Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Control y supervisión mediante un sistema microcontrolador de los parámetros de calidad de agua de un estanque

Autor: Antonio Pérez Laguarda

Tutor: Juan de la Cruz García Ortega

Departamento de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Control y supervisión mediante un sistema microcontrolador de los parámetros de calidad de agua de un estanque

Autor:
Antonio Pérez Laguarda

Tutor:
Juan de la Cruz García Ortega
Profesor titular

Departamento de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2017

Trabajo Fin de Grado: Control y supervisión mediante un sistema microcontrolador de los parámetros de calidad de agua de un estanque

Autor: Antonio Pérez Laguarda

Tutor: Juan de la Cruz García Ortega

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

A mi familia y amigos

A mis profesores

Resumen

En la actualidad, existen multitud de personas interesadas en el mundo de los acuarios marinos. La mayoría de estas personas realizan un control manual de ciertos parámetros de calidad del agua o de la iluminación, es decir, se necesita de una persona para poder modificar y conocer parámetros tales como la temperatura y pH del agua, nivel de llenado del acuario, o intensidad y horas de iluminación.

Existen sistemas comerciales en el mercado cuya función es el control del acuario sin la necesidad de la actuación de una persona. El problema de estos controladores comerciales es su elevado precio, que no está al alcance de la mayoría.

Por este motivo, se propone la creación de un controlador de acuarios con una funcionalidad similar a los comerciales pero mucho más económico.

A lo largo del proyecto se estudiarán los aspectos teóricos necesarios, se diseñará y se realizará el montaje del sistema, se programará el código necesario para el microcontrolador Arduino y se realizarán una serie de tests que demuestren su correcto funcionamiento.

Abstract

Nowadays, many people are interested in aquariums. Most of them use their own hands to control it, in other words, it is necessary someone who amends and understands some settings like the temperature, the capacity of the aquarium, the illumination or the pH among other aspects.

In the market, there are commercial systems to control all of those settings, but, their high price reduces the possibility of acquisition to many people.

That is why it is proposed the creation of an aquarium controller similar to the ones that are marketed but more cost-effectively.

Throughout the project, necessary theoretical aspects are going to be studied, I will design and build the system, I will plan the necessary code to the Arduino microcontroller and will be applying a number of tests to check it.

Índice

Resumen **ix**

Abstract **xi**

Índice **xii**

Índice de Tablas **xiv**

Índice de Figuras **xv**

Notación **xvii**

1 Introducción **1**
1.1 Motivación *1*
1.2 Objetivos *1*

2 Estado del Arte **3**

3 Parámetros del Agua **7**
3.1 PH *7*
3.2 Conductividad *8*
3.3 Temperatura *9*
3.4 Dureza *9*
3.5 CO_2 *10*

4 Requerimientos **11**

5 Estudios previos **13**

6 Desarrollo Hardware **17**
6.1 Elección del microcontrolador *17*
6.2 Arduino *17*
6.3 Placas Arduino *18*
6.4 Arduino Mega *23*
6.5 Sensores *24*
6.5.1 Sensor de temperatura 24
6.5.2 Sensor nivel de agua 26
6.5.3 Sensor de pH 28
6.5.4 Sensor de conductividad 31
6.6 Otros elementos del montaje *33*
6.6.1 Teclado 4x4 33
6.6.2 Display LCD 16x2 34
6.7 Actuadores *35*
6.7.1 Resistencia calefactora 35
6.7.2 Ventilador 36
6.7.3 Iluminación 37
6.7.4 Bomba de llenado y bomba de vaciado 38
6.7.5 Válvula de CO_2 39
6.8 Montaje de las salidas del sistema *41*
6.9 Montaje final del sistema *45*
6.10 Diseño del PCB *46*

6.10.1 Proceso de fabricación del PCB 48
6.10.2 Montaje de los componentes 48

7 Desarrollo Software 51
7.1 IDE Arduino 51
7.2 Programa de control utilizado 53
7.2.1 Funciones creadas 53
7.2.2 Funciones usadas de las librerías 61

8 Pruebas realizadas 69
8.1 Prueba 1. Correcto encendido del sistema 69
8.1.1 Descripción general 69
8.1.2 Procedimiento detallado para la comprobación 69
8.1.3 Resultado 69
8.2 Prueba 2. Activación correcta de la iluminación 69
8.2.1 Descripción general 69
8.2.2 Procedimiento detallado para la comprobación 69
8.2.3 Resultado 69
8.3 Prueba 3. Activación correcta de la electroválvula de CO_2 69
8.3.1 Descripción general 69
8.3.2 Procedimiento detallado para la comprobación 70
8.3.3 Resultado 70
8.4 Prueba 4. Lectura correcta de los potenciómetros que simulan el sensor de pH y el sensor de conductividad 70
8.4.1 Descripción general 70
8.4.2 Procedimiento detallado para la comprobación 70
8.4.3 Resultado 70
8.5 Prueba 5. Activación o desactivación de la resistencia calefactora y del ventilador dependiendo de la temperature medida por el sensor DS18B20 71
8.5.1 Descripción general 71
8.5.2 Procedimiento detallado para la comprobación 71
8.5.3 Resultado 72
8.6 Activación o desactivación de la bomba de llenado y de la bomba de vaciado dependiendo de las señales leídas por las boyas de nivel 72
8.6.1 Descripción general 72
8.6.2 Procedimiento detallado para la comprobación 72
8.6.3 Resultado 72
8.7 Visualización correcta del menú por el display LCD 72
8.7.1 Descripción general 72
8.7.2 Procedimiento detallado para la comprobación 72
8.7.3 Resultado 73
8.8 Lectura correcta del teclado matricial 4x4 por parte de Arduino 73
8.8.1 Descripción general 73
8.8.2 Procedimiento detallado para la comprobación 73
8.8.3 Resultado 73
8.9 Cuadro resumen de las pruebas realizadas 73

9 Conclusiones y posibles mejoras 75
9.1 Conclusiones 75
9.2 Posibles mejoras 75

Referencias 77

Anexos 79
A. Comunicación I2C 79

Glosario **81**

Índice de Tablas

Tabla 3-1. Escala de valores del pH 8
Tabla 3-2. Valores de la conductividad en diferentes aguas 9
Tabla 3-3. Rangos de temperatura en diferentes tipos de agua 9
Tabla 3-4. Clasificación del agua en función del nivel de dureza 10
Tabla 3-5. Escalas de dureza 10
Tabla 6-1. Clasificación de las placas Arduino según características 18
Tabla 6-2. Características principales Arduino Mega 23
Tabla 6-3. Características sensor de temperatura DS18B20 25
Tabla 6-4. Relación resolución-precisión 25
Tabla 6-5. Relación temperatura-salida digital 26
Tabla 6-6. Características principales sensor de pH 28
Tabla 6-7. Relación tensión-valor de pH 29
Tabla 6-8. Características principales sensor de conductividad 31
Tabla 6-9. Clasificación intensidad de la luz 37
Tabla 7-1. Funciones de la librería Wire.h y su descripción 62
Tabla 7-2. Funciones de la librería LCD.h y su descripción 62
Tabla 7-3. Funciones de la librería LyquidCrystal_I2C.h y su descripción 63
Tabla 7-4. Funciones de la librería stdlib.h y su descripción 64
Tabla 7-5. Funciones de la librería Keypad.h y su descripción 64
Tabla 7-6. Funciones de la librería OneWire.h y su descripción 65
Tabla 7-7. Funciones de la librería DallasTemperature.h y su descripción 65
Tabla 8-1. Tabla resumen de las pruebas realizadas 73

Índice de Figuras

Figura 2-1. Acuario plantado 3
Figura 2-2. Sistema comercial AT-Control 3
Figura 2-3. Sistema comercial ProfiLux. 4
Figura 2-4. Placa Ferduino. 4
Figura 2-5. Montaje placa Jarduino. 5
Figura 5-1. Entradas y salidas al microcontrolador 15
Figura 6-1. Arduino UNO 19
Figura 6-2. Arduino 101 19
Figura 6-3. Arduino Esplora 19
Figura 6-4. Arduino Leonardo 20
Figura 6-5. Arduino Mega 20
Figura 6-6. Arduino Mega ADK. 20
Figura 6-7. Arduino Due 21
Figura 6-8. Arduino Zero 21
Figura 6-9. Arduino Ethernet 21
Figura 6-10. Arduino Yun 22
Figura 6-11. Arduino Gemma. 22
Figura 6-12. Arduino Lilypad 22
Figura 6-13. Materia 101 23
Figura 6-14. Sensor de temperatura DS18B20 24
Figura 6-15. Conexión sensor DS18B20 a Arduino 24
Figura 6-16. Boya de nivel 26
Figura 6-17. Esquema montaje boyas de nivel 27
Figura 6-18. Sensor de ultrasonido HC-SR04 27
Figura 6-19. Esquema de conexión sensor de pH de DFRobot 28
Figura 6-20. Esquemático conexión potenciómetro 30
Figura 6-21. Esquemático conexión potenciómetro con resistencia en serie 30
Figura 6-22. Esquema de conexión sensor de conductividad 31
Figura 6-23. Pinout teclado 4x4 33
Figura 6-24. Conexión teclado 4x4 a Arduino 34
Figura 6-25. Pinout display LCD 16x2 34
Figura 6-26. Modulo I2C del display LCD 16x2 35
Figura 6-27. Resistencia calefactora 36

Figura 6-28. Montaje cable calefactor 36
Figura 6-29. Opciones para disminuir la temperatura en un acuario 37
Figura 6-30. Panel iluminación LED 38
Figura 6-31. Bomba de agua sumergible 38
Figura 6-32. Bombona CO_2 39
Figura 6-33. Manoreductor con manómetro 39
Figura 6-34. Válvula de retención 40
Figura 6-35. Cuentagotas 40
Figura 6-36. Electroválvula 41
Figura 6-37. Elementos de un relé 41
Figura 6-38. Esquemático de un relé 42
Figura 6-39. Circuito necesario para la conexión de un relé a Arduino 42
Figura 6-40. Elementos de un módulo de cuatro relés 43
Figura 6-41. Esquemático de un canal del módulo de cuatro relés 43
Figura 6-42. Conexión del módulo de cuatro relés a Arduino usando una fuente externa 44
Figura 6-43. Montaje final del sistema usando placa de pruebas 45
Figura 6-44. Esquemático completo en Eagle 46
Figura 6-45. Diseño layout del circuito. 47
Figura 6-46. Diseño layout del circuito mostrando solo vías y pads 47
Figura 6-47. Diseño layout del circuito con el plano de tierra añadido 48
Figura 7-1. Interfaz gráfica del IDE Arduino 51
Figura 7-2. Menú del IDE de Arduino 52
Figura 7-3. Menú de acceso rápido del IDE de Arduino 52
Figura 7-4. Editor de texto del IDE de Arduino. 52
Figura 7-5. Mensaje de compilación del IDE de Arduino 53
Figura 7-6. Mensaje de error del IDE de Arduino 53
Figura 7-7. Consola del IDE de Arduino 53
Figura 7-8. Diagrama de flujo del menú de inicio 54
Figura 7-9. Diagrama de flujo del "modo lectura" 54
Figura 7-10. Diagrama de flujode la función "lecturaGeneral" 56
Figura 7-11. Diagrama de flujo del menú "fijar parámetros" 57
Figura 7-12. Diagrama de flujo del menú "mostrar parámetros" 58
Figura 7-13. Diagrama de flujo genérico para fijar parámetros 59
Figura 7-14. Diagrama de flujo para la lectura de un potenciómetro 60
Figura 7-15. Diagrama de flujo para la lectura de las boyas de nivel 60
Figura 7-16. Código de ejemplo del sensor de temperatura DS18B20 61
Figura 7-17. Diagrama de flujo para la lectura del monitor serie 61
Figura A-1. Esquema funcionamiento del bus I2C 79
Figura A-2. Ejemplo de funcionamiento del bus I2C 80

Notación

pH	Potencial de hidrógeno
µS/cm	Microsiemens por centímetros
ºC	Grados centígrados
S	Siemens
ppm	Partes por millón
CO_2	Dioxido de carbono
Mhz	Megahercio
mA	Miliamperio
V	Voltios
kB	Kilobyte
kΩ	Kiloohmio
Máx.	Máximo
Mín.	Mínimo
m/s	Metros por segundo
%	Porcentaje
W	Vatio
L	Litro
cc	Centímetro cúbico
mg/L	Miligramos por litro
O	Oxígeno
C	Carbono

1 INTRODUCCIÓN

En este primer capítulo se expondrán los motivos por los cuales se ha desarrollado el presente trabajo, y posteriormente se comentarán los objetivos fijados.

1.1 Motivación

Desde hace bastante tiempo, a la gente le gusta tener y cuidar personalmente su acuario, y su intención es tener el mayor número de parámetros controlados para evitar problemas en él. Para ello se recurre a la tecnología actual, que permite automatizar las tareas de cuidado de los acuarios.

Los dispositivos electrónicos actuales creados para el control de los acuarios tienen un coste elevado. Es por este motivo por el que surge la necesidad de la creación de un sistema que permita realizar las mismas funciones de estos sistemas comerciales, sin que tenga un precio elevado, y dando además la posibilidad al usuario de adaptar el sistema a las necesidades de este.

1.2 Objetivos

El objetivo fundamental del proyecto es el desarrollo de un sistema electrónico de bajo coste, de fácil manejo e intuitivo que cuente con un Arduino Mega como microcontrolador, encargado de monitorizar y controlar la información recibida mediante una serie de sensores, para el mantenimiento del acuario.

El presente proyecto abarca varios campos y objetivos. A continuación se detallan los pasos a realizar que se encuentran implícitos en el desarrollo del proyecto para llegar al objetivo fundamental:

En primer lugar será necesario realizar un estudio previo de los aspectos teóricos que son de importancia en el desarrollo del sistema. Para ello será necesario realizar una investigación sobre los diferentes sistemas ya creados para el mantenimiento de acuarios y sobre qué parámetros del agua son interesantes en el entorno de la acuariofilia.

En segundo lugar se sitúa el diseño y fabricación del sistema, encontrándose implícito en este apartado la selección del hardware que se usará, el desarrollo del programa de control, y el diseño final del circuito sobre una placa PCB y su posterior fabricación.

Por último debe comprobarse que el sistema funciona de manera correcta, mediante la simulación de distintos eventos de importancia para el sistema, de manera manual interactuando con un acuario, o mediante la activación a través del código de algunas variables que simularían la ocurrencia de estos eventos.

