UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Alcoy

Robot cuidador de plantas

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Bazaga Buades, Patricia

Tutor/a: Cuesta Frau, David

CURSO ACADÉMICO: 2023/2024

Resumen

Este trabajo de final de grado trata sobre la construcción de un robot autónomo capaz de ocuparse de los cuidados de una planta. Para ello, mediante la información que recibe de los sensores, será capaz de hallar la mejor solución a los diferentes problemas que conlleva cuidar una planta. Estos problemas son falta o exceso de luz solar y falta de agua, aunque estos problemas dependerán de las características de la planta. Para indicar el tipo de planta a cuidar y sus características se diseñará una aplicación móvil que configure el robot.

Palabras clave: Robot; Arduino; planta.

Abstract

This final degree project is about the construction of an autonomous robot capable of taking care of a plant. Through the information it receives from the sensors, it will be able to find the best solution to the different problems that come with caring for a plant. These problems are lack or excess of sunlight and lack of water, although these problems will depend on the haracteristics of the plant. To indicate the type of plant to care for and its characteristics a mobile application will be designed to configure the robot.

***Keywords**: Robot; Arduino; plant.*

Índice general

Índice de figuras

Índice de tablas

1 Introducción

Un sector de la población quiere tener plantas en sus hogares porque les agrada, son su pasatiempo, las utilizan en recetas culinarias o puramente como decoración. No obstante, entre ese sector hay personas que no tienen los conocimientos para hacerse cargo de ellas debido a que requieren una serie de cuidados. Entre los cuidados se encuentran el riego, la poda, el abono, el control de luz y temperatura. Estos últimos, el riego y el control de luz y temperatura, son los más críticos ya que por falta de conocimientos o descuido pueden llegar a causar la muerte.

En este proyecto nos centraremos en el riego y control de luz. En el caso del riego, el exceso de humedad de la tierra pudrirá la planta desde las raíces. Por el contrario, la falta de agua impedirá que tenga suficiente alimento y, por consecuencia, se seque. En el caso del control de luz, si se trata de luz solar el exceso puede quemar las hojas, mientras que su escasez, independientemente de origen, puede llegar a hacer que no sobreviva.

1.1 Motivación

La principal motivación de este proyecto es dar una solución al problema del cuidado de plantas. Aunando por un lado un sistema de riego automático con un sistema de movimiento para un correcto posicionamiento. Actualmente en el mercado no hay ningún robot con su misma funcionalidad. Existen maceteros inteligentes o sistemas de riego automáticos, pero no tienen en cuenta la luz o no son capaces de moverse por sí solos.

Por otro lado, este proyecto está concebido con el objetivo de tener un coste reducido y accesible. Para que todas las personas que lo deseen con unos conocimientos básicos de electrónica sean capaces de llegar a replicarlo.

1.2 Objetivos

El objetivo principal del proyecto es eliminar las dificultades que presenta el exceso o escasez de sol y riego, ayudando así a que los usuarios sean capaces de mantener una planta. Para ello diseñaremos un software para la plataforma de Arduino que detecte el estado de la planta y calcule las condiciones óptimas. A su vez, se diseñará un chasis donde colocar la maceta y nuestro Arduino con sus sensores.

La parte exterior del chasis consistirá en una zona donde se podrá colocar la maceta, una pantalla donde se muestre la información y las acomodaciones necesarias para los sensores, mientras que por dentro estarán todos los componentes electrónicos y el depósito de agua.

Adicionalmente, para la configuración del Arduino con los parámetros específicos de la planta se creará una pequeña aplicación móvil que se comunicará mediante Bluetooth.

2 Estado del arte

En la actualidad las soluciones que tenemos al alcance se pueden dividir en dos grandes grupos, soluciones comerciales y prototipos. Estos últimos normalmente son creados por personas individuales y a cuenta propia o de carácter académico.

2.1 Soluciones comerciales

2.1.1 Sybil

Sybil[20] es un robot inteligente que se alimenta mediante energía solar. Diseñado y fabricado por la empresa Sterling salió al mercado en 2021. Los únicos parámetros que necesita son las esquinas y el centro del área que quieres que se encargue. Con sus dos cámaras frontales diferencia entre malas hierbas y plantas haciendo uso de una red neuronal. En el caso de que se encuentre con una mala hierba está preparado para poder encargarse de ella mediante unas cuchillas en su parte trasera. En la siguiente Figura 2.1 se muestra el robot Sybil en el campo.

Figura 2.1: Sybil.

Por otro lado, se le pueden introducir semillas para que en aquellos lugares donde no detecte vegetación vaya plantando las semillas. Cabe destacar que al ser únicamente alimentado por energía solar no necesita de ningún aparato externo para su carga y por lo tanto podrá estar las 24 horas del día, gracias a que por la mañana irá cargando las

baterías y por la noche las gastará. A su vez, si se encuentra en una zona donde hay pocas horas de luz es posible que no sea capaz de trabajar las 24 horas.

2.1.2 Flower Power

Flower Power[17] es un dispositivo creado por Parrot. Se encarga de hacer lecturas a la tierra de la maceta para enviar alertas a la aplicación del móvil con el fin de avisarnos de que necesita riego la planta. Adicionalmente nos avisará de la luz y la humedad, pues cuenta con un sensor de luz que registrará la cantidad de luz diaria que recibe y un sensor que leerá la humedad a la que se encuentra el suelo. En la siguiente Figura 2.2 se muestra el dispositivo en una maceta.

Figura 2.2: Flower power en una maceta.

En cuanto al diseño intenta imitar un tronco o una rama partida, haciendo que no destaque, pero sabiendo en todo momento donde está por los colores no característicos que tiene. Los sensores se encuentran ocultos pues la mayoría de ellos están en la parte inferior, la cual está enterrada en la tierra. En el caso del sensor de luz, aunque se encuentra en la parte más alta del dispositivo pasa desapercibido pues parece como si fuera parte del tronco. Por último, la zona de la batería se encuentra donde se puede apreciar que el tronco no es continuo.

La manera que tiene de saber cuándo dar el aviso es mediante una base de datos con la que cuenta la aplicación donde seleccionaremos el tipo de plantas que tenemos en la maceta. El intercambio de información entre el dispositivo y la aplicación móvil se hará mediante Bluetooth, por lo que tendremos que estar a una distancia cercana si deseamos recibir información a diario de ella. Si por el contrario nos olvidamos de pasar no habrá ninguna alerta que nos informe de la falta de interacción entre el dispositivo y la aplicación.

2.1.3 Yoolax

Yoolax[27] es una maceta inteligente que fue creada por la empresa del mismo nombre. Su finalidad es monitorizar la humedad, luz y temperatura que recibe la planta y avisar cuando no sean las correctas.

Figura 2.3: Yoolax.

Las principales características de Yoolax son la pantalla LCD táctil, con la cual se interactúa con el robot. El diseño está enfocado a ser una mascota que se comportará en relación al estado de la planta, haciendo gestos o ruidos. Además, posee conexión a internet para poder recibir el estado de la planta en tiempo real. En cuanto al diseño intenta parecerse a una maceta, y los sensores para la toma de medidas se encuentran dentro del cuerpo del robot a excepción del sensor de luz y temperatura. Estos se encuentran en la parte trasera del macetero inteligente para no interferir con la forma habitual de cogerlo.

2.1.4 HEXA

HEXA[24] es un robot araña multiusos creado por la compañía Vincross. Uno de sus experimentos consistía en cuidar una planta, modificando el cuerpo estándar que tiene los HEXA para poder albergar en su cuerpo una maceta.

Figura 2.4: Robot HEXA llevando una maceta. [8]

La característica que define un robot HEXA son las seis patas en forma de araña que posee, de ahí su nombre. Tiene una cámara de 720p, sensores infrarrojos, acelerómetros y

puertos USB y de audio. En cuanto a la parte de software, tiene una aplicación móvil con la que puedes controlar el robot y su movimiento de diferentes maneras. Adicionalmente, también tiene acceso para dos programas extra, uno donde puedes crear código que quieres que ejecute el robot, y otro donde puedes simular de forma 3D las acciones del robot.

Su principal función es ir a buscar la luz para la planta e ir rotando poco a poco para que reciba por todos los sitios la luz. No puede enviar notificaciones para avisar que le falta agua a la maceta, por lo que se pone a bailar para avisar de ello. A su vez, está programado para interactuar con las personas reaccionado al ser tocado o a los gestos que le hacen, como si fuera una mascota.

2.2 Prototipos

2.2.1 Elowan [26]

Elowan es un híbrido robot-planta pues quien controla el robot es la planta, ha sido creado por científicos del Instituto de Tecnología de Massachusetts en Estados Unidos. Su función principal es llevar a la planta a un espacio con luz, basándose en las señales bio-electroquímicas de la planta para saber en qué dirección se encuentra la luz.

Figura 2.5: Robot Elowan llevando una maceta. [9]

Su funcionamiento se basa en una serie de electrodos repartidos por las zonas de la planta: tallo, hojas y raíces. Cada zona tiene unos parámetros a medir: en las hojas se mide la respiración de la planta; en el tallo, el crecimiento de la planta; y en las raíces, la absorción de humedad de la planta, en otras palabras, la variación de la humedad en la tierra. Posteriormente a la lectura de las señales son aumentadas y enviadas a la placa para ser interpretadas y mover la planta al mejor sitio posible.

Su diseño no es de un producto para venderlo cara al público como en los anteriores robots. Es un prototipo que quiere mostrar las utilidades del aumento de datos relacionados con la naturaleza, priorizando la funcionalidad a la estética.

2.2.2 Prototipo 1

Es un proyecto que hicieron varios alumnos del máster de mecatrónica de la universidad de Bruface. Su función principal es regar y llevar la planta a un sitio con luz.

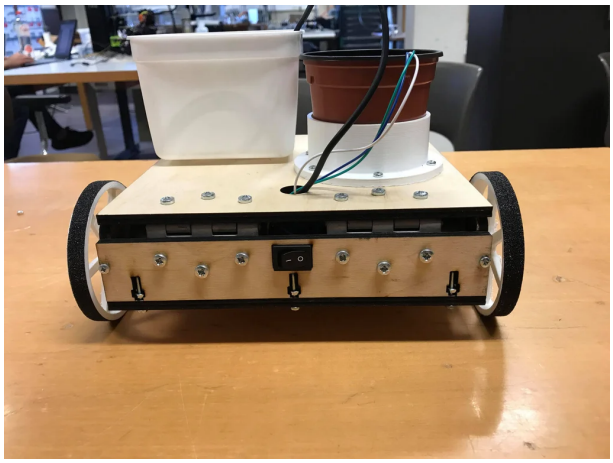


Figura 2.6: Protitopo 1 [14]

Su funcionamiento se basa en comprobar la humedad de la tierra para regarla en el caso de que sea poca y posicionar la planta en el lugar donde el nivel lumínico sea apropiado, evitando obstáculos a su paso. No hay manera de comunicarse con el robot.

2.2.3 Prototipo 2

Es un proyecto que hicieron varios alumnos de la Universidad Libre de Bruselas. Su función principal es regar y darle luz a la planta homogéneamente mediante las tiras led.

Figura 2.7: Prototipo 2 [18]

Su funcionamiento se basa en que el sensor de humedad de la tierra detecta cuando es necesario regar. Por otro lado, los motores rotan la planta sobre su eje para que reciba la luz homogéneamente.

Por último, vamos a resumir los parámetros más importantes de cada robot en una tabla comparativa. En está tabla solo se tendrán en cuenta las características que tienen relación con el proyecto.

Parámetros	Sybil	Flower Power	HEXA	Yoolax	Elowan	Prototipo 1	Prototipo 2
Medición de la luz		X	X	X	X	X	X
Sistema de riego				X	X	X	
Control humedad		X		X		X	X
Movimiento	X		X		X	X	X[2]
Comunicación		Bluetooth	Baile	Internet			LCD + botones
Transporte	X[1]		X		X	X	

1 Transporta semillas.
2 Gira sobre su eje para mover la planta para que reciba la luz homogéneamente.

Tabla 2.1: Comparativa de las características de los robots

Como se puede observar, si comparamos las diferencias entre prototipos y productos comerciales, estos últimos no están capacitados para cuidar una planta autónomamente. La mayoría han sido diseñados con el concepto de mascota en mente y por ello recaen ciertas tareas en el usuario.

3 Fundamentos

En este apartado vamos explicar el funcionamiento de ciertas funciones del Arduino, así como la comunicación entre Arduino y móvil. Primero explicaremos las interrupciones en la sección 3.1; los modos de suspensión del Arduino en la sección 3.2; y, por último, la comunicación Bluetooth en la sección 3.3.

3.1 Interrupciones en Arduino

Para diseñar nuestro programa de Arduino tendremos dos bloques diferenciados: el *setup* y el *loop*. El *setup* se ejecutará una vez al encender el Arduino, para inicializar variables necesarias, los *pinModes*, comunicaciones y librerías. El *loop*, como su propio nombre indica, será ejecutado en bucle hasta que el Arduino deje de ser alimentado.

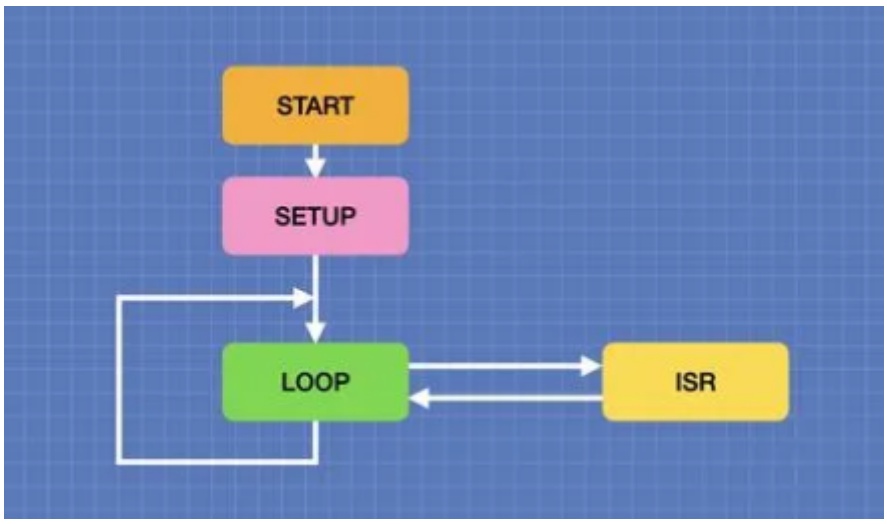


Figura 3.1: Flujo básico de un programa de Arduino [25]

Las interrupciones cortan el funcionamiento normal del *loop* en el momento que se detecta una. En ese caso, el *loop* se detiene temporalmente y se ejecuta una llamada al método que gestiona dicha interrupción, Figura 3.1. Para declarar una interrupción se utiliza el siguiente comando:

```
1 attachInterrupt(digitalPinToInterrupt(pin), ISR, mode);
```

El parámetro pin será el número del pin del Arduino, hay solo unos pines que pueden ser usados, Tabla 3.1. Todos los pines serán digitales, por ello, *pin* deberá estar dentro de *digitalPinToInterrupt( )*.

	Modelos Arduino					
	UNO	Nano	Mini Pro	Mega	Leonardo	Due
INT0	2	2	2	2	3	En todos los pines
INT1	3	3	3	3	2	
INT2	-	-	-	21	0	
INT3	-	-	-	20	1	
INT4	-	-	-	19	7	
INT5	-	-	-	18	-	

Tabla 3.1: Pines susceptibles de generar interrupciones [3]

ISR es el método que se ejecutará cuando se detecte la interrupción. El método que se llama no deberá de devolver ningún valor y tampoco utilizarse ningún tipo de *delay()* o función similar , así como cálculos que requieran la suspensión del programa principal. Mientras esté ejecutándose la interrupción todo el *loop* estará paralizado y, por lo tanto, estaremos alterando el flujo del programa.

Por último, *mode* define el motivo por el que ha de activarse la interrupción. Hay un total de 6 modos, pero para el Arduino Mega no todos los pines se pueden utilizar con estos modos [15].

Pines	Modos de interrupciones				
	LOW	CHANGE	RISING	FALLING	HIGH
2	Si	-	-	-	-
3	Si	-	-	-	-
18	Acepta todos				
19	Acepta todos				
20	Acepta todos				
21	Acepta todos				

Tabla 3.2: Modos de interrupciones en Arduino Mega [15]

LOW : se activará cuando el valor sea 0.

CHANGE : se activará cuando cambie el valor.

RISING : se activará cuando haya un flanco de subida

FALLING : se activará en el flanco de bajada.

HIGH se activará cuando el valor sea 1.

3.2 Modos de suspensión del Arduino [15]

Los modos de suspensión se utilizan como recurso para ahorrar batería en las aplicaciones que no necesitan estar siempre activas. Existen un total de seis modos de suspensión: *Idle*, *ADC Noise Reduction*, *Power-down*, *Power-save*, *Standby* y *Extended Standby*. Siendo *idle* el modo en el que menos componentes del Arduino están desactivados y *Power-down* siendo el modo en el que más componentes lo están.

Independientemente del modo a utilizar primero ha de ponerse el bit SE del *Sleep Mode Control Register* (SMCR) a uno. En el caso de que no esté a uno cuando se le mande la instrucción de domir no la ejecutará. Para evitar que el Arduino entre en modo suspensión cuando no debe recomiendan que se ponga a uno el bit de SE de manera previa a mandar la instrucción de dormir. De la misma manera, cuando el Arduino salga del modo suspensión, lo primero que se hará será cambiar el bit SE a cero para evitar problemas.

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	–	–	–	–	SM2	SM1	SM0	SE	SMCR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 3.2: SMCR – Sleep Mode Control Register

Posteriormente, para elegir el modo de suspensión, modificaremos los bits SM0, SM1 y SM2. Estos bits nos darán un total de 8 combinaciones pero tendremos dos de ellas reservadas, pues solo hay seis modos de suspensión como podemos ver en la figura 3.3.

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby(1)
1	1	1	Extended Standby(1)

Note: 1. Standby modes are only recommended for use with external crystals or resonators.

