Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería electrónica industrial y automática

Modelado, simulación y desarrollo de una práctica docente con el robot ABB irb 120 basada en el juego “Plinko”

Autor:
Diez Rey, Francisco Javier

Tutor(es):
Fraile Marinero, Juan Carlos

Departamento de Ingeniería de sistemas y automática

Valladolid, Junio de 2022.

Agradecimientos

Quiero agradecer todo el apoyo y ayuda que he recibido de mi familia y amigos, en especial de mis padres que han dedicado mucho tiempo y esfuerzo en formarme profesional y personalmente. También quiero agradecer a todos los que durante mi paso por la Escuela de ingenierías industriales de Valladolid han despertado mi curiosidad y ganas de seguir aprendiendo.

Resumen

Este trabajo de fin de grado consiste en la elaboración de un proyecto para su utilización como material docente. Está conformado por un ABB irb120 con un controlador IRC5, el cual se relaciona con un espacio de trabajo a través de entradas y salidas digitales. Además, se comunica con unos microcontroladores ESP32 mediante protocolo de comunicación TCP-IP a través de la red de la escuela.

La práctica se basa en el juego del Plinko, en el cual el robot se encarga de coger una bola del plano inclinado y soltarla en la parte superior del tablero con los clavos. Entre las funciones del robot se encuentra la comunicación con los microcontroladores, los cuales son capaces de gobernar el ABB y de gestionar unos displays. Por último, se añade un estudio estadístico para determinar la probabilidad que existe de que la bola caiga en cada casillero antes de ser soltada.

Palabras clave (key words)

Simulación entornos robóticos, RobotStudio, RAPID, práctica docente.

Abstract

This final degree project consists of the elaboration of a project for use as teaching material. It consists of an ABB irb120 with an IRC5 controller, which relates to a workspace through digital inputs and outputs. In addition, it communicates with ESP32 microcontrollers using TCP-IP communication protocol over the school network.

The practice is based on the game of Plinko, in which the robot is responsible for taking a ball from the inclined plane and releasing it at the top of the board with the nails. Among the functions of the robot is the communication with the microcontrollers, which can govern the ABB and manage displays. Finally, a statistical study is added to determine the probability that the ball will fall into each box before being released.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. Introducción y objetivos

Este proyecto se centra en la utilización de un robot ABB IRB120, disponible en el laboratorio de robótica de la Escuela de Ingenierías Industriales de la Universidad de Valladolid, para desarrollar una maqueta que permita realizar el juego del Plinko en un entorno académico

El robot IRB120 es el robot industrial antropomórfico más pequeño de ABB, ya que solo pesa 25 Kg y soporta una carga de 3 Kg.

Este robot ya ha sido utilizado en otros proyectos para desarrollar prácticas docentes. En la figura 3 se muestra el escenario creado para utilizar el robot IRB120 para tareas de manipulación y pegado de piezas [3].

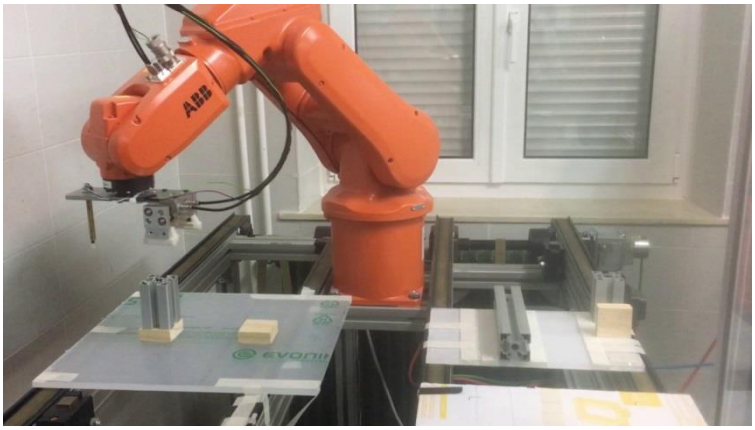


Ilustración 1: Robot IRB120 de ABB - Ejercicio de manipulación y pegado de piezas. **[3].**

Para ello hace uso de una herramienta compuesta por una pinza y un láser. Este simula la aplicación del adhesivo. Las trayectorias que realiza dependen de la pieza que deba ser pegada.

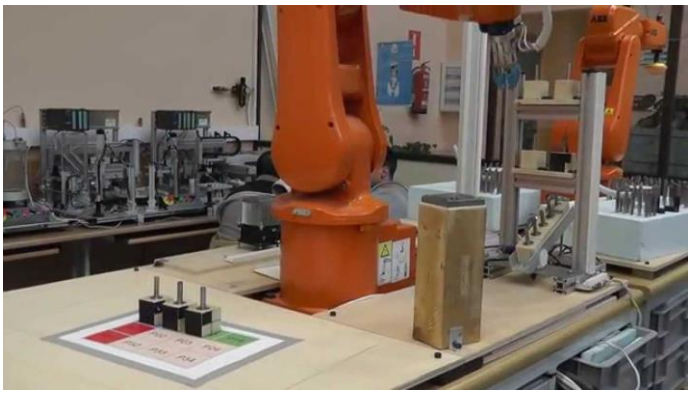


Ilustración 2: PICK & PLACE con ROBOT INDUSTRIAL ABB IRB120.**[4].**

En la figura 4 se muestra otra aplicación del robot IRB 120 para el reposicionamiento de las piezas, que son suministradas mediante un plano inclinado, en las distintas casillas marcadas en el folio.

Ilustración 3: Trabajo de fin de grado de Miguel Rodríguez López **[5].**

En la figura 5 se muestra la práctica docente realizada en el TFG de Miguel Rodríguez López [5], en la Escuela de Ingenierías Industriales de la Universidad de Valladolid. En esta práctica, dos torres verticales suministran bolas, que, de forma aleatoria, caen en las distintas casillas del tablero. El tablero está sensorizado, para que el robot "conozca" donde ha caído la bola, cogerla y recolocarla en una de las torres.

Este proyecto es en el cual más me he inspirado ya que presenta elementos muy interesantes como los accionamientos neumáticos y la aleatoriedad.

El objetivo del proyecto es el desarrollo de una estación de simulación utilizando el software Robotstudio, y la construcción de varias maquetas con las que interactuará el robot ABB IRB120, para implementar el juego del Plinko en un entorno académico de prácticas con robot.

En la **Ilustración 3** se muestra la maqueta de la práctica académica que ha sido desarrollada en este TFG.

Ilustración 4: Estación de trabajo del ABB irb120.

El espacio de trabajo desarrollado para esta práctica tiene los siguientes elementos:

1. Un depósito de bolas, denominado a lo largo del proyecto plano inclinado, donde se almacenarán 4 bolas.
2. Un pistón neumático para "sacar" la bola fuera del plano inclinado, y posicionarla en un lugar más accesible para que el robot pueda coger dicha bola.
3. Una electroválvula que active el pistón neumático.
4. Un sensor conformado por un diodo y un fototransistor que sirva para indicar que hay alguna bola en el plano y se puede activar el pistón.
5. Una plataforma del juego plinko.
6. Cinco finales de carrera que sirvan para identificar en que casillero ha caído una bola.
7. Un robot ABB IRB120

En un inicio todas las bolas se encuentran en el plano inclinado, cuyo sensor indica que hay una bola dispuesta para ser empujada por el pistón.

Al indicar el inicio del programa el robot activa la electroválvula haciendo avanzar el vástago del pistón, y empujando la bola hasta una posición en la que será recogida por el robot.

Tras esta activación el robot recoge la bola y la posiciona en un lugar aleatorio de la parte superior del plinko. Al abrir la pinza del robot, la bola cae por la plataforma inclinada del "plinko", y se posiciona en uno de los 5 casilleros. Esto hace que se active una entrada digital del robot, indicando a que posición debe dirigirse la pinza del robot para recoger la bola que acaba de caer.

Tras recoger la bola este se dirige a la parte superior del plano inclinado y la suelta almacenándola en este, repitiendo el ciclo.

Además de este proceso se ha añadido valor a este TFG mediante la incorporación de unos displays gobernados por un microcontrolador. Este haciendo uso del protocolo TCP/IP y mediante sockets, se comunica con el ABB IRB120, mostrando en los displays el recuento de bolas que han caído en los distintos casilleros, y la estimación de la probabilidad de que caiga la bola en cada casillero, tras conocer la posición desde donde se deja caer la bola.

Este microcontrolador se comunica con otro microcontrolador igual, mediante comunicación serial. Este segundo microcontrolador permite gestionar una página web desde la cual se pueden controlar los distintos modos de funcionamiento de la plataforma docente, ofreciendo al usuario una interfaz sencilla y de fácil manejo.

Este proyecto está constituido de cinco partes:

1. Modelado y simulación de la estación para el juego del Plinko. Esta parte se constituye del modelado de la estación y la programación de los componentes inteligentes.
2. Programación del ABB, los dos microcontroladores y su comunicación.
3. Diseño de los componentes electrónicos y su construcción.
4. Fabricación de las maquetas y conexionado de los sensores y actuadores.
5. Registro y estudio estadístico.

Para alcanzar este objetivo se han realizado los siguientes pasos.

- Brainstorming junto al tutor para plantear un proyecto lo más completo posible con el fin de conocer todas las posibilidades del ABB-irb120.
- Desarrollo del espacio de trabajo en Robotstudio y evaluación de posibles mejoras o añadidos en la realización de las maquetas.
- Búsqueda de información y estudio de componentes inteligentes con Robotstudio los cuales permite simular el comportamiento real del proceso.
- Programación del proceso principal.
- Estudio del diseño en 3D para la realización de las maquetas lo más similares posibles a las creadas en Robotstudio.
- Diseño de las maquetas en madera y diseño en 3D de aquellas piezas cuya realización en madera presenta mucha complejidad tanto por la precisión requerida como por el conformado.
- Realización de un primer prototipo de las maquetas en madera para corregir posibles errores en el planteamiento inicial.
- Evaluación junto al tutor de los avances realizados con el fin de mejorar o añadir elementos tanto a las maquetas como al espacio de trabajo en Robotstudio.

- Practicas con el ABB irb-120 para conocer su funcionamiento con la flexpendant, los riesgos y precauciones en su manejo además de cómo se realizan los ajustes entre la realidad y la simulación.
- Posicionamiento de las maquetas en el espacio de trabajo y ajuste de las posiciones que debe alcanzar el TCP del robot para llegar a los distintos puntos críticos de proceso.
- Implantación del proceso principal sin colocar ningún sensor ni actuador y actuando sobre el sistema mediante la botonera que simula los sensores y muestra las distintas activaciones de las salidas.
- Modificación de las maquetas para evitar problemas con el ABB que no se encontraban en la simulación.
- Simplificación de la trayectoria.
- Programación de algunos modos de funcionamiento: Posición en el Plinko y movimiento manual.
- Programación de los microcontroladores y desarrollo en una protoboard siendo la comunicación mediante un router domestico entre el microcontrolador y la simulación en Robotstudio.
- Desarrollo del sensor, que se encuentra en el plano inclinado, en una PCB ya que la bola no es capaz de activar el final de carrera.
- Desarrollo de dos PCBs para la implantación de los microcontroladores y los displays y la maqueta donde se alojan.
- Corrección de errores en la programación y diseño del circuito conformado por los dos microcontroladores.
- Implantación de los sensores en las maquetas.
- Conexionado de los sensores y actuadores a una DB25 conforme las entradas y salidas programas.
- Conexionado neumático.
- Modificación de las maquetas con el fin de hacerlas más estéticas.
- Implantación del programa final y comprobación de su correcto funcionamiento.
- Realización de un registro con la posición en el Plinko donde se soltaba la bola y el casillero en el que caía con el fin de realizar un estudio estadístico.
- Evaluación del registro y realización de un estudio en el cual se establecen intervalos de confianza del 90% con errores de probabilidad de hasta un 25%.
- Comunicación entre el microcontrolador y el ABB mediante las redes de la escuela de ingenierías industriales.
- Implantación y validación final de proyecto junto al tutor.

2. Modelado y simulación de la estación en Robotstudio para el juego Plinko

2.1 Creación del robot y la pinza en la estación virtual

Para crear la estación virtual se comienza seleccionando el robot que se va a utilizar para ello se selecciona la pestaña “Posición inicial” y se selecciona Biblioteca ABB para que parezca el catálogo de equipos de ABB que están disponibles en Robotstudio.

Para este proyecto se selecciona el ABB irb120 que es el cual se encuentra en el laboratorio de robótica y con el que se realiza este proyecto.

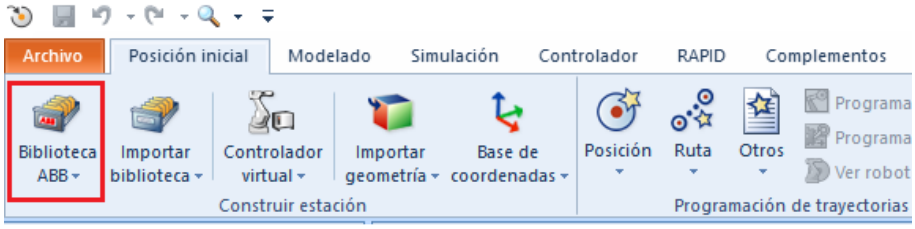


Ilustración 5: Menú superior.

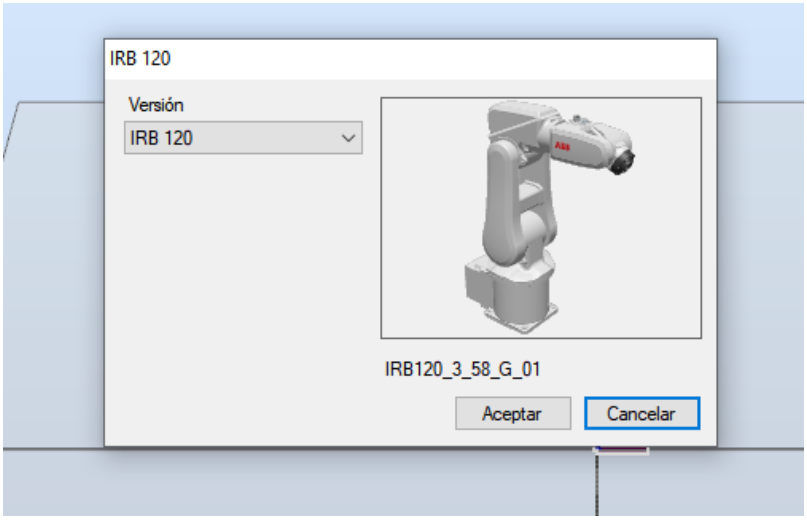


Ilustración 6: Selección del robot ABB irb120.

El robot traído de la librería no dispone de herramienta por lo que se la ha tenido que importar.

La herramienta utilizada es una pinza neumática realizada por el departamento de ingeniería de sistemas y automática por lo cual se ha importado de una estación creada por este departamento para la docencia de Robotstudio y Rapid.

La herramienta importada no exactamente igual que la herramienta utilizada en la realidad debido a que se le debieron añadir dos almohadillas a los lados de la pinza para conseguir que esta agarrase las bolas.

Para asignar al ABB de Robotstudio la herramienta final que debía utilizar se partió de la pinza creada por el departamento de sistemas y automática aprovechando parte de la morfología creada además de la programación del componente inteligente.

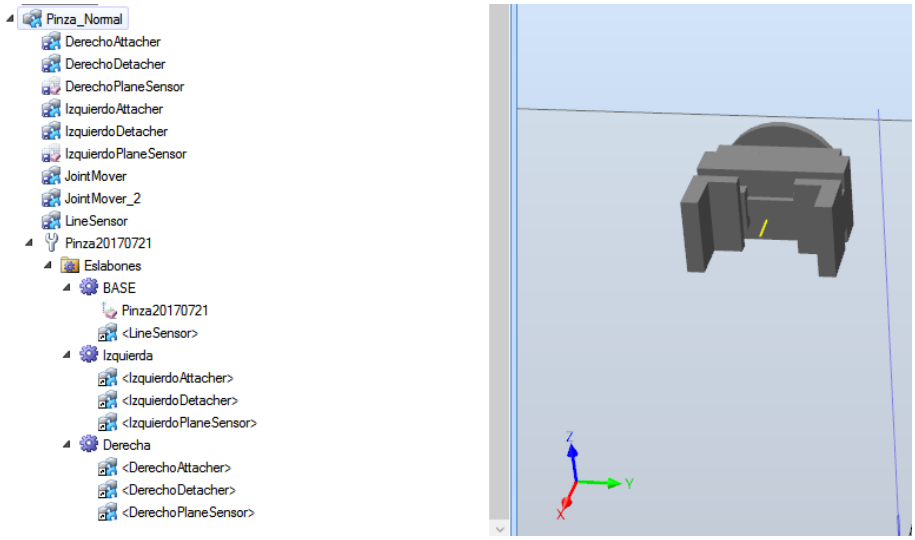


Ilustración 7: Herramienta Pinza_Normal.

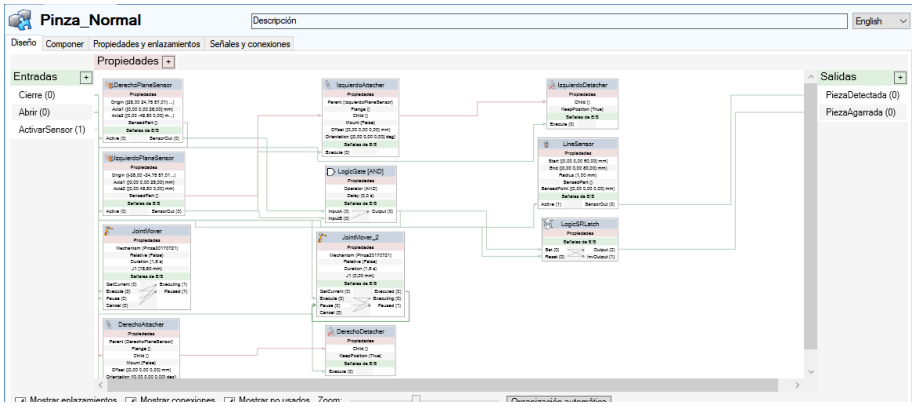


Ilustración 8: Componente inteligente de Pinza_Normal.

Este componente inteligente presenta tres entradas:

- Cierre: Representa la orden del ABB de que se junten los eslabones.
- Abrir: Representa la orden del ABB de que se separen los eslabones.
- Activar sensor: Se encuentra constantemente activado y solo es para que se active un sensor representado en el componente inteligente que no se encuentra en la realidad.

Como salidas se encuentran Pieza detectada y Pieza agarrada las cuales no se encuentran conectadas a la estación.

El único eslabón que no requiere ninguna modificación es el eslabón Base.

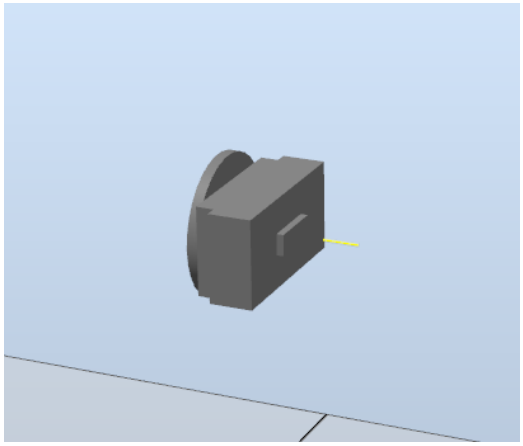


Ilustración 9: Eslabón Base de Pinza_Normal.

Los eslabones Izquierda y derecha fueron modificados añadiendo grosor a sus paredes laterales y modificando la posición de los sensores planares programados con “SmartComponents”.

Se crearon utilizando la herramienta de Modelado de Robotstudio los eslabones siendo estos casi idénticos a los de la Pinza_Normal.

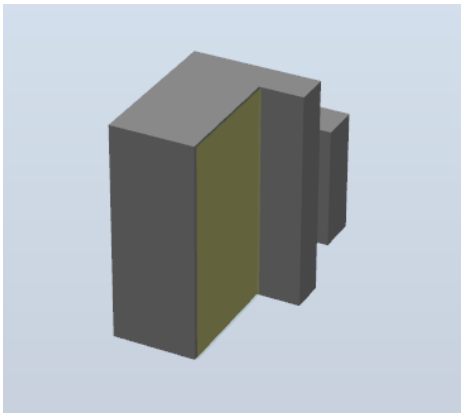


Ilustración 10: Eslabón Izquierdo de Pinza_Ancha.

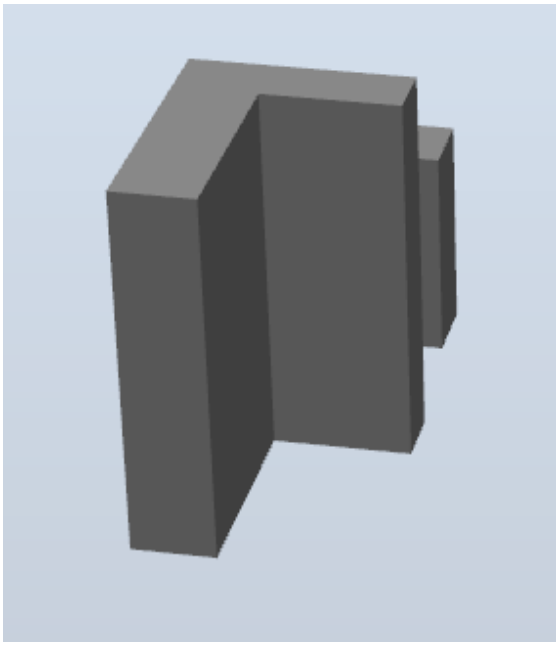


Ilustración 11: Eslabón Izquierdo de Pinza_Normal.

El eslabón derecho es simétrico a este por lo que solo fue necesario la realización de un cuerpo con la herramienta de modelado.

Para que los sensores planares izquierdo y derecho detectasen la bola se modificó la posición de estos en la programación de componentes inteligentes.

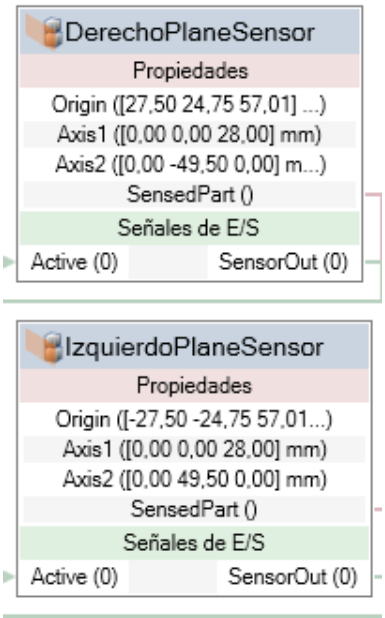


Ilustración 12: Posición sensores planares en la Pinza_Ancha.

Tras crear una estación virtual con el ABB irb120 y su herramienta, Pinza_Ancha, se procede a crear los elementos que conforman el espacio de trabajo.

El espacio de trabajo debe encontrarse sobre un tablero móvil el cual podrá ser colocado y retirado en cualquier momento, para representarlo en RobotStudio se selecciona la pestaña Modelado, hacer click sobre sólido y dentro de las opciones seleccionar tetraedro.

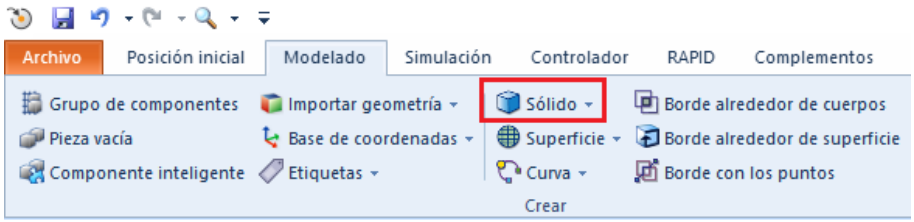


Ilustración 13: Menú superior selección de sólido.

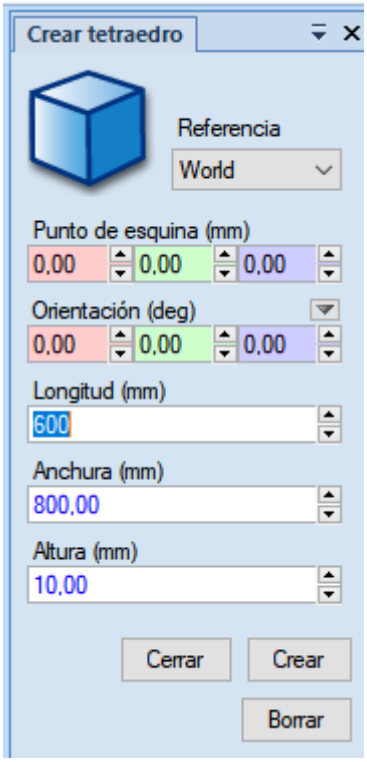


Ilustración 14: Creación del tablero móvil.

El tablero móvil como se muestra en la **Ilustración *14*** presenta unas medidas de 600x800x10 mm.

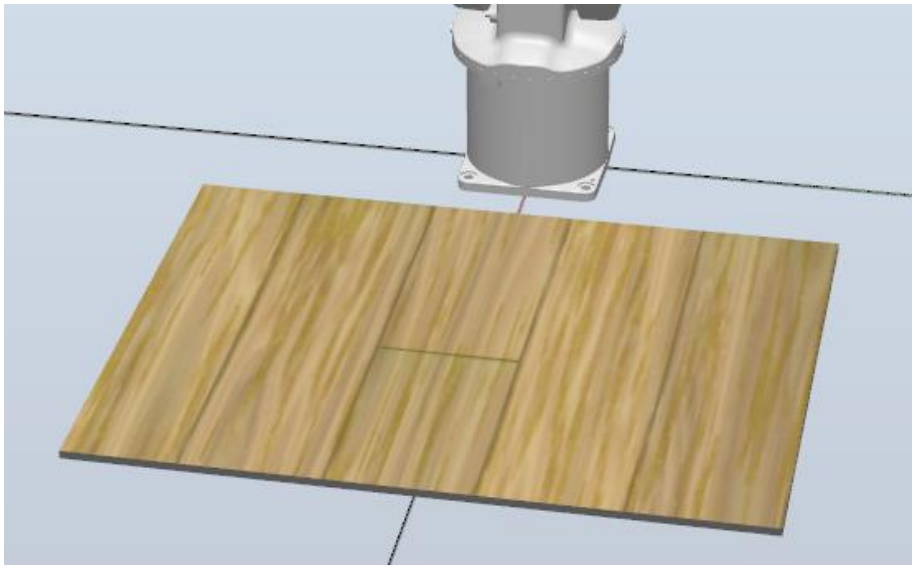


Ilustración 15: Tablero móvil.

2.2 Modelado del depósito de bolas.

Sobre el tablero móvil se sitúan dos maquetas. Una primera es un plano inclinado que sirve como depósito de bolas donde estas son almacenadas y dispensadas al ABB irb120. La otra maqueta es el Plinko.

El modelado del plano inclinado se ha realizado mediante la suma y resta de cuerpos.

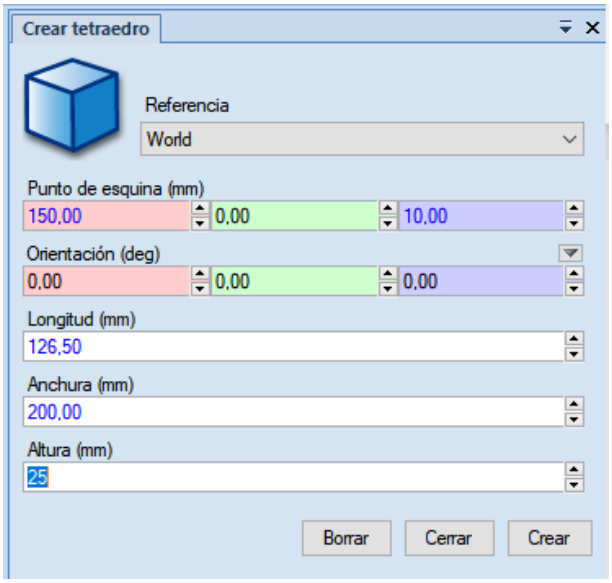


Ilustración 16: conformado de la base del plano inclinado.

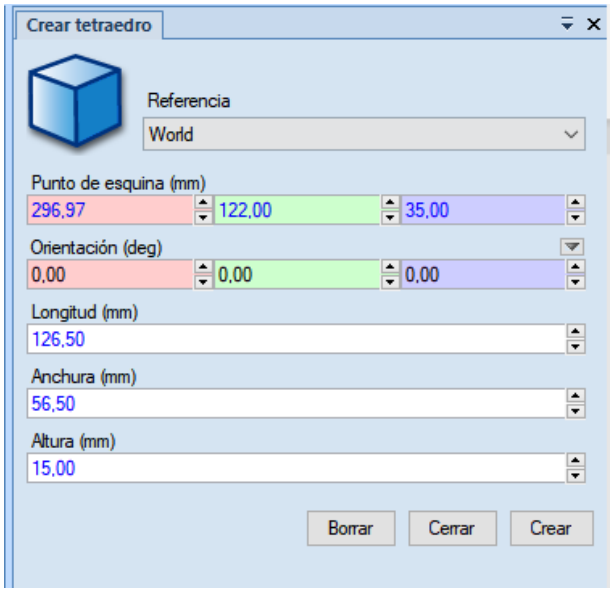


Ilustración 17: Creación del carril dispensador de bolas.

Con las dos piezas creadas como se muestra en las **Ilustración *16***e **Ilustración *17*** se puede ver como se ha ido creado el plano inclinado.

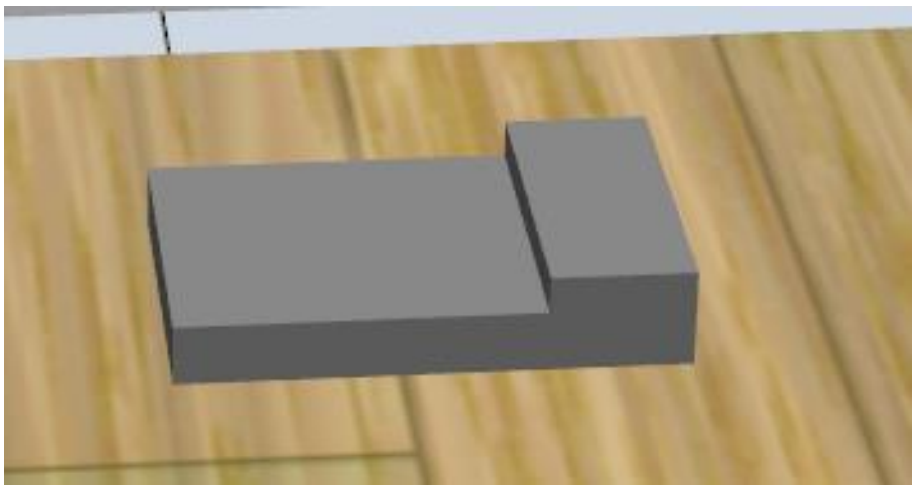


Ilustración 18: Creación del carril dispensador de bolas.

Para poder hacer el carril se utiliza una herramienta para restar cuerpos como se muestra a continuación.

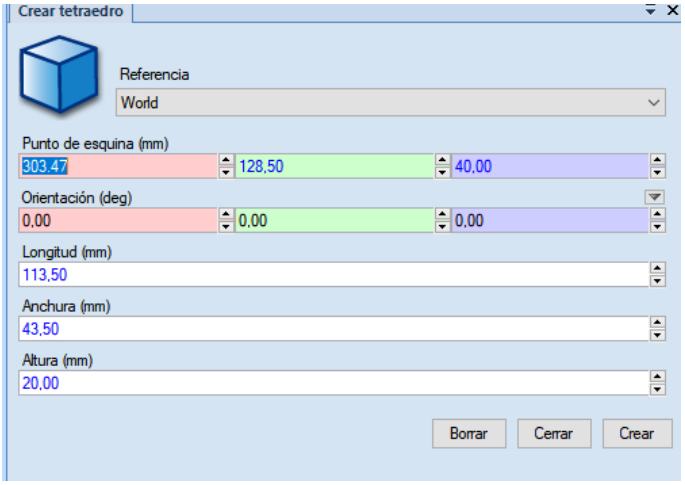


Ilustración 19: Creación del cuerpo que resta al anterior.

Tras crear el cuerpo que resta al anterior se selecciona la herramienta de resta y se indican los cuerpos que se quien restar en este caso, se quiere restar un cuerpo de la pieza 3 con el cuerpo de la pieza 5.

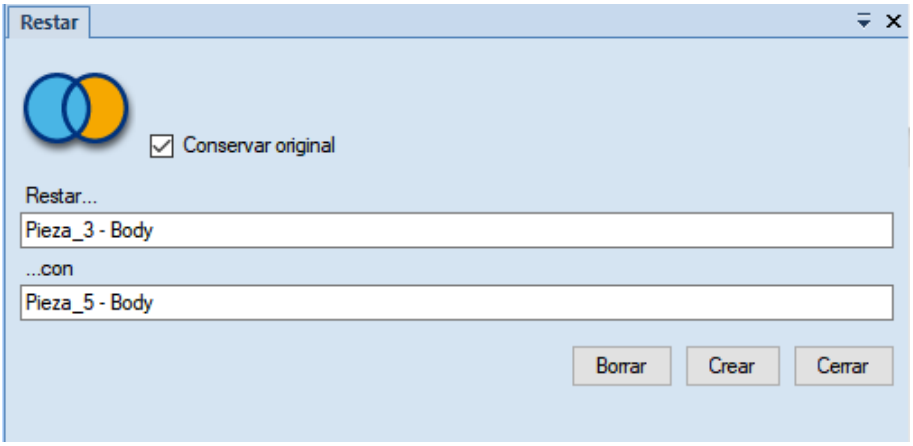


Ilustración 20: Herramienta de resta.

El resultado de los anteriores pasos es el cuerpo de la derecha de la **Ilustración 3**.

Después de realizar diversas operaciones de suma y resta de cuerpos en la pestaña de modelado se llega al resultado final mostrado en la **Ilustración 3**.

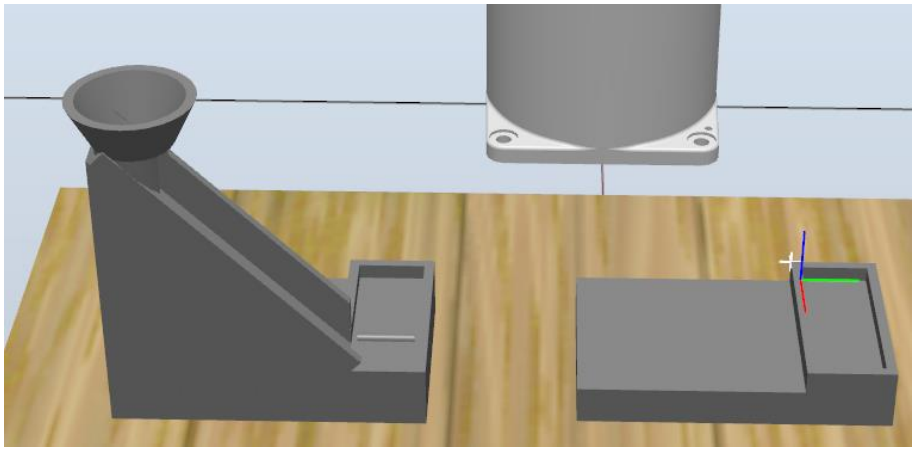


Ilustración 21: Modelado del plano inclinado.

2.3 Modelado del Plinko.

La segunda maqueta que se ha creado en la estación es el Plinko, el cual dota de dinamismo al proyecto añadiéndole una componente aleatoria además de varias posiciones donde se depositan las bolas obligando al ABB irb120 a elegir la posición donde debe dirigirse a dejar o coger las bolas.

Un plinko es un juego de feria que consiste en un tablero con unos clavos equidistantes sobre los cuales cae una bola chocando con ellos y cambiando su trayectoria hasta que llega a unos casilleros donde se deposita.

Los ejemplos sobre los que se ha basado el diseño son los siguientes:

Ilustración 22: Juego del Plinko. [6].

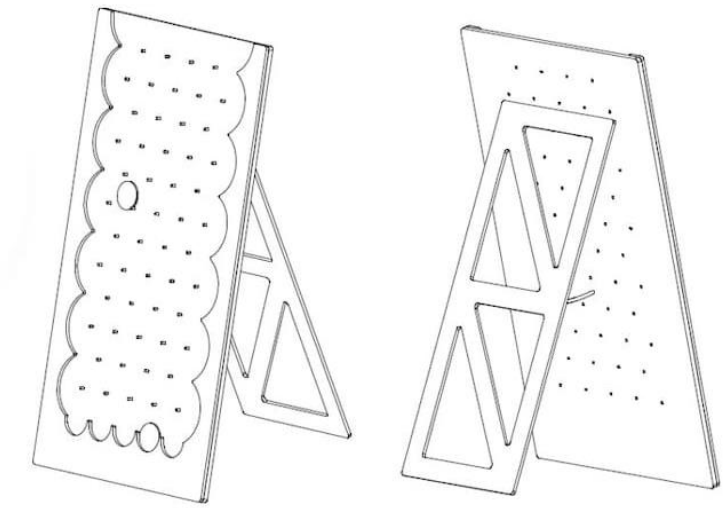


Ilustración 23: Planos Plinko. **[7].**

El modelado del Plinko se ha realizado de forma similar al plano inclinado, esta maqueta ha sufrido cambios desde su primer prototipo hasta la maqueta final debido a la proximidad de los clavos con el cableado.

Esta maqueta se ha diseñado en Robotstudio ya que sus dimensiones y posición en el espacio de trabajo deben adecuarse al espacio de trabajo del robot.

Las dimensiones del tablero, que contiene los clavos, es de 400x300 mm y se encuentra inclinado 55º con respecto del eje Z, logrando un tablero amplio donde la caída de las bolas en los distintos casilleros es lo suficientemente aleatoria como para no conocer de forma intuitiva donde va a caer la bola.

El Plinko dispone de cinco casilleros los cuales deben ser lo suficientemente hondos como para conseguir que la bola active un final de carrera y se mantenga en esa posición sin entorpecer al robot la cogida de la bola, conociendo el diámetro de la esfera se determinó una profundidad de 10mm, profundidad menor que su radio, para permitir que la pinza cogiese la bola sin complicaciones.

El primer diseño del Plinko fue el que se muestra en la **Ilustración *24***.

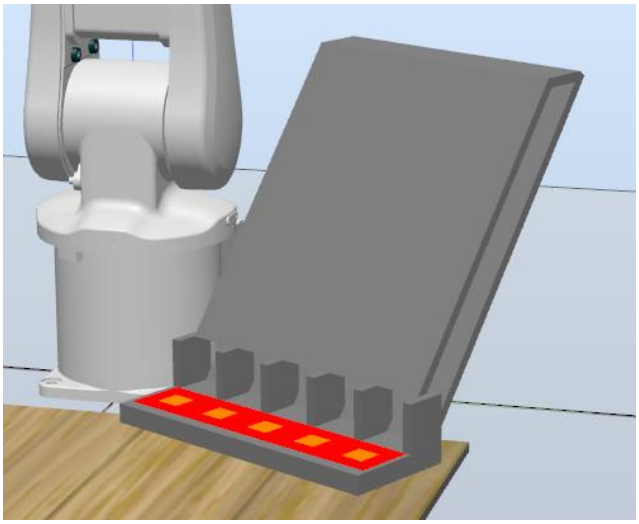


Ilustración 24: Primer diseño del Plinko.

En el primer diseño, el Plinko presenta una base muy estrecha donde los finales de carrera elegidos presentan una palanca muy pequeña y se encontrarían justo debajo.

Este primer diseño se maquetó como se muestra en la **Ilustración 25**.

Ilustración 25: Maquetado del primer diseño del Plinko.

El primer prototipo presentaba tres inconvenientes:

1. El primero es la cercanía de los casilleros con el tablero de clavos donde los cables que gobernaban la pinza se podían enredar además en el primer casillero el robot se encontraba cerca de una singularidad.

2. El segundo es que los finales de carrera no presentaban la sensibilidad suficiente como para activarse con el peso de las bolas.
3. El tercero es el apoyo del plano inclinado el cual necesitaba de una estructura más rígida de lo esperado.

Todas estos estos problemas fueron subsanados en el modelo de Robotstudio además de una mejor estética añadiendo los clavos al modelo, el resultado final es el siguiente.

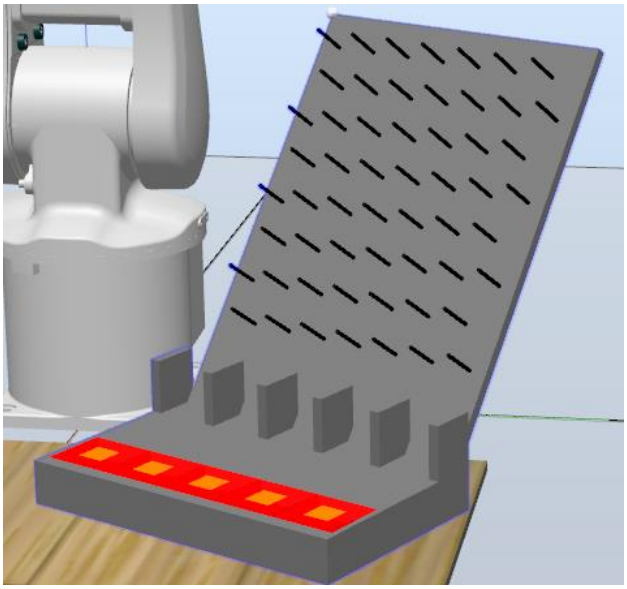


Ilustración 26: Diseño final del Plinko.

2.4 Creación de los WorkObjects.

En Robotstudio los WorkObjects son usados para definir todos los movimientos del robot, las trayectorias de un robot es recomendable que sigan unos puntos dados cuya posición en el espacio esta referenciada a un sistema de coordenadas denominado WorkObject.

En Robotstudio existe un WorkObject por defecto que es wobj0 y es el sistema de referencia fijo sobre el cual se referencian el resto.

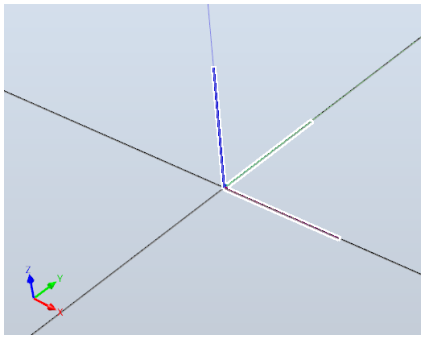


Ilustración 27: WorkObject base.

El resto de WorkObject se deben crear para ello se selecciona en Posición inicial Base de coordenadas donde hay dos métodos para crearlo, el que se ha elegido es el sistema de coordenadas por tres puntos debido a que mediante las herramientas disponibles en Robotstudio este sistema se puede localizar unido a la maqueta de una forma muy sencilla.

