UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

DISEÑO E IMPLEMENTACIÓN DE UNA HERRAMIENTA ROBÓTICA GAMIFICADA MULTIUSUARIO PARA LA EDUCACIÓN DIABETOLÓGICA INFANTIL

AUTOR: CARLOS VISERAS RUIZ

TUTOR: JOSÉ LUIS DÍEZ RUANO

Seleccion JORGE BONDIA COMPANY

Curso Académico: 2019-20

Resumen

La diabetes mellitus tipo 1 es una enfermedad que necesita un amplio conocimiento y dedicación para su adecuada gestión. Este problema se acrecienta en edades tempranas, donde la asimilación de conceptos y rutinas requiere del uso de métodos más didácticos y enfocados al rango de edad.

En el presente proyecto se desarrolla una herramienta multiplataforma, enfocada en la educación diabetológica, con la que poder transmitir de una forma más empática los conceptos básicos que un niño que se inicia en la enfermedad debe conocer.

La herramienta implementada constará de una aplicación desarrollada en Unity donde se integrarán, en un concepto multiusuario, juegos de competición y aprendizaje mutuo. Todo esto será complementado mediante la adición de dispositivos robóticos, basados en Arduino, que dotarán a la herramienta de una interfaz mucho más amigable y cercana hacia los usuarios finales.

Palabras clave: diabetes mellitus, educación diabetológica, Unity, multiusuario, dispositivos robóticos, Arduino.

Resum

La diabetis mellitus tipus 1 és una malaltia que necessita d'un ampli coneixement i dedicació per una adequada gestió. Aquest problema s'incrementa a tempranes edats, on l'assimilació de conceptes i rutines requereix de l'ús de mètodes més didàctics i enfocats a la franja d'edat.

En el present projecte es desenvolupa una eina multiplataforma, enfocada en l'educació diabetològica, amb la qual poder transmetre d'una forma més empàtica els conceptes bàsics que un xiquet que s'inicia en la malaltia ha de conèixer.

L'eina implementada constarà d'una aplicación desenvolupada en Unity on s'integren, en un concepte multiusuari, jocs de competició i aprenentatge mutu. Tot això serà complementada mitjançant l'addició de dispositius robòtics, basats en Arduino, que dota l'eina d'una interfície molt més amigable i propera cap als usuaris finals.

Paraules clau: Diabetis mellitus, educació diabetològica, Unity, multiusuari, dispositius robòtics, Arduino.

Abstract

Type 1 diabetes mellitus is a disease that requires extensive knowledge and dedication for its proper management. This problem increases in early ages, where the assimilation of concepts and routines requires the use of more didactic methods focused on the age range.

In the present project, a multiplatform tool is developed, focused on diabetes education, with which to transmit in a more empathetic way the basic concepts that a child who is beginning to suffer from the disease must know.

The tool implemented will consist of an application developed at Unity where competitive games and mutual learning will be integrated into a multi-user concept. All this will be complemented by the addition of robotic devices, based on Arduino, which will provide the tool with a much more friendly interface and closer to the end users.

Keywords: diabetes mellitus, diabetes education, Unity, multi-user, robotic devices, Arduino.

Índice general

Índice de figuras

Índice de tablas

Parte I

Memoria

Capítulo 1

Introducción

«Todo gran proyecto, debe tener un principio, pero es en la continuidad, hasta el final, en dónde se obtiene la verdadera ganancia» — Linares

1.1 Objetivo del proyecto

El objetivo de este proyecto es la creación de una herramienta educativa que sirva de apoyo a los niños entre 6 y 12 años en el aprendizaje acerca de cómo vivir con una enfermedad como la diabetes tipo 1. Concretamente, se ha enfocado el diseño y la carga educativa a un uso de la herramienta en talleres de educación diabetológica y en la preparación previa a estos.

Como complemento de este objetivo principal se anticipan una serie de subobjetivos necesarios para el correcto cumplimiento del principal:

- Crear una herramienta capaz de transmitir los conocimientos básicos que un niño debutante en diabetes debe conocer.
- Transmitir la información de una forma atractiva y visual acorde al público objetivo al que está destinada.
- Conocimiento de la diabetes tipo 1 y los mecanismos utilizados por los profesionales para la transmisión de los conocimientos asociados a ella.
- Integración y entendimiento de un simulador de diabetes capaz de modelar la curva de glucosa en sangre de una persona tipo tras un escenario de ingestas, ejercicio y terapias a lo largo de un día.
- Estudio del efecto de los juegos de competición en el mantenimiento de la motivación y como aplicarlo a una herramienta con componente físico y virtual.
- Integración de un dispositivo robótico móvil al ecosistema generado para la simulación virtual.
- Incorporación de carreras entre vehículos alrededor de un circuito como elementos para generar competitividad y motivación.

- Virtualización y modelado del funcionamiento del dispositivo robótico para su uso como interfaz sin necesidad de poseer el dispositivo físico.
- Integración en la herramienta de una base de datos en la nube capaz de gestionar las comunicaciones entre usuarios de forma síncrona y asíncrona.
- Generación de distintos perfiles de usuario y privilegios de uso en función del rol que desempeñen en la utilización de la herramienta.
- Creación mediante visión artificial de circuitos tanto físicos como virtuales sobre el que el dispositivo robótico pueda actuar como interfaz en función de la simulación realizada.
- Programación y diseño de la herramienta para ser escalable tanto en número de usuarios simultáneos como en juegos o modos adicionales.

1.2 Motivación

La diabetes tipo 1 es una enfermedad crónica, la cual requiere una implicación y conocimiento muy alto por parte del paciente para su tratamiento. Esta ingente cantidad de información y metodologías pueden llegar a abrumar a adultos debutantes. Es por ello la necesidad del uso de los talleres de formación y los educadores en diabetes para la correcta asimilación de los conceptos asociados a esta enfermedad.

Todo esto se agrava en el momento en el que el paciente debutante es un niño, el cual desde el momento de su debut debe realizar las mismas acciones que realizaría un adulto en su misma situación. Este hecho implica la importancia de la existencia de herramientas que ayuden a los educadores a transmitir los conocimientos de una forma atractiva y eficaz a los niños que se inicien en la diabetes tipo 1.

En este contexto se engloba el proyecto detallado a lo largo del presente documento, la creación de una herramienta útil para la sociedad. La cual, permita hacer en la medida de lo posible un poco más amable a los niños los inicios en una enfermedad que les exige tanto esfuerzo y dedicación.

Este proyecto se encuentra enmarcado en una iniciativa en educación diabetológica puesta en marcha en el Instituto de Automática e informática Industrial (Ai2) de la UPV por el grupo de investigación Tecnodiabetes, sirviendo este de soporte para la adquisición de recursos y el establecimiento de contactos con profesionales de la educación diabetológica y pacientes de los que recibir realimentación.

1.3 Interés Académico

El presente proyecto se enmarca en la realización del Trabajo Fin de Máster que culmina los estudios del Máster en Ingeniería Industrial impartido por la Universidad Politécnica de Valencia. Debido a este hecho, los contenidos del proyecto deben tener la suficiente entidad y demostrar la adquisición de los conocimientos adquiridos durante la maestría.

Al ser la herramienta creada un sistema que integra la implementación de un software educativo y el uso de un dispositivo robótico tanto en su versión física como su modelado virtual, implica

el uso de conocimientos y conceptos muy variados adquiridos durante el Máster y que justifican su interés académico:

- Aprendizaje de nuevos lenguajes de programación y entornos de desarrollo software.
- Aplicación de conocimientos de robótica y modelado adquiridos durante la maestría.
- Uso de conexiones inalámbricas para la comunicación entre dispositivos y sincronización de la información.
- Aplicación de conceptos básicos matemáticos y de la ingeniería para la resolución de problemas complejos.
- Creación de protocolos de comunicación que doten de robustez a la gestión multiusuario a tiempo real.
- Uso de software de diversas ramas de la ingeniería para la construcción de una herramienta útil y completa.
- Aplicación de conceptos de visión artificial y reconocimiento de formas estudiados en el Máster.

1.4 Estructura del documento y del proyecto

El presente documento se estructura siguiendo el orden cronológico seguido para el desarrollo del proyecto.

En primer lugar, se inició una fase de documentación tanto de la diabetes y los mecanismos usados para su aprendizaje como de los factores psicológicos implicados en el mantenimiento de la motivación de los alumnos.

En segundo término, debido a que este proyecto parte de un desarrollo previo realizado en el instituto de investigación Ai2 de la UPV, se ha debido entender en detalle su funcionamiento para abordar con éxito la profunda modificación y ampliación realizada.

El tercer paso ha sido el de conocer las posibilidades y limitaciones que ofrece el dispositivo robótico utilizado al ser este un recurso ya disponible en el grupo de investigación. Estas características y limitaciones justificarán muchas de las decisiones tomadas en el desarrollo de la herramienta educativa que lo gobierna.

En cuarto lugar, ha sido necesario el aprendizaje de un nuevo lenguaje de programación y de un entorno de desarrollo de software. Con ellos es con los que se ha realizado la profunda modificación y ampliación del desarrollo previo.

En el quinto paso se ha abordado la modelización y virtualización del dispositivo robótico para poder ser usada la herramienta con todas sus funcionalidades principales sin necesidad de disponer del robot ni de asistir a un taller de educación diabetológica.

Para finalizar se ha diseñado una herramienta externa basada en visión artificial con la que crear circuitos y trayectorias a seguir por el robot que permitan la personalización de los escenarios de una manera muy rápida y sencilla.

Siguiendo esta estructura cronológica en el desarrollo del proyecto se presentarán los distintos epígrafes del documento donde se desarrollarán con profundidad los conceptos mencionados en este apartado introductorio.

Capítulo 2

La diabetes

«La diabetes es un ejemplo de enfermedad con la cual, al darle al paciente las herramientas, puede manejarse muy bien» — Clayton Christensen

2.1 Definición

La diabetes es una enfermedad crónica que se produce porque el páncreas no regula adecuadamente los niveles de glucosa en sangre, esto puede ser causado por tres motivos, el primero es porque el páncreas no produce insulina de forma adecuada, el segundo porque la insulina se produce correctamente, pero el organismo no la puede utilizar de forma eficaz haciéndose resistente a la misma, el tercero y último se trata de una mezcla de los dos anteriores. Como resultado de lo anterior se produce hiperglucemia, es decir, un aumento de los niveles de glucosa en sangre. En la actualidad esta enfermedad no tiene cura alguna, únicamente se pueden prevenir las posibles complicaciones.[1]

2.2 Epidemiología

En el primer Informe mundial publicado por la OMS sobre diabetes ya se estimaba que en el año 2014, 422 millones de adultos en todo el mundo sufrirían esta enfermedad, frente a los 108 millones de 1980.[2]

En el año 2014 más de 5 millones de personas en España sufrían esta enfermedad crónica, formando el colectivo más grande de pacientes crónicos del país. En ese momento más de 5.301.314 personas sufrían diabetes tipo 2, y la Federación Española de Diabetes estimaba que más de 2 millones en ese momento estaban aún sin diagnosticar. Por otro lado, la diabetes tipo 1 representaba un 13 % del total de los diabéticos, de los cuales 29.000 eran niños menores de 15 años, con una incidencia anual de 1.100 casos.[3]

Por otro lado, el coste anual de este problema en nuestro país es de 23.077 millones de euros, de los cuales 17.630 son ocasionados por costes indirectos como el absentismo laboral, las jubilaciones anticipadas y los gastos sociales, y 5.447 son costes directos ocasionados por tratamientos y hospitalizaciones.[3]

En un ranking de 30 países europeos España se sitúa en el puesto número 18. En el mundo 382 millones de personas tienen esta enfermedad, lo que supone un 8,3 % de la población adulta, y se prevé que esta cifra aumente a 592 millones en el año 2035, lo cual supone un aumento del 55 % en dos décadas.[3]

2.3 Mecanismo de producción

En el momento en el que ingerimos un alimento este se descompone en los elementos que lo forman, muchos de los cuales son moléculas de glucosa, comúnmente conocidos como hidratos de carbono, estas moléculas son absorbidas en el intestino delgado, a través del cual entran al torrente sanguíneo, después de lo cual el páncreas, un órgano tanto endocrino como exocrino, produce insulina, que se encargará de que esas moléculas de glucosa anteriormente nombradas sean utilizadas por el organismo para la producción de energía.[1]

2.4 Tipos

Existen principalmente dos tipos de diabetes, detallados en la siguiente tabla (ver Tabla 2.1).[1]

Tabla 2.1: Tipos de diabetes

Tipo	Descripción
Diabetes tipo 1	También conocida como diabetes insulino dependiente, este tipo de diabetes se debe a que el páncreas no produce insulina de forma total o parcial, debido a lo cual las personas que la sufren han de inyectarse insulina de forma diaria. Este tipo de diabetes se da con mayor frecuencia en niños. La causa que produce esta enfermedad es todavía desconocida.
Diabetes tipo 2	Este tipo de diabetes se debe a que, aunque el páncreas produce de forma eficaz la insulina, esta no es utilizada de forma adecuada por el organismo, este tipo de diabetes es más frecuente en adultos, la mayoría de los cuales desconocen que la padecen.

2.5 Síntomas

Los síntomas más frecuentes causados por la diabetes descompensada son los siguientes (ver Figura 2.1).[1]

Mientras que la diabetes tipo 1 debuta de forma temprana, la diabetes tipo 2 lo hace de forma insidiosa, por lo que muchas de las personas que la sufren pueden no tener síntomas en un principio.[1]

Figura 2.1: Síntomas de la diabetes

2.6 Consecuencias

Si esta enfermedad no se detecta y se trata de forma precoz puede acarrear otros problemas más graves a largo plazo, como, por ejemplo: retinopatía diabética, úlceras e infecciones, neuropatía, insuficiencia renal, arteriopatías, etc.[1]

2.7 Prevención

Actualmente la diabetes tipo 1 no tiene ningún tipo de prevención, pero si existen investigaciones que están tratando de averiguar cómo retrasar la aparición de esta enfermedad en personas de alto riesgo.[1]

Por otra parte, la prevención de la diabetes tipo 2 consiste en mantener un estilo de vida saludable, manteniendo un peso corporal adecuado junto a actividad física regular.[1]

2.8 Tratamiento

El tratamiento de la diabetes tipo 1 consiste fundamentalmente en seguir una dieta equilibrada, ejercicio regular y el tratamiento con insulina. Mientras que el tratamiento de la diabetes tipo 2 consiste en dieta y ejercicio físico, dependiendo de la gravedad de la enfermedad se puede necesitar o no tratamiento farmacológico y/o insulina. Todo ello junto a una adecuada educación en salud y seguimiento profesional en cualquiera de ambos casos.[1]

Hay que tener en cuenta que poco después de haber hecho el diagnóstico de diabetes tipo 1 en niños se produce un periodo conocido como 'luna de miel', el cual consiste en una fase en la que el páncreas vuelve a secretar insulina, de forma que en muchos casos hay que disminuir la dosis de insulina o incluso anularla, sin embargo, hay que tener en cuenta que esta fase dura poco tiempo y es necesario prevenir a los niños y a la familia de que esta fase de remisión es transitoria y por tanto una vez haya pasado se volverán a requerir mayores cantidades de insulina.[4]

Existen varios tipos de insulina que se pueden combinar para el tratamiento de la diabetes, estos tipos se diferencian en el momento en el que empieza el efecto, el momento en el que se produce el pico de máximo efecto y la duración de la insulina en el organismo. Actualmente en el mercado existen insulinas ya mezcladas, sin embargo, estas no son adecuadas para la edad pediátrica puesto que las necesidades de los niños son muy variables. Además, hay que tener en cuenta que existen factores que aceleran o disminuyen la velocidad de la insulina una vez administrada (ver Tabla 2.2).[5]

Tabla 2.2: Factores que modifican la velocidad del efecto de la insulina.

Tipos	**Ejemplos**
Factores que aceleran la acción	▪ Actividad física en la zona de administración. ▪ Calor en la zona de punción. ▪ Masaje sobre la zona de administración. ▪ Administrar la inyección a mayor profundidad de la adecuada.
Factores que retrasan la acción	▪ El humo del tabaco. ▪ Frío sobre la zona de punción. ▪ Administrar la inyección a menor profundidad de la adecuada.

Para el correcto uso de la insulina hay que tener en cuenta varios factores: mirar la fecha de caducidad antes de su administración, si la presentación normal se vuelve turbia no debe inyectarse, es importante tener insulina de reserva y conservarla en la nevera, el vial que este en uso en ese momento puede conservarse a temperatura ambiente, pero sin superar los 24ºC, además una vez abierto es importante anotar la fecha de la apertura puesto que este no debe usarse pasadas tres semanas. Si por el contrario se guarda el frasco en uso en la nevera hay que calentarlo con las manos antes de administrar la insulina, ya que el niño sentirá más dolor en su administración y se absorberá peor una vez puesta. Por otra parte, si se van a hacer viajes largos es importante

que el fármaco se mantenga a una temperatura más o menos constante, evitando exponerla a temperaturas muy frías o muy cálidas.[5]

A partir de los 8 años es importante que los niños con diabetes comiencen a aprender a administrarse la insulina de forma autónoma. Para la correcta administración los niños deben de seguir una serie de pasos (ver Figura 2.2).[5]

Figura 2.2: Pasos a seguir para la administración de la insulina

Es importante que si se observan anomalías en la zona de punción se consulte con el equipo sanitario, y se vayan alternando los puntos de inyección para que estas no lleguen a producirse. Existen varios lugares en los que se puede puncionar: en el muslo tanto en la parte anterior como lateral externa, en el cuadrante externo superior de las nalgas, en el abdomen dejando una zona libre de aproximadamente dos dedos alrededor del ombligo, y en la parte superior externa de ambos brazos en caso de que los niños ya no sean muy pequeños.

Es importante seleccionar una de las zonas y puncionar siempre en la misma, puesto que la velocidad de absorción varía entre ellas.[5]

En la realización de los controles de glucemia se debe tener en cuenta dos fases (ver Figura 2.3).[5]

Todo ello con el fin de llegar a unos niveles ideales de glucemia en distintos momentos (ver Figura 2.4).[5]

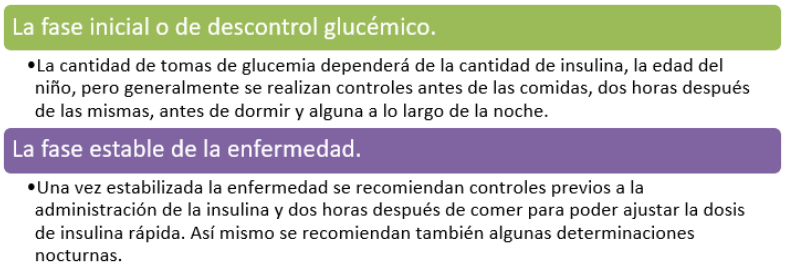


Figura 2.3: Fases de la diabetes

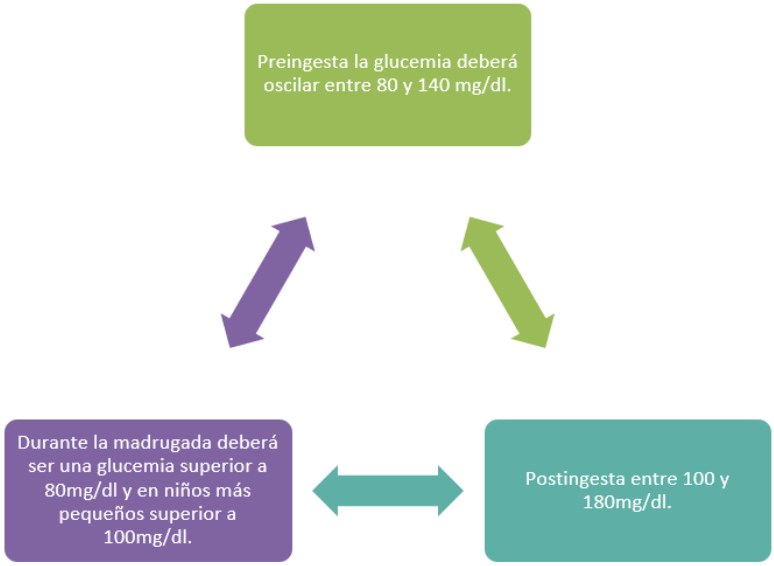


Figura 2.4: Niveles de glucemia en función del momento del día

Actualmente existen unos dispositivos que se colocan a nivel subcutáneo de forma invasiva para lograr una monitorización continua de la glucosa con el fin de conocer como fluctúan los niveles de esta a lo largo del día dependiendo de la ingesta y el ejercicio.[5]

2.9 La alimentación en la diabetes

La finalidad de la alimentación en el niño es cubrir todas sus necesidades energéticas para conseguir un correcto desarrollo tanto psicológico como físico. Para tal fin hay que diferenciar entre los diferentes grupos de alimentos.[5]

Por un lado, los hidratos de carbono o glúcidos popularmente conocidos como azúcares podemos separarlos en dos grupos: simples y complejos. Los hidratos de carbono simples están compuestos por una o dos moléculas de glucosa, es por ello por lo que su absorción es muy rápida y aumentan muy rápido los niveles de glucemia; como ejemplo encontramos el azúcar, la miel, las frutas y la leche. Los hidratos de carbono complejos están formados por muchas moléculas de glucosa por lo

que pasarán más lentamente a sangre, ya que previamente deberán descomponerse en sus formas más simples y por tanto subirán la glucemia más lentamente; como ejemplo encontramos el arroz, las pastas, el pan, las legumbres, las verduras, las patatas, etc. Por todo ello se recomienda que los niños tomen preferentemente hidratos de carbono complejos frente a los simples.[5]

Los lípidos, conocidos como grasas son los alimentos que mayor aporte energético aportan al organismo. Dado que las personas diabéticas tienen un mayor riesgo de sufrir arterioesclerosis es importante prevenir esta mediante una alimentación saludable, por lo que las grasas saturadas presentes en embutidos, carnes grasas, mantequilla, nata, etc. están desaconsejadas de la dieta. Sin embargo, sí que se recomiendan grasas insaturadas como el aceite de oliva, las presentes en los frutos secos, en los pescados azules, etc.[5]

Por último, las proteínas son las encargadas de formar estructuralmente el organismo, las encontramos en alimentos como la carne, el pescado, los huevos, la leche, etc.[5]

2.10 La actividad física

Junto a la dieta y el tratamiento con insulina es el tercer pilar fundamental en el tratamiento de la diabetes, ya que ayuda a controlar los niveles de glucosa en sangre, por lo que es aconsejable realizar al menos treinta minutos diarios de actividad física. Entre los beneficios podemos encontrar los mostrados en la Figura 2.5).[5]

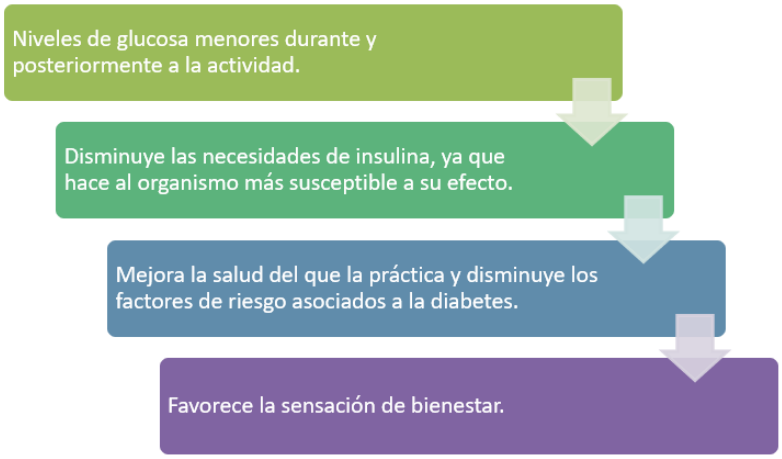


Figura 2.5: Beneficios de la actividad física en la diabetes

Existen dos tipos de actividad física, la aeróbica y la anaeróbica. La aeróbica se trata de una actividad mantenida en el tiempo de moderada o baja intensidad, de manera que mantiene una correcta oxigenación, entre este tipo de ejercicio encontramos el ciclismo, la natación, el futbol, etc. Sin embargo, la actividad física anaeróbica se trata de una actividad física intensa y de duración corta, como el levantamiento de pesas, que disminuye la oxigenación celular, además de aumentar la presión arterial, es por ello por lo que este tipo de actividades no son recomendables.[5]

Previa a la realización de cualquier actividad física hay que medir el nivel de glucosa plasmática, adecuar las necesidades energéticas y de insulina, no inyectar la insulina en los principales grupos musculares que vayan a intervenir en el ejercicio, y estar atento a las fluctuaciones de glucemia entre las doce y veinticuatro horas posteriores a la actividad física.[5]

2.11 Educación diabetológica

Numerosos estudios desarrollados a nivel internacional recalcan que la importancia de la educación en diabetes es fundamental para poder prevenir las complicaciones tanto a corto como a largo plazo que podría sufrir el niño. También son fundamentales la implantación de programas educativos que enseñen al niño y a la familia cómo actuar ante una hiperglucemia o hipoglucemia, cómo administrarse la insulina, conocer qué tipo de dieta deben llevar y dominar cuantas raciones supone un determinado plato, qué aspectos son importantes a la hora de realizar actividad física.[6]

Se ha demostrado que una correcta educación reduce el número de episodios agudos y por tanto de ingresos hospitalarios relacionados con un mal control de la enfermedad, así como la cantidad de días de ingreso si este llega a producirse. Es por ello importante proporcionar una adecuada educación diabetológica desde el primer momento en el que se detecta un debut diabético.[6]

Una parte fundamental de la educación diabetológica es la dietética ya que en gran medida va a determinar junto con la insulina y el ejercicio físico los niveles de glucosa en el organismo. A la hora de planificar la alimentación de una persona que padece diabetes hay que tener en cuenta una serie de aspectos como son la cultura, la dieta familiar y el contexto social. La dieta por raciones consiste en diferenciar los alimentos según son frutas y verduras, hidratos de carbono y proteínas, los cuales pueden intercambiarse dentro del mismo grupo, cada ración es aquella que contiene 10g de hidratos de carbono, que dependiendo del alimento se corresponderá con una cantidad u otra, ejemplo: una ración de arroz son 15g, mientras que una ración de pan se corresponde con 20g del alimento. Dependiendo de la edad del niño su dieta se compondrá de un número de raciones u otro, que se irán ajustando según sus necesidades y momento del desarrollo. En la Tabla 2.3 se muestran algunos alimentos en los que se puede ver representado a qué cantidad de alimento equivale una ración.[7]

2.12 La robótica y la educación

Hoy en día la robótica forma parte de la vida de los individuos de muchas formas diferentes, llegando a ser parte en su día a día. También la medicina ha apostado por las nuevas tecnologías con todos los beneficios que ello supone como por ejemplo la disminución de costos, ya que evita muchos desplazamientos por parte del personal sanitario, también puede servir como apoyo educativo en numerosos procesos, etc. Sin embargo, también es muy importante que las personas que los utilicen reciban una correcta formación para poder darles un uso adecuado y sacar el máximo provecho, por esto mismo es importante tener en cuenta que hay que adaptar los robots al contexto de las personas que lo van a utilizar.[8]

Tabla 2.3: Tabla de equivalencias entre raciones y alimentos

Alimento	**Cantidad**
Arroz	15g
Pasta	15g
Galleta tipo María	15g
Garbanzos cocidos	50g
Lentejas cocidas	50g
Pan blanco	20g
Patatas	50g
Melocotón	100g
Manzana	100g
Naranja	100g
Fresas	200g
Melón	200g
Plátano	50g
Uva	50g
Leche	200g

Capítulo 3

Motivación psicológica de la competición educativa

«El ascensor hacia el éxito está fuera de servicio, tienes que subir las escaleras poco a poco» — Mireia Belmonte

3.1 Definición

Como se ha anticipado en los objetivos generales del proyecto se pretende incorporar a la herramienta educativa los efectos de la competición entre alumnos para fomentar la motivación y la asimilación de información. En este contexto se realiza una breve revisión de los artículos científicos publicados donde se analizan los efectos positivos y negativos de incorporar la competición al medio educativo. La mayoría de los artículos relacionados con este tema tratan el efecto de la competición en la enseñanza de la educación física, pero se entiende que se puede realizar una correcta extrapolación pues el fin es el mismo, mantener la motivación y mejorar la asimilación de información y valores.

Además, los conceptos resultantes del proceso de revisión y documentación serán tenidos en cuenta para el diseño de los juegos en los que exista el componente de competición para así lograr el efecto de motivación deseado.

3.2 La competición como vector de cambio

Toda competición debe estar fundamentada en algún tipo de juego o desafío que suponga un reto para el que lo realiza. Siguiendo el entorno de trabajo Octalysis, de Yu-Kai Chou, el primer concepto que se introduce es la definición de gamificación, siendo esta el proceso por el cual se es capaz de trasladar todo aquello divertido y atractivo de los juegos al mundo real. Este concepto será el primero que habrá que tratar de implementar con éxito, como hacer que una tarea tan tediosa como el aprendizaje en diabetes puede asemejarse a un juego. Según Yu-Kai Chou la mayoría de las tareas relativas al trabajo o cosas que no le despiertan interés a la persona son tareas que siguen el diseño "basado en la función final". Es decir, se busca realizar la tarea lo más eficientemente posible sin tener en cuenta que el que las realiza es un ser complejo muy diferente

a un robot o máquina. La segunda forma de diseño de tareas es la que el autor llama tareas con un diseño "enfocado al ser humano", en ellas se entiende al ejecutor como un sistema con sentimientos, motivaciones e inseguridades. En estos conceptos será en los que habrá que hacer énfasis para que el ejecutor realice la tarea lo más eficientemente posible. [9]

Ligado al concepto anterior entra en juego el concepto de la competición como vector para hacer énfasis en esos elementos relacionados con los sentimientos, motivaciones e inseguridades. En un juego la competición es el componente principal que hace que el usuario desee seguir jugando, muchas veces la competición se ejecuta contra uno mismo al sentir ese impulso de superación o contra un ser inanimado como es el propio juego. Esto, con el avance de las nuevas tecnologías, está cambiando y está permitiendo conectar a muchas personas para poder competir entre ellas. Esta crea una nueva dimensión ya que la interacción con un competidor similar y en las mismas condiciones fomenta ese sentimiento de que no hay ningún motivo para ser peor que él.

Es importante también establecer las diferencias entre el juego y la gamificación. Según Foncubierta y Rodríguez (2014) el juego es un producto acabado cuya finalidad es el disfrute, en cambio la gamificación parte de un contenido didáctico que ellos definen como "una actividad aderezada con elementos o pensamientos del juego". Otros autores como Hamari y Koivisto (2013) también definen la gamificación como ese proceso cuya finalidad es la de influir en los participantes, sin centrarse exclusivamente en el disfrute y la diversión. La idea de gamificación que engloba todo lo anterior es la dada por Simoes et al. (2013), el cual describe la diferencia entre gamificación y juego como que: "los estudiantes aprenden, no jugando a juegos específicos, sino que aprenden como si estuvieran jugando a un juego". [10, 11, 12]

Otro concepto muy importante a la hora de incorporar la gamificación y la competición a una tarea es la necesidad de que el usuario pueda sentir claramente la recompensa de lo que realiza. Es decir, debe existir una relación directa evidente entre el desempeño del usuario en la tarea y la recompensa que recibe por el grado de éxito en ella. Esto comúnmente en los juegos online se consigue mediante sistemas de puntuación, barras de progreso o recompensas en monedas propias de la aplicación. Esto son métodos de recompensas que funcionan, pero no traspasan el mundo virtual siendo por ello su efecto limitado como vector de motivación. Por ello, es importante centrar los esfuerzos en buscar nexos de unión entre el mundo virtual y la realidad que es donde de verdad es necesaria la aplicación de los conocimientos adquiridos.

Autores como Ramón-Cortes (2011), defienden la idea de que la competición fomenta la capacidad de buscar nuevas estrategias para superar los retos y progresar tanto en el ámbito en el que se desarrolla la tarea concreta como en el desempeño de la vida y las relaciones humanas en general. Recalca también, la idea de que la competición desde nuestros orígenes es la que nos ha permitido progresar tanto individualmente como colectivamente. [13]

La competición está comprobada que tiene una influencia positiva en el mantenimiento de la motivación y el énfasis con en el que los usuarios realizan las tareas. Pero su uso incorrecto puede llevar a los usuarios a sentimientos contrarios a los buscados, como desilusión, desesperación o rechazo. Artículos como el de Durán (2013) evocan la idea de que la competición en último término debe referirse a una competición donde el competidor final sea uno mismo. Los pasos intermedios hasta lograr la asimilación de este concepto, donde se sucederán las derrotas y decepciones deberán ser tratadas como un elemento de superación y aprendizaje. Estos conceptos serán tenidos en cuenta en el diseño de la herramienta dando siempre la posibilidad al usuario

de poder volver a enfrentarse a la tarea para realizarla con mayor éxito. También se planteará un sistema de entrenamiento previo que permitirá al usuario dominar la herramienta y tener menores inseguridades a la hora de enfrentarse a ella y otros compañeros. [14]

De la breve revisión realizada se concluye que la incorporación de la gamificación y la competición al aprendizaje de tareas y transmisión de información presenta en general un efecto positivo. Este efecto se mantendrá siempre que exista una estrecha coordinación entre la forma que la herramienta transmite la información y como lo realiza el educador encargado de guiar a los alumnos en el uso de esta. Otra idea importante extraída es la necesidad de gestionar correctamente las derrotas y desilusiones encauzándolas como una oportunidad de superación y aprendizaje en lugar de representar menosprecio y hundimiento.

Capítulo 4

Recursos previos y limitaciones de partida

«Deberíamos usar el pasado como trampolín y no como sofá» — Harold MacMillan

4.1 Desarrollo previo de partida

Una vez realizada la documentación y elaboración del contexto en el que se va a desarrollar el proyecto es necesario plantear que forma va a tener el producto resultante de él. Basados en esta idea, y teniendo en cuenta que los objetivos generales del proyecto son bastantes ambiciosos se plantea no empezar el desarrollo desde cero. En este contexto se plantea la utilización de un desarrollo previo propiedad del grupo de investigación Tecnodiabetes que permitiera mediante su modificación y ampliación alcanzar los objetivos planteados en el Capítulo 1 de este documento.

El desarrollo previo escogido se trata de un pequeño minijuego desarrollado en el software de desarrollo Unity, cuyo objetivo es el de ayudar a los niños con diabetes tipo 1 a aprender sobre la planificación diaria que deben realizar en cuanto ingestas, realización de ejercicio y administración de terapias (insulina o snack). Este desarrollo se compone de tres pantallas con una interfaz gráfica adecuada para el público al que va destinado.