Figura 3.3: Combinación de bits para activar los modos de suspensión del Arduino

Antes de entrar a explicar los diferentes modos primero explicaremos algunos de los módulos a los que se hará referencia. Estos módulos serán desactivados para ahorrar batería y, en el caso que se necesiten, se activarán para que el Arduino despierte de su modo de suspensión.

Analog to Digital Converter (ADC)

En el caso de que se habilite estará disponible en todos los modos. Para ahorrar batería, si no se usa, ha de ser deshabilitado antes de entrar en cualquier modo.

Analog Comparator

Módulo que se encarga de comparar el voltaje entre pines. En la mayoría de modos está desactivado. No obstante, en modo *Idle* y *Analog Comparator* está activo pero para ahorrar energía debería desactivarse si no se necesita.

Brown-out Detector

Es el módulo que se encarga de si el voltaje que le llega a la placa es menor al configurado apagar el procesador. Si se activa estará activo en todos los modos consumiendo energía.

Internal Voltage Reference

Módulo que se encarga de asignar una referencia a los pines. Por defecto usa el mismo voltaje al que va el Arduino. Este módulo se activará a raíz de *Brown-out Detection*, el *Analog Comparator* o del ADC, y en el caso contrario estará desactivado. Si se activa hay que dejar un tiempo de recuperación entre su puesta en marcha y uso para su correcto funcionamiento.

Watchdog Timer (WDT)

Es un procedimiento para asegurar el correcto funcionamiento del Arduino. En el caso de que se detecte un problema reinicia el Arduino. Si el módulo está activo se puede usar en todos los modos.

Port Pins

Cuando se configure el modo de suspensión hay que asegurar que todos los pines usan la mínima energía. En los modos en los que el I/O clock (clkI/O) y el ADC clock (clkADC) están parados automáticamente se desactivarán, asegurando así el ahorro de energía.

Por otro lado, los pines analógicos deben de estar siempre desactivados. Si se requiere que los pines digitales estén también desactivados es cambiar los *Digital Input Disable Registers* (DIDR2, DIDR1 y DIDR0).

On-chip Debug System

Este módulo no se desactiva automáticamente al entrar en cualquier modo de suspensión, conllevando así un elevado consumo de energía en los modos de más bajo consumo. Para desactivar este módulo se puede hacer de tres maneras: desactivando el OCDEN Fuse, el JTAGEN Fuse, o poniendo a uno el bit de JTD en el MCUCR.

3.2.1 Idle

El modo *Idle* se activa cuando los bits del SM2 al SM0 están a cero. Este modo detiene la CPU, pero deja la mayoría de módulos del Arduino encendidos. El SPI, USART, *Analog Comparator*, ADC, la comunicación por I2C, los contadores, el *Watchdog*, el sistema de interrupciones y todos los relojes exceptuando clkCPU y clkFLASH son los que se quedarían activos.

Para despertar del modo *Idle* se puede hacer mediante interrupciones externas, *Timer Overflow* o comunicación por USART. En este modo si no se necesita el comparador analógico puede no alimentarse, pero en el caso de que se mantenga activo la comparación empezará cuando se active el modo, haciendo que aumente el consumo de energía.

3.2.2 ADC Noise Reduction

El modo *ADC Noise Reduction* se entra cuando los bits SM2 a SM0 están en 001 respectivamente. En este modo la CPU se detiene, pero siguen funcionando el ADC, las interrupciones externas, la comunicación por I2C, el Timer/Counter2 y el *Watchdog*. Además, los relojes a excepción de clkI/O, clkCPU, y clkFLASH siguen funcionando.

Para despertar de este modo se tiene que utilizar una interrupción externa, una interrupción o *reset* generado por el *Watchdog*, *Brown-out*, una interrupción por I2C, una interrupción del Timer/Counter2 o una interrupción por ADC, que en este modo tiene mejor resolución al reducir el ruido.

3.2.3 Power-down

El modo en el que menos energía se gasta, para activarlo se ha de poner 010 en los bits SM2 a SM0 respectivamente. En este modo el oscilador externo se detiene al igual que todos los relojes. Lo único que sigue funcionando son las interrupciones externas, la comunicación I2C y el *Watchdog*. Este último se podría deshabilitar para reducir el consumo de energía.

En el caso de que se vaya a despertar mediante una interrupción de nivel, el nivel del bit ha de mantenerse durante un tiempo. Además, siempre va a haber un retraso hasta que la activación es efectiva. Este retraso es para que la activación de los relojes sea correcta y estable después de haber sido parados.

3.2.4 Power-save

Cuando los bits del SM2 al SM0 se ponen en 011 respectivamente y posteriormente se manda la instrucción de dormir se activa este modo. Es idéntico a *Power-down*, exceptuando que el Timer/Counter 2 esté activo.

Timer/Counter 2 si está activo puede ser tanto síncrono como asíncrono dependiendo de la necesidad. En el caso de que sea asíncrono el reloj está parado, mientras que si es síncrono el oscilador estará parado pero el reloj solo podrá ser usado por Timer/Counter 2.

Las formas de salir de este modo son iguales que las de *Power-down* añadiendo que puede ser despertado mediante un *Timer Overflow* o mediante la comparación de la salida del Timer/Counter 2 en el caso de que los bit de control estén activados. En el caso de que no se vaya a utilizar el Timer/Counter 2 se recomienda utilizar *Power-down*.

3.2.5 Standby

Cuando los bits del SM2 a SM0 se ponen a 110 se activará el modo *Standby* al darle la orden de dormir. El modo es idéntico a *Power-down* exceptuando que el oscilador sigue en marcha. El dispositivo se pone en funcionamiento en seis ciclos de reloj.

3.2.6 Extended Standby

Cuando los bits del SM2 a SM0 se ponen a 111 al darle la instrucción de dormir se entrará en este modo, no obstante, se ha de utilizar un cristal o reloj externo. Este modo es igual que el de *Power-save*.

En la siguiente tabla se hará un resumen de los diferentes relojes, osciladores y las formas de salir del modo de suspensión para todos los modos anteriormente mencionados.

	Active Clock Domains					Oscillators		Wake-up Sources						
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Osc Enabled	INT7:0 and Pin Change	TWI Address Match	Timer2	SPM/ EEPROM Ready	ADC	WDT Interrupt	Other I/O
Idle			X	X	X	X	X(2)	X	X	X	X	X	X	X
ADCNRM				X	X	X	X(2)	X(3)	X	X(2)	X	X	X	
Power-down								X(3)	X				X	
Power-save					X		X(2)	X(3)	X	X			X	
Standby(1)						X		X(3)	X				X	
Extended Standby					X(2)	X	X(2)	X(3)	X	X			X	

Note: 1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT7:4, only level interrupt.

Figura 3.4: Resumen de los relojes, osciladores y formas de despertar al Arduino

3.3 Bluetooth en Arduino

Las placas de Arduino no vienen por defecto con la posibilidad de establecer conexiones Bluetooth, por ende, se necesita de un módulo separado para manejar este tipo de conexión.

Los módulos de Bluetooth que podemos encontrar se diferencian entre los que son solo esclavos y los que pueden ser esclavos o maestros. Los módulos que se comportan como esclavo esperan peticiones de conexión mientras que los módulos maestros son los que emiten peticiones de conexión.

En nuestra aplicación usaremos Bluetooth Low Energy, puesto que nos permite casi el mismo rango de alcance de comunicación pero con un rendimiento energético más eficiente.

4 Hardware

En este apartado veremos el diseño y los componentes del robot. Estará dividido en cuatro secciones: la primera será el diseño,4.1, donde explicaremos el razonamiento detrás del diseño y las características del mismo; la segunda contendrá los componentes del robot,4.2, donde explicaremos con detalle cada componente utilizado; la tercera será el presupuesto,4.3, tanto para los componentes eléctricos como para el chasis del robot; y, por último, el esquema eléctrico del robot,4.4.

4.1 Diseño

La sección estará dividida en dos subsecciones: la primera será las necesidades el diseño,4.1.1, donde hablaremos sobre las características que necesita cumplir el diseño; y la segunda será los diseños realizados,4.1.2.

4.1.1 Necesidades del diseño

En esta subsección hablaremos sobre las razones de la elección de los componentes y de las pautas para el diseño del chasis del robot.

Forma

Principal característica del chasis del robot. Las posibles formas son rectangular o circular.

Utilizando la forma rectangular se puede aprovechar mejor el espacio debido a que todos los componentes electrónicos son rectangulares, no obstante, las macetas normalmente son circulares. Por otro lado, un diseño con esquinas provoca más situaciones donde el robot se atasque.

En contraposición, la forma circular hace que los componentes electrónicos no puedan colocarse de manera que cubran todo el espacio de la circunferencia, pero gracias a su

forma podrá parecer una maceta normal. A su vez, la forma circular evita las posibles situaciones donde una esquina produciría un atasco.

Agua

A la hora de decidir las dimensiones del depósito de agua del robot tenemos que calcular la cantidad de agua necesaria en un riego. Por ello primero tenemos que acotar el diámetro de la maceta. Para conseguir un compromiso entre capacidad y tamaño hemos decidido acotar el diámetro de la maceta en 15 cm

Diametro cm.	Altura cm.	Volumen en Lt.	Plato Sugerido
13	10.8	1.1	12
15	12.3	1.6	14
17	14.5	2.4	16
19	15.5	3.1	18
21	17	4.3	18
24	19	6.5	20
27	23	9.5	22
30	26	13.5	24
35	32	23	26
40	36	35	32
45	40	46	36
50	45	65	40
60	47	90	45

Figura 4.1: Tamaños de maceta y sus correspondientes litros. [12]

Se recomienda utilizar del 5 % al 10 % de la capacidad de la maceta de agua para regarla. Una maceta de 15cm de diámetro contiene 1.6 litros de tierra, por lo tanto, la cantidad recomendada máxima serían 0.16 litros de agua. Para reducir la interacción necesaria por parte del usuario se calculará para renovar el agua cada mes. Siendo el depósito de 0.48 litros de agua.

Otro tema a tener en cuenta es el agua residual que expulsa la planta al ser regada. Ya sea debido a la composición de la tierra o a una sobredosis de agua. En el caso de que la maceta tenga agujeros, habrá una salida de agua. Para evitar una posible humedad innecesaria y el reaprovechamiento del agua sobrante proponemos una unión entre el depósito y la zona del macetero.

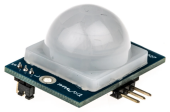
Luz

A la hora del funcionamiento del robot uno de los temas más importantes es la medición de luz y detección de las variaciones de esta misma. Para la medición de luz se dispone de dos formas: utilizando únicamente el sensor ldr o módulos donde viene montado el ldr. No obstante, debido a que necesitamos convertir esas variaciones en interrupciones para el Arduino y, por lo tanto, tiene que ser dadas mediante pines digitales se usarán módulos que devuelvan una señal digital.

Por último, la posición de los sensores tiene que estar alejada de las posibles hojas de la planta y dar una lectura lo suficiente amplia para cubrir toda la circunferencia. Siendo necesario dos sensores, cada uno de ellos abarcando media circunferencia.

Detección de obstáculos

Necesitamos detectar el mayor tipo de materiales con nuestro sensor. Por ello, aunque existan diferentes sensores para detectar obstáculos, sensores PIR, infrarrojos y ultrasónicos, utilizaremos los ultrasónicos. Esto es debido a que los sensores PIR solo detectan la variación de calor por lo que podríamos evitar personas o animales, pero no objetos. Por el contrario, los infrarrojos si detectan objetos, pero se ven afectados por los colores y materiales de estos. Debido a estas razones, los sensores ultrasónicos son la opción más adecuada.



(a) Sensor PIR

(b) Sensor infrarrojo

(c) Sensor ultrasónico

Figura 4.2: Tipos de sensores para la detección de obstáculos.

Placa Arduino

La parte más importante del sistema, pero a su vez la más dependiente de los componentes que se vayan a utilizar. Al necesitarse 3 pines capaces de soportar interrupciones, Tabla 3.1, se eligió la placa de Arduino Mega para el control.

Baterías

Debido a que se usa un Arduino Mega y ha de ser alimentado entre 7 a 12V se decidió por utilizar 9v.

Motores

El robot ha de ser capaz de mover el peso de la planta más el robot. En el caso del peso del robot partimos de que el contenido del depósito van a ser 0.4 kg, añadiendo una estimación de que chasis mas componentes serán 400g obtenemos 0.8kg de peso total. Por otro lado, las plantas no tienen un peso específico y varían en función del tipo de la maceta y el tipo de planta por lo que apuntaremos a soportar una planta que pese 0.7 Kg o menos.

Por lo tanto, el robot con maceta pesará alrededor de 1.5 kg. Dando un coeficiente de seguridad del 1.5 necesitaremos unos motores capaces de mover 3kg. Otro factor importante a decidir es la cantidad de ruedas que se va a utilizar. Las opciones más adecuadas serían 2 ruedas motoras y una de tracción, o bien 4 ruedas motoras.

4.1.2 Diseños

Teniendo en consideración las características habladas en la subsección 4.1.1 y los componentes electrónicos a usarse se proponen varios diseños.

Primer diseño

El primer diseño imita un macetero convencional teniendo en la parte superior el hueco para la maceta. Interiormente está dividido en dos partes, la zona del depósito y la zona de los componentes electrónicos. Esta última, a su vez, está dividida en varios pisos. La zona superior cuenta con diferentes agujeros para el paso de los cables de los sensores, pantalla lcd y manguera de riego. El peso de la maceta y de los componentes de las zonas superiores es soportado por las cinco columnas que tiene.

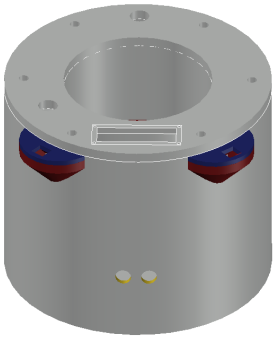


Figura 4.3: Primer diseño

Segundo diseño

El segundo diseño se plantea teniendo más en cuenta el diseño de cara al usuario que la practicidad. Cuenta con un macetero propio de forma irregular y acceso al depósito desde la parte superior. Al ser un cilindro desde la base hasta la parte superior hace que no sea posible reaprovechar el agua saliente.

Figura 4.4: Segundo diseño del chasis del robot.

Diseño final

El diseño final se basa principalmente en el primer diseño. Por una parte, la estructura que soporta el peso ha sido cambiada de 6 columnas a solo 2, debajo de la zona del macetero. No obstante, para la unión de la parte superior con el cuerpo se han creado hendiduras para que no gire el macetero.

Figura 4.5: Diseño final.

Para el interior del robot se ha usado de guía el segundo diseño contado con únicamente una segunda media planta cercana a la abertura para el sensor de ultrasonidos. A su vez, el depósito en vez de estar sujeto a las columnas está metido a presión en la base y dispone de una puerta trasera para su fácil acceso a la hora de rellenar o limpiarlo. Las ruedas se han mantenido iguales a los dos diseños pues su posición ofrece protección contra posibles elementos externos.

Figura 4.6: Vista lateral del diseño final.

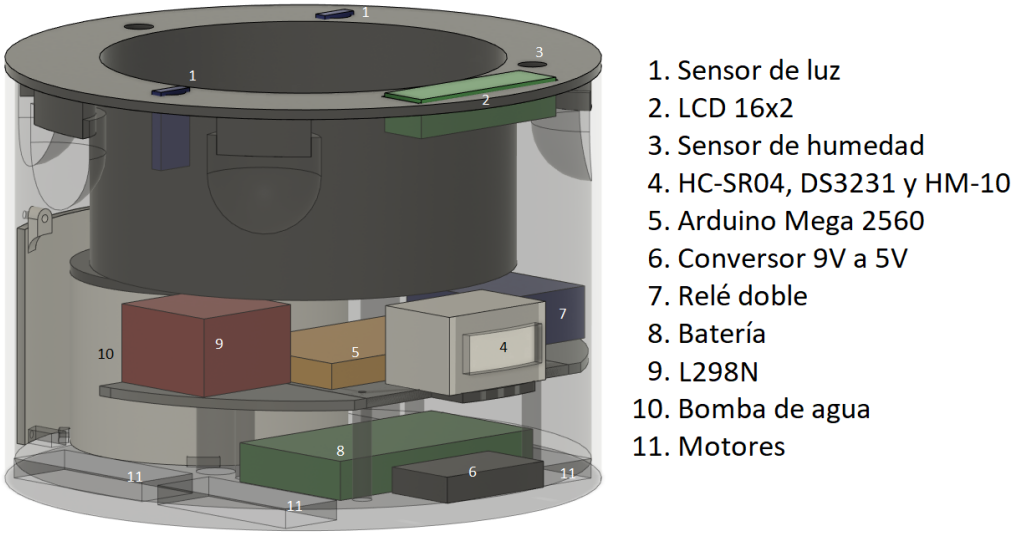


Figura 4.7: Posición de los componentes.

El depósito tiene unas medidas de 9.7cm de diámetro y 6.5cm de altura consiguiendo así una capacidad de 0.48 litros de agua. En la tapa del depósito se encuentran dos agujeros uno que conecta con el fondo del depósito para recolectar el exceso de agua y otro por el que sale el tubo de riego hacia la planta.

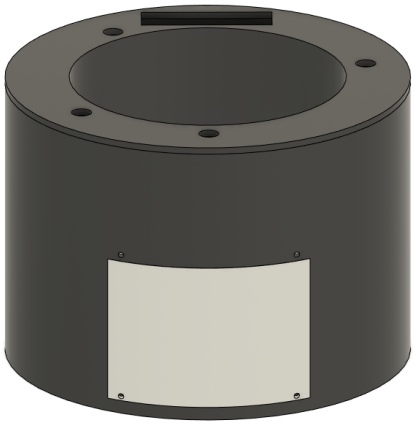


Figura 4.8: Vista trasera del robot.

Despiece del diseño final

El diseño está compuesto de 6 partes: el cuerpo del robot, el depósito, la puerta trasera, la tapa del depósito, el piso superior y la parte superior. Los planos de las piezas se encuentran en el Anexo 7.2.