2 ESTADO DEL ARTE

La cría de seres acuáticos en cautividad es una práctica muy antigua, que se remonta varios siglos antes de Cristo. Sin embargo, los acuarios tal y como se conocen hoy nacieron en el siglo XVIII, al surgir entre la gente acomodada la moda de coleccionar animales y sobre todo plantas. Para lo cual se desarrollaron recipientes sellados que podían contener cierta cantidad de líquido. Más adelante, dicha cantidad fue aumentando hasta convertirse en los modelos actuales.

A la hora de tener un acuario, es muy importante controlar a la perfección todos los parámetros del agua para así conseguir que el ecosistema se desarrolle a la perfección, con plantas y peces saludables, libre de algas y enfermedades, y seguro para el usuario.

Figura 2-1. Acuario plantado

Existen tanto sistemas comerciales como sistemas de código abierto desarrollados principalmente sobre Arduino y Raspberry Pi para el control de los parámetros del agua en un acuario.

Entre los sistemas comerciales podemos nombrar dos controladores: AT-Control de la empresa Aqua Medic [1] y Profilux controller 4 de la empresa Profilux [2]. El primero de ellos permite el control a través del ordenador, dando la posibilidad de descargar gráficas o datos importantes en él, y el segundo, es ligeramente superior ya que cuenta con la posibilidad de interactuar con el controlador desde un PC, desde una tablet o desde un dispositivo móvil.

Figura 2-2. Sistema comercial AT-Control

Figura 2-3. Sistema comercial ProfiLux.

La ventaja de estos sistemas es su exactitud y sencillez a la hora de la instalación. Su principal inconveniente es su elevado coste, el cual va desde los 600€ que puede costar los controladores de Aqua Medic hasta los más de 1000€ que pueden llegar a costar los desarrollados por Profilux.

También existen gran cantidad de proyectos open source en la web relacionados con el control de acuarios. Entre los cuales se pueden destacar tanto Jarduino como Ferduino, ambos usando el microcontrolador Arduino Mega. La gran ventaja de este tipo de sistemas es la facilidad con la que podemos adaptarlo a nuestras necesidades, modificando lo que creamos conveniente. [3]

Jarduino está pensado para el control del riego, pero al ser un tema bastante similar al control de acuarios, la gente ha adaptado el código y ha sustituido algunos de los sensores por otros más convenientes en el ámbito de la acuariofilia. [4]

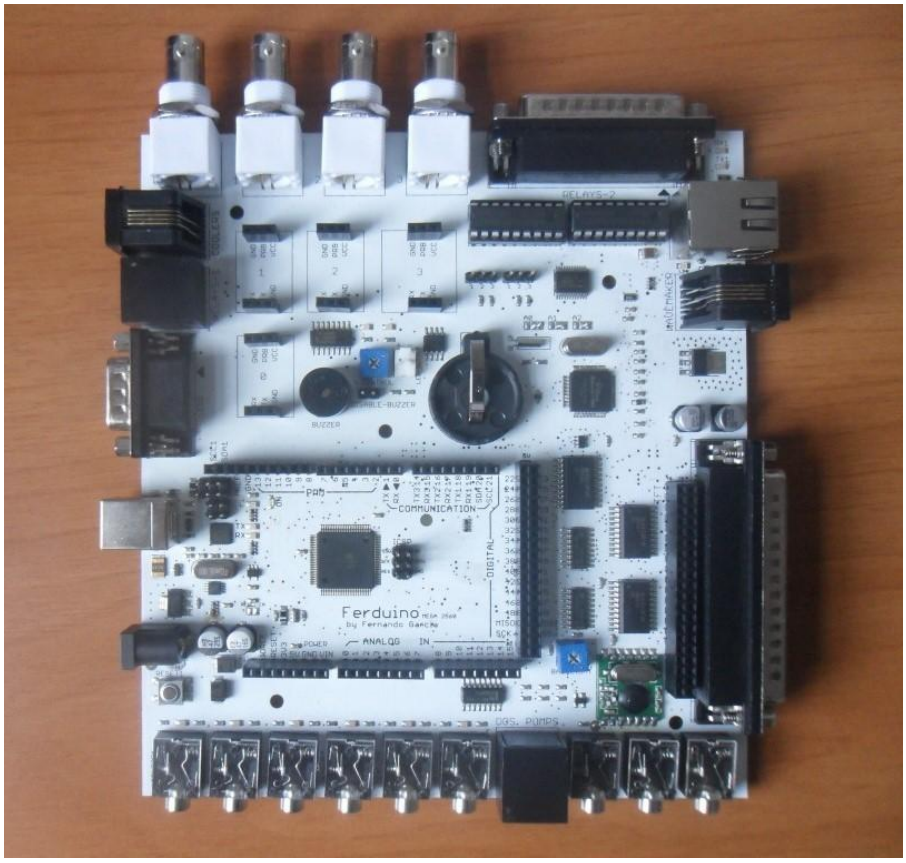


Figura 2-4. Placa Ferduino.

Figura 2-5. Montaje placa Jarduino.

3 PARÁMETROS DEL AGUA

El presente trabajo se va a centrar en la monitorización de diversos parámetros del agua, enfocándose al uso en acuarios.

Según la RAE el agua es un líquido transparente, incoloro, inodoro e insípido en estado puro, cuyas moléculas están formadas por dos átomos de hidrógeno y uno de oxígeno, y que constituye el componente más abundante de la superficie terrestre y el mayoritario de todos los organismos vivos.

Orientando el uso de agua a acuarios, cabe destacar la importancia del mantenimiento de una serie de parámetros en unos márgenes adecuados para así posibilitar la vida en su interior.

A continuación, se va a proceder a comentar algunos de los más importantes.

3.1 PH

El pH es una medida de acidez o alcalinidad de una disolución. Indica la concentración de iones de hidrógeno H^+ presentes en determinadas disoluciones. En la mayoría de las sustancias presentes en la naturaleza, estas concentraciones son muy bajas y expresarlas en forma decimal o exponencial resulta engorroso, y con frecuencia es fuente de errores. En el año 1909, el químico danés Sören Sörensen propuso una alternativa para la expresión de la concentración de H^+. Sugirió que en lugar de usar números en forma decimal o exponencial, se empleara una transformación logarítmica de la concentración molar de protones a la que denominó pH y definió matemáticamente como:

$$pH = \log\frac{1}{[H^+]} = -\log[H^+]$$

Como resultado de esta transformación, los números fraccionarios se convierten en números con enteros positivos, y como es inversa, mientras mayor sea la concentración de H^+, el valor del pH es menor. Este aspecto indica la importancia de regular correctamente el valor del pH del acuario, ya que una simple variación de 1 en la escala de pH, representa un cambio de diez veces en la concentración del agua.

A pesar del tiempo transcurrido desde la aparición de esta forma de medida, hoy día sigue siendo el pH la forma más común de expresar la acidez y la alcalinidad.

La escala que se usa para la medición de este parámetro va de 1 a 14. El pH 7 equivale a un pH neutro y es el que tiene el agua químicamente pura. Valores menores a 7 indican acidez del agua y valores mayores indican alcalinidad, estos niveles serán mayores cuanto más se acerquen al extremo.

Tabla 3-1. Escala de valores del pH

Reacción	pH	Ejemplo
Ácida	0	Ácido de baterías
	1	Ácido sulfúrico
	2	Jugo de limón o vinagre
	3	Jugo de naranja o refresco de cola
	4	Lluvia ácida
	5	Banana
	6	Lago saludable o leche
Neutra	7	Agua pura
Básica	8	Agua de mar o huevos
	9	Bicarbonato de sodio
	10	Detergente
	11	Amoniaco
	12	Soda caustica
	13	Lavandina
	14	Limpiador líquido de cañerías

Como pequeño resumen a lo descrito anteriormente se pueden destacar dos propiedades importantes:

- Varía de forma inversa a la concentración de protones, a mayor concentración, mayor acidez, pero menor valor de pH.
- Es logarítmica, es decir, un cambio de unidad de pH, representa un cambio diez veces en la concentración de protones.

3.2 Conductividad

En general, el flujo de electricidad a través de un conductor es debido a un transporte de electrones. Según la forma de llevarse a cabo este transporte, los conductores pueden ser de dos tipos: conductores metálicos o electrónicos y conductores iónicos o electrolíticos.

Las disoluciones acuosas pertenecen a los conductores iónicos o electrolíticos. En ellas la conducción de electricidad al aplicar un campo eléctrico se debe al movimiento de los iones en disolución, los cuales transfieren los electrones a la superficie de los electrodos para completar así el paso de corriente.

La conductividad eléctrica del agua se define como la medida de la capacidad de ésta para transportar la corriente eléctrica y permite conocer la concentración de especies iónicas disueltas en su interior. Dependerá además del voltaje aplicado, de la viscosidad del medio, del tipo, número, carga y movilidad de los iones

presentes. En disoluciones acuosas, puesto que la viscosidad disminuye con la temperatura, la conductividad aumentará a medida que aumente la temperatura.

La conductividad es la inversa de la resistividad por lo que su unidad es el S/m o $\Omega^{-1} \cdot m^{-1}$. Para el caso concreto del agua la forma de medida habitual son los $\mu S/cm$.

Tabla 3-2. Valores de la conductividad en diferentes aguas

Tipos de agua	Valores aproximados
Agua ultra pura	0.055 $\mu S/cm$
Agua destilada	0.5 $\mu S/cm$
Agua de montaña	1.0 $\mu S/cm$
Agua doméstica	500 a 800 $\mu S/cm$
Max. Para agua potable	1055 $\mu S/cm$
Agua de mar	56 mS/cm
Agua salobre	100 mS/cm

3.3 Temperatura

La temperatura del acuario es un factor de gran importancia para el desarrollo correcto de los seres vivos que se encuentren en su interior, un cambio brusco en la temperatura puede ocasionarles grandes daños, por eso es de vital importancia garantizar una estabilidad de la temperatura. Para ello, nuestro sistema hará uso de actuadores que aumenten o disminuyan la temperatura según sea necesario.

En la siguiente tabla se muestran los rangos de temperatura medidos en diversos tipos de agua.

Tabla 3-3. Rangos de temperatura en diferentes tipos de agua

Tipos de agua	Rango de temperatura	Temperatura media
Tropical	20ºC - 30ºC	27ºC
Subtropical	16ºC - 27ºC	22ºC
Boreal	1ºC - 17ºC	11ºC
Ártica	-1ºC - 9ºC	3ºC
Mediterránea	11ºC – 28ºC	19ºC

3.4 Dureza

El término dureza se refiere al contenido total de iones alcalinotérreos que hay en el agua. Como la concentración de Ca^{2+} y Mg^{2+} es, normalmente, mucho mayor que la del resto de iones alcalinotérreos, la dureza puede aproximarse a la suma de las concentraciones de estos dos iones. Se expresa por lo general como el número equivalente de miligramos de carbonato de calcio por litro ($\frac{mg}{L} de\ CaCO_3$) y constituye un parámetro muy significativo en la calidad del agua. Se puede subdividir en dos grupos:

- La dureza permanente (GH). Determinada por todas las sales de calcio y magnesio excepto carbonatos y bicarbonatos.
- La dureza temporal o de carbonatos (KH). Determinada por el contenido de carbonatos y bicarbonatos de calcio y magnesio.

Por tanto:

$$Dureza\ total = Dureza\ temporal + Dureza\ permanente$$

Tabla 3-4. Clasificación del agua en función del nivel de dureza

Dureza expresada en $CaCO_3$	Clasificación
0-100	Agua blanda
101-200	Agua moderadamente dura
201-300 (límite agua potable)	Agua dura
>300	Agua muy dura

La dureza, además de en $\frac{mg}{L}\ de\ CaCO_3$(equivalente a ppm), también puede expresarse en diversas escalas:

Tabla 3-5. Escalas de dureza

Escalas	Valor equivalente a $1\frac{mg}{L}de\ CaCO_3$ ó 1ppm
Grados ingleses	14.3
Grados Americanos	17.2

La conductividad y la dureza guardan una relación destacable que será de utilidad en el desarrollo del proyecto:

Grados ingleses: $1.4\frac{\mu S}{cm} = 1ppm\ o\ \frac{mg}{l}de\ CaCO_3$

Grados americanos: $2\frac{\mu S}{cm} = 1ppm\ o\ \frac{mg}{l}de\ CaCO_3$

3.5 CO_2

El dióxido de carbono es un gas inodoro, incoloro, ligeramente ácido y no inflamable. Es soluble en agua cuando la presión se mantiene constante, y está formado por una molécula lineal de un átomo de carbono ligado a dos átomos de oxígeno, de la forma O = C = O.

A pesar de que a temperatura y condiciones ordinarias se encuentra en forma gaseosa, puede solidificarse si se somete a temperaturas inferiores de -79º C, y licuarse cuando se disuelve en agua.

El dióxido de carbono es, junto a la iluminación, el nutriente imprescindible para la alimentación de nuestras plantas. Gracias al CO_2, las plantas realizan la fotosíntesis, liberando oxígeno.

En acuarios con altas prestaciones lumínicas, el CO_2 se hace totalmente necesario, puesto que sin él algunas plantas no serán capaces de procesar todos los nutrientes y apenas crecerán, y otras morirán.

4 REQUERIMIENTOS

Antes de comenzar a desarrollar el proyecto, hay que indicar unos requerimientos previos que se tendrán en cuenta a la hora de tomar decisiones durante la realización del proyecto.

1. El sistema debe conectarse a la red eléctrica y comenzar a funcionar de manera autónoma con unos parámetros fijados por defecto, que son:
    a. Temperatura: 25ºC.
    b. Horario de iluminación: 10 horas.
    c. Horario de encendido de la electroválvula de CO_2: 10 horas.
2. El sistema debe medir la temperatura con una precisión de ±1ºC, es decir, si se fija una temperatura de 25ºC, el sistema considerará correcta cualquier temperatura incluida en el rango de 24ºC a 26ºC. La temperatura mínima que deberá medir será de 5ºC y la temperatura máxima de 40ºC.
3. El sistema debe medir el pH con una precisión de ±0.5. El nivel mínimo de pH será de 1 y el nivel máximo de 13, ya que los valores extremos no son de interés.
4. El sistema debe medir la dureza con una precisión de ±10 ppm. El nivel mínimo de dureza que se deberá medir será 1500 ppm y el nivel máximo de 20000 ppm.
5. El sistema debe medir el nivel de llenado del acuario mediante la fijación de dos niveles, nivel máximo y nivel mínimo. Se considerará correcto el nivel de llenado si el nivel de agua se encuentra entre estos dos niveles fijados manualmente.
6. Cualquier parámetro medido por el sistema podrá ser visualizado por el display en cualquier momento.
7. El horario de activación de la iluminación del acuario y de la electroválvula de CO_2 podrá ser modificada mediante el teclado y el display en cualquier momento.
8. La temperatura deseada en el acuario puede ser programada en cualquier instante. Haciendo uso del sensor de temperatura, para alcanzar esta temperatura fijada, si es necesario, se hará uso de una resistencia calefactora o un ventilador.