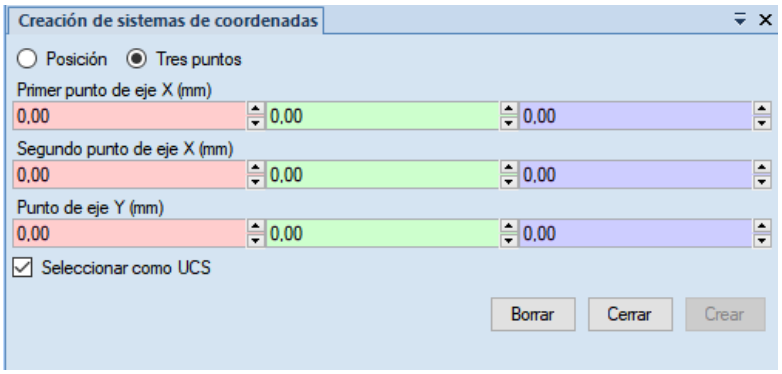


Ilustración 28: Creación de sistemas de coordenadas por tres puntos.

Ilustración 29: Herramientas de diseño para localizar puntos.

El sistema de coordenadas creado se localiza en la pestaña de diseño donde al pulsar click derecho se despliega una serie de opciones entre las cuales se encuentra “Convertir base de coordenadas en objeto de trabajo”.

Tras convertirlo en WorkObject este se localiza en la pestaña de trayectorias y puntos. Es recomendable conectar el WorkObject al objeto sobre el que se ha creado con el fin de que cuando se modifique su posición el WorkObject se desplace con él.

Tras crear el WorkObject se crean los puntos cuya posición y orientación utilizaran como referencia el objeto de trabajo creado, para ello se pulsa sobre posición y en crear punto.

Se utiliza como referencia un WorkObject, se establece su posición, orientación, se le da un nombre y se selecciona el WorkObject sobre el que debe ir referenciado.

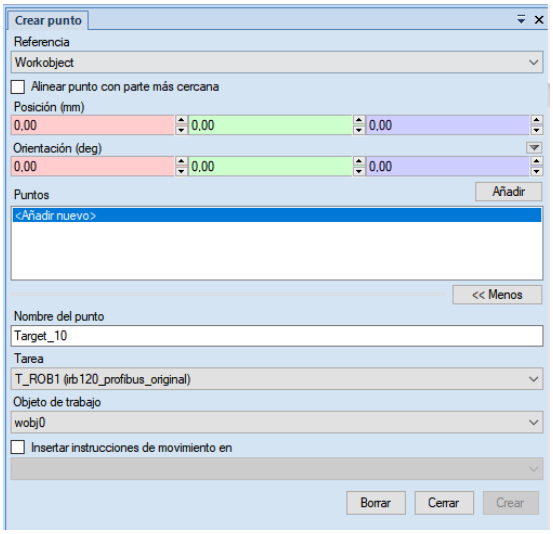


Ilustración 30: Creación de puntos.

En este proyecto hay creados tres WorkObjects, dos sobre el Plinko y uno sobre el plano inclinado, sus posiciones y orientaciones quedan reflejados en la siguiente tabla.

WorkObject	Posición (mm)		
	x	y	z
Casillero_Plinko	165,44	93,44	50
Posicionador_esferas	-66,46	224,61	377,5
Plano_inclinado	302,14	-173,12	35,04

Tabla 1: Posición WorkObjects.

WorkObject	Orientación (mm)			
	q1	q2	q3	q4
Casillero_Plinko	0,971	0	0	-0,236
Posicionador_esferas	0,971	0	0	-0,236
Plano_inclinado	1	0	0	0

Tabla 2: Orientación WorkObjects.

La orientación de los WorkObjects son las rotaciones del sistema de coordenadas expresada como un cuaternio con sus cuatro componentes q1, q2, q3 y q4.

El Plinko presenta dos WorkObject debido a que es la maqueta más compleja, y presenta dos partes a la que debe posicionarse el robot.

El primer WorkObject es el casillero, lugar donde las bolas se depositan tras recorrer los clavos.

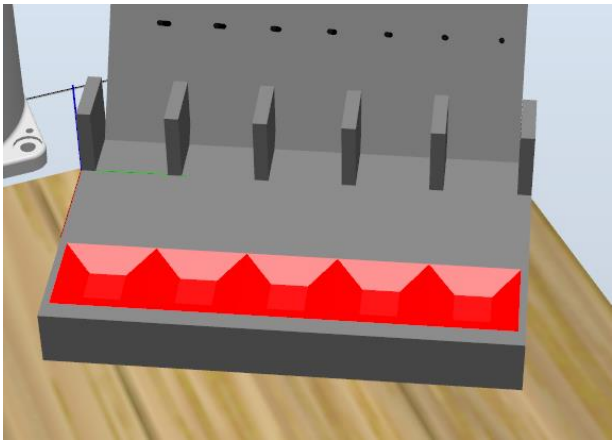


Ilustración 31: WorkObject Casillero_Plinko.

El segundo WorkObject se encuentra en la parte superior del Plinko lugar donde se sueltan las bolas.

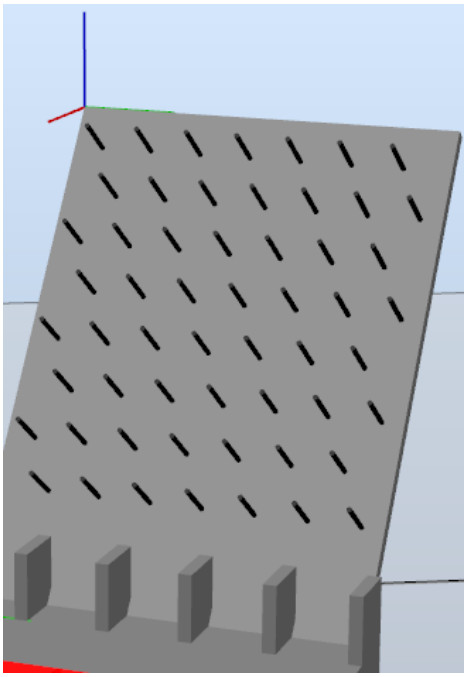


Ilustración 32: WorkObject Posicionador_esferas.

Por último, se encuentra el WorkObject del plano inclinado, esta maqueta es más sencilla y solo presenta dos puntos. El punto al que debe dirigirse el TCP para la recogida de las bolas es el más crítico por lo que este sistema de referencia se encuentra próximo a él.

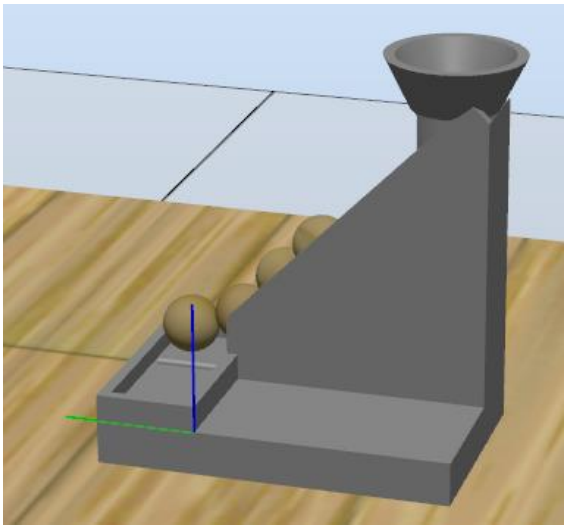


Ilustración 33: WorkObject Plano_inclinado.

El conjunto de los WorkObjects se encuentra en “Posición inicial” en la carpeta de &Objetos de trabajo y puntos.

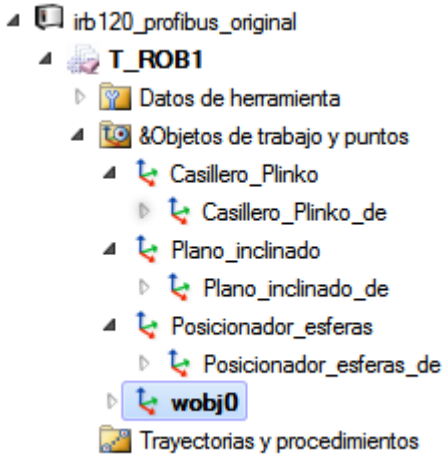


Ilustración 34: WorkObjects de la estación.

Tras crear los WorkObjects se crean los Robtargets, puntos referenciados a los WorkObjects, sobre los que se creará la trayectoria.

Los Robtargets contienen la información de su posición y orientación, esta última es importante ya que es la cual indica la posición del TCP.

El TCP presenta un sistema de coordenadas el cual se posiciona con los ejes coincidentes con los del Robtarget.

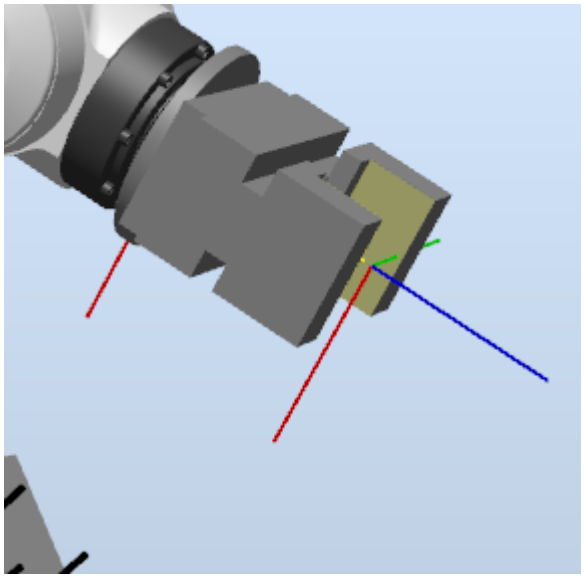


Ilustración 35: Sistema de referencia del robot y su pinza, TCP.

Tras crear un punto este presenta una orientación que no siempre es la preferible para que el robot coloque su TCP en ese punto con el mejor posicionamiento posible.

Para ello se puede ver la herramienta en el punto, esto se logra pulsando sobre el Robtarget creado y seleccionando ver herramienta en posición.

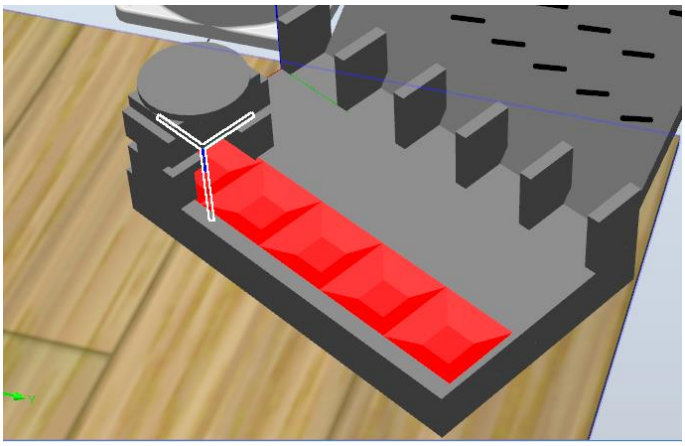


Ilustración 36: Sistema de referencia del robot y su pinza, TCP.

También se puede ver el robot en la posición y así corregir la posición de la maqueta o realizar las modificaciones pertinentes para facilitar la maniobrabilidad del robot.

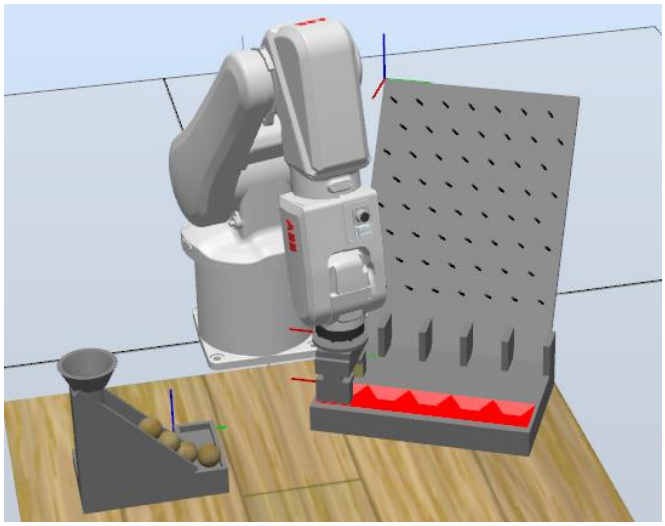


Ilustración 37: Posicionamiento del robot en el casillero.

Todos los Robtargets de la estación están registrados en las siguientes tablas.

Casillero_Plinko	Posición (mm)		
	x	y	z
Casillero_1	111,0	36	15
Casillero_2	110,99	98	15
Casillero_3	110,99	160	15
Casillero_4	110,99	222	15
Casillero_5	110,99	284	15

Tabla 3: Posiciones de los Robtargets vinculados con Casillero_Plinko.

Casillero_Plinko	Orientación (mm)			
	q1	q2	q3	q4
Casillero_1	0	-0,707	0,707	0
Casillero_2	0	-0,707	0,707	0
Casillero_3	0	-0,707	0,707	0
Casillero_4	0	-0,707	0,707	0
Casillero_5	0	-0,707	0,707	0

Tabla 4: Orientaciones de los Robtargets vinculados con Casillero_Plinko.

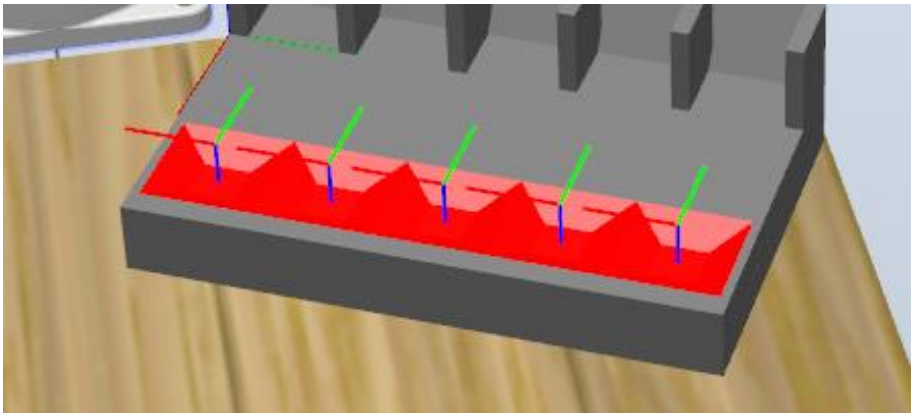


Ilustración 38: Robtargets del casillero.

Posicionador_esferas	Posición (mm)		
	x	y	z
posición esfera inicial	24,893	30	30

Tabla 5: Posiciones de los Robtargets vinculados con Posicionador_esferas.

Posicionador_esferas	Orientación (mm)			
	q1	q2	q3	q4
posición esfera inicial	0	-0,707	0,707	0

Tabla 6: Orientaciones de los Robtargets vinculados con Posicionador_esferas.

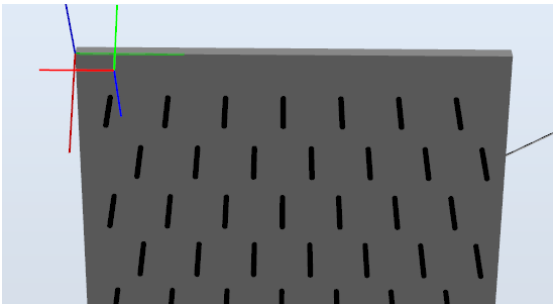


Ilustración 39: Posición de esfera inicial.

Este Robtarget sirve como referencia para indicar el inicio de una recta horizontal paralela al eje Y del WorkObject desde la cual se sueltan las bolas.

Plano_inclinado	Posición (mm)		
	x	y	z
Recoge_bolas	41,6	28,25	28
Traga_bolas	101,5	-118,5	215

Tabla 7: Posiciones de los Robtargets vinculados con Plano_inclinado.

Plano_inclinado	Orientación (mm)			
	q1	q2	q3	q4
Recoge_bolas	0	0	1	0
Traga_bolas	0	0,383	0,924	0

Tabla 8: Orientaciones de los Robtargets vinculados con Plano_inclinado.

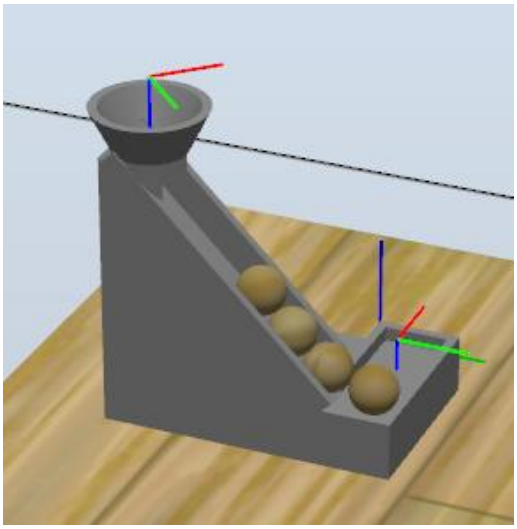


Ilustración 40: Robtargets del plano inclinado.

2.5 Entradas y salidas digitales

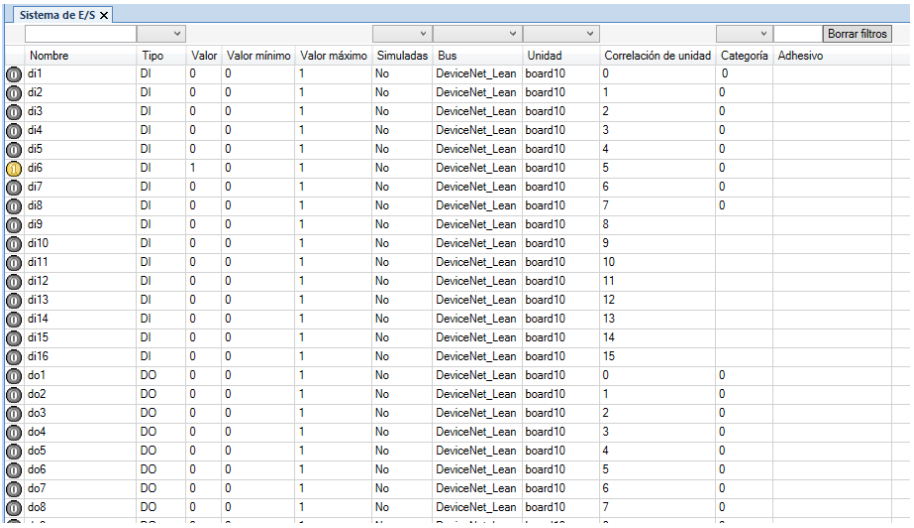
Las entradas y salidas digitales son los pines por los cuales se puede establecer una comunicación entre los elementos de la estación, sensores y actuadores, con la controladora. La comunicación entre la estación y la controladora es imprescindible para que el robot conozca la situación del sistema y pueda actuar sobre el.

2.5.1 Descripción de E/S digitales

Las señales de entradas y salidas digitales pueden ser simuladas mediante valores que se pueden introducir desde la propia interfaz de Robotstudio desde la cual se encuentra un simulador de una botonera o programar unas instrucciones por las cuales se activan o desactivan estas entradas y salidas cuando se cumplen unas determinadas condiciones.

El sistema de entradas y salidas muestra todos los datos de las entradas y salidas del sistema con las que trabaja el controlador IRC5.

En este sistema de entradas y salidas, como se aprecia en la **Ilustración 41**, se puede conocer el nombre de la entrada y salida, si está activa o no, su valor actual, su valor máximo y mínimo, a que bus pertenecen y la unidad a la que están conectadas.



Nombre	Tipo	Valor	Valor mínimo	Valor máximo	Simuladas	Bus	Unidad	Correlación de unidad	Categoría	Adhesivo
di1	DI	0	0	1	No	DeviceNet_Lean	board10	0	0	
di2	DI	0	0	1	No	DeviceNet_Lean	board10	1	0	
di3	DI	0	0	1	No	DeviceNet_Lean	board10	2	0	
di4	DI	0	0	1	No	DeviceNet_Lean	board10	3	0	
di5	DI	0	0	1	No	DeviceNet_Lean	board10	4	0	
di6	DI	1	0	1	No	DeviceNet_Lean	board10	5	0	
di7	DI	0	0	1	No	DeviceNet_Lean	board10	6	0	
di8	DI	0	0	1	No	DeviceNet_Lean	board10	7	0	
di9	DI	0	0	1	No	DeviceNet_Lean	board10	8		
di10	DI	0	0	1	No	DeviceNet_Lean	board10	9		
di11	DI	0	0	1	No	DeviceNet_Lean	board10	10		
di12	DI	0	0	1	No	DeviceNet_Lean	board10	11		
di13	DI	0	0	1	No	DeviceNet_Lean	board10	12		
di14	DI	0	0	1	No	DeviceNet_Lean	board10	13		
di15	DI	0	0	1	No	DeviceNet_Lean	board10	14		
di16	DI	0	0	1	No	DeviceNet_Lean	board10	15		
do1	DO	0	0	1	No	DeviceNet_Lean	board10	0	0	
do2	DO	0	0	1	No	DeviceNet_Lean	board10	1	0	
do3	DO	0	0	1	No	DeviceNet_Lean	board10	2	0	
do4	DO	0	0	1	No	DeviceNet_Lean	board10	3	0	
do5	DO	0	0	1	No	DeviceNet_Lean	board10	4	0	
do6	DO	0	0	1	No	DeviceNet_Lean	board10	5	0	
do7	DO	0	0	1	No	DeviceNet_Lean	board10	6	0	
do8	DO	0	0	1	No	DeviceNet_Lean	board10	7	0	

Ilustración 41: Sistema de entradas y salidas

Las entradas se utilizan para comunicar eventos que se producen en la estación con el controlador con el fin de que el robot ejecute una serie de instrucciones determinadas como por ejemplo la activación de una salida.

En el caso de este proyecto las señales de entrada corresponden con los finales de carrera que se encuentran en los casilleros y el sensor de bola disponible en el plano inclinado.

En este proyecto solo se usan dos señales de salida, una para activar la pinza y otra para conmutar la electroválvula encargada de permitir el paso de aire hacia el pistón que dispensa las bolas.

2.5.2 Simulación de E/S digitales

Para simular las entradas y salidas en Robotstudio es necesario activar la vista de controlador en la pestaña controlador.

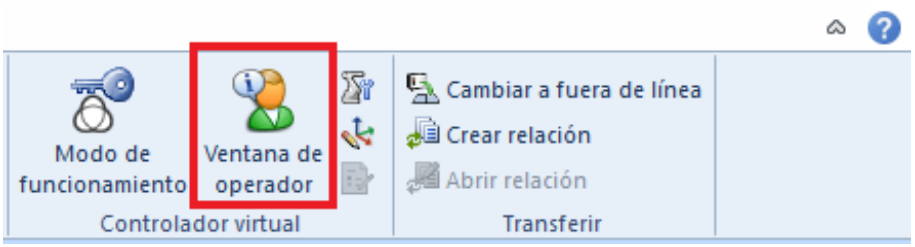


Ilustración 42: Ventana operador

Se debe establecer el modo de funcionamiento en automático.

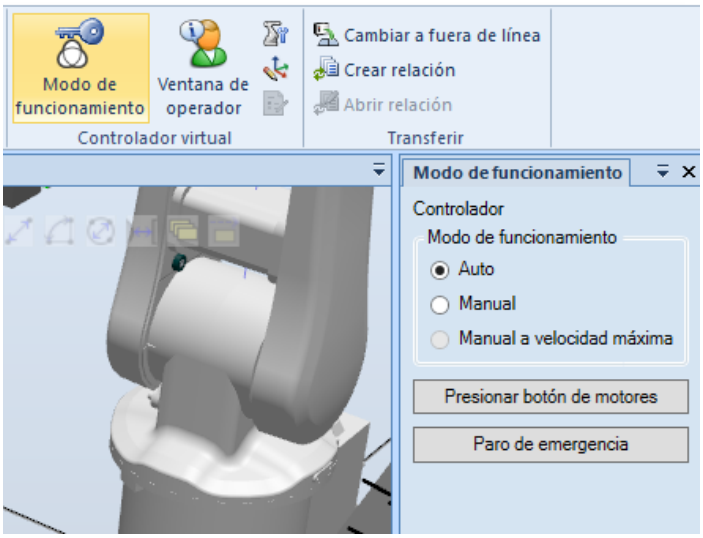


Ilustración 43: Modo de funcionamiento

En este caso, y como se muestra en la **Ilustración 41** la unidad de las entradas y salidas es la boar10 la cual contiene 16 entradas y 16 salidas, de las cuales solo se usan 8 entradas y 1 salida. 6 de las entradas corresponden con los finales de carrera y el sensor que indica la existencia de una bola para ser dispensada, las otras dos entradas se encuentran en la botonera y son utilizadas para gestionar interrupciones.

La salida solo corresponde con la activación de la electroválvula conectada al pistón que dispensa las bolas.

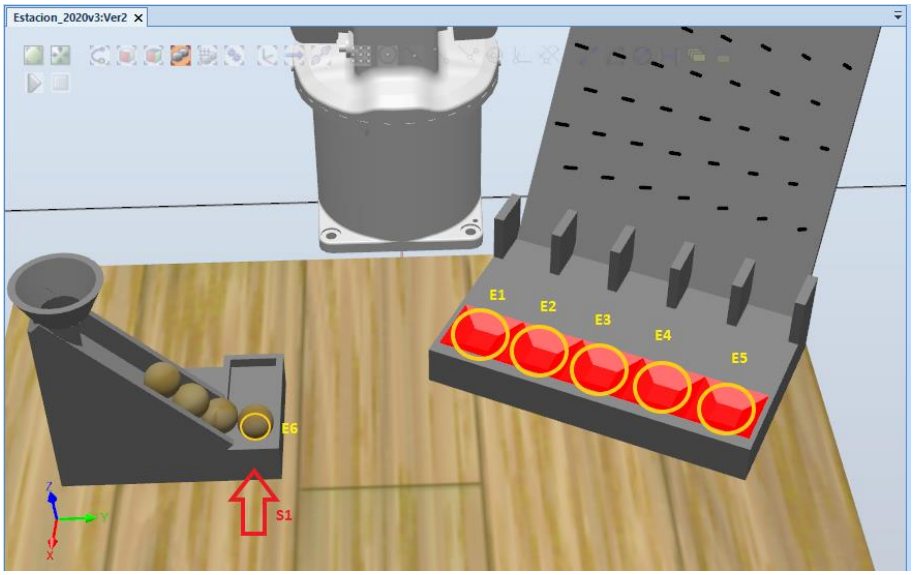


Ilustración 44: Disposición de las entradas y salidas

En la siguiente tabla se recogen las distintas entradas y salidas utilizadas y donde son empleadas.

Nombre	Tipo de señal	Identificación
di1	Digital Input	Sensor casillero 1
di2	Digital Input	Sensor casillero 2
di3	Digital Input	Sensor casillero 3
di4	Digital Input	Sensor casillero 4
di5	Digital Input	Sensor casillero 5
di6	Digital Input	Sensor plano inclinado
di7	Digital Input	Botón di7
di8	Digital Input	Botón di8
do1	Digital Output	Electroválvula

Tabla 9: Tabla de entradas y salidas.

2.6 Programación de Smartcomponents

Para la simulación del comportamiento de las bolas durante el proceso, se ha recurrido a la utilización de componentes inteligentes o SmartComponents los cuales permiten simular el comportamiento de los objetos de la estación.

En este proyecto existen cuatro componentes inteligentes:

1. Componente inteligente botonera: Este componente se ha tomado de la estación en Robotstudio proporcionada por el departamento de automatización y no se ha modificado. Este componente simula una botonera real que se encuentra en el laboratorio de robótica y cambia el color de las bombillas según las salidas son activadas.

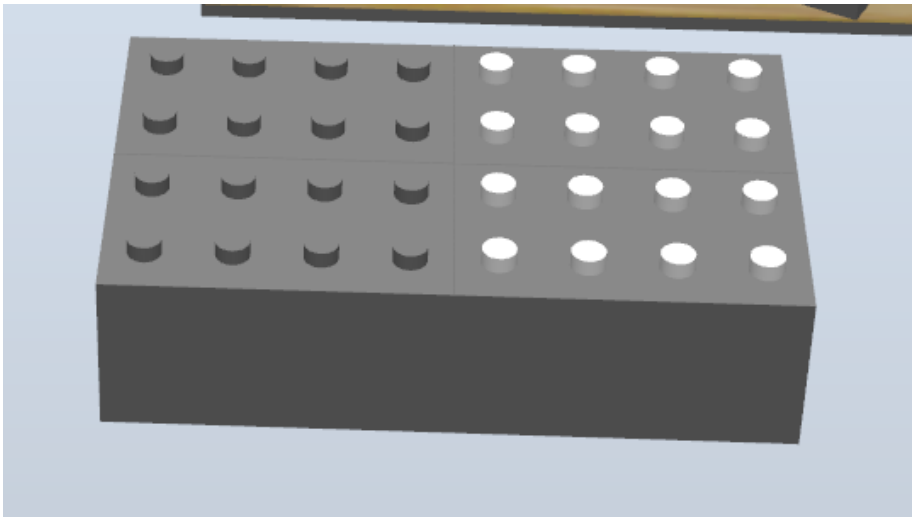


Ilustración 45: Componente inteligente botonera.

2. El segundo componente inteligente es la pinza, esta se llama en la estación Pinza_Normal, sin embargo, se ha debido adaptar al

tamaño de las bolas por lo que a la herramienta se la denomina PinzaAncha con el fin de diferenciarla de la original.

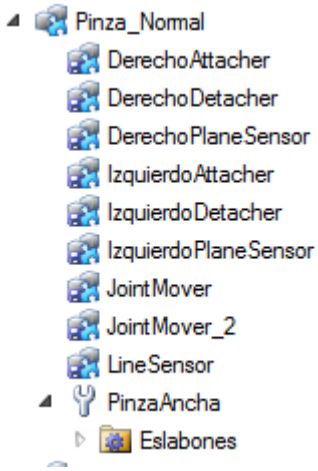


Ilustración 46: PinzaAncha.

3. El tercer componente inteligente es el plano inclinado el cual se encarga de almacenar las bolas, este objeto ha sido creado en su totalidad con el fin de satisfacer las necesidades de este trabajo.
   Este objeto debe ser un componente inteligente ya que simula la actuación del pistón.
4. Por último, el Plinko, este objeto es el más complejo de los cuatro y es el que simula la caída de la bola a través de los clavos y su posicionamiento en el casillero de forma aleatoria.

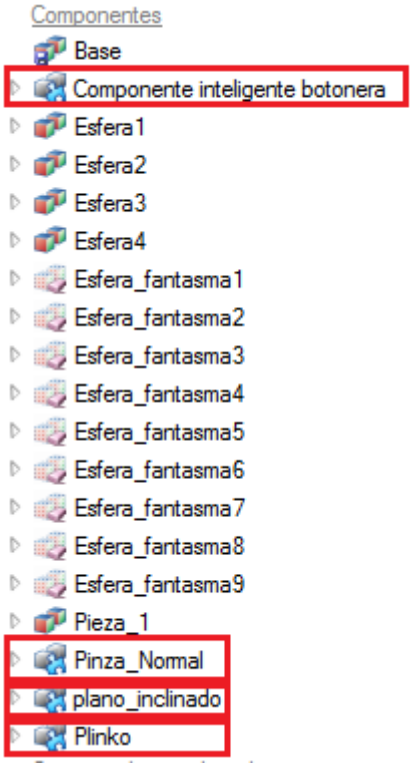


Ilustración 47: Componentes inteligentes en la estación.

2.6.1 Smartcomponent de la Pinza ancha

Dentro del uso de los componentes inteligentes se modificó la pinza la cual al aumentar el grosor de las paredes laterales cambió también la posición de unos planos encargados de unir la bola con la pinza.

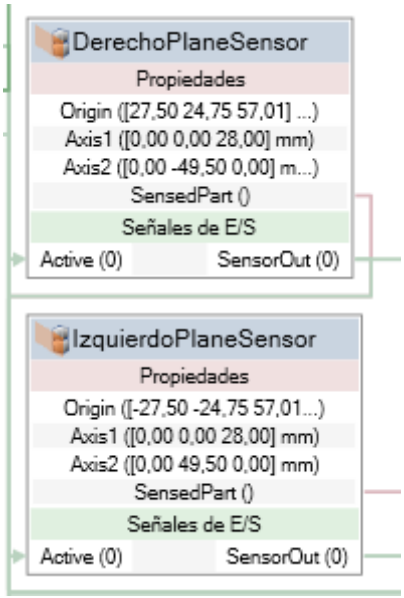


Ilustración 48: Sensores planares.

Estos planos son importantes ya que cuando detectan un objeto activan un bloque denominado “Attacher” el cual une el objeto que ha sido detectado por uno de los dos planos, en este caso la bola.

Hay dos bloques “Attacher”: “Attacher Izquierdo” y “Attacher Derecho” los cuales son activados con respectivos planos que se encuentran pegados a la pared interior de la pinza.

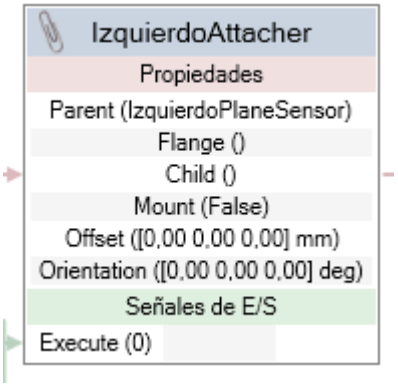


Ilustración 49: Attacher.

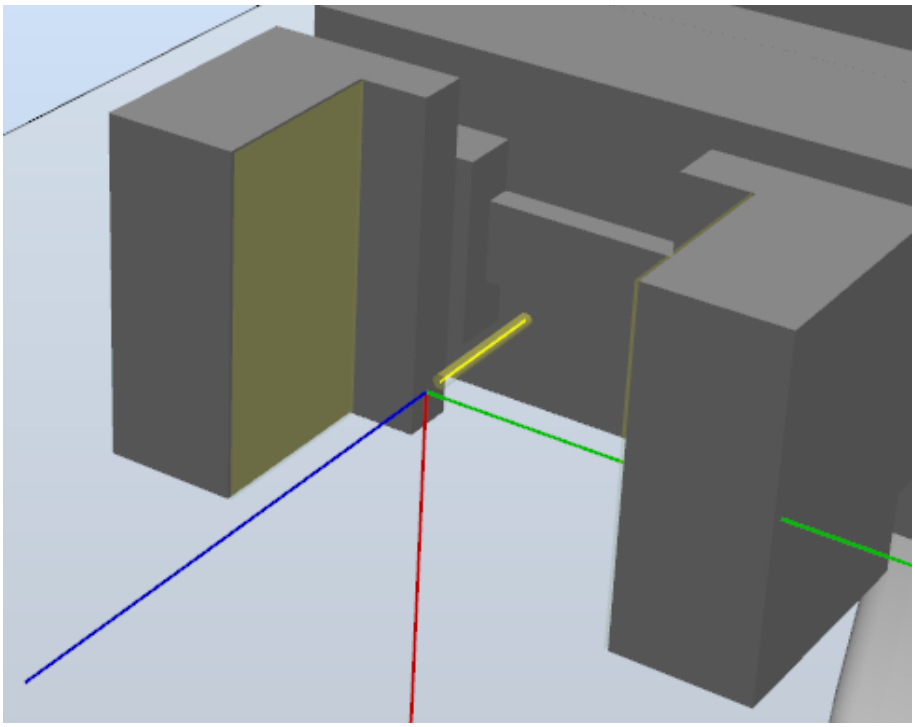


Ilustración 50: Plane Sensor.

Para separar las bolas de las paredes de la pinza se debe ejecutar el bloque Detacher el cual se activa cuando recibe la orden de abrir pinza.

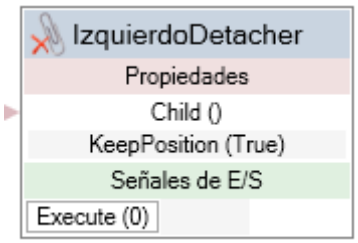


Ilustración 51: Detacher.

Cuando se envía la orden de abrir pinza se separa la bola de las paredes laterales de la pinza y se empiezan a separar estas del centro. El bloque encargado de realizar este movimiento es "JointMover".

Este mismo bloque debe ser utilizado cuando se cierra la pinza.

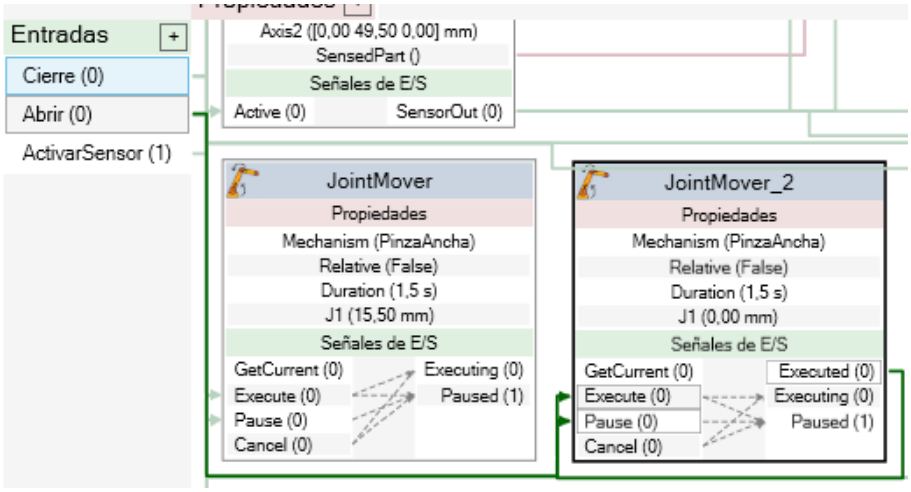


Ilustración 52: JointMover.

2.6.2 Smartcomponent del Plano inclinado

El plano inclinado realiza dos modificaciones en la posición de las bolas:

Una de ellas se encarga de hacer desaparecer la bola de la parte superior donde el robot las suelta para que queden almacenadas y hacerla aparecer en el carril, esto se realiza mediante SmartComponents debido a la dificultad de crear un objeto completamente funcional con la herramienta de modelado.

Esta funcionalidad se ha realizado con un sensor planar el cual detecta una bola y ejecuta un bloque Source el cual crea una copia de la bola en el carril del plano inclinado donde se almacenan las bolas. Tras la ejecución del bloque Source se envía una orden a un bloque Sink el cual elimina la bola original detectada por el sensor planar.

El sensor planar se activa cuando la pinza se abre y es el encargado de detectar e identificar el objeto que lo toca para enviar dicha información al resto de bloques siguientes.

Para configurar el sensor se le deben dar una posición y unas dimensiones las cuales se establecen por un origen y dos ejes.

A este sensor se le posiciona justo encima del lugar donde la pinza suelta la bola y se conecta con la entrada de apertura de pinza.

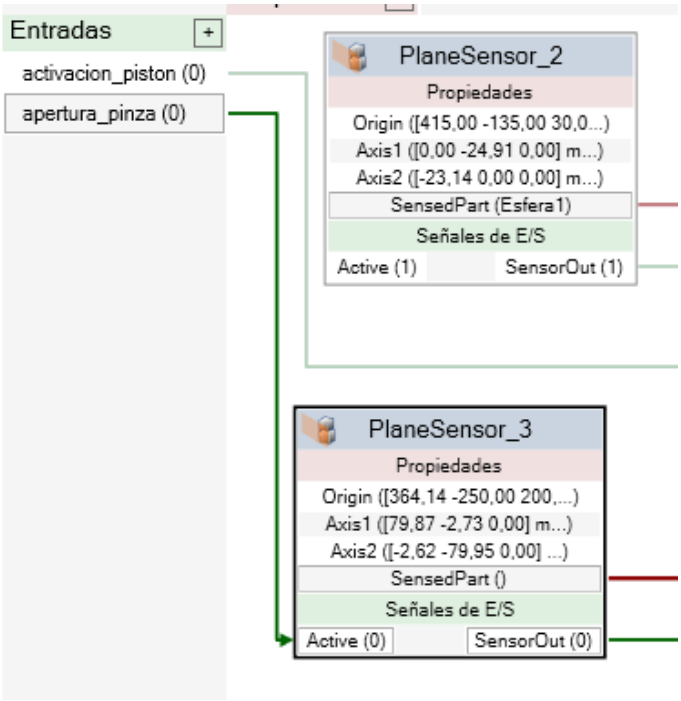


Ilustración 53: Sensor planar encargado del almacenamiento de las bolas.

Como se muestra en la **Ilustración 53** la casilla Active se encuentra a cero por defecto ya que esta se activa con la entrada de apertura_pinza.

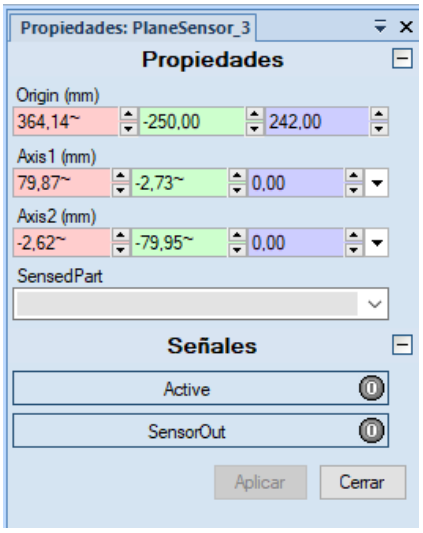


Ilustración 54: Configuración del sensor planar encargado del almacenamiento de las bolas.

Como se aprecia en la **Ilustración 54** se establece un origen y unas dimensiones haciendo uso de Origin, Axis1 y Axis2.

Los ejes Axis1 y Axis2 pueden ser configurados más fácilmente haciendo uso de una pestaña que se despliega tras pulsar en la flecha que se encuentra a la derecha de los casilleros.

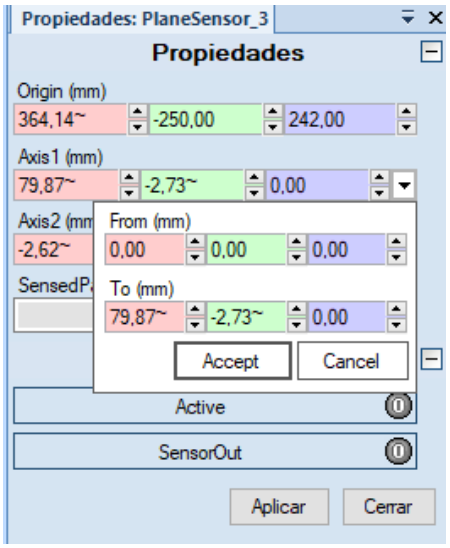


Ilustración 55: Configuración de los ejes.

Para que este bloque se mantenga unido al modelo en tres dimensiones cuando este cambie de posición se selecciona el bloque en el desplegable de diseño donde se encuentra el sensor, se selecciona conectar y se elige el cuerpo al que se desea trasladar.

