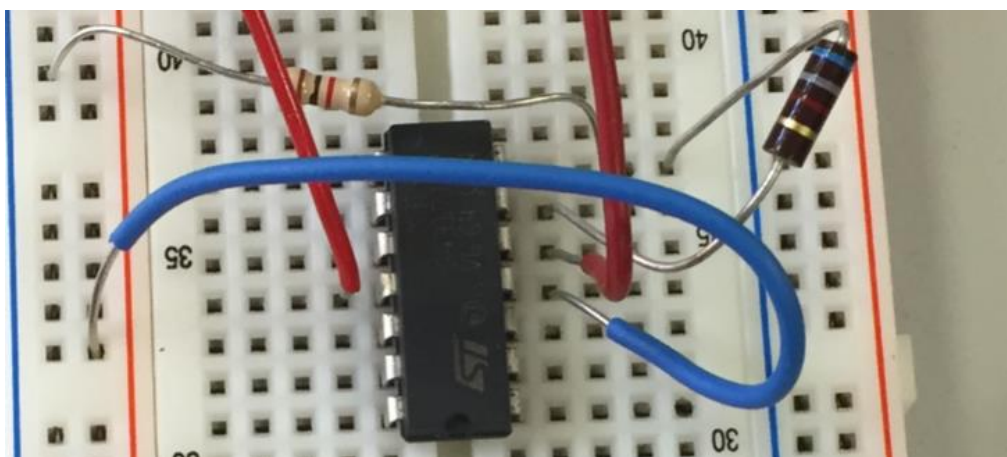


Ilustración 74. Amplificador No Inversor con un Lm234

Es un amplificador no inversor en el que la resistencia R1 de 1k y la R2 7k.
Quedando el montaje de varios detectores como se ve en la Ilustración 75.

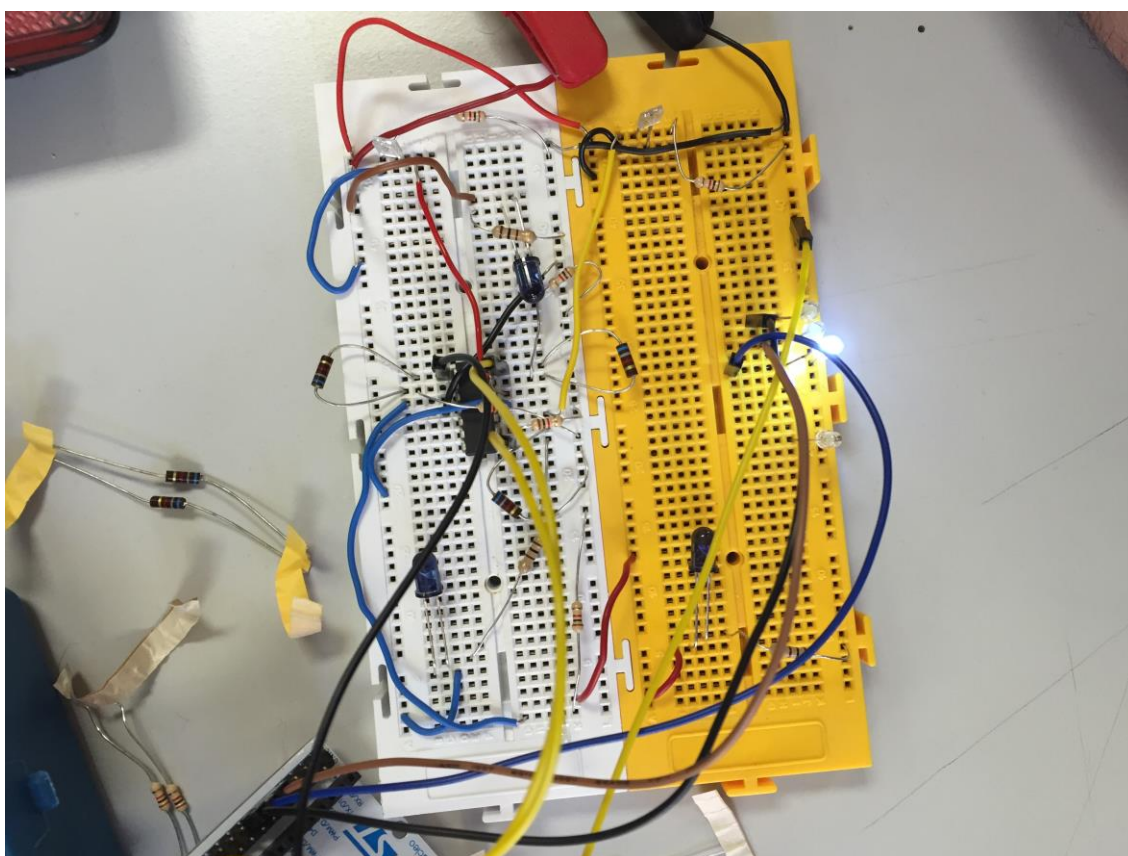


Ilustración 75. Prueba con varios leds y fototransistores

También para el control de los servomotores, es necesario el uso de una señal PWM generada por el micro cuyo el ciclo variable provoca el giro de estos servomotores. Pero para poder realizar el movimiento del servomotor la señal que le llegue tiene que ser de 5V y la señal que se genera mediante el micro tiene un valor de tensión menor.

La solución que se adoptó fue la de poner un comparador entre medias que nos aumentase el valor de la señal, estando el comparador con V_+ a 5V y con V_- a 0V, y estando alimentado por la patilla de la entrada negativa con 1V y en la positiva la señal del micro. Aunque de esta forma no conseguimos los 5V que se requieren teóricamente, pues tenemos a la salida 4V, el servomotor funciona correctamente, véase en la Ilustración 76.

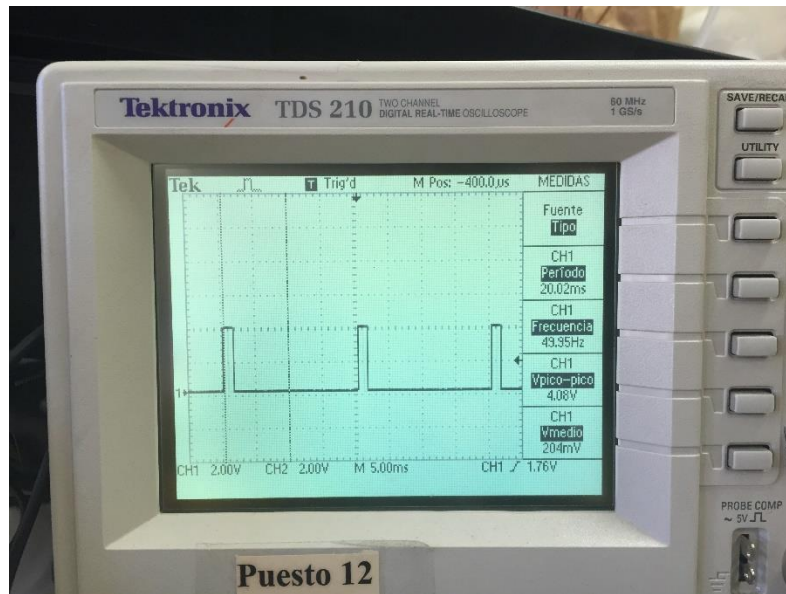


Ilustración 76. Señal después de amplificarla

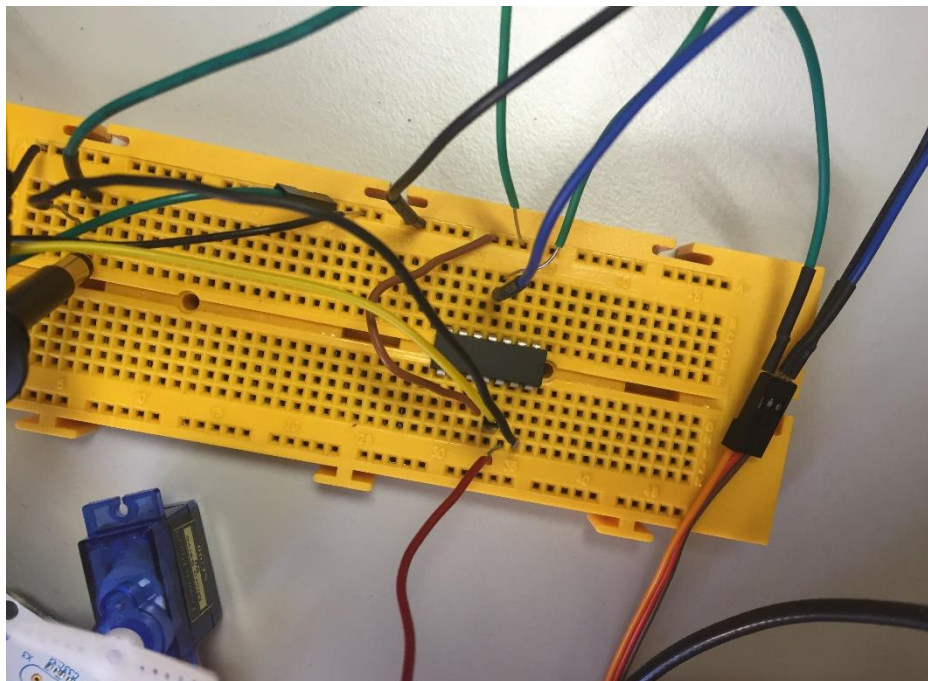


Ilustración 77. Circuito integrado para controlar los servos

El cable amarillo que vemos en la Ilustración 77, es el que lleva la señal PWM del microcontrolador y el cable azul es el de la salida del comparador que llega al servo.

Se hicieron pruebas con el pulsador para poder comprobar el rebote que nos generaba a la hora de la pulsación, ya que este tipo de rebotes pueden afectar en la interrupción del programa dando fallos en el futuro. Para ello se realizó una prueba de dicho pulsador y se comprobó la señal en el osciloscopio, obteniendo resultados satisfactorios, aunque era algo previsible según los datos del fabricante, ya que tiene un efecto rebote muy pequeño, aunque por seguridad durante la programación se adaptó por si hubiera algún tipo de rebote por si se cambiase el pulsador por otro. Las conexiones que se realizaron al pulsador se pueden ver en la Ilustración 78, esta imagen se corresponde al montaje.

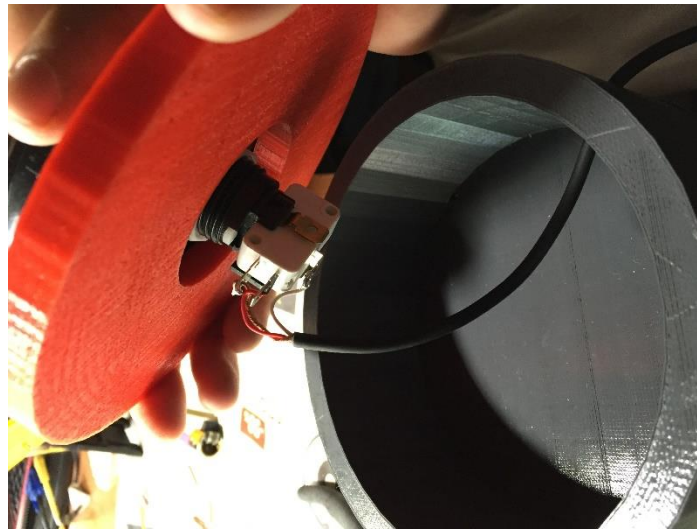


Ilustración 78. Colocación de los cables en el pulsador

También se realizaron pruebas en el sistema de radiofrecuencia, esta es la parte que conecta los dos bloques, ya que la parte transmisora se encuentra en el Bloque Rampa y la parte receptora en el Bloque Carriles. Se mandaba información de un sistema a otro en el que con la pulsación de unos botones se activaban una serie de led al otro lado, esta fue la primera prueba que se realizó. Luego se aplicó a los intereses del proyecto, mandando la información necesaria para mover los motores paso a paso.

Este tipo de pruebas se explican más detalladamente en la memoria de mi compañero Sergio Castillo Mohedano.

3.3 Construcción Mecánica del Sistema

En este apartado se explica de forma detallada la construcción del juego de carreras al completo, los dos bloques. Tanto el diseño de las piezas 3D como la construcción de la estructura de madera.

3.3.1 Material

3.3.1.1 Madera

Para la construcción del juego se utilizó madera de contrachapado de calabo de chapas varias de 7mm, ya que es un material que tiene cuerpo, es resistente y ligero. Este material se consigue con la unión de finas capas pegadas con las fibras transversalmente unas sobre otras, consiguiendo de esta forma esa resistencia a pesar de su ligereza.



Ilustración 79. Madera de calabo

3.3.1.2 Filamento PLA

El plástico que se emplea en la impresora 3D es filamento PLA de 1.75mm.

Es uno de los materiales que se suelen encontrar para trabajar con las impresoras 3D junto al ABS, el PLA tiene un rápido endurecimiento, una mínima tensión térmica y una mínima deformación.

Su temperatura de deformación óptima es de 220°C, con una temperatura de fusión entre los 180 y 220°C. Aunque según la velocidad de impresión hay que modificar la temperatura, como se explicará en el siguiente apartado 3.3.2 Impresora 3D.

Además, el PLA es un producto ecológico derivado del maíz, en el que no se usan derivados del petróleo con lo que es totalmente biodegradable.

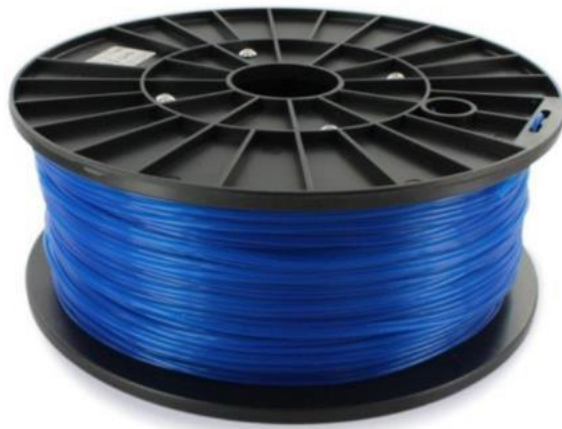


Ilustración 80. Rollo de filamento PLA

3.3.2 Impresora 3D

La impresora 3D que se ha utilizado para este TFG es la Impresora 3D BQ Witbox 2, que está en el laboratorio 1.2.C.12 del GDAF-UC3M del Departamento de Tecnología Electrónica.

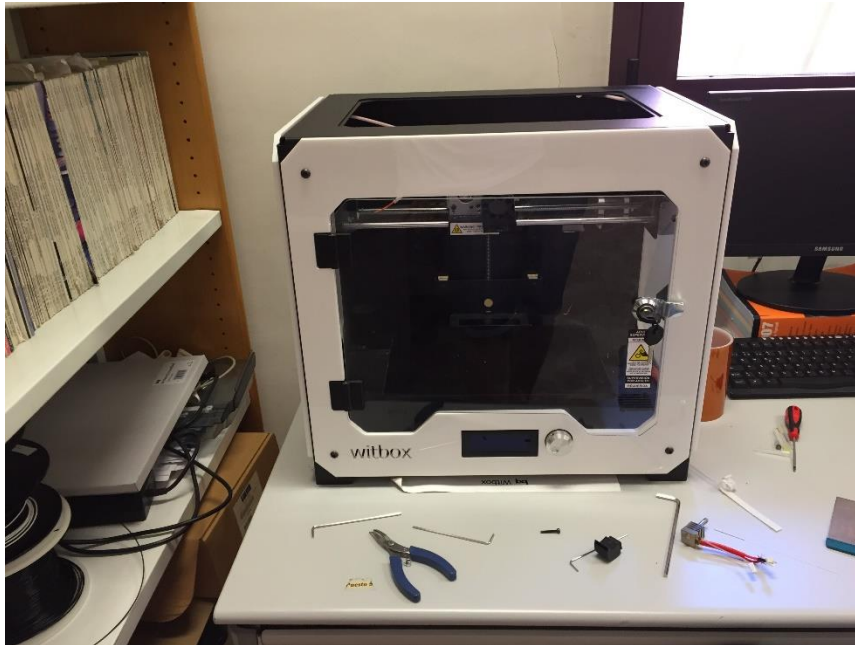


Ilustración 81. Impresora Witbox 2

A la hora de la impresión de las piezas aparecieron una serie de problemas sobre todo en los últimos días, en los meses de verano, ya que la impresora comenzó a quedarse atascada en la zona del extrusor, sin saber muy bien porque ocurría esto.

Se tuvo que cambiar el extrusor varias veces para limpiarlo y que así funcionase correctamente, aunque solo por unos días ya que sin una explicación lógica a entender, porque se tuvo especial cuidado entre la gente del laboratorio a la hora de su uso.

En una ocasión comenzó a imprimir la pieza correctamente y se quedó a mitad de la impresión ya que comenzó a descender la temperatura drásticamente y se tuvo que parar la impresión (Ilustración 82), para así intentar evitar problemas mayores, quedándose la pieza a medias. Esto fue solventado con la actualización del software de control de la impresora.



Ilustración 82. Pieza completa a la izquierda y otras dos que se quedaron a medias

También debido a los problemas con el extrusor las pocas piezas que se podían sacar, no tenían los acabados correctos como los que tenían las primeras piezas. El extrusor se atascaba debido a errores en la extracción del mismo, motivados por el salto de una instrucción por parte de la impresora, véase en la Ilustración 83, como se puede apreciar la pieza tiene una textura rugosa, mientras que si se hubiera tenido una impresión correcta la textura sería lisa.



Ilustración 83. Pieza con fallos en el extrusor

En un principio se creyó que podría estar mal calibrado, pero se realizó una nueva calibración de la placa, de la siguiente forma: con el menú de la impresora se selecciona Control→ Level Plate y luego hay que ajustar 3 puntos en los que tiene que pasar el folio entre el extrusor y la placa de manera muy ajustada, y finalmente se coloca en el centro de la placa para comprobar de nuevo la altura.

Debido a estos problemas que nos encontramos y el hecho de que la universidad cerraba por un tiempo y nos faltaban algunas últimas piezas, se usó una impresora 3D a la cual tenía acceso mi compañero en su trabajo en Indra, era una impresora Ultimaker 2, para la cual nos valían los diseños 3D de las piezas ya realizados, lo único que había que modificar era el archivo .gcode que es el que regula las características de la impresora a la hora de la impresión y que cada .gcode es propio de cada tipo de impresora 3D.

3.3.3 Bloque Rampa

Este bloque está hecho principalmente de madera salvo la parte del pulsador que está diseñado e impreso en 3D.

Primeramente se comenzó la construcción de la estructura mediante unos listones de DM de 3x2 cm de grosor, que forman el esqueleto del bloque (ver Ilustración 84).

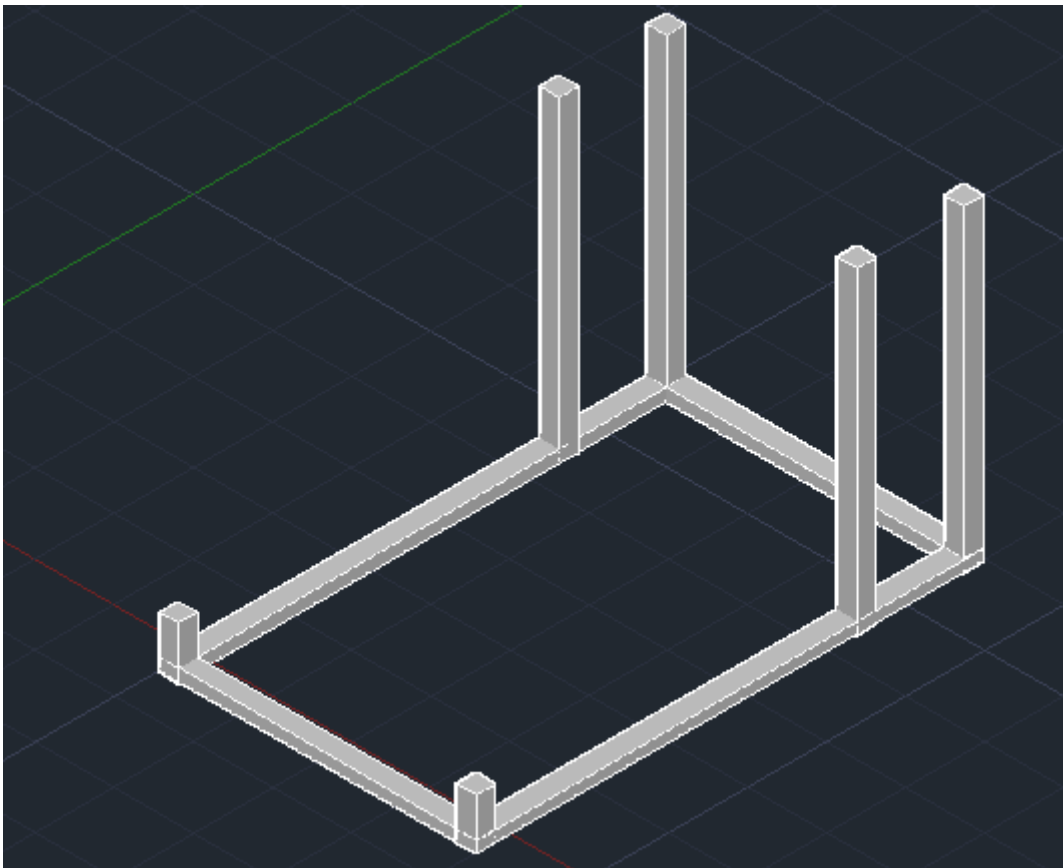


Ilustración 84. Esqueleto de listones de la estructura

Los listones estaban cortados de la siguiente forma:

- 4 listones de 48 cm.
- 2 listones de 50 cm.
- 2 listones de 74 cm.