5 ESTUDIOS PREVIOS

Como análisis previo al desarrollo inicial del proyecto, se deben dejar claro algunos aspectos. Entre los cuales está diferenciar los parámetros de interés de nuestro sistema y de nuestro microcontrolador, es decir, entradas al microcontrolador y parámetros que se estudiarán en el sistema, el número de entradas y salidas de las que constará nuestro microcontrolador, si se precisa de señales digitales o analógicas, qué sensores son los más apropiados para nuestro entorno de trabajo etc.

Como bien se ha comentado en capítulos anteriores, se van a monitorizar una serie de parámetros del agua mediante un microcontrolador, los cuales serán modificados en la medida de lo posible mediante una serie de actuadores. Como punto de partida de nuestro proyecto tendremos que responder la pregunta de cuántas entradas y salidas tendremos en nuestro sistema y como ponerlas en funcionamiento a través del microcontrolador.

Los parámetros de interés para nuestro sistema serán la dureza, la temperatura, el CO_2, nivel de agua, pH y luz.

El siguiente paso sería comprobar qué sensores serán necesarios para controlar los parámetros descritos anteriormente, también comprobar si su comunicación con el microcontrolador es analógica o digital. De forma paralela debe comprobarse lo mismo con las salidas, y por último comentar la forma en la que el sistema comunicará al usuario la situación monitorizada.

Para la **temperatura** será necesario un sensor de temperatura. Existen multitud de sensores de temperatura en el mercado, tanto analógicos como digitales.

Entre los análogicos se puede destacar el TMP36 de Analog Devices, el cual cumple con los requisitos de rango de temperatura medible, visto en la tabla expuesta en el capítulo de aspectos teóricos. Sin embargo, no cuenta con un encapsulado apropiado para el contacto con el agua, por lo que esto puede ser a priori un inconveniente. Aunque podría realizarse de manera casera un encapsulado al sensor para poder introducirlo en el agua y poder trabajar con él sin problema.

En cuanto a los sensores digitales de temperatura, el sensor DS18B20 es sin duda el más apropiado debido a la posibilidad de conseguir el sensor encapsulado en una sonda que puede estar en contacto con el agua. Otra ventaja importante de este sensor es que cuenta con la comunicación "1-Wire", que sólo necesita de un cable para comunicación, y en el mismo cable pueden incluirse diferentes sensores, lo que nos permitiría conectar otro sensor más de temperatura para medir la temperatura del exterior por ejemplo. Además cumple con los rangos de temperatura necesarios.

En nuestro sistema se desea mantener la temperatura en unos márgenes adecuados, por lo que necesitaremos un actuador para aumentarla en caso de que esta esté por debajo del margen inferior fijado, o para disminuirla en caso contrario. Para ello haremos uso de un ventilador y una resistencia calefactora.

Por lo tanto se necesitará un sensor de temperatura como entrada al microcontrolador, y dos salidas, una para el ventilador y otra para la resistencia calefactora.

Para la medida del **pH**, se necesitará un sensor de pH. Una medida bastante extendida y de reducido coste es usar tiras de papel tornasol, los cuales se introducen en el agua y según el color que toma puede saberse el nivel de pH.

Nuestro objetivo es lograr monitorizar el pH sin la necesidad de la participación de una persona. Para ello existe un sensor analógico de pH llamado SEN0161, diseñado para su uso en microcontroladores, fabricado por DFRobot, el cual consta de una sonda con conexión BNC y una pequeña placa que hace las funciones de puente entre la sonda y el microcontrolador.

Para la modificación del pH no es posible realizarlo de manera automática, sino que será necesario añadir una serie de sustancias químicas por parte del usuario, que sabrá cuándo es necesaria su utilización ya que el sistema constará con un display LCD que mostrará los niveles medidos.

Resumiendo, será necesario un sensor analógico de pH como entrada, y como salida no se necesitará nada

puesto que se hará de manera manual.

Para medir la **dureza** del acuario, no hay dispositivos específicos, por lo que se hará uso de la relación que esta medida guarda con la conductividad del agua, ya que para la conductividad si existen sensores. Por tanto, necesitaremos un sensor de conductividad para posteriormente realizar las oportunas operaciones hasta conseguir el nivel de dureza. Como sensor de EC se usará el sensor DFR0300 producido por DFRobot. Este nos dará el valor de conductividad en mS/cm.

Para modificar la dureza, al igual que el pH, el usuario tendrá que modificarlo de manera manual, disolviendo en el agua una serie de sustancias químicas específicas para la modificación de la dureza, aspecto que no es de interés en nuestro trabajo.

Así pues, para controlar la dureza del agua se necesitará un sensor analógico de conductividad, y no se necesitará añadir ninguna salida ya que se hará de manera manual.

El **nivel de agua** será controlado mediante una herramienta muy común y con un funcionamiento muy sencillo, la boya de nivel, la cual actúa como interruptor dependiendo si el nivel de agua está por encima o por debajo de su colocación en el acuario. Para lograr mantener el agua dentro unos niveles fijados, sería necesario el uso de dos boyas, una señalando el nivel superior y otra el inferior.

También es posible el uso de un sensor de ultrasonido situado en la tapadera del acuario, la implementación de esta alternativa sería algo más complicada en cuanto a desarrollo, pero la precisión sería mucho mayor, ya que a raíz del tiempo que tarde el sensor en recibir el eco y sabiendo la altura del acuario, puede determinarse el porcentaje de llenado de este.

Debido a que la precisión necesaria en cuanto al nivel de llenado en el proyecto no es muy elevada, se opta por el uso de dos boyas de nivel, por lo tanto precisaríamos de dos entradas digitales.

Para controlar el nivel de agua del acuario se utilizarán dos bombas de agua. Una de ellas encargada del vaciado del acuario, y otra encargada del llenado. Para el llenado podría usarse también una válvula de llenado, donde el grifo daría la suficiente presión para conseguir llenar el tanque.

Por lo tanto para el nivel de agua se necesitarán 4 elementos. Como entradas al microcontrolador se tendrán dos boyas de nivel digitales, y dos salidas, una para cada bomba de agua necesaria.

En cuanto a la medida de la **luz**, existen diferentes componentes electrónicos los cuales determinan la cantidad de luz existente, pero para nuestro proyecto no será necesaria la utilización de ninguno de estos ya que el acuario estará ubicado en una zona interior y se le deberá proporcionar una fuente de luz directa al agua durante un intervalo de tiempo fijo sin tener en cuenta la cantidad de luz exterior.

Por lo tanto no se usará un sensor de este tipo, simplemente se fijarán unas horas de encendido y apagado de las luces que pueden ser modificadas a petición del usuario mediante un teclado con el que contará nuestro sistema.

No será necesario un sensor de **CO_2**, ya que conociendo los niveles de dureza y pH puede hacerse una estimación de la cantidad de CO_2 disuelto en el agua, siendo así innecesario el uso de un sensor de CO_2. Aunque en este proyecto no se realizarán las conversiones pertinentes para conocer el nivel de CO_2.

Para añadir el CO_2 usaremos una válvula solenoide, un contador de burbujas y una bombona de CO_2.

Según lo comentado, como entradas al microcontrolador necesitaremos:

1. Sensor de temperatura
2. Sensor de pH

3. Sensor de EC
4. Dos boyas de nivel

Y en cuanto a salidas:

1. Resistencia calefactora
2. Ventilador
3. Válvula CO_2
4. Bomba de agua para llenado
5. Bomba de agua para vaciado
6. Iluminación

Quedando nuestro sistema de la siguiente manera:

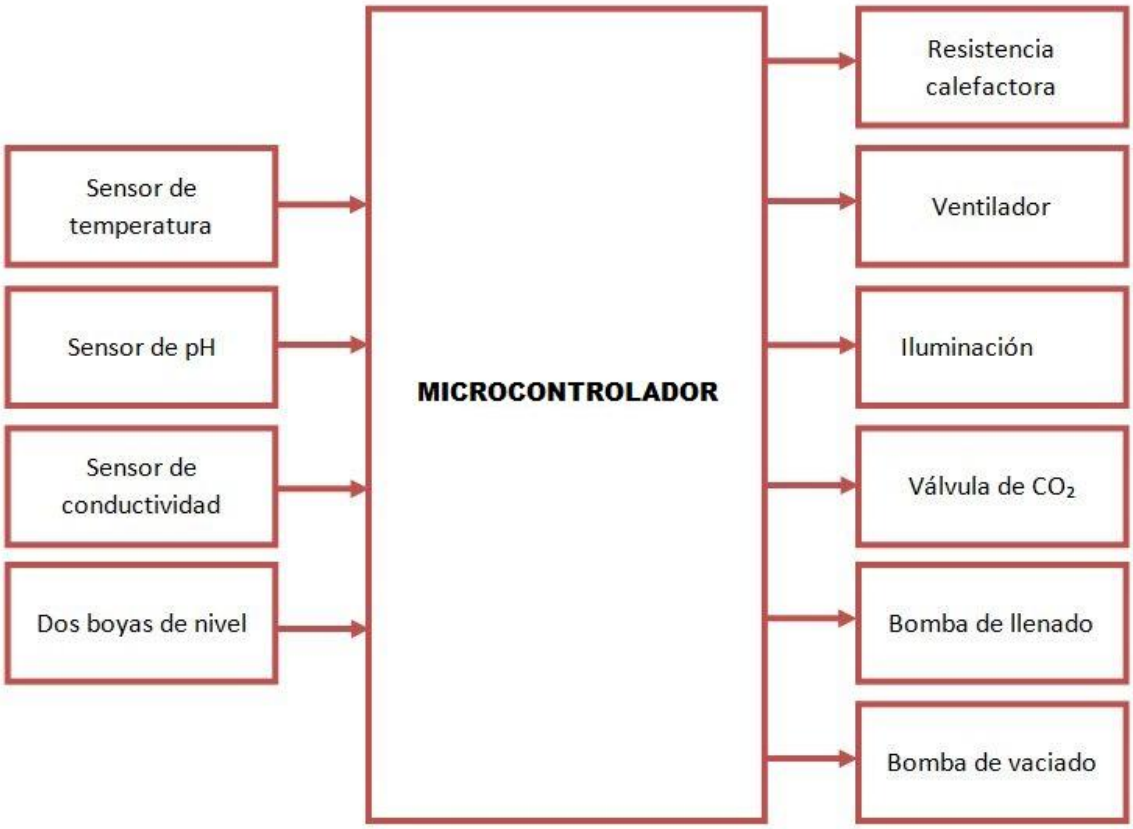


Figura 5-1. Entradas y salidas al microcontrolador

En capítulos siguientes se explicará con más detalle el funcionamiento de los elementos citados anteriormente.

6 DESARROLLO HARDWARE

6.1 Elección del microcontrolador

Existen multitud de microcontroladores en el mercado capaces de soportar el presente proyecto. Nuestro objetivo es que la plataforma sea de código abierto, para así facilitar el desarrollo del trabajo. Entre estas plataformas encontramos, entre otras, Raspberry Pi, BeagleBone, Sharks Cove, Waspmote o Arduino.

Se ha elegido la plataforma Arduino debido a que, además de simplificar el proceso de trabajo con microcontroladores, ofrece una serie de ventajas respecto a otros sistemas:

La elección de la plataforma Arduino se debe a que, además de simplificar el proceso de trabajo con microcontroladores, ofrece una serie de ventajas respecto a otros sistemas:

- Precio. Las placas Arduino son relativamente baratas en comparación con otras plataformas de microcontroladores.
- Multiplataforma. El software Arduino (IDE) puede ejecutarse en sistemas operativos Windows, Mac y Linux. La mayoría de los sistemas de microcontroladores están limitados a Windows.
- Entorno de programación simple y claro. El software Arduino (IDE) es sencillo de usar para gente sin experiencia previa, y aún lo suficientemente flexible para que los usuarios más avanzados puedan sacarle provecho.
- Código libre y software extensible. El software de Arduino está publicado como herramientas de código abierto, disponibles para la extensión por programadores experimentados. El lenguaje puede ser expandido a través de bibliotecas C++. Gente que quiera entender los detalles técnicos puede realizar el salto de Arduino al lenguaje de programación AVR-C en el cual está basado. También se puede agregar código AVR-C directamente en los programas Arduino.
- Código libre y hardware extensible. Los diseños de las placas Arduino están publicados bajo una licencia de Creaciones Comunes, por tanto, experimentados diseñadores de circuitos pueden realizar su propia versión de la placa, ampliando y mejorando las características dependiendo de su posterior uso.

6.2 Arduino

Arduino es una plataforma electrónica de código abierto (open source) basada en el sencillo manejo tanto de software como de hardware. Las placas Arduino son capaces de leer entradas, como puede ser la temperatura mediante un sensor, y convertirlas a unas salidas, activando un ventilador o encendiendo un LED. Se le puede decir a la placa lo que se desea que haga mediante el envío de una serie de instrucciones al microcontrolador de la placa. Para ello se utiliza el lenguaje de programación Arduino, basado en Wiring, y el software Arduino (IDE), basado en Processing.

Con el paso de los años Arduino ha sido el "cerebro" de miles de proyectos, desde objetos cotidianos a complejos instrumentos científicos. Una comunidad de diseñadores alrededor del mundo (estudiantes, aficionados, progamadores, profesionales) se han reunido en torno a esta plataforma de código abierto, sus contribuciones han proporcionado una gran cantidad de conocimiento que puede ser de tanto para novatos como expertos.

Arduino nació en el Instituto de Diseño de Interacciones de Ivrea, como una herramienta sencilla para el rápido prototipado, enfocado a estudiantes sin una base de electrónica y programación. Tan pronto como Arduino comenzó a convertirse en una gran comunidad, la placa Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y retos, diferenciándose su oferta de placas simples de 8 bits a productos para IoT, weareable, impresión 3D, y entornos embebidos. Todas las placas Arduino son completamente de código abierto, permitiendo a los usuarios construirlos de manera independiente y eventualmente adaptarlos a sus

necesidades particulares.