4.1.1 Pantalla de inicio del juego

Esta es la pantalla a la que se accede al arrancar el juego y es una pantalla que permite acceder a la siguiente pantalla de la aplicación, salir de ella, o ver los créditos de los creadores y productores. A nivel estético presenta un paisaje por donde una animación del personaje del juego montado en un avión va atravesando la pantalla constantemente describiendo una trayectoria diferente la cual se repite por ciclos (Figura 4.1).

Figura 4.1: Captura de la pantalla inicial del desarrollo previo

4.1.2 Pantalla de planificación diaria

En esta pantalla se muestra una escala de tiempos que recoge todas las horas de un día en intervalos de 30 minutos. En la parte izquierda de la pantalla hay una serie de menús desplegables que permiten seleccionar iconos entre diversos tipos de ingesta, ejercicios o terapias. El usuario podrá arrastrar los iconos que desee y colocarlos en la línea de tiempos en la posición (franja horaria) que desee. Una vez colocados, mediante la pulsación en cada uno de ellos, aparecerá un menú donde poder introducir la cantidad en raciones asociada a ese evento representado por el icono. De esta forma el usuario podrá crear a su antojo la planificación diaria que desee para luego ver su efecto sobre la simulación de la glucosa en sangre de un paciente virtual. Además de lo anterior, esta pantalla permite la eliminación mediante el arrastre a la papelera de alguno de los iconos colocados o cambiar el vehículo en el que el personaje del minijuego realizará la visualización de la simulación en la segunda pantalla (ver Figura 4.2).

Cuando el usuario finalice la composición de los eventos a lo largo del día este podrá mediante el botón “Despegar” acceder a la siguiente pantalla donde visualizar el efecto de su planificación en una simulación. En este momento el sistema realizará la lectura de los eventos colocados y los codificará para poder ser introducidos en el simulador.

Figura 4.2: Captura de la pantalla de planificación diaria del desarrollo previo

4.1.3 Pantalla de visualización de la simulación

En esta pantalla el minijuego introduce los eventos colocados por el usuario en un simulador de un paciente tipo con diabetes. Este simulador devuelve la curva de glucosa en sangre durante un día dividida en 235 pasos. Con esta información, en esta pantalla se representa a velocidad baja la evolución de esta curva de glucosa mediante un personaje que representa a un cerebro montado en un avión cuya trayectoria es la propia curva de glucosa dada por el simulador. A lo largo de la simulación el cerebro ira atravesando diferentes lugares y se mostrarán los cambios entre el día y la noche simulando que el personaje está realizando el viaje a lo largo de todo un día (Figura 4.3).

Además, tanto las expresiones del personaje como la interfaz gráfica se adaptarán al estado glucémico en el que se encuentre la simulación para avisar al usuario si algo va mal. En cualquier momento el usuario podrá realizar una medición de la glucosa en sangre con el glucómetro. En este momento la visualización de la simulación se detendrá y se le permitirá a este administrar una acción terapéutica (bolo de insulina o snack) con el objetivo, si es necesario, de volver y mantenerse el mayor porcentaje de tiempo posible en normoglucemia.

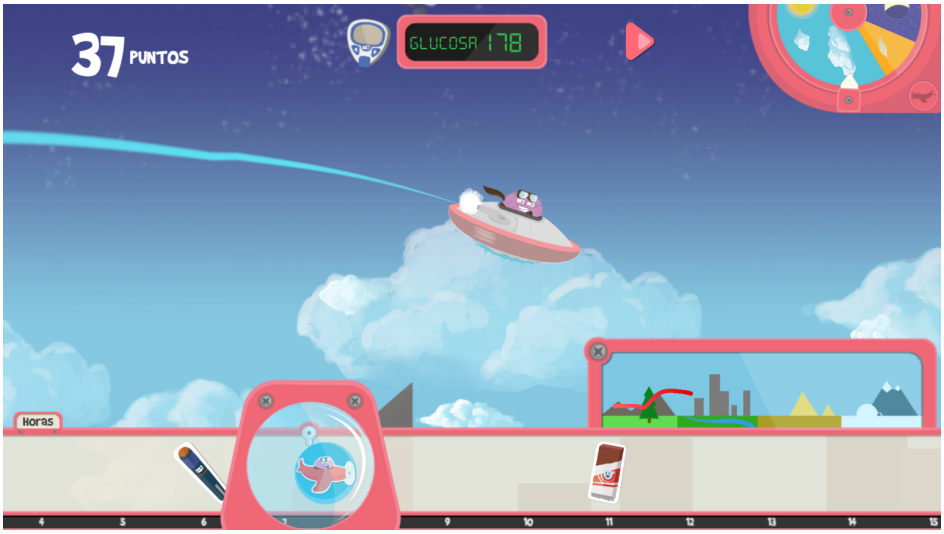


Figura 4.3: Captura de la pantalla de visualización de la simulación del desarrollo previo

Al finalizar la simulación se le muestra al usuario un panel donde se encuentran los porcentajes de tiempo que ha pasado en hiperglucemia, normoglucemia e hipoglucemia. También se mostrará una puntuación calculada en base a esos tres porcentajes de tiempo.

Las ecuaciones utilizadas en el simulador que modelan al paciente con diabetes tipo 1 se pueden consultar en el apartado 4.1.4.

4.1.4 Simulador de paciente con diabetes tipo 1

El simulador consiste en la modelización de un subsistema de la glucosa que recoge las fases de absorción, distribución y eliminación. Equivalentemente, se modela el subsistema de la insulina. Además, el modelo tiene en cuenta la acción que la insulina produce en el transporte de la glucosa, así como en su eliminación como producción endógena. [15]

El modelo se fundamenta en dos ecuaciones diferenciales que representan la cinética de la glucosa en el cuerpo humano:[15]

$$\frac{dQ_1(t))}{dt} = -[\frac{F_{01}^c}{(V_G G(t)} + x_1(t)]Q_1(t) + k_{12}Q_2(t) - F_R + U_G(t) + EGP_0[1 - x_3(t)] \tag{4.1}$$

$$\frac{dQ_1(t)}{dt} = x_1(t)\ Q_1(t) - [k_{12} + x_2(t)]Q_2(t)y(t)G(t) = \frac{Q_1(t)}{V_G} \tag{4.2}$$

Q_1 y Q_2 representa, respectivamente, las masas de glucosa en la zona de medida (accesible) y en la zona donde no se realiza medida alguna (no accesible). K_{12} representa la tasa de transferencia constante entre la zona no accesible y la accesible, V_G simboliza el volumen distribuido de la zona accesible y $G(t)$ es la concentración de glucosa medible. EGP_0 representa la producción endógena de insulina extrapolada al cero de la concentración de insulina. F_{01}^c es el flujo de glucosa no dependiente de la insulina. Y F_R representa a la glucosa renal.[15] La absorción de la insulina se modela de la siguiente forma:[15]

$$\frac{dS_1(t)}{dt} = u(t) - \frac{(S_1(t))}{t_{(max,I)}} \tag{4.3}$$

$$\frac{dS_2(t)}{dt} = \frac{S_1(t)}{t_{(max,I)}} - \frac{(S_2(t))}{t_{(max,I)}} \tag{4.4}$$

S_1 y S_2 representan la absorción de la insulina de corta acción administrada de forma subcutánea, $u(t)$ modela la administración de la insulina (bolo e infusión). Y $t_{(max,I)}$, es el tiempo para que se produzca la absorción completa de la insulina.[15]

La concentración de insulina en plasma se modela:[15]

$$\frac{dI(t)}{dt} = \frac{U_1(t)}{V_1} - k_e\ I(t) \tag{4.5}$$

Donde k_e es la tasa de eliminación fraccional y V_1 es el volumen de distribución. Por su parte U_1 se define como $U_1 = S_2(t)/t_{max}$.[15]

El modelo define tres acciones de la insulina sobre la cinética de la glucosa:[15]

$$\frac{dx_1}{dt} = -k_{a1}\ x_1\ (t) + k_b 1\ I(t) \tag{4.6}$$

$$\frac{dx_2}{dt} = -k_{a2}\ x_2\ (t) + k_b 2\ I(t) \tag{4.7}$$

$$\frac{dx_3}{dt} = -k_{a3}\ x_3\ (t) + k_b 3\ I(t) \tag{4.8}$$

x_1 y x_2 representa el efecto de la insulina sobre el transporte de la glucosa, su eliminación y sobre la creación endógena. El factor K_{ai} con $i = 1, 2, 3$, representan tasas constantes de desactivación, así como K_{bi} representa las tasas de activación constantes.[15]

La implementación de estas ecuaciones en un código ejecutable requiere la utilización de un algoritmo de integración numérica. El utilizado en este simulador ha sido el método de Dormand-Prince. Este método es una adaptación del conocido método de Runge-Kutta. El algoritmo de Dormand-Prince mejora la exactitud con respecto a su algoritmo de base, se categoriza como un método de paso adaptativo. El algoritmo se basa en evaluar en cada paso la ecuación:[16]

$$y_{(n+1)} = y_n + h \sum_{i=1}^{s} b_i \; k_i \tag{4.9}$$

Donde:

$$k_1 = f(t_n \;\; , \; y_n), \tag{4.10}$$

$$k_2 = f(t_n + \;\; c_2 \; h \; , y_n + h(a_2 1 \; k_1 \;)), \tag{4.11}$$

$$k_3 = f(t_n + \;\; c_3 \; h \; , y_n + h(a_3 1 \; k_1 + a_3 2 \; k_2 \;)), \tag{4.12}$$

$$k_s = f(t_n + \;\; c_s \; h \; , y_n + h(a_s 1 \; k_1 + a_s 2 \; k_2 + \cdots + a_(s, s-1) \; k_(s-1) \;)) \tag{4.13}$$

y los parámetros b_i, c_i, a_{ij}, se obtienen de la tabla de Butcher correspondiente a este método.[16] A continuación, se evalúa la ecuación:

$$y^*_{(n+1)} = y_n + h \sum_{i=1}^{s} b^*_i k_i \tag{4.14}$$

Donde b_i corresponde al método de Runge-Kutta de quinto orden mientras que los términos b^*_i son del de orden cuatro.[16]

Por último, se calcula el error como la diferencia entre ambas ecuaciones:[16]

$$e_{(n+1)} = y_{(n+1)} - y^*_{(n+1)} = h \sum_{i=1}^{s} (b_i - b^*_i) k_i \tag{4.15}$$

y se define el paso utilizado como:[16]

$$s = \left(\frac{\varepsilon h}{2 \left| y_{(n+1)} - y^*_{(n+1)} \right|} \right)^{(1/5)} \tag{4.16}$$

Donde ε hace referencia a la tolerancia utilizada y h es el intervalo de tiempo anterior.[16]

4.2 Limitaciones del desarrollo previo

El desarrollo analizado en este capítulo cumple la función para la que fue diseñado, pero se queda muy lejos de poder cumplir los objetivos planteados al inicio de este documento. A continuación, se procede a describir las limitaciones que presenta este minijuego y que por tanto deberán ser desarrolladas a lo largo de este proyecto para poder ser suplidas:

- No permite interacción entre usuarios, está pensado para ser usado por un único usuario en su dispositivo electrónico.
- No incorpora ninguna funcionalidad de registro de usuarios ni de creación de perfiles de usuario.
- No permite la creación de escenarios previos, ni ser utilizado en talleres educativos donde exista un moderador y diversos alumnos.
- No dispone de la infraestructura necesaria para poder comunicarse con una base de datos y poder almacenar información.
- No incorpora ningún aspecto que pueda fomentar la competitividad entre usuarios.
- No integra a ningún dispositivo robótico ni físico ni virtual que pueda fomentar la interfaz hacia el usuario.

Capítulo 5

Planteamiento general de la herramienta educativa

«Tener una visión global de las cosas nos puede poner adelante en el camino hacia el éxito» — Anónimo

5.1 Enfoque principal

Este proyecto, teniendo en cuenta los objetivos generales y las limitaciones de recursos, se ha centrado en la creación de un juego donde puedan competir dos o más alumnos simultáneamente entre ellos, donde se podrán valorar los conocimientos que este posea en planificación de terapias (momento óptimo de administración de bolo de insulina o de snack) y toma de decisiones ante momentos en los que no se encuentre en un estado de normoglucemia.

Este enfoque didáctico tendrá tres interfaces:

- Interfaz en pantalla, donde el alumno verá de una forma lúdica el resultado de la simulación en función de la resolución propuesta al escenario planteado y las decisiones tomadas a lo largo de esta.
- Interfaz de robot físico, en ella el alumno verá como el robot es capaz de seguir un circuito de forma más rápida y eficiente en función del estado de glucemia en el que se encuentre la simulación. En este tipo de interfaz es donde competirán varios robots simultáneamente, controlados por las curvas de glucemia derivadas de las decisiones tomadas por los alumnos para la resolución de los escenarios planteados.
- Interfaz de robot virtual, esta modalidad será usada por el alumno para practicar la resolución de escenarios antes de acudir al taller educativo, la simulación de este escenario también podrá ser visualizada de forma virtual.

5.2 Interfaz en pantalla

Toda la interacción del alumno con el problema a resolver se realizará a través de esta vía. Es decir, este a través de su ordenador, teléfono o tableta podrá realizar por este orden las siguientes acciones:

1. Visualización, de forma visualmente atractiva, de los escenarios (ingestas y ejercicio a lo largo de un día) planteados por su educador en diabetes.
2. Resolución interactiva de los escenarios, colocando en el momento del día y cantidad óptimos las terapias a aplicar (administración de bolo de insulina o snack).
3. Visualización lúdico-interactiva de la curva de glucosa derivada de la simulación a partir de las entradas provenientes del escenario planteado y de las terapias aplicadas por el alumno. Durante el transcurso de la simulación, el alumno podrá en cualquier momento realizar una medición del nivel de glucosa en sangre y tomar la acción correctora (administración de terapia) que considere oportuna.

Para la programación de toda esta interfaz en pantalla se partirá de la base del desarrollo previo propiedad del grupo de investigación en diabetes enmarcado en el Instituto Ai2.

5.3 Interfaz de robot físico

En esta vertiente un pequeño robot móvil, simultáneamente al transcurso de la simulación, será el encargado de reproducir los efectos derivados de los estados glucémicos que esté atravesando la simulación. Esto se ha materializado en un robot que debe dar vueltas a un circuito de manera autónoma. La velocidad y precisión con las que el robot sea capaz de avanzar en el circuito vendrá marcada por el estado glucémico en el que se encuentre la curva de glucosa en sangre simulada en el dispositivo que posee el alumno.

Con este concepto de una forma muy visual, extrapolable a cualquier deporte de motor, el alumno asociará a que si quiere que su robot avance más rápido y gane al de su compañero, deberá resolver de la forma más adecuada posible (mayor tiempo en normoglucemia durante la simulación) los escenarios planteados por el educador en diabetes.

5.4 Interfaz de robot virtual

La herramienta debe contar con la modelización y virtualización del dispositivo robótico para poder ser accesible desde cualquier lugar. De esta forma el alumno podrá prepararse y entrenar con la herramienta antes de acudir a los talleres de educación en diabetes.

El hecho de que se disponga de la virtualización del circuito y pueda verse como un modelo del robot real actúa de forma similar a la realidad fomenta en el alumno la capacidad de utilizar la herramienta como si en el taller educativo se encontrara.

Capítulo 6

MBot, el dispositivo robótico

«La robótica abre la puerta a un micro-mundo de aprendizaje motivador y entretenido. El establecimiento de un vínculo entre el mundo digital y el mundo físico ayuda a presentar a los alumnos una tecnología clave para el futuro» — Didier Roy

6.1 Descripción del dispositivo robótico

MBot es un robot educativo móvil creado por la empresa Makeblock, principalmente diseñado para iniciar en la robótica y en la programación a los niños desde la educación primaria. Su chasis es de aluminio extruido anodizado de 2 mm de grosor el cual le confiere un aspecto robusto y con la apariencia de estar dirigido a un público adulto.(Figura 6.1).

Figura 6.1: Imagen del dispositivo robótico mBot

6.1.1 Placa base del dispositivo robótico

MBot está basado en una placa Arduino UNO gobernada por el microcontrolador Atmega328 alimentada por cuatro pilas LR6 o una batería de lítio de 1450 mAh de capacidad a 3.7 V. Este chip es desarrollado por la casa Atmel y se trata de un circuito integrado de 8 bits que entre muchas características posee 32 Kb de memoria flash que permite lectura y escritura, 1Kb de memoria EEPROM y un convertidor analógico digital de 10 bits de resolución. Estas características pueden parecer muy básicas, pero como se justificará en el Apartado 6.2 son suficientes para la aplicación a la que se va a destinar el dispositivo robótico (Figura 6.2). [17]

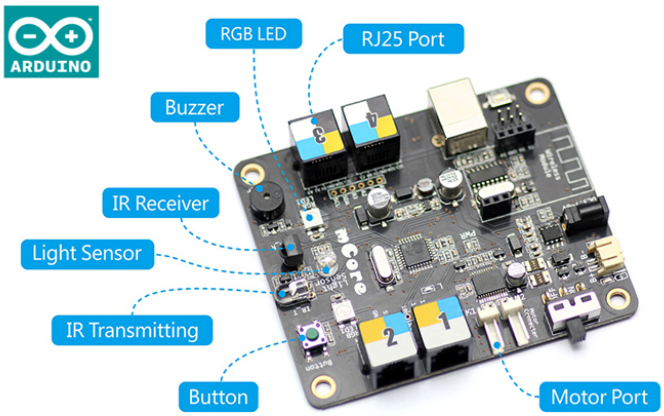


Figura 6.2: Imagen de la placa base del robot mBot

6.1.2 Conexiones de los sensores a la placa base

Las conexiones a esta placa están configuradas para ser a través de puertos RJ25 por lo que no es necesario realizar ningún tipo de soldadura para acoplar los hasta cuatro sensores que admite la placa. Esta conexión es una variante del conector RJ11, usado en líneas telefónicas, con la diferencia de que dispone de 6 pines en vez de 4.[17]

El hecho de que las conexiones con la placa no sean realizadas a través de los pines de esta, implica que su control no sea mediante la colocación a nivel alto o bajo de un determinado bit. Es por ello, por lo que la programación de los sensores utilizados requiera de la utilización de las librerías para Arduino proporcionadas por Makeblock.

6.1.3 Accionamiento del robot

El movimiento del mBot está basado en una configuración diferencial con dos ruedas motrices traseras y una rueda delantera que gira libre solo en una dirección. Las ecuaciones que definen este movimiento y su modelado serán definidas en el Apartado 8.3.

Por su parte las ruedas motrices están accionadas mediante dos motores de corriente continua a 6v con una velocidad angular máxima de 200 rpm y una reducción de 1/48 lo que permite un par motor mayor y que el robot pueda avanzar a velocidades bajas. No disponen de encoder ni son motores paso a paso por lo que es imposible realizar ningún tipo de control en bucle cerrado.

Es por ello por lo que la única forma de interactuar con estos motores es variando su voltaje de entrada, el cual cuando es negativo hace girar a los motores en sentido contrario provocando el retroceso del dispositivo robótico (Figura 6.3).[17]

Figura 6.3: Imagen de uno de los motores del robot mBot

6.1.4 Sensores y componentes de mBot

Los sensores incluidos en el dispositivo robótico son un sensor de ultrasonidos con el que medir distancias a objetos, un sensor de luminosidad y un sensor siguelíneas. Este último va a ser el único utilizado en este proyecto.

Por otro lado, como elementos de interfaz el dispositivo robótico presenta dos leds RGB, un zumbador y una matriz de leds de 8x16 leds azules. Todos ellos serán utilizados para acentuar la interfaz proporcionada por el robot creando expresiones y rutinas características de cada estado glucémico que atraviese la simulación de la curva de glucosa del paciente.[17]

El sensor siguelíneas

Este sensor dispone de dos diodos emisores y dos fototransistores los cuales se basan en la emisión y recepción de radiación lumínica en el espectro infrarrojo. De esta forma si uno de los sensores se encuentra sobre un fondo negro este absorberá toda la radiación emitida y el receptor no recibirá ninguna señal. Al disponer de dos pares emisor-receptor se puede saber si el robot se ha desviado de la línea por la izquierda o por la derecha y corregir su trayectoria. En la Figura 6.4 se observa el funcionamiento de este sensor.[18]

Con este concepto de codificación binaria y teniendo en cuenta que los motores que dan movimiento al robot no cuentan con encoder, el único control posible es un todo o nada siguiendo el siguiente algoritmo basado en toda la casuística posible (ver Tabla 6.1).

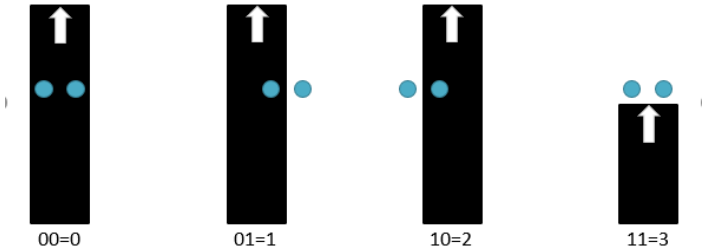


Figura 6.4: Casuística del sensor siguelíneas gráficamente

Tabla 6.1: Tabla resumen de la casuística del sensor siguelíneas

Casuística (basada en la Figura 6.4)	**Acción de control**
Sensor siguelíneas= 0	Continuar recto (misma velocidad en ambas ruedas)
Sensor siguelíneas= 1	Girar a la izquierda (rueda derecha en sentido contrario al de avance del movimiento)
Sensor siguelíneas= 2	Girar a la derecha (rueda izquierda en sentido contrario al de avance del movimiento)
Sensor siguelíneas= 3	Marcha atrás recto hasta volver a encontrar la línea.

6.1.5 Conexión inalámbrica del dispositivo robótico

La versión de mBot utilizada incorpora un módulo Bluetooth dual (compatible con versión clásica y BLE) conectado directamente a la placa base. Este es alimentado a 5v entre los pines "Vcc" y "GND" y conectado a la placa en la configuración típica donde el pin "Tx" (transmisión) se conecta al de recepción de la placa y el pin "Rx" (recepción) al de transmisión de esta (Figura 6.5). [19]

Este módulo es de tipo "esclavo" por lo que no puede iniciar la comunicación. En esta aplicación como el robot solo se va a usar como interfaz que recibe órdenes esto no será un problema. Además, el módulo solo permite un canal de comunicación, por lo que no es posible establecer ni alternar la comunicación entre más de un dispositivo y el robot.

La transmisión de información a través de este módulo se realiza a través del puerto serie (interfaz de comunicación transmisor-receptor) mediante el cual se envía la información en bits. Para este proyecto se ha establecido una velocidad de transmisión ("baud rate") de 115200 bits/s la cual empíricamente se ha comprobado que ofrece buenos resultados.

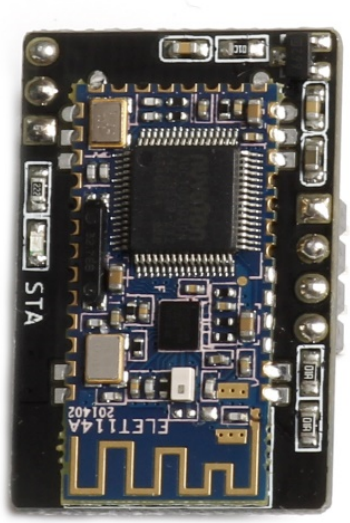


Figura 6.5: Imagen del módulo Bluetooth del robot mBot

6.2 Justificación de la elección del dispositivo robótico

Se ha elegido al mBot como el dispositivo robótico más adecuado frente a otras alternativas como los robots de Lego (menos económicos) o el robot Zowi de BQ (menos funcional), por las siguientes cuestiones:

- Es económico, pues tiene un precio de 100 € en su versión bluetooth con la matriz de leds incluida.
- Es un robot móvil mediante ruedas, el cual lo hace ideal para el tipo de interfaz siguelíneas planteada.
- Ofrece un sensor siguelíneas y elementos de interfaz como un zumbador, leds RGB y la matriz de leds.
- Está basado en Arduino por lo que permite una rápida y sencilla programación pudiendo centrar los esfuerzos en otras tareas del desarrollo.
- Su movimiento basado en una configuración diferencial permite su modelado y virtualización de una forma más sencilla que por ejemplo una configuración bípeda.

6.3 Entornos de programación del mBot

6.3.1 Programación por bloques mediante mBlock

Este software es propiedad de Makeblock, permite la actualización del firmware del robot y de las bibliotecas de funciones, así como su programación mediante bloques. La carga de los programas en la placa Arduino se pueden realizar mediante cable o Bluetooth. Su funcionamiento está basado en Scratch, un lenguaje de programación gráfico creado por el MIT para permitir la programación sencilla del robot sin necesitar conocimientos profundos de programación. [20]

Una de las características más útiles de mBlock es que realiza la traducción simultánea del algoritmo creado por bloques a código Arduino. Por ello, en este proyecto este software ha sido usado para conocer la sintaxis de las funciones necesarias para el control del robot y el manejo de sus sensores y actuadores.

6.3.2 Programación por código mediante el IDE de Arduino

Arduino es un proyecto desarrollado en la primera década del siglo XXI a partir de un proyecto de investigación en el Interaction Design Institute de Ivrea (Italia). Este engloba tanto la producción hardware (placas Arduino), como software (lenguaje de programación y Arduino IDE).[21]

Arduino es ampliamente usado como primer eslabón para iniciarse en la programación por código (basado en C/C++) y la robótica debido a su sencillez y disponibilidad. Al ser un producto muy económico y accesible es muy frecuente encontrarlo en el centro de muchos proyectos de creación propia y prototipado.

Como se ha comentado anteriormente, Arduino cuenta con su propio entorno de programación (Arduino IDE) el cual permite la generación de algoritmos mediante código, su depuración y su transmisión a las placas Arduino o aquellas basadas en este sistema. Por su versatilidad y sencillez de uso este entorno de programación ha sido el utilizado para el desarrollo del código que dará vida al dispositivo robótico. Se ha primado esta opción frente a mBlock, ya que con cierto conocimiento de programación es mucho más eficiente y versátil la programación por código que por bloques gráficos.

6.4 Estructura del código Arduino implementado en el robot

Como se ha comentado en el Apartado 6.1.5, el dispositivo robótico solo va a actuar como interfaz recibiendo órdenes a través del único canal de comunicación con la aplicación, siendo este el puerto serie Bluetooth.

Es por ello, por lo que el código se ha estructurado como un bucle infinito en el que el robot está comprobando con un cierto periodo de tiempo el valor almacenado en ese puerto serie. Cuando ese valor coincida con el valor que indique que el robot entre en el modo competición, este dejará ese bucle principal y entrará en otro donde seguirá leyendo continuamente el puerto serie a la espera de comandos. Pero en este caso, además de seguir leyendo el puerto serie el robot comenzará autónomamente a seguir las líneas y curvas que conforman el circuito gracias al sensor siguelíneas y a las funciones creadas para su movimiento.

6.4.1 Comandos que puede recibir el robot en el modo competición

El robot en este modo puede recibir las siguientes instrucciones con las que cambiar su comportamiento autónomo en función de la simulación que está teniendo lugar en el dispositivo del alumno. Estas pueden ser:

- Cambio del estado a normoglucemia.
- Cambio del estado a hiperglucemia.
- Cambio del estado a hipoglucemia.
- Detención del movimiento por medida de glucosa.

Estado de normoglucemia

Este es el estado natural y con el que comenzará la simulación. En este estado el robot avanzará a una velocidad alta y precisión en el seguimiento de las líneas máxima. Además, sus leds interiores serán encendidos de color verde y se mostrará una expresión facial en la matriz de leds indicando que el estado es el adecuado y en el que se debería estar el mayor tiempo posible (ver Figura 6.6).

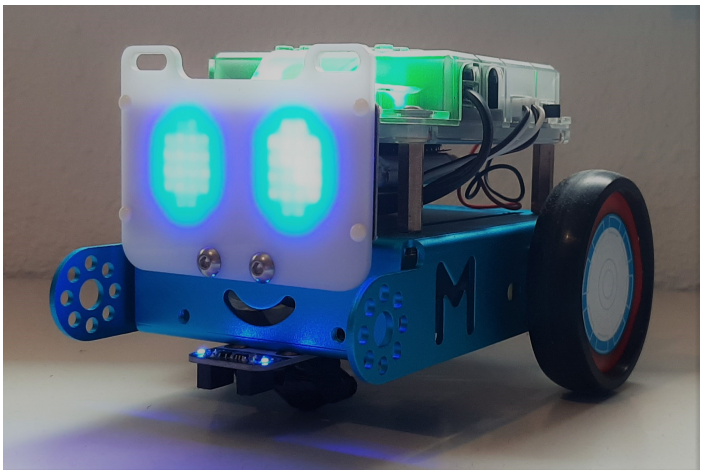


Figura 6.6: Robot en estado de normoglucemia

Estado de hiperglucemia

En este estado el robot intenta transmitir los síntomas que tendría una persona atravesando una hiperglucemia. Por ello las acciones asociadas a este estado son la reducción de la velocidad del robot a baja simulando el cansancio, y una expresión facial con los ojos indicando ese mismo estado de debilidad. Además, los leds interiores se colocarán en color rojo indicando rápidamente que se ha cambiado a un estado no deseado. Este efecto de llamada de la atención será complementado con el sonido del zumbador en señal de alerta en el momento del cambio a este estado (ver Figura 6.7).[22]

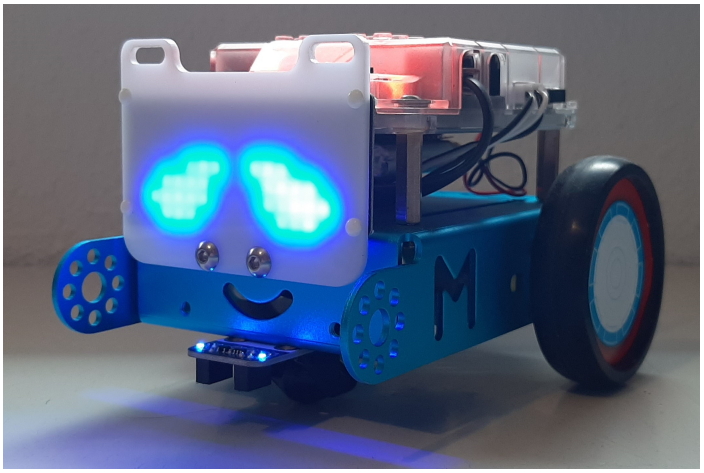


Figura 6.7: Robot en estado de hiperglucemia

Estado de hipoglucemia

Del mismo modo que en el anterior, cuando el robot recibe la orden de simular una hipoglucemia baja su velocidad a media y disminuye también la precisión en el seguimiento de las líneas del circuito simulando los síntomas de falta de concentración y mareos.[22]

La falta de precisión en el seguimiento del circuito se consigue con la implementación de un retardo entre la lectura del sensor siguelíneas y el envío a los motores de la acción de control. De esta forma se consigue simular un control "torpe" que actúa con acciones poco precisas y ajustadas a la realimentación que debería estar recibiendo.

En adición a estos efectos, se suman el encendido de los leds interiores a un color azul cyan intentado simular de algún modo los síntomas de entumecimiento de manos y temblores asociados a la hipoglucemia. También se muestra mediante la matriz de leds una expresión con los ojos muy abiertos denotando ese estado de irritabilidad y comportamiento extraño (ver Figura 6.8).

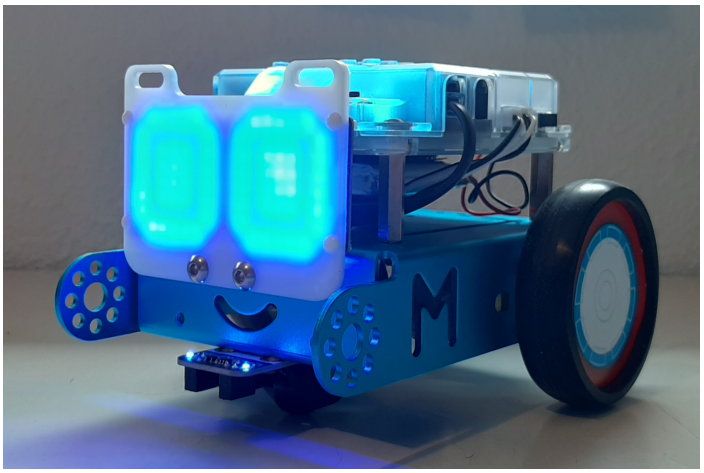


Figura 6.8: Robot en estado de hipoglucemia

Al igual que en el estado anterior, en el momento de entrada a este sonará el zumbador con una secuencia corta diferente que alerte que se está atravesando un momento no deseado.

Lectura de la medida de glucosa

A esta situación se puede llegar, o bien porque el usuario ha planificado la lectura del glucómetro en la resolución del escenario, o porque el alumno durante el transcurso de la simulación ha visto necesario realizar una medición de la glucosa en sangre para poder aplicar una terapia correctora.

En cualquiera de los casos, con el objetivo de que el robot no pueda dar más vueltas al circuito por el hecho de permanecer en esta pausa de manera indefinida, se ha programado un comando que hace detenerse al robot cuando la aplicación le envía por Bluetooth que se está realizando una medición con el glucómetro.

Durante el transcurso de esta medición el robot continuará con los elementos de interfaz activos correspondientes al estado glucémico que estuviese atravesando en el momento de la parada. De esta forma, el usuario podrá conocer de forma reafirmada que acción correctora necesitaría aplicar para volver a un estado de normoglucemia en el caso de que esto fuese necesario.

De esta parada se saldrá cuando el usuario aplique la terapia correctora o simplemente no aplique nada porque piense que no es necesario. Desde ese momento el robot reanudará la marcha entrando en el estado glucémico correspondiente al nivel de glucosa que se esté experimentando en ese momento.

Programación de la función de movimiento del robot

Como todas las demás acciones que realiza el robot, el movimiento de este también es gobernado por una función de la librería proporcionada por Makeblock. Existe una función independiente para mover cada una de las ruedas del robot necesitando como parámetro de entrada el porcentaje sobre la velocidad máxima de giro del motor.