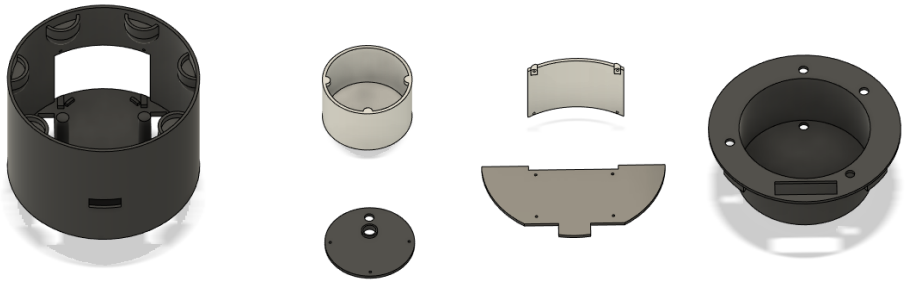


Figura 4.9: Despiece visto desde el frente

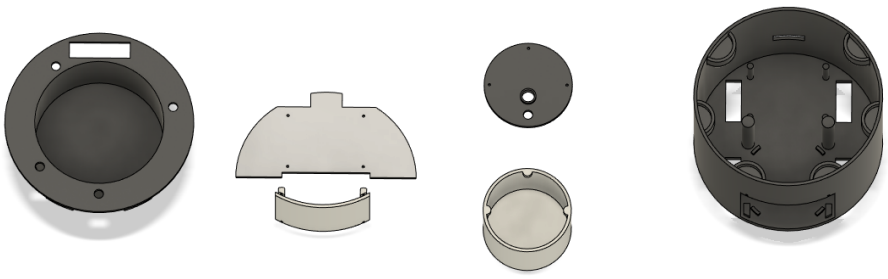


Figura 4.10: Despiece visto desde detrás

4.2 Componentes electrónicos

Los componentes electrónicos utilizados en el proyecto van a estar divididos en cuatro subsecciones: la primera subsección será la de control, 4.2.1, la cual se encarga del correcto funcionamiento del robot; la segunda subsección será la de actuadores y sensores, 4.2.2, siendo los encargados de recoger la información y enviarla para su posterior procesado; la tercera subsección será la de baterías, 4.2.3, encargada de proporcionar energía al robot; y por último, la cuarta subsección será la de motores, 4.2.5, utilizados para mover el robot y el agua.

4.2.1 Elementos de control

Arduino Mega 2560

Las características de la placa son: puerto USB de tipo C, 54 pines digitales de los cuales 15 pines con capacidad de trabajar en PWM, 16 pines analógicos y 4 pines para comunicación serial. Además, el voltaje en el que se alimenta es de 7V a 12V, mientras que los pines trabajan a 5v.

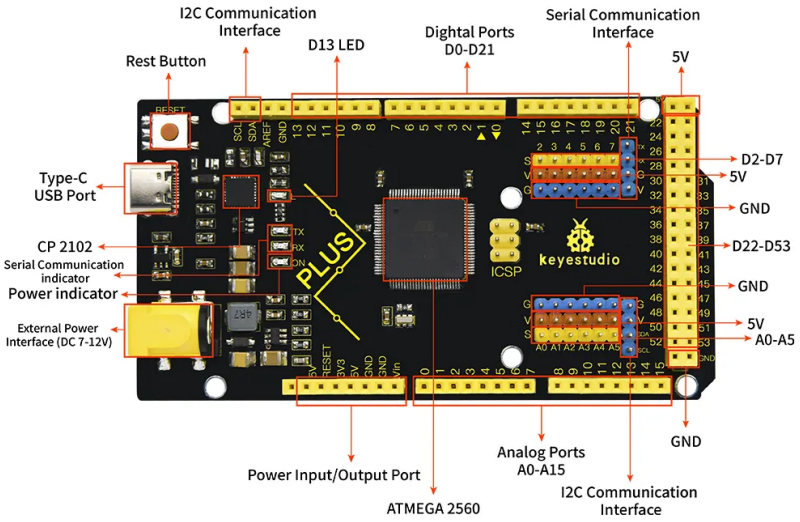


Figura 4.11: Arduino Mega 2560

L298N

El L298N[22] se encarga de controlar los motores mediante los 6 pines que dispone, puede controlar dos motores a la vez. Cada salida de motor tiene asignado 3 pines dos de ellos cambiarán la polaridad para indicar el giro del motor y el restante indicará la velocidad del motor. Este último tendrá que estar conectado a un pin PWM mientras que los otros dos será a pines digitales.

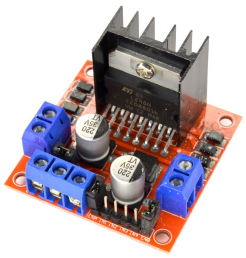


Figura 4.12: L298N.

DS3231

El DS3231[13] es un reloj en tiempo real que mantiene la hora actual y puede producir alarmas. Consta de 6 pines: **VCC**, alimentado a 5v; **GND**, tierra; **SCL** y **SCA**, para la comunicación I2C con el Arduino siendo **SCA** para la señal de datos y **SCL** para la señal de reloj; **SQW**, pin utilizado para comunicar las alarmas al Arduino; y, por último, **32K**, que manda una señal de reloj.

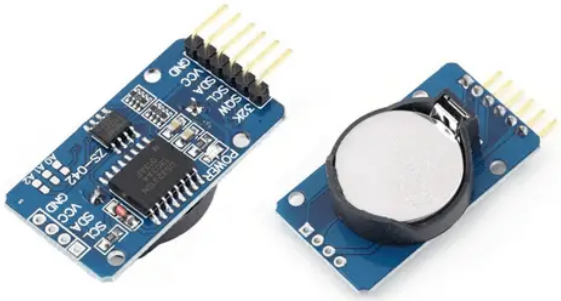


Figura 4.13: DS3231

4.2.2 Actuadores y sensores

Relé doble

El relé es un dispositivo electromagnético, que funciona como un interruptor. Está compuesto internamente por una bobina y un electroimán. Los relés se controlarán mediante el Arduino, el primero cerrará el circuito para la activación de la bomba de agua para regar la planta y el segundo alimentará a los sensores cuando se necesiten tomar medidas.

Figura 4.14: Relé doble

Consta de un total de 6 pines de salida, 3 para cada relé y 6 pines de entrada. Los pines de entrada: VCC, alimentación a 5v; GND, tierra; IN1, entrada de control del relé 1; IN2; entrada de control del relé 2; GND y COM, para en el caso que se desea separar las diferentes tierras de las entradas de control. Los pines de salida serán iguales en ambos casos, el común, normalmente cerrado y normalmente abierto.

HCSR04

El sensor HC-SR04 [7] es un sensor de distancia mediante ultrasonido. Para calcular la distancia envía una onda ultrasónica y cuenta el tiempo que tarda en recibir la onda reflejada que retorna el objeto. Se pueden encontrar sensores que envíen y reciban la onda con un mismo transductor, pero en nuestro caso cuenta con dos transductores, uno para enviar y otro para recibir.

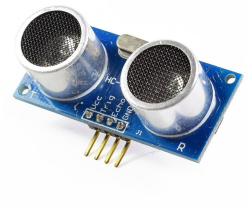


Figura 4.15: HCSR04.

Cabe destacar que el sensor puede detectar objetos trasparentes o objetos que tengan formas complejas como por ejemplo malla y no se ve afectado por la niebla o humo. A la vez no puede detectar objetos que por su forma al devolver la onda se desvíe del

transductor, objetos muy pequeños o que por el material que estén hecho absorban la onda y no la devuelvan.

Sensor de humedad del suelo

El sensor de humedad del suelo puede encontrarse en diferentes formas, las más comunes son en forma de estaca plana o en forma de tenedor de dos puntas. Los dos extremos son conductores que funcionan como una resistencia variable. El valor varía dependiendo de la cantidad de agua en el suelo.

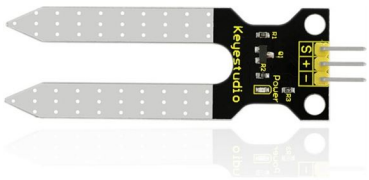


Figura 4.16: Sensor de humedad del suelo.

Debido a que trabaja en un ambiente con agua, para alargar la vida del sensor, lo ideal es solo tenerlo encendido cuando se vayan a tomar las medidas. En el caso de que se dejase todo el rato encendido terminaría degradándose con el tiempo, debido a que acabaría produciéndose óxido.

KS-028

El sensor KS-028 está basado en una resistencia dependiente de la luz[11], el valor devuelto será correspondiente al nivel de luz disponible. El que utilizamos cuenta con dos salidas de valores: una salida analógica, que dará el valor constante de la lectura y una salida digital que se activará al sobrepasar una cantidad de luz[23].

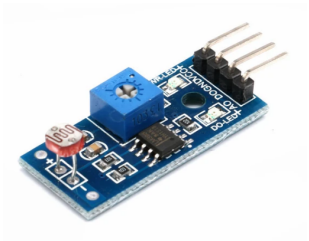


Figura 4.17: Sensor de luz KS-028.

HM-10

El modulo HM-10[6] es un módulo de Bluetooth 4.0 BLE para aplicaciones de bajo consumo. Puede llegar a comunicarse a una distancia de 100m. Se trabaja igual que con un módulo de Bluetooth, pero la capacidad de información que se puede enviar baja de 3Mbps a 1Mbps.

Figura 4.18: HM-10 Bluetooth-4.0 BLE.

I2C Serial Interface 1602 LCD

La pantalla LCD tiene 2 líneas de 16 caracteres cada una. Además, detrás de la pantalla podremos encontrar un potenciómetro para ajustar el brillo de la pantalla. La comunicación se hará a través del protocolo I2C y por defecto la dirección estará configurada en la 0x3F[21].

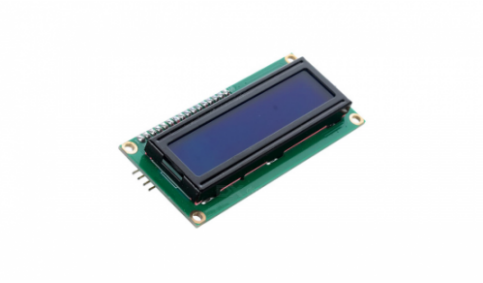


Figura 4.19: Pantalla LCD

4.2.3 Baterías

Porta pilas

Utilizaremos un porta pilas de capacidad para 6 pilas para que nos dé 9V. Las pilas utilizadas serán AA que darán cada una de ellas 1,5V.

Figura 4.20: Porta pilas de 6 pilas AA.

Tendremos una vida útil de 6h a 25h debido a que la batería da 2500mA y el consumo en reposo del Arduino con los sensores es de 100mA y en activo 400mA.

4.2.4 Conversor de voltaje

Conversor de corriente continua, capaz de aceptar voltajes entre 9V y 36V en su entrada y ofreciendo 5V de salida. La entrada se puede alimentar tanto por enchufe de corriente continua como por terminales, mientras que la salida puede ser por USB o terminales.

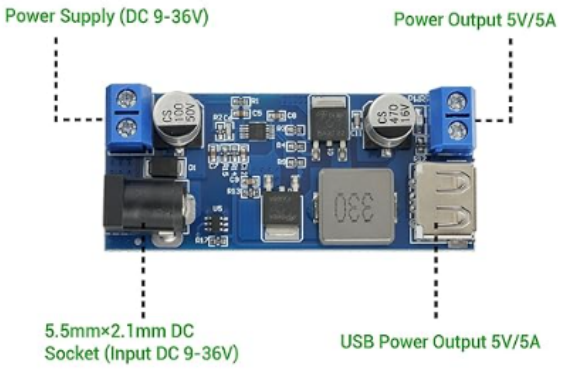


Figura 4.21: Conversor de 9V a 5V.

4.2.5 Motores

Motor TT

Los motores TT[10] son motores de corriente continua. Disponen de dos cables, positivo y negativo. Dependiendo de la manera de conectarlos el giro del motor variará. A estos motores irán acopladas las ruedas que han sido creadas para estos motores.

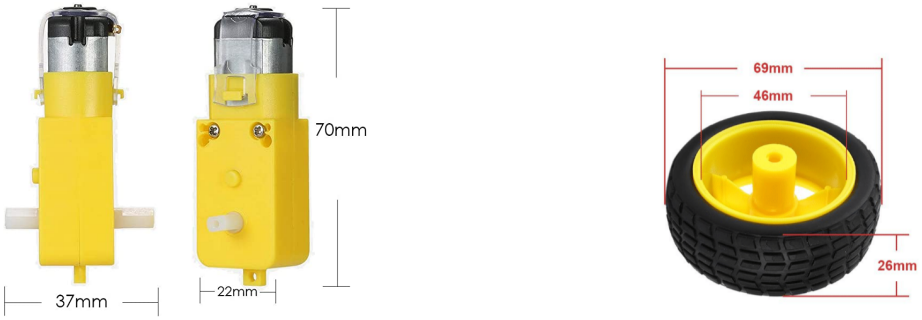


Figura 4.22: Motor TT y ruedas

Bomba de agua

La bomba que utilizaremos para regar la planta será capaz de bombear agua a una elevación entre 0.2 y 1.10 metros y el flujo es de 1.3 a 2L / min dependiendo de la elevación[19].

Figura 4.23: Bomba de agua

4.3 Presupuesto

En esta subsección estudiaremos el presupuesto de los componentes necesarios para la creación del robot.

Componentes eléctricos			
	Cantidad	Precio	Total
Arduino Mega 2560	1	19,12	19,12
DS3231	1	3.99	3.99
L298N	1	3.45	3.45
HCSR04	1	5.95	5.95
Relé doble	1	4.99	4.99
Sensor de humedad del suelo	1	3.95	3.95
Sensor LDR analógico KS-028	2	1.5	3
HM-10 Bluetooth-4.0	1	10	10
Pantalla LCD	1	4.99	4.99
Motor DC	pack	12	12
Bomba de agua	1	2	2
Portapilas 9 V	1	6	6
Pilas 1,5V	6	0.30	1.8
Cableado	pack	9.99	9.99
TOTAL			91.23€

Se ha pedido también presupuesto a empresas de impresión 3D para en el caso que no se disponga de una impresora 3D tener los precios de fabricación. Los precios son por las 5 piezas.

Impresión 3D	
Material	Precio
ABS	200€
SAS	200€
PLA	120€

4.4 Esquema eléctrico

En esta subsección mostraremos en detalle las conexiones entre los diferentes componentes electrónicos. Si se desea el esquema completo se encuentra en el Anexo en el apartado 7.3.

Arduino

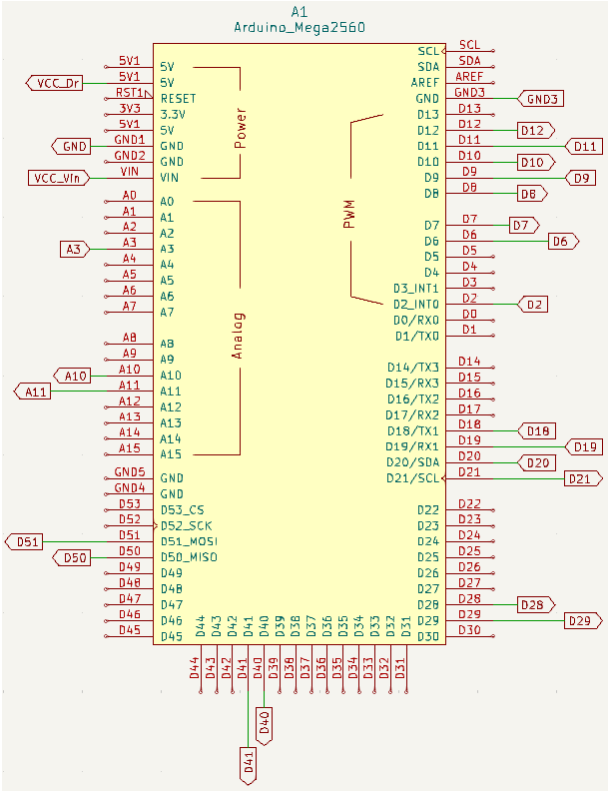


Figura 4.24: Pin out Arduino.

Si se desea modificar alguno de los pines utilizados hay 5 pines que no pueden ser cambiados. Serían **D2**, **D18**, **D19**, **D20** y **D21**, esto es debido a que **D2**, **D18** y **D19** son los pines que se utilizan para las interrupciones y **D20** y **D21** son los pines del Arduino Mega que se utilizan para la comunicación del bus I2C. Así mismo, **D6**, **D7**, **D8** y **D9** han de mantenerse en pines PWM debido a los componentes a los que van conectados. Los pines **D6** y **D7** están encargados de modificar la velocidad de los motores y **D8** y **D9** los utilizados por el sensor HC-SR04. Todos los demás pines respetando el tipo de pin pueden ser cambiados a otros si se desea.

Los pines digitales **D10** y **D11** serán utilizados para la conexión Bluetooth configurados en el código pues los pines por defecto para este tipo de comunicación son los pines **D18** y **D19** que son utilizados para las interrupciones. A su vez, los pines **D28**, **D29** y **D12** serán utilizados para, los dos primeros, activar los relés y el último, para activar el sensor de humedad. Por último, los pines **D40**, **D41**, **D50** y **D51** serán utilizados para controlar el giro de los motores.

Los pines analógicos son utilizados para recibir los parámetros de los sensores de luz, **A10** y **A11**, y el sensor de humedad de la tierra, **A3**.

LN298

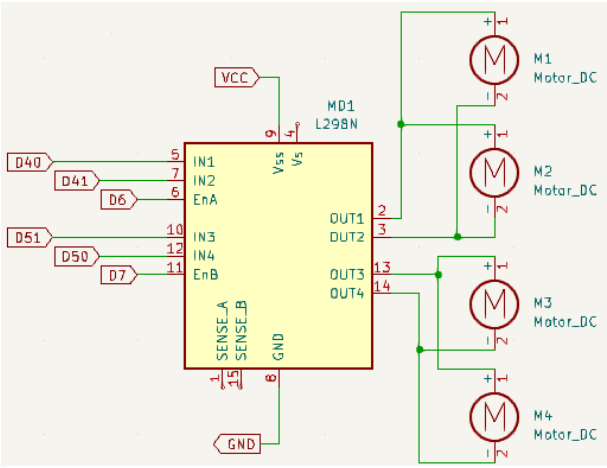


Figura 4.25: Pin out LN298

Debido a que no existe indicaciones para saber el giro del motor de antemano es posible que a la hora de conectarlos se deban intercambiar los pines.