Gracias a su experiencia de usuario sencilla y accesible, Arduino ha sido usada en miles de proyectos y aplicaciones diferentes. El software Arduino es sencillo de usar para principiantes, y suficientemente flexible todavía para usuarios avanzados. Puede ejecutarse en Mac, Windows y Linux. Profesores y estudiantes lo usan para construir instrumentos científicos baratos, para probar principios físicos y químicos, o para comenzar a iniciarse en el mundo de la programación y la robótica. Diseñadores y arquitectos construyen prototipos interactivos, músicos y artistas lo usan para instalaciones y experimentar con nuevos instrumentos musicales. Arduino es la herramienta clave para aprender cosas nuevas. Cualquiera (niños, aficionados, artistas, programadores) puede empezar a probar con Arduino simplemente siguiendo una serie de instrucciones paso a paso de un kit, o compartir sus ideas en internet con otros miembros de la comunidad Arduino.

Existen otros muchos microcontroladores y plataformas disponibles para la computación física. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, y muchos otros ofrecen funcionalidades muy parecidas. Todas estas herramientas toman los detalles problemáticos de la programación del microcontrolador y las agrupan en un paquete fácil de usar.

6.3 Placas Arduino

A día de hoy, la variedad de productos ofrecidos por Arduino es inmensa, ofreciendo placas, módulos, accesorios, y kits enfocados a una aplicación concreta. Como se puede apreciar en el siguiente gráfico existen placas pensadas para personas poco o nada experimentadas en el mundo de Arduino y la programación de microcontroladores, placas con características un poco más avanzadas, Arduino orientado a IoT, pensados para llevar encima (wearable), o incluso otros pensados exclusivamente para el manejo de impresoras 3D.

Tabla 6-1. Clasificación de las placas Arduino según características

NIVEL BÁSICO	CARACTERÍSTICAS AVANZADAS	IoT	WEARABLES	IMPRESIÓN 3D
Arduino UNO	Arduino Mega	Arduino Yun	Arduino Gemma	Materia 101
Arduino Leonardo	Arduino Mega ADK		Lilypad Arduino USB	
Arduino 101	Arduino Zero		Lilypad Arduino Main Board	
Arduino Robot	Arduino Due		Lilypad Arduino Simple	
Arduino Esplora	Arduino ISP		Lilypad Arduino Simple Snap	
	Arduino Ethernet			

A continuación se comentarán brevemente las características más importantes de algunas de las placas Arduino:

- Arduino UNO. Esta es la placa más conocida de Arduino y, sin duda, la más recomendada para comenzar. Esta fue la primera en salir al mercado, y el resto de placas posteriores están apoyadas en ella para su diseño. Posee un microcontrolador ATmega320 de 8 bits a 16 Mhz con una alimentación de 5 V. Uno de sus defectos puede ser la memoria, ya que esta es algo limitada, pero no impide que no

sea compatible con multitud de proyectos. Contiene 14 pines digitales, de los cuales 6 de ellos pueden usarse como PWM, y 6 pines analógicos. Los pines pueden trabajar con corrientes de hasta 40 mA.

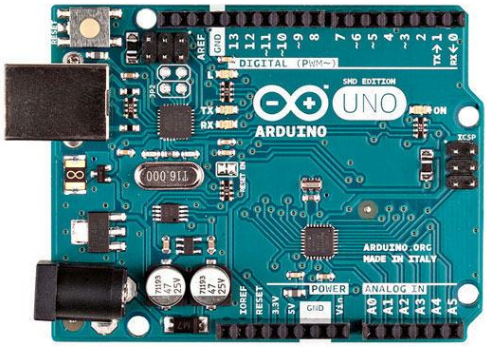


Figura 6-1. Arduino UNO

- Arduino 101. Es una placa nueva, presentada en este año. Aspectos a destacar de ella son su módulo Intel Curie, con dimensiones reducidas y bajo consumo. Contiene un microcontrolador x86, DSP, bluetooth, acelerómetro, giroscopio etc. El tamaño y conexiones es igual a Arduino UNO.

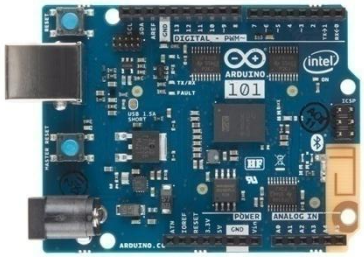


Figura 6-2. Arduino 101

- Arduino Esplora. Esta placa a simple vista se diferencia mucho del resto de la oferta de Arduino. Esto se debe a su forma ovalada y a la inclusión de una serie de sensores y botones en la superficie de la placa (acelerómetro, sensor de temperatura, sensor de luz, zumbador, botones, joystick, micrófono y socket para conectar una pantalla TFT LCD). El microcontrolador es un ATmega23u4. El principal inconveniente de esta placa es su escasa conectividad, ya que todo lo tiene integrado. Por lo que no es aconsejable para personas que quieran indagar más en el mundo de la electrónica.

Figura 6-3. Arduino Esplora

- Arduino Leonardo. Es muy similar a Arduino UNO. Sus diferencias son su tamaño más reducido (usando sólo conexión mini-USB), el número de pines (20 pines digitales y 12 pines analógicos), y que los pines son solo perforaciones en la placa, no cuentan con las tiras de pines para la conexión. También el microcontrolador es diferente, ATmega32u4.

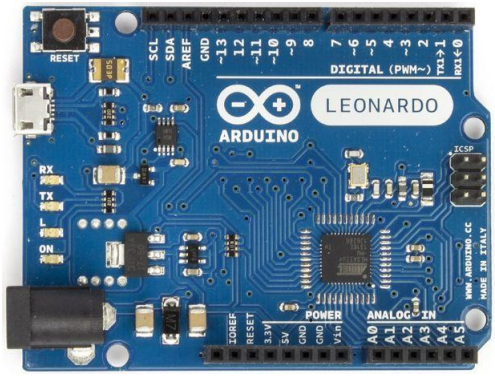


Figura 6-4. Arduino Leonardo

- Arduino Mega. EL microcontrolador que usa es el ATmega2560, con capacidades superiores al Arduino UNO. Una de las características más destacables de Arduino Mega es su gran cantidad de pines, 54 pines digitales, de los cuales 15 de ellos son PWM, y 16 pines analógicos.

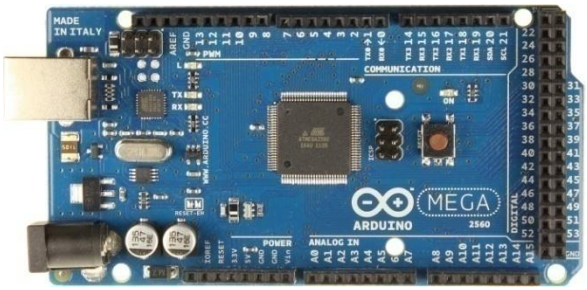


Figura 6-5. Arduino Mega

- Arduino Mega ADK. La principal diferencia con Arduino Mega es que dispone de una interfaz preparada para ser conectada mediante USB a dispositivos móviles basados en Android, gracias a su IC MAX3421e.

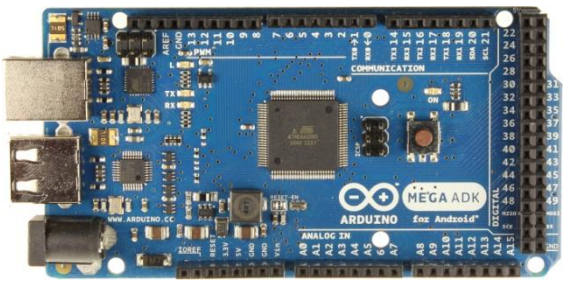


Figura 6-6. Arduino Mega ADK.

- Arduino Due. Es la primera placa basada en un microcontrolador ARM de 32 bits, el Atmel SAM3X8E ARM Cortex-M3. A diferencia de la mayoría de placas Arduino maneja un voltaje de 3.3 V. Está orientada a proyectos con alta capacidad de procesamiento. Posee 54 pines digitales, 12 de ellos PWM, y 12 analógicos.

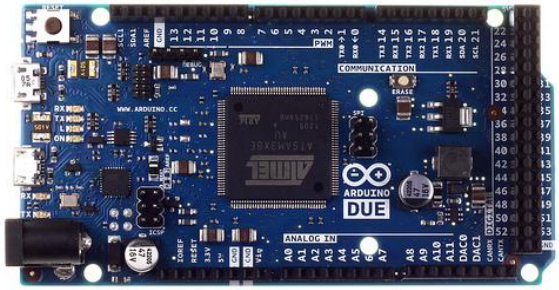


Figura 6-7. Arduino Due

- Arduino Zero. Aspecto muy similiar a Arduino UNO, con la diferencia del potente microcontrolador que contiene, Atmel SAMD21 MCU. Básicamente se puede decir que está placa está recomendada para personas que con Arduino UNO comiencen a verse limitadas.

Figura 6-8. Arduino Zero

- Arduino Ethernet. Se trata de un Arduino UNO con conectividad Ethernet.

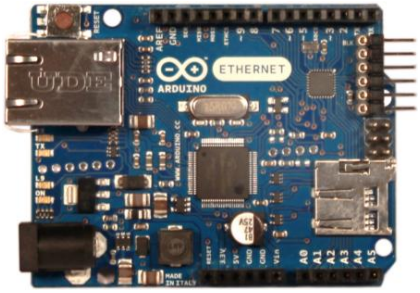


Figura 6-9. Arduino Ethernet

- Arduino Yun. Es la placa perfecta cuando se desea conectar dispositivos, en general proyectos basados en IoT. Combina la potencia de Linux con el fácil manejo de Arduino. Basado en ATmega32u4 y en el chip Atheros AR9331, que soporta la distribución de Linux basada en OpenWrt llamada OpenWrt-Yun. Es una placa muy parecida a Arduino UNO con la inclusión de conexión Ethernet, WiFi, USB y micro-SD.

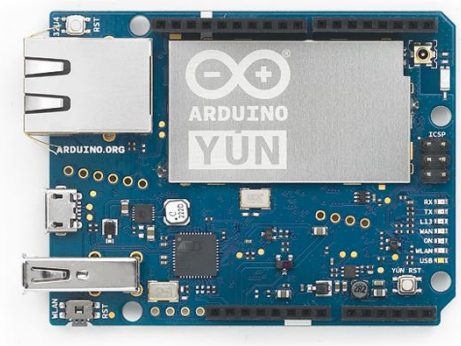


Figura 6-10. Arduino Yun

- Arduino Gemma. Es una placa miniatura realizada por Adafruit basada en el microcontrolador ATtiny85. Posee todo lo necesario, simplemente se conecta a la alimentación mediante un cable micro-USB o una batería (conector JST para una batería de 3.7 V) y ya se puede comenzar a realizar proyectos wearables. Posee 3 pines, dos pueden usarse como PWM y uno como analógico.

Figura 6-11. Arduino Gemma.

- Arduino Lilypad. Este conjunto de placas miniatura están especialmente diseñadas para su uso en prendas y textiles. Se basa en dos versiones de microcontrolador diferentes, ATmega168V y ATmega328V, siendo la segunda más potente (5.5 V frente a los 2.7 V de la primera). Dispone de 14 pines digitales, 6 de ellos PWM, y 6 analógicos.

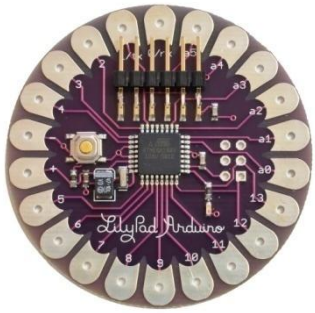


Figura 6-12. Arduino Lilypad

- Materia 101. Aunque no es una placa, cabe hacer una mención a Materia 101, al ser la primera impresora 3D oficial de Arduino. Su diseño está apoyado sobre un Arduino Mega.

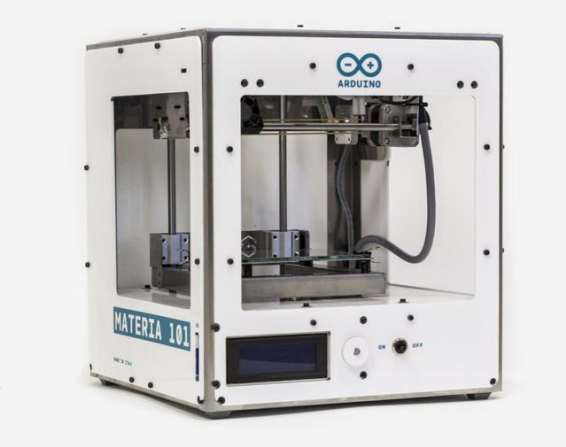


Figura 6-13. Materia 101

6.4 Arduino Mega

La elección del Arduino Mega como núcleo de nuestro sistema ha sido debida principalmente al gran número de pines de los que dispone el mismo, aspecto que facilita una posible ampliación futura del sistema, añadiéndole más sensores o actuadores. También posee una gran capacidad de memoria, lo que permite que, en el caso de ampliarse en un futuro, se puedan incluir más variables al programa. [5]

Las características más importantes de la placa Arduino Mega se pueden ver en la siguiente tabla:

Tabla 6-2. Características principales Arduino Mega

CARACTERÍSTICAS	VALORES
Microcontrolador	ATmega2560
Tensión de alimentación	5 V
Tensión de entrada recomendada	7-12 V
Límite de entrada	6-20 V
Pines digitales	54 (14 con PWM)
Entradas analógicas	16
Corriente máxima por pin	40 mA
Corriente máxima para el pin 3.3V	50 mA
Memoria flash	256 KB
SRAM	8 KB
EEPORM	4 KB
Velocidad de reloj	16 MHz

6.5 Sensores

El sistema constará de una serie de sensores para monitorizar los parámetros anteriormente citados, los cuales se conectarán al Arduino mediante sus pines analógicos o digitales, dependiendo del sensor. Estos sensores medirán la temperatura, el pH, la conductividad y el nivel de agua, como ya se ha comentado en los Estudios Previos. En este apartado se realizará un análisis más detallado sobre ellos.

6.5.1 Sensor de temperatura

El sensor de temperatura DS18B20 [6] se trata de un sensor digital con dos características bastante útiles, por un lado está disponible una versión encapsulada y cableada del sensor que permite su uso en contacto con líquidos, aspecto de interés para nuestro trabajo. Y por otro lado está la utilización de un protocolo de comunicación llamado 1-Wire que nos permite utilizar varios sensores de forma simultánea conectados al mismo pin de nuestro Arduino. Esto es posible gracias a un código de 64 bits que posee cada sensor y lo distingue de cualquier otro. Por tanto el sensor contará con tres cables, dos para la alimentación (GND y VDD) y únicamente se necesitará uno para la comunicación ya que es bidireccional (DATA).