Con el objetivo de hacer más eficiente el manejo del movimiento del robot se ha programado una función auxiliar que a partir de unos parámetros más generales pueda aplicar las funciones de movimiento elementales comentadas anteriormente. Esta función general precisa como parámetros de entrada un valor entero (identifica si se requiere seguir recto, realizar un giro a izquierda o derecha o ir marcha atrás), el porcentaje sobre la velocidad máxima a la que se quiere realizar el movimiento y, por último, el tiempo en milisegundos de retardo en el control en el caso de estar en un estado de hipoglucemia (Figura 6.9).

```
void move(int direction,int retardohipo, int speed)
{
      int leftSpeed = 0;
      int rightSpeed = 0;
      if(direction == 1){
          delay(retardohipo);
          leftSpeed = speed;
          rightSpeed = speed;
      }else if(direction == 2){
          delay(retardohipo);
          leftSpeed = -speed;
          rightSpeed = -speed;
      }else if(direction == 3){
          delay(retardohipo);
          leftSpeed = -speed;
          rightSpeed = speed;
      }else if(direction == 4){
          delay(retardohipo);
          leftSpeed = speed;
          rightSpeed = -speed;
      }
     motor_9.run((9)==M1?-(leftSpeed):(leftSpeed));
     motor_10.run((10)==M1?-(rightSpeed):(rightSpeed));
}
```

Figura 6.9: Función de movimiento del código del robot

Capítulo 7

La aplicación multiusuario

«La tecnología no es nada. Lo importante es que tengas fe en la gente, que sean básicamente buenas e inteligentes, y si les das herramientas, harán cosas maravillosas con ellas» — Steve Jobs

7.1 Descripción de la aplicación

Tras valorar todas las limitaciones presentes en el desarrollo previo y teniendo como meta lograr los objetivos planteados en el Capítulo 1 de este documento, se ha elaborado una herramienta educativa basada en una aplicación multiplataforma y multiusuario programada en Unity. La interfaz que proporcione esta será complementada con un dispositivo robótico como es el mBot totalmente integrado en la aplicación. El objetivo principal de la aplicación creada es la de enseñar en un ambiente de competitividad, controlada con amigos, los conocimientos necesarios que un niño debutante en diabetes debe conocer sobre planificación de la administración de terapias.

7.2 Software de desarrollo

Como entorno de desarrollo se ha utilizado el motor de desarrollo de videojuegos Unity, el cual se programa dentro de él con el lenguaje C#.

7.2.1 Unity, el entorno de programación

Unity es un motor de desarrollo de videojuegos multiplataforma en 2D o 3D desarrollado por la compañía Unity Technologies. Permite la compilación del juego o aplicación a plataformas como Windows, Android, iOS, WebGL, PlayStation, Xbox, Android TV, Linux o VR. Esto hace que, junto a la disponibilidad de su versión gratuita con plenas capacidades, sea uno de los entornos de programación más utilizados por desarrolladores de juegos y aplicaciones independientes. Aunque este motor de desarrollo también es usado para la creación de videojuegos tan conocidos como Pokemon GO o Super Mario Run .[23]

A todo lo anterior se le añade un muy potente motor físico como es el PhysX de NVIDIA y una gran comunidad de desarrolladores que implementan complementos (*assets*) con los que añadir funcionalidades a la aplicación de forma sencilla.[23]

Todas estas características, unido a que el desarrollo previo de partida estaba realizado en este entorno de desarrollo, ha hecho que Unity haya sido la plataforma elegida para este proyecto. Se ha utilizado la versión estable más reciente de Unity (2019) con el objetivo de mantener actualizada la herramienta el mayor tiempo posible. Esto ha provocado que haya sido necesario la actualización del desarrollo previo de partida a esta versión realizando el intercambio o actualización de los métodos y funciones obsoletas para que funcionen sin problemas en la versión actual.

7.2.2 C#, el lenguaje de programación

Este es un lenguaje de programación multiparadigma y orientado a objetos diseñado por la empresa Microsoft. Fue creado para facilitar la programación en su plataforma .NET y ha sido estandarizado por la ECMA e ISO. [24]

C# es una evolución de C/C++ como su nombre indica, ya que su almohadilla ("sharp") representa el sostenido musical que representa a un tono superior. Además, la almohadilla se puede interpretar como cuatro signos "+" que representan la evolución frente a los dos que posee C++.[24]

Al ser un lenguaje de programación orientado a objetos permite la creación y manejo de clases, siendo este un enfoque mucho más intuitivo a la hora de programar.

7.2.3 Visual Studio

Este es un entorno de desarrollo integrado (IDE) muy potente desarrollado también por Microsoft. Admite la creación de aplicaciones en multitud de lenguajes de programación como C++, C#, Visual Basic, .Net, o Java entre otros.[25]

Visual Studio es muy potente por la cantidad de parámetros configurables y extensiones disponibles. También lo es por la incorporación de tecnologías que facilitan y hacen más ágil la programación como "Intellisense", la cual autorrellena pequeños fragmentos de código y automáticamente muestra características de las clases como los métodos y atributos que contiene.[25]

En este proyecto ha sido el IDE usado para la escritura de los *scripts* que dan vida a la herramienta en lenguaje C# debido a que Visual Studio ofrece una muy buena integración con Unity.

7.3 Concepto multiusuario

Con el objetivo de crear una herramienta que permita la competición a tiempo real entre varios usuarios se necesita crear la infraestructura necesaria que sostenga este concepto. Esto se hará posible con la definición de diferentes perfiles de usuario que puedan mantener una comunicación fluida mediante una base de datos en la nube. Para que todos los perfiles de usuarios se puedan comunicar entre sí también será necesario la definición de un protocolo de comunicación común a todos.

7.3.1 Perfiles de usuario

El sistema de educación diabetológica desarrollado no es solo una herramienta multiusuario porque permita la interacción a tiempo real de varios alumnos en una competición, sino también porque maneja diferentes perfiles de usuarios cada uno de los cuales tendrán unos privilegios e itinerarios distintos dentro de la aplicación. La gestión de estos usuarios se realizará mediante un fichero alojado en la base de datos que contendrá los nombres de usuario que pertenezcan al grupo de los moderadores y otro para el grupo de superusuarios. Los usuarios cuyos nombres no estén en estos ficheros tendrán rol de alumnos.

7.3.2 Perfil de moderador

Este perfil es el segundo con más privilegios en la herramienta y es el dedicado a los educadores en diabetes que actúen como profesores en los talleres educativos. Este perfil tendrá acceso a la creación y modificación de escenarios, así como tendrá el control absoluto del transcurso de la simulación y la sincronización entre usuarios. Además, tendrá acceso a los resultados de las simulaciones y las resoluciones de los escenarios de los alumnos en el taller educativo.

7.3.3 Perfil de alumno

Este tipo de perfil solo permitirá el acceso a la resolución de los escenarios y a la visualización de la simulación tanto de forma virtual como real en los talleres educativos. En todo momento los usuarios con este rol serán guiados y sincronizados a través de la herramienta por los usuarios moderadores.

7.3.4 Perfil de superusuario

Este es el perfil con mayores privilegios dentro de la herramienta. Engloba todas las funcionalidades del perfil de moderador, pero además se le añade la posibilidad de agregar o eliminar usuarios como moderadores y de crear escenarios predeterminados visibles por todos los usuarios.

Este perfil se debe asignar desde el entorno de desarrollo de la aplicación y solo lo disfrutarán el desarrollador de la herramienta y los tutores de este proyecto, al ser estos los que han contribuido con los recursos necesarios para su creación.

7.4 Complementos de Unity incorporados al proyecto

Unity de forma nativa no incorpora ninguna funcionalidad de manejo de usuarios ni de comunicaciones con bases de datos en la nube. Por ello se hace necesario el uso de complementos (*assets*) creados por la comunidad de desarrolladores de Unity.

A estos complementos se accede mediante la tienda de Unity, los cuales previo pago, pueden ser incorporados al proyecto en desarrollo. Este modelo permite la disponibilidad de complementos que agilizan mucho el desarrollo de proyectos al no tener que implementar todas las funcionalidades que se quieran usar en un proyecto. Además, permite la consulta de dudas a los creadores de los *assets* sobre la integración y manejo de estos. Este recurso de consultas ha sido frecuentemente utilizado para el correcto manejo de estos complementos.

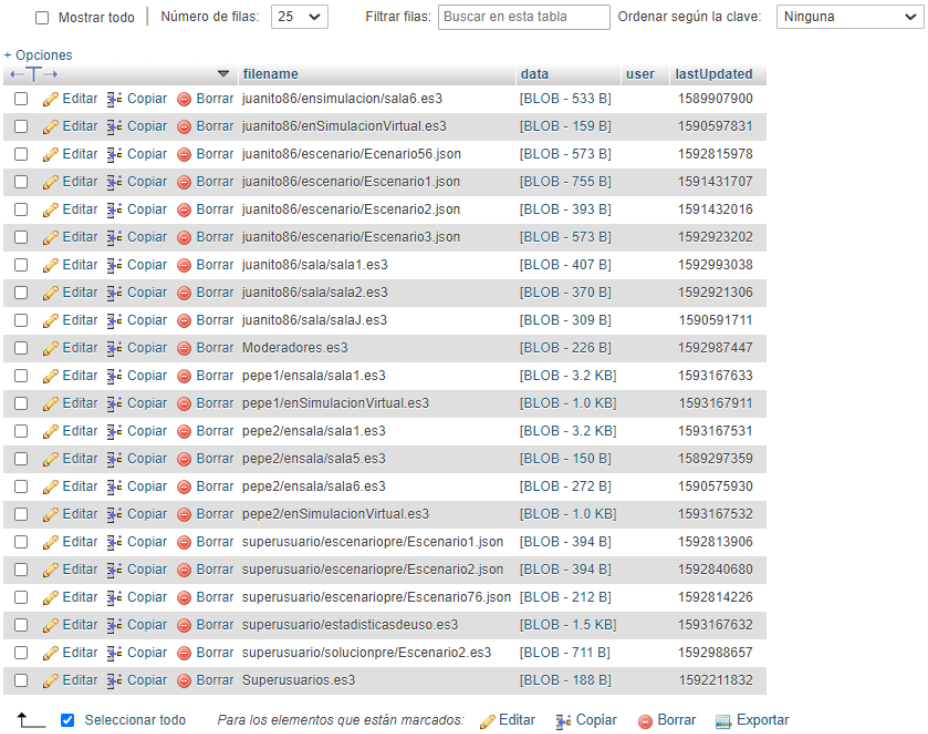
Para este proyecto, en concreto, han sido utilizados los *assets* siguientes:

- Registro e inicio de sesión de usuarios.
- Comunicación mediante ficheros con la base de datos.
- Comunicación Bluetooth con dispositivos basados en Arduino.

7.5 Base de datos

Como se ha comentado en apartados anteriores se necesita algún medio con el que gestionar el inicio de sesión y las comunicaciones de los distintos usuarios de la herramienta. Para ello se ha hecho uso de una base de datos alojada en los servidores del Instituto Ai2, a la cual poder escribir y realizar lecturas remotamente.

La gestión de esta base de datos SQL, con el fin de depurar errores y comprobar su correcto funcionamiento, se ha realizado mediante la herramienta software phpMyAdmin escrita en lenguaje PHP. Esta permite la administración, desde un navegador web, de bases de datos MySQL incorporando la visualización, subida de archivos y descarga de estos (Figura 7.1).


Mostrar todo | Número de filas: 25 Filtrar filas: Buscar en esta tabla Ordenar según la clave: Ninguna

+ Opciones

	filename	data	user	lastUpdated
Editar Copiar Borrar	juanito86/ensimulacion/sala6.es3	[BLOB - 533 B]		1589907900
Editar Copiar Borrar	juanito86/enSimulacionVirtual.es3	[BLOB - 159 B]		1590597831
Editar Copiar Borrar	juanito86/escenario/Ecenario56.json	[BLOB - 573 B]		1592815978
Editar Copiar Borrar	juanito86/escenario/Escenario1.json	[BLOB - 755 B]		1591431707
Editar Copiar Borrar	juanito86/escenario/Escenario2.json	[BLOB - 393 B]		1591432016
Editar Copiar Borrar	juanito86/escenario/Escenario3.json	[BLOB - 573 B]		1592923202
Editar Copiar Borrar	juanito86/sala/sala1.es3	[BLOB - 407 B]		1592993038
Editar Copiar Borrar	juanito86/sala/sala2.es3	[BLOB - 370 B]		1592921306
Editar Copiar Borrar	juanito86/sala/salaJ.es3	[BLOB - 309 B]		1590591711
Editar Copiar Borrar	Moderadores.es3	[BLOB - 226 B]		1592987447
Editar Copiar Borrar	pepe1/ensala/sala1.es3	[BLOB - 3.2 KB]		1593167633
Editar Copiar Borrar	pepe1/enSimulacionVirtual.es3	[BLOB - 1.0 KB]		1593167911
Editar Copiar Borrar	pepe2/ensala/sala1.es3	[BLOB - 3.2 KB]		1593167531
Editar Copiar Borrar	pepe2/ensala/sala5.es3	[BLOB - 150 B]		1589297359
Editar Copiar Borrar	pepe2/ensala/sala6.es3	[BLOB - 272 B]		1590575930
Editar Copiar Borrar	pepe2/enSimulacionVirtual.es3	[BLOB - 1.0 KB]		1593167532
Editar Copiar Borrar	superusuario/escenariopre/Escenario1.json	[BLOB - 394 B]		1592813906
Editar Copiar Borrar	superusuario/escenariopre/Escenario2.json	[BLOB - 394 B]		1592840680
Editar Copiar Borrar	superusuario/escenariopre/Escenario76.json	[BLOB - 212 B]		1592814226
Editar Copiar Borrar	superusuario/estadisticasdeuso.es3	[BLOB - 1.5 KB]		1593167632
Editar Copiar Borrar	superusuario/solucionpre/Escenario2.es3	[BLOB - 711 B]		1592988657
Editar Copiar Borrar	Superusuarios.es3	[BLOB - 188 B]		1592211832

Seleccionar todo *Para los elementos que están marcados:* Editar Copiar Borrar Exportar

Figura 7.1: Base de datos implementada en el proyecto

7.5.1 Asset de registro e inicio de sesión

Para poder diferenciar entre tipo de usuarios y poder organizar sus archivos en la base de datos es necesario establecer un mecanismo de registro e inicio de sesión de usuarios en la herramienta. Para incorporar esta funcionalidad se ha hecho uso del *asset* "Online Account System" desarrollado por Tall Guy Productions. [26]

Este complemento incorpora un *script* en lenguaje PHP que permite la creación y modificación de una tabla en la base de datos donde se almacenan los nombres de usuario y las contraseñas de aquellas personas que estén registradas en la herramienta. Este *script* debe ser alojado en el servidor para servir de puente entra la base de datos y la aplicación desarrollada en Unity.

Además, este complemento incorpora los *scripts* en C# necesarios para, desde la aplicación, realizar el registro de nuevos usuarios y gestionar sus inicios de sesión a través de la comprobación de la contraseña introducida con la almacenada en la base de datos.

7.5.2 Asset de comunicación con la base de datos

Como se ha anticipado en apartados anteriores para la comunicación entre usuarios y el almacenamiento de información para su posterior uso se ha optado por la utilización de una base de datos en la nube.

La comunicación entre la base de datos y la aplicación desarrollado con Unity se ha realizado mediante el *asset* "Easy Save" desarrollado por Moodkie. Este complemento al igual que el anterior incorpora un *script* PHP que crea en la base de datos una tabla de almacenamiento de información y hace de puente de comunicación entre esta y la aplicación. [27]

Este *asset* permite solo el almacenamiento y descarga de ficheros en multitud de formatos incluido uno propio con extensión ".es3". Es decir, no es posible el almacenamiento y actualización de variables por separado. Siendo necesario en el caso de querer consultar el valor de una variable, descargar el fichero de la base de datos a la memoria local del dispositivo que esté ejecutando la aplicación, abrir dicho fichero, leerlo y buscar la variable almacenada en él. Lógicamente esta metodología no es la más rápida ni eficiente para establecer una comunicación a tiempo real entre usuarios, pero tomando las medidas adecuadas expuestas en el Subapartado 7.5.3 siguiente se pueden conseguir resultados aceptables.

Cabe destacar que el uso de este complemento para la comunicación con la base de datos se debe principalmente al propósito de reducir el coste económico de desarrollo del proyecto. Ya que el equipo de investigación en el que se desarrolla este proyecto había adquirido con anterioridad este *asset* para otros proyectos previos. Además de que, para el almacenamiento y gestión de ficheros en la base de datos este si aporta funcionalidades muy interesantes, las cuales también serán ampliamente usadas en la programación de la herramienta.

7.5.3 Estructura de ficheros en la base de datos

La definición de una estructura de ficheros o protocolo de comunicación se hace necesaria por la asociación de cada uno de los ficheros a su usuario correspondiente y también con el objetivo de crear ficheros livianos que permitan una rápida descarga y lectura.

Con este objetivo presente se ha optado por no crear un solo fichero por usuario donde almacenar toda su información, sino que se ha implementado el concepto de salas. Este se basa en crear un fichero común al que tengan acceso tanto el moderador como todos los alumnos asistentes a la sala o en este caso taller educativo. Este formato facilita la sincronización de las acciones y el envío de información entre el moderador y los alumnos conectados en ese momento a la sala. De esta forma con el propósito de crear varios ficheros livianos en lugar de solo uno existirán los siguientes tipos de archivo en la base de datos:

Archivos de sala

Contendrá los campos identificativos de esta como el nombre de la sala y su contraseña, aparte de otros datos sobre los alumnos conectados a ella. Este archivo. al ser conocida su ruta por el moderador y los alumnos conectados en la sala, será el utilizado para la sincronización y envío de ordenes por parte del moderador a las aplicaciones en los dispositivos de los alumnos (ver Tabla 7.1).

Tabla 7.1: Información de archivos de sala

Trama	*nombreUsuario*/**sala**/*nombreSala*.**es3**
Ubicación	Base de datos
Usuarios con permiso de edición	▪ Moderadores (edición y visualización) ▪ Alumnos (solo visualización) ▪ Superusuarios (solo visualización)
Pantallas en las que se usa	▪ Pantalla de búsqueda de salas para el alumno ▪ Pantalla de búsqueda de salas para el moderador ▪ Pantalla de espera en sala para el alumno ▪ Pantalla de control de la sala por parte del moderador ▪ Pantalla de creación y modificación de escenarios ▪ Pantalla de control de superusuario

Archivos de estancia en sala

Cada alumno que se conecte a una sala generará un archivo de este tipo asociado a la sala a la que se ha conectado. En este archivo se almacenarán datos como la resolución del escenario y los resultados de la simulación para ser posteriormente visualizadas por el moderador (ver Tabla 7.2).

Tabla 7.2: Información de archivos de estancia en sala

Trama	*nombreUsuario*/**ensala**/*nombreSala*.**es3**
Ubicación	Base de datos
Usuarios con permiso de edición	▪ Moderadores (solo visualización) ▪ Alumnos (edición y visualización) ▪ Superusuarios (solo visualización)
Pantallas en las que se usa	▪ Pantalla de espera en sala para el alumno ▪ Pantalla de control de la sala por parte del moderador ▪ Pantalla de resolución de escenarios ▪ Pantalla de visualización de la simulación ▪ Pantalla de control de superusuario

Archivos de simulación virtual

Este tipo de fichero será generado cuando el usuario acceda al modo de simulación con el robot virtual. En él se almacenará la información relativa a la resolución del escenario propuesto en la simulación virtual, así como las variables necesarias para la sincronización con el compañero o el propio dispositivo con el que se va a competir (ver Tabla 7.3).

Tabla 7.3: Información de archivos de simulación virtual

Trama	*nombreUsuario*/**enSimulacionVirtual.es3**
Ubicación	Base de datos
Usuarios con permiso de edición	▪ Alumnos (edición y visualización) ▪ Superusuarios (solo visualización)
Pantallas en las que se usa	▪ Pantalla de selección de simulación con robot físico o virtual ▪ Pantalla de búsqueda de amigo para simulación virtual ▪ Pantalla de resolución de escenarios ▪ Pantalla de visualización de la simulación ▪ Pantalla de simulación con robot virtual ▪ Pantalla de control de superusuario

Archivos de escenarios creados por el moderador

Estos son archivos que almacenan los datos de los escenarios creados por el moderador y solo el tendrá acceso de edición a ellos para poder cargarlos en las salas que haya creado (ver Tabla 7.4).

Tabla 7.4: Información de archivos de escenarios creados por el moderador

Trama	*nombreUsuario*/**escenario**/*nombreEscenario*.**json**
Ubicación	Base de datos
Usuarios con permiso de edición	▪ Moderadores (edición si es propietario y visualización) ▪ Alumnos (solo visualización) ▪ Superusuarios (edición si es propietario y visualización)
Pantallas en las que se usa	▪ Pantalla de creación y modificación de escenarios ▪ Pantalla de creación y modificación de escenarios predeterminados ▪ Pantalla de resolución de escenarios ▪ Pantalla de visualización de la simulación ▪ Pantalla de simulación con robot virtual

Archivos de escenarios predeterminados

Estos ficheros almacenan los datos de los escenarios creados por los superusuarios, almacenando estos escenarios básicos y progresivos que permitan al moderador comenzar la educación de sus alumnos por ellos (ver Tabla 7.5).

Tabla 7.5: Información de archivos de escenarios predeterminados

Trama	**superusuario**/**escenariopre**/*nombreEscenario*.**json**
Ubicación	Base de datos
Usuarios con permiso de edición	▪ Moderadores (solo visualización) ▪ Alumnos (solo visualización) ▪ Superusuarios (edición y visualización)
Pantallas en las que se usa	▪ Pantalla de creación y modificación de escenarios ▪ Pantalla de creación y modificación de escenarios predeterminados ▪ Pantalla de resolución de escenarios ▪ Pantalla de visualización de la simulación ▪ Pantalla de simulación con robot virtual

Archivos de soluciones predeterminadas

En ellos se almacenan los estados relativos a la simulación de la resolución del escenario predeterminado dado por el usuario (ver Tabla 7.6).

Tabla 7.6: Información de archivos de soluciones predeterminadas

Trama	**superusuario**/**solucionpre**/*nombreSolucion*.**es3**
Ubicación	Base de datos
Usuarios con permiso de edición	▪ Alumnos (solo visualización) ▪ Superusuarios (edición y visualización)
Pantallas en las que se usa	▪ Pantalla de creación y modificación de escenarios ▪ Pantalla de creación y modificación de escenarios predeterminados ▪ Pantalla de resolución de escenarios ▪ Pantalla de visualización de la simulación ▪ Pantalla de simulación con robot virtual

Archivo de moderadores

En él se almacenan los nombres de usuario de los moderadores habilitados por los superusuarios (ver Tabla 7.7).

Tabla 7.7: Información de archivos de moderadores

Trama	**Moderadores.es3**
Ubicación	Base de datos
Usuarios con permiso de edición	▪ Moderadores (solo visualización) ▪ Alumnos (solo visualización) ▪ Superusuarios (edición y visualización)
Pantallas en las que se usa	▪ Pantalla de inicio de sesión

Archivo de superusuarios

En él se almacenan los nombres de usuario de los superusuarios (ver Tabla 7.8).

Tabla 7.8: Información de archivos de superusuarios

Trama	**Superusuarios.es3**
Ubicación	Base de datos
Usuarios con permiso de edición	▪ Moderadores (solo visualización) ▪ Alumnos (solo visualización) ▪ Superusuarios (solo visualización) ▪ Desarrollador de la herramienta (edición y visualización)
Pantallas en las que se usa	▪ Pantalla de inicio de sesión

Archivo de transferencia de información en la aplicación

Será el encargado de almacenar la información necesaria que se desea que compartan varias pantallas de la aplicación (ver Tabla 7.9).

Tabla 7.9: Información de archivos de transferencia de información en la aplicación

Trama	**Datos.es3**
Ubicación	Local en cada dispositivo
Usuarios con permiso de edición	▪ Moderadores (edición y visualización) ▪ Alumnos (edición y visualización) ▪ Superusuarios (edición y visualización)
Pantallas en las que se usa	▪ Todas las pantallas

Archivo de estadísticas de uso

En él se almacenan las estadísticas de las que el superusuario dispondrá en su panel de control sobre el uso que se está dando a la aplicación (ver Tabla 7.10).

Tabla 7.10: Información de archivos de estadísticas de uso

Trama	**superusuario/estadisticasdeuso.es3**
Ubicación	Base de datos
Usuarios con permiso de edición	▪ Moderadores (solo edición) ▪ Alumnos (solo edición) ▪ Superusuarios (edición y visualización)
Pantallas en las que se usa	▪ Pantalla de resolución de escenarios ▪ Pantalla de control de superusuario

7.6 Comunicación entre aplicación y robot

Para la comunicación entre la herramienta y el dispositivo robótico se ha utilizado la conexión Bluetooth. Unity tampoco tiene implementado por defecto los *scripts* que modulan este tipo de conexiones, por lo que es necesario recurrir a *assets* implementados por otros desarrolladores.

En este caso como el robot utilizado está basado en Arduino se ha recurrido a un *asset* específico para comunicar aplicaciones desarrollados con Unity con dispositivos basados en Arduino. Este complemento es el "Arduino Bluetooth Plugin" desarrollado por Tony Abou Zaidan, el cual incorpora tres *scripts* escritos en C# que permiten el establecimiento de la comunicación y el envío de datos a través de Bluetooth entre este tipo de dispositivos. [28]

7.7 Estética de la aplicación

Como para el desarrollo de esta herramienta se partía de un desarrollo previo se ha mantenido en todo lo posible la estética del que este disponía. De esta forma se han aprovechado todas las imágenes y diseños incorporados en el desarrollo previo para darles un nuevo uso completamente diferente en la herramienta desarrollada en este proyecto.

Esta reutilización permite un importante abaratamiento de costes temporales y económicos al no tener que crear nuevas imágenes. Además, permite crear cohesión en la aplicación al compartir todas las pantallas una misma filosofía de transmisión de la información mediante la interfaz gráfica.

Uno de los recursos estéticos más reutilizados del desarrollo previo es el fondo y la animación del avión incluidos en la pantalla principal. Estos recursos han sido incluidos en la mayoría de las pantallas para crear dinamismo y continuidad en la herramienta.

7.8 Estructura de la aplicación y las diferentes pantallas que la componen

La herramienta desarrollada en este proyecto presenta una complejidad significativa. Se ha partido del desarrollo previo de tres pantallas gobernadas por 15 archivos programados en C# con más de 2700 líneas en total.

De este desarrollo previo se han modificado sus dos pantallas principales y se han añadido 12 pantallas más junto a 25 *scripts* programados en C# sumando un total de más de 6000 líneas de código escritas para la realización de este proyecto.

En los apartados siguientes se detalla el funcionamiento e implementación de cada una de las pantallas desarrolladas en este proyecto.

7.8.1 Pantalla de inicio de sesión

Esta es la primera pantalla que se encuentra cualquier usuario de la herramienta y tiene dos funciones principales:

- Inicio de sesión mediante usuario y contraseña: Esto permite conocer quien está utilizando la aplicación y de este modo cargar y descargar de la base de datos la información propia a ese usuario.
- Discriminación entre perfiles de usuario: cuando el usuario inicia sesión se realiza una descarga y lectura del fichero de moderadores y superusuarios para conocer si pertenece o no a este perfil de usuario. Esta comprobación actualizará una variable en el fichero personal de cada usuario que hará saber a la herramienta que perfil de usuario la está usando.

Para la consecución de la primera de las funciones se han implementado dos campos de introducción de texto (usuario y contraseña) los cuales a través del *asset* "Online Account System" realizan con esos datos una petición de inicio de sesión a la base de datos. Con la misma filosofía se ha implementado un panel con campos de introducción de texto que formalizan el registro de un nuevo usuario en la herramienta en caso de que nunca la haya utilizado con anterioridad (ver Figura 7.2).

Para la implementación de la segunda función de discriminación de perfiles, se ha hecho uso del complemento de comunicación con la base de datos "Easy Save". Con sus métodos propios se realiza la descarga y lectura de los ficheros de usuarios moderadores y superusuarios para conocer si la persona que está usando la aplicación pertenece a alguno de estos perfiles.

En la parte baja de la pantalla se encuentra un gran botón de salir, el cual cerrará la aplicación por completo eliminándola también de las aplicaciones en segundo plano.

Figura 7.2: Captura de la pantalla de inicio de sesión

7.8.2 Pantalla de selección de simulación con robot físico o virtual

A esta pantalla solo llegarán aquellos usuarios catalogados como alumnos inmediatamente después de iniciar sesión. Aquí se les mostrará un panel con dos botones para que el alumno decida si quiere realizar la simulación junto al robot físico o junta a la virtualización de este (Figura 7.3).

Figura 7.3: Captura de la pantalla de selección de simulación con robot físico o virtual

En caso de decidir que el dispositivo robótico real le acompañe, aparecerá por pantalla un gran botón para realizar la conexión entre el robot y la aplicación. Esta acción desencadenará una serie de funciones apoyadas en el *asset* "Arduino Bluetooth Plugin" que intentarán realizar la conexión entre ambos dispositivos. Si esta no es satisfactoria el usuario será notificado mediante un texto. Por consiguiente, si la conexión entre ambos dispositivos se ha establecido correctamente aparecerá un nuevo panel en el que se le dará la posibilidad al usuario de realizar la desconexión de la comunicación o ser trasladado a la pantalla de selección de sala (Figura 7.4).

La otra vertiente que puede ser escogida por el usuario en esta pantalla es la de decidir realizar la simulación con el robot virtualizado. Por ello, en el caso de ser esta la opción seleccionada, se obviará todo el proceso de establecimiento de la conexión con el robot y se deberá elegir si se quiere realizar la simulación contra un amigo o contra la máquina. En el primero de los casos el usuario será trasladado a la pantalla de búsqueda de amigo. En el segundo el sistema realizará una búsqueda en la base de datos de todos aquellos escenarios predeterminados existentes que también tengan su correspondiente resolución realizada por los superusuarios. De todos estos escenarios que cumplan ambas condiciones será elegido uno al azar y será el que deberá resolver el alumno para luego enfrentarse en la competición virtual a la solución predeterminada que representa la máquina (Figura 7.5).

En este caso el botón de salir devolverá al usuario a la pantalla de inicio de sesión.

Figura 7.4: Captura del panel de conexión Bluetooth exitosa

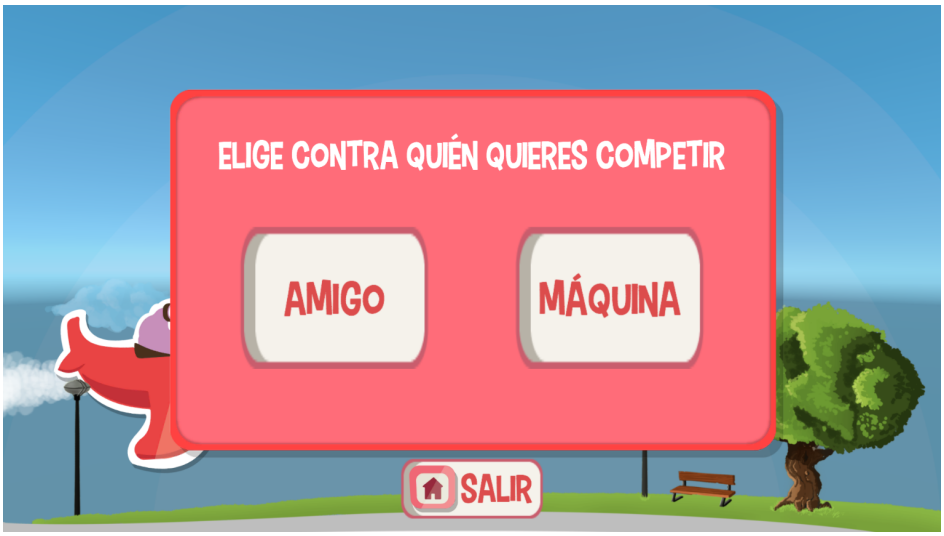


Figura 7.5: Captura del panel de elección de contrincante para simulación virtual

7.8.3 Pantalla de búsqueda de salas para el alumno

Cuando un usuario catalogado como alumno decida realizar la simulación junto al robot real y haya establecido la conexión, será llevado a esta pantalla. En ella tendrá a su disposición dos campos de introducción de texto mediante teclado (Figura 7.6).

En el primero de ellos deberá introducir el nombre de la sala a la que desea acceder, automáticamente la herramienta confirmará si esa sala existe en la base de datos y se le informará al usuario mediante un texto la existencia o no de la sala introducida.

El segundo campo de entrada de texto está destinado a la introducción de la contraseña de la sala. Este campo permanece deshabilitado hasta que la herramienta verifica que el nombre de la sala introducido por el alumno coincide con alguna de las salas existentes en la base de datos.

Cuando el alumno pulse en el botón de entrar en la sala, con los parámetros previamente introducidos, el sistema descargará el fichero de la base de datos correspondiente a la sala solicitada y comprobará si la contraseña introducida coincide con la especificada en ese archivo. En el caso de que no exista coincidencia el usuario será notificado mediante un texto invitándole a introducir de nuevo la contraseña. Si las contraseñas coinciden el usuario accederá a la sala y se mantendrá en ella a la espera de recibir órdenes por parte del moderador.

La información necesaria para acceder a una determinada sala (nombre y contraseña) deberá ser previamente transmitida al alumno por parte del moderador, el cual las conocerá porque será el creador y dueño de la sala.

En el momento de entrada a la sala si es la primera vez que accede se creará un fichero en la base de datos único para el usuario. En él se inicializarán todas las variables necesarias para el correcto funcionamiento de la comunicación y sincronización entre el alumno y el moderador.

Esta pantalla, al igual que la mayoría en la herramienta, dispone de un gran botón para salir, que en este caso hará retroceder a la pantalla de inicio de sesión.

Figura 7.6: Captura de la pantalla de búsqueda de salas para el alumno

7.8.4 Pantalla de búsqueda de salas para el moderador

A esta pantalla accederán- una vez iniciada la sesió,- los usuarios catalogados como moderadores. En ella, a diferencia de en la pantalla de búsqueda de salas paro los alumnos, el moderador se encontrará con una lista desplegable y un campo de introducción de texto (Figura 7.7).

En la lista desplegable se mostrarán las salas creadas por el moderador que está usando la herramienta en ese momento. Esta acción se puede realizar gracias a la estructura de ficheros creada en la base de datos y al método del complemento "Easy Save" que permite la descarga de

los nombres de todos los ficheros contenidos en la base de datos. De esta forma, se realizará la lectura de todos los nombres y mediante la separación de cadenas de caracteres a partir de un delimitador se filtran los nombres escogiendo solo aquellos que contienen el nombre del usuario activo y el identificador de que ese fichero denota a una sala. De esta forma el usuario podrá seleccionar de la lista una de las salas propia a la que quiera acceder.

Una vez seleccionada la sala, deberá introducir en el segundo campo de entrada de texto la contraseña de la sala, la cual conocerá pues es el mismo el que ha creado previamente la sala, dándole nombre y contraseña. Pulsando en el botón de entrar al igual que en la pantalla anterior se desencadenará el proceso de descarga y lectura del fichero de sala para realizar la comprobación de coincidencia de contraseñas. El moderador será informado en todo momento mediante un texto de si la contraseña es correcta o no.

En el caso de que el moderador aún no disponga de ninguna sala creada por él o simplemente desee crear una nueva, en esta misma pantalla se le da la posibilidad de registrar una nueva sala. Para ello deberá pulsar en un texto que le indica a ello, esto gesto hará superponerse un panel similar al de registro de un nuevo usuario, pero en este caso será para registrar una nueva sala.