Una vez formada la estructura y encolada se pasa a poner los laterales de contrachapado, estos se fijan a la estructura con tornillos para que sea robusta y no se suelte, como se ve en la Ilustración 85, donde está en color la parte nueva para apreciarla claramente.

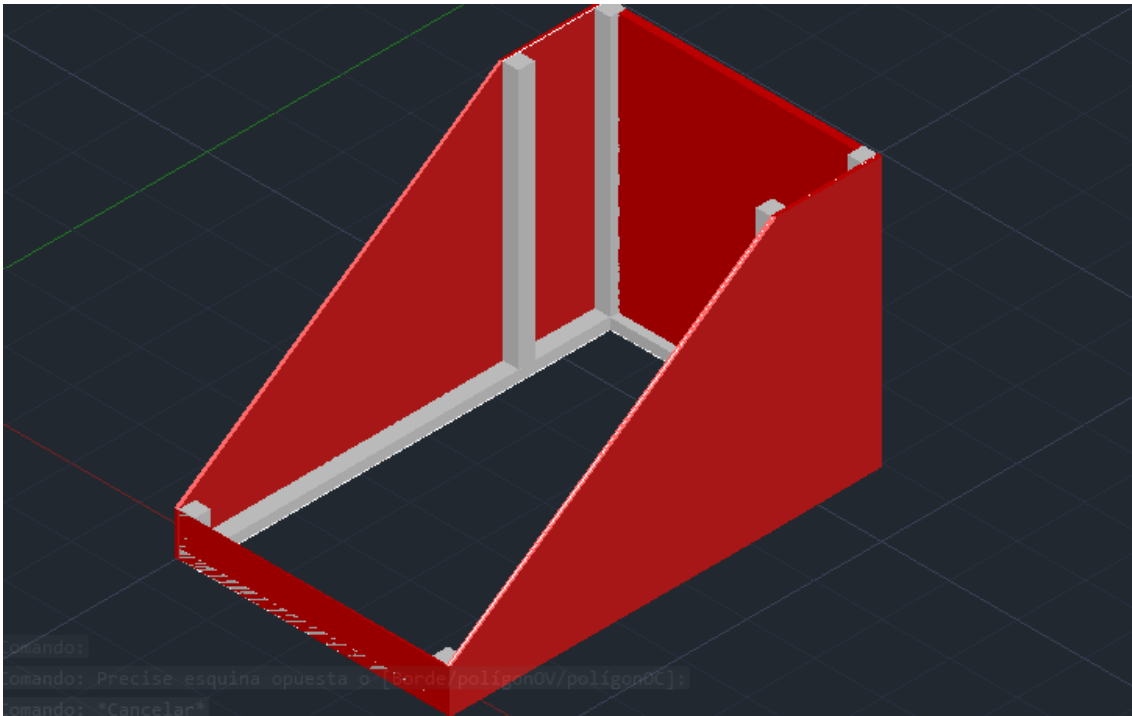


Ilustración 85. Estructura con los lados colocados

Después de colocar los laterales, se debe colocar otros listones inclinados en los laterales para colocar la posterior rampa, también se hacen las casillas donde caerá la pelota para dictaminar los puntos obtenidos en cada tirada, como se ve en la Ilustración 86.

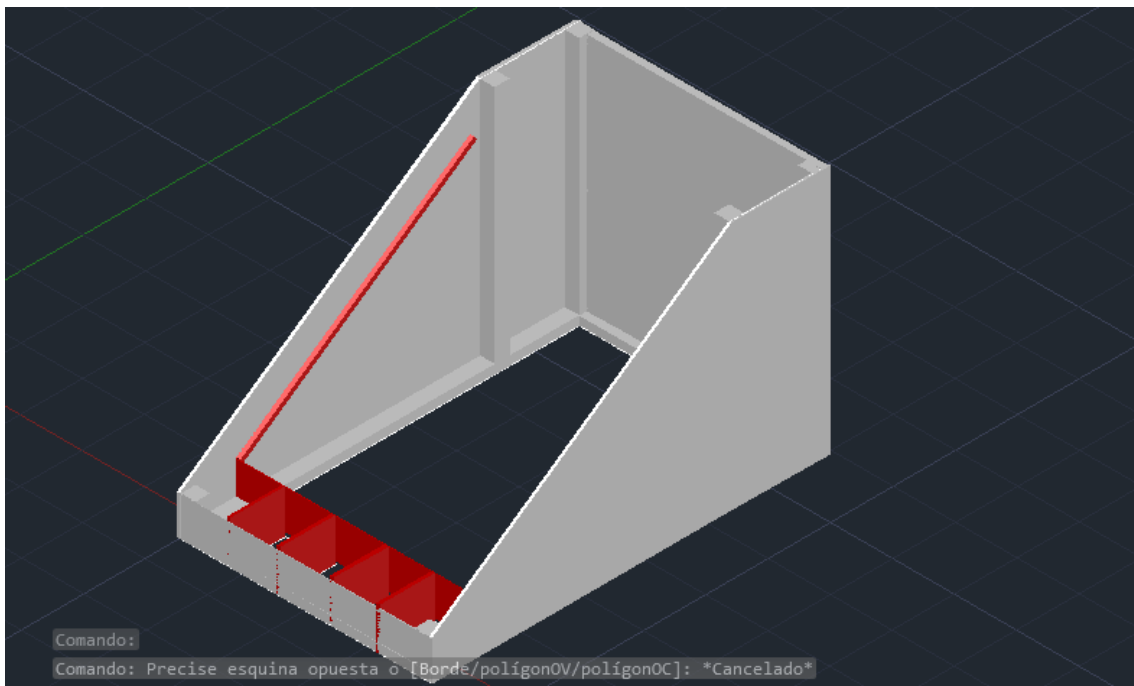


Ilustración 86. Colocados las casillas

Con la estructura del bloque ya construida pasamos a hacer la caja donde se colocará la puerta para comenzar la partida (Ilustración 87), esto irá en la parte superior de la estructura. Será una caja con el suelo inclinado para que la pelota caiga por su propio peso y el efecto de la gravedad, pero para que esto ocurra cuando se quiere, la puerta se abre accionándola con el pulsador.

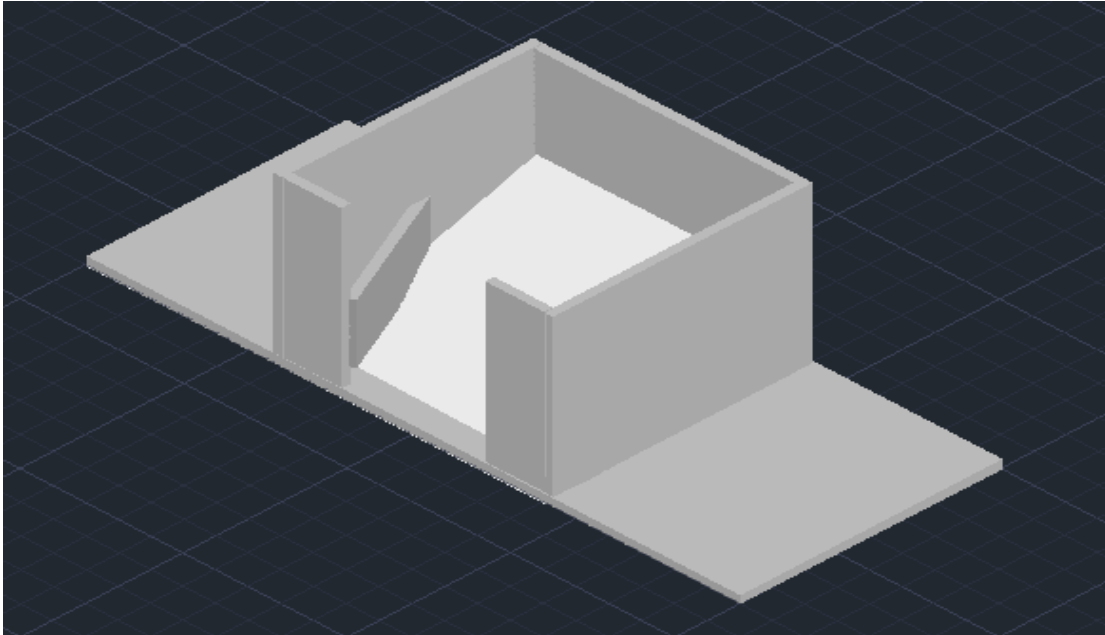


Ilustración 87. Caja donde se coloca la pelota

La rampa como ya dijimos en el apartado 2.2.2 Rampa tiene una máquina de Galton en la parte superior. Para hacer los pivotes de ese diseño se ha empleado espigas de madera, colocadas en una serie de agujeros de M6 realizados en la madera, formando la máquina de Galton. La parte inferior tiene tras palas que se activan con los servos, estas palas son de contrachapado, tienen las siguientes dimensiones 15x6 cm (Ilustración 88).

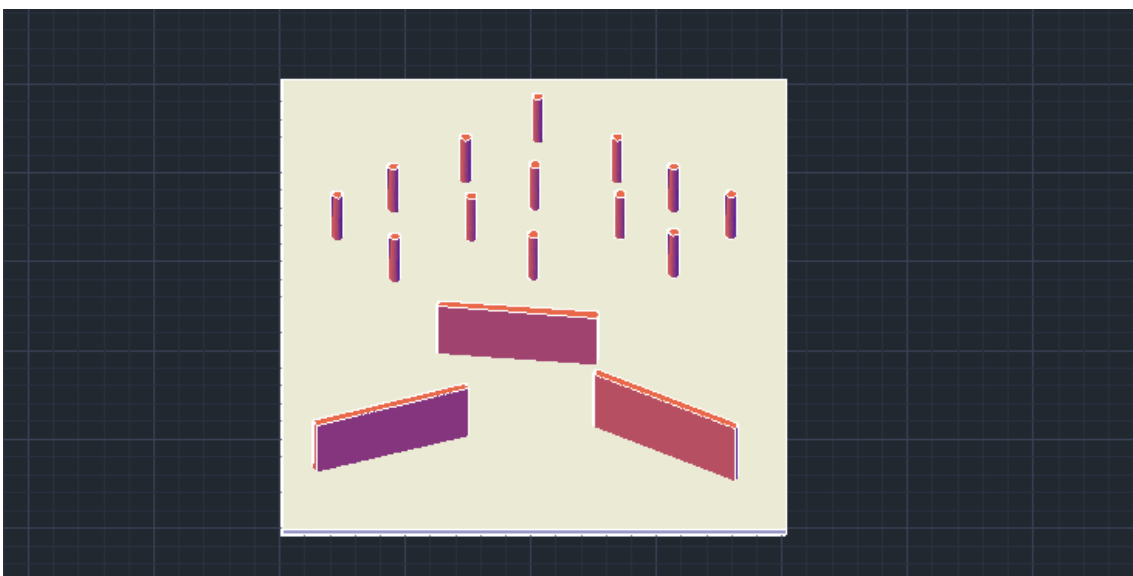


Ilustración 88. Colocación de los obstáculos de la rampa

Una vez construido todo esto se puede montar y colocar todo quedando de la forma que se ve en la Ilustración 89.

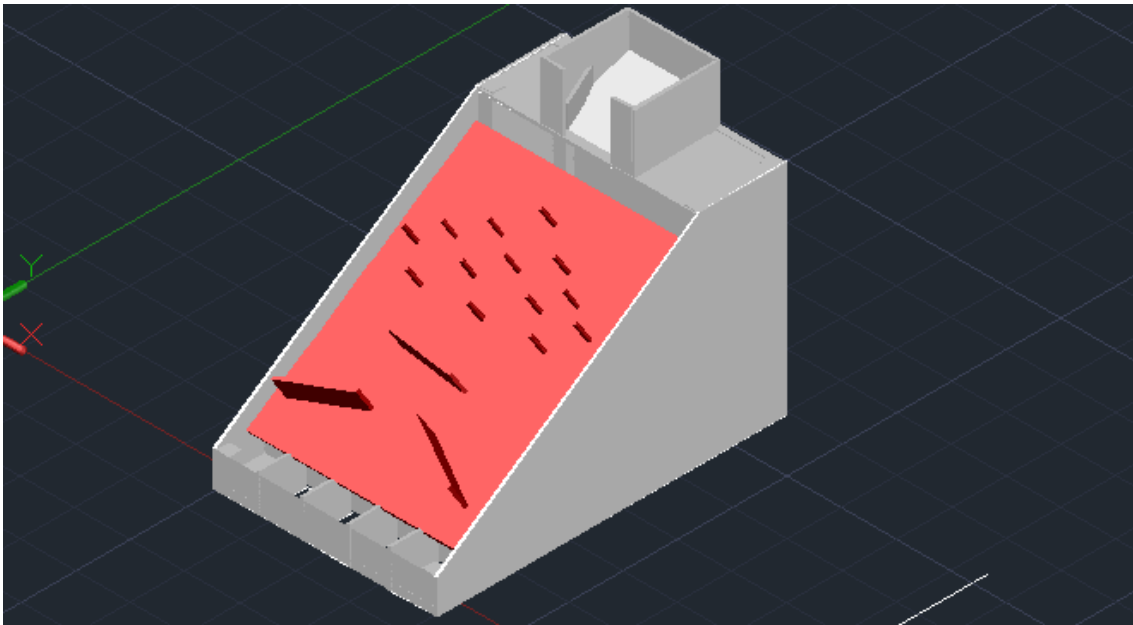


Ilustración 89. Bloque Rampa completo

Pero con esto no acaba la construcción de este bloque, ya que en esta parte también se encuentra el botón, y hay que hacerle una caja o estructura que lo proteja, porque tiene que estar aparte del bloque de la rampa, para que se pueda usar de una manera fácilmente alcanzable según el Diseño Para Todos. Para ello esta parte la hemos diseñado en 3D y se imprimirá con la impresora, ya que de esta forma será más estético y resistente.

Por una parte tenemos la caja donde estará el botón protegido (Ilustración 90).

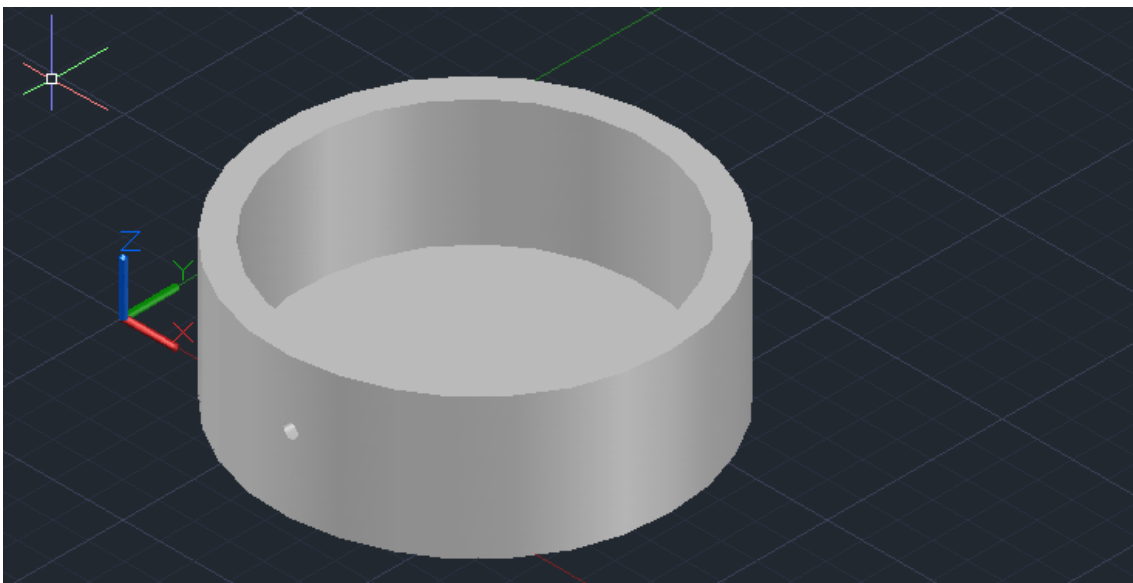


Ilustración 90. Caja del Pulsador

Y por otro lado la tapa que cerrará el bloque, y protegerá las conexiones internas (Ilustración 91).

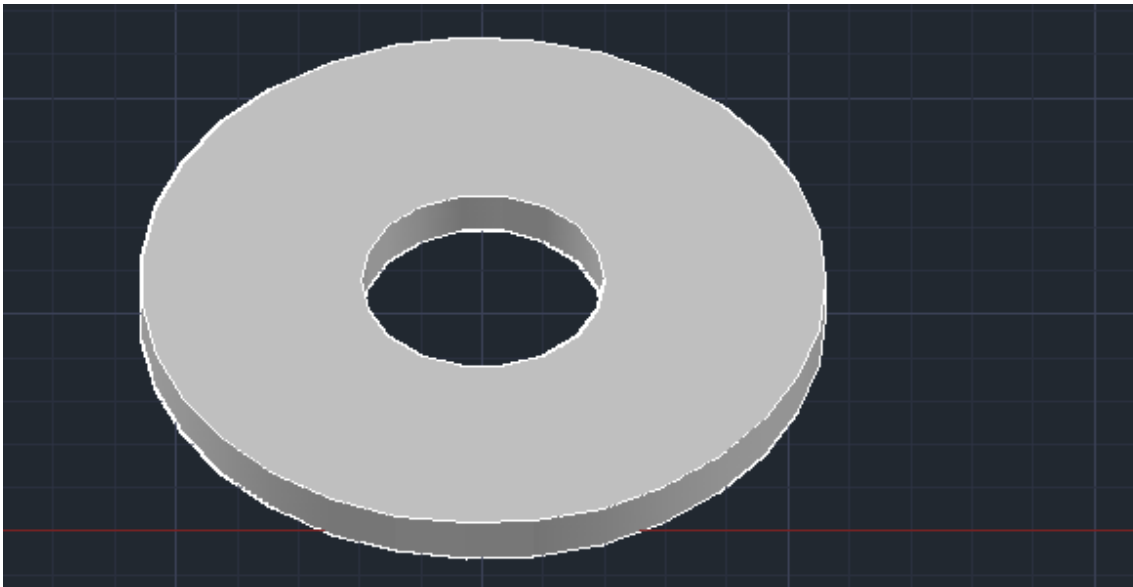


Ilustración 91. Tapa de la caja del pulsador

Con esto ya tendríamos toda la construcción de este bloque de la rampa, pero hay que colocar toda la parte electrónica.

Para poder colocarla se deben hacer un par de agujeros en el panel lateral derecho del bloque.

Por una parte se coloca el botón de encendido de la alimentación y los cables de la alimentación de la fuente que va conectada a la red eléctrica doméstica. El agujero de los cables está ubicado en la esquina inferior derecha y es de M6. Para el botón de la alimentación se requiere un hueco de 5x2 cm que colocaremos en la parte delantera del lateral para ser más accesible, junto al agujero de la entrada Jack de M6. Lo podemos ver en la Ilustración 92.



Ilustración 92. Interruptor de alimentación y entrada Jack

La otra parte del sistema que necesita hacer agujeros es el sistema de control, ya que se necesitan 3 agujeros para los botones que controlan el sistema, estos serán de M15, colocándolos en un fila vertical que estará a la izquierda de la pantalla, como esta en la Ilustración 93.



Ilustración93. Colocación de los botones de control

También se necesita un hueco para la pantalla LCD, que por un lado está anclada a la PCB, y por el otro se ve en el lateral, por eso tiene que ir colocado en un lugar en el que haya espacio para anclar y fijar la PCB por dentro. Las dimensiones del hueco de la pantalla son 7.1x2.7 cm, como se puede ver en la Ilustración 94.



Ilustración 94. Hueco pantalla LCD

Con esto ya están hechos todos los agujeros que se precisan en la madera, pero ahora queda colocar los componentes electrónicos en los sus lugares correspondientes.