Figura 6-14. Sensor de temperatura DS18B20

Para la conexión con Arduino es necesaria la colocación de una resistencia de pull-up como se muestra en la figura siguiente. El valor recomendado para esta resistencia por el fabricante es de 4.7kΩ.

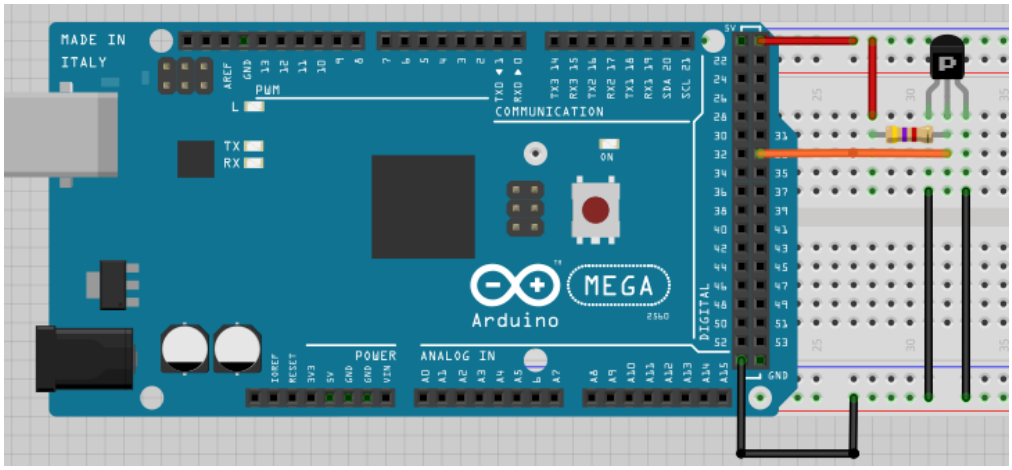


Figura 6-15. Conexión sensor DS18B20 a Arduino

Las características más importantes de este sensor se muestran en la siguiente tabla.

Tabla 6-3. Características sensor de temperatura DS18B20

ESPECIFICACIONES	VALORES
Rango de alimentación	3V – 5.5V
Rango de temperatura	-55ºC – 125ºC
Precisión	±0.5ºC (entre -10ºC y 85ºC)
Resolución del sensor	Programable de 9 a 12 bits

El funcionamiento del sensor viene detallado en el datasheet de este, por lo que no se va ahondar demasiado en él. Es importante conocer la relación que guarda la resolución configurada con la precisión obtenida y la relación entre el valor digital que proporciona el sensor con la temperatura. Ambas relaciones proporcionadas por el fabricante.

Tabla 6-4. Relación resolución-precisión

RESOLUCIÓN CONFIGURADA	PRECISIÓN OBTENIDA
12 bits	0.0625ºC
11 bits	0.1250ºC
10 bits	0.2500ºC
9 bits	0.5000ºC

Tabla 6-5. Relación temperatura-salida digital

Temperatura (ºC)	Salida digital
125	0000 0111 1101 0000
85	0000 0101 0101 0000
25.0625	0000 0001 1001 0001
10.125	0000 0000 1010 0010
0.5	0000 0000 0000 1000
0	0000 0000 0000 0000
-0.5	1111 1111 1111 1000
-10.125	1111 1111 0101 1110
-25.0625	1111 1110 0110 1111
-55	1111 1100 1001 0000

6.5.2 Sensor nivel de agua

6.5.2.1 Boya de nivel

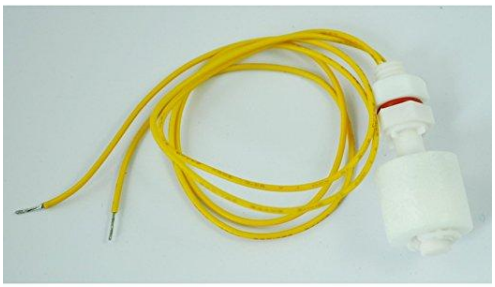


Figura 6-16. Boya de nivel

Los sensores que se van a exponer son realmente interruptores electromecánicos orientados a la medida del nivel del agua en un deposito. Existen distintos sensores, tanto en el montaje (vertical u horizontal), el material y diversos aspectos que hace que haya una gran variedad de sensores de este tipo.

En el montaje que se quiere realizar, habría que colocar dos sensores para mantener el nivel del agua en el nivel deseado, como se muestra en la imagen.

El funcionamiento de estos sensores digitales es muy sencillo, proporciona un nivel alto o bajo dependiendo de la situación de la boya. [7]

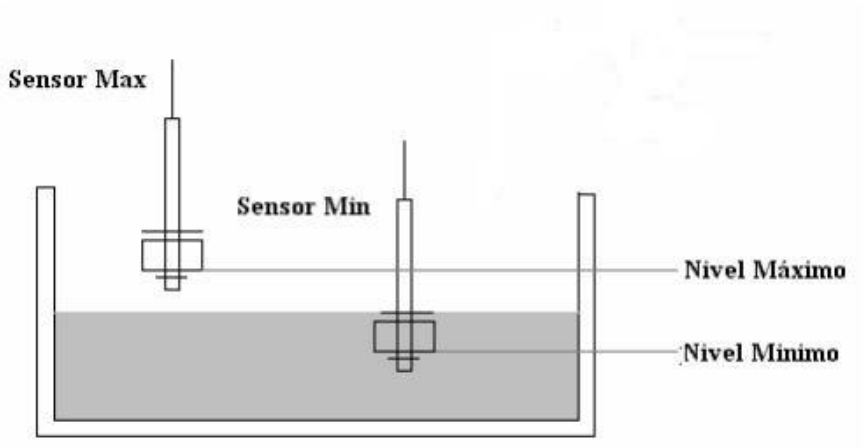


Figura 6-17. Esquema montaje boyas de nivel

6.5.2.2 Sensor de ultrasonido HC-SR04

El uso de este sensor permitiría mucha más exactitud en la medida del nivel del agua, ya que se podría calcular el porcentaje de llenado. Consta de cuatro pines, VCC, GND, trig (disparo del ultrasonido), y echo (recepción del ultrasonido). Su funcionamiento consiste en el envío de un pulso de alta frecuencia no perceptible por el ser humano, este rebotará en el objeto más cercano y será recibido por un micrófono funcionando para esa frecuencia. Si se mide el tiempo entre los pulsos y sabiendo la velocidad del sonido es posible calcular la distancia a la que se encuentra el objeto. [8]
El cálculo empleado para la medición de la distancia es el siguiente:

$$343\frac{m}{s}\cdot 100\frac{cm}{m}\cdot\frac{1}{1\cdot 10^{6}}\frac{s}{\mu s}=0.0343\frac{cm}{\mu s}=\frac{1\ cm}{29.15451895\ \mu s}$$

Por tanto se puede afirmar que tarda 29.15451895 µs en recorrer 1 cm. Se puede obtener entonces la distancia de la siguiente manera:

$$Distancia(cm)=\frac{Tiempo\ (\mu s)/29.15451895}{2}$$

La división por dos se realiza ya que lo que se ha medido es el tiempo que tarda el pulso en ir y volver, por lo que si no se realiza esta división se estaría obteniendo el doble de la distancia que se desea medir.

Figura 6-18. Sensor de ultrasonido HC-SR04

La elección para nuestro montaje será el uso de dos boyas de nivel, como ya se había comentado previamente en el capítulo de estudios previos, debido a su bajo coste y su fácil manejo.

6.5.3 Sensor de pH

El pH es el coeficiente que indica el grado de acidez o basicidad de una solución acuosa. Para la medida de éste se usará el sensor de pH analógico SEN0161 diseñado DFRobot que consta de tres elementos que conforman el instrumento en su totalidad como puede verse en la imagen, una sonda de pH, cable BNC y la placa PCB del circuito sensor. [9]

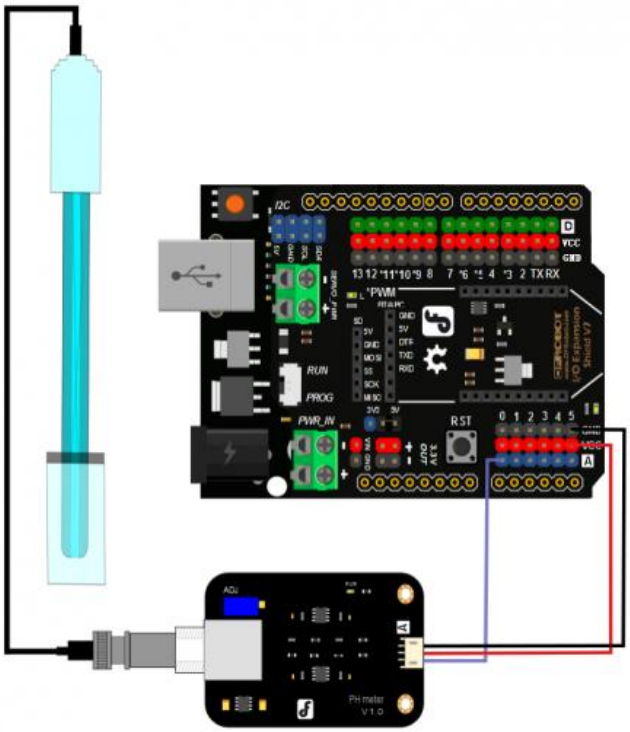


Figura 6-19. Esquema de conexión sensor de pH de DFRobot

En la siguiente tabla se recogen los aspectos más importantes del sensor.

Tabla 6-6. Características principales sensor de pH

Especificaciones	Valores
Tensión de operación	5V
Rango de medida de pH	0 – 14
Precisión	±0.1pH (a 25ºC)

La salida que proporciona el electrodo es en mV, y la relación que guarda con el pH se muestra en la tabla.

Tabla 6-7. Relación tensión-valor de pH

Tensión (mV)	Valor de pH
414.12	0
354.96	1
295.80	2
236.64	3
177.48	4
118.32	5
59.16	6
0.00	7
-59.16	8
-118.32	9
-177.48	10
-236.64	11
-295.80	12
-354.96	13
-414.12	14

Para su uso es necesaria una calibración previa del sensor, la cual puede realizarse mediante software o mediante hardware a través de un potenciómetro, ambos métodos precisan de unas soluciones proporcionadas con el sensor de pH 4.00 y 10.00.

En este proyecto, no se dispone del sensor de pH comentado, por lo que se simulará mediante el uso de un potenciómetro su funcionamiento.

Si colocásemos un potenciómetro alimentado por los 5V que proporciona Arduino, el rango de tensión que introduciríamos a Arduino sería de 0V a 5V.

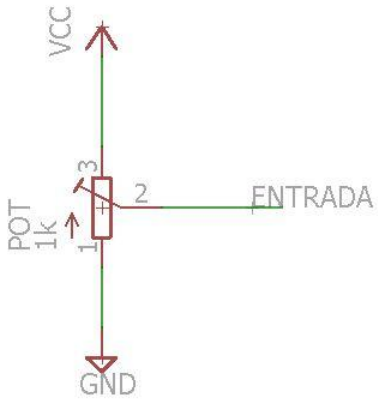


Figura 6-20. Esquemático conexión potenciómetro

Este rango de tensiones debe ser modificado ya el sensor de pH tiene un rango de tensión diferente a este. Su rango de salida de -414.12mV a 414.12mV. Como la simulación de tensiones negativas es problemática, se considerará un rango de salida de 0V a 828.24mV. Si se contara con el sensor, no habría que preocuparse de estas tensiones, ya que el sensor cuenta con una pequeña placa de circuito impreso que, situada entre éste y el Arduino, realiza las conversiones necesarias para su correcto funcionamiento.

La forma de conseguir que, mediante el uso de un potenciómetro, se obtenga un valor de tensión a la entrada de Arduino como la proporcionada por el sensor es realizando el siguiente circuito, que conseguirá mediante la incorporación de una resistencia de valor fijo que el rango de tensión sea el deseado.

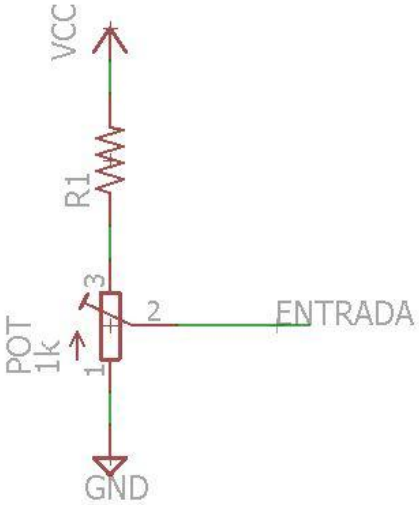


Figura 6-21. Esquemático conexión potenciómetro con resistencia en serie

De esta forma, la tensión a la entrada del Arduino (V_{in}) tomaría el siguiente valor:

$$V_{in} = \frac{R_{potenci\ ómetro}}{R_{potenci\ ómetro} + R_{fija}} \cdot V_{cc} \rightarrow R_{fija} = \frac{R_{potenci\ ómetro} \cdot V_{cc}}{V_{in}} - R_{potenci\ ómetro}$$

Donde $R_{potenci\ ómetro} = 1k\Omega$, $V_{cc} = 5V$ y V_{in} es 828.24mV.

Conocidos estos datos se puede determinar el valor de la resistencia fija que se deberá colocar para llegar al rango de tensión deseado.

$$R_{fija}^{pH} = \frac{1k\Omega \cdot 5V}{828.24mV} - 1k\Omega = 5k\Omega$$

La resistencia que se usará en el montaje tendrá un valor de 4.7kΩ, lo que hará que el valor aportado por el potenciómetro no sea exactamente el mismo que el sensor, pero sí sea bastante similar.

6.5.4 Sensor de conductividad

La empresa DFRobot ha desarrollado el sensor DFR0300, que es un sensor analógico de conductividad para su uso con Arduino. La conductividad es la habilidad de una sustancia de conducir la corriente, y su medida en el sistema internacional son los Siemens/metro, S/m. También es común ver otras unidades como mS/cm o µS/cm. [10]

Los valores característicos más importantes de este sensor son los siguientes:

Tabla 6-8. Características principales sensor de conductividad

Especificaciones	Valores
Tensión de operación	5V
Rango de medida	1mS/cm – 20mS/cm
Precisión	±10%

Este dispositivo incluye un electrodo que mide la conductividad con una conexión BNC a una placa PCB que sirve como etapa de procesamiento de datos antes de la lectura del Arduino. Además cuenta con un sensor de temperatura DS18B20, explicado anteriormente.