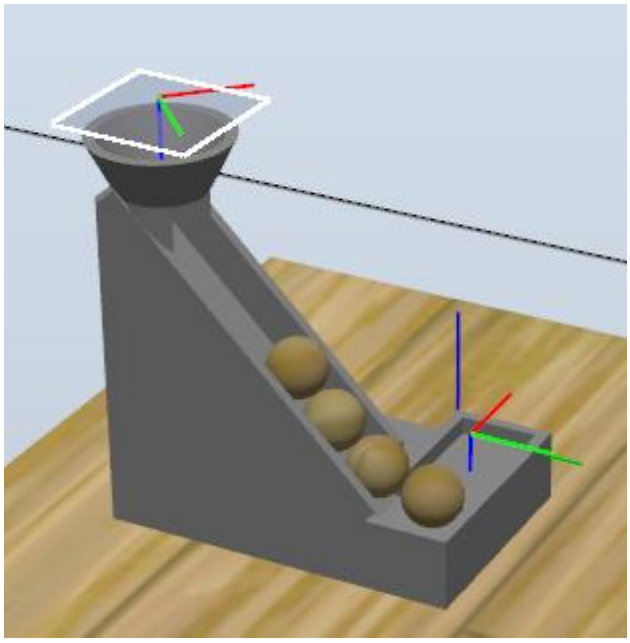


Ilustración 56: Posición del sensor planar.

De este sensor se aprovechan dos salidas, si se ha detectado una pieza y que pieza se ha detectado ya que esta información se le debe dar a el bloque de Source y de Sink.

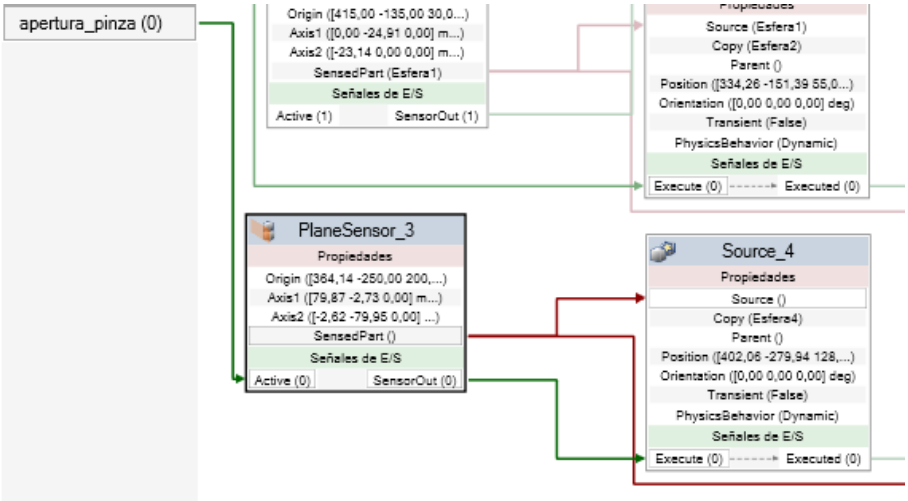


Ilustración 57: Conexiones entre el sensor planar y el bloque Source.

El bloque Source es el encargado de realizar una copia del objeto que se indique en la posición de source() en cualquier otra parte de la estación.

Para el funcionamiento de este bloque se le aporta como pieza a copiar la bola detectada y se ejecuta en el momento en el que se detecta una pieza, el lugar donde aparece la segunda es un punto en el carril del plano inclinado y el comportamiento físico que debe tener la copia se establece como dinámico para que se comporte como lo haría una bola en la realidad.

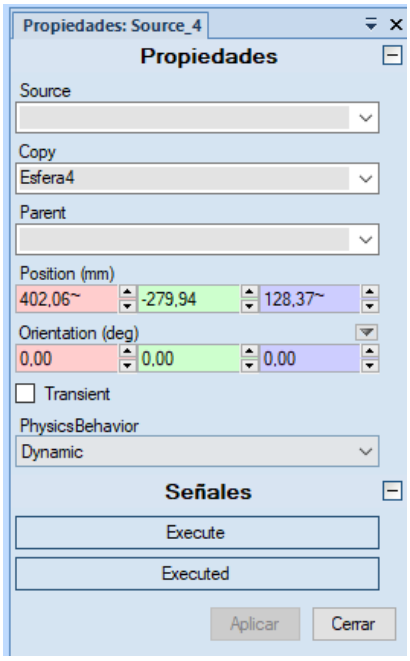


Ilustración 58: Configuración del bloque Source.

Tras la ejecución de este bloque se ejecuta el bloque Sink el cual es el encargado de eliminar una bola, este tiene una configuración más sencilla y solo hay que indicarle cuando debe ser ejecutado y la pieza que debe eliminar de la estación.

Este bloque elimina la pieza que ha sido detectada por el plano y se ejecuta tras la creación de la copia.

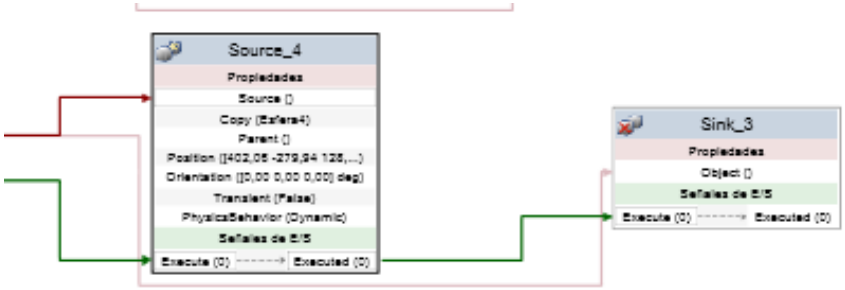


Ilustración 59: Conexión del bloque Sink.

Otra funcionalidad del plano inclinado es simular el comportamiento del pistón para ello se hace uso de una entrada llamada activacion_piston.

El funcionamiento del componente inteligente en esta funcionalidad es muy parecido al comportamiento de este componente cuando la intención es almacenar las bolas.

De igual forma presenta un sensor planar, el cual simula el comportamiento sensor que indica que hay una bola disponible para ser

dispensada, un bloque Source y un bloque Sink los cuales simulan el cambio de posición de la bola tras ser golpeada por el vástago del pistón.

Sin embargo, presenta algunas modificaciones.

El sensor planar se encuentra siempre activo y la detección de una bola se indica en una salida denominada bola_disponible.

El bloque Source se ejecuta con la entrada activacion_piston y el bloque Sink tras la ejecución del bloque Source.

En bloque Source la copia presenta una física dinámica como en el caso anterior.

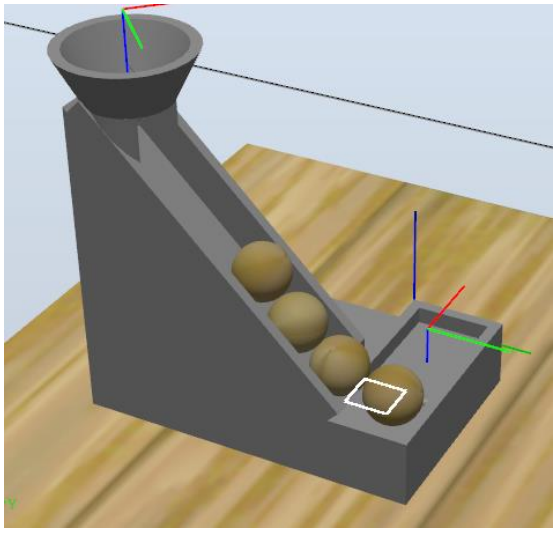


Ilustración 60: Posición del plano que simula el comportamiento de un sensor.

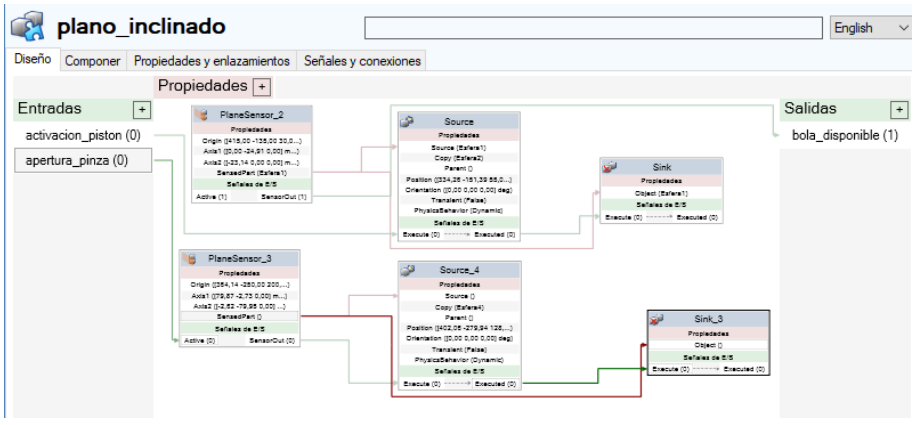


Ilustración 61: Componente inteligente plano inclinado.

2.6.3 Smartcomponent del Plinko

El Plinko es el componente inteligente más complejo debido a que simula el comportamiento aleatorio de las bolas al caer por el tablero lleno de

clavos y su posicionamiento en los distintos casilleros. Antes de detallar la programación del componente inteligente se añaden unas imágenes que muestran su funcionamiento.

Primeramente, el robot posiciona la bola en la parte superior del Plinko.

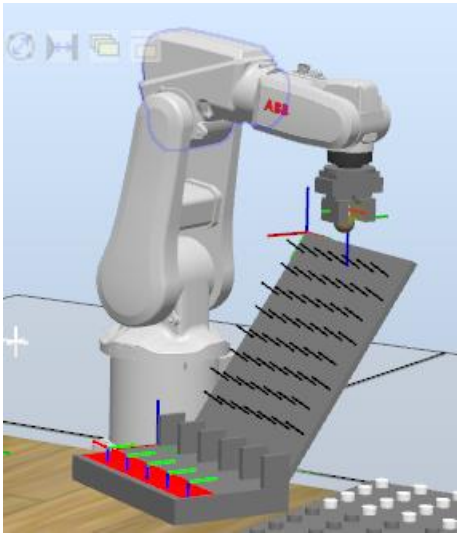


Ilustración 62: Posicionamiento de la bola en la parte superior del Plinko.

Desaparece la bola en la parte superior y aparece en la primera línea del Plinko.

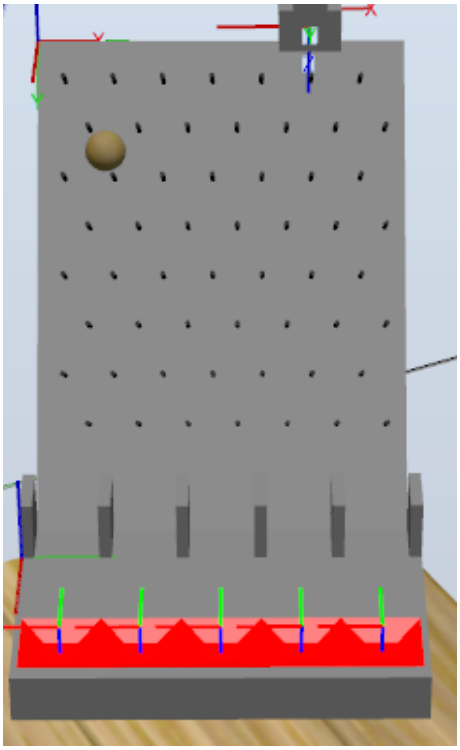


Ilustración 63: Primera aparición de la bola.

Desaparece la bola de la primera línea y aparece en la segunda línea.

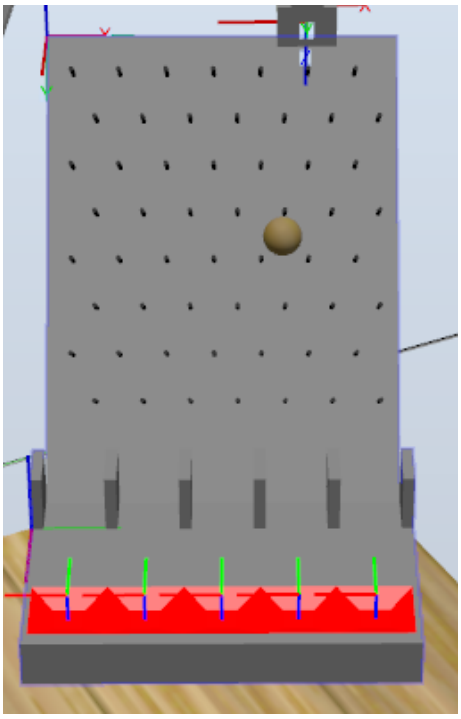


Ilustración 64: Segunda aparición de la bola.

Desaparece la bola de la segunda línea de aparición y aparece en la última línea.

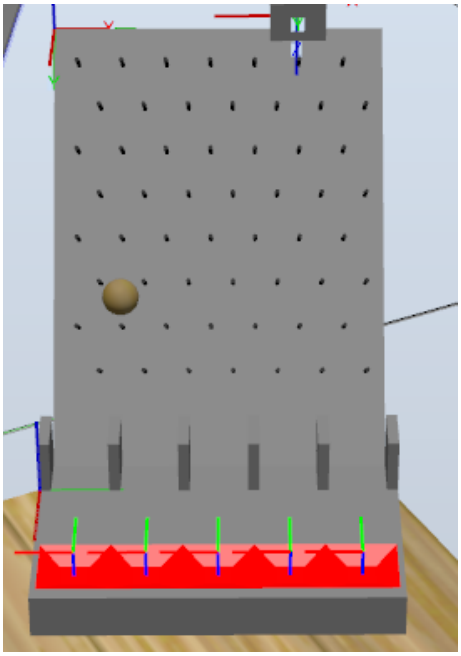


Ilustración 65: Tercera aparición de la bola.

Desaparece la bola de la última línea y aparece en un casillero de forma aleatoria.

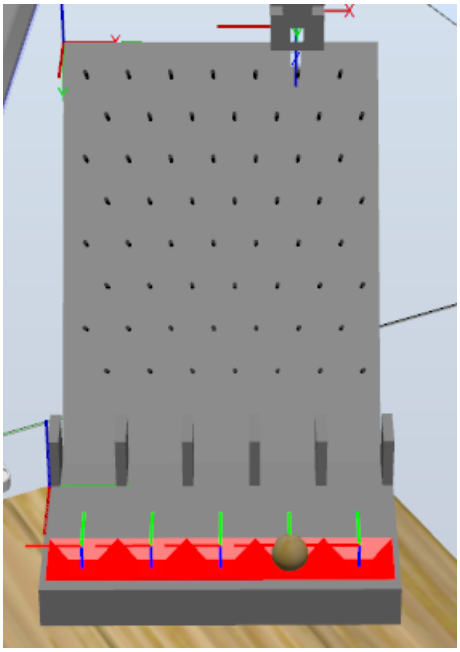


Ilustración 66: Aparición de la bola en el casillero.

Por último, el robot se desplaza hacia el casillero donde ha caído la bola y la recoge.

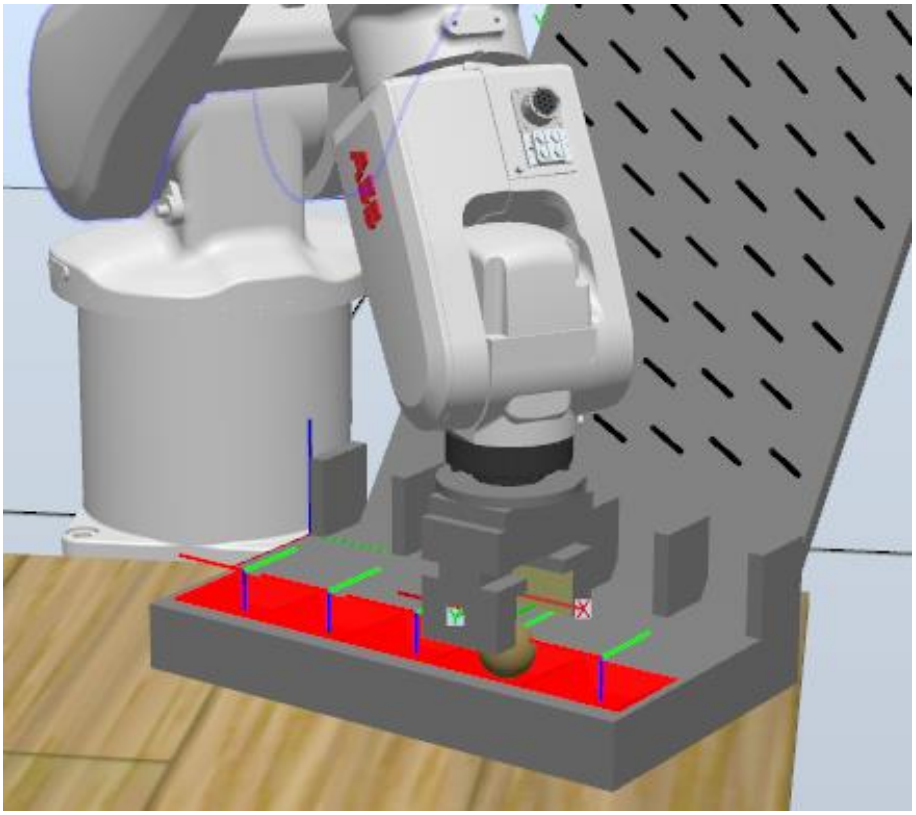


Ilustración 67: Recogida de la bola.

La programación del Plinko para que pueda simular el lugar donde y como ha caído la bola y la comunicación con la controladora para indicar dicha posición se ha realizado haciendo uso de una entrada y cinco salidas.

Como entrada se encuentra apertura_pinza y como salidas: Casillero1, Casillero2, Casillero3, Casillero4 y Casillero5.

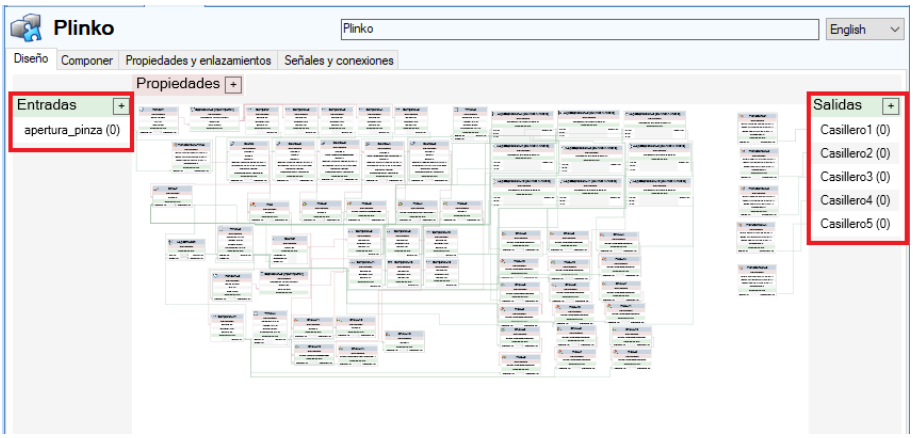


Ilustración 68: Entradas y salidas del componente inteligente Plinko.

El Plinko presenta un sensor planar el cual se encuentra en la parte superior de este y es activado mediante la entrada “apertura_pinza”.

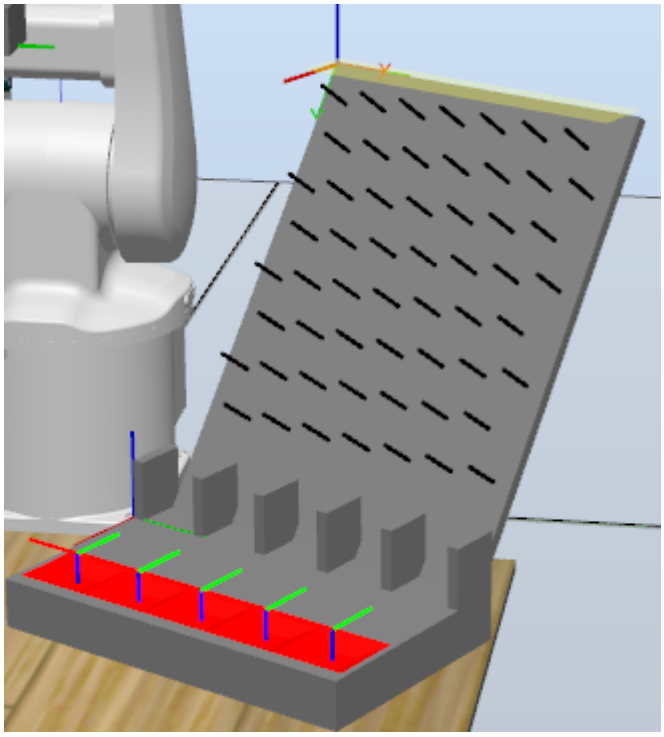


Ilustración 69: Posición del sensor planar en la parte superior del Plinko.

Este sensor es el encargado de reconocer la bola que se va a soltar y tras detectarla ejecuta un bloque que genera números aleatorios entre dos números. Este será el encargado de decidir en qué casillero aparecerá la bola finalmente. Como los números generados por este bloque son reales los límites impuestos a la generación son 1 y 5.99.

Tras obtener un número real del bloque de números aleatorios se truca haciendo uso de la siguiente ecuación.

$$input - input\%1$$

Ecuación 1: Truncamiento del número aleatorio.

Para truncar el número se resta dicho número con el resto de su división por la unidad de esta forma se le consigue restar su parte decimal.

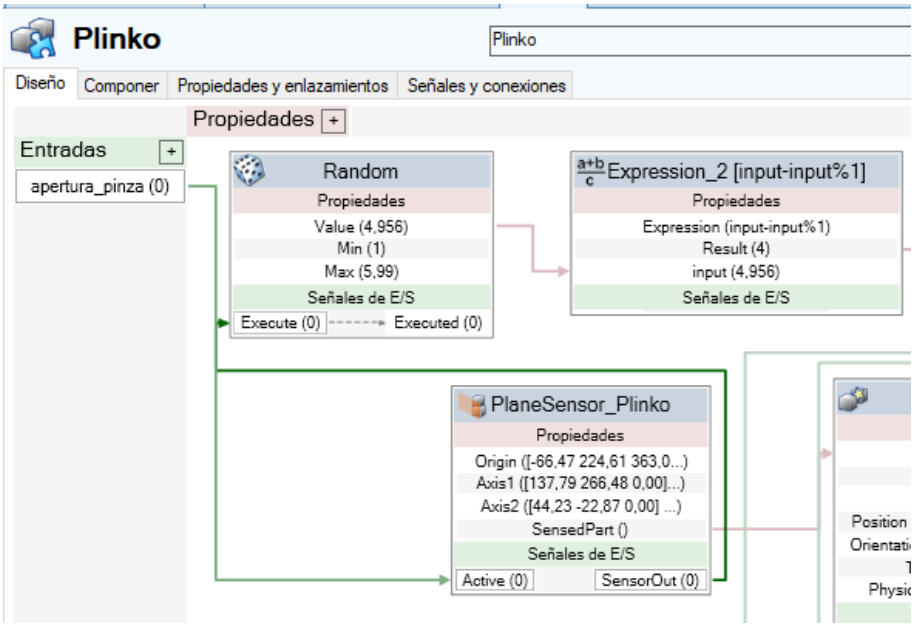


Ilustración 70: Reconocimiento de la bola y generación de números aleatorios.

Como resultado de la operación se obtiene un numero entero del 1 al 5 el cual se compara en cinco comparadores distintos, en el caso en el que el número sea igual al número establecido en la configuración de uno de los comparadores se activará una salida.

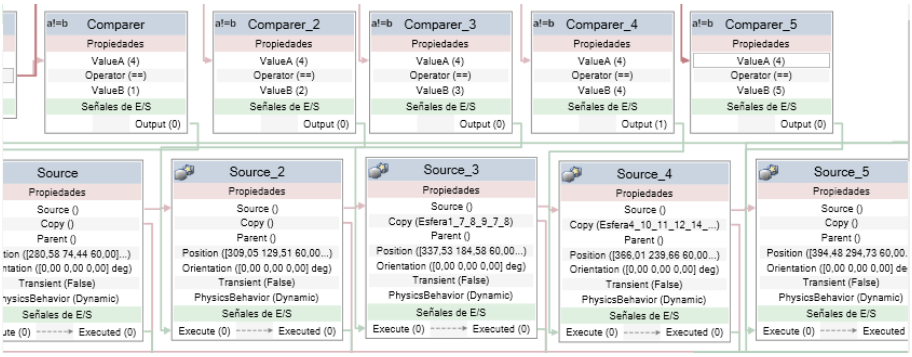


Ilustración 71: Generación de la bola en uno de los casilleros.

Como se aprecia en la **Ilustración 70** y en la **Ilustración 71** el valor generado es 4.956 el cual tras pasar por el bloque encargado de truncar dicho número se obtiene el valor de 4. El resultado se compara en cada uno de los comparadores con un numero diferente, en el caso de que la comparación sea correcta, es decir, que los dos números sean iguales, se activa la salida del comparador donde se ha cumplido la igualdad, en este caso en el comparador 4, y seguidamente se crea una copia de la bola en el casillero 4 mediante la ejecución del bloque Soruce_4.

Los bloques Source reciben la pieza que deben copiar de una de las salidas del sensor planar que se encuentra en la parte superior del Plinko.

Tras la creación de la copia esta se oculta haciendo uso del bloque Hide al cual se le debe indicar que pieza se debe ocultar, como en este caso no conocemos el nombre de la bola recién creada se debe utilizar una de las salidas del bloque Source para indicar dicho nombre.

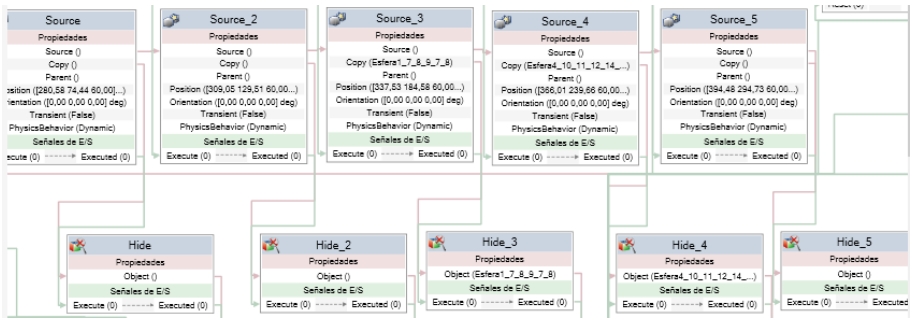


Ilustración 72: Ocultación de la bola creada.

Como se muestra en la **Ilustración 72** existen cinco bloques Hide uno por cada bloque Source, de esta forma tras la ejecución de un bloque Source el bloque Hide conectado a él toma el nombre de la bola creada y la oculta.

Se oculta dicha bola debido a que se debe crear la ilusión de que la bola está bajando por el Plinko.

Tras ocultar la copia la bola original, la cual fue detectada por el sensor planar, es eliminada.

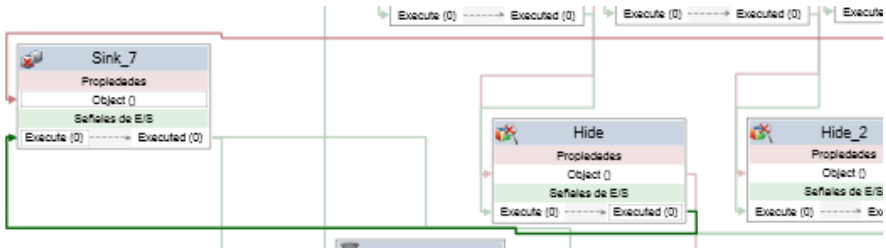


Ilustración 73: Ejecución del bloque Sink.

Para crear la ilusión que se muestra en la **Ilustración 63**, **Ilustración 64** y la **Ilustración 65** se han creado nueve bolas a las cuales se les a puesto el apellido fantasma y se han hecho invisibles.

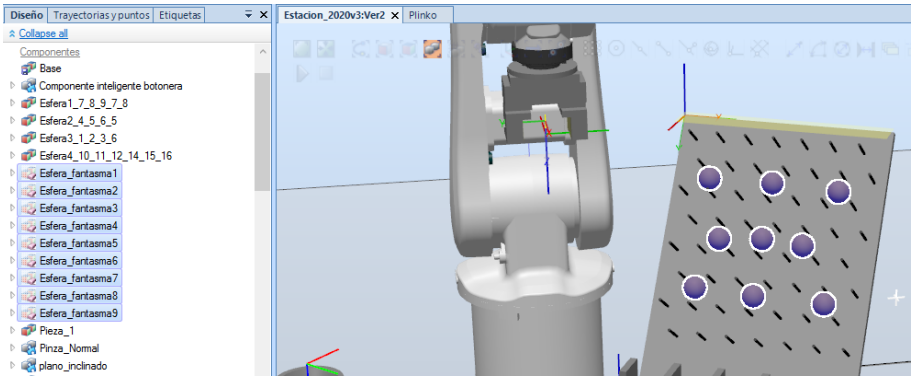


Ilustración 74: Esferas fantasma.

El comportamiento físico de dichas bolas es diferente de las que se encuentran en movimiento, en este caso se ha establecido como inactivas.

Dichas bolas se muestran aleatoria e individualmente tras cierto tiempo, esto se ha logrado de la siguiente forma.

Este efecto solo se debe reproducir después de que el ABB irb120 suelte la bola en el Plinko y no se repita hasta que se suelte la siguiente bola, para ello se ha utilizado un bloque LogicSRLatch el cual presenta dos entradas, Set y Reset, y dos salidas, Output e InvOutput.

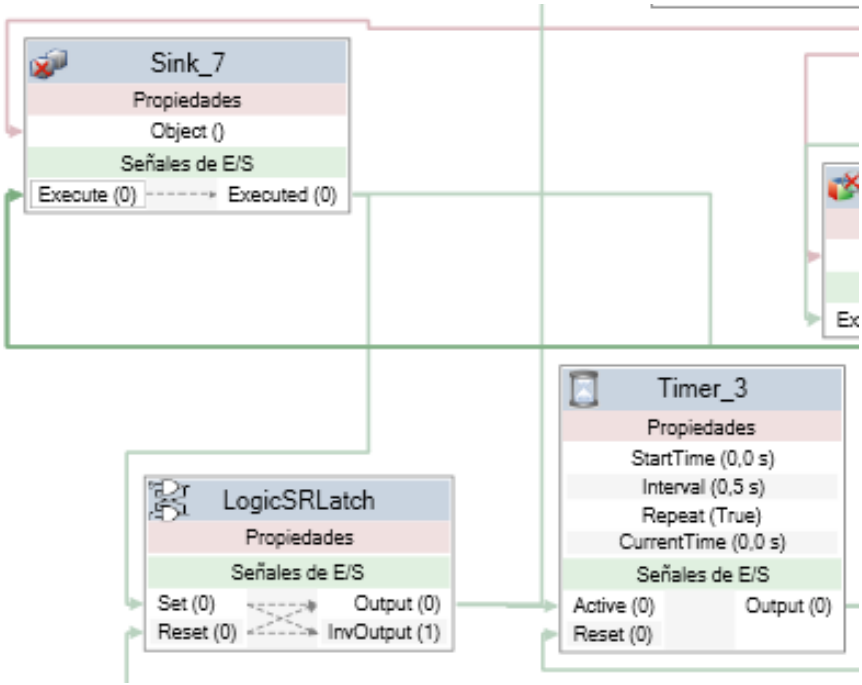


Ilustración 75: Activación del efecto y tiempo entre apariciones.

Cuando LogicSRLatch reciba por Set un '1' activa la salida "Output" y esta se mantiene en ese estado, independientemente del valor de Set, hasta recibir por la entrada Reset otro nivel alto el cual desactiva dicha salida, pasando de nivel alto a bajo.

La entrada Set de este bloque se activa tras la eliminación de la bola original y la salida Output activa un temporizador de medio segundo que es el tiempo que se muestra cada bola fantasma.

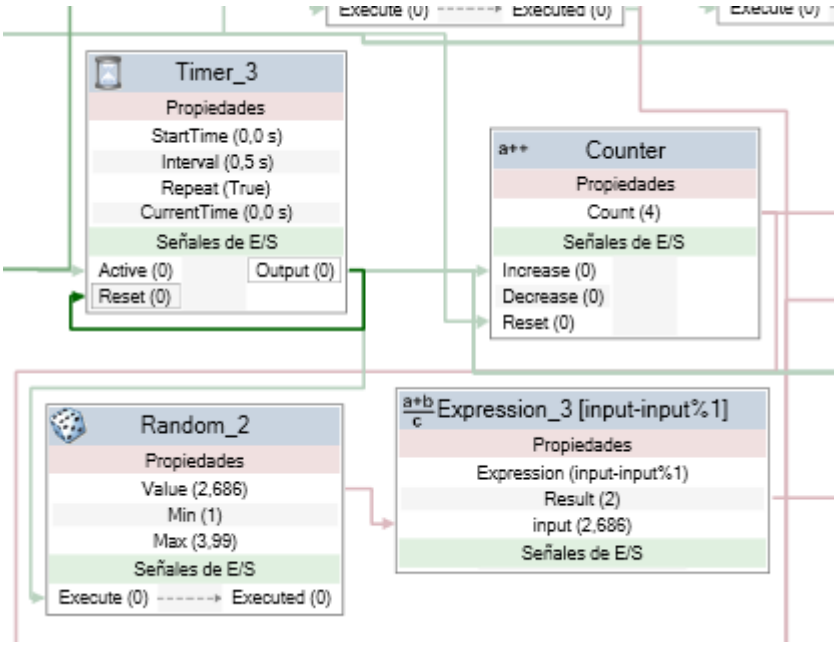


Ilustración 76: Reset del temporizador y contador.

Cada vez que pasa medio segundo se incrementa la cuenta de un contador, se ejecuta un bloque generador de números aleatorios, como en el caso anterior, y se reinicia el temporizador.

El resultado proporcionado por Random_2 se envía a un bloque el cual, de igual forma que en el caso anterior, lo trunca.

Existen nueve bolas fantasmas en el Plinko, de las cuales se muestran tres aleatorias en cada ejecución. Para dar la ilusión de que la bola va cayendo se utiliza un contador, un bloque generador de números aleatorios y una expresión que trunca el resultado.

		RANDOM_2		
		1	2	3
CONTADOR	1	Bola fantasma 1	Bola fantasma 2	Bola fantasma 3
	2	Bola fantasma 4	Bola fantasma 5	Bola fantasma 6
	3	Bola fantasma 7	Bola fantasma 8	Bola fantasma 9

Tabla 10: Bolas fantasmas.

La **Tabla 10** muestra la matriz con la que se hacen aparecer las bolas. El contador indica la fila en la cual se encuentra una de bolas que se debe mostrar, por lo que si la cuenta vale uno se elige una de las tres bolas que se muestran en la fila uno, para elegir la bola se emplea el bloque random y la expresión que trunca el resultado obteniendo un número del 1 al 3 que corresponde con la columna de dicha matriz.

De esta manera y repitiendo la operación tres veces a la vez que se incrementa el valor del contador se eligen las bolas que deben ir apareciendo.

El contador se reinicia cada vez que se elimina la bola original de esta forma tras soltar dicha bola aparece una bola fantasma en la primera fila.

Tras conocer la fila y la columna de la bola fantasma que debe aparecer se debe ejecutar un bloque “Show” asociado con la bola que se va a mostrar para ello se utilizan unos comparadores.

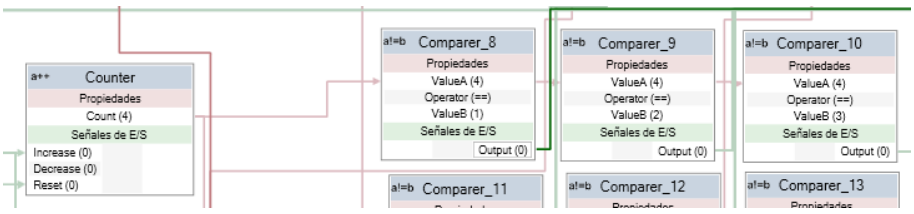


Ilustración 77: Comparadores que indican la fila.

En caso de que la cuenta sea igual a 1 el comparador 8 presenta una salida a nivel alto mientras que el resto de los comparadores, comparador 9 y 10, presentan una salida a nivel bajo.

Lo mismo ocurre para hacer aparecer la bola de una columna, en este caso la entrada a los comparadores es el numero aleatorio tras ser truncado.

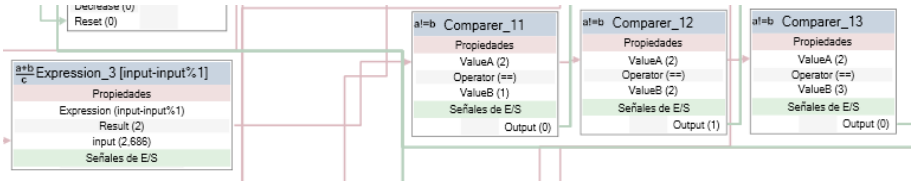


Ilustración 78: Comparadores que indican la columna.

Tres comparadores reciben como entrada el valor del contador y se utilizan para obtener una señal activa que indica la fila en la que va a aparecer una bola fantasma.

Después de estas comparaciones se obtienen seis valores binarios los cuales están relacionados con la posición de las bolas en el Plinko para seleccionar la bola se deben hacer pasar por unas expresiones lógicas las cuales están sincronizadas con un temporizador cuya salida reinicia su cuenta y presenta un periodo de una décima de segundo.

Las expresiones lógicas que se muestran en la **Ilustración 79** son nueve, una por bola, y ejecutan el bloque Show correspondiente a la bola que se encuentra en una posición determinada por su fila y columna.

Estas expresiones presentan tres entradas y dos puertas AND, en el caso de que el temporizador, el comparador de filas y el comparador de columnas conectados al bloque lógico se activen la salida se activa.

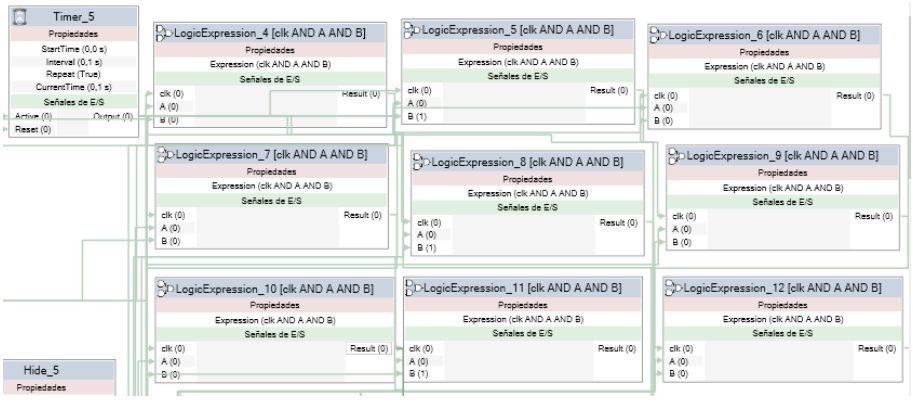


Ilustración 79: Expresiones lógicas que ejecutan el bloque Show correspondiente

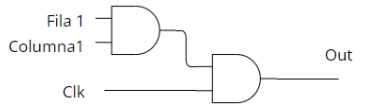
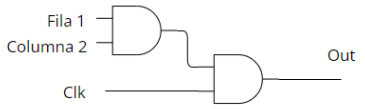
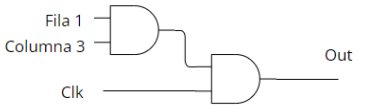
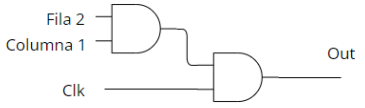
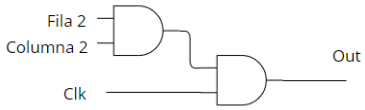
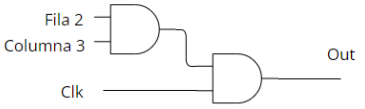
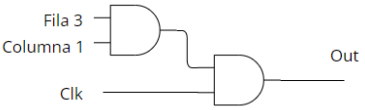
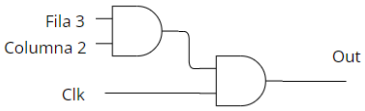
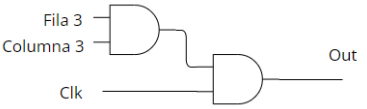
Nombre del bloque	Expresión lógica	Diagrama
LogicExpression_4	Fila 1 AND Columna 1 AND Clk	
LogicExpression_5	Fila 1 AND Columna 2 AND Clk	
LogicExpression_6	Fila 1 AND Columna 3 AND Clk	
LogicExpression_7	Fila 2 AND Columna 1 AND Clk	
LogicExpression_8	Fila 2 AND Columna 2 AND Clk	
LogicExpression_9	Fila 2 AND Columna 3 AND Clk	
LogicExpression_10	Fila 3 AND Columna 1 AND Clk	
LogicExpression_11	Fila 3 AND Columna 2 AND Clk	
LogicExpression_12	Fila 3 AND Columna 3 AND Clk	

Tabla 11: Expresiones lógicas.

Los comparadores ejecutan los bloques Show correspondientes con cada bola.

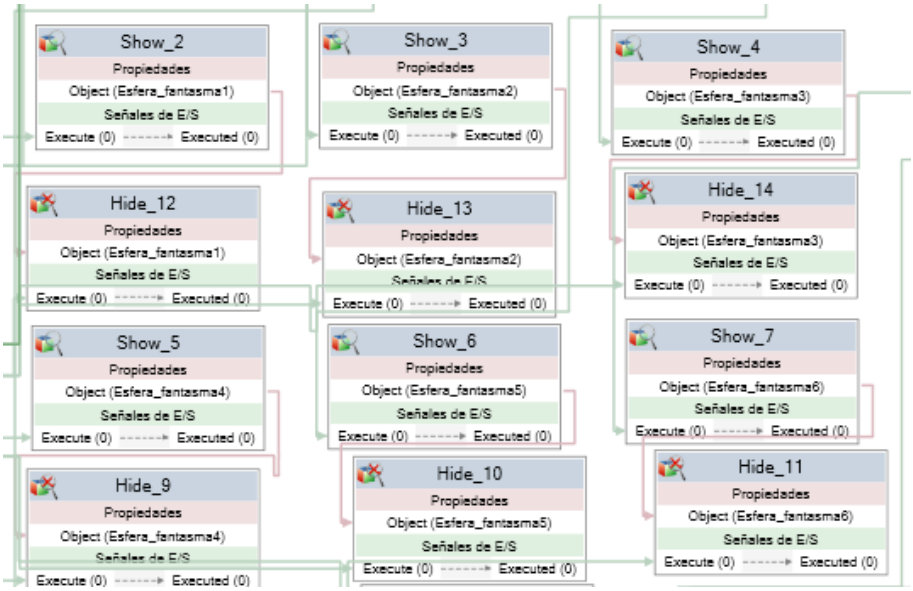


Ilustración 80: Bloques Show y Hide.

Los bloques Hide se ejecutan con la salida a nivel alto del temporizador anteriormente explicado Timer_3.

Primero se ocultan todas las bolas y después se van mostrando las indicadas esto es así debido al temporizador que se encuentra como entrada en todas las expresiones lógicas el cual añade un retardo a la aparición de las bolas logrando que el bloque Hide oculte la anterior bola mostrada y tras esto se muestre la siguiente hasta el siguiente flanco en el que se ocultan todas las bolas nuevamente y se repite el proceso.