Para el sistema de detección se debe colocar los led y los fototransistores en las casillas donde cae la pelota indicando el valor obtenido. Para que se tenga una buena detección, se han colocado en la esquina superior para que lo detecte nada más pasar y no corte el haz de luz varias veces pudiendo provocar algún fallo, aunque en la programación se ha preparado para que no se produzcan rebotes. Dicha colocación se puede ver en la Ilustración 95 donde están colocados y como se han fijado, para lo cual se ha empleado pegamento termoplástico.



Ilustración 95. Colocación de uno de los leds

Para fijar las palas de los servomotores y la puerta de la caja que se acciona con servos también se han empleado unos clavos de 1.1mm de diámetro y 15mm de longitud para fijarlos al plástico que se encaja en la polea dentada de los servos.



Ilustración 96. Fijación de palas a engranaje del servomotor

Y para fijar los servomotores a la estructura lo que se ha empleado es pegamento termo plástico.

3.3.4 Bloque Carriles

Este es el segundo bloque y para este se ha tenido que diseñar más piezas para la impresora 3D, debido a las exigencias que se nos plantea con el bloque.

Al igual que el bloque de la rampa se comienza con la construcción de la estructura de listones. Se usarán los mismos listones aunque con diferentes longitudes, y están colocados de la forma que indica la Ilustración 97.

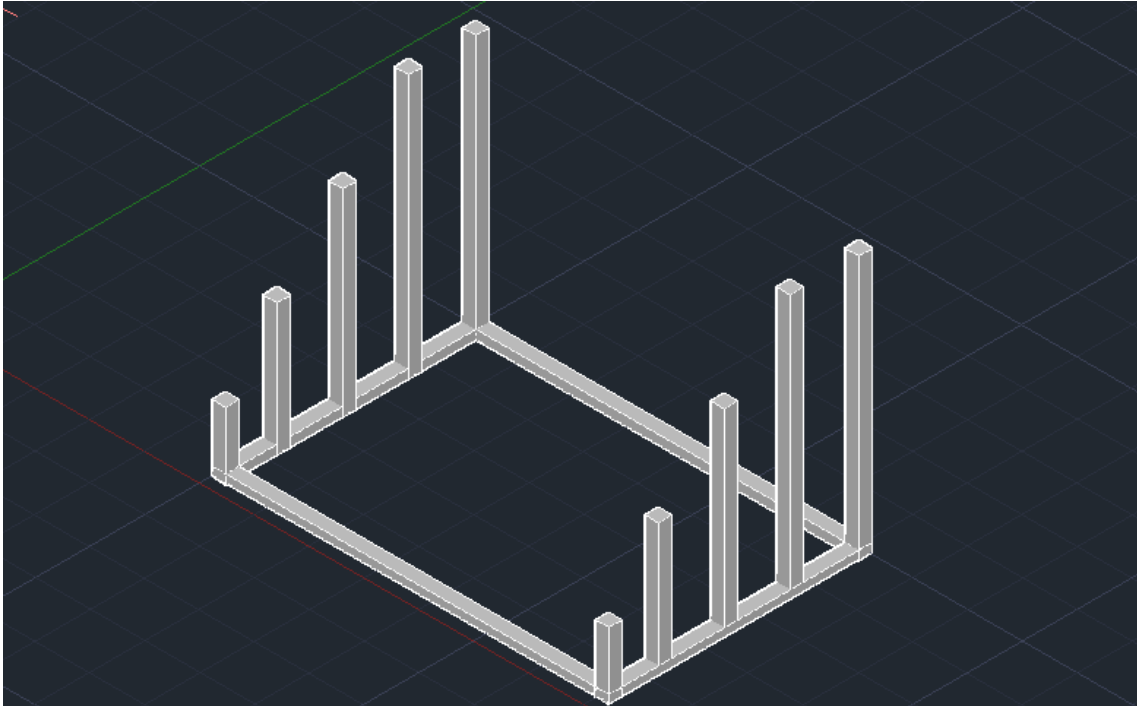


Ilustración 97. Estructura de listones de los Carriles

Los listones estaban cortados de la siguiente forma:

- 4 listones de 58 cm.
- 2 listones de 43 cm.
- 2 listones de 28 cm.
- 2 listones de 13 cm.
- 2 listones de 90 cm.
- 2 listones de 47 cm.

Al igual que la anterior colocaremos los paneles laterales de contrachapado para ir formando el bloque y se fijarán de la misma forma, ver Ilustración 98.

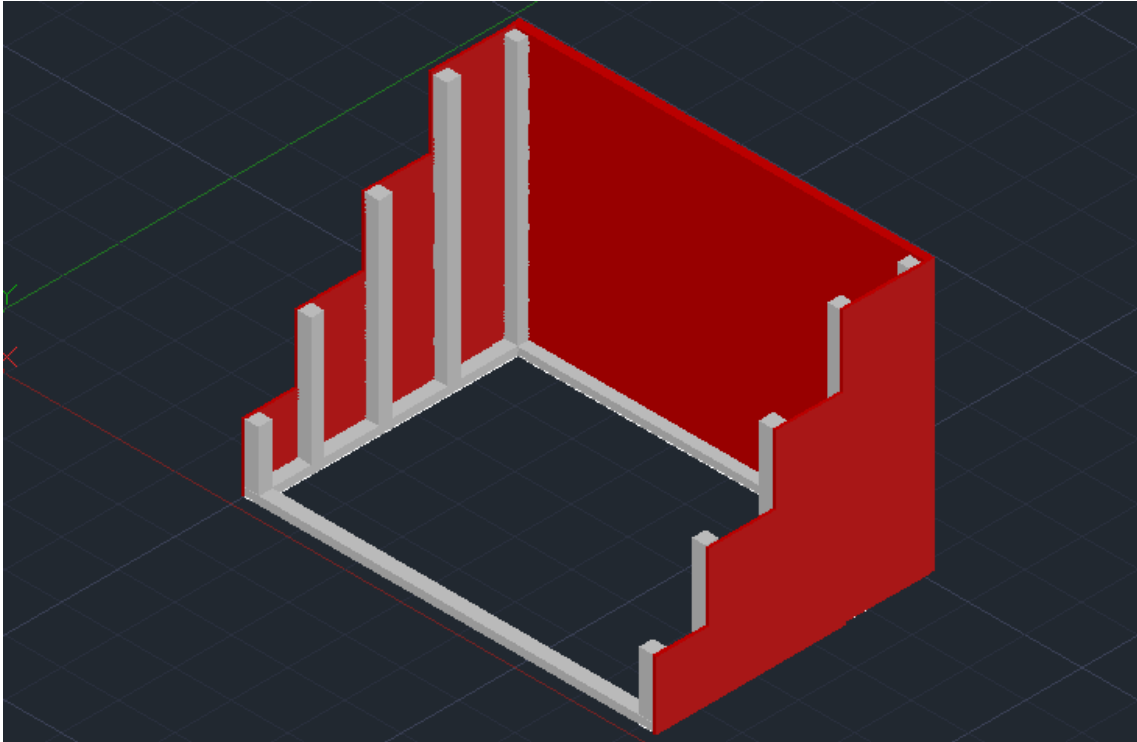


Ilustración 98. Estructura con los laterales

Una vez ubicados los escalones, colocamos los paneles frontales de los escalones (Ilustración 99), donde se fijarán los motores paso a paso una vez ya tengamos puestas todas las maderas.

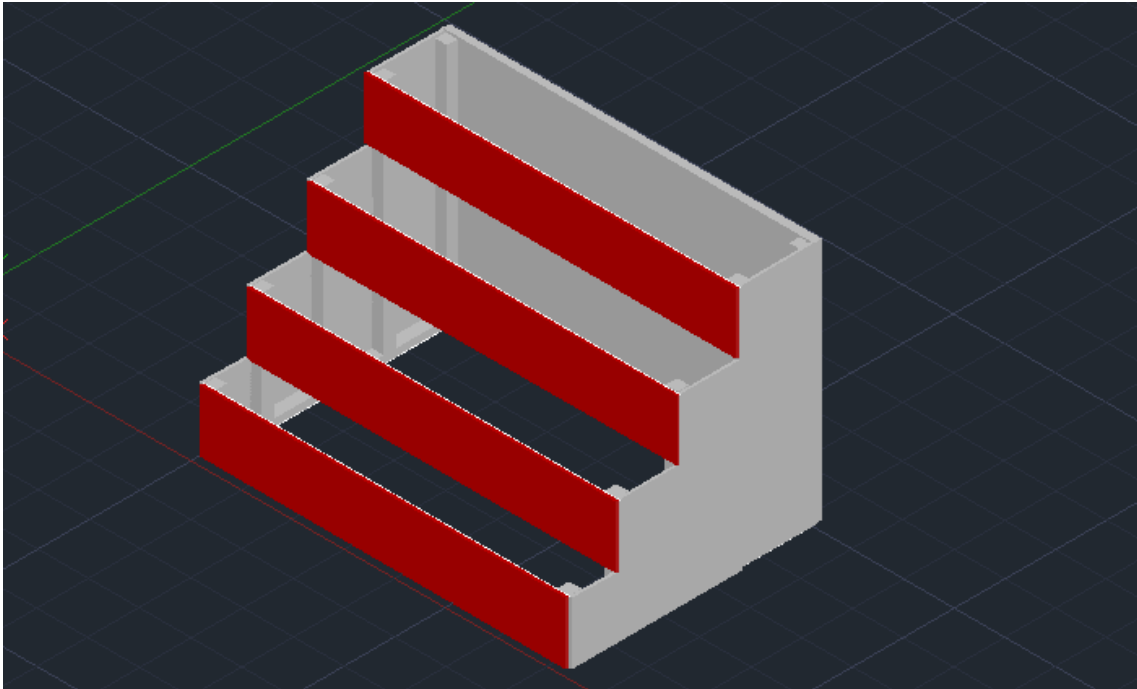


Ilustración 99. Colocación de los paneles frontales

Después cortamos las tapas de los carriles con la forma correspondiente, 90x15cm y un hueco interior centrado de 6x80cm, la pieza se puede ver en la Ilustración 100.



Ilustración 100. Tapa de cada carril

Luego se colocarán en sus posiciones quedando la estructura como en la Ilustración 101.

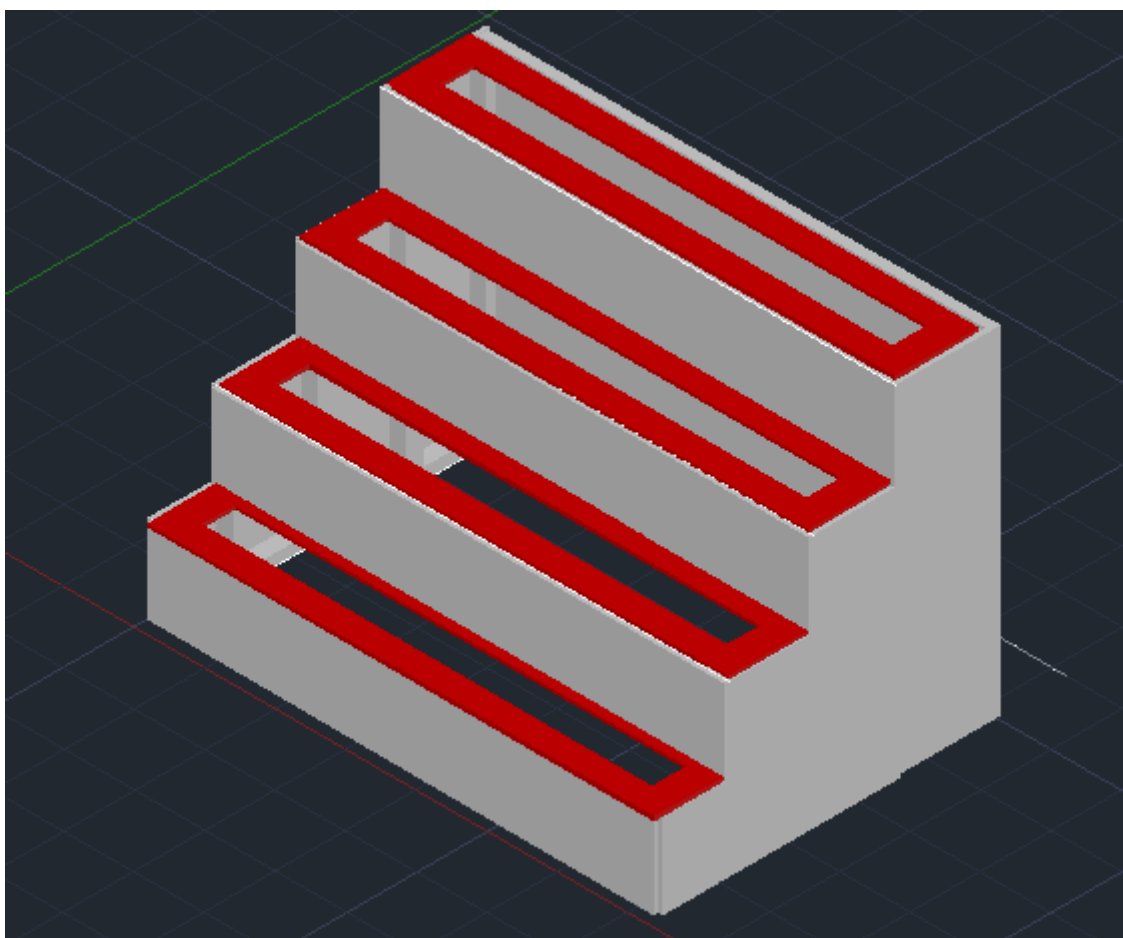


Ilustración 101. Bloque Carriles completo

Los motores pasos a paso van colocados en los paneles frontales con 4 tornillos M5 de 50 mm por cada motor, a 10 cm del lado izquierdo, como se ve en la Ilustración 102.



Ilustración 102. Colocación de un motor paso a paso

En estos mismos paneles pero al lado contrario y también a 10 cm se colocan las poleas esclavas, estas van ancladas mediante 1 tornillo M5 de 15 mm con una tuerca M5 a M5 y en el otro lado una varilla M5 donde irá la polea, esta está ajustada con los tuercas auto-frenantes (Ilustración 103), y para que gire bien la polea se colocará un rodamiento en el interior (Ilustración 104), quedando finalmente como en la Ilustración 105.



Ilustración 103. Varilla de la polea esclava

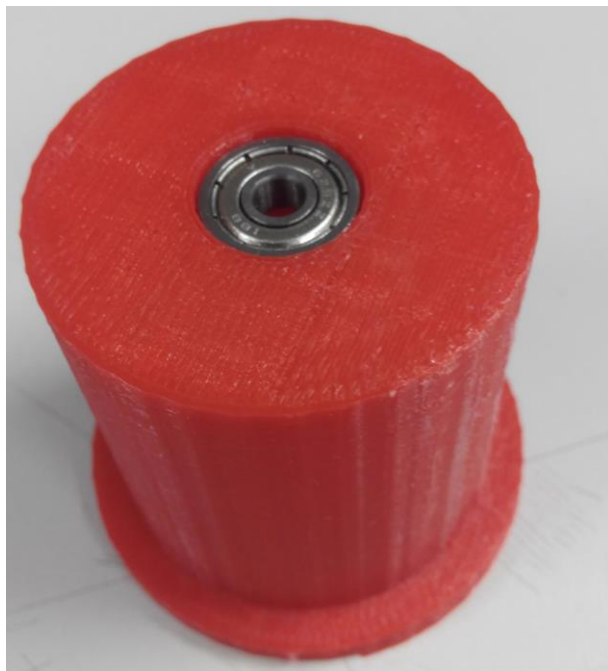


Ilustración 104. Polea esclava con el rodamiento interno.



Ilustración 105. Polea esclava colocada en la estructura

Para el motor paso a paso se tuvo que hacer una polea en 3D, para que ajustase con la marca que tiene hecha el motor, en la Ilustración 106 se puede ver la polea y su tapa.

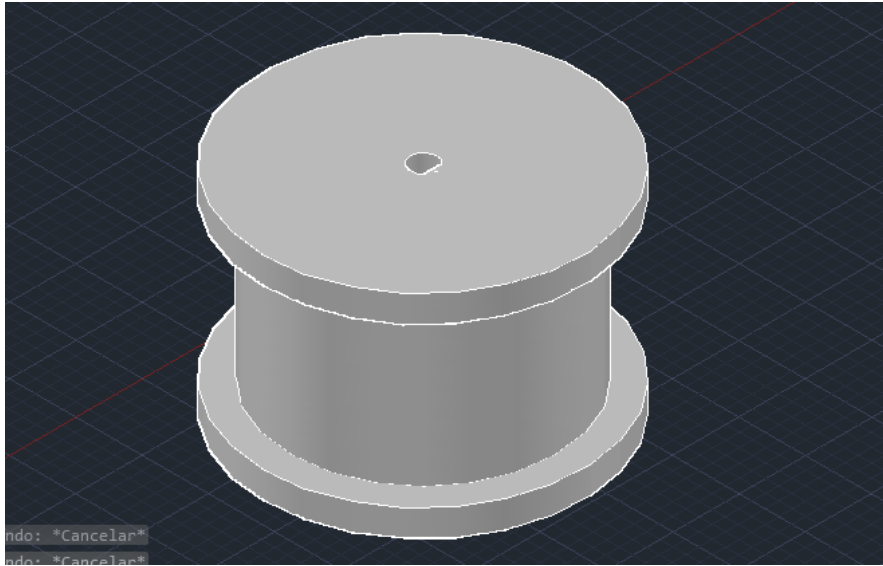


Ilustración 106. Polea completa

Para que el carril funcione correctamente es necesaria una cinta que mueva la polea, para esto se ha utilizado una cinta de tiracol (correa empleada en la tapicería) de ancho 5cm y longitud 150cm, ya que el carril tiene 70cm de longitud. Para adaptarla al proyecto las tuvimos que unir y para ello la grapamos entre sí, quedándonos una superficie de unos 6 cm con doble cinta que quedará en la parte de abajo para que no haya problemas durante el movimiento del carril. Estos 6 cm fueron para que el carril estuviera lo más tenso posible, quedando la cinta como en la Ilustración 107.

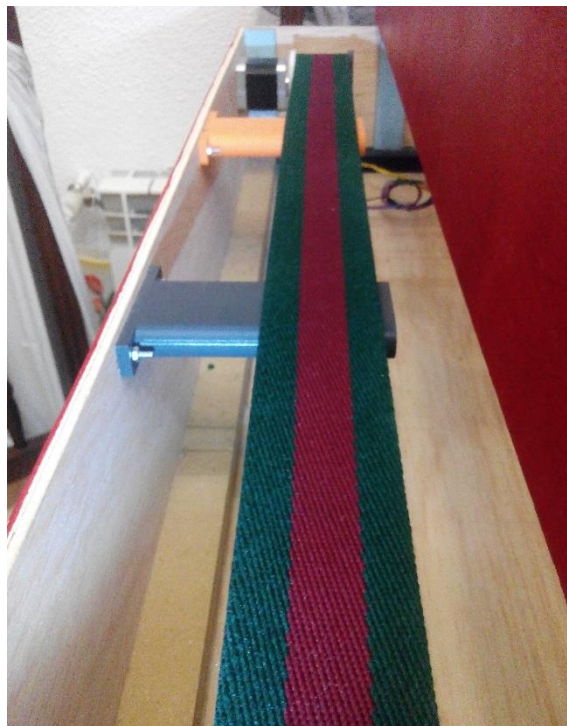


Ilustración 107. Correa de los carriles

Debido al peso que puedan tener los muñecos que van sobre los carriles, se ha diseñado unos apoyos 3D que sujeten la cinta (Ilustración 108), habrá dos por cada panel frontal colocados de la forma que se puede ver en la Ilustración 109.