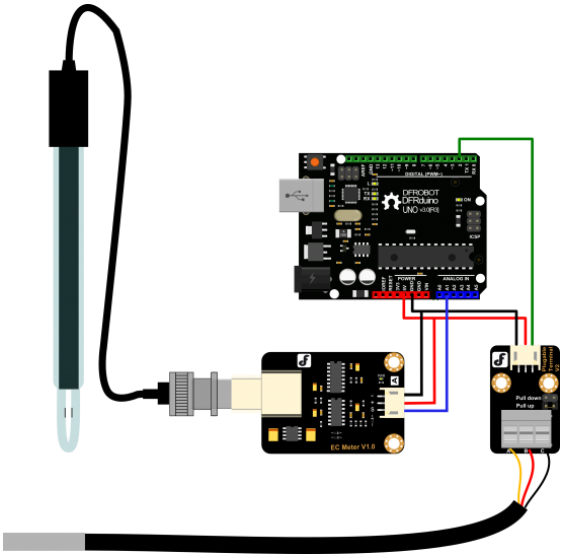


Figura 6-22. Esquema de conexión sensor de conductividad

Aunque este sensor incluya un hardware y un software que te permite hallar el valor de la conductividad sin apenas complicación, sería conveniente conocer los principios teóricos y la relación que guarda la conductividad con la tensión dada por el sensor.

En el esquemático, la función de transferencia del chip U3B es la siguiente:

$$V_0 = \frac{R_{10}}{R} \cdot V_i$$

Donde R_{10} es una resistencia de realimentación cuyo valor es 820Ω. La resistencia R es la que surge al introducir el electrodo en el agua, relacionada con la conductividad de la siguiente manera:

$$R = \rho \frac{L}{A}$$

Siendo ρ la resistividad, L la distancia entre las dos láminas conductoras, y A el área enfrentada entre ellas. La conductividad es inversamente proporcional a la resistividad:

$$k = \frac{1}{\rho} = \frac{1}{R} \cdot \frac{L}{A}$$

A la relación 1/R se le denomina conducción G, y a la relación que L/A constante de Vessel Q.

$$k = G \cdot Q$$

Con los cálculos anteriormente realizados se puede llegar a la ecuación que dará el valor de la conductividad en función de la tensión medida:

$$k = \frac{Q}{R_{10} \cdot V_{in}} \cdot V_{out}$$

La constante de Vessel es diferente para cada electrodo. En la documentación proporcionada por el fabricante puede verse como para una V_{out} de 209mV, el valor de la conductividad es 1.41mS/cm. Haciendo uso de esta relación y sabiendo que V_{in} puede aproximarse a una constante de valor 200mV se puede hallar el valor de la constante de Vessel:

$$Q = \frac{k \cdot R_{10} \cdot V_{in}}{V_{out}} = 1106.41 \frac{mS \cdot \Omega}{cm}$$

Quedando finalmente la siguiente relación lineal que guarda la conductividad con la tensión medida.

$$k = 6.746411 \cdot 10^{-3} \cdot V_{out}$$

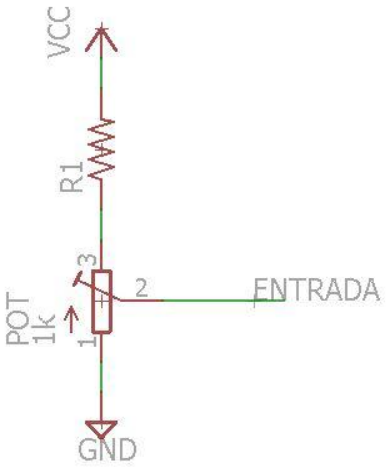
Por tanto el sensor tendrá un rango de salida de

1mS/cm: $\frac{1\frac{mS}{cm}}{6.746411 \cdot 10^{-3}} = 148.24mV$

20mS/cm: $\frac{20\frac{mS}{cm}}{6.746411 \cdot 10^{-3}} = 2.965V$

Al igual que pasa con el sensor de pH, tampoco se dispone en el presente proyecto del sensor de conductividad, por lo que también se simulará su funcionamiento mediante el uso de un potenciómetro.

De forma análoga a lo explicado antes con el sensor de pH, el circuito que resultaría para simular el sensor sería el siguiente:



A la hora de trabajar con Arduino consideraremos un rango de salida de 0 a 2.965V.

De esta forma, la tensión a la entrada del Arduino (V_{in}) tomaría el siguiente valor:

$$V_{in} = \frac{R_{potenci\ ómetro}}{R_{potenci\ ómetro} + R_{fija}} \cdot V_{cc} \rightarrow R_{fija} = \frac{R_{potenci\ ómetro} \cdot V_{cc}}{V_{in}} - R_{potenci\ ómetro}$$

Donde $R_{potenci\ ómetro} = 1k\Omega$, $V_{cc} = 5V$ y V_{in} 2.965V.

Conocidos estos datos se puede determinar el valor de la resistencia fija que se deberá colocar para llegar al rango de tensión deseado.

$$R_{fija}^{EC} = \frac{1k\Omega \cdot 5V}{2.965V} - 1k\Omega = 686.3\Omega$$

La resistencia que se usará en el montaje tendrá un valor de 680Ω, lo que hará que el valor aportado por el potenciómetro no sea exactamente el mismo que el sensor, pero sí sea bastante similar.

6.6 Otros elementos del montaje

6.6.1 Teclado 4x4

Un teclado no es más que una colección de botones, a cada uno de los cuales le asignamos un símbolo o una función determinada.

Para que nuestro Arduino pueda saber que tecla se pulsa, basta con poner tensión en las filas de forma secuencial y luego leer las columnas para ver cuál de ellas tiene HIGH.Los teclados matriciales usan una combinación de filas y columnas para conocer el estado de los botones. Cada tecla es un pulsador conectado a una fila y a una columna. Cuando se pulsa una de las teclas, se cierra una conexión única entre una fila y una columna.

Por ejemplo, ponemos HIGH en la primera fila (hilo 8 en el diagrama de la derecha) y después leemos sucesivamente los hilos correspondientes a las columnas (hilos 4, 3, 2,1). Si ninguno está en HIGH es que no se ha pulsado ninguna tecla de la primera fila.

Pasamos a la segunda fila (hilo 7) y ponemos HIGH, si al leer los hilos 4, 3, 2,1 encontramos que el hilo 1 está en HIGH, es que se ha pulsado la tecla correspondiente a la "B".

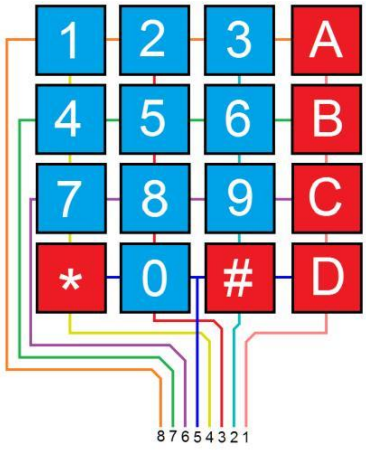


Figura 6-23. Pinout teclado 4x4

La conexión con el Arduino no tiene ninguna complicación, simplemente hay que conectarlo a ocho pines digitales de Arduino. [11]

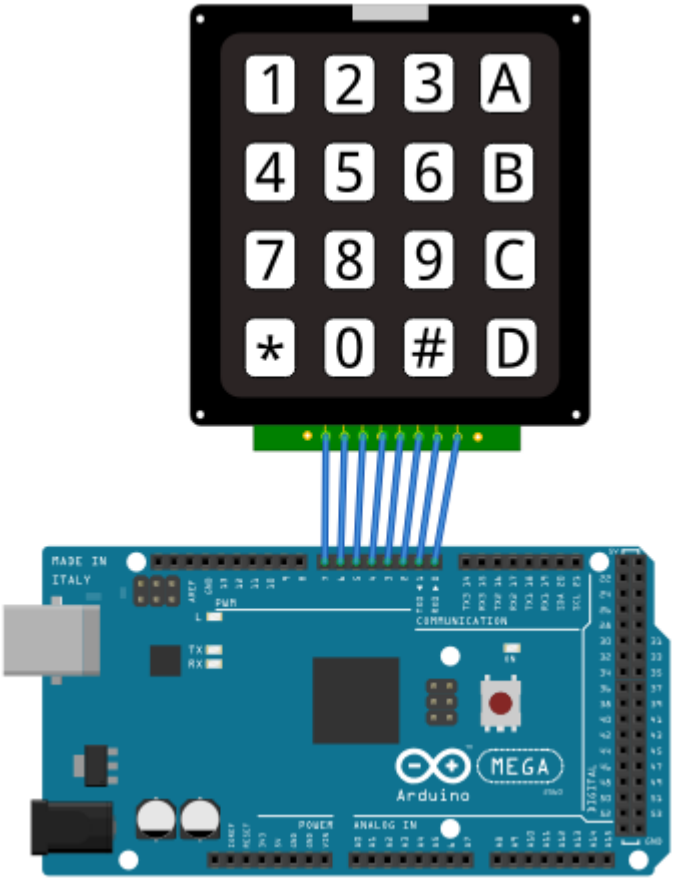


Figura 6-24. Conexión teclado 4x4 a Arduino

6.6.2 Display LCD 16x2

Como necesidad principal del presente proyecto, está la de proporcionar información al usuario sobre los parámetros que se están controlando en el acuario, para ello se usará un display LCD 16x2, es decir, dieciséis caracteres y dos líneas. [12]

La conexión con el Arduino es relativamente sencilla, a pesar de la gran cantidad de conexiones a realizar, como puede verse en la siguiente imagen.

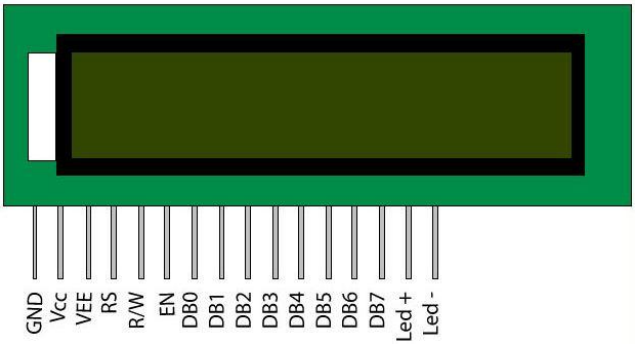


Figura 6-25. Pinout display LCD 16x2

A pesar de que el Arduino Mega cuenta con una gran cantidad de pines, como mejora a la conexión anteriormente propuesta, se usará una conexión mediante comunicación I2C que hará que pasemos de 16 pines a un total de 4, que serán VCC, GND, SDA y SCL. Este modo de comunicación será posible gracias a un

módulo I2C. Esta comunicación será explicada en más detalle en un anexo del trabajo.

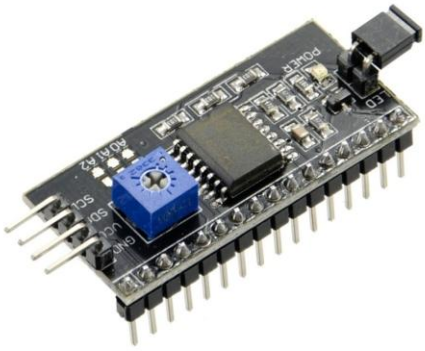


Figura 6-26. Modulo I2C del display LCD 16x2

6.7 Actuadores

Como se expuso anteriormente en el apartado de estudios previos, necesitaremos 6 actuadores que se conectarán a nuestro Arduino: resistencia calefactora, ventilador, iluminación, válvula de CO_2, bomba de llenado y bomba de vaciado. [13]

6.7.1 Resistencia calefactora

Calcular correctamente la potencia que necesita una resistencia para calentar un acuario no resulta sencillo. En ello intervienen el volumen del tanque, la oxigenación del mismo y sobretodo la temperatura exterior al acuario. Por convención, siempre se ha dicho que debe usarse un vatio (W) por litro de tanque. Para hacernos una idea, esto supone, por ejemplo, una resistencia calefactora de 100W de potencia en un acuario de 100L. Si en pleno invierno, nuestra estancia donde está el acuario se encuentra a 10º C, nuestro tanque de cien litros necesitará algo más de 100W. Si la calefacción central del domicilio se pone en funcionamiento alcanzando el ambiente los 21ºC durante la noche, con unos 30W tendríamos suficiente para alcanzar los 25-28ºC de un acuario tropical.

Litros	Temperatura mínima estancia (ºC)	Potencia (W)
100	5	150
100	12	100
100	17	75

Figura 6-27. Resistencia calefactora

Otra opción para calentar el agua podría ser un cable calefactor, que consiste en una resistencia de filamento recubierta por silicona. Es totalmente sumergible y muy manejable lo que nos permite instalarlo haciendo zigzag en el fondo del tanque. Algunos incluyen un grupo de pinzas con ventosa para fijarlas al cristal del fondo del acuario, otra opción es añadir una rejilla a la que fijar el cable como se puede ver en la imagen.

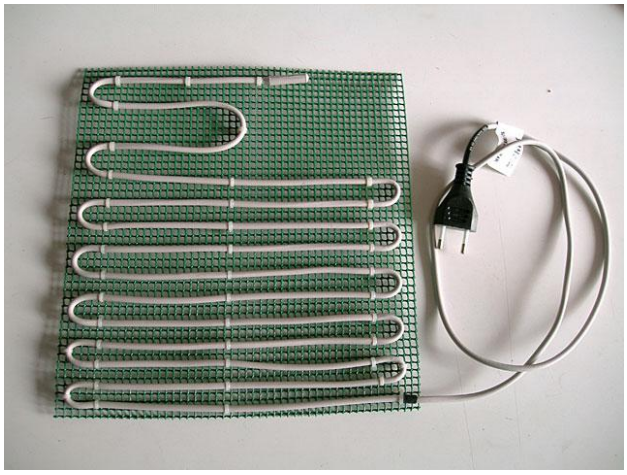


Figura 6-28. Montaje cable calefactor

6.7.2 Ventilador

El principal causante del aumento de la temperatura del agua en un acuario, aunque parezca extraño, no es la temperatura exterior, sino la iluminación que tenemos sobre nuestro acuario, causando una alta temperatura sobre la superficie que hace que el agua se caliente.

