Universidad Zaragoza
1542

Trabajo Fin de Grado

Futbolín electrónico
Electronic foosball

Autor

Rodrigo Pérez García

Director/es

Roberto Casas Nebra
Alberto Mur López

Escuela de Ingeniería y Arquitectura de Zaragoza
Año académico: 2019 - 2020

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe entregarse en la Secretaría de la EINA, dentro del plazo de depósito del TFG/TFM para su evaluación).

D./Dª. Rodrigo Pérez García ,en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza, Declaro que el presente Trabajo de Fin de (Grado/Máster) Ingenieria Electrónica y automática (Título del Trabajo)

Futbolín electrónico.

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 1 de Septiembre de 2020

Fdo: Rodrigo Pérez García

Resumen

En el presente trabajo de fin de grado, se va a hacer uso de circuitería electrónica para modificar un futbolín de madera convencional. Mediante la incorporación de sensores, microcontroladores y de nuevas tecnologías, podremos dotar al futbolín de nuevas funcionalidades añadidas y formas de juego.

El alumno deberá de investigar aquellas mejoras que puedan resultar de interés al jugador de futbolín. Una vez conocidas, se tendrá que familiarizar con los distintos lenguajes de programación y sus entornos de desarrollo, para desarrollar las mejoras de forma eficaz.

En este TFG se programará: un microcontrolador en C++, para implementar toda la lógica de las partidas y rotaciones de los equipos; una Raspberry pi en Python, para incorporar funcionalidades multimedia; y un servidor web en HTML5, para que futbolín y usuario puedan comunicarse.

Se hará uso de la tecnología de impresión 3D para fabricar piezas que permitan integrar la electrónica en el futbolín proporcionado por la empresa aragonesa VAL. Por petición de la empresa, la estética del futbolín debe de ser lo más parecida posible a la original. Además, la jugabilidad deberá de ser mejorada manteniéndose la "esencia" del juego tradicional.

Abstract

The aim of this final degree project is to equip a traditional wooden foosball with new features and functionalities using 3D 3d printed pieces and electronic devices. It will use sensors to measure the speed, a microcontroller to manage all the sensors, and different game modes. It will incorporate multimedia elements managed with Raspberry Pi, and a web applications to interact with the user.

Índice de contenido

Lista de figuras

Lista de ilustraciones

Lista de tablas

1. Introducción

El futbolín tradicional es un juego que tiene una mesa (llamada futbolín) con dos porterías y en el que se enfrentan dos equipos formados por dos jugadores. Cada equipo ha de controlar 4 barras en las cuales se disponen una serie de muñecos. Mediante el movimiento coordinado de las barras se tiene que golpear una bola para introducirla en la portería del equipo rival. Una vez introducida en la portería contraria, se reincorpora otra al terreno de juego para poder realizar la misma mecánica.

Ilustración 1. Futbolín de madera tradicional

La finalización de la partida depende del acuerdo previo que hayan pactado ambos equipos respecto al número de goles que ha de anotarse para acabar la partida. Cuando uno de los equipos llega al número de goles pactado ese equipo se da como ganador de la partida

El conteo de goles de ambos equipos también llamado marcador, lo gestionan los equipos de manera memorística. Siendo este tradicionalmente recordado cada vez que se anota un gol.

En el caso de que exista algún equipo a la espera de jugar, generalmente se aplican las reglas de "rey de la pista". Este modo de juego establece que el equipo ganador permanece jugando en el futbolín, mientras que el equipo perdedor se pone el último en la cola a la espera de volver a jugar. Esta mecánica hace que los equipos que acumulen más victorias tiendan a jugar más partidas que los equipos con tendencia a perder.

1.1 Objetivos

A continuación, se pasará a enumerar los objetivos planteados:

1. Mediante la integración de diversos componentes electrónicos en el futbolín, podemos liberar a los jugadores de la tediosa tarea de gestionar las mecánicas de juego. Así los equipos únicamente se centran en disfrutar de la experiencia de juego y no en tener que recordar cuántos goles han marcado cada uno, o de conocer qué equipo es el siguiente en jugar. Por ello, el objetivo principal del trabajo de fin de grado es innovar en el concepto de futbolín mediante el uso de elementos electrónicos.

2. Mediante los elementos electrónicos y la programación de un microcontrolador se busca poder disputar modos de juego alternativos al tradicional, como un sistema de liga en el que todos los equipos juegan el mismo número de partidas o un torneo eliminatorio.

3. Se va a realizar un prototipo físico y funcional, que posteriormente será comercializado. Se requiere que el prototipo satisfaga las necesidades no sólo del usuario sino también las del fabricante. Por ello, otro objetivo principal es el de reducir los costes de fabricación, materiales y montaje para hacer del futbolín un prototipo competente en el mercado.

4. Puesto que se va a comercializar el futbolín, es necesario conocer las necesidades del usuario para implementar las mejoras más demandadas. Por ello, se deberá hacer un estudio de mercado para saber cuáles son dichas mejoras.

5. Como último objetivo, se espera que este trabajo de fin de grado sea el inicio y esqueleto de posibles futuros trabajos de fin de grado o máster. Por ello se tratará de realizar un diseño modular lo más flexible posible.

1.2 Planificación y fases

Para cumplir con los objetivos planteados, se van a planificar las siguientes etapas:

- **Análisis del estado del arte**: En esta fase se contextualiza y se revisa todo lo relacionado con el futbolín tradicional. Además, también se realiza un estudio de la competencia o de posibles prototipos relacionados.

- **Estudio de mercado**: Posteriormente a la fase anterior, se necesita saber que mejoras tendrán buena recepción en los jugadores de futbolín. Seguidamente, se analizarán aquellas que son interesantes de implementar y que aportan un valor añadido a la jugabilidad y experiencia de juego.

- **Análisis de alternativas**: Conocida la mejora a añadir, se debe de analizar todas las opciones posibles y seleccionar la más adecuada para incorporar.

- **Fase de diseño**: En esta fase se programa el firmware necesario para controlar a los distintos elementos electrónicos del futbolín. También, se programa todo lo relacionado con mejoras para la gestión de partidas.

- **Implementación y prototipado**: Acabada la fase de diseño, queda incorporar todas las mejoras en el futbolín físico. Adicionalmente en este apartado se añade el diseño y fabricación de piezas 3D.

- **Verificación de funcionamiento y optimización**: Finalmente, se debe de comprobar que el prototipo real funciona como corresponde y, por último, realizar funciones de calibración y puesta a punto.

1.3 Herramientas empleadas

Para la elaboración de este trabajo, se van a utilizar las siguientes herramientas:

- **Microsoft Word**, **Excel** y **Visio** como herramientas ofimáticas.
- **Google forms** para la creación y administración de encuestas
- **IDE de Arduino** para la programación en C++.
- **Dreamviewer** para la programación de HTML5.
- **Raspbian** como sistema operativo para Raspberry Pi.
- **Pycharm** y **Thonny** para la programación en Python.
- **CircuitMaker** para diseño electrónico de la PCB.
- **Thingsboard** como plataforma IoT para la publicación de datos.
- **Fusion 360**, **Meshmixer** y **Ultimaker Cura** para modelado e impresión 3D.

1.4 Motivación

La motivación principal de este proyecto es mejorar la experiencia de juego del jugador. Con el fin de aumentar los beneficios producidos por el futbolín, o en el caso de que el propietario sea particular, en un mayor uso del futbolín.

Ser pionero en esta tecnología aporta una imagen de empresa referente y es sinónimo de innovación y de calidad. Además, dotar al futbolín con elementos electrónicos hace que éste tenga un carácter diferenciador alejándolo de la competencia y creando un nuevo nicho de mercado.

Como motivación personal, me atrae mucho poder materializar en un prototipo físico funcional las ideas de mejora que den como resultado una mejor experiencia de juego. Además, me motiva bastante aprender tanto los distintos lenguajes de programación, como todas las demás herramientas a utilizar.

2. Estudio de la cuestión y definición de especificaciones

En este apartado, se estudiará el futbolín tradicional y aquellas patentes relacionadas con el presente trabajo de fin de grado. Asimismo, también se analizarán las necesidades de usuario y fabricante para decidir cuáles son las mejoras más convenientes a implementar.

2.1 Estado del arte

La primera patente de futbolín tiene su origen en España a finales del siglo XX donde se creó un modelo de tamaño reducido y con los jugadores con las piernas unidas.

Posteriormente se inventa otro tipo de futbolín, con un diseño mucho más parecido al actual. El inventor de este nuevo futbolín es Alejandro Finisterre (poeta, inventor y editor). Como se relata en la bibliografía de [1], Alejandro fue herido en los bombardeos de la guerra civil y en el hospital vio que allí había muchos niños que no podían jugar al futbol. Fue entonces cuando se le ocurrió la idea del futbolín. Contactó con su amigo carpintero Francisco Javier Altuna y se pusieron a materializar la idea. La invención la patentó en 1937, pero debido al triunfo franquista tuvo que exiliarse en Francia. En una tormenta Alejandro perdió los papeles de la patente, por ello no hay forma de saber cómo era el diseño original ni sus medidas.

Ilustración 2. Alejandro Finisterre y el primer futbolín. Recuperado de https://www.elespanol.com/quincemil/articulos/cultura/alejandro-finisterre-la-increible-vida-del-gallego-que-invento-el-futbolin

En la actualidad existen multitud de variantes de futbolines, como puede ser el futbolín español, el bonzini de Francia, el fireball de China, el Roberto-sport de Italia, el Garlando también de Italia o el Leonhart de Alemania.

Pese al gran avance de las nuevas tecnologías, el futbolín tradicional ha permanecido al margen de ellas. La implantación de luz bien en los laterales o bien mediante un travesaño ha sido la única modificación relevante.

Como futbolín tradicional adaptado con elementos electrónicos, sólo destacan pequeños proyectos caseros donde se implementan sonido ambiente y juegos luminosos cuando se marca gol y un proyecto realizado por un grupo universitario de la Universidad Politécnica de Madrid llamado B105 y detallado en [2].

El proyecto que ellos desarrollan es un proyecto similar. Introducen una plataforma de desarrollo como Raspberry Pi 2b que les permite tener una interfaz gráfica muy potente para poder seleccionar un icono de jugador y un escudo de equipo.

Los jugadores se registran en el futbolín con un lector de huellas automático. Estos eligen su avatar y equipo mediante una pantalla táctil y ya estarían listos para jugar.

La velocidad de la bola es medida por sensores infrarrojos y en la interfaz gráfica aparece la velocidad y el jugador que ha anotado el gol. Las jugadas son grabadas por una cámara y las repeticiones de gol son reproducidas en una pantalla.

También incluye funcionalidades adicionales como la notificación automática de resultados en Twitter o la iluminación de la portería cuando se marca un gol.

El carácter y la finalidad del proyecto comentado anteriormente es: la aplicación de la electrónica y la motivación de los estudiantes. Ese futbolín no está especialmente pensado para su comercialización sino para el ejercicio de la electrónica y para disfrutar de la funcionalidad que este aporta.

El futbolín anteriormente comentado ha sido sujeto de pequeños proyectos e incluso de un trabajo de fin de grado para la monitorización inalámbrica de sensores.

2.2 Análisis de mercado y posibles mejoras

Antes de implementar cualquier mejora al futbolín es necesario conocer aquello que más le puede interesar al usuario. Las posibles mejoras han de evaluarse dependiendo de varios factores fundamentales como pueden ser:

- **Nivel de recepción del jugador:** Un buen nivel de aceptación resultaría aquel cuya implementación tuviera un impacto positivo en el jugador. Un bajo nivel de recepción sería aquella mejora cuya implementación dejara indiferente al jugador.

- **Coste de implementación:** Poder poner un robot que permitiera jugar partidas en solitario resultaría una gran mejora, pero por otra parte sería algo no realista que supondría encarecer los costes de producción. Por ese motivo, tienen que incorporarse mejoras que generen un impacto positivo al jugador con el menor coste posible.

- **Capacidad de mejora:** Resulta muy interesante realizar mejoras que puedan desarrollarse y mejorarse en un futuro. Emplear herramientas que permitan el desarrollo de nuevas ideas evita que el producto se quede obsoleto. Un ejemplo de ello sería implementar una mejora que permitiera comunicar futbolín y usuario por medio de tecnología móvil. El índice de mejoras en este caso resultaría mucho más favorable que realizar una comunicación entre usuario futbolín por medio de botones.

- **Facilidad de montaje:** Un requisito obligatorio de cualquier montaje es la implementación rápida y fácil. Lo contrario supondría tiempo y/o especialización, y en ambos casos se daría un aumento no deseado en el coste de producción.

- **Diseño modular:** Cada comprador de futbolines tiene distinto presupuesto, gusto e interés. Por lo tanto, es buena práctica dar la posibilidad al comprador de poder seleccionar entre todas las mejoras posibles aquellas que más se adecuen a sus gustos e intereses.

2.2.1 Ideas de mejora

Para saber qué implementar se realiza una tormenta de ideas. Posteriormente, se realizó una encuesta a los usuarios de futbolín sobre qué posibles mejoras les resultaban atractivas. Se obtuvieron 259 respuestas donde los resultados se pueden consultar en el apartado 1 de anexos.

a) Incorporación de luz.
b) Sistemas de pago alternativos como uso de monederos electrónicos.
c) Posibilidad de pago con tarjeta de socio.
d) Posibilidad de pago con diferentes tipos de moneda.
e) Pago mediante aplicaciones bancarias o tarjetas de crédito.
f) Incorporación de un marcador que registre los goles automáticamente.
g) Medición de la velocidad de la bola al anotar un gol.
h) Implementar nuevos modos de juego para poder realizar otro tipo de mecánicas de juego como torneos o ligas.
i) Juego por tiempo mediante una cuenta atrás que, una vez finalizada, indique el resultado de la partida.
j) Reproducción de efectos sonoros en ocasiones especiales, como goles y finales de partida. Ejemplo: sonido de la grada celebrando un gol.
k) Obtención de estadísticas de juego como goles, tiros a puerta, tiros fuera, posesión, etc.
l) Visionado de repeticiones de gol.
m) Detección automática de jugadas no permitidas como ruletas o jugadas.
n) Diseño de una aplicación móvil que permita tener un perfil de jugador para poder consultar sus estadísticas y almacenar sus créditos.
o) Acceder mediante código QR a una interfaz web para poder seleccionar los parámetros de juego.
p) Posibilidad de geolocalización de un futbolín
q) Selección del número de bolas que se desean jugar.
r) Notificación al propietario del futbolín las ganancias producidas y la posibilidad de notificarle de la necesidad de vaciar el cajón de monedas.
s) Disputa de partidos profesionales de futbolín y actualización de los resultados de partida de una manera cómoda.

2.2.2 Análisis cualitativo de mejoras

A continuación, se pasa a analizar las mejoras teniendo en cuenta los factores anteriormente comentados, y clasificando las mejoras dependiendo si son aceptadas o clasificadas como posible mejora para implementar en un futuro.

Un punto a destacar es que las mejoras no son independientes unas de otras; existen mejoras que no pueden coexistir y otras cuya implementación de una favorece la existencia de otra. Por ejemplo, el visionado de las repeticiones de gol favorece la detección automática de jugadas no permitidas.

Mejoras aceptadas

La mejora de añadir luz al terreno de juego (a) fue, por su simplicidad y necesidad, la primera en ser aceptada.

Existen varias propuestas para mejorar el sistema de pago. Pago mediante tarjeta socio, de crédito, mediante monederos que admitan diferentes tipos de moneda, etc. Entre ellas se aceptó la mejora de pago con tarjeta de socio (c), ya que existen sensores NFC que tienen un precio reducido y que permiten realizar a la perfección esa función.

La opción de incorporar un marcador (f) aporta un gran valor a un precio reducido. Además, también con ella se aceptó la opción de medir la velocidad de la bola al marcar gol (g) ya que la detección de esa velocidad supone comprobar el momento cuando ha habido un gol, y por ello poder actualizar el marcador.

Las opciones (h), (i), (k), (p) y (o) también fueron aceptadas, ya que el coste de implementación de estas mejoras es bajo debido a que se realizan enteramente por software. Por ello, tanto la realización de modos de juego con rotaciones a tiempo, como el almacenamiento las estadísticas de la partida o de geolocalización, únicamente supone tener un programa cargado en un microcontrolador que se encargue de toda la lógica requerida.

Por lo que respecta al visionado de las repeticiones de gol (l), se trata de una mejora que aporta un alto valor en la jugabilidad y que permite mejoras futuras relacionadas con procesamiento de imagen como podría ser la detección de la bola o de jugadas no permitidas.

Mejoras para implementar en un futuro

Entre las mejoras para añadir en un futuro entran las que suponen modificar el monedero tradicional, como pueden ser la incorporación de un monedero electrónico (b) que permita pagar con diferentes tipos de moneda (d), y poder seleccionar el número de bolas (q).

Por razones de tiempo y complejidad existen mejoras como el pago con tarjeta bancaria (e) que para el procesamiento de pagos conlleva bastante trabajo que se escapan de los objetivos del mismo. Por ese motivo también se incluye realizar efectos sonoros (j), detectar jugadas no permitidas (m), disponer una aplicación móvil (n), notificar al propietario de las ganancias producidas (r) y disputar partidos profesionales de futbolín de manera cómoda (s).

Como resumen podemos agrupar las mejoras en 2 grupos:

Mejoras aceptadas	Mejoras pospuestas
(a) Incorporación de luz (c) pago con tarjeta socio (f) marcador de goles (g) medición de velocidad de bola (h) nuevos modos de juego (i) Jugar partidas por tiempo (k) Obtención de estadísticas de juego (l) Visionado de repeticiones de gol (o) Interfaz gráfica con acceso mediante QR (p) geolocalización de futbolín	(b) sistemas de pago alternativos como uso de monederos electrónicos (d) pago con diferentes monedas (e) pago con aplicaciones bancarias (j) Reproducción de efectos sonoros (m) Detección automática de jugadas no permitidas (n) diseño de una aplicación móvil (q) Selección del número de bolas (r) Notificaciones al propietario (s) Gestión automática de partidos profesionales

Tabla 1. Tabla de mejoras aceptadas/pospuestas

3 El futbolín y sus bloques principales

A continuación, dos diagramas de bloques donde se observan las partes principales del futbolín que posteriormente serán analizadas en detalle.

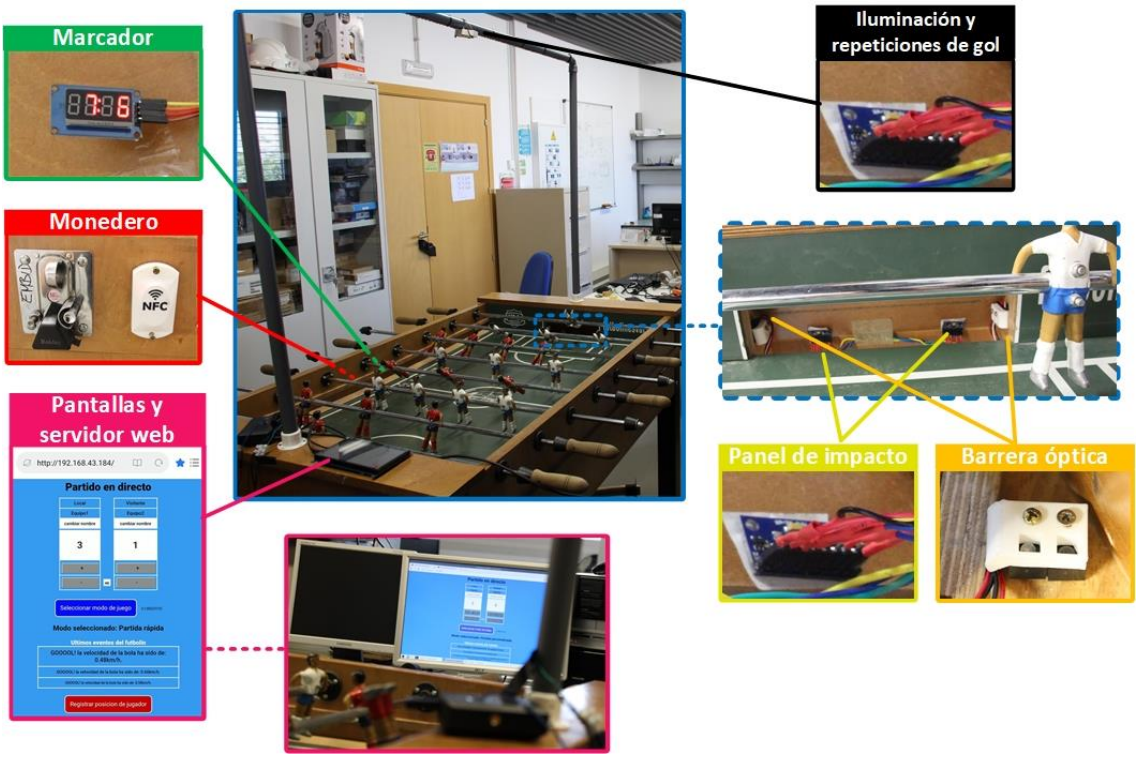


Fig. 1. Diagrama de bloques principales (parte visible).

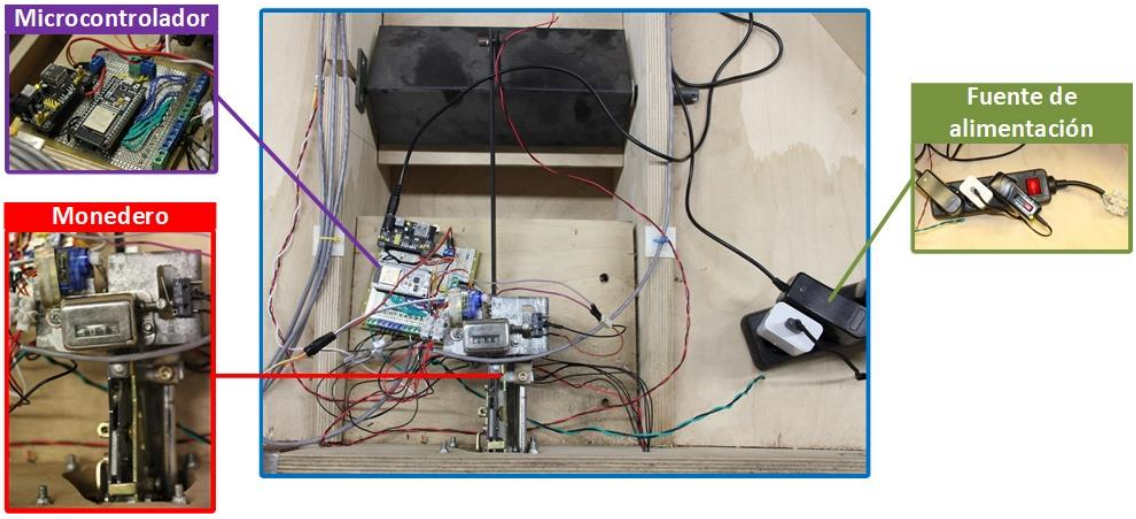


Fig. 2. Diagrama de bloques principales (parte interior).

Para ver más fotos del prototipo físico, revisar apartado 7 de anexos.

3.1 Microcontrolador

El microcontrolador es el elemento principal para la gestión de la mecánica de juego del futbolín. Es el circuito integrado programable capaz de ejecutar las órdenes que hayan sido grabadas en su memoria. En otras palabras, es el microcontrolador el que te va a decir cuántos goles ha marcado cada equipo, a quién le toca jugar o incluso cuántos goles o tiempo faltan para finalizar la partida. Además, también va a ser el encargado de incrementar el marcador cuando un jugador lo pide o de pausar una partida si alguno de los jugadores se tiene que ausentar.

El microcontrolador utilizado es el ESP32 diseñado por la compañía china Espressif. Dispone de unas características de funcionalidad, coste y potencia que hacen que sea un candidato ideal para las tareas requeridas. Tiene memoria suficiente, un bajo coste (de 2 a 5€), módulos wifi y bluetooth y puede ser programado en el entorno de programación (IDE) de Arduino.

Para más detalles relacionados con las características del microcontrolador consultar anexos apartado 2.

El módulo empleado para programar el microcontrolador es el DevKit ESP32-WROOM 32. Para más detalles consultar [3].

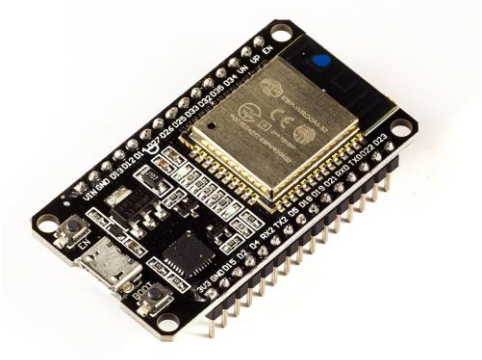


Ilustración 3. DevKit ESP32-WROOM 32. (imagen sin copyright)

Como ya hemos comentado, el microcontrolador es el cerebro del futbolín electrónico, es aquel que se encarga de ejecutar todas las órdenes que han sido programadas. Por ello en él han de conectarse todos los sensores y actuadores que aportan al microcontrolador datos de entrada y salida respectivamente. Para facilitar el conexionado de los distintos elementos electrónicos, se diseñó una placa de circuito impreso PCB mediante el software de diseño electrónico de Circuit Maker.

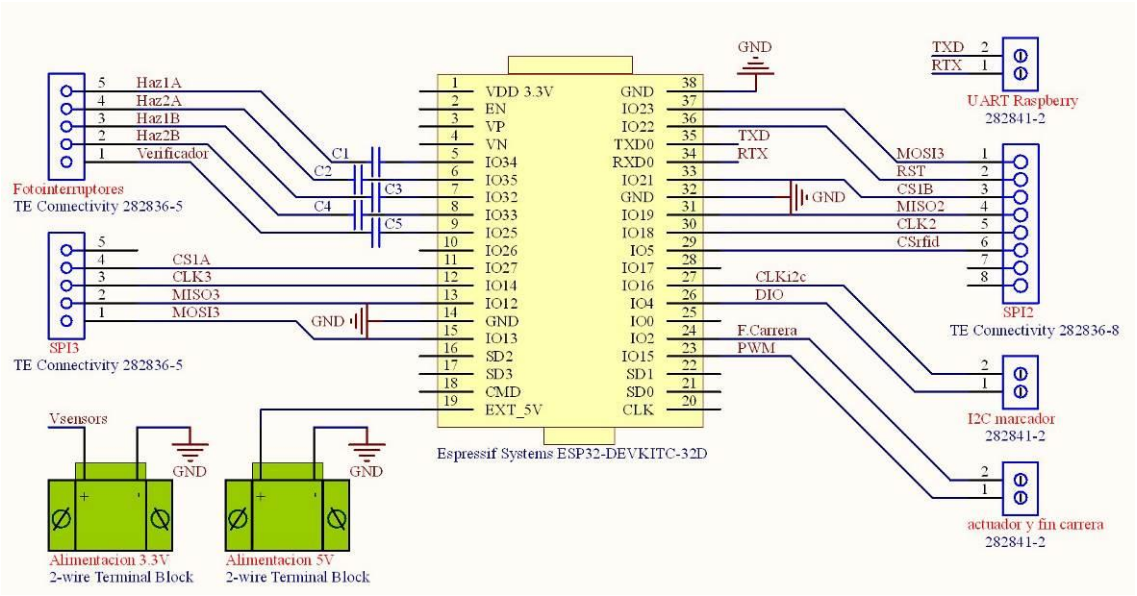


Fig.3. Esquemático de PCB

A partir del esquemático diseñado y una placa de baquelita de cobre perforada se sueldan y cablean los correspondientes componentes; de esta manera se obtiene una PCB con carácter provisional, pero plenamente funcional.

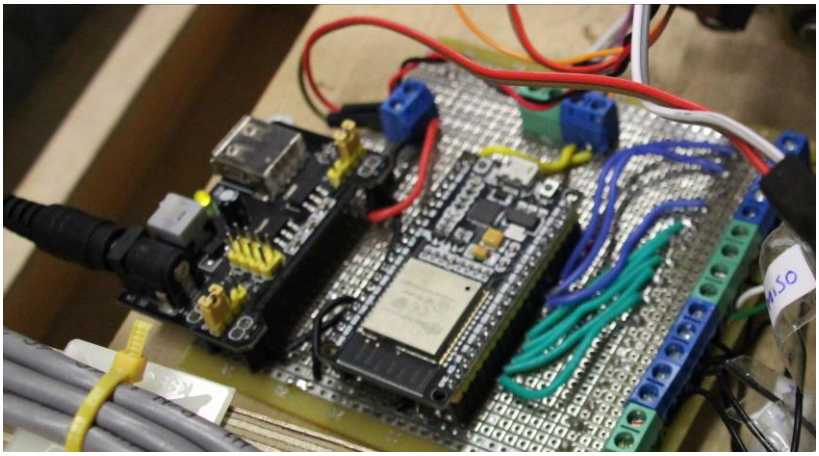


Ilustración 4. PCB futbolín

Como breve explicación de la PCB fabricada, se ha de resaltar existen dos circuitos de alimentación de 5V y 3.3V para alimentar el microcontrolador y los sensores respectivamente.

3.2 Marcador

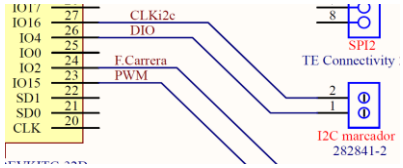


Fig. 4. Parte de esquemático del marcador.

Como hemos comentado en el apartado anterior, el microcontrolador es el encargado de gestionar la mecánica de juego y dentro de esta se incluye el conteo de goles de ambos equipos o también llamado marcador.

La información correspondiente a los goles de ambos equipos se visualiza por dos vías: por un display de 7 segmentos y por un servidor web que se accede a él al escanear un código QR.

El display de 7 segmentos dispone de cuatro dígitos para poder representar hasta un máximo de 99 goles por cada equipo. Como tipología de bus utiliza tecnología I2C. (Para más información sobre tipologías de bus y protocolos de comunicación revisar apartado 3 de anexos).

Ilustración 5. Marcador, códigos QR y monedero.

En la ilustración 5 se observa a la izquierda el display de 7 segmentos, en él se muestran los goles marcados por el equipo local y visitante respectivamente. El equipo local será aquel que se sitúe en la parte del futbolín donde se encuentra el cajón de bolas.

Para seleccionar la ubicación del display del marcador se seleccionó la de la imagen por motivos de proximidad a sus respectivas conexiones. Se ha de aclarar que se trata de una ubicación temporal. Una versión más avanzada del futbolín deberá situar el display en una zona visible para todos los jugadores.

3.3 Barrera óptica

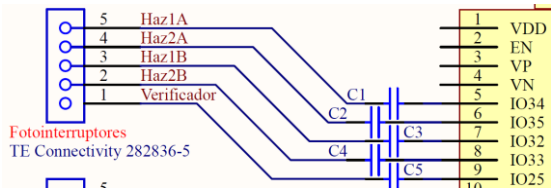


Fig. 5. Parte de esquemático de la barrera óptica.

En el apartado 4 de anexos, se estudia las diferentes alternativas para el conteo de goles. Además de detectar el paso de la bola (indicativo de gol), resulta interesante medir la velocidad con la que la bola entra a la portería. Por ello, en el apartado 5 de anexos se ponen en cuestión diferentes alternativas para el sensado de velocidad de la bola.

Tras analizar los apartados comentados anteriormente, el sistema que por sus características resulta más atractivo es el de barrera óptica. Este sistema se encarga del sensado de velocidad, y por tanto, también del de detección de goles.

Ilustración 6. Vista inferior de barrera óptica.

Para medir la velocidad de la bola se hace uso de dos fotointerruptores (barrera óptica). Cada fotointerruptor consta de un emisor de luz infrarroja y un receptor que la detecta. Cuando un objeto irrumpe el haz generado entre emisor y receptor, a este último le deja de llegar la luz infrarroja. Cuando se irrumpe el haz entre el cable de señal y de masa (cable blanco y negro respectivamente) deja de haber 3.3V de tensión y pasa a ser 0V.

Los fotointerruptores elegidos son detectores de haz infrarrojo DC5V RAD100CM que se detallan en [4]. Se escogieron por su bajo coste (1.5€), su rapidez de respuesta y por ser sensores discretos (2 valores lógicos), capaces de generar una interrupción por flanco de bajada.

Los dos fotointerruptores se montan enfrentados para evitar que el emisor de uno pueda interferir en el receptor del otro (en una pared va emisor y receptor).

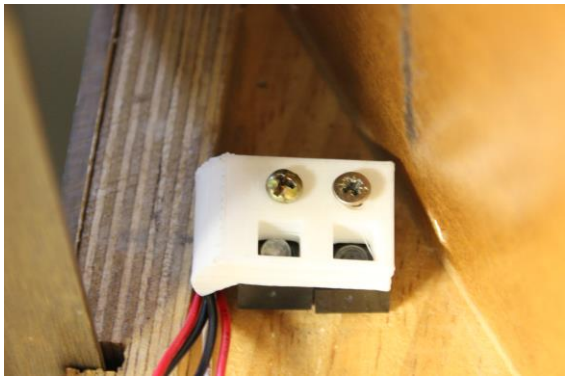


Ilustración 7. Emisor y receptor con pieza de protección

Para evitar posibles daños en los sensores se utilizan dos piezas impresas en 3D, como la de la imagen anterior. La vista isométrica del diseño es la siguiente:

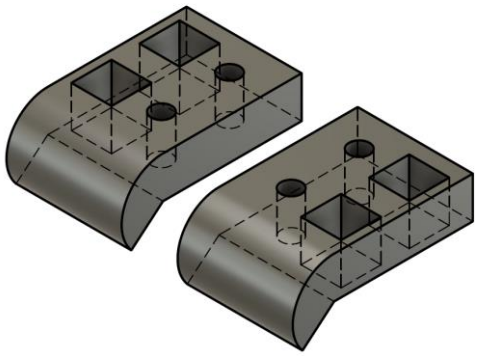


Fig. 6. Diseño 3D de protectores de sensores de la barrera óptica.

Para ver la pieza de la figura 6 en detalle con sus medidas, revisar apartado 10 de anexos.

Si se quiere medir la velocidad de la bola, se ha de cronometrar el tiempo transcurrido desde que el primer fotointerruptor detecta la presencia de la bola (0V entre cable de señal y de masa) hasta que el segundo fotointerruptor emita la misma señal. Conocido el tiempo entre señales, la distancia entre fotointerruptores y sabiendo que la velocidad es el cociente entre el espacio recorrido y el tiempo, podemos calcular fácilmente la velocidad.

Adicionalmente a las barreras ópticas de cada portería, ha de existir un sistema que evite falsos goles que podrían darse al meter la mano en cualquiera de las dos porterías. Ese sistema, el verificador, es un interruptor fotoeléctrico ubicado dentro del conducto por donde pasa la bola. Una ilustración del verificador es la siguiente:

Ilustración 8. Verificador

El verificador es el encargado de confirmar que ha habido gol y por ello el marcador debe incrementarse.

El conjunto de las dos barreras ópticas y el verificador suponen cinco cables de señal que deben conectarse al microcontrolador para que se encargue de la algoritmia encargada de calcular la velocidad, incrementar el marcador, etc.

Al igual que para las barreras ópticas, el emisor y receptor del verificador llevan una pieza impresa en 3D para evitar posibles daños.

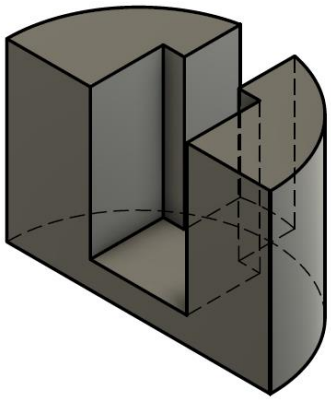


Fig. 7. Diseño 3D de protectores de sensores de verificador.

Para ver la pieza de la figura 7 en detalle con sus medidas, revisar apartado 10 de anexos.

3.4 Pared de impacto

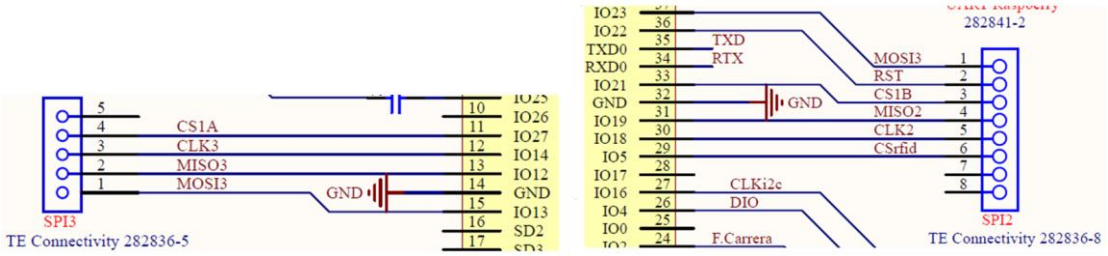


Fig. 8. Parte de esquemático de la pared de impacto.

Cada vez que se marca un gol, la bola impacta contra la madera del fondo de la portería. El impacto de la bola se traduce en vibración capaz de ser medida para estimar la velocidad.

Por otra parte, es lógico pensar que el módulo de la fuerza de impacto variará dependiendo de la zona donde golpee la bola, por ello con los sensores adecuados se podría, no sólo estimar la velocidad con la fuerza de impacto sino también la zona donde golpea.

A partir de estas dos hipótesis nace el concepto de pared de impacto. Con este término nos referimos al conjunto de sensores capaces de detectar la fuerza y la zona de impacto.

Se utilizan los módulos MPU92/65 de nueve ejes, sus especificaciones se detallan en [5]. Estos disponen de acelerómetro magnetómetro y giroscopio. Utilizando los acelerómetros para determinar la gravedad y sus variaciones en los tres ejes X, Y, Z podemos detectar lo anteriormente comentado.

Ilustración 9. Pared de impacto con sus 2 módulos MPU 92/65

Pese a no tener mucha relevancia para el desarrollo de partidas en el futbolín, sí que tiene mucho interés en el ámbito de la investigación. Por ello se realizó un ensayo de tiradas para obtener datos que puedan ser útiles en futuras investigaciones.

Ensayo de sensado de velocidad e impacto con acelerómetros

El ensayo se basa en soltar una bola dejándola caer desde una determinada altura hasta la pared de impacto. Se realizan diez tiradas en cinco posiciones de la pared a tres alturas diferentes. Por cada tirada dos acelerómetros toman 500 muestras de la fuerza X, Y, Z medida. Todos los datos son enviados desde puerto serie hasta el ordenador donde son procesados.

El objetivo del ensayo es analizar las medidas de los acelerómetros para encontrar una correlación entre el valor medido, y la velocidad y ubicación de impacto de la bola. Para ello, se añaden los dos fotointerruptores para sensar la velocidad de la bola y dos acelerómetros:

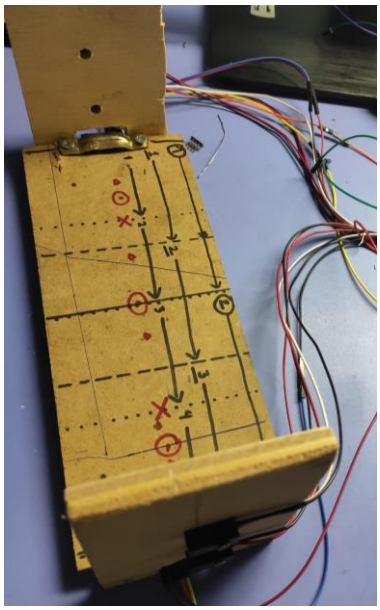


Ilustración 10 Tabla de ensayo de tiradas.

Mediante un programa implementado en lenguaje Python, podemos leer el puerto serie y cargar de manera automática los datos leídos a un documento Excel. En el programa se especifica el número de líneas que se han de leer hasta finalizar el programa y generar el documento correspondiente. Para ello, se pone un número ligeramente superior a las tiradas a realizar, multiplicado por el número de muestras tomadas por cada una de ellas.

Para separar las tiradas válidas de las que no lo son, se hace uso de un pulsador a modo de verificador que se encarga de enviar por el puerto serie la última tirada realizada.

3.5 Monedero modificado

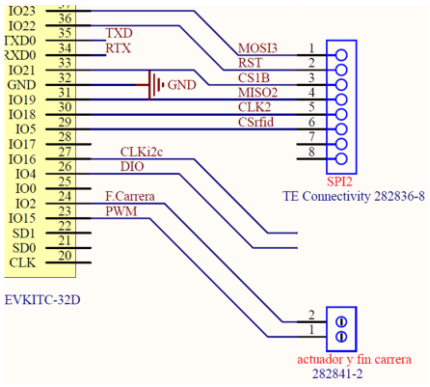


Fig. 9. Parte de esquemático del monedero modificado.

El sistema encargado de la gestión del saldo monetario es lo que se denomina monedero. En el futbolín tradicional, el monedero es totalmente mecánico. La discretización entre una moneda válida y una no válida se realiza enteramente por un sistema con imanes, palancas y piezas metálicas. Por tamaño o por atracción magnética, el monedero expulsa la moneda si es incorrecta o la introduce en un cargador en caso contrario.

Para dispensar las bolas se tiene que accionar un pulsador del monedero que mueve un mecanismo y bloquea el paso de bolas al cajón exterior. El pulsador puede estar bloqueado o no mediante una palanca. Si está bloqueado, se puede desbloquear añadiendo una moneda al cargador. Con la propia curvatura de la moneda, la palanca no se bloquea y permite el paso de bolas al cajón exterior. Una ilustración de un monedero mecánico puede ser la siguiente.

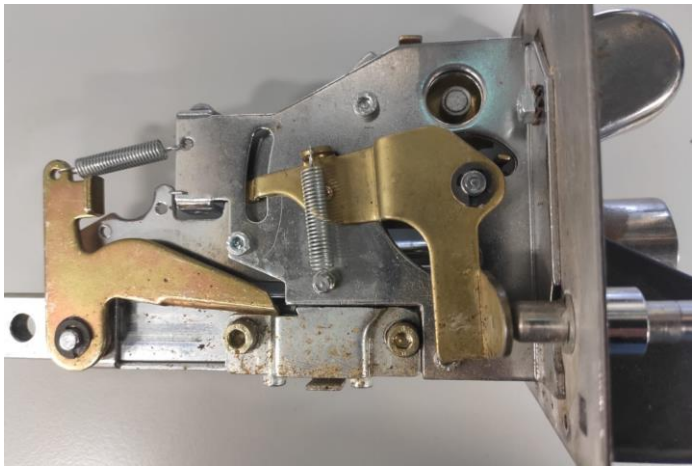


Ilustración 11. Monedero mecánico tradicional

Una de las grandes limitaciones de este sistema mecánico es que únicamente se puede pagar con un tipo de moneda, y ello en ocasiones resulta tedioso para los jugadores. Este problema se podría solucionar con monederos electrónicos multimoneda, que permiten conocer el valor de la moneda introducida y generar una señal electrónica proporcional a ese valor.

En el apartado 6 de anexos, se ha realizado un estudio para poder comparar las propiedades de distintos monederos electrónicos.

Por petición de la empresa interesada, para no añadir un sistema adicional de dispensado y por simplicidad, se decidió mantener el monedero tradicional y modificarlo con distintos elementos electrónicos que puedan añadir otras funcionalidades.

Mediante la incorporación de un final de carrera para conocer cuando se dispensan bolas, y un actuador para levantar la palanca que bloquea que sean dispensadas; podemos controlar el monedero mecánico y el flujo de monedas que han sido introducidas. A continuación una ilustración de lo comentado:

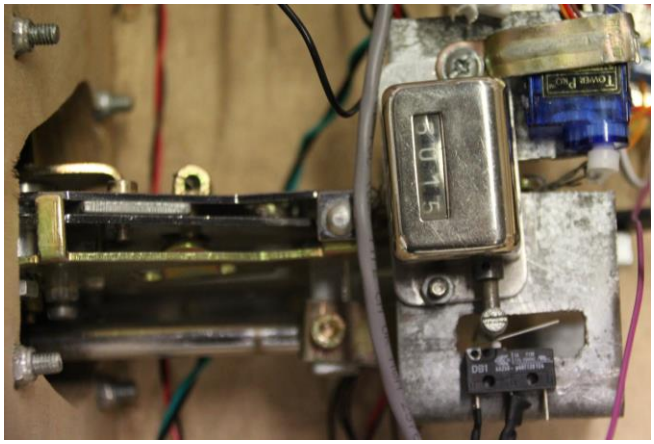


Ilustración 12. Actuador y fin de carrera

El final de carrera únicamente da señal cuando el botón se ha pulsado hasta el fondo, en otras palabras, cuando se realiza el dispensado de bolas.

El actuador es un servomotor SG90 [6] con un cable atado a su extremo, el cable a su vez se une a la palanca que bloquea el botón de dispensado. De esta forma cuando se acciona el actuador, se mueve el extremo del servomotor hacia arriba y como está ligado a un cable, también lo hace la palanca. La única fuerza que tiene que vencer el actuador es la del muelle que hace retornar la palanca a la posición natural.

Adicionalmente se añade un sensor lector de tarjetas RFID-RC522 cuyas características se detallan en [7]. Gracias a éste, se permite dar al jugador sistemas de pagos alternativos (la tipología de bus es SPI). Aproximando la tarjeta en el lector, se habilita el actuador permitiéndose así dispensar un cajón de bolas. Cuando se pulse el botón, el final de carrera ordena al microcontrolador que el actuador se deshabilite.

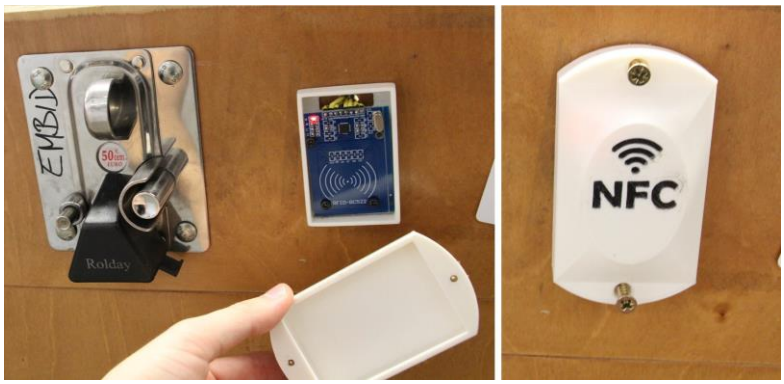


Ilustración 13. Sensor lector de tarjeta sin tapa y con tapa

Como se observa en la ilustración anterior, el lector de tarjetas dispone de una carcasa protectora hecha con impresión 3D.