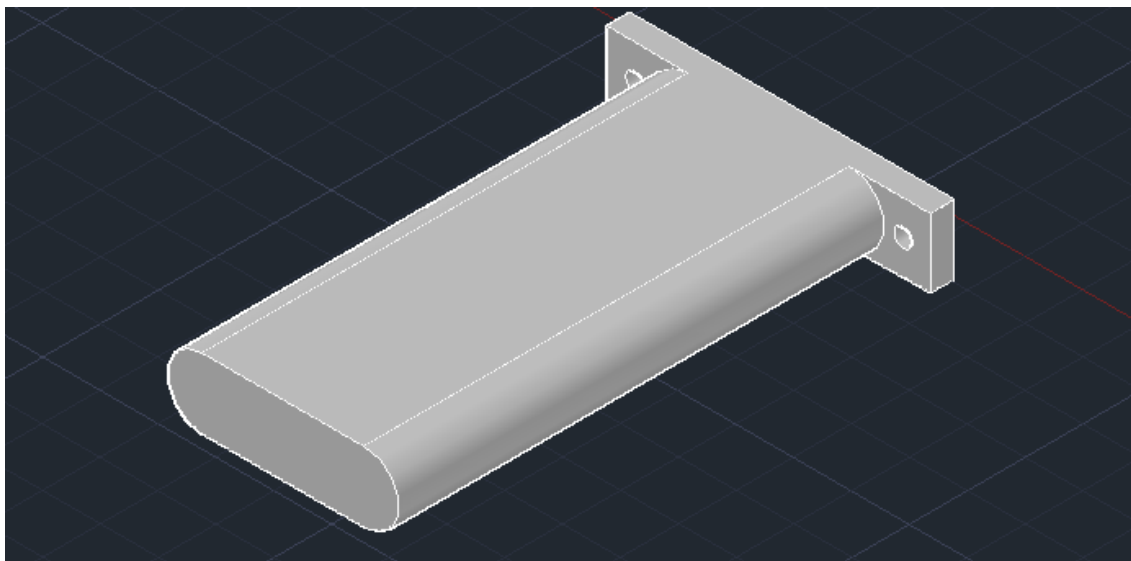


Ilustración 108. Diseño 3D de apoyo para correa de los carriles

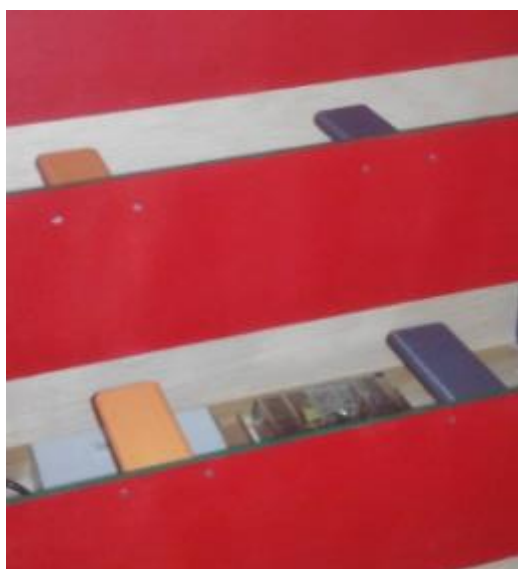


Ilustración 109. Colocación de los apoyos en la estructura

Estos están sujetos a la madera con 2 tornillos M3 de 15 mm y su tuerca correspondiente.

Con todo esto estaría completada toda la parte mecánica del bloque aunque falta colocar los microinterruptores. Son finales de carrera, colocados en las planchas que tapan los carriles, en el hueco interior, una en cada extremo por la parte de abajo, para detectar tanto la meta final como el principio cuando se colocan.

Para el sistema de audio hay que hacer unos huecos en los paneles laterales de la estructura para poder colocar los altavoces. Tendrá un hueco en cada lateral de la forma mostrada en la Ilustración 110, con una dimensión de 5x3 cm.



Ilustración 110. Hueco para los altavoces

Para la alimentación de bloque también se le hace unos agujeros, uno para colocar el botón de encendido y apagado y otro para pasar los cables que van a la corriente, como se aprecia en la Ilustración 111.

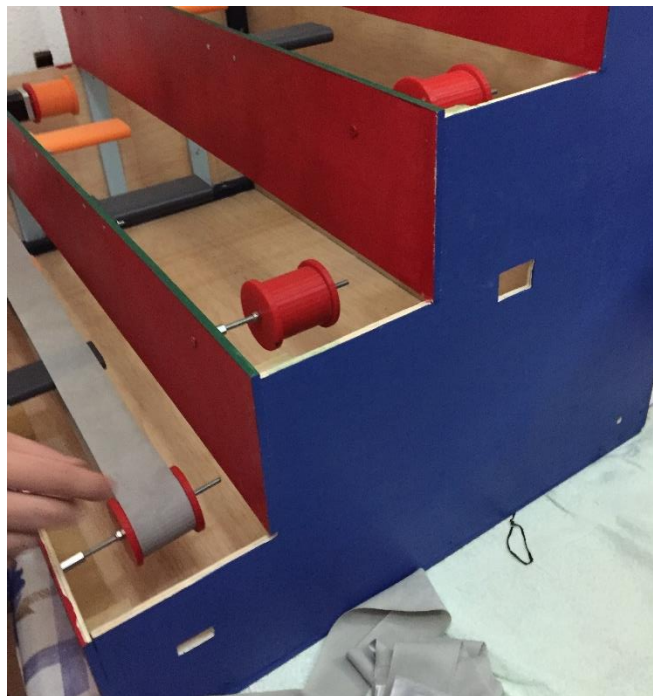


Ilustración 111. Lateral de la estructura para ver los huecos necesarios

El botón tiene una dimensiones de 5x2 cm y el agujero de los cables en de M6.

CAPITULO 4. PRUEBAS Y RESULTADOS EXPERIMENTALES

4.1 Pruebas iniciales

Aunque la mayor parte del programa se desarrolló y se probó previamente en las placas de circuito impreso antes de ser integradas en el chasis. Una vez que estas se integraron hubo que cambiar ciertos aspectos del programa para facilitar el uso del juego lo máximo posible. Por otro lado, también se tuvo que hacer modificaciones sobre la placa debido a errores de diseño. A continuación, se describen algunos de los problemas principales de los que se tuvieron constancia mientras se realizaban pruebas.

- Bloqueo de los motores 2 y 3

La primera vez que se procedió a mover los motores a través de la caída de la pelota sobre las casillas de la rampa. Se tenía que solo se movían los motores 1 y 4. El problema estaba en la forma en que se actuaba sobre los motores a través de los flags de codificador 8:3 de los microinterruptores. El programa nunca entraba en la instrucción que hacía el *toggle* sobre el correspondiente pin. Se solucionó cambiando la forma en que los flags eran leídos, añadiendo una variable local llamada *stop*.

- Modificación en la interfaz de accionamiento de trampilla de la rampa. (11)

El botón mediante el que se produce el accionamiento de la apertura de la rampa es un relé que tiene sus tres terminales conectados a un conector Jack macho TRS de 3.5mm. Sin embargo, al no haber profundizado lo suficiente en el tipo de botón que se utilizaba en el Colegio San Rafael, no se tuvo en cuenta que allí se utiliza un Jack TS. Es decir, mientras allí se usa un conector de dos terminales, el juego se diseñó para uno de tres.

Según las ilustraciones 107 y 108, que muestran el esquema eléctrico de la interfaz de nuestro botón con el microcontrolador, en estado de reposo el camino que está cerrado es el que se forma entre *sleeve* y *ring* (GND y entrada a microcontrolador). Cuando se produce la pulsación el camino que se cierra es el que forman *tip* y *ring* (3.3V y GND).

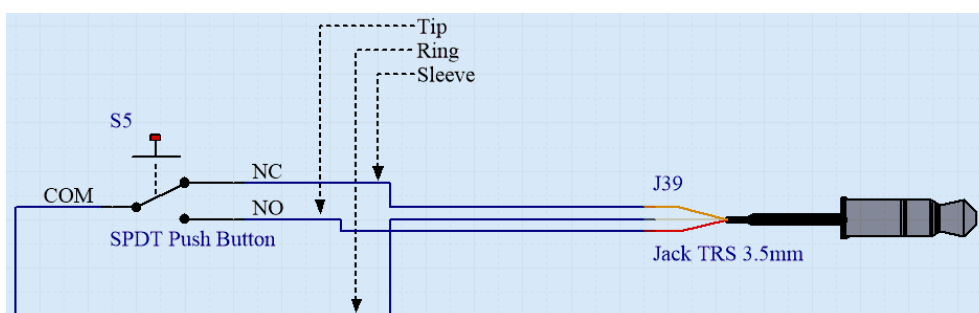


Ilustración 112. Interfaz de conexión del botón con el microcontrolador .

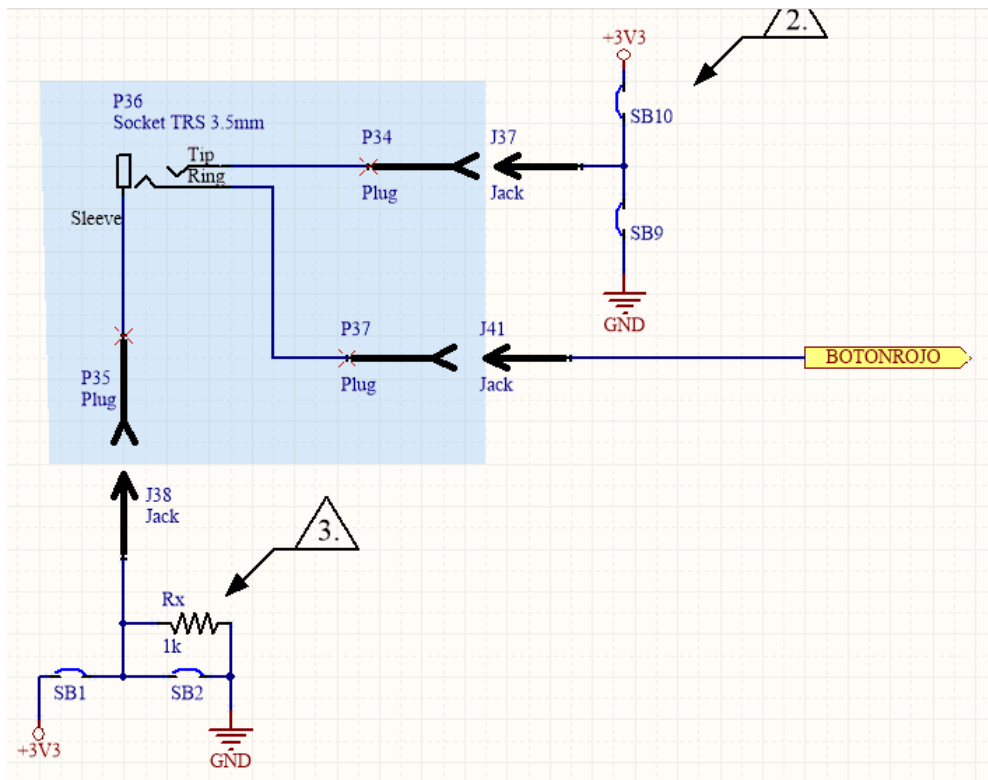


Ilustración 113. Interfaz de conexión del botón con el microcontrolador.

Si se conectara un Jack *TS* a la interfaz tal y como se ha descrito, se produciría un corto entre el *sleeve* y el *ring* debido a que el terminal *sleeve* en los conectores *TS* son más largos (ilustración 109).

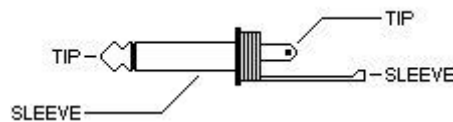


Ilustración 114. Conector TS de 3.5mm.

En estado de reposo no pasaría nada, pero si se pulsara y se cerrara el camino entre *tip* y *ring*, se produciría un cortocircuito entre *tip* (3.3V), *ring* y *sleeve* (GND). Para evitarlo, se tuvo que desmontar la placa del chasis, desoldar el punto de soldado SB2 y colocar una resistencia de 1kohm en paralelo (ilustración 110).

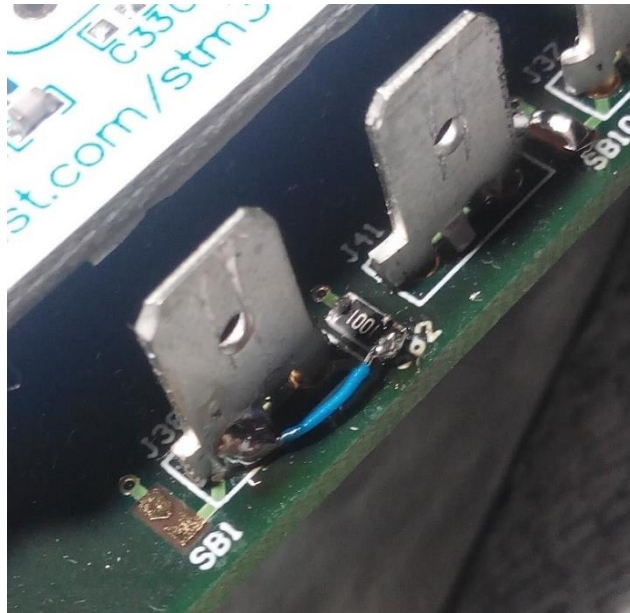


Ilustración 115. Modificación en el diseño de la PCB. Resistencias de montaje superficial colocadas en paralelo al punto de soldado SB2.

- Peso excesivo de las poleas esclavas

Debido al exceso de peso que soportaba las poleas esclavas se vencían hacia abajo, pero no solo era el peso que soportaría, sino la tensión que le provocaba las tiras de tiracol, que hacen de carriles.

Debido a esta tensión las varillas de las poleas se vencían hacia el interior, ya que estas solo están sujetas por un tornillo que conforma varilla (que hace de eje).

Por esta razón se debió de colocar unas alcayatas en el lateral del bloque rampa por el interior para que con unas bridas tiraran para el lado contrario al que lo hacía la tira de tiracol, contrarrestando las tensiones y que de esta forma se quedar recta. Puede que esto exija más par a los motores, pero los motores que se han utilizado tienen suficiente par, y además la tensión que se produce no es excesiva, simplemente, es para que se mantenga recta los carriles y no se hundan sin ponerles tan si quiera peso alguno.



Ilustración 116. Uso de las bridas como tensores

4.2 Pruebas experimentales de Campo

Una vez realizadas todas las pruebas en el laboratorio y se corrigieron los problemas que se comentaron anteriormente, ya solo quedaba la prueba final.

Esta prueba final fue llevarlo al Colegio de Educación Especial del Hospital San Rafael, donde fue probado por sus alumnos y monitores, sin ninguno de los problemas anteriores.

Durante la prueba en el colegio, nos encontramos un pequeño fallo, relativo al funcionamiento del juego, y era que los sensores infrarrojos no detectaban el paso de la pelota de ping pong. Pero sí detectaban la mano cuando se iba a coger la pelota de cada casilla, este fue un fallo que no nos había pasado durante ninguna de las pruebas iniciales durante el laboratorio.

La única cosa que se hizo para solucionarlo fue ajustar la direccionalidad de los led, por si se hubieran movido durante el transporte al colegio. Y tras ese ajuste en todas las partidas siguientes funcionaron correctamente.

A continuación, se muestran unos enlaces de los videos en los que se puede ver una prueba del laboratorio donde se ve la selección del juego de forma más detallada, y otro video donde se ve el transcurso normal de la partida allí en el colegio.

Prueba Juego Carreras Adaptado (1): <https://www.youtube.com/watch?v=pyFjskidAT8>

CAPITULO 5. CONCLUSIONES Y POSIBLES LINEAS FUTURAS

5.1 Conclusiones

Al terminar este proyecto, se puede concluir que se llegaron a cumplir todos los objetivos que nos planteamos al comenzarlo.

La satisfacción completa la conseguimos cuando fuimos al colegio, y se pudo comprobar que todo el sistema funcionaba correctamente, y ver la cara de felicidad e impresión de los niños que se divertían con el juego, pero no solo tenían una cara de felicidad los niños sino que también los monitores estaban sorprendidos con el funcionamiento del juego. Y esto fue una gran alegría, ya que a pesar de todo el trabajo y sufrimiento que se pasó durante algunos momentos del proyecto, todo tenía su recompensa.

Pero no solo obtuvimos la recompensa y satisfacción de que funcionase, sino que también conseguimos conocimientos nuevos durante el desarrollo del proyecto, ya que se emplearon herramientas y productos, que en mi caso eran desconocidos hasta el momento, ya que en mi caso yo no había programado un microcontrolador y ahora ya se los comandos necesarios para programar uno, también he adquirido conocimientos necesarios para el diseño de PCBs aunque no hayan sido diseñadas por mi durante este proyecto, pero si he seguido el proceso de su diseño por parte de mi compañero, otra herramienta en la que he mejorado ha sido en la utilización de AutoCad, debido al diseño de las piezas en 3D y los planos del proyecto.

Y el hecho de ser un proyecto orientado a usuario final, con una serie de requisitos y especificaciones funcionales, daba la sensación de estar trabajando en una empresa. En la que el producto iba a salir al mercado y tenía que funcionar correctamente, en unos plazos de limitados.

5.2 Líneas Futuras

Aunque el presente proyecto funciona correctamente, se podrían añadir una serie de mejoras, ya que siempre se pueden añadir cosas nuevas para mejorar el sistema.

En la presente memoria se dirán las mejoras principalmente respecto a la parte mecánica.

La primera mejora que nos encontramos, es sobre el diseño de los carriles, porque se ha diseñado con poleas lisas, las cuales mueven la cinta gracias a que ésta está tensa y por lo cual no resbala por la polea. Pero este sistema podría fallar si se dieran de sí las cintas aunque en principio no debería de ocurrir. Por lo que como gran mejora aunque aumentaría considerablemente el coste, sería el utilizar unas poleas dentadas, con lo que se usaría unos carriles dentados, y esto haría que se controlase de forma muy precisa los pasos que hacen los motores paso a paso. Aunque con la solución que se ha optado funcionan correctamente y por seguridad se han puesto finales de carrera por si se saltara alguna vez un paso, aunque no hemos tenido este problema, pero previsiblemente se piensa que con la correa dentada sea algo mucho más controlado y preciso.

También se puede mejorar en el aspecto visual, ya que si se le añaden una serie de leds a los bloques, que se activaran según la puntuación obtenida o mientras que se mueven los personajes y llega a la meta el ganador. Esto haría que el sistema fuera más espectacular, pero como no era uno de los requisitos esenciales, nos decantamos por centrarnos más en el sistema de audio.

Otra mejora podría ser añadir nuevos personajes para que se pudieran elegir, esto se haría primero consiguiendo un personaje nuevo y colocándole una base como las del resto de personajes, y una vez hecho esto habría que añadir al firmware de la rampa el nuevo personaje, en la cadena que almacena los personajes, para que nos saliera en la pantalla LCD. También se iría al firmware de los carriles, donde habría que añadir los sonidos del nuevo personaje en la tarjeta microSD. Con esto y modificando un poco el código se podría añadir nuevos personajes.