Sensores

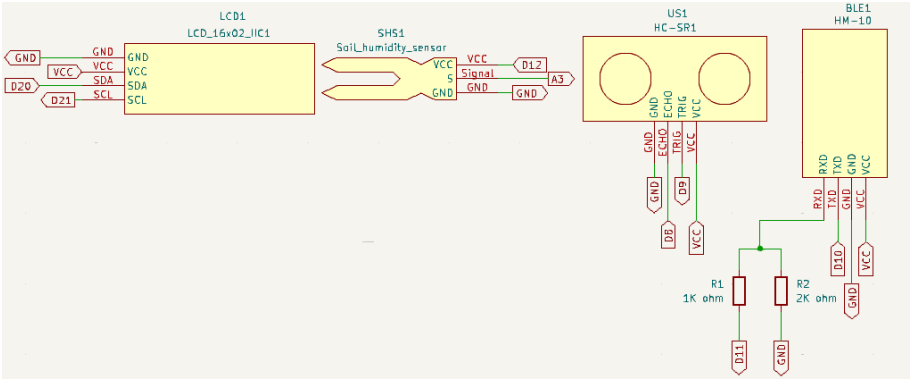


Figura 4.26: Pin out del sensor de humedad, pantalla lcd, HC-SR04 y HM-10

El sensor de humedad en vez de ser alimentado por la batería es alimentado por el Arduino, esto es debido a lo explicado en el apartado 4.2.2. Por otro lado, aunque el pin **RXD** del HM-10 puede conectarse directamente al pin **D11** y funcionaría, se recomienda hacer el divisor de tensiones pues la lógica del pin trabaja sobre 3.3V.

Alimentación

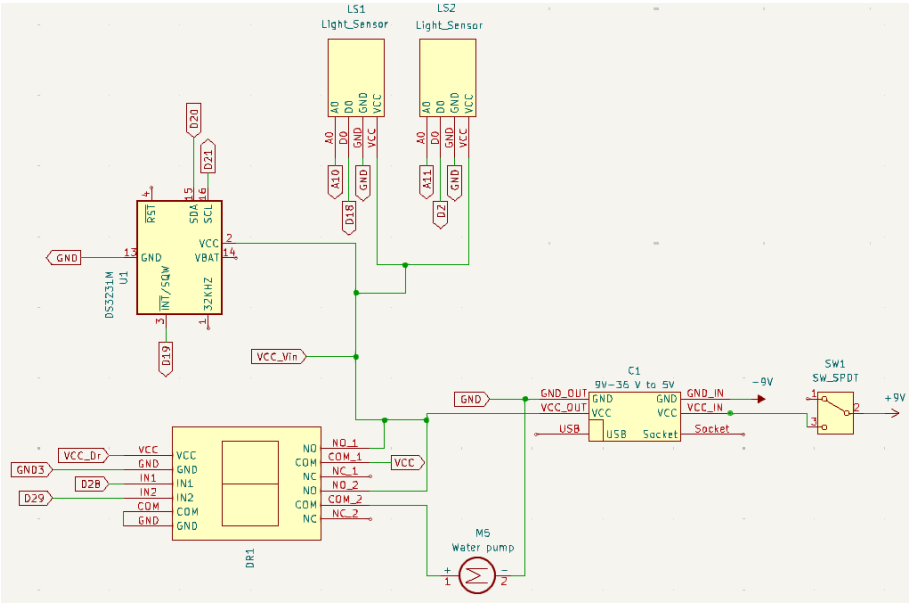


Figura 4.27: Conexiones batería y relés

El Arduino Mega acepta un voltaje entre 7V a 12V por lo tanto irá directamente alimentado desde la batería, no obstante, todos los demás componentes necesitan 5v.Por esta razón se ha se ha introducido un conversor para conseguir los 5v.

Los únicos componentes que necesitan alimentación continua son el RTC, el relé doble, el Arduino Mega y los dos sensores de luz. Los demás componentes serán alimentados solo si se activa el relé consiguiendo que no consuman batería mientras no se están utilizando.

5 Software

El apartado del software estará dividido en dos secciones, la primera tratará sobre el programa que tiene el robot 5.1 , y la segunda hablará sobre la aplicación móvil, que se utiliza para comunicarse con él, 5.2.

5.1 Robot

El programa se encargará, después de ser configurado con los parámetros requeridos, de reposicionar la planta al lugar donde tenga la luz objetivo. Después entrará en modo suspensión hasta que se detecte una interrupción. Las interrupciones posibles son por cambio brusco de luz o por alarma del módulo DS3231. Estas últimas son porque es hora de regar la planta o porque ha pasado una hora desde que entró en modo suspensión. En el caso de que sea de noche solo la alarma de riego se mantendrá activa.

Para llevar la planta al lugar correcto se usarán las lecturas de los sensores de luz, siguiendo el nivel de luz que más cerca de la luz objetivo esté. El sensor de ultrasonidos se usará para evitar obstaculos en el recorrido.

El robot tiene una pantalla de 2 filas de 16 segmentos que será actualizada con los parámetros de humedad, luz y la configuración de riego. También será donde se notifiquen los problemas que aparezcan, como que no se haya configurado el robot correctamente o haya demasiada humedad en la tierra para el riego.

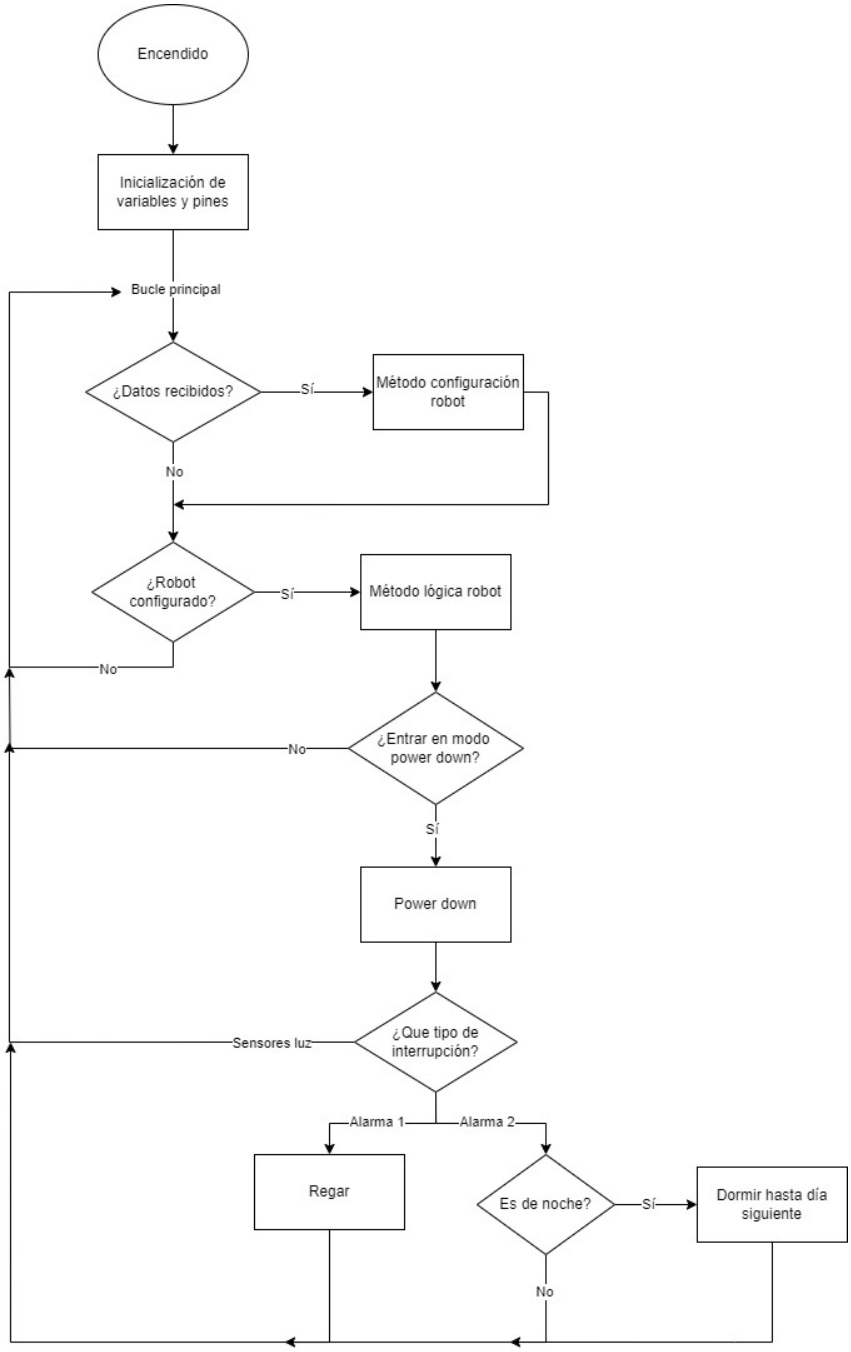


Figura 5.1: Diagrama de flujo principal

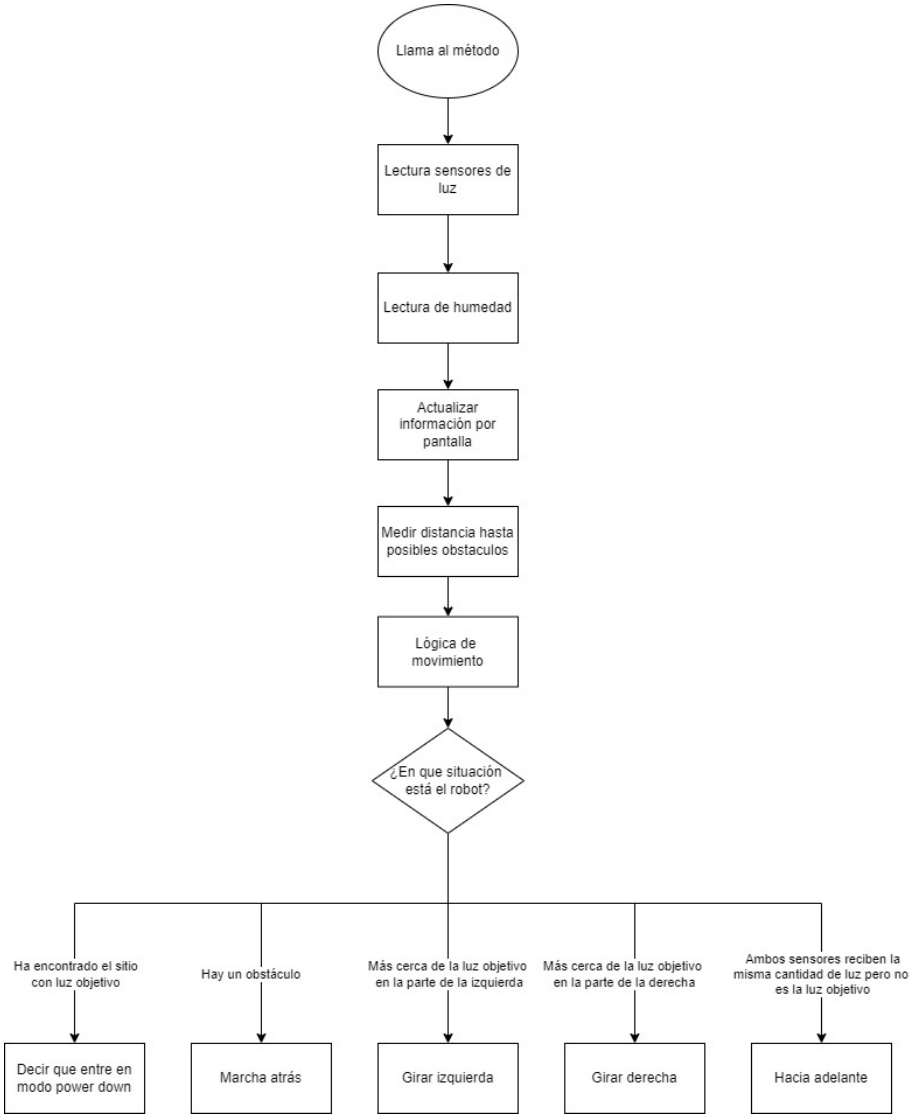


Figura 5.2: Diagrama de flujo método lógica del robot

El programa en su totalidad está compuesto de 13 métodos:

- **setup()**,5.1.2, inicializará los pines y variables.
- **loop()**,5.1.3, bucle principal del programa.
- **getDistance()**,5.1.4, donde calcularemos la distancia a los posible obstáculos.
- **movementLogic()**,5.1.5, encargado de la lógica de movimiento del robot.
- **wakeUp()**,5.1.6, se ejecutará al saltar una interrupción.
- **getHumidity()**,5.1.7, encargado de tomar y guardar las lecturas de la humedad del suelo.
- **waterPlant()**,5.1.8, quien se encarga de regar la planta
- **showInfo()**,5.1.9, encargado de actualizar la información de la pantalla.
- **logicRobot()**5.1.10, método maestro encargado de hacer la lectura de la cantidad de luz y de llamar a los métodos necesarios para el control del robot.
- **setRTCAlarm2()**,5.1.11, se encarga de crear y modificar la Alarma2 del módulo DS3231.
- **setRTC()**,5.1.12, función de configuración del módulo DS3231.
- **convertToByte()**,5.1.13, método para convertir chars en bytes
- **configureRobot()**,5.1.14, encargado de configurar el robot cuando reciba los datos por el puerto Serial

5.1.1 Librerías y declaración de variables

Las librerías que vamos a utilizar son ***L298NX2*** [2] para el manejo del driver de los motores. ***DS3231*** [1] para el control del RTC. ***Wire*** librería que depende tanto ***DS3231*** como ***LiquidCrystalI2C*** [4], está última se utilizará para controlar la pantalla de segmentos. ***SoftwareSerial*** [5] la cual utilizaremos para cambiar los pines por defecto de RX y TX pues están ocupados con las interrupciones y por último dos librerías que vienen ya incluidas en el IDE de Arduino, ***avr/interrupt*** y ***avr/sleep*** [16].

```
1 #include <L298NX2.h>
2 #include <DS3231.h>
3 #include <Wire.h>
4 #include <LiquidCrystal_I2C.h>
5 #include <SoftwareSerial.h>
6 #include <avr/interrupt.h>
7 #include <avr/sleep.h>
```

Para un fácil cambio a la hora de modificar los pines se definirán los nombres que harán referencia a estos. En todo el código se hará referencia al nombre en vez de al número del pin.

```
9 #define ENA  6       |
10 #define IN1A 41      | Pines control motor izquierda
11 #define IN2A 40      |
12 #define IN1B 51          |
13 #define IN2B 50          | Pines control motor derecha
14 #define ENB 7            |
15
16 #define ECHO 8          | Sensor ultrasonidos
17 #define TRIG 9          |
18
19 #define LIGHTLA A10      | Sensores de luz
20 #define LIGHTRA A11      |
21
22 #define HUM A3           | Sensor humedad
23 #define HUMACT 12        | Control alimentación
24
25 #define RXD 10           | Pines HM-10
26 #define TXD 11           |
27
28 #define RTC 18           | Interrupción RTC
29 #define LIGHT 19         | Interrupción sensor luz 1
30 #define LIGHT2 2         | Interrupción sensor luz 2
31
32 #define RELE1 28         | Control relé sensores
33 #define RELE2 29         | Control relé para bomba agua
```

```
36 const int lightMargin = 10;
37 const int humidityMax = 20;
38 const int distMin = 40;
39 int lightSensorR, lightSensorL,lightL, lightR ;
40 double light, dist, humidity;
41 long timeUltrasonicSensor = 0;
42 bool flagHumidity, flagConfiguration, flagSleep = false;
43 byte year, month, date, dOW, hour, minute, second, hourA,
     dOWA, dOWB, cap, plantLightNeeds, beginNightH, endNightH
     ;
44 bool flagDay = true;
45 String week= "-------";
46
47 DS3231 myRTC;
```

```
48 L298NX2 motors(ENA, IN1A, IN2A, ENB, IN1B, IN2B);
49 LiquidCrystal_I2C lcd(0x3F ,16,2);
50 SoftwareSerial mySerial(RXD, TXD);
```

5.1.2 Método setup()

Declaramos los modos de los pines, inicializamos el puerto Serial para la comunicación Bluetooth y el mensaje de base de la pantalla.

```
323 void setup(){
324     mySerial.begin(115200);
325 
326     pinMode(RTC, INPUT_PULLUP);
327     pinMode(LIGHT, INPUT);
328     pinMode(BOMB, OUTPUT);
329     pinMode(LIGHTLA , INPUT);
330     pinMode(LIGHTRA , INPUT);
331     pinMode(HUMACT , OUTPUT);
332     pinMode(HUM , INPUT);
333     pinMode(TRIG , OUTPUT);
334     pinMode(ECHO , INPUT);
335     pinMode(ENA, OUTPUT);
336     pinMode(IN1A, OUTPUT);
337     pinMode(IN2A, OUTPUT);
338     pinMode(ENB, OUTPUT);
339     pinMode(IN1B, OUTPUT);
340     pinMode(IN2B, OUTPUT);
341     pinMode(RELE1, OUTPUT);
342     pinMode(RELE2, OUTPUT);
343 
344     lcd.init();
345     lcd.backlight();
346     lcd.print("No␣configuración");
347     lcd.setCursor(0, 1);
348     lcd.print("Introduzca␣datos");
349 
350     digitalWrite(RELE1,HIGH);
351 }
```

5.1.3 Método loop

El bucle principal del programa, en primer lugar comprobamos que ha sido configurado el robot. En el caso de no estar inicializado, el robot esperará hasta que reciba dichos datos, aplicándolos mediante el método **configureRobot()**. Si por el contrario está configurado, se ejecutará el método **logicRobot()**, quien se encargará de mover el robot. Al finalizar comprobará si ha llegado a su destino.