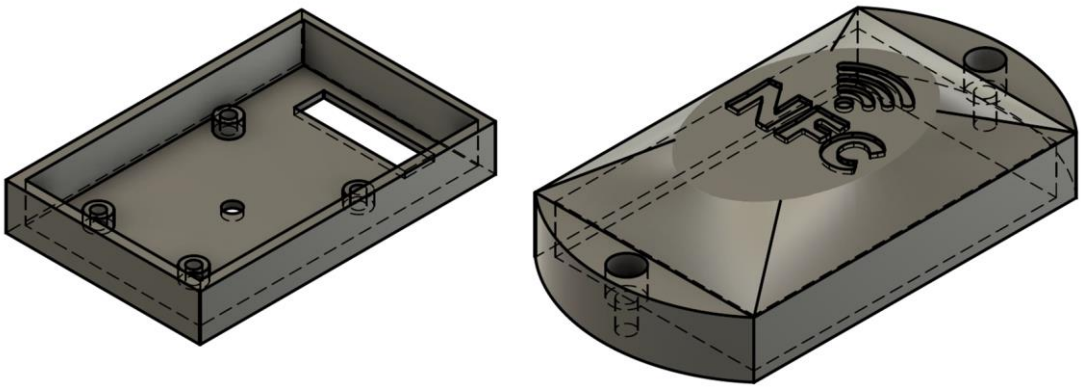


Fig. 10. Diseño 3D de protectores de lector de tarjeta.

Para ver la pieza de la figura anterior en detalle con sus medidas, revisar apartado 10 de anexos.

3.6 Iluminación y repeticiones de gol

Una parte sencilla pero no menos importante es la iluminación. Una mala iluminación puede dificultar el desarrollo del juego. Por ese motivo se incorpora un sistema de iluminación integrado en el futbolín que asegura que las condiciones de juego sean óptimas independientemente de agentes externos.

El sistema de iluminación se basa en tres tubos de PCV dispuestos en forma de arco. Sobre el tubo central se adhiere una tira de leds y en uno de los tubos laterales se integra un interruptor de encendido y apagado.

Ilustración 14. A la izquierda pieza de fijación, en el centro tira de leds, a la derecha interruptor

Como se observa en la imagen anterior, para fijar el tubo del futbolín, se diseñó la siguiente pieza impresa en 3D:

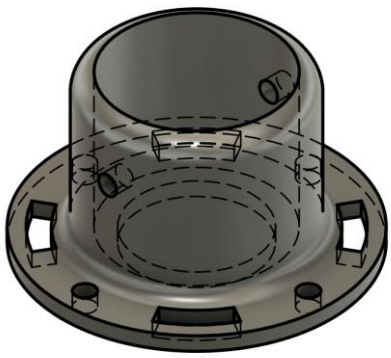


Fig. 11. Diseño 3D de sujeciones de la barra.

En el apartado 2.2.2 se ha comentado el interés y atractivo que supondría incorporar una cámara para poder ver las repeticiones de los goles. Para ello se hace uso de un ordenador de bajo coste como Raspberry pi con una cámara específica. El conexionado entre la cámara y la raspberry es el siguiente:

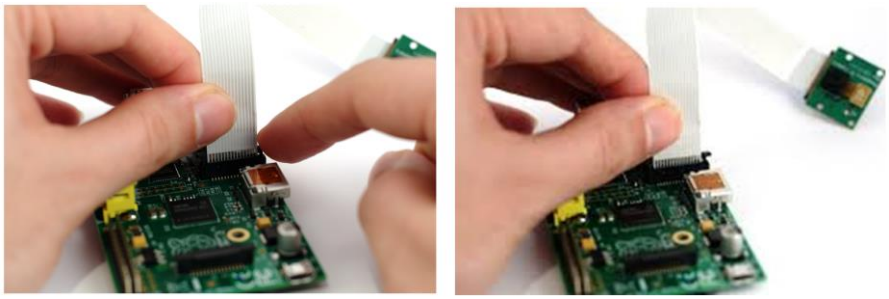


Ilustración 15. Conexionado de Pi Camera a Raspberry (imagenes sin copyright)

Mediante el algoritmo de programación cargado en la Raspberry, la cámara va grabando los 7 últimos segundos transcurridos; cuando se marca gol el clip es guardado y reproducido en una pantalla.

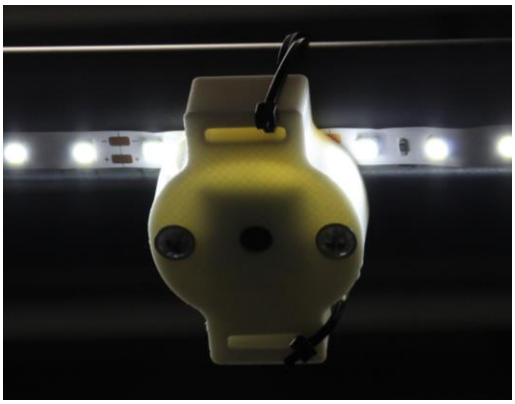


Ilustración 16. Pi camera con carcasa

Como se observa en la ilustración anterior, para proteger la cámara de agentes externos se diseñó una pieza impresa en 3D.

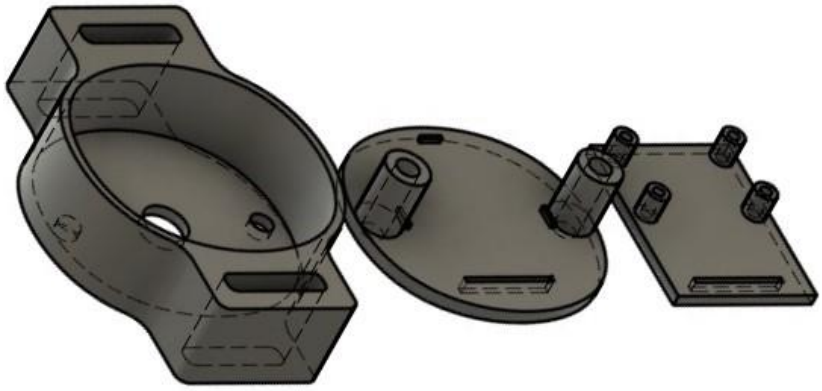


Fig. 12. Diseño 3D de protectores de cámara.

Al igual que para las sujeciones de la barra, las medidas de las piezas se detallan en apartado 10 de anexos.

3.7 Servidor web y pantallas

El servidor web es la interfaz o el medio con el que el jugador se comunica con el microcontrolador. El elemento físico donde se visualizará serán las pantallas asociadas (pantalla táctil, televisión y/o dispositivo móvil). El microcontrolador es el elemento que proporciona el servidor web

La aplicación web del servidor, se desarrolla con la especificación HTML5. Ésta es un lenguaje de marcado que entienden los navegadores para representar el contenido al usuario.

A través de él se pueden realizar principalmente dos acciones; visualizar variables y seleccionar distintas opciones de juego. Como resumen de ambas:

Variables visualizadas	Acciones de juego
- Goles del equipo local - Goles del equipo visitante - Nombre del equipo local - Nombre del equipo visitante - Créditos disponibles - Modo de juego seleccionado - Último evento de futbolín - Penúltimo evento de futbolín - Antepenúltimo evento de futbolín	- Sumar/restar goles al equipo local - Sumar/restar goles al equipo visitante - Renombrar al equipo local - Renombrar al equipo visitante - Seleccionar el modo de juego - Seleccionar el número de equipos - Seleccionar el modo de rotación a goles o tiempo - Registrar a un jugador para actualizar sus estadísticas de juego

Tabla 2. Variables visualizadas y acciones de juego del servidor web

La aplicación web se guarda en el sistema SPIFFS del microcontrolador (sistema de archivos que funcionan en memorias flash conectadas por SPI en dispositivos embebidos). Los ficheros se guardan en una carpeta llamada data; dentro de ella se guarda uno o varios archivos con extensión .html y el fichero que dota de estilos a la página con extensión .css.

Para operar con el servidor web se puede utilizar una pantalla táctil integrada, o bien cualquier dispositivo móvil que sea capaz de escanear códigos QR (que representa una dirección IP). Para visualizar el servidor web también se puede conectar un monitor externo o televisor.

Ilustración 17. Izquierda acceso al servidor web por QR, a la derecha pantalla táctil y monitor.

Para conectarte al servidor web del futbolín, podemos acceder mediante la conexión a la misma red wifi a la que esté conectada el microcontrolador, y escanear un código QR (en la imagen el derecho). También se podría acceder conectándonos al punto de acceso que genera el futbolín; para ello habría que buscar la red generada entre las redes wifi disponibles y una vez localizada, introducir la contraseña y escanear el código QR (en la imagen el izquierdo).

Al igual que un dispositivo móvil puede conectarse al punto de acceso generado, también puede la Raspberry pi, que únicamente le faltaría acceder a la dirección IP (que representa el código QR) del punto de acceso. Una ilustración del servidor web podría ser la siguiente:

Ilustración 18. Servidor web

Las repeticiones de los goles, comentadas en el apartado de iluminación, son visualizados en la pantalla táctil (que aparece en la derecha de la ilustración 17), televisión o monitor conectados a la Raspberry Pi. Si se quieren utilizar dos dispositivos (como la pantalla táctil para seleccionar los parámetros de juego y una televisión para visualizar el servidor y repeticiones), existen dos opciones: utilizar una raspberry pi 4 ya que viene incorporado con 2 puertos HDMI, o utilizar un dispositivo llamado splitter, cuya función es mostrar por dos o más puertos de salida HDMI los datos recibidos por un puerto de entrada HDMI.

3.8 La fuente de alimentación

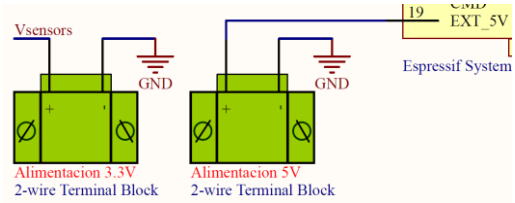


Fig. 13.Parte de esquemático de la alimentación de sensores.

El microcontrolador, la raspberry, la pantalla la iluminación, los sensores, etc; son elementos electrónicos que necesitan una fuente de alimentación. Para alimentarlos desde la red eléctrica se hace uso de una regleta convencional.

- Fuente de alimentación de raspberry 5V-3A (consumo típico 1A).
- Módulo de alimentación MB102 de (3,3 o 5V)-700mA (Consumo típico 300mA) para más detalles consultar [8].
- Fuente de alimentación de iluminación 12V-1A (Consumo típico 100mA)
- Alimentación splitter HDMI (Prescindible) 5V.1A (Consumo típico 80mA)

Ilustración 19. Regleta y fuentes de alimentación.

4 Especificaciones de software

Los algoritmos, funciones y diseño del software, aunque pasan desapercibidos en el producto final, son una parte muy importante a tener en cuenta.

Un buen estilo de programación aporta flexibilidad para realizar con facilidad futuras mejoras y modificaciones. De lo contrario, un mal diseño puede suponer grandes dificultades para realizar cualquier cambio.

Otro factor que puede ahorrar bastante tiempo al programador, es organizar y documentar correctamente el código.

Los lenguajes de programación utilizados tienen que adaptarse a la aplicación que desarrollan. Por ello se han usado tres tipos de lenguaje como es Python, HTML5 (Javascript, HTML y CSS) y C++.

C++	HTML5	Python
- Gestión de monedero. - Sensorizado de goles, de acelerómetros y de lector de tarjetas. - Cronometraje y gestión del tiempo. - Rotaciones y administración de modos de juego.	- Diseño de la apariencia del servidor web (CSS). - Estructura del servidor web (HTML) - Actualización de variables del servidor web (JavaScript).	-Escritura automática en Excel de los datos enviados por puerto serie. - Visualización del servidor web en pantallas táctil y monitores. - Procesado y reproducción de las repeticiones de gol.

Tabla 3. Aplicaciones de lenguajes de programación utilizados

En Python, por el reducido tamaño del código, el programa no está estructurado, el código está formado por un único bloque contiguo de instrucciones.

Para la aplicación web, la apariencia de la página se da mediante un fichero externo con extensión .css; en cambio, para la programación de JavaScript se realiza una programación como script en línea, es decir en el archivo con extensión .html se introduce el código de JavaScript entre las etiquetas <script> y </script>.

En C++ se realiza una programación orientada a objetos. Es decir, un paradigma de programación en el que se crean objetos o entidades que tienen asociados unos atributos y métodos. Los objetos manipulan los datos de entrada para obtener unos datos de salida específicos.

Un ejemplo podría ser el objeto portería1. Este objeto dispone de atributos como podría ser la distancia entre los sensores, los dos pines que ocupan los dos fotointerruptores de la barrera óptica, o incluso podría ser otro objeto un atributo, por ejemplo, un objeto cronómetro. En base a los datos de entrada obtenidos por la barrera óptica, se puede utilizar un método propio del objeto, como podría ser "calculovelocidad()". El método haría uso del atributo cronómetro y los valores de entrada para generar un dato de salida como puede ser la velocidad de la bola.

Para trabajar con objetos se utilizan las clases. Las clases no son más que una plantilla para la creación de objetos. En el ejemplo anterior, existiría una clase que serviría para crear dos objetos como podría ser "portería_Local" y "portería_Visitante". Ambas tendrían los mismos atributos y métodos, pero los datos de cada atributo podrían ser diferentes, como pasa con en el ejemplo anterior con los pines de las barreras ópticas de cada portería.

Las clases se almacenan en librerías. Para la creación de librerías como se explica en [9], se necesitan dos ficheros: uno con extensión .h y otro con extensión .cpp. El fichero principal tiene extensión .ino y los ficheros para la aplicación web se almacenan en la carpeta data.

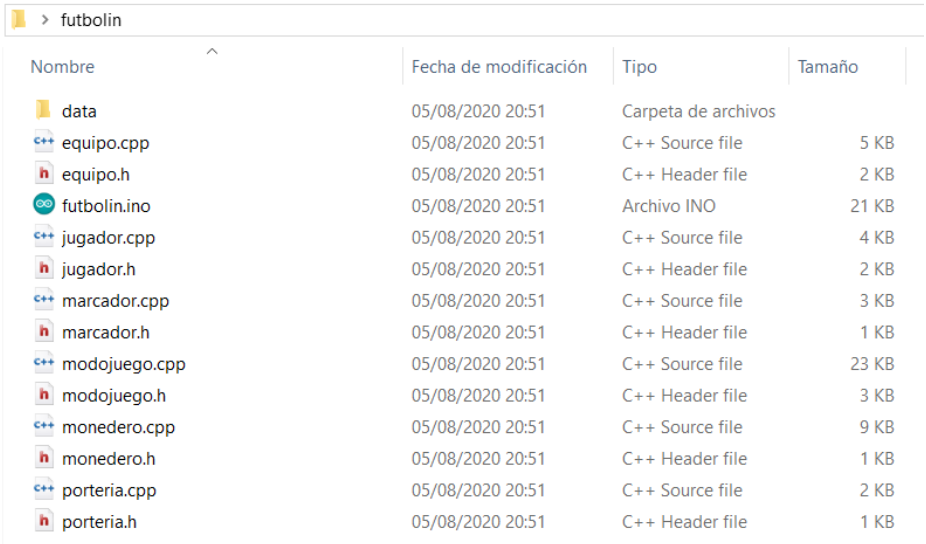


Ilustración 20 Archivo futbolín.

Ilustración 20. Archivo futbolín.

Para pasar a explicar el software, es necesario separarlo por capas para facilitar su compresión. A continuación, un diagrama de bloques con lo comentado:

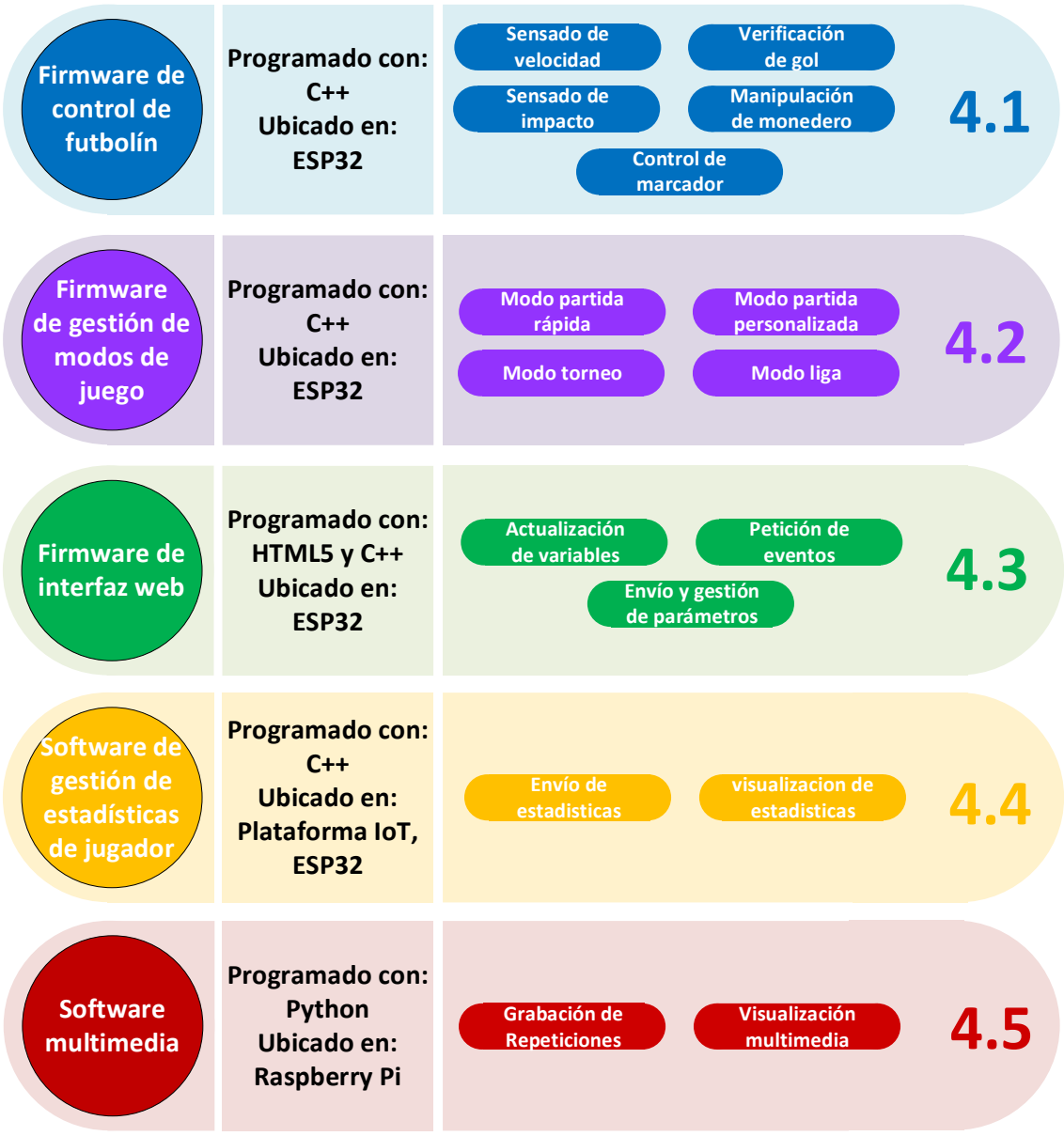


Fig. 14. Capas de software.

Como se observa en la figura anterior, todas las capas tienen un gestor de comunicaciones. Este se encarga de ser el medio por el que se comunican los distintos elementos de la misma u otra capa.

4.1 Firmware de control de futbolín

El software es el conjunto de rutinas y programas que hacen posible la realización de tareas específicas. El programa o conjunto de programas encargados de controlar a los circuitos electrónicos de un dispositivo (hardware) se denomina firmware.

La función del firmware de control de futbolín es gestionar todos los periféricos conectados y gestionados por el microcontrolador ESP32. A continuación, pasaremos a comentar las distintas subcapas del mismo.

4.1.1 Sensado de velocidad y verificación de gol

Como comentamos en el apartado 3.3, para medir la velocidad de la bola se requiere cronometrar el tiempo transcurrido entre que dos fotointerruptores emiten una señal. Para la aplicación se hace uso de lo que se denominan interrupciones. Como se indica en [10], éstas son funciones que interrumpen el curso de ejecución del programa y pasan a ser ejecutadas. El curso de ejecución tras existir una interrupción se representa en la siguiente figura:

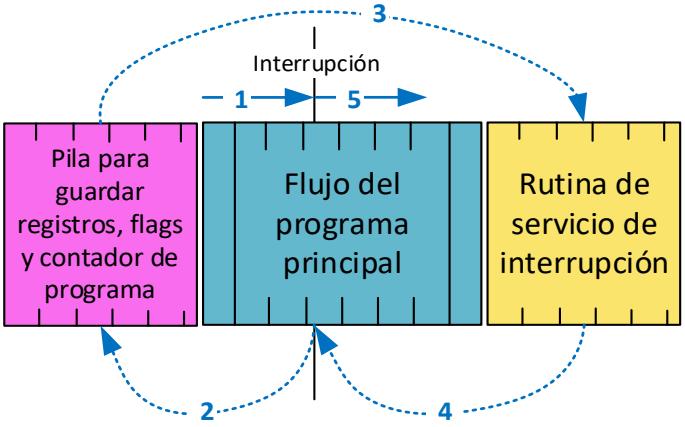


Fig.15. Curso de programa con una interrupción.

Para los pines de señal de los fotointerruptores se le asocian una interrupción por flanco de bajada, de modo que cada vez que la tensión del pin pase de 3.3V a 0V, se parará el flujo del programa principal y se pasará inmediatamente a ejecutar la rutina de interrupción.

Para sensar la velocidad se utiliza la función micros(). Con esta función se obtiene un contador que se incrementa 1000 veces cada milisegundo. En el contador se pasa del valor máximo a cero (Overflow) cada 70 minutos.

Un valor alto de velocidad podría ser 50 m/s y uno bajo 2 m/s (1m/s = 3.6km/h).

$$v = \frac{dist}{t};\ t = \frac{dist}{v} = \frac{0.005\ m}{50\ m/s} = 0.0001s = 100us \rightarrow 100\ unidades$$

$$t = \frac{dist}{v} = \frac{0.005}{2} = 0.0025s \quad ; \quad \frac{0.0025\ s}{70 * 60\ s} * 100\% \approx 0.00006\%$$

Se puede comprobar que con medidas altas de velocidad podemos obtener una resolución aceptable de 100 unidades. Para ver la influencia del Overflow estudiamos un caso desfavorable, como una velocidad baja. Se deduce una muy baja probabilidad de ocurrencia: 6 de cada 10 millones.

Como hemos comprobado, la función micros() es una muy buena opción para el cálculo de velocidad. Por ello, se guarda su valor en una variable y cuando el segundo fotointerruptor genere su interrupción, además de guardar el valor de micros(), se activará un flag (variable booleana) para calcular la velocidad y verificar si ha sido gol.

Los diagramas de flujo de las interrupciones de gol de cada fotointerruptor son:

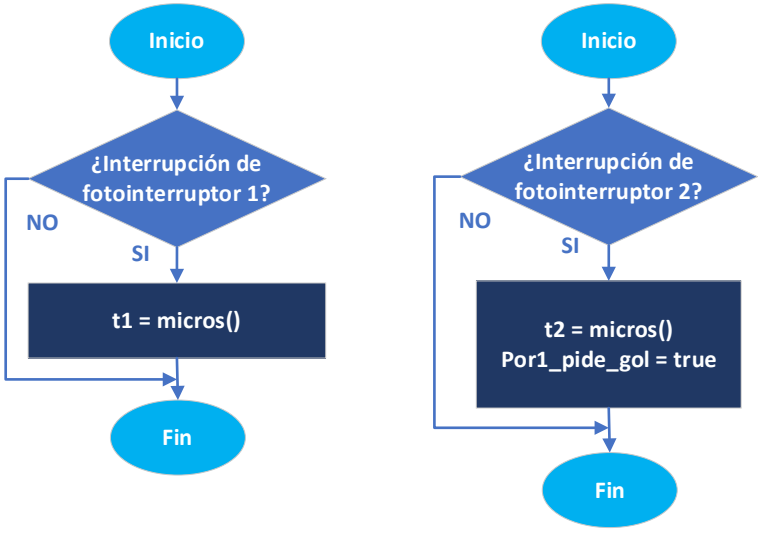


Fig.16. Diagramas de flujo de interrupciones de gol.

Para comprobar si ha sido gol, se observa durante unos segundos si ha pasado la bola por el fotointerruptor del verificador. A continuación, un diagrama de flujo del sensado de velocidad de la bola y verificación de gol.

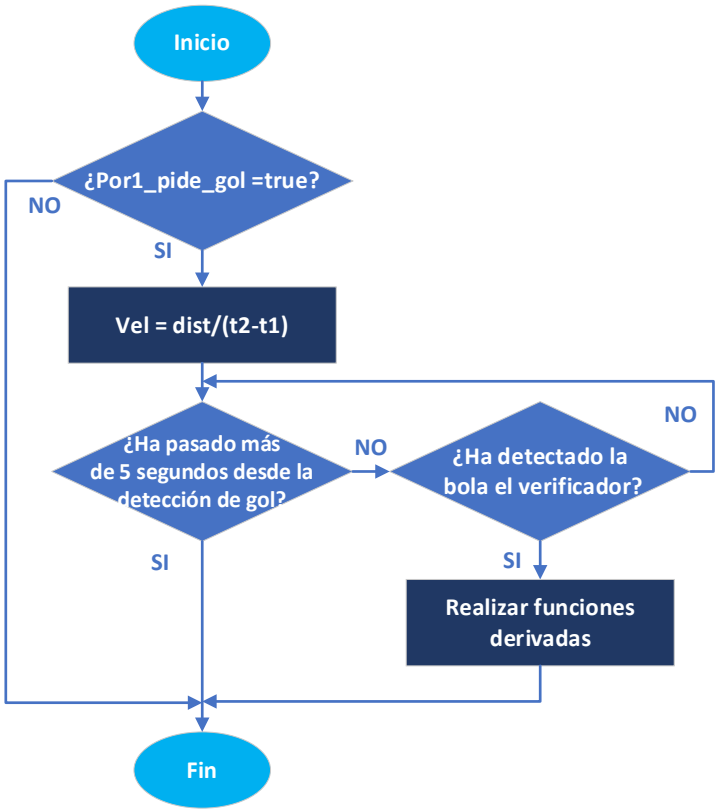


Fig.17. Diagrama de flujo del sensado de velocidad de la bola y verificación de gol.

4.1.2 Sensado de impacto

Como vimos en el apartado 3.4, el futbolín dispone de dos módulos MPU92/65 para poder obtener medidas de la vibración producida por la bola al golpear el fondo de portería donde se ubican los acelerómetros (llamada pared de impacto).

Para el uso de los módulos MPU92/65, se utiliza la librería MPU9250-Master [11]. Esta librería, permite leer los valores del módulo mediante comunicación I2C o SPI.

Debido a la naturaleza del problema, el módulo debe funcionar y ser leído en el menor tiempo posible. Ya que el tiempo en el que se produce la vibración debida al impacto es muy reducida, resulta totalmente necesario obtener el mayor número de muestras de los módulos.

La librería dispone de una función setSrd() para modificar la frecuencia de muestreo, de tal forma que el índice de muestreo se calcula:

$$índice\ de\ muestreo = \frac{1000}{1+SDR} = \frac{1000}{1+0} = 1000\ muestras/\text{s}$$

Para obtener el mayor número de muestras, se emplea la función con 0 como parámetro, de tal forma que conseguimos obtener 1000 muestras cada segundo o lo que es equivalente muestrear a 1000 Hz.

Dado que el módulo MPU92/65 dispone de tres elementos y únicamente se utiliza el acelerómetro, se ha modificado el código de la librería para inhabilitar el magnetómetro y giroscopio, y así liberar carga de procesamiento al microcontrolador.

Adicionalmente, se emplea la tipología de bus SPI para realizar la comunicación lo más rápido posible.

Para realizar la lectura de los acelerómetros de la manera más óptima posible, se realizó una versión de futbolín para que este se encargase únicamente del sensado de velocidad e impacto de la bola. En esta versión se emplean seis arrays de 500 datos tipo float para almacenar las lecturas de la fuerza en los ejes X, Y, Z de los dos acelerómetros.

Tras cada lectura se espera 1ms, esto se debe a que el acelerómetro toma 1000 muestras cada segundo, sin embargo, la lectura y guardado de la muestra se realiza a 100us. Si no esperásemos ese tiempo, se guardaría la misma muestra en diez posiciones del array.

Una vez se han guardado las medidas de los acelerómetros en los arrays, éstas tienen que ser enviadas por el puerto serie.

El diagrama de flujo para el sensado de impacto es el siguiente:

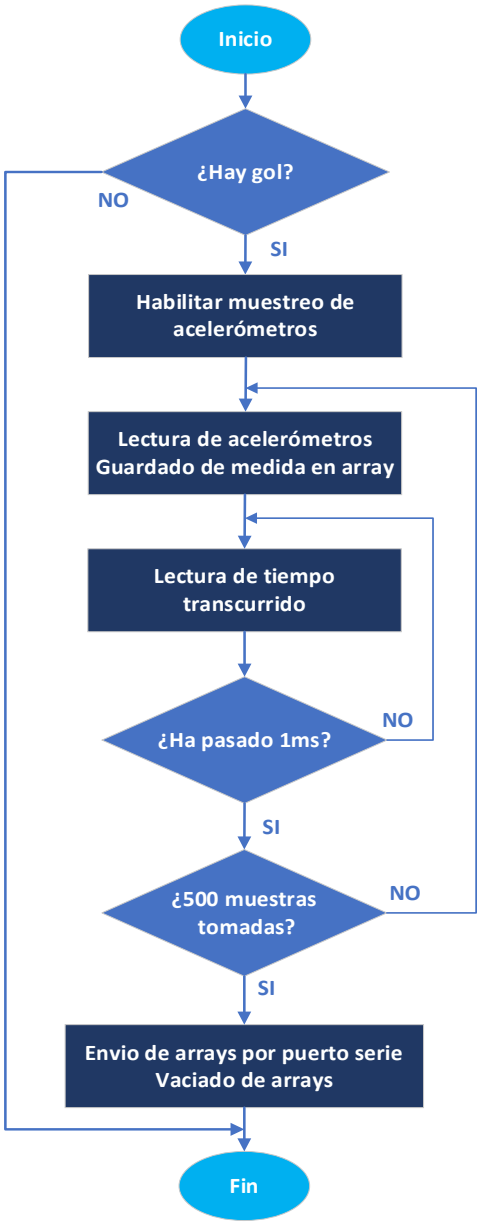


Fig.18. Diagrama de flujo para sensado de impacto

Según se comentó en el apartado 3.4, mediante un programa implementado en lenguaje Python inspirado en el ejemplo de la librería SerialToExcel [12], se lee el puerto serie y se tabulaban los datos en un documento Excel. S e realizan 15 ficheros debido a que las tiradas son a tres alturas y en cinco posiciones. En esos ficheros se ponen las diez tiradas cada una con sus 500 muestras de X, Y y Z de cada acelerómetro. Finalmente, se analizan los datos para obtener una correlación entre la medida y la velocidad y zona de impacto. Para más detalle de los resultados

4.1.3 Manipulación del monedero

Para la gestión del monedero adaptado se emplea una librería propia con la clase monedero. En ella se incluyen dos librerías adicionales de código abierto como son Servo [13] y MFRC522 [14], para controlar el actuador y el lector de tarjetas respectivamente.

Como comentamos en el apartado de "3.3 monedero adaptado" el monedero está formado por tres elementos: final de carrera, actuador y lector de tarjetas. Estos componentes se relacionan entre sí mediante dos atributos de la clase monedero como son el coste de la partida y el saldo del futbolín.

La lógica del actuador es habilitarse o ponerse "en alto" cuando el saldo del futbolín sea mayor que el coste de la partida. En ese caso el servomotor con su biblioteca servo cambiará su posición angular levantando la palanca que bloquea el dispensado de las bolas. A continuación, su diagrama de flujo.

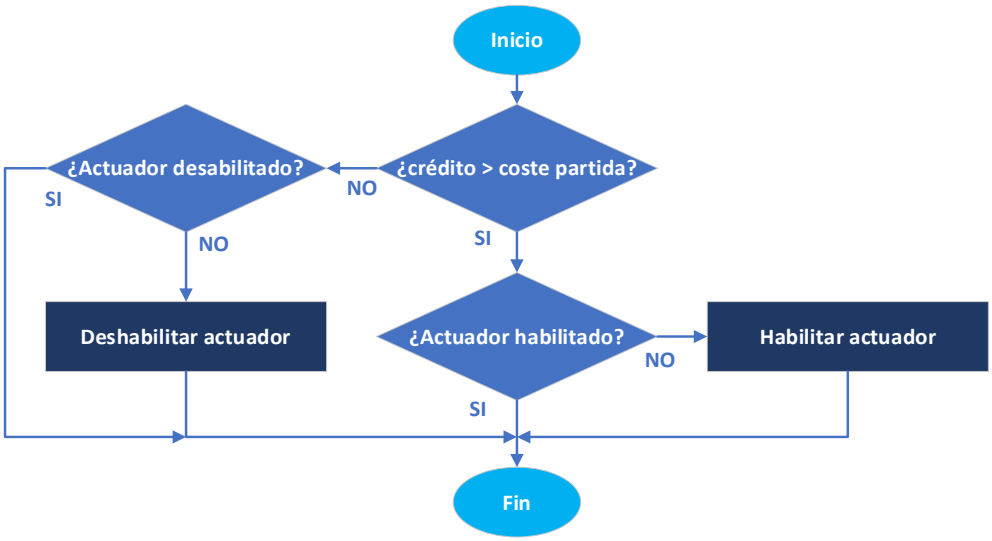


Fig.19. Diagrama de flujo del actuador

El final de carrera es un pulsador con una rutina de interrupción asociada. Cuando se produce la interrupción se activa un flag para que se gestione la interrupción en el bucle principal.

Cuando se pasa a gestionar la interrupción generada, al saldo del futbolín se le resta el coste de partida. Para evitar los rebotes del pulsador se esperan 100 ms, de este modo conseguimos evitar que el coste de partida se reste más de una vez. Finalmente se actualiza el estado del actuador.

Para el uso del detector de tarjetas, aproximamos la tarjeta al sensor y así, leer y/o escribir datos como su saldo.

En la clase monedero existen distintos métodos para poder explotar al máximo todas las posibles funcionalidades de la tarjeta. Así pues, se puede leer el saldo de la tarjeta o escribir el saldo deseado (bien enviándolo como un parámetro de función o escribiendo el valor de recarga por el puerto serie).

Se puede combinar la lectura y escritura. Por ejemplo, se lee la tarjeta y tiene saldo suficiente, carga el saldo de la tarjeta al futbolín y escribe en la tarjeta el nuevo saldo ya restado. Ésta será la ejecución que realizará por defecto el futbolín. Si la tarjeta se queda sin saldo tendrá que ser recargada con el uso de un dispositivo externo de recarga de tarjetas.

Para que el usuario conozca el saldo que tiene su tarjeta, se utiliza un atributo de monedero donde se almacena el valor del saldo de la última tarjeta leída y se visualiza por la aplicación web. El diagrama de flujo del lector de tarjetas es este:

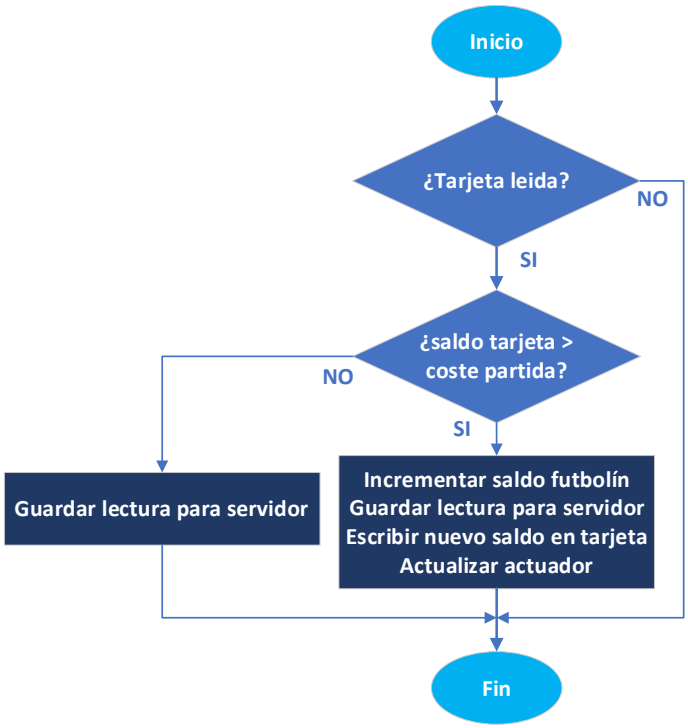


Fig.20. Diagrama de flujo del lector de tarjetas

4.1.4 Control de marcador

Para hacer el recuento de los goles, se utiliza una clase denominada marcador. Ésta únicamente lleva el registro de los goles de los equipos que están jugando la partida. Dispone de funciones para ser leído o modificarlo. Adicionalmente, se hace uso de la librería TM1367 detallada en [15], la cual es empleada para controlar el display que se comentó en el apartado 3.2.

4.2 Firmware de gestión de modos de juego

Los modos de juego son un conjunto de reglas y mecánicas de juego que indican cuándo finaliza la partida y qué equipos juegan. Al igual que para el sensado de velocidad o monedero, los modos de juego también se incorporan en una clase propia; en este caso llamada modo de juego. La presente clase también hace uso de otras clases que todavía no han sido explicadas como puede ser la clase equipo o la clase marcador.

Existen principalmente 4 modos de juego:

- Partida rápida.
- Partida personalizada.
- Torneo.
- Liguilla.

En los modos de juego se gestionan las partidas que juegan cada equipo. La rotación de estas partidas se realiza de dos formas:

- Cuando se llega a un número determinado de goles.
- Cuando se sobrepasa un tiempo determinado y un equipo lleva más goles que el otro.

Todos los modos de juego disponen de 3 parámetros para seleccionar la tipología de partida. El único modo que no requiere la selección de ningún parámetro es partida rápida y por ello es el que se juega por defecto cuando no se ha seleccionado nada. Los parámetros son:

- Número de equipos.
- Rotación por goles/Rotación por cuenta atrás.
- Goles para rotar/Tiempo para rotar.

El número de equipos va a determinar cuántos objetos de la clase equipo van a ser generados. Estos equipos van a tener una serie de atributos como su nombre, identificación o estadísticas, los cuales van a ser utilizados junto con sus métodos. Habrá métodos para obtener esos atributos y métodos para modificarlos.

Los equipos generados se guardarán en un array de equipos. Un array es un tipo de dato estructurado que permite almacenar en él un conjunto de datos

homogéneos, en este caso de equipos. Moviendo la posición de los equipos en el array, podemos realizar con facilidad las mecánicas de juego.

Si se decide rotar por goles, se hará uso de la clase marcador para obtener el número de goles marcados por cada equipo. Cuando un equipo llegue al número necesario para la rotación, éste se llevará la victoria y el array de equipos cambiará para dar paso a la siguiente partida.

En el caso de rotación por tiempo, se hace uso de la librería "Chrono". Esta librería se empleará para iniciar un cronómetro al inicio de la partida. Cuando en el cronometro obtengamos un tiempo mayor al establecido, se procederá a finalizar la partida. En el caso de que haya un equipo con más goles anotados, este se llevará la victoria, en caso de empate, se realizará una muerte súbita hasta desempatar. Posteriormente cambiará el array y se dará una nueva partida.

Todos los modos de juego van a depender de dos flags o variables booleanas que indicarán el estado del modo de juego. Estos dos flags son "preparado" y "jugando" que, como es lógico, hacen referencia al modo de juego. A continuación, se analiza qué sucede en función del valor de los flags:

preparado	jugando	Resultado
falso	falso	- El modo y demás funciones deben inicializarse y crearse el array de equipos correspondiente.
falso	verdadero	- Situación incompatible, no se realizan las mecánicas de juego (pensado para futuras mejoras).
verdadero	falso	- El modo está parado a la espera de ser reanudado (pensado para futuras mejoras).
verdadero	verdadero	- El modo de juego transcurre con normalidad realizándose las mecánicas de juego establecidas.

Tabla 4. Análisis de resultados dependiendo los flags preparado y jugando de modo de juego.

A continuación, describiremos en detalle el funcionamiento de los distintos modos de juego.

4.2.1 Partida rápida

El modo partida rápida no requiere de parámetros y es el modo por defecto. Cuando se juega al modo partida rápida, sólo juegan dos equipos; no hay ninguna rotación de equipos, ni límite de tiempo o goles. Por tanto, la partida no finaliza hasta que los equipos decidan cambiar de modo o dejar de jugar.

La partida rápida, al igual que los demás modos de juego, utiliza los flags comentados anteriormente. Cuando los dos tienen el valor "falso", el modo de juego inicializa el marcador poniendo a 0 los goles de ambos equipos. Una vez inicializado, ambos flags pasan a valer "verdadero" y comienza la partida. En el transcurso de la partida, el modo no realiza ninguna acción, el marcador se incrementa por interrupciones como se vio en el apartado 4.11.

4.2.2 Partida personalizada

El modo partida personalizada es el equivalente a jugar a "rey de la pista".
Cuando se selecciona este modo, al igual que con el de partida rápida, el marcador se reinicia. Además, a partir del parámetro del número de equipos, se crean tantos como este parámetro indique, guardándolos en un array de equipos denominado "listadeequipos". Para determinar cuáles son los equipos que van a jugar, se toman las dos primeras posiciones del array correspondiendo la posición de local y visitante a la primera y segunda posición del array. El equipo perdedor pasará al final del array, y el que estaba esperando empezará a jugar.

Los equipos del array "listadeequipos" se gestionan principalmente por la posición que ocupan en el array y por su identificador o ID. Cuando se crean los equipos al inicio de la partida, se pone un nombre por defecto a los equipos formado por "Equipo" + el valor de ID. El nombre puede ser modificado por cualquier otro de entre 2 y 16 caracteres. Esto será común en los demás modos de juego.

Para entender la mecánica de rotación de equipos en partida personalizada, podemos poner un ejemplo con ocho equipos en los cuales la mayoría se han cambiado el nombre.

El array de equipos evolucionaría de la siguiente forma:

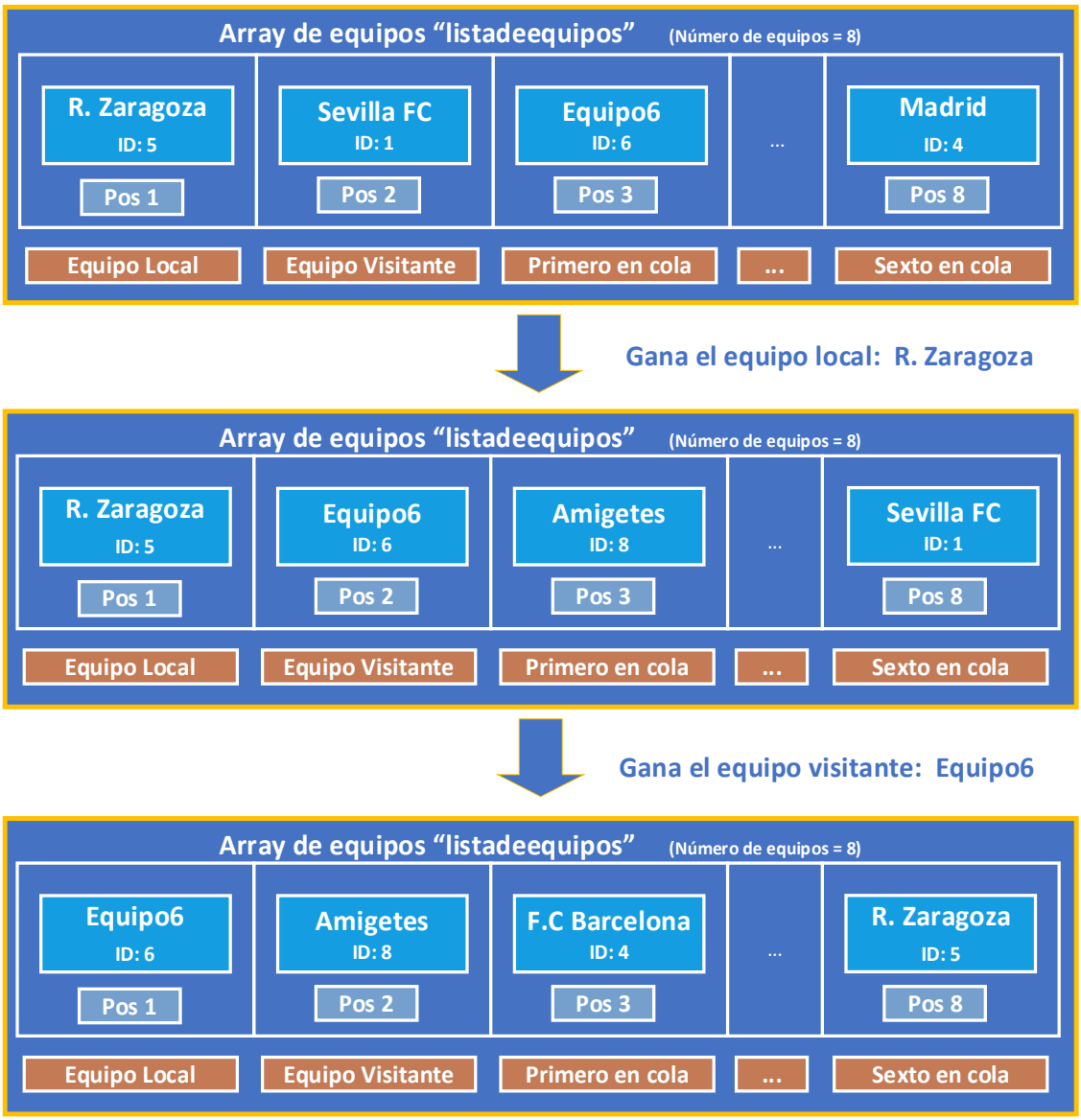


Fig.21. mecánica de rotación de equipos en partida personalizada

Podemos observar en la figura anterior cómo el equipo que gana la partida permanece jugando, mientras que el perdedor se pone al final de la cola. Es importante mencionar que cuando un equipo visitante gana al equipo local que estaba jugando, éste pasará en la siguiente partida a ser el equipo local. El equipo local se ubicará en el lado del cajón de las bolas y concederá la bola al equipo visitante para que realice el saque inicial.