Esto generará un fichero asociado al moderador que cree la sala inicializándose todas las variables de identificación de esta (Figura 7.8).

Un gran botón de salir permitirá en cualquier momento volver a la pantalla de inicio de sesión.

Figura 7.7: Captura de la pantalla de búsqueda de salas para el moderador

Figura 7.8: Captura del panel de registro de una nueva sala

7.8.5 Pantalla de espera en sala para el alumno

En el momento en el que el alumno acceda a una determinada sala, este permanecerá esperando en esta pantalla. En ella aparecerá una animación con unas luces que se irán encendiendo la siguiente y apagando la anterior síncronamente de forma que se dé la sensación de que la luz va avanzando a lo largo de las diferentes bombillas. Este efecto será acompañado por un texto que le avisará al usuario de que permanece a la espera de que el moderador de la orden de enviarles el escenario que quiere que sus alumnos resuelvan (Figura 7.9).

Por tanto, en esta pantalla la única interacción que puede tener el alumno es la de pulsar el botón de salir para volver a la pantalla de búsqueda de salas. En cambio, aunque el alumno no tenga muchas opciones en esta interfaz el sistema por dentro está constantemente ejecutando diversas rutinas y funciones. Estas son destinadas a la comprobación con un periodo de 2 segundos si una determinada variable en el fichero de sala creado por el moderador cambia indicando que ya tiene preparado el escenario a resolver y cargado en la sala. Los alumnos tienen permiso a acceder a este fichero pues en el momento de entrar en la sala se realiza la correspondiente transferencia de identificadores de ficheros a compartir durante el actual uso de la herramienta.

En estas rutinas también se actualiza constantemente una variable llamada “ultimaActualizacion” donde se almacena el valor de la fecha, hora, minutos y segundos actuales. Esta variable será de gran utilidad para el moderador y el correcto funcionamiento de la pantalla de control de la sala por parte del moderador.

Figura 7.9: Captura de la pantalla de espera en sala para el alumno

7.8.6 Pantalla de control de la sala por parte del moderador

Esta es una de las pantallas con mayor carga de programación e interfaz desarrolladas, a ella se llegará cuando el usuario catalogado como moderador acceda a una de sus salas propias. Nada más entrar el usuario verá un panel con tres botones, un texto y una zona sombreada (ver Figura 7.10).

Figura 7.10: Captura de la pantalla de control de la sala por parte del moderador con alumnos en espera

La zona sombreada será utilizada para ver información acerca de los alumnos que están actualmente dentro de la sala. Esta información se consigue gracias a la variable "ultimaActualizacion" y una serie de rutinas que se van ejecutando periódicamente. Además de comprobar que alumnos

están dentro de la sala el sistema también informa al moderador de en qué pantalla se encuentran esos alumnos condicionando las acciones que pueda tomar el moderador a esos estados de sincronismo. Con todo ello el proceso llevado a cabo para la recabación de toda esta información es el siguiente:

1. Descarga desde la base de datos de todos los nombres de fichero y filtrado de aquellos que contengan la cadena "ensala" y la cadena coincidente con el nombre de la sala en el que el moderador está actualmente. Estos nombres de usuarios que alguna vez accedieron a esa sala y ficheros correspondientes son almacenados en dos listas separadas.
2. Llamada a una función que recibe como parámetro las dos listas anteriores y las va recorriendo elemento a elemento. De esta información el sistema descarga el fichero correspondiente a cada uno de los elementos y lee la variable "ultimaActualizacion".
3. El valor de esa variable es comparado con el de la fecha en ese momento extraída del dispositivo en uso. Si la diferencia es menor a un minuto se entiende que ese alumno está activo actualmente en la sala, y por tanto su nombre aparecerá en la zona sombreada.
4. Adicionalmente, al moderador se le notificará de en qué pantalla se encuentra cada uno de esos alumnos activos en la sala. Esto se consigue, una vez filtrados los alumnos actualmente activos, leyendo una variable de sus ficheros asociados a la pertenencia en la sala donde la aplicación del alumno irá actualizando esta en función de en la pantalla en la que se encuentren.
5. Se generará un nuevo campo de texto a continuación del que muestra el nombre de usuario del alumno activo donde se mostrará la pantalla en la que este se encuentra.

La información que se podrá visualizar en este campo de texto asociado a la pantalla en la que se encuentra el alumno será:

- El alumno se encuentra "en espera" hasta que el moderador envíe el escenario a todos los alumnos activos.
- El alumno se encuentra "en mapa", esto le indicará al moderador que el alumno está resolviendo el escenario planteado, el cual como se detallará en su pantalla correspondiente muestra una interfaz de un mapa con una línea de tiempos (Figura 7.11).
- Tiempo que el alumno ha necesitado para dar una solución al escenario. Cuando el alumno envíe la solución al escenario planteado el moderador podrá saber cuánto tiempo en minutos ha necesitado para su resolución (Figura 7.12).
- El alumno se encuentra "en simulación", indicará al moderador que ese alumno ya ha comenzado su simulación a partir de la resolución que este haya dado al escenario planteado. Este estado permitirá saber si alguno de los alumnos ha tenido algún problema de comunicaciones pues como se detallará a continuación la orden de comenzar la simulación es dada por el moderador a todos los alumnos a la vez.

A parte de la visualización de los estados de los alumnos, la herramienta también incorpora otra comprobación a modo de seguridad para asegurar y facilitar la sincronización entre todos los alumnos y el moderador. Esta es que el sistema cerrará la entrada a la sala de nuevos alumnos en el momento que el moderador decida enviarles el escenario propuesto a todos los que se

Figura 7.11: Captura de la pantalla de control de la sala por parte del moderador con alumnos resolviendo el escenario

Figura 7.12: Captura de la pantalla de control de la sala por parte del moderador con alumnos tras haber resuelto el escenario

encuentran actualmente dentro de la sala. Desde este momento el sistema comprobará, aparte de individualmente, el estado en la aplicación de los alumnos también el estado en conjunto de todos los que se encuentren en la sala. De esta forma, acciones como inicializar la simulación síncrona en todos los dispositivos de los alumnos y por tanto el movimiento del robot solo estará disponible para el moderador cuando el sistema verifique que todos los alumnos han resuelto el escenario y están en disposición de comenzar la simulación. Tampoco se le permitirá al moderador el envío de un nuevo escenario a resolver hasta que el sistema compruebe que todos los alumnos han finalizado la simulación y han vuelto a la sala de espera.

Por otro lado, como se ha comentado con anterioridad, en esta pantalla existen tres botones más y un texto informativo cuyas funciones se listan a continuación:

- Botón de actualizar: permite actualizar en ese momento la comprobación de alumnos activos en la sala y el estado en la aplicación en el que se encuentran. Por defecto el sistema realiza una actualización de los datos cada 8 segundos.
- Botón de crear escenario: traslada al moderador a la pantalla de creación de escenarios que se detallará en su apartado correspondiente (Apartado 7.8.7).
- Botón de enviar escenario: será el encargado de dar la orden a todos los alumnos activos en la sala de acceder a la pantalla de resolución del escenario habiéndoles cargado este previamente en la sala.
- Texto informativo: en él se mostrará en el caso de que lo haya, el nombre del escenario actualmente cargado en la sala. Este será el escenario que recibirán los alumnos cuando el moderador pulse en el botón de enviar escenario.

Adicionalmente a todo lo anterior esta pantalla también incluye acceso a un panel de informes de cómo ha transcurrido la simulación y han resuelto el escenario los alumnos.

Panel de informe

A este panel se accederá mediante un pequeño botón donde se identifica una letra "R" (*Report*) que se encuentra junto a cada uno de los nombres de los alumnos activos. Pulsando sobre él aparecerá un panel con una gran línea de tiempos donde se ubicarán las horas del día empezando desde las 7:00 h hasta las 6:30 h en intervalos de 30 minutos. El objetivo de este panel es el de dar al moderador una visión rápida de que elementos de terapia y en qué momento los han utilizado los alumnos para conseguir que la simulación permanezca el mayor porcentaje de tiempo posible en normoglucemia.

La información mostrada en la escala de tiempos sigue un formato o codificación condensada para poder mostrar toda la información en una sola pantalla. Esta será condensada en cuadros compuestos por tres campos tal y como muestra en la Figura 7.13.

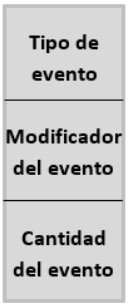


Figura 7.13: Formato de la información en el panel de report

En el campo tipo de evento podrán mostrarse los siguientes elementos (Tabla 7.11):

Tabla 7.11: Elementos posibles en el campo evento

Símbolo	**Evento**
I	Evento de una ingesta
E	Evento de ejercicio
B	Evento de administración de un bolo de insulina
S	Evento de administración de un snack
G	Evento de lectura del glucómetro

En el campo de modificador de evento solo se mostrará información de aquellos eventos que necesiten información adicional para su definición (ver Tabla 7.12):

Tabla 7.12: Elementos posibles en el campo modificadores

Evento	**Modificadores**		
Ingesta	D → Desayuno	Co → Comida	Ce → Cena
Ejercicio	A → Intensidad alta	M → Intensidad media	B → Intensidad baja
Bolo de insulina	No requiere de modificadores		
Snack	No requiere de modificadores		
Glucómetro	Modificador en función de la simulación		

En el tercer campo cantidad del evento se mostrará la cantidad en raciones asociada a ese evento.

El evento glucómetro presenta un comportamiento particular pues su valor dependerá de cómo transcurra la simulación y que acciones correctoras tome el alumno. Es decir, en el panel informe si en una determinada franja horaria se ha producido un evento de medición con glucómetro este se mostrará de forma condensada de la siguiente forma.

- Campo tipo de evento: se mostrará el valor de la glucosa en sangre en el momento de la simulación en el que se realice la medición con el glucómetro.
- Campo modificador del evento: en él se mostrará una "B" o una "S" en función de la medida correctora que tome el alumno cuando se realice la medición. Si no se tomara ninguna terapia correctora este campo aparecería sin rellenar.
- Campo cantidad del evento: este mostrará la cantidad, en caso de aplicación de alguna acción correctora, de las raciones aplicadas de bolo de insulina o snack.

Otro aspecto importante y de interés para el moderador es saber si un determinado evento proviene del escenario planteado por el mismo, de la resolución dada por el alumno o de la aplicación de alguna medida correctora durante el transcurso de la simulación. Para diferenciar estos momentos diferentes se ha establecido un código de color, aplicado en el cuadro de información, el cual se detalla en la Tabla 7.13.

Estos cuadros de información se generarán automáticamente a partir de la información recibida por parte del alumno una vez finalizada la simulación en su dispositivo. El sistema contempla el caso más extremo de máxima información simultánea, este se producirá cuando en una misma franja horaria se den tres eventos simultáneamente, pero provenientes de diferentes fuentes. Esto se solucionará con el apilamiento de los cuadros informativos hacia arriba siguiendo la jerarquía

Tabla 7.13: Colores identificativos en función de la procedencia del evento

Procedencia del evento	**Color identificativo**
Escenario planteado por el moderador	Rojo
Resolución del escenario por el alumno en la pantalla de resolución	Amarillo
Acción correctora del alumno durante la simulación	Verde

de en la parte más baja los eventos provenientes del escenario, un escalón hacia arriba los situados por el alumno en su resolución y arriba del todo las acciones correctoras durante la simulación. Este apilamiento no se dará si solo existe un evento en esa determinada franja horaria, ya que este se colocará directamente en la zona baja.

Como complemento a esta información el moderador también dispondrá de los porcentajes de tiempo que la simulación del alumno ha permanecido en cada uno de los tres estados glucémicos definidos en la herramienta.

Por último, para tener una interfaz conectada existen botones de volver para cerrar el panel de informe y un botón de salir, como en todas las pantallas, que devolverá al usuario a la pantalla de búsqueda de sala.

Toda la información detallada sobre el panel de informe es visible, su aplicación, en la Figura 7.14.

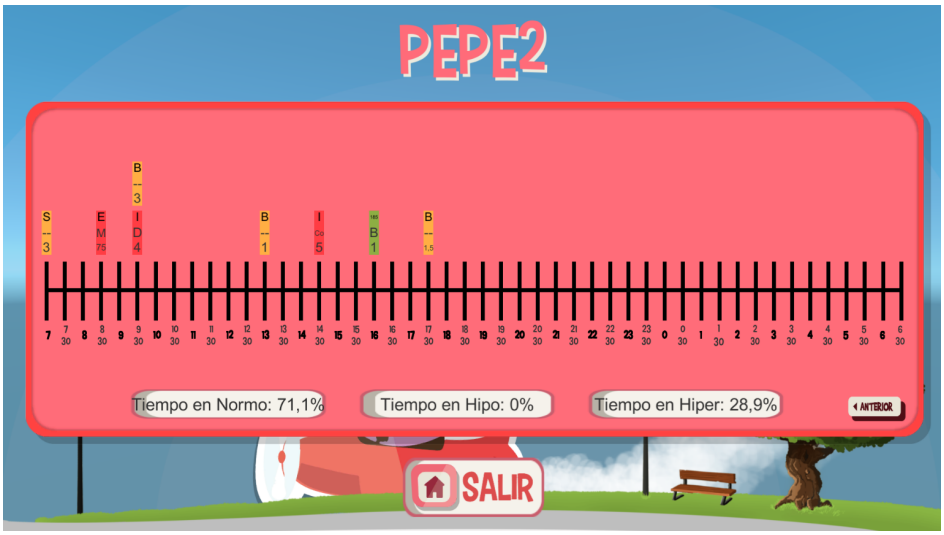


Figura 7.14: Captura del panel de report con información de los alumnos para el moderador

7.8.7 Pantalla de creación y modificación de escenarios

A esta pantalla se accederá con la pulsación del botón de creador de escenarios en el panel de control de la sala. En ella el moderador tendrá control absoluto de sus escenarios pudiendo visualizarlos, modificarlos, eliminarlos o crear nuevos.

La interfaz de esta pantalla se basa en un primer panel donde al usuario se le muestran dos listas desplegables, en la primera de ellas aparecerán los nombres de todos los escenarios creados por él, en cambio en la segunda se mostrarán los nombres de los escenarios predeterminados. Si el moderador no elije ninguno de los escenarios mostrados solo se le dará la opción de crear uno nuevo, en el caso de que este seleccione uno de los escenarios predeterminados se le dará la posibilidad de visualizarlo, y por último si este marca uno de sus escenarios propios se habilitarán las opciones de modificar sus campos o eliminarlo de la base de datos. Todas estas opciones o permisos se manejarán a través de la habilitación o deshabilitación de los botones encargados de realizar estas acciones (Figura 7.15).

Figura 7.15: Captura de la pantalla de creación y modificación de escenarios en el momento de la elección del escenario

En el caso de que el moderador decida modificar alguno de sus escenarios se le mostrará un panel en el que en primera instancia podrá ver un listado de los eventos y características que componen el escenario y también una serie de listas desplegables con las que poder añadir eventos. Los eventos del escenario, tanto los ya incorporados como los nuevos, son ordenados automáticamente en función de la franja horaria en la que van a tener lugar. Con esto se busca que el moderador tenga una visión más clara de cómo quedarían distribuidos los eventos a lo largo del día (Figura 7.16).

En cuanto a la creación de nuevos eventos, como se ha comentado anteriormente, se ponen a disposición del moderador una serie de listas desplegables para que este pueda ir confeccionando los eventos a su criterio, siendo estas de izquierda a derecha:

- Tipo de evento (Ingesta, Ejercicio, Bolo, Snack, Glucómetro)

- Intensidad del ejercicio (Alta, Media, Baja)
- Tipo de ingesta (Desayuno, Comida, Cena)
- Cantidad del evento
- Hora del evento

Figura 7.16: Captura de la pantalla de creación y modificación de escenarios en el momento de la visualización del escenario

Estas listas desplegables son dinámicas, es decir se irán activando en función de la opción marcada en la lista anterior. Por defecto todas las listas estarán deshabilitadas excepto la de tipo de evento, si por ejemplo se selecciona en ella ingesta, se habilitarán solo las listas que sean necesarias para la correcta definición de este tipo de evento (tipo de ingesta, cantidad del evento y hora del evento).

En adición a esta primera medida de seguridad para evitar errores por parte del moderador, la herramienta tiene implementada otra medida para evitar la introducción en el escenario de eventos mal definidos. Esta consiste en que si el usuario pulsa el botón de añadir evento sin haber seleccionado toda la información necesaria para la correcta definición del evento la aplicación no le permitirá añadirlo. Además, se le notificará mediante un texto de que campos le falta por seleccionar para definir correctamente el evento.

Como se ha anticipado anteriormente, en la pantalla creador de escenarios aparte de visualizar escenarios ya creados también se permite su modificación. Esta modificación se puede dar o bien por la adición de un nuevo evento o por la eliminación de uno de los ya presentes mediante el botón asociado a cada uno de ellos. Como medida de seguridad para evitar modificaciones indeseadas el sistema cuestionará al usuario la primera vez que realice alguna de las dos acciones anteriores sobre si desea modificar el escenario o bien crear una copia de este con otro nombre. Si se elige la primera opción se podrá llevar a cabo la modificación y el nuevo escenario será actualizado en la base de datos. En el caso de elegir la opción de crear una copia se le pedirá al usuario que introduzca un nuevo nombre de escenario, el cual será comprobado que no exista ningún otro con ese nombre, y se creará en la base de datos un escenario nuevo con los mismos

eventos que el original. Esta última alternativa permitirá al moderador crear de una forma más ágil nuevos escenarios al no tener que partir de cero.

Por último, esta pantalla también incorpora un botón de salir que hace retroceder al usuario a la pantalla de control de la sala. También existe un botón que permite la carga del escenario en la sala, es decir el usuario podrá cargar en la sala el escenario que seleccione, modifique o cree para poder ser enviado a los alumnos desde la pantalla de panel de control de la sala.

7.8.8 Pantalla de resolución de escenarios

A esta pantalla llegarán los alumnos en el momento que el moderador les envíe el escenario que deben resolver. Esta es una modificación de una de las pantallas disponibles en el desarrollo previo, a la cual se le han realizado diversas modificaciones y ampliaciones para ajustarla al funcionamiento requerido en la nueva herramienta. De este modo, esta pantalla mantendrá todas las funciones de la original con las siguientes modificaciones:

- Colocación automática de eventos en función del escenario cargado: la herramienta leerá el nombre del escenario cargado en la sala y lo descargará de la base de datos para proceder a colocar los iconos, con sus características marcadas por el evento guardado, en su lugar correspondiente. Estos eventos no podrán ser modificados por el alumno, permitiéndoles solo acceder a ellos para ver sus características de tipo y cantidad del evento. Estos iconos serán coloreados con una capa de color rojo superpuesta indicando al alumno que son eventos colocados ahí por el moderador y no pueden ser eliminados ni modificados (Figura 7.17).
- Desactivación de todos los iconos de los eventos excepto los relativos a las terapias que serán los que el alumno deberá utilizar para la resolución del escenario.
- Actualización con cierto periodo de tiempo de la variable "ultimaActualización": con ella se le indicará al moderador que ese usuario sigue activo en la sala.
- Medida del tiempo de permanencia en esta pantalla: se realizará una medida de la fecha, con precisión de segundos, disponible en el dispositivo a la entrada a esta pantalla y otra en el momento en el que se pulse el botón de "Despegar" que indicará que se ha terminado de resolver el escenario y se está listo para la simulación. La diferencia de tiempo entre estas dos variables será cargada en el fichero del alumno de pertenencia a la sala para poder ser visualizada por el moderador.
- Actualización de la variable de estancia en esta pantalla: el objetivo de esta es que el moderador pueda ser notificado de ello en su panel de control.
- Almacenamiento y envío de los eventos colocados por el alumno para la resolución del usuario para poder ser mostrados al moderador en su panel de informe.

En esta pantalla el alumno podrá tomar todo el tiempo que precise oportuno para la resolución del escenario pues gracias a los métodos de sincronización desarrollados los demás compañeros deberán esperar a que todos hayan enviado sus resultados para poder comenzar la simulación.

Figura 7.17: Captura de la pantalla de resolución de escenarios

7.8.9 Pantalla de visualización de la simulación

Esta pantalla es la consecución de la anterior y también parte de una de las pantallas presentes en el desarrollo previo. En ella de nuevo se guardan todas las características de la original, pero se incorporan una serie de modificaciones para lograr el funcionamiento requerido en esta nueva herramienta. Estas modificaciones y ampliaciones son (Figura 7.18):

- Sistema de sincronización de inicio de simulación: tanto el robot real como la simulación se mantendrán detenidos al entrar en esta pantalla, pero el sistema internamente estará constantemente comprobando un cambio en el fichero de la sala para comenzar. Se ha programado de esta forma para conseguir la máxima sincronización en el comienzo de la competición para todos los alumnos. Ya que en vez de descargar el fichero correspondiente y leer el estado de una variable que indique el comienzo de la carrera, se comprueba simplemente el cambio en la fecha de última actualización del fichero correspondiente a la sala. Este método es mucho más rápido e independiente del rendimiento del dispositivo que esté ejecutando la simulación. Además, esto se hace posible gracias a que la herramienta está programada para que, llegados a este punto, la única alternativa que tiene el moderador para realizar un cambio en este fichero y en toda la aplicación es pulsar un gran botón de iniciar carrera.
- Envío de ordenes al robot en función del estado glucémico: se ha creado una función que a partir de los valores de glucosa en sangre en un determinado momento de la simulación evalúa en qué estado glucémico se encuentra. Dependiendo del estado glucémico en ese momento la aplicación enviará al robot real la orden de simular el comportamiento asociado a los síntomas típicos de ese estado.
- Actualización de la fecha "ultimaActualización" con un cierto periodo de tiempo para mostrar al moderador que el usuario sigue activo.

- Envío de orden de detención al robot físico durante la realización de una medida con el glucómetro: como se ha comentado anteriormente con el objetivo de que este proceso no suponga una ventaja competitiva en la carrera el robot deberá detenerse cuando la simulación atraviese un evento de tipo glucómetro.
- Almacenamiento y envío de la información de interés para el moderador: la herramienta almacena los datos de interés durante las mediciones con glucómetro (nivel de glucosa actual y tipo de acción y cantidad aplicada). Además de esta información al final de la simulación se envían los porcentajes de tiempo que ha pasado la simulación en cada uno de los tres estados glucémicos definidos.
- Almacenamiento del estado glucémico en cada instante para simulación con robot virtual: se ha programado la función de que cuando el sistema detecte, mediante la lectura de una variable del fichero propio de cada usuario, que el usuario ha decidido realizar la simulación con robot virtual, se almacenarán en un vector en cada paso de la simulación que estado glucémico se tenía. Esta información será usada para modelar el comportamiento del robot virtualizado en su pantalla correspondiente.
- Modo velocidad máxima: a este modo se accederá cuando se elija el itinerario de simulación con robot virtual, siendo su cometido el de realizar la visualización de la simulación a una velocidad muy alta. Se ha programado así en lugar de ocultar este proceso al usuario con el objetivo de que este perciba que lo que está ocurriendo en la modalidad con robot físico o virtual es la misma simulación.

A nivel visual en esta pantalla el único cambio apreciable respecto al desarrollo previo de partido es la inclusión de una barra de gasolina que va consumiéndose a lo largo de la visualización de la simulación. De esta forma se le indica gráficamente al usuario que porcentaje de la simulación se ha completado en cada momento.

Figura 7.18: Captura de la pantalla de visualización de la simulación

Al finalizar esta pantalla en el modo simulación con robot real, saltará un panel que informará al alumno de los puntos conseguidos y de los porcentajes de tiempo en los que la simulación ha

estado atravesando cada uno de los estados glucémicos definidos. Será en este momento cuando el moderador será notificado en su panel de control de que ese alumno ha finalizado la simulación. El moderador no podrá volver a enviar otro escenario hasta que todos los alumnos activos en la sala hayan salido de esta pantalla y hayan vuelto a la sala de espera.

En cambio, si la simulación finaliza en el modo de simulación con robot virtual, la aplicación pasará directamente a la pantalla con el robot y el circuito virtualizado, permaneciendo a la espera de que su contrincante finalice también su simulación correspondiente.

7.8.10 Pantalla de búsqueda de amigo para simulación virtual

A ella se accederá cuando se escoja la opción de realizar la simulación con robot virtual contra un amigo. En esta pantalla se le presentará al usuario un panel de entrada de texto en el que deberá introducir el nombre de usuario del amigo con el que quiere conectar. Este mandará una especie de petición de conexión a la aplicación del amigo mediante la escritura en un fichero destinado a gestionar la información de la simulación virtual. Si el amigo introduce en su pantalla el nombre de la persona que quiere conectar con él, el sistema reconocerá que ambos usuarios desean conectarse entre sí (Figura 7.19).

Figura 7.19: Captura de la pantalla de búsqueda de amigo para simulación virtual

Una vez ambas aplicaciones han detectado la conexión se debe elegir un escenario común que ambos usuarios deben resolver. Este escenario será elegido aleatoriamente entre todos los escenarios almacenados en la base de datos creados por cualquier moderador. El problema reside en que las aplicaciones de ambos usuarios seleccionarán un escenario aleatoriamente que no tienen por qué coincidir entre ellos. Esta situación no es viable pues no se estaría compitiendo en las mismas condiciones. Por ello, la solución tomada ha sido la de seleccionar a uno de los usuarios como moderador cuyo único privilegio será que el escenario que ambos deberán resolver será el que su sistema aleatoriamente haya escogido. La selección de este moderador se escoge seleccionando aquel cuya suma de los caracteres en formato ASCII que conforman su nombre de usuario dividida por el número de estos sea mayor. Este método casi siempre elegirá a uno de los dos usuarios

pues las probabilidades de que dos nombres de usuario en forma ASCII y con la aplicación de esa regla devuelvan el mismo valor es muy pequeña. En el caso de que esta situación remota se diese el sistema elegiría el escenario de aquel que más tarde lo haya seleccionado aleatoriamente pues será el último escrito en el fichero.

7.8.11 Pantalla de simulación con robot virtual

En ella se realiza la virtualización del circuito y del robot para que simulen un comportamiento similar al que ocurriría en la vida real, siendo percibido por el usuario como un entrenamiento útil para cuando deba enfrentarse a esta situación en el taller educativo.

Debido a la complejidad y extensión de esta pantalla, así como los desarrollos y procesos previos necesarios para su funcionamiento se ha dedicado un apartado exclusivamente para tratar la modelización, virtualización y simulación en pantalla de la competición entre ambos robots (ver Apartado 8.5).

7.8.12 Pantalla de control de superusuario

Esta es una pantalla donde este perfil de usuario tendrá acceso a las siguientes estadísticas sobre el uso de la aplicación:

- Número de alumnos activos en talleres educativos en las últimas 24 horas, última semana y último mes.
- Número de alumnos activos entrenando en las últimas 24 horas, última semana y último mes.
- Número de moderadores activos en las últimas 24 horas, última semana y último mes.
- Tiempo medio que pasan los alumnos resolviendo los escenarios mientras entrenan con el robot virtual.
- Tiempo medio que utilizan los alumnos para resolver los escenarios cuando se encuentran en el taller.

Toda esta información es anonimizada para preservar la privacidad de los usuarios.

Para la recabación de las estadísticas de números de alumnos activos, se va a usar la lectura de la fecha de última actualización de sus ficheros correspondientes en la base de datos. De esta forma comparando esta fecha con la actual procedente del dispositivo se puede saber cuánto tiempo hace que se ha conectado cada usuario. Con este dato ya es trivial saber cuántos usuarios han estado activos hace menos de un día, una semana o un mes.

El manejo de las estadísticas de tiempos medios se basará en una serie de variables que irán actualizando todos los usuarios en un fichero de estadísticas de uso compartido, a las cuales solo tendrán acceso de visualización los superusuarios. Se ha programado de esta forma, a pesar de que puede ocasionar problemas como que dos usuarios estén intentando actualizar el fichero compartido a la vez y solo uno finalmente lo pueda realizar. La situación ideal sería que cada usuario tuviera una variable o vector en la base de datos donde fuera actualizando los tiempos que pasa usando la aplicación, pero debido al funcionamiento basado en ficheros del *asset* de comunicaciones utilizado, esta solución se presenta imposible para una herramienta con un

volumen de usuarios considerable. Esto es debido a que para saber el tiempo medio que pasan los alumnos en la aplicación habría que descargar todos los ficheros asociados a los alumnos a la memoria local del dispositivo, abrirlos, y leer la variable correspondiente. El mismo proceso habría que realizarlo para los moderadores. Esto desembocaría en la descarga de una gran cantidad de ficheros en la memoria local del dispositivo del superusuario, así como unos tiempos de computación inabordables si existe una cantidad de usuarios en la herramienta apreciable.

Además de la visualización de estas estadísticas de uso, esta pantalla permitirá a sus usuarios entrar en las pantallas de registro de nuevos moderadores y de creación de escenarios predeterminados o continuar como moderadores haciendo un uso con los mismos privilegios que tienen estos últimos del resto de la herramienta (Figura 7.20).

Figura 7.20: Captura de la pantalla de control de superusuario

7.8.13 Pantalla de registro de nuevos moderadores

A esta pantalla solo accederán los usuarios catalogados como superusuarios. En ella como se ha venido anticipando, estos podrán visualizar que usuarios son actualmente moderadores. Además, tendrán control absoluto para añadirles o eliminarles a los usuarios la capacidad de ser moderadores (Figura 7.21).

La interfaz para la realización de estas acciones será la siguiente:

- Campo de entrada de texto donde introducir el nombre del usuario que se desea dar capacidad de ser moderador.
- Botón para añadir a ese usuario al fichero que almacena todos los moderadores dados de alta en la herramienta.
- Texto que informa de si un usuario que se quiere hacer moderador ya lo es impidiendo añadirlo por duplicado. Este texto también informará de si se está intentando añadir a un usuario cuando el campo de texto donde introducir su nombre está vacío.

- Lista desplegable donde visualizar todos los nombres de usuario de aquellos que son moderadores. También permite seleccionarlos para proceder a la eliminación de su capacidad de ser moderador.
- Botón para eliminar del fichero en la base de datos a aquel usuario seleccionado en la lista desplegable.

Figura 7.21: Captura de la pantalla de registro de nuevos moderadores

7.8.14 Pantalla de creación de escenarios predeterminados

Esta pantalla es una versión de la pantalla de creación de escenarios para los moderadores. Será de solo acceso para aquellas personas que usen la aplicación y tengan privilegios de superusuario.

En ella se mostrará una lista desplegable con los escenarios predeterminados alojados en la base de datos. Estos podrán ser visualizados, modificados y eliminados de la misma forma que en su versión análoga para moderadores (Figura 7.22).

Además de la creación de estos escenarios predeterminados, la herramienta instará a los superusuarios a la resolución de estos para poder así almacenar esta solución en la base de datos. Esta será utilizada para poder realizar la competición entre el alumno y la máquina en la versión virtual de la herramienta.

Cabe recordar que, aunque estos ficheros serán solo accesibles para su modificación por parte de los usuarios, los moderadores si tendrán acceso a ellos para visualizarlos, crear copias con modificaciones de ellos y usarlos con sus alumnos.

Figura 7.22: Captura de la Pantalla de creación de escenarios predeterminados

7.9 Itinerarios dentro de la aplicación en función del perfil de usuario

Al contener la herramienta desarrollada en este proyecto una aplicación que debe conjugar diferentes perfiles de usuario, se producen diferentes itinerarios o caminos a seguir dentro de ella. Por ello con el objetivo de clarificar las distintas opciones que tendrá cada usuario dentro de la aplicación se presentan en la Figura 7.23, Figura 7.24 y Figura 7.25 los mapas de decisión que pueden tomar cada uno de estos.