Si se ha llegado al sitio objetivo y esta marcado el valor de **flagSleep** entrará en modo *power down* hasta que reciba una interrupción. Cuando se despierte comprobará que tipo de interrupción ha sido. Si es por una interrupción de la alarma 1, significa que es hora de regar la planta y llamará al método **waterPlant()**. Comprobará si está en el periodo de noche o de día. Si es de noche levantará la bandera para que vuelva a entrar en modo *power down*.

```
356 void loop(){
357   if (flagConfiguration){
358     if(flagDay){
359       logicRobot();
360     }
361     if (flagSleep){
362       Serial.println("SLEEP");
363       digitalWrite(RELE1,LOW);
364       lcd.noBacklight();
365       set_sleep_mode(SLEEP_MODE_PWR_DOWN);
366       sleep_enable();
367       delay(100);
368       sleep_mode();
369       delay(100);
370
371       if (myRTC.checkIfAlarm(1)){
372         waterPlant();
373       }
374       bool auxBool = false;
375       byte theHour = myRTC.getHour(auxBool, auxBool);
376       if((theHour >= beginNightH || theHour <= endNightH)){
377           flagDay = false;
378           flagSleep = true;
379       }else{
380           flagDay = true;
381           flagSleep = false;
382           digitalWrite(RELE1,HIGH);
383           lcd.backlight();
384       }
385     }
386   }
387   if (mySerial.available() > 0 ){
388     flagConfiguration = false;
```

```
389     configureRobot();
390   }
391   delay(1000);
392 }
```

5.1.4 *Método getDistance()*

Es el encargado de medir la distancia que hay ente los objetos y el robot, para ello activaremos durante 10 microsegundos el pin de TRIG que enviará una onda. Cuando la onda llegué al obstáculo rebotará y será captado por el pin de ECHO, con la que calcularemos la distancia.[7].

```
67 void getDistance(){
68
69     digitalWrite(TRIG ,HIGH);
70     delayMicroseconds(10);
71     digitalWrite(TRIG , LOW);
72
73     timeUltrasonicSensor = (pulseIn(ECHO , HIGH));
74
75     dist = float(timeUltrasonicSensor /58);
76 }
```

5.1.5 *Método movementLogic()*

En primer lugar comprobamos que la luz que reciben los sensores tiene el valor objetivo con el margen de error estipulado. En ese caso, levantaremos la bandera para indicar que el robot puede entrar en modo *power down*. Si a raíz de los movimientos anteriores se encuentra en una posición donde puede producirse un choque usará la marcha atrás para evitar el obstáculo. En los casos que la luz objetivo esté más cerca en uno de los laterales se procederá a girar hacia ese lado. Si no se ha dado ninguno de los casos anteriores se procederá a seguir hacia delante.

```
89 void movementLogic(){
90   int aux =  (abs(plantLightNeeds - lightR) <= lightMargin)
         && (abs(plantLightNeeds - lightL) <= lightMargin);
91  if (aux){
92     Serial.println("Buen␣sitio");
93     flagSleep = true;
94
95   }else if (dist < distMin && dist > 0){
96     Serial.println("Choque");
97      motors.setSpeed(200);
98      motors.backward();
99      delay(600);
```

```
100
101       if(flagTurn){
102         motors.setSpeedB(255);
103         motors.setSpeedA(200);
104         motors.forwardA();
105         motors.backwardB();
106       }else{
107         motors.setSpeedB(200);
108         motors.setSpeedA(255);
109         motors.forwardB();
110         motors.backwardA();
111       }
112
113       delay(600);
114       motors.stop();
115     }
116     else if(abs(plantLightNeeds - lightR) > abs(
          plantLightNeeds - lightL)){
117       Serial.println("Hay mas luz sensor Derecha");
118       flagTurn = false;
119       motors.setSpeedB(255);
120       motors.setSpeedA(200);
121       motors.forwardB();
122       motors.backwardA();
123       delay(650);
124       motors.forward();
125       delay(300);
126       motors.stop();
127
128     }
129     else if(abs(plantLightNeeds - lightR) < abs(
          plantLightNeeds - lightL)){
130       Serial.println("Hay mas luz sensor Izquierda");
131       flagTurn = true;
132       motors.setSpeedA(255);
133       motors.setSpeedB(200);
134       motors.forwardA();
135       motors.backwardB();
136       delay(650);
137       motors.forward();
138       delay(300);
139       motors.stop();
140
141     }
142     else {
143       Serial.println("Hacia delante");
144       motors.setSpeed(200);
```

```
145     motors.forward();
146     delay(500);
147     motors.stop();
148 
149   }
150 }
```

A la hora de girar para que el giro sea más preciso se invertirá el giro de los motores contrarios al lado de giro. Al terminar la maniobra, se moverá brevemente hacia delante para evitar que quede dando vueltas en círculos. A su vez, cuando se termine de dar marcha atrás se girará en función del sentido de giro antes del posible choque.

5.1.6 Método wakeUp()

Cuando se produzca una interrupción se llamará a este método que deshabilitará el modo *power down* y quitará la bandera para que no pueda volver a entrar en el modo.

```
136 void wakeUp(){
137     sleep_disable();
138     flagSleep = false;
139 }
```

5.1.7 Método getHumidity()

Método encargado de tomar las medidas de humedad de la tierra. Para evitar que el sensor se degrade se activará mediante otro pin del Arduino cuando se vaya a tomar la medida, y se desactivará al terminar. Se evita el uso prolongado de la medición ya que hace uso del voltaje entre dos electrodos, y acabaría produciendo óxido.

```
146 void getHumidity(){
147     digitalWrite(HUMACT , HIGH);
148     humidity = analogRead(HUM );
149     digitalWrite(HUMACT , LOW);
150     humidity = abs (humidity/10);
151 }
```

5.1.8 Método waterPlant()

Será llamado cuando sea hora de regar la planta. Leerá la humedad que hay en la maceta y, posteriormente, si es inferior a la humedad máxima, se procederá al riego. En el caso de que haya demasiada humedad no se regará y se notificará por pantalla.

```
159 void waterPlant(){
160     getHumidity();
161     if ( humidity < humidityMax){
162         flagHumidity = 0;
163         Serial.println("RIEGO");
164         digitalWrite(RELE2 ,HIGH);
165         delay((cap/100)*10);
166         digitalWrite(RELE2 ,LOW);
167     }else{
168         flagHumidity = 1;
169     }
170 }
```

5.1.9 Método showInfo

Se utilizará para mostrar por pantalla la información de los sensores. Al ser una pantalla de 2x16 segmentos en la primera línea mostraremos el porcentaje de luz actual y en la segunda línea mostraremos el porcentaje de humedad y el día seleccionado para riego. En el caso de que hubiese alarma de humedad se modificaría la parte del día seleccionado.

```
180 void showInfo(){
181     lcd.clear();
182     lcd.setCursor(0, 0);
183     lcd.print("L:"+(String)light +"%␣Riego␣" );
184     lcd.setCursor(0,1);
185     lcd.print("H:"+ (String)humidity + "%␣" + week);
186     lcd.setCursor(9,1);
187     if (flagHumidity){
188         lcd.print("Cambiar");
189     }
190 }
```

5.1.10 Método logicRobot

Método que se encargará de tomar medidas de luz y llamar a los métodos encargados de medir parámetros. los cuales se usarán en el movimiento del robot o la actualización de la pantalla.

```
196 void logicRobot(){
197     lightSensorR = analogRead(LIGHTRA);
198     lightSensorL = analogRead(LIGHTLA);
199 
200     lightR = map(lightSensorR, 1023, 0, 0, 100);
201     lightL = map(lightSensorL, 1023, 0, 0, 100);
202     Serial.print("Luz␣real␣␣R:␣");
203     Serial.print(lightR);
204     Serial.print("␣␣I:");
205     Serial.println(lightL);
206     light = (lightR + lightL )/2;
207 
208     getHumidity();
209     showInfoDebug();
210     getDistance();
211     movementLogic();
212 }
```

Al tomar los valores medidos por los sensores de luz los escalaremos entre 0 y 100, donde 0 sería oscuridad y 100 la máxima luz posible. Para sacar la luz global se hará una media entre los dos valores.

5.1.11 Método setRTCAlarm2()

Método encargado de configurar la alarma2, encargada de despertar al robot cada hora.

```
222 void setRTCAlarm2(byte dayA2, byte hourA2, byte typeA, bool
        dayOrWeekDay, bool h12H24, bool pmAm){
223 
224     myRTC.setA2Time(dayA2, hourA2, 0, typeA, dayOrWeekDay,
          h12H24, pmAm);
225     myRTC.turnOnAlarm(2);
226     myRTC.checkIfAlarm(2);
227 
228 }
```

La alarma 2 necesita los parámetros: día, hora, minuto, tipo de alarma, si el día se refiere a día o días de la semana, el formato de la hora y por último, si el formato es de 12h si son AM o PM.

A su vez para que funcione la alarma habrá que encenderla y limpiar la activación de la alarma pues si no se hace no lanzará la interrupción correspondiente.

5.1.12 Método setRTC()

Este método se encargará de configurar el reloj del RTC con la fecha y hora del móvil que se utilice para configurar los parámetros del robot. Además, crearemos las dos alarmas que serán usadas para hacer las interrupciones que despertarán al Arduino.

```
233 void setRTC(byte year, byte month, byte date, byte dOW,
    byte hour, byte minute, byte second,
234             byte hourA, byte dOWA){
235
236         myRTC.setClockMode(false);
237         myRTC.setYear(year);
238         myRTC.setMonth(month);
239         myRTC.setDate(date);
240         myRTC.setDoW(dOW);
241         myRTC.setHour(hour);
242         myRTC.setMinute(minute);
243         myRTC.setSecond(second);
244
245         myRTC.setA1Time(dOWA, hourA, 0, 0, 0x0, true,
246                         false, false);
247         myRTC.turnOnAlarm(1);
248         myRTC.checkIfAlarm(1);
249
250         byte aux = 0b00001100;
251         setRTCAlarm2(dOWB, hourA, aux, true, false, false);
252 }
```

Los parámetros a configurar para que el reloj funcione correctamente serán el formato de la hora (12h o 24h), el año, mes, día, día de la semana (1 a 7), hora, minutos y segundos.

Las alarmas necesitarán de los parámetros: día, hora, minuto, tipo de alarma, si el día se refiere a día o días de la semana, el formato de la hora y, por último, si el formato es de 12h si son AM o PM. Particularmente la alarma 1 también habrá que decirle los segundos.

Los posibles tipos de alarma son cada segundo, cada minuto, cada hora, cada día, cada semana y cada mes.

5.1.13 Método convertToByte()

Debido a que para la configuración del RTC necesitamos los parámetros en bytes y recibimos por Bluetooth un *String* con los datos, este método ha sido creado para facilitar la conversión. Se resta 48 debido a que el valor en ASCII del 0 es 48.

```
257 byte convertToByte(char a, char b){
258   byte temp1, temp2;
259
260   temp1 = (byte)a -48;
261   temp2 = (byte)b -48;
262
263   return temp1*10 + temp2;
264 }
```

5.1.14 Método configureRobot()

En el momento en el que se detecte que en el puerto *mySerial* hay datos recibidos se empezará a guardar los datos en un *array* de *chars* para su posterior conversión a los datos que se necesitan para configurar el robot.

La cadena recibida estará compuesta por 24 caracteres siendo el último para asegurar que la cadena ha sido recibida correctamente. Si no se recibe 'x' en el último lugar se tomará que ha habido un fallo en el envío de datos y se notificará para su posible nuevo intento. La notificación se hará en la pantalla del robot.

```
274 void configureRobot(){
275     char inChar;
276     char inString[25];
277     int i = 0;
278     while (mySerial.available()) {
279         inChar = mySerial.read();
280         inString[i] = inChar;
281         i++;
282     }
283     if (inString[24] == 'x'){
284         year = convertToByte(inString[0],inString[1]);
285         month = convertToByte(inString[2],inString[3]);
286         date = convertToByte(inString[4],inString[5]);
287         dOW = (byte)inString[6] - 48;
288         hour = convertToByte(inString[7],inString[8]);
289         minute = convertToByte(inString[9],inString[10]);
290         second = convertToByte(inString[11],inString[12]);
291
292         hourA = convertToByte(inString[13],inString[14]);
293         dOWA = (byte)inString[15] - 48;
294
```

```
295         cap = convertToByte(inString[16],inString[17]);
296         plantLightNeeds = convertToByte(inString[18],
               inString[19]);
297         beginNightH = convertToByte(inString[20],inString
               [21]);
298         endNightH = convertToByte(inString[22],inString
               [23]);
299
300         setRTC(year, month, date, dOW, hour, minute, second
               , hourA, dOWA);
301         showInfoDebug();
302         flagConfiguration = true;
303
304         delay(1000);
305         attachInterrupt(digitalPinToInterrupt(RTC), wakeUp,
                FALLING);
306         attachInterrupt(digitalPinToInterrupt(LIGHT),
               wakeUp, FALLING);
307         attachInterrupt(digitalPinToInterrupt(LIGHT2),
               wakeUp, LOW);
308
309         week.setCharAt(dOWA-1, 'R');
310     }
311     else{
312         lcd.clear();
313         lcd.setCursor(0, 0);
314         lcd.print("Fallo␣en␣la");
315         lcd.setCursor(0, 1);
316         lcd.print("configuración");
317     }
318 }
```

El formato completo de la cadena estará dividido en seis grandes grupos: el primero referente a la fecha y día de la semana actual, el segundo a la hora actual, el tercero al tiempo de riego, el cuarto al día y hora del riego, el quinto a la luz objetivo para la planta y el último al periodo de noche siendo la hora de comienzo y finalización respectivamente. Además, para control, se añade la letra x significando el final de la cadena.

Figura 5.3: Formato de la cadena recibida

5.2 Aplicación para el móvil

Para la configuración del robot crearemos una app para la transmisión de los datos. Se utilizará MIT App inventor para realizarla y se programará mediante bloques. Además, debido a que estamos utilizando el módulo de Bluetooth HM-10 tendremos que añadir el módulo Bluetooth BLE para poder comunicarnos con este tipo de Bluetooth, pues no está soportado por defecto.

5.2.1 Diseño de la app

La aplicación constará de dos vistas que irán intercambiándose dependiendo de si se está conectado al robot o no.

La primera vista será para pedir los permisos de Bluetooth para poder usarlos en la aplicación y la selección del dispositivo. La lista será poblada con los diferentes dispositivos que encuentre el móvil. En el momento que se seleccione el dispositivo se pasará a la segunda vista.

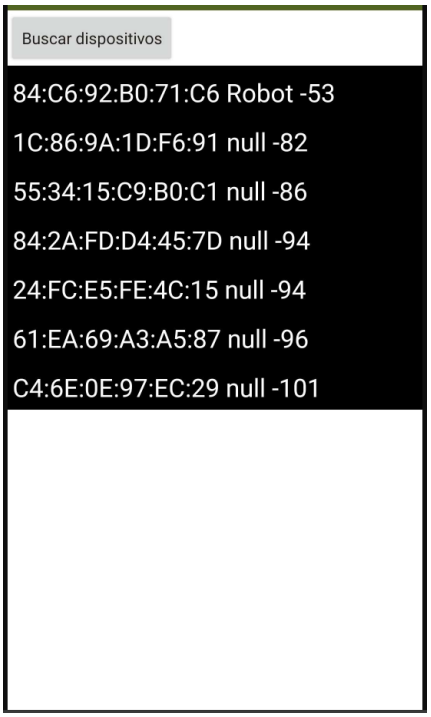


Figura 5.4: Pantalla búsqueda de dispositivo.

La segunda vista, la vista de configuración del robot, será la encargada de enviar la información al robot. Constará de varios selectores para elegir las horas del periodo de noche, el día de la semana y hora de riego, la cantidad de luz y la capacidad de riego.

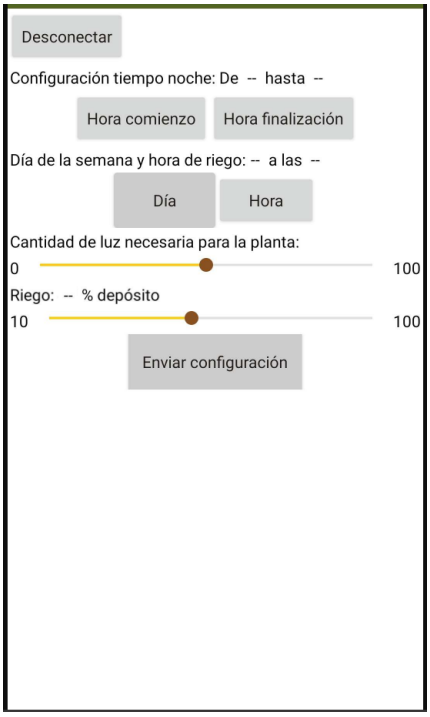


Figura 5.5: Pantalla de configuración del robot.

5.2.2 Código

En este apartado explicaremos en más detalle los diferentes bloques del programa, serán agrupados mediante las funcionalidades que desempeñen. El código completo se encuentra en el Anexo 7.1.2.

Inicialización

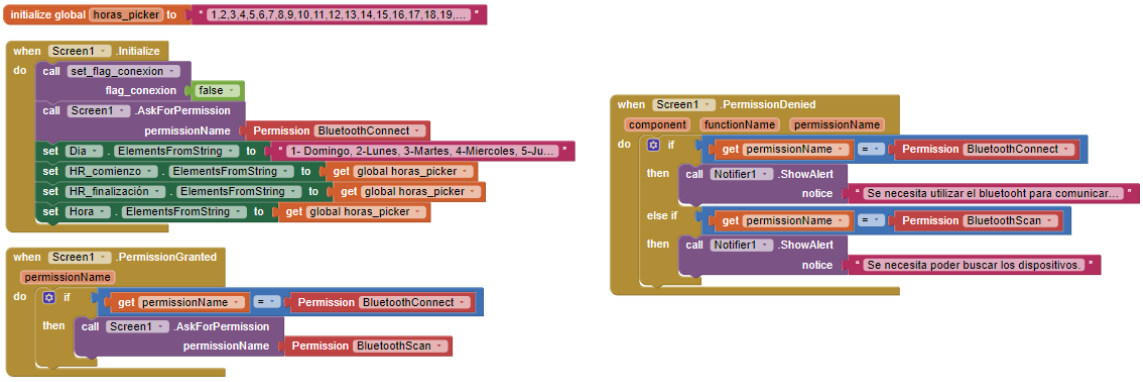


Figura 5.6: Bloques de la parte de inicialización del programa.