5.3 Presupuesto

CÓDIGO	UD	DESCRIPCIÓN	MEDICIÓN	P.U.	P.T.
01		MATERIALES Y CONSTRUCCIÓN MECÁNICA			
01.01	2	Madera de calabo Tablero de madera de 250x100x7 cm	Ud.	32,42	64,84
01.02	8	Listones madera Listones de DM de 100x3x3 cm	Ud.	1,50	12
01.03	1	Bobina de Plástico PLA Filamento de plástico PLA de 1.75mm. 1Kg	Ud.	19,95	19,95
01.04	3	Pintura Pintura al agua de distintos colores. Bote de 500ml	Ud.	4,50	13,50
01.05	1	Bobina de Aluminio Bobina de aluminio 1 mm diámetro	Ud.	16,52	16,52
01.06	1	Pegamento Pack barras de pegamento termoplástico	Ud.	3	3
01.07	1	Pegamento Cola blanca rayt500gr 429-07 Cola blanca para madera	Ud.	2,60	2,60
01.08	1	Bridas Bolsa con 10 unidades grandes	Ud.	0,60	0,60
01.10	4	Alcayatas	Ud.	0,15	0,60

01.11	1	Espigas Bolsa con 100 unidades	Ud.	1	1
01.12	4	Personajes Juguetes distintos de plástico para usarlos de personajes	Ud.	2	8
01.13	4	Rodamientos Rodamiento de bolas RS Pro, Miniatura, 625-2Z	Ud.	1,43	5,72
01.14	1	Imanes Tiras magnéticas 100mm PP, Adhesivo trasero, 2.3mm de grosor, 10u	Ud.	5,56	5,56
01.15	1	Varilla Varilla roscada M5 1metro+8tuercas M5 autoblocantes+4 tornillos M5X20 (para sujetar poleas conducidas)	Ud.	1,8	1,80
01.16	8	Tuerca Tuercas auto-freanantes de M5	Ud.	0,1	0,80
01.17	1	Manguitos manguitos metálicos M5-M5 Hembra-Hembra 8 unidades para sujetar poleas conducidas	Ud.	1	1
01.18		Tornillería 16 Tornillos M3X20 y 16 Tuercas M3 (para los apoyos intermedios de los carriles) 16 Tornillos M3X50 (4 por motor) Pack Tornillos (para la construcción de las estructuras)	Ud.		4.88
		TOTAL MATERIALES			162.37

CÓDIGO	UD.	DESCRIPCIÓN	MEDICIÓN	P.U.	P.T.
02		MATERIAL ELECTRÓNICO COMÚN A LOS DOS BLOQUES			
02.01	51	Terminal desconexión rápida hembra PCB	Ud.	0,0754	3,83
02.02	51	Terminal desconexión rápida macho aéreo	Ud.	0,0744	3,79
02.03	5	led Amarillo 2mA smd	Ud.	0,1020	0,51
02.04	2	Material temo retráctil Tubo Termo retráctil, Flexible, Ignífugo, 2.4 mm, 0.094 ", 2:1, Negro, 3.94 ft, 1.22 m	Ud.	1,16	2,32
02.05	8	Condensadores para el desacoplo condensador 1206 0.1uF	Ud.	0,0861	0,69
02.06	24	Resistencia 1K Ω MULTICOMP MC0125W120 611K Resistencia SMD de Tipo Chip, Serie MC, 1 kohm, 200 V, 1206 [Métrica 3216], 125 mW, \pm 1%	Ud.	0,0034	0,08
02.07	19	Resistencia 15 Ω MULTICOMP MCWR12X15 02FTL Resistencia SMD de Tipo Chip, Película Gruesa, Serie MCWR, 15 kohm, 200 V, 1206 [Métrica 3216]	Ud.	0,0121	0,23
02.08	9	Resistencia 100 Ω MULTICOMP MCWR12X10 00FTL Resistencia SMD de Tipo Chip, Película Gruesa, Serie MCWR, 100 ohm, 200 V, 1206 [Métrica 3216]	Ud.	0,0118	0,11

02.09	5	Resistencia 22K Ω MULTICOMP MC0125W120 6122K Resistencia SMD de Tipo Chip, Película Gruesa, Serie MC, 22 kohm, 200 V, 1206 [Métrica 3216]	Ud.	0,0036	0,2
		TOTAL			11,76
03	BLOQUE RAMPA				
03.01	<u>Microcontrolador</u>				
03.01.01	1	Microcontrolador STMICROELECTRONICS N UCLEO-F411RE PLACA DESAR, STM32F411RE CORTEX-M4 MCU	Ud.	9,97	9,97
03.01.02	4	Soporte PCB para montaje en panel M4 ETTINGER 06.84.256 SEPA RADOR, 25/M4/1.8-PA	Ud.	0,523	2,09
03.02	<u>Zona de detección</u>				
03.02.01	10	OPTEK TECHNOLOGY OP550C PH OTOTRANSISTOR, SIDE VIEW	Ud.	0,662	6,62
03.02.02	10	VISHAY TSAL6200 Emisor de Infrarrojos, Alta Potencia, 100 mA, 15 ns, 15 ns, 17 °, 1.6 V, -40 °C	Ud.	0,289	2,89
03.02.03	2	Amplificador Operacional LM234n	Ud.	0,526	1,05
03.03	<u>Radiofrecuencia</u>				
03.03.01	1	RF SOLUTIONS RF600E IC, ENCODER KEELOQ, DIP8	Ud.	3,63	3,63

03.03.02	1	RF SOLUTIONS AM-RT4-433FR RF MODULE, TRANSMITTER, AM, 433MHZ	Ud.	8,41	8,41
03.04 <u>Servomotores</u>					
03.04.01	1	Servomotores SG-90 SODIAL(R) 6 piezas de Nueva SG-90 SG90 9g micro servos para el helicóptero de coches Plano Barco	Ud.	8,32	8,32
03.04.02	2	Amplificador Operacional LM234n	Ud.	0,526	1,05
03.04.03	6	conector cable a placa 3p 2.54mm	Ud.	0,0935	0,56
03.04.04	6	receptáculo cable a placa 3p 2.54mm	Ud.	0,117	0,70
03.05 <u>Circuito de Acondicionamiento</u>					
03.05.01	1	Botón Pulsador gran botón dome rojo	Ud.	19,24	19,24
03.05.02	1	cable jack 2m pelado	Ud.	2,576	2,58
03.05.03		conector jack montaje en chasis	Ud.	1,24	1,24
03.06 <u>Panel de control</u>					
03.06.01	3	Botones de control Rafi 1.10.001.011/0301 Momentary Switch (ON)-OFF	Ud.	2,69	8,07
03.06.02	1	pantalla lcd 16x02 MIDAS MC21605B6WR-BNMLW Pantalla LCD Alfanumérica, 16 x 2, Blanco sobre Azul, 5 V, Paralelo, Inglés, Cirílico, Transmisivo	Ud.	11,06	11,06
03.06.03	1	Potenciómetro 20kohm	Ud.	0,65	0,65

03.06.04	1	16x socket 2.54mm agujero pasante	Ud.	1,39	1,39
03.06.05	1	16x header 2.54mm agujero pasante	Ud.	0,416	0,42
03.06.06	4	separador latón hembra hex m2.5	Ud.	0,365	1,46
03.07	<u>Alimentación</u>				
03.07.01	1	Fuente de Alimentación Cerrada AC/DC, Compacta, Ajustable, Fijo, 90 V, 264 V, 25 W, 5 V, 5 A	Ud.	18,8	18,8
03.07.02	1	interruptor basculante montaje en panel spst	Ud.	1,20	1,20
TOTAL BLOQUE RAMPA					111,85
04	BLOQUE CARRILES				
04.01	<u>Microcontrolador</u>				
04.01.01	1	Microcontrolador STMICROELECTRONICS NUCLEO-F411RE PLACA DESAR, STM32F411RE CORTEX-M4 MCU	Ud.	9,97	9,97
04.01.02	4	Soporte base adhesiva	Ud.	0,341	1,36
04.02	<u>Motores paso a paso</u>				
04.02.01	1	STEPPER MOTOR AMAZON 5 UNIDADES BQ	Ud.	59	59
04.02.02	4	Drivers Motores paso a paso Stepper Motor Driver A4988	Ud.	5,95	23,8

04.03		<u>Finales de carrera</u>			
04.03.01	8	Microrruptores OMRON ELECTRONIC COMPONENTS D2FD- 01L30-1H Microinterruptor, SPDT, Palanca con Roldana Simulada, Soldable, 100 mA, 125 V	Ud.	1,078	8,62
04.03.02	1	Encoder 8:3 TEXAS INSTRUMENTS SN74LS348 N LOGIC, 8 - 3 LINE ENCODER, 16DIP	Ud.	5,48	5,48
04.04		<u>Sistema de Audio</u>			
04.04.01	2	altavoz 4ohm 4w	Ud.	4,97	9,94
04.04.02	1	Driver de sonido Mini MP3 Player	Ud.	7,99	7,99
04.05		<u>Radiofrecuencia</u>			
04.05.01	1	RF SOLUTIONS AM- HRR30-433 RF MODULE, RECEIVER, SUPER-REGEN, AM	Ud.	5,04	5,04
04.05.02	1	RF SOLUTIONS RF600D IC, DECODER KEELOQ, DIP18	Ud.	5,41	5,41
04.06		<u>Alimentación</u>			
04.06.01	1	Fuente de Alimentación AC/DC Cubierta, Compacta, Ajustable, Fija, 5V, 8A, 12V, 4ª	Ud.	57,62	57,62
04.06.02	1	Interruptor basculante montaje en panel spst	Ud.	1,20	1,20
		TOTAL BLOQUE CARRILES			195.43

El precio que se ha obtenido por partes en la tabla anterior, solo corresponde a los materiales empleados en la construcción, sin tener en cuenta los honorarios para los dos ingenieros que han llevado acabo el diseño y la construcción del proyecto.

Tomando como referencia el salario medio de un ingeniero titulado de la Universidad Carlos III de Madrid: 25€/hora

Esos 26.34€ se deben a:

Sueldo medio ingeniero junior 20,5€

Seguridad Social 28%

Desempleo 1,5%

Las horas empleadas corresponde a 4 horas/día, 20 días al mes en 6 meses, esto nos da un total de 480 horas.

$$\text{Honorarios} = \text{Personas} \cdot \frac{\text{Precio}}{\text{Hora}} \cdot \text{Número de horas}$$

$$\text{Honorarios} = 2 \cdot 26,34 \cdot 480 = 25286,40\text{€}$$

APARTADO	PRECIO
Materiales y construcción mecánica	162,37€
Material electrónico común a los dos Bloques	11,76€
Bloque Rampa	111,82€
Bloque Carriles	195,43€
Honorarios Ingenieros	25286,40€
COSTE TOTAL	25.667,78€

Bibliografía

1. Guía Diseño Para Todos. [En línea] [Citado el: 3 de Septiembre de 2016.] http://www.prodintec.es/catalogo/ficheros/aplicaciones/fichero_44_4441.pdf .
2. Fundación Sidar. [En línea] [Citado el: 3 de Septiembre de 2016.] <http://www.sidar.org/recur/desdi/usable/dudt.php> .
3. Colegio San Rafael. [En línea] [Citado el: 4 de Septiembre de 2016.] <http://www.sanrafaelcolegio.com/>.
4. Artículo: Juego, Juguetes y Discapacidad. La importancia del Diseño Universal. [En línea] [Citado el: 5 de Septiembre de 2016.] <http://www.cesya.es/sites/default/files/documentos/folleto%20AIJU.pdf>.
5. Productos de Ferias. [En línea] [Citado el: 4 de Septiembre de 2016.] <http://www.elpoderdelasideas.com/web-page/atg-carreras-de-caballos-de-suecia-la-carrera/>.
6. Máquina de Galton. [En línea] [Citado el: 9 de Septiembre de 2016.] https://es.wikipedia.org/wiki/M%C3%A1quina_de_Galton.
7. Explicación Máquina de Galton. [En línea] [Citado el: 8 de Septiembre de 2016.] <https://prezi.com/nwcdiaszaojc/maquina-de-galton/>.
8. Codificación Manchester. [En línea] [Citado el: 17 de Septiembre de 2016.] https://es.wikipedia.org/wiki/Codificaci%C3%B3n_Manchester.
9. RFSolutions RF600E. [En línea] [Citado el: 17 de septiembre de 2016.] <http://www.farnell.com/rf-solutions/rf600d/ic-decoder-keeloq-dip18/dp/1200973>.
10. DataSheet Pantalla LCD MIDAS MC21605B6WR-BNMLW. [En línea] [Citado el: 10 de Septiembre de 2016.] <http://www.farnell.com/datasheets/2051065.pdf>.
11. Tipos de conexión Jack. [En línea] [Citado el: 22 de Septiembre de 2016.] http://cefire.edu.gva.es/pluginfile.php/194573/mod_resource/content/0/contenidos/106/4_conectores_xlr_trs_ts_rca.html.
12. BOE . [En línea] [Citado el: 22 de Septiembre de 2016.] <https://www.boe.es/buscar/act.php?id=BOE-A-2003-22066>.

Índice de Acrónimos

AM	Amplitud Modulada
GND	Ground
GPIO	General Purpose Input/Output
HIS	High Speed Internal
LED	Light-Emitting Diode (Diodo emisor de luz)
microSD	micro Secure Digital
PCB	Printed Circuit Board (Placa de circuito impreso)
PLA	Poliácido Láctico
PLL	Phase-Locked Loop
RF	Radio Frecuencia

Anexos

ANEXO 1. Data Sheet

- Microcontrolador (nucleo f411re)

http://www.st.com/content/ccc/resource/technical/document/user_manual/98/2e/fa/4b/e0/82/43/b7/DM00105823.pdf/files/DM00105823.pdf/jcr:content/translations/en.DM00105823.pdf

- Led Infrarrojo (tsal6200)

<http://www.vishay.com/docs/81010/tsal6200.pdf>

- Fototransistor (optek op550a)

<http://optekinc.com/datasheets/op550a.pdf>

- Pantalla LCD (MC21605B6WR-BNMLW)

<http://www.farnell.com/datasheets/2051065.pdf>

- Botones (Rafi 1.10.001.011/0301)

<http://www.farnell.com/datasheets/10359.pdf>

- RF600E:

<http://www.circuitstoday.com/wp-content/uploads/2009/03/RF600D.pdf>

- AM-RT4-433:

<http://www.rfsolutions.co.uk/acatalog/DS013-9%20AM-RTx.pdf>

- Fuente de Alimentación (TXM 025-105)

<http://www.tracopower.com/products/txm.pdf>

- Servomotor (SG-90)

<http://www.micropik.com/PDF/SG90Servo.pdf>

ANEXO 2. Manual de Instrucciones

JUEGO DE CARRERAS ADAPTADO

DESCRIPCIÓN DE COMPONENTES



El juego de carreras está compuesto por los siguientes elementos:

El Bloque Rampa: Desde el cual se efectúa todo el control (selección de jugadores, selección de personajes...) a través del panel de control situado en el lateral. Una vez que se ha terminado de configurar la partida, basta con pulsar un botón para accionar la apertura de la trampilla, que hará que la bola caiga en una de las cinco casillas de la parte inferior de la rampa y que hará que se mueva el personaje del jugador que en ese momento esté tirando.



El Bloque Carriles: Compuesto por cuatro carriles donde pueden jugar hasta cuatro jugadores. Durante toda la partida se produce un barrido auditivo de todo lo que está sucediendo gracias a los dos altavoces que hay situados en cada lateral.



Botón Rojo: Se conecta a un terminal JACK hembra de 3.5mm situado en el lateral de la rampa. Es el encargado del accionamiento de la trampilla. Al terminal JACK se le puede conectar otro pulsador más conveniente para el jugador, si fuera necesario. Y se pueden cambiar los pulsadores en medio de la partida para que cada jugador use el pulsador que mejor le convenga.



Muñecos: Hay cuatro muñecos distintos en función del personaje que se elija: un dinosaurio, un caballo, un coche y una moto. Cada uno de los muñecos tiene una base con un imán en la parte inferior de la misma. Las cintas transportadoras tienen otra base con otro imán. Es posible intercambiar los muñecos entre los distintos jugadores en función del personaje que haya elegido cada uno.

MONTAJE



Cada Bloque es independiente y pueden estar separados, aunque ambos deben estar conectados para poder jugar. Ambos tienen un enchufe y un botón como el de la siguiente figura. Para detectar que el bloque está enchufado y conectado, el botón de alimentación debe estar encendido.

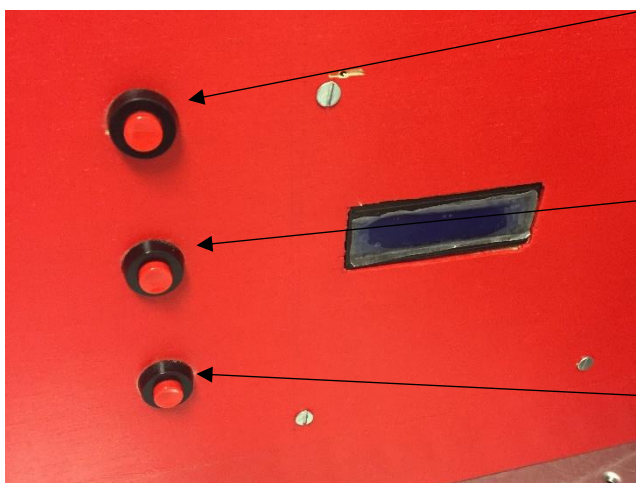
Ambos bloques deben estar en la misma habitación y los dos deben ser visibles por parte de los jugadores para disfrutar plenamente de la experiencia.

Aunque el orden en que se enciendan los dos bloques no influye en el funcionamiento del juego, se recomienda encender primero el Bloque Carriles y a continuación el Bloque Rampa. Si se hace al contrario se recomienda apretar antes de empezar a jugar el botón RESET del panel de control del Bloque Rampa.

Es requisito indispensable antes de empezar la configuración desde el panel de control que ambos bloques estén encendidos.

CONFIGURACIÓN

El panel de control dispone de 3 botones y un display.



BOTÓN MOVER: Es el botón situado en la parte superior de la pantalla. Mediante este botón nos desplazamos a través de las distintas opciones que nos ofrece el menú de la pantalla.

BOTÓN OK: Botón central mediante el cual se selecciona la opción que se desea elegir. También es el botón que hay que presionar antes de habilitar una nueva pulsación del botón de accionamiento.

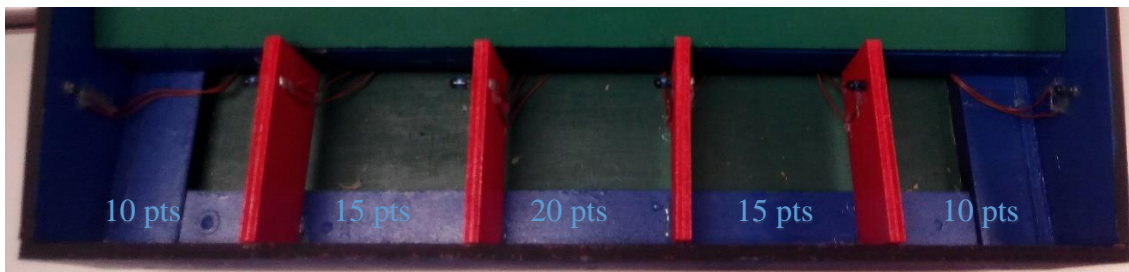
BOTÓN RESET: Botón habilitado para poder reiniciar la partida en cualquier momento si fuera necesario.

Para configurar una nueva partida se deben seguir los siguientes pasos:

1. El primer mensaje que aparecerá en la pantalla será *“Seleccione número de jugadores:”*. Inmediatamente pondrá *“número de jugadores: 4”*. Para cambiar el número de jugadores presione el BOTÓN MOVER. Cuando esté de acuerdo con su elección presione el BOTÓN OK.
2. A continuación, aparecerá un mensaje en la pantalla: *“Seleccione Personajes”*. Seleccione qué personaje desea que tenga cada jugador en la partida. Y en función de su elección coloque los muñecos en su posición correspondiente.
3. Comienza la partida. En la pantalla aparece un mensaje: *“Jugador 1: Presione Botón Rojo”*. En este momento comienza además a sonar la música característica que indica que la configuración de la partida ya ha terminado y la partida ha comenzado.

JUEGO

El objetivo del juego es que el jugador llegue al final de la carrera antes que los demás jugadores. Para ello la bola debe caer en una de las cinco casillas de la parte inferior de la rampa. Cada una de las casillas tiene una puntuación en función de la cual se moverá el personaje una distancia proporcional.



La puntuación necesaria para ganar la partida es de 70 puntos. La longitud de la cinta transportadora sobre la que se desplazan los carriles es de 70 cm. Es decir, cada punto conseguido corresponde a un centímetro. Si el jugador consigue 20 puntos, el personaje suyo se desplazará 20 centímetros.

Con tal de que todos los jugadores tengan las mismas oportunidades de ganar, se establecen las siguientes condiciones:

- Si en una misma ronda, dos o más jugadores llegan al final de la carrera. Ganará aquel que hubiera conseguido una mayor puntuación.
- Si dos o más jugadores llegan al final de la carrera con la misma puntuación, ganará el último de ellos que hubiera logrado esa puntuación.
- El ganador se establece al final de cada ronda. Si hay cuatro jugadores, los cuatro jugadores tienen la oportunidad de jugar en esa ronda, aunque el primero hubiera llegado ya a meta.

Cuando empieza el juego:

- Es el turno del primer jugador, que deberá haber colocado su pulsador conveniente antes de terminar la configuración de la partida.
- Cuando el primer jugador haya terminado su turno y se haya movido su personaje, se debe volver a colocar la pelota en la caseta de la rampa.
- Antes de que el siguiente jugador empiece su turno, se debe pulsar el BOTÓN OK en el panel de control para habilitar la pulsación del siguiente jugador. Si el siguiente jugador tiene que cambiar su pulsador, es imprescindible que lo haga antes de pulsar el BOTÓN OK, ya que de lo contrario el accionamiento de la trampilla se puede producir de forma inesperada.
- Este proceso se repite continuamente hasta que uno de los jugadores gane la partida. Cuando termina el juego, los personajes vuelven automáticamente a su posición inicial. En este momento se muestra en la pantalla del panel de control el jugador que ha ganado y su puntuación. Para volver a empezar otra partida se debe pulsar el BOTÓN RESET.
- Si se desea interrumpir el transcurso de juego antes de que finalice, basta con pulsar el BOTÓN RESET. Los personajes vuelven automáticamente a su posición inicial. Si no lo hicieran, se pueden desplazar fácilmente con la mano.

GUÍA RÁPIDA ANTE POSIBLES INCIDENCIAS.

- **Cuando cae la pelota, la casilla no ha detectado su paso y en la pantalla continúa saliendo el mensaje de “Bola cayendo, buena suerte.”**

Es posible que en este caso alguno de los sensores de las casillas no esté correctamente alineado. Asegura que todas las cabezas de los leds estén en ángulo recto con la madera.

- **Durante la configuración de la partida, la pantalla se ha quedado congelada y ni el BOTÓN OK ni el BOTÓN MOVER responden.**

En este caso, la solución está en pulsar el BOTÓN RESET y volver a configurar la partida.