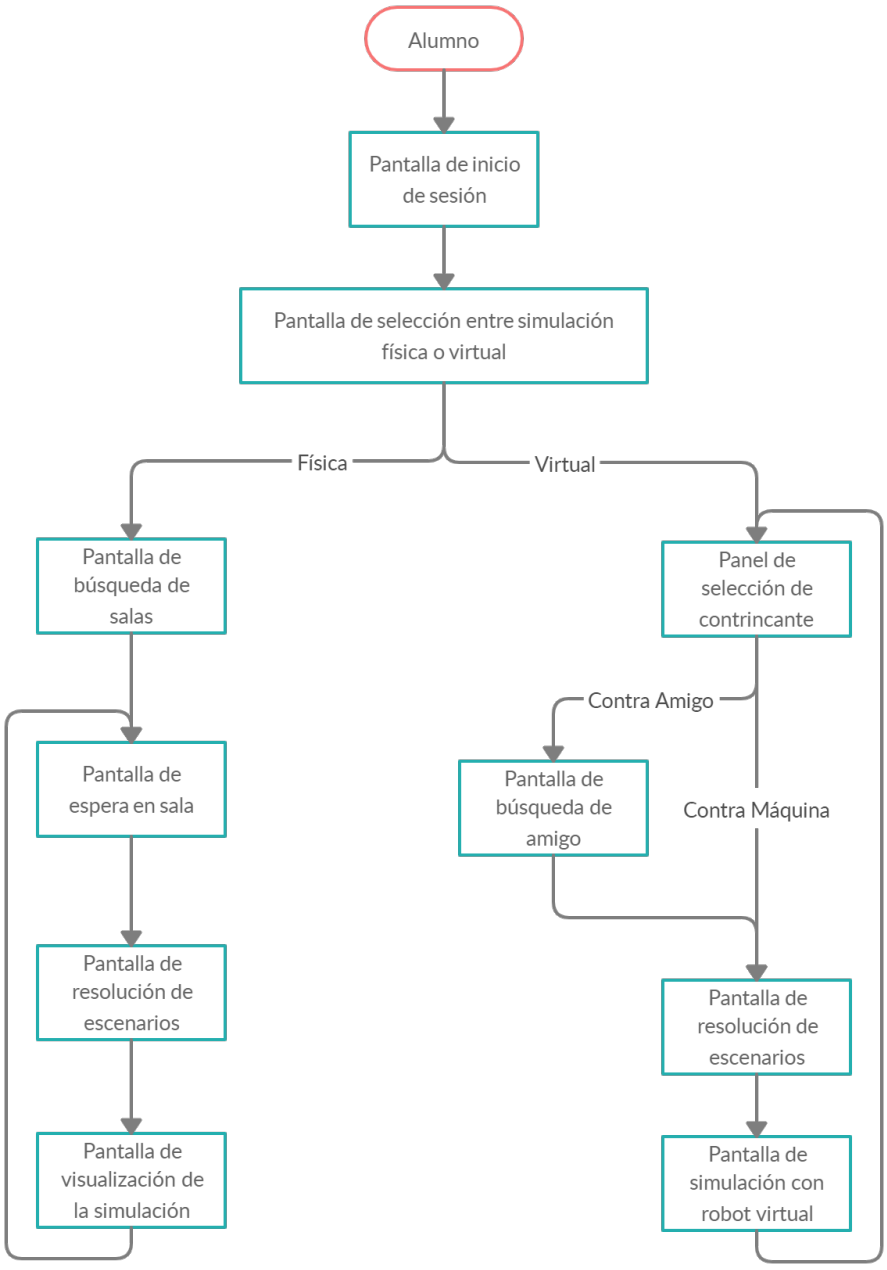


Figura 7.23: Itinerario dentro de la aplicación desde el punto de vista del alumno

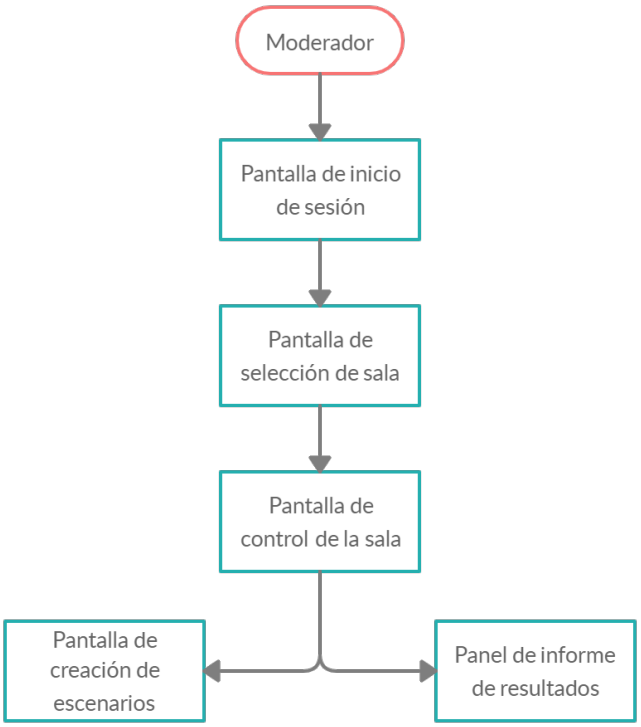


Figura 7.24: Itinerario dentro de la aplicación desde el punto de vista del moderador

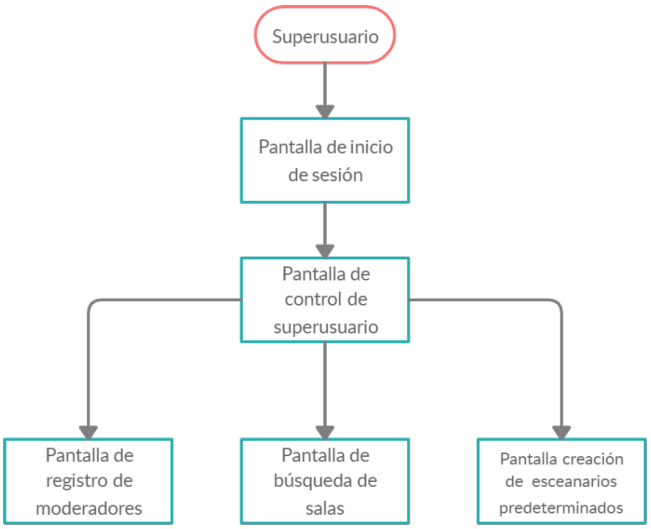


Figura 7.25: Itinerario dentro de la aplicación desde el punto de vista del superusuario

Capítulo 8

Virtualización del dispositivo robótico

«El mundo de la realidad tiene sus límites, pero el mundo de la imaginación es ilimitado» — Jean-Jacques Rousseau

8.1 Definición

Como se ha anticipado en los apartados anteriores la herramienta debía incorporar algún sistema que permitiera a los alumnos poder poner a prueba sus conocimientos y entrenar antes de acudir a los talleres de educación diabetológica. Para ello se decidió, con el objetivo de intentar imitar al máximo posible la experiencia que vivirían estos en los talleres, realizar la virtualización del dispositivo robótico para que se comportase de forma similar a como lo haría en la realidad. Esta virtualización se ha llevado a cabo en dos fases. La primera de ellas ha hecho uso del software Matlab para conocer de una forma prototipada como se comportaría el modelo cinemático del robot. Y en la segunda fase se hace uso del motor físico de Unity para conseguir un modelo del robot que tenga en cuenta también la dinámica del sistema y su interacción con el entorno, además de poder ser este integrado de forma adecuada con el resto de la herramienta desarrollada en Unity.

8.2 Dimensiones y características del robot mBot

En este apartado se detallan las características que definen al dispositivo robótico utilizado, las cuales serán usadas para el modelado de este tanto en Matlab como en Unity.

- Masa: 0.5 Kg
- Largo: 16.5 cm
- Ancho: 9 cm
- Alto: 7.5 cm
- Diámetro de la rueda: 6 cm
- Masa de la rueda: 30 g
- Separación entre ruedas motoras: 11.5 cm

- Distancia entre el punto medio del eje imaginario de unión de las ruedas y el sensor siguelíneas: 9.5 cm
- Separación entre pares emisor-receptor del sensor siguelíneas: 1.5 cm

8.3 Modelo cinemático directo del dispositivo robótico

En el Apartado 6.1.3 se avanzó que el robot físico integrado en esta herramienta (mBot) basa su movimiento en lo que se conoce como configuración diferencial. En ella existen dos ruedas traseras motoras y una delantera móvil que solo sirve de apoyo. Su movimiento se basa, por tanto, en la diferencia de velocidad entre ambas ruedas motoras.

Con todo ello las ecuaciones diferenciales que modelan su comportamiento cinemático en el plano son: [29]

$$\dot{x}(t) = v(t)\cos(t) \tag{8.1}$$

$$\dot{y}(t) = v(t)\sin(t) \tag{8.2}$$

$$\dot{\varphi}(t) = \omega(t) \tag{8.3}$$

φ será el ángulo que forma el vector perpendicular a ese eje imaginario con el eje horizontal (Eje X). Por otro lado v y ω son respectivamente la velocidad lineal y la velocidad angular de ese punto imaginario, estas son definidas por las siguientes expresiones:[29]

$$v = \frac{V_{derecha} + V_{izquierda}}{2} \tag{8.4}$$

$$\omega = \frac{V_{derecha} + V_{izquierda}}{d_{ruedas}} \tag{8.5}$$

En estas expresiones $V_{derecha}$ y $V_{izquierda}$ son las velocidades lineales de cada una de las ruedas motoras del robot. El término d_{ruedas} hace referencia a la separación entre ambas ruedas, es decir a la longitud de ese eje imaginario que las une.[29]

La expresión que relaciona la velocidad angular de la rueda con su velocidad lineal es la siguiente:[29]

$$V_{derecha} = R_{derecha} \cdot \omega_{derecha} \tag{8.6}$$

$$V_{izquierda} = R_{izquierda} \cdot \omega_{izquierda} \tag{8.7}$$

En este caso y en casi todos los dispositivos robóticos con esta configuración motora esta dimensión será igual para ambas ruedas.

Con la Ecuación 8.1, Ecuación 8.2 y Ecuación 8.3 se modela el comportamiento cinemático del robot para cada instante de tiempo, pero para poder ser estas implementadas en un ordenador y poder realizar un control sobre ellas es necesario su discretización. Esta será una buena aproximación si se evalúa en intervalos de tiempo pequeños. Con este concepto, la discretización de las ecuaciones anteriores será:[29]

$$x(t + \triangle t) = x(t) + v(t)\ \triangle t\ \cos(\varphi(t)) \tag{8.8}$$

$$y(t + \triangle t) = y(t) + v(t)\ \triangle t\ \sin(\varphi(t)) \tag{8.9}$$

$$\varphi(t + \triangle t) = \varphi(t) + \triangle t\, \omega(t) \tag{8.10}$$

8.4 Modelado en Matlab del comportamiento del robot

Con el objetivo de realizar un prototipo virtual que reproduzca el movimiento del robot real y su comportamiento de seguidor de líneas se ha utilizado Matlab para su implementación.

8.4.1 Matlab, el software de cálculo

Matlab (Matrix Laboratory) es un software de cálculo, análisis y representación de datos y creación de algoritmos propiedad de la compañía MathWorks. Ofrece un entorno de desarrollo integrado basado en su propio lenguaje (lenguaje M). Además, es muy versátil pues sus funciones son muy ampliables a través de la instalación de sus Toolboxes desarrolladas tanto por la propia compañía como por terceros. [30]

Este software es mayoritariamente utilizado en el ámbito académico e investigador, aunque también es usado en algunas empresas para el modelado y prototipado de sus sistemas y la realización de experimentos simulados sobre partes del sistema previos a su implantación.

8.4.2 Virtualización del circuito y del robot

Para realizar la representación gráfica del movimiento del robot se ha partido de un desarrollo realizado por Edison Sasig. Este incluye las funciones necesarias para la representación del plano sobre el que se moverá el modelo 3D del robot con configuración diferencial. Además, incorpora el trazado de la trayectoria que sigue el punto medio del eje de unión de ambas ruedas motoras. [31]

Este desarrollo ha sido complementado con las siguientes funcionalidades:

- Programación de las ecuaciones discretas de movimiento del robot detalladas en el Apartado 8.3.
- Creación de una trayectoria a seguir por el robot.
- Simulación del sensor siguelíneas.
- Control todo o nada a partir de la realimentación dada por la simulación del sensor siguelíneas.

Las ecuaciones han sido programadas, basándose en el desarrollo de Edison Sasig, en un bucle que se evaluará con un periodo de muestreo de 0.01 segundos. Este bucle será el encargado de establecer la posición y orientación del robot en cada periodo de muestreo.

Para probar el funcionamiento del control sobre el movimiento del robot se ha trazado una trayectoria a seguir por este. Está se compone de diversos segmentos y arcos con cambios de dirección entre ellos.

Por otro lado, el sensor siguelíneas presente en el robot ha sido simulado evaluando la posición del punto donde se encontraría el sensor respecto a la trayectoria de referencia. De esta forma, se va calculando a partir del punto medio del eje imaginario, la distancia entre este punto y el sensor en el robot real y la orientación en cada instante del robot, la posición que tendría ese sensor sobre el plano de simulación. Conocido ese punto el siguiente paso es saber si este se ha desviado más de la mitad del ancho de la línea y por tanto no estaría sobre ella. Este efecto se ha programado a partir del cálculo de la mínima distancia euclídea entre el punto que define el sensor y todos los puntos que definen la trayectoria. Esta distancia será perpendicular a la trayectoria y permitirá compararla con la mitad del ancho establecido para la línea para saber si el robot está sobre ella o no.

El último paso es el de saber si el robot se ha desviado de la línea por la derecha o por la izquierda de esta para poder aplicar la acción de control apropiada. Esta funcionalidad se ha implementado mediante el cálculo de la tangente inversa en el cuarto cuadrante (atan2 en Matlab) del punto que define el sensor y del punto de la trayectoria con el cual se ha obtenido la menor distancia euclídea entre ellos. Con su diferencia se obtendrá un signo que denotará si el sensor está por la derecha o por la izquierda de la línea. Este signo presenta el siguiente comportamiento (Figura 8.1) en función del cuadrante en el que se encuentre el robot:

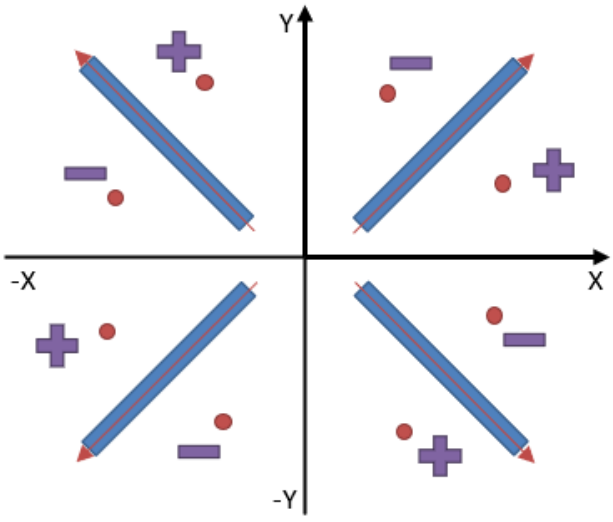


Figura 8.1: Signo en función del cuadrante y la posición

En la Figura 8.1 se representan los cuatro cuadrantes en los que se divide el plano, además en cada uno de ellos se ha representado en azul la línea que conformaría la trayectoria junto con una flecha roja que indica el sentido de movimiento del robot. Esto permitirá siempre, teniendo como base este sentido de movimiento, saber si el robot se ha desplazado de la trayectoria por la derecha o por la izquierda. Los signos posibles en cada cuadrante se representan por unos puntos fuera de la trayectoria junto a un signo positivo o negativo en función de si en ese cuadrante desplazarse de la trayectoria por la derecha o la izquierda genera con el cálculo anterior un signo positivo o negativo. Como se puede observar siempre que el sensor se salga de la línea por la derecha en el sentido del movimiento el signo del cálculo será positivo, y viceversa para cuando se sale por la izquierda.

Con el sensor siguelíneas modelado, permitiendo así saber si el sensor se ha desplazado de los límites de la trayectoria por la derecha o por la izquierda, se puede realizar el control del robot mediante el siguiente algoritmo equivalente al implementado en el robot real (Figura 8.2).

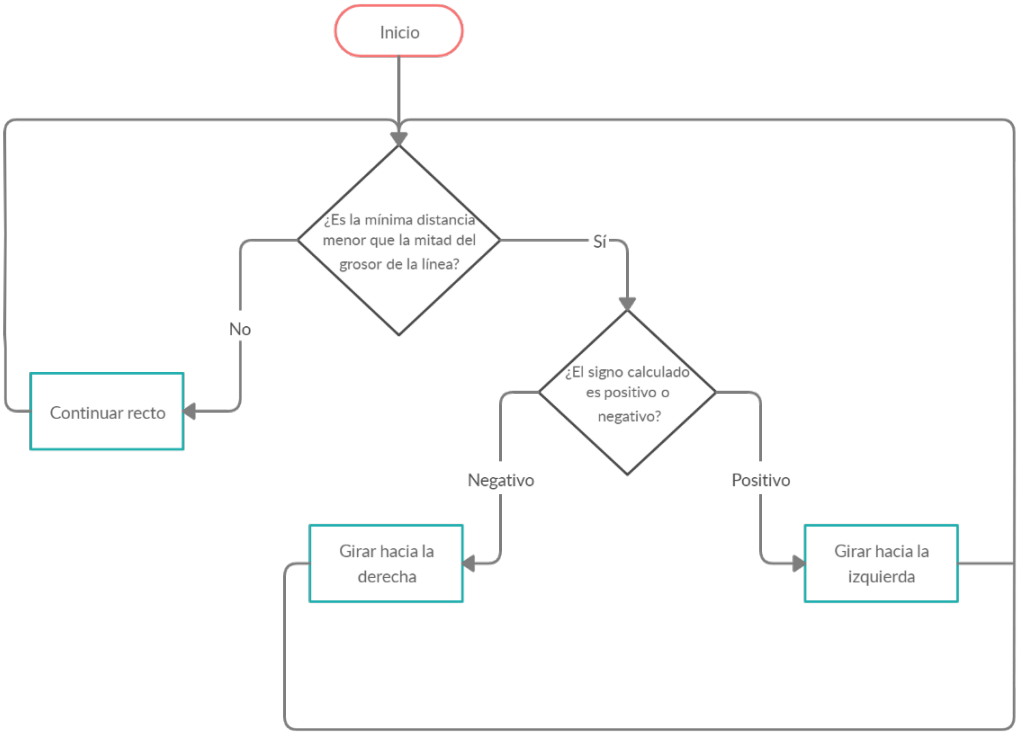


Figura 8.2: Funcionamiento del sensor siguelíneas modelado en Matlab

Una vez realizada la simulación del sensor siguelíneas, e implementado el control todo o nada que actúa sobre el modelo cinemático que rige el movimiento del robot se está en disposición de ejecutar la simulación y comprobar si el funcionamiento es el correcto. Para ello se le ha presentado al robot y al algoritmo de sensorización y control una trayectoria compleja compuesta por segmentos con cambios de dirección y curvas y se ha realizado la simulación (ver Figura 8.3).

En ella se puede observar como el algoritmo siguelíneas implementado es capaz de recorrer la trayectoria con bastante fidelidad, experimentando ciertas oscilaciones en los cambios de dirección siendo estás más pronunciadas cuanto más brusco es este cambio.

En la Figura 8.4 se muestran las acciones de control que el controlador suministra a las ruedas motoras en forma de velocidad angular. En ella se pueden apreciar perfectamente los tramos que se corresponden con una curva (en la zona media y final) ya que el control debe realizar muchos más cambios de dirección en esta geometría compleja que en una recta.

Una vez comprobado con este prototipo que es posible modelar con un buen resultado el comportamiento del robot en Matlab el siguiente paso es tratar de traspasar todo este conocimiento a Unity.

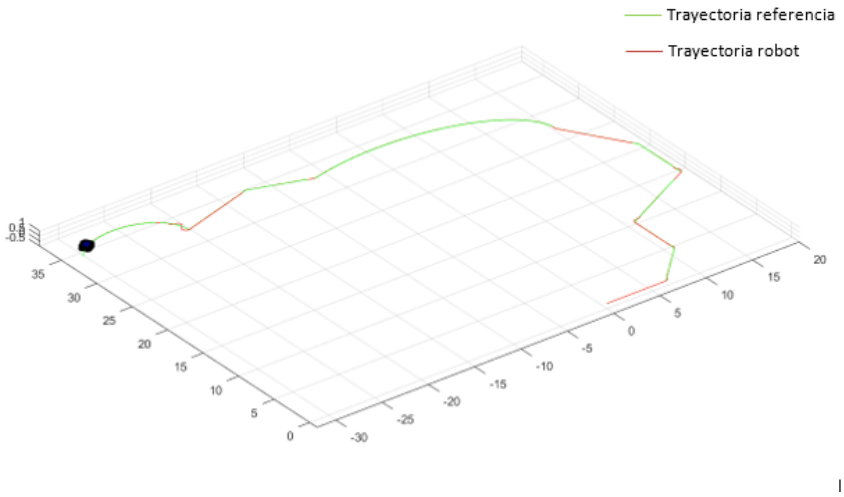


Figura 8.3: Resultado de la simulación del comportamiento del robot siguelíneas en Matlab

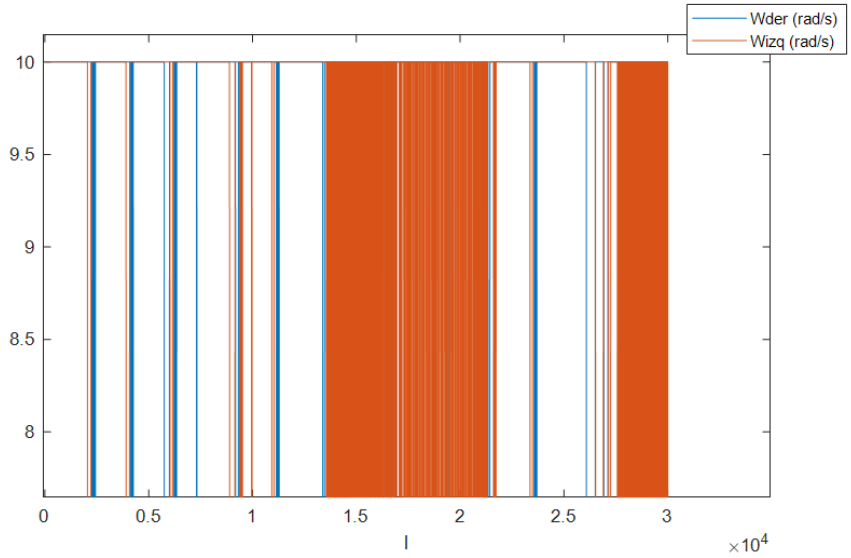


Figura 8.4: Acciones de control transmitidas a las ruedas motoras del robot siguelíneas en Matlab

8.5 Virtualización del robot en Unity

Con el objetivo de integrar esta funcionalidad en la herramienta en desarrollo, así como por tratar de modelar más fielmente el comportamiento del robot se va a emplear las funcionalidades que ofrece Unity. En este software como se ha comentado anteriormente se incluye el motor físico Nvidia PhysX el cual es de los más potentes para la creación de videojuegos y simulaciones. Para implementar esta virtualización del comportamiento siguelíneas se ha partido de un desarrollo previo realizado por Wiktor Rott.

8.5.1 Desarrollo previo de partida

En el proyecto creado por Wiktor Rott se modela un robot con configuración diferencial el cual debe seguir una trayectoria gracias a la implementación de un controlador de tipo PID y la simulación de un sensor siguelíneas con 7 pares emisor-receptor (Figura 8.5).[32]

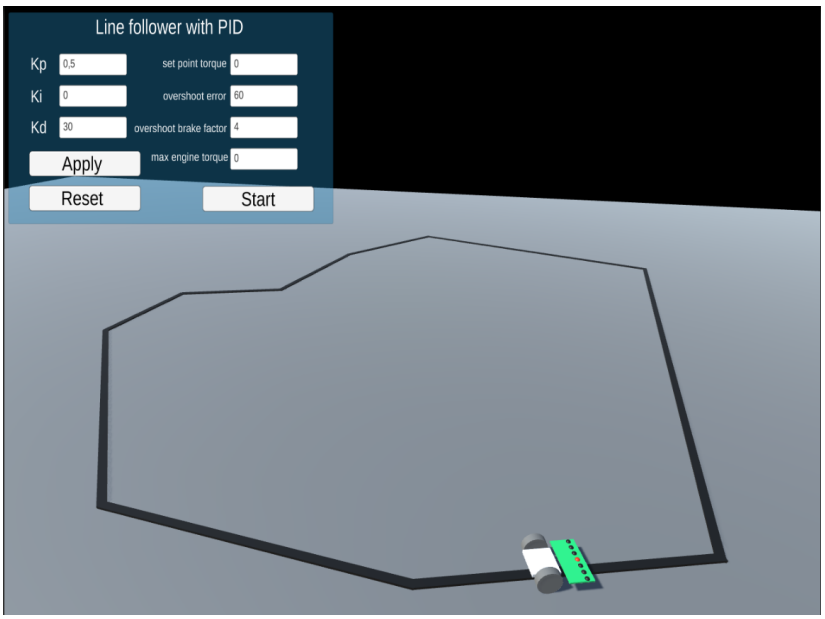


Figura 8.5: Captura del desarrollo previo de robot siguelíneas con control basado en PID

Este desarrollo se basa en la utilización del motor físico PhysX embebido en Unity. Para ello utiliza funcionalidades y conceptos que serán más detallados en el Apartado 8.5.3.

A pesar del buen funcionamiento de este proyecto, este presentaba una serie de limitaciones y características que lo hacían no ajustarse demasiado bien al sistema que se quería modelar en la herramienta. Estas son:

- Las dimensiones, peso, parámetros de suspensión, fricción no estaban personalizadas para el modelo de robot que se quiere virtualizar.
- El aspecto del robot no se parece en nada al del dispositivo robótico utilizado en la herramienta (mBot).
- El sensor siguelíneas utilizado es de 7 pares emisor-receptor, pero en el robot físico utilizado solo se dispone de dos pares.
- El control implementado en el desarrollo de partida es de tipo PID, en cambio en el robot real al no disponer de encoders no es posible realizar este tipo de control. Esto será una característica importante que habrá que modificar para conseguir un comportamiento lo más fidedigno posible a la realidad.
- En la herramienta que se ha desarrollado se pretende simular una competición entre mínimo dos robots. En el desarrollo de partida solo estaba realizada la implementación de uno de los robots.

- El proyecto creado por Wiktor Rott no incorpora ninguna funcionalidad de modificación del comportamiento dinámico del sistema en función de parámetros externos a él. Por ello será necesario la inclusión de funciones que permitan modificar el comportamiento del robot virtual en función del estado glucémico que esté atravesando la simulación.

8.5.2 Modelado del aspecto físico del robot virtualizado

Con el objetivo de brindar al usuario una experiencia lo más ajustada a la realidad, se ha buscado que la representación virtual del robot tuviera el mayor parecido con el aspecto físico del robot real. Para lograr esto, facilitando el trabajo, se ha partido de un modelo 3D ensamblado por Daniel Robles a partir de modelos CAD de las piezas disponibles a través de la propia compañía Makeblock (Figura 8.6).

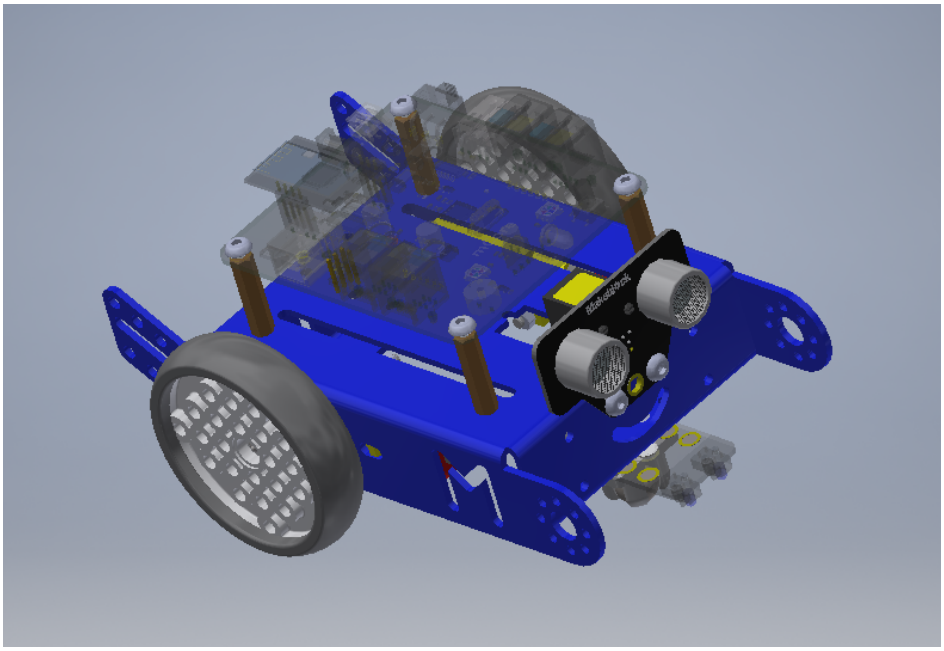


Figura 8.6: Captura del desarrollo previo del modelado 3D del robot mBot

Este archivo del modelo 3D del robot presenta una serie de limitaciones que no lo hace idóneo para la función que se buscaba:

- El archivo está en formato .stp, el cual no es reconocible por Unity.
- El ensamblado del robot incorpora el sensor de ultrasonidos en vez de la matriz leds que incorpora el dispositivo robótico utilizado.
- Las dimensiones del ensamblado del robot no concuerdan con la realidad.
- Algunas piezas no presentan el color que deberían posiblemente motivado por la conversión de tipos de archivo.

Para solventar todas estas limitaciones se ha hecho uso del software Autodesk Inventor. Este es un paquete de modelado 3D mediante diseño asistido por computador propiedad de la empresa

Autodesk. Su funcionamiento se basa en la generación de bocetos en 2D a los cuales se le brinda la tercera dimensión a partir de operaciones como la extrusión, revolución, o barrido de una superficie. Luego permite ensamblar el conjunto de piezas creado mediante uniones rígidas o móviles. [33]

El formato de exportación del modelo 3D final ha sido .OBJ ya que este es el único soportado por Unity de todos los que dispone Autodesk Inventor. Este es un formato adecuado ya que mantiene las dimensiones y unidades establecidas en el software CAD.

Para la sustitución del sensor de ultrasonidos por la matriz leds se debió crear la pieza expresamente pues no existe en la librería pública de Makeblock ni por parte de ningún diseñador independiente. Por ello el primer paso ha sido el de la generación del boceto a partir de las medidas tomadas del robot real. Este boceto, por la característica de simetría que presenta la pieza, se ha realizado solo de la mitad de esta. Este boceto se ha basado en un rectángulo al cual se ha incorporado en su parte superior una doble ranura. La transición entre el rectángulo y las ranuras son suavizadas mediante empalmes circulares y tangentes tal y como los presenta la pieza real. Con todo ello el boceto resultante acotado y totalmente restringido en grados de libertad es el mostrado en la Figura 8.7.

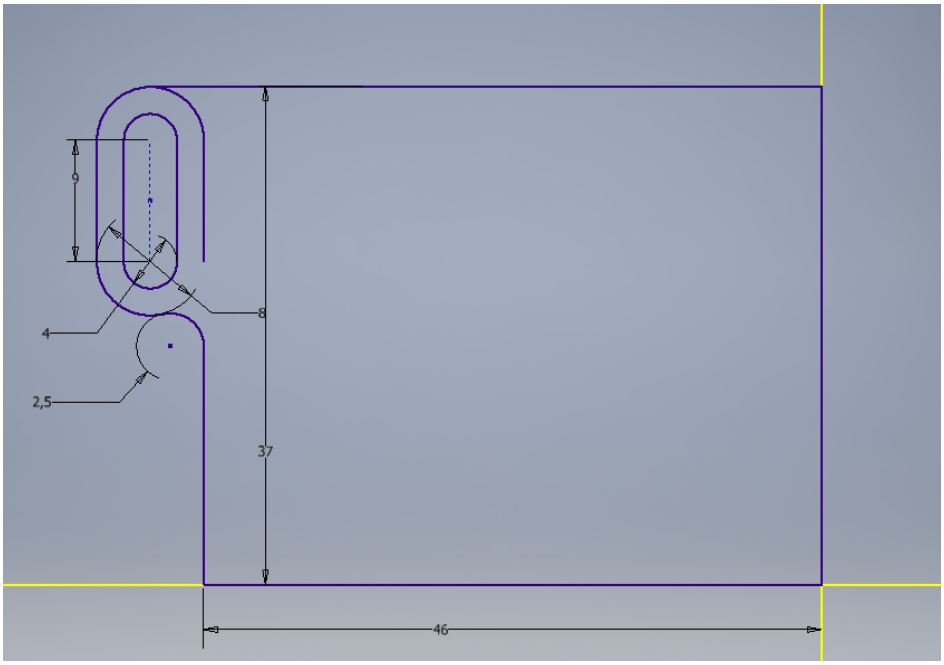


Figura 8.7: Boceto de la pantalla de matriz de leds del robot mBot

Sobre este boceto, al estar bien definido, se puede realizar una operación de extrusión simétrica de 2.5 mm de grosor y tras ella el redondeo de sus esquinas inferiores mediante empalmes circulares. También se han realizado los dos agujeros pasantes que servirán de anclaje con el robot mediante tornillos. Finalmente, todo esto se ha completado con una operación de simetría alrededor del plano XZ para lograr el aspecto final. El resultado del efecto de todas estas operaciones es visible en la Figura 8.8.

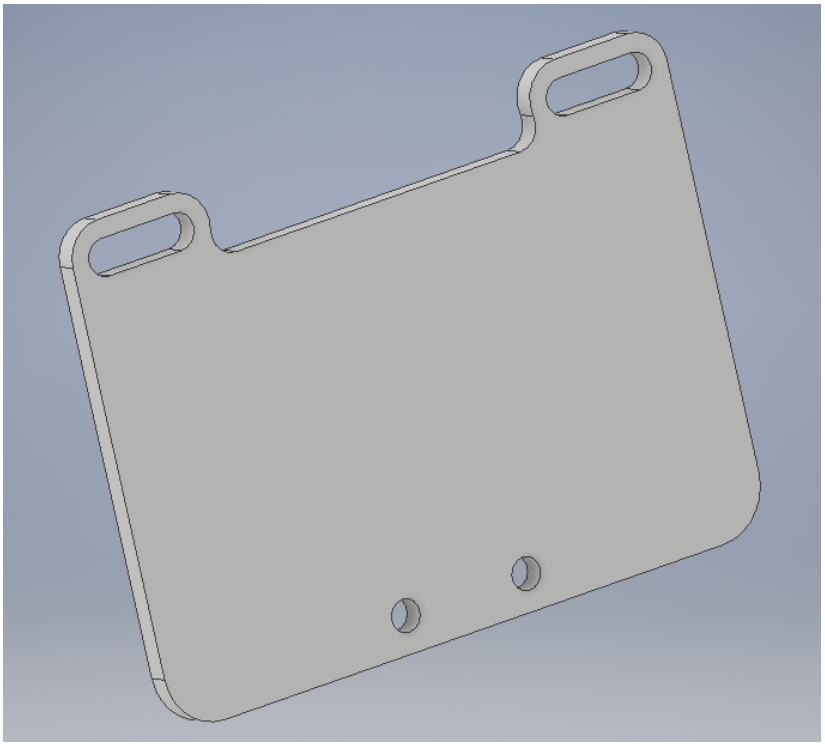


Figura 8.8: Modelo 3D de la pantalla de matriz de leds del robot mBot

Finalmente, una vez solventadas las demás limitaciones de dimensiones y pérdida de colores, el modelo 3D del robot, una vez incorporado en Unity, presenta la siguiente apariencia (Figura 8.9)

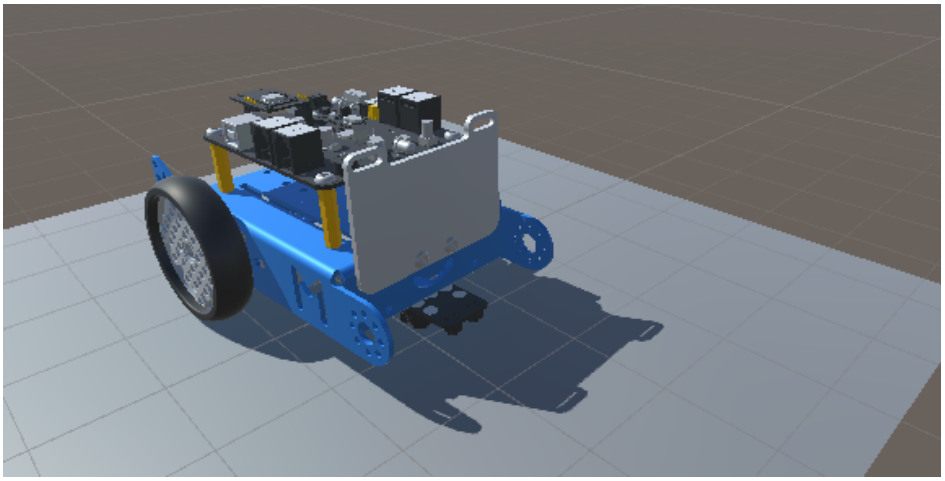


Figura 8.9: Modelo 3D del robot mBot en Unity

Una vez resuelto el aspecto físico del robot es necesario abordar como se va a mover y de qué forma implementarlo en Unity. Estos conceptos serán tratados en los siguientes apartados.