A continuación, el diagrama de flujo correspondiente a la mecánica de juego de partida personalizada:

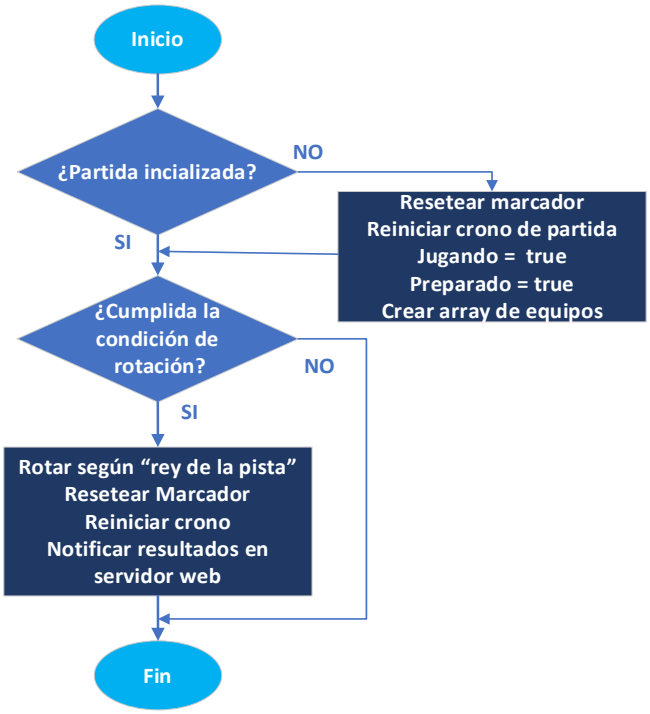


Fig.22. Diagrama de flujo de partida personalizada.

4.2.3 Torneo

A diferencia del modo anterior en donde los equipos podían estar rotando de manera indefinida, el modo torneo tiene una duración determinada. Esto se debe a que los equipos que han sido derrotados no vuelven a jugar como en el caso anterior, sino que son directamente eliminados. Cuando sólo quede un equipo invicto, este se hará con la victoria del torneo. Un ejemplo de torneo utilizando el esquema típico con 13 equipos generados por defecto podría ser:

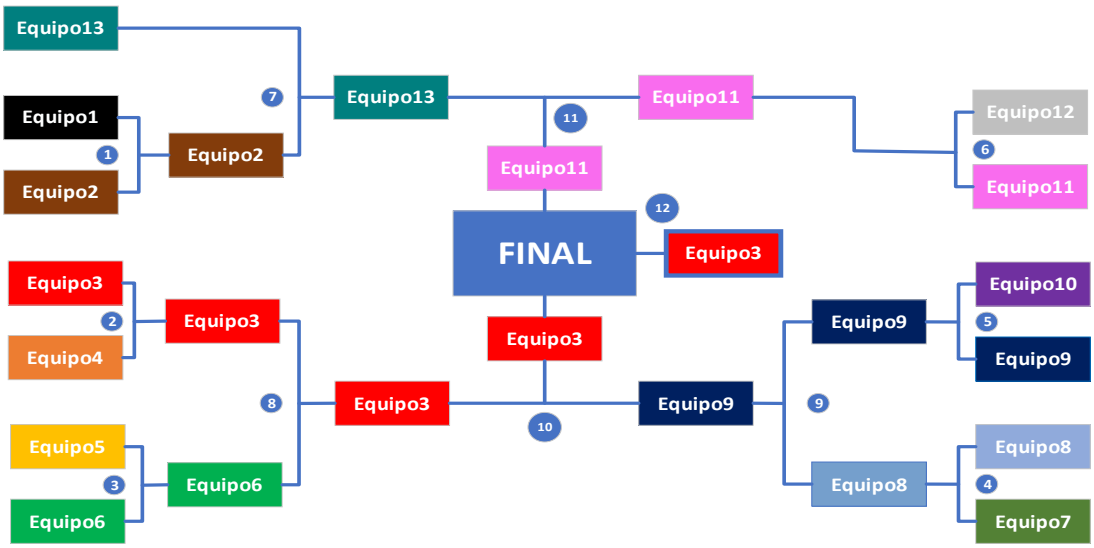


Fig.23. Ejemplo de esquema de torneo con 13 equipos

El torneo empieza y los equipos son creados y guardados en el array "listadeequipos". A partir de entonces, se jugarán un número de partidos una unidad menor al número de equipos. Por tanto, en el ejemplo comentado se disputarán 12 partidos. Recolocando adecuadamente los equipos en el array, podemos llegar a efectuar los partidos en el orden correcto. A continuación, se muestra el modo de rotación del array de equipos según los resultados de la figura 23.

Fig.24. Mecánica de rotación de equipos en torneo

Como se puede observar en la figura 24, los equipos que por estar en las dos primeras posiciones les toca jugar, cuando roten, el equipo ganador se ubicará en la posición cuyo valor será igual al número de partidos restantes (incluyendo el que se está jugando). El equipo perdedor se situará una posición por detrás del equipo ganador.

Tomando como ejemplo la figura 11 y el partido número 3 en la que gana el equipo 6, en la partida 4 quedarán 10 partidos y la posición del equipo ganador será la décima y la del equipo perdedor la decimoprimera.

Cuando el último partido finaliza, el modo de juego acaba. El equipo vencedor se notifica en el servidor web. Los equipos restantes quedan ordenados en el array de acuerdo a la posición en la que han quedado en el torneo.

Una vez finalizado el torneo, se pasará a jugar partida rápida. El diagrama de flujo de la mecánica de juego de torneo será la siguiente:

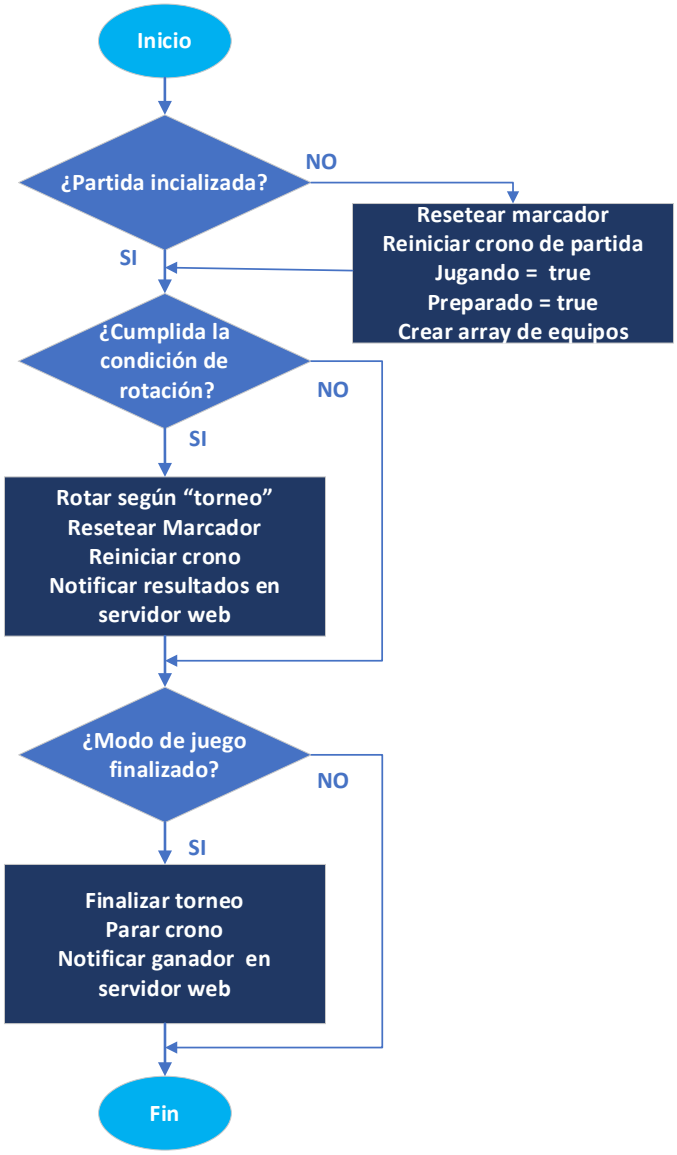


Fig.25. Diagrama de flujo de torneo

Podemos observar cómo el diagrama de flujo de torneo es muy similar al de partida personalizada, salvo que el de torneo incluye una condición adicional para revisar si ha finalizado el modo de juego.

4.2.4 Liga

El modo liga es el que permite a los equipos poder jugar el mismo número de partidos que los demás. Gracias a este tipo de mecánicas, podemos evitar que un equipo que juegue mejor que el resto permanezca invicto en el futbolín por mucho tiempo, como pasa en partida personalizada.

El número de partidos que se juegan en modo liga se obtendrá mediante la siguiente expresión matemática:

$$número\ de\ partidos = \sum_{n=1}^{n=Nequipos-1} n$$

Para comprobar de forma gráfica la expresión anterior, se puede emplear el siguiente diagrama:

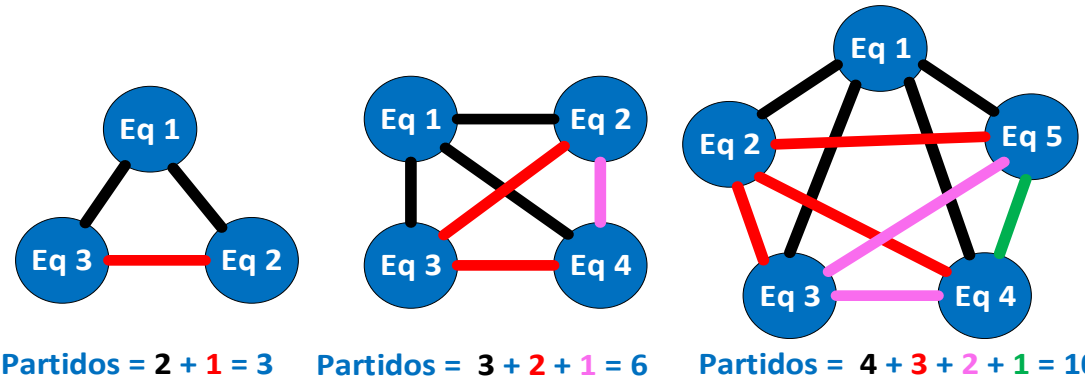


Fig.26. Esquema para obtener el número de partidos de liga

Se observa de forma gráfica cómo el número de partidos coincide con la expresión matemática anterior. Si se quisiera jugar 2 partidos con cada equipo (ida y vuelta), el número de partidos sería el doble.

Con el fin de almacenar las estadísticas necesarias en los equipos para posteriormente obtener la clasificación de la liga, se hace uso del array "listadeequipos" comentado previamente. Este almacenará los partidos ganados, goles a favor y goles en contra de cada partido.

Para gestionar la rotación de los equipos, al contrario que con torneo o partida personalizada que usaban el array "listadeequipos", se va a utilizar un array llamado "listadepartidos". En dicho array se creará una copia de los equipos ordenados según los partidos que se vayan a jugar. Según vayan jugando, se desplazarán hacia las primeras posiciones.

Un ejemplo sencillo con 3 equipos podría ser:

Fig.27. Array "listadepartidos" con 3 equipos diferentes.

Para obtener la lista de partidos se ponen dos columnas con los equipos ordenados de arriba a abajo según han sido creados. Posteriormente se desplaza la columna de la derecha hacia abajo y se ubica en la primera posición el último equipo. Una vez realizada esta operación, se comparan las dos columnas obteniéndose un número de partidos igual al número de equipos. Finalmente se repetirá esta operación hasta obtener el número de partidos requerido.

Un ejemplo de generación de partidos de liga con 5 equipos puede ser:

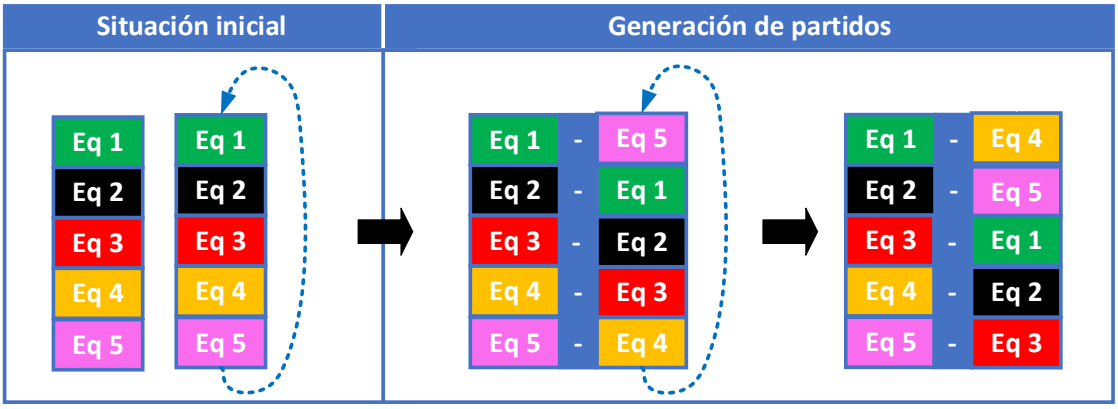


Fig.28. Modo de generación de partidos de liga.

Rellenado el array “listadepartidos, se jugarán las correspondientes partidas evolucionando el array según muestra la figura anterior. Cuando se acaben los equipos de “listadepartidos” se ordena el array “listadeequipos” y se visualiza por el servidor web la clasificación de la liga. Al igual que pasa con torneo, acabada la liga se pasará a jugar partida rápida.

El diagrama de flujo de liga es similar al del torneo, lo único que cambia es que se usan dos arrays con equipos en vez de uno; y que acabado el modo se tiene que ordenar “listadeequipos”.

4.3 Firmware de interfaz web

En el apartado 3.7, vimos qué es el servidor web y para qué sirve, e introdujimos brevemente cómo funciona. En este apartado veremos más detalladamente el funcionamiento del servidor web y la forma de comunicarse con el microcontrolador.

Para poder cargar el servidor web en el microcontrolador a través de la IDE de Arduino, se necesita de un plugin llamado "ESP32 Filesystem Uploader" [16].

Además de utilizar SPIFFS para almacenar los ficheros HTML y CSS como se vio en el apartado 3.7, también se usa una librería "ESPAsyncWebServer" [17]. Esta librería destaca por permitir realizar el servidor totalmente asíncrono. Con esto, nos referimos a que el envío y la recepción de datos no se hace de manera coordinada, sino que cuando un cliente hace una petición (request") al servidor web guardado en el microcontrolador (como por ejemplo incrementar marcador), esta petición es gestionada por el microcontrolador cuando a éste le sea conveniente en vez de hacerlo de una vez. Cuando pueda ser gestionada la petición, se envía una respuesta ("respond") con los ficheros html y css.

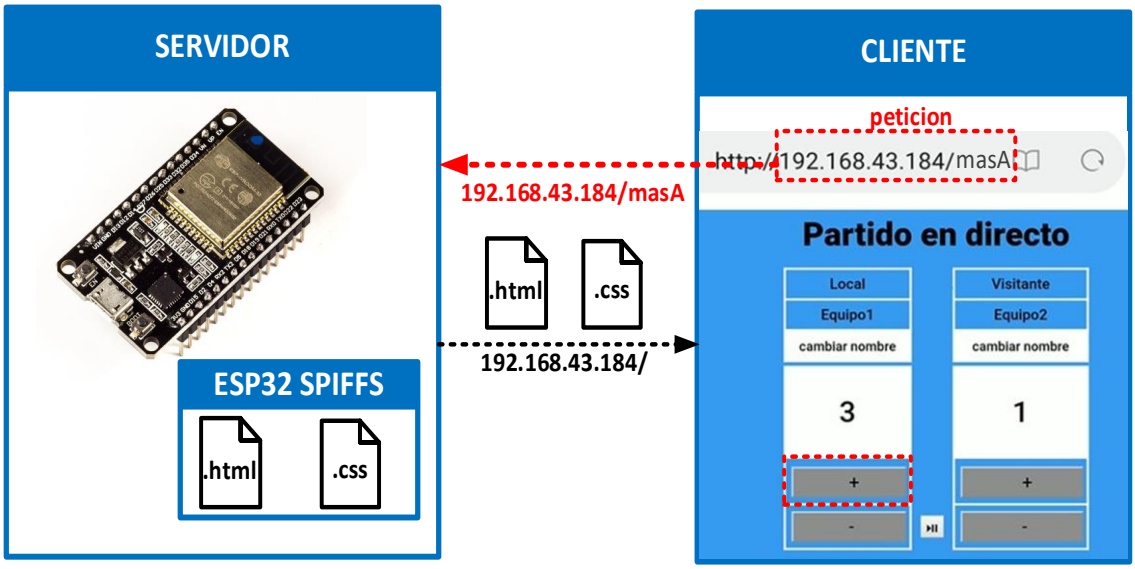


Fig.29. Diagrama de comunicación cliente-servidor

Entre las ventajas de utilizar un servidor asíncrono destacan: su alta velocidad, permite dejar libre el bucle del programa principal para realizar otras tareas, y se pueden gestionar más de una conexión a la vez.

Las peticiones anteriormente comentadas, se hacen en la URL del navegador. Esta URL está compuesta por el protocolo (http://), el directorio raíz (que coincide con la dirección IP de la red privada 192.168.43.184) y un directorio (como por ejemplo /masA).

Enviar un directorio precedido con "/" se usa para dos finalidades realizar peticiones de eventos y actualizar variables.

4.3.1 Realización de peticiones de eventos

Estas peticiones se realizan mediante la pulsación de un botón en el servidor web, y sirven para modificar el estado del futbolín. Entre ellas están incrementar o decrementar el marcador y pausar-reanudar el cronómetro de la partida.

Este tipo de peticiones cuando se gestionan siempre van seguidas de una o varias funciones a realizar. Además, la URL será redireccionada al directorio raíz "/".

4.3.2 Actualización de variables

Este tipo de peticiones son realizadas automáticamente de manera periódica por el código realizado en JavaScript dentro del fichero HTML. Para dar el valor correspondiente a los contenedores de variables ("placeholders") del servidor web, se utiliza una función processor(). (revisar las funciones del fichero .ino del apartado 11 de anexos).

4.3.3 Envío y gestión de parámetros

Hay situaciones como la selección del modo de juego, que requieren el envío de parámetros desde el servidor web al microcontrolador. En este caso, desde el directorio raíz, se realiza la petición para obtener los parámetros de la URL.

Para enviar parámetros, se añade después del directorio raíz precedidos de "/?": el nombre, el parámetro, el símbolo "=" y el valor del parámetro. Para el envío de más parámetros se añade el símbolo "&" y se repite lo anterior. Un ejemplo para enviar los parámetros de nombre de jugador, equipo y posición podría ser:

http://192.168.43.184/?jug=Rodri512&jug_eq=1&jug_pos=1

Para la obtener los parámetros se utiliza una función como se especifica en [18].

Se requiere el uso de parámetros en los siguientes casos:

- Elección del modo de juego: Se selecciona el modo, número de jugadores, la condición de rotación y la forma de rotación.
- Cambio de nombre de equipo Se introduce el nombre de equipo.
- Registro de jugador: Se introduce el nombre del usuario, el equipo y la posición en la que juega.

Los parámetros se suelen enviar mediante formularios añadidos en pestañas emergentes llamadas "popups". A continuación, una imagen detallada del servidor web y sus partes principales.

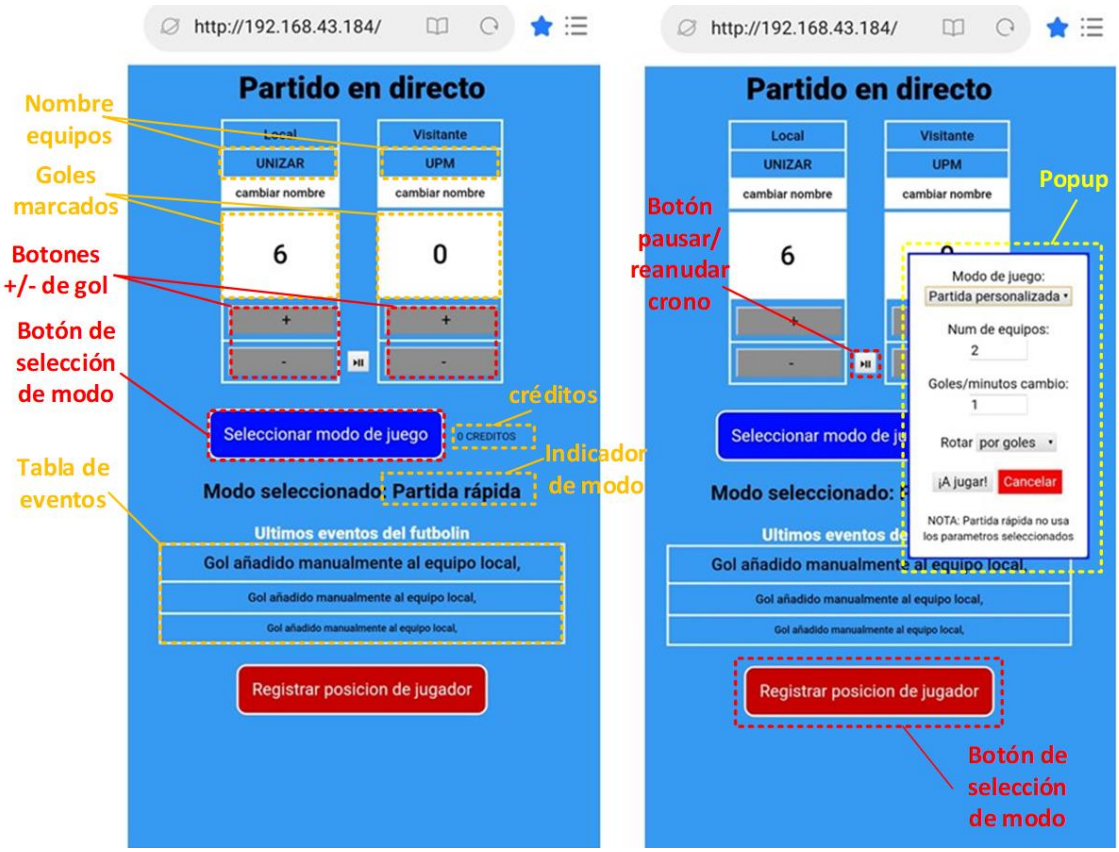


Ilustración 21. Partes del servidor web

En la ilustración de arriba, podemos ver de color amarillo las pestañas emergentes o "popups", de naranja los contenedores de variables o placeholders, y de rojo los botones de acción que modifican el estado del futbolín.

4.4 Software de gestión de estadísticas de jugador

Como se vio en el apartado 2.2.2, una de las mejoras que podían resultar bastante interesantes era la obtención de estadísticas de juego (goles, tiros a puerta, tiros fuera, posesión, etc.)

Pese a que hay estadísticas como los tiros fuera y la posesión que resultan bastante complicadas de obtener, estadísticas como las partidas jugadas, ganadas o goles marcados sí que resultan fácilmente implementables.

Las estadísticas de partida son propias de cada jugador y por tanto deben gestionarse con una clase propia a la que llamaremos clase "Jugador". Esta clase permite crear objetos "jugador" para asociarlos al objeto "equipo"; éste, en su creación o instanciación, generará dos objetos" jugador" (defensa y atacante). Cada equipo dispondrá de dos flags usados para habilitar la recopilación de las estadísticas del jugador defensor y/o del jugador atacante.

Para registrar un jugador se hará uso de la aplicación web. Los jugadores que se pueden registrar son aquellos que estén jugando la partida (máximo cuatro jugadores por partida). Para registrarse, el jugador tendrá que indicar el nombre de usuario (que en un futuro podría ser un código QR), el equipo y la posición. Una vez registrado el jugador, se habilitarán los flags para recopilar las estadísticas correspondientes.

Para visualizar, almacenar y en un futuro, gestionar las estadísticas, se usa una plataforma web integrada al Internet of Things (IoT) llamada "Thingsboard". Como breve resumen explicativo a lo ya comentado, se ha de decir que la plataforma se basa en un servidor en línea accesible desde internet, en el que dispositivos inteligentes con conectividad a internet (dispositivos IoT) envían datos para ser procesados.

4.4.1 Envío de estadísticas

El envío de las estadísticas del jugador a la plataforma se realiza a través de la clase modo de juego. En partida personalizada, torneo o liga, previamente a la rotación de los equipos, se comprueba los flags de los dos equipos que jugaron y se envían las correspondientes estadísticas de los jugadores registrados.

Además de las estadísticas de jugador, también se envían otras como la del marcador, la velocidad de la última bola marcada en cada portería y el modo de

juego y forma de rotación de los equipos. Para enviar los datos se hace vía MQTT, que es un protocolo de red ligero de publicación y suscripción (para más información de los protocolos utilizados revisar apartado 3 de anexos).

La librería encargada de enviar estos datos se llama "PubSubClient" [19]; mediante ella podemos tanto enviar(publish) como recibir datos(subscribe) al servidor de Thingsboard vía MQTT.

4.4.2 visualización de estadísticas

En base a la documentación [20], el servidor de Thingsboard dispone de un tablero o "dashboard" donde se usan unos "widgets" para la correcta visualización de los datos enviados. Un ejemplo del tablero principal de Thingsboard podría ser el siguiente:

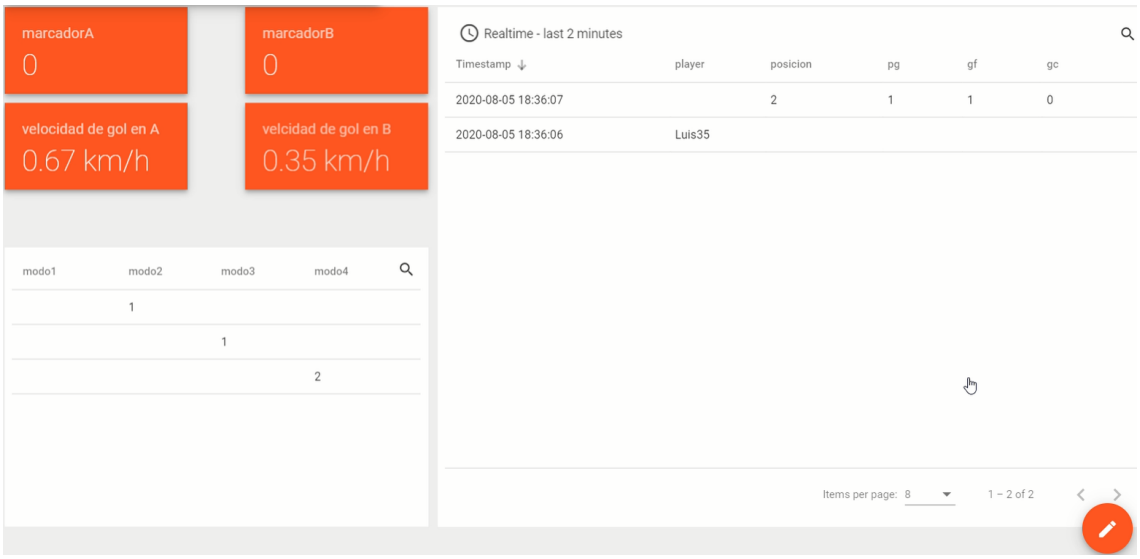


Ilustración 22. Tablero Thingsboard para la visualización de estadísticas.

En la parte superior izquierda, podemos visualizar el marcador y la velocidad de la bola marcada de cada portería. En la esquina inferior izquierda nos encontramos con un widget que indica el modo que ha sido seleccionado (partida rápida, personalizada, torneo y liga) y la forma de rotación. Así pues, cuando se juega a un modo especifico en la celda correspondiente al mismo aparece un valor numérico; este valor es 1 o 2 dependiendo de si se rota por goles o por tiempo. En la derecha aparece el mismo tipo de widget, su función es mostrar las estadísticas del jugador que ha sido registrado.

En un futuro, los datos recibidos deberán de ser procesados para llevar un seguimiento del historial de estadísticas de cada jugador.

4.5 Software multimedia

Como ya se ha comentado en el apartado 3.6, disponemos de una cámara de Raspberry para grabar y reproducir los últimos 7 segundos de video. Además, se explicó que el procesado y la reproducción de las repeticiones de gol se realizaba mediante un código desarrollado en el lenguaje Python. Por ello, se hace uso de la librería "PiCamera" [21] que contiene todas las funciones para manipular la cámara.

Adicionalmente se usa la librería "RPi.GPIO" para leer los pines de señal, la librería "time" para hacer temporizaciones, y la librería "webbrowser" para acceder al servidor web.

4.5.1 Grabación de repeticiones

Para grabar los últimos 7 segundos de video se emplea lo que se denomina un buffer circular. Para ello se usa la función PiCameraCircularIO. En el buffer se van añadiendo los fotogramas grabados por la cámara, cuando el número sea mayor al tamaño del buffer, el fotograma más antiguo se sobrescribe por el segundo más antiguo y así se van eliminando del buffer los fotogramas más antiguos y añadiendo los más recientes. Se mantiene así, el buffer con un número de fotogramas constante correspondiente a 7 segundos de vídeo.

Para saber cuándo ha habido gol y finalizar la grabación de la repetición es necesario comunicar a la raspberry con el microcontrolador. Para ello se podría utilizar el protocolo de comunicación UART, pero por la simplicidad del problema, se optó por que el microcontrolador cada vez que se marque gol y por tanto se requiera mostrar la repetición, genere dos flancos de 15 ms; uno cuando entra por la barrera óptica y otro cuando pase por el verificador. Así pues, basta con un cable de señal y otro de masa y conectarlos a los pines GPIO de la Raspberry.

La Rasberry cuando detecte que en los pines GPIO ha habido un flanco de bajada (es decir, cuando se pase de 3.3V a 0V) parará de grabar, y haciendo uso de la librería time esperará unos segundos a recibir el segundo flanco. Si detecta el flanco, el buffer grabado será reproducido en las pantallas y guardado. En caso contrario, el programa seguirá grabando como antes.

El diagrama de flujo del funcionamiento de la grabación de las repeticiones es el siguiente:

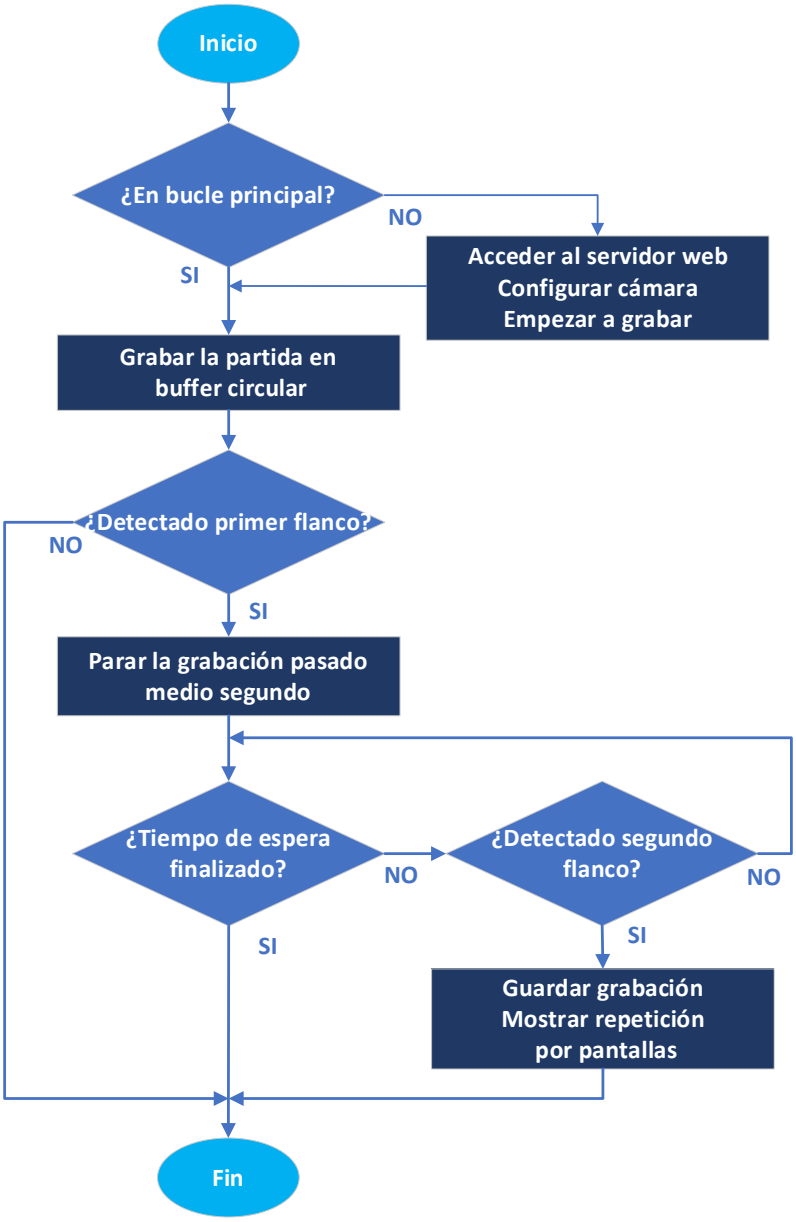


Fig.30. Diagrama de flujo de grabación de las repeticiones

Finalmente, se ha de resaltar que existe una relación directa entre la resolución grabada y los fotogramas por segundo. Si se quiere grabar a alta resolución se hará en detrimento de los fotogramas cuyo número será bajo y viceversa. Además, si se graba a baja resolución puede disminuir el campo de visión o FoV. Por ello, se ha grabado a 1080x720 ya que es una resolución que encuadra a todo el futbolín y que permite grabar a 90 FPS. (Para más información de las opciones de resolución, campo de visión FPS, revisar apartado 8 de anexos).

4.5.2 Visualización multimedia

Tanto el servidor web como las repeticiones de gol se muestran por las pantallas comentadas en el apartado 3.7.

Para visualizar la aplicación web, la Raspberry Pi se configura para conectarse al punto de acceso generado por el ESP32 [22]. Para conectarse únicamente se tiene que ir a redes wifi y poner el nombre del punto de acceso y su contraseña. Posteriormente, al iniciar el código Python se accederá a dicha aplicación web.

Cuando se marque un gol, se guarda la repetición y se reproduce en las pantallas. Para ver las repeticiones de gol, se usa el reproductor de video Omxplayer ya que fue específicamente creado para la GPU de raspberry y es el que viene por defecto.

Aunque siempre se graba los mismos segundos de video en directo, las repeticiones de gol pueden durar más o menos dependiendo de la velocidad a las que son reproducidas. La velocidad de reproducción es fácilmente modificable. Por comodidad en el testeo se visualizan las repeticiones a cámara rápida; en la práctica resultará más atractivo visualmente reproducirlas a cámara lenta.

Ilustración 23. Monitor con repetición de gol.

4.6 Relaciones entre capas y el fichero principal

Como es lógico, las capas anteriormente comentadas no son independientes las unas con las otras. Por ejemplo, para realizar la grabación de repeticiones de la capa de software multimedia, es necesario que la capa de firmware de futbolín registre un gol.

La comunicación entre capas la lleva el fichero principal con extensión .ino ubicado en el ESP32 y denominado "futbolín". Este fichero tiene la siguiente estructura:

1. Declaración de librerías, constantes y variables.
2. Instanciación de objetos de clases y creación de punteros de objeto.
3. Definición de funciones (funciones de interrupción, gestión del modo de juego, generación de pulsos para Raspberry, etc.).
4. Setup o configuración inicial. Aquí dentro encontramos:
   - 4.1 Inicializaciones del puerto serie, de pines, variables y objetos.
   - 4.2 Configuración y conexión a red wifi y generación de AP.
   - 4.3 Configuración de la dinámica del servidor web (almacenamiento de parámetros y funciones de respuesta a peticiones e inicialización).
   - 4.4 Conexión a plataforma IoT.
5. Loop o bucle
   - 5.1 Gestor de interrupciones.
   - 5.2 Verificación de monedero.
   - 5.3 Loop de cliente MQTT.

En el fichero principal es necesario diferenciar 3 partes: por un lado, está aquello que se ejecuta una sola vez, como es el caso de la definición de librerías, funciones, instanciación de objetos o la configuración de la dinámica del servidor web. Por otro, está el bucle que se ejecuta repetidamente de manera indefinida; y por otro lado, están las interrupciones que "saltan" y se ejecutan cada vez que se produce un cambio de estado en los pines a los que se le asocia la interrupción. Las interrupciones cada vez que salten habilitarán un flag para ser gestionadas en el bucle principal.

A continuación, un esquema que indica la comunicación entre capas y el fichero principal:

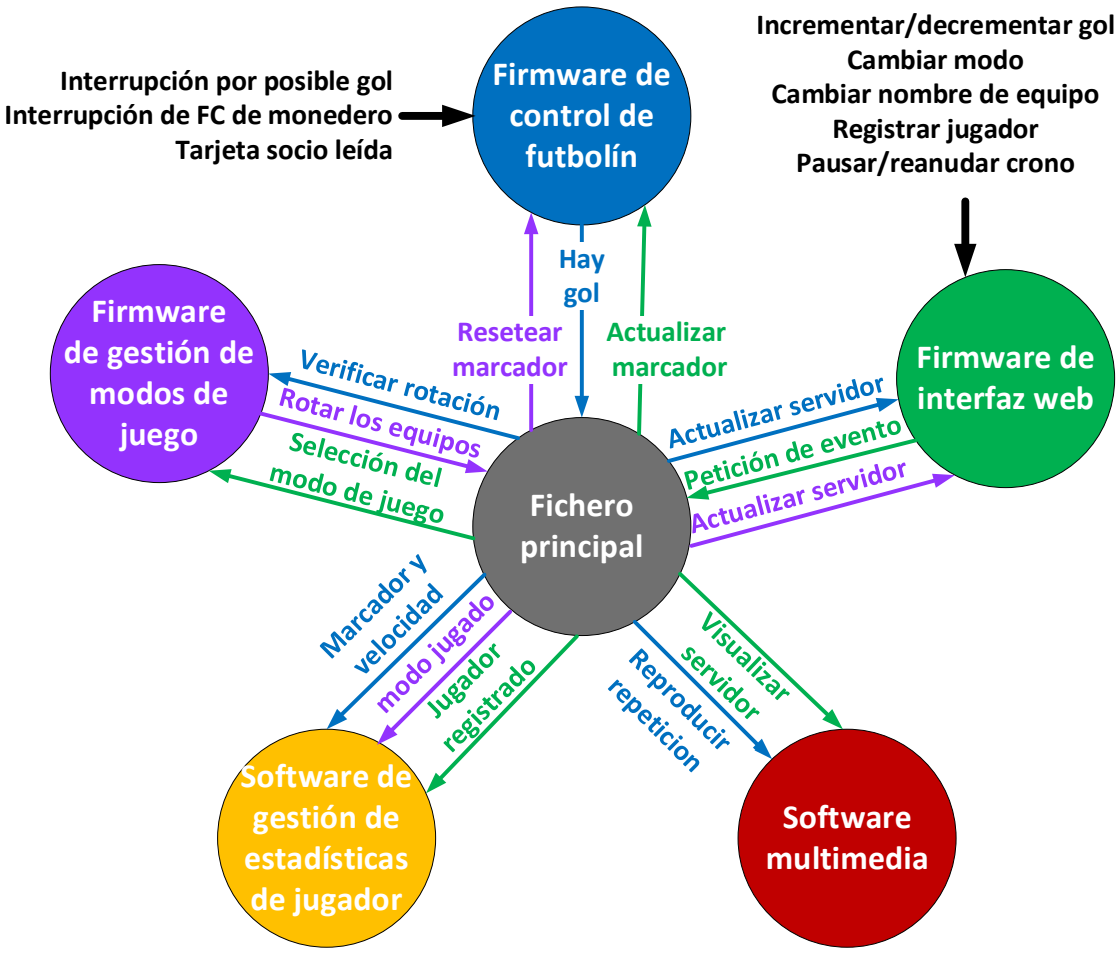


Fig. 31. esquema de relaciones entre capas

Como se observa en el esquema anterior, la capa de firmware de control de futbolín y la capa de firmware de interfaz web van a generar una salida en base a los datos de entrada que se indican.

Así pues, los datos de entrada correspondientes al firmware de interfaz web son los correspondientes a los distintos botones y parámetros que el usuario selecciona desde la interfaz web.

Respecto a los datos de entrada del firmware de control de futbolín, las entradas provienen de interrupciones y de la lectura de la tarjeta socio. Tanto para gestionar la interrupción como para conocer si se ha pagado con la tarjeta, las entradas derivan del bucle del fichero principal (más concretamente del gestor de interrupciones). Un diagrama de flujo del bucle principal podría ser el siguiente:

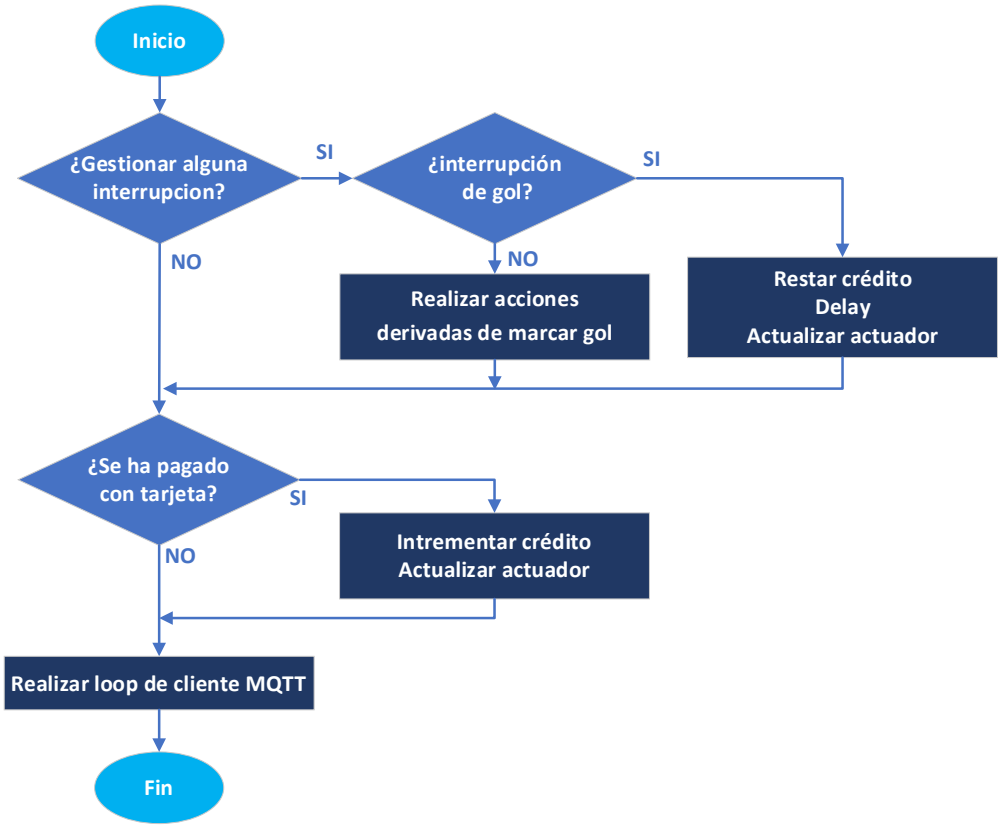


Fig. 32 Diagrama de flujo del bucle principal.

Si volvemos al esquema de relaciones entre capas, observamos como debido a unas entradas de las 2 capas anteriormente comentadas, se genera una salida que se gestiona en el fichero principal desencadenando unas entradas en las otras capas.

Por ejemplo, supongamos que el usuario selecciona un modo de juego distinto al que estaba seleccionado. En base a esa entrada, el firmware de interfaz web genera una petición de evento. En el fichero principal se especifica la respuesta ante dicha petición. Esa respuesta será un conjunto de acciones derivadas de cambiar de modo; como puede ser resetear el marcador del firmware de control de futbolín, seleccionar ese modo en el firmware de gestión de modo de juego o visualizar en la capa de software multimedia la aplicación web actualizada.