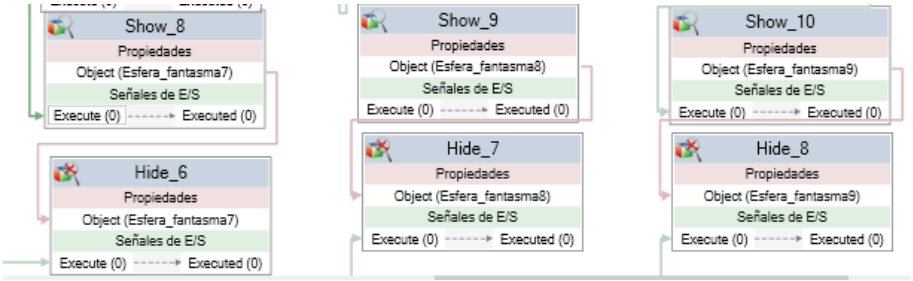


Ilustración 81: Últimos Bloques Show y Hide.

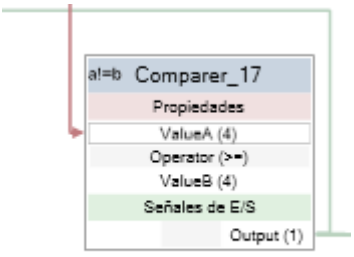


Ilustración 82: Comparador de cuenta.

Cuando la cuenta es mayor o igual a cuatro se activa la salida de este comparador y resetea el LogicSRLatch del inicio finalizando la simulación de la caída de la bola, ejecuta los últimos bloques Hide correspondientes con la última fila de bolas fantasma y un temporizador de una décima de segundo que retrasa la aparición de la bola en el casillero donde se ha copiado la original en el inicio del proceso.

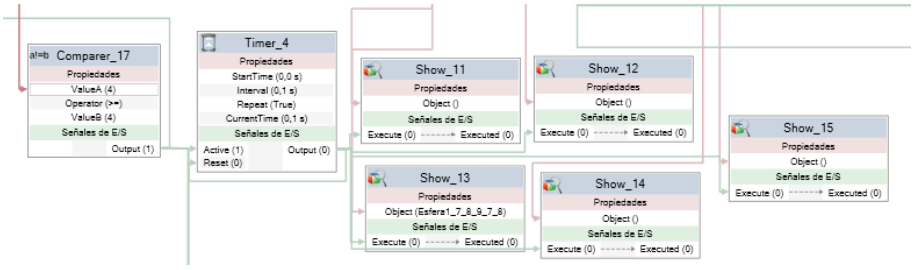


Ilustración 83: Aparición de la bola en el casillero.

Por último, para que el robot detecte las bolas en el casillero se crean cinco sensores planares, uno por casillero y se conectan las salidas de estos a las salidas creadas en el componente inteligente.

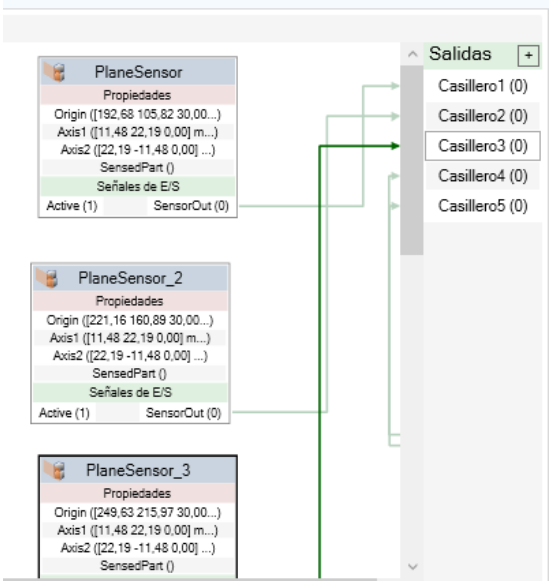


Ilustración 84: Sensores planares y la salida del SmartComponent.

2.6.4 Lógica de estación

Con el fin de simular el funcionamiento de los sensores y de los actuadores se deben conectar las entradas y salidas de los componentes inteligentes con la controladora, para ello se selecciona lógica de estación, la cual se encuentra en la pestaña de simulación.

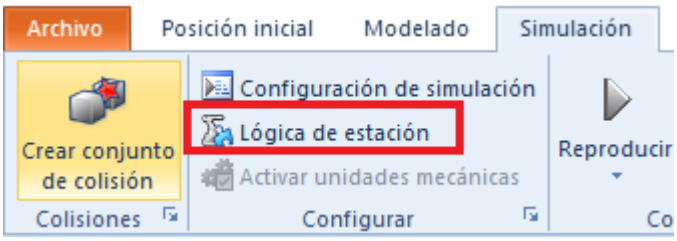


Ilustración 85: Lógica de estación.

Tras seleccionar Lógica de estación aparece una ventana en la cual se puede ver los componentes inteligentes de la estación con las entradas y salidas creadas conectadas a la controladora.

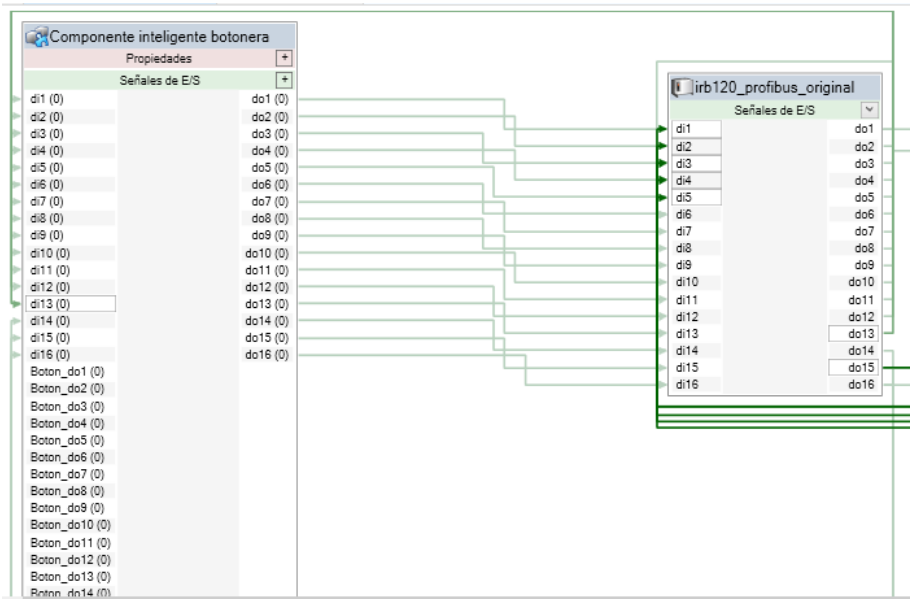


Ilustración 86: Botonera y controladora en la lógica de estación.

Como se observa en la imagen las salidas de la botonera están conectadas con las entradas de la controladora y las salidas de la controladora están conectadas con las entradas de la botonera.

BOTONERA	IRB120
Di1	Do1
Di2	Do2
Di3	Do3
Di4	Do4
Di5	Do5
Di6	Do6
Di7	Do7
Di8	Do8
Di9	Do9
Di10	Do10

Di11	Do11
Di12	Do12
Di13	Do13
Di14	Do14
Di15	Do15
Di16	Do16
Do1	Di1
Do2	Di2
Do3	Di3
Do4	Di4
Do5	Di5
Do6	Di6
Do7	Di7
Do8	Di8
Do9	Di9
Do10	Di10
Do11	Di11
Do12	Di12
Do13	Di13
Do14	Di14
Do15	Di15
Do16	Di16

Tabla 12: Conexionado de las entradas y salidas de la botonera con la controladora.

En el caso de la pinza se utilizan dos entradas y una salida y se conectan a la entrada di16 y las salidas do15 y do16.

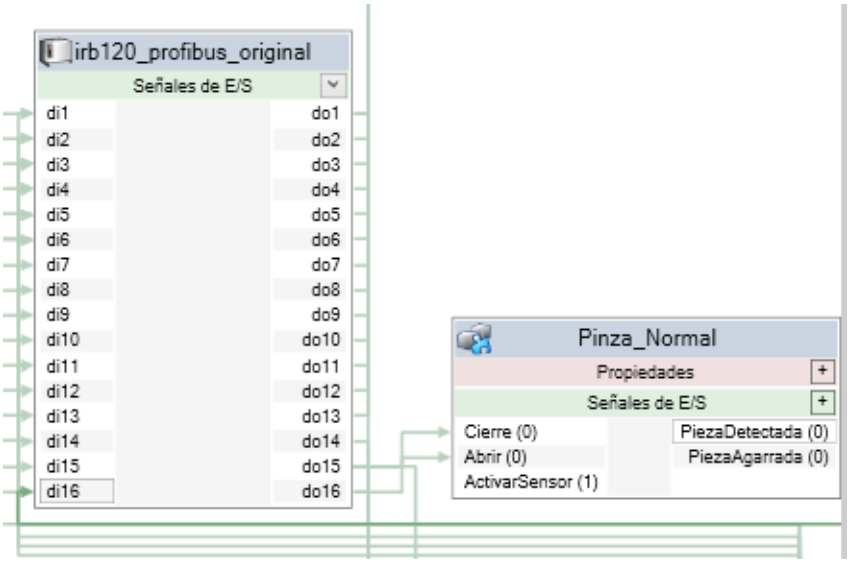


Ilustración 87: Pinza y controladora en la lógica de estación.

Pinza_Normal	IRB120
Cierre	Do16
Abrir	Do15
PiezaDetectada	Di16

Tabla 13: Conexionado de las entradas y salidas de la pinza con la controladora.

En el plano inclinado se utilizan dos entradas, la activación del pistón que simula la apertura de la electroválvula y apertura_pinza que sincroniza el componente inteligente con el robot, y una salida que simula el sensor que indica que hay una bola disponible.

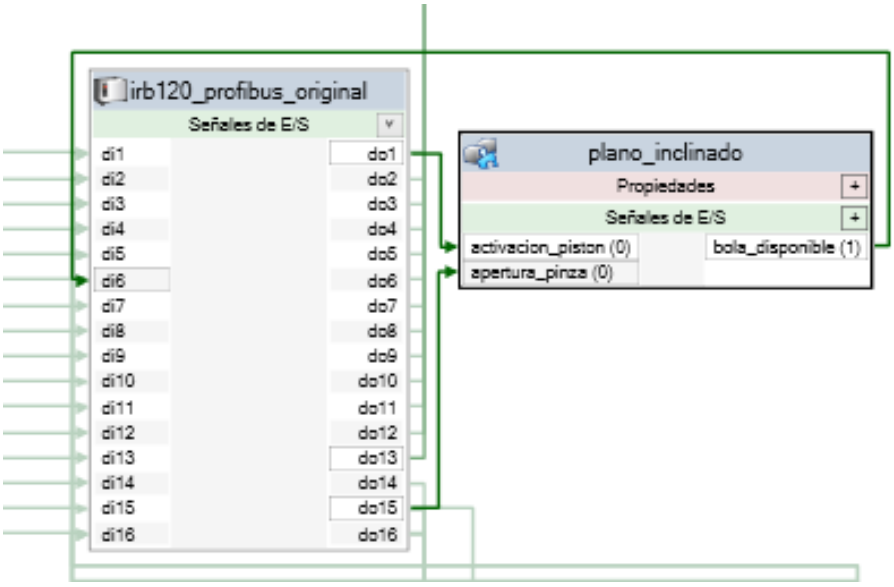


Ilustración 88: Pinza y controladora en la lógica de estación.

Plano_inclinado	IRB120
activacion_piston	Do1
apertura_pinza	Do15
bola_disponible	Di6

Tabla 14: Conexionado de las entradas y salidas del plano inclinado con la controladora.

Por último, el Plinko tiene una entrada que indica en que momento se abre la pinza, al igual que en el plano inclinado esto se usa para sincronizarlo con los movimientos del ABB irb120, y cinco salidas las cuales simulan los finales de carrera que se encuentran en los casilleros e indican donde ha caído la bola.

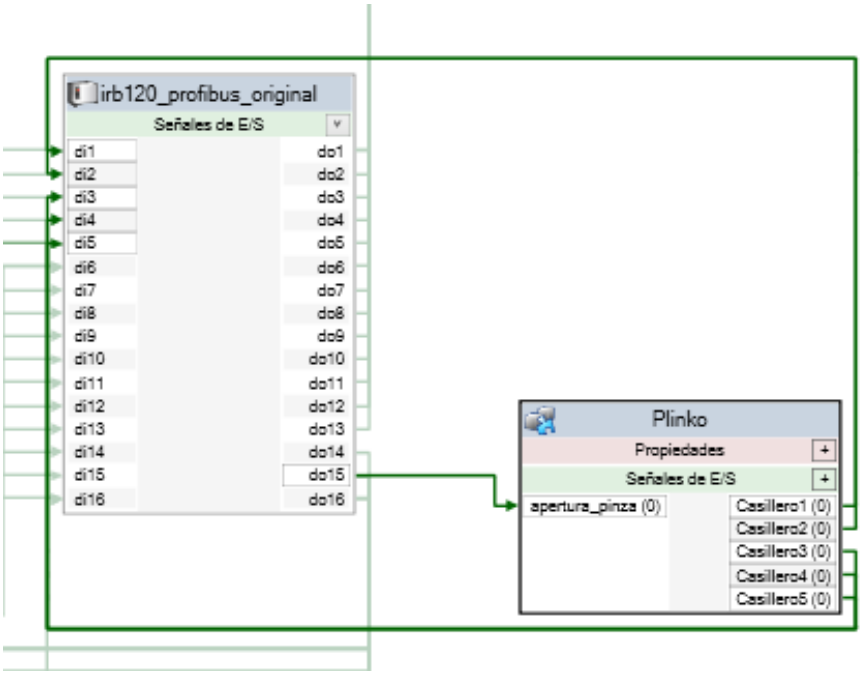


Ilustración 89: Plinko y controladora en la lógica de estación.

Plinko	IRB120
Apertura_pinza	Do15
Casillero1	Di1
Casillero2	Di2
Casillero3	Di3
Casillero4	Di4
Casillero5	Di5

Tabla 15: Conexionado de las entradas y salidas del Plinko con la controladora.

3. Software desarrollado para la implantación en la práctica docente

El juego del Plinko se basa en cuatro modos de funcionamiento:

1. Posicionamiento en el Plinko: Modo en el cual se puede decidir desde que parte del Plinko se quiere dejar caer la bola.
2. Movimiento manual: Este modo es un valor añadido al proyecto ya que no está relacionado con el juego del Plinko, pero si con el espacio de trabajo en el que se mueve. Este modo permite modificar la posición de los ejes de forma manual, es decir, decidir la posición articular de todos los ejes evitando que el TCP choque con alguna de las maquetas.
3. Elección del casillero: Tras un estudio estadístico, se logra conocer cuáles son los casilleros en los que son más probables que acabe la bola sabiendo desde donde se ha dejado caer.
4. Movimiento automático: En este modo el robot deja caer la bola en un lugar aleatorio de la parte superior del Plinko y recoge o no la bola dependiendo de las instrucciones que envíe el usuario.

Todos estos modos de funcionamiento van complementados con una comunicación con unos displays donde se muestra un recuento de las bolas que han caído en cada casillero y la probabilidad de que caiga en cada uno de los casilleros conociendo donde se va a soltar la bola.

Además, se puede controlar el ABB desde una página web desde la cual se envían las instrucciones presentando al usuario una interfaz más intuitiva, universal y amigable desde la cual gobernar el robot.

Todas estas funcionalidades se logran mediante una adecuada programación del ABB y de los autómatas.

El código de estos se encuentra detallado en este apartado.

3.1 Código del ABB

En este apartado no se van a explicar todas las funciones y procesos programados, sin embargo, si los que considero más relevantes.

3.1.1 Variables globales

La programación del ABB se realiza mediante código de programación en Rapid.

Para este proyecto se ha realizado un módulo, M_TFG, en el cual se estructura todo el funcionamiento del robot.

En este módulo se encuentran una serie de variables globales declaradas con el fin de que puedan ser utilizadas desde cualquier función o proceso.

```
!Variables
Local PERS num seed:=406;
LOCAL CONST speeddata v{3}:=[v200,v100,v50];
LOCAL VAR num bolas:=0;
LOCAL VAR num web;
LOCAL VAR num repeticiones:=0;!variable que indica el numero de repeticion en modo automatico
LOCAL VAR bool repeticion:=TRUE;
LOCAL VAR bool funcionamiento:=TRUE;
LOCAL VAR bool errbolas:=TRUE;
LOCAL VAR bool errsockets:=TRUE;
LOCAL VAR bool colision:=FALSE;
LOCAL VAR bool automatico_web:=TRUE;
```

Ilustración 90: Variables globales del proyecto.

- La variable “seed” es una variable persistente y es utilizada para la obtención de números pseudoaleatorios.
- El vector constante “v” presenta tres parámetros los cuales corresponden con las velocidades del ABB irb120 en los distintos movimientos.
- La variable “web” es una variable numérica utilizada para los formularios que se observan desde la flexpendant.
- “repeticiones” es la variable que indica el número de repetición en modo automático.
- “repetición” es una variable booleana cuya finalidad es la de establecer un bucle en el que se repite un modo de funcionamiento hasta que se indica la intención de abandonarlo.
- “funcionamiento” es una variable usada en un bucle donde se encuentra la selección del modo de funcionamiento.
- La variable “errbolas” se encarga de salir de una interrupción.
- La variable “errsockets” indica si se ha establecido comunicación con el controlador esp32.
- “colisión” es una variable que indica si se ha establecido alguna colisión del TCP con las maquetas de la estación.

La variable “automático_web” se gestiona desde una interrupción y se encarga de indicar si se quiere de salir del modo automático cuando el control se realiza desde la página web.

3.1.2 Función principal

El proceso principal de dicho módulo se llama Main_TFG y es el encargado de llamar al resto de funciones y procesos que se encuentran declarados de forma local.

En este proceso se encuentran declaradas variables y las llamadas a las interrupciones creadas.

```
PROC Main_TFG()

    !___________VARIABLES_______
    VAR num opcion;
    VAR num recoger;
    VAR num posicion_plinko;
    VAR num displays;
    VAR num modo;
    VAR num valor;!valor que se procede de la instruccuion web
    VAR string stval;
    VAR string inf:="";
    VAR bool menu:=TRUE;
    VAR bool reintentar_conexion:=TRUE;
    VAR num articulaciones{6};
```

Ilustración 91: Variables locales del módulo Main_TFG.

El módulo Main_TFG requiere el uso de una serie de variables las cuales solo son utilizadas en este módulo.

Estas variables solo pueden ser utilizadas en Main_TFG y cuya función es la siguiente.

- La variable “opción” se utiliza en un formulario de la Flexpendant en el cual se decide si se quiere repetir el modo o salir de este.
- “recoger” es una variable usada en un formulario de la Flexpendant e indica si se quieren recoger las bolas de los casilleros.
- “posición_plinko” es la variable que indica donde se debe colocar la bola en la parte superior del Plinko.
- “displays” se utiliza en un formulario de la Flexpendant en el cual se decide si se quiere establecer comunicación con los displays.
- La variable “modo” indica que modo de funcionamiento se utiliza.
- “valor” es el valor que procede de la instrucción web.
- “stval” se utiliza en el modo de posicionamiento en Plinko en el que se establece la posición desde donde va a dejar caer la bola.
- La variable “inf” es la variable que almacena toda la trama que se recibe desde los displays.

- “menú” es una variable que te permite salir del bucle más externo finalizar el proceso y terminar de recorrer el proceso Main_TFG.
- La variable “reintentar conexión” se utiliza para indicar si se quiere reintentar la conexión con los displays.
- Por último, el vector “articulaciones” se utiliza para almacenar el valor de los ejes del robot a los que se debe posicionar tras recibir la orden de movimiento manual.

```
!________INTERRUPCIONES____________

!ERROR BOLAS
CONNECT Errorbolas WITH Error_bolas;
ITimer 5, Errorbolas;
ISleep Errorbolas;
!DESCONEXION WEB
CONNECT SalidaWeb WITH Salida_web;
ISignalDI di8,high,SalidaWeb;
ISleep SalidaWeb;
!COLISIONES
CONNECT colisiones WITH colision_m2;
ISignalDI di7,high,colisiones;
ISleep colisiones;
!SALIDA AUTOMATICO
CONNECT SalidaAutomatico WITH Salida_automatico;
ISignalDI di7,high,SalidaAutomatico;
ISleep SalidaAutomatico;
```

Ilustración 92: Declaración de las interrupciones en el Main_TFG.

Hay cuatro interrupciones tres se realizan con las entradas digitales di7 y di8, estas son:

- Salida web: Para salir del modo web y volver al control desde la flexpendant.

```
TRAP Salida_web
    !TPWrite("En la interrupcion");
    Envia_instruccion("stop");
    repeticion:=FALSE;
    funcionamiento:=FALSE;
    web:=1;
    ISleep SalidaWeb;
ENDTRAP
```

Ilustración 93: Interrupción Salida_web.

- Colisiones: Se utiliza para evitar que el robot colisione con alguna maqueta y el control de colisiones no lo haya detectado.

```
TRAP colision_m2
    colision:=TRUE;
ENDTRAP
```

Ilustración 94: Interrupción Colisiones.

- SalidaAutomatico: Se utiliza para dejar de repetir el modo automático antes de que hayan completado todas las repeticiones.

```
TRAP salida_automatico
    !TPWrite("En la interrupcion");
    automatico_web:=FALSE;
    ISleep SalidaAutomatico;
ENDTRAP
```

Ilustración 95: Interrupción Salida_automático.

La única interrupción que no se activa con una entrada digital es la de error bolas, y es la encargada de tratar los errores derivados de que se produzca un fallo en la simulación o en el caso de que se produzca un fallo durante el proceso, como el caso de que la bola no active el final de carrera del casillero.

Las interrupciones se deben declarar como variables globales.

```
!interrupciones
LOCAL VAR intnum Errorbolas;
LOCAL VAR intnum SalidaWeb;
LOCAL VAR intnum colisiones;
LOCAL VAR intnum SalidaAutomatico;
```

Ilustración 96: Declaración de las interrupciones.

Tras la declaración de las interrupciones se empieza a estructurar el código de la función principal.

```
WHILE menu=TRUE DO
    funcionamiento:=TRUE;
    repeticion:=TRUE;
    reintentar_conexion:=TRUE;
    !_______COMUNICACION SOCKETS_____
    TPWrite "¿Comunicar con displays?";
    TPWrite "SI";
    TPWrite "NO";
    TPWrite "Salir";
    TPReadFK displays, "Opcion= ","SI","NO","Salir","","";
    IF...ENDIF

    WHILE...ENDWHILE
ENDWHILE
```

Ilustración 97: Bucle principal

Todas las funciones del proyecto se encuentran dentro de un bucle que evalúa la variable menú, esta variable en caso de ser false sale del bucle y termina el módulo.

El bucle establece las variables de "funcionamiento", "repetición" y "reintentar_conexión" a "true" y muestra un menú en el cual se le da valor a la variable display, en caso de querer conectar el ABB irb120 con los displays adquiere el valor de "1" en caso de no querer conectarlos adquiere el valor de "2" y en caso de querer salir adquiere el valor de "3".

```
IF displays=1 THEN
    WHILE...ENDWHILE
!_______CONEXION WEB_________
    IF errsockets=FALSE THEN
        TPWrite "¿Control desde la flexpendant o desde la pagina web?";
        TPWrite "flexpendant";
        TPWrite "web";
        TPReadFK web, "Opcion= ","flexpendant","web","","","";
        IF...ENDIF
    ENDIF
ELSEIF displays=2 THEN
    SocketClose server_socket;
    SocketClose client_socket;
    web:=1;
ELSEIF displays=3 THEN
    SocketClose server_socket;
    SocketClose client_socket;
    repeticion:=FALSE;
    funcionamiento:=FALSE;
    menu:=FALSE;
    web:=1;
ENDIF
```

Ilustración 98: Evaluación de la variable display

Tras esto se evalúa el valor asignado a la variable "display" y se realizan los siguientes procesos.

En el caso de que la variable “display” sea igual a “1” se realizan las funciones y procesos del bucle debido a que en un primer momento la variable “reintentar_conexión” es igual a true.

Se abren los sockets y en caso de no haberlo conseguido se pregunta al usuario si quiere reintentar la conexión, en caso afirmativo se vuelve a intentar y si se consigue la comunicación o se decide no comunicarles se sale del bucle.

```
IF displays=1 THEN
    WHILE reintentar_conexion=TRUE DO
        Abrir_Sockets;
        IF displays=2 OR errsockets=FALSE THEN
            reintentar_conexion:=FALSE;
        ELSE
            TPWrite "¿Reintentar conexion?";
            TPWrite "SI";
            TPWrite "NO";
            TPReadFK displays, "Opcion= ","SI","NO","","","";
        ENDIF
    ENDWHILE
!_______CONEXION WEB__________
    IF errsockets=FALSE THEN
        TPWrite "¿Control desde la flexpendant o desde la pagina web?";
        TPWrite "flexpendant";
        TPWrite "web";
        TPReadFK web, "Opcion= ","flexpendant","web","","","";
        IF...ENDIF
    ENDIF
```

Ilustración 99: Menú de comunicación con los displays

Tras salir del bucle y si se ha comunicado con los displays se vuelve a mostrar otro cuestionario en el que se pregunta al usuario si se quiere establecer el control desde la flexpendant o desde la página web.

```
TPWrite "¿Control desde la flexpendant o desde la pagina web?";
TPWrite "flexpendant";
TPWrite "web";
TPReadFK web, "Opcion= ","flexpendant","web","","","";
IF web=2 THEN
    Envia_instruccion("web");
    WHILE Recibe_instruccion() <> "ok" DO
    ENDWHILE
    IWatch SalidaWeb;
    TPWrite "Puedes cerrar el control web pulsando di8";
    TPWrite "Se cerrará el control tras haber finalizado el proceso";
ENDIF
```

Ilustración 100: Menú de comunicación con la página web o la flexpendant

Se evalúa el valor de web y si es igual a “2” se envía a los displays “web” y espera a recibir “ok” del controlador, después de esto se activa la interrupción “SalidaWeb” y se muestran unas instrucciones para salir de este modo.

```
ELSEIF displays=2 THEN
    SocketClose server_socket;
    SocketClose client_socket;
    web:=1;
ELSEIF displays=3 THEN
    SocketClose server_socket;
    SocketClose client_socket;
    repeticion:=FALSE;
    funcionamiento:=FALSE;
    menu:=FALSE;
    web:=1;
ENDIF
```

Ilustración 101: Menú de comunicación con los displays

En el caso de que la variable “display” sea igual a “2” o “3” se cierra el cliente y el servidor de sockets y se da el valor de “1” a web. Si “display” es igual a “3” se sale de los bucles repetición, funcionamiento y menú terminando así la ejecución del proceso principal.

```
WHILE funcionamiento=TRUE DO
    repeticion:=TRUE;
    salto:
    IF web=1 THEN
        TPWrite "Modo de funcionamiento:";
        TPWrite "Elige posicion en plinko";
        TPWrite "Manual";
        TPWrite "Elige casillero";
        TPWrite "Normal";
        TPWrite "Salir";
        TPReadFK modo, "Opcion= ","Posicion Plinko","Manual","Casillero","Normal","Salir";
    ELSEIF web=2 THEN
        inf:=Recibe_instruccion();
        TPWrite inf;
        !TPWrite informacion(inf,1);
        TEST...ENDTEST
        IF inf="" THEN
            modo:=6;
        ENDIF
    ENDIF
```

Ilustración 102: Bucle del funcionamiento

Tras el menú en el que decide si se quiere conectar con los displays y si se quiere gobernar el ABB irb120 desde la página web, se entra en un bucle y se elige el modo de funcionamiento desde el menú de la flexpendant o desde la interfaz web.

La ejecución del programa se encontrará dentro de dicho bucle hasta que se decida salir.

En el caso de que se decida la comunicación web la trama que recibe ya indica el modo en el que se encuentra por lo que directamente se evalúa la trama, se establece el modo y se trata la información recibida.

```
ELSEIF web=2 THEN
    inf:=Recibe_instruccion();
    !TPWrite inf;
    !TPWrite informacion(inf,1);
    TEST informacion(inf,1)
    CASE "m1":
        modo:=1;
    CASE "m2":
        modo:=2;
    CASE "m3":
        modo:=3;
    CASE "m4":
        modo:=4;
    CASE "stop":
        modo:=5;
    ENDTEST
    IF inf="" THEN
        modo:=6;
    ENDIF
ENDIF
```

Ilustración 103: Selección del modo en el caso web

Se recibe la información y se almacena en una variable llamada “inf”, esta información contiene el modo y la información que este modo requiere para que el robot pueda ejecutar las tareas programadas.

Tanto los modos como los datos se envían separados por una “/” lo que permite conocer en qué momento finalizan los datos y la trama.

Se estudia la variable “inf” con la función información y dependiendo de la cadena recibida se asigna un valor a la variable “modo” la cual también se utiliza en el menú que se visualiza en la flexpendant.

En el caso de que no se haya recibido nada la variable “modo” adquiere el valor “6” evitando así todos los bucles y volviendo a esperar a que reciba algo.

```
WHILE repeticion=TRUE DO

    IF modo=1 OR modo=3 OR modo=4 THEN
        IF...ENDIF
    ELSEIF modo=2 THEN
        TPWrite "En caso de colision con alguna maqueta pulsar di7";
        IWatch colisiones;
        IF...ENDIF
        Movimiento_manual(articulaciones);
        ISleep colisiones;
    ELSEIF modo=5 THEN
        repeticion:=FALSE;
        funcionamiento:=FALSE;
        web:=1;
        ISleep SalidaWeb;
    ENDIF

    IF...ENDIF

    IF...ENDIF
ENDWHILE
```

Ilustración 104: Bucle de repetición de modo

Tras la selección del modo, desde la flexpendant o desde la página web, se evalúa un bucle el cual se ejecuta al menos una vez debido a que el valor de la variable “repetición” es igual a true desde el inicio.

En este bucle se evalúan el modo 1, el modo 3 y el modo 4 de forma muy similar ya que en todos ellos la secuencia de tareas que ejecuta el robot es muy similar con pequeñas diferencias.

```
IF modo=1 THEN
    IF web=2 THEN
        TPWrite informacion(inf,2);
        stval:=informacion(inf,2);
        valor:=str_num(stval);
        TPWrite "Tramo: " \Num:=seleccion_tramo(valor);
        !proceso modo \posicion:=valor;
    ELSE
        TPWrite "Posición: Elige un numero entre 0 y 240";
        valor:=histeresis(240,0);
        !controla_display(probabilidades{seleccion_tramo(valor)});
        !proceso modo \posicion:=valor;
    ENDIF
    controla_display(probabilidades{seleccion_tramo(valor)});
    proceso modo \posicion:=valor;
```

Ilustración 105: Modo 1.

En el modo 1 se decide en que posición del Plinko se quiere dejar caer la bola, para ello se distinguen los dos modos por los cuales puede llegarle dicha información, el primero es desde la página web para ello utiliza la función información y se selecciona el segundo tipo de dato de la trama ya que la trama para este modo es de la siguiente forma.

Trama:“modo”/”posición”/

El valor obtenido de la trama se le asigna a la variable “stval” la cual es utilizada como variable auxiliar para el uso de la función “str_num” la cual transforma una cadena de caracteres en un número.

Cuando la información se recibe desde la flexpendant se utiliza una función llamada histéresis en la cual se establecen los márgenes superior e inferior del rango en el cual se puede soltar la bola.

Seguido de esto, se envía la información a los displays indicándoles que números deben aparecer. En este caso aparecen las probabilidades de que caiga en alguno de los casilleros.

El modo 2 es el modo manual y presenta su propio tratamiento de la información recibida y el modo 5 equivale a salir de los bucles donde se evalúan las variables repetición y funcionamiento y volver al bucle general donde se evalúa la variable menú.

Por último, se ejecuta el proceso indicándole como variable opcional la posición desde la cual se quiere dejar caer la bola.

```
ELSEIF modo=3 THEN
    IF web=2 THEN
        valor:=str_num(informacion(inf,2));
        proceso modo \casillero:=valor;
        !IF StrToVal(informacion(inf,2),valor) THEN
        !    !TPWrite "posicion:" \Num:=valor;
        !    proceso modo \casillero:=valor;
        !ENDIF
    ELSE
        TPWrite "Elección casillero:";
        TPWrite "Casillero 1";
        TPWrite "Casillero 2";
        TPWrite "Casillero 3";
        TPWrite "Casillero 4";
        TPWrite "Casillero 5";
        TPReadFK valor, "Opcion= ","Casillero 1","Casillero 2","Casillero 3","Casillero 4","Casillero 5";
        proceso modo \casillero:=valor;
    ENDIF
```

Ilustración 106: Modo 3.

En este modo de funcionamiento se elige en que casillero se quiere que caiga la bola, esto se logra gracias a un estudio estadístico en el cual se indican los tramos donde es más probable que acabe la bola tras ser soltada desde un punto en concreto de la parte superior del Plinko.

Este modo, al igual que el resto, evalúa la información de dos formas distintas, en el caso de que proceda de la página web o desde la flexpendant.

Si la información procede de la página web se obtiene el valor numérico enviado de igual forma que en el caso anterior y se envía en una variable opcional distinta dicho valor.

La trama enviada en este caso es de la siguiente forma.

“modo”/”casillero donde debe acabar la bola”/

En el caso de que la información se obtenga desde menú en la flexpendant la variable ligada a dicho menú es la variable “valor”.

```
ELSEIF modo=4 THEN
    IF web=2 THEN
        IWatch SalidaAutomatico;
        TPWrite "se repetirá 10 veces en caso de querer finalizar pulsar di7";
        repeticiones:=0;
    ELSE
        repeticiones:=9;
    ENDIF
    !TPWrite ""\Num:=repeticiones;
    !TPWrite ""\Bool:=automatico_web;
    WHILE repeticiones<10 AND automatico_web=TRUE DO
        TPWrite "se debe reproducir";
        controla_display(contador_casilleros());
        proceso(modo);
        repeticiones:=repeticiones+1;
    ENDWHILE
    !IF web=2 THEN
        ISleep SalidaAutomatico;
    !ENDIF
    repeticiones:=0;
ENDIF
```

Ilustración 107: Modo 4.

El modo 4 es el modo automático en él la posición desde la cual se tira la bola se obtiene de forma aleatoria, en el caso de que el control se realice desde la página web el número de repeticiones es diez mientras que en el caso de la flexpendant solo es uno.

Como en el control desde la web el número de repeticiones es muy elevado se ha creado una interrupción la cual solo se activa cuando se entra en el modo desde ese tipo de control y se desactiva al salir.

En el proceso que es repite cada vez que se tira una bola se envía una actualización del recuento de bolas en cada casillero para que se muestren en los displays, se ejecuta el proceso conforme con el modo seleccionado y se aumenta en uno el valor de la variable repeticiones.

```
ELSEIF modo=2 THEN
    TPWrite "En caso de colision con alguna maqueta pulsar di7";
    IWatch colisiones;
    IF web=2 THEN
        FOR i FROM 2 TO 7 DO
            valor:=str_num(informacion(inf,i));
            articulaciones{(i-1)}:=valor;
        ENDFOR
    ELSE
        TPWrite "EJE1: Elige un numero entre 165 y -165";
        articulaciones{1}:=histeresis(165,-165);
        TPWrite "EJE2: Elige un numero entre 110 y -110";
        articulaciones{2}:=histeresis(110,-110);
        TPWrite "EJE3: Elige un numero entre 70 y -110";
        articulaciones{3}:=histeresis(70,-110);
        TPWrite "EJE4: Elige un numero entre 160 y -160";
        articulaciones{4}:=histeresis(160,-160);
        TPWrite "EJE5: Elige un numero entre 120 y -120";
        articulaciones{5}:=histeresis(120,-120);
        TPWrite "EJE6: Elige un numero entre 150 y -150";
        articulaciones{6}:=histeresis(150,-150);
    ENDIF
    Movimiento_manual(articulaciones);
    ISleep colisiones;
```

Ilustración 108: Modo 2.

El modo de funcionamiento dos es completamente distinto al resto ya que no está relacionado con el juego del Plinko, en este modo los ejes se mueven a una determinada posición articular como si se ejecutase una instrucción “MoveAbsJ”, sin embargo, en este modo de funcionamiento se establecen unas protecciones para evitar que el robot se choque con las maquetas.

Existen dos tipos de protecciones:

Una establecida mediante una interrupción y una entrada de la botonera que se utiliza en caso de que el robot vaya a chocar con alguna de las maquetas.

Y otra protección establecida haciendo uso de una serie de Robtargects declarados en Robotstudio y con los cuales se han creado unos planos matemáticos que el robot no debe traspasar con el TCP.

Al igual que en el resto de los modos la obtención de la información necesaria para ejecutar dicho modo de funcionamiento depende de si esta es enviada desde la flexpendant o desde el entorno web.

En caso de que se envíe desde la flexpendant se hace uso de una función llamada histéresis en la cual se establecen el margen superior e inferior máximos admisibles de cada eje. Sin embargo, si la información es enviada

desde la página web debe ser obtenida y transformada ya que esta se encuentra en una trama que presenta el siguiente aspecto.

“modo”/”articulacion1”/ ”articulacion2”/ ”articulacion3”/
”articulacion4”/ ”articulacion5”/ ”articulacion6”/

Los distintos argumentos de la trama se obtienen mediante la función información y la función “str_num”.

Después de la obtención de las coordenadas articulares se ejecuta la función “Movimiento_manual” la cual requiere la posición de las articulaciones a la cual debe desplazarse.

Por último, se desconecta la interrupción de colisiones ya que solo debe estar activa durante el movimiento de los ejes.

```
IF web=2 THEN
    IF modo<>6 THEN
        Envia_instruccion("finish");
        TPWrite "finish";
    ENDIF
    GOTO salto;
ENDIF
```

Ilustración 109: finalización del proceso.

Después de la finalización del proceso y si el control se realiza desde la página web se envía la palabra “finish” a los displays para permitir al controlador que gestiona la página web el envío de nuevas instrucciones. Inmediatamente se vuelve a la etiqueta salto ya que en cada trama que envía el control web indica a qué modo de funcionamiento se refiere y los argumentos que este necesita para ejecutarse.

```
WHILE funcionamiento=TRUE DO
    repeticion:=TRUE;
    salto:
    IF...ENDIF

    WHILE...ENDWHILE
ENDWHILE
```

Ilustración 110: Etiqueta salto.

Finalmente, en caso de que el control sea desde la flexpendant se pregunta al usuario si quiere repetir el modo de funcionamiento o salir de él regresando de nuevo al menú en el cual se pregunta de nuevo el modo de funcionamiento.

```
IF funcionamiento=TRUE AND web=1 THEN
    TPWrite "Opciones:";
    TPWrite "   Repetir modo";
    TPWrite "   Salir del modo";
    TPReadFK opcion, "Opcion= ","Repetir","Salir","","","";
    IF opcion=2 THEN
        repeticion:=FALSE;
    ENDIF
ENDIF
```

Ilustración 111: Etiqueta salto.

3.1.3 Proceso principal

"Proceso" es el proceso en el cual se ejecutan los principales movimientos del robot. El código de este proceso queda resumido y esquematizado en el siguiente diagrama de flujo.

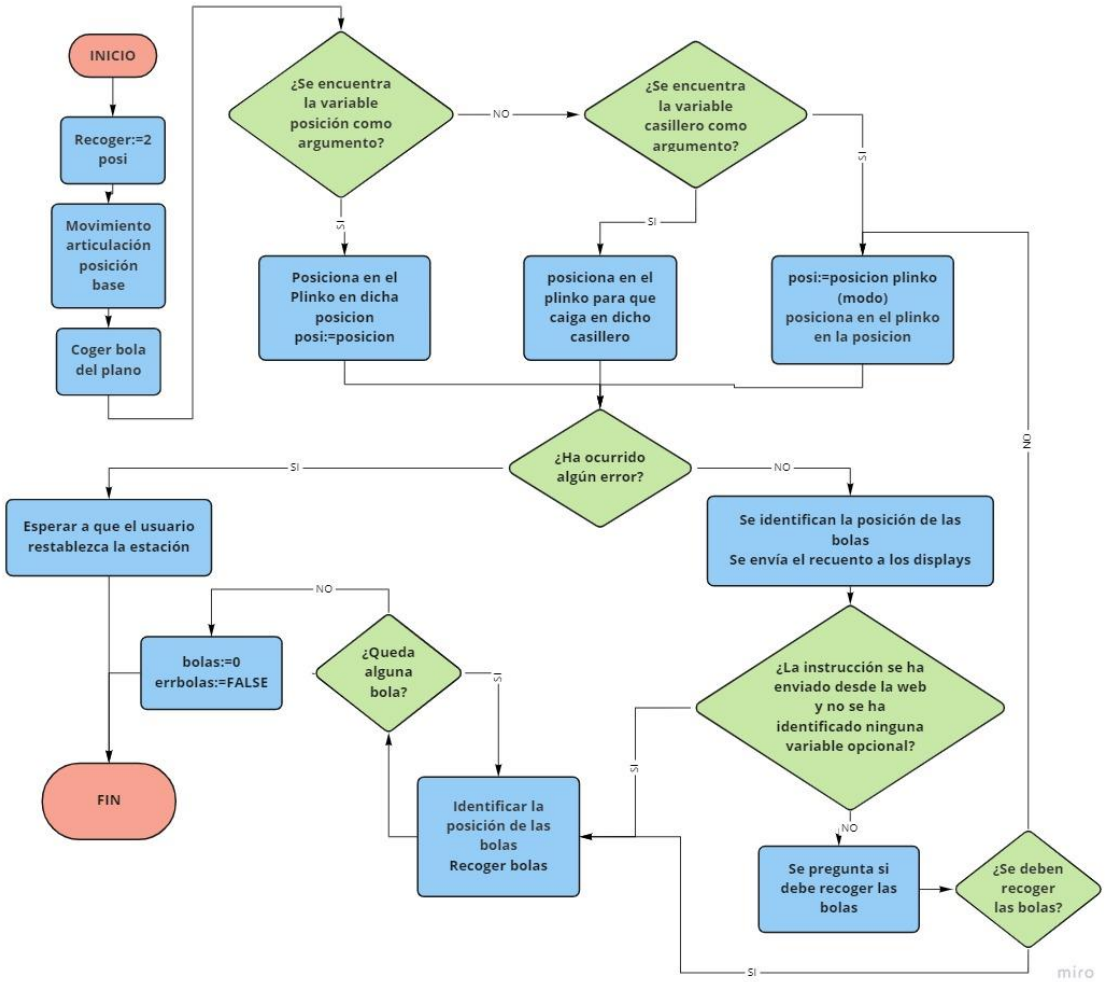


Ilustración 112: Diagrama de flujo principal.

En el inicio se declaran dos variables una es “recoger”, la cual esta inicializada a “2” y otra es “posi”, esta última se inicializa a “0” por defecto.

La variable “recoger” es una variable entera la cual se utiliza para evaluar si se deben recoger las bolas.

Tras la declaración de las variables el robot se dirige a una posición articular cómoda desde la cual se inician el resto de los movimientos.