8.5.3 Elementos de PhysX y Unity utilizados para la virtualización del robot

Como se ha comentado PhysX es un motor físico muy potente capaz de simular multitud de sistemas y escenarios gobernados por las leyes de la física clásica. En este desarrollo se ha utilizado una pequeña parte de su potencial al tratarse el sistema a modelar de un sistema físico de complejidad moderada. Con todo ello los elementos de PhysX usados en este proyecto han sido:

- RigidBody (sólido rígido): es un componente que se debe incorporar a cualquier objeto que se quiera que sea afectado por las leyes físicas como la gravedad, rozamiento, colisiones, etc.
- Collider: es una simplificación de la geometría del objeto al que se incorpora con el objetivo de que sea más sencillo realizar los cálculos de colisiones entre objetos.
- Raycast: esta función permite la propagación de un rayo en una determinada dirección y a una determinada distancia desde un objeto y poder así detectar que objetos colisionan con este rayo. Esta función será la utilizada para la implementación del sensor siguelíneas.

El componente "RigidBody" en este proyecto ha sido incorporado a un "collider" de forma cúbica que engloba a todo el conjunto del robot. Gracias a que este componente implementa las ecuaciones de movimiento en el espacio 3D de un sólido rígido se ha utilizado una de sus características como es la limitación de grados de libertad para simplificar el modelado del sistema. Todo sólido rígido en el espacio dispone de 6 grados de libertad (3 desplazamientos y 3 rotaciones) por ello como el robot se quiere que se desplace en el plano se ha limitado la rotación en los ejes X y Z y el desplazamiento en el eje Y. En la Figura 8.10 se puede observar el modelo del robot en Unity junto a todos sus grados de libertad, en ella se ha colocado una cruz en aquellos desplazamientos o giros, que como se han comentado anteriormente, han sido limitados.

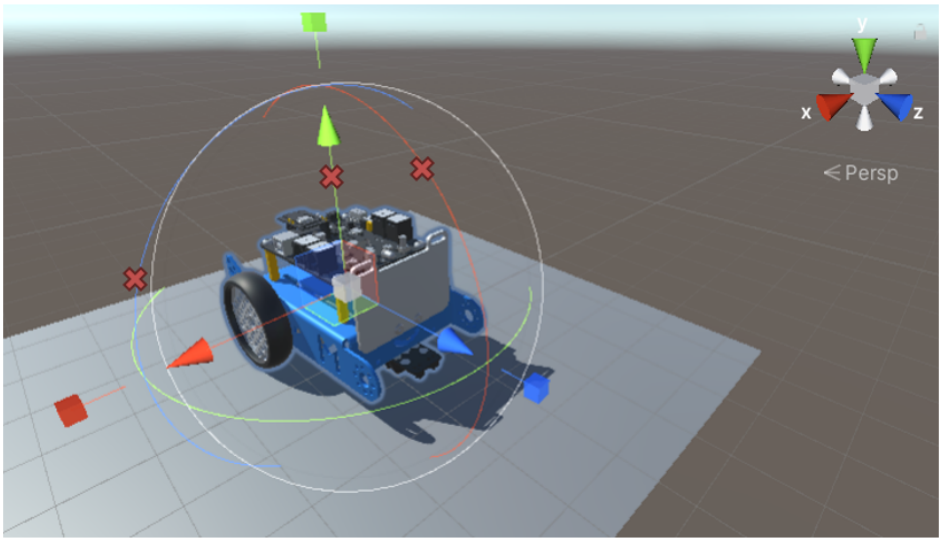


Figura 8.10: Grados de libertad del movimiento del robot en Unity

Por otro lado, para lograr una geometría más sencilla que la planteada por el modelo complejo en 3D del robot se ha hecho uso de los "colliders". En concreto se ha colocado uno de tipo caja envolviendo al cuerpo del robot y otros dos de tipo rueda en cada una de las ruedas motoras.

Este último tipo de "collider" es el más complejo y el que más posibilidades brinda, ya que no solo simula el rozamiento y fricción con el suelo, sino que también permite aplicar velocidades angulares a cada uno de ellos. Será sobre estos elementos sobre los que se aplicarán las acciones de control.

Por último, como se ha comentado anteriormente, el modelado del sensor siguelíneas ha sido implementado a través de la funcionalidad de "Raycast". Para ello se crea una pequeña esfera por cada par emisor-receptor que se tenga. A ella, en cada periodo de muestreo, se le indica que lance un rayo hacia abajo en la dirección perpendicular al suelo. Al contener las líneas que forman el circuito también "colliders", el rayo será capaz de detectarlos si chocara con alguno de ellos. Esta colisión activaría una señal indicando que ese par emisor-receptor ha detectado la línea. El manejo de estos rayos también permite incorporar filtros de colisiones, es decir, decidir con que capa de objetos se va a detectar la colisión. Las capas son identificadores diseñados para dotar de ciertas características iguales a un conjunto de objetos. Por ello la función de creación y difusión del haz tiene aplicado el filtro de detección de solo la capa que engloba a los objetos de tipo línea del circuito.

8.5.4 Implementación de las modificaciones del desarrollo de partida en Unity

Como se ha detallado en el Apartado 8.5.1, el desarrollo previo de partida presentaba una serie de limitaciones que lo hacían insuficiente para la función que se buscaba que desempeñara en la herramienta creada en este proyecto. Por ello además de brindar al modelo de un aspecto físico más realista se han abordado las modificaciones necesarias para solventar las limitaciones comentadas.

Para dotar de las dimensiones, masa, y fricción con el aire más acordes al dispositivo robótico con el que se está trabajando se han modificado los parámetros que definen al componente "RigidBody" mediante las magnitudes expresadas en el Apartado 8.2. En cuanto a la fricción con el aire, al no disponer de la posibilidad de realizar experimentos para conocer su valor se ha ajustado este realizando pruebas hasta seleccionar un valor que proporcione resultados razonables. Cabe destacar que esta fricción con el aire en PhysX es modulada a través de un parámetro llamado "Drag" el cuál actúa en una fuerza de sentido contrario al movimiento del robot siguiendo la siguiente expresión cuadrática: [34]

$$\| F_{Drag} \| = \| Velocidad \|^2 \cdot Drag \tag{8.11}$$

En este caso este parámetro de "Drag" ha sido finalmente definido en un valor de 0.05 al considerarse que la geometría del robot, con muchos espacios y huecos, no provocaría una gran resistencia al aire.

En cuanto al sensor siguelíneas se ha seguido la misma implementación que en el desarrollo previo. Pero en este caso se han utilizado, como en el robot real, solo dos pares emisor-receptor los cuales han sido colocados en la posición determinada en el modelo 3D incluido. Para hacer aún más realista su simulación se ha ajustado su tamaño y se ha intercambiado su lógica para que estos pares se enciendan cuando no detecten la línea como es en el caso del sensor que incluye el robot mBot.

Como se ha comentado en el apartado dedicado a la explicación del desarrollo previo (Apartado 8.5.1), este implementaba un control de tipo PID para actuar sobre las ruedas motoras del robot. Evidentemente, esta es una mejor solución que la implementada en este proyecto, pero no es adecuada porque no es la utilizada para el control del robot real. Con esta limitación se ha sustituido este control por un control todo o nada el cual sigue un algoritmo muy similar al implementado para el control del robot real (Figura 8.11):

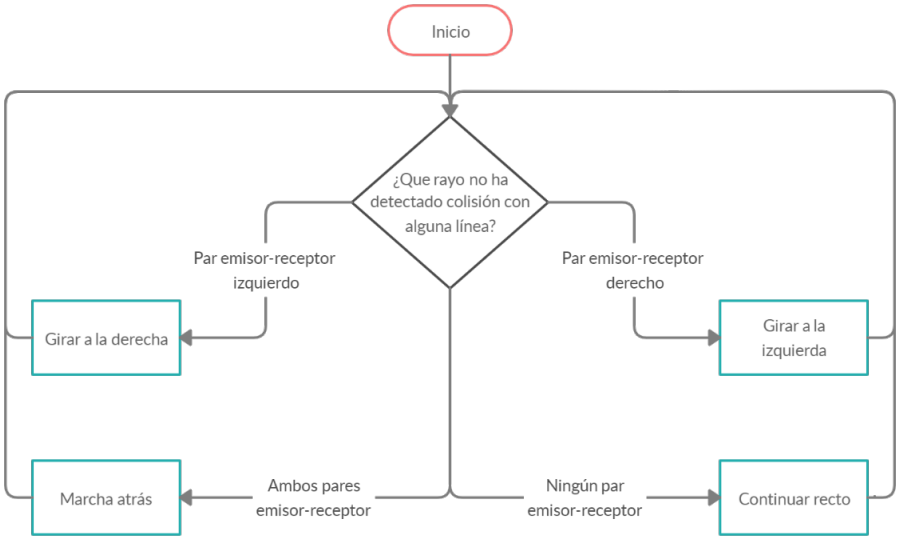


Figura 8.11: Funcionamiento del sensor siguelíneas modelado en Unity

Relativo a la administración de las acciones de control esto es modelado a través del "collider" de tipo rueda. Este posee parámetros ajustables como la masa y diámetro de la rueda ajustados según los parámetros definidos en el Apartado 8.2. También incorpora otro tipo de parámetros como el coeficiente de amortiguamiento de la junta de revolución o el de fricción del neumático con el suelo. El primero de ellos al no disponer de datos ni experimentos desde los que obtenerlo se ha definido de forma que ofrezca un resultado razonable en la simulación. Para la fricción entre la rueda y el suelo se ha definido el coeficiente de fricción relativo al contacto entre el caucho y el hormigón (0.8) el cual será la situación más aproximada a la realidad. Por último, este componente también incorpora parámetros para modelar la suspensión entre la rueda y el vehículo, estos han sido todos colocados a cero al no disponer el robot mBot de ningún sistema de amortiguamiento.

Tal y como se ha desarrollado a lo largo de todo este proyecto, la herramienta educativa diseñada está basada en el uso de la competición para fomentar el aprendizaje de los alumnos. Con esta filosofía era necesario trasladar esa competición entre los robots al mundo virtual. Para ello se ha debido duplicar todos los elementos constituyentes del robot virtual y los *scripts* que modelan su comportamiento y control. Además, para crear un efecto magnificado de competición se ha aprovechado las ventajas que proporciona el mundo virtual para implementar que ambos robots sigan las líneas del mismo circuito. Esto producía el problema de que ambos robots se chocaban entre ellos al tener ambos "colliders" entre sus componentes. Esta problemática se ha solucionado, de nuevo, con el uso de las capas de objetos. Estas, entre sus características, permiten

la modulación de las detecciones de colisión entre distintos conjuntos de objetos identificados por estas capas. Toda esta casuística está recogida en la conocida como Matriz de Colisión (ver Figura 8.12).

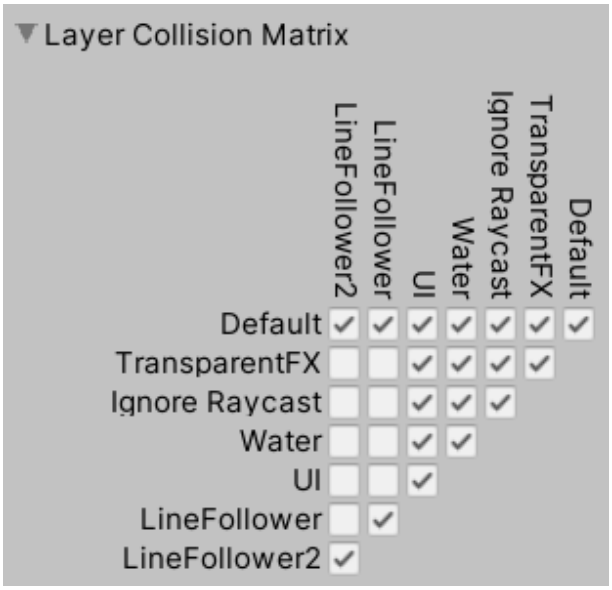


Figura 8.12: Matriz de colisiones de Unity

En ella se puede observar cómo los objetos "LineFollower", que son aquellos que engloban todos los componentes del robot virtual, solo tienen activada la detección de colisión con ellos mismos (para que todos sus elementos permanezcan unidos) y con la capa Default (incluida en ella el suelo) para que el robot detecte este límite y no sea capaz de atravesarlo.

Para finalizar la implementación de la virtualización de la competición entre los dispositivos robóticos era necesario incorporar funciones que permitieran antes de iniciar la carrera conocer los resultados de la simulación del amigo o de la máquina con los que se va a competir. Esto se ha realizado compartiendo entre los usuarios conectados o la máquina la información relativa a la simulación del competidor. Esta información es un vector que contiene, para cada uno de los 235 puntos en los que se ha evaluado la simulación, el valor del estado glucémico en el que se encontraba la simulación en ese momento. Para ajustar la duración de la simulación, conjugando el tiempo de muestreo del control y la lectura de estos estados, se ha implementado que para que esta dure aproximadamente el mismo tiempo que en la realidad se haga la lectura de la siguiente posición del vector de estados cada 16 repeticiones del bucle de control. Estos estados son manifestados por el robot de la misma forma que en la realidad. Para ello se han colocado dos esferas en el modelo 3D que cambian de color emitiendo luz simulando los leds interiores del robot mBot. Y las modificaciones del comportamiento de este también han sido implementadas igual que en el robot real, disminuyendo su velocidad en el estado de hiperglucemia e introduciendo un retardo en el control en el caso de estar atravesando una hipoglucemia.

Una vez finalizada la simulación, cuando se haya terminado de leer el vector de estados, aparecerá un panel donde el usuario podrá ver tanto los porcentajes que su simulación ha permanecido en cada estado glucémico como los de su competidor.

Todas estas modificaciones al desarrollo previo de partida dan como resultado las pantallas visibles en la Figura 8.13 y Figura 8.14.

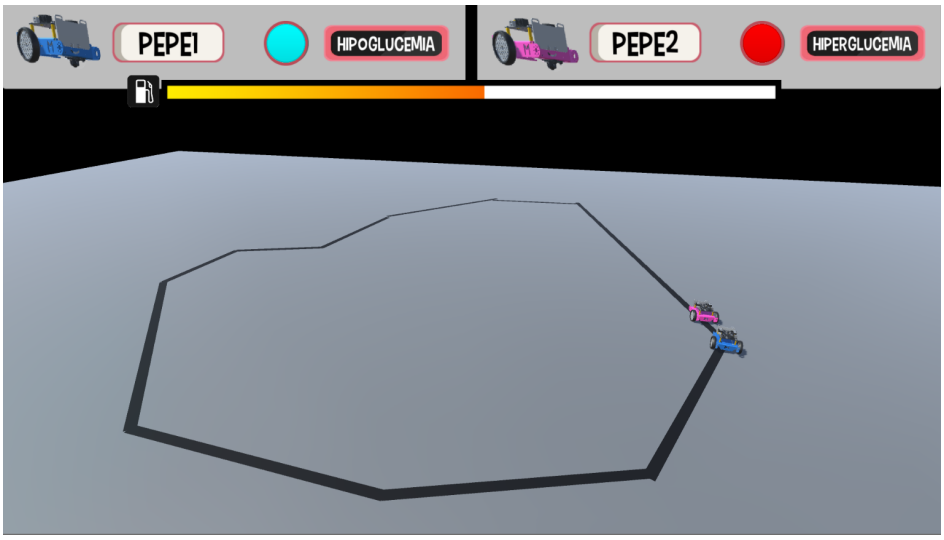


Figura 8.13: Captura de la pantalla de simulación con robot virtual

Figura 8.14: Captura del panel de muestra de los porcentajes de tiempo en cada uno de los estados glucémicos

Capítulo 9

Virtualización de circuitos siguelíneas mediante visión artificial

«Si se ve a simple vista, se puede resolver por visión artificial» — Anónimo

9.1 Definición

El objetivo de este capítulo es detallar el funcionamiento de la subherramienta creada para facilitar la creación y virtualización de circuitos para robots siguelíneas. Esta herramienta se trata de una aplicación programada en lenguaje C++, la cual con ayuda de la librería OpenCV será capaz de reconocer y virtualizar la imagen de un circuito dibujado en un papel. Esta virtualización tendrá dos vertientes, como el sistema desarrollado, una física y otra virtual. Esta aplicación de visión artificial creará dos ficheros independientes, uno será usado en el software AutoCAD para poder imprimir el circuito a escala real y ser utilizado en los talleres educativos, y el otro fichero servirá para ser leído por Unity para ser utilizado en la pantalla de simulación con robot virtual.

9.2 C++, el lenguaje de la aplicación de visión artificial

Este lenguaje es una versión del lenguaje C diseñado en 1979 por Bjarne Stroustrup. Su intención era la de crear un lenguaje con la capacidad de poder manipular objetos. Por ello este lenguaje incorpora la programación estructurada y la orientado a objetos, conociéndose a este lenguaje como multiparadigma. [35]

Este lenguaje junto a su antecesor C están en la mayoría de grandes proyectos informáticos gracias a que se mantienen actualizados y son muy versátiles. Además, son lenguajes que han servido de base para la creación de otros para utilidades más específicas.[35]

En este proyecto la programación de la aplicación de visión artificial en C++ va a ser programada en el IDE VisualStudio debido entre otras cosas a las características que se detallaron de este entorno de desarrollo en el Apartado 7.2.3.

9.3 Opencv, la librería de visión artificial

Esta es una biblioteca de código abierto dedicada a suplir al programador de funciones relacionadas con la visión artificial, el tratamiento de imágenes y el machine learning. De esta forma, con sus más de 2500 algoritmos optimizados es capaz de acelerar los desarrollos de herramientas de visión por ordenador en multitud de proyectos. Esta librería es usada por grandes compañías como Google, Yahoo o Microsoft hasta pequeñas startups y proyectos de investigación. Este uso masivo ha provocado que se disponga de una gran comunidad de desarrolladores que aportan información sobre su uso y sobre la resolución de los problemas más frecuentes. [36]

Esta librería está totalmente implementada en C++, pero también incorpora de forma nativa conectores para otros lenguajes como Python, Java o Matlab. [36]

Por todos estos motivos ha sido la librería elegida para la implementación de esta aplicación de virtualización de circuitos siguelíneas.

9.4 Restricciones en el formato de la imagen analizada

Como se ha anticipado en el Apartado 9.1 de definición de la herramienta de visión artificial, el propósito de esta, es virtualizar la información de un circuito contenida en la fotografía de una hoja de papel. Debido a que esta aplicación es una subherramienta dentro de todo el sistema creado, no se han centrado todos los esfuerzos en su implementación y funcionalidad. Por ello aún presenta un carácter de prototipo, aunque totalmente funcional teniendo en cuenta las siguientes limitaciones:

- El circuito dibujado sobre el papel debe presentar un contraste suficiente sobre la superficie del folio que permita una correcta segmentación de este.
- En la fotografía tomada al papel con el circuito dibujado no puede haber elementos distintos a las propias líneas que definen a este.
- El circuito solo puede estar basado en segmentos, no se admiten curvas debido a su complejidad de virtualización y su imposibilidad de ser introducidas en Unity con el método utilizado.
- Los segmentos que componen el circuito deberán estar ligeramente separados entre sí en cada cambio de dirección para poder ser captados por la aplicación.
- Los escalados aplicados para ser exportados a AutoCAD y Unity tendrán un correcto funcionamiento si el circuito está dibujado sobre un folio tamaño A4 siendo capturada toda su superficie.

9.5 Algoritmo de procesado de la imagen

Como toda aplicación de visión artificial, por básica que esta resulte, presenta una serie de pasos a seguir necesarios:

1. Calibración de la cámara.
2. Preprocesado de la imagen capturada.
3. Procesado y aplicación de operaciones sobre la imagen.
4. Exportación de resultados.

Cada uno de estos pasos básicos que conforman la macroestructura de un algoritmo de visión artificial serán analizados con mayor detalle a lo largo de los subapartados posteriores.

9.5.1 Calibración de la cámara

Este es un proceso necesario a la hora de capturar imágenes tratadas por visión artificial si se quiere lograr un correcto resultado ya sea con una cámara profesional o la de un teléfono móvil. Estas calibraciones se realizan a través de la captura de una imagen patrón de la cual se conocen las dimensiones y separaciones de los objetos que contiene. Existen multitud de diseños de patrones, pero uno de los más utilizados y que mejor resultado da es el de matriz de círculos alterna, utilizado en este proyecto (Figura 9.1).

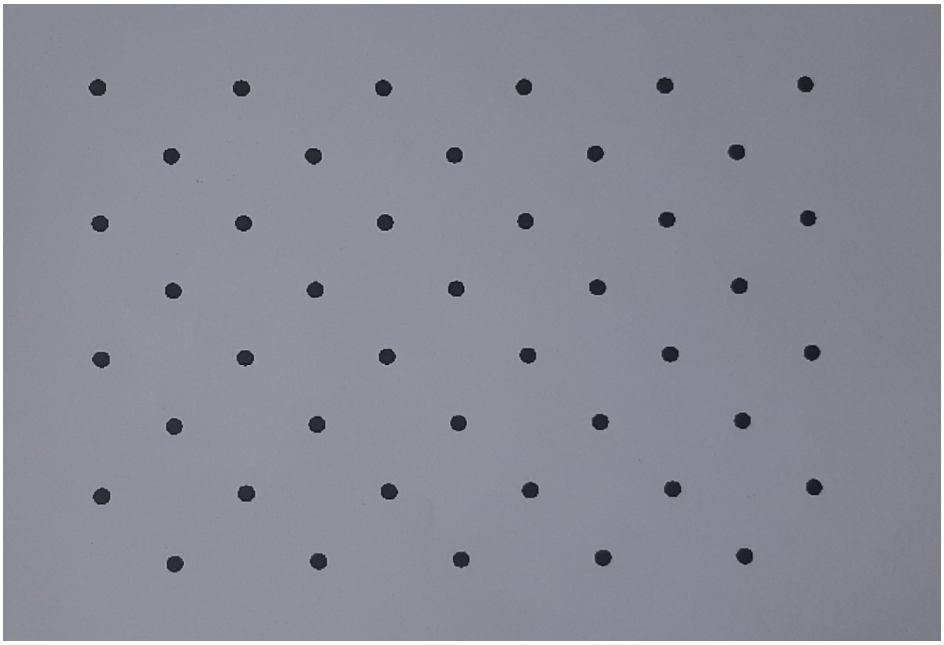


Figura 9.1: Fotografía del patrón utilizado para la calibración de la cámara

Por la sencillez de la aplicación en este proyecto solo se va a realizar la calibración de la homografía.

Para el reconocimiento de los círculos que conforman el patrón se utiliza la función de la librería OpenCV llamada "findCirclesGrid", esta a partir de la imagen patrón es capaz de calcular las

coordenadas en pixeles de los centroides de cada uno de los círculos que lo componen almacenándolos en un vector de puntos. Para lograr un correcto funcionamiento de esta función, es necesario ajustar sus parámetros de búsqueda (filas y columnas del patrón, umbral máximo y mínimo de segmentación, y tamaño en pixeles del área máxima y mínima de los objetos de interés). Tras aplicar esta función, correctamente parametrizada, sobre la imagen patrón se logra el resultado visible en la Figura 9.2.

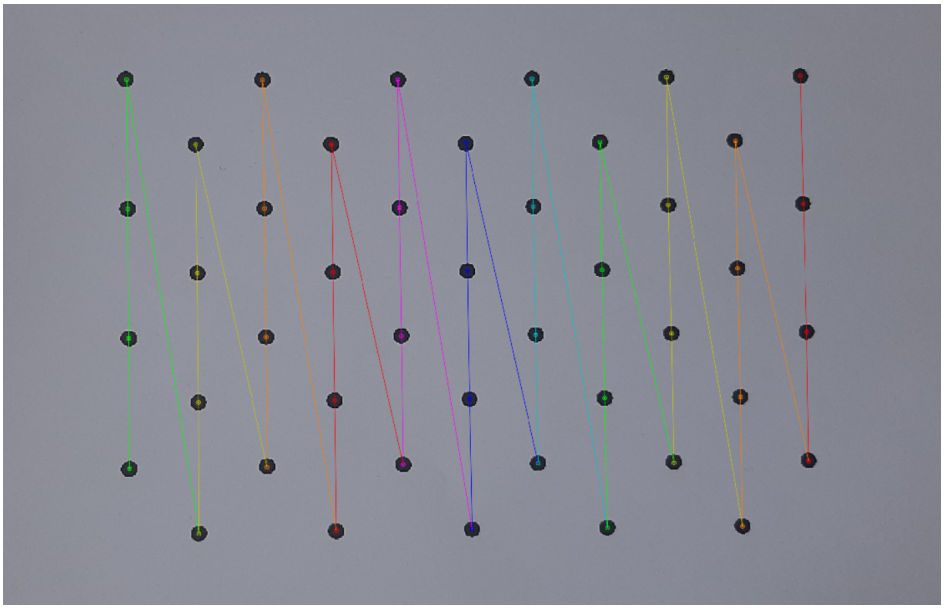


Figura 9.2: Detección de los centroides de los puntos en el patrón

Una vez obtenidas las coordenadas de los centroides de los objetos de interés del patrón el siguiente paso es el de la generación de las coordenadas de los puntos que corresponden con la realidad. Esto se puede realizar ya que se conocen las dimensiones con las que ha sido generado el patrón.

La obtención de los puntos del mundo real y de la imagen para el patrón son el paso previo para el cálculo de lo que se conoce como matriz de homografía. Esta no es más que una matriz cuya aplicación realiza una transformación proyectiva entre dos figuras planas, en este caso los elementos de unas imágenes. Por ello conociendo esta matriz, aplicando su inversa a partir de las coordenadas pixélicas de los objetos detectados se podrán calcular las coordenadas de sus homólogos en el mundo real. [37]

Esta matriz de dimensiones 3x3 puede ser obtenida a partir de cuatro puntos no colineales, hasta un factor de escala, utilizando el método de resolución de fijar su último elemento a la unidad. De esta forma el cálculo de la matriz de homografía se reducirá a la resolución del siguiente sistema matricial:[37]

$$\begin{bmatrix} u \\ v \\ s \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{9.1}$$

En él, X e Y son las coordenadas de los puntos de interés en el mundo real y u, v, s, definen las siguientes relaciones con los puntos x e y en la imagen:[37]

$$x = \frac{u}{s} \tag{9.2}$$

$$y = \frac{v}{s} \tag{9.3}$$

Por ello por cada par de puntos homólogos se cumplirán las siguientes ecuaciones:[37]

$$s_i \cdot x_i = h_{11}X_i + h_{12}Y_i + h_{13} \tag{9.4}$$

$$s_i \cdot y_i = h_{21}X_i + h_{22}Y_i + h_{23} \tag{9.5}$$

$$s_i = h_{31}X_i + h_{32}Y_i + h_{33} \tag{9.6}$$

Finalmente sustituyendo la Ecuación 9.6 en la Ecuación 9.4 y en la Ecuación 9.5, y fijando $h_{33} = 1$, para los cuatro puntos homólogos se obtiene el siguiente sistema matricial fácilmente resoluble:[37]

$$\begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -y_1X_1 & -y_1Y_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -y_2X_2 & -y_2Y_2 \\ X_3 & Y_3 & 1 & 0 & 0 & 0 & -x_3X_3 & -x_3Y_3 \\ 0 & 0 & 0 & X_3 & Y_3 & 1 & -y_3X_3 & -y_3Y_3 \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -x_4X_4 & -x_4Y_4 \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -y_4X_4 & -y_4Y_4 \end{bmatrix} \cdot \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \\ x_4 \\ y_4 \end{bmatrix} \tag{9.7}$$

Lógicamente OpenCV tiene una función que realiza todos estos cálculos automáticamente. En este proyecto la homografía ha sido calculada de la forma manual y con OpenCV reflejando resultados muy similares, pero finalmente se ha usado la calculada por OpenCV para el resto de los cálculos. Esta matriz ha sido almacenada en un fichero XML para poder ser utilizada en cualquier momento sin necesidad de volver a calibrar la cámara siempre que se mantengan las condiciones en las que se capturan las imágenes.

9.5.2 Preprocesado de la imagen capturada

Para que las operaciones de transformación y análisis de la imagen se lleven a cabo con éxito es necesario preparar esta mediante un preprocesado. Este consiste en realizar la lectura en escala de grises para poder realizar la correcta segmentación de la imagen (obtención de una imagen binaria). Para evitar problemas de indefinición en los bordes de la imagen se realiza un recorte de esta de un 1 % en cada una de sus dimensiones (Figura 9.3).

La segmentación sobre la imagen recortada se realizará con una función de OpenCV que implementa el algoritmo de Otsu (análisis de la dispersión de los niveles de gris) para la selección del umbral óptimo de segmentación (Figura 9.4).

Esta matriz binaria resultante será la base sobre la que se aplicaran las sucesivas operaciones de detección de contornos.

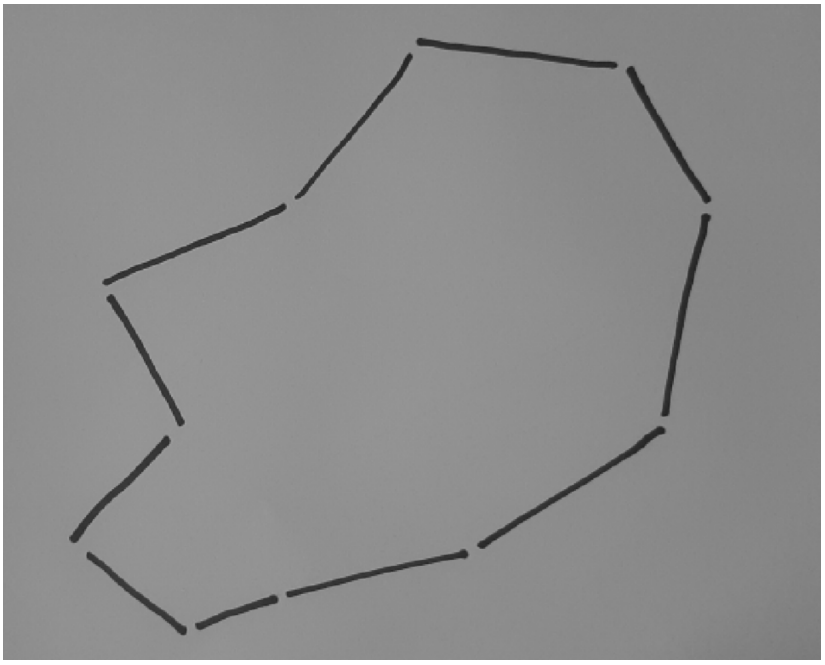


Figura 9.3: Dibujo del circuito a virtualizar

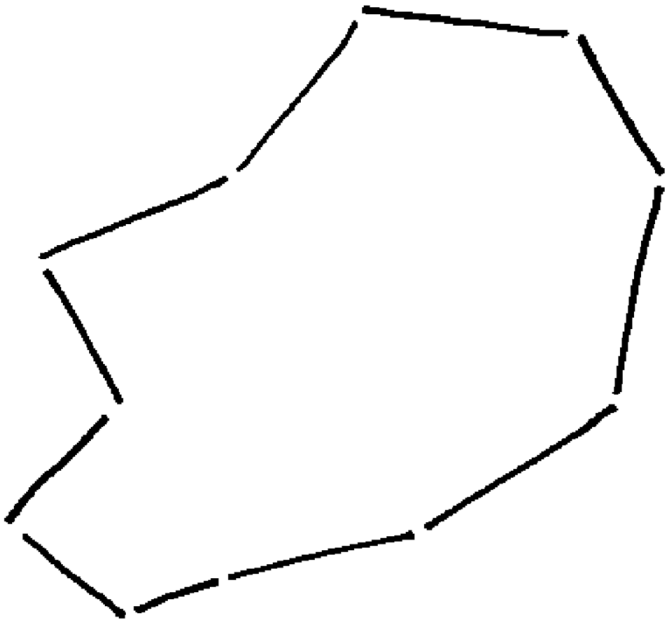


Figura 9.4: Segmentación de la imagen analizada

9.5.3 Procesado de la imagen segmentada

La primera operación que realizar sobre la matriz binaria será la de detección de los contornos de los objetos que contiene. Para ello se hace uso de la función "findcontours" de OpenCV, la

cual implementa el algoritmo de seguimiento de bordes desarrollado por Satoshi Suzuki y Keiichi Abe. Este algoritmo va recorriendo la matriz de derecha a izquierda y de abajo a arriba por lo que los contornos serán detectados en este orden, el cual probablemente no coincida con el orden o secuencia que siguen los segmentos que conforman el circuito. Esto será un problema que habrá que solucionar para lograr ordenar los bordes encontrados del mismo modo que lo están en el circuito dibujado.

El propósito final es el de localizar los puntos de unión (donde cambian de dirección) cada uno de los segmentos. Este proceso se ha llevado a cabo con la implementación del siguiente algoritmo:

1. Obtención de la recta que mejor se ajuste a cada uno de los contornos identificados. Esta operación se realiza mediante la función "fitline" de OpenCV, que realiza el ajuste mediante la minimización por mínimos cuadrados de la distancia entre cada uno de los puntos del contorno y la recta resultante. Esta función devuelve un punto de la recta (x0,y0) y un vector normalizado colineal a la línea (vx,vy) (ver Figura 9.5).

2. Hallar las ecuaciones de la recta a partir de la obtención de los puntos extremos con las siguientes expresiones:

$$m = \frac{v_y}{v_x} \tag{9.8}$$

$$P1_x = 0 \rightarrow P1_y = m \cdot (0 - x_0) + y_0 \tag{9.9}$$

$$P2_x = Columnas_{matrizbinaria} \rightarrow P2_y = m \cdot (P2_x - x_0) + y_0 \tag{9.10}$$

3. Detección de una aproximación de los puntos iniciales y finales de cada uno de los segmentos de la recta contenidos en los contornos. Esto se utilizará para poder ordenar esos segmentos en el orden correcto. Esta aproximación se logra cogiendo como punto inicial del segmento el primer punto del vector del contorno detectado del objeto y como punto final el correspondiente a la posición media de este vector ya que se detecta el contorno cerrado completo. Si estos contornos fueran rectángulos perfectos este cálculo obtendría con exactitud los puntos buscados, pero en la realidad estos segmentos identificados no se asemejarán demasiado a un rectángulo perfecto y el punto medio del vector de contornos no coincidirá con el extremo de la línea ajustada. En la Figura 9.6 se representa este problema de precisión y en la Figura 9.7 se pueden observar donde se encuentran los puntos iniciales y finales siguiendo este método aproximado.