- **Durante el intercambio del pulsador para el turno del siguiente jugador, se ha accionado la apertura de la trampilla antes incluso de conectar el siguiente pulsador.**

Suele ocurrir cuando el BOTÓN OK se ha pulsado antes de cambiar el pulsador. Hay que asegurarse de conectar el nuevo pulsador antes de presionar el BOTÓN OK.

- **La partida aparentemente ya ha llegado a su fin, pero no ocurre nada.**

Del mismo modo que cada vez que se cambia de turno hay que pulsar el BOTÓN OK, también hay que hacerlo al final de esta última ronda.

- **El interruptor ON/OFF del Bloque Carriles no se enciende. No sé si el Bloque Carriles está conectado o no.**

Si el botón de ON/OFF no se enciende, una forma de comprobar si el Bloque Carriles está realmente encendido es mirar a través de las cintas transportadoras hacia el interior del bloque. Si está encendido se observarán una serie de luces en la placa de circuito impreso.

ANEXO 3. Programa Bloque rampa

Main.c

```
/**
*****
* File Name      : main.c
* Description    : Main program body
*****

COPYRIGHT(c) 2016 STMicroelectronics

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:
    1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
    2. Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with the
distribution.
    3. Neither the name of STMicroelectronics nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Created on: Aug 9, 2016
Author: scastillom

*****/

/* Includes -----*/
-----*/
#include "stm32f4xx_hal.h"
#include "RF.h"
#include "LCD.h"
#include "controlpanel.h"
#include "servomotors.h"
/* Defines -----*/
#define MAXPOINTS 70
/* Private variables -----*/
TIM_HandleTypeDef htim3;
TIM_HandleTypeDef htim4;

volatile uint8_t IR_Flag = 0;
volatile uint8_t servos_Cnt = 0;

/* Private function prototypes -----*/
void SystemClock_Config(void);
void Error_Handler(void);
static void MX_GPIO_Init(void);
static void MX_TIM3_Init(void);
static void MX_TIM4_Init(void);
```

```
static void MX_NVIC_Init(void);

void HAL_TIM_MspPostInit(TIM_HandleTypeDef *htim);

int main(void) {
    uint8_t commandToSend = 0;
    uint8_t pointPlayers[4] = {0,0,0,0};
    /* MCU Configuration-----
    */
    /* Reset of all peripherals, Initializes the Flash interface and the
    SysTick. */
    HAL_Init();
    /* Configure the system clock */
    SystemClock_Config();
    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_TIM3_Init();
    MX_TIM4_Init();
    /* Initialize interrupts */
    MX_NVIC_Init();
    //Reinicializa variables del bloque carriles
    commandToSend = RESTART;
    RF_Command(commandToSend);
    HAL_Delay(160);
    commandToSend = 0;
    RF_Command(commandToSend);
    //menú pantalla inicio
    init_PanelConfig();
    //elige el número de jugadores que va a haber en la partida
    uint8_t jugadores;
    jugadores = numeroJugadores();
    //guarda el comando correspondiente al número de jugadores
    //para a continuación mandarlo
    switch (jugadores)
    {
    case 4:
        commandToSend = PLAYER4;
        break;
    case 3:
        commandToSend = PLAYER3;
        break;
    case 2:
        commandToSend = PLAYER2;
        break;
    case 1:
        commandToSend = PLAYER1;
        break;
    }
    //manda el comando correspondiente al número de jugadores mediante
    RF
    //y a continuación se reinicia la variable commandToSend
    RF_Command(commandToSend);
    HAL_Delay(160);
    commandToSend = 0;
    RF_Command(commandToSend);
    //se crea array para guardar el personaje elegido por cada jugador
    uint8_t characters[jugadores];
    //se inicializa el array con el tamaño en función del número de
    //jugadores
    uint8_t i;
    for(i=0; i<jugadores; i++) //inicialización de array
```

```
{
    characters[i] = 0;
}
//se asigna a cada jugador un personaje y se manda mediante RF
for ( i=0 ; i<jugadores ; i++)
{
    characters[i] = seleccionPersonaje(i+1);
    switch(characters[i])
    {
    case 1:
        commandToSend = CHARACTER1;
        break;
    case 2:
        commandToSend = CHARACTER2;
        break;
    case 3:
        commandToSend = CHARACTER3;
        break;
    case 4:
        commandToSend = CHARACTER4;
        break;
    }
    RF_Command(commandToSend);
    commandToSend = 0;
    HAL_Delay(160);
    RF_Command(commandToSend);
}
startServomotors();
moveServomotor(5, 180);
moveServomotor(6, 0);
//mientras que no se haya superado la puntuación máxima, se continúa
lanzando y esperando
//recepción de bola de pingpong por parte de sensores infrarrojos y
posterior lectura
// de puntuación y su envío mediante rf
while ((pointPlayers[0] < MAXPOINTS) && (pointPlayers[1] <
MAXPOINTS) && (pointPlayers[2] < MAXPOINTS) && (pointPlayers[3] <
MAXPOINTS))
{
    for (i=0 ; i<jugadores ; i++)
    {
        redButtonScreen(i+1);

        moveServomotor(5, 90);
        moveServomotor(6, 105); //para "igualar" con respecto
al otro servo se pone 105 en vez de 90
        servos_Cnt = 24; //para evitar delay que
se produce entre apertura y mov de servos rampa
        IR_Flag = 0;

        while(IR_Flag == 0)
        {
            if (servos_Cnt == 25) //cada vez que
tim_it_update se genera se aumenta el contador
            {
                //tim_update se genera cada 20ms, y queremos mover los servos de
la //rampa
                //rampa
                cada medio sec (0.5sec/20e-3=25)
                moveServosRamp();
                servos_Cnt = 0;
            }
        }
    }
}
```

```
        }
    }
    switch(IR_Flag)
    {
    case 10:
        pointPlayers[i] = pointPlayers[i] + 10;
        pointPlayersScreen(10, i+1);
        commandToSend = POINTS10;
        break;
    case 15:
        pointPlayers[i] = pointPlayers[i] + 15;
        pointPlayersScreen(15, i+1);
        commandToSend = POINTS15;
        break;
    case 20:
        pointPlayers[i] = pointPlayers[i] + 20;
        pointPlayersScreen(20, i+1);
        commandToSend = POINTS20;
        break;
    }
    RF_Command(commandToSend);
    commandToSend = 0;
    HAL_Delay(160);
    RF_Command(commandToSend);
    moveServomotor(5, 180);
    moveServomotor(6, 0);
    lcd_clear();
    lcd_on();
    lcd_goto(0,0);
    lcd_puts("PULSE OK PARA");
    lcd_goto(0,1);
    lcd_puts("CONTINUAR.");
    waitOkButton();
    }
}
stopServomotors();
//se envía comando de fin de juego al bloque de los carriles, que
hace que
//los motores retrocedan a su posición inicial.
HAL_Delay(5000);
commandToSend = ENDGAME;
RF_Command(commandToSend);
commandToSend = 0;
HAL_Delay(160);
RF_Command(commandToSend);
while (1)
{
    //en caso de que más de un jugador llegue al final en la misma
ronda, gana aquel que haya conseguido
    //una mayor puntuación a lo largo de las tiradas. Si más de un
jugador tiene la misma puntuación, gana
    //aquel que haya tirado el último.
    endScreen(pointPlayers);
}
return 0;
}

/** System Clock Configuration
*/
void SystemClock_Config(void)
{
```

```
RCC_OscInitTypeDef RCC_OscInitStruct;
RCC_ClkInitTypeDef RCC_ClkInitStruct;

__HAL_RCC_PWR_CLK_ENABLE();

__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = 16;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
RCC_OscInitStruct.PLL.PLLM = 16;
RCC_OscInitStruct.PLL.PLLN = 336;
RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
RCC_OscInitStruct.PLL.PLLQ = 4;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
    Error_Handler();
}

RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) !=
HAL_OK)
{
    Error_Handler();
}

HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

/* SysTick_IRQn interrupt configuration */
HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}

/** NVIC Configuration
*/
static void MX_NVIC_Init(void)
{
    /* EXTI0_IRQn interrupt configuration */
    HAL_NVIC_SetPriority(EXTI0_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(EXTI0_IRQn);
    /* EXTI1_IRQn interrupt configuration */
    HAL_NVIC_SetPriority(EXTI1_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(EXTI1_IRQn);
    /* EXTI2_IRQn interrupt configuration */
    HAL_NVIC_SetPriority(EXTI2_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(EXTI2_IRQn);
    /* EXTI3_IRQn interrupt configuration */
    HAL_NVIC_SetPriority(EXTI3_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(EXTI3_IRQn);
    /* EXTI4_IRQn interrupt configuration */
```

```
HAL_NVIC_SetPriority(EXTI4_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(EXTI4_IRQn);
/* EXTI9_5_IRQn interrupt configuration */
HAL_NVIC_SetPriority(EXTI9_5_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);
/* EXTI15_10_IRQn interrupt configuration */
HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
/* TIM3_IRQn interrupt configuration */
HAL_NVIC_SetPriority(TIM3_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(TIM3_IRQn);
/* TIM4_IRQn interrupt configuration */
HAL_NVIC_SetPriority(TIM4_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(TIM4_IRQn);
}

/* TIM3 init function */
static void MX_TIM3_Init(void)
{
    TIM_MasterConfigTypeDef sMasterConfig;
    TIM_OC_InitTypeDef sConfigOC;

    htim3.Instance = TIM3;
    htim3.Init.Prescaler = 84;
    htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim3.Init.Period = 20000;
    htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    if (HAL_TIM_PWM_Init(&htim3) != HAL_OK)
    {
        Error_Handler();
    }

    sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
    sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
    if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) !=
    HAL_OK)
    {
        Error_Handler();
    }

    sConfigOC.OCMode = TIM_OCMODE_PWM1;
    sConfigOC.Pulse = 2650;
    sConfigOC.OCpolarity = TIM_OCPOLARITY_HIGH;
    sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
    if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_1) !=
    HAL_OK)
    {
        Error_Handler();
    }

    if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_2) !=
    HAL_OK)
    {
        Error_Handler();
    }

    if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_3) !=
    HAL_OK)
    {
        Error_Handler();
    }
}
```



```
    }

    HAL_TIM_MspPostInit (&htim3);
}

/* TIM4 init function */
static void MX_TIM4_Init(void)
{
    TIM_MasterConfigTypeDef sMasterConfig;
    TIM_OC_InitTypeDef sConfigOC;

    htim4.Instance = TIM4;
    htim4.Init.Prescaler = 84;
    htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim4.Init.Period = 20000;
    htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    if (HAL_TIM_PWM_Init(&htim4) != HAL_OK)
    {
        Error_Handler();
    }

    sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
    sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
    if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) !=
        HAL_OK)
    {
        Error_Handler();
    }

    sConfigOC.OCMode = TIM_OCMODE_PWM1;
    sConfigOC.Pulse = 2650;
    sConfigOC.OCpolarity = TIM_OCPOLARITY_HIGH;
    sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
    if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_1) !=
        HAL_OK)
    {
        Error_Handler();
    }

    if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_2) !=
        HAL_OK)
    {
        Error_Handler();
    }

    if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_3) !=
        HAL_OK)
    {
        Error_Handler();
    }

    HAL_TIM_MspPostInit (&htim4);
}

/** Configure pins as
    * Analog
    * Input
    * Output
```

```
        * EVENT_OUT
        * EXTI
    PA2    -----> USART2_TX
    PA3    -----> USART2_RX
*/
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;

    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOH_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();

    /*Configure GPIO pin : B1_Pin */
    GPIO_InitStructure.Pin = B1_Pin;
    GPIO_InitStructure.Mode = GPIO_MODE_EVT_RISING;
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStructure);

    /*Configure GPIO pins : IR1_Pin IR2_Pin IR3_Pin IR4_Pin */
    GPIO_InitStructure.Pin = IR1_Pin|IR2_Pin|IR3_Pin|IR4_Pin;
    GPIO_InitStructure.Mode = GPIO_MODE_IT_FALLING;
    GPIO_InitStructure.Pull = GPIO_PULLDOWN;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStructure);

    /*Configure GPIO pins : USART_TX_Pin USART_RX_Pin */
    GPIO_InitStructure.Pin = USART_TX_Pin|USART_RX_Pin;
    GPIO_InitStructure.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStructure.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStructure);

    /*Configure GPIO pin : IR5_Pin */
    GPIO_InitStructure.Pin = IR5_Pin;
    GPIO_InitStructure.Mode = GPIO_MODE_IT_FALLING;
    GPIO_InitStructure.Pull = GPIO_PULLDOWN;
    HAL_GPIO_Init(IR5_GPIO_Port, &GPIO_InitStructure);

    /*Configure GPIO pin : LD2_Pin */
    GPIO_InitStructure.Pin = LD2_Pin;
    GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStructure);

    /*Configure GPIO pins : BOTONROJO_Pin BOTON1_Pin BOTON2_Pin
    BOTON3_Pin */
    GPIO_InitStructure.Pin =
    BOTONROJO_Pin|BOTON1_Pin|BOTON2_Pin|BOTON3_Pin;
    GPIO_InitStructure.Mode = GPIO_MODE_IT_RISING;
    GPIO_InitStructure.Pull = GPIO_PULLDOWN;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStructure);

    /*Configure GPIO pins : S0_Pin S3_Pin S2_Pin S1_Pin */
    GPIO_InitStructure.Pin = S0_Pin|S3_Pin|S2_Pin|S1_Pin;
    GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStructure.Pull = GPIO_PULLDOWN;
```

```
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_MEDIUM;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/*Configure GPIO pins : RS_Pin RW_Pin */
GPIO_InitStruct.Pin = RS_Pin|RW_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_MEDIUM;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/*Configure GPIO pin : E_Pin */
GPIO_InitStruct.Pin = E_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_MEDIUM;
HAL_GPIO_Init(E_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pins : D0_Pin D1_Pin D2_Pin */
GPIO_InitStruct.Pin = D0_Pin|D1_Pin|D2_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_PULLDOWN;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_MEDIUM;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

/*Configure GPIO pins : D3_Pin D4_Pin D5_Pin D6_Pin
D7_Pin */
GPIO_InitStruct.Pin = D3_Pin|D4_Pin|D5_Pin|D6_Pin
|D7_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_PULLDOWN;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_MEDIUM;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOA, LD2_Pin|D3_Pin|D4_Pin|D5_Pin
|D6_Pin|D7_Pin, GPIO_PIN_RESET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOB, S0_Pin|RS_Pin|RW_Pin|S3_Pin
|S2_Pin|S1_Pin, GPIO_PIN_RESET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOC, E_Pin|D0_Pin|D1_Pin|D2_Pin,
GPIO_PIN_RESET);
}

/**
 * @brief This function is executed in case of error occurrence.
 * @param None
 * @retval None
 */
void Error_Handler(void)
{
    /* USER CODE BEGIN Error_Handler */
    /* User can add his own implementation to report the HAL error
return state */
    while(1)
    {
    }
    /* USER CODE END Error_Handler */
}
```

```
}

#ifdef USE_FULL_ASSERT

/**
 * @brief Reports the name of the source file and the source line
 number
 * where the assert_param error has occurred.
 * @param file: pointer to the source file name
 * @param line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t* file, uint32_t line)
{
    /* USER CODE BEGIN 6 */
    /* User can add his own implementation to report the file name and
 line number,
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
 line) */
    /* USER CODE END 6 */
}

#endif

/***** (C) COPYRIGHT STMicroelectronics *****/
FILE*****/
```

Servomotor.c

```
/*
 * servomotors.c
 * Created on: Aug 9, 2016
 * Author: scastillom
 ****
 ****
 */
#include "servomotors.h"
/** @moveServomotor
 * @brief mueve el servomotor que se quiera un número determinado
 * de grados entre 0 y 180.
 * @param servoMotor: el servo que se quiere mover, hay hasta 6.
 * @param degrees: número de grados que se desea mover el servo.
 * @retval none
 */
void moveServomotor(uint8_t servoMotor, double degrees)
{ //y=mx+n -> n=2650, m=-((1700-750)/(210-105))=-9.05,
y=ccrx, x= degrees
uint32_t CCRX = 0;
double m = -9.05;
double n = 2650;

CCRX = round((m*degrees) + n);

if (CCRX > 2650)
{
    CCRX = 2650;
}

if (CCRX < 750)
{
    CCRX = 750;
}

switch(servoMotor)
{
case 1:
    TIM3->CCR1=CCRX;
    break;
case 2:
    TIM3->CCR2=CCRX;
    break;
case 3:
    TIM3->CCR3=CCRX;
    break;
case 4:
    TIM4->CCR1=CCRX;
    break;
case 5:
    TIM4->CCR2=CCRX;
    break;
case 6:
    TIM4->CCR3=CCRX;
    break;
}
} //moveServomotor

/** @startServomotors
 * @brief inicializa todos los servos y la interrupción del tim3 por
 "update".
```

```
* @param none
* @retval none
*/
void startServomotors (void)
{
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2);
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_3);
    HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_1);
    HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_2);
    HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_3);

    HAL_TIM_Base_Start_IT(&htim3);
} //startServomotors

/** @stopServomotors
 * @brief Envía servomotores a posición inicial y los para.
 * @param none
 * @retval none
 */
void stopServomotors (void)
{
    uint32_t CCRX = 2650;

    TIM3->CCR1=CCRX;
    TIM3->CCR2=CCRX;
    TIM3->CCR3=CCRX;
    TIM4->CCR1=CCRX;
    TIM4->CCR2=CCRX;
    TIM4->CCR3=CCRX;

    HAL_TIM_Base_Stop_IT(&htim3);

    HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_1);
    HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_2);
    HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_3);
    HAL_TIM_PWM_Stop(&htim4, TIM_CHANNEL_1);
    HAL_TIM_PWM_Stop(&htim4, TIM_CHANNEL_2);
    HAL_TIM_PWM_Stop(&htim4, TIM_CHANNEL_3);
} //stopServomotors

/** @moveServosRamp
 * @brief Mueve lo servomotores de la rampa a una posición aleatoria
 *         entre 0 y 180 grados.
 * @param none
 * @retval none
 */
void moveServosRamp (void)
{
    uint8_t random = 0;
    do {random = rand();} while (random > 180);
    moveServomotor(1, random);
    do {random = rand();} while (random > 180);
    moveServomotor(2, random);
    do {random = rand();} while (random > 180);
    moveServomotor(3, random);
    do {random = rand();} while (random > 180);
    moveServomotor(4, random);
}
}
```