Posteriormente, si el sensor del plano inclinado detecta que hay una bola para ser dispensada, activa la salida do1 excitando la bobina de la electroválvula y desplazando el vástago del pistón empuja la bola al lugar donde debe ser recogida por el ABB.

El robot dirige el TCP a la posición donde se encuentra la bola dispensada y la recoge. Se evalúa la presencia de distintos argumentos opcionales y en caso de haber sido recibidos se ejecutan diferentes variantes del proceso.

Si se ha recibido la posición desde la cual se debe dejar caer la bola traslada el TCP a dicha posición y abriendo la pinza deja caer la bola.

Si se ha recibido el casillero donde se desea que acabe la bola se deja caer la bola desde un tramo en el cual la probabilidad de que acabe en ese casillero sea la máxima posible.

Si no se reciben ninguno de los dos argumentos se posiciona la bola en un punto aleatorio de la parte superior del Plinko y desde donde seguidamente se deja caer.

Tras abrir la pinza el ABB activa una la interrupción de error bolas la cual se ejecuta si tras cinco segundos no se detecta su posición en el casillero del Plinko, esto es considerado un error y deberá ser tratado.

Para poder detectar los errores se hace uso de una variable “bolas” la cual incrementa su valor cada vez se suelta una bola y decrementa su valor cada vez que se recoge una.

Se activa la interrupción que gestiona el error de las bolas y se espera a que el numero de la variable bolas sea igual a la suma de todas las entradas procedentes del casillero, si dicha igualdad se cumple en menos de cinco segundos se desactiva dicha interrupción y si dicha igualdad no se cumple se entra en el tratamiento, el cual es gestionado por “Error_bolas”.

```
IWatch Errorbolas;
WaitUntil bolas = di1+di2+di3+di4+di5 OR errbolas=FALSE;
ISleep Errorbolas;
```

Ilustración 113: Activación y desactivación de la interrupción "Error_bolas".

Si se ha producido un error se restablecer la estación, dejar todas las bolas en el plano inclinado, y pulsar el botor di7 de la botonera.

Entre los errores que se pueden producir se encuentran:

- La bola se posiciona en un casillero, pero no activa un final de carrera.
- La bola se queda atascada en la cama de clavos del Plinko.
- La bola no se posiciona en ningún casillero, esto puede ser porque cae fuera de la estación o porque se queda en el tramo intermedio entre los casilleros y el tablero que contiene los clavos.

Este último posible es el menos probable de los tres y no se ha dado en ningún ensayo.

Si no se han producido errores se identifica la posición de las bolas y se envía el recuento de bolas a los displays.

El proceso presenta dos variantes para el modo automático dependiendo de si este es seleccionado desde la flexpendant o desde la página web.

En el caso de que las instrucciones se envíen desde la página web o desde un modo distinto al de automático se envía alguna de las dos variables opcionales, por lo que se recogen las bolas tras ser lanzadas, sin embargo, si se envían desde la flexpendant y en modo automático se pregunta al usuario si se quiere o no recoger las bolas de los casilleros antes de lanzar la siguiente.

Para recoger las bolas se identifican los casilleros ocupados y se recoge una de las bolas que están activando un final de carrera, tras esto se vuelve a ejecutar el bucle hasta que no se encuentra ningún final de carrera activo.

Tras recoger las bolas se establece la variable "bolas" a "0" y se finaliza el proceso.

No se puede salir del proceso si no se han recogido todas las bolas, esto permite volver a utilizar el proceso conociendo la situación de la estación sin la necesidad de realizar ningún registro.

3.1.4 Volúmenes de las maquetas

El modo 2 permite colocar al ABB irb120 en una posición articular determinada teniendo en cuenta la posición y el volumen que las maquetas ocupan en la estación para evitar colisionar con ellas. Existen dos protecciones para evitar una colisión, la primera es mediante una interrupción llamada

“colisión” y otra mediante unas funciones en las cuales se indica si se a vulnerado el volumen de alguna de las maquetas, en ambos casos se vuelve a la posición base.

```
LOCAL PROC Movimiento_angular(num articulaciones{*})

    VAR num tolerancia:=3;
    VAR num final:=0;
    VAR num incremento{6};
    VAR pos posicion;

    VAR jointtarget destino;

    VAR jointtarget posicion_actual;
    VAR num arti_actual{6};

    posicion_actual:=CJointT();

    MoveAbsJ Punto_Sing,V80,fine,PinzaInv;

    arti_actual:=[0,0,0,0,0,0];
    destino:=posicion_actual;
```

Ilustración 114: Movimiento angular.

En este proceso se declaran una serie de variables:

- “tolerancia” es una variable numérica que evita que los ejes se muevan de forma indefinida debido a pequeños desajustes. Esta variable es utilizada para que en caso de que la posición del eje en ese instante sea muy parecida a la posición final que se quiere alcanzar se deje mover.
- “final” es una variable numérica que indica el número de ejes que han alcanzado la posición final.
- “incremento” es un vector que almacena seis parámetros los cuales se establecen para intentar que los seis ejes alcancen la posición de tras un numero de repeticiones del bucle similar. “incremento” almacena el paso de cada eje en cada ejecución del bucle.
- “posición” es una variable que almacena la posición del TCP en el instante inicia.
- “destino” es la variable que almacena el destino final de los ejes en cada ciclo del bucle.
- El jointtarget “posición_actual” almacena la posición de los ejes en el instante inicial.
- “arti_actual” es el vector que almacena la posición de los ejes en el instante inicial, como en ese instante se encuentra en una posición conocida se sabe que el valor de todos los ejes es “0”.

```
FOR i FROM 1 TO 6 DO
    incremento{i}:=(abs(arti_actual{1}-articulaciones{1}))/60;
ENDFOR
```

Ilustración 115: Determinación del paso de movimiento

Primeramente, se determina el paso de movimiento de los seis ejes, para ello se resta la posición de los ejes actual de la posición final que se desea alcanzar y se divide entre el número de veces que se quiere repetir el bucle.

Cuantas más veces se repita el bucle menos riesgo existe de que se choque el Plinko con la maqueta, pero el movimiento será menos continuo y más forzado, en cambio cuantas menos veces se repita el bucle el control será peor y el movimiento más natural y menos forzado.

```
WHILE volumen_plinko(Posicion)=FALSE AND volumen_plano_inclinado(posicion)=FALSE AND final<6 AND colision=FALSE DO
    final:=0;
    IF abs(arti_actual{1}-articulaciones{1})>tolerancia THEN
        IF arti_actual{1}<articulaciones{1} THEN
            destino.robax.rax_1:=destino.robax.rax_1+incremento{1};
        ELSE
            destino.robax.rax_1:=destino.robax.rax_1-incremento{1};
        ENDIF
    ELSE
        final:=final+1;
    ENDIF
```

Ilustración 116: Bucle de posicionamiento en la coordenada articular establecida

En la anterior ilustración se puede observar cómo funciona uno de los bucles para un eje en concreto.

En un primer momento se establece la variable “final” a “1” y se resta la posición de la articulación actual y la posición final que se desea alcanzar, en caso de que dicha resta sea mayor a la tolerancia establecida al inicio del proceso se plantean dos escenarios:

Si la posición de la articulación en ese instante sea menor que la posición final, el incremento se debe sumar a la variable destino, en caso contrario se deberá restar.

Tras alcanzar la posición se incrementa en uno el valor de la variable final.

Este proceso se repite con todos los ejes, se ejecuta un MoveAbsJ hasta la posición de destino y se actualizan las variables.

```
MoveAbsJ destino,v80,fine,PinzaInv;
!TPWrite "final: "\Num:=final;

posicion:= CPos(\Tool:=PinzaNorm \WObj:=wobj0);
posicion_actual:=CJointT();
arti_actual:=[posicion_actual.robax.rax_1,posicion_actual.robax.rax_2,posicion_actual.robax.rax_3,
posicion_actual.robax.rax_4,posicion_actual.robax.rax_5,posicion_actual.robax.rax_6];
```

Ilustración 117: Movimiento hasta la posición de la variable destino

Este proceso se repite hasta que todos los ejes alcanzan la posición final o hasta que el TCP invade alguno de los volúmenes establecidos.

Estos volúmenes se establecen de manera matemática haciendo uso de unos Robtargects declarados en el espacio de trabajo wobj0.

Para la creación de estos volúmenes se hace uso de planos matemáticos conformados por tres puntos.

La ecuación de un plano genérico conformado por tres puntos se realiza de la siguiente manera.

$$\begin{vmatrix} x-a\{1\} & y-a\{2\} & z-a\{3\} \\ b\{1\}-a\{1\} & b\{2\}-a\{2\} & b\{3\}-a\{3\} \\ c\{1\}-a\{1\} & c\{2\}-a\{2\} & c\{3\}-a\{3\} \end{vmatrix} = 0 \rightarrow$$

$$\rightarrow (x-a\{1\})\cdot(b\{2\}-a\{2\})\cdot(c\{3\}-a\{3\}) + (y-a\{2\})\cdot(b\{3\}-a\{3\})\cdot(c\{1\}-a\{1\}) +$$
$$+(z-a\{3\})\cdot(b\{1\}-a\{1\})\cdot(c\{2\}-a\{2\}) - (z-a\{3\})\cdot(b\{2\}-a\{2\})\cdot(c\{1\}-a\{1\}) -$$
$$-(y-a\{2\})\cdot(b\{3\}-a\{3\})\cdot(c\{1\}-a\{1\}) - (x-a\{1\})\cdot(b\{3\}-a\{3\})\cdot(c\{2\}-a\{2\}) = 0$$
$$\rightarrow$$

$$\rightarrow (x-a\{1\})\cdot[(b\{2\}-a\{2\})\cdot(c\{3\}-a\{3\}) - (b\{3\}-a\{3\})\cdot(c\{2\}-a\{2\})] +$$
$$\rightarrow (y-a\{2\})\cdot[(b\{3\}-a\{3\})\cdot(c\{1\}-a\{1\}) - (b\{1\}-a\{1\})\cdot(c\{3\}-a\{3\})] +$$
$$\rightarrow (z-a\{3\})\cdot[(b\{1\}-a\{1\})\cdot(c\{2\}-a\{2\}) - (b\{2\}-a\{2\})\cdot(c\{1\}-a\{1\})] = 0$$

Se utiliza el cambio de variable para simplificar la ecuación final.

$$\alpha = [(b\{2\}-a\{2\})\cdot(c\{3\}-a\{3\}) - (b\{3\}-a\{3\})\cdot(c\{2\}-a\{2\})]$$
$$\beta = [(b\{3\}-a\{3\})\cdot(c\{1\}-a\{1\}) - (b\{1\}-a\{1\})\cdot(c\{3\}-a\{3\})]$$
$$\gamma = [(b\{1\}-a\{1\})\cdot(c\{2\}-a\{2\}) - (b\{2\}-a\{2\})\cdot(c\{1\}-a\{1\})]$$

$$\alpha\cdot(x-a\{1\}) + \beta\cdot(y-a\{2\}) + \gamma\cdot(z-a\{3\}) = 0 \rightarrow$$
$$\rightarrow -\alpha\cdot(x-a\{1\}) - \beta\cdot(y-a\{2\}) = \gamma\cdot(z-a\{3\}) \rightarrow$$

Llegando a la ecuación final del plano

$$\rightarrow \frac{-\boldsymbol{\alpha}\cdot(\boldsymbol{x}-\boldsymbol{a}\{\mathbf{1}\}) - \boldsymbol{\beta}\cdot(\boldsymbol{y}-\boldsymbol{a}\{\mathbf{2}\})}{\gamma} + \boldsymbol{a}\{\mathbf{3}\} = \boldsymbol{z}$$

Ecuación 2: Ecuación del plano genérico.

Los puntos que limitan los volúmenes han sido declarados como variables globales.

```
!_________Volumenes________
!volumen plinko
LOCAL CONST robtarget p1_plinko:=[[
LOCAL CONST robtarget p2_plinko:=[[
LOCAL CONST robtarget p3_plinko:=[[
LOCAL CONST robtarget p4_plinko:=[[
LOCAL CONST robtarget p5_plinko:=[[

!volumen plano inclinado
LOCAL CONST robtarget p1_plano_inc:
LOCAL CONST robtarget p2_plano_inc:
LOCAL CONST robtarget p3_plano_inc:
LOCAL CONST robtarget p4_plano_inc:
```

Ilustración 118: Declaración de los Robtargets que definen los volúmenes.

El volumen establecido para el plano inclinado es aquel que se encuentra debajo del plano definido por las ecuaciones antes descritas y los puntos que se muestran en el código.

```
VAR num a{3}:=[p1_plano_inc.trans.x,p1_plano_inc.trans.y,p1_plano_inc.trans.z];
VAR num b{3}:=[p2_plano_inc.trans.x,p2_plano_inc.trans.y,p2_plano_inc.trans.z];
VAR num c{3}:=[p3_plano_inc.trans.x,p3_plano_inc.trans.y,p3_plano_inc.trans.z];
VAR num d{3}:=[p4_plano_inc.trans.x,p4_plano_inc.trans.y,p4_plano_inc.trans.z];

VAR num alpha;
VAR num beta;
VAR num gamma;

VAR num Z;!El punto se debe encontrar por debajo del plano
VAR num Y1;!El punto se debe enctontrar detras de la recta
VAR num Y2;!El punto se debe enctontrar detras de la recta
VAR num X1;!El punto se debe enctontrar detras de la recta
VAR num X2;!El punto se debe enctontrar detras de la recta

alpha:=(b{2}-a{2})*(c{3}-a{3})-(b{3}-a{3})*(c{2}-a{2});
beta:=(b{3}-a{3})*(c{1}-a{1})-(b{1}-a{1})*(c{3}-a{3});
gamma:=(b{1}-a{1})*(c{2}-a{2})-(b{2}-a{2})*(c{1}-a{1});

Z:=((-alpha*(posicion.x-a{1})-beta*(posicion.y-a{2}))/gamma)+gamma*a{3};
```

Ilustración 119: Declaración de variables (volumen plano inclinado).

```
IF posicion.z>50 THEN
    IF posicion.z<c{3} AND posicion.z<Z AND posicion.y>Y1 AND posicion.y<Y2 AND posicion.x>X1 AND posicion.x<X2 THEN
    !TPWrite "y: "\Num:=posicion.y;
    !TPWrite "z: "\Num:=posicion.z;
    !TPWrite "Z: "\Num:=Z;
    !IF posicion.z<c{3} AND posicion.z<Z AND posicion.y>Y THEN
        TPWrite "colision con plano inclinado";
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    ENDIF
ELSE
    RETURN TRUE;
ENDIF
```

Ilustración 120: Definición de los límites del volumen del plano inclinado.

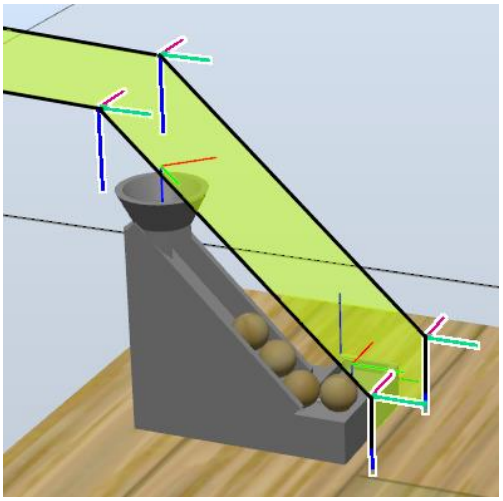


Ilustración 121: Plano que define la frontera del volumen del plano inclinado.

El volumen establecido para el Plinko, al igual que para el plano inclinado, es aquel que se encuentra debajo del plano definido por las ecuaciones antes descritas y los puntos que se muestran en el código.

```
VAR num a{3}:=[p1_plinko.trans.x,p1_plinko.trans.y,p1_plinko.trans.z];
VAR num b{3}:=[p2_plinko.trans.x,p2_plinko.trans.y,p2_plinko.trans.z];
VAR num c{3}:=[p3_plinko.trans.x,p3_plinko.trans.y,p3_plinko.trans.z];
VAR num d{3}:=[p4_plinko.trans.x,p4_plinko.trans.y,p4_plinko.trans.z];
VAR num e{3}:=[p5_plinko.trans.x,p5_plinko.trans.y,p5_plinko.trans.z];

VAR num alpha;
VAR num beta;
VAR num gamma;

VAR num Z;!El punto se debe encontrar por debajo del plano
VAR num Y;!El punto se debe enctontrar detras de la recta conformada por los puntos 4 y 5

alpha:=(b{2}-a{2})*(c{3}-a{3})-(b{3}-a{3})*(c{2}-a{2});
beta:=(b{3}-a{3})*(c{1}-a{1})-(b{1}-a{1})*(c{3}-a{3});
gamma:=(b{1}-a{1})*(c{2}-a{2})-(b{2}-a{2})*(c{1}-a{1});

Z:=((-alpha*(posicion.x-a{1})-beta*(posicion.y-a{2}))/gamma)+gamma*a{3};
Y:=((e{2}-d{2})/(e{1}-d{1}))*(posicion.x-e{1})+e{2};
```

Ilustración 122: Declaración de variables (volumen Plinko).

```
IF posicion.z>50 THEN
    IF posicion.z<e{3} AND posicion.y>e{2} AND Y<posicion.y THEN
        TPWrite "colision con plinko";
        !TPWrite "z:"\Num:=posicion.z;
        !TPWrite "e{3}:"\Num:=e{3};
        !TPWrite "y: "\Num:=posicion.y;
        !TPWrite "e{2}:"\Num:=e{2};
        !TPWrite "Y: "\Num:=Y;
        RETURN TRUE;
    ELSEIF posicion.z<a{3} AND posicion.z>e{3} AND posicion.z<Z THEN
        TPWrite "colision con plinko";
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    ENDIF
ELSE
    RETURN TRUE;
ENDIF
```

Ilustración 123: Definición de los límites del volumen del plano inclinado.

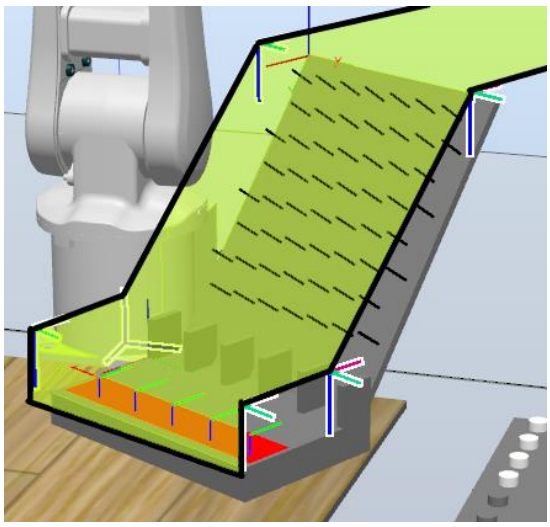


Ilustración 124: Plano que define la frontera del volumen del Plinko.

Estas dos funciones devuelven una variable booleana siendo esta "TRUE" en caso de atravesar el espacio con el TCP.

Este tipo de protección presenta vulnerabilidades ya que el robot puede golpear con cualquier eslabón y estos no serían identificados por las funciones, para evitar esto se ha decidido crear una interrupción en la cual al pulsar el botón di7 se finaliza el movimiento y se retrocede a la posición base.

3.1.5 Información recibida desde la web

La función información es la encargada de extraer los datos de las tramas enviadas desde la web, para ello es necesario conocer la posición del dato a extraer dentro de la trama y la trama recibida.

Sabiendo que cada dato en la trama finaliza con un “/” se recorre la trama y se contabiliza el número de veces que se encuentra dicho carácter en el momento en el que el numero de la cuenta es igual a la posición del dato que necesitas se devuelve el último dato leído.

Esto se logra haciendo uso de la función StrPart la cual devuelve una cadena de caracteres. Para ello debe recibir como parámetros el string principal en donde se encuentra la sección definida por la posición en la que se halla el primer carácter del string contenido en el string principal y la longitud de dicha cadena.

```
LOCAL FUNC string informacion(string str, num indice)

    VAR num i:=0;!Variable que inidica el orden del dato ejemplo m1/54/ m1 es 1 y 54 es 2
    VAR num aux:=0;!Variable que ayuda a conocer el numero de caracteres entre barras inclinadas
    VAR string inf:="";
    !IF StrLen(str)<15 and StrPart(str,StrLen(str),1)="/"THEN

    IF StrPart(str,StrLen(str),1)="/"THEN
        FOR j FROM 1 TO StrLen(str) DO
            IF StrPart(str,j,1)="/" OR j=StrLen(str) THEN
                i:=i+1;
                IF i=indice THEN
                    !TPWrite StrPart(str,aux+1,j-(aux+1));
                    RETURN StrPart(str,aux+1,j-(aux+1));
                ENDIF
                aux:=j;
            ENDIF
        ENDFOR
    ENDIF
ENDFUNC
```

Ilustración 125: Código de la función información

3.1.6 Función “Posición Plinko”

La función posición Plinko devuelve un valor numérico en el que se indica en que posición del Plinko se debe colocar la bola.

Esta posición es dependiente del modo de funcionamiento.

En el caso de que el modo de funcionamiento sea el modo 1 se debe enviar el parámetro opcional de posición y se estudia si el valor introducido se encuentra en los márgenes establecidos. Si la posición introducida es correcta se devuelve la posición recibida.

Si el modo de funcionamiento es el modo 3 se debe enviar el parámetro opcional de casillero y haciendo uso del registro estadístico se devuelve una posición en la parte superior del Plinko desde la cual la probabilidad de que caiga en dicho casillero es la mayor posible.

Si el modo de funcionamiento es el modo 4 la posición se obtiene haciendo uso de la función “rand” la cual devuelve un valor aleatorio entre “0” y “1”, dicho valor se multiplica por el margen superior y se obtiene un numero entre “0” y “240” el cual es enviado como la posición en la que se debe colocar la bola.

```
LOCAL FUNC num posicion_plinko(num modo \num posi \num casillero)

    var num posicion;

    TEST modo
        CASE 1:
            IF Present(posi) THEN
                posicion:=posi;
            ENDIF
        CASE 3:
            IF Present(casillero) THEN
                TEST casillero
                CASE 1:
                    posicion:=70;
                CASE 2:
                    posicion:=30;
                CASE 3:
                    posicion:=150;
                CASE 4:
                    posicion:=210;
                CASE 5:
                    posicion:=10;
                ENDTEST
            ELSE
                TPWrite "¿En que parte del plinko quieres colocar la bola?";
                TPWrite "Elige un numero entre 0 y 240 (se permiten decimales)";
                TPReadNum posicion,"posición";
                WHILE posicion>240 OR posicion<0 DO
                    TPWrite "Elige un numero entre 0 y 240 (se permiten decimales)";
                    TPReadNum posicion,"posición";
                ENDWHILE
            ENDIF
        CASE 4:
            posicion:=240*rand();
    ENDTEST
    RETURN posicion;
ENDFUNC
```

Ilustración 126: Código de la función posición plinko

3.1.7 Proceso que identifica la posición de las bolas

El proceso “identifica posición bolas” identifica en que casillero han caído las bolas y realiza dos funciones principales. La primera y que siempre realiza es actualizar el recuento de veces que las bolas han acabo en los distintos casilleros.

Las variables que almacenan dicha información son unas variables globales de tipo persistente, lo cual permite que el valor de dichas variables no cambie tras la finalización del programa, logrando así que el recuento de bolas sea el total de todas las veces que se ha ejecutado el proceso.

```
!Contador casilleros
LOCAL PERS num casillero1:=40;
LOCAL PERS num casillero2:=33;
LOCAL PERS num casillero3:=40;
LOCAL PERS num casillero4:=35;
LOCAL PERS num casillero5:=34;
LOCAL VAR num di_last{5}:=[0,0,0,0,0];
```

Ilustración 127: Variables del recuento de casilleros

Para lograr esto, se hace uso de un vector de cinco parámetros los cuales almacenan el valor del estado anterior de las entradas correspondientes a los finales de carrera situados en el casillero.

El recuento de casilleros se consigue evaluando el estado de los finales de carrera.

Si al entrar en la función el final de carrera esta presionado se compara con la posición anterior de la última vez que se ejecutó el proceso, para ello se utiliza el vector “di_last”.

En caso de que el parámetro “di_last” correspondiente a la entrada estudiada sea distinto de dicha entrada se incrementa el valor del recuento del casillero y se actualiza el valor del parámetro de “di_last” para evitar que la siguiente vez que se acceda a la función se vuelva a incrementar el recuento.

La principal intención del recuento de los casilleros es el envío de este recuento al controlador encargado de gestionar los displays, mostrando el número de veces que han caído las bolas en los distintos casilleros al usuario.

Como solo hay dos displays de siete segmentos por casillero el recuento de los casilleros debe ser menor de cien para evitar desbordamientos, para esto se ha establecido un reseteo del recuento en caso de que alguno de los casilleros alcance el valor de 100 veces.

La otra intención de la función, aunque esta es opcional, es la de recoger las bolas del casillero.

Si se recibe por parámetro la variable booleana “recoger” y esta es igual a “True” se ejecuta el proceso “Recoge del casillero” al cual se le indica el Robtarget que corresponde con el casillero donde se ha detectado la bola.

```
LOCAL PROC identifica_posicion_bolas(\num posicion, \bool recoger)
    WaitUntil di1=1 OR di2=1 OR di3=1 OR di4=1 OR di5=1;
    !Open "HOME:"\File:="reg_plinko.txt",fp\Append;
    IF di1=1 THEN
        IF di1<>di_last{1} THEN
            casillero1:=casillero1+1;
            IF Present(posicion) THEN
                !Write fp,"posicion: "+NumToSTR(posicion,2)+" casillero: "+"1";
                !Close fp;
            ENDIF
            di_last{1}:=di1;
        ENDIF
        IF Present(recoger) AND recoger=TRUE THEN
            Recoge_del_casillero(Casillero_1);
            di_last{1}:=0;
            Posiciona_en_recoge_bola;
        ENDIF
    ENDIF
    IF...ENDIF
    IF...ENDIF
    IF...ENDIF
    IF...ENDIF

    IF casillero1>99 OR casillero2>99 OR casillero3>99 OR casillero4>99 OR casillero5>99 THEN
        casillero1:=0;
        casillero2:=0;
        casillero3:=0;
        casillero4:=0;
        casillero5:=0;
    ENDIF

ENDPROC
```

Ilustración 128: Función “identifica posición bolas”.

3.2 Código del esp32 que gestiona los displays

En el proyecto se hace uso de dos esp32, uno gestiona los displays y el otro ofrece una página web con la que comunicarse con el ABB irb120.

Este primer microcontrolador presenta dos cometidos:

1. Comunicarse con el ABB por protocolo TCP/IP haciendo uso de sockets y con el otro esp32 por puerto serial.
2. Mostrar en los displays la información recibida por el ABB irb120.

La programación de este esp32 se resume en el siguiente diagrama de flujo.

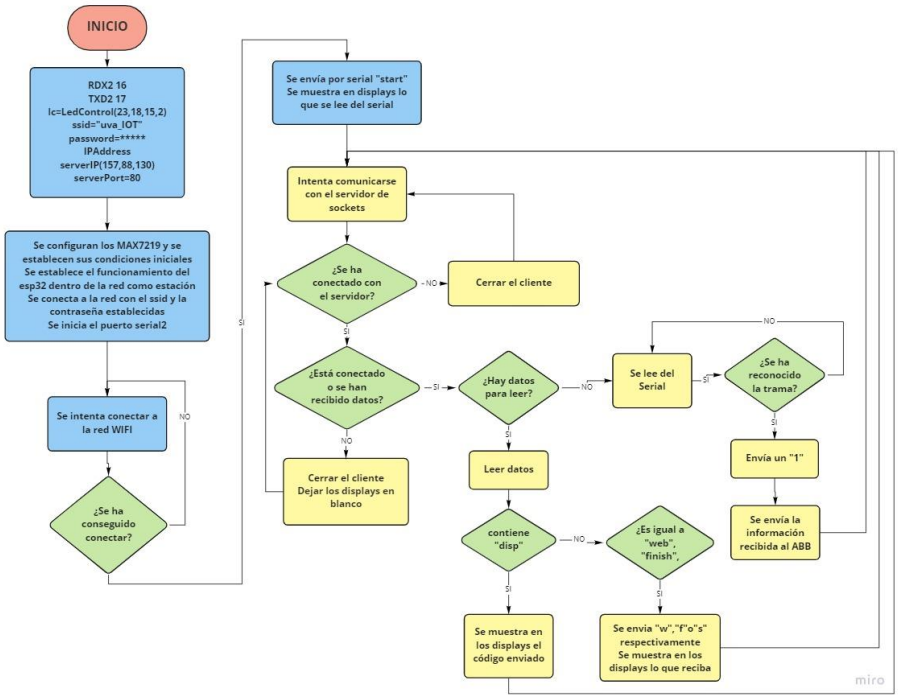


Ilustración 129: Diagrama de flujo del esp32 que gestiona los displays.

En un inicio se declaran los pines desde los que se va a realizar la comunicación I2C con los MAX7219, se declara la variable “lc” y se indica el ssid, la contraseña de la red a la que se debe conectar el microcontrolador, la ip del servidor de sockets y el puerto por el que se va a comunicar.

Tras estas declaraciones se configuran los circuitos integrados estableciendo sus condiciones iniciales, se establece el funcionamiento del esp32 dentro de la red como estación, se conecta dicha red e inicia el puerto serial.

Se evalúa si se ha conseguido conectar a la red, en caso de que no lo haya conseguido se vuelve a intentar, sin embargo, si se consigue conectar envía por el puerto serial la palabra “start”, esto se realiza para iniciar la comunicación por el puerto serial ya que ante la primera trama la comunicación no responde bien.

Las siguientes instrucciones se realizan dentro de la función void loop por lo que para diferenciarlos se ha cambiado el color de los procesos a amarillo.

Se intenta comunicar por sockets, en caso de no conseguirlo se cierra el cliente y se vuelve a intentar, si se conecta se evalúa si está conectado o se han recibido datos, en caso de que no se cumpla se cierra el cliente de sockets y se dejan los displays en blanco volviendo a intentar la conexión. Si está conectado o se han recibido datos se evalúa si hay datos para leer en caso de que les haya se leen y se estudia la trama.

Si la trama contiene la cadena de caracteres “disp” se muestra en los displays la información que se encuentre justo detrás de dicha cabecera. Si no contiene “disp” se evalúan “web”, “finish” o “stop”, en caso de que ocurra alguna de estas se actualizan los displays si es necesario, y se envía por serial al otro esp32 “w”, “f” o “s” respectivamente.

En el caso de que no haya datos para leer se lee del serial la información enviada desde el esp32 que gestiona la página web, si se ha reconocido la trama se envía un “1” y se envía la información recibida por sockets al ABB en caso contrario no se envía nada y se vuelve a leer el serial, de esta forma se logra reducir el número de errores de comunicación.

Tras la ejecución de todo esto se vuelve a ejecutar el primer bloque amarillo.

Para realizar la comunicación con el ABB se necesita utilizar la librería <WiFi.h> la cual gestiona toda la comunicación del microcontrolador con la red.

La librería <WiFi.h> se utiliza para establecer el modo en el que se va a conectar a la red, en este caso como estación (STA).

Gestiona la suspensión del Wifi logrando un más bajo consumo, en este proyecto se prima la velocidad de transmisión por lo que esta suspensión queda desactivada.

```
WiFi.mode(WIFI_STA);
WiFi.setSleep(false); // Desactiva la suspensión de wifi en modo STA para mejorar la velocidad de respuesta
```

Ilustración 130: Funciones de la librería <WiFi.h> WiFi.mode y WiFi.setSleep.

Se conecta a la red enviando al router indicado por la SSID la intención de formar parte de esa red y la contraseña para acceder.

```
WiFi.begin(ssid, password);
```

Ilustración 131: Función WiFi.begin encargada de conectarse a la red.

Gracias a esta librería se puede conocer tanto la IP asignada como la Mac del módulo Wifi integrado.

```
Serial.println(WiFi.localIP());
Serial.println(WiFi.macAddress());
```

Ilustración 132: Dirección IP y dirección Mac.

Establece la comunicación por sockets en la red en la que se encuentra logrando enviar tramas al ABB y recibir las instrucciones del robot.

```
if (client.connect(serverIP, serverPort)) // Intenta acceder a la dirección de destino
```

Ilustración 133: Intento de conexión por sockets con el servidor

```
if (client.available())
```

Ilustración 134: Evaluación de si hay archivos para leer por sockets.

```
String line = client.readStringUntil('\n'); // Leer datos a nueva línea
```

Ilustración 135: Lectura de los datos por sockets.

```
client.print(inf);
```

Ilustración 136: Envío de los datos por sockets.

Para la comunicación con el otro esp32 se utiliza el tercer puerto serial para ello se utiliza la función “Serial2.println” la cual no es necesario importarla de ninguna librería.

El uso de dicho puerto debe inicializarse indicando la velocidad de trasferencia de archivos en baudios y los pines habilitados para ellos.

```
#define RXD2 16
#define TXD2 17
```

Ilustración 137: Declaración de los pines RX y TX.

```
Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2);// serial1 pines 16 y 17
```

Ilustración 138: Inicialización del puerto serie 2.

Con el puerto serial se puede enviar y recibir información.

```
Serial2.println("start");
```

Ilustración 139: Envío de información por puerto serial2.

Para saber si hay información disponible para ser leída se usa la función "Serial2.available".

```
if (Serial2.available()>0)
```

Ilustración 140: Evaluación de si hay archivos para leer por serial2.

La lectura del serial se realiza de carácter en carácter por lo que para leer una trama utilizo el siguiente bucle.

```
if (Serial2.available()>0)
{
  char mens=Serial2.read();
  while(mens!='\n')
  {
    mensaje=mens+mensaje;
    mens=Serial2.read();
  }
}
```

Ilustración 141: Lectura de datos por serial2.

Para controlar los displays el esp32 hace uso de un protocolo de comunicación llamado I2C el cual permite el envío y la recepción de tramas de datos mediante un dispositivo digital como en este caso el esp32.

La comunicación I2C se basa en dos cables, uno SCL controlado por el maestro por donde se envía la señal de reloj y otro SDA que es el encargado de la transmisión de los datos.

Este tipo de comunicación presenta una ventaja con respecto a la comunicación serial y es que permite tener una confirmación de la recepción de los datos.

El maestro I2C, en este caso el esp32, inicia y finaliza la comunicación con los circuitos integrados.

El esclavo I2C, en este caso los circuitos integrados MAX7219, recibe la información y responde con un bit de ACK.

La trama que se utiliza para el envío de información es de la siguiente forma.

Ilustración 142: Trama I2C. [9]

Para mostrar en los displays la información recibida se ha importado la librería LedControl.h la cual está diseñada para funcionar con los circuitos integrados esp32.

En la declaración de un objeto LedControl se debe especificar el pin DIN, SCL, CS y el número de MAX7219 que están conectados al esp32.

```
LedControl lc= LedControl(23,18,15,2);
```

Ilustración 143: Declaración del objeto LedControl lc.

Al inicio, los displays deben establecerse encendidos o apagados para ello se hace uso de la función "lc.shutdown" a la cual se le indica el dispositivo al que se refiere considerando que se inicia desde "0" y se le indica el estado inicial, false (estado inicial del MAX7219 apagado) o true (estado inicial del MAX7219 encendido).

Tras esto se utiliza la función "setIntensity" para establecer por software el nivel de intensidad lumínica que deben tener los displays y se utiliza "cearDisplay" para borrar los dígitos.

```
lc.shutdown(0,false);
lc.shutdown(1,false);
lc.setIntensity(0,8);
lc.setIntensity(1,8);
lc.clearDisplay(0);
lc.clearDisplay(1);
```

Ilustración 144: Control de los MAX7219.

Para mostrar una serie de números o de caracteres en los displays se debe hacer uso de la función “lc.setChar” la cual envía un carácter a un MAX7219 indicándole el display que debe activar para que dicho carácter se muestre, además se le envía un parámetro independiente para indicar si se quiere iluminar el punto del display o no.

```
lc.setChar(0,disp,informacion[i],punto);
```

Ilustración 145: Envío de caracteres a los MAX7219.

3.3 Código del esp32 que gestiona la página web.

El esp32 encargado de brindar la página web crea una red propia a la cual se puede conectar cualquier dispositivo e introduciendo en el navegador la dirección ip de este microcontrolador se puede visualizar una página web dinámica con una mejor interfaz que desde la flexpendant.

La programación de este esp32 se resume en el siguiente diagrama de flujo.

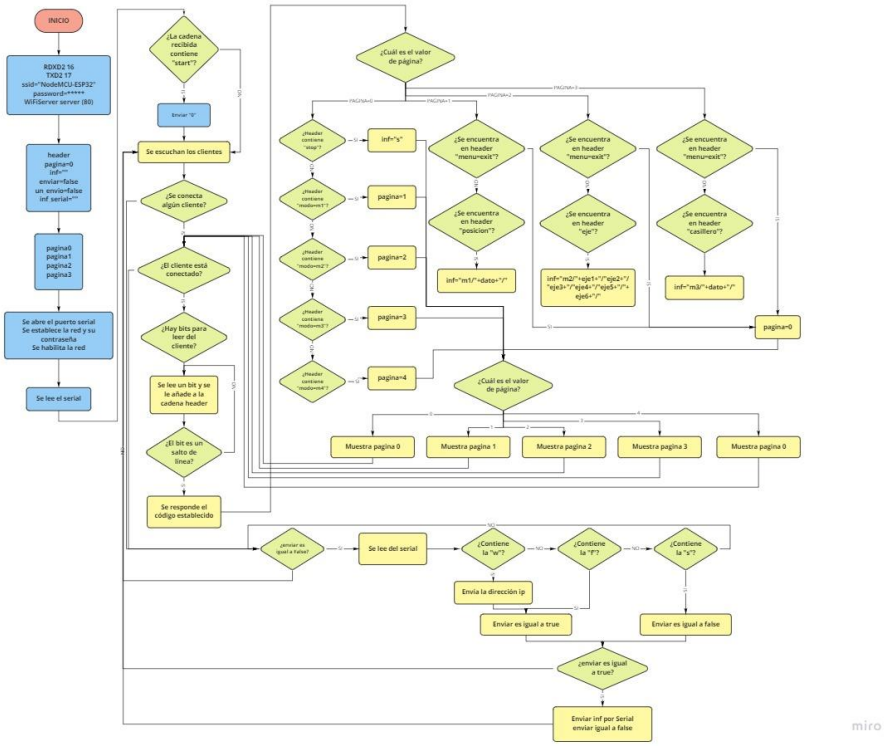


Ilustración 146: Diagrama de flujo del segundo esp32.

La explicación se divide en partes.

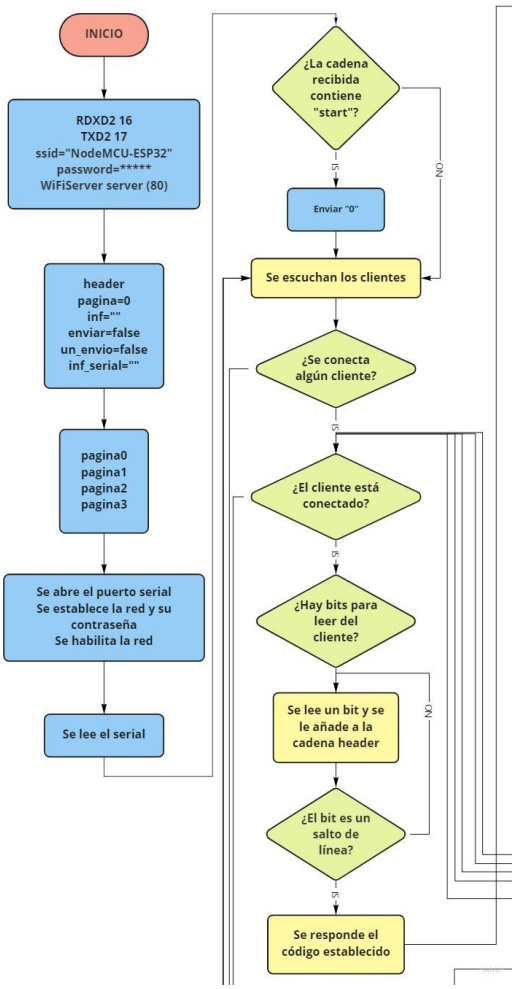


Ilustración 147: Primera parte del diagrama de flujo del segundo esp32.

El segundo esp32 se comunica con el primero mediante comunicación serial por lo que al igual que en el anterior caso se necesitó declarar los pines de RDX y TDX, se definió la ssi, contraseña y puerto de comunicación del punto de acceso que debe crear.

Se declaran unas variables necesarias para el funcionamiento del programa.

La variable “header” es la información que recoge de la comunicación con el cliente, es esencial para saber cuál es la información recogida del formulario debido a que el método GET es por el cual obtiene la información del formulario.

“pagina” sirve para conocer la página web que debe mostrar al usuario, de esta forma se consigue una navegación entre las páginas web.

Las variables pagina0, pagina1, pagina2, pagina3 son aquellas que almacenan el código de las páginas web. El código de dichas paginas es muy

simple y debido a que este es un proyecto enfocado al control de un ABB irb120 no se va a comentar.

Antes de pasar al bucle “loop” se abre el puerto serial, se habilita el punto de acceso, se inicia el servidor web y se lee el serial.

Tras esto se evalua si la cadena recibida contiene “start”, en caso de tenerlo se envia “0”. Esta comunicación no suele ser satisfactoria, sin embargo, se establece para inicializar la comunicación serial y lograr que durante la ejecución del programa la comunicación sea mucho más estable.

Lo siguiente que se realiza es una escucha de los clientes, si se conecta algún cliente se evalúa si se ha recibido algo, en caso de haber recibido algo se leen los caracteres hasta llegar a un salto de línea.

Después de leer todo lo proveniente del cliente web se responde con un código preestablecido que indica que la conexión ha sido correcta.

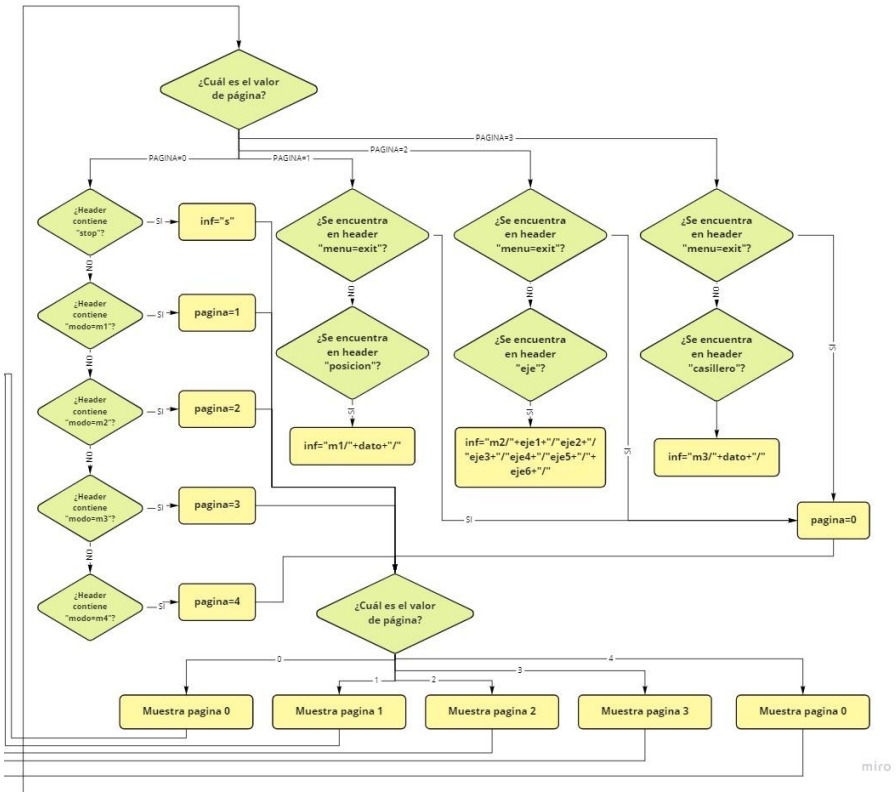


Ilustración 148: Segunda parte del diagrama de flujo del segundo esp32.