Otro ejemplo sería cuando el usuario, desde el firmware de interfaz web, se registra como jugador para obtener sus estadísticas. En ese caso, intervienen 3 capas: firmware de gestión de modo de juego, firmware de interfaz web y software de gestión de estadísticas de jugador (en verde en el siguiente esquema). Un esquema más detallado de este ejemplo podría ser el siguiente:

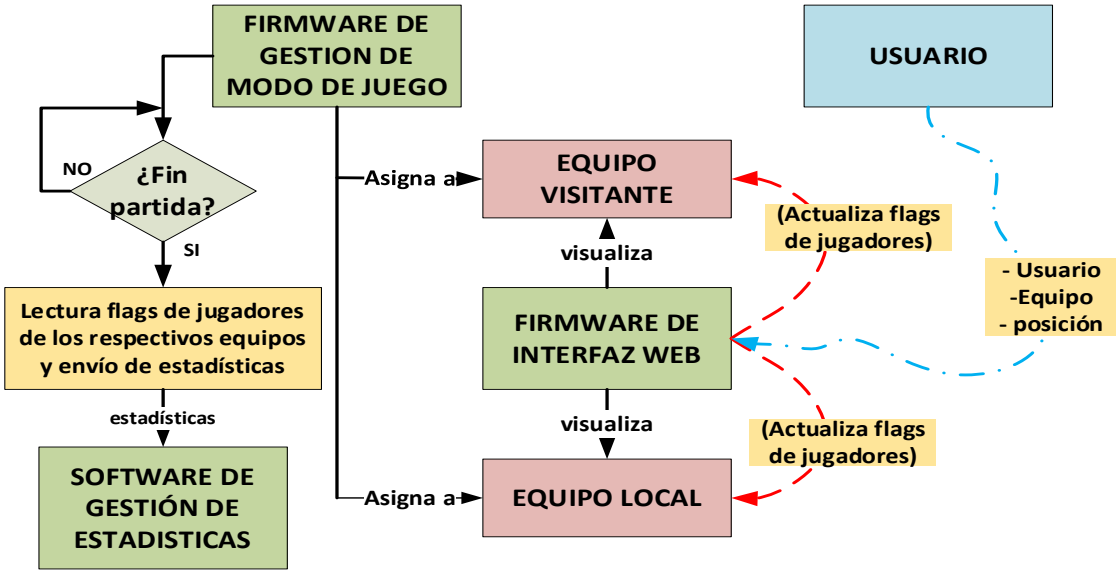


Fig. 33. Esquema de acciones derivadas de registrar un jugador

También resulta de interés conocer las acciones derivadas de la salida generada por la capa de firmware de control de futbolín al producirse un gol:

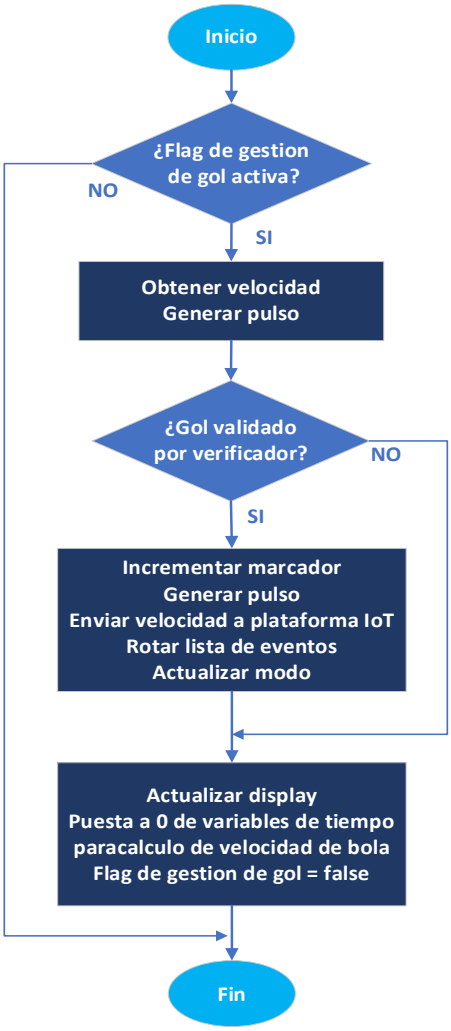


Fig.34. Diagrama de flujo de acciones derivadas de marcar gol.

Finalmente comentar que todo este punto 4 de especificaciones de software, se ha estudiado teniendo una visión por capas. Cada capa tiene una o varias funcionalidades diferentes de las demás. Sin embargo, hay otros posibles enfoques para desarrollar este punto. Un enfoque que se planteó, fue explicar el software por sus clases o bloques principales. Debido a que las clases están muy interconectadas y que, por tanto, pueden intervenir muchas clases para realizar una sola función, este enfoque fue descartado.

A continuación, un diagrama de las relaciones entre clases u otros bloques:

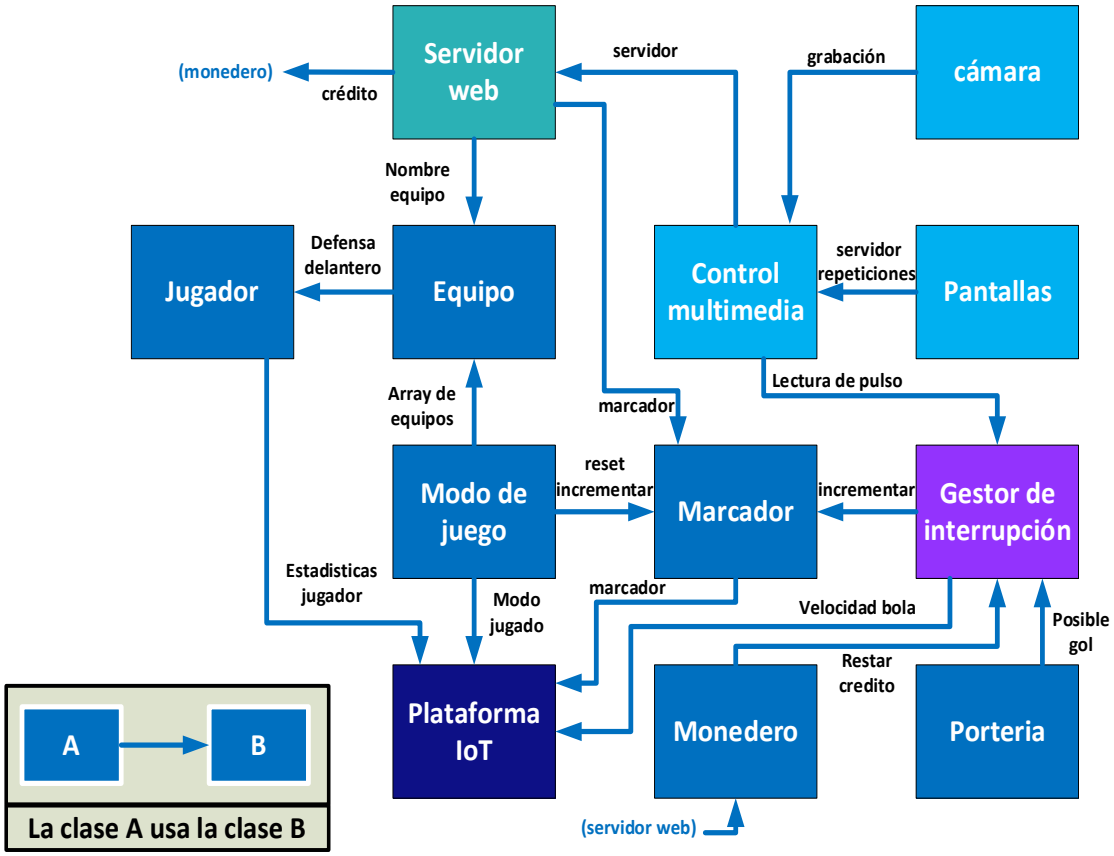


Fig.35. Diagrama de clases y bloques principales.

5. Conclusiones y Mejoras futuras

Mediante este proyecto, se han conseguido cumplir la mayoría de objetivos propuestos. Se ha integrado un sistema capaz de relevar al usuario tener que contar el número de goles marcados. Además, se han implementado modos de juego y sistemas de rotación que aumentan las posibilidades de juego.

Además, con 259 respuestas, se ha conseguido una gran participación en las encuestas, lo que ha posibilitado conocer que mejoras eran más atractivas para el usuario.

Se ha conseguido obtener una buena cantidad de muestras para estudiar la medida de velocidad mediante impacto pese a no obtenerse unos resultados concluyentes.

Se ha diseñado un producto modular, donde el cliente podrá seleccionar aquellas características de su futbolín que más le resulta conveniente. Además, se han tenido bastante en cuenta el coste de fabricación, obteniéndose así, un producto muy competente e innovador en el mercado.

Se ha conseguido un prototipo funcional y eficaz donde todos los componentes electrónicos se han integrado satisfactoriamente en el futbolín. Todos han sido probados y optimizados para obtener unos resultados bastante buenos.

Como valoración personal, la realización de este TFG me ha permitido comprender y utilizar los distintos lenguajes de programación. Asimismo, he asimilado conocimientos relativos a los distintos protocolos de comunicación, plataformas IoT, sistemas operativos, uso de librerías, diseño y fabricación de piezas 3D, etc. He podido materializar las ideas de mejora en un juego muy valorado y utilizado por un gran público de todas las edades: la esencia del juego sigue siendo la misma y el jugador experimenta una mayor satisfacción en la experiencia de juego por los cambios realizados.

El presente trabajo de fin de grado es el primer paso para otras posibles mejoras en el futuro. A partir del futbolín tradicional de madera, se han añadido mejoras electrónicas y se han pospuesto aquellas que por su extensión, dificultad o naturaleza no han podido ser implementadas. A continuación, comentaremos brevemente una serie de mejoras que podrían resultar bastante interesantes en futuros trabajos.

5.1 Sensado de velocidad y posición mediante acelerómetros

Como ya se vio antes en el ensayo de tiradas, no se pudieron obtener resultados concluyentes. Por ello queda pendiente realizar un estudio más detallado del ensayo en unas condiciones más favorables. El objetivo sería sensar la velocidad y el punto de impacto únicamente mediante dos acelerómetros. Las barreras ópticas sólo se usarían en los ensayos para comparar el valor de velocidad medido con el estimado a partir de las lecturas de impacto de los acelerómetros.

5.2 Implementación de una base de datos para gestionar las estadísticas de los distintos jugadores de futbolín

Pese a que el futbolín permite gestionar los jugadores y enviar sus estadísticas a la plataforma IoT, éstas son únicamente visualizadas. Como posible mejora, se podría gestionar las estadísticas para que el usuario pudiera acceder a ellas en todo momento.

5.3 Diseño de una aplicación móvil

Mediante una aplicación móvil, el usuario podría acceder y controlar el futbolín, visualizar sus estadísticas, descargarse las repeticiones de gol o incluso, se podría utilizar para realizar micropagos a una cuenta bancaria asociada al futbolín.

5.4 Desarrollo de una interfaz multimedia en Raspberry

La interfaz gráfica de usuario que ha sido desarrollada, se trata de un servidor web gestionado enteramente por el microcontrolador. Como mejora futura, se puede realizar una interfaz en Raspberry que sea capaz de gestionar el servidor web; y que también se pudieran añadir funcionalidades multimedia como pequeños clips de audio y video. Por ejemplo, una animación de gol, un árbitro pitando los finales de partido, revisando el video arbitraje (VAR), etc.

5.5 Diseño de un prototipo para el dispensado de bolas de futbolín

El monedero mecánico, pese a que ha sido adaptado, podría mejorarse sustituyéndose por uno electrónico. De esta forma, se podrían realizar pagos con diferentes tipos de moneda. El problema de implementar un monedero electrónico es la necesidad de un sistema que permita dispensar las bolas. Por ello, se tendría que diseñar un prototipo capaz de dispensar el número de bolas.

5.6 Elaboración de una fuente de alimentación de futbolín

Como se vio en el apartado 3.8, la fuente de alimentación del futbolín se basa en una regleta que alimenta las distintas fuentes. Por motivos de disminución de coste y eficiencia energética, se debería diseñar y elaborar una fuente que cumpla con los requerimientos de tensión y corriente solicitados por los distintos elementos del futbolín.

5.7 Identificación y seguimiento mediante una cámara inteligente de las bolas jugadas en el futbolín

El objetivo sería utilizar una cámara de Raspberry Pi para detectar la posición de la bola, velocidad y trayectoria. Esto permitiría implementar estadísticas de juego adicionales (tiros a puerta, posesión, pases, etc.). Adicionalmente, se podría realizar un software que fuera capaz de detectar acciones no permitidas como “ruletas” o “jugadas”.

6. Bibliografía

- [1] Wikipedia "futbolín" (13 de agosto de 2005). Disponible en: https://es.wikipedia.org/wiki/Futbol%C3%ADn [Accedido el 1 de septiembre de 2020].

- [2] Santiago Real "Customized foosball", 9 julio de 2018 [Online]. Disponible en: http://elb105.com/the-twelve-of-b105-customized-foosball/ [Accedido el 1 de septiembre de 2020].

- [3] Espressif "Datasheet DevKit ESP32-WROOM 32", agosto de 2019 [Online]. Disponible en: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf [Accedido el 1 de septiembre de 2020].

- [4] Página comercial de fotointerruptores, sin fecha [Online]. Obtenida de: https://es.aliexpress.com/item/32998692609.html [Accedido el 1 de septiembre de 2020].

- [5] Invensense "datasheet MPU 9250", 9 de septiembre de 2019 [Online]. Disponible en: https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/MPU-9250-Register-Map.pdf [Accedido el 1 de septiembre de 2020].

- [6] SG90 "Datasheet SG90", sin fecha [Online]. Disponible en: http://www.datasheetcafe.com/wp-content/uploads/2015/12/SG90.pdf [Accedido el 1 de septiembre de 2020].

- [7] NXP Semiconductors "Datasheet RFID MRC522", 27 de abril de 2019 [Online]. Disponible en: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf [Accedido el 1 de septiembre de 2020].

- [8] Handson Technology "Datasheet MB102 Breadboard", 1 de enero de 2015 [Online]. Disponible en: https://components101.com/sites/default/files/component_datasheet/MB102-Datasheet.pdf [Accedido el 1 de septiembre de 2020].

- [9] Blog de WordPress "Crear Librerías Arduino", 9 de julio de 2019 [Online]. Disponible en: https://aprendiendoarduino.wordpress.com/2018/07/09/crear-librerias-arduino/ [Accedido el 1 de septiembre de 2020].

- [10] Blog de WordPress "Configuración y manejo de interrupciones GPIO ESP32 en Arduino IDE", 1 de Noviembre 2019 [Online]. Disponible en: https://aprendiendoarduino.wordpress.com/2018/07/09/crear-librerias-arduino/ [Accedido el 1 de septiembre de 2020].

- [11] flybrianfly "Librería MPU9250", 20 de Mayo de 2020 [Online]. Disponible en: https://github.com/bolderflight/MPU9250 [Accedido el 1 de septiembre de 2020].

- [12] gsampampallo "Librería serialToExcel", 3 de agosto de 2020 [Online]. Disponible en: https://github.com/gsampallo/serialToExcel [Accedido el 1 de septiembre de 2020].

- [13] Librería ServoESP32, 17 de julio de 2020 [Online]. Disponible en: https://github.com/RoboticsBrno/ServoESP32 [Accedido el 1 de septiembre de 2020].

- [14] JarekParal "Librería RFID-RC52", 24 de Julio de 2020 [Online]. Disponible en: https://github.com/miguelbalboa/rfid [Accedido el 1 de septiembre de 2020].

- [15] avishorp "Librería TM1637", 30 de octubre de 2018 [Online]. Disponible en: https://github.com/avishorp/TM1637 [Accedido el 1 de septiembre de 2020].

- [16] Sara Santos "Instalar el cargador del sistema de archivos ESP32 en Arduino IDE", 5 de Julio de 2019 [Online]. Disponible en: https://randomnerdtutorials.com/install-esp32-filesystem-uploader-arduino-ide/ [Accedido el 1 de septiembre de 2020].

- [17] me-no-dev "Librería ESPAsyncWebServer", 17 de octubre de 2019 [Online]. Disponible en: https://github.com/me-no-dev/ESPAsyncWebServer [Accedido el 1 de septiembre de 2020] .

- [18] Techtutorialsx "Servidor HTTP ESP32 Arduino: obtener parámetros", 17 de diciembre de 2017 [Online]. Disponible en: https://techtutorialsx.com/2017/12/17/esp32-arduino-http-server-getting-query-parameters/ [Accedido el 1 de septiembre de 2020].

- [19] knolleary "Librería MQTT pubsubclient", 20 de mayo de 2020 [Online]. Disponible en: https://github.com/knolleary/pubsubclient [Accedido el 1 de septiembre de 2020].

- [20] Thingsboard "Referencia de API de Thingsboard", Año 2020 [Online]. Disponible en: https://thingsboard.io/docs/reference/mqtt-api/ [Accedido el 1 de septiembre de 2020].

- [21] Dave Jones "Documentación PiCamera", 28 de abril de 2020 [Online]. Disponible en: https://readthedocs.org/projects/picamera/downloads/pdf/latest/ [Accedido el 1 de septiembre de 2020].

- [22] Sara Santos "Configurar un punto de acceso (AP)", 23 de abril de 2019 [Online]. Disponible en: https://randomnerdtutorials.com/esp32-access-point-ap-web-server/ [Accedido el 1 de septiembre de 2020].

- [A23] Wikipedia "I2C", 19 de junio de 2020 [Online]. Disponible en: https://es.wikipedia.org/wiki/I%C2%B2C [Accedido el 1 de septiembre de 2020].

- [A24] Wikipedia "SPI", 11 de febrero de 2020 [Online]. Disponible en https://es.wikipedia.org/wiki/Serial_Peripheral_Interface [Accedido el 1 de septiembre de 2020].

- [A25] Wikipedia "UART",16 de Enero de 2016 [Online]. Disponible en: https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter [Accedido el 1 de septiembre de 2020].

- [A26] Luis llamas "¿Qué es MQTT? su importancia como protocolo IoT (17 de abril de 2019). Disponible en: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/ [Accedido el 1 de septiembre de 2020].

ANEXOS

Trabajo Fin de Grado

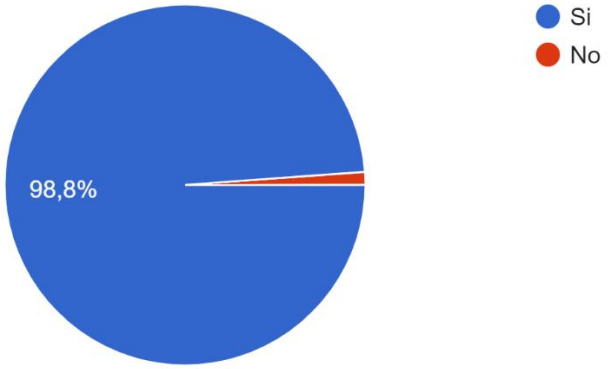
Futbolín electrónico
Electronic foosball

Escuela de ingeniería y arquitectura
2020

Anexos 1. Resultados de estudio de mercado

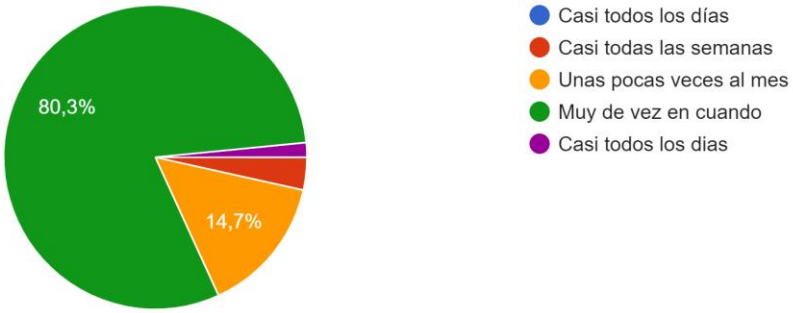
¿Ha jugado o sabe jugar al futbolín?

259 respuestas



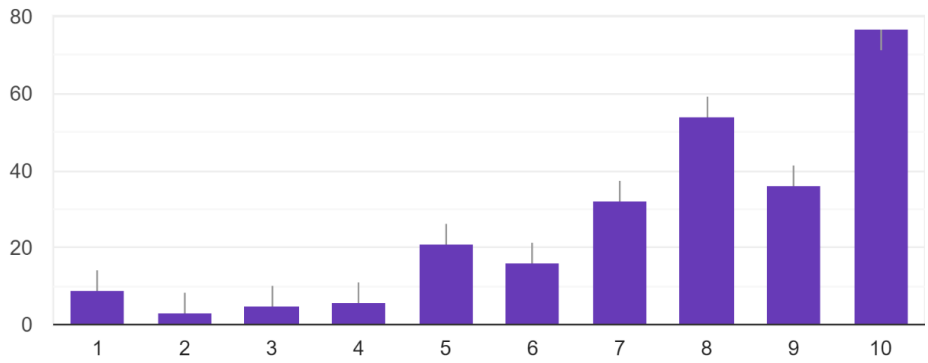
¿Con qué frecuencia juega al futbolín?

259 respuestas



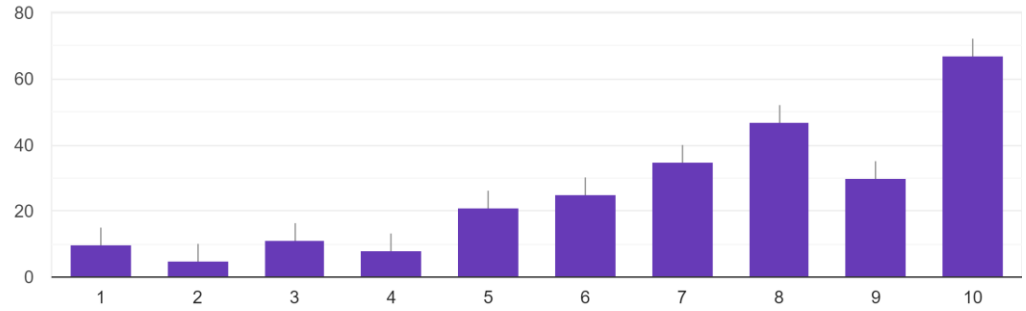
¿Considera interesante incorporar un marcador electrónico que permita contar los goles automáticamente (también manualmente)?

259 respuestas



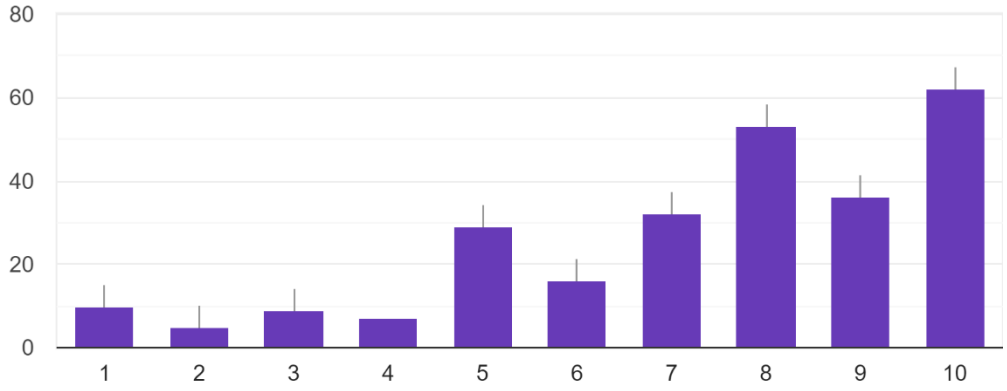
¿Le gustaría que el futbolín en ocasiones especiales tipo goles, paradas, inicio-fin de partida realizara un sonido acorde a dicha situación?

259 respuestas



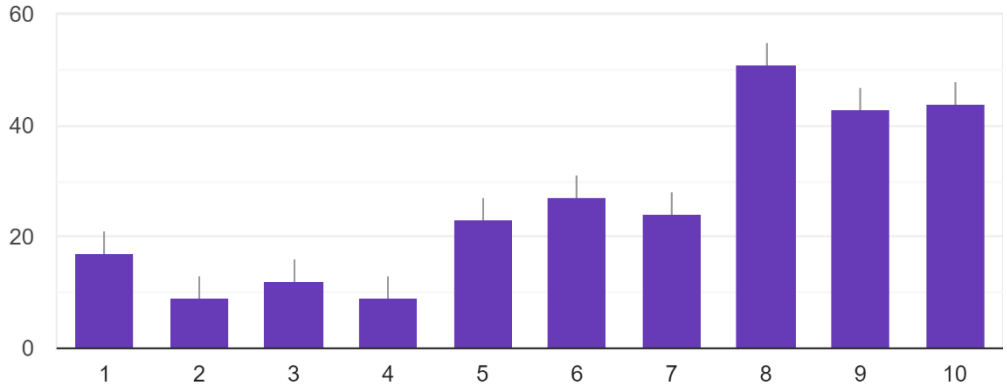
¿Una iluminación del terreno de juego regulable es una mejora...?

259 respuestas

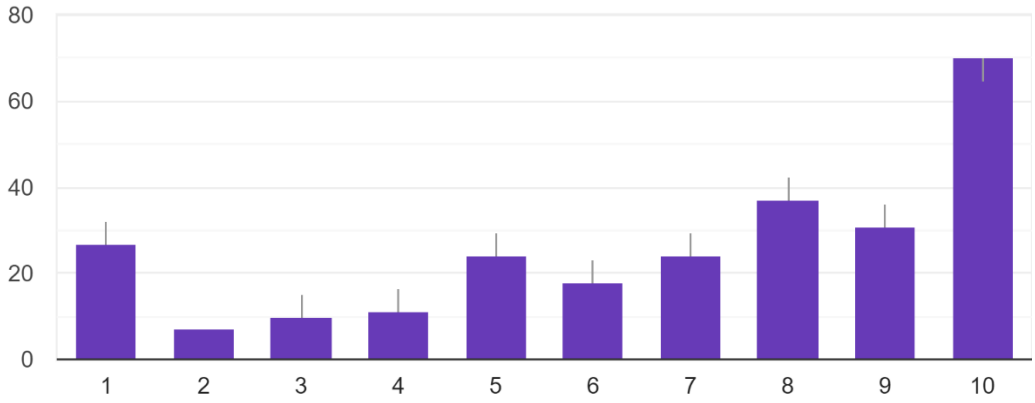


¿Le parecería interesante poder ver las estadísticas de juego (goles, tiros a puerta, tiros fuera, posesión, velocidad...en una pantalla externa tipo televisor?

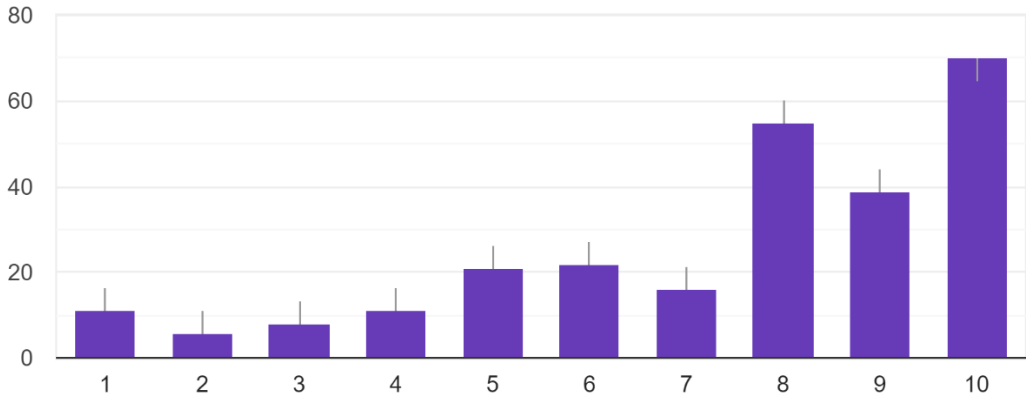
259 respuestas



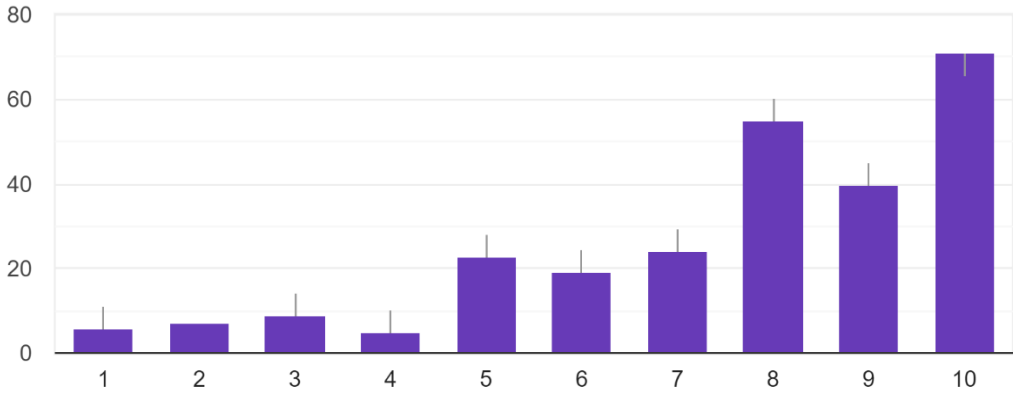
¿Y ver las repeticiones de gol a cámara lenta en la pantalla externa?
259 respuestas



¿Considera de interés la posibilidad de habilitar/deshabilitar un sistema automatizado para detectar jugadas no permitidas (“ruleta”, “jugada”...)?
259 respuestas

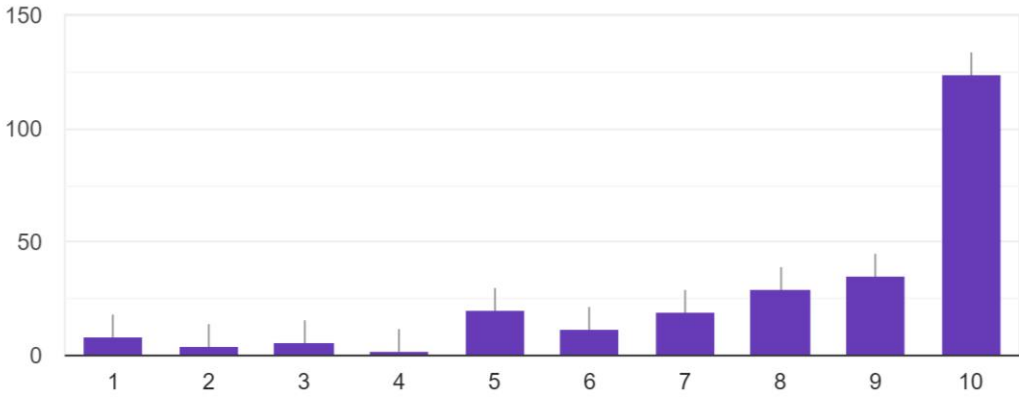


¿Qué le parecería poder hacer torneos y liguillas pudiendo ver los resultados en una pantalla externa o aplicación móvil?
259 respuestas



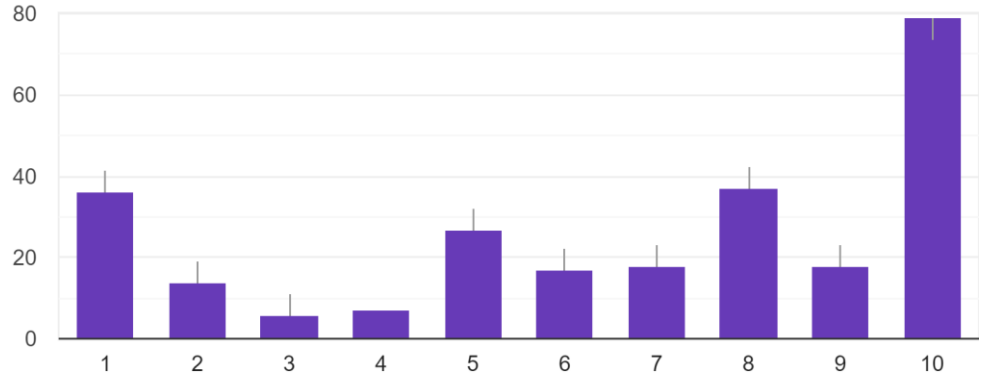
Si pudieras pagar con varios valores de moneda, ¿te parecería relevante que se devolviese el cambio?

259 respuestas



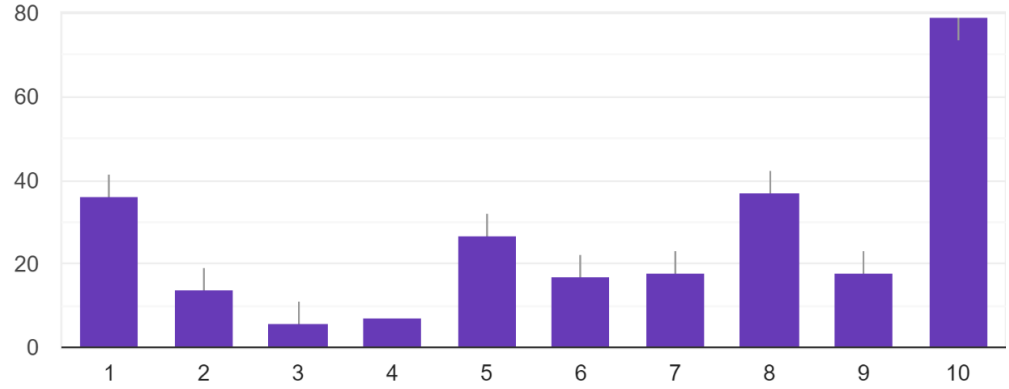
¿Le gustaría tener la posibilidad de pagar con tarjeta?

259 respuestas



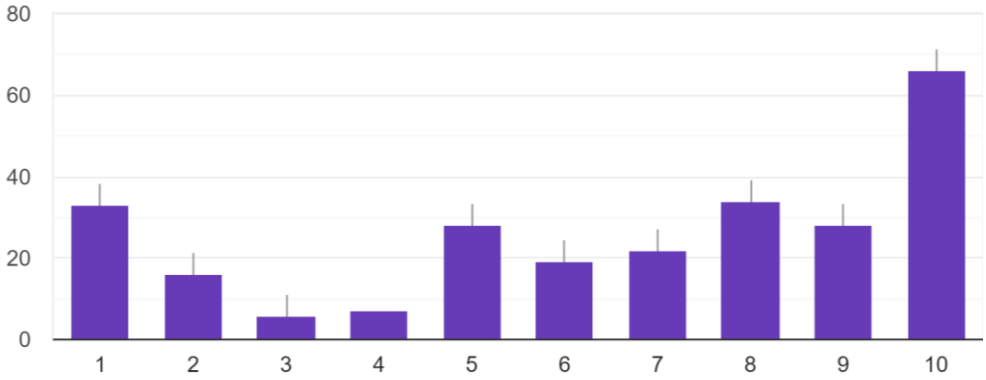
¿Le gustaría poder pagar con tarjeta?

259 respuestas



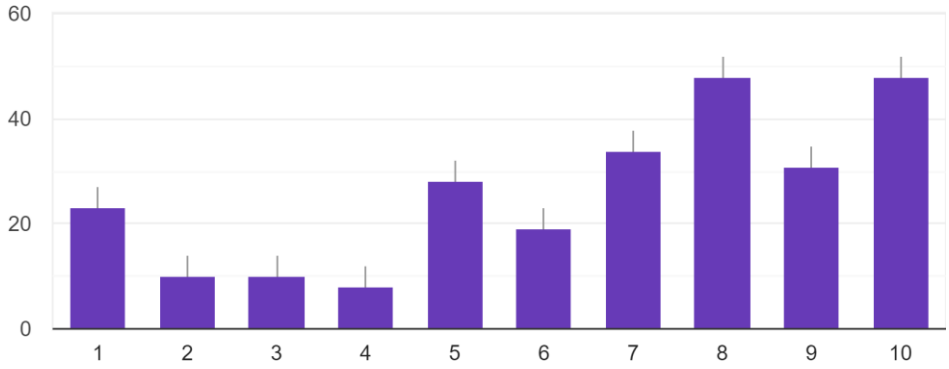
¿Y con una app con saldo en el móvil?

259 respuestas



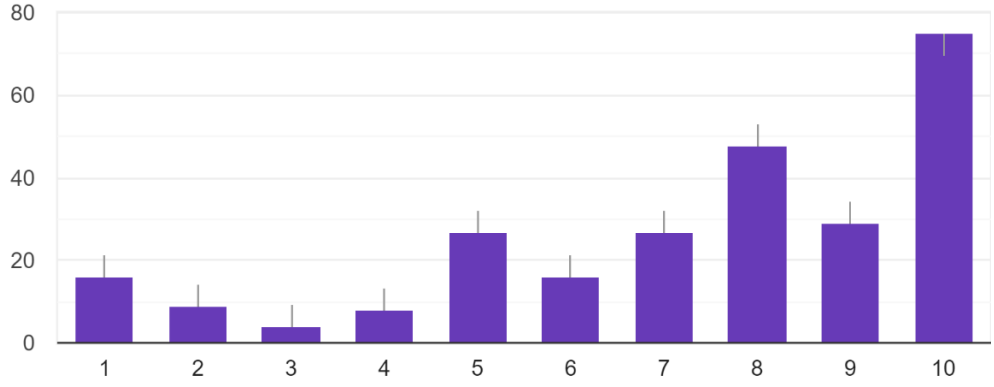
¿Qué le parecería poder jugar partidas en las que se rota tras acabar un tiempo determinado (y no por goles)?

259 respuestas



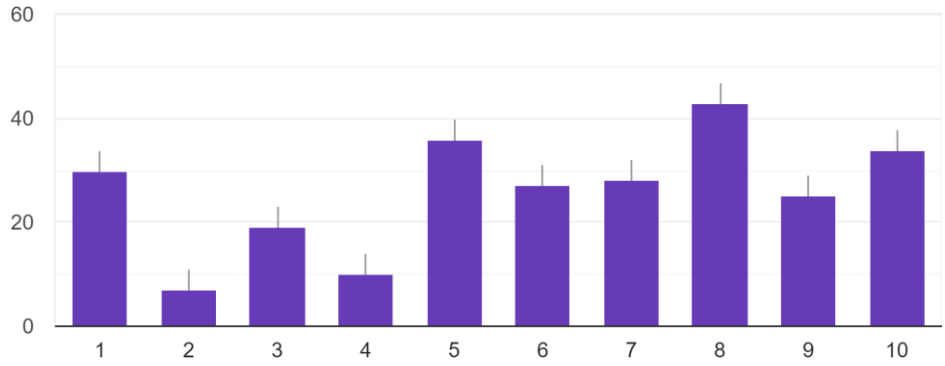
¿Cree interesante poder seleccionar el número de bolas a jugar y pagar en correspondencia a ese número?

259 respuestas



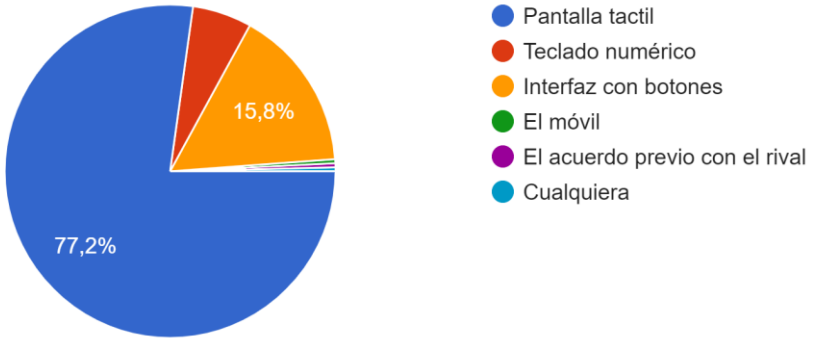
¿Le gustaría tener una aplicación móvil para consultar sus estadísticas de juego?

259 respuestas



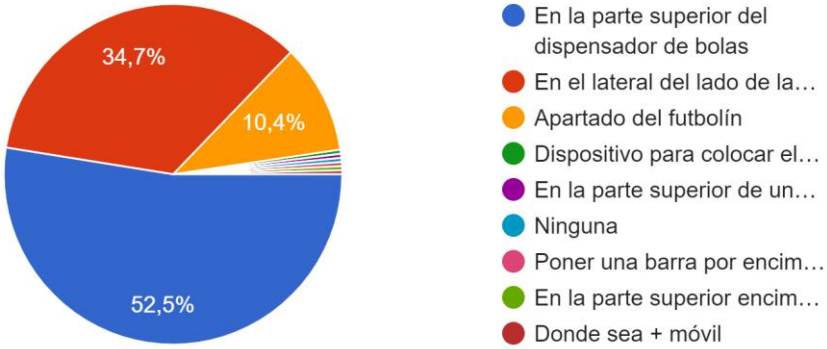
Para seleccionar y modificar las distintas opciones de juego,¿cuál cree que es el medio más adecuado?

259 respuestas

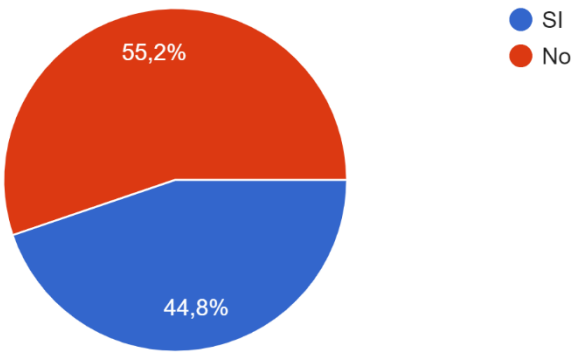


¿Y cuál sería la ubicación más adecuada?

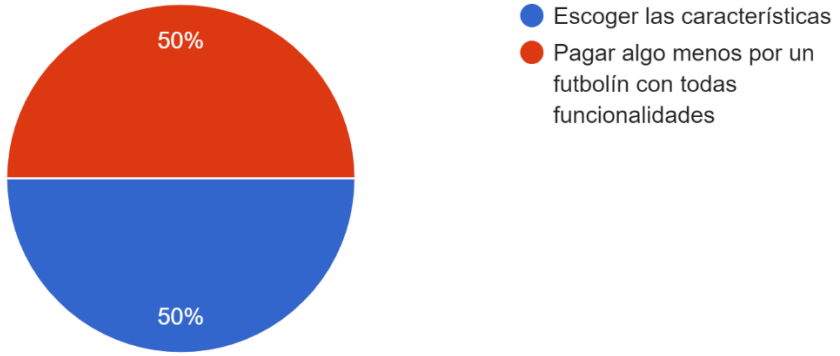
259 respuestas



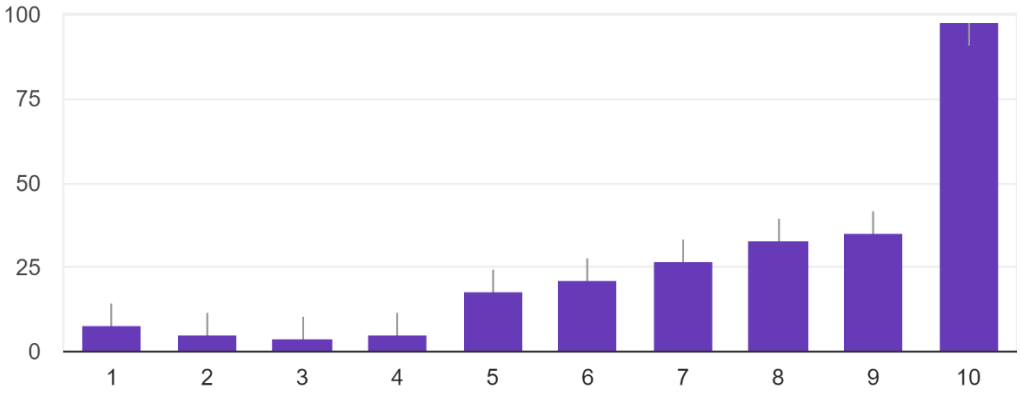
¿Se plantearía comprar un futbolín electrónico o cualquier otro futbolín?
259 respuestas



¿Preferiría comprar un futbolín pudiendo escoger las características o comprar un modelo concreto?
254 respuestas



¿Como propietario cree interesante disponer de una aplicación móvil que le notifique el dinero recaudado, necesidades de mantenimiento, errores...)?
254 respuestas



Anexos 2. Características ESP32-WROOM-32

- Voltaje de operación 2.2-3.6V.
- Temperatura de operación: -40 a +85 C.
- Velocidad de reloj: Entre 160 Mhz y 240 Mhz.
- 520 Kb de RAM.
- 8 KBytes SRAM en RTC SLOW.
- 8 KBytes SRAM en RTC FAST.
- 36 pins GPIO.
- Wifi integrado: con acces point & Station.
- Bluetooth 4.2 2.4 Ghz; BT 2.0 y 4.0 BLE.
- Procesador dual core Xtensa® LX6 de 32 bits.
- 2 Digital to Analog Converter DAC de 8 Bits.
- 16 Analog to digital ADC de 12 bits..
- 2 puertas series UART.
- 2 canales I2C.
- 4 canales SPI (aunque solo 2 están operativas actualmente).
- 16 canales PWM.
- Interfaces: SD-card, SDIO, I2S, LED PWM, Motor PWM, IR, GPIO, sensor táctil capacitivo, sensor Hall, sensor de temperatura.

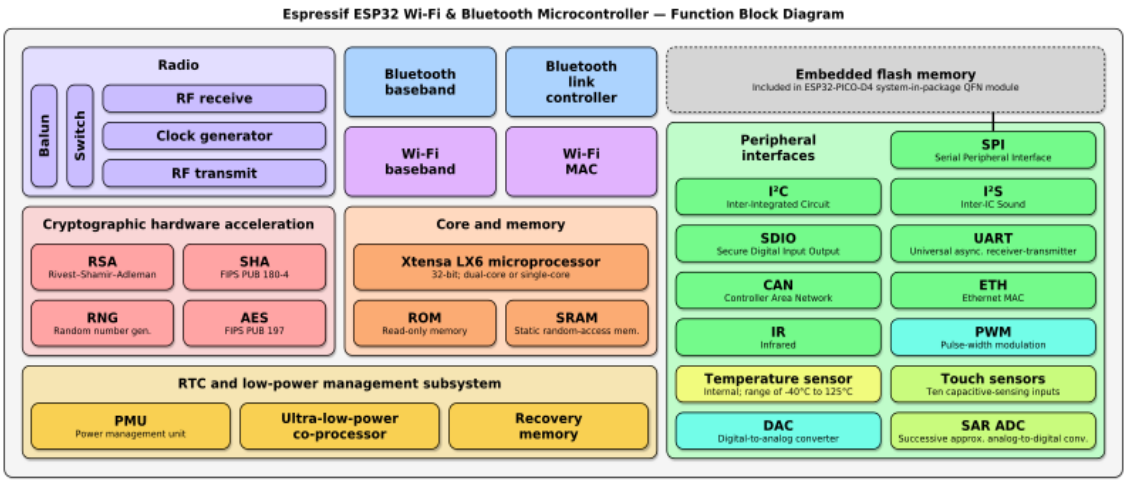


Fig. 36. Diagrama de bloques de funciones de ESP32 (imagen sin copyright).

Anexos 3. Protocolos de comunicación empleados

En la memoria se comentaron conceptos relativos a tipologías de bus y protocolos de comunicación (o estándar de comunicación). La diferencia entre tipología de bus y protocolo de comunicación, es que la primera es el medio físico donde se transmiten los datos; mientras que la segunda es la forma en la que se comunican.