Controlpanel.c

```
/*
 * controlpanel.c
 * Created on: Aug 18, 2016
 * Author: scast
 */

/* Includes -----
-----*/
#include <string.h>
#include "controlpanel.h"

/* Extern Variables -----
-----*/
volatile uint8_t boton_Flag = 0;
volatile uint8_t debounce_Cnt = 0;

void init_PanelConfig(void)
{
    // Initialize lcd driver
    lcd_init(0, 16, 2);
    // Clear screen to end write to character generator
    lcd_clear();
    // Turn on display
    lcd_on();
    // Display message on the first row
    lcd_puts("Bienvenido!");
    // Move cursor to the fourth column on second line
    lcd_goto(0, 1);
    // Print a message there
    lcd_puts("Pulse OK o MOVER");

    while ((boton_Flag != 3) && (boton_Flag != 2));
    HAL_Delay(200);
    boton_Flag = 0;
    debounce_Cnt = 0;
}

uint8_t numeroJugadores(void)
{
    uint8_t players = 4;
    uint8_t refresh = 0; //para evitar que el lcd se
    refresque continuamente.

    lcd_clear();
    lcd_goto(0,0);
    lcd_on();
    lcd_puts("Seleccione nume-");
    lcd_goto(0,1);
    lcd_puts("ro de jugadores.");

    HAL_Delay(3000);

    lcd_clear();
    lcd_goto(0,0);
    lcd_on();
    lcd_puts("Numero de");
    lcd_goto(0,1);
    lcd_puts("jugadores: 4");
}
```

```
while (boton_Flag != 3)
{
    if ((boton_Flag == 2))
    {
        HAL_Delay(200);
        boton_Flag = 0;
        debounce_Cnt = 0;
        refresh = 0;
        players++;
        if ( (players > 4) || (players < 1) ) {players = 1;}
        switch (players)
        {
            case 1:
                if (refresh == 0)
                {
                    lcd_clear();
                    lcd_goto(0,0);
                    lcd_on();
                    lcd_puts("Numero de");
                    lcd_goto(0,1);
                    lcd_puts("jugadores: 1");
                }
                refresh = 1;
                break;
            case 2:
                if (refresh == 0)
                {
                    lcd_clear();
                    lcd_goto(0,0);
                    lcd_on();
                    lcd_puts("Numero de");
                    lcd_goto(0,1);
                    lcd_puts("jugadores: 2");
                }
                refresh = 1;
                break;
            case 3:
                if (refresh == 0)
                {
                    lcd_clear();
                    lcd_goto(0,0);
                    lcd_on();
                    lcd_puts("Numero de");
                    lcd_goto(0,1);
                    lcd_puts("jugadores: 3");
                }
                refresh = 1;
                break;
            case 4:
                if (refresh == 0)
                {
                    lcd_clear();
                    lcd_goto(0,0);
                    lcd_on();
                    lcd_puts("Numero de");
                    lcd_goto(0,1);
                    lcd_puts("jugadores: 4");
                }
                refresh = 1;
                break;
        }
    }
}
```



```
        }  
    }  
    return players;  
}  
  
uint8_t seleccionPersonaje(uint8_t jugador)  
{  
    HAL_Delay(200);  
    boton_Flag = 0;  
    debounce_Cnt = 0;  
    uint8_t personajes [4][16] =  
        {  
            {"Caballo", "\0"},  
            {"Dinosaurio", "\0"},  
            {"Coche", "\0"},  
            {"Moto", "\0"}  
        };  
    uint8_t charLCD[16] = "          ";  
    uint8_t personaje = 0;  
    uint8_t refresh = 0;  
  
    if (jugador == 1)  
    {  
        lcd_clear();  
        lcd_goto(0,0);  
        lcd_on();  
        lcd_puts("Seleccione ");  
        lcd_goto(0,1);  
        lcd_puts("personajes.");  
  
        HAL_Delay(2000);  
        boton_Flag = 0;  
        debounce_Cnt = 0;  
    }  
  
    lcd_clear();  
    lcd_on();  
    lcd_goto(0,1);  
    lcd_puts("Presione MOVER");  
    lcd_goto(0,0);  
    lcd_puts("Jugador  :");  
    lcd_goto(8,0);  
    lcd_send( (jugador + 48) , 1);  
  
    while (boton_Flag !=2)  
    {  
        if (boton_Flag == 3)  
        {  
            HAL_Delay(200);  
            boton_Flag = 0;  
            debounce_Cnt = 0;  
        }  
    }  
  
    while (boton_Flag != 3)  
    {  
        if ((boton_Flag == 2))  
        {  
            HAL_Delay(200);  
            boton_Flag = 0;  
        }  
    }  
}
```

```
        debounce_Cnt = 0;
        refresh = 0;

        personaje = checkPersonajes(personaje);
        strncpy(charLCD, personajes[personaje - 1], 16);

        if (refresh == 0)
        {
            lcd_clear();
            lcd_on();
            lcd_goto(0,1);
            lcd_puts(&charLCD);
            lcd_goto(0,0);
            lcd_puts("Jugador  :");
            lcd_goto(8,0);
            lcd_send( (jugador + 48) , 1);
        }
        refresh = 1;
    }
}

usedFlag |= (1 << (personaje-1));
return personaje;
}

uint8_t checkPersonajes (uint8_t personaje)
{
    personaje++;

    if (usedFlag == 0x00)
    {
        if (personaje > 4)
        {
            personaje = 1;
        }
    }

    if (usedFlag == 0x01)
    {
        if (personaje > 4 || personaje == 1)
        {
            personaje = 2;
        }
    }

    if (usedFlag == 0x02)
    {
        if (personaje == 2)
        {
            personaje++;
        }

        if (personaje > 4)
        {
            personaje = 1;
        }
    }

    if (usedFlag == 0x03)
    {
        if (personaje > 4 || personaje == 1 || personaje == 2)
```

```
        {
            personaje = 3;
        }
    }

    if (usedFlag == 0x04)
    {
        if (personaje == 3)
        {
            personaje++;
        }
        else
        {
            if (personaje > 4)
            {
                personaje = 1;
            }
        }
    }

    if (usedFlag == 0x05)
    {
        if (personaje == 1 || personaje == 3)
        {
            personaje++;
        }

        if (personaje > 4)
        {
            personaje = 2;
        }
    }

    if (usedFlag == 0x06)
    {
        if (personaje == 2 || personaje == 3)
        {
            personaje = 4;
        }

        if (personaje > 4)
        {
            personaje = 1;
        }
    }

    if (usedFlag == 0x07)
    {
        personaje = 4;
    }

    if (usedFlag == 0x08)
    {
        if (personaje >= 4)
        {
            personaje = 1;
        }
    }

    if (usedFlag == 0x09)
    {
        if (personaje == 1 || personaje >= 4)
```

```
        {
            personaje = 2;
        }
    }

    if (usedFlag == 0x0A)
    {
        if (personaje == 2)
        {
            personaje++;
        }

        if (personaje >=4)
        {
            personaje = 1;
        }
    }

    if (usedFlag == 0x0B)
    {
        personaje = 3;
    }

    if (usedFlag == 0x0C)
    {
        if (personaje == 3 || personaje >= 4)
        {
            personaje = 1;
        }
    }

    if (usedFlag == 0x0D)
    {
        personaje = 2;
    }

    if (usedFlag == 0x0E)
    {
        personaje = 1;
    }

    return personaje;
}

void redButtonScreen (uint8_t jugador)
{
    HAL_Delay(200);
    boton_Flag = 0;
    debounce_Cnt = 0;
    uint8_t refresh = 0;

    while (boton_Flag != 4)
    {
        if (refresh == 0)
        {
            lcd_clear();
            lcd_on();
            lcd_goto(0,1);
            lcd_puts("Pulse BOTON ROJO");
            lcd_goto(0,0);
            lcd_puts("Jugador  :");
        }
    }
}
```

```
        lcd_goto(8,0);
        lcd_send( (jugador + 48) , 1);
    }
    refresh = 1;
}

refresh = 0;
if (refresh == 0)
{
    lcd_clear();
    lcd_on();
    lcd_goto(0,0);
    lcd_puts("BOLA CAYENDO,");
    lcd_goto(0,1);
    lcd_puts("BUENA SUERTE.");
}

HAL_Delay(200);
refresh = 1;
boton_Flag = 0;
debounce_Cnt = 0;
}

void endScreen (uint8_t *puntuaciones)
{
    uint8_t i;
    uint8_t max = 0;
    uint8_t max1 = 0;
    uint8_t max2 = 0;
    uint8_t jugador = 0;

    for (i=0 ; i<4 ; i++)
    {
        if (puntuaciones[i] >= max)
        {
            max=puntuaciones[i];
            jugador = i+1;
        }
    }

    max1 = max % 10;
    max = max / 10;
    max2 = max % 10;

    HAL_Delay(2000);

    lcd_clear();
    lcd_on();
    lcd_goto(0,0);
    lcd_puts("JUGADOR   GANA");
    lcd_goto(0,1);
    lcd_puts("CON     PUNTOS!");
    lcd_goto(8,0);
    lcd_send( (jugador + 48) , 1);
    lcd_goto(5,1);
    lcd_send( (max1 + 48) , 1);
    lcd_goto(4,1);
    lcd_send( (max2 + 48) , 1);

    HAL_Delay(2000);
}
```

```
    lcd_clear();
    lcd_on();
    lcd_goto(0,0);
    lcd_puts("PULSE RESET PARA");
    lcd_goto(0,1);
    lcd_puts("NUEVA PARTIDA.");
}

void pointPlayersScreen ( uint8_t puntuacion, uint8_t jugador)
{
    uint8_t puntuacion1 = 0;
    uint8_t puntuacion2 = 0;

    puntuacion1 = puntuacion % 10;
    puntuacion = puntuacion / 10;
    puntuacion2 = puntuacion % 10;

    HAL_Delay(500);

    lcd_clear();
    lcd_on();
    lcd_goto(0,0);
    lcd_puts("JUGADOR   LOGRA");
    lcd_goto(0,1);
    lcd_puts("   PUNTOS!!");
    lcd_goto(8,0);
    lcd_send( (jugador + 48) , 1);
    lcd_goto(1,1);
    lcd_send( (puntuacion1 + 48) , 1);
    lcd_goto(0,1);
    lcd_send( (puntuacion2 + 48) , 1);

    HAL_Delay(2000);
}

void waitOkButton(void)
{
    debounce_Cnt = 0;
    boton_Flag = 0;
    while (boton_Flag != 3)
    {
        if (boton_Flag == 2)
        {
            HAL_Delay(200);
            boton_Flag = 0;
            debounce_Cnt = 0;
        }
    }

    HAL_Delay(200);
    boton_Flag = 0;
    debounce_Cnt = 0;
}
```

LCD.c

```
/*
 * LCD.c
 *     fuente: https://github.com/geekfactory/LCD/blob/master/LCD.c
 *           http://www.geekfactory.mx/tutoriales/tutoriales-pic/pantalla-lcd-16x2-con-pic-libreria/
 *     Created on: Aug 16, 2016
 *     Author: scastillom
 ****
 */
#include "LCD.h"
// Local variables
const uint8_t rowaddr[4] = {0x00, 0x40, 0x14, 0x54};
uint8_t lcdrows = 2;
uint8_t lcdcolumns = 16;
// Local copy of the Display on off control register
uint8_t dispctrl = 0x00;
uint8_t iomode = 0;

uint8_t lcd_init(void * iodata, uint8_t cols, uint8_t rows)
{
    // Initialize IO pins
    iomode = lcd_ioinit(iodata);
    lcdrows = rows;
    lcdcolumns = cols;
    // Initial delay after power-up
    HAL_Delay(150);
    // Begin LCD controller Initialization (HD44780 page 45-46)
    lcd_command(E_FUNCTION_SET | BIT_DL_DATALENGTH_8);
    HAL_Delay(5);
    lcd_command(E_FUNCTION_SET | BIT_DL_DATALENGTH_8);
    HAL_Delay(2);
    lcd_command(E_FUNCTION_SET | BIT_DL_DATALENGTH_8);
    HAL_Delay(2);
    lcd_command(E_FUNCTION_SET | BIT_DL_DATALENGTH_8 |
    BIT_N_DISP_LINES_2 | BIT_F_FONT_5_8);
    HAL_Delay(1);
    // Configure display after power up
    lcd_command(E_DISPLAY_ON_OFF_CTRL | BIT_D_DISPLAY_OFF);
    HAL_Delay(1);
    lcd_command(E_CLEAR_DISPLAY);
    HAL_Delay(2);
    lcd_command(E_ENTRY_MODE_SET | BIT_S_AUTOSCROLL_OFF |
    BIT_ID_INCREMENT_CURSOR);
    HAL_Delay(1);
    return TRUE;
}

void lcd_clear()
{
    lcd_command(E_CLEAR_DISPLAY);
    HAL_Delay(2);
}

void lcd_home()
{
    lcd_command(E_RETURN_HOME);
    HAL_Delay(2);
}

```

```
void lcd_on()
{
    dispctrl |= BIT_D_DISPLAY_ON;
    lcd_command(E_DISPLAY_ON_OFF_CTRL | dispctrl);
    HAL_Delay(1);
}

void lcd_off()
{
    dispctrl &= ~BIT_D_DISPLAY_ON;
    lcd_command(E_DISPLAY_ON_OFF_CTRL | dispctrl);
    HAL_Delay(1);
}

void lcd_cursor(enum enLCDCursorModes mode)
{
    dispctrl &= 0xFC;
    dispctrl |= mode;
    lcd_command(E_DISPLAY_ON_OFF_CTRL | dispctrl);
    HAL_Delay(1);
}

void lcd_cursor_left()
{
    lcd_command(E_CURSOR_DISPLAY_SHIFT | BIT_SC_SHIFT_CURSOR |
    BIT_RL_SHIFT_LEFT);
    HAL_Delay(1);
}

void lcd_cursor_right()
{
    lcd_command(E_CURSOR_DISPLAY_SHIFT | BIT_SC_SHIFT_CURSOR |
    BIT_RL_SHIFT_RIGHT);
    HAL_Delay(1);
}

void lcd_scroll_left()
{
    lcd_command(E_CURSOR_DISPLAY_SHIFT | BIT_SC_SHIFT_DISPLAY |
    BIT_RL_SHIFT_LEFT);
    HAL_Delay(1);
}

void lcd_scroll_right()
{
    lcd_command(E_CURSOR_DISPLAY_SHIFT | BIT_SC_SHIFT_DISPLAY |
    BIT_RL_SHIFT_RIGHT);
    HAL_Delay(1);
}

void lcd_autoscroll_on()
{
    lcd_command(E_ENTRY_MODE_SET | BIT_S_AUTOSCROLL_ON |
    BIT_ID_INCREMENT_CURSOR);
    HAL_Delay(1);
}

void lcd_autoscroll_off()
{
    lcd_command(E_ENTRY_MODE_SET | BIT_S_AUTOSCROLL_OFF |
    BIT_ID_INCREMENT_CURSOR);
}
```



```
        HAL_Delay(1);
    }

void lcd_goto(uint8_t col, uint8_t row)
{
    // Apply limits for Rows and Columns
    if (row >= lcdrows)
    {row = lcdrows - 1;}
    if (col >= lcdcolumns)
    {col = lcdcolumns - 1;}
    lcd_command(E_SET_DDRAM_ADDR | (col + rowaddr[ row ]));
}

void lcd_send(uint8_t data, uint8_t rs)
{
    // Write logic one to send characters or logic 0 to send a
    command
    if (rs)
    {lcd_iohigh(E_RS_PIN);}
    else
    {lcd_iolow(E_RS_PIN);}
    // Clear the RW pin if used
    lcd_iolow(E_RW_PIN);
    lcd_iowrite8(data);
}

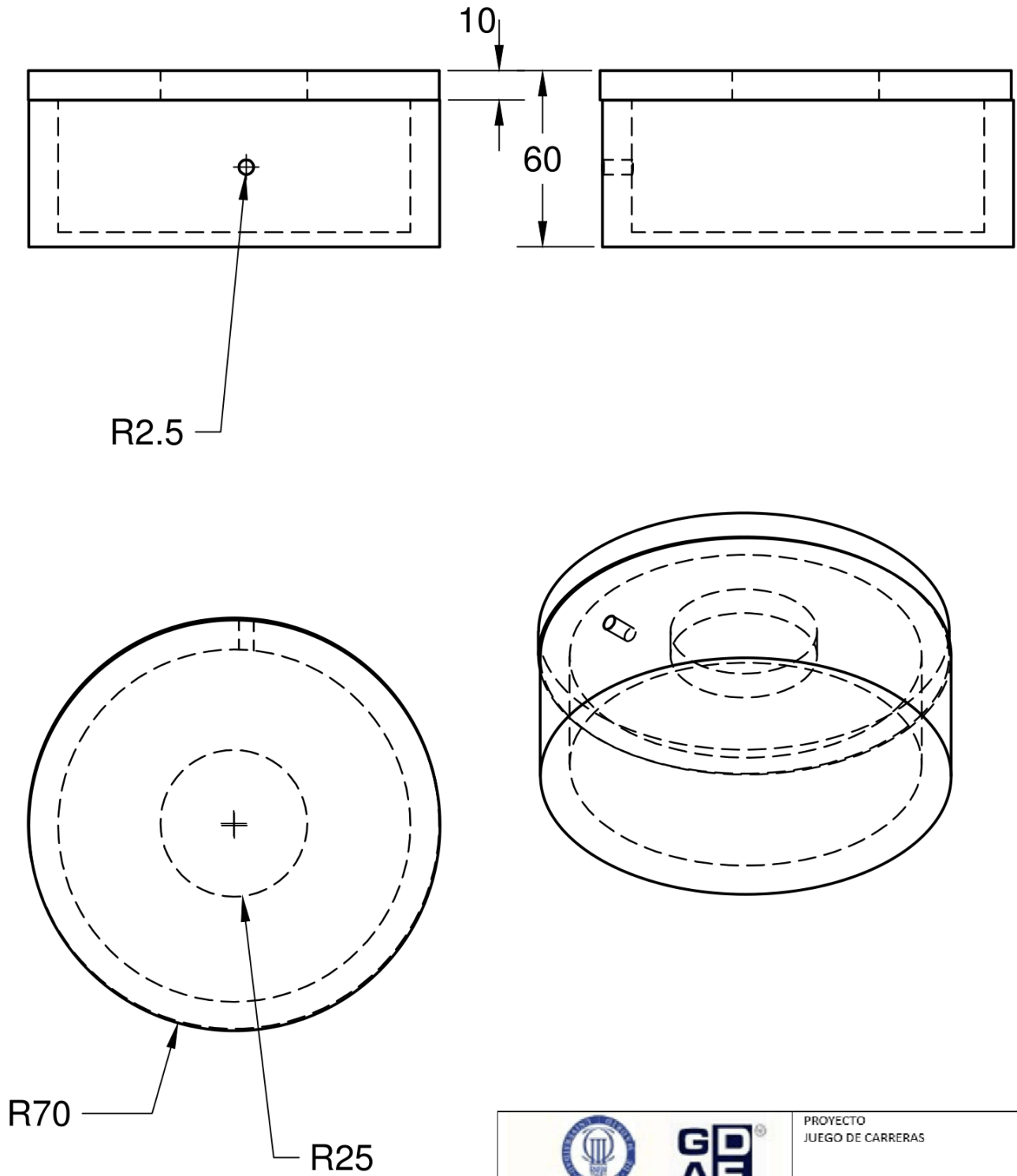
void lcd_puts(const char * string)
{
    while (*string != '\0')
        lcd_write(*string++);
}

void lcd_create_char(uint8_t charnum, const uint8_t * chardata)
{
    uint8_t i;
    charnum &= 0x07;
    lcd_command(E_SET_CGRAM_ADDR | (charnum << 3));
    for (i = 0; i < 8; i++)
        lcd_write(chardata[i]);
}
```

RF.c

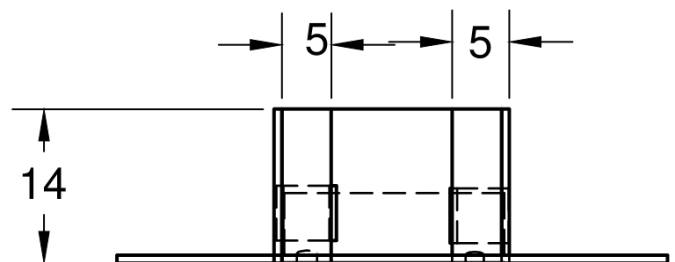
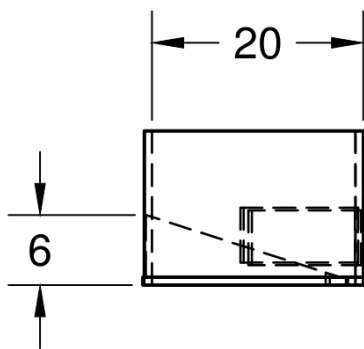
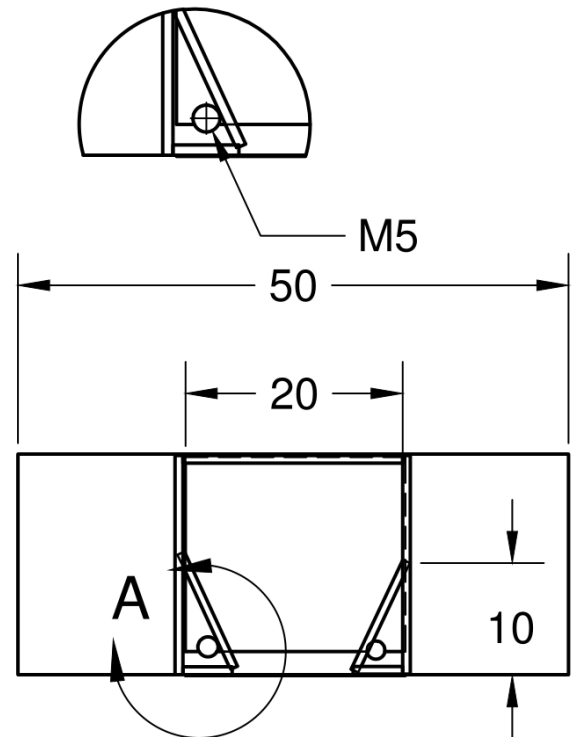
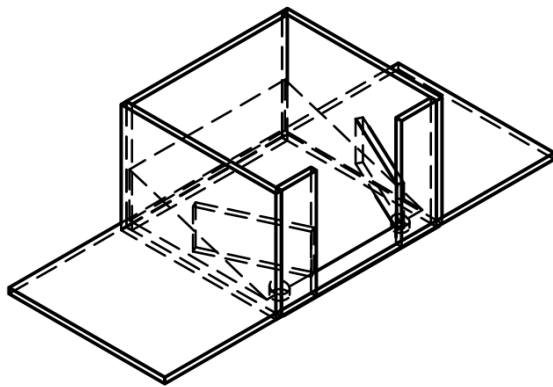
```
/*
 * RF.c
 * Created on: Aug 9, 2016
 * Author: scastillom
 *****/
/* Includes -----*/
#include "RF.h"
/** @RF_Command
 * @brief Función que separa bit a bit el comando que se desea
mandar y se asigna a su output correspondiente.
 * @param RF_Command: Byte que se desea mandar a través de RF
 * @retval none
 */
void RF_Command (uint8_t RF_Command)
{
    int i;
    uint8_t singleBit = 0;
    for(i = 3; 0 <= i; i --) //extrae los 4 bits menos
significativos del Byte
    {
        singleBit = (RF_Command >> i) & 0x01;
        switch (i)
        {
            case 3: //lectura bit mas significativo
                if (singleBit == 0)
                    {HAL_GPIO_WritePin(S3_GPIO_Port, S3_Pin,
GPIO_PIN_RESET);}
                else
                    {HAL_GPIO_WritePin(S3_GPIO_Port, S3_Pin,
GPIO_PIN_SET);}
                break;
            case 2: //lectura segundo bit mas significativo
                if (singleBit == 0)
                    {HAL_GPIO_WritePin(S2_GPIO_Port, S2_Pin,
GPIO_PIN_RESET);}
                else
                    {HAL_GPIO_WritePin(S2_GPIO_Port, S2_Pin,
GPIO_PIN_SET);}
                break;
            case 1: //lectura segundo bit menos significativo
                if (singleBit == 0)
                    {HAL_GPIO_WritePin(S1_GPIO_Port, S1_Pin,
GPIO_PIN_RESET);}
                else
                    {HAL_GPIO_WritePin(S1_GPIO_Port, S1_Pin,
GPIO_PIN_SET);}
                break;
            case 0: //lectura bit menos significativo
                if (singleBit == 0)
                    {HAL_GPIO_WritePin(S0_GPIO_Port, S0_Pin,
GPIO_PIN_RESET);}
                else
                    {HAL_GPIO_WritePin(S0_GPIO_Port, S0_Pin,
GPIO_PIN_SET);}
                break;
            default:
                break;
        }
    }
} //RF_Command
```