4. Ordenamiento de los segmentos detectados en la misma secuencia que componen el circuito. Esto se ha realizado con la implementación de un algoritmo que calcula la mínima distancia entre uno de los puntos extremos aproximados anteriores y todos los puntos iniciales y finales de los demás segmentos. De esta forma el segmento que contenga ese punto con mínima distancia será el siguiente contorno de búsqueda. Si ese punto de mínima distancia ha sido un punto inicial, siguiendo el recorrido, se inicializará de nuevo la búsqueda con el punto final aproximado de ese contorno y viceversa. Con este método se obtendrá una lista de contornos ordenados en sentido horario o antihorario cuya secuencia conforma el circuito (ver Figura 9.8).

5. Obtención, siguiendo la secuencia de contornos ordenados, de los puntos de corte entre cada una de las rectas ajustadas. Esto creará una lista secuencial de puntos que corresponden

con los puntos de corte de las prolongaciones de estos segmentos en los que se divide el circuito. En la Figura 9.9 se muestra el resultado final del procesado de la imagen donde se pueden ver las coordenadas de los puntos buscados ordenados secuencialmente según el circuito analizado.

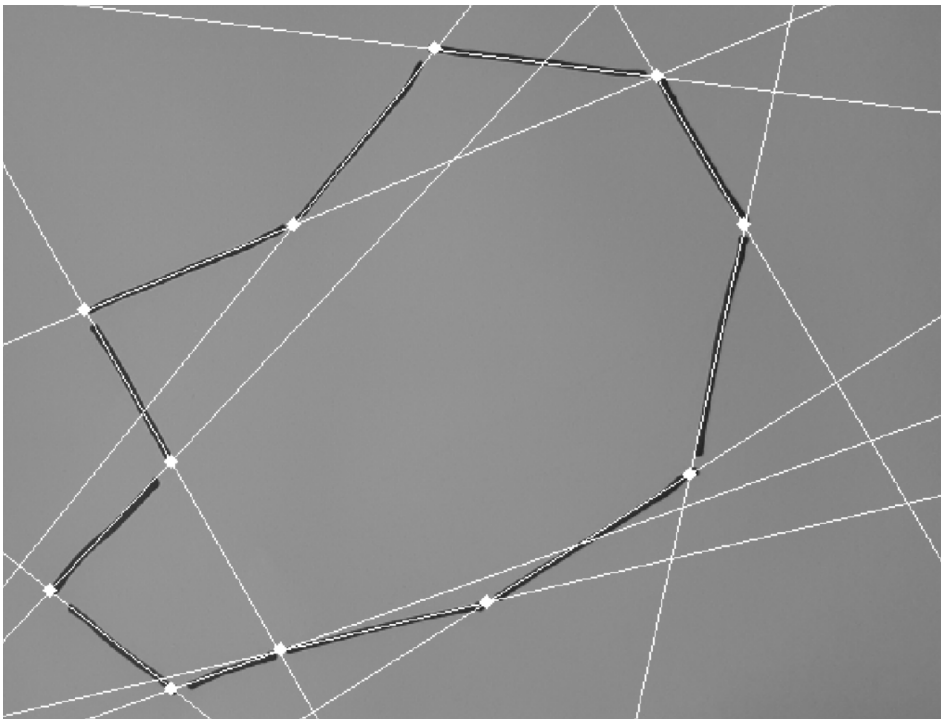


Figura 9.5: Ajuste de la recta que atraviesa longitudinalmente a cada segmento

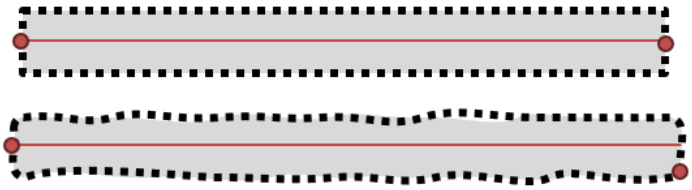


Figura 9.6: Representación del problema de detección de los puntos extremos del segmento

Una vez obtenidos los puntos de corte de todas las rectas ordenados secuencialmente, cada uno de estos se debe multiplicar por la inversa de la matriz de homografía y ser normalizados dividiendo cada una de sus componentes por la última de ellas, de modo que en esta última posición se encuentre la unidad. Estas coordenadas ya estarán trasladadas al mundo y serán las que se exporten a los ficheros correspondientes.

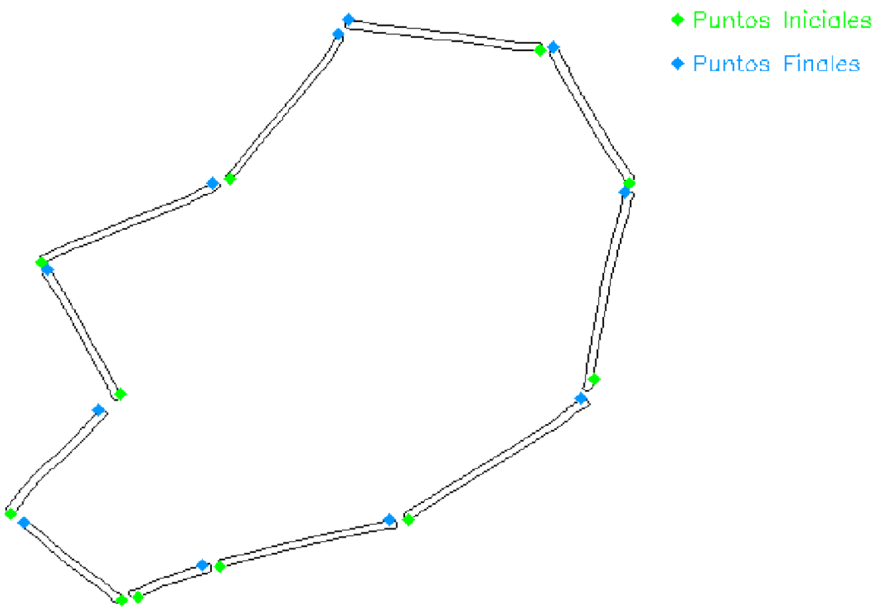


Figura 9.7: Detección de los puntos iniciales y finales de forma aproximada

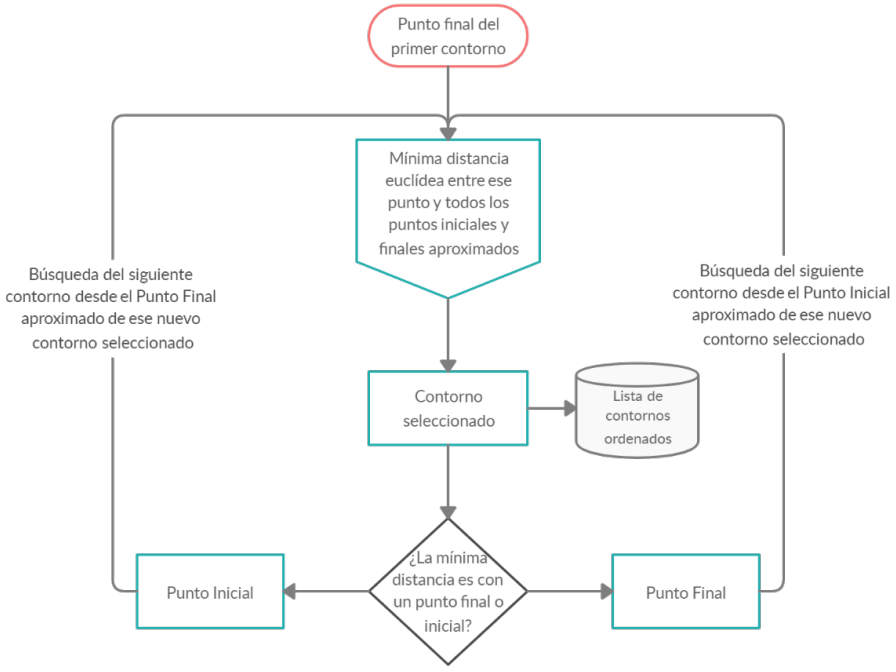


Figura 9.8: Algoritmo de ordenación de los contornos en función de la secuencia del circuito

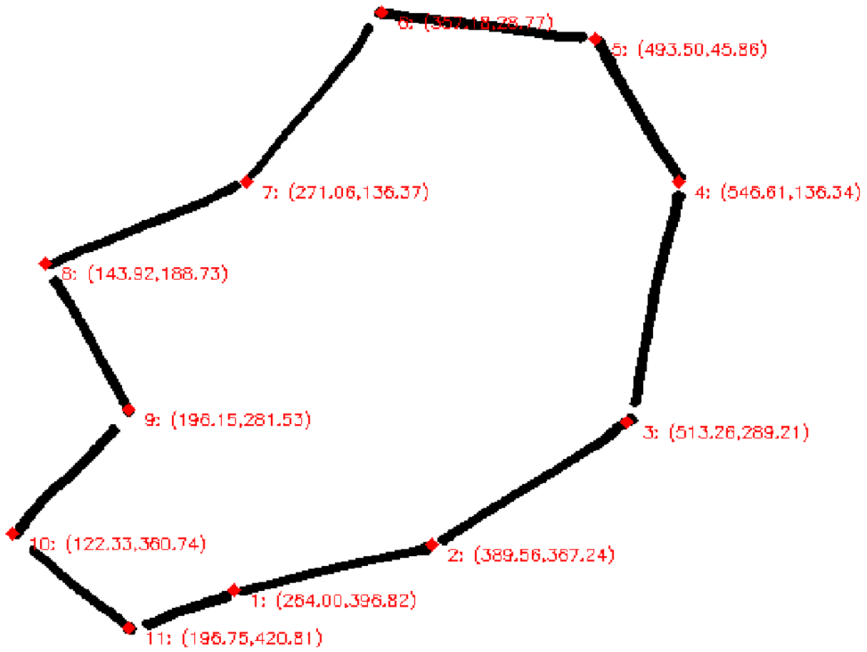


Figura 9.9: Coordenadas de los puntos de corte de los segmentos ordenados

9.5.4 Exportación de resultados

Como se ha especificado en el Apartado 9.1, el propósito de los resultados obtenidos de la aplicación de visión artificial es poder usarlos para generar circuitos a tamaño real para ser usados en talleres, y también para generar circuitos virtuales donde un robot virtualizado los pueda recorrer.

El primero de ellos será un fichero exportado directamente desde la aplicación de visión artificial creada en formato ".src", este es un archivo de comandos que puede ser interpretado por programas capaces de ejecutarlos. En este caso este fichero ha sido creado con comandos del software Autodesk AutoCAD para que este pueda interpretarlos y preparar automáticamente el circuito para su impresión.

Autodesk AutoCAD es un software de diseño asistido por ordenador, diseñado para que profesionales de la arquitectura y la ingeniería puedan crear dibujos y planos precisos tanto en 2D como en 3D. Pertenece a la misma compañía que el software Inventor (Autodesk) teniendo una gran compatibilidad con él. [38]

El fichero sigue la misma estructura de pasos que seguiría una persona realizando la construcción del circuito en AutoCAD a mano:

1. Invocación del comando pol: este inicializa la creación de una polilínea con la que generar el circuito.
2. Escritura de las coordenadas X e Y del primer punto del circuito o contorno que se quiere reproducir.

3. Establecimiento del grosor inicial y final de la línea construida por el comando polilínea.
4. Escritura secuencial de todos los puntos de inflexión de la polilínea, los cuales coincidirán con los calculados por la aplicación de visión artificial.
5. Cierre del comando polilínea finalizando su construcción.
6. Invocación del comando AI_SELALL para seleccionar todos los objetos presentes en el dibujo.
7. Invocación del comando MOVE para poder desplazar el circuito al origen de coordenadas consiguiendo así centrar el objeto en el plano. Para ello se hará coincidir el punto inicial de la polilínea con el origen de coordenadas.
8. Aplicación del comando SIMETRIA (respecto al eje X) y luego el comando _rotate (giro de 270°) para deshacer la reflexión producida por la captura de la imagen con la cámara. De esta forma el usuario verá en AutoCAD el circuito en la misma orientación en la que lo dibujó.
9. Invocación del comando AI_SELALL para seleccionar de nuevo todos los objetos presentes en el dibujo.
10. Aplicación del comando ESCALA para realizar un escalado del circuito a la escala correspondiente al formato de hoja que se precise.
11. Ejecución de un zoom de tipo extensión para poder visualizar todo el circuito a tamaño máximo en la pantalla.

La Figura 9.10 muestra como se ve la ejecución de este fichero en AutoCAD.

Llegados a este punto será el usuario el que manualmente deberá realizar la impresión del circuito para la cual simplemente deberá precisar que la escala es 1:1 y que desea imprimir en el formato de papel para el que haya generado la virtualización del circuito. También, deberá precisar que la orientación sea horizontal y la impresión a PDF respete los grosores de las líneas. Un ejemplo del circuito impreso (en formato DIN A1) se puede ver en la Figura 9.11.

La herramienta de virtualización de circuitos también realiza la exportación de otro archivo en este caso en formato de texto (.txt). El objetivo de este es poder ser procesado por la aplicación en Unity generando así el circuito en la pantalla de simulación con robot virtual para que este sea capaz de seguir las líneas que lo componen. Este fichero contendrá las coordenadas X e Y separadas por el carácter "/" de los puntos ordenados según la secuencia que compone el circuito. Cada línea del fichero contendrá las coordenadas de uno de los puntos.

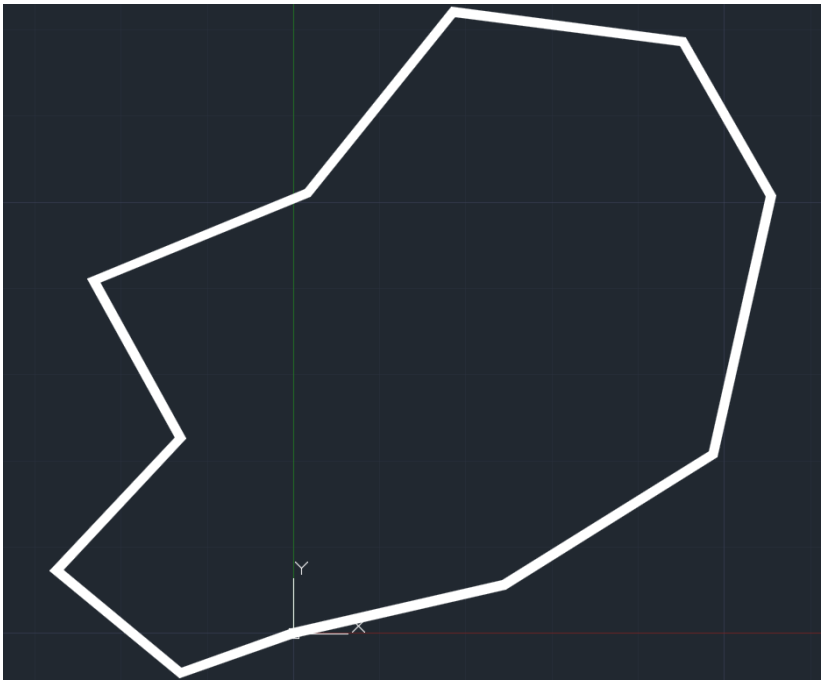


Figura 9.10: Resultado de ejecución del fichero del circuito virtualizado en Autodesk AutoCAD

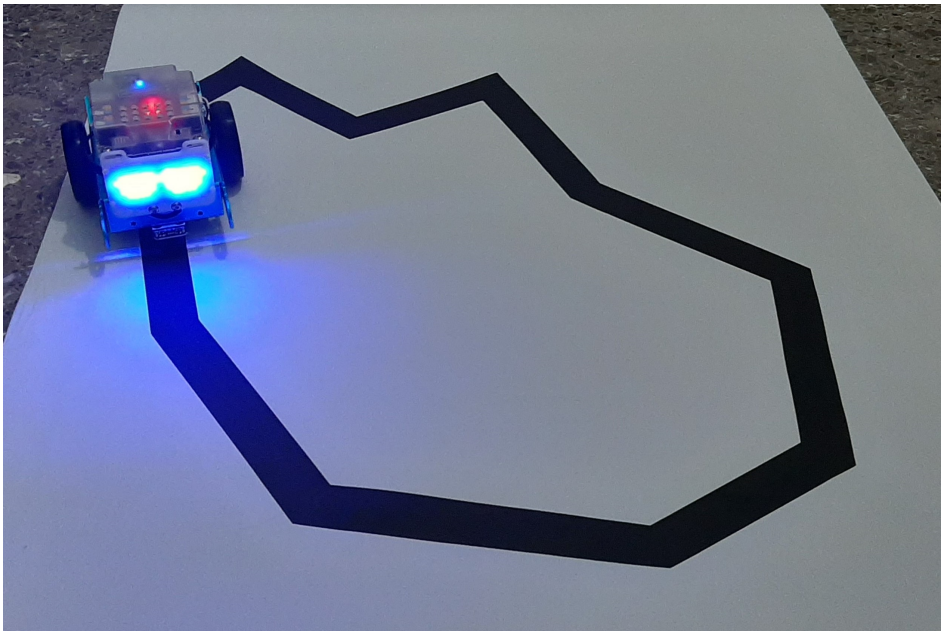


Figura 9.11: Impresión del circuito generado por el fichero en AutoCAD

9.5.5 Interfaz gráfica de usuario

La herramienta mostrada en este capítulo se trata de un prototipo destinado a usuarios avanzados de la aplicación. Por ello no se han centrado esfuerzos en crear una interfaz de usuario sencilla y agradable a la vista. La interfaz desarrollada se basa en la consola de VisualStudio donde se le irán pidiendo por pasos al usuario que introduzca acciones y configuraciones en la herramienta. Un usuario que utilice la aplicación de virtualización de circuitos deberá seguir los siguientes pasos:

1. Elección de si se desea calibrar la cámara o no.
2. Si se desea realizar la calibración de la cámara, se le pedirá al usuario que indique la ruta del archivo donde se encuentra la imagen capturada del patrón y que indique el numero de objetos en la dimensión vertical y horizontal que contiene así como la separación entre ellos. También deberá indicar la ruta donde desea almacenar el fichero que contiene la matriz de homografía calculada.
3. Indicar la ruta de la imagen capturada del circuito a virtualizar y si se desea utilizar la matriz de homografía la ruta del fichero que la contiene.
4. Indicar si se quiere generar el fichero de AutoCAD y el de Unity, solo uno de los dos o ninguno.
5. Introducir en el caso de seleccionar que se quiere generar algún fichero, la ruta en la que almacenarlos. El nombre del archivo será dado por la aplicación automáticamente.
6. Indicar si se desea generar el fichero de AutoCAD a que formato DIN se desea exportar el circuito.

El usuario será guiado por la interfaz en todo momento indicándole que comandos debe introducir para realizar cada acción que desee.

La interfaz mostrará por defecto al usuario las siguientes imágenes representativas del procesado que se ha realizado a la imagen:

- Imagen original capturada del circuito por el usuario.
- Imagen segmentada del circuito
- Imagen con las líneas ajustadas a cada segmento y su punto de corte
- Imagen con los puntos de unión entre segmentos ordenados y con sus coordenadas cartesianas.

Algunos ejemplos de como es la interfaz básica generada se muestran en la Figura 9.12 y la Figura 9.13 .

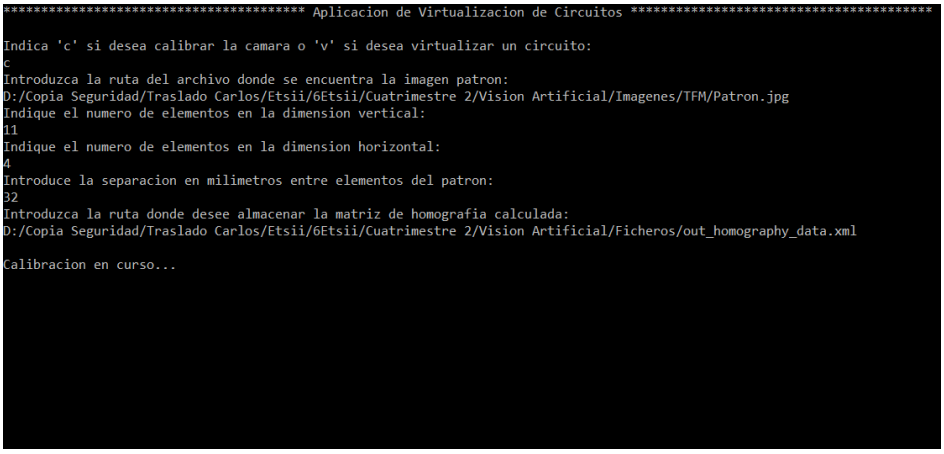


Figura 9.12: Interfaz de cálculo de la homografía para calibración

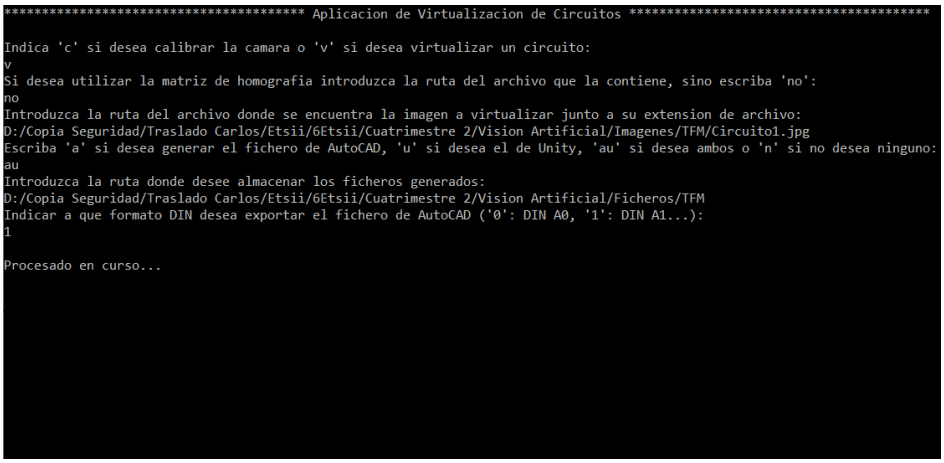


Figura 9.13: Interfaz de virtualizacion de circuitos

9.6 Implementación del circuito digitalizado en Unity

Cuando un usuario entra a la pantalla de visualización de la simulación con robot virtual se encontrará un panel informativo donde se le pedirá que elija entre si quiere que el robot siga un circuito por defecto (Figura 9.14) o bien si quiere cargar un circuito diseñado por él. En el segundo caso se le precisará al usuario que indique la ruta en la que se encuentra el fichero .txt generado por la aplicación de visión artificial.

Si el usuario decide incorporar un nuevo circuito e introduce correctamente la ruta del fichero correspondiente se desencadenarán una serie de funciones que materializarán el circuito en la pantalla. Estas se pueden plasmar en los siguientes pasos:

1. Apertura del fichero seleccionado y lectura línea a línea de las coordenadas de los puntos. Almacenamiento de las coordenadas X e Y por separado en dos variables de tipo lista ordenadas.

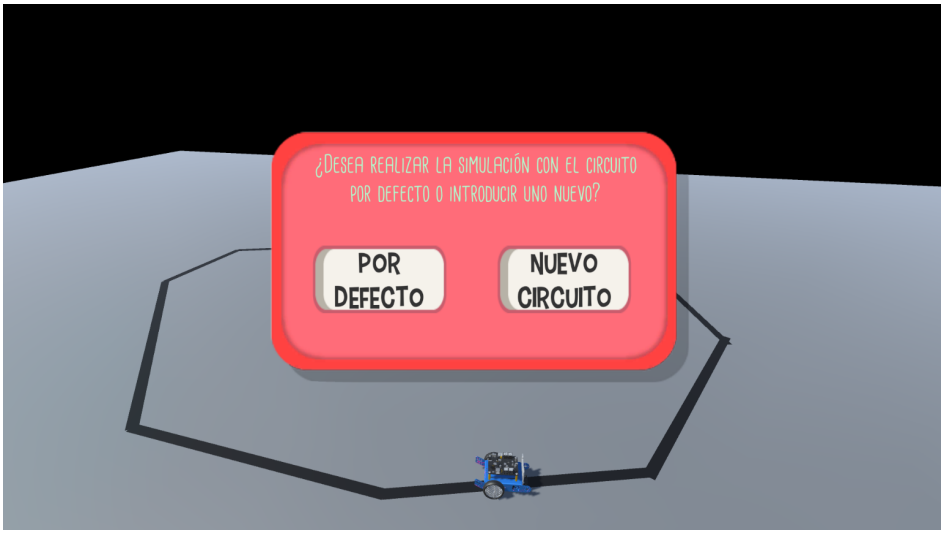


Figura 9.14: Captura del panel de selección del circuito en pantalla de simulación con robot virtual

2. Instanciación de un objeto de tipo renderizado de línea e incorporación a un objeto que la contendrá a ella y a todas las demás que se generen. Ajuste de su grosor y grado de redondeo.
3. Colocación de los puntos inicial y final de la línea para la renderización de esta. El punto inicial para la primera línea será el relativo a las primeras coordenadas de las listas y el final a las siguientes. La coordenada Z en ambos puntos es situada en un valor fijo que permita a la línea verse a través del suelo.
4. Inicialización de la posición local del objeto línea al origen de coordenadas al igual que su rotación. La orientación vendrá dada directamente por la posición de sus puntos iniciales y finales.
5. Adición de un componente "collider" de tipo caja a la línea para poder ser detectada por los rayos que lance el sensor siguelíneas implementado. La colocación en el espacio de este componente requiere precisar su longitud, posición del punto medio, y orientación.
6. Cálculo de la longitud del "collider" como la distancia euclídea entre el punto inicial y final de la línea renderizada. Esto se puede realizar ya que ambos puntos comparten la misma coordenada Z.
7. Cálculo del punto medio del eje imaginario que une ambos puntos inicial y final de la línea.
8. Cálculo del ángulo de giro alrededor de cada eje para el "collider". Este solo será necesario alrededor del eje Z al ser este el eje perpendicular al plano. Este ángulo se calcula con la aplicación de la trigonometría básica la cual se plasma en la Figura 9.15 y la Ecuación 9.11.
9. Este proceso se realizará para cada una de las líneas o segmentos que dicte el fichero leído. Con estos pasos ya estará construido el circuito con todos sus "colliders" colocados correctamente.

10. Colocación del objeto padre de todas las líneas del circuito en la posición correcta para poder ser visualizado en su totalidad. Este proceso se realiza conociendo cual es la coordenada máxima a la que es visible un punto en cada extremo de la pantalla para la configuración de cámara prefijada. Se calculan los puntos en cada dirección más extremos de todas las líneas del circuito los cuales coincidirán con alguno de los puntos iniciales o finales que las conforman. Una vez recopilada esta información se puede situar el centro del objeto padre (Unity calcula automáticamente el centro geométrico de un conjunto de objetos contenidos en otro) en la posición adecuada que permita que ninguno de estos puntos extremos supere los límites de la pantalla. Esto se realiza mediante un proceso iterativo comprobando que punto extremo supera los límites de la pantalla para corregir la posición de todo el circuito hasta que todo el conjunto de líneas sea perfectamente visible.
11. Colocación de los robots en la posición de inicio, esta posición vendrá dada por el nuevo punto medio de la primera de las líneas renderizadas. La orientación de estos robots también coincidirá con el ángulo de giro respecto al eje Z que tenga esta línea.

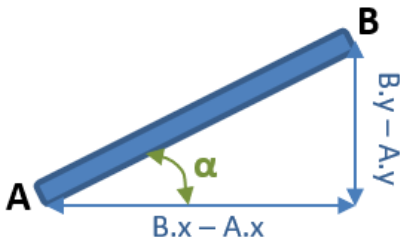


Figura 9.15: Descripción gráfica del ángulo de orientación del segmento

$$\alpha = \arctan\left(\frac{B_y - A_y}{B_x - A_x}\right) \tag{9.11}$$

Tras todo este proceso de generación del circuito virtual, se puede ver en la Figura 9.16 el resultado de la virtualización de este y de la colocación de los robots en él. En este caso ambos robots participantes están superpuestos al tratarse de una instantánea del comienzo de la carrera.

Figura 9.16: Resultado de ejecución del fichero del circuito virtualizado en Unity

Capítulo 10

Resultados y conclusiones del proyecto

«Grande es el arte de comenzar, pero mayor es el arte de concluir» — Henry Wadsworth

El presente proyecto culmina con la formalización de una herramienta educativa completa para servir de apoyo en la educación diabetológica de los niños debutantes. Se considera una propuesta útil que cumple todos los objetivos globales y parciales establecidos al inicio de este.

La herramienta creada en su conjunto trata de facilitar la transmisión de conceptos complejos a niños que poseen una enfermedad como la diabetes donde se debe manejar una cantidad de información muy importante, sobre todo en el debut, para poder llevarla de la mejor manera posible. Se centra sobre todo en la asimilación de conceptos relacionados con la planificación de la administración de terapias correctivas (bolo de insulina o snack) ante un escenario prefijado de ingestas y sesiones deportivas a lo largo de un día común.

Esta herramienta educativa se considera un prototipo que debe ser probado por pacientes y profesionales de la educación diabetológica para decidir si aporta o no beneficios significativos en la asimilación de conceptos por parte del menor. Por ello, se recomienda centrar su utilización en varias sesiones de talleres educativos en diabetes donde esta herramienta sirva de complemento de otras ya utilizadas.

Capítulo 11

Trabajo futuro y propuestas de mejora

«Cuanto más numerosas son las cosas que quedan para aprender, menos tiempo queda para hacerlas» — Marcel Prévost

1. **Ampliación de la herramienta con nuevos juegos:** se podrían incluir nuevos juegos y pantallas tratando otros aspectos de la educación diabetológica. Esto sería sencillo al tratarse de una aplicación multiplataforma donde una actualización solo significaría hacer llegar la nueva versión a los usuarios que la hubiesen descargado.
2. **Pruebas de la herramienta con usuarios finales:** sería muy interesante que tanto los pacientes debutantes en diabetes como los educadores probaran la herramienta y dieran su opinión sobre la utilidad de esta y que mejoras o cambios debería incluir.
3. **Potenciar el uso del robot:** dar mayor visibilidad al robot y que no actúe solo como interfaz, sino que permita cierta interacción con el alumno. Posiblemente el dispositivo robótico utilizado en este proyecto tendría demasiadas limitaciones y sería necesario el uso de un robot con mayores capacidades.
4. **Desarrollar algoritmos para resolver escenarios automáticamente:** mediante técnicas de inteligencia artificial que el sistema fuese capaz de proponer una solución a este maximizando el porcentaje de tiempo en normoglucemia que atravesaría la simulación de ese escenario resuelto.
5. **Mejorar la aplicación de reconocimiento de circuitos mediante visión artificial:** esta debería reconocer y digitalizar circuitos que incluyeran curvas y no debieran estar separadas estas en diferentes segmentos. También se podría incluir todo el algoritmo de procesado y captura de la imagen en la herramienta desarrollada, incluyendo la librería de OpenCV en Unity.
6. **Simplificación del código:** una reestructuración y limpieza de los *scripts* y funciones implementadas podría mejorar la eficiencia de la aplicación y hacerla más liviana.

Parte II

Presupuesto

Capítulo 12

Presupuesto

12.1 Mano de obra

La mano de obra utilizada en este proyecto ha sido la de únicamente un titulado superior en Ingeniería Industrial cuyo precio por hora es de 24.00 €/h.

12.2 Materiales

La mayoría de los materiales utilizados son software cuyo periodo de licencia abarca más tiempo que el ocupado en el desarrollo de este proyecto (ver Tabla 12.1). Por ello, se va a computar para el presupuesto de este proyecto el porcentaje del precio relativo a los meses de utilización suponiendo que el software va a estar en todo momento utilizado al ser un proyecto desarrollado por una sola persona. Este precio se dividirá entre las horas laborables del periodo de ocupación del software para obtener el precio por hora que supondría el uso del programa informático debido al pago de su licencia. Con todo ello en la Tabla 12.2 se calculan los precios por hora de los softwares utilizados en el proyecto teniendo en cuenta que la duración del proyecto es de 4 meses (85 días laborables).

El resto del software utilizado es de código abierto o se ha utilizado su versión gratuita por lo que no computan para el desarrollo del presupuesto del proyecto. Se recoge este en la Tabla 12.3, pues será tenido en cuenta para el computo de la medición relativa a la instalación de todo el software.

Los recursos cuya medida unitaria es la unidad y son de uso exclusivo al proyecto serán computados una vez en su correspondiente unidad de obra relativa a la instalación o montaje de estos.

Los gastos adicionales no medibles fácilmente serán computados como costes directos complementarios en cada unidad de obra mediante la aplicación de un 2 % sobre el subtotal del presupuesto de la unidad de obra.

La unidad monetaria contemplada en todo el presupuesto es el euro (€).