Al abrir la aplicación serán los primeros bloques que se ejecuten. Primero pondrá la aplicación en la vista para conectarse al robot. Si es la primera vez que se abre la aplicación se pedirán los permisos para poder usar el Bluetooth, al no permitirlo se mostrarán alertas para que el usuario sepa que es necesario tener acceso a estos permisos.

Control de visibilidad de las partes

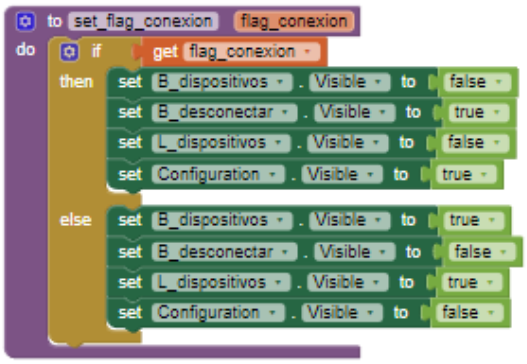


Figura 5.7: Bloque función para control de la visibilidad de las partes de la aplicación

El método de setFlagConexion se utilizará para cambiar entre las dos vistas de la aplicación. Es el encargado de ocultar o mostrar los elementos dependiendo de si se ha producido una conexión o no.

Conexión Bluetooth

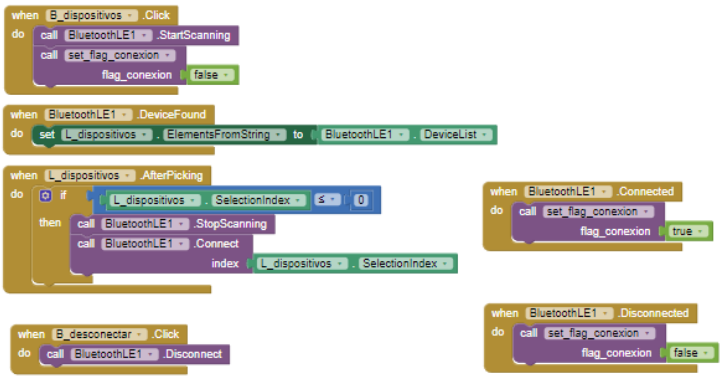


Figura 5.8: Bloques de la lógica para administrar la conexión y desconexión de los dispositivos Bluetooth

Estos bloques serán los encargados de manejar la lógica de conexión al robot. Al pulsar el botón de *Buscar dispositivos* empezará a escanear, y los dispositivos compatibles que encuentre los añadirá a la lista. Al seleccionar uno de los elementos de esta lista parará de escanear y procederá a conectarse al dispositivo. Si se ha conseguido conectar cambiará la vista de la app a la de configuración.

En la vista de configuración tendremos el botón de desconectar que al ser presionado mandará la orden de desconectarse. Cuando se ejecute cambiará a la vista de conexión de dispositivos.

Recogida de los datos de los parámetros requeridos

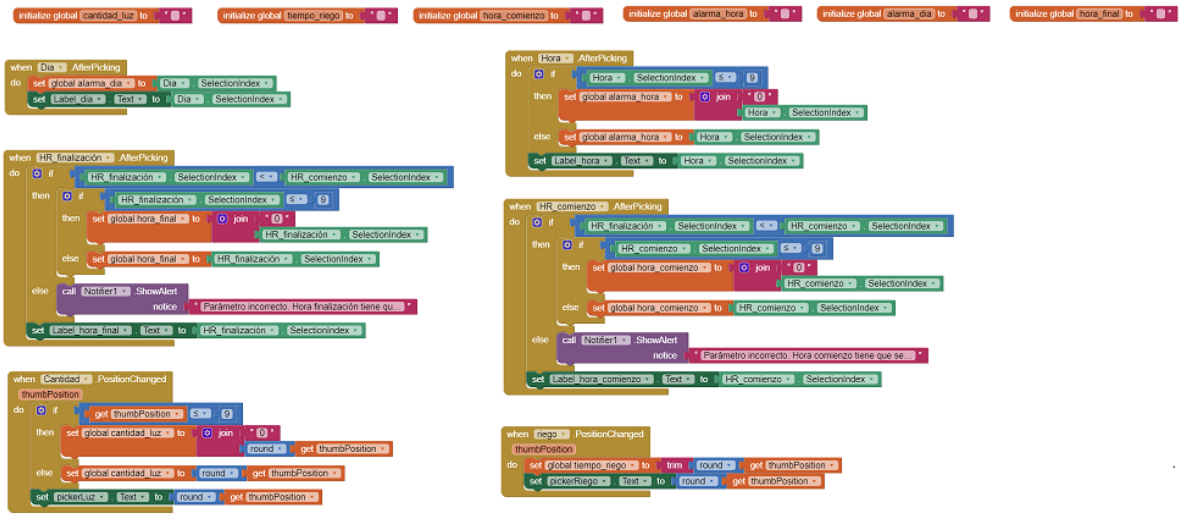


Figura 5.9: Bloques del programa encargados de guardar los datos que el usuario introduce.

En la pantalla de configuración de los parámetros del robot nos encontraremos con 4 apartados. El primero pedirá al usuario configurar el tiempo de la noche, siendo comprobado que incluya un periodo válido. Posteriormente se configurará el día y hora de la semana para el riego. Los dos últimos apartados serán dos *Sliders* que configurarán la cantidad de luz necesaria para la planta y la cantidad del deposito en cada riego.

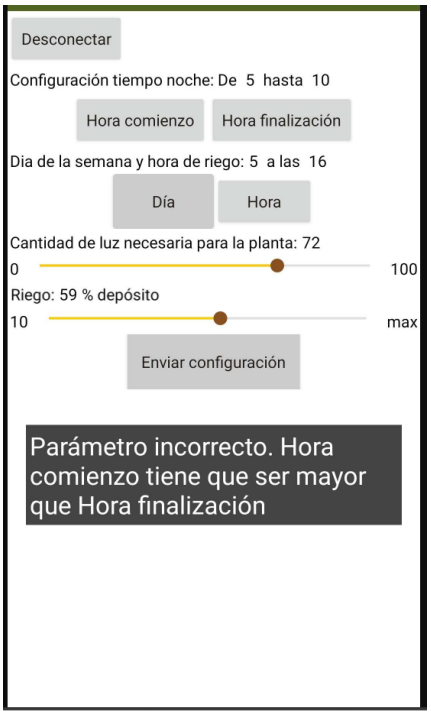


Figura 5.10: Alerta de valores mal introducidos en la app.

Envío de datos al Arduino

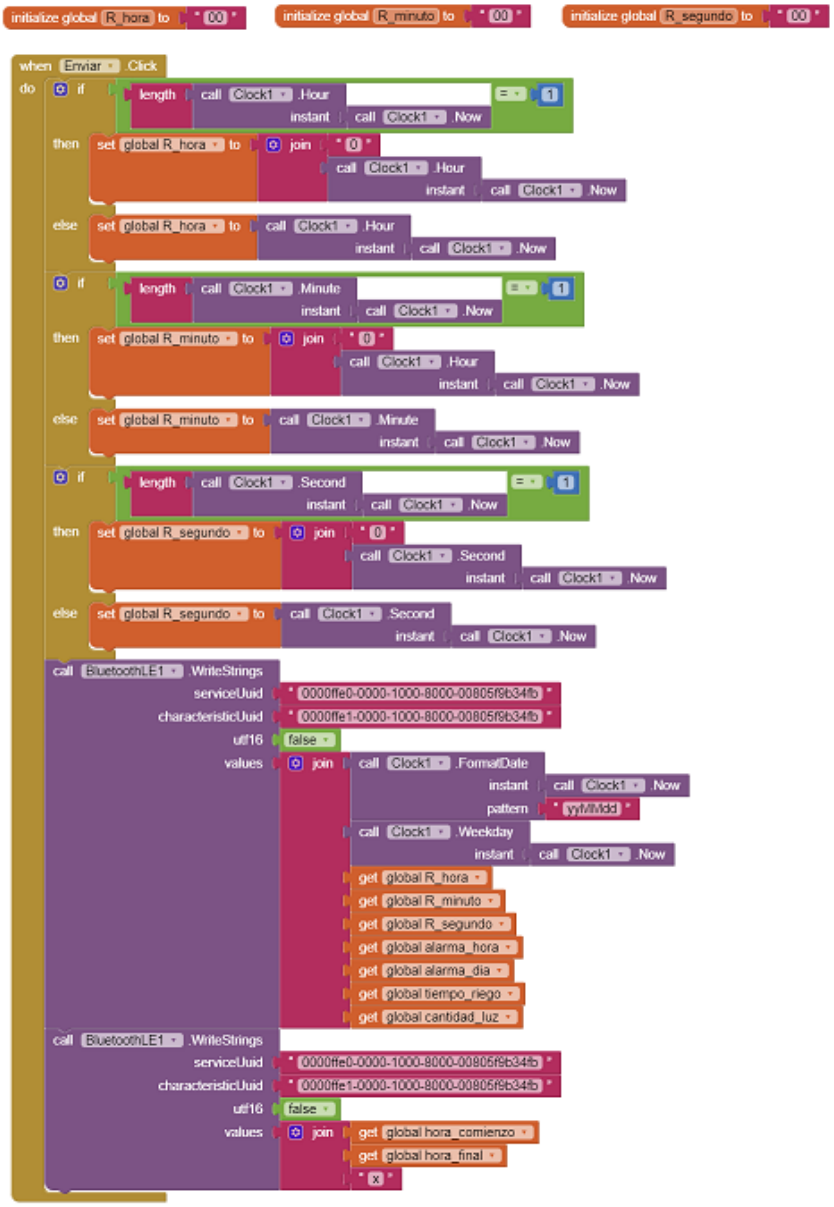


Figura 5.11: Bloques del programa encargados del envío de datos

Antes de proceder al envío de los datos se pondrán en el formato correcto. Se enviarán dos cadenas de caracteres debido a que el campo *value* tiene un tamaño máximo. La primera cadena tendrá el siguiente formato: fecha y día de la semana actual(rojo), hora actual(verde), capacidad del deposito de riego(negro), día y hora de riego (amarillo), cantidad de luz objetivo(rosa). La segunda cadena tendrá periodo de noche(azul) y la letra x(naranja).

YYMMddwHHMMSSssHHw00HHHHx

Figura 5.12: Formato cadena enviada

Para enviar los datos se necesitará saber dos Uuids del dispositivo Bluetooth. Para ello utilizaremos Ble Scanner. Es una aplicación que muestra todos los dispositivos de Bluetooth Ble y los parámetros que tienen. Los dos parámetros a utilizar serán *Custom Service* y *Custom Characteristic*.

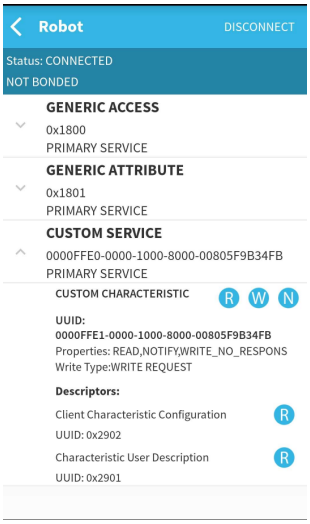


Figura 5.13: Propiedades del dispositivo Bluetooth HM-10.

6 Conclusión

Nuestro objetivo era desarrollar un robot capaz de cuidar una planta, llevarla al sitio correcto de luz y regarla cuando fuese necesario. En un primer momento se planteó la utilización de la placa de Arduino siempre en funcionamiento para tener una lectura constante de los cambios de luz y poder reaccionar al momento. Esta idea se descartó debido a que consumía las baterías y no llegaba a durar medio día. Se procedió a rebajar la reacción a los cambios de luz para así aprovechar y ahorrar batería utilizando el modo *power down* del Arduino y las interrupciones para despertarlo. Utilizando esta técnica se mejoró la duración de las baterías a un día. Con un funcionamiento correcto del robot se procedió a crear una pequeña app para poder modificar los parámetros en relación a la planta a cuidar.

6.1 Resultados

Se ha conseguido desarrollar un robot capaz de transportar una planta siguiendo la cantidad de luz del ambiente hasta llegar a la luz objetivo. Además, es capaz de evitar los obstáculos que se encuentre a su paso y es capaz de detectar en el momento del riego si se necesita regar o hay aún demasiada humedad en la tierra. También se ha conseguido configurarlo mediante Bluetooth utilizando la aplicación creada. La aplicación es funcional, sin embargo, no cuenta con la opción de pedir la información actual al robot. La única manera de informase de los valores de los sensores es mediante la pantalla lcd que se actualiza cuando el robot no esté en modo suspensión. Por último, se ha conseguido que el robot sea autónomo durante 25 horas sin embargo, no tiene mucha respuesta ante cambios sutiles de luz debido a que solo cuenta con dos sensores de luz para reducir el consumo de energía.

6.2 Mejoras futuras

La principal mejora sería crear una zona de carga de baterías y relleno de agua, para así conseguir que una vez el robot ha sido programado con las características de las necesidades de la planta pueda ser autónomo. Aprovechando que por la noche la planta no necesita luz se llevaría a la estación para así estar completamente cargado para el día siguiente.

Otras posibles mejoras sería aumentar la potencia de los motores de las ruedas, consiguiendo así poder llevar plantas de mayor peso. Por otro lado, aumentar la sensorización del robot para que detecte cambios menos significativos en la luz ayudando así a la hora de despertar el robot. Otros sensores que se podrían añadir serían medidores de humedad en el aire y temperatura.

Por último, una mejora futura sería añadir la capacidad de registrar los alrededores y tener un mapa de la zona. Este mapa a lo largo de varios días sería rellenado con datos de luz, temperatura, humedad y posibles obstáculos haciendo que el robot tuviese zonas recomendadas para la planta.

Bibliografía

[1] A. Wickert, E. Ayars, J. C. Wippler, N. W. LLC. *DS3231*. URL: https://github.com/NorthernWidget/DS3231/tree/master. (accessed: 5.04.2023) (vid. pág. 37).

[2] Andrea Lombardo, Fernando Brucher, Randy, Tomáš Roj. *L298NX2*. URL: https://github.com/AndreaLombardo/L298N/tree/master?tab=readme-ov-file#l298nx2-methods. (accessed: 16.06.2022) (vid. pág. 37).

[3] Arduino. *attachInterrupt*. URL: https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/. (accessed: 10.04.2023) (vid. pág. 10).

[4] Arduino. *LiquidCrystalI2C*. URL: https://www.arduino.cc/reference/en/libraries/liquidcrystal-i2c/. (accessed: 16.06.2022) (vid. pág. 37).

[5] Arduino. *SofwareSerial*. URL: https://docs.arduino.cc/learn/built-in-libraries/software-serial/. (accessed: 23.10.2022) (vid. pág. 37).

[6] DSD TECH. *HM-10/HM-11 DataSheet*. URL: https://wds-service-1258344699.file.myqcloud.com/20/12636/pdf/1694067402916eed7c34c55e813358278626d7271051f.pdf. (accessed: 20.04.2023) (vid. pág. 27).

[7] Elec freaks. *HC-SR04 datasheet*. URL: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf. (accessed: 10.08.2022) (vid. págs. 25, 41).

[8] David Grossman. *The Plant That Became a Six-Legged Robot To Chase the Sun*. URL: https://www.popularmechanics.com/technology/robots/a22131882/hexa-robot-plant/. (accessed: 16.06.2022) (vid. pág. 5).

[9] Harpreet Sareen, Pattie Maes. *Elowan: A plant-robot hybrid*. URL: https://www.media.mit.edu/projects/elowan-a-plant-robot-hybrid/overview/. (accessed: 6.06.2022) (vid. pág. 6).

[10] Electro Hobby. *Motor TT datasheet*. URL: https://www.electrohobby.es/motor-dc/263-motor-tt.html. (accessed: 20.07.2022) (vid. pág. 29).

[11] LIDA OPTICALI&ELECTRONIC CO., LTD. *GL5528 datasheet*. URL: https://pi.gate.ac.uk/pages/airpi-files/PD0001.pdf. (accessed: 10.08.2022) (vid. pág. 26).

[12] Raiz Madre. *MACETAS CLASICAS PLÁSTICAS*. URL: https://www.raizmadre.com/index.php?route=product/product&product_id=142. (accessed: 16.06.2022) (vid. pág. 17).

[13] Maxim Integrated. *DS3231 datasheet*. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/DS3231.pdf. (accessed: 20.10.2023) (vid. pág. 24).

[14] Mercedes Arévalo Suárez, Daniel Blanquez, Baudouin Cornelis, Kaat Leemans, Marcos Martínez Jiménez, Basile Thisse. *PLANT ROBOT*. URL: https://www.instructables.com/PLANT-ROBOT/. (accessed: 1.03.2024) (vid. pág. 7).

[15] Microchip. *ATmega640/V-1280/V-1281/V-2560/V-2561/V*. URL: https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega640-1280-1281-2560-2561-Datasheet-DS40002211A.pdf. (accessed: 16.06.2022) (vid. págs. 10, 11).

[16] Microchip. *avr/sleep*. URL: https://onlinedocs.microchip.com/pr/GUID-317042D4-BCCE-4065-BB05-AC4312DBC2C4-en-US-2/index.html?GUID-F889605B-692F-493A-8BE7-F0FBACF1715B. (accessed: 10.04.2023) (vid. pág. 37).

[17] Parrot. *Flower power*. URL: https://www.parrot.com/assets/s3fs-public/2021-09/flower-power_user-guide_uk.pdf. (accessed: 10.07.2022) (vid. pág. 4).