Un diagrama de los diferentes protocolos de comunicación utilizados por los distintos bloques del futbolín podría ser:

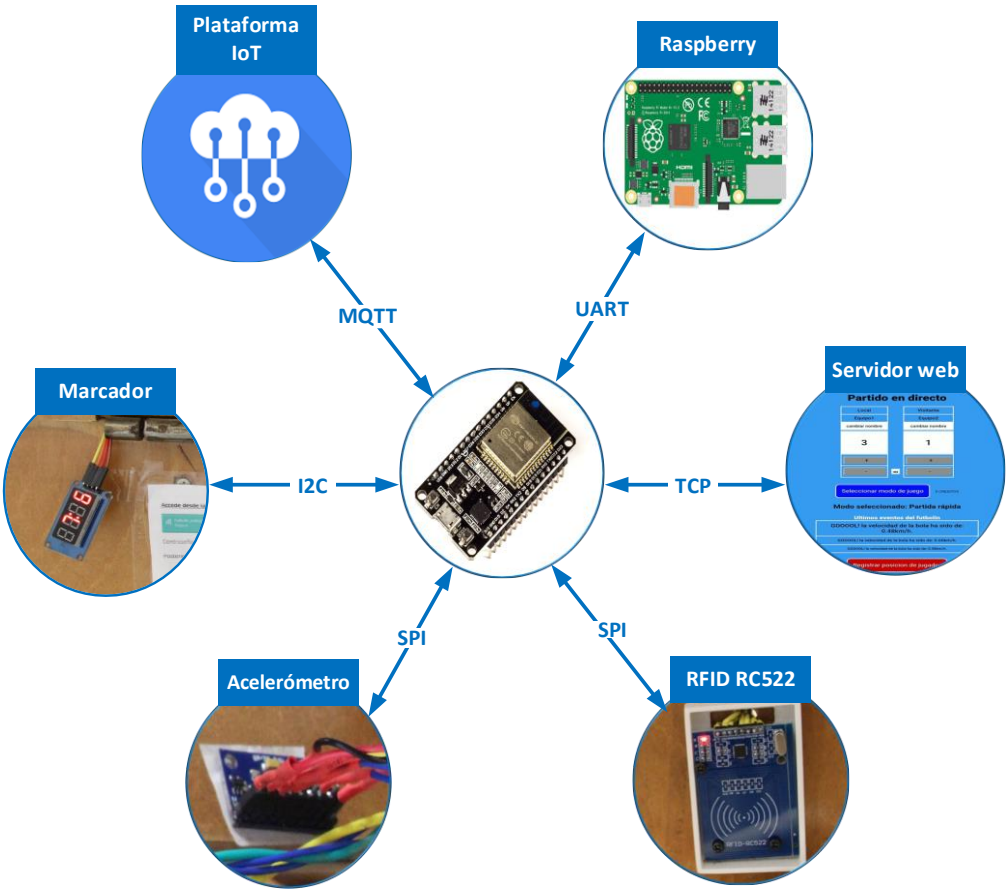


Fig. 37. Diagrama de componentes y protocolos de comunicación

A continuación, pasaremos a explicar brevemente las características de los distintos protocolos de comunicación.

Anexos 3.1. I2C

La abreviatura hace referencia a “inter integrated circuit”. Es un tipo de bus empleado para conectar varios circuitos integrados en los que se hace uso de dos líneas de señal y un común o masa.

Basándonos en [A23], SPI es un bus con tipología maestro-esclavo. Por maestro, nos referimos al dispositivo capaz de iniciar una comunicación (determina los tiempos de tráfico y la dirección de bus); y por esclavo, al dispositivo que recibe y reacciona a las señales del maestro (es incapaz de generar pulsos de reloj).

La tipología I2C puede tener múltiples maestros, es decir, se puede conectar varios chips al mismo bus y todos ellos pueden funcionar como maestro.

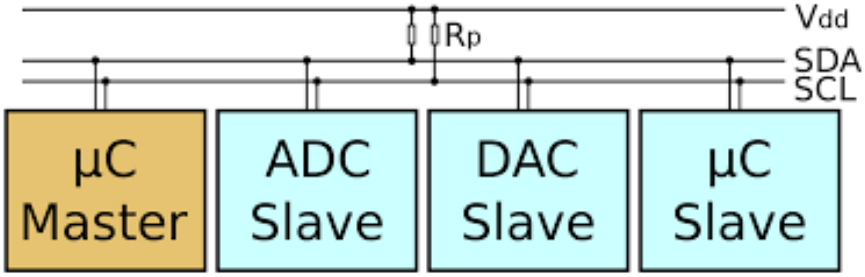


Fig. 38. Ejemplo de maestro con tres esclavos I2C (imagen sin copyright).

Como se observa en la figura anterior, existen dos líneas de datos: reloj (SCLK) y la línea de datos (SDA). Ambas líneas precisan de dos resistencias de “pull up”.

Cada circuito integrado dispone de una dirección que servirá como identificador. El maestro enviará un byte con la dirección del esclavo con el que quiere establecer una comunicación. Si el esclavo se identifica con esa dirección, contesta con un bit en bajo y ambos inician la comunicación.

Como característica principal de I2C decir que sólo requiere dos pines, lo que es una gran ventaja para conectar otros dispositivos al microcontrolador. Otra característica es su tasa de transferencia de datos de hasta 400 KHz. Finalmente, se ha de comentar que la longitud de los cables que trabajen con este tipo de bus es limitada; se pueden alcanzar longitudes de hasta 50 m, pero todo dependiendo de la frecuencia de trabajo y tipología de cable.

Anexos 3.2 SPI

Según [A24], la abreviatura corresponde a "Serial Peripheral Interface" el bus SPI es un estándar para controlar casi cualquier dispositivo electrónico capaz de gestionar un flujo de bits serie regulado por un reloj.

Al igual que con el caso anterior, la tipología de bus SPI, también es una tipología maestro-esclavo.

Además de I2C, SPI es un protocolo síncrono, ya que los datos entre el emisor y el receptor se transmiten de manera consecutiva, con un flujo constante determinado por la señal del reloj.

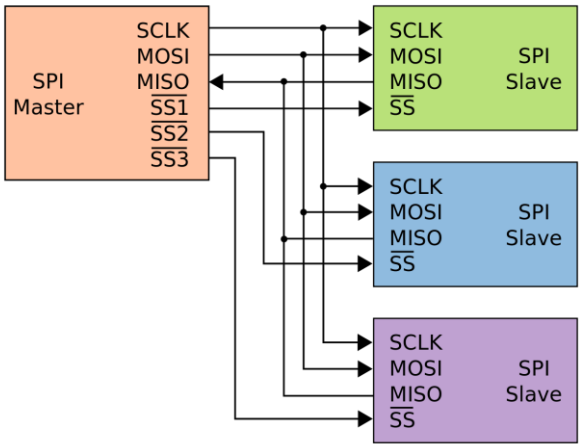


Fig. 39. Ejemplo de maestro con tres esclavos SPI (imagen sin copyright).

Como se observa en la imagen anterior, existen tres líneas de datos comunes para cada esclavo y una adicional por cada esclavo Las líneas son:

- SCLK: Es el bus encargado de la sincronización, con cada pulso de este reloj se envía o recibe un bit.

- MOSI y MISO: son las líneas de salida de datos entre maestro-esclavo y esclavo-maestro respectivamente.

- SS: llamado también CS (chip select), es la línea para seleccionar o activar un esclavo.

SPI tiene la gran ventaja de tener una alta tasa de transferencia de datos de en torno a 10 MHz. Por el contrario, requiere de más pines para su utilización.

Anexos 3.3. TCP/IP

La abreviatura significa "Protocolo de control de transmisión/Protocolo de Internet". TCP/IP es un conjunto de protocolos que permiten la comunicación entre distintos elementos conectados a una red.

Se trata de un software de comunicación complejo en el que, para conseguir un intercambio fiable de datos, se deben realizar muchos procedimientos separados.

Anexos 3.4 UART

Su abreviatura significa "recepción-transmisión de datos universal". UART es de los protocolos serie más utilizados [A25]. El hardware UART, permite controlar puertos y dispositivos serie.

Al contrario que con I2C o SPI que son protocolos síncronos, UART es un protocolo asíncrono, es decir que no está sincronizado con una señal de reloj.

UART usa una línea de datos simple para transmitir y otra para recibir datos. Tanto el formato de los datos como la velocidad de transmisión son configurables.

Normalmente, se suelen utilizar dispositivos de interfaz separados para convertir las señales de nivel lógico del UART. Esto se realiza hacia y desde los niveles de señalización externos como los utilizados para señalización por voltaje RS-232, RS-422 o RS-485.

Anexos 3.5 MQTT

MQTT son las siglas de "Message Queuing Telemetry Transport". Como se indica en [A26], es un protocolo máquina a máquina (M2M) de tipo de mensajes queue. Está basado en TCP/IP, aunque en este caso la conexión se mantiene abierta y se "reutiliza" hasta que el cliente la finaliza.

Los clientes se conectan a un servidor central llamado "broker", y los mensajes se envían a cada cliente y se disponen en "topics" organizados jerárquicamente. Un cliente puede subscribirse a ese "topic" y el broker le enviará los mensajes suscritos.

Como características de este protocolo, se han de destacar su ligereza, robustez, fiabilidad y sencillez. Además, requiere de muy poco ancho de banda. Como punto negativo, destacaría que, al ser un protocolo tan ligero, la seguridad no es muy elevada (aunque dispone de medidas adicionales como la seguridad y calidad del servicio QoS).

Anexos 4. Análisis de alternativas para el conteo de goles

Existen un gran número de sensores para una alta diversidad de aplicaciones. Por ello, nos centraremos en aquellos cuyo funcionamiento y aplicación son los más adecuados para la tarea a realizar. Por tanto, los sensores de distancia, de sonido o de fibra óptica, pese a que podrían cumplir los requisitos necesarios para sensar el gol, no se analizarán ya que no han sido diseñados para la funcionalidad que se desea.

Para detectar el gol, se ha de utilizar un dispositivo que mediante el paso o contacto de la bola actué en consecuencia. Por ello, debería ubicarse en la canaleta que dirige la bola al depósito donde se encuentran las otras (con las restricciones de tamaño a tener en cuenta).

Anexos 4.1 Sensores de proximidad

Este tipo de sensores consisten en transductores que detectan los objetos que se encuentran cerca de un elemento sensor. Según el principio físico que utilizan, podemos distinguir varios tipos entre los que destacan: los detectores capacitivos, los inductivos y los fotoeléctricos o infrarrojos.

Sensor capacitivo

Su función es detectar cambios de estado midiendo una variación de capacitancia. Están compuestos por un oscilador cuya capacidad la forman un electrodo interno (parte del propio sensor), y otro externo (pieza conectada a la masa). La bola a detectar se utiliza como dieléctrico que se introduce entre la masa y la placa activa, variando así las características del condensador equivalente.

Sus ventajas son: detectar los objetos metálicos y no metálicos, sensar sin contacto físico y ser bastante robustos y fiables.

Por el contrario, el alcance es bastante limitado; depende del diámetro del sensor y puede alcanzar hasta 6 cm. Además, al depender del objeto a sensar se la asocia una constante eléctrica cuya función es ajustar la sensibilidad del sensor (típicamente con un potenciómetro).

Otra desventaja es el precio, que puede variar mucho dependiendo de la distancia de detección. Algunos ejemplos comerciales son:

- Sensor LJC30A3-H-Z (Max 10mm): 3,83€ + 2,61€ g.d.e.
- Sensor CJM30-10A2-S (Max 10mm): 10,41€.
- Sensor KI6000 - KI-3250NFPK (Max 25mm): 83,13€.

Ilustración 24. Sensor capacitivo ifm electronic KI6000 - KI-3250NFPKG. Recuperado de: https://www.automation24.es/sensor-capacitivo-ifm-electronic-ki6000-ki-3250nfpkg-pl-2p-us-io

Sensor fotoeléctrico

Los llamados sensores fotoeléctricos o fotocélulas son dispositivos que responden al cambio de intensidad de luz, por lo que requieren de un emisor para generar la luz y de un receptor para detectarla. Una vez recibidas, se utiliza un transductor para convertir la luz a una señal eléctrica. Existen tres tipos de sensores fotoeléctricos:

Fotocélulas de barrera de luz.

El emisor y el receptor están separados en cuerpos distintos quedando ambos enfrentados.

Tiene la ventaja de ser el método más fiable ya que la totalidad de la potencia emitida es enviada al receptor, es apto para condiciones ambientales desfavorables, obtienes unas distancias de funcionamiento elevadas (unos 60m), pero por el contrario necesitan calibrarse de manera precisa.

Fotocélulas autorreflexivas

El emisor y el receptor están dentro de la misma carcasa; la luz emitida incide en el objeto y es el receptor el encargado de detectarla.

Es económico pero el menos adecuado de las fotocélulas para trabajar en ambientes desfavorables, ya que la suciedad, polvo o humedad puede inhabilitar la detección de la fotocélula.

Pueden a su vez subdividirse en tres. Si se le añade la técnica de supresión de fondo, permite ignorar objetos situados detrás del objeto de sensado (añadiendo técnicas de triangulación para calcular la posición exacta del objeto), o con técnicas de supresión de primer plano, cuyo principio es similar al anterior pero el ajuste se hace apuntando a la superficie de fondo y delimitando la zona de detección (usual en cintas transportadoras).

Fotocélulas reflexivas con reflector

Al igual que las anteriores, el receptor y el emisor se encuentran dentro de la misma carcasa. En estas fotocélulas, la luz emitida es reflejada por un espejo, y cuando se interrumpe el haz por la entrada del objeto, la luz no llega al receptor y se produce el sensado.

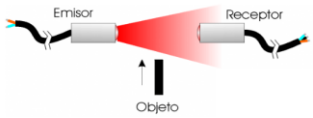
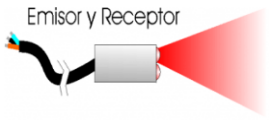
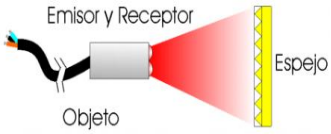
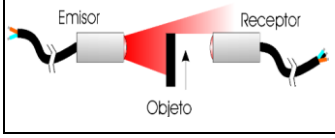
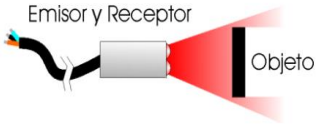
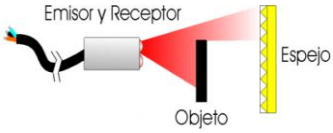
Barrera de luz	Autorreflexivas	Reflexivas con reflector
		
		

Ilustración 25 Tipos de sensores fotoeléctricos. Recuperada de:
https://www.contaval.es/que-tipos-de-sensores-fotoelectricos-existen/

Algunos ejemplos comerciales:
-DC5V NPN de barrera de luz (1m): 1,80€+1,94€ g.d.e.
- M12JG-30N1 de barrera de luz (20 m): 8,18€ + 4,71€ g.d.e.
-TCRT5000 autorreflexivo (25mm): 0,37 € +0,53€ g.d.e.
-E3F-DS30C4 autorreflexivo (300 mm): 1,77€ + 1,64€ g.d.e.
-E3JK-R4M1reflexiva con reflector (4m): 3,89€ +2,94€ g.d.e.
-Panel reflexivo TD-08 (+ otro reflector): 0,38€ +1,52€ g.d.e.

Ilustración 26. Sensor E3F-DS30C4 y sensor M12JG-30N1 a izquierda y derecha respectivamente. Recuperadas de: https://es.aliexpress.com/i/32862213669.html y https://es.aliexpress.com/i/32854695984.html

Sensores de ultrasonidos

Estos sensores, también llamados ultrasónicos, miden la distancia mediante ondas ultrasónicas. Un cabezal emite un pulso ultrasónico que es reflejado por el objeto y recibido por el mismo sensor. Mediante la medición de la distancia y el tiempo de llegada entre la emisión y recepción, se puede conocer si existe o no un objeto (Distancia L = 1/2 × T × C siendo T el tiempo recorrido de ida y vuelta y C la velocidad del sonido).

Las ventajas de estos sensores son la capacidad de medir todo tipo de objetos y su bajo coste. Por el contrario, no son muy robustos y pueden dar falsas alarmas. Además, presenta problemas con las zonas ciegas.

Algunos ejemplos comerciales son:
- Módulo HC-SR04 +HC-SR00 (2-400 cm): 0,33-0,95€+0,40€ g.d.e.
- Módulo Jsn-Sr04T (25-250 cm): 5,84€.

Ilustración 27. Módulo Jsn-Sr04T. Recuperada de: https://es.aliexpress.com/item/32332773388.html

Sensor inductivo

Los sensores inductivos de proximidad funcionan generando un campo magnético y detectando las pérdidas de corriente de dicho campo. Esto se realiza cuando los objetos de detección férricos se introducen en el sensor.

Puesto que las bolas de futbolín no tienen componentes férricos, su estudio carece de interés.

Sensor magnético

Estos sensores, detectan imanes a larga distancia, son de reducidas dimensiones y se caracterizan por tener grandes distancias de conmutación.
Al igual que con los sensores anteriores, su estudio carece de interés.

Anexos 4.2 Sensor de contacto

El sensor de contacto o final de carrera es un dispositivo electrónico, mecánico o neumático en el que se envía una señal cuando se modifica el estado del mismo. En su interior suelen contener interruptores normalmente abiertos, normalmente cerrados o conmutadores.

Es el dispositivo más sencillo de instalar, tiene un coste muy bajo, es bastante robusto y puede trabajar a altas tensiones. Como desventajas, tenemos: la baja velocidad de detección, la existencia de rebotes y que, si no se llega hasta el final, puede llegar a quemarse en caso de falso contacto.

En el futbolín debería tenerse en cuenta el peso de la bola y la mínima velocidad a la que ésta pudiera bajar, de este modo se calcularía la fuerza a vencer por el resorte del final de carrera.

También se podría incorporar un pequeño sistema mecánico que accionara un final de carrera, pulsador o cualquier otro dispositivo.

Algunos ejemplos comerciales:
- Interruptor de ruedaLXW5-11G1: (10 uni): 12,78€ → 1,278€/uni.
- Interruptor de patilla V-153-1C25 (5 uni): 1,02€ +1,36€ g.d.e →0,475€/uni.
- Interruptor de límite momentáneo ME -8169(1 uni): 1,06€+1,36€ g.d.e.

Anexos 4.3 Patentes de posible interés

Detector de goles para la detección de un objeto que sobrepasa un plano de gol

"*Objeto móvil para su utilización en un sistema provisto de unos medios para la determinación de si el objeto móvil sobrepasa un plano objetivo llano del sistema, estando provisto el objeto móvil de una pluralidad de medios sensores y unos medios de emisión de ondas de radio colocados en el objeto móvil, caracterizado porque el objeto móvil comprende unos medios de control para controlar el funcionamiento de los medios de emisión de ondas de radio, estando colocados los medios de control para tomar muestras de la intensidad del campo electromagnético medido por los medios sensores y transmitir datos relativos a la intensidad de campo medida por los sensores individuales por medio de dichos medios de emisión de ondas de radio, en el que los datos transmitidos permiten una identificación única de cuál de dicha pluralidad de medios sensores midieron los datos transmitidos.*"

(ESPAÑA Patente nº 2 355 630, 28.05.2008)

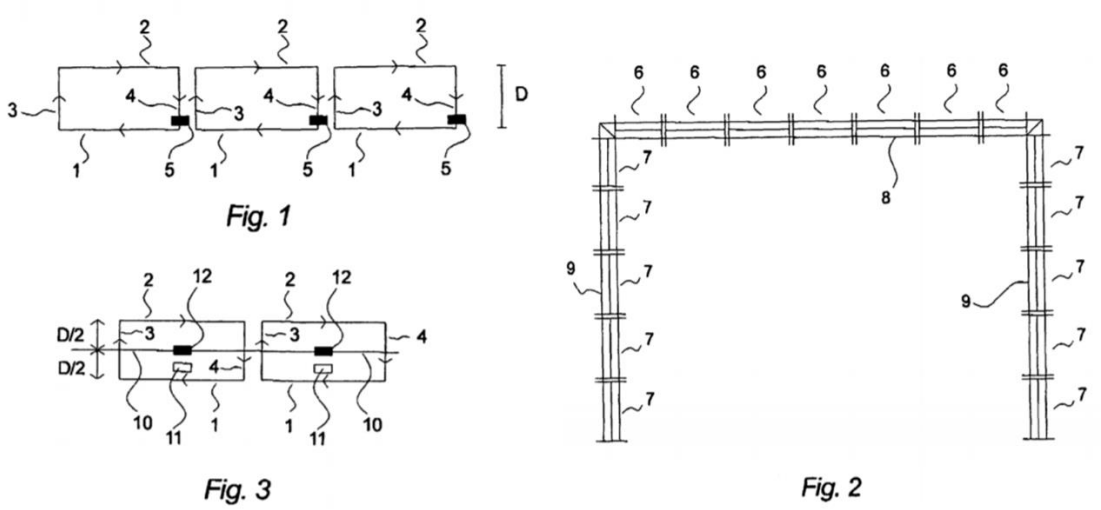


Fig. 40 figuras explicativas de la patente nº 2 355 630, 28.05.2008).

"*La figura 1 muestra tres secciones de una primera forma de realización de la presente invención colocada a lo largo del travesaño de un gol, la figura 2 muestra un gol con secciones según la forma de realización primera o segunda colocadas a lo largo del perímetro del plano de gol, y la figura 3 muestra dos secciones de una segunda forma de realización de la presente invención.*"

(ESPAÑA Patente nº 2 355 630, 28.05.2008).

Anexos 5. Análisis de alternativas para sensado de velocidad de la bola y el conteo de goles

El sensado de la velocidad se podría realizar en todo momento del juego, pero dado lo complejo que resultaría el sensado, se reducirá a medir la velocidad exclusivamente cuando se marca gol.

Puesto que el sensado de la velocidad se realiza cuando se marca gol, mediante estos sistemas se podrían omitir las alternativas anteriormente mencionadas para el conteo de goles.

Anexos 5.1 Barrera óptica

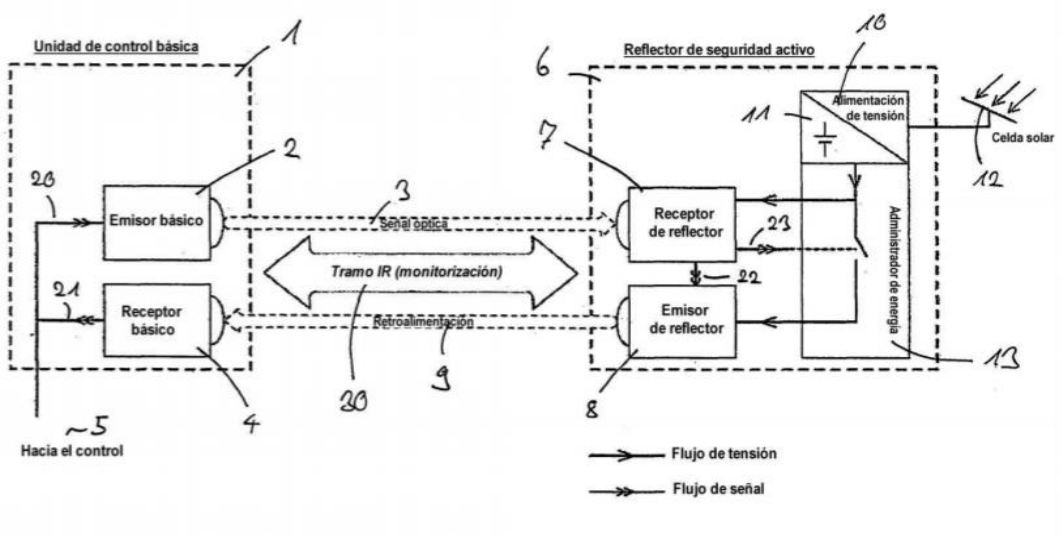


fig 41 Esquema de la patente nº 2 566 504.

"*Barrera óptica para monitorizar una trayectoria óptica (30) con una disposición básica (1) formada por un primer emisor (2), que emite una señal óptica (3) y un primer receptor (4), que genera una señal de salida para un control, y una disposición de reflector activa (6) con un segundo receptor (7) para recibir la señal óptica (3) emitida por el primer emisor (2) y un segundo emisor (8) para emitir una señal de retroalimentación (9) en función de la señal óptica (3) recibida, caracterizada porque la disposición de reflector activa (6) presenta una unidad de valoración, que comprueba si la señal óptica (3) emitida por el primer emisor (2) llega hasta el segundo receptor (7), y porque el segundo emisor (8) envía una señal de retroalimentación (9) al primer receptor (7) en el caso de que la señal óptica (3) emitida por el primer emisor (2) llegue hasta el segundo receptor (7), en donde el primer receptor (7) pone a disposición una señal de conmutación*

correspondiente, después de la valoración de la señal de retroalimentación (9), a la salida (21) de la disposición básica (1)."
(ESPAÑA Patente nº 2 566 504, 09.03.2016)

Para el sensado de la velocidad cuando entra la bola en la portería, basta conocer los instantes en el que los dos receptores han obtenido la señal y la distancia entre ambos. Así pues, con la definición de velocidad (v=distancia/tiempo) se obtendría una estimación de la velocidad de la bola (aunque se obtendría exclusivamente la velocidad axial al plano formado por la posición de los dos receptores).

Esta misma patente puede servir de inspiración para elaborar un dispositivo capaz de medir la velocidad mediante el uso de dos fotocélulas en paralelo.

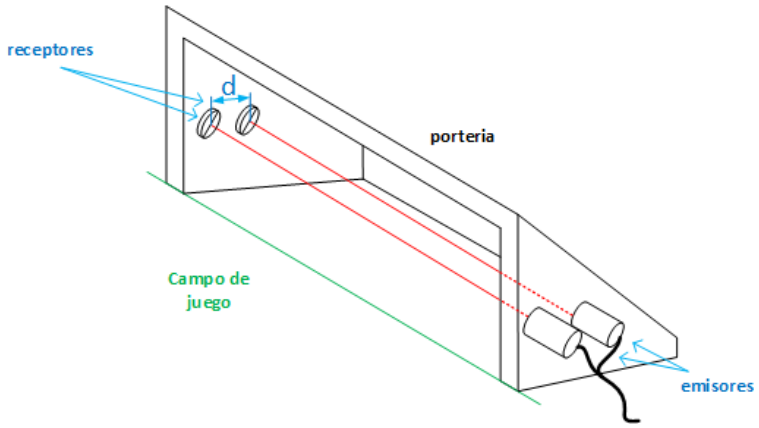


Fig. 42. Esquema para el sensado de velocidad con barrera óptica.

Los principales inconvenientes de esta alternativa son: la no muy alta sensibilidad (ésta depende de la distancia y del tiempo de muestreo de las fotocélulas), la necesidad de calibrado, una ubicación desfavorable que hace que los sensores sean vulnerables a colisiones con la bola, que sólo se permite sensar la velocidad perpendicular al plano de la portería y que se requiere que la trayectoria de la bola corte con los haces de los emisores.

Como ventajas se han de señalar las siguientes: el bajo coste, la ubicación es apenas visible, ocuparía relativamente poco espacio y la complejidad de instalación no es muy elevada.

Anexos 5.2 Sensor piezoeléctrico de impacto

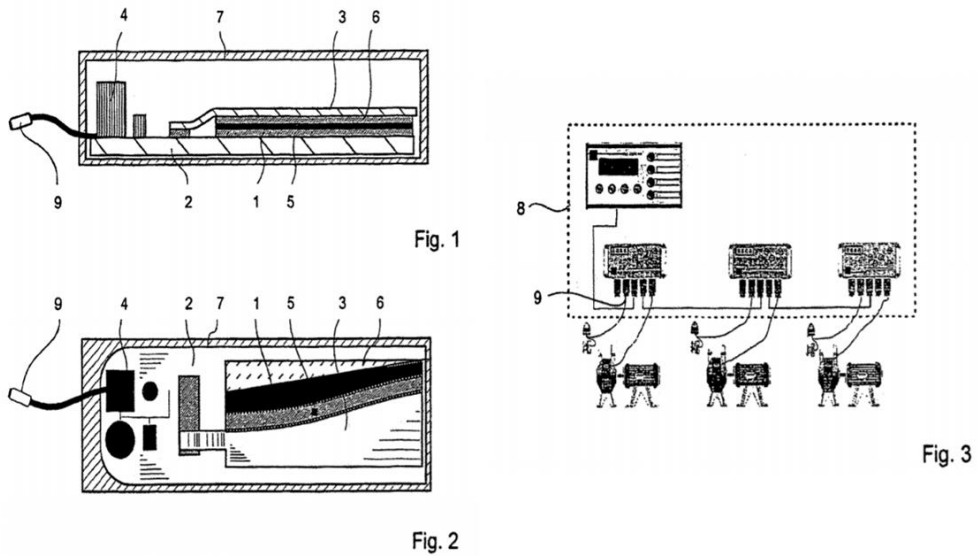


Fig. 43. Esquema de la patente nº 2 295 184

"*Sensor piezoeléctrico con un soporte de base (2), un captador de valor de medida dispuesto sobre el soporte de base (2), una capa de cubierta (3) que tapa el captador de valor de medida y una electrónica de evaluación (4), estando formado el captador de valor de medida por una capa piezoeléctrica (1), presentando el soporte de base (2) una primera capa de contacto (5) eléctricamente unida con la capa piezoeléctrica (1), presentando la capa de contacto (3) una segunda capa de contacto (6) que está eléctricamente unida con la capa piezoeléctrica (1), y siendo la electrónica de evaluación (4) capaz de determinar una carga mecánica de la capa piezoeléctrica (1) por medio de la evaluación de la diferencia del potencial eléctrico entre la primera capa de contacto (5) y la segunda capa de contacto (6), caracterizado porque el sensor está configurado, junto con la electrónica de control, con el espesor de una película, presentando la capa piezoeléctrica (1) un espesor de menos de 1 mm y estando dispuesta la electrónica de evaluación (4), junto con el captador de valor de medida, sobre el soporte de base (2) configurado a modo de película, que está fabricado de un material elástico que sólo amortigua un poco las vibraciones*."

(ESPAÑA Patente nº 2 295 184, : 16.04.2003)

Para el sensado de la velocidad de la bola tras entrar a la portería, basta con saber que la fuerza de impacto es proporcional a la velocidad y masa de la bola; por lo tanto, si se conoce la masa y la fuerza se conoce la velocidad.

Al igual que en el caso anterior, esta patente se puede para el sensado de velocidad si se añaden a la portería los sensores de impacto necesarios.

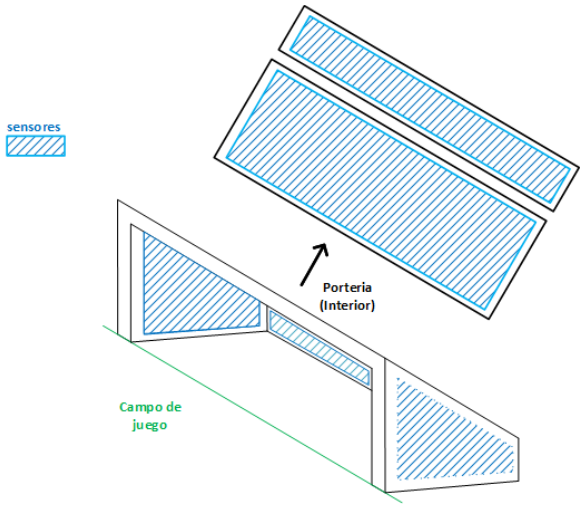


Fig. 44. Esquema para el sensado de velocidad con sensores de impacto.

En el esquema, la parte rayada hace referencia al lugar donde se ubicarán los sensores de impacto. Si se deseara sensar las bajas velocidades cuando la bola no golpea en la parte trasera de la portería, habría que añadir un sensor de impacto (u otro sensor) debajo de la portería en la canaleta.

Otra alternativa utilizando un sensor de impacto, sería la de diseñar una carcasa rígida móvil donde el impacto se concentrará íntegramente en el sensor.

La gran ventaja de esta alternativa es que mide la velocidad real de la bola, además permite implementar funcionalidades adicionales (si se ‘’juega’’ con el sensor que ha recibido el impacto). Como desventajas tenemos: su baja robustez, su precio elevado, la dificultad de tratar los datos y la dependencia del valor estimado de velocidad con el peso de la bola (el cual podrá ser variable).

Otra patente de interés relacionada: *“Dispositivo y método para la medición de una fuerza de lanzamiento ejercida sobre un aparato de juego móvil.”*.

(ESPAÑA Patente nº 2 321 339 : 27.07.2006).

Anexos 5.3 Método y dispositivo para la medición de trayectorias de objetos de geometría conocida

"Método y dispositivo para la medición de trayectorias de objetos de geometría conocida. La invención se refiere a un método y un dispositivo para la detección y análisis de trayectorias de objetos con geometría conocida, que posibilita la obtención de los parámetros cinemáticos y/o geométricos de dicho objeto, aumentando la precisión de la medida respecto a sistemas conocidos independientemente del tamaño o forma de los objetos. Un conjunto de elementos sensores detectan por interferencia la presencia o ausencia objetos con geometría conocida, y se registra los datos procedentes de los sensores. Cuando un objeto atraviesa la región del espacio entre los receptores y la fuente de emisión, el sistema registra la interferencia de cada sensor, el instante y la duración. De esta manera, y mediante el procesado posterior, se obtienen datos como: geometría, velocidad, aceleración, spin (rotación y eje). La invención tiene como una de sus principales aplicaciones el entrenamiento de deportes de pelota o balón como fútbol, baloncesto, tenis, balonmano, rugby y todos aquellos en los que intervenga un objeto de geometría conocida.

La invención se encuadra en el sector técnico de aparatos para la medición de trayectorias, teniendo una aplicación directa sobre lanzamiento y tiro de pelota o balón. Así mismo, también tiene una aplicación directa sobre el sector de las máquinas recreativas, especialmente en periféricos para consolas y ordenadores domésticos. Además, también tiene aplicaciones de utilidad en sectores como el militar y el aeronáutico."

(ESPAÑA Patente nº 2 264 324, 16.11.2007)

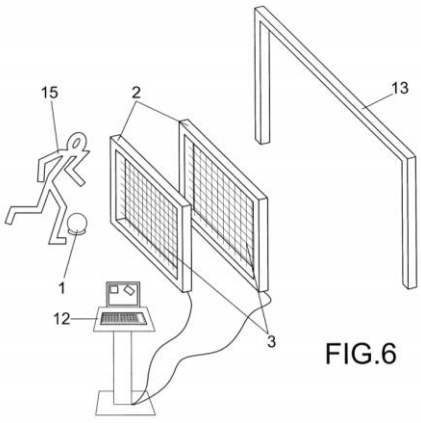


Fig. 45. Esquema de la Patente nº 2 264 324.

Anexos 5.4 Sensor doppler HB100

Se basa en el principio doppler en el cual un emisor fijo (velocidad 0m/s) emite ondas de radio, cuando se encuentran con un objeto son reflejadas y llegan al receptor. Se hace una comparación de las frecuencias de onda reflejadas y emitidas si son iguales al objeto que está quieto; si son diferentes, estará en movimiento y bastará con analizar la frecuencia de la señal recibida para poder obtener una medida de la velocidad.

Ilustración 28. Sensor doppler HB100. Recuperada de:
http://www.arduinove.com/index.php?route=product/product&product_id=519

Tiene la ventaja de detectar sin contacto, no se ve afectado por temperatura, humedad, ruido, aire, polvo o luz y es bastante económico. Tiene limitaciones en cuanto a precisión y robustez; Además, el movimiento del portero puede afectar al sensado.

- Sensor HB100 1,73€.

Anexos 6. Análisis de alternativas de monederos

Dependiendo de si es capaz o no de discriminar el valor de las monedas introducidas, podemos distinguir dos tipos de monedero:

Monedero electrónico discriminante o multimoneda

Esta tipología permite incrementar el saldo en proporción al valor monetario introducido. Así pues, permite utilizar monedas de distinto valor.

La ventaja principal de esta tipología es la versatilidad de pago que ofrece al consumidor.

Monedero electrónico no discriminante

Incrementa el saldo en función del valor monetario introducido. Realiza una comparativa entre la moneda introducida y una moneda de referencia.

Las ventajas principales de esta tipología son la sencillez de programación y la capacidad de trabajar con cualquier tipo de moneda (siempre que se cumplan las dimensiones requeridas).

A su vez ambos monederos pueden clasificarse dependiendo de si devuelven o no cambio.

Anexos 6.1 Alternativas comerciales de monederos multimoneda

FT modular X DSP de azkoyen:

Ilustración 29. Monedero FT Modular X DSP de Azkoyen. Recuperada de: https://www.azkoyenpayment.com/productos/

- No devuelve cambio.
- Incluye tres pares de sensores ópticos, un sensor acústico y dos pares de sensores inductivos (o un par en su versión anterior).
- Alimentación a 12V.
- Requerimientos de corriente promedio: 50-150 mA.
- Acepta hasta 32 tipos de monedas diferentes.
- Las dimensiones de moneda que acepta son:
    Diámetro: de 16.25mm a 32.5mm.
    Espesor: de 1.2 mm a 3.3 mm.
- Capacidad de insertar módulos como funnels (embudos), sorters y herramientas de programación y verificación.
- Dimensiones: 89mm x 102 mm x 48 mm (dimensiones no verificadas)

Caesium ES110 de coges

Ilustración 30. Caesium ES110 de coges. Recuperada de: https://www.coges.es/productos/

-No devuelve cambio.
-Sistema anti-fishing (para evitar recuperar monedas).
-Existe el modelo con el protocolo MDB (Multi Drop Bus).
- Acepta hasta 32 tipos de monedas diferentes.
-Las dimensiones de moneda que acepta son:
Diámetro: de 16.25mm a 28 mm.
Espesor: de 1 mm a 3.2 mm.
- Programación a través de PC conectado a una interfaz serial o programador portátil. O mediante dip-switch externo.
- Acepta máximo tres monedas por segundo.
-Dimensiones 89mm x 103 mm x 49 mm.

C-10 asm7 de somic (validator)

Ilustración 31. C-10 VEND 0 de somic. Recuperada de: https://somyc.com.ar/productos.html

-No devuelve cambio.
-Existen varios modelos como C-10 VEND (permite además funcionar como totalizador, es decir sumar monedas hasta llegar al precio del servicio, momento en el cual envía un pulso de aviso); o C-10 ECO-2 (permite ademas funcionar como totalizador y temporizador); o variantes del modelo para funcionar como validator, totalizador o temporizador.
- Alimentación de 9 a 18Vcc.
- Requerimientos de corriente promedio: 60-300mA.
-Las dimensiones de moneda que acepta son:
Diámetro: de 16mm a 31 mm.
Espesor: de 1 mm a 3 mm.
- Acepta hasta 16 tipos de monedas o cospeles.
- Programación mediante PC (Conexión por puerto COM con protocolo SOMYC) con el software gratuito NEXUS C-10.
- Dimensiones: 310mm x 115mm x 185mm.
- Precio: 87 a 157€ (depende del modelo).

HI-09FCS de HUAI I electronics

Ilustración 32. HI-09FCS. Recuperada de:
https://www.arcadexpress.com/es/monederos-puertas/220-monedero-arcade-euros-hi-09fcs-.html

-No devuelve cambio.
-Existe el modelo HI-09UCS (con mismas características, pero con la ranura superior).
- Alimentación a 12V.
- Requerimientos de corriente promedio: 50 mA (max 300mA).
- Acepta hasta 8 tipos de monedas diferentes (máx 4 espesores diferentes).

- Espesor de moneda admitido: de 1.8 mm a 3 mm.
- Autoprogramable, sin PC (se programa mediante switches y potenciómetros).
- Acepta máximo tres monedas por segundo.
- Dimensiones:
  HI-09UCS: 102mm x 99mm x 55mm.
  HI-09FCS: 124.5 mm x 120.5 mm x 64.5 mm.
- Precio 40€.

Aeterna de coges

Ilustración 33. Aeterna de coges. Recuperada de: https://www.coges.es/productos/

-Devuelve cambio.
- Dispone de cinco tubos que pueden contener hasta 97 monedas de 0,05 €.
- Permite conectar Un sistema de pago "cashless" de llave o tarjeta.
- Protocolos de comunicación: Executive, MDB, BDV.
- Programación de los parámetros mediante teclado frontal o Maxi Programmer.
- Las monedas que acepta son de 0.05 €, 0.10€, 0.20€, 0.50€ y 1€ (la versión UNICA de coges dispone de 6 tubos pudiendo aceptar además 2€).
- Dimensiones: 138 mm x 381 mm x 80 mm.

J-2000 de Jofemar

Ilustración 34. J-2000 de Jofemar. Recuperada de:
https://www.maxvending.es/monederos-selectores/compacto-jofemar-j2000.html

-Devuelve cambio.
- Dispone de cinco tubos que pueden contener entre 50 y 70 monedas.
- Permite programar descuentos según la fecha y la opción de extracción de datos a través de infrarrojos (en formato DEX).
- Protocolos de comunicación MDB.
- Sistema de programación integrado, display numérico y botones de selección.
- Las monedas que acepta son de 0.05 €, 0.10€, 0.20€, 0.50€ y 1€.
- Dimensiones: no específicas).
- Precio 110€.

Anexos 6.2 Alternativas comerciales de monederos monomoneda

TW-333 de TL

Ilustración 35. TW-333 de TL. Recuperada de:
https://www.arcadexpress.com/es/monederos-puertas/219-monedero-arcade-universal-tw-333-.html

-No devuelve cambio.
- Acepta todo tipo de monedas.
- Posee un display para contar el saldo restante.
- Las dimensiones de moneda que acepta son:
    Diámetro: de 22mm a 28 mm.
    Espesor: de 1.7 mm a 22 mm.

-Precio 8€.

TW-130B de TL

Ilustración 36. TW - 130B de TL. Recuperada de: https://en.china.cn/search/tw-130b.html

-No devuelve cambio.
- Acepta todo tipo de monedas.
- Las dimensiones de moneda que acepta son:
    Diámetro: de 20mm a 30 mm.
    Espesor: de 1.7 mm a 24 mm.
- Alimentación de 12Vcc.
- Requerimientos de corriente promedio: 75mA.
- Precio: 6€.

CH-926 de CH

Ilustración 37. CH-926 DE CH. Recuperada de:
https://www.amazon.co.uk/Akozon-Acceptor-Selector-Mechanism-Vending/dp/B07HKPDDC5/

-No devuelve cambio.
- Acepta todo tipo de monedas.
- Las dimensiones de moneda que acepta son:
    Diámetro: de 15 mm a 32 mm.
    Espesor: de 1.2 mm a 38 mm.
- Alimentación de 12Vcc.
- Requerimientos de corriente promedio: 65mA.
- Precio 16€.

Anexos 7 Galería de fotos del montaje.

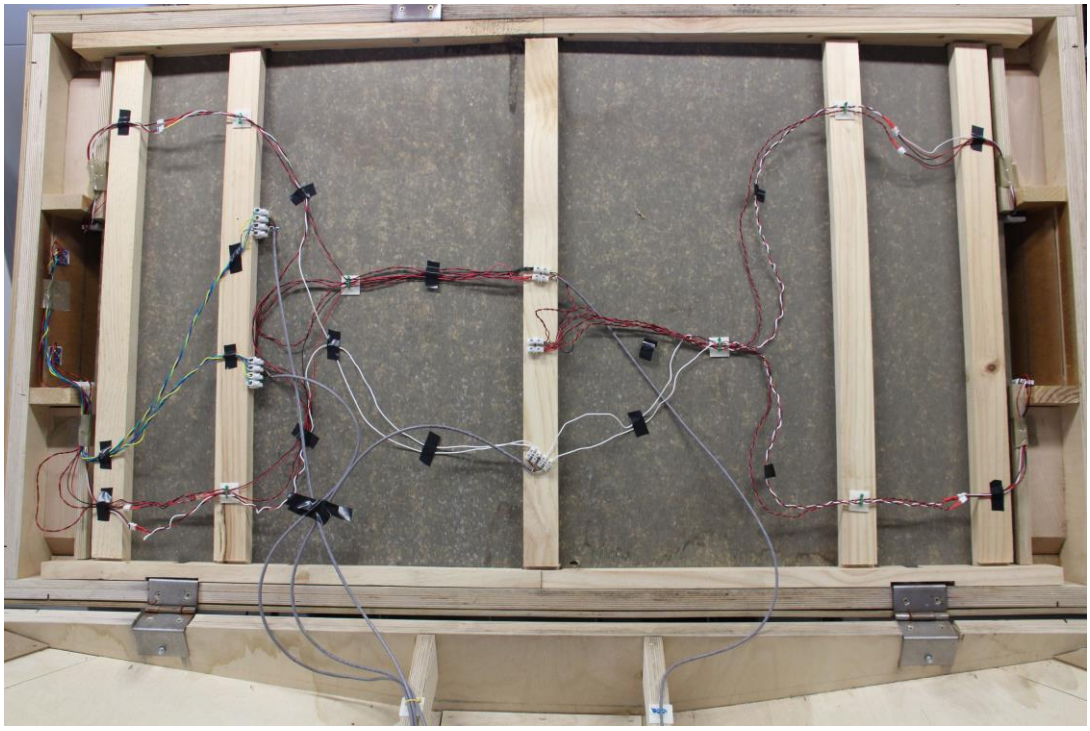


Ilustración 38. Conexionado de futbolín (parte superior).