El método más económico para conseguir reducir la temperatura del acuario es el uso de un ventilador (dependiendo de la potencia necesaria pueden usarse varios). Con esto se consigue que la temperatura se renueve constantemente.

Pueden usarse tanto ventiladores que se puedan encontrar por el hogar apuntando en la dirección adecuada, como dispositivos diseñados específicamente para el uso en acuarios.

Figura 6-29. Opciones para disminuir la temperatura en un acuario

6.7.3 Iluminación

Cuando hablamos de necesidades de iluminación para nuestro acuario, nos referimos a la cantidad de luz que emiten las lámparas y a la vez a su calidad. Las plantas necesitan una intensidad suficiente para poder realizar la fotosíntesis, y una calidad de luz que se asemeje lo máximo posible a la que emite nuestro sol.

En cuanto a intensidad, surge la pregunta de, ¿qué intensidad tiene que tener la iluminación de mi acuario? Todo depende de las plantas que queramos meter y del tipo de iluminación que tengamos. Básicamente podemos agrupar los tipos de iluminación en dos tipos, fluorescencia y LEDs. A continuación va a explicarse únicamente la iluminación LED, ya que es la más utilizada.

Para este tipo de iluminación la forma de medir la intensidad es mediante la cantidad de lúmenes por litro, y puede clasificarse de la siguiente manera.

Tabla 6-9. Clasificación intensidad de la luz

Nivel de iluminación	Cantidad de lúmenes por litro
Iluminación baja	0-15 lúmenes/litro
Iluminación media	15-25 lúmenes/litro
Iluminación alta	25-35 lúmenes/litro
Iluminación muy alta	>35 lúmenes/litro

Las diferencias entre la iluminación LED y fluorescente son varias, como la eficiencia, siendo la iluminación LED mucho más eficiente, por lo que consiguen la misma potencia lumínica con menos potencia eléctrica. Otra ventaja de los LEDs es que emiten en una única dirección, no siendo necesario añadir elementos para reflejar la luz y así no tener que desperdiciar rayos de luz. Como inconveniente, el precio es mayor, aunque debido a su rendimiento y su larga vida útil, la amortización es casi inmediata.

Figura 6-30. Panel iluminación LED

6.7.4 Bomba de llenado y bomba de vaciado

Para llenar y vaciar el acuario usaremos dos bombas de agua sumergibles, una para realizar el llenado y otra para el vaciado. El funcionamiento de la bomba de agua es sencillo, se coloca dentro del agua, y esta, a través de un tubo del tamaño deseado, bombea el agua hacia el exterior (donde coloquemos el extremo del tubo).

Una vez explicado esto, el montaje en nuestro sistema será el siguiente: una bomba será colocada en el interior del acuario y otra en un pequeño tanque de agua exterior al acuario. La bomba interior, cuando se active la señal que avisa que hemos superado el nivel fijado, se activará comenzando a vaciar el acuario hasta que vuelva de nuevo a ese nivel. La bomba exterior tendrá que activarse cuando el nivel medido esté por debajo del deseado, comenzando así a bombear el agua del tanque exterior al acuario.

Figura 6-31. Bomba de agua sumergible

6.7.5 Válvula de CO_2

Existen diversos métodos caseros para añadir CO_2 al acuario, pero sin duda lo más recomendable es contar con un sistema profesional para ello, que nos permitirá controlar con exactitud la cantidad de burbujas que se añaden al acuario.

Estos sistemas son los más beneficiosos para el acuario, porque gracias a su aportación constante, consiguen que las plantas siempre tengan la misma cantidad de CO_2 disponible para realizar la fotosíntesis, así como aseguran la estabilización del pH en niveles un poco ácidos, facilitando la absorción de nutrientes de las plantas del acuario.

Un sistema profesional de CO_2 cuenta con, al menos, los siguientes componentes:

- Bombona de CO_2: pueden ser de diferentes volúmenes. Y existen tanto bombonas desechables como recargables.

Figura 6-32. Bombona CO_2

- Manoreductor con manómetro: debido a que el CO_2 dentro de la bombona está a una alta presión necesitamos este aparato para disminuirla y poder enviarla al acuario sin peligro. Gracias a la válvula de regulación y a los manómetros, podemos visualizar la presión y ajustar el flujo de CO_2 que sale de la bombona.

Figura 6-33. Manoreductor con manómetro

- Válvula de retención: Para evitar que el agua del acuario acceda al sistema de CO_2, se coloca una válvula antirretorno que solo permite el paso del fluido en una dirección.

Figura 6-34. Válvula de retención

- Cuentagotas: el cuentagotas de vidrio nos permite visualizar el número de gotas que salen de la bombona, para junto al manoreductor, ajustar la cantidad de CO_2 que queremos añadir al acuario.

Figura 6-35. Cuentagotas

- Electroválvula: La electroválvula nos permite cerrar el circuito de CO_2 a través de una señal eléctrica. Esto permite que podamos abrir o cerrar el circuito según las lecturas de nuestros sensores.

Figura 6-36. Electroválvula

6.8 Montaje de las salidas del sistema

En el montaje de este proyecto no se contará con los actuadores mencionados anteriormente, se usarán unos diodos LED que simularán el encendido y apagado de estos actuadores. Los elementos necesarios serían una resistencia y un diodo LED por cada actuador que queremos simular.

Pero la idea de nuestro proyecto es que, si en un futuro se desean integrar los actuadores en lugar de los LEDs, esta integración apenas suponga problema. En el caso de que se contara con los actuadores, estos irán alimentados por la red eléctrica, lo que lleva a pensar que será necesaria la utilización de un dispositivo situado entre el actuador y el Arduino para así evitar quemar el microcontrolador. Este dispositivo es el relé.

Un relé es un dispositivo electromagnético que, estimulado por una corriente eléctrica muy débil (Arduino), abre o cierra un circuito en el cual se disipa una potencia mayor que en el circuito estimulador (actuador conectado a red eléctrica). [14]

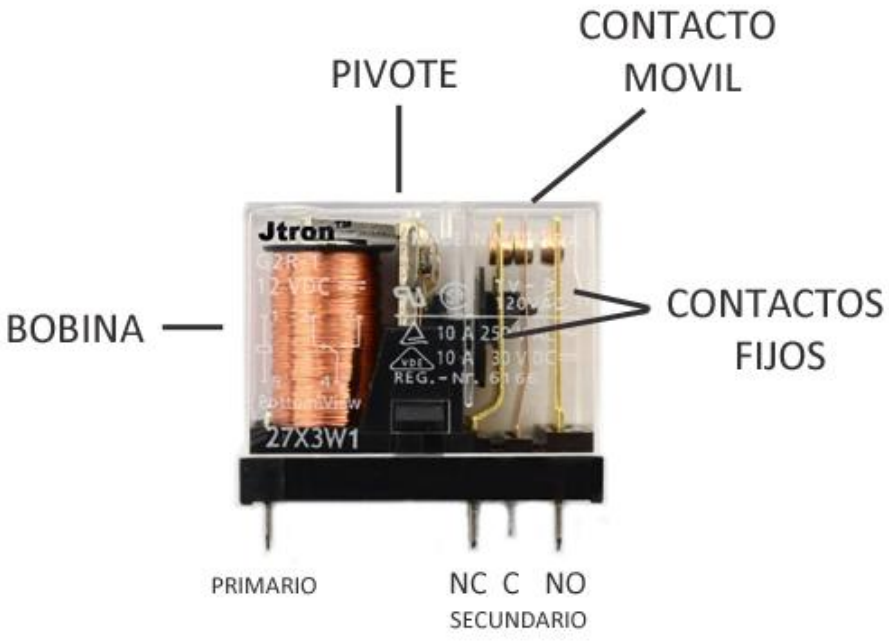


Figura 6-37. Elementos de un relé

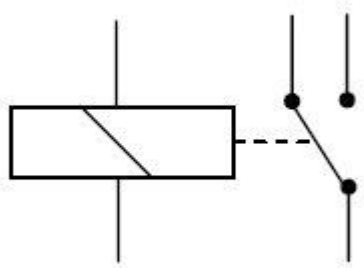


Figura 6-38. Esquemático de un relé

En el símbolo básico del relé se muestra una bobina y un interruptor que se mueve entre dos contactos, aunque existen relés de más contactos.

Los relés existentes en el mercado no precisan demasiada potencia para excitar la bobina, pero en el caso del Arduino esta potencia es demasiado pequeña por lo que habrá que diseñar un circuito que consiga nuestro propósito. En la siguiente imagen se muestra el circuito correspondiente a la activación de un actuador cualquiera (en este caso en lugar del actuador se coloca un diodo LED, LED2).

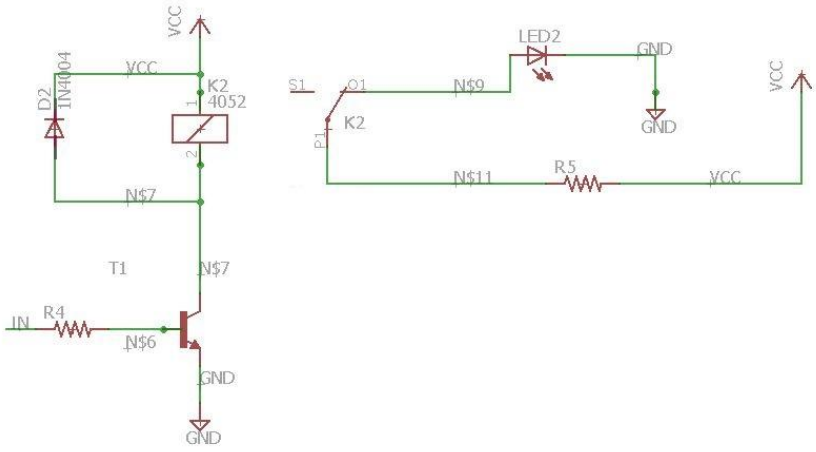


Figura 6-39. Circuito necesario para la conexión de un relé a Arduino

Normalmente los microcontroladores suministran unas tensiones muy pequeñas, en el caso de Arduino de 40mA, no suficiente para activar el relé, por lo que se coloca una etapa de amplificación (resistencia de 10kΩ y transistor BJT NPN), para así conseguir la activación del relé.

El diodo que se encuentra en paralelo con la bobina se necesita para eliminar picos de corriente producidos por esta.

Como alternativa al montaje del circuito mostrado anteriormente, existen en el mercado unas placas que incluyen el relé y la circuitería necesaria para realizar la conexión directamente a Arduino. Su principal ventaja frente a la realización del circuito anterior, es el tamaño, mucho más reducido. Existen desde placas con un solo relé a placas con gran cantidad de relés. En nuestro proyecto se usarán dos placas de cuatro relés como la mostrada a continuación.

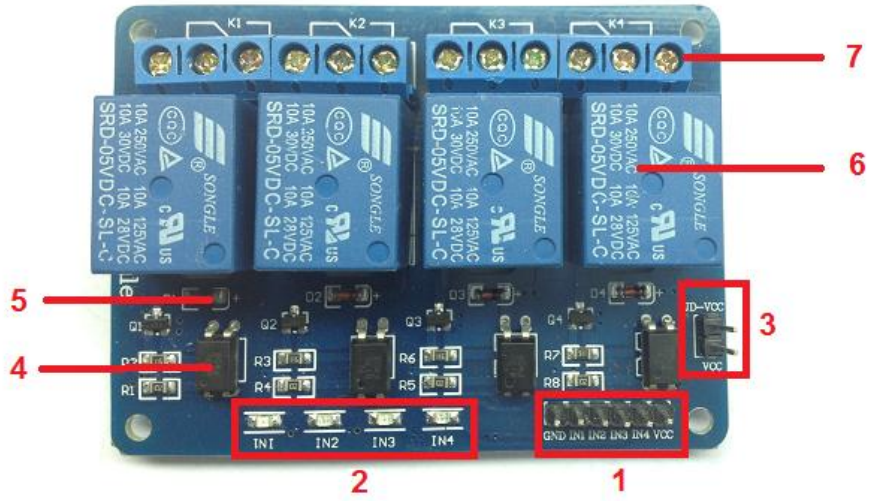


Figura 6-40. Elementos de un módulo de cuatro relés

Como se puede apreciar, la placa consta de:

1. Conector de entradas (IN1 a IN4) y alimentación (GND y VCC).
2. Cuatro LEDs que indican el estado de las entradas.
3. Un jumper selector para la alimentación de los relés.
4. Cuatro optoacopladores del tipo FL817C.
5. Cuatro diodos de protección.
6. Cuatro relés marca SONGLE con bobinas de 5V y contactos capaces de controlar hasta 10 Amperios y una tensión de 250V.
7. Cuatro bornas, con tres contactos cada una (Común, Normalmente abierto y Normalmente cerrado), para las salidas de los relés.

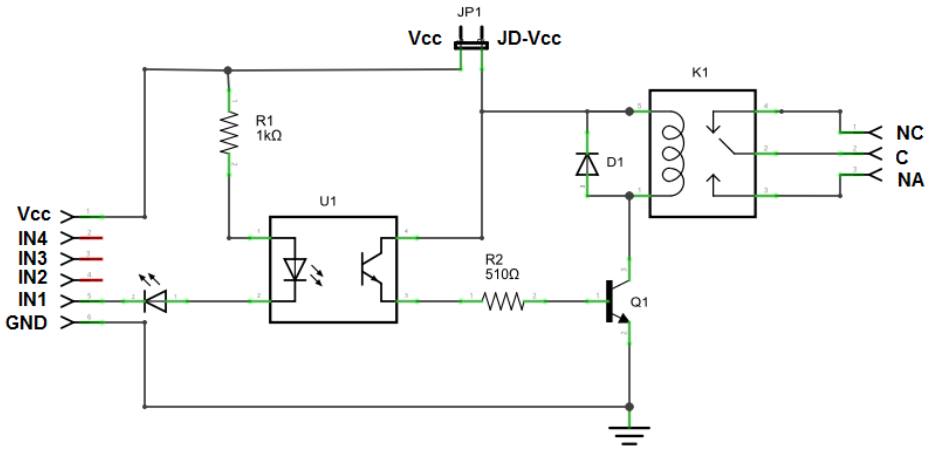


Figura 6-41. Esquemático de un canal del módulo de cuatro relés

A raíz del circuito anterior, puede analizarse el funcionamiento de este. La entrada IN1 está conectada al cátodo del diodo del optoacoplador a través del LED indicador. El ánodo del diodo del optoacoplador se conecta a VCC mediante R1, una resistencia de 1kΩ. Estos tres componentes, el diodo indicador, el diodo del optoacoplador y R1 forman un circuito serie por donde circula la corriente si la entrada está a un nivel bajo (GND) y no circula corriente si la entrada está a un nivel alto (VCC).