ANEXO 4. Planos

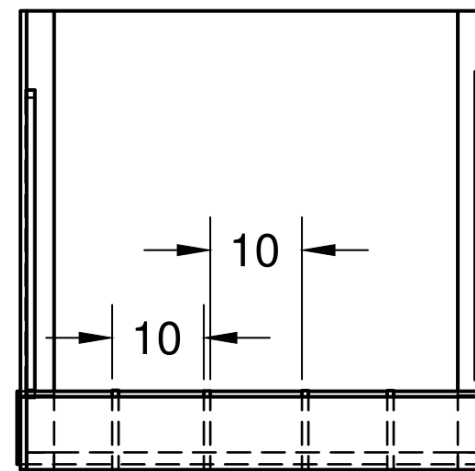
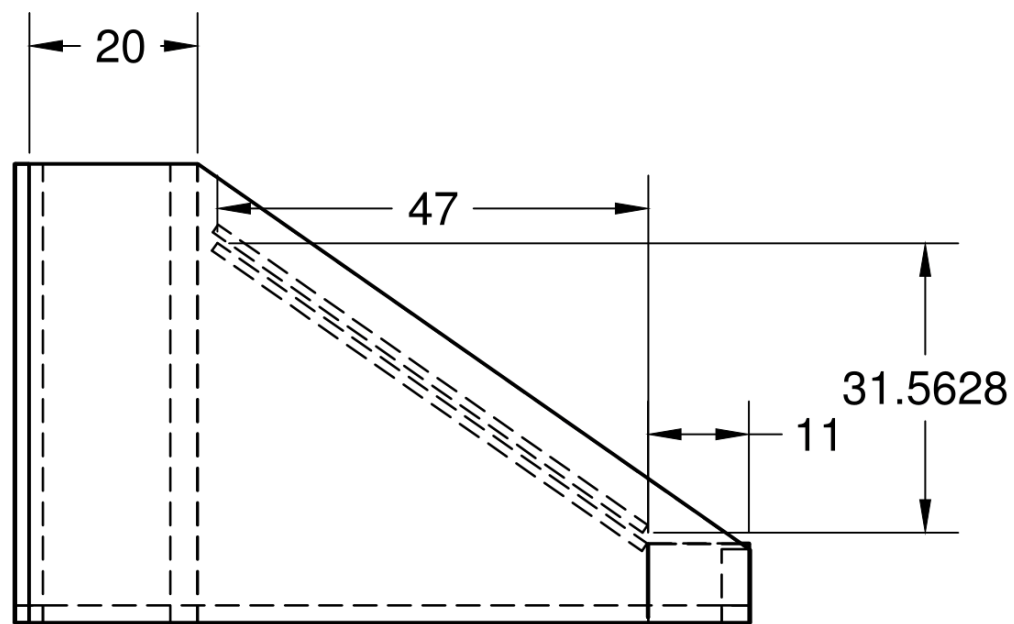
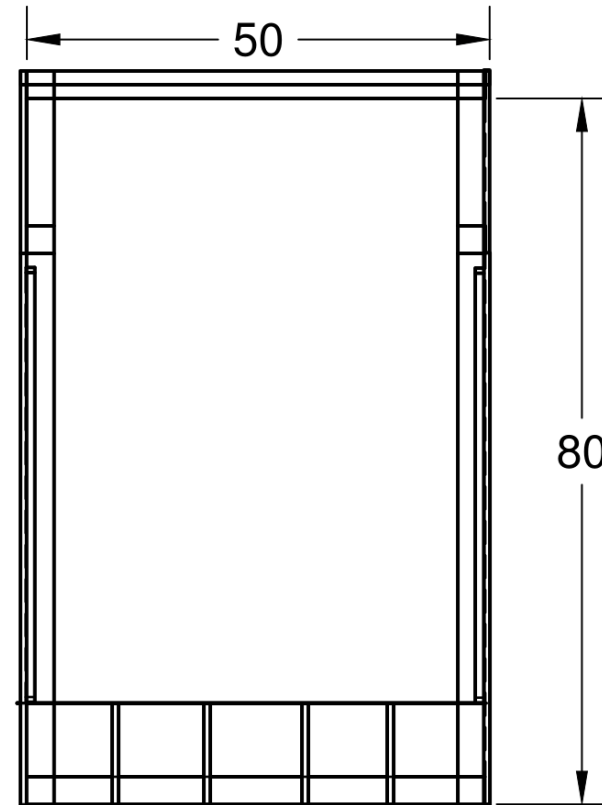
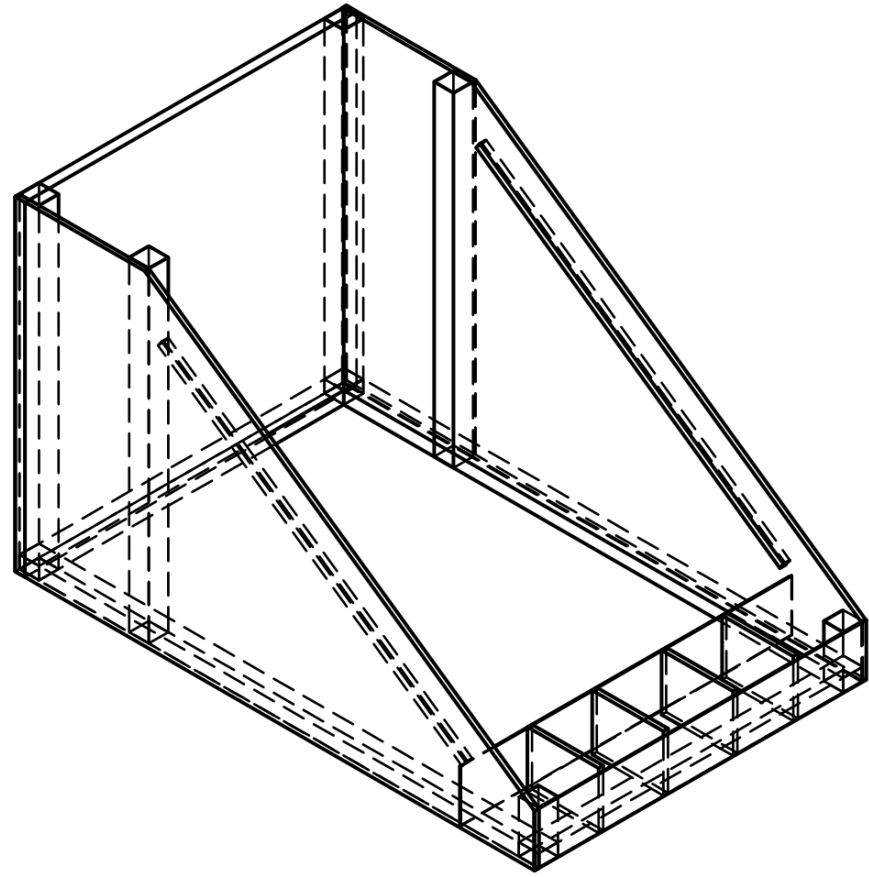


 	PROYECTO JUEGO DE CARRERAS	
Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael(15): Transición a la Vida Adulta-Juego de Carreras	TITULO Pulsador	
Autores: Sergio Aguilera Sergio Castillo	ESCALA 1/4	FORMATO A4

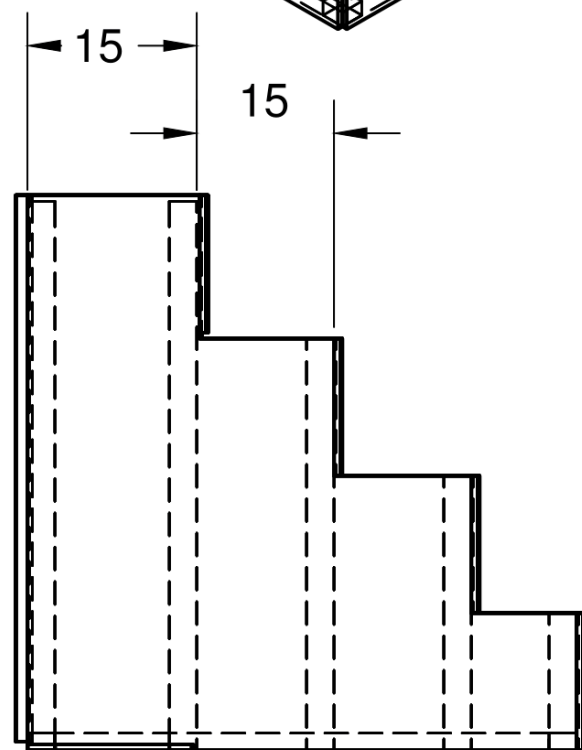
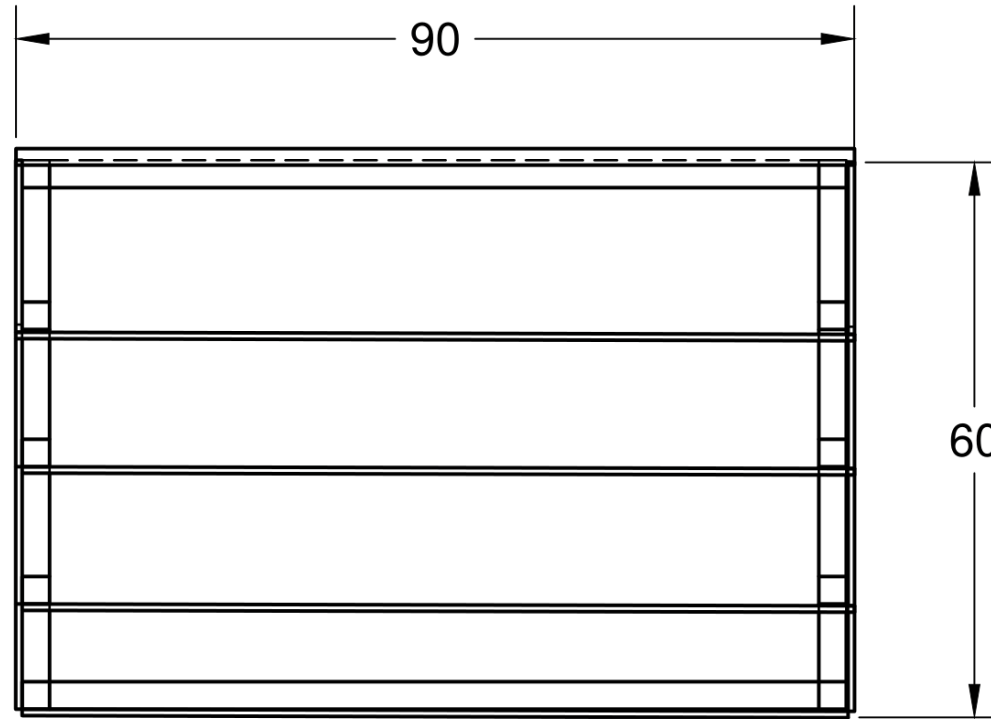
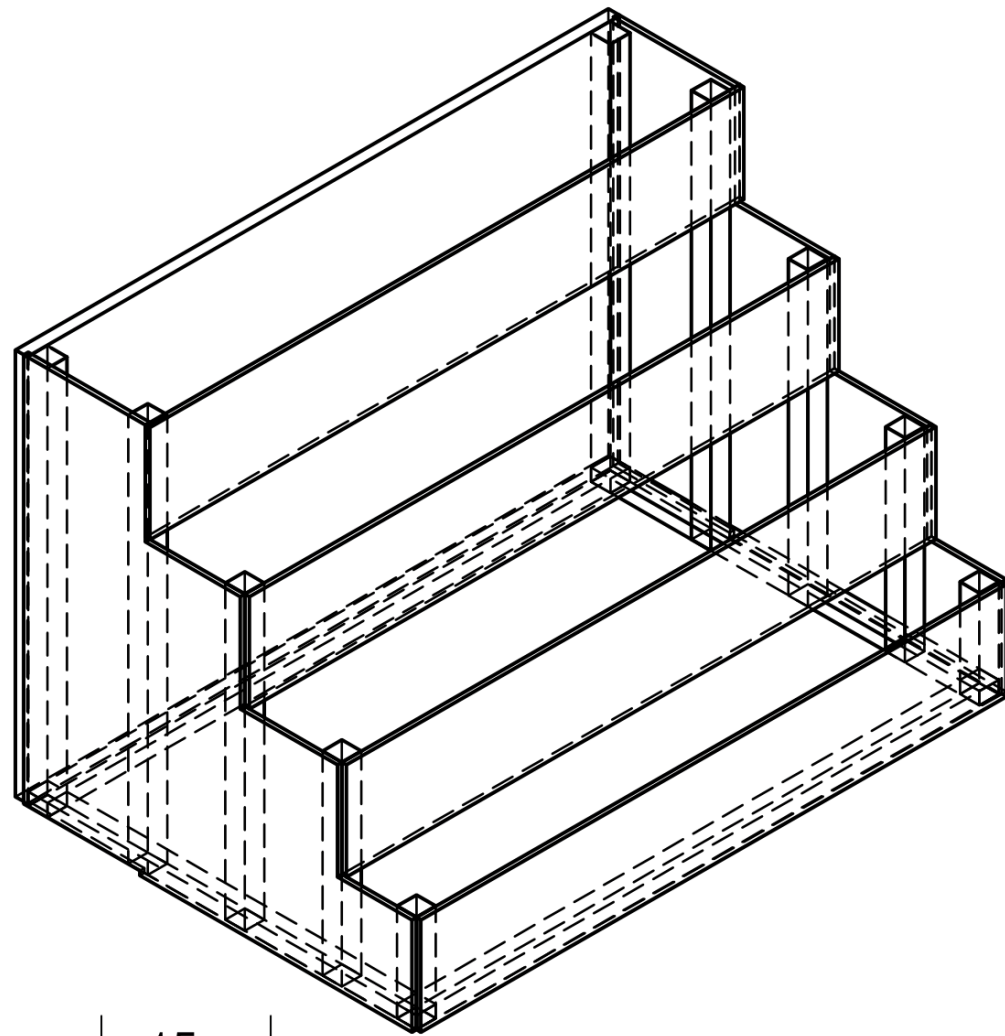
DETALLE A SCALE 1" = 1'-0"




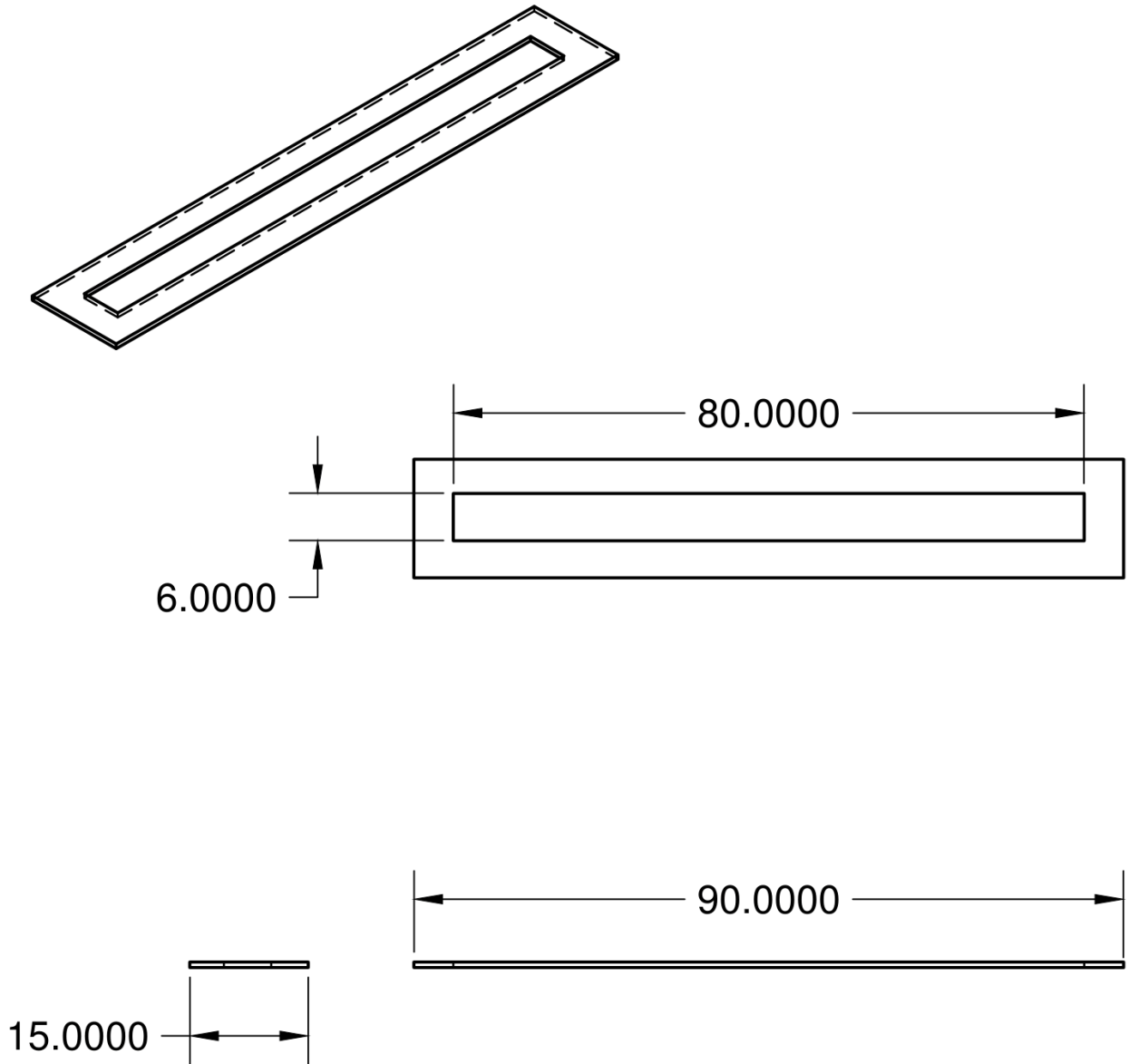
 		PROYECTO JUEGO DE CARRERAS	
Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael(15): Transición a la Vida Adulta-Juego de Carreras		TITULO Caja Pelota	
Autores: Sergio Aguilera Sergio Castillo		ESCALA 1:16	FORMATO A4



 		PROYECTO JUEGO DE CARRERAS	
Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael(15): Transición a la Vida Adulta-Juego de Carreras		TITULO Bloque Rampa	
Autores: Sergio Aguilera Sergio Castillo		ESCALA 1:20	FORMATO A3



 		PROYECTO JUEGO DE CARRERAS	
Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael(15): Transición a la Vida Adulta-Juego de Carreras		TITULO Bloque Carriles	
Autores: Sergio Aguilera Sergio Castillo		ESCALA 1:20	FORMATO A3



 	PROYECTO JUEGO DE CARRERAS	
Universidad Carlos III de Madrid Desarrollo de una ayuda técnica para alumnos del Colegio San Rafael(15): Transición a la Vida Adulta-Juego de Carreras	TITULO Base carril	
Autores: Sergio Aguilera Sergio Castillo	ESCALA 1:20	FORMATO A4