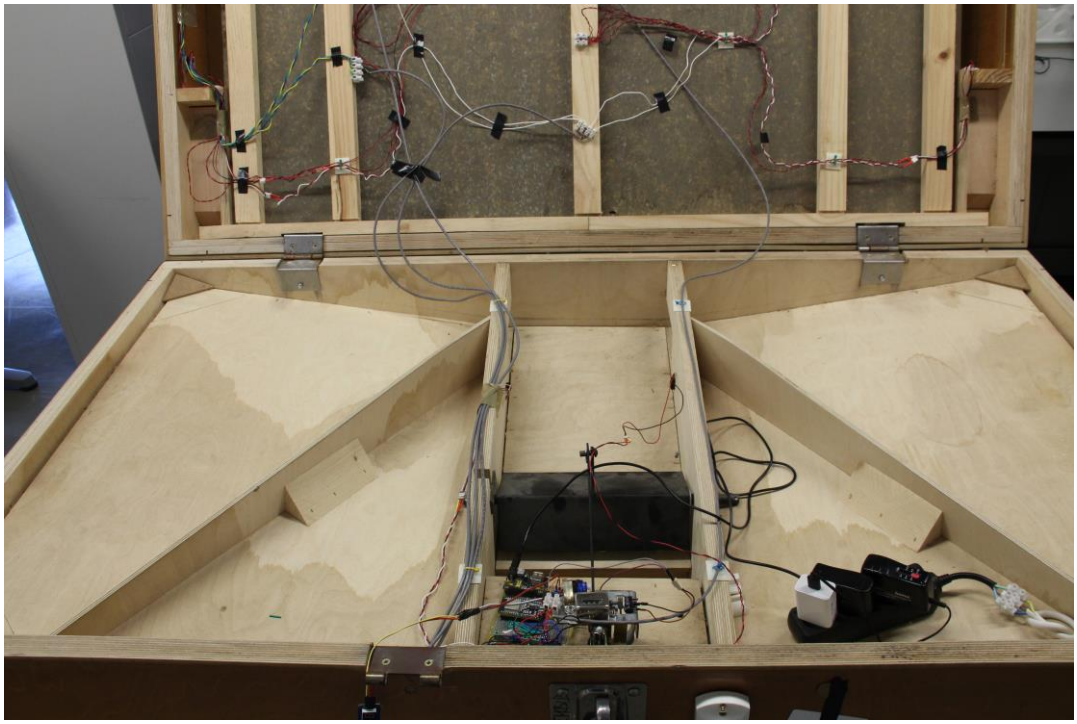


Ilustración 39. Conexionado de futbolín (parte inferior).

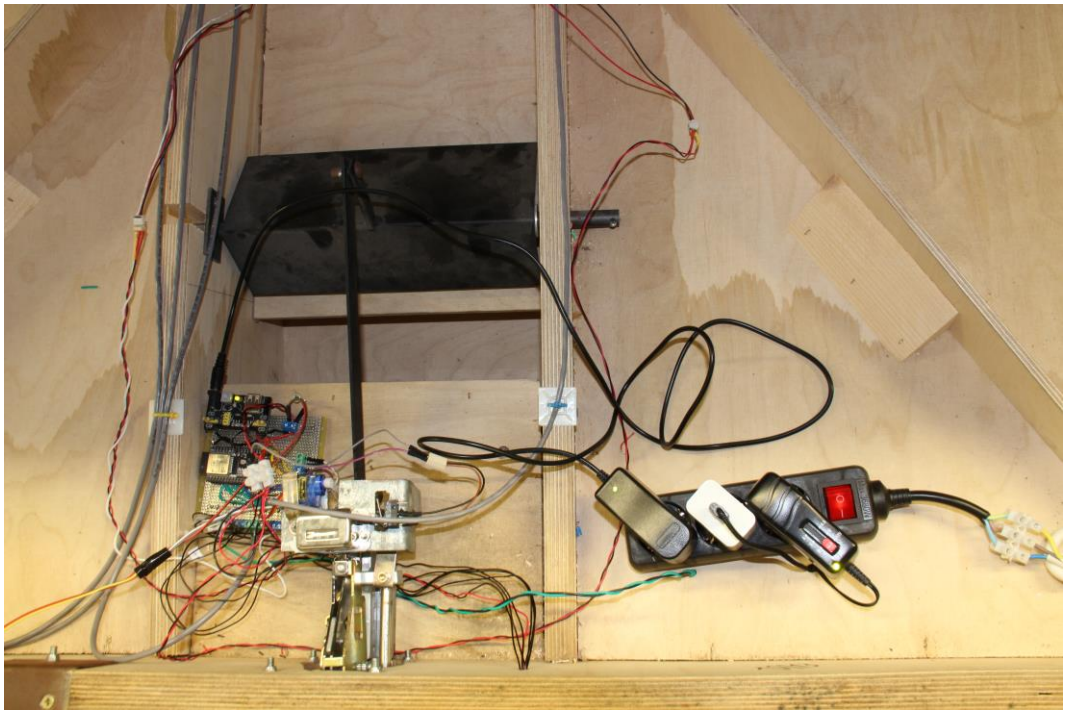


Ilustración 40. PCB, monedero y fuentes de alimentación.

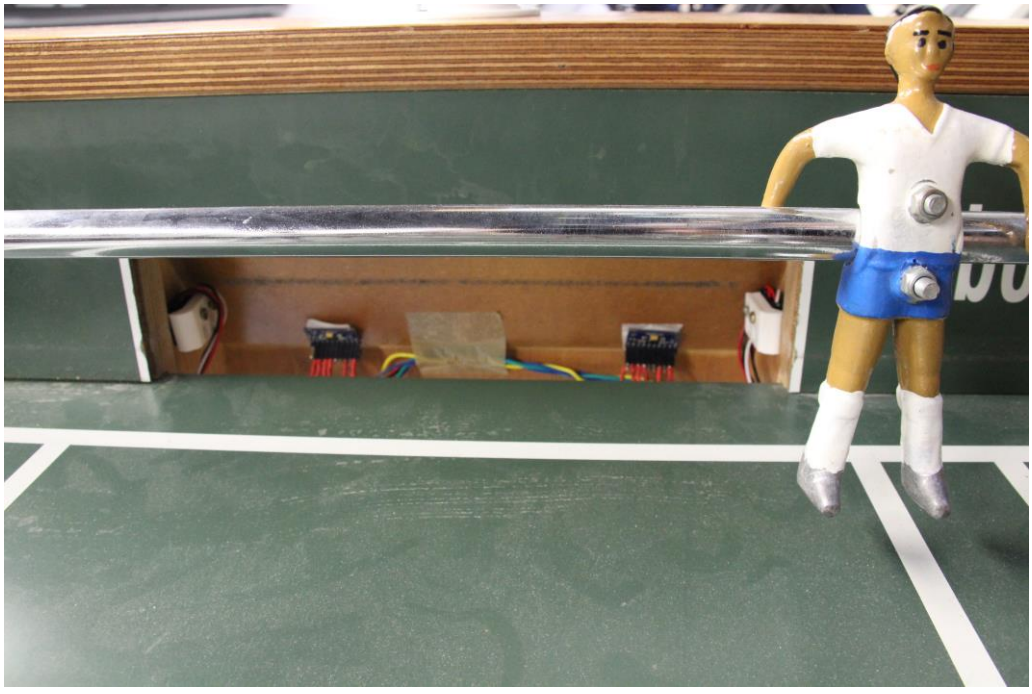


Ilustración 41. Portería desde el exterior.

Anexos 8. Tablas de características de Pi Camera

Para la Pi Camera V1:

#	Resolución	Ratio	FPS	Video	Imagen	FoV	Binning
1	1920x1080	16:9	1< fps <= 30	Si	No	Parcial	No
2	2592x1944	4:3	1< fps <= 15	Si	Si	Total	No
3	2592x1944	4:3	1/6< fps <= 1	Si	Si	Total	No
4	1296x972	4:3	1< fps <= 42	Si	No	Total	2x2
5	1296x730	16:9	1< fps <= 49	Si	No	Total	2x2
6	640x480	4:3	42< fps <= 60	Si	No	Total	4x4
7	640x480	4:3	60< fps <= 90	Si	No	Total	4x4

Tabla 5. Tabla de características Pi Camera V1.

Para la Pi Camera V2:

#	Resolución	Ratio	FPS	Video	Imagen	FoV	Binning
1	1920x1080	16:9	0.1<=fps<=30	Si	No	Parcial	No
2	3280x2464	4:3	0.1<=fps<=30	Si	Si	Total	No
3	3280x2464	4:3	0.1<=fps<=30	Si	Si	Total	No
4	1640x1232	4:3	0.1<=fps<=30	Si	No	Total	2x2
5	1640x922	16:9	0.1<=fps<=30	Si	No	Total	2x2
6	1280x720	16:9	40< fps <= 90	Si	No	Parcial	2x2
7	640x480	4:3	40< fps <= 90	Si	No	Parcial	2x2

Tabla 6 Tabla de características Pi Camera V2.

Anexos 9. Ensayo de sensado de velocidad e impacto con acelerómetros

Cuando se realizan las correspondientes tiradas, sólo nos queda adaptar manualmente los datos de Excel. Por ello, se deben eliminar los saltos de línea, las tabulaciones y cambiar el punto por la coma.

Para trabajar con los datos, se realizan 15 ficheros debido a que las tiradas son a tres alturas y cinco en posiciones. En esos ficheros se ponen las diez tiradas cada una con sus 500 muestras de X, Y y Z de cada acelerómetro.

Finalmente se hace una tabla que engloba datos estadísticos de todos los ficheros.

- En las filas, se ponen los 150 ensayos (10 tiradas,5 posiciones, 3 alturas) añadiendo el valor de cada acelerómetro y su diferencia (450 líneas).

- En las columnas, se pone la media, valor eficaz, valor mínimo y máximo, pico a pico, el régimen permanente y el valor eficaz sin offset; de las medidas en X, Y, Z y del promedio de estas tres.

De los valores anteriores se toma como el más representativo el valor eficaz (o media cuadrática) sin offset. Este valor eficaz por ejemplo para la variable Z, tiene la siguiente la siguiente expresión:

$$Vef_{Z_{S.O}} = \sqrt{\frac{\left[\left(\sum_{n=1}^{300} Z^2\right) - 300 * RP_Z{}^2\right]}{300}}$$

Se resta el valor del régimen permanente porque interesa conocer el valor eficaz de la variación de las medidas de fuerza. Ya que, si el régimen permanente no tiende a 0, no tiene sentido calcular el valor eficaz tradicional.

Finalmente, se grafica el valor de cada valor eficaz sin offset y se comparan los resultados para los dos acelerómetros y para las distintas alturas y zonas.

De las numerosas gráficas realizadas, podemos tomar una de ejemplo:

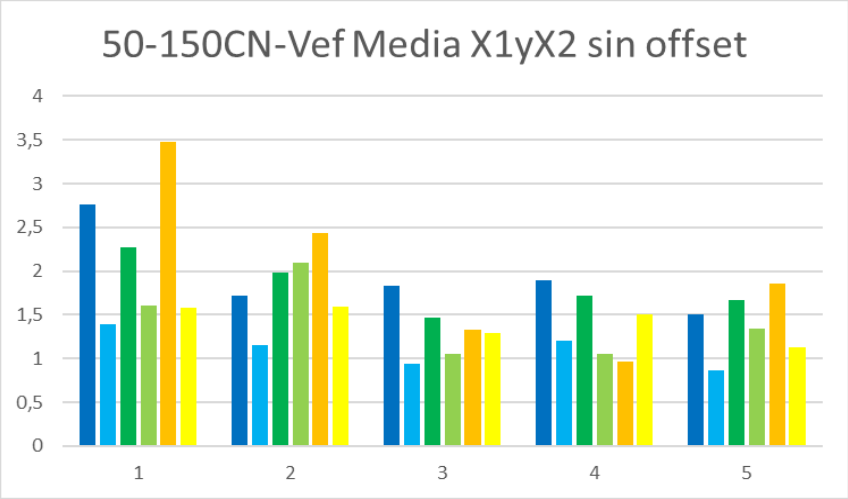


Ilustración 42. Gráfica de ensayo de tiradas

En ella se muestra en azul, verde y amarillo las medidas de los dos acelerómetros para una altura de 50, 100 y 150 cm respectivamente en cinco posiciones distintas.

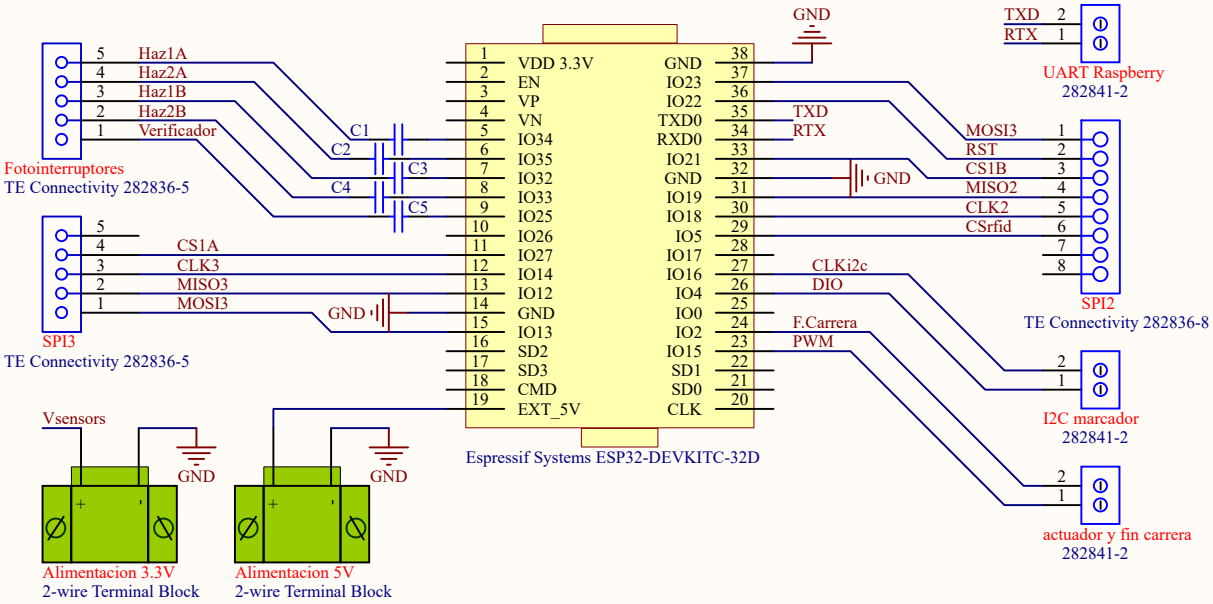
Se puede observar que, aunque exista una ligera correlación entre los valores y la altura de las tiradas, no hay una tendencia clara en los datos que permita obtener unas conclusiones favorables, debido a la gran variabilidad de los datos, la no idoneidad del ensayo y los defectos en la tabla y sujeción de los acelerómetros. Por lo tanto, el ensayo deberá volver a realizarse en condiciones más adecuadas.

Anexos 10. Planos de piezas 3D

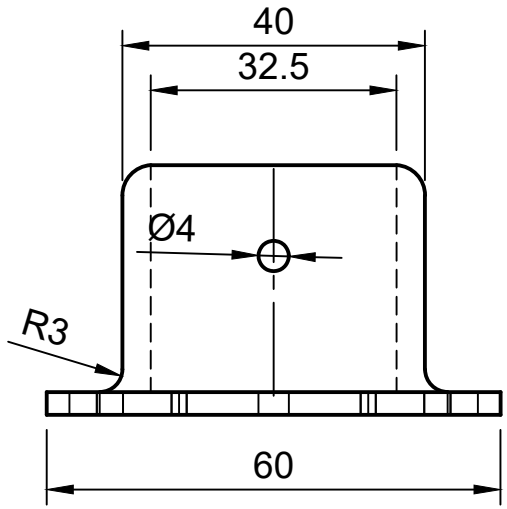
(Páginas 112 a 122).

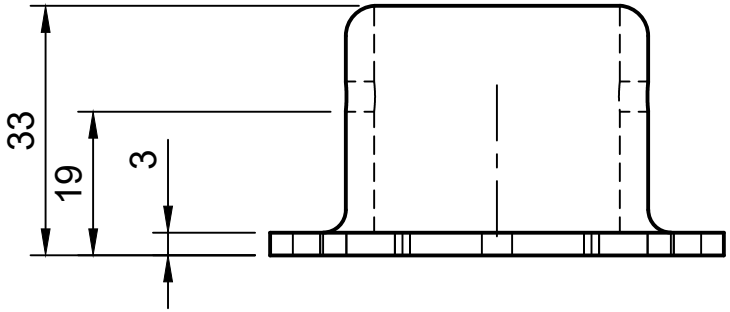
Anexos 11. Código de programación

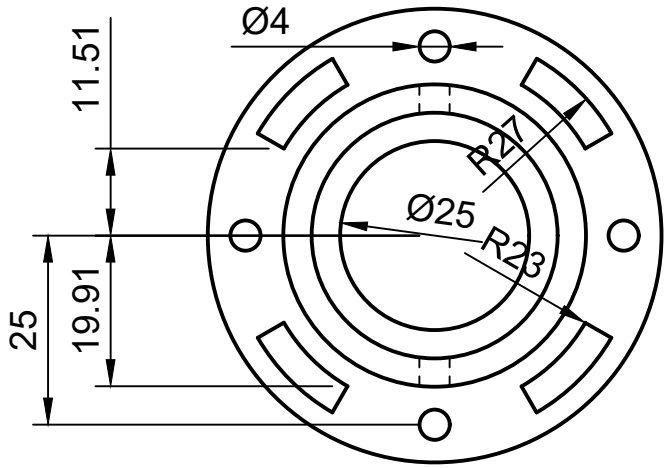
(Páginas 123 a 177).

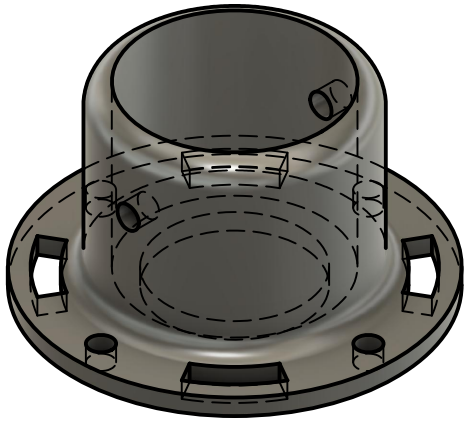


Title	Esquemático PCB	
Size: A4	Number: s/n	Revision:
Date: 21/08/2020		Sheet 1of 1
File: Sheet1.SchDoc		Drawn By: Rodrigo Pérez

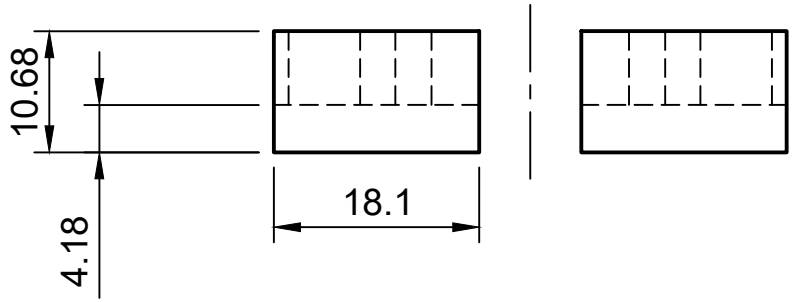


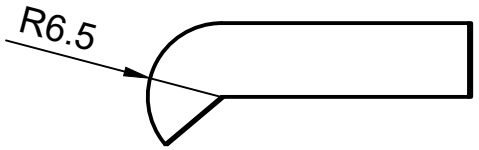


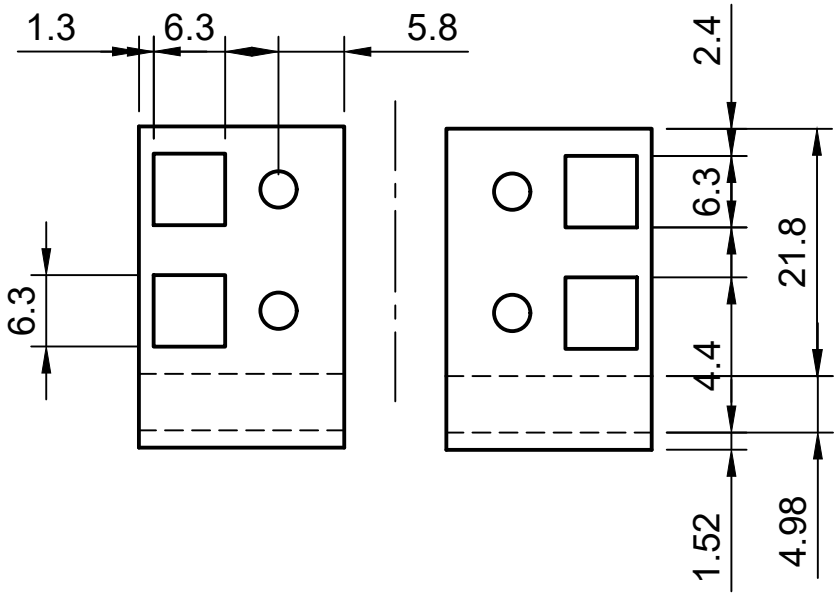


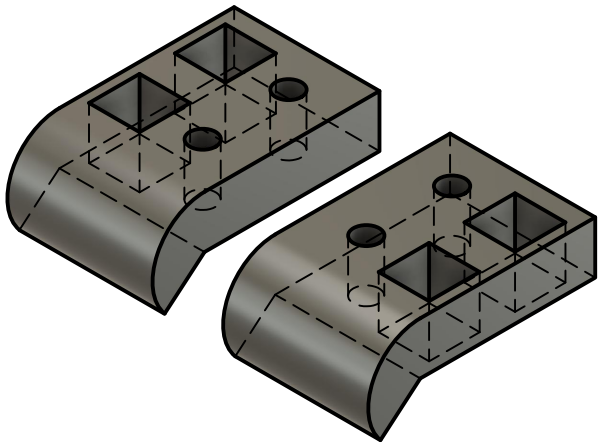


Dept.	Technical reference	Created by	Approved by
TFG	futbolín 2.0	Rodrigo Perez 19/07/2020	
		Document type: plano de pieza 3D	Document status
		Title: soporte barra	DWG No.: 1
			Rev. / Date of issue / Sheet: 1/1

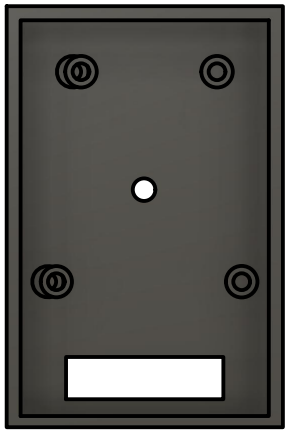


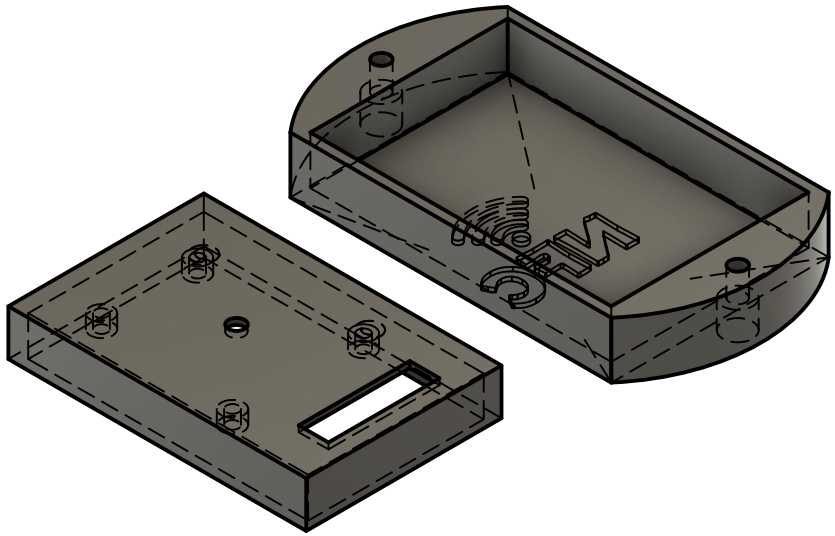




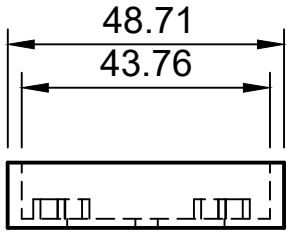


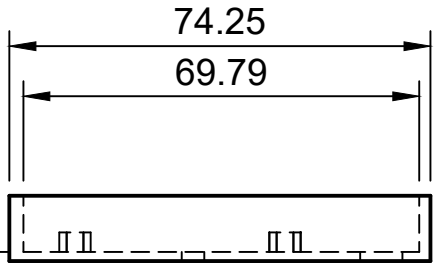
Dept.	Technical reference	Created by	Approved by
TFG	futbolín 2.0	Rodrigo Perez 19/07/2020	
		Document type: plano de pieza 3D	Document status
		Title: protectores de sensores	DWG No.: 2
			Rev. / Date of issue / Sheet: 1/1

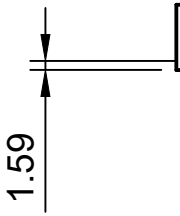


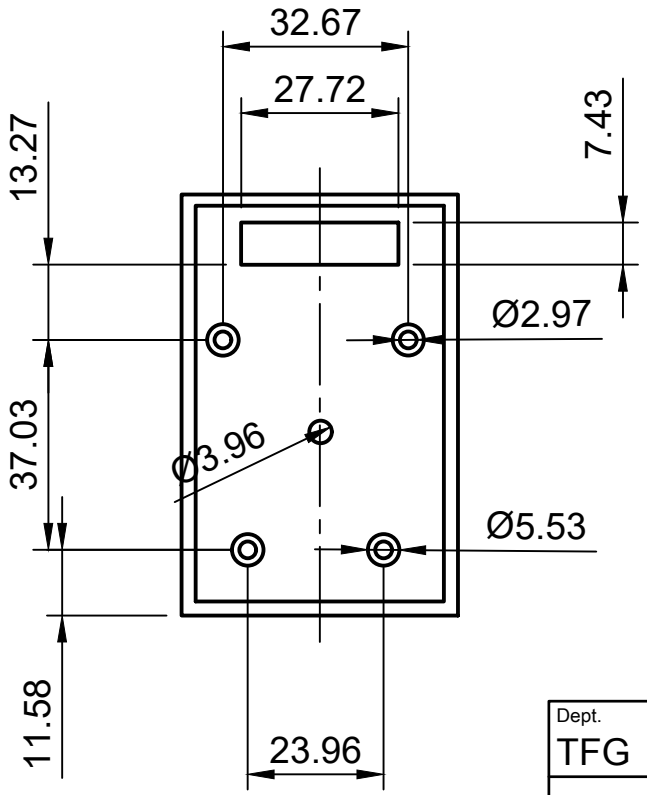


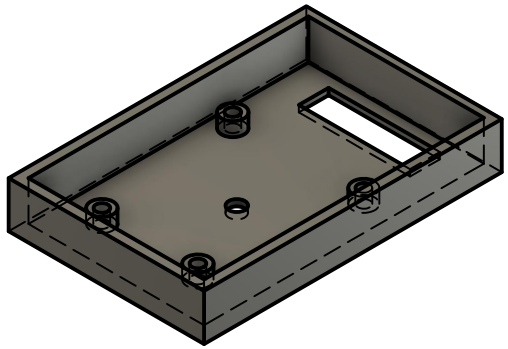
Dept.	Technical reference	Created by	Approved by
TFG	futbolín 2.0	Rodrigo Perez 19/07/2020	
		Document type: plano de pieza 3D	Document status
		Title: Carcasa RFID	DWG No.: 3
			Rev. / Date of issue / Sheet: 1/3



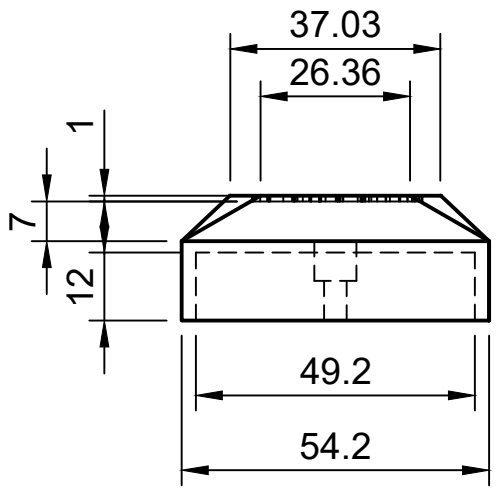


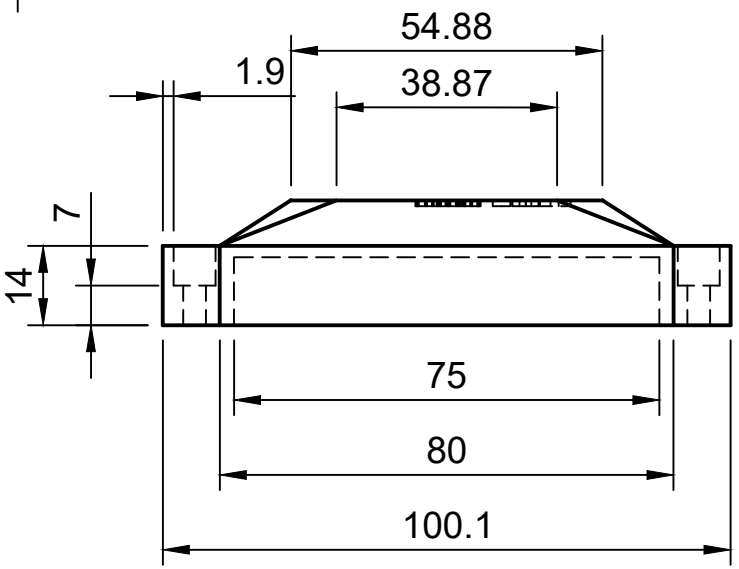


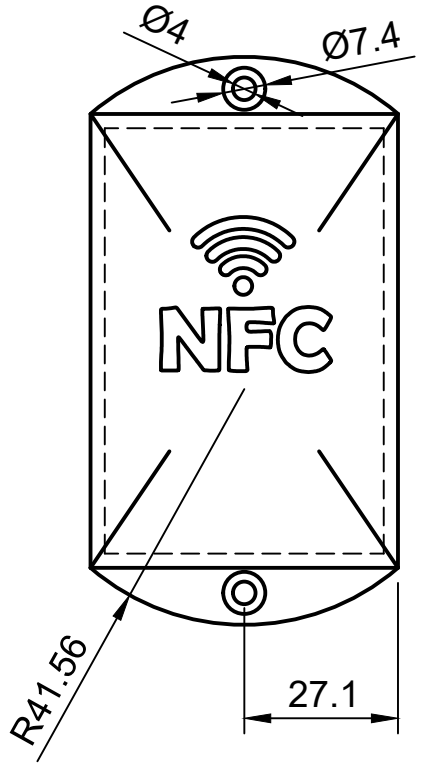


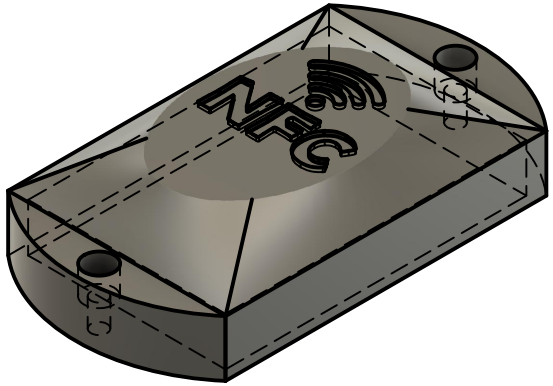


Dept.	Technical reference	Created by	Approved by
TFG	futbolín 2.0	Rodrigo Perez 19/07/2020	
		Document type: plano de pieza 3D	Document status
		Title: Carcasa interior RFID	DWG No.: 4
			Rev. / Date of issue / Sheet: 2/3

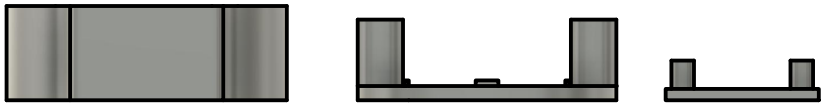


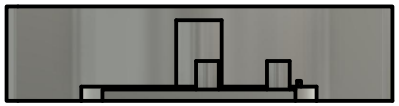


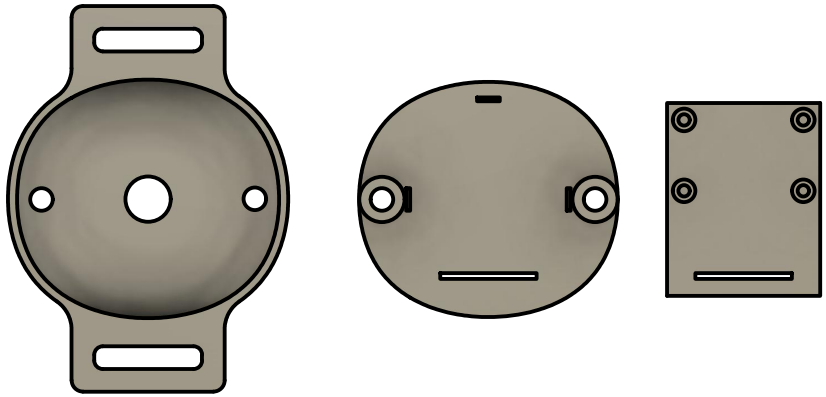


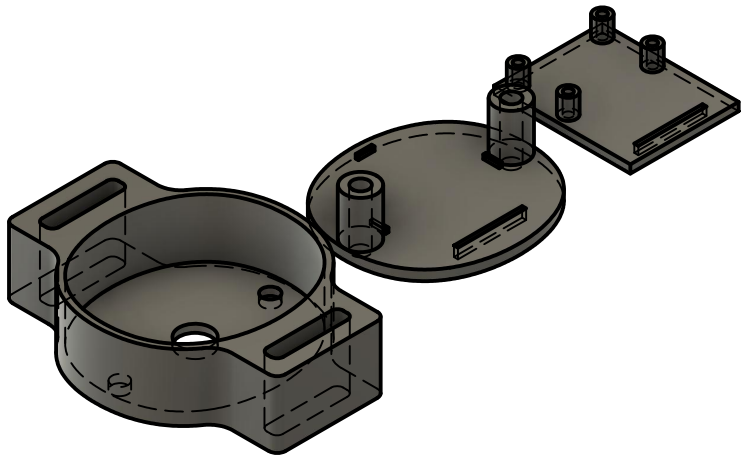


Dept.	Technical reference	Created by	Approved by
TFG	futbolin 2.0	Rodrigo Perez 19/07/2020	
		Document type: plano de pieza 3D	Document status
		Title: carcasa superior RFID	DWG No.: 5
			Rev. / Date of issue / Sheet: 3/3

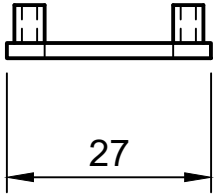


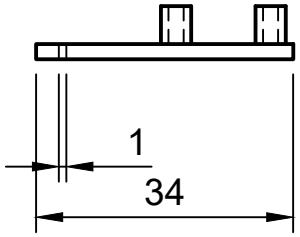


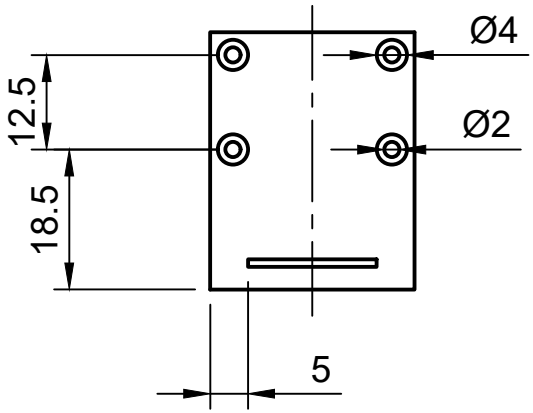


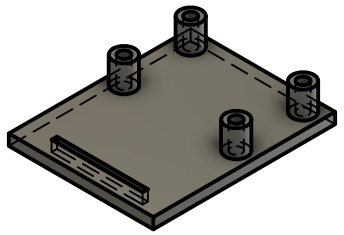


Dept.	Technical reference	Created by	Approved by
TFG	futbolín 2.0	Rodrigo Perez 19/07/2020	
		Document type: plano de pieza 3D	Document status
		Title: Carcasa de cámara	DWG No.: 6
			Rev. / Date of issue / Sheet: 1/4

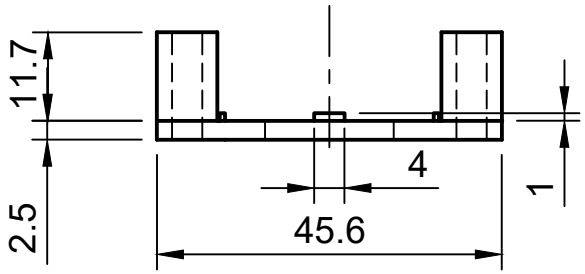


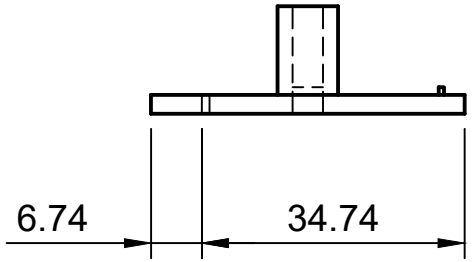


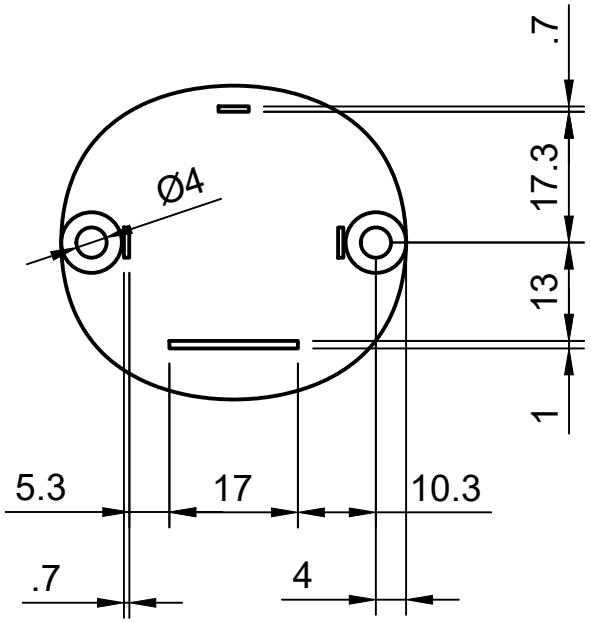


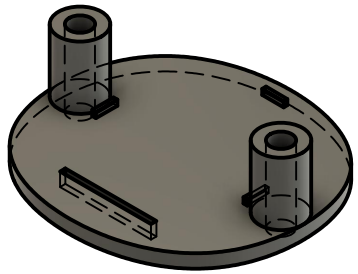


Dept.	Technical reference	Created by	Approved by
TFG	futbolín 2.0	Rodrigo Perez 19/07/2020	
		Document type: plano de pieza 3D	Document status
		Title: base de camara	DWG No.: 7
			Rev. / Date of issue / Sheet: 2/4

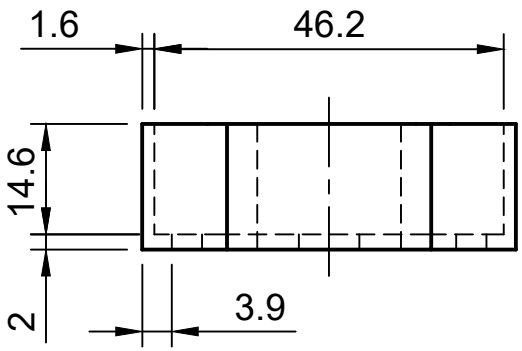


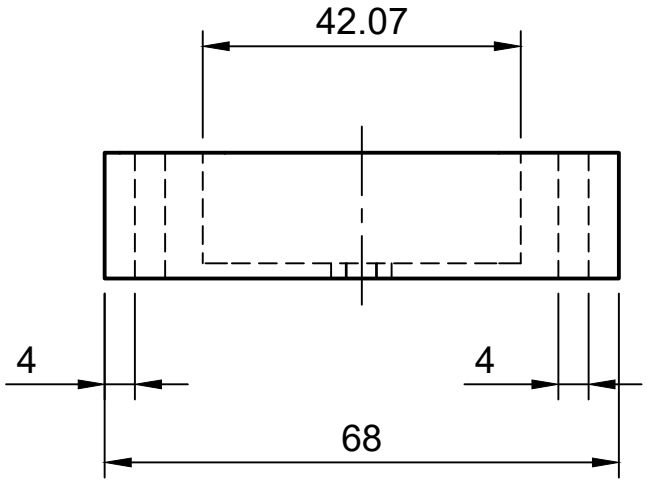


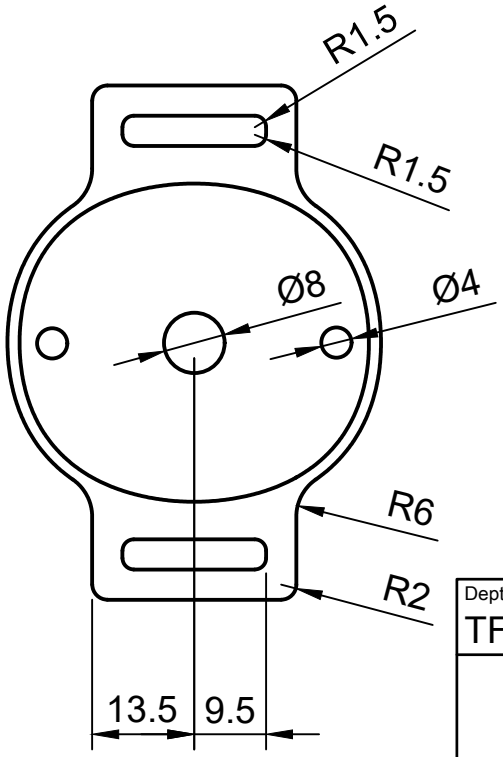


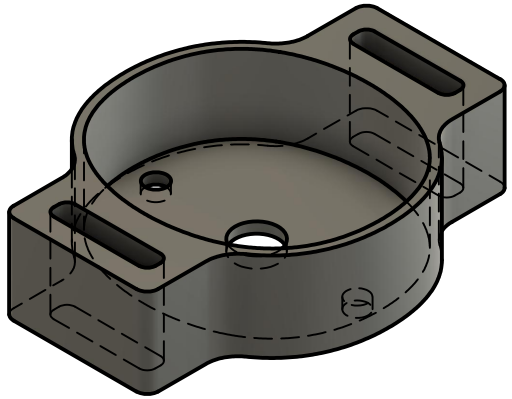


Dept.	Technical reference	Created by	Approved by
TFG	futbolin 2.0	Rodrigo Perez 19/07/2020	
		Document type: plano de pieza 3D	Document status
		Title: carcasa interior camara	DWG No.: 8
			Rev. / Date of issue / Sheet: 3/4

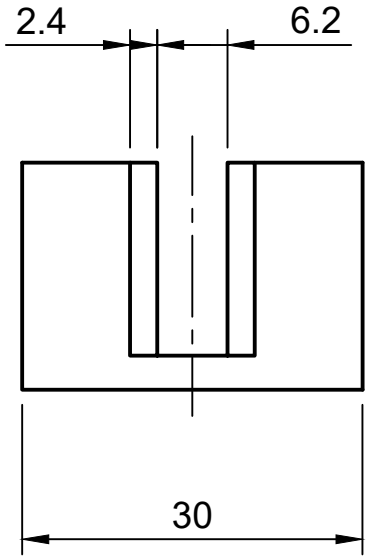


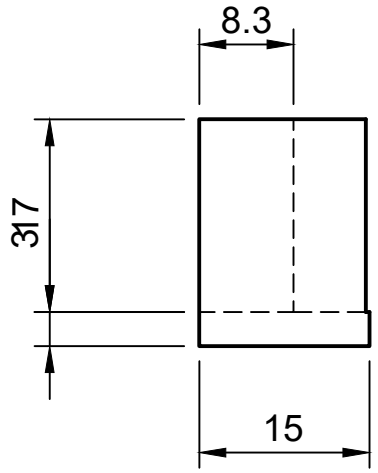


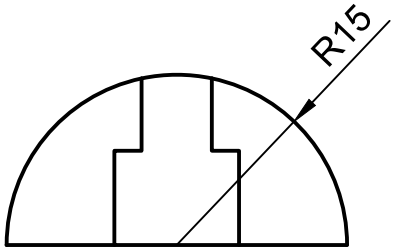


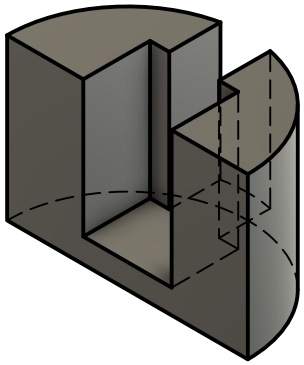


Dept. TFG	Technical reference futbolín 2.0	Created by Rodrigo Perez 19/07/2020	Approved by
		Document type plano de pieza 3D	Document status
		Title cubierta de cámara	DWG No. 9
			Rev. / Date of issue / Sheet 4/4