La información recibida será tratada de forma independiente de la página de la que proceda.

1. En caso de proceder de la pagina0 la información podrá ser “s”, de salir del modo web, “1”, de pagina 1,”2”, de pagina 2, “3” de pagina 3 y “4” de pagina 4.
   Esta información decide los modos de funcionamiento y la pagina visitada para poder crear la trama correctamente.

Ilustración 149: Selección de modo de funcionamiento desde la pagina 0.

2. Si procede de la pagina1 solo se van a evaluar dos posibles, el primero es que se quiera regresar al menú principal y otro la captación de la zona donde se quiere posicionar la bola.

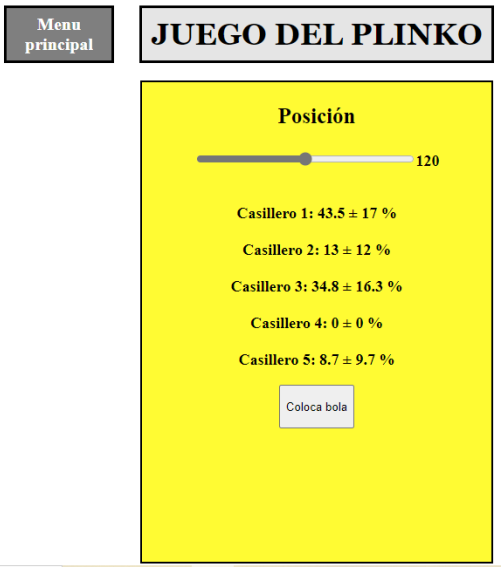


Ilustración 150: posicionamiento en plinko desde la pagina1.

3. Si procede de la pagina2 se estudian las posibilidades igual que en la página anterior, puede querer regresar al menú principal o puede enviar la posición de los ejes.

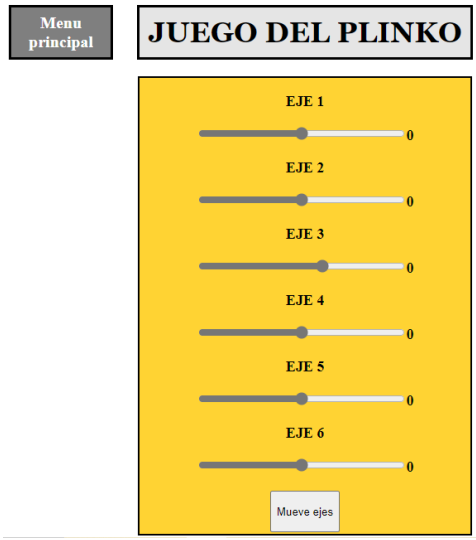


Ilustración 151: Movimiento de los ejes desde la pagina2.

4. El caso de la pagina3 es igual que en los dos casos anteriores, se puede salir al menú principal o se puede enviar la posición del casillero donde quieres que caiga.

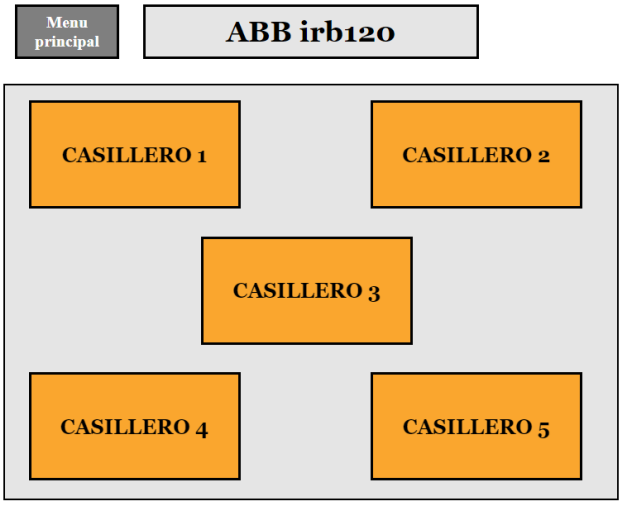


Ilustración 152: Selección del casillero donde quieres que caiga.

5. El modo de funcionamiento automático no requiere de página web por lo que se regresa a la página para poder seleccionar otro modo o el mismo.

En un inicio la variable “pagina” es igual a “0” por lo que se debe seleccionar uno de los modos, tras esto la variable página se actualiza y se envía la página correspondiente al cliente.

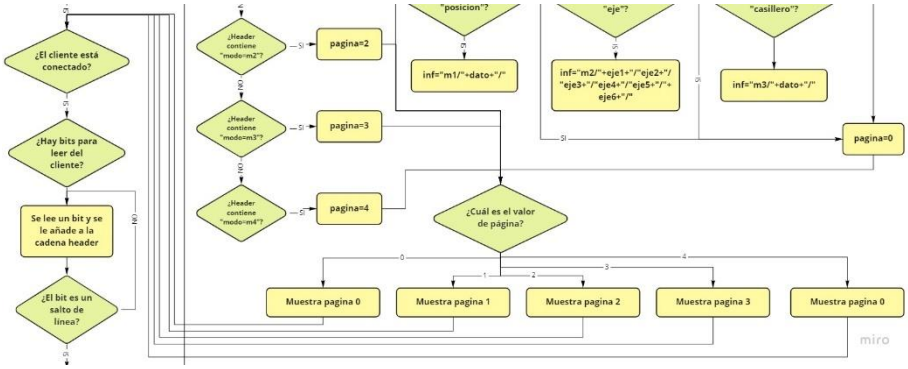


Ilustración 153: tercera parte del diagrama de flujo del segundo esp32.

Tras mostrar la página web se cierra la comunicación con el cliente y se vuelve a verificar que el cliente esté conectado.

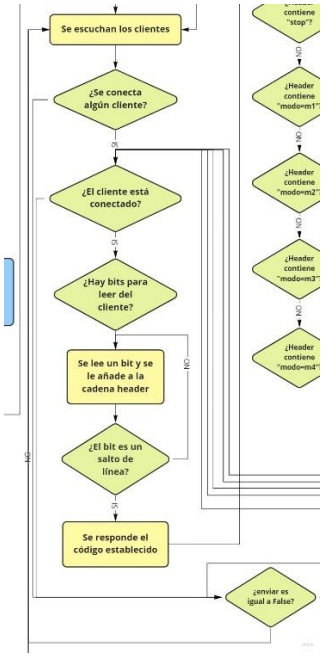


Ilustración 154: Cuarta parte del diagrama de flujo del segundo esp32.

Si no está conectado ningún cliente se evalúa la variable “enviar”.

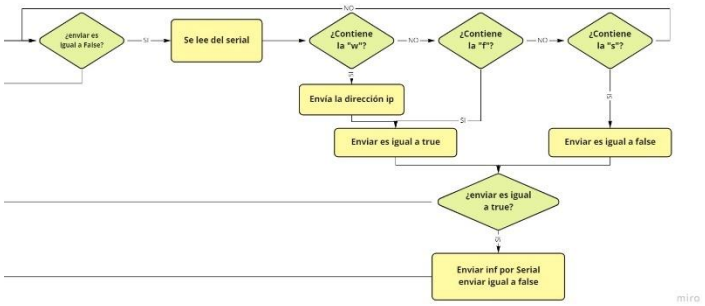


Ilustración 155: Quinta parte del diagrama de flujo del segundo esp32.

Si la variable enviar es igual a “false” se lee el serial y se estudia si contiene la “w”, la “f” o la “s”. En caso de recibir una “w” envía la dirección ip por Serial y establece la variable igual a true, en caso de recibir una “f” establece la variable “enviar” igual a true, sin embargo, en caso de recibir una “s” la variable “enviar” se establece en “false”.

Si enviar es igual a true se envía la información recibida desde la página web por el cliente.

En este microcontrolador, al igual que en el anterior, debe utilizar la librería <WiFi.h> además de la librería <WebServer.h>, estas librerías son necesarias para gestionar el módulo Wifi y para poder ser el servidor de una página web.

Se declara el servidor web y el puerto que va a utilizar haciendo uso del tipo de dato WiFiServer.

```
WiFiServer server(80);
```

Ilustración 156: Declaración del servidor y el puerto.

Se establece el ssid de la red Wifi que va a crear y la contraseña y se establecen en la función “WiFi.softAP”, la cual crea un punto de acceso.

```
const char* ssid = "NodeMCU-ESP32";
const char* password = [redacted]
```

Ilustración 157: Establecer ssid y contraseña de la red wifi.

```
WiFi.softAP(ssid, password);
```

Ilustración 158: Indicar la ssid y la contraseña del punto de acceso.

Tras esto, el servidor escucha a los clientes entrantes con la función “WiFiClient”.

```
WiFiClient client = server.available();
```

Ilustración 159: Escucha de clientes desde la página web.

Se inicia el servidor con la función “server.begin” para que pueda ser visitado.

```
server.begin();
```

Ilustración 160: Inicia el servidor web.

Para poder visualizar la página web hay que entrar en el punto de acceso creado por el esp32 e introducir en el navegador la dirección ip del

microcontrolador. Para conocer la dirección ip del esp32 se utiliza la función “WiFi.softAPIP” y se envía al ordenador por el puerto serial 0.

```
IPAddress IP = WiFi.softAPIP();
Serial.print("AP IP address: ");

Serial.println(IP);
```

Ilustración 161: Obtención de la dirección ip.

Para identificar si se conecta un nuevo cliente se evalúa el dato cliente declarado en la **Ilustración 159**.

```
if (client) {                             // Si se conecta un nuevo cliente
```

Ilustración 162: Identificación de la conexión de un nuevo cliente.

Tras el envío de información por parte del cliente se le responde para establecer la comunicación.

```
client.println("HTTP/1.1 200 OK");
client.println("Content-type:text/html");
client.println("Connection: close");
client.println();
```

Ilustración 163: Respuesta del servidor.

Para enviar información al cliente se utiliza la función “client.println” a la cual se le pasa por argumento la cadena de caracteres que debe enviar para que interprete el navegador del cliente.

```
switch(pagina)
{
  case 0:
    client.println(pagina0);
    break;
  case 1:
    client.println(pagina1);
    break;
  case 2:
    client.println(pagina2);
    break;
  case 3:
    client.println(pagina3);
    break;
  case 4:
    client.println(pagina0);
    break;
}
```

Ilustración 164: Envío de información por parte del servidor.

Las variables pagina0, pagina1, pagina2 y pagina3 son strings guardados en la memoria rom del microcontrolador.

Se guardan en la memoria ROM debido a que su declaración es más simple y a una mejor interpretación de los scripts, programados en javascript, por parte del cliente.

```
String pagina0 PROGMEM = R"=====(
<!doctype html>
<html>
<head>
<meta charset='utf-8'>
<meta name='viewport' content='width=device-width, initial-scale=1'>
<title>TFG</title>


<script>

    function modo()
```

Ilustración 165: Declaración de la página web “pagina0”.

Las funciones de la comunicación serial son las mismas que en el anterior esp32 por lo que, aunque este microcontrolador también las utiliza, no van a ser explicadas.

3.2.2 Comunicación entre los microcontroladores y el ABB

La comunicación entre los microcontroladores y el ABB es imprescindible para poder integrar dentro del proyecto el control del ABB desde una página web y el uso de unos displays para mostrar el recuento de veces que ha caído la bola en los distintos casilleros.

Estas comunicaciones se han realizado mediante una comunicación serial entre los esp32 y una comunicación TCP/IP mediante el uso de sockets entre un esp32 y el ABB.

3.2.3 Comunicación Serial.

La comunicación serial se basa en un puerto capaz de enviar y recibir bytes de información bit a bit de forma asíncrona pudiendo enviar y recibir información de forma simultánea.

Este método es utilizado para la trasmisión de caracteres en formato ASCII y para que esta sea posible se requieren de tres líneas. Una línea para trasmitir información (TX), otra línea para recibir los datos (RX), además se deben conectar las tierras para referenciar el neutro.

La comunicación serial se caracteriza por:

- La velocidad de transmisión medida en baudios (bits por segundo que se trasfieren), en este proyecto es de 115200.

- La cantidad de bits de datos que se envían en cada paquete.
- El bit de paridad el cual es utilizado para verificar la existencia de errores en la transmisión serial. Existen cuatro tipos de paridad, aunque también se puede prescindir del uso de paridad.
    - Par
    - Impar
    - Marcada
    - Espaciada
- El bit de parada es usado para indicar el fin de la comunicación de un solo paquete. Como el tipo de comunicación es asíncrona y cada dispositivo tiene un reloj propio el bit de parada tiene un margen de tolerancia para evitar errores derivados de la diferencia de los relojes.

Las características de la comunicación entre los esp32 en este proyecto es la siguiente.

- Velocidad de trasmisión: 115200 baudios.
- La cantidad de bits de datos que se envían son 8.
- En este caso no se utiliza un bit de paridad.
- Hay solo un bit de parada.
- Los pines de comunicación Serial TX y RX quedan definidas en el inicio del programa y son 17 y 16 respectivamente.

Esta configuración queda declarada en esta línea de código donde se indica la velocidad de transmisión, los bits de datos paridad, parada y los pines RX y TX. Los bits de datos, la paridad y un bit de parada quedan establecidos en “SERIAL_8N1”.

```
Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2);
```

Ilustración 166: Declaración de las características de la comunicación Serial.

3.2.4 Comunicación TCP/IP y Sockets.

La comunicación TCP/IP lo conforman un conjunto de protocolos por los cuales se pueden comunicar distintos dispositivos conectados a una misma red.

Este tipo de comunicación se basa en la dirección ip, es decir, cada equipo de la red presenta una dirección con una primera parte común y otra distinta. La parte común se establece mediante el uso de una máscara y en función de esta se clasifican los distintos tipos de redes.

Máscara	Clase

255.255.255.255	D
255.255.255.0	C
255.255.0.0	B
255.0.0.0	A

Tabla 16: Clase de subred en función de la máscara

La principal función de este tipo de comunicación es garantizar una correcta transmisión entre dispositivos para lograrlo divide los datos a enviar en varios paquetes, utiliza un sistema de direcciones, antes de enviar el paquete se define el camino por el cual se va a hacer llegar los datos del origen al destino y presenta distintos protocolos para identificar los errores durante la trasmisión de los datos.

TCP/IP es universal ya que es independiente del sistema operativo, el hardware o el software. Esto se logra debido a que este protocolo TCP establece la manera en la que se fragmentan los datos y conformando varios paquetes que son enviados individualmente y el protocolo IP enruta los paquetes desde el origen hasta su destinatario.

La comunicación en este proyecto es TCP/IP haciendo uso de sockets los cuales permiten que la comunicación sea bidireccional entre las máquinas.

La comunicación por sockets se basa en la filosofía cliente-servidor donde el servidor se encarga de crear un socket con un nombre conocido por el cliente y el cliente crea otro socket. Los dos sockets permiten una comunicación bidireccional.

Los sockets presentan dos características principales:

1. El tipo de socket, indica el tipo de comunicación que puede establecerse entre los sockets y existen estos cuatro tipos:
    - **SOCK_DGRAM:** Basado en UDP y por el cual se envían datagramas de tamaño limitado.
    - **SOCK_STREAM:** Basado en TCP y por el cual el tamaño de los datos capaces enviar es variable.
    - **SOCK_RAW:** Permite el acceso a protocolo de más bajo nivel como el IP.
    - **SOCK_SEQPACKET:** Mismas características que el SOCK_STREAM pero con un tamaño de mensajes fijo.
2. El dominio del socket especifica las direcciones de los sockets y los protocolos por los que se podrán comunicar entre sí.
    - **AF_UNIX**: El cliente y el servidor deben estar en la misma máquina.
    - **AF_INET**: El cliente y el servidor pueden estar en cualquier máquina de la misma red.

Para establecer una comunicación por sockets el proceso servidor crea un socket con un nombre establecido y espera la conexión de un cliente, al mismo tiempo el cliente crea un socket sin nombre y pide una con el socket servidor tras esto el cliente realiza la conexión a través de su socket con el socket del servidor.

3.2.5 Diagrama de secuencia de la comunicación en este proyecto.

Las tramas que se envían entre los dispositivos quedan detalladas en el siguiente diagrama de secuencia.

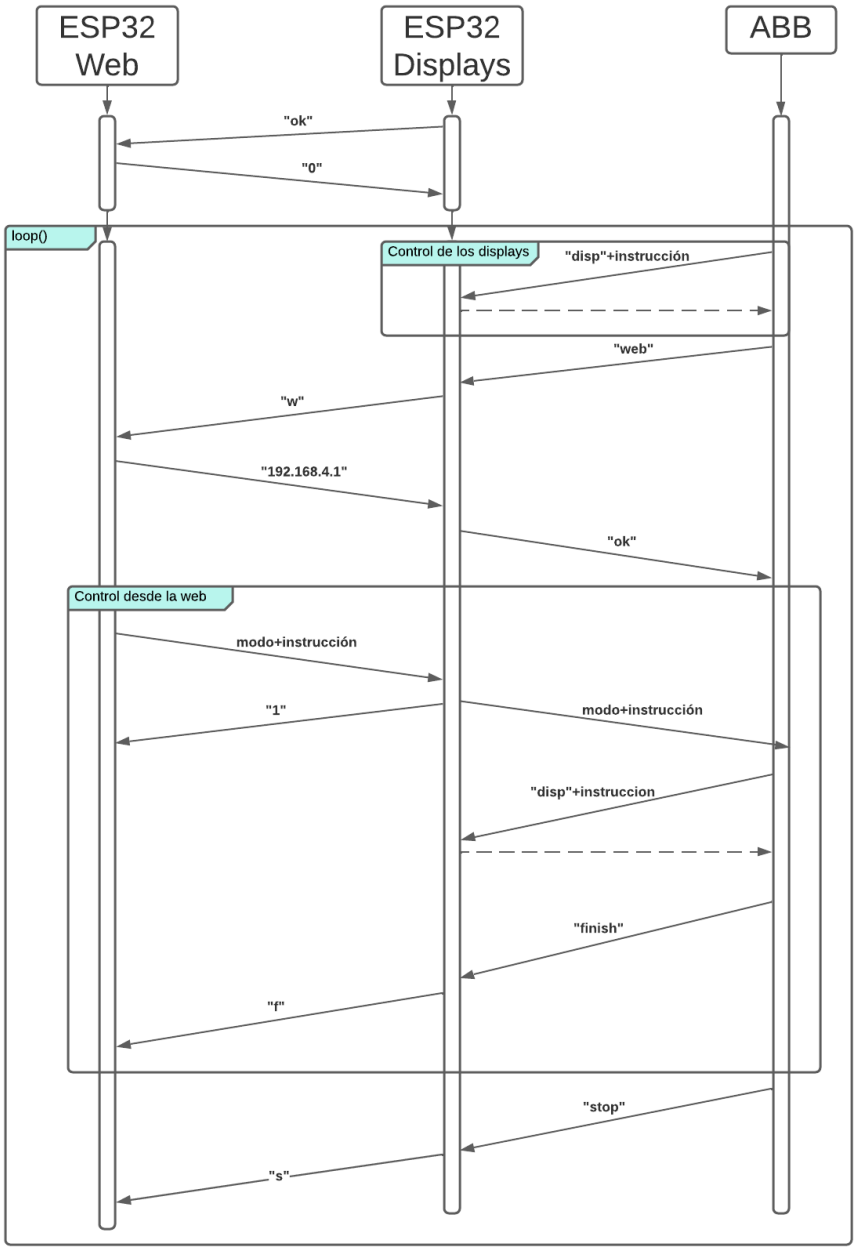


Ilustración 167: Diagrama de secuencia que explica la comunicación.

Los microcontroladore en un inicio ejecutan sus funciones de setup en las cuales hay una comunicación serial entre ellos, en ella un esp32 le envía una trama al otro y este le responde. Esta comunicación no se completa correctamente, sin embargo, mejora la comunicación de las siguientes tramas que se envían y reciben.

El ABB en el inicio de su programa decide si crear y abrir el socket servidor para que se conecte el esp32 cliente, si se conecta, durante la ejecución de distintos procesos le va enviando distinta información para que sea mostrada en los displays. Las tramas que envía para gobernar los displays son de la siguiente forma.

"disp"+instrucción

Instrucción debe presentar diez caracteres, uno por display.

Si el usuario le comunica a través de la fexpendant su intención de controlar el ABB desde la página web este envía "web" por sockets al esp32 intermedio y este le envía "w" al segundo.

En el esp32 intermedio se acorta la trama a enviar al segundo para reducir las probabilidades de errores de comunicación.

Seguidamente el esp32 que crea un punto de acceso le envía su dirección ip a los displays para indicar al usuario cual es la dirección donde se encuentra la página web. Tras esto el microcontrolador intermedio le envía "ok" al ABB, que se encuentra en espera, para indicarle que la comunicación con la página web ha sido satisfactoria.

Como el control se realiza desde la página web, las tramas son enviadas del segundo microcontrolador al primero. En dichas tramas se indica el modo y la instrucción. El esp32 que controla los displays le envía la trama recibida al ABB y este ejecuta el proceso seleccionado.

Para evitar evaluar una trama errónea, el microcontrolador intermedio contesta con un "1" si la trama recibida es correcta, en caso contrario se espera a que reenvíe la trama, la cual se enviará de forma periódica hasta que reciba un "1".

Durante el proceso el ABB envía información a los displays, y cuando este acaba le envía un "finish" que, tras el procesamiento de esta trama en el primer microcontrolador, se recibe "f" en el segundo indicando que puede enviar la siguiente instrucción.

Este proceso se repite de forma cíclica hasta que el usuario desde el botón di8 de la botonera o desde la página web decide salir de este modo volviendo al menú inicial de la flexpendant.

El mensaje enviado o recibido desde el segundo microcontrolador para finalizar el modo web es “s”, mientras que el caso del ABB es “stop”.

4. Diseño electrónico para la gestión de la práctica docente.

La electrónica en este proyecto la conforman dos circuitos, el primero y más complejo es el conformado por dos PCBS una contiene dos microcontroladores y los circuitos integrados MAX7219 la otra PCB donde se interconectan todos los displays. El segundo, un sensor que identifica si hay alguna bola disponible para ser dispensada.

En este apartado se explican los esquemas de los circuitos electrónicos y se muestran sus diseños para la posterior fabricación en la PCB.

4.1 Microcontroladores y circuitos integrados.

Este circuito está conformado por dos microcontroladores esp32 comunicados entre si mediante comunicación serial.

Uno de los dos microcontroladores es el encargado de gestionar dos circuitos integrados MAX7219 los cuales gobiernan los displays.

El otro microcontrolador es el encargado de gestionar la página web y la única comunicación que presenta es con el otro ESP32 para el envío de información mediante dos pistas TX y RX.

Este circuito se alimenta haciendo uso de un Buck de potencia ya que el ABB presenta una salida de alimentación a 24V, mientras que los microcontroladores deben ser alimentados a 5V.

El Buck de potencia es necesario para reducir la tensión de 24V a 5V sin grandes pérdidas energéticas en forma de calor.

4.1.1 ESP32

ESP32 es un microcontrolador diseñado por Espressif Systems.

Este microcontrolador presenta una serie de características de las cuales las que considero más relevantes se resumen en la siguiente tabla.

CARACTERÍSTICAS	ESP32
Microprocesador	Xtensa Dual-Core 32-bit LX6 con 600DMIPS
Wi-Fi	HT40
Bluetooth	Bluetooth 4.2 y BLE
Frecuencia de operción	160Mhz
SRAM	448 KB
Flash	520 KB
Pines de entradas y salidas	34
PWM (software)	16 canales
SPI	4

I2C	2
I2S	2
Comunicación Serial	2
Convertidor A/D	12-bits de resolución
Interfaz MAC Ethernet	Sí
Temperatura de trabajo	-40ºC a 125ºC

Tabla 17: Resumen de características del ESP32

Estas características son las que se tuvieron en cuenta en la elección de los microcontroladores con los cuales se realizaron la página web y la gestión de los displays.

4.1.2 MAX7219

Para reducir el número de pistas, resistencias y simplificar el código y el circuito se ha hecho uso de los circuitos integrados MAX7219 los cuales mediante una comunicación síncrona con el microcontrolador logran controlar hasta ocho displays con ocho leds cada uno, los siete segmentos y el punto.

Los MAX7219 se pueden conectar en cascada permitiendo ampliar el número de displays controlados por el esp32 sin necesidad de aumentar el número de pistas.

Estos circuitos integrados consiguen gobernar hasta ocho displays con ocho leds cada uno con solo 16 pines, para ello se conectan todos los segmentos de los siguientes displays en serie y con un pin de habilitación conectado a cada digito se logra iluminar los segmentos deseados.

Los MAX7219 presentan la siguiente configuración de pines.

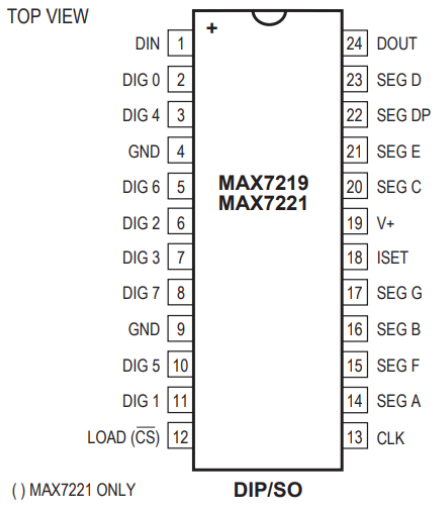


Ilustración 168: configuración de pines del MAX7219. [11]

En la Ilustración 168 se muestra la disposición de los pines. Los pines cuyo nombre empieza por "DIG" corresponden con los pines de habilitación y

deben ir conectados a cada uno de los digitos, los pines cuyo nombre empieza por “SEG” son aquellos que deben ir conectados a los segmentos de los displays.

Los pines “CS”, “CLK” y “DIN” son los dispuestos para la comunicación con el microcontrolador.

El pin “CLK” recibe la señal de reloj del microcontrolador y es el encargado de sincronizar ambos dispositivos.

El pin “CS” es la entrada que seleccionadora del chip. Los datos Serial se cargan en el “shift register” mientras CS se encuentre a nivel bajo.

El pin “DIN” es en el cual recibe la información bit a bit que necesita para gestionar los displays, esta información puede ser enviada desde el microcontrolador o desde otro MAX7219 conectado en cascada a través del pin “DOUT”.

El pin “ISET” se emplea para gestionar mediante hardware la intensidad que circula por cada uno de los diodos, para ello se utiliza una resistencia conectada a ese pin y a la alimentación de este.

Esta resistencia se elige conociendo las características del display y haciendo uso de la siguiente tabla que se encuentra en la hoja de características del circuito integrado.

1. Electro-Optical Characteristics(Ta=25℃)

PARAMETER	SYMBOL	DEVICES (ULTRA-BRIGHT RED)		UNIT	TEST CONDIONS
		TYP	MAX		
Peak Emission Wavelengrth	λp	640		nm	IF=10mA
Forward Voltage	VF	1.8		V	IF=10mA
Reverse Current	IR		50	μA	VR=5V
Segment To Segment (Dot To Dot) Luminonous Intensity Ratio	IV-M	1.5:1			IF=20

Ilustración 169: Corriente y tensión características de los diodos que conforman los displays. **[12]**

I_{SEG} (mA)	V_{LED} (V)				
	1.5	2.0	2.5	3.0	3.5
40	12.2	11.8	11.0	10.6	9.69
30	17.8	17.1	15.8	15.0	14.0
20	29.8	28.0	25.9	24.5	22.6
10	66.7	63.7	59.3	55.4	51.2

Note: *R_{SET} values are in Kilo Ohms (kΩ)*

Ilustración 170: Configuración de pines del MAX7219. [11]

Como se muestra en las tablas anteriores la tensión de los diodos es de 1.8V con una intensidad de paso de 10mA por lo que la resistencia a elegir deberá ser entre 66.7 y 63.7 KΩ.

Como las resistencias normalizadas más cercanas son de 56KΩ y 68KΩ se ha decidido por conectar una de 68KΩ.

x 1	x 10	x 100	x 1.000 (K)	x 10.000 (10K)	x 100.000 (100K)	x 1.000.000 (M)
1 Ω	10 Ω	100 Ω	1 KΩ	10 KΩ	100 KΩ	1 M Ω
1,2 Ω	12 Ω	120 Ω	1K2 Ω	12 KΩ	120 KΩ	1M2 Ω
1,5 Ω	15 Ω	150 Ω	1K5 Ω	15 KΩ	150 KΩ	1M5 Ω
1,8 Ω	18 Ω	180 Ω	1K8 Ω	18 KΩ	180 KΩ	1M8 Ω
2,2 Ω	22 Ω	220 Ω	2K2 Ω	22 KΩ	220 KΩ	2M2 Ω
2,7 Ω	27 Ω	270 Ω	2K7 Ω	27 KΩ	270 KΩ	2M7 Ω
3,3 Ω	33 Ω	330 Ω	3K3 Ω	33 KΩ	330 KΩ	3M3 Ω
3,9 Ω	39 Ω	390 Ω	3K9 Ω	39 KΩ	390 KΩ	3M9 Ω
4,7 Ω	47 Ω	470 Ω	4K7 Ω	47 KΩ	470 KΩ	4M7 Ω
5,1 Ω	51 Ω	510 Ω	5K1 Ω	51 KΩ	510 KΩ	5M1 Ω
5,6 Ω	56 Ω	560 Ω	5K6 Ω	56 KΩ	560 KΩ	5M6 Ω
6,8 Ω	68 Ω	680 Ω	6K8 Ω	68 KΩ	680 KΩ	6M8 Ω
8,2 Ω	82 Ω	820 Ω	8K2 Ω	82 KΩ	820 KΩ	8M2 Ω
						10M Ω

Ilustración 171: Resistencias normalizadas [10]

Por último, a través de los pines V y GND se alimenta el microcontrolador con 5V.

En este proyecto se requieren diez displays, dos por casillero, como cada circuito integrado solo es capaz de gestionar hasta 8 displays se necesitan usar dos MAX7219, uno gestionará los ocho primeros y el segundo los dos siguientes.

La siguiente imagen muestra un ejemplo de cómo se conectan los MAX7219 en cascada. Los condensadores son importantes ya que estabilizan el funcionamiento de los integrados.

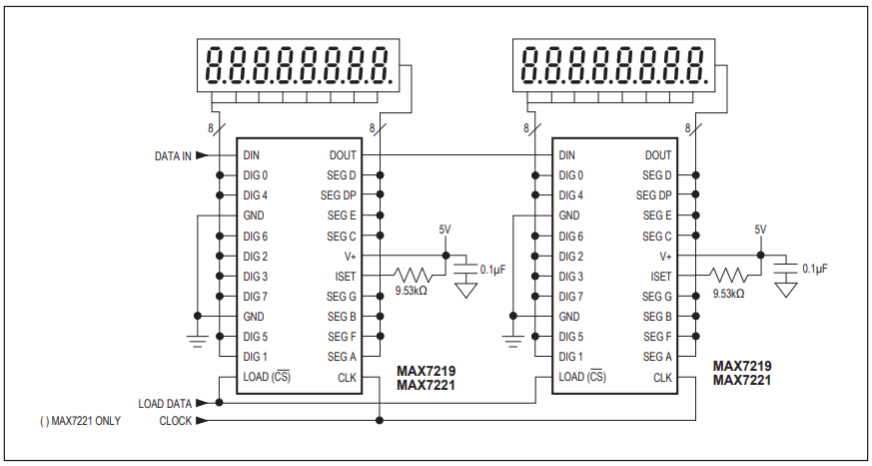


Ilustración 172: Conexionado de MAX7219 en cascada. **[11]**

El funcionamiento del MAX7219 se basa en la visualización dinámica. La visualización dinámica consiste en iluminar los segmentos de los displays en distintos instantes de tiempo. Esto se logra mediante los pines de control de los dígitos, en cada ciclo de reloj se activa el control de uno de los displays y se desactiva el resto.

Los segmentos iluminados a distintos instantes de tiempo son procesados por el cerebro humano dando la ilusión de que todos están iluminados al mismo tiempo.

4.1.3 Circuito electrónico

Se realizó un primer diseño en el cual la fuente de tensión del circuito la proporcionaba un regulador de tensión, este regulador debía gestionar un salto de 19 voltios, esta potencia se disipaba en forma de calor.

La temperatura que alcanzaba era demasiado elevada incluso después de añadir varios radiadores, este aumento de temperatura presentaba una ineficiencia debido a que el consumo era muy superior.

El esquema del circuito es el siguiente donde las resistencias R1 y R2 son de 68KΩ y el condensador C1 de 32pF.

En este circuito se pueden observar ocho displays gestionados por un único MAX7219 cuyos segmentos se encuentran conectados al mismo pin, y cada digito es gobernado desde una pista a un pin de habilitación.

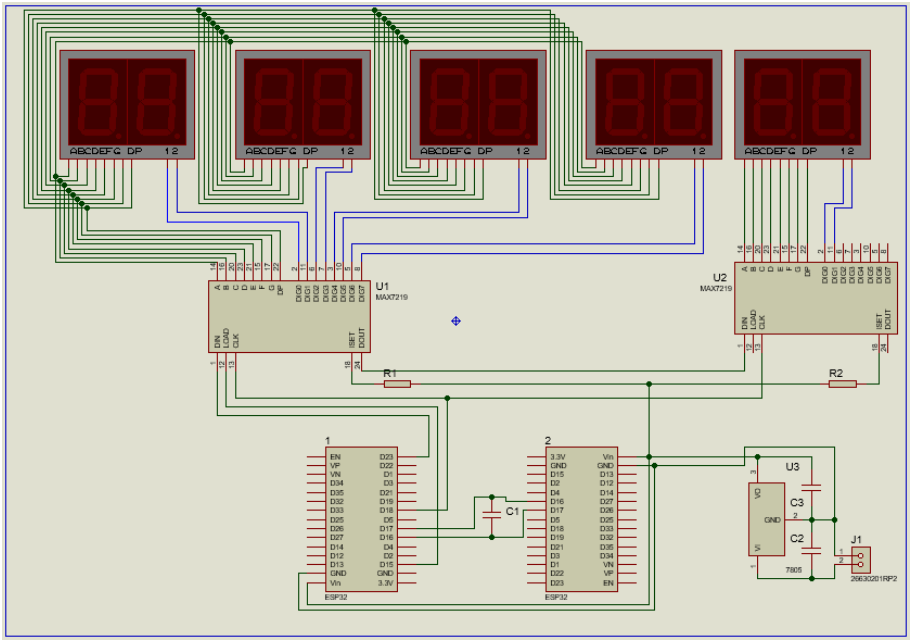


Ilustración 173: Esquema del circuito electrónico.

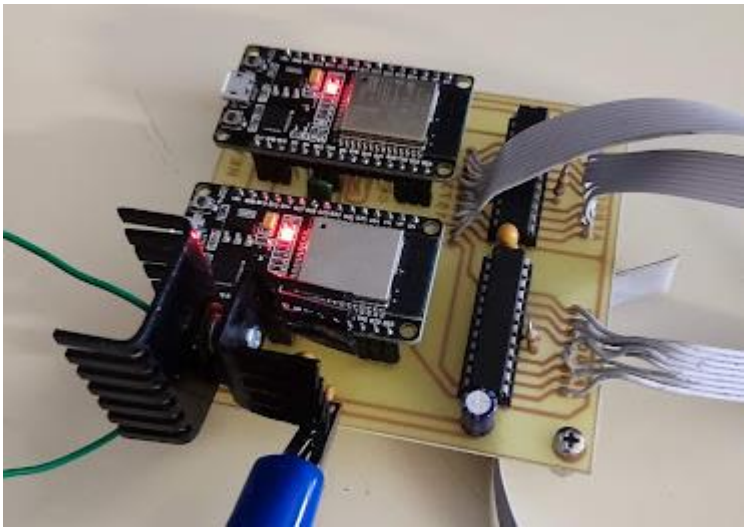


Ilustración 174: Primer prototipo circuito electrónico.

Como se aprecia en la imagen se añadieron los condensadores de 0.1µF encargados de estabilizar el funcionamiento del microcontrolador

Tras este primer diseño se adaptó el circuito sustituyendo el regulador de tensión por un Buck de potencia, el cual al ser un reductor de voltaje conmutado permite transformar tensiones continuas sin grandes pérdidas de energía.

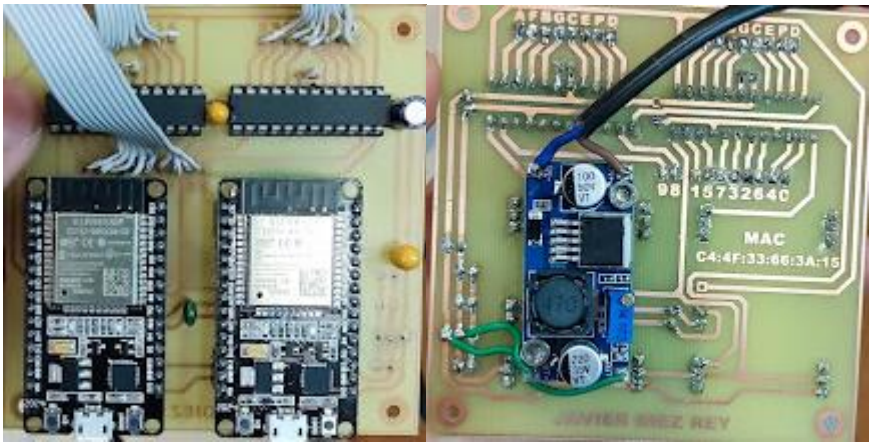


Ilustración 175: Adaptación para el Buck de potencia

4.1.4 BUCK de potencia

El Buck de potencia es un convertidor DC-DC cuya función es proporcionar una tensión de salida continua menor que la tensión de entrada y mantenerla ante variaciones en la carga.

Este convertidor se logra mediante el siguiente circuito.

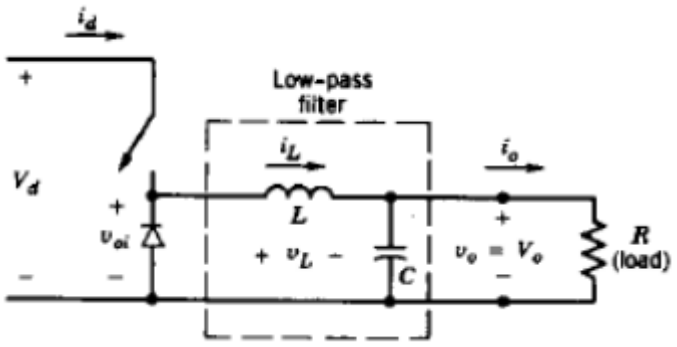


Ilustración 176: Circuito del Buck de potencia [13].

El funcionamiento del Buck de potencia se basa en un conmutador electrónico el cual presenta dos estados, cuando cierra el circuito y cuando lo abre.

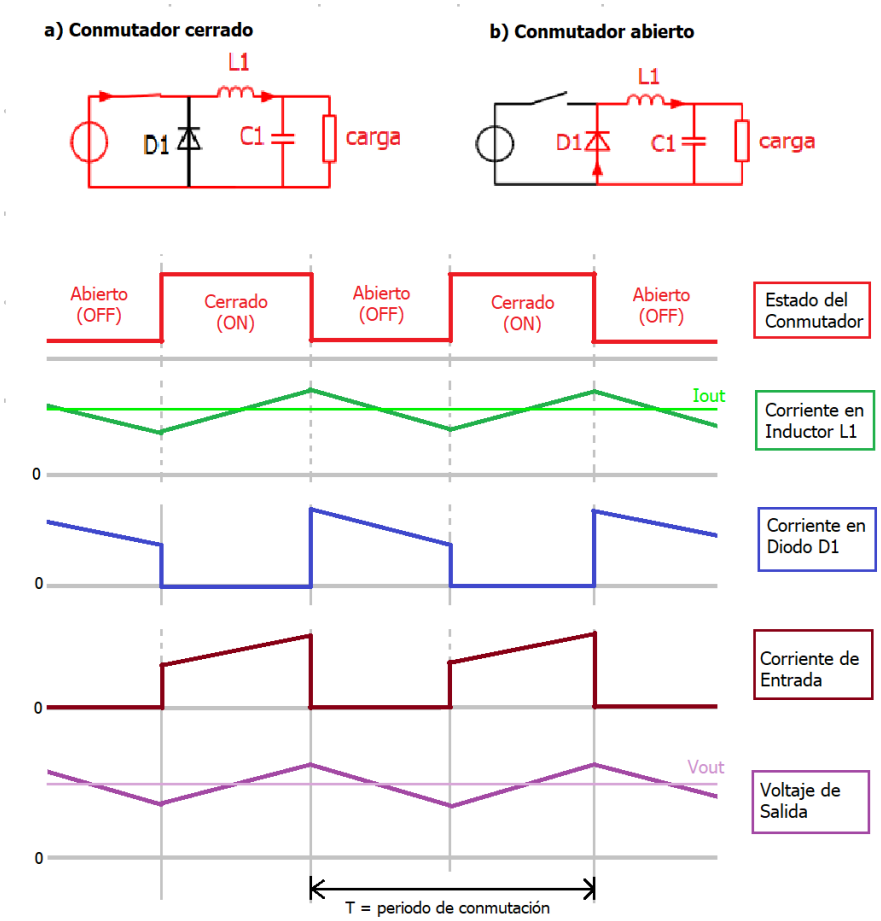


Ilustración 177: Funcionamiento del Buck de potencia **[14].**

El tiempo que el conmutador se encuentra cerrando el circuito se nombre “Ton”, en este tiempo la corriente “id”, proporcionada por la fuente de alimentación que se quiere regular, circula por la bobina “iL” y se divide en dos intensidades una que alimenta la carga que se conecta y la otra el condensador el cual no se encuentra totalmente cargado.

El tiempo que el conmutador se encuentra abriendo el circuito se denomina “Toff”, en este momento la bobina descarga la energía que tiene almacenada en forma de corriente la cual, igual que en el caso anterior, alimenta la carga y el condensador. El circuito se cierra a través del diodo el cual solo conduce durante Toff.

El ciclo de trabajo del conmutador se establece según la siguiente ecuación.

$$D = \frac{T_{on}}{T_{on} + T_{off}}$$

Ecuación 3: Ecuación del plano genérico.