Tabla 12.1: Materiales utilizados en el proyecto

Material/Software	**Descripción**
Ordenador portátil ASUS ROG-GL553VD	Ordenador portátil con procesador Intel®Core™i7,de séptima generación, gráfica dedicada NVIDIA®GeForce®GTX 1050, 250GB de almacenamiento SSD y 1Tb de almacenamiento HDD, 8Gb de memoria RAM y soporte para Microsoft®DirectX®12
Matlab	Licencia anual del software de cálculo y simulación propiedad de MathWorks®
Autodesk AutoCAD 2020	Licencia anual del software de dibujo en 2D y 3D propiedad de Autodesk®
Autodesk Inventor 2020	Licencia anual del software modelado 3D propiedad de Autodesk®
MS Word	Licencia anual del software de edición y creación de documentos tipográficos propiedad de Microsoft®
Robot mBot	Robot móvil basado en Arduino de la compañía Makeblock®. Versión Bluetooth con pack añadido de matriz led de 8x16 pixeles y batería de ion litio recargable
Tableta Samsung S6	Tableta basada en Android de la compañía Samsung®. Pantalla de 10.5 pulgadas, procesador Qualcomm Snapdragon®855 y 6 Gb de memoria RAM
Asset Arduino Bluetooth Plugin	Complemento de Unity para integrar en la aplicación la conectividad Bluetooth entre la herramienta y el dispositivo robótico
Asset Easy Save 3	Complemento de Unity para integrar en la aplicación la conectividad con una base de datos en la nube mediante la subida y descarga de ficheros
Asset Online Account System	Complemento de Unity para integrar en la aplicación la gestión de usuarios a través de una base de datos en la nube

Tabla 12.2: Calculo de los precios unitarios de los materiales utilizados en el proyecto

Material/Software	**Precio licencia o compra**	**Precio para el proyecto**	**Precio por hora**
Ordenador portátil ASUS ROG-GL553VD	1069 € (vida útil de 5 años de uso)	81.49 € (7.62 % de ocupación en el proyecto respecto a su vida útil)	**0.12 €/h** (680 h laborables a lo largo del proyecto con jornada de 8 h)
Matlab	800 € (licencia de un año)	304.93 € (38.11 % de uso en el proyecto respecto al año de licencia de 223 días laborables)	**0.45 €/h** (680 h laborables a lo largo del proyecto con jornada de 8 h)
Autodesk AutoCAD 2020	2227 € (licencia de un año)	848.71 € (38.11 % de uso en el proyecto respecto al año de licencia de 223 días laborables)	**1.25 €/h** (680 h laborables a lo largo del proyecto con jornada de 8 h)
Autodesk Inventor 2020	2747 € (licencia de un año)	1046.88 € (38.11 % de uso en el proyecto respecto al año de licencia de 223 días laborables)	**1.54 €/h** (680 h laborables a lo largo del proyecto con jornada de 8 h)
MS Word	69 € (licencia de un año)	26.30 € (38.11 % de uso en el proyecto respecto al año de licencia de 223 días laborables)	**0.04 €/h** (680 h laborables a lo largo del proyecto con jornada de 8 h)
Robot mBot	104.95 € (unidad)	**104.95 €** (uso exclusivo en el proyecto)	—
Tableta Samsung S6	679 € (unidad)	**679 €** (uso exclusivo en el proyecto)	—
Asset Arduino Bluetooth Plugin	16.97 € (unidad)	**16.97 €** (uso exclusivo en el proyecto)	—
Asset Easy Save 3	44.66 € (unidad)	**44.66 €** (uso exclusivo en el proyecto)	—
Asset Online Account System	17.87 € (unidad)	**17.87 €** (uso exclusivo en el proyecto)	—

Tabla 12.3: Software gratuito utilizado en el proyecto

Software	**Descripción**
Unity 3D	Licencia gratuita del entorno de desarrollo de aplicaciones y videojuegos propiedad de Unity Technologies®
VisualStudio	Licencia gratuita del entorno de desarrollo integrado propiedad de Microsoft®
OpenCV	Librería open source de visión artificial
mBlock	Licencia gratuita del software de programación mediante Scratch propiedad de la empresa Makeblock®
Arduino IDE	Licencia gratuita del entorno de programación propiedad de Arduino®.
LaTeX	Sistema de composición de textos open source

12.3 Precios descompuestos

INS001 **h** **Instalación del software utilizado en el proyecto**

Puesta en marcha del ordenador e instalación de todo el software utilizado para el desarrollo del proyecto

Código	Unidad	Descripción	Rendimiento	Precio Unitario	Importe
1		**Materiales**			
	h	Ordenador portátil ASUS	1.000	0.12	0.12
	h	Matlab	0.600	0.45	0.27
	h	Autodesk AutoCAD 2020	0.600	1.25	0.75
	h	Autodesk Inventor 2020	0.600	1.54	0.92
		Subtotal de materiales:			**2.06**
2		**Mano de obra**			
	h	Ingeniero Industrial	0.800	24.00	19.20
		Subtotal mano de obra:			**19.20**
3		**Costes complementarios**			
	%	Costes complementarios	2.000	21.26	0.43
		Costes directos (1+2+3):			**21.69**

INS002 **Ud** **Instalación e integración de los assets en el proyecto**

Instalación de los assets en el software Unity, entendimiento de ellos e integración en el proyecto.

Código	Unidad	Descripción	Rendimiento	Precio Unitario	Importe
1		**Materiales**			
	h	Ordenador portátil ASUS	2.400	0.12	0.29
	Ud	Asset Arduino Bluetooth	1.000	16.97	16.97
	Ud	Asset Easy Save 3	1.000	44.66	44.66
	Ud	Asset Online Account	1.000	17.87	17.87
		Subtotal de materiales:			**79.79**
2		**Mano de obra**			
	h	Ingeniero Industrial	2.500	24.00	60.00
		Subtotal mano de obra:			**60.00**
3		**Costes complementarios**			
	%	Costes complementarios	2.000	139.79	2.80
		Costes directos (1+2+3):			**142.59**

INS003 **Ud** **Montaje y puesta en marcha del robot**

Ensamblado del robot, actualización de firmware e integración con softwares de programación

Código	Unidad	Descripción	Rendimiento	Precio Unitario	Importe
1		**Materiales**			
	h	Ordenador portátil ASUS	1	0.12	0.12
	Ud	Robot mBot	1.000	104.95	104.95
			Subtotal de materiales:		**105.07**
2		**Mano de obra**			
	h	Ingeniero Industrial	1.500	24.00	36
			Subtotal mano de obra:		**36**
3		**Costes complementarios**			
	%	Costes complementarios	2.000	141.07	2.82
			Costes directos (1+2+3):		**143.89**

INS004 **Ud** **Puesta en marcha de la tableta**

Puesta en marcha de la tableta Samsung e integración con el software Unity

Código	Unidad	Descripción	Rendimiento	Precio Unitario	Importe
1		**Materiales**			
	h	Ordenador portátil ASUS	0.400	0.12	0.048
	Ud	Tableta Samsung S6	1.000	679.00	679.00
			Subtotal de materiales:		**679.05**
2		**Mano de obra**			
	h	Ingeniero Industrial	0.600	24.00	14.4
			Subtotal mano de obra:		**14.4**
3		**Costes complementarios**			
	%	Costes complementarios	2.000	693.45	13.87
			Costes directos (1+2+3):		**707.32**

DOC001 **h** **Documentación y recogida de información**

Proceso de documentación acerca de la diabetes y de la motivación psicológica de la competición educativa

Código	Unidad	Descripción	Rendimiento	Precio Unitario	Importe
1		**Materiales**			
	h	Ordenador portátil ASUS	1.000	0.12	0.12
	h	MS Word	0.400	0.04	0.02
			Subtotal de materiales:		**0.14**
2		**Mano de obra**			
	h	Ingeniero Industrial	1.100	24.00	26.40
			Subtotal mano de obra:		**26.40**
3		**Costes complementarios**			
	%	Costes complementarios	2.000	26.54	0.53
			Costes directos (1+2+3):		**27.07**

DES001 **h** **Programación del robot**

Creación del programa en Arduino que llevará integrado el dispositivo robótico

Código	Unidad	Descripción	Rendimiento	Precio Unitario	Importe
1		**Materiales**			
	h	Ordenador portátil ASUS	1.000	0.12	0.12
			Subtotal de materiales:		**0.12**
2		**Mano de obra**			
	h	Ingeniero Industrial	1.100	24.00	26.40
			Subtotal mano de obra:		**26.40**
3		**Costes complementarios**			
	%	Costes complementarios	2.000	26.52	0.53
			Costes directos (1+2+3):		**27.05**

DES002 **h** **Desarrollo de la aplicación en Unity**

Creación de la interfaz gráfica y programación en C de todos los scripts necesarios

Código	Unidad	Descripción	Rendimiento	Precio Unitario	Importe
1		**Materiales**			
	h	Ordenador portátil ASUS	1.000	0.12	0.12
			Subtotal de materiales:		**0.12**
2		**Mano de obra**			
	h	Ingeniero Industrial	1.100	24.00	26.40
			Subtotal mano de obra:		**26.40**
3		**Costes complementarios**			
	%	Costes complementarios	2.000	26.52	0.53
			Costes directos (1+2+3):		**27.05**

DES003 **h** **Modelado del robot y control en Matlab**

Modelado del comportamiento cinemático del dispositivo robótico e implementación del sensor siguelíneas y el control de su trayectoria en Matlab

Código	Unidad	Descripción	Rendimiento	Precio Unitario	Importe
1		**Materiales**			
	h	Ordenador portátil ASUS	1.000	0.12	0.12
	h	Matlab	1.000	0.45	0.45
			Subtotal de materiales:		**0.57**
2		**Mano de obra**			
	h	Ingeniero Industrial	1.100	24.00	26.40
			Subtotal mano de obra:		**26.40**
3		**Costes complementarios**			
	%	Costes complementarios	2.000	26.97	0.54
			Costes directos (1+2+3):		**27.51**

DES004 **h** **Modelado del robot y control en Unity**

Modelado del comportamiento cinemático del dispositivo robótico e implementación del sensor siguelíneas y el control de su trayectoria en Unity

Código	Unidad	Descripción	Rendimiento	Precio Unitario	Importe
1		**Materiales**			
	h	Ordenador portátil ASUS	1.000	0.12	0.12
	h	Autodesk Inventor 2020	0.200	1.54	0.31
			Subtotal de materiales:		**0.43**
2		**Mano de obra**			
	h	Ingeniero Industrial	1.100	24.00	26.40
			Subtotal mano de obra:		**26.40**
3		**Costes complementarios**			
	%	Costes complementarios	2.000	26.83	0.54
			Costes directos (1+2+3):		**27.37**

DES005 **h** **Diseño e implementación de la aplicación de virtualización de circuitos**

Diseño y programación de la herramienta para la virtualización de circuitos basada en la librería OpenCV e integración con Unity y AutoCAD

Código	Unidad	Descripción	Rendimiento	Precio Unitario	Importe
1		**Materiales**			
	h	Ordenador portátil ASUS	1.000	0.12	0.12
	h	Autodesk AutoCAD 2020	0.200	1.25	0.25
			Subtotal de materiales:		**0.37**
2		**Mano de obra**			
	h	Ingeniero Industrial	1.100	24.00	26.40
			Subtotal mano de obra:		**26.40**
3		**Costes complementarios**			
	%	Costes complementarios	2.000	26.77	0.54
			Costes directos (1+2+3):		**27.31**

RED001 **h** **Redacción del proyecto**

Redacción de la memoria del proyecto y del presupuesto de este

Código	Unidad	Descripción	Rendimiento	Precio Unitario	Importe
1		**Materiales**			
	h	Ordenador portátil ASUS	1.000	0.12	0.12
	h	MS Word	0.200	0.04	0.008
			Subtotal de materiales:		**0.13**
2		**Mano de obra**			
	h	Ingeniero Industrial	1.100	24.00	26.40
			Subtotal mano de obra:		**26.40**
3		**Costes complementarios**			
	%	Costes complementarios	2.000	26.53	0.53
			Costes directos (1+2+3):		**27.06**

12.4 Presupuesto parcial

Código	Unidad	Descripción	Medición	Precio	Importe
MEINS001	**h**	Instalación del software utilizado en el proyecto	4.00	21.69	86.76
			Importe total:		**86.76**

Código	Unidad	Descripción	Medición	Precio	Importe
MEINS002	**Ud**	Instalación e integración de los assets en el proyecto	1.00	142.59	142.59
			Importe total:		**142.59**

Código	Unidad	Descripción	Medición	Precio	Importe
MEINS003	**Ud**	Montaje y puesta en marcha del robot	2.00	143.89	287.78
			Importe total:		**287.78**

Código	Unidad	Descripción	Medición	Precio	Importe
MEINS004	**Ud**	Puesta en marcha de la tableta	2.00	707.32	1414.64
			Importe total:		**1414.64**

Código	Unidad	Descripción	Medición	Precio	Importe
MEDOC001	**h**	Documentación y recogida de información	12.00	27.07	324.84
			Importe total:		**324.84**

Código	Unidad	Descripción	Medición	Precio	Importe
MEDES001	**h**	Programación del robot	10.00	27.05	270.50
			Importe total:		**270.50**

Código	Unidad	Descripción	Medición	Precio	Importe
MEDES002	**h**	Desarrollo de la aplicación en Unity	320.00	27.05	8656.00
			Importe total:		**8656.00**

Código	Unidad	Descripción	Medición	Precio	Importe
MEDES003	**h**	Modelado del robot y control en Matlab	12.00	27.51	330.12
			Importe total:		**330.12**

Código	Unidad	Descripción	Medición	Precio	Importe
MEDES004	**h**	Modelado del robot y control en Unity	25.00	27.37	684.25
			Importe total:		**684.25**

Código	Unidad	Descripción	Medición	Precio	Importe
MEDES005	**h**	Implementación de la aplicación de virtualización de circuitos	35.00	27.31	955.85
			Importe total:		**955.85**

Código	Unidad	Descripción	Medición	Precio	Importe
MERED001	**h**	Redacción y maquetado del proyecto	130.00	27.06	3517.80
			Importe total:		**3517.80**

12.5 Presupuesto de ejecución material

Código	Descripción	Importe (€)
MEINS001	Instalación del software utilizado en el proyecto	**86.76**
MEINS002	Instalación e integración de los assets en el proyecto	**142.59**
MEINS003	Montaje y puesta en marcha del robot	**287.78**
MEINS004	Puesta en marcha de la tableta	**1414.64**
MEDOC001	Documentación y recogida de información	**324.84**
MEDES001	Programación del robot	**270.50**
MEDES002	Desarrollo de la aplicación en Unity	**8656.00**
MEDES003	Modelado del robot y control en Matlab	**330.12**
MEDES004	Modelado del robot y control en Unity	**684.25**
MEDES005	Programación aplicación de virtualización de circuitos	**955.85**
MERED001	Redacción y maquetado del proyecto	**3517.80**
	Presupuesto de Ejecución Material:	**16671.13**

La suma total asciende a: DIECISÉIS MIL SEISCIENTOS SETENTA Y UN EUROS CON TRECE CÉNTIMOS.

12.6 Presupuesto de ejecución por contrata

Descripción	Importe (€)
Presupuesto de Ejecución Material	**16671.13**
Gastos generales 13 %	**2167.25**
Beneficio Industrial 6 %	**1000.27**
Presupuesto total	**19838.65**
I.V.A 21 %	**4166.12**
Presupuesto de Ejecución por Contrata:	**24004.77**

Asciende el presente presupuesto a la expresada cantidad de: VEINTICUATRO MIL CUATRO EUROS CON SETENTA Y SIETE CÉNTIMOS.

Parte III

Anexos

Capítulo 13

Manual de usuario

13.1 Definición

En este documento se pretenden dar las pautas necesarias para que cualquier usuario pueda iniciarse en el uso de la herramienta educativa diseñada. Su uso y desempeño en la herramienta dependerá del perfil de usuario que represente.

13.2 Registro de un nuevo usuario

En la primera pantalla que aparece al iniciar la aplicación se encontrará un botón que permitirá el registro de un nuevo usuario. Se deberá introducir un nombre de usuario y una contraseña para formalizar el registro. Este nombre de usuario no puede ser igual que el de otra persona ya registrada en la herramienta, en ese caso el sistema mostrará una advertencia y no formalizará el registro.

Por defecto, todo nuevo usuario tendrá asignado el perfil de alumno. Si se desea tener permisos de moderador se deberá contactar con los productores de la herramienta para que estos actualicen el listado de usuarios con permisos de moderador.

Una vez realizado el registro correctamente se podrá utilizar la herramienta con normalidad iniciando sesión en ella cada vez que se inicie. Esta solo mostrará las pantallas y funcionalidades asociadas al perfil de usuario logeado.

Se deberá de disponer de conexión a internet en todo momento para el correcto funcionamiento de la herramienta.

13.3 Funcionamiento para perfil de alumno

Un usuario con perfil de alumno que asista a un taller educativo donde se utilice esta herramienta deberá seguir la siguiente secuencia de pasos dentro de ella:

1. Inicio de sesión.
2. Pulsación en el menú del botón de simulación con robot físico.
3. Conexión del dispositivo robótico con la aplicación. Para ello deberá estar previamente emparejados ambos sistemas con el gestor de conexiones propio de cada sistema operativo.
4. Entrada en la sala que el moderador le haya solicitado.
5. Espera en la sala hasta que el moderador envíe el escenario a resolver.
6. Resolución del escenario y pulsación en el botón despegar cuando se finalice esta.
7. Espera a que todos los compañeros del taller presentes en la sala finalicen la resolución del escenario y el moderador de permiso a iniciar la visualización de la simulación.
8. Visualización de la simulación tanto en la aplicación como en el dispositivo robótico físico. Durante la visualización el alumno tendrá la posibilidad de realizar una medida de la glucosa en sangre, pulsando el glucómetro, y de administrar alguna terapia correctiva.

Un usuario con perfil de alumno que desee entrenar en su casa para estar preparado cuando acuda al taller educativo deberá seguir los siguientes pasos:

1. Inicio de sesión.
2. Pulsación en el menú del botón de simulación con robot virtual.
3. Elección de si desea como contrincante al propio sistema o a un amigo. Si desea un amigo deberá introducir el nombre de usuario de este y esperar que el amigo haga lo mismo con su nombre de usuario.
4. Visualización de la simulación a cámara super rápida donde se realizan los cálculos necesarios para poder mostrar la competición entre robots virtuales.
5. Elección de si se desea realizar la carrera entre robots en el circuito por defecto o en uno nuevo. Si se desea uno nuevo se deberá introducir la ruta del dispositivo donde se encuentra el fichero para Unity creado por la aplicación de virtualización de circuitos detallada en apartados posteriores de este manual.
6. Espera a que el amigo en el caso de que se haya seleccionado esta opción finalice la resolución del escenario y la simulación a cámara super rápida.
7. Visualización de la competición entre los robots virtuales.

13.4 Funcionamiento para perfil de moderador

El usuario que utilice la herramienta teniendo permisos de moderador seguirá los siguientes pasos dentro de ella:

1. Inicio de sesión.
2. Entrada en la sala propia que previamente haya creado. También se puede crear una nueva sala siguiente un método muy similar al utilizado para el registro de un nuevo usuario (ver Apartado 13.2 de este manual).
3. En la parte izquierda de la pantalla se podrá visualizar los alumnos presentes en la sala y del estado en la aplicación en el que se encuentran. Existe una actualización automática de esta información cada 8 segundos, si se desea actualizar en cualquier momento se puede pulsar el botón actualizar.
4. Selección del escenario a enviar a los alumnos. Si no hay ninguno cargado se deberá acceder al creador de escenarios y o bien seleccionarlo de los creados anteriormente o crearlo en el momento. Si se desea crear en el momento se accederá a una pantalla donde mediante una serie de listas desplegables podrá conformar los distintos eventos que conformarán el escenario. Si desea modificar un escenario existente se le dará la posibilidad de crear una copia de este y no afectar al original o modificar el escenario seleccionado directamente.
5. Envío del escenario a los alumnos mediante el botón de envío de escenarios. En un texto en la parte superior de la pantalla se indicará que escenario está cargado en la sala y por tanto será enviado a los alumnos.
6. Espera a la resolución de los alumnos del escenario. Se sabrá la finalización de cada uno de ellos pues se mostrará en el panel de información el tiempo que han pasado resolviendo cada uno el escenario.
7. Cuando todos los alumnos presentes en la sala finalicen la resolución del escenario se habilitará para el moderador un botón con el que iniciar la carrera.
8. Cuando todos los alumnos hayan finalizado la visualización de sus simulaciones y por tanto la carrera de robots, se le habilitarán al moderador los botones correspondientes a la selección/creación de escenarios y el relativo al envío de estos.
9. Existirá un botón al lado de cada usuario en el panel de información que el moderador podrá pulsar o no. Estos botones dan acceso a un *report* de los parámetros de la simulación que ha realizado cada alumno.

13.5 Funcionamiento para perfil de superusuario

Si el usuario en posesión y uso de la herramienta educativa disfruta de permisos de superusuario seguirá los siguientes pasos durante su utilización:

1. Inicio de sesión.
2. Visualización de estadísticas acumuladas de uso anonimizadas de los usuarios de la herramienta educativa.

3. En esta pantalla el superusuario tendrá la posibilidad de acceder a la pantalla de registro de moderadores, a la de creación de escenarios predeterminados o continuar en la herramienta como si de un moderador se tratase.
4. En la pantalla de registro de moderadores el superusuario deberá escribir el nombre de usuario del moderador a registrar en el campo de entrado de texto y pulsar sobre el botón de añadir. También, existe la posibilidad de visualizar los usuarios a los que se les ha dotado de permisos de moderador mediante una lista desplegable, esta también da acceso a la revocación de estos permisos seleccionando el nombre de usuario de la lista y pulsando el botón de eliminar.
5. En la pantalla de creación de escenarios predeterminados, el superusuario podrá, de igual modo que los moderadores, crear escenarios para su posterior resolución. La diferencia es que si el superusuario selecciona la casilla de escenario predeterminado (solo accesible a este tipo de usuarios) el escenario que diseñe podrá ser utilizado por todos los moderadores para que lo resuelvan sus alumnos, pero solo modificado por él. Si se crea un escenario predeterminado se le dará la posibilidad al superusuario de resolverlo el mismo para que esta solución sirva en las competiciones con robots virtuales entre el alumno y la máquina.
6. Si se selecciona continuar como moderador el superusuario seguirá el mismo recorrido con exactamente los mismos privilegios y funcionalidades que este. Cuando el superusuario inicie de nuevo sesión en la herramienta volverá a poder elegir que desea hacer entre las opciones mostradas en los puntos anteriores.

Capítulo 14

Manual del programador

14.1 Definición

Con este manual se pretenden dar las indicaciones básicas para poder entender el funcionamiento interno de la herramienta educativa y poder proceder a su modificación. Por ello se dividirá este manual en tres partes, cada una de las cuales hace referencia respectivamente a la programación del robot, la aplicación de Unity, y la programación de la aplicación de virtualización de circuitos.

14.2 Programación del robot

En la herramienta el robot solo actúa como interfaz, esto quiere decir que todos los cálculos y procesamiento es realizado por la aplicación de Unity. El robot, el único procesamiento externo que realiza será el de la lectura del sensor siguelíneas y actuación en consecuencia. Toda esta casuística se ha programado con un bloque "switch" que ejecuta la acción correspondiente a la lectura del sensor.

Externamente la aplicación se comunicará con el robot mediante Bluetooth para ello se ha utilizado el asset Android Bluetooth Plugin el cual, para su configuración básica se necesita indicar el tipo de módulo Bluetooth del que dispone el dispositivo Arduino y el nombre visible de este. Esto implica que actualmente la herramienta está programada para solo conectarse con dispositivos robóticos de la casa Makeblock que incorporen el módulo Bluetooth utilizado en el mBot. En el caso de querer conectar otro dispositivo será necesario cambiar estos parámetros e incluso el asset puede no ser válido si el dispositivo no está basado en Arduino.

La herramienta enviará al robot acciones como comenzar la simulación, entrar en un determinado estado glucémico, pausar la simulación o detenerla. Esto está basado en el envío por el puerto serie de una codificación que entienden ambos dispositivos para acceder a cada uno de los modos o acciones de funcionamiento.

14.3 Programación de la aplicación en Unity

La aplicación ha sido desarrollada en la última versión estable disponible en el momento de la realización de este proyecto (2019.3.7f1). Si se desea modificar esta aplicación en una versión diferente existe la posibilidad de que algunos métodos estén obsoletos y no funcionen por lo que sería necesario la actualización manual de estas funciones y partes que no funcionen para poder partir de un funcionamiento correcto de la aplicación.

En cuanto a organización, cada escena o pantalla creada, dispone de un *script* con el mismo nombre que el controlador principal de la escena. También pueden existir *scripts* adicionales de menor importancia que sean utilizados en la escena. Los scripts están organizados en carpetas que identifican a que pantalla pertenecen.

Todas las imágenes incorporadas en la aplicación han sido transformadas al tipo nativo de Unity Sprite 2D para asegurar una correcta incorporación de estas. Estas, están almacenadas en la carpeta "Textures" la cual, a su vez esta subdividida en otras carpetas más identificadoras de la pantalla en la que se usan esas imágenes.

Por tanto, en el árbol principal del proyecto aparecerán las siguientes carpetas:

- Animaciones (como por ejemplo la del avión moviéndose en las pantallas).
- Una relativa a cada uno de los tres *assets* incluidos en el proyecto. En ellas se almacenan los *scripts* que incorporan estos *assets*. Algunos de ellos han sido ligeramente modificados para dotar de otras funcionalidades incorporadas en la herramienta. Por ejemplo, la discriminación por perfil de usuario en el momento de iniciar sesión o que haya que abrir el puerto serie del Bluetooth cada vez que se cambia de pantalla.
- Fuentes, en ella se almacenan las fuentes de texto utilizadas en la aplicación.
- Materiales, se encuentran en ella los materiales utilizados en la aplicación como por ejemplo los aplicados al modelo 3D del robot virtualizado.
- Prefabs, en esta carpeta se encuentran aquellos objetos que se instanciarán en repetidas ocasiones mediante código como por ejemplo las fichas de eventos en el panel de report.
- Escenas, se almacenan aquí los ficheros propios de Unity que definen una escena, es decir que imágenes incorpora y como están colocadas, que objetos existen en la escena y como están relacionados entre ellos, que *scripts* y en que objetos intervienen.
- *Scripts*, es una carpeta que incorpora otras carpetas donde están organizados todos los *scripts* en C# que conforman la herramienta.
- Texturas, esta es la última carpeta y en ella se encuentran todas las imágenes utilizadas en la aplicación en formato Sprite 2D.

14.4 Programación de la aplicación de virtualización de circuitos

Para poder modificar esta aplicación se deberá tener instalado y configurado VisualStudio para poder utilizar funciones de la librería OpenCV que habrá que tener también instalada. La programación de la aplicación se divide en dos partes, una relativa a la calibración de la cámara y otra a la propia virtualización del circuito.

De esta forma en la calibración de la cámara se ha configurado los parámetros para ser calibrada con un patrón de círculos asimétricos de 4x11. En el caso de querer utilizar otro tipo de patrón se deberá modificar la función de reconocimiento de objetos y el bundle de parámetros de entrada a ella con los parámetros específicos del nuevo patrón. La matriz inversa de la matriz de homografía resultante de la calibración será almacenada en un fichero XML en la ruta especificada al inicio del código.

En la virtualización de circuitos la aplicación ha sido diseñada para solo soportar circuitos dibujados en un folio tamaño A4 dividido en segmentos rectos y con un buen contraste con el fondo. Cualquier imagen que no cumpla estos requisitos puede no virtualizarse correctamente.

La aplicación segmentará la imagen, detectará sus contornos y puntos de corte entre segmentos y los ordenará siguiendo la secuencia que conforma el circuito. Como salida la aplicación creará dos ficheros que se almacenarán en la ruta que el usuario indique mediante la interfaz. Sobre todo, el fichero que incorpora los comandos de AutoCAD a ejecutarse automáticamente está probado en la versión 2019 por lo que no se garantiza su correcto funcionamiento en otra versión donde los comandos utilizados puedan ser diferentes.

Para la correcta definición de los puntos que conforman el circuito, la aplicación está programada de forma que pida al usuario que indique la ruta donde se encuentra el fichero XML que almacena la matriz inversa de homografía.

Bibliografía

[1] Medline Plus. U.S. National Library of Medicine. *Diabetes*. Disponible en: https://medlineplus.gov/spanish/ency/article/001214.htm. Accedido 15-04-2020. 2017 (vid. págs. 7-10).

[2] Organización Mundial de la Salud. *Informe mundial sobre la diabetes*. Disponible en: http://apps.who.int/iris/bitstream/handle/10665/254649/9789243565255-spa.pdf?sequence=1. Accedido 15-04-2020. 2016 (vid. pág. 7).

[3] Federación Española de Diabetes. *Las cifras de diabetes en España*. Disponible en: https://www.fedesp.es/portal/1/main$_n$*oticias.aspx*?*idnoticia* = 2306*idportal* = 1. Accedido 15-04-2020. 2014 (vid. págs. 7, 8).

[4] Torres M Garrido R. *Protocolos de urgencias pediátricas*. 2ª ed. Barcelona; 75-81. Capítulo 8. Urgencias endocrinas: Diabetes. Disponible en: https://www.aeped.es/sites/default/files/documentos/diabetes.pdf. Accedido 18-04-2020. 2010 (vid. pág. 10).

[5] Barrio Castellanos R. *Lo que debes saber sobre la diabetes en la edad pediátrica*. 3ª ed. Sociedad Española de Endocrinología Pediátrica. Disponible en: http://www.seep.es/privado/gdiabetes/libro$_d$*iabetes*$_i$*nfantil.pdf*. Accedido 19-04-2020. 2008 (vid. págs. 10-14).

[6] López-de-Andrés Ana Montilla-Pérez Manuel Mena-López Natalia. *Efectividad de la educación diabetológica sistematizada en niños que debutan con Diabetes Mellitus tipo 1*. Disponible en: http://dx.doi.org/10.4321/S1132-12962012000100005. Accedido 19-04-2020. 2012 (vid. pág. 14).

[7] Jansà M Vidal M. *Entrenamiento del paciente y de la familia en el cálculo de raciones de hidratos de carbono. Av Diabetol*. Disponible en: http://www.avancesendiabetologia.org/gestor/upload/revistaAvances/22-4-4.pdf. Accedido 20-04-2020. 2006 (vid. pág. 14).

[8] López Ramírez. *Aprendizaje con robótica, algunas experiencias. Revista Educación*. Disponible en: http://ucsj.redalyc.org/articulo.oa?id=44028564003>ISSN 0379-7082. Accedido 20-04-2020. 2013 (vid. pág. 14).

[9] Yu-Kai Chou. *The complete gamification framework*. Disponible en: https://yukaichou.com/gamification-examples/octalysis-complete-gamification-framework/. Accedido 21-04-2020. 2014 (vid. pág. 18).

[10] Foncubierta JM. *Didáctica de la gamificación en la clase de español*. Disponible en: https://scholar.google.es/citations?user=OshUJxgAAAAJhl. Accedido 21-04-2020. 2014 (vid. pág. 18).

[11] Hamari Juho. *Social motivations to use gamification: An empirical study of gamifying exercise*. Disponible en: https://www.researchgate.net/publication/236269293_Social_motivations_to_use_gamification_An_empirical_study_of_gamifying_exercise. Accedido: 22-04-2020. 2013 (vid. pág. 18).

[12] Simoes. *A social gamification framework for a K-6 learning platform*. Disponible en: https://www.researchgate.net/publication/230854806_A_social_gamification_framework_for_a_K-6_learning_platform. Accedido: 22-04-2020. 2013 (vid. pág. 18).

[13] F Ramón-Cortés. *Cuidado con la obsesión por ganar*. El País Semanal. Disponible en: https://elpais.com/diario/2011/09/11/eps/1315722413_850215*.html*. Accedido: 22-04-2020. 2011 (vid. pág. 18).

[14] J Durán. *Ética de la competición deportiva: Valores y contravalores del deporte competitivo*. Disponible en: https://www.upo.es/revistas/index.php/materiales_historia_deporte/article/view/803/655. Accedido: 22-04-2020. 2013 (vid. pág. 19).

[15] Roman Hovorka. *Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes*. Disponible en: http://www.stat.yale.edu/ jtc5/diabetes/Nonlinear ModelPredictiveControl$_{H}ovorka_{0}4.pdf$. Accedido: 25-04-2020. 2004 (vid. págs. 25, 26).

[16] Toshinori Kimura. *On Dormand-Prince Method*. Disponible en: http://depa.fquim.unam.mx/amyd/archivero/DormandPrince$_{1}9856.pdf$. Accedido: 25-04-2020. 2009 (vid. pág. 26).

[17] *mBot Bluetooth*. Disponible en: https://www.makeblock.es/productos/robot_educativo_mbot/? Accedido: 28-04-2020 (vid. págs. 32, 33).

[18] *Sensor siguelíneas del robot mBot con mBlock*. Disponible en: https://www.programoergosum.com/cursos-online/robotica-educativa/. Accedido: 28-04-2020 (vid. pág. 33).

[19] *Módulo Bluetooth HC06*. Disponible en: https://www.prometec.net/bt-hc06/. Accedido: 28-04-2020 (vid. pág. 34).

[20] *mBlock*. Disponible en: http://www.mblock.cc/. Accedido: 28-04-2020 (vid. pág. 36).

[21] *What ii Arduino?* Disponible en: https://www.arduino.cc/en/Guide/Introduction. Accedido: 28-04-2020 (vid. pág. 36).

[22] Asociación Diabetes Madrid. *Hipoglucemia e hiperglucemia, ¿cómo se presenta y qué debo hacer?* Disponible en: https://diabetesmadrid.org/hipoglucemia-e-hiperglucemia-como-se-presenta-y-que-debo-hacer/. Accedido: 30-04-2020 (vid. págs. 37, 38).

[23] *Te damos la bienvenida a Unity.* Disponible en: https://unity.com/es/our-company. Accedido: 30-04-2020 (vid. pág. 41).

[24] *C#. Qué es y para qué se utiliza.* Disponible en: https://negociosyestrategia.com/blog/que-es-csharp/. Accedido: 01-05-2020 (vid. pág. 42).

[25] *VisualStudio.* Disponible en: https://visualstudio.microsoft.com/es/. Accedido: 01-05-2020 (vid. pág. 42).

[26] Tall Guy Productions. *Online Account System.* Disponible en: https://assetstore.unity.com/packages/tools/network/online-account-system-18487. Accedido: 03-05-2020 (vid. pág. 45).

[27] Moodkie. *Easy Save The Complete Save Load Asset.* Disponible en: https://assetstore.unity.com/packages/tools/input-management/easy-save-the-complete-save-load-asset-768. Accedido: 03-05-2020 (vid. pág. 45).

[28] Tony Abou Zaidan. *Arduino Bluetooth Plugin.* Disponible en: https://assetstore.unity.com/packages/tools/input-management/arduino-bluetooth-plugin-98960. Accedido: 03-05-2020 (vid. pág. 51).

[29] Mellado Arteche Martin. *Rmóvil Cinemática Diferencial UPV.* Disponible en: https://www.youtube.com/watch?v=SirC01o7kx8. Accedido: 05-05-2020 (vid. pág. 78).

[30] *Matlab.* Disponible en: https://es.mathworks.com/products/matlab.html. Accedido: 05-05-2020 (vid. pág. 79).

[31] Edison Sasig. *Robots Móviles Autónomos 1 : Modelo Cinemático y Simulación Robot Diferencial.* Disponible en: https://www.conectados-online.com/proyectos-robotica/. Accedido: 05-05-2020 (vid. pág. 79).

[32] Wiktor Rott. *Line Follower PID.* Disponible en: https://github.com/vvrvvd/Line-Follower-PID. Accedido: 05-05-2020 (vid. pág. 83).

[33] *Autodesk Inventor.* Disponible en: https://www.autodesk.es/products/inventor/overview. Accedido: 09-05-2020 (vid. pág. 85).

[34] Nvidia. *Vehicles Nvidia PhysX.* Disponible en: https://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/Vehicles.html. Accedido: 06-05-2020 (vid. pág. 88).

[35] *C++ A Brief Description.* Disponible en: https://www.cplusplus.com/info/description/. Accedido: 14-05-2020 (vid. pág. 93).

[36] *OpenCV*. Disponible en: https://opencv.org/about/. Accedido: 14-05-2020 (vid. pág. 94).

[37] Antonio J. Sánchez-Salmerón. *OpenCV*. Disponible en: https://poliformat.upv.es/portal/site/DOC_33780_2019/ tool/2229deca-cf3e-49da-b12d-3abb20ae2ba3. Accedido: 14-05-2020 (vid. págs. 96, 97).

[38] *Autodesk AutoCAD*. Disponible en: https://www.autodesk.es/products/autocad/overview?plc=ACDISTterm=1-YEARsupport=ADVANCEDquantity=1. Accedido: 17-05-2020 (vid. pág. 102).