[18] Philipp, Dylan, Nassim, Colbish, Geoffrey, Hilary. *Plant Bot Taking Care of Plant*. URL: https://www.instructables.com/Plant-Bot-Taking-Care-of-Plant/. (accessed: 20.05.2023) (vid. pág. 7).

[19] Rambal. *Mini Bomba Sumergible DC 2.5V a 6V Brushless*. URL: https://rambal.com/bomba-valvula-cerradura/969-mini-bomba-sumergible-dc-25v-a-6v.html. (accessed: 20.07.2022) (vid. pág. 29).

[20] Sterling. *Sybil*. URL: https://www.kickstarter.com/projects/starlingsystems/sybil-the-smart-garden-robot-powered-by-machine-learning?lang=es. (accessed: 6.06.2022) (vid. pág. 3).

[21] Handson Technology. *I2C Serial Interface 1602 LCD datasheet*. URL: https://www.handsontec.com/dataspecs/module/I2C_1602_LCD.pdf. (accessed: 10.08.2022) (vid. pág. 27).

[22] Handson Technology. *L298N Motor Driver datasheet*. URL: https://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf. (accessed: 10.06.2022) (vid. pág. 23).

[23] Texas instruments. *LMx93-N, LM2903-N Low-Power, Low-Offset Voltage, Dual Comparators*. URL: https://www.ti.com/lit/ds/symlink/lm393-n.pdf?ts=1716943679840&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FLM393-N%252Fpart-details%252FLM393TL%252FNOPB. (accessed: 10.08.2022) (vid. pág. 26).

[24] Vincross. *HEXA*. URL: https://www.vincross.com/products/hexa. (accessed: 16.06.2022) (vid. pág. 5).

[25] DroneBot Workshop. *Using Arduino Interrupts – Hardware, Pin Change and Timer*. URL: https://dronebotworkshop.com/interrupts/. (accessed: 12.05.2023) (vid. pág. 9).

[26] Yi, Guyi y Di Carlo, Ilaria. "Cyborgian Approach of Eco-interaction Design Based on Machine Intelligence and Embodied Experience". En: ene. de 2021, págs. 79-90. ISBN: 978-981-33-4399-3. DOI: 10.1007/978-981-33-4400-6_8. (accessed: 6.06.2022) (vid. pág. 6).

[27] Yoolax. *Yoolax Smart Plant Pot*. URL: https://www.yoolax.com/products/yoolax-smart-plant-pot?variant=41640273576076. (accessed: 1.06.2022) (vid. pág. 5).

7 Anexo

7.1 Código

7.1.1 Robot

```
#include <L298NX2.h>
#include <DS3231.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>

#define ENA  6       |
#define IN1A 41      | Pines control motor izquierda
#define IN2A 40      |
#define IN1B 51          |
#define IN2B 50          | Pines control motor derecha
#define ENB 7            |

#define ECHO 8         | Sensor ultrasonidos
#define TRIG 9         |

#define LIGHTLA A10      | Sensores de luz
#define LIGHTRA A11      |

#define HUM A3           | Sensor humedad
#define HUMACT 12        | Control alimentación

#define RXD 10           | Pines HM-10
#define TXD 11           |

```

```
28 #define RTC 18           | Interrupción RTC
29 #define LIGHT 19         | Interrupción sensor luz 1
30 #define LIGHT2 2         | Interrupción sensor luz 2
31
32 #define RELE1 28         | Control relé sensores
33 #define RELE2 29         | Control relé para bomba agua
34
35  // Variación de luz a la hora de encontrar el sitio
       indicado
36 const int lightMargin = 10;
37
38 // Máximo de humedad para poder regar, por encima no se
     riega
39 const int humidityMax = 20;
40
41 // Distancia mínima a la que puede estar el robot antes de
     evitar el choque
42 const int distMin = 40;
43
44 int lightSensorR, lightSensorL,lightL, lightR ;
45 double light, dist, humidity;
46 long timeUltrasonicSensor = 0;
47 bool flagHumidity, flagConfiguration, flagSleep = false;
48 bool flagDay = true;
49
50 String week= "-------";
51
52 DS3231 myRTC;
53
54 byte year, month, date, dOW, hour, minute, second,
55      hourA, dOWA, dOWB,
56      cap, plantLightNeeds, beginNightH, endNightH;
57
58
59 L298NX2 motors(ENA, IN1A, IN2A, ENB, IN1B, IN2B);
60 LiquidCrystal_I2C lcd(0x3F ,16,2);
61
62 SoftwareSerial mySerial(RXD, TXD); // RX, TX
63
64 /*
65  *  Obtener la distancia a la que se encuentra el robot de
       los posibles obstáculos
66  */
67 void getDistance(){
68
69   digitalWrite(TRIG ,HIGH);
70   delayMicroseconds(10);
```

```
71    digitalWrite(TRIG , LOW);
72
73    timeUltrasonicSensor = (pulseIn(ECHO , HIGH));
74
75    dist = float(timeUltrasonicSensor /58);
76  }
77
78
79  /*
80   * Método de movimiento
81   *  Se pueden dar 5 casos
82   *    1. La luz que recibe los sensores de la planta es la luz que necesita +- margen.
83   *       Se pondrá la flagSleep a true para que el robot se duerma
84   *    2. Hay obstaculos, por lo tanto se irá marcha atrás.
85   *    3. La luz que recibe por el sensor de la derecha está mas cerca de la luz que necesita. Girará en esa dirección.
86   *    4. La luz que recibe por el sensor de la izquierda está mas cerca de la luz que necesita. Girará en esa dirección.
87   *    5. Ambos sensores reciben la misma cantidad de luz y no es aun la luz que necesita +-margen.
88   */
89  void movementLogic(){
90   int aux =  (abs(plantLightNeeds - lightR) <= lightMargin) && (abs(plantLightNeeds - lightL) <= lightMargin);
91   if (aux){
92      Serial.println("Buen sitio");
93      flagSleep = true;
94
95    }else if (dist < distMin && dist > 0){
96      Serial.println("Choque");
97       motors.setSpeed(200);
98       motors.backward();
99       delay(600);
100
101      if(flagTurn){
102        motors.setSpeedB(255);
103        motors.setSpeedA(200);
104        motors.forwardA();
105        motors.backwardB();
106      }else{
107        motors.setSpeedB(200);
108        motors.setSpeedA(255);
109        motors.forwardB();
```

```
110         motors.backwardA();
111       }
112
113       delay(600);
114       motors.stop();
115   }
116   else if(abs(plantLightNeeds - lightR) > abs(
          plantLightNeeds - lightL)){
117     Serial.println("Hay mas luz sensor Derecha");
118     flagTurn = false;
119     motors.setSpeedB(255);
120     motors.setSpeedA(200);
121     motors.forwardB();
122     motors.backwardA();
123     delay(650);
124     motors.forward();
125     delay(300);
126     motors.stop();
127
128   }
129   else if(abs(plantLightNeeds - lightR) < abs(
          plantLightNeeds - lightL)){
130     Serial.println("Hay mas luz sensor Izquierda");
131     flagTurn = true;
132     motors.setSpeedA(255);
133     motors.setSpeedB(200);
134     motors.forwardA();
135     motors.backwardB();
136     delay(650);
137     motors.forward();
138     delay(300);
139     motors.stop();
140
141   }
142   else {
143     Serial.println("Hacia delante");
144     motors.setSpeed(200);
145     motors.forward();
146     delay(500);
147     motors.stop();
148
149   }
150 }
151
152 /*
153  * Función que se llama al salir del modo Power down
154  */
```

```
155 void wakeUp(){
156   sleep_disable();
157   flagSleep = false;
158 }
159
160 /*
161  * Función que mide la humedad de la tierra, para evitar
       que el sensor
162  * deje de funcionar en poco tiempo solo estará activo a la
       hora de la
163  * medición.
164  */
165 void getHumidity(){
166   digitalWrite(HUMACT , HIGH);
167   humidity = analogRead(HUM );
168   digitalWrite(HUMACT , LOW);
169   humidity = abs (humidity/10);
170 }
171
172 /*
173  * Función encargada de regar la planta. Primero pedirá la
       medición de
174  * humedad, depues comparará la humedad con la humedad
       maxima en el caso
175  * de que sea mayor no regará pues hay demasiada agua en la
       tierra.
176  * Modificará la flagHumidity para mostrar la alarma.
177  */
178 void waterPlant(){
179
180   getHumidity();
181   if ( humidity < humidityMax){
182     flagHumidity = 0;
183     Serial.println("RIEGO");
184     digitalWrite(RELE2,HIGH);
185     delay((cap/100)*10);
186     digitalWrite(RELE2,LOW);
187   }else{
188     flagHumidity = 1;
189   }
190
191 }
192
193 /*
194  * Método para actualizar los parámetros mostrados en la
       lcd
195  * En el caso que la flagHumidity sea cierta modificará el
```

```
196  * apartado de días por "Cambiar" para notificar que hay
       que
197  * cambiar el riego.
198  */
199 void showInfo(){
200   lcd.clear();
201   lcd.setCursor(0, 0);
202   lcd.print("L:"+(String)light +"%␣Riego␣" );
203   lcd.setCursor(0,1);
204   lcd.print("H:"+ (String)humidity + "%␣" + week);
205   lcd.setCursor(9,1);
206   if (flagHumidity){
207       lcd.print("Cambiar");
208   }
209
210 }
211
212 /*
213  * Método maestro encargado de leer los sensores de luz y
       todos los demás sensores
214  */
215 void logicRobot(){
216   lightSensorR = analogRead(LIGHTRA);
217   lightSensorL = analogRead(LIGHTLA);
218
219   lightR = map(lightSensorR, 1023, 0, 0, 100); // 0 maxima
       luz
220   lightL = map(lightSensorL, 1023, 0, 0, 100); // 1023
       oscuridad
221   light = (lightR + lightL )/2;
222
223   getHumidity();
224   showInfo();
225   getDistance();
226   movementLogic();
227
228 }
229
230 /*
231  * Método para crear y modificar la Alarma 2
232  *  Parámetros:
233  *    dayA2          : dia
234  *    hourA2         : hora
235  *    minuteA2       : minuto
236  *    typeA          : que tipo de alarma: cada segundo,
       minuto, hora, dia, semana, mes o año
```

```
237  *    dayOrWeekDay  : el parámetro dayA2 hace referencia a
       los dias del mes o los días de la semana
238  *    h12H24        : el parámetro hourA2 está en formato
       12h o 24h
239  *    pmAm          : en el caso de estar en formato 12h si
        la hora es AM o PM
240  */
241 void setRTCAlarm2(byte dayA2, byte hourA2, byte typeA, bool
      dayOrWeekDay, bool h12H24, bool pmAm){
242
243   myRTC.setA2Time(dayA2, hourA2, 0, typeA, dayOrWeekDay,
        h12H24, pmAm);
244   myRTC.turnOnAlarm(2);
245   myRTC.checkIfAlarm(2);
246
247 }
248
249 /*
250  * Método para poner en fecha y hora el RTC como las dos
       alarmas
251  */
252 void setRTC(byte year, byte month, byte date, byte dOW,
     byte hour, byte minute, byte second,
253             byte hourA, byte dOWA){
254
255         myRTC.setClockMode(false);
256         myRTC.setYear(year);
257         myRTC.setMonth(month);
258         myRTC.setDate(date);
259         myRTC.setDoW(dOW);
260         myRTC.setHour(hour);
261         myRTC.setMinute(minute);
262         myRTC.setSecond(second);
263
264         myRTC.setA1Time(dOWA, hourA, 0, 0, 0x0, true,
265                         false, false);
266         myRTC.turnOnAlarm(1);
267         myRTC.checkIfAlarm(1);
268
269         byte aux = 0b00001100;
270         setRTCAlarm2(dOWB, hourA, aux, true, false, false);
271 }
272
273 /*
274  * Método para convertir los chars a bytes
275  */
276 byte convertToByte(char a, char b){
```

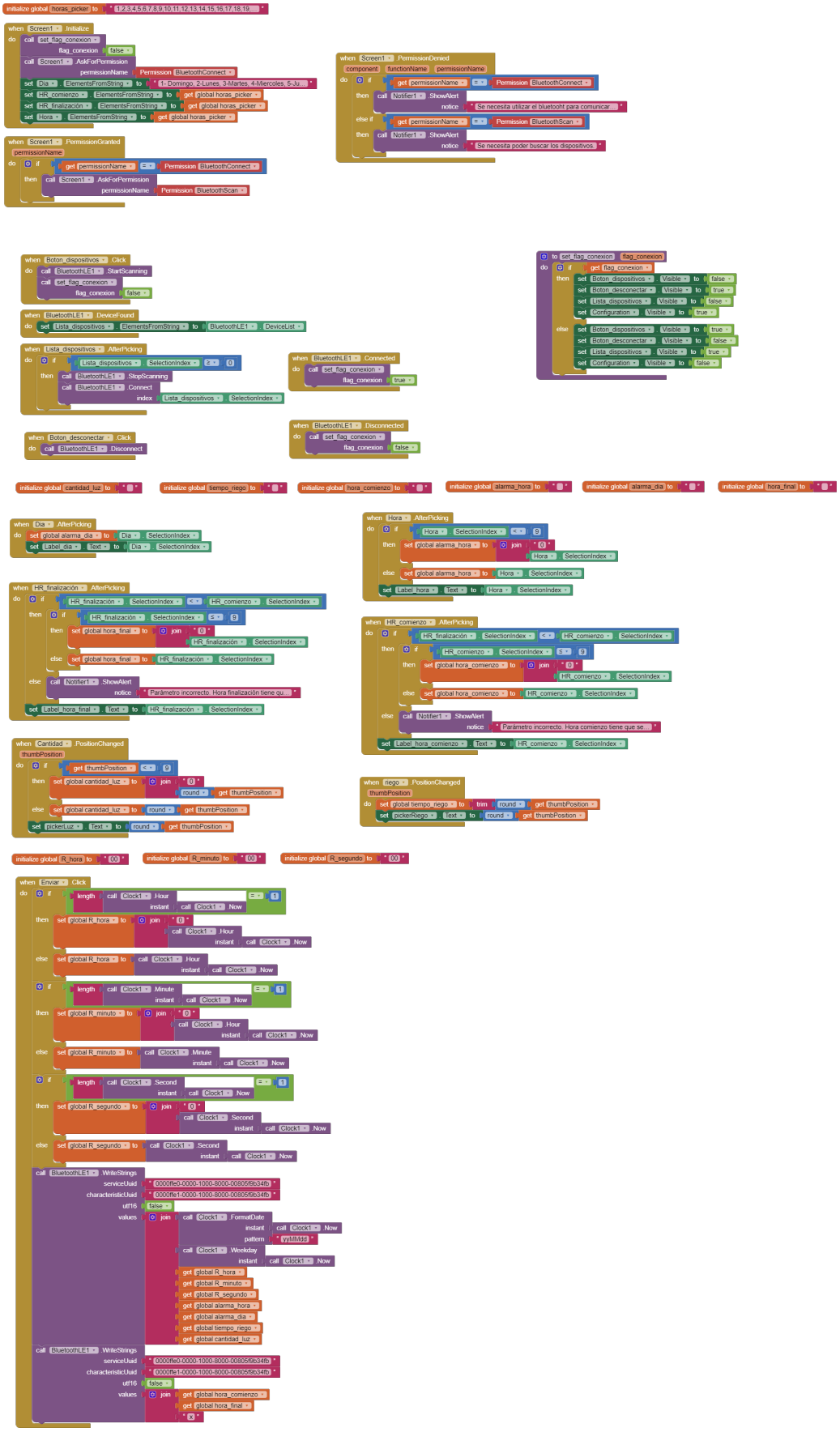
```
277   byte temp1, temp2;
278
279   temp1 = (byte)a -48;
280   temp2 = (byte)b -48;
281
282   return temp1*10 + temp2;
283 }
284
285
286 /*
287  * Método prinicpal para la configuración del robot.
288  * Guardará la información recibida por el puerto serie,
289  * en el caso que la información recibida no acabe con x
290  * se tomará como falladia el envio de información y se
291  * mostrará por pantalla.
292  */
293 void configureRobot(){
294   char inChar;
295   char inString[25];
296   int i = 0;
297   while (mySerial.available()) {
298         inChar = mySerial.read();
299          Serial.println(inChar);
300         inString[i] = inChar;
301         i++;
302   }
303   if (inString[24] == 'x'){
304     year = convertToByte(inString[0],inString[1]);
305     month = convertToByte(inString[2],inString[3]);
306     date = convertToByte(inString[4],inString[5]);
307     dOW = (byte)inString[6] - 48;
308     hour = convertToByte(inString[7],inString[8]);
309     minute = convertToByte(inString[9],inString[10]);
310     second = convertToByte(inString[11],inString[12]);
311
312     hourA = convertToByte(inString[13],inString[14]);
313     dOWA = (byte)inString[15] - 48;
314
315     cap = convertToByte(inString[16],inString[17]);
316     plantLightNeeds = convertToByte(inString[18],inString
          [19]);
317     beginNightH = convertToByte(inString[20],inString[21]);
318     endNightH = convertToByte(inString[22],inString[23]);
319
320     setRTC(year, month, date, dOW, hour, minute, second,
          hourA, dOWA);
321     showInfo();
```

```
322     flagConfiguration = true;
323     delay(1000);
324     attachInterrupt(digitalPinToInterrupt(RTC), wakeUp,
          FALLING);
325     attachInterrupt(digitalPinToInterrupt(LIGHT), wakeUp,
          FALLING);
326     attachInterrupt(digitalPinToInterrupt(LIGHT2), wakeUp,
          LOW);
327
328     week.setCharAt(dOWA-1, 'R');
329   }
330   else{
331       lcd.clear();
332       lcd.setCursor(0, 0);
333       lcd.print("Fallo␣en␣la");
334       lcd.setCursor(0, 1);
335       lcd.print("configuración");
336   }
337 }
338
339 /*
340  * Declaración de los pines y inicialización de la
        comunicación serial y lcd
341  */
342 void setup(){
343   Serial.begin(9600);
344   mySerial.begin(115200);
345
346   motors.setSpeed(100);
347
348   pinMode(RTC, INPUT_PULLUP);
349   pinMode(LIGHT, INPUT);
350   pinMode(LIGHT2, INPUT);
351   pinMode(LIGHTLA , INPUT);
352   pinMode(LIGHTRA , INPUT);
353   pinMode(HUMACT , OUTPUT);
354   pinMode(HUM , INPUT);
355   pinMode(TRIG , OUTPUT);
356   pinMode(ECHO , INPUT);
357   pinMode(ENA, OUTPUT);
358   pinMode(IN1A, OUTPUT);
359   pinMode(IN2A, OUTPUT);
360   pinMode(ENB, OUTPUT);
361   pinMode(IN1B, OUTPUT);
362   pinMode(IN2B, OUTPUT);
363   pinMode(RELE1, OUTPUT);
364   pinMode(RELE2, OUTPUT);
```