Dept.	Technical reference	Created by	Approved by
TFG	futbolín 2.0	Rodrigo Perez 19/07/2020	
		Document type: plano de pieza 3D	Document status
		Title: protector verificador	DWG No.: 10
			Rev. / Date of issue / Sheet: 1/1

```
/*
Página principal donde se gestionan el setup, interrupciones y servidor web entre otros.
Por Rodrigo Perez Garcia 02/08/2020
Todos los derechos reservados
*/
//_____________________________________________________________________________________
//_____________________________________LIBRERIAS_____________________________________

#include <MFRC522.h>
#include <SPI.h>
#include <PubSubClient.h>
#include "monedero.h"
#include "marcador.h"
#include "equipo.h"
#include "jugador.h"
#include "modojuego.h"
#include "porteria.h"
#include "WiFi.h"
#include "ESPAsyncWebServer.h"
#include "SPIFFS.h"


//_____________________________________________________________________________________
//_____________________________________CONSTANTES____________________________________

//pines de porterías verificador (pin 34,35 sin resistencias internas de pull up)
#define pin_haz1A 34
#define pin_haz2A 35
#define pin_haz1B 32
#define pin_haz2B 33
#define pin_verifi 25

//pines de marcador (pines I2C)
#define display_CLK 16
#define display_DIO 4

//pin para comunicacion de raspi (pin UART)
#define pin_replay 3

//pines para monedero (tarjeta,actuador y fin de carrera)
#define SS_PIN    21
#define RST_PIN   22
#define PWM_PIN 15
#define COIN_PIN 2


//_____________________________________________________________________________________
//_____________________________________VARIABLES_____________________________________

//variables para el gestor de interrupciones
volatile int cont_interrupt=0;
boolean porA_pide_gol = false;
boolean porB_pide_gol = false;
unsigned long  time_us1=0;
unsigned long  time_us2=0;
boolean monedero_resta= false;

//variable para seleccion de modo
unsigned char modoseleccionado=4;

//variables para conexion Wifi y AP
const char* ssid = "Foscam";
const char* password = "*********";
const char* ssidAP = "Futbolin_rodrigo";
const char* passwordAP = "********";
```

```
//variables para MQTT
const char* mqttServer = "***.***.***.**";
const int mqttPort = *****;
const char* mqttUser = "********************";
const char* mqttPassword = "";
char velocidadMQTT[30];
char marcadorAMQTT[20];
char marcadorBMQTT[20];

//variables para servidor web
unsigned int parametros_modo[5];
String eventoA ="";
String eventoB ="";
String eventoC ="";



//_____________________________________________________________________________
//_____________________________DEFINICION DE CLASES Y OBJETOS_____________________________

//declaracion de los 2 vias SPI
SPIClass vspi = SPIClass(VSPI);
SPIClass hspi = SPIClass(HSPI);

//declaracion de objetos y punteros de objetos de librerías propias
Monedero elmonedero(SS_PIN,RST_PIN,PWM_PIN);
Monedero* punteromonedero = &elmonedero;
Porteria porteriaA(pin_haz1A,pin_haz2A,pin_verifi);
Porteria porteriaB(pin_haz1B,pin_haz2B,pin_verifi);
Marcador elmarcador(display_CLK, display_DIO);
Marcador* punteromarcador= &elmarcador;
Modojuego elmododejuego;
Modojuego* punteromodo = &elmododejuego;

//declaracion del servidor y ajuste de IP estática
AsyncWebServer server(80);
IPAddress local_IP(192, 168, 43, 184);
IPAddress gateway(192, 168, 43, 1);
IPAddress subnet(255, 255, 255, 0);
IPAddress primaryDNS(8, 8, 8, 8);
IPAddress secondaryDNS(8, 8, 4, 4);

//declaracion de cliente MQTT
WiFiClient espClient;
PubSubClient client1(espClient);
PubSubClient* punterocliente = &client1;



//_____________________________________________________________________________
//_____________________________DEFINICION DE FUNCIONES_____________________________
//-------------------------------funciones de interrupcion-------------------------------
//funciones de interrupcion de porteria
void IRAM_ATTR interr_hazA() {
  time_us1=micros();
}
void IRAM_ATTR interr_hazB() {
  time_us1=micros();
}
void IRAM_ATTR interr_hazA2() {
  time_us2=micros();
  cont_interrupt++;
  porA_pide_gol =true;
}
void IRAM_ATTR interr_hazB2() {
  time_us2=micros();
  cont_interrupt++;
  porB_pide_gol =true;
```

```
}

//funciones de interrupcion de saldo
void IRAM_ATTR interr_coin() {
  cont_interrupt++;
  monedero_resta = true;
}


//-------------------------funcion selectora del modo de juego---------------------------
String actualizarmodo(){
  switch(parametros_modo[0]){
      case 2:
         return
elmododejuego.reydelapista(parametros_modo[2],parametros_modo[3],parametros_modo[1],punter
omarcador,punterocliente);
      break;

      case 3:
         return
elmododejuego.partidatorneo(parametros_modo[2],parametros_modo[3],parametros_modo[1],punte
romarcador,punterocliente);
      break;

      case 4:
         return
elmododejuego.partidaliga(parametros_modo[2],parametros_modo[3],parametros_modo[1],false,p
unteromarcador,punterocliente);
      break;

      default:
        return elmododejuego.partidarapida(punteromarcador,punterocliente);
      break;
  }
}


//----------------------------funciones para el servidor web-----------------------------
//funcion principal para actualizar las variables que se muestran en el servidor web
String processor(const String& var){
  Serial.println(var);
  if(var == "GOLES_A"){
    return String(elmarcador.get_goles(true));
  }
  else if(var == "GOLES_B"){
      elmarcador.get_goles(true);
    return String(elmarcador.get_goles(false));
  }
  else if(var == "NOMBRELOCAL"){
    return elmododejuego.daelnombredeequipo(0,parametros_modo[0]);
  }
  else if(var == "NOMBREVISITANTE"){
    return elmododejuego.daelnombredeequipo(1,parametros_modo[0]);
  }
  else if(var == "NOMBREMODO"){
    return darnombremodo();
  }
  else if(var == "EVENTO_A"){
    return String(eventoA);
  }
  else if(var == "EVENTO_B"){
    return eventoB;
  }
  else if(var == "EVENTO_C"){
    return eventoC;
  }
  else if(var == "CREDITOS"){
    return String(punteromonedero->ver_saldo());
```

```
  }
  else if(var == "CRONO"){
    return print_crono();
  }
  return String();
}

//funcion para actualizar la variable para mostrar en el servidor web el modo de juego
String darnombremodo(){
String elnombre ="";
  if(parametros_modo[0]==2){
    elnombre="Partida personalizada";
  }
  else if(parametros_modo[0]==3){
    elnombre="Torneo";
  }
  else if(parametros_modo[0]==4){
    elnombre="Liguilla";
  }
  else{
    elnombre="Partida rápida";
  }
  return elnombre;
}

//funcion para actualizar la variable de tiempo o goles a mostrar en el servidor web
String print_crono(){
  if(parametros_modo[0]==1){
    return "";
  }
  else if(parametros_modo[3]!=2){
    if(parametros_modo[2]>0){
      return ("a "+String(parametros_modo[2])+" goles");
    }
    else{
      return "";
    }
  }
  else{
    unsigned long lectura_crono = punteromodo->ver_crono();
    if(parametros_modo[2]*60000>lectura_crono){
      return String((parametros_modo[2]*60000-
lectura_crono)/60000)+":"+String(((parametros_modo[2]*60000-lectura_crono)/1000)%60);
    }
    else{
      return "tiempo extra";
    }
  }
}

//funcion para actualizar los eventos que se visualizan en el servidor web
void RotarListaEventos(){
  eventoC = eventoB;
  eventoB = eventoA;
  eventoA = "";
}


//---------------------funciones para pedir repeticion a Raspberry-----------------------
void genera_pulso(){
digitalWrite(pin_replay,1);
delay(15);
digitalWrite(pin_replay,0);
}
```

```
//------------------------funciones para enviar velocidad via MQTT-----------------------
void enviaMQTTmarcador(){
  sprintf(marcadorAMQTT,"{\"marcadorA\":%u}",elmarcador.get_goles(true)) ;
  client1.publish("v1/devices/me/telemetry",marcadorAMQTT);
  sprintf(marcadorBMQTT,"{\"marcadorB\":%u}",elmarcador.get_goles(false)) ;
  client1.publish("v1/devices/me/telemetry",marcadorBMQTT);
}



//_____________________________________________________________________________________
//_____________________________________SETUP___________________________________________

void setup() {
  //---------------------------inicializacion del puerto Serie-------------------------
  Serial.begin(115200);

  //----------------------inicializacion del pines, variables y objetos-----------------
  //inicializacion de pines y asignacion de interrupciones

  pinMode(pin_replay,OUTPUT);
  pinMode(COIN_PIN,INPUT_PULLUP);
  digitalWrite(pin_replay,0);
  attachInterrupt(digitalPinToInterrupt(pin_haz1A), interr_hazA, FALLING);
  attachInterrupt(digitalPinToInterrupt(pin_haz1B), interr_hazB, FALLING);
  attachInterrupt(digitalPinToInterrupt(pin_haz2A), interr_hazA2, FALLING);
  attachInterrupt(digitalPinToInterrupt(pin_haz2B), interr_hazB2, FALLING);
  attachInterrupt(digitalPinToInterrupt(COIN_PIN), interr_coin, FALLING);

  //inicializacion de array de lectura de parametros de modo de juego
  for (int i=1;i>5;i++){
    parametros_modo[i]=5;
  }
  //inicializacion de modo de juego y del display del marcador
  actualizarmodo();
  elmarcador.actualizar_display();

  //inicializacion del sistema Flash Fail System integrado en el bus SPI
  if(!SPIFFS.begin(true)){
    Serial.println("An Error has occurred while mounting SPIFFS");
    return;
  }


  //--------------------------------inicializacion de Wi-fi-------------------------------
  //configuracion de la red (con IP estática)

  if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {
    Serial.println("STA Failed to configure");
  }
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }
  Serial.println(WiFi.localIP());

  //configuracion del punto de acceso
  WiFi.softAP(ssidAP, passwordAP);
  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);
```

```
  //_____________________________________________________________________________________
  //_____________________________________Servidor web____________________________________

  //-----------funcion de lectura de parametros (servidor web--param-->ESP32)------------
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    int paramsNr = request->params();

    //if para tomar obtener y modificar el nombre del equipo
    if(paramsNr==1){
       AsyncWebParameter* p = request->getParam(0);
      if((p->name())=="loc"){
        if(parametros_modo[0]==4){
          for( int i=0;i<250;i++){
            if(punteromodo->getequipo(i,parametros_modo[0])->getIDequipo() == punteromodo-
>getequipo(0,parametros_modo[0])->getIDequipo()){
              punteromodo->getequipo(i,parametros_modo[0])->modificarnombre(p->value());
            }
          }
          for( int i=0;i<parametros_modo[1];i++){
            if(punteromodo->getequipo(i,1)->getIDequipo() == punteromodo-
>getequipo(0,parametros_modo[0])->getIDequipo()){
              punteromodo->getequipo(i,1)->modificarnombre(p->value());
            }
          }
        }
        else{
          punteromodo->getequipo(0,parametros_modo[0])->modificarnombre(p->value());
        }
      }
      else if((p->name())=="vis"){
        if(parametros_modo[0]==4){
          for( int i=0;i<250;i++){
            if(punteromodo->getequipo(i,parametros_modo[0])->getIDequipo() == punteromodo-
>getequipo(1,parametros_modo[0])->getIDequipo()){
              punteromodo->getequipo(i,parametros_modo[0])->modificarnombre(p->value());
            }
          }
          for( int i=0;i<parametros_modo[1];i++){
            if(punteromodo->getequipo(i,1)->getIDequipo() == punteromodo-
>getequipo(1,parametros_modo[0])->getIDequipo()){
              punteromodo->getequipo(i,1)->modificarnombre(p->value());
            }
          }
        }
        else{
          punteromodo->getequipo(1,parametros_modo[0])->modificarnombre(p->value());
        }
      }
    }

    //if para asignar equipo y poscion a un jugador y habilitar envio de sus estadisticas
    else if(paramsNr==3){
      String jugadornombre;
      unsigned char jugadorequipo;
      unsigned char jugadorposcion;
      for(int i=0;i<paramsNr;i++){
        AsyncWebParameter* p = request->getParam(i);
        Serial.print("Nombre Parametro: ");
        Serial.println(p->name());
        Serial.print("Valor parametro: ");
        Serial.println(p->value());
        if(i==0){
            jugadornombre = p->value();
        }
        else if(i==1){
          jugadorequipo = (p->value()).toInt();
        }
        else if(i==2){
```

```
          jugadorposcion = (p->value()).toInt();
        }
      }
      punteromodo-
>registra_jugador(jugadornombre,jugadorequipo,jugadorposcion,parametros_modo[0]);
    }

    //if para determinar el modo de juego a jugar
    else if(paramsNr>=4){
      for(int i=0;i<paramsNr;i++){
        AsyncWebParameter* p = request->getParam(i);
        Serial.print("Nombre Parametro: ");
        Serial.println(p->name());
        Serial.print("Valor parametro: ");
        Serial.println(p->value());
        parametros_modo[i]=(p->value()).toInt();
      }
      elmododejuego.resetear_modo();
      actualizarmodo();
    }

    request->send(SPIFFS, "/index.html", String(), false, processor);
  });

  //--------------------funcion para asignar el css del servidor web----------------------
  server.on("/style.css", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/style.css", "text/css");
  });

  //------------------funciones requests para envio de datos a servidor------------------
  server.on("/goles_a", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(200, "text/plain", (String(elmarcador.get_goles(true))).c_str());
  });

    server.on("/goles_b", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(200, "text/plain", (String(elmarcador.get_goles(false))).c_str());
  });

   server.on("/nombrelocal", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(200, "text/plain",
elmododejuego.daelnombredeequipo(0,parametros_modo[0]).c_str());
  });

  server.on("/nombrevisitante", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(200, "text/plain",
elmododejuego.daelnombredeequipo(1,parametros_modo[0]).c_str());
  });

  server.on("/nombremodo", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(200, "text/plain", darnombremodo().c_str());
  });

  server.on("/evento_a", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(200, "text/plain", eventoA.c_str());
  });

  server.on("/evento_b", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(200, "text/plain", String(eventoB).c_str());
  });

  server.on("/evento_c", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(200, "text/plain", eventoC);
  });

  server.on("/creditos", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(200, "text/plain", String(punteromonedero->ver_saldo()).c_str());
  });
```

```
  server.on("/crono", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(200, "text/plain", print_crono());
  });

  //-------------------------funciones mandadas por el servidor-------------------------
  server.on("/masA", HTTP_GET, [](AsyncWebServerRequest *request){
    elmarcador.inc_marcador(true);
    elmarcador.actualizar_display();
    RotarListaEventos();
    eventoA="Gol añadido manualmente al equipo local, ";
    eventoA=eventoA+actualizarmodo();
    enviaMQTTmarcador();
    request->redirect("/");
  });

  server.on("/masB", HTTP_GET, [](AsyncWebServerRequest *request){
    elmarcador.inc_marcador(false);
    elmarcador.actualizar_display();
    RotarListaEventos();
    eventoA="Gol añadido manualmente al equipo visitante, ";
    eventoA=eventoA+actualizarmodo();
    enviaMQTTmarcador();
    request->redirect("/");
  });

  server.on("/menosA", HTTP_GET, [](AsyncWebServerRequest *request){
    elmarcador.dec_marcador(true);
    elmarcador.actualizar_display();
    RotarListaEventos();
    eventoA="Gol restado manualmente al equipo local, ";
    enviaMQTTmarcador();
    request->redirect("/");
  });

  server.on("/menosB", HTTP_GET, [](AsyncWebServerRequest *request){
    elmarcador.dec_marcador(false);
    elmarcador.actualizar_display();
    RotarListaEventos();
    eventoA="Gol restado manualmente al equipo visitante, ";
    enviaMQTTmarcador();
    request->redirect("/");
  });

  server.on("/playpause", HTTP_GET, [](AsyncWebServerRequest *request){
    punteromodo->pausar_reanudar_crono(parametros_modo[3]);
    request->redirect("/");
  });

  //-------------------------------inicializacion del servidor----------------------------
  server.begin();

  //_____________________________________________________________________________________
  //_____________________________________Setup MQTT______________________________________
  client1.setServer(mqttServer, mqttPort);
  while (!client1.connected()) {
    Serial.println("Conectando a Broquer MQTT...");
    if (client1.connect("IOT-ESP32", mqttUser, mqttPassword )) {
      Serial.println("conectado");
    }
    else {
      Serial.print("conexion fallida ");
      Serial.print(client1.state());
      delay(1000);
    }
  }
}
```

```
//_____________________________________________________________________________________
//_____________________________________Bucle____________________________________________
void loop() {
  //----------------------------Gestor de interrupciones-------------------------------
  if(cont_interrupt>0){
    noInterrupts();
    cont_interrupt--;

    //gestion de interrupcion de porteria A
    if (porA_pide_gol==true){
      float vel_bola1=porteriaA.revbarrera(5,time_us1,time_us2);
      elmarcador.escribir_mensaje(1);
      genera_pulso();
      if(porteriaA.revgol()==true){
        elmarcador.inc_marcador(true);
        genera_pulso();
        sprintf(velocidadMQTT,"{\"vel_bola1\": %f}",vel_bola1) ;
        client1.publish("v1/devices/me/telemetry",velocidadMQTT);
        RotarListaEventos();
        eventoA="GOOOOL! la velocidad de la bola ha sido de: "+String(vel_bola1)+"km/h. ";
        eventoA=eventoA+actualizarmodo();
        enviaMQTTmarcador();
      }
      elmarcador.actualizar_display();
      porA_pide_gol=false;
      time_us1=0;
      time_us2=0;
    }

    //gestion de interrupcion de porteria B
    if (porB_pide_gol==true){
      float vel_bola2=porteriaB.revbarrera(5,time_us1,time_us2);
      elmarcador.escribir_mensaje(1);
      genera_pulso();
      if(porteriaB.revgol()==true){
        elmarcador.inc_marcador(false);
        genera_pulso();
        sprintf(velocidadMQTT,"{\"vel_bola2\": %f}",vel_bola2) ;
        client1.publish("v1/devices/me/telemetry",velocidadMQTT);
        RotarListaEventos();
        eventoA="GOOOOL! la velocidad de la bola ha sido de: "+String(vel_bola2)+"km/h.
";
        eventoA=eventoA+actualizarmodo();
        enviaMQTTmarcador();
      }
      elmarcador.actualizar_display();
      porB_pide_gol=false;
      time_us1=0;
      time_us2=0;
    }

    //gestion de interrupcion de final de carrera de monedero
    if(monedero_resta == true){
      Serial.println("restando credito!");
      punteromonedero->restar_credito();
      monedero_resta = false;
      delay(100);
    }

    interrupts();
  }
```

```
  //----------------------------Verificacion de monedero--------------------------------
  if(punteromonedero->actualizar_credito()==true){
    RotarListaEventos();
    eventoA="Saldo de restante de la tarjeta: "+String(punteromonedero-
>credito_ultima_tarjeta());
  };
  punteromonedero->actualizar_actuador();

  //------------Condicion de rotacion  de equipos para partidas a tiempo---------------
  if ((parametros_modo[3]==2) and ((punteromodo->finpartido(true, 2, parametros_modo[2],
punteromarcador))or(punteromodo->finpartido(false, 2, parametros_modo[2],
punteromarcador)))){
    RotarListaEventos();
    eventoA="Finaaaaal del partido, ";
    eventoA=eventoA+actualizarmodo();
  }

  //--------------------------------Loop de cliente MQTT--------------------------------
  client1.loop();
}
```

```
/*
Equipo.h - Libreria para administrar los atributos y funciones de equipo
Por Rodrigo Perez Garcia 04/08/2020
Todos los derechos reservados
*/
#ifndef Equipo_h
#define Equipo_h
#include "Arduino.h"
#include "jugador.h"

//--------------------------------definicion de la clase-------------------------
--------
class Equipo
{
 public:
  Equipo();
  unsigned int getdato(unsigned char eldato);
  String danombreequipo();
  unsigned char quejugadoreshay();
  Jugador* getjugador(unsigned char posicionjug);
  unsigned long getIDequipo();
  void modificarnombre(String elnuevonombre);
  void sumapartida(boolean fueganada, unsigned int golesafavor, unsigned int
golesencontra);
  void asignajugador( unsigned char su_posicion);
  void asignarIDequipo(unsigned long laID);
  void eliminajugador( boolean esdefensa);
  void printestadisticas();

 private:
        String nombreequipo;
        unsigned int num_jugadas;
        unsigned int num_ganadas;
        unsigned int num_goles_a_favor;
        unsigned int num_goles_en_contra;
        unsigned long equipoID;
        boolean defensa_select;
        boolean delantero_select;
        Jugador eldefensa;
        Jugador eldelantero;
} ;

#endif
```

```
/*
Equipo.cpp -Libreria para administrar los atributos y funciones de equipo
Por Rodrigo Perez Garcia 04/08/2020
Todos los derechos reservados
*/
#include "equipo.h"
#include "jugador.h"

//---------------------------------constructor de la clase-------------------------------
Equipo::Equipo():eldefensa(),eldelantero(){
  equipoID=0;
  nombreequipo = "Equipo"+String(equipoID);
  num_jugadas=0;
  num_ganadas=0;
  num_goles_a_favor=0;
  num_goles_en_contra=0;
  defensa_select=false;
  delantero_select=false;
}



//-------------------------------------funciones get-------------------------------------
//funcion para obtener los datos (y poder ordenarlos en modo liga)
unsigned int Equipo::getdato(unsigned char eldato){
  if(eldato==1){return num_ganadas;}
  else if(eldato==2){return num_jugadas;}
  else if(eldato==3){return num_goles_a_favor;}
  else if(eldato==4){return num_goles_en_contra;}
  else if(eldato==5){return equipoID;}
  else{return 0;}
}


//funcion para dar el nombre del equipo
String Equipo::danombreequipo(){
  return nombreequipo;
}


//funcion para conocer los jugadores registrados en el equipo
unsigned char Equipo::quejugadoreshay(){
  if (defensa_select==true and delantero_select==true){
    return 3;
  }
  else if (delantero_select==true){
    return 2 ;
  }
  else if (defensa_select==true){
    return 1 ;
  }
  else{
    return 0;
  }
}


//funcion para obtener un jugador de los 2 asignados al equipo
Jugador* Equipo::getjugador(unsigned char posicionjug){
  if (posicionjug==1){
    return &eldefensa;
  }
  else{
    return &eldelantero;
  }
}
```

```
//funcion para obtener el ID (el ID se usa para organizar listas de equipos)
unsigned long Equipo::getIDequipo(){
  return equipoID;
}



//-------------------------------------funciones set-------------------------------------
//funcion para modificar el nombre del equipo
void Equipo::modificarnombre(String elnuevonombre){
  nombreequipo = elnuevonombre;
}


//funcion para actualizar los datos de partida
void Equipo::sumapartida(boolean fueganada, unsigned int golesafavor, unsigned int
golesencontra){
  num_jugadas = num_jugadas + 1;
  num_goles_a_favor = num_goles_a_favor + golesafavor;
  num_goles_en_contra = num_goles_en_contra + golesencontra;
  if(fueganada==true){
    num_ganadas=num_ganadas+1;
  }
}


//funcion para registrar la posicion de jugadores y habilitar el envio de sus estadisticas
void Equipo::asignajugador(unsigned char su_posicion){
  if (su_posicion==1){
    //eldefensa = eljugador;
    defensa_select=true;
    eldefensa.seleccionaposicion(1);
  }
  else{
    //eldelantero = eljugador;
    delantero_select=true;
    eldelantero.seleccionaposicion(2);
  }
}


//funcion para asignar ID (y así poder utilizar algo para ordenar equipos)
void Equipo::asignarIDequipo(unsigned long laID){
  equipoID = laID;
}


//funcion para eliminar a un jugador de su posicion (no se usa)
void Equipo::eliminajugador( boolean esdefensa){
  if (esdefensa==true){
    //habria que actualizar los datos de jugador
    defensa_select=false;
    eldefensa.seleccionaposicion(0);
  }
  if (esdefensa==false){
    //habria que actualizar los datos de jugador
    delantero_select=false;
    eldelantero.seleccionaposicion(0);
  }
}
```

```
//-------------------------------funciones de verificacion-------------------------------
//funcion para imprimir las estadisticas de los equipos (no se usa)
void Equipo::printestadisticas(){
  Serial.println("---------------ESTADISTICAS DE EQUIPO------------------");
  Serial.print("El equipo: ");
  Serial.print(nombreequipo);
  Serial.println(" tiene estas estadisticas: ");
  Serial.print("Partidas jugadas: ");
  Serial.println(num_jugadas);
  Serial.print("Partidas ganadas: ");
  Serial.println(num_ganadas);
  Serial.print("Goles favor/contra: ");
  Serial.print(num_goles_a_favor);
  Serial.print("/");
  Serial.println(num_goles_en_contra);
  Serial.println("--------------ESTADISTICAS DEl DEFENSA------------------");
  if(defensa_select==true){
    eldefensa.printestadisticas();
  }
  else{
    Serial.println("el defensor no se ha registrado");
  }
  Serial.println("-------------ESTADISTICAS DEl DELANTERO------------------");
  if(delantero_select==true){
    eldelantero.printestadisticas();
  }
  else{
    Serial.println("el delantero no se ha registrado");
  }
  Serial.println("");
}
```

```
/*
jugador.h - Libreria para determinar las caracteristicas de jugador
Por Rodrigo Perez Garcia 06/04/2020
Todos los derechos reservados
*/

#ifndef Jugador_h
#define Jugador_h

#include "Arduino.h"
#include <PubSubClient.h>

//--------------------------------definicion de la clase----------------------------------
class Jugador
{
 public:
  Jugador();
  String printnombre();
  void modificarnombre(String elnuevonombre);
  void seleccionaposicion(unsigned char laposicion);
  void sumapartidajugador(boolean fueganada, unsigned int golesafavor, unsigned int
golesencontra, PubSubClient* &clienteMQTT);
  void printestadisticas();

 private:
        String nombrejugador;
        unsigned int posiciondejuego;
        unsigned int partidas_jugadas;
        unsigned int partidas_ganadas;
        unsigned int goles_a_favor;
        unsigned int goles_en_contra;
        unsigned int partidas_ganadas_defensa;  //no se usa
        unsigned int partidas_ganadas_ataque;   //no se usa
        unsigned int partidas_perdidas_defensa; //no se usa
        unsigned int partidas_perdidas_ataque;  //no se usa
        unsigned int goles_marcados_ataque;     //no se usa
        unsigned int goles_sufridos_defensa;    //no se usa
} ;

#endif
```

```
/*
jugador.cpp - Libreria para gestion de estadisticas de los jugadores de futbolin
Por Rodrigo Perez Garcia 04/08/2020
Todos los derechos reservados
*/
#include "jugador.h"
#include <PubSubClient.h>

//--------------------------------constructor de la clase-------------------------------
Jugador::Jugador(){
  nombrejugador = "JugadorX";
  posiciondejuego=0;
  partidas_jugadas=0;
  partidas_ganadas=0;
  goles_a_favor=0;
  goles_en_contra=0;
  partidas_ganadas_defensa=0;
  partidas_ganadas_ataque=0;
  partidas_perdidas_defensa=0;
  partidas_perdidas_ataque=0;
  goles_marcados_ataque=0;
  goles_sufridos_defensa=0;
}

//-------------------------------------funciones get-------------------------------------
//funcion para obtener el nombre de jugador
String Jugador::printnombre(){
  //Serial.print(nombrejugador);
  return nombrejugador;
}



//-------------------------------------funciones set-------------------------------------
//funcion para modificar el nombre
void Jugador::modificarnombre(String elnuevonombre){
//void Jugador::modificarnombre(String elnuevonombre){
  nombrejugador=elnuevonombre;
}


//funcion para asignar una posicion al jugador (habilita flags para envio de sus datos)
void Jugador::seleccionaposicion(unsigned char laposicion){
  if(laposicion==1){
    posiciondejuego = 1; //juega de defensa
  }
  else if(laposicion==2){
    posiciondejuego = 2; //juega de ataque
  }
  else{
    posiciondejuego = 3; //su posicion se desconoce
  }
}


//funcion para cargar las estadisticas de jugador al broquer via MQTT
void Jugador::sumapartidajugador(boolean fueganada, unsigned int golesafavor, unsigned int
golesencontra, PubSubClient* &clienteMQTT){
  char array30MQTT[30];
  char array20MQTT[20];
  sprintf(array30MQTT,"{\"player\": %s}",printnombre()) ;
  clienteMQTT->publish("v1/devices/me/telemetry",array30MQTT);
  sprintf(array20MQTT,"{\"posicion\": %u}",posiciondejuego);
  clienteMQTT->publish("v1/devices/me/telemetry",array20MQTT);
  partidas_jugadas = partidas_jugadas + 1;
  //clienteMQTT->publish("v1/devices/me/telemetry","{\"pj\": 1}");
  if(fueganada==true){
```

```
    partidas_ganadas=partidas_ganadas+1;
    clienteMQTT->publish("v1/devices/me/telemetry","{\"pg\": 1}");
  }
  else{
    clienteMQTT->publish("v1/devices/me/telemetry","{\"pg\": 0}");
  }
  //actualizacion de goles
  goles_a_favor = goles_a_favor + golesafavor;
  goles_en_contra = goles_en_contra + golesencontra;
  sprintf(array20MQTT,"{\"gf\": %u}",golesafavor) ;
  clienteMQTT->publish("v1/devices/me/telemetry",array20MQTT);
  sprintf(array20MQTT,"{\"gc\": %u}",golesencontra) ;
  clienteMQTT->publish("v1/devices/me/telemetry",array20MQTT);
}



//--------------------------------funciones de verificacion-------------------------------
//funcion para imprimir las estadisticas de los jugadores (no se usa)
void Jugador::printestadisticas(){;
  Serial.print("El jugador ");
  Serial.print(nombrejugador);
  Serial.println(" tiene estas estadisticas: ");
  Serial.print("Partidas jugadas: ");
  Serial.println(partidas_jugadas);
  Serial.print("Partidas ganadas: ");
  Serial.println(partidas_ganadas);
  Serial.print("Goles favor/contra: ");
  Serial.print(goles_a_favor);
  Serial.print("/");
  Serial.println(goles_en_contra);
  Serial.print("Has ganado como defensor: ");
  Serial.print(partidas_ganadas_defensa);
  Serial.print(" de ");
  Serial.println(partidas_ganadas_defensa+partidas_perdidas_defensa);
  Serial.print("Y te han marcado : ");
  Serial.print(goles_sufridos_defensa);
  Serial.println(" goles");
  Serial.print("Has ganado como atacante: ");
  Serial.print(partidas_ganadas_ataque);
  Serial.print(" de ");
  Serial.println(partidas_ganadas_ataque+partidas_perdidas_ataque);
  Serial.print("Y marcaste : ");
  Serial.print(goles_marcados_ataque);
  Serial.println(" goles");
}
```

```
/*
marcador.h - Libreria para control del marcador y del display de marcador
Por Rodrigo Perez Garcia 04/08/2020
Todos los derechos reservados
*/

#ifndef Marcador_h
#define Marcador_h

#include <Arduino.h>
#include <TM1637Display.h>

//--------------------------------definicion de la clase----------------------------------
class Marcador
{
public:
  Marcador(int pin_CLK, int pin_DIO);
  int get_goles(boolean delocal);
  void inc_marcador(boolean delocal);
  void dec_marcador(boolean delocal);
  void reset_marcador();
  void actualizar_display();
  void escribir_mensaje(int num_mensaje);
private:
        int _pin_CLK;
        int _pin_DIO;
        unsigned int goles_local;
        unsigned int goles_visitante;
} ;

#endif
```

```
/*
marcador.cpp - Libreria para control del marcador y del display de marcador
Por Rodrigo Perez Garcia 04/08/2020
Todos los derechos reservados
*/
#include "marcador.h"
#include <Arduino.h>
#include <TM1637Display.h>

//---------------------------------constructor de la clase--------------------------------
Marcador::Marcador(int pin_CLK, int pin_DIO){
  _pin_CLK = pin_CLK;
  _pin_DIO = pin_DIO;
  goles_local = 0;
  goles_visitante = 0;
}



//-------------------------------------funciones get-------------------------------------
//funcion para obtener el valor de goles marcados en una porteria
int Marcador::get_goles(boolean delocal){
  if(delocal==true){
    return goles_local;
  }
  else{
    return goles_visitante;
  }
}



//-------------------------------------funciones set-------------------------------------
//funcion para incrementar una unidad los goles marcados en una porteria
void Marcador::inc_marcador(boolean delocal){
  if(delocal==true){
    goles_local=goles_local+1;
  }
  else{
    goles_visitante=goles_visitante+1;
  }
}



//-------------------------------------funciones set-------------------------------------
//funcion para decrementar una unidad los goles marcados en una porteria
void Marcador::dec_marcador(boolean delocal){
  if(delocal==true){
    if (goles_local>=1){
      goles_local=goles_local-1;
    }
  }
  else{
    if(goles_visitante>=1){
      goles_visitante=goles_visitante-1;
    }
  }
}

//funcion para resetear los goles marcados en ambas porterias
void Marcador::reset_marcador(){
  goles_local=0;
  goles_visitante=0;
}
```

```
//funcion para actualizar el display y visualizar los goles marcados en ambas porterias
void Marcador::actualizar_display(){
  uint8_t segto;
  TM1637Display display(_pin_CLK, _pin_DIO);
  display.setBrightness(7, true);
  display.showNumberDec(goles_local,false,2,0);
  display.showNumberDec(goles_visitante,false,2,2);
  segto = 0x80 | display.encodeDigit(goles_local%10);
  display.setSegments(&segto, 1, 1);
}


//funcion para actualizar el display y visualizar un mensaje como GOOL
void Marcador::escribir_mensaje(int num_mensaje){
  TM1637Display display(_pin_CLK, _pin_DIO);
  display.setBrightness(7, true);
  if(num_mensaje==1){
    uint8_t data[] = { 0b00111101, 0b00111111, 0b00111111, 0b00111000 }; //GOOL
    display.setSegments(data);
  }
  else if(num_mensaje==2){
    uint8_t data[] = { 0b01110111, 0b01010100, 0b00111110, 0b00111000 }; //ANUL
    display.setSegments(data);
  }
  else if(num_mensaje==3){
    uint8_t data[] = { 0b01110001, 0b00110000, 0b01010100, 0b00000000 }; //FIN
    display.setSegments(data);
  }
  else if(num_mensaje==4){
    uint8_t data[] = { 0b01110110, 0b00111111, 0b00111000, 0b01110111 }; //HOLA
    display.setSegments(data);
  }
  else{
    uint8_t data[] = { 0x00, 0x00, 0x00, 0x00 };
    display.setSegments(data);
  }
}
```

```
/*
modojuego.h - Librería empleada para gestionar las rotaciones de los distintos
                equipos que se seleccionan en el servidor web.
Por Rodrigo Perez Garcia 04/08/2020
Todos los derechos reservados
*/
#ifndef Modojuego_h
#define Modojuego_h

#include <Arduino.h>
#include "marcador.h"
#include "equipo.h"
#include <Chrono.h>
#include <PubSubClient.h>

//--------------------------------definición de la clase----------------------------------
class Modojuego
{
 public:
  Modojuego();
  unsigned long ver_crono();
  boolean finpartido(boolean eslocal, unsigned char agoles, unsigned char x_cambio,
Marcador* &marcadorpartida);
  Equipo* getequipo(unsigned char su_posicion, unsigned char numerodelmodo);
  String daelnombredeequipo(unsigned char laposicion, unsigned char numerodelmodo);
  void resetear_modo();
  void pausar_reanudar_crono(unsigned char agoles);
  void registra_jugador(String nombrejugador, unsigned char queequipo, unsigned char
queposicion, unsigned char numerodelmodo);
  void actualizaajugadores(Equipo elequipo, boolean fueganada, unsigned int golesafavor,
unsigned int golesencontra, PubSubClient* &clienteMQTT);
  void organizarequipos(unsigned char numeroequipos);
  void reydelapistarotar(boolean ganaprimeroenlista);
  void torneorotar(boolean ganaprimeroenlista);
  void obtenerlistadepartidos(unsigned char numerodequipos, boolean idayvuelta);
  void ordenar_clasificacion();
  String partidarapida(Marcador* &marcadorpartida,PubSubClient* &clienteMQTT);
  String reydelapista(unsigned int x_cambio, unsigned char agoles, unsigned char
numerodeequipos, Marcador* &marcadorpartida,PubSubClient* &clienteMQTT);
  String partidatorneo(unsigned int x_cambio, unsigned char agoles, unsigned char
numerodeequipos, Marcador* &marcadorpartida,PubSubClient* &clienteMQTT);
  String partidaliga( unsigned char x_cambio, unsigned char agoles, unsigned char
numerodeequipos, boolean idayvuelta, Marcador* &marcadorpartida,PubSubClient*
&clienteMQTT);
  void printquienjuega(boolean deliga);

 private:
   Chrono crono;
   boolean preparada;
   boolean jugando;
   unsigned char numequipos;
   unsigned char equiposinvictos;
   unsigned char modoseleccionado;
   Equipo listaequipos[256];
   Equipo listapartidosliga[250];
   unsigned int contadorpartidosliga;
   unsigned int numpartidos;
} ;

#endif
```

```
/*
modojuego.cpp - Libreria empleada para gestionar las rotaciones de los distintos
                equipos que se seleccionan en el servidor web.
Por Rodrigo Perez Garcia 04/08/2020
Todos los derechos reservados
*/
#include "marcador.h"
#include "modojuego.h"
#include "equipo.h"
#include <Chrono.h>
#include <PubSubClient.h>

//--------------------------------constructor de la clase-------------------------------
Modojuego::Modojuego(){
  preparada=false;
  jugando=false;
  numequipos=2;
}



//-------------------------------------funciones get-------------------------------------
//funcion para obtener el tiempo de partida transcurrido
unsigned long Modojuego::ver_crono(){
  return crono.elapsed();
}


//funcion para conocer si la partida ha finalizado (puede finalizar por goles o tiempo)
boolean Modojuego::finpartido(boolean eslocal, unsigned char agoles, unsigned char
x_cambio, Marcador* &marcadorpartida){
  boolean resultado;
  if(agoles==1){
    if(eslocal==true){
      resultado=(marcadorpartida->get_goles(true) >= x_cambio);
    }
    else{
      resultado=(marcadorpartida->get_goles(false) >= x_cambio);
    }
  }
  else{
    if(eslocal==true){
      resultado=((marcadorpartida->get_goles(true) > marcadorpartida->get_goles(false))
and (crono.hasPassed(60000*x_cambio)==true));
    }
    else{
      resultado=((marcadorpartida->get_goles(true) < marcadorpartida->get_goles(false))
and (crono.hasPassed(60000*x_cambio)==true));
    }
  }
  return resultado;
}


//funcion para obtener un equipo de de los arrays de equipos
Equipo* Modojuego::getequipo(unsigned char su_posicion, unsigned char numerodelmodo){
  Equipo* punteroequipo;
  if(numerodelmodo==4){
    punteroequipo = &listapartidosliga[su_posicion];
  }
  else{
    punteroequipo = &listaequipos[su_posicion];
  }
  return punteroequipo;
}
```

```
//funcion para obtener el nombre del equipo de los arrays de equipos
String Modojuego::daelnombredeequipo(unsigned char laposicion , unsigned char
numerodelmodo){
  if(numerodelmodo==4){
    return listapartidosliga[laposicion].danombreequipo();
  }
  else{
    return listaequipos[laposicion].danombreequipo();
  }
}



//-------------------------------------funciones set-------------------------------------
//funcion para resetear los flags indicadores del estado del modo de juego
void Modojuego::resetear_modo(){
   preparada=false;
   jugando=false;
}


//funcion empleada para pausar/reanudar el crono de partida (se usa en servidor web)
void Modojuego::pausar_reanudar_crono(unsigned char agoles){
  if(agoles==2){
    if(crono.isRunning()==true){
      crono.stop();
    }
    else{
      crono.resume();
    }
  }
}


// funcion que registra la posicion de jugadores en un equipo (se usa en servidor web)
void Modojuego::registra_jugador(String nombrejugador, unsigned char queequipo, unsigned
char queposicion, unsigned char numerodelmodo){
  if(numerodelmodo==4){
    for( int i=0;i<250;i++){
      if(listapartidosliga[i].getIDequipo() ==
listapartidosliga[queequipo].getIDequipo()){
        getequipo(i,4)->asignajugador(queposicion);
        getequipo(i,4)->getjugador(queposicion)->modificarnombre(nombrejugador);
      }
    }
  }
  else{
    listaequipos[queequipo].asignajugador(queposicion);
    listaequipos[queequipo].getjugador(queposicion)->modificarnombre(nombrejugador);
  }
}


//funcion para comprobar los jugadores registrados y enviar sus estadisticas por MQTT
void Modojuego::actualizaajugadores(Equipo elequipo, boolean fueganada, unsigned int
golesafavor, unsigned int golesencontra, PubSubClient* &clienteMQTT){
  unsigned char datodenumjug = elequipo.quejugadoreshay();
  Serial.println("Actualizando a jugadores:"+String(elequipo.quejugadoreshay()));
  if(datodenumjug==3){
    elequipo.getjugador(1)-
>sumapartidajugador(fueganada,golesafavor,golesencontra,clienteMQTT);
    elequipo.getjugador(2)-
>sumapartidajugador(fueganada,golesafavor,golesencontra,clienteMQTT);
  }
  else if(datodenumjug==2){
    elequipo.getjugador(2)-
>sumapartidajugador(fueganada,golesafavor,golesencontra,clienteMQTT);
  }
```

```
  else if(datodenumjug==1){
    elequipo.getjugador(1)-
>sumapartidajugador(fueganada,golesafavor,golesencontra,clienteMQTT);
  }
}



//----------------------funciones para modificar los arrays de equipo--------------------
//funcion para inicializar el array de lista de equipos
void Modojuego::organizarequipos(unsigned char numeroequipos){
  numequipos = numeroequipos;
  Equipo elequipo;
  for(int i=0;i<numeroequipos;i++){
    String elnombredeequipo="Equipo"+String(i+1);
    elequipo.modificarnombre(elnombredeequipo);
    elequipo.asignarIDequipo(i+1);
    listaequipos[i]=elequipo;
  }
}


//funcion para rotar el array de lista de equipos por las leyes del rey de la pista
void Modojuego::reydelapistarotar(boolean ganaprimeroenlista){
  if (ganaprimeroenlista==true){
    listaequipos[numequipos]=listaequipos[1];
    for(int i = 1; i<numequipos;i++){
      listaequipos[i]=listaequipos[i+1];
    }
  }
  else{
   listaequipos[numequipos]=listaequipos[0];
   listaequipos[0]=listaequipos[1];
    for(int i = 1; i<numequipos;i++){
      listaequipos[i]=listaequipos[i+1];
    }
  }
}


//funcion para rotar el array de lista de equipos por las leyes de torneo
void Modojuego::torneorotar(boolean ganaprimeroenlista){
  if(equiposinvictos>1){
    equiposinvictos = equiposinvictos - 1;
    if (ganaprimeroenlista==true){
      listaequipos[equiposinvictos+1]=listaequipos[0];
      listaequipos[equiposinvictos+2]=listaequipos[1];
      for(int i = 2;i<=equiposinvictos+1;i++){
        listaequipos[i-2]=listaequipos[i];
      }
    }
    else{
      listaequipos[equiposinvictos+1]=listaequipos[1];
      listaequipos[equiposinvictos+2]=listaequipos[0];
      for(int i = 2;i<=equiposinvictos+1;i++){
        listaequipos[i-2]=listaequipos[i];
      }
    }
  }
  else{
    Serial.print("-----------> El equipo: ");
    Serial.print(listaequipos[0].danombreequipo());
    Serial.println(" es el ganador <-----------");
    for(int i = 3;i<=numequipos+1;i++){
        listaequipos[i-2]=listaequipos[i];
    }
  }
}
```

```
//funcion para inicializar el array de lista de partidos (para modo liga)
void Modojuego::obtenerlistadepartidos(unsigned char numerodequipos, boolean idayvuelta){
  organizarequipos(numerodequipos);
  numequipos = numerodequipos;
  if((idayvuelta==true and numerodequipos<=11)or (idayvuelta==false and
numerodequipos<=16)){
    unsigned char numpartidos = numerodequipos*(numerodequipos-1);
    if(idayvuelta==false){
      numpartidos = numpartidos/2;
    }
    unsigned char listarotacion[numerodequipos];
    unsigned char listalocales[numpartidos];
    unsigned char listavisitantes[numpartidos];
    unsigned char auxiliar;
    for (int x=0;x<numerodequipos;x++){
      listarotacion[x]=x+1;
    }
    for(int i=0;i<numpartidos;i++){
      if(((i+1)%numerodequipos) == 0){
        listalocales[i] = numerodequipos;
      }
      else{
        listalocales[i] = ((i+1)%numerodequipos);
      }
      if ((i)%numequipos == 0){
        auxiliar = listarotacion[numerodequipos-1];
        for (int j=numerodequipos-1; j>0 ; j--){
          listarotacion[j]=listarotacion[j-1];
        }
        listarotacion[0] = auxiliar;
        listavisitantes[i] = listarotacion[0];
      }
      else{
        listavisitantes[i] = listarotacion[(i)%numequipos];
      }
      for(int j =0;j<numequipos;j++){
        if (listaequipos[j].getIDequipo() == listalocales[i]){
          listapartidosliga[2*i]=listaequipos[j];
        }
      }
      for(int j =0;j<numequipos;j++){
        if (listaequipos[j].getIDequipo() == listavisitantes[i]){
          listapartidosliga[2*i+1]=listaequipos[j];
        }
      }
    }
  }
  else{
    Serial.println("Porfavor, selecciona un numero de jugadores inferior.");
  }
}


//funcion para ordenar la lista de partidos segun los resultados (para modo liga)
void Modojuego::ordenar_clasificacion(){
  Equipo equipoauxiliar;
  Equipo equipoenblanco;
  unsigned int posicion_aux = 0;
  for(int i=0;i<numequipos;i++){
    equipoauxiliar=equipoenblanco;
    for(int j=i;j<numequipos;j++){
      if(listaequipos[j].getdato(1)>equipoauxiliar.getdato(1)){
        equipoauxiliar =listaequipos[j];
        posicion_aux=j;
      }
      else if(listaequipos[j].getdato(1)==equipoauxiliar.getdato(1)){
```

```
        if((listaequipos[j].getdato(2)<equipoauxiliar.getdato(2)and
equipoauxiliar.getdato(2)!=0)or(listaequipos[j].getdato(2)>equipoauxiliar.getdato(2)and
equipoauxiliar.getdato(2)==0)){
          equipoauxiliar =listaequipos[j];
          posicion_aux=j;
        }
        else if(listaequipos[j].getdato(2)==equipoauxiliar.getdato(2)){
          if(listaequipos[j].getdato(3)>equipoauxiliar.getdato(3)){
            equipoauxiliar =listaequipos[j];
            posicion_aux=j;
          }
          else if(listaequipos[j].getdato(3)==equipoauxiliar.getdato(3)){
            if(listaequipos[j].getdato(4)<equipoauxiliar.getdato(4)){
              equipoauxiliar =listaequipos[j];
              posicion_aux=j;
            }
            else if(listaequipos[j].getdato(4)==equipoauxiliar.getdato(4)){
              if(listaequipos[j].getdato(5)<equipoauxiliar.getdato(5)){
                equipoauxiliar =listaequipos[j];
                posicion_aux=j;
              }
            }//else if dato gc iguales
          }//else if dato gf iguales
        }//else if num partidos iguales
      }//else if partidos gandos iguales
    }//for j
    for(int a=posicion_aux;a>i;a--){
      listaequipos[a]=listaequipos[a-1];
    }
    listaequipos[i]=equipoauxiliar;
  }
}



//-------------------------------funciones de modo de juego-----------------------------
//modo partida rapida
String Modojuego::partidarapida(Marcador* &marcadorpartida, PubSubClient* &clienteMQTT){
  if(preparada == false and jugando==false){
    crono.restart();
    crono.resume();
    organizarequipos(2);
    preparada = true;
    jugando = true;
    clienteMQTT->publish("v1/devices/me/telemetry","{\"modo1\": 1}");
  }
  if(preparada == true and jugando==true){
      actualizaajugadores(listaequipos[1],true,marcadorpartida-
>get_goles(false),marcadorpartida->get_goles(true),clienteMQTT);
      actualizaajugadores(listaequipos[0],false,marcadorpartida-
>get_goles(true),marcadorpartida->get_goles(false),clienteMQTT);
  }
  if(preparada == true and jugando==false){
    crono.stop();
    //partida finalizada . Reiniciar?(en este caso imposible)
  }
  return " ";
}
```

```
//modo rey de la pista
String Modojuego::reydelapista(unsigned int x_cambio, unsigned char agoles, unsigned char
numerodeequipos, Marcador* &marcadorpartida, PubSubClient* &clienteMQTT){
  String mensaje_a_enviar="";
  if(preparada == false and jugando==false){
    numequipos=numerodeequipos;
    preparada = true;
    jugando=true;
    organizarequipos(numerodeequipos);
    printquienjuega(false);
    marcadorpartida->reset_marcador();
    crono.restart();
    crono.resume();
    if(agoles==1){
      clienteMQTT->publish("v1/devices/me/telemetry","{\"modo2\": 1}");
    }
    else{
      clienteMQTT->publish("v1/devices/me/telemetry","{\"modo2\": 2}");
    }
  }
  if(preparada == true and jugando==true){
    marcadorpartida->actualizar_display();
    if(finpartido(true, agoles, x_cambio, marcadorpartida)==true){
      Serial.print("Ha ganado el equipo local: ");
      Serial.print(listaequipos[0].danombreequipo());
      listaequipos[0].sumapartida(true,marcadorpartida->get_goles(true),marcadorpartida-
>get_goles(false));
      listaequipos[1].sumapartida(false,marcadorpartida->get_goles(false),marcadorpartida-
>get_goles(true));
      actualizaajugadores(listaequipos[0],true,marcadorpartida-
>get_goles(true),marcadorpartida->get_goles(false),clienteMQTT);
      actualizaajugadores(listaequipos[1],false,marcadorpartida-
>get_goles(false),marcadorpartida->get_goles(true),clienteMQTT);
      reydelapistarotar(true);
      printquienjuega(false);
      marcadorpartida->reset_marcador();
      crono.restart();
      marcadorpartida->actualizar_display();
      mensaje_a_enviar=listaequipos[0].danombreequipo()+" es el ganador, juega contra el:
"+listaequipos[1].danombreequipo()+". "+listaequipos[2].danombreequipo()+" ves
calentando...";
    }
    else if(finpartido(false, agoles, x_cambio, marcadorpartida)==true){
      Serial.print("Ha ganado el equipo vistante: ");
      Serial.print(listaequipos[1].danombreequipo());
      listaequipos[1].sumapartida(true, marcadorpartida->get_goles(false),marcadorpartida-
>get_goles(true));
      listaequipos[0].sumapartida(false,marcadorpartida->get_goles(true),marcadorpartida-
>get_goles(false));
      actualizaajugadores(listaequipos[1],true,marcadorpartida-
>get_goles(false),marcadorpartida->get_goles(true),clienteMQTT);
      actualizaajugadores(listaequipos[0],false,marcadorpartida-
>get_goles(true),marcadorpartida->get_goles(false),clienteMQTT);
      reydelapistarotar(false);
      printquienjuega(false);
      marcadorpartida->reset_marcador();
      crono.restart();
      marcadorpartida->actualizar_display();
      mensaje_a_enviar=listaequipos[0].danombreequipo()+" es el ganador, juega contra el:
"+listaequipos[1].danombreequipo()+". "+listaequipos[2].danombreequipo()+" ves
calentando...";
    }
  }
  if(preparada == true and jugando==false){
    crono.stop();
  }
  return mensaje_a_enviar;
}
```

```
//modo torneo
String Modojuego::partidatorneo(unsigned int x_cambio,unsigned char agoles, unsigned char
numerodeequipos, Marcador* &marcadorpartida, PubSubClient* &clienteMQTT){
  String mensaje_a_enviar="";
  if(preparada == false and jugando==false){
    numequipos=numerodeequipos;
    equiposinvictos=numerodeequipos;
    preparada = true;
    jugando=true;
    organizarequipos(numerodeequipos);
    printquienjuega(false);
    marcadorpartida->reset_marcador();
    crono.restart();
    crono.resume();
    if(agoles==1){
      clienteMQTT->publish("v1/devices/me/telemetry","{\"modo3\": 1}");
    }
    else{
      clienteMQTT->publish("v1/devices/me/telemetry","{\"modo3\": 2}");
    }
  }

  else if(preparada == true and jugando==true and equiposinvictos>1){
    marcadorpartida->actualizar_display();
    if(finpartido(true, agoles, x_cambio, marcadorpartida)==true){
      Serial.println("Ha ganado el equipo local: ");
      Serial.print(listaequipos[0].danombreequipo());
      Serial.println("");
      listaequipos[0].sumapartida(true,marcadorpartida->get_goles(true),marcadorpartida-
>get_goles(false));
      listaequipos[1].sumapartida(false,marcadorpartida->get_goles(false),marcadorpartida-
>get_goles(true));
      actualizaajugadores(listaequipos[0],true,marcadorpartida-
>get_goles(true),marcadorpartida->get_goles(false),clienteMQTT);
      actualizaajugadores(listaequipos[1],false,marcadorpartida-
>get_goles(false),marcadorpartida->get_goles(true),clienteMQTT);
      mensaje_a_enviar="El ganador es: "+listaequipos[0].danombreequipo()+"
,"+String(listaequipos[1].danombreequipo())+" ha sido eliminado. ";
      torneorotar(true);
      printquienjuega(false);
      marcadorpartida->reset_marcador();
      crono.restart();
      marcadorpartida->actualizar_display();
      if(equiposinvictos>1){
        mensaje_a_enviar=mensaje_a_enviar+"Juegan: "+listaequipos[0].danombreequipo()+" vs
"+listaequipos[1].danombreequipo();
      }
    }

    else if(finpartido(false, agoles, x_cambio, marcadorpartida)==true){
      Serial.println("Ha ganado el equipo vistante: ");
      Serial.print(listaequipos[1].danombreequipo());
      Serial.println("");
      listaequipos[1].sumapartida(true,marcadorpartida->get_goles(false),marcadorpartida-
>get_goles(true));
      listaequipos[0].sumapartida(false,marcadorpartida->get_goles(true),marcadorpartida-
>get_goles(false));
      actualizaajugadores(listaequipos[1],true,marcadorpartida-
>get_goles(false),marcadorpartida->get_goles(true),clienteMQTT);
      actualizaajugadores(listaequipos[0],false,marcadorpartida-
>get_goles(true),marcadorpartida->get_goles(false),clienteMQTT);
      mensaje_a_enviar="El ganador es: "+listaequipos[1].danombreequipo()+"
,"+listaequipos[0].danombreequipo()+" ha sido eliminado. ";
      torneorotar(false);
      printquienjuega(false);
      marcadorpartida->reset_marcador();
      crono.restart();
      marcadorpartida->actualizar_display();
```

```
      if(equiposinvictos>1){
        mensaje_a_enviar=mensaje_a_enviar+"Juegan: "+listaequipos[0].danombreequipo()+" vs
"+listaequipos[1].danombreequipo();
      }
    }

    if(equiposinvictos<=1){
        jugando= false;
        //el equipo X  ha ganado
        crono.stop();
        Serial.print("El equipo ganador del torneo es: ");
        Serial.print(listaequipos[0].danombreequipo());
        Serial.println("");
        mensaje_a_enviar="YA TENEMOS UN GANADOR DEL TORNEO!! el ganador es:
"+listaequipos[0].danombreequipo();
    }
  }

  return mensaje_a_enviar;
}


//modo liga
String Modojuego::partidaliga( unsigned char x_cambio,unsigned char agoles, unsigned char
numerodeequipos, boolean idayvuelta, Marcador* &marcadorpartida, PubSubClient*
&clienteMQTT){
  String mensaje_a_enviar="";
  if(preparada == false and jugando==false){
    numequipos=numerodeequipos;
    if((idayvuelta==true and numerodeequipos<=11)or (idayvuelta==false and
numerodeequipos<=16)){
      numpartidos = numerodeequipos*(numerodeequipos-1);
      if(idayvuelta==false){
        numpartidos=numpartidos/2;
      }
      organizarequipos(numerodeequipos);
      obtenerlistadepartidos(numerodeequipos,idayvuelta);
      printquienjuega(true);
      crono.restart();
      crono.resume();
      contadorpartidosliga = 0;
      marcadorpartida->reset_marcador();
      preparada = true;
      jugando=true;
    }
    else{
      Serial.println("El numero de jugadores debe ser: max 16 (solo ida) o max 11(ida y
vuelta)");
    }
    if(agoles==1){
      clienteMQTT->publish("v1/devices/me/telemetry","{\"modo4\": 1}");
    }
    else{
      clienteMQTT->publish("v1/devices/me/telemetry","{\"modo4\": 2}");
    }
  }
  else if(preparada == true and jugando==true and contadorpartidosliga<numpartidos){
    marcadorpartida->actualizar_display();
    if(finpartido(true, agoles, x_cambio, marcadorpartida)==true){
      Serial.println("Ha ganado el equipo local: ");
      Serial.print(listapartidosliga[0].danombreequipo());
      listaequipos[listapartidosliga[0].getIDequipo()-1].sumapartida(true,marcadorpartida-
>get_goles(true),marcadorpartida->get_goles(false));
      listaequipos[listapartidosliga[1].getIDequipo()-
1].sumapartida(false,marcadorpartida->get_goles(false),marcadorpartida->get_goles(true));
      actualizaajugadores(listapartidosliga[0],true,marcadorpartida-
>get_goles(true),marcadorpartida->get_goles(false),clienteMQTT);
```

```
      actualizaajugadores(listapartidosliga[1],false,marcadorpartida-
>get_goles(false),marcadorpartida->get_goles(true),clienteMQTT);
      mensaje_a_enviar=listaequipos[0].danombreequipo()+" ha ganado el partido
NÂ°"+String(contadorpartidosliga+1)+"de" +String(numpartidos)+", los siguientes en jugar
son: ";
      for (int j=0;j<(numpartidos-contadorpartidosliga);j++){
         listapartidosliga[2*j] =  listapartidosliga[2*j+2];
         listapartidosliga[2*j+1] =  listapartidosliga[2*j+3];
      }
      mensaje_a_enviar=mensaje_a_enviar+listaequipos[0].danombreequipo()+ " vs
"+listaequipos[1].danombreequipo();
      contadorpartidosliga = contadorpartidosliga +1 ;
      printquienjuega(true);
      marcadorpartida->reset_marcador();
      crono.restart();
      marcadorpartida->actualizar_display();
    }
    else if(finpartido(false, agoles, x_cambio, marcadorpartida)==true){
      Serial.println("Ha ganado el equipo vistante: ");
      Serial.print(listapartidosliga[1].danombreequipo());
      listaequipos[listapartidosliga[1].getIDequipo()-1].sumapartida(true,marcadorpartida-
>get_goles(false),marcadorpartida->get_goles(true));
      listaequipos[listapartidosliga[0].getIDequipo()-
1].sumapartida(false,marcadorpartida->get_goles(true),marcadorpartida->get_goles(false));
      actualizaajugadores(listapartidosliga[0],false,marcadorpartida-
>get_goles(true),marcadorpartida->get_goles(false),clienteMQTT);
      actualizaajugadores(listapartidosliga[1],true,marcadorpartida-
>get_goles(false),marcadorpartida->get_goles(true),clienteMQTT);
      mensaje_a_enviar=listaequipos[1].danombreequipo()+" ha ganado el partido
NÂ°"+String(contadorpartidosliga+1)+"de" +String(numpartidos)+", los siguientes en jugar
son: ";
      for (int j=0;j<(numpartidos-contadorpartidosliga);j++){
         listapartidosliga[2*j] =  listapartidosliga[2*j+2];
         listapartidosliga[2*j+1] =  listapartidosliga[2*j+3];
      }
      mensaje_a_enviar=mensaje_a_enviar+listaequipos[0].danombreequipo()+ " vs
"+listaequipos[1].danombreequipo();
      contadorpartidosliga = contadorpartidosliga +1 ;
      printquienjuega(true);
      marcadorpartida->reset_marcador();
      crono.restart();
      marcadorpartida->actualizar_display();
    }
    if(contadorpartidosliga>=numpartidos){
      for(int i=0;i<numequipos;i++){
        listaequipos[i].printestadisticas();
        Serial.println("--------------------------------------------------");
      }
      ordenar_clasificacion();
      mensaje_a_enviar="FIIIIN DE LA LIGA, el ganador es:
"+listaequipos[0].danombreequipo()+"\r\n"+"La clasificacion ha quedado:"+"\r\n";
      mensaje_a_enviar=mensaje_a_enviar+"<br>"+"Equipo        PG     PJ      GF      GC"+"
\r\n";
      for(int i=0;i<numequipos;i++){
        mensaje_a_enviar=mensaje_a_enviar+"<br>"+listaequipos[i].danombreequipo()+"\t"+"\t
"+listaequipos[i].getdato(1)+"\t"+listaequipos[i].getdato(2)+"\t"+listaequipos[i].getdato(
3)+"\t"+listaequipos[i].getdato(4)+"\r\n";
      }
      jugando=false;
      crono.stop();
      }
    }
  return mensaje_a_enviar;
}
```

```
//--------------------------------funciones de verificacion------------------------------
void Modojuego::printquienjuega(boolean deliga){
  Serial.print("Juegan: Local->  ");
  if (deliga==true){
    Serial.print(listapartidosliga[0].danombreequipo());
    Serial.print("  vs  ");
    Serial.print(listapartidosliga[1].danombreequipo());
  }
  else{
    Serial.print(listaequipos[0].danombreequipo());
    Serial.print("  vs  ");
    Serial.print(listaequipos[1].danombreequipo());
  }
  Serial.println(" <-Visitante");
}
```

```
/*
monedero.h - Libreria para monedero
Por Rodrigo Perez Garcia 04/08/2020
Todos los derechos reservados
*/

#ifndef Monedero_h
#define Monedero_h

#include "Arduino.h"
#include <Servo.h>
#include <MFRC522.h>

//--------------------------------definicion de la clase----------------------------------
class Monedero
{
  public:
    Monedero(unsigned char SS_PIN, unsigned char RST_PIN, unsigned char PWM_PIN);
    int ver_saldo();
    int credito_ultima_tarjeta();
    int leer_tarjeta();
    void actualizar_actuador();
    boolean actualizar_saldo(boolean incrementar, int cantidad);
    void restar_credito();
    boolean actualizar_credito();
    void escribir_tarjeta(int elvalor);
    void recargar_tarjeta();
    void verificartarjeta(int selecopcion);

  private:
    MFRC522 mfrc522;
    Servo elservo;
    unsigned int coste;
    unsigned int saldo;
    boolean pwm_on;
    const int SIZE_BUFFER=18;
    const int MAX_SIZE_BLOCK=16;
    int saldo_last_card=0;
};
#endif
```

```
/*
monedero.cpp - Libreria para pagar con targeta socio y habilitar el dispensado de bolas
Por Rodrigo Perez Garcia 04/08/2020
Todos los derechos reservados
*/

#include "monedero.h"
#include "Arduino.h"
#include <Servo.h>
#include <MFRC522.h>

//---------------------------------constructor de la clase-------------------------------
Monedero::Monedero(unsigned char SS_PIN, unsigned char RST_PIN, unsigned char
PWM_PIN):mfrc522(SS_PIN, RST_PIN){
  //MFRC522 mfrc522(SS_PIN, RST_PIN);    // Defined pins to module RC522
  SPI.begin(); // Init SPI bus
  mfrc522.PCD_Init();
  elservo.attach(PWM_PIN);
  coste = 1;
  saldo = 0;
  pwm_on = false;
}



//-------------------------------------funciones get-------------------------------------
//funcion para ver el saldo del futbolin
int Monedero::ver_saldo(){
  return saldo;
}


//funcion para conocer el credito de la ultima tarjeta (se visualizara en servidor web)
int Monedero::credito_ultima_tarjeta(){
  return saldo_last_card;
}


//funcion de lectura de la tarjeta
int Monedero::leer_tarjeta(){
  MFRC522::MIFARE_Key key;             //used in authentication
  MFRC522::StatusCode status;          //authentication return status code
  String stringcredito = "";
  int creditodetarjeta = 0;
  Serial.println("leyendo tarjeta...");
  //mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid)) prints the technical details of the
card/tag

  //prepare the key - all keys are set to FFFFFFFFFFFFh
  for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

  //buffer for read data
  byte buffer[SIZE_BUFFER] = {0};

  //the block to operate
  byte block = 1;
  byte size = SIZE_BUFFER;  //authenticates the block to operate
  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key,
&(mfrc522.uid)); //line 834 of MFRC522.cpp file
  if (status != MFRC522::STATUS_OK) {
    Serial.println("La autentificacion ha fallado: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    return creditodetarjeta;
  }
  status = mfrc522.MIFARE_Read(block, buffer, &size);
  if (status != MFRC522::STATUS_OK) {
    Serial.println("La lectura ha fallado: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
```

```
    return creditodetarjeta;
  }
  else{
    Serial.print("Credito en la tarjeta: ");
    for (uint8_t i = 0; i < MAX_SIZE_BLOCK; i++){
      Serial.write(buffer[i]);
      try{
        stringcredito += (char)buffer[i];
      }
      catch (const std::exception& ex){
        Serial.printf("Error:%s",ex.what());
      }
    }
    Serial.println("");
    creditodetarjeta = stringcredito.toInt();
  }
  return creditodetarjeta;

}



//-------------------------------------funciones set-------------------------------------
//funcion para habilitar y deshabilitar el dispensado de bolas
void Monedero::actualizar_actuador(){
  if(saldo>0){
    if(pwm_on==false){
      elservo.write(100);
      delay(1000);
      pwm_on=true;
    }
  }
  else{
    elservo.write(10);
    delay(1000);
    pwm_on=false;
  }
}


//funcion para incrementar el saldo de monedero
boolean Monedero::actualizar_saldo(boolean incrementar, int cantidad){
  if(incrementar==true){
    saldo= saldo+cantidad;
    return true;
  }
  else{
    if(cantidad>saldo){
      Serial.println("Se dispone de saldo insuficiente");
    }
    saldo = saldo-cantidad;
  }
}



//funcion para decrementar el saldo de monedero
void Monedero::restar_credito(){
  if(saldo>0){
    saldo--;
  }
}


//funcion para leer la tarjeta restarle un credito e incrementar el saldo de monedero
boolean Monedero::actualizar_credito(){
  int saldotarjeta;
  MFRC522::MIFARE_Key key;
```

```
  MFRC522::StatusCode status;
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return false;
  }
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return false;
  }
  saldotarjeta = leer_tarjeta();
  if (saldotarjeta>0){
    Serial.print(" (-) coste de partida: ");
    Serial.println(coste);
    escribir_tarjeta(saldotarjeta-1);
    saldo = saldo +1;
    Serial.println("");
    saldo_last_card=saldotarjeta-1;
  }
  else{
    Serial.println("no tienes creditos suficientes");
    saldo_last_card=saldotarjeta;
  }
  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();
  return true;
}


//funcion para escribir el numero de creditos en la tarjeta
void Monedero::escribir_tarjeta(int elvalor){
  MFRC522::MIFARE_Key key;             //used in authentication
  MFRC522::StatusCode status;          //authentication return status code
  Serial.println("escribiendo tarjeta...");
  for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;
  //buffer para almacenar los datos a escribir
  byte buffer[MAX_SIZE_BLOCK] = "";
  byte block; //the block to operate
  byte dataSize; //size of data (bytes)
  String valordeescritura=String(elvalor);
  dataSize = valordeescritura.length();
  //void positions that are left in the buffer will be filled with whitespace
  for(byte i=0; i < dataSize; i++){
    buffer[i] = valordeescritura[i];
  }
  for(byte i=dataSize; i < MAX_SIZE_BLOCK; i++){
    buffer[i] = ' ';
  }
  block = 1; //the block to operate
  String str = (char*)buffer; //transforms the buffer data in String
  Serial.print("El credito de la tarjeta será: ");
  Serial.println(str);
   //Authenticate is a command to hability a secure communication
  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key,
&(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
    Serial.println("fallo de autentificacion.");
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  status = mfrc522.MIFARE_Write(block, buffer, MAX_SIZE_BLOCK);
  if (status != MFRC522::STATUS_OK) {
    Serial.println("fallo de escritura");
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  else{
    Serial.println("(escritura realizada)");
  }
}
```

```
//funcion para escribir el numero de creditos en la tarjeta mediante la IDE
void Monedero::recargar_tarjeta(){
  MFRC522::MIFARE_Key key;              //used in authentication
  MFRC522::StatusCode status;           //authentication return status code
  Serial.setTimeout(10000L) ;    // esperar 15 segundos para introducir la cantidad
deseada
  Serial.println(F("Introduce el valor a escribir finalizado por '#' [max 16 digitos] \n
valor escrito:"));
  //prepare the key - all keys are set to FFFFFFFFFFFFh
  for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

  //buffer para almacenar los datos a escribir
  byte buffer[MAX_SIZE_BLOCK] = "";
  byte block; //the block to operate
  byte dataSize; //size of data (bytes)

  //recover on buffer the data from Serial
  //all characters before chacactere '#'
  dataSize = Serial.readBytesUntil('#', (char*)buffer, MAX_SIZE_BLOCK);
  //void positions that are left in the buffer will be filled with whitespace
  for(byte i=dataSize; i < MAX_SIZE_BLOCK; i++){
    buffer[i] = ' ';
  }
  block = 1; //the block to operate
  String str = (char*)buffer; //transforms the buffer data in String
  Serial.println(str);
   //Authenticate is a command to hability a secure communication
  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key,
&(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
    Serial.print("Fallo de autentificacion");
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  //Writes in the block
  status = mfrc522.MIFARE_Write(block, buffer, MAX_SIZE_BLOCK);
  if (status != MFRC522::STATUS_OK) {
    Serial.println("escritura fallada:");
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  else{
    Serial.println("Escritura realizada");
  }
}
```

```
//--------------------------------funciones de verificacion-------------------------------
//funcion para utilizar las funciones set anteriores
void Monedero::verificartarjeta(int selecopcion) {
  MFRC522::MIFARE_Key key;              //used in authentication
  MFRC522::StatusCode status;           //authentication return status code
   // Aguarda a aproximacao do cartao
   //waiting the card approach
  if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }
  // Select a card
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
  }
  if(selecopcion == 4) {
    recargar_tarjeta();
  }
  else if(selecopcion == 3){
    escribir_tarjeta(619);
  }
  else if(selecopcion == 2){
    actualizar_credito();
  }
  else{
    int valorleido = -1;
    valorleido=leer_tarjeta();
    Serial.println(valorleido);
  }
  //instructs the PICC when in the ACTIVE state to go to a "STOP" state
  mfrc522.PICC_HaltA();
  // "stop" the encryption of the PCD, it must be called after communication with
authentication, otherwise new communications can not be initiated
  mfrc522.PCD_StopCrypto1();
}
```

```
/*
Porteria.h - Libreria para control de porterias de un futbolin
Por Rodrigo Perez Garcia 04/08/2020
Todos los derechos reservados
*/
#ifndef Porteria_h
#define Porteria_h

#include "Arduino.h"
#include <SPI.h>

//--------------------------------definición de la clase----------------------------------
class Porteria
{
 public:
  Porteria(int haz1, int haz2, int verificador);
  float revbarrera(float dist_mm ,unsigned long time_us1,unsigned long time_us2);
  boolean revgol();
 private:
        int _haz1;
        int _haz2;
        int _verificador;
        int valorverificador;
        unsigned long time_ms1;
        unsigned long time_ms2;
        unsigned long time_us1;
        unsigned long time_us2;
        float vel_bola;

} ;

#endif
```

```
/*
Porteria.h - Libreria para control de porterias de un futbolin
Por Rodrigo Perez Garcia 04/08/2020
Todos los derechos reservados
*/

#include "porteria.h"
#include <SPI.h>

//---------------------------------constructor de la clase--------------------------------
Porteria::Porteria(int haz1, int haz2, int verificador){
  pinMode(haz1,INPUT_PULLUP);
  pinMode(haz2,INPUT_PULLUP);
  pinMode(verificador,INPUT_PULLUP);
  _haz1 = haz1;
  _haz2=haz2;
  _verificador=verificador;
}

//-------------------------------------funciones get-------------------------------------
//funcion para obtener la velocidad de la bola
float Porteria::revbarrera(float dist_mm ,unsigned long time_us1,unsigned long time_us2){
  if(time_us1>time_us2){
    vel_bola = 3600*dist_mm/(time_us2);
  }
  else{
    vel_bola = 3600*dist_mm/(time_us2-time_us1);
  }
  Serial.println("GOOOOOOL, la velocidad de la bola es: ");
  Serial.println(vel_bola);
  return vel_bola;
}


//funcion para verificar si ha habido gol o no
boolean Porteria::revgol(){
  valorverificador=digitalRead(_verificador);
  time_ms1=millis();
  time_ms2=millis();
  while ((time_ms2-time_ms1)<5000 ){
    valorverificador=digitalRead(_verificador);
    if(valorverificador==LOW){
      Serial.println("--------EL VAR: HA SIDO GOL----------");
      return true;
    }
    time_ms2=millis();
  }
  Serial.println("--------EL VAR: GOL ANULADO----------");
  return false;
}
```

```
/*
Porteria.h - Libreria para control de porterias de un futbolin
Por Rodrigo Perez Garcia 04/08/2020
Todos los derechos reservados
*/
#ifndef Porteria_h
#define Porteria_h

#include "Arduino.h"
#include <SPI.h>

//--------------------------------definicion de la clase----------------------------------
class Porteria
{
 public:
  Porteria(int haz1, int haz2, int verificador);
  float revbarrera(float dist_mm ,unsigned long time_us1,unsigned long time_us2);
  boolean revgol();
  void revpared(MPU9250 Acel_1, MPU9250 Acel_2);
  void printpared();
 private:
        int _haz1;
        int _haz2;
        int _verificador;
        int velestimada;
        int valorhaz2;
        int valorverificador;
        unsigned long time_ms1;
        unsigned long time_ms2;
        unsigned long time_us1;
        unsigned long time_us2;
        double vel_bola;

        float X_Acel_1[500];
        float Y_Acel_1[500];
        float Z_Acel_1[500];
        float X_Acel_2[500];
        float Y_Acel_2[500];
        float Z_Acel_2[500];

} ;

#endif
```

```
/*
Porteria.cpp - Libreria para control de porterias de un futbolin
Por Rodrigo Perez Garcia 04/08/2020
Todos los derechos reservados
*/

#include "porteria.h"
#include <SPI.h>

//---------------------------------constructor de la clase--------------------------------
Porteria::Porteria(int haz1, int haz2, int verificador){
  pinMode(haz1,INPUT_PULLUP);
  pinMode(haz2,INPUT_PULLUP);
  pinMode(verificador,INPUT_PULLUP);
  _haz1 = haz1;
  _haz2=haz2;
  _verificador=verificador;
}

//-------------------------------------funciones get-------------------------------------
//funcion para obtener la velocidad de la bola
float Porteria::revbarrera(float dist_mm ,unsigned long time_us1,unsigned long time_us2){
  if(time_us1>time_us2){
    vel_bola = 3600*dist_mm/(time_us2);
  }
  else{
    vel_bola = 3600*dist_mm/(time_us2-time_us1);
  }
  Serial.println("GOOOOOOL, la velocidad de la bola es: ");
  Serial.println(vel_bola);
  return vel_bola;
}


//funcion para verificar si ha habido gol o no
boolean Porteria::revgol(){
  valorverificador=digitalRead(_verificador);
  time_ms1=millis();
  time_ms2=millis();
  while ((time_ms2-time_ms1)<5000 ){
    valorverificador=digitalRead(_verificador);
    if(valorverificador==LOW){
      Serial.println("--------EL VAR: HA SIDO GOL----------");
      return true;
    }
    time_ms2=millis();
  }
  Serial.println("--------EL VAR: GOL ANULADO----------");
  return false;
}
```

```
//--------------------------------------funciones set-----------------------------------
void Porteria::revpared(MPU9250 Acel_1, MPU9250 Acel_2){

  for (int i = 0 ; i<500 ; i++){
    time_us1=micros();
    Acel_1.readSensor();
    Acel_2.readSensor();
    X_Acel_1[i] = Acel_1.getAccelX_mss();
    Y_Acel_1[i] = Acel_1.getAccelY_mss();
    Z_Acel_1[i] = Acel_1.getAccelZ_mss();
    X_Acel_2[i] = Acel_2.getAccelX_mss();
    Y_Acel_2[i] = Acel_2.getAccelY_mss();
    Z_Acel_2[i] = Acel_2.getAccelZ_mss();
    time_us2=micros();
    while (time_us2-time_us1<1000){
      time_us2=micros();
    }
  }
}
void Porteria::printpared(){
  for (int i = 0 ; i<500 ; i++){
    Serial.print(X_Acel_1[i]);
    Serial.print("\t");
    Serial.print(X_Acel_2[i]);
    Serial.print("\t");
    Serial.print(Y_Acel_1[i]);
    Serial.print("\t");
    Serial.print(Y_Acel_2[i]);
    Serial.print("\t");
    Serial.print(Z_Acel_1[i]);
    Serial.print("\t");
    Serial.println(Z_Acel_2[i]);
  }
}
```