La tensión de salida en modo continuo se logra mediante la siguiente ecuación.

$$V_o = \frac{T_{on}}{T_{on} + T_{off}} \cdot V_d = D \cdot V_d$$

Ecuación 4: Tensión de salida en función del ciclo de trabajo y la tensión de entrada.

4.2 Sensor de bola disponible.

Debido al que el peso de la bola no es suficiente como para activar un final de carrera se ha recurrido al uso de un circuito electrónico el cual utiliza como sensor de proximidad un diodo y un fototransistor.

Este circuito se compone de un sensor óptico CNY70, un diodo, tres resistencias y un transistor.

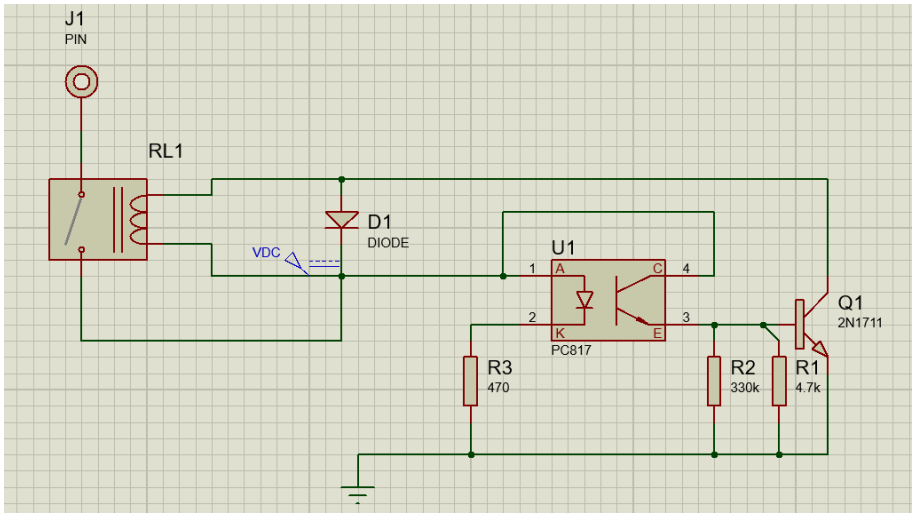


Ilustración 178: Circuito del sensor de posición.

El CNY70 está conformado por un diodo y un fototransistor. La luz emitida por el diodo se refleja en el objeto que se encuentra encima y excita la base del fototransistor permitiendo la circulación de corriente.

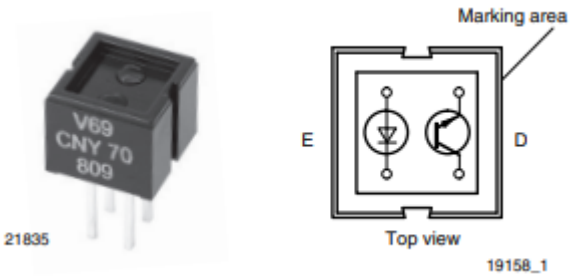


Ilustración 179: Descripción del CNY70 [15].

El circuito es alimentado por 24V por lo que para determinar si es necesario establecer una regulación de tensión antes de alimentar el CNY70 se ha estudiado sus características máximas.

ABSOLUTE MAXIMUM RATINGS (T_{amb} = 25 °C, unless otherwise specified)				
PARAMETER	**TEST CONDITION**	**SYMBOL**	**VALUE**	**UNIT**
OUTPUT (DETECTOR)				
Collector emitter voltage		V_{CEO}	32	V
Emitter collector voltage		V_{ECO}	7	V
Collector current		I_C	50	mA
Power dissipation	$T_{amb} \leq 25$ °C	P_V	100	mW
Junction temperature		T_j	100	°C

Ilustración 180: Características máximas del CNY70 **[15].**

En este caso no se necesita ninguna etapa de regulación de tensión ya que el transistor soporta hasta 32V de emisor-colector.

Se dispone una resistencia en serie con el diodo con el fin de limitar la corriente que circule por este.

BASIC CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)						
PARAMETER	**TEST CONDITION**	**SYMBOL**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
COUPLER						
Collector current	V_{CE} = 5 V, I_F = 20 mA, d = 0.3 mm (figure 1)	I_C (2)	0.3	1.0		mA
Cross talk current	V_{CE} = 5 V, I_F = 20 mA, (figure 2)	I_{CX} (3)			600	nA
Collector emitter saturation voltage	I_F = 20 mA, I_C = 0.1 mA, d = 0.3 mm (figure 1)	V_{CEsat} (2)			0.3	V
INPUT (EMITTER)						
Forward voltage	I_F = 50 mA	V_F		1.25	1.6	V
Radiant intensity	I_F = 50 mA, t_p = 20 ms	I_e			7.5	mW/sr
Peak wavelength	I_F = 100 mA	λ_P	940			nm
Virtual source diameter	Method: 63 % encircled energy	d		1.2		mm
OUTPUT (DETECTOR)						
Collector emitter voltage	I_C = 1 mA	V_{CEO}	32			V
Emitter collector voltage	I_E = 100 µA	V_{ECO}	5			V
Collector dark current	V_{CE} = 20 V, I_F = 0 A, E = 0 lx	I_{CEO}			200	nA

Ilustración 181: Características básicas del CNY70 **[15].**

$$R3 = \frac{24}{0.05} = 480\Omega \simeq 475\Omega$$

Ecuación 5: Cálculo de la resistencia del diodo.

En el circuito la resistencia de 475Ω estaba sometida a mucha potencia por lo que se colocaron varias resistencias en paralelo con el fin de reducir la cantidad de watios que se disipaban en cada una de ellas, evitando así que se quemasen.

Las resistencias de $330K\Omega$ y $4.7K\Omega$ se utilizan para dar más rapidez en la conmutación del segundo transistor.

Por último, el diodo es necesario para proteger el segundo transistor.

Cuando el sensor detecta una bola aporta una corriente a la base del transistor Q1 el cual entra en saturación permitiendo el paso de corriente entre el colector y el emisor, se cierra el circuito y se excita la bobina RL1.

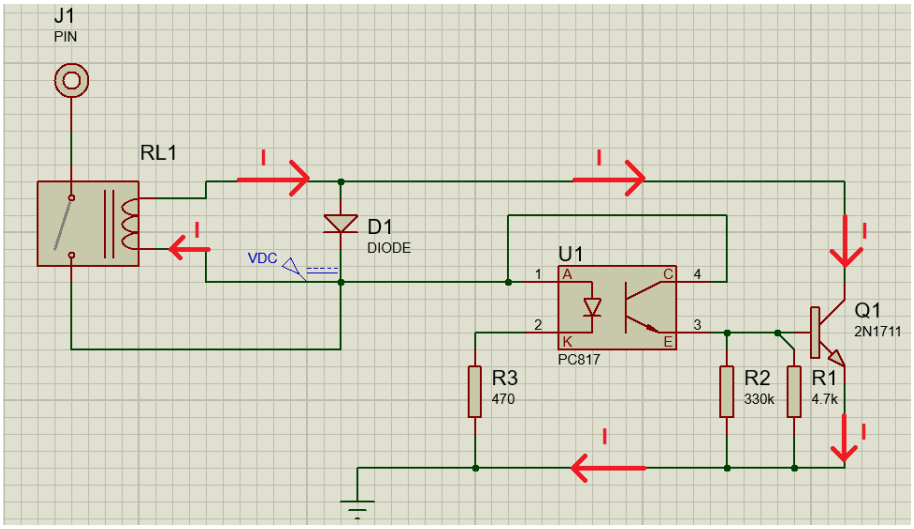


Ilustración 182: Circulación de corriente con el transistor Q1 en saturación.

La ecuación que relaciona la caída de tensión en una bobina y la corriente que circula por ella es la siguiente.

$$V_L = L \cdot \frac{di_L}{dt}$$

Ecuación 6: Relación de tensión y corriente en una bobina.

De esta ecuación se puede deducir que la corriente en una bobina no puede variar de forma brusca, debe ser continua. En caso de que el sensor deje de detectar una bola y el transistor Q1 se establezca en corte, la bobina descargará la energía acumulada en el diodo.

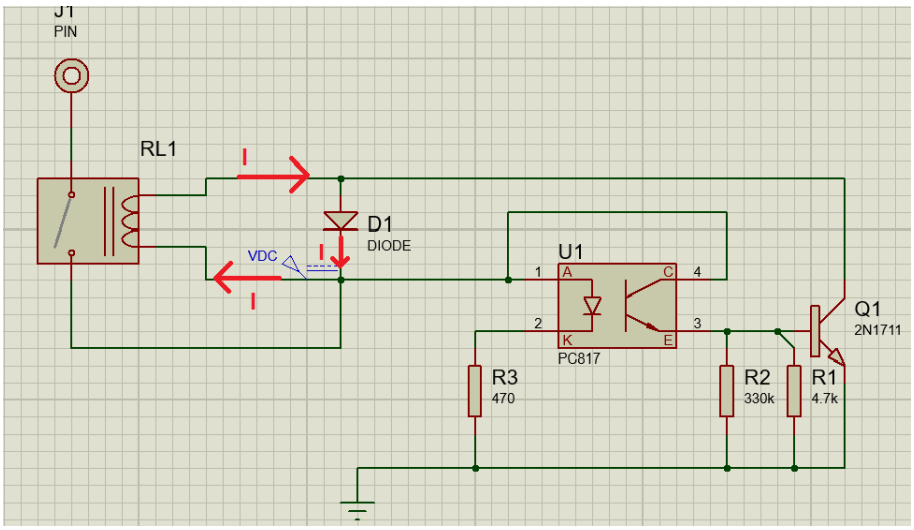


Ilustración 183: Circulación de corriente con el transistor Q1 en corte.

En caso de prescindir del diodo la corriente circulará por el transistor independientemente de que este se encuentre en corte. Esta circulación de corriente quemará el semiconductor.

5. Fabricación de todos los elementos del sistema

En este apartado se explica cuál ha sido el planteamiento inicial de las maquetas y circuitos, y como estos han evolucionado hasta la versión final.

La fabricación de todos los elementos que conforman el trabajo de fin de grado se divide en cuatro puntos.

1. **Diseño, evolución y fabricación de los prototipos de las maquetas.**

   En este apartado se muestra cual ha sido el planteamiento inicial del plano inclinado y el Plinko, las modificaciones necesarias que se han realizado en los prototipos para adaptarse a los distintos problemas y el resultado final.

2. **Diseño y fabricación de la electrónica.**

   Para el conexionado de los circuitos electrónicos, explicados anteriormente, se han realizado varias PCBs las cuales han requerido de un diseño y varios ataques químicos.

3. **Circuito neumático y componentes que lo conforman**

   Se muestra el circuito electroneumático y los componentes que lo integran-

4. **Conexionado del control del ABB con los sensores y actuadores de la estación.**

En este punto se muestra cómo se han conectado los sensores y actuadores con la controladora y como esta alimenta los circuitos electrónicos.

5.1 Diseño, evolución y fabricación de los prototipos de las maquetas.

En este proyecto se han realizado dos maquetas, el Plinko y el plano inclinado, las cuales han requerido de un planteamiento y diseño previo el cual se ha ido modificando para afrontar los distintos problemas que surgían durante la puesta en marcha del proyecto.

En un primer diseño las dos maquetas se plantearon en plástico para poder ser fabricadas por una impresora 3D. Sin embargo, se optó por la fabricación de los prototipos en madera para poder corregir errores de diseño con mayor facilidad.

5.1.1 Plano inclinado

El primer diseño del plano inclinado se diseñó en FreeCAD, para ello se descargó el modelo en 3D del pistón que debía empujar las bolas y a partir de este se diseñó el soporte.

En este diseño se creó un tronco de cono hueco a modo de embudo para no necesitar mucha precisión en el almacenamiento de las bolas.

Se pensó en un rebaje de las paredes laterales en el lugar donde se depositaba la bola, tras ser empujada por el pistón, para ser recogida y un aumento de la pared final para evitar que en caso de que la bola alcanzase mucha velocidad saliese de la maqueta.

El sensor que detectaba la presencia de bola para ser dispensada estaba conformado por un final de carrera con una palanca corta.

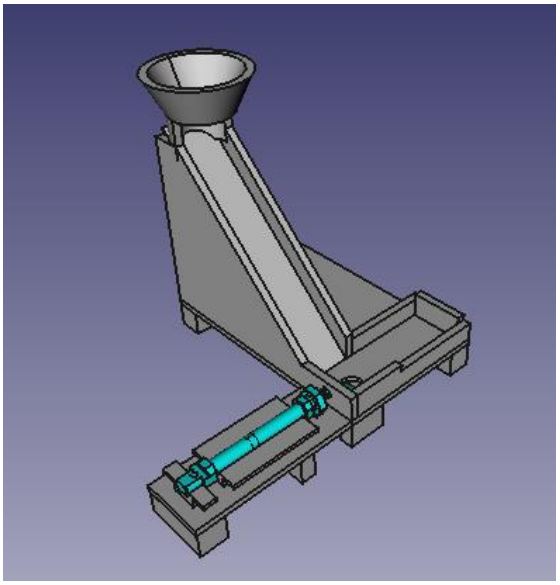


Ilustración 184: Vista 3D del plano inclinado.

Aprovechando que el diseño se realizó en FreeCAD se ha utilizado esta herramienta para obtener los planos del planteamiento inicial de dicha maqueta. En estos planos, que se encuentran en anexos, no se han acotado los soportes que daban mayor altura a la maqueta debido a que no considero que dichas dimensiones sean relevantes y a que el exceso de cotas dificultaría su interpretación.

El principal problema de este diseño fue el final de carrera encargado de identificar la disponibilidad de bola para ser dispensada, esto se debe a que las bolas presentan un muy bajo peso, unos 10g.

Una solución que se estudió fue el cambio del final de carrera de palanca corta por uno con la palanca más larga y mayor sensibilidad, esta solución no fue viable y se terminó optando por una solución electrónica.

Tras el diseño en FreeCAD se construyó un primer prototipo en madera, el cual ya sufrió alguna modificación debido a un más fácil montaje.

En este prototipo se combinan tanto piezas en 3D como en madera. Las piezas en 3D son el embudo, el soporte del pistón y una pieza con un relieve encargada de que tras el empujón del pistón la bola termine en una determinada posición.

El primer prototipo presenta dos cambios con respecto al diseño original:

La pieza diseñada para el posicionamiento de la bola en la posición donde posteriormente es recogida.

La altura se establece con pequeñas planchas de madera debido a que no necesitan sostener mucho peso.

Ilustración 185: Primera maqueta del plano inclinado.

Finalmente, la maqueta se realizó en madera debido a un carácter estético.

La maqueta final no dista mucho del diseño original salvo por la sustitución del final de carrera por un sensor electrónico.

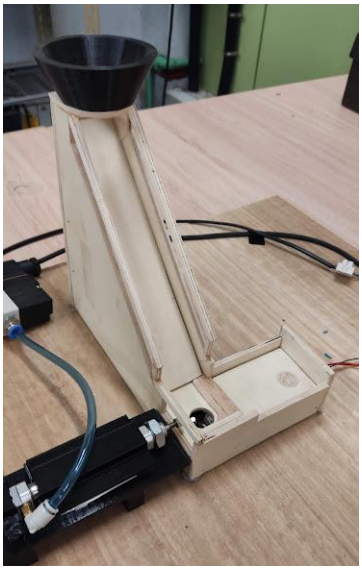


Ilustración 186: Maqueta final del plano inclinado.

5.1.2Plinko

Al igual que en el plano inclinado el primer diseño del Plinko se realizó haciendo uso de la herramienta de diseño en 3D FreeCAD, con la intención de imprimirla en una Anycubic i3 mega,

El Plinko consta de dos partes muy diferenciadas, la primera es un tablero con clavos donde la bola golpea según va descendiendo, cambiando su trayectoria. La otra parte es un casillero donde la bola acaba su recorrido y de donde es recogida para posteriormente almacenarla en el plano inclinado.

El tablero con clavos presenta las dimensiones representadas en el plano adjunto en anexos y sus clavos se encuentran equiespaciados según el siguiente plano.

La segunda parte es la que más modificaciones ha sufrido y cuyo primer diseño se compone de dos piezas debido a que el tamaño de la pieza global es demasiado grande para poder ser impresa por la impresora del departamento.

Esta parte tuvo un primer diseño en FreeCAD, herramienta en la que se importó un diseño en 3D de un final de carrera utilizado para crear un soporte para este.

Estas piezas presentan unos casilleros en forma de tronco de pirámide que finalizan en un final de carrera como en el caso del plano inclinado.

Se diseñaron unas paredes laterales cuya finalidad es conducir la bola a los casilleros además de para servir de soporte para el tablero con los clavos.

En la parte trasera se diseñó un pequeño soporte para que la cama de clavos se apoyase sobre este, logrando que las dos partes encajasen formando siempre el mismo ángulo.

La segunda parte está compuesta por dos piezas las cuales se unen haciendo uso de unos taladros y de unos cilindros salientes que encajan unos en otros.

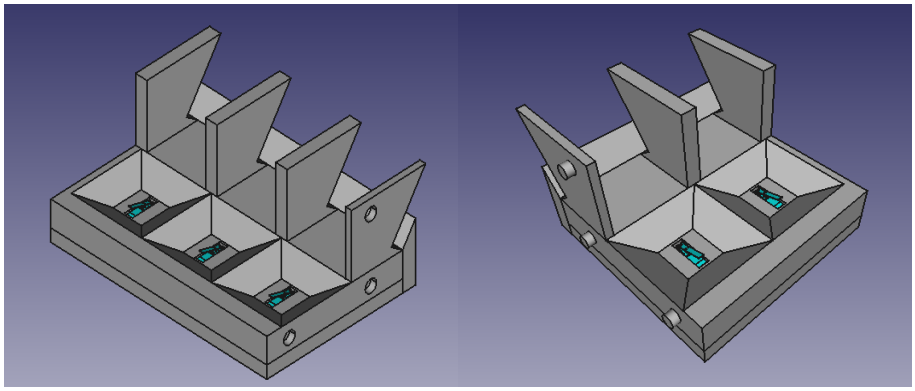


Ilustración 187: Vista 3D del casillero del Plinko.

Al igual que en la anterior maqueta el diseño se realizó en FreeCAD y se ha utilizado esta herramienta para obtener los planos del planteamiento inicial de dicha maqueta.

En un primer prototipo se añadió una tabla en la parte frontal cuya función era hacer de tope evitando que la bola saliese del casillero y se modificó el soporte final haciéndolo más grande para dar mayor apoyo al tablero sobre el cual se encuentran los clavos.

En este prototipo no se pensó en el posicionamiento de los finales de carrera ya que el principal cometido de este era conocer las deficiencias del diseño.

El resultado del primer prototipo fue el siguiente.

Ilustración 188: Primera maqueta del casillero del Plinko.

En las imágenes del prototipo se pueden observar cómo los aspectos estructurales de la maqueta presentan una gran importancia debido a la inclinación y al peso del tablero que sostiene el casillero.

En esta maqueta las piezas que sostienen el peso del tablero, sobre el cual descienden las bolas, se encuentran en los extremos a diferencia del primer diseño que eran más pequeñas y se encontraban a lo largo de la parte posterior del casillero.

Esta maqueta presentaba tres inconvenientes importantes al robot. El primero era que cuando el TCP se dirigía a recoger la bola del primer casillero se encontraba cerca de una singularidad, el segundo inconveniente era que los cables que conectan la controladora con la pinza corrían el riesgo de enredarse con los clavos estropeando la maqueta, y el tercero era que la tabla frontal era insuficiente para frenar la bola y evitar que saliese del casillero.

Estos inconvenientes fueron solventados separando el casillero del soporte del tablero y añadiendo una tabla frontal como se muestra en la siguiente imagen.

Ilustración 189: Primera modificación del casillero del Plinko.

En esta maqueta se solventaron la mayoría de los problemas, sin embargo, limitaba el número de bolas que se podían soltar en la parte superior del Plinko, antes de ser recogidas, en un máximo de tres.

Este problema ocurría ya que las paredes de la pinza debían posicionarse de tal forma que no golpeasen la pared frontal, esta posición de la pinza podía romper la maqueta si dos bolas se encontraban en casilleros contiguos, ya que al descender la pinza para recoger una de las dos bolas empujaba la otra contra el casillero.

Para solucionar este problema y conseguir aprovechar al máximo el casillero se optó por sustituir la pared frontal de madera por otra de otro material flexible el cual pudiese ser comprimido por la pinza sin ocasionar daños al casillero. Cuando la pinza no se encontrase comprimiéndolo debía recuperar su posición original.

Uno de los materiales que por su precio y características más se adaptaba a las necesidades del proyecto fue una esponja, el principal inconveniente de este material era puramente estético.

Se modificó la maqueta sustituyendo la pared frontal de madera por una de esponja y añadiendo los sensores conformados por finales de carrera con una palanca más larga que la propuesta en el primer diseño. Esta modificación en los sensores fue necesaria ya que el peso de las bolas no era suficiente para activar el microinterruptor.

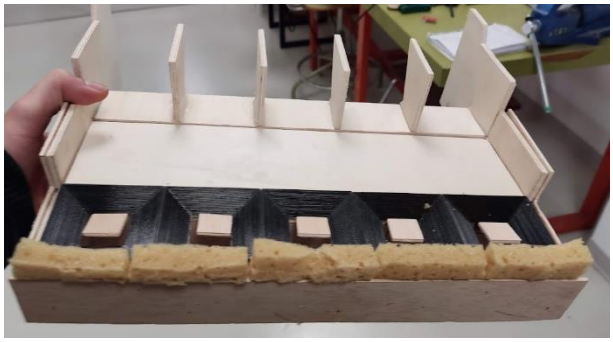


Ilustración 190: Segunda modificación del casillero del plinko.

Este prototipo cumplía perfectamente los objetivos del proyecto, sin embargo, no era estético por lo que se le realizaron unas modificaciones.

Se le cambio la esponja por unas piezas, de baja densidad, impresas en 3D con un filamento elástico el cual cumple con el cometido de la esponja y soluciona el aspecto estético.

La maqueta final resultante es la siguiente.

Ilustración 191: Maqueta del Plinko final.

5.2 Diseño y fabricación de la electrónica.

Para el diseño y fabricación de los circuitos impresos utilizados en este proyecto es imprescindible una placa formada por un material resistente al fuego y aislante eléctrico el cual sirve como soporte a las pistas de cobre y a los componentes electrónicos que conforman el circuito.

En este proyecto las placas son de baquelita un material compuesto de resina Fenólica y fibra de papel. Entre las características de este material se encuentran su bajo precio, baja resistencia al calor y malas características mecánicas y eléctricas.

Estos circuitos deben presentar unas características que les permitan soportar la mecanización de la placa, el peso de los componentes, evitar descargas eléctricas debidas a un mal aislamiento entre pistas muy próximas, una buena refrigeración, un tamaño de las pistas adecuado para soportar las corrientes…

El material conductor que recubre las dos capas de la baquelita está compuesto por cobre electrolítico cuyos espesores más comunes son 35µm y 70µm. El cobre electrolítico tras un insolado, rebelado y atacado forman las pistas y los pads.

El cobre electrolítico puede encontrarse en varias capas, los circuitos impresos pueden ser de simple cara, doble cara o multicapa, en este caso con la placa de simple cara ha sido suficiente.

La fabricación de las placas de circuito impreso se compone de varias partes:

1. Diseño de los fotolitos
2. Insolado de la placa, proceso en el cual se transfieren el dibujo del fotolito a la placa.
3. Revelado de la película fotosensible mediante el uso de un medio básico.
4. Ataque ácido al cobre no protegido por la fotorresina.
5. Limpieza de la placa.
6. Taladrado de los pads.

5.2.1 Diseño de los fotolitos

Se han diseñado tres placas de circuito impreso:

- La que conforman los esp32 y los circuitos integrados MAX7219.
- Sobre la que se conectan los displays.

- El sensor de bola disponible para ser dispensada.

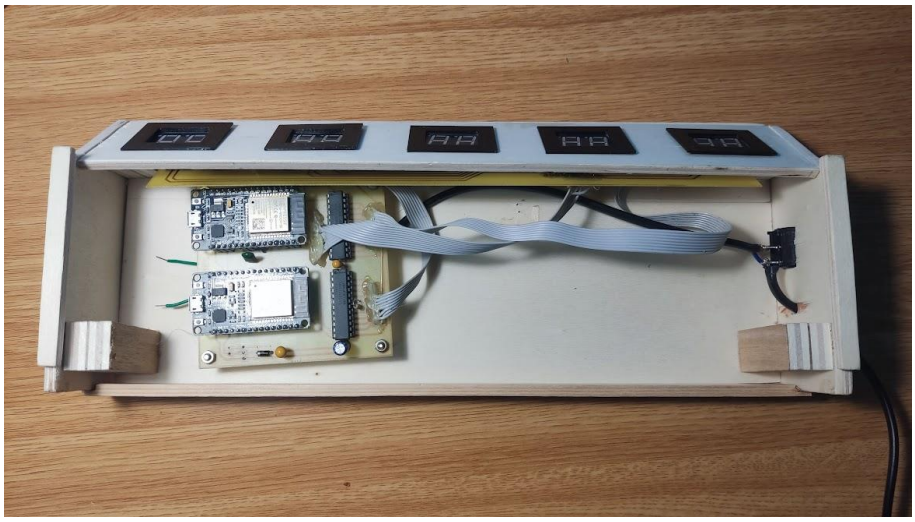


Ilustración 192: Placas de circuito impreso finales y maqueta que las contiene.

El software de diseño de PCBs elegido ha sido Proteus en el cual se han rutado los circuitos.

El resultado del diseño del fotolito de la primera PCB es el siguiente, en él se puede observar cómo se han impreso nombre a los pines para que el conexionado con los displays sea los más rápido y sencillo posible minimizando el número de errores.

Conociendo el esp32 encargado de conectarse a la red de la escuela se ha apuntado su MAC para identificar sin necesidad de cambiar el software cual es el identificativo de dicho dispositivo en la red.

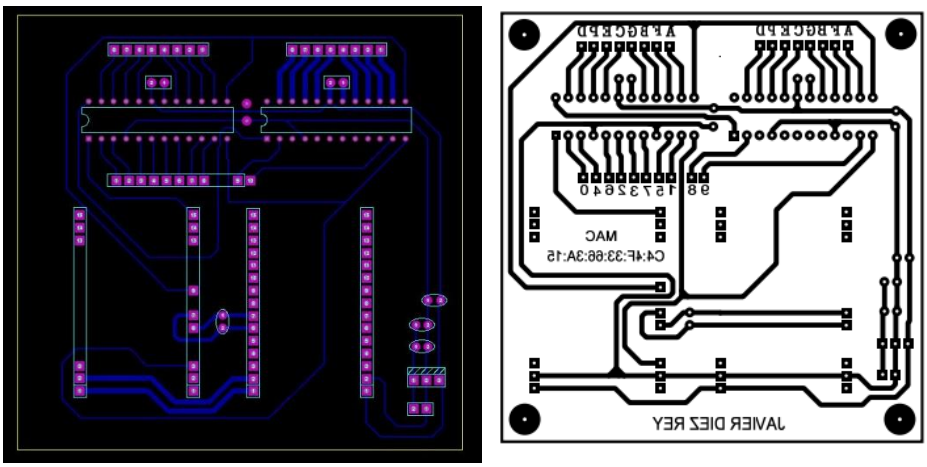


Ilustración 193: Fotolito de la PCB que contiene los dos esp32 y MAX7219.

La placa de circuito impreso que contiene los displays presenta tres tiras de pines.

La tira superior es la que se conecta con el multiplexor que elige a que leds se les debe suministrar corriente.

La tira lateral izquierda alimenta los leds de los ocho primeros dígitos mientras que la tira lateral derecha alimenta los dos últimos.

En este caso no se añadió ningún identificativo en los pines debido a la falta de espacio.

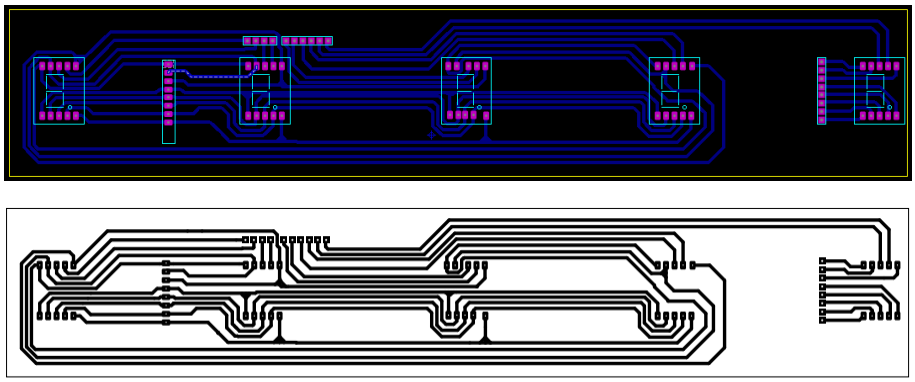


Ilustración 194: Fotolito de la PCB que contiene los displays.

La placa de circuito impreso del sensor identificador de bolas presentaba como mayor problemática el espaciado de los componentes y la posición del sensor ya que el espacio debajo del plano inclinado es muy reducido con muy poco margen para desplazar la placa.

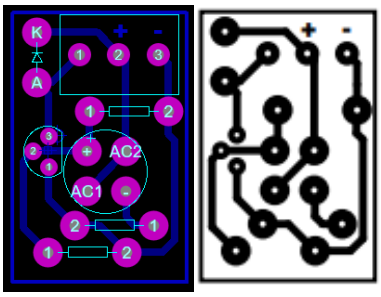


Ilustración 195: Fotolito de la PCB que conforma el sensor de bola disponible.

Los fotolitos resultantes se han impreso en modo espejo en un papel traslucido como el papel cebolla para que la tinta se encuentre lo más cerca de la PCB y el ataque con luz actínica sobre las partes que se quieren proteger sea menor.

5.2.2 Taladrado de los pads

En el caso de la primera placa el taladrado se realizó haciendo uso de la CNC del laboratorio de Métodos y herramientas de diseño electrónico, y se hizo antes del insolado. El resto se taladró al finalizar todo el proceso y de forma manual.

En este documento solo se documenta con imágenes la fabricación de la primera placa ya que el resto de las placas se realizaron de forma muy similar.

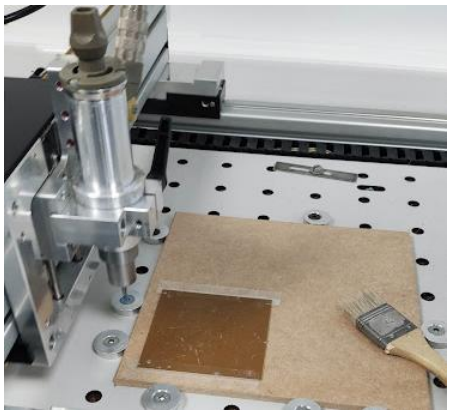


Ilustración 196: Taladrado mediante el uso de CNC.

5.2.3 Insolado de la placa

Tras el taladrado, se situó el fotolito sobre la PCB para que los taladros y los pads fueran coincidentes y tras esto se insoló durante 4 minutos aproximadamente.

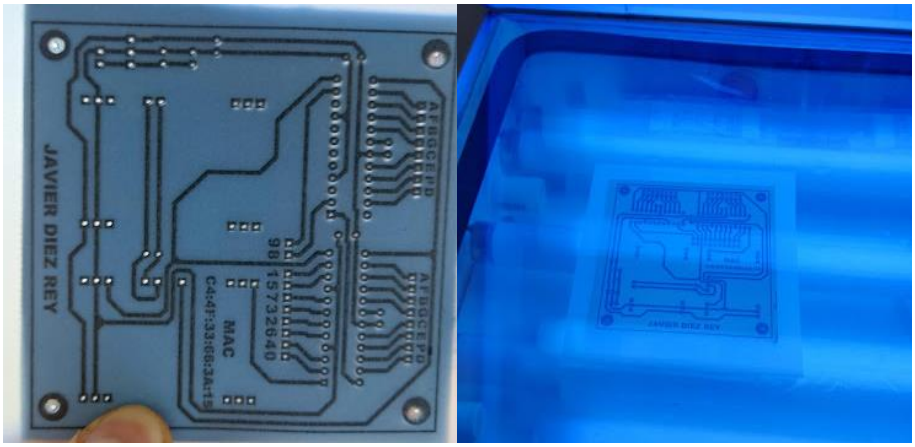


Ilustración 197: Insolado de las PCBs.

5.2.4 Revelado de la película fotosensible

El resultado se rebeló haciendo uso de un medio básico conformado por 50ml de una mezcla de agua y sosa cáustica.

Este proceso requirió el uso de unas protecciones como fueron una bata, unos guantes de látex, unas pinzas además de realizar la reacción química en un lugar perfectamente aireado y con acceso a agua corriente.

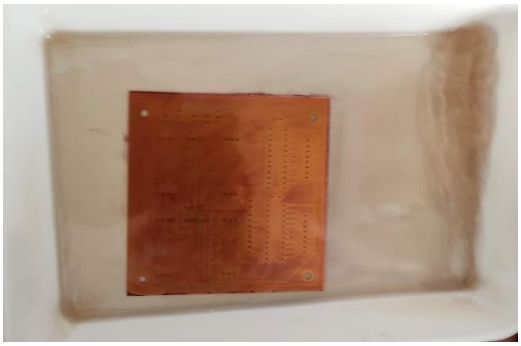


Ilustración 198: Rebelado de las PCBs.

Después de unos minutos en contacto con este medio y tras un lavado con agua el resultado fue el siguiente.

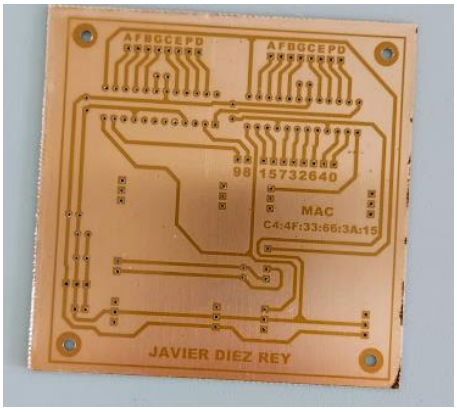


Ilustración 199: Resultado tras el rebelado.

5.2.5 Ataque ácido al cobre

El siguiente paso consiste en el atacado del cobre, expuesto a la luz actínica, sumergiéndolo en un medio ácido.

El medio ácido está compuesto por 20 ml de agua, 20 ml de ácido clorhídrico al 35% y 20 ml de agua oxigenada.

Las semirreacciones por las cuales el agua oxigenada es capaz de oxidar el cobre son las siguientes.

$$Cu^0 \rightarrow Cu^{+2} + 2e^-$$

$$H_2O_2 + 2H^+ + 2e^- \rightarrow 2H_2O$$

Ecuación 7: Semirreacciones de la oxidación del cobre.

El potencial de la primera semirreacción es de -0.34V y el potencial de la segunda semirreacción es de +1.77V al sumar los dos potenciales el resultado es positivo siendo esta reacción espontánea y cuyo excedente energético se disipa en forma de calor.

El resultado de dichas semirreacciones es el siguiente.

$$Cu^0 + H_2O_2 + 2H^+ \rightarrow Cu^{+2} + 2H_2O$$

Ecuación 8: Reacción resultante de la oxidación del cobre.

Los protones de hidrógeno proceden del ácido clorhídrico por lo que la reacción real es la siguiente, en la que el desperdicio adquiere un color verdoso derivado del cloruro cúprico.

Ilustración 200: Ataque con ácido clorhídrico.

5.2.6 Limpieza de la placa.

Finalmente, el resultado de los anteriores ataques se limpia con alcohol para eliminar los restos del revelador y la fotorresina.

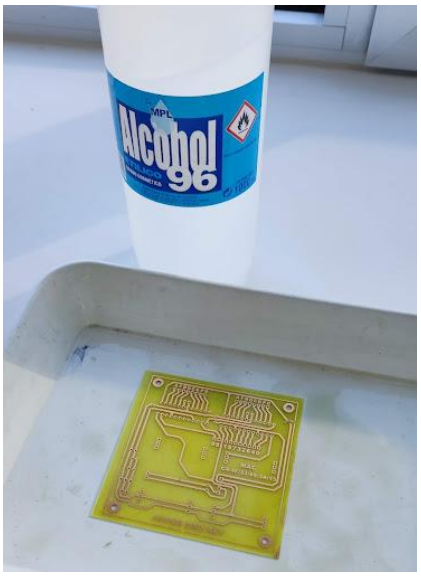


Ilustración 201: Limpieza con alcohol de la PCB final.

5.3 Circuito neumático y componentes que lo conforman

El circuito neumático es muy simple y se conforma de un pistón neumático de simple efecto con retroceso por muelle, una válvula 3/2 normalmente cerrada con accionamiento eléctrico, es decir, una electroválvula

de tres vías y dos posiciones la cual se activa con corriente eléctrica y retrocede a la posición normal mediante un muelle.

Este circuito es alimentado por la toma de aire a presión del taller de robótica el cual presenta una válvula reguladora de presión.

El pistón neumático es del fabricante SMC y su referencia es la siguiente: CD85N10-40S-B.

La electroválvula es del mismo fabricante y su referencia es la siguiente: VT307-5DZ1-02F-Q.

El circuito final se muestra en el siguiente esquema.

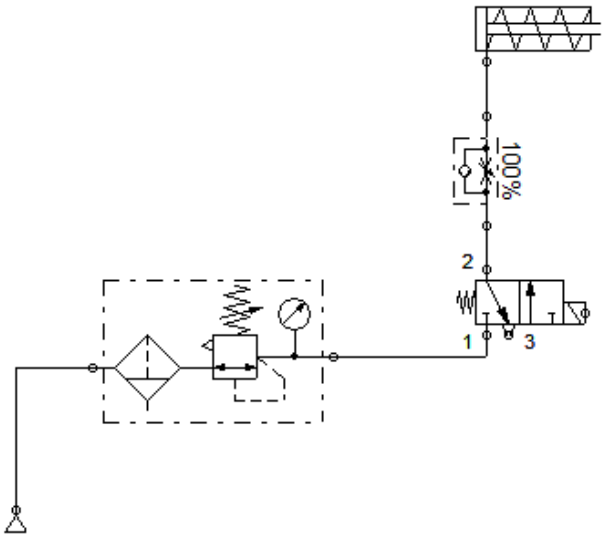


Ilustración 202: Circuito neumático.

5.4 Conexionado del control del ABB con los sensores y actuadores de la estación.

El conexionado de los sensores y actuadores con el controlador del irb120 se realiza a través de una DB25, con 25 pines, de la cual se emplean 10 pines que se conectan con las entradas y salidas digitales de la Board10.

En este proyecto el conexionado del controlador con la DB25 no ha sufrido ninguna modificación con el objetivo de permitir intercambiar las estaciones de trabajo de distintos proyectos de la forma más fácil y rápida posible.

Para explicar cómo se conectan los distintos elementos de la estación con el controlador es necesario definir el punto de vista desde el cual se va a explicar. En este proyecto el conexionado se va a detallar siguiendo la numeración de pines de la DB25 Macho.

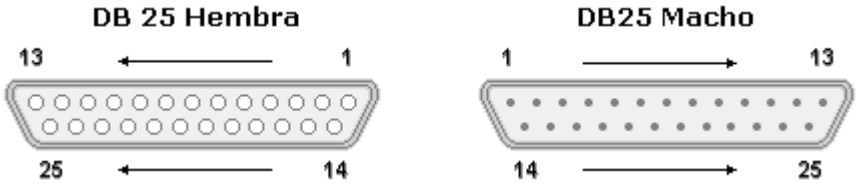


Ilustración 203: configuración de pines de la DB25 **[16].**

La conexión entre las entradas y salidas del controlador con la DB25 es la siguiente.

PIN	Entrada/Salida digital	Sensor/Actuador
1	Di1	Final de carrera 1
2	Di3	Final de carrera 3
3	Di5	Final de carrera 5
4	¿?	-
5	¿?	-
6	¿?	-
7	¿?	-
8	¿?	-
9	¿?	-
10	¿?	-
11	¿?	-
12	GND	GND
13	GND	GND
14	Di2	Final de carrera 2
15	Di4	Final de carrera 4
16	Di6	Sensor de bola disponible
17	¿?	-
18	24V	24V
19	¿?	-
20	¿?	-
21	24V	24V
22	¿?	-
23	¿?	-
24	Do1	Electroválvula
25	¿?	-

Tabla 18: Asignación de pines de la DB con entradas y salidas del controlador.

Los finales de carrera presentan tres patillas las cuales se conectan conforme el siguiente circuito, logrando que mientras no se encuentren presionados la tensión en el pin de entrada al controlador sea de 0V, y cuando sean presionados la tensión sea de 24V.

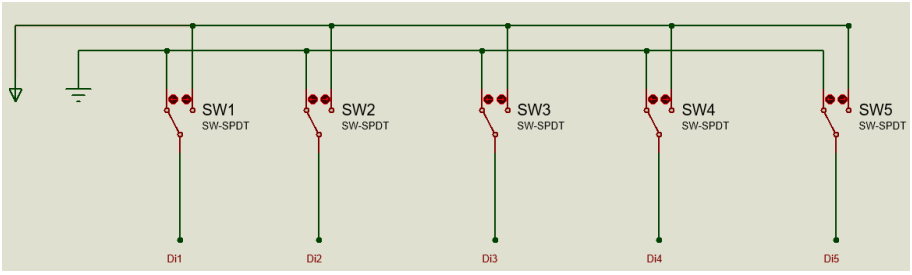


Ilustración 204: Conexionado de los finales de carrera.

La electroválvula se conecta con dos pines, uno se encuentra en GND y el otro se conecta a Do1, cuando do1 presente una salida de 24V la bobina de la electroválvula se excitará permitiendo el paso de aire al pistón.

Hay dos pines de los cuales se han obtenido tanto 24V como GND, una pareja de estos se ha empleado para alimentar los sensores y la otra pareja para alimentar la electrónica conformada por los esp32 y los displays.

6. Introducción al análisis de los datos de un primer muestreo

Se ha realizado un estudio estadístico sobre la probabilidad de que la bola acabe en cada uno de los casilleros a parir del lugar desde el cual se suelta. Este estudio se ha realizado mediante una muestra de 218 valores.

La bola puede ser soltada sobre una línea imaginaria, que se encuentra en la parte superior del Plinko, de longitud 240mm, por lo que para estimar la probabilidad de que caiga en uno u otro casillero se ha de dividir dicha línea en varios tramos y realizar un estudio estadístico de cada uno de ellos.

Los valores obtenidos en la muestra se representan en la siguiente gráfica donde se pude observar desde donde han sido lanzados y en que casillero han terminado.