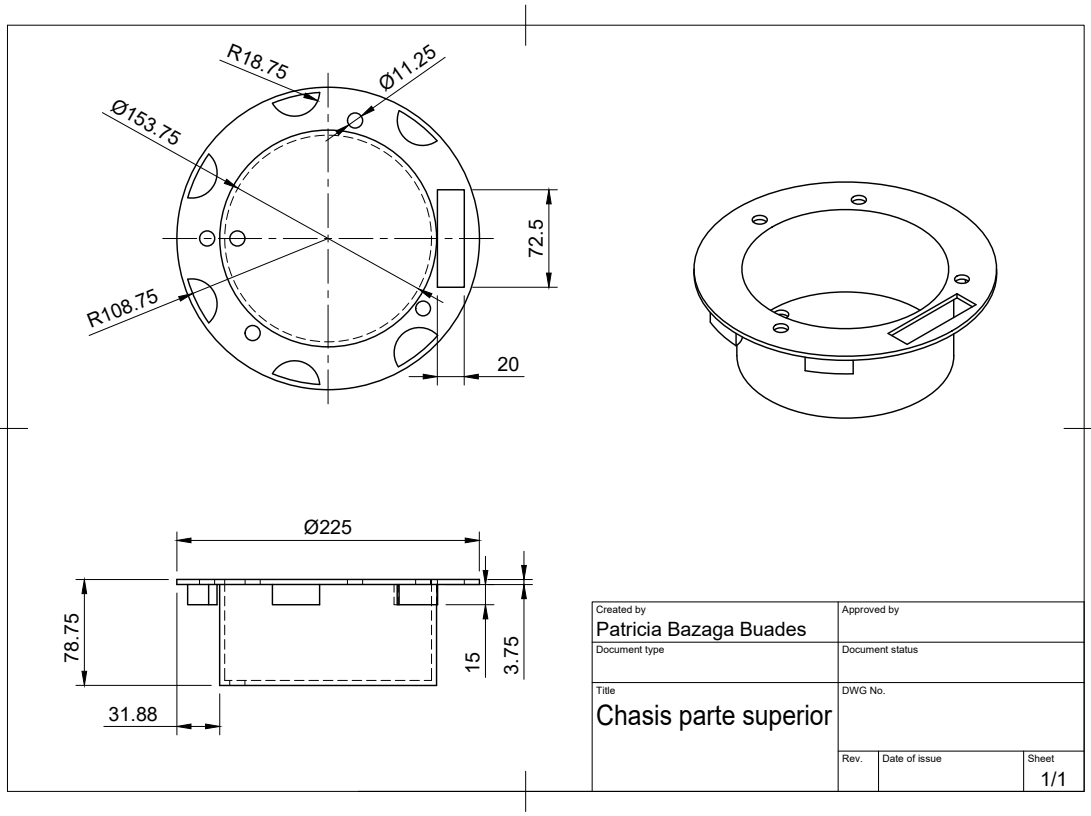
```
365
366   lcd.init();
367   lcd.backlight();
368   lcd.print("No config");
369   lcd.setCursor(0, 1);
370   lcd.print("Introduzca");
371
372   digitalWrite(RELE1,HIGH);
373   }
374
375 void loop(){
376   // En el caso de que el robot esté configurado empezará a
          cuidar la planta.
377   if (flagConfiguration){
378     if(flagDay){
379       logicRobot();
380     }
381     // Si se ha encontrado el sitio se pondrá en modo power
          down
382     if (flagSleep){
383       Serial.println("SLEEP");
384       digitalWrite(RELE1,LOW);
385       lcd.noBacklight();
386       set_sleep_mode(SLEEP_MODE_PWR_DOWN);
387       sleep_enable();
388       Serial.println("next line will put Arduino in sleep
            mode");
389       delay(100);
390       sleep_mode();
391       delay(100);
392
393       /*
394        * En el caso de que se despierte por las alarmas del
              RTC se mirará por cual.
395        * Alarma 1: asegurarse que no es de noche y si es de
              noche dormir hasta la mañana sigueinte
396        * Alarma 2: regar
397        */
398       if (myRTC.checkIfAlarm(1)){
399         waterPlant();
400       }
401       bool auxBool = false;
402       byte theHour = myRTC.getHour(auxBool, auxBool);
403       if((theHour >= beginNightH || theHour <= endNightH)){
404           flagDay = false;
405           flagSleep = true;
406       }else{
```

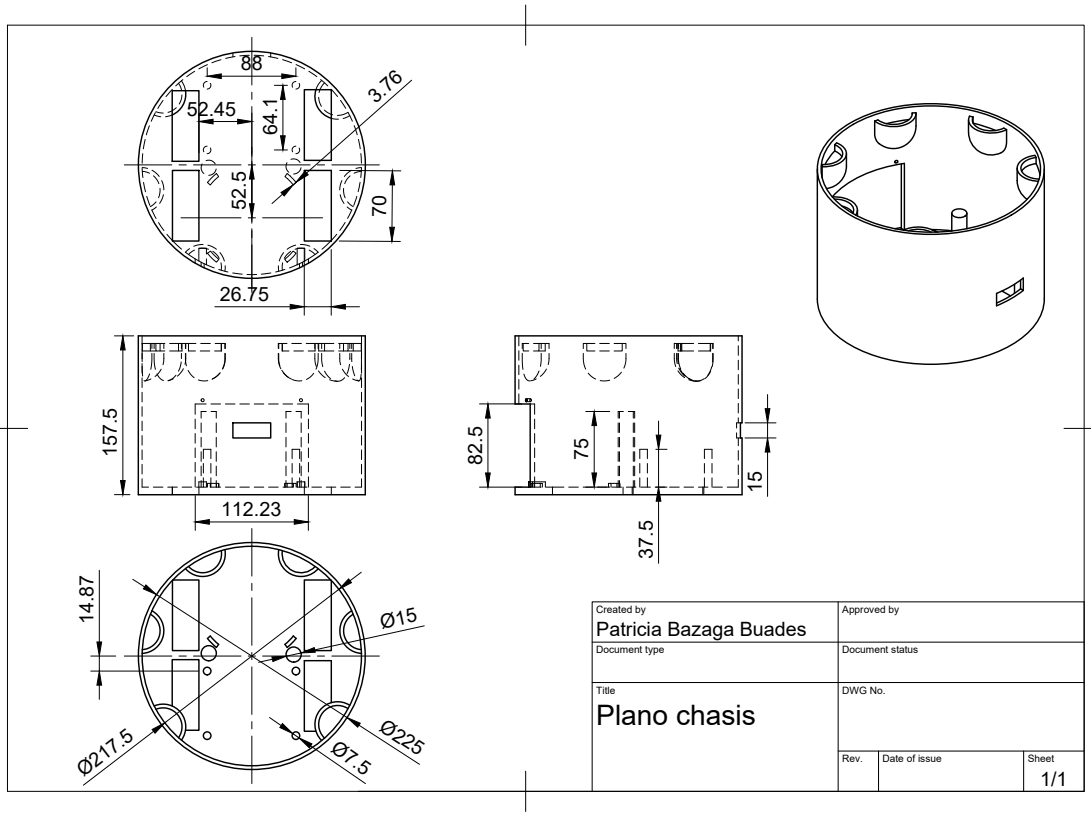
```
407             flagDay = true;
408             flagSleep = false;
409             digitalWrite(RELE1,HIGH);
410             lcd.backlight();
411         }
412       }
413   }
414   // En el caso de que se reciba información por bluetooth.
415   if (mySerial.available() > 0 ){
416     flagConfiguration = false;
417     configureRobot();
418   }
419   delay(1000);
420 }
```

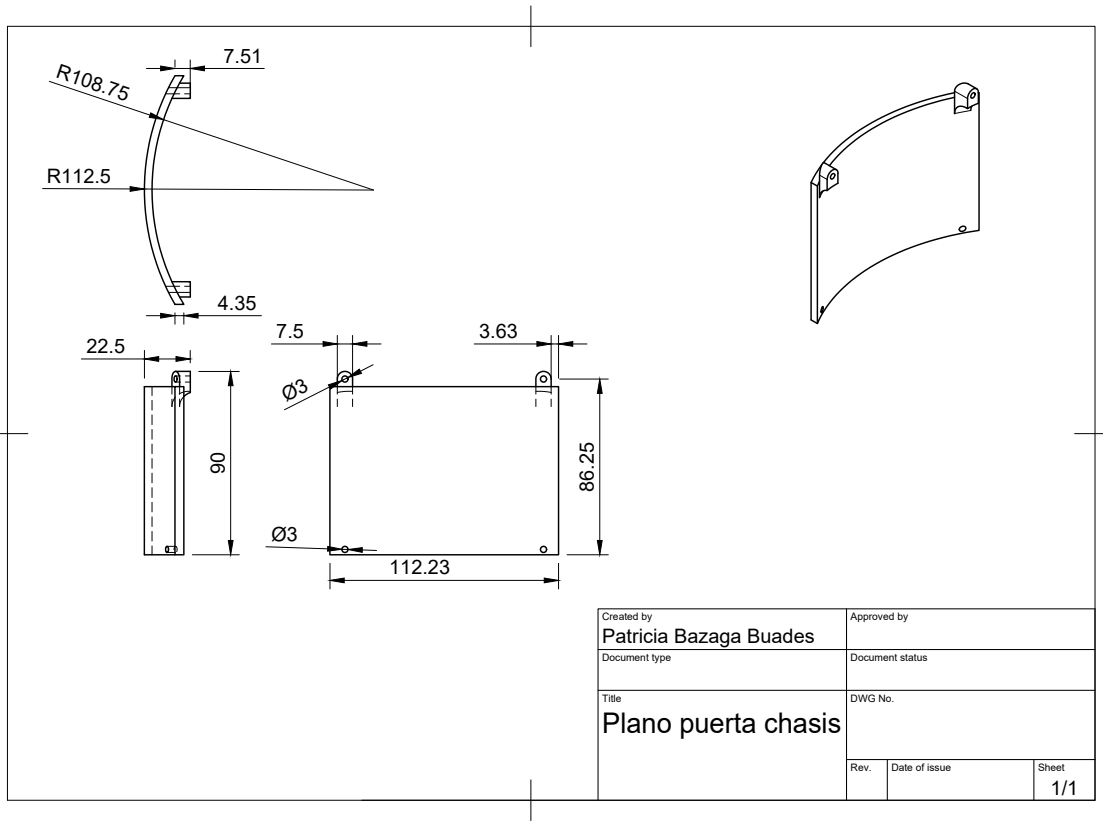
7.1.2 App



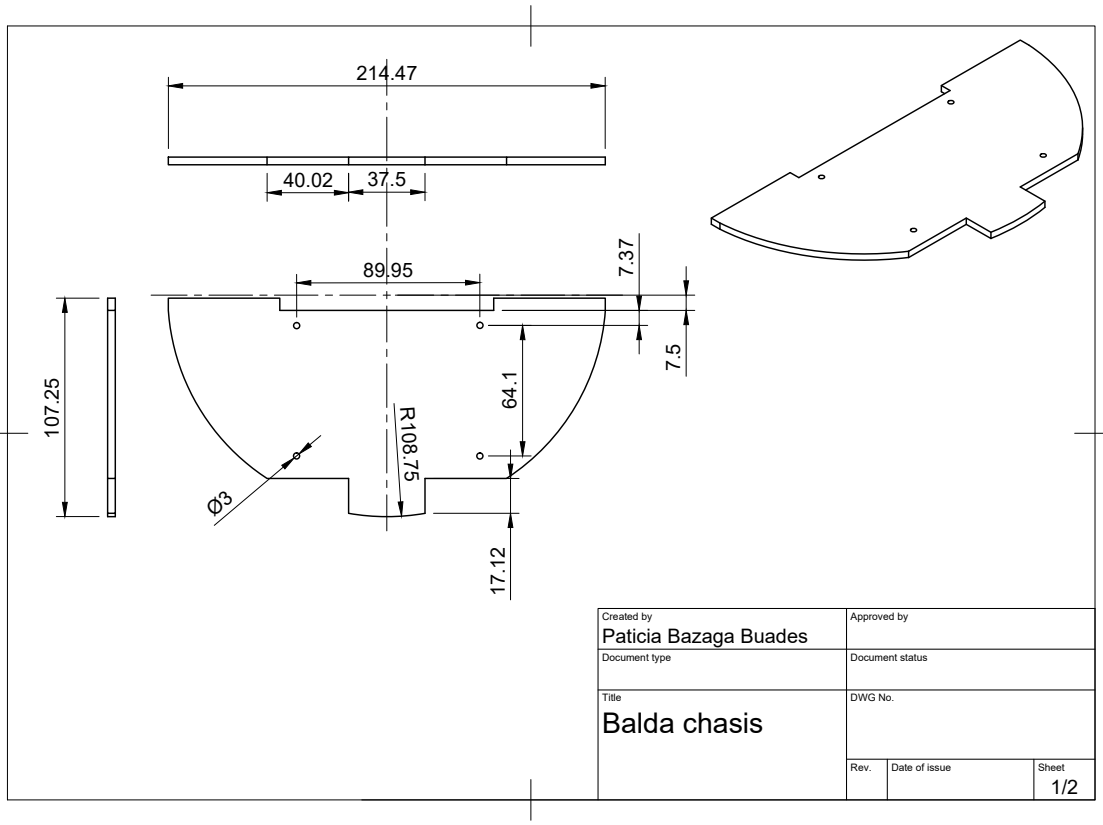
7.2 Planos


R18.75
Ø11.25
Ø153.75
R108.75
72.5
20
Ø225
78.75
15
3.75
31.88
Created by
Patricia Bazaga Buades
Approved by
Document type
Document status
Title
Chasis parte superior
DWG No.
Rev.
Date of issue
Sheet
1/1

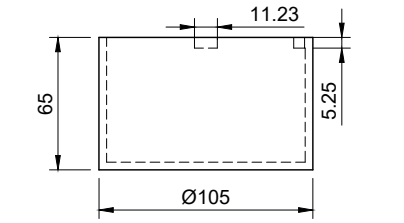

88
52.45
64.1
3.76
52.5
70
26.75
157.5
112.23
82.5
75
15
37.5
14.87
Ø15
Ø217.5
Ø225
Ø7.5
Created by
Patricia Bazaga Buades
Approved by
Document type
Document status
Title
Plano chasis
DWG No.
Rev.
Date of issue
Sheet
1/1

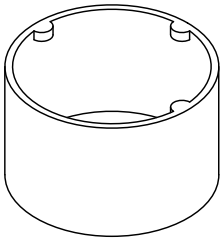


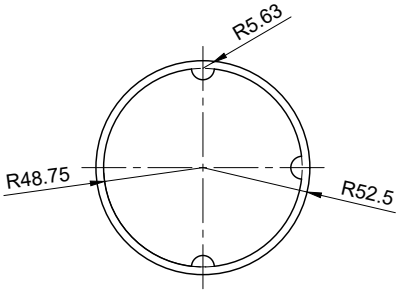
Created by: Patricia Bazaga Buades	Approved by:
Document type:	Document status:
Title: Plano puerta chasis	DWG No.:
	Rev. / Date of issue / Sheet: 1/1



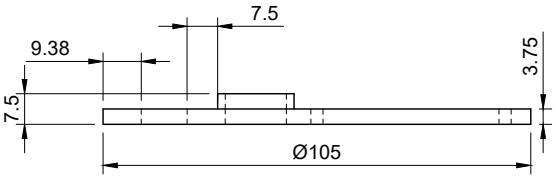
Created by: Paticia Bazaga Buades	Approved by:
Document type:	Document status:
Title: Balda chasis	DWG No.:
	Rev. / Date of issue / Sheet: 1/2

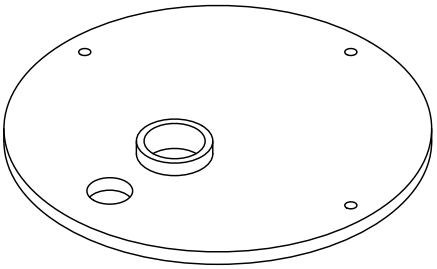


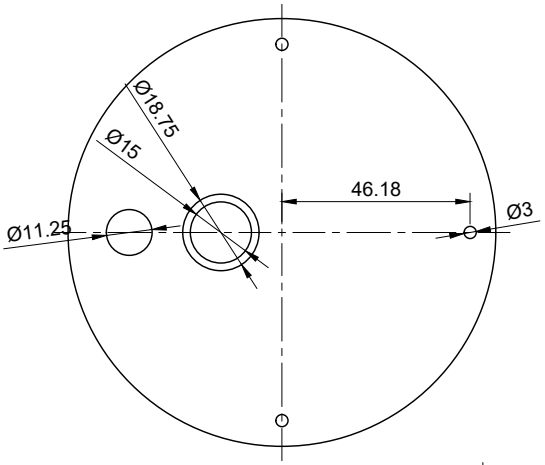




Created by: Patricia Bazaga Buades	Approved by:
Document type:	Document status:
Title: Plano depósito	DWG No.:
Rev. / Date of issue	Sheet: 1/1







Created by: Patricia Bazaga Buades	Approved by:
Document type:	Document status:
Title: Tapa depósito	DWG No.:
Rev. / Date of issue	Sheet: 1/1

7.3 Esquema eléctrico