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="style.css">
<title>marcador en directo</title>
</head>

<body>
  <h1 style="font-size: 45px;
  font-weight: 800;">Partido en directo</h1>

<script>
  setInterval(function ( ) {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
           document.getElementById("golesA").innerHTML = this.responseText;
        }
  };
  xhttp.open("GET", "/goles_a", true);
  xhttp.send();
  }, 1000 ) ;

  setInterval(function ( ) {
    var xhttp = new XMLHttpRequest();
         xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
           document.getElementById("golesB").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/goles_b", true);
  xhttp.send();
  }, 1000 ) ;

  setInterval(function ( ) {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
           document.getElementById("nombrelocal").innerHTML = this.responseText;
    }
         };
         xhttp.open("GET", "/nombrelocal", true);
         xhttp.send();
  }, 1000 ) ;

  setInterval(function ( ) {
    var xhttp = new XMLHttpRequest();
           xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
             document.getElementById("nombrevisitante").innerHTML = this.responseText;
      }
    };
         xhttp.open("GET", "/nombrevisitante", true);
         xhttp.send();
  }, 1000 ) ;
```

```
setInterval(function ( ) {
var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
        document.getElementById("nombre_modo").innerHTML = this.responseText;
  }
};
xhttp.open("GET", "/nombremodo", true);
xhttp.send();
}, 1000 ) ;

setInterval(function ( ) {
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("el_evento_a").innerHTML = this.responseText;
  }
};
xhttp.open("GET", "/evento_a", true);
xhttp.send();
}, 1000 ) ;

setInterval(function ( ) {
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
        document.getElementById("el_evento_b").innerHTML = this.responseText;
  }
};
xhttp.open("GET", "/evento_b", true);
xhttp.send();
}, 1000 ) ;

setInterval(function ( ) {
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
        document.getElementById("el_evento_c").innerHTML = this.responseText;
  }
};
xhttp.open("GET", "/evento_c", true);
xhttp.send();
}, 1000 ) ;

setInterval(function ( ) {
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
if (this.readyState == 4 && this.status == 200) {
      document.getElementById("los_creditos").innerHTML = this.responseText;
}
};
xhttp.open("GET", "/creditos", true);
xhttp.send();
}, 1000 ) ;
```

```
  setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
          document.getElementById("el_crono").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/crono", true);
  xhttp.send();
  }, 1000 ) ;

</script>

<table style="margin: auto">
  <tr>
    <td style="  padding: 0px 60px 0px 60px">Local</td>
    <td class="marcadormitad">.........</td>
    <td style="  padding: 0px 50px 0px 50px;">Visitante</td>
  </tr>
  <tr>
        <td id="nombrelocal">%NOMBRELOCAL%</td>
        <td class="marcadormitad"></td>
        <td id="nombrevisitante">%NOMBREVISITANTE%</td>
  </tr>
  <tr style="background-color: #FFFFFF">
        <td><button class="boton botonrename" id="botnombreA">cambiar
nombre</button></td>
        <td class="marcadormitad"></td>
        <td><button class="boton botonrename" id="botnombreB">cambiar
nombre</button></td>
  </tr>
  <tr>
        <td class="celdamarcador" id="golesA">%GOLES_A%</td>
        <td class="marcadormitad"></td>
        <td class="celdamarcador" id="golesB">%GOLES_B%</td>
  </tr>
  <tr >
        <td><a href="/masA"><button class="botondegol">+</button></a></td>
        <td class="marcadormitad" id="el_crono" style="color: white;font-size:
20px">%CRONO%</td>
        <td><a href="/masB"><button class="botondegol">+</button></a></td>
  </tr>
  <tr >
    <td><a href="/menosA"><button class="botondegol">-</button></a></td>
        <td class="marcadormitad"><a href="/playpause"><button style="font-size:
18px">⏯</button></a></td>
        <td><a href="/menosB"><button class="botondegol">-</button></a></td>
  </tr>
</table>
```

```
<div class="popup" id="popup1">
  <div class="popup-contenido">
        <form  method="post" target="_self" type="hidden" id="formularioseleccion">
      <a>Modo de juego:</a><br>
      <select name="mod">
                <option value="1" selected="selected">Partida Rapida</option>
                <option value="2">Partida personalizada</option>
                <option value="3">Torneo</option>
                <option value="4">Ligilla</option>

      </select><br><br>
      <a>Num de equipos:<br> <input type="number" name="equ" value="2"  min="2" max="256"
step="1"></a><br><br>
      <a>Goles/minutos cambio:<br> <input type="number" name="gol" value="1" min="1"
max="100" step="1"></a><br><br>
      Rotar <select name="rot">
        <option value="1" selected="selected">por goles</option>
        <option value="2">por tiempo</option>
      </select><br><br>
      <a><input type="submit" value="¡A jugar!" formmethod="get" id="botonaceptar"></a>
      <a href="/"><button class="boton botonpopup" id="botoncanelar"
>Cancelar</button></a><br><br>
      <small>NOTA: Partida rápida no usa<br>los parametros seleccionados</small>
    </form>
  </div>
</div>

<div class="popup" id="popup2">
  <div class="popup-contenido">
        <form  method="post" target="_self" type="hidden" id="formularionombreA">
      <a>Nombre de equipo local:</a><br>
          <a><i>%NOMBRELOCAL%</i></a><br><br>
      <a>Nuevo nombre:<br><input type="text" autocomplete="off" name="loc" minlength="2"
maxlength="16" required="true" style="WIDTH: 250px" ></a><br><br>
      <a><input type="submit" value="¡cambiar!" formmethod="get" id="botonaceptar2"></a>
    </form>
        <a href="/"><button class="boton botonpopup" id="botoncanelar2"
>Cancelar</button></a>
  </div>
</div>

<div class="popup" id="popup3">
  <div class="popup-contenido">
        <form  method="post" target="_self" type="hidden" id="formularionombreb">
          <a>Nombre de equipo visitante:</a><br>
      <a><i>%NOMBREVISITANTE%</i></a><br><br>
      <a>Nuevo nombre:<br><input type="text" autocomplete="off" name="vis" minlength="2"
maxlength="16" required="true" style="WIDTH: 250px"></a><br><br>
      <a><input type="submit" value="¡cambiar!" formmethod="get" id="botonaceptar3"></a>
    </form>
        <a href="/"><button class="boton botonpopup" id="botoncanelar3"
>Cancelar</button></a>
  </div>
</div>
```

```
<div class="popup" id="popup4">
  <div class="popup-contenido">
        <form  method="post" target="_self" type="hidden" id="formulariojugador">
          <a>Id de jugador:<br><input type="text" autocomplete="off" name="jug"
minlength="2" maxlength="16" required="true" style="WIDTH: 250px"></a><br><br>
      <a>Equipo:</a>
      <select name="jug_eq">
        <option value="0" selected="selected">Local</option>
                <option value="1">Visitante</option>

      </select><br><br>
      <a>Posicion:</a>
      <select name="jug_pos">
                <option value="1" selected="selected">Defensa</option>
                <option value="2">Ataque</option>
      </select><br><br>
      <a><input type="submit" value="aceptar" formmethod="get" id="botonaceptar4"></a>
     </form>
         <a href="/"><button class="boton botonpopup" id="botoncanelar4"
>Cancelar</button></a>
  </div>
</div>

<br><br><p style="display: inline"><a><button class="boton botonseleccion"
id="bot1">Seleccionar modo de juego</button></a></p>
     <a id="los_creditos" style="font-size: 20 ;display:
inline">%CREDITOS% </a> CREDITOS

<p>Modo seleccionado: <a id="nombre_modo">%NOMBREMODO%</a></p>

<p style="font-size: 25px; font-weight: 800; color: white ;margin-bottom: 0px"> Ultimos
eventos del futbolin</p>

<table style="width:600px">
  <tr>
    <td style="font-size: 25px" id="el_evento_a"> %EVENTO_A% </td>
  </tr>
  <tr>
        <td style="font-size: 18px" id="el_evento_b"> %EVENTO_B% </td>
  </tr>
  <tr>
        <td style="font-size: 15px" id="el_evento_c"> %EVENTO_C% </td>
  </tr>
</table>

<p><a><button class="boton botonseleccion"  id="bot2" >Registrar posicion de
jugador</button></a></p>

<script>
  document.getElementById("bot1").addEventListener("click",function(){
    document.getElementById("popup1").style.display="flex";
  })

  document.getElementById("botoncanelar").addEventListener("click",function(){
        document.getElementById(".popup1").style.display="none";
        document.getElementById(".popup2").style.display="none";
```

```
        document.getElementById(".popup3").style.display="none";
        document.getElementById(".popup4").style.display="none";
    })

    document.getElementById("botnombreA").addEventListener("click",function(){
        document.getElementById("popup2").style.display="flex";
    })

    document.getElementById("botoncanelar2").addEventListener("click",function(){
        document.getElementById(".popup2").style.display="none";
        document.getElementById(".popup1").style.display="none";
        document.getElementById(".popup3").style.display="none";
        document.getElementById(".popup4").style.display="none";
    })

    document.getElementById("botnombreB").addEventListener("click",function(){
        document.getElementById("popup3").style.display="flex";
    })

    document.getElementById("botoncanelar3").addEventListener("click",function(){
        document.getElementById(".popup3").style.display="none";
        document.getElementById(".popup1").style.display="none";
        document.getElementById(".popup2").style.display="none";
        document.getElementById(".popup4").style.display="none";
    })

    document.getElementById("bot2").addEventListener("click",function(){
        document.getElementById("popup4").style.display="flex";
    })

    document.getElementById("botoncanelar4").addEventListener("click",function(){
        document.getElementById(".popup4").style.display="none";
        document.getElementById(".popup1").style.display="none";
        document.getElementById(".popup2").style.display="none";
        document.getElementById(".popup3").style.display="none";
    })


</script>
</body>
</html>
```

```css
html {
  font-family: Helvetica;
  display: inline-block;
  margin: 0px auto;
  text-align: center;
  background-color: #369AF3;
}


h1{
  color: #000000;
  padding:  0px 0px 20px;
  margin: 0px    ;
}

h2{
  color: #000000;
  padding: 1px;
  margin: 0px;
}

p{
  font-size: 30px;
  font-weight: 600;
}

input, select{
         margin-top:4px;
         font-size: 20px;
}


table, th, td{
         width:20%;
         border: 3px solid white;
         border-collapse: collapse;
         padding: 2%;
         font-size: 20px;
         font-weight: 500;
         margin: auto;
}

.celdamarcador{
         padding: 10% 0% 10% 0%;
         font-size: 40px;
         background-color: white;
         border-bottom: 3px solid #369AF3;
         border-top: 5px solid #369AF3;
}
```

```css
.boton{
    display: inline-block;
        background-color: #f44336;
        border: 3px solid #FFFFFF;
        border-radius: 25px;
        color: white;
        padding: 75px 50px;
        text-decoration: none;
        font-size: 50px;
}


.botondegol{
        background-color: #8D8D8D;
        font-size: 25px;
        margin: 0%;
        padding: 3% 50% 3% 50% ;
}

.botonseleccion {
        background-color: #000CFF;
        border-radius: 15px;
        padding: 20px 20px;
        font-size: 25px;
        text-align: center;
}

.botonpopup{
        background-color: #FB0000;
        padding: 6px 8px;
        border-radius: 0px;
        font-size: 20px;
}


.botonsalir{
        background-color: #FF000;
        border-radius: 15px;
        padding: 20px 100px;
        font-size: 25px;
        text-align: center;
}

.botonrename{
        background-color: #FFFFFF;
        color:black;
        font-size: 18px;
        border-radius: 0px;
        margin: 0%;
        padding: 0% 0% 0% 0%;
        font-weight: 500;
}
```

```
.marcadormitad{
        background-color: #369AF3;
        border-bottom: 3px solid #369AF3;
        border-top: 3px solid #369AF3;
        color:#369AF3;
}

.popup{
        display: none;
        position: absolute;
        align-items: center;
        text-align: center;
        justify-content: center;
        margin-left: 55%;
        margin-top: -20%;
        border: 3px solid #000CFF;
}

.popup-contenido{
        height: auto;
        width: auto;
        background-color: white;
        padding: 20px;
        border-radius: 5px;
        position: relative;
        display: block;
        margin: auto;
        font-size: 20px;
}


#bot2{
        background-color: #C70000;
}

#popup2{
        margin-top: -20%;
        margin-left: 27.8%
}

#popup3{
        margin-top: -20%;
        margin-left: 51%
}
```

```
import sys
import io
import picamera
import RPi.GPIO as GPIO
import time
import os
import webbrowser
url="http://192.168.4.1/";
#Set up confirguration
isoVal = 800
expmode = 'auto'

totalseconds = 3

thisframerate = 60
playbackframerate = 15
theoplaytime = totalseconds * thisframerate / playbackframerate

#set up GPIO using BCM Numbering
GPIO.setmode(GPIO.BCM)
GPIO.setup(4, GPIO.IN, GPIO.PUD_UP)

camera = picamera.PiCamera()
camera.awb_mode = 'flash'
camera.vflip = True
camera.hflip = True
camera.resolution = (1280, 720)
camera.framerate = thisframerate
camera.exposure_mode = 'auto'
camera.iso = isoVal
stream = picamera.PiCameraCircularIO(camera, seconds=10)
camera.start_recording(stream, format='h264' ,bitrate=25000000,quality=30)
videocount = 1
#set up directory for files
webbrowser.open(url,new=2);
directory = "/home/pi/Desktop/TFG repeticiones/";

while True:
    try:
        camera.wait_recording(0.2)

        GPIO.wait_for_edge(4, GPIO.FALLING)

        filemp4 = directory + "gol" + str(videocount) + ".mp4"
        filename = directory + "gol" + str(videocount) + ".h264"
        stream.copy_to(filename, seconds=totalseconds+5)
        #convertstring = "MP4Box -fps " + str(playbackframerate) + " -add " + filename + "
" + filemp4
        #playerstring = "omxplayer " + filemp4
        playerstring = "omxplayer " + filename
        #os.system(convertstring)
        os.system(playerstring)
        t_ini=time.time()
```

```
         while (time.time()-t_ini<4):
            if GPIO.input(4)==1:
                omxc = Popen(['omxplayer', filename])
                print("17 pulsado")
                break
        #omxc = Popen(['omxplayer',
filename],stdin=PIPE,stdout=PIPE,stderr=PIPE,close_fds=True)
        #omxc = Popen(['omxplayer', filemp4])

    finally:
        print("gol")
        GPIO.cleanup()
        camera.stop_recording()
        camera.close()
```

```
import serial
import xlwt
from datetime import datetime

class SerialToExcel:

    def __init__(self,port,speed):

        self.port = port
        self.speed = speed

        self.wb = xlwt.Workbook()
        self.ws = self.wb.add_sheet("Data from Serial",cell_overwrite_ok=True)
        self.ws.write(0, 0, "Data from Serial")
        self.columns = ["Date Time"]
        self.number = 100


    def setColumns(self,col):
        self.columns.extend(col)

    def setRecordsNumber(self,number):
        self.number = number

    def readPort(self):
        ser = serial.Serial(self.port, self.speed, timeout=1)
        c = 0
        for col in self.columns:
            self.ws.write(1, c, col)
            c = c + 1
        self.fila = 2

        i = 0
        while(i<self.number):
            line = str(ser.readline())
            if(len(line) > 0):
                now = datetime.now()
                date_time = now.strftime("%m/%d/%Y, %H:%M:%S")
                print(date_time,line)
                if(line.find(",")):
                    c = 1
                    self.ws.write(self.fila, 0, date_time)
                    columnas = line.split(",")
                    for col in columnas:
                        self.ws.write(self.fila, c, col)
                        c = c + 1

                i = i + 1
                self.fila = self.fila + 1

    def writeFile(self,archivo):
         self.wb.save(archivo)
```

```
from serialToExcel import SerialToExcel

serialToExcel = SerialToExcel("COM8",1000000)

columnas = ["num","acel1_x","acel2_x","acel1_y","acel2_y","acel1_z","acel2_z"]

serialToExcel.setColumns(["Nro Lectura","Valor"])
serialToExcel.setRecordsNumber(1300)
serialToExcel.readPort()

serialToExcel.writeFile("150cm_B.xls")
```