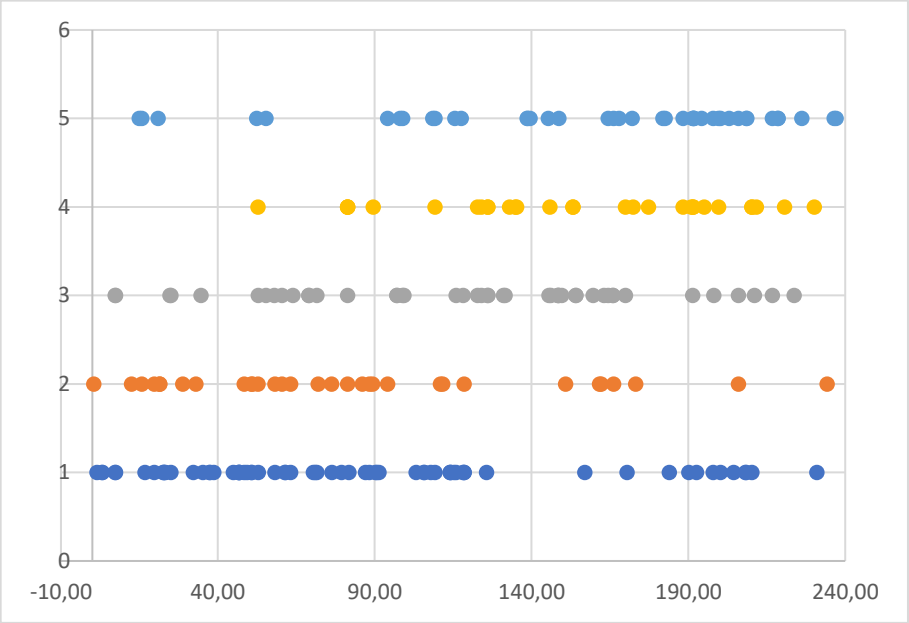


Ilustración 205: Bolas caídas en cada casillero en función de la posición inicial.

La división se ha realizado en 12 tramos ya que si se mira el tablero desde la parte superior hay 13 columnas de clavos a lo largo de la línea desde la cual puede ser soltada la bola.

Como cada vez que la bola golpea un clavo esta puede ser dirigida a uno de los dos lados posibles cada vez que la bola golpea un clavo se produce un ensayo de Bernoulli.

La distribución que sigue la bola en su caída por el tablero es una distribución binomial ya que esta se define como una distribución de probabilidad discreta que contabiliza el número de éxitos de una secuencia de

varios ensayos de Bernoulli independientes entre sí con una probabilidad fija de que se produzca este éxito.

Conociendo que la estimación del parámetro de proporción de una distribución binominal se calcula de la siguiente forma.

$$\hat{P} = X/n$$

Ecuación 9: Estimación de probabilidad

La varianza corresponde con la siguiente fórmula.

$$Var(\hat{P}) = p \cdot \frac{1-p}{n}$$

Ecuación 10: Estimación de varianza

Haciendo uso del teorema central del límite el cual establece que si el número de variables aleatorias independientes es muy elevado con una media conocida y una varianza distinta de cero la distribución se aproxima a una distribución Normal.

$$\frac{X - np}{\sqrt{np \cdot (1-p)}} \rightarrow N(0,1)$$

Ecuación 11: Distribución de probabilidad

A partir de la anterior formula se llega a la estimación del intervalo de confianza al 100·(1- α)%.

$$\left[\hat{p} - Z_{1-\frac{\alpha}{2}} \cdot \sqrt{\hat{p} \cdot \frac{1-\hat{p}}{n}}, \hat{p} + Z_{1-\frac{\alpha}{2}} \cdot \sqrt{\hat{p} \cdot \frac{1-\hat{p}}{n}} \right]$$

Ecuación 12: Intervalo de confianza

Donde la probabilidad y el error son:

$$Probabilidad: \hat{p} = \frac{x}{n}$$

$$Error: \ Z_{1-\frac{\alpha}{2}} \cdot \sqrt{\hat{p} \cdot \frac{1-\hat{p}}{n}}$$

Ecuación 13: Estimación de probabilidad y error de estimación.

Particularizando los anteriores conceptos al estudio del comportamiento de este proyecto se obtienen las siguientes tablas.

RECUENTO DE CAIDAS EN CASILLEROS POR TRAMOS						
TRAMO	CASILLERO 1	CASILLERO 2	CASILLERO 3	CASILLERO 4	CASILLERO 5	TOTAL
1	2	3	1	3	6	15
2	3	6	4	0	5	18
3	9	6	1	2	4	22
4	8	0	3	2	4	17
5	10	2	3	4	4	23
6	10	3	8	0	2	23
7	6	2	4	4	1	17
8	2	2	7	2	3	16
9	3	4	3	1	6	17
10	7	3	6	4	2	22
11	6	1	4	6	3	20
12	2	0	2	2	2	8
						218

Tabla 19: Recuento de caídas en casilleros por tramos

En la siguiente gráfica se muestra la proporción de bolas caídas en cada casillero.

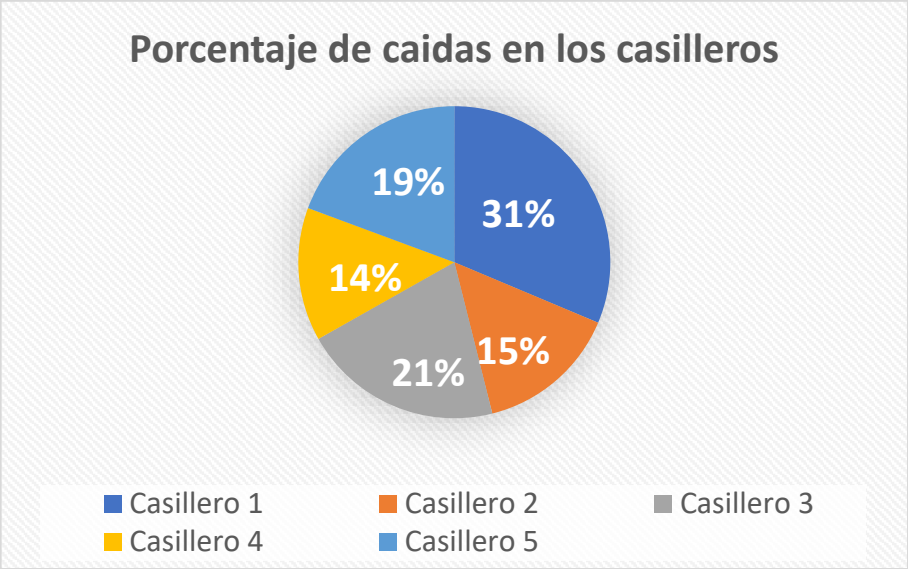


Ilustración 206: Bolas caídas en cada casillero en función de la posición inicial.

El recuento de caídas en los casilleros por tramos recogidos en la anterior tabla es necesario para conocer la probabilidad de que caiga una bola en un casillero tras ser soltada en un determinado tramo, para ello ha de usarse la **Ecuación 9.**

ESTIMACIÓN DE PROBABILIDADES					
TRAMO	CASILLERO 1	CASILLERO 2	CASILLERO 3	CASILLERO 4	CASILLERO 5
1	0,133	0,200	0,067	0,200	0,400
2	0,167	0,333	0,222	0,000	0,278
3	0,409	0,273	0,045	0,091	0,182
4	0,471	0,000	0,176	0,118	0,235
5	0,435	0,087	0,130	0,174	0,174
6	0,435	0,130	0,348	0,000	0,087
7	0,353	0,118	0,235	0,235	0,059
8	0,125	0,125	0,438	0,125	0,188
9	0,176	0,235	0,176	0,059	0,353
10	0,318	0,136	0,273	0,182	0,091
11	0,300	0,050	0,200	0,300	0,150
12	0,250	0,000	0,250	0,250	0,250

Tabla 20: Estimación de probabilidades

Los errores de estimación se calculan con la **Ecuación 13** y se obtiene la siguiente tabla.

ERRORES DE ESTIMACIÓN					
TRAMO	CASILLERO 1	CASILLERO 2	CASILLERO 3	CASILLERO 4	CASILLERO 5
1	0,144	0,170	0,106	0,170	0,208
2	0,144	0,183	0,161	0,000	0,174
3	0,172	0,156	0,073	0,101	0,135
4	0,199	0,000	0,152	0,129	0,169

5	0,170	0,097	0,116	0,130	0,130
6	0,170	0,116	0,163	0,000	0,097
7	0,191	0,129	0,169	0,169	0,094
8	0,136	0,136	0,204	0,136	0,161
9	0,152	0,169	0,152	0,094	0,191
10	0,163	0,120	0,156	0,135	0,101
11	0,169	0,080	0,147	0,169	0,131
12	0,252	0,000	0,252	0,252	0,252

Tabla 21: Errores de estimación

El intervalo de confianza se construye a partir de las estimaciones de probabilidades que se muestran en la anterior tabla y de un nivel de confianza, dependiendo del nivel de confianza el intervalo es mayor o menor. Cuanto mayor sea el nivel de confianza mayor es el intervalo.

El nivel de confianza utilizado en este estudio es de 0.9 lo cual da un valor a Z de 1,65.

Nivel de confianza	Z
0,8	1,282
0,9	1,645
0,95	1,960
0,99	2,576

Tabla 22: Nivel de confianza

Los intervalos de confianza se construyen de acuerdo con la **Ecuación 12** los cuales quedan definidos en la siguiente tabla.

INTERVALOS DE CONFIANZA						
TRAMO	CASILLERO 1		CASILLERO 2		CASILLERO 3	
1	-0,011	0,278	0,030	0,370	-0,039	0,173
2	0,022	0,311	0,151	0,516	0,061	0,383

3	0,237	0,582	0,117	0,429	-0,028	0,119
4	0,271	0,670	0,000	0,000	0,024	0,329
5	0,265	0,605	-0,010	0,184	0,015	0,246
6	0,265	0,605	0,015	0,246	0,184	0,511
7	0,162	0,544	-0,011	0,246	0,066	0,405
8	-0,011	0,261	-0,011	0,261	0,234	0,641
9	0,024	0,329	0,066	0,405	0,024	0,329
10	0,155	0,482	0,016	0,257	0,117	0,429
11	0,132	0,469	-0,030	0,130	0,053	0,347
12	-0,002	0,502	0,000	0,000	-0,002	0,502

INTERVALOS DE CONFIANZA				
TRAMO	CASILLERO 4		CASILLERO 5	
1	0,030	0,370	0,192	0,608
2	0,000	0,000	0,104	0,451
3	-0,010	0,192	-0,030	0,317
4	-0,011	0,246	0,066	0,405
5	0,044	0,304	0,044	0,304
6	0,000	0,000	-0,010	0,184
7	0,066	0,405	-0,035	0,153
8	-0,011	0,261	0,027	0,348
9	-0,035	0,153	0,162	0,544
10	0,047	0,317	-0,010	0,192
11	0,131	0,469	0,019	0,281
12	-0,002	0,502	-0,002	0,502

Tabla 23: Intervalos de confianza.

De esta forma podemos conocer la probabilidad que hay de que una bola caiga en un determinado casillero conociendo desde donde es lanzada.

7. Conclusiones y líneas futuras

En este Trabajo Final de Grado se ha realizado el modelado, diseño, simulación y fabricación de un entorno didáctico robotizado basado en el juego "Plinko", para su utilización como material para una práctica docente. Se ha empleado un robot ABB IRB120, disponible en el taller de robótica de la Escuela de Ingenierías industriales de la Universidad de Valladolid.

Inicialmente se realizó el diseño y modelado del espacio de trabajo utilizando el software Robotstudio. Mediante esta simulación se pudo determinar la mejor ubicación del robot para poder acceder a todas las posiciones que requería el juego, y las trayectorias más adecuadas del robot.

Tras esta simulación se construyeron las maquetas para la práctica docente. Las maquetas, junto con el programa del robot desarrollado con RobotStudio, permitieron comprobar la capacidad real del robot IRB120 para realizar las tareas programadas. Fue necesario realizar ciertas modificaciones en las maquetas con el fin de corregir algunas deficiencias que en la simulación con Robotstudio no eran tan evidentes.

Posteriormente se añadió a las maquetas los sensores y actuadores necesarios, que se cablearon con el robot ABB utilizando la tarjeta de E/S digitales del robot.

Una vez montado el escenario real completo, se desarrolló y programó la electrónica necesaria para poder comunicar el robot, a través de protocolo de comunicación TCP-IP, con una página web desde la cual se pueden controlar los distintos modos de funcionamiento de la práctica docente, ofreciendo al usuario una interfaz sencilla y de fácil manejo. La electrónica desarrollada también permite mostrar en unos "displays" el recuento de bolas caídas en cada casillero.

Por último, se realizó un estudio estadístico teniendo en cuenta el lugar desde el que se había soltado la bola y el casillero en el que había finalizado.

Entre las líneas futuras de desarrollo poder indicar:

- Sustitución de los finales de carrera por sensores electrónicos conformados por diodos y fototransistores.
- Mejora de diseño del circuito electrónico con el cual se determina la disponibilidad de bola en el plano inclinado.
- Estudio del uso del método POST para obtener la información recibida de la página web por el ESP32.

8. Bibliografía

[1] Pagina web: https://www.infoplc.net/actualidad-industrial/item/102921-mercado-robots-soldadura.

[2] Pagina web: https://www.infoplc.net/actualidad-industrial/item/102471-robotica-industrial-china-supera-eeuu-eu Acceso realizado marzo 2022.

[3] Video: https://www.youtube.com/watch?v=U5IjvFBxfdY. Accesible desde el 17 de febrero de 2017.

[4] Video: https://www.youtube.com/watch?v=v3BX0ZOCWgI. Accesible desde el 27 de febrero de 2015.

[5] Rodriguez Lopez, Miguel. Modelado, simulación y desarrollo de una práctica docente con el robot ABB IRB 120 (uva.es). Trabajo fin de grado. Escuela de ingenierías industriales de Valladolid. Junio 2021.

[6] Página web: https://i5.walmartimages.com/asr/da8c936d-7b0f-457e-8d47-3ab94af63ced_1.7000d92ff97ef0ce2d12ab80d089c30e.jpeg?odnHeight=612&odnWidth=612&odnBg=FFFFFF Acceso realizado marzo 2022.

[7] Pagina web: https://www.etsy.com/es/listing/267992640/plinko-game-board-dxf-files-plans?ref=landingpage_similar_listing_top-4.. Acceso realizado marzo 2022

[8] Skretting, Karl. Departament of Electrical Engineering and Computer Science. ELE610 Robot Technology, spring 2022. https://www.ux.uis.no/~karlsk/ELE610/rs1.pdf. Acceso realizado marzo 2022.

[9] Página web: https://hetpro-store.com/TUTORIALES/i2c/ Acceso realizado marzo 2022.

[10] Página web: https://www.electrontools.com/Home/WP/valores-comerciales-de-resistencias/. Acceso realizado abril 2022.

[11] Hoja de características: https://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf. Acceso realizado abril 2022.

[12] Hoja de características: 5621AS.pdf (xlitx.com) . Acceso realizado abril 2022.

[13] Ramos Flores, Cristina. Análisis de un convertidor DC/DC destinado al almacenamiento híbrido de energía. https://oa.upm.es/48060/1/TFG_CRISTINA_RAMOS_FLORES.pdf. Trabajo fin de grado. Universidad Politécnica de Madrid. Julio 2017.

[14] Página web: https://eming.cl/2019/05/11/convertidor-buck-reductor-de-tension-con-dsp-tms320f28335-trabajo-en-contexto-academico/. Acceso realizado abril 2022.

[15] Página web: https://www.vishay.com/docs/83751/cny70.pdf. Acceso realizado abril 2022.

[16] Página web: http://r-luis.xbot.es/puerto/port01.html Acceso realizado mayo 2022.

[17] Página web: https://hetpro-store.com/TUTORIALES/i2c/ Acceso realizado mayo 2022.

[18] Página web: http://www.sinaptec.alomar.com.ar/2018/09/esp32-desde-cero-tutorial-6-servidor.html. Acceso realizado mayo 2022.

[19] Página web: https://www.monografias.com/trabajos104/comunicacion-serial-conceptos-generales/comunicacion-serial-conceptos-generales. Acceso realizado mayo 2022.

[20] Página web: http://diccionario.sensagent.com/8N1/es-es/.Acceso realizado mayo 2022.

[21] Página web: https://upanama.e-ducativa.com/archivos/repositorio/6000/6126/html/51_proto.htm.Acceso realizado mayo 2022.

[22] Página web: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Acceso realizado mayo 2022.

[23] Página web: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf. Acceso realizado mayo 2022.

[24] Página web: https://smc-static-resources-prd.s3.eu-central-1.amazonaws.com/products/datasheet/gen/DS_CD85N10-40S-B_es_ES.pdf. Acceso realizado mayo 2022.

[25] Página web: https://smc-static-resources-prd.s3.eu-central-1.amazonaws.com/products/datasheet/gen/DS_VT307-5DZ1-02F-Q_es.pdf. Acceso realizado mayo 2022.

[26] Página web: https://programarfacil.com/esp8266/esp32/. Acceso realizado mayo 2022.

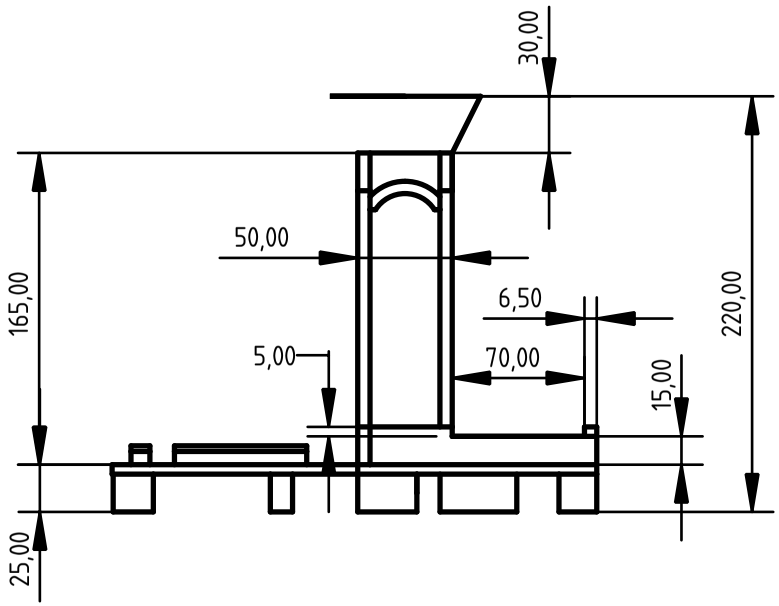
[27] Página web: https://programarfacil.com/blog/arduino-blog/matriz-led-arduino-max7219/. Acceso realizado mayo 2022.

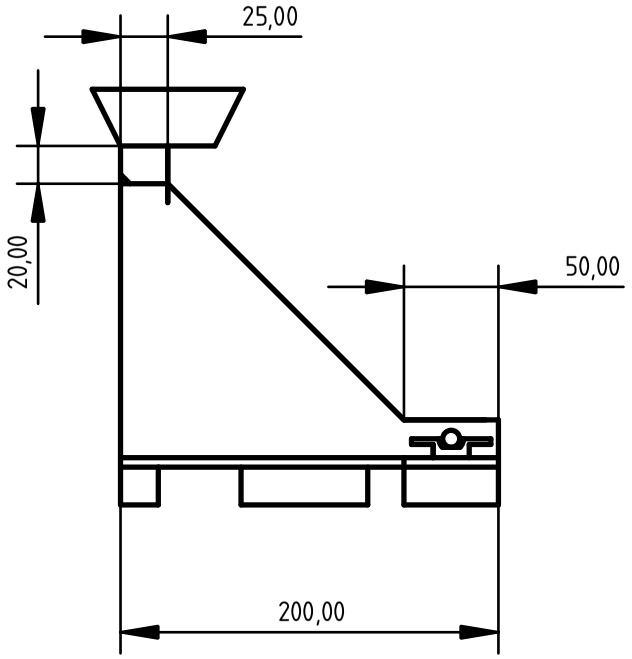
[28] Página web: http://www.coffeebrain.org/wiki/index.php?title=Max7219. Acceso realizado mayo 2022.

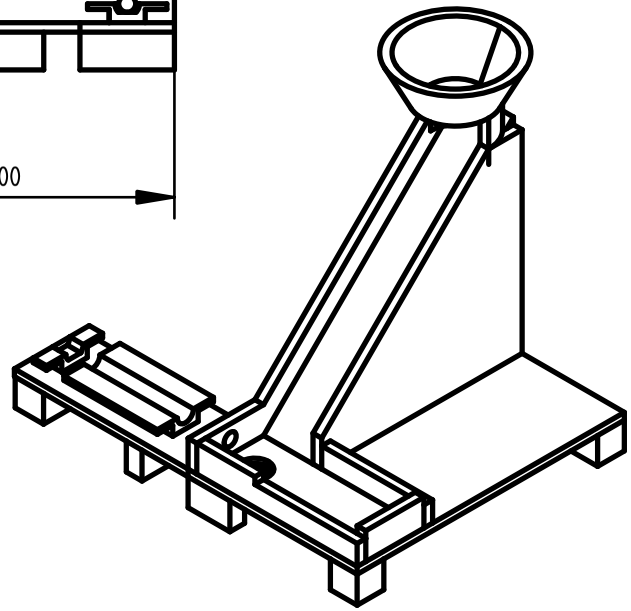
[29] Video: https://www.youtube.com/watch?v=slNPfrQaVlo. Acceso realizado mayo 2022.

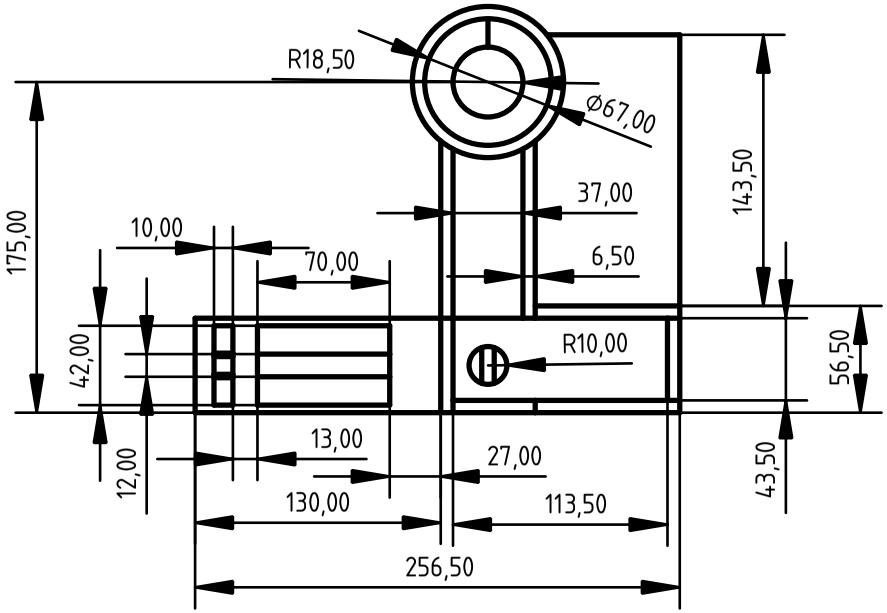
[30] Video: https://www.ea6sb.es/como-construir-una-pcb/. Acceso realizado mayo 2022.

9. Anexos

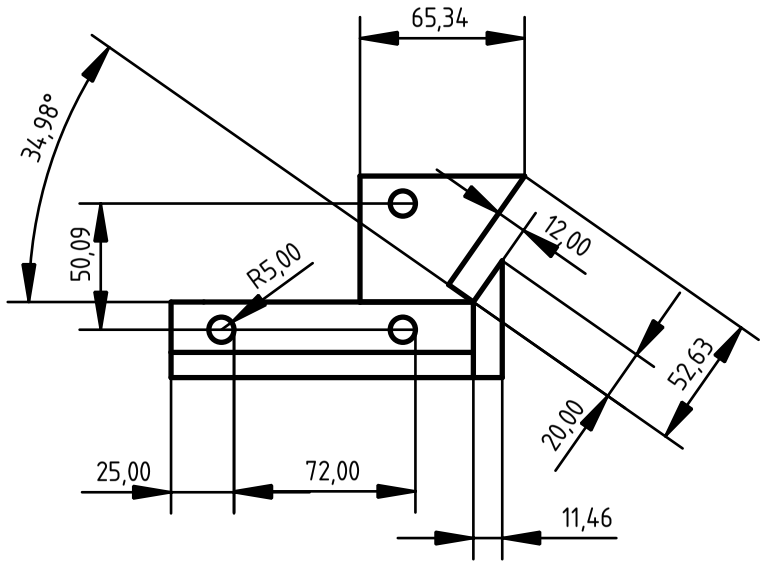


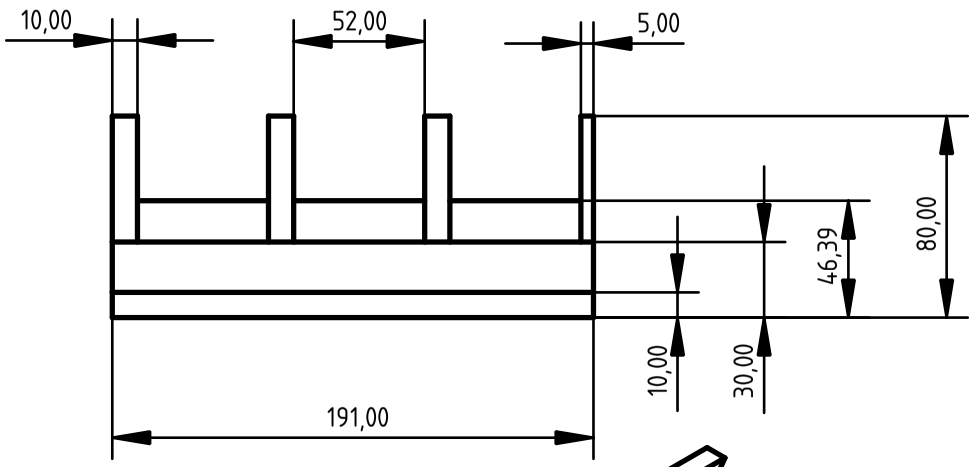


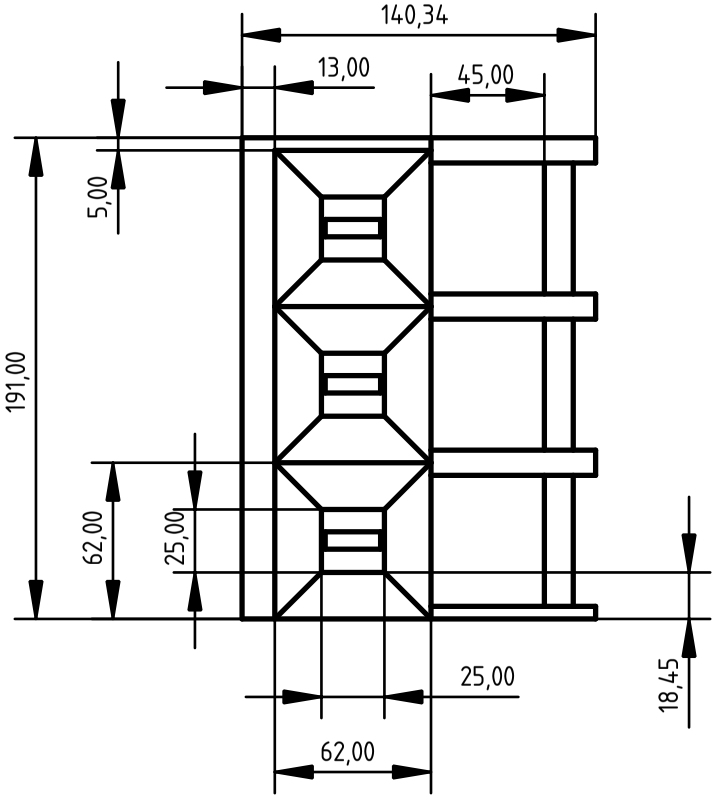


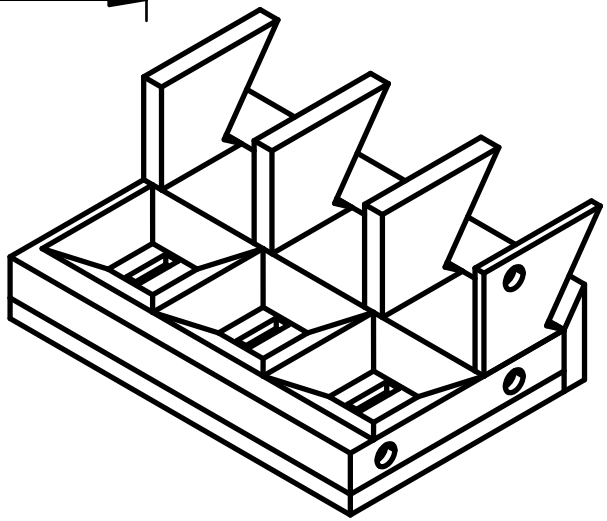


DESIGNED BY: Javier Diez DATE: Octubre 2021		Primer diseño Plano inclinado	
SIZE A4			
SCALE 1:4	WEIGHT (kg)	DRAWING NUMBER 1	SHEET 1

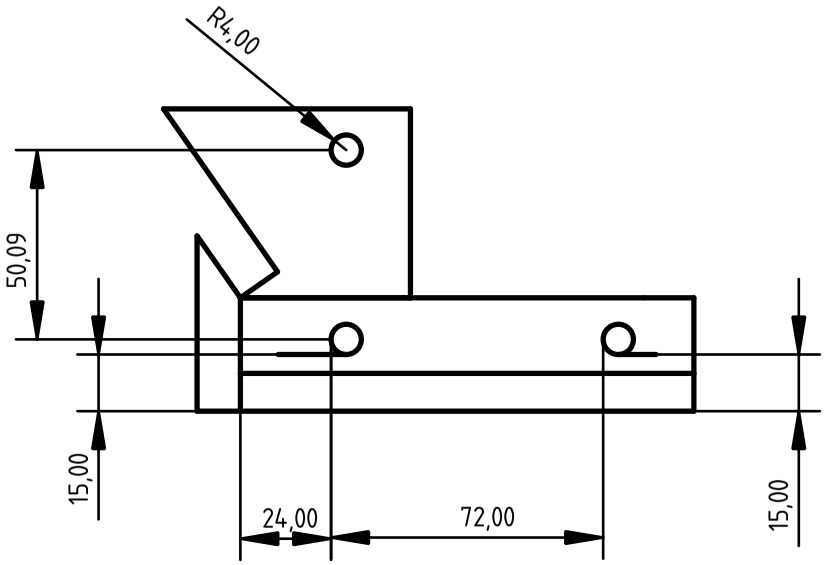


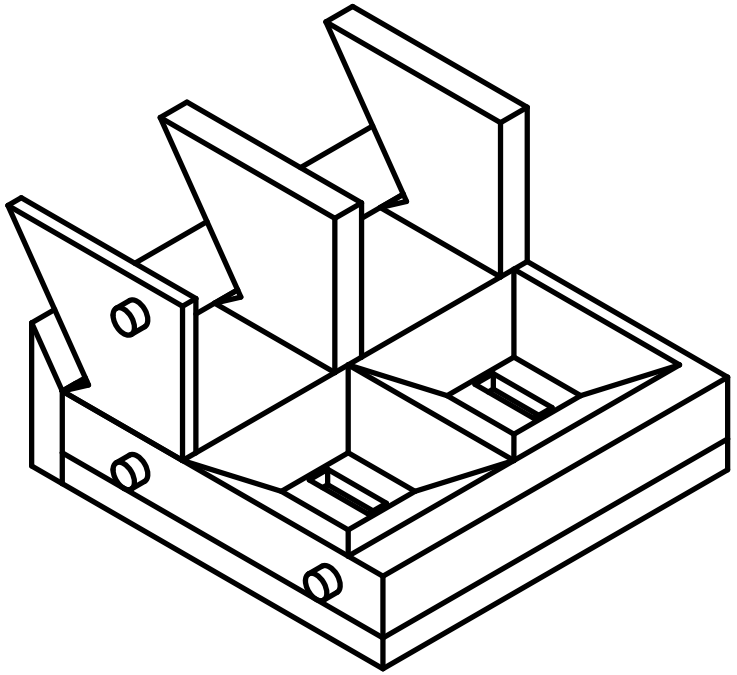


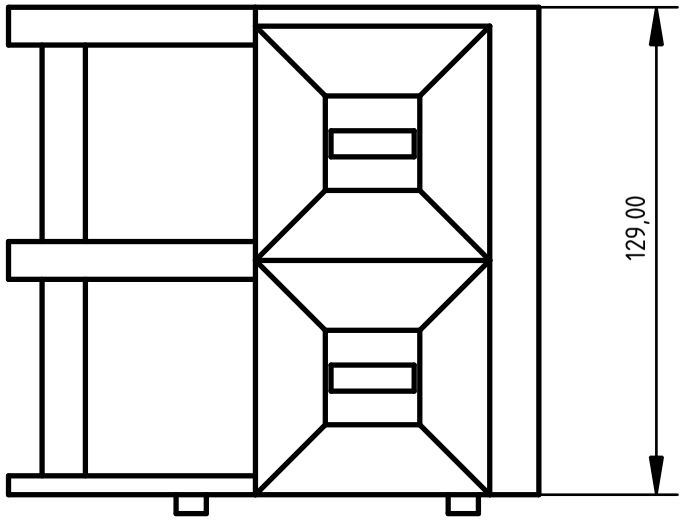




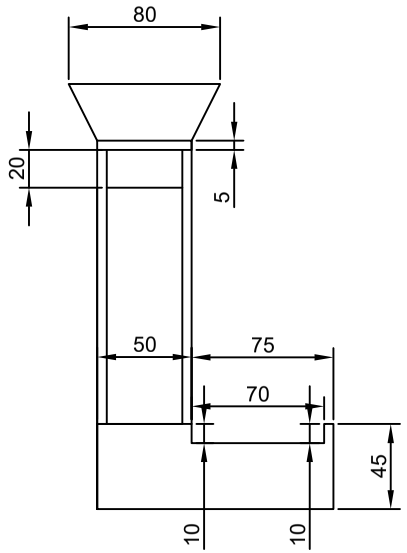
DESIGNED BY: Javier Diez DATE: Octubre 2021		Pieza 1 Casillero Plinko	
SIZE A4			
SCALE 1:3	WEIGHT (kg)	DRAWING NUMBER 2	SHEET 2

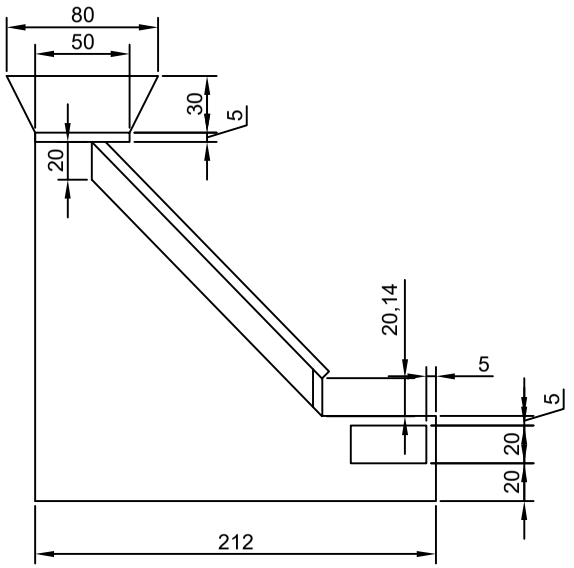


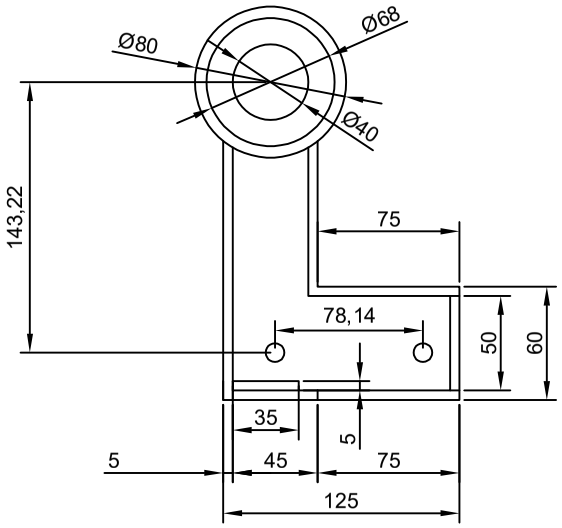




DESIGNED BY: Javier Diez DATE: Octubre 2021		Pieza 2 Casillero Plinko	
SIZE A4			
SCALE 1:2	WEIGHT (kg)	DRAWING NUMBER 3	SHEET 3







UNIVERSIDAD DE VALLADOLID
ESCUELA DE INGENIERÍAS INDUSTRIALES

TITULO PROYECTO:

TRABAJO DE FIN DE GRADO JUEGO DEL PLINKO

PLANO:

PLANO INCLINADO FINAL

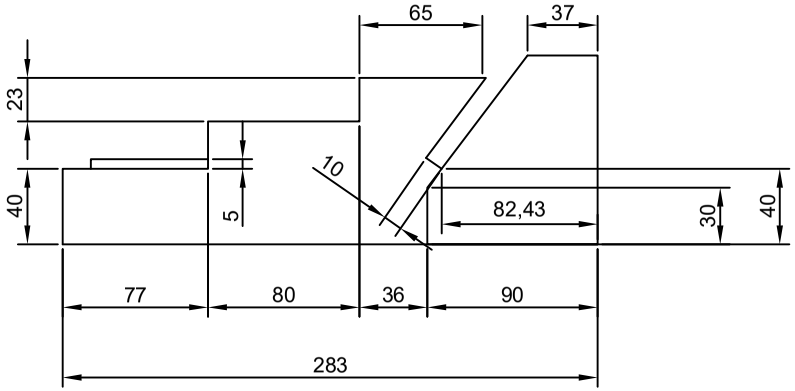
FECHA: Mayo-2022

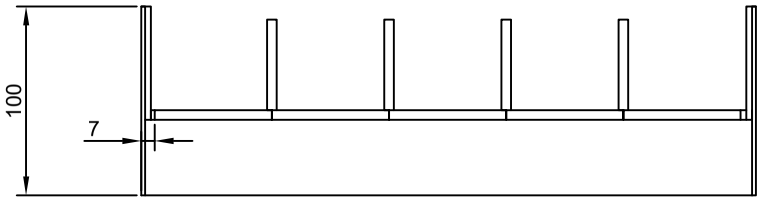
N° PLANO: 4

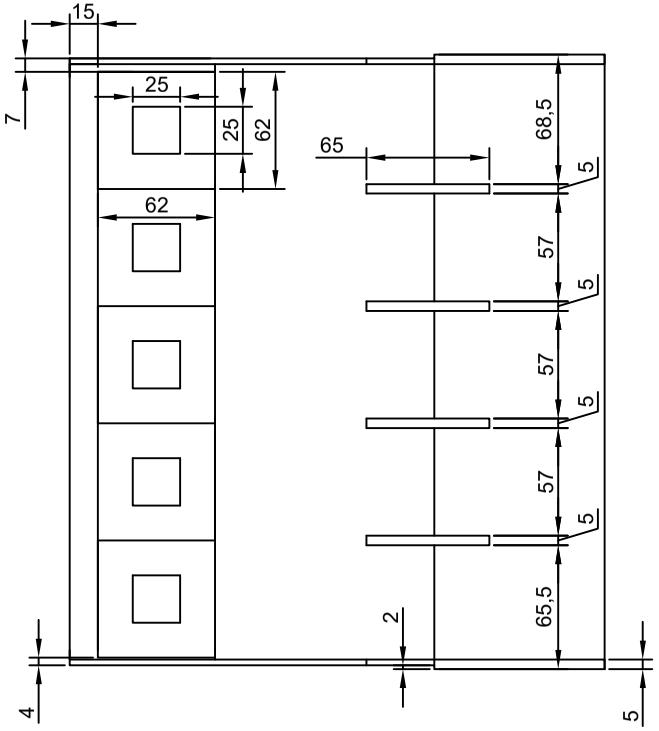
ESCALA: 1:4

FIRMA:
EL/LOS ALUMNO/S:

Grado en Ingeniería electrónica industrial y automática
Fdo: Francisco Javier Diez Rey







UNIVERSIDAD DE VALLADOLID
ESCUELA DE INGENIERÍAS INDUSTRIALES

TITULO PROYECTO:

TRABAJO DE FIN DE GRADO JUEGO DEL PLINKO

PLANO:

CASILLERO PLINKO

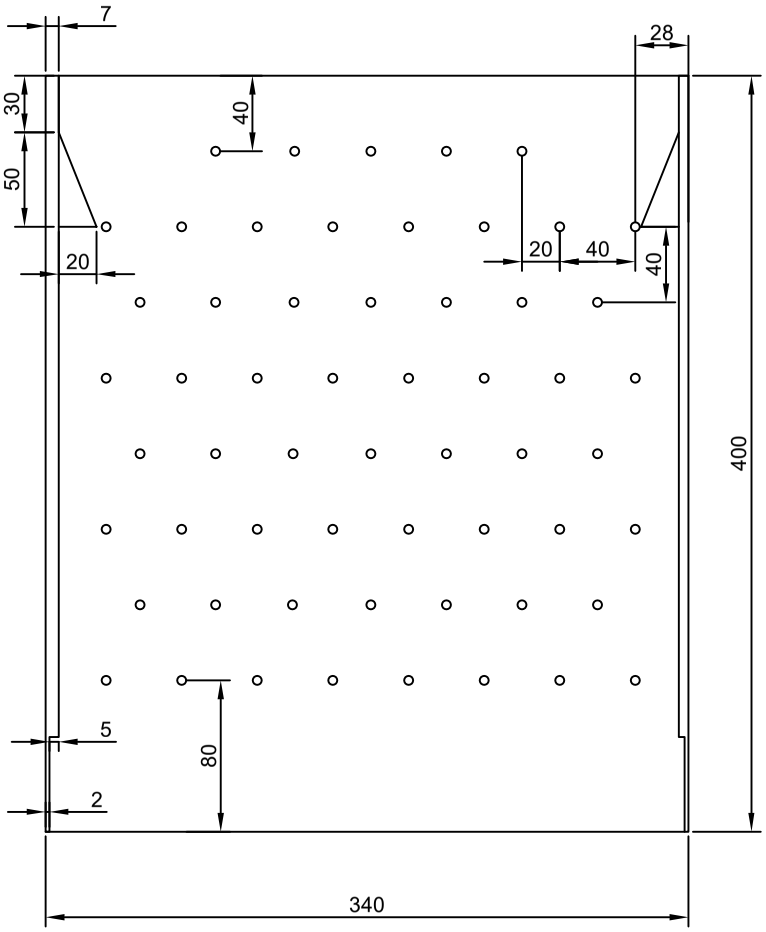
FECHA: Mayo-2022

N° PLANO: 5

ESCALA: 1:4

FIRMA:
EL/LOS ALUMNO/S:

Grado en Ingeniería electrónica industrial y automática
Fdo: Francisco Javier Diez Rey



UNIVERSIDAD DE VALLADOLID
ESCUELA DE INGENIERÍAS INDUSTRIALES

TITULO PROYECTO:

TRABAJO DE FIN DE GRADO JUEGO DEL PLINKO

PLANO:

TABLERO PLINKO

FECHA: Mayo-2022

N° PLANO: 6

ESCALA: 1:4

FIRMA:
EL/LOS ALUMNO/S:

Grado en Ingeniería electrónica industrial y automática
Fdo: Francisco Javier Diez Rey