El transistor del optoacoplador tiene su colector a JD-VCC y su emisor conectado al transistor Q1 a través de R2, una resistencia de 510Ω. Este es otro circuito serie donde circula corriente cuando el transistor del optoacoplador conduce (diodo LED encendido), con lo que se introduce corriente en la base de Q1 a través de

R2.

Por último, el transistor Q1 está conectado en una configuración emisor común, con su emisor a tierra (GND) y la bobina del relé como carga en el colector. Cuando circula corriente por la base desde el optoacoplador, Q1 se satura permitiendo el paso de la corriente a través de la bobina del relé, lo que produce que se cierren los contactos del mismo (común con normalmente abierto). El diodo D1 protege al transistor de la tensión que aparece en la bobina del relé cuando deja de circular corriente por la misma.

A grandes rasgos puede resumirse lo comentado de la siguiente forma: al ponerse la entrada **a** nivel bajo satura el transistor Q1 a través del optoacoplador con lo que se cierra el contacto normalmente abierto del relé.

Un aspecto importante que también hay que tener en cuenta sobre el uso de la placa de relés es el consumo y la alimentación.

La forma más sencilla de alimentar este módulo es usando la alimentación VCC y GND del Arduino, manteniendo el Jumper en su lugar, con lo que JD-VCC sería la VCC de Arduino. Si realizamos esta conexión, hay que considerar dos limitaciones importantes:

- Se pierde la aislación eléctrica que proporcionan los optoacopladores, lo que aumenta la posibilidad de daño al Arduino si hay algún problema con las cargas de los relés.

- La corriente consumida por las bobinas de los relés debe ser proporcionada por el Arduino. Cada bobina consume unos 30mA. Si en nuestro montaje vamos a tener seis relés correspondientes a los seis actuadores que van a usarse, esto conllevaría una corriente de aproximadamente 180mA. Es necesario considerar siempre estas corrientes ya que un puerto USB proporciona 500mA y puede ser necesario que se realice otro tipo de alimentación en lugar del USB para que aumentara este límite de corriente.

La forma más segura de alimentación es quitar el jumper y alimentar la placa de relés con dos fuentes: la de la placa Arduino conectada a VCC y una segunda fuente, con el positivo a JD-VCC y el negativo a GND, sin estar éste unido al Arduino. Esta conexión presenta dos ventajas:

- Hay una completa aislación entre la carga y el Arduino.

- Todo el consumo de los relés es tomado de la segunda fuente y no del Arduino o el puerto USB.

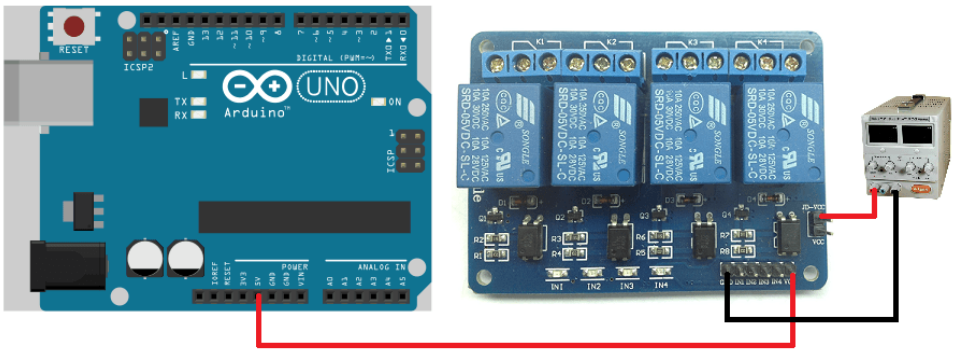


Figura 6-42. Conexión del módulo de cuatro relés a Arduino usando una fuente externa

6.9 Montaje final del sistema

Tras todo lo explicado hasta ahora, el esquema de nuestro sistema (diseñado usando el software Fritzing [15]) quedaría de la siguiente manera:

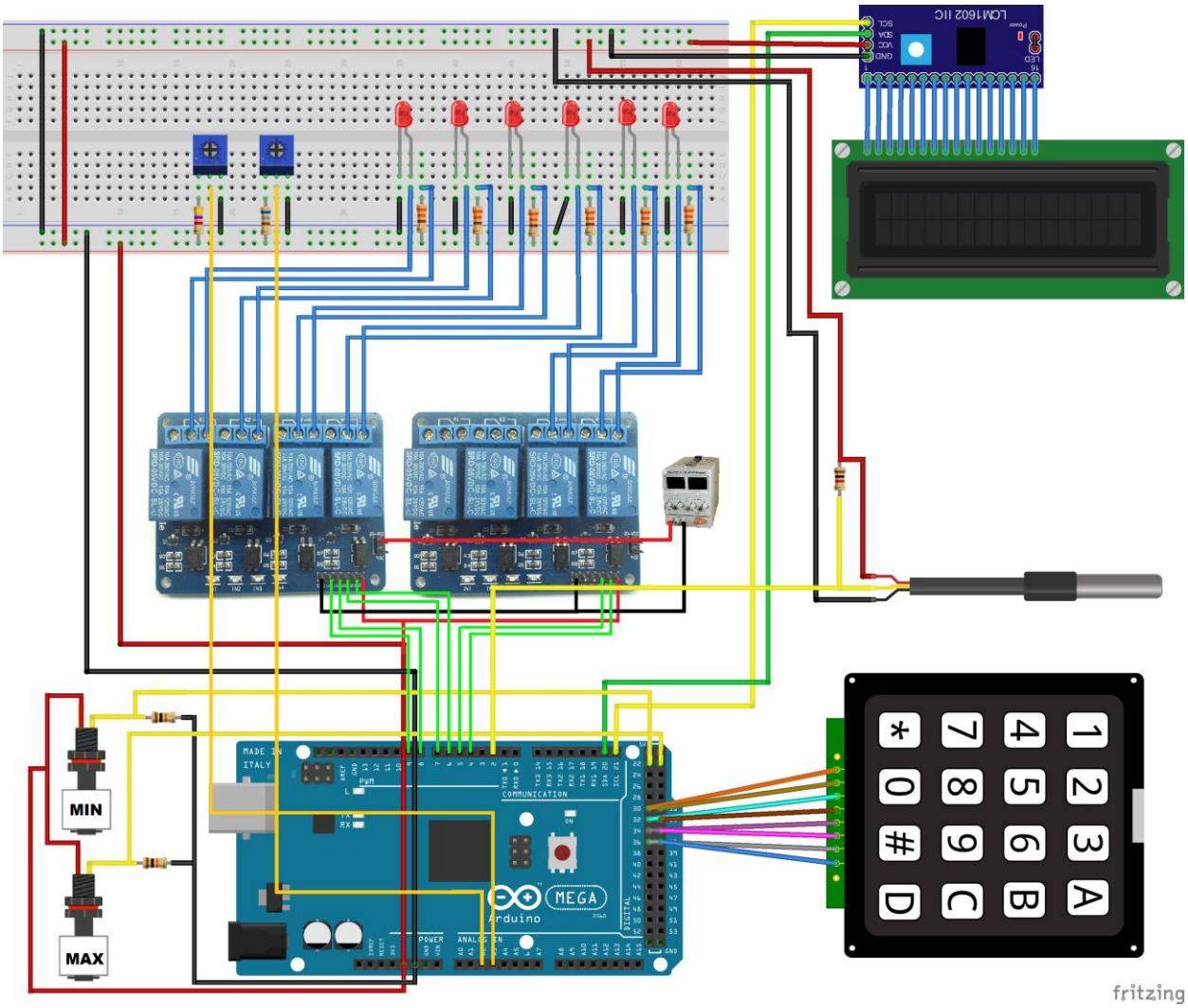


Figura 6-43. Montaje final del sistema usando placa de pruebas

En la siguiente tabla queda reflejado, a modo de resumen, los pines que se usan de nuestro Arduino Mega y cuales son sus funciones.

Pin	Función que desempeña	Elemento conectado
D2	Sensor	Sensor de temperatura DS18B20
D4	Actuador	Resistencia calefactora
D5	Actuador	Ventilador
D6	Actuador	Bomba de llenado
D7	Actuador	Iluminación
D8	Actuador	Bomba de vaciado

D9	Actuador	Electroválvula de CO_2
A2	Sensor	Potenciómetro que simula el sensor de conductividad
A3	Sensor	Potenciómetro que simula el sensor de pH
D30-D37	Entrada	Teclado matricial 4x4
D22	Sensor	Boya de nivel mínimo
D23	Sensor	Boya de nivel máximo
GND, 5V	Alimentación	Todos
D20	Serial Data	Display LCD 16x2
D21	Serial Clock	Display LCD 16x2

6.10 Diseño del PCB

Una vez probado en la placa de pruebas el correcto funcionamiento del proyecto, se procede a diseñar un pequeño circuito en PCB donde se ubicarán los componentes situados sobre la placa de pruebas. Para ello se hará uso de un software gratuito llamado Eagle. [16]

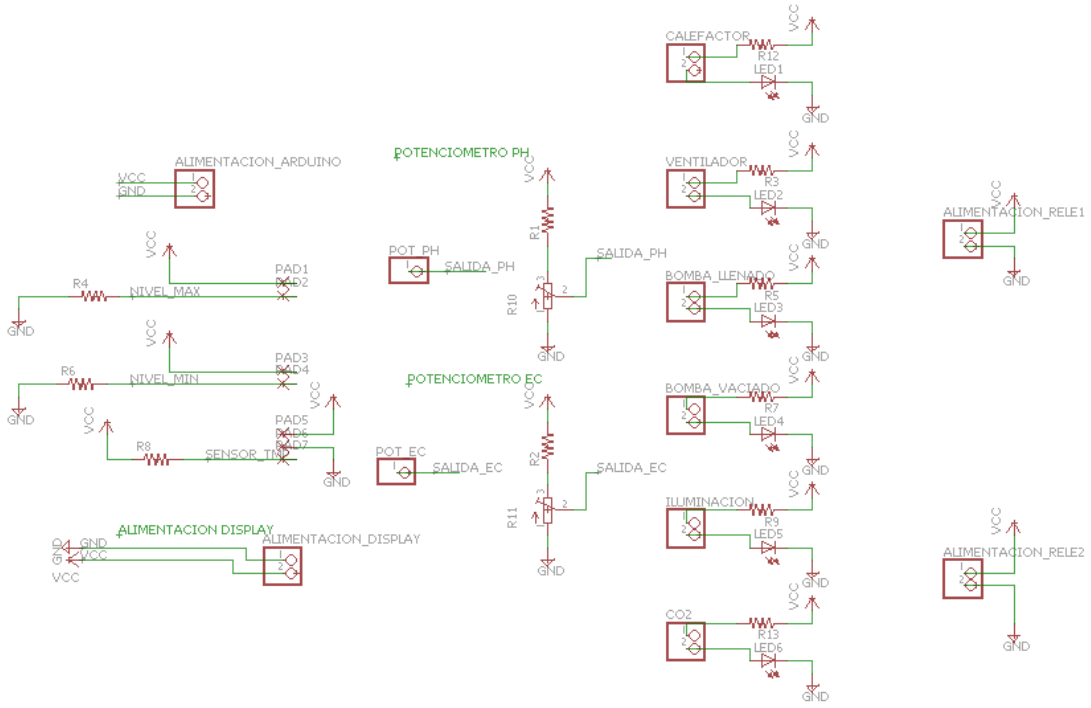


Figura 6-44. Esquemático completo en Eagle

Tras realizarse el esquemático y las pertinentes conexiones en Eagle, se procede a la colocación y conexión de los componentes sobre la placa PCB de la siguiente manera

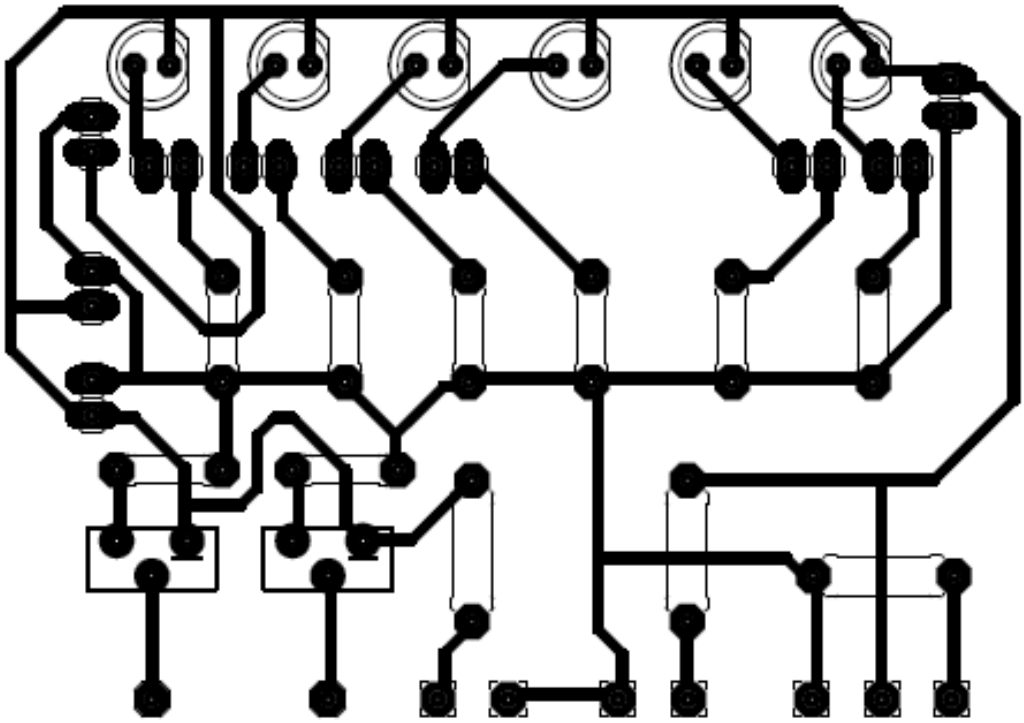


Figura 6-45. Diseño layout del circuito.

Tras varias pruebas se consigue una colocación adecuada de los componentes como se muestra en la anterior figura. El siguiente paso es eliminar las capas no necesarias y dejar solo los pads y las vías, para continuar con el proceso de fabricación de la placa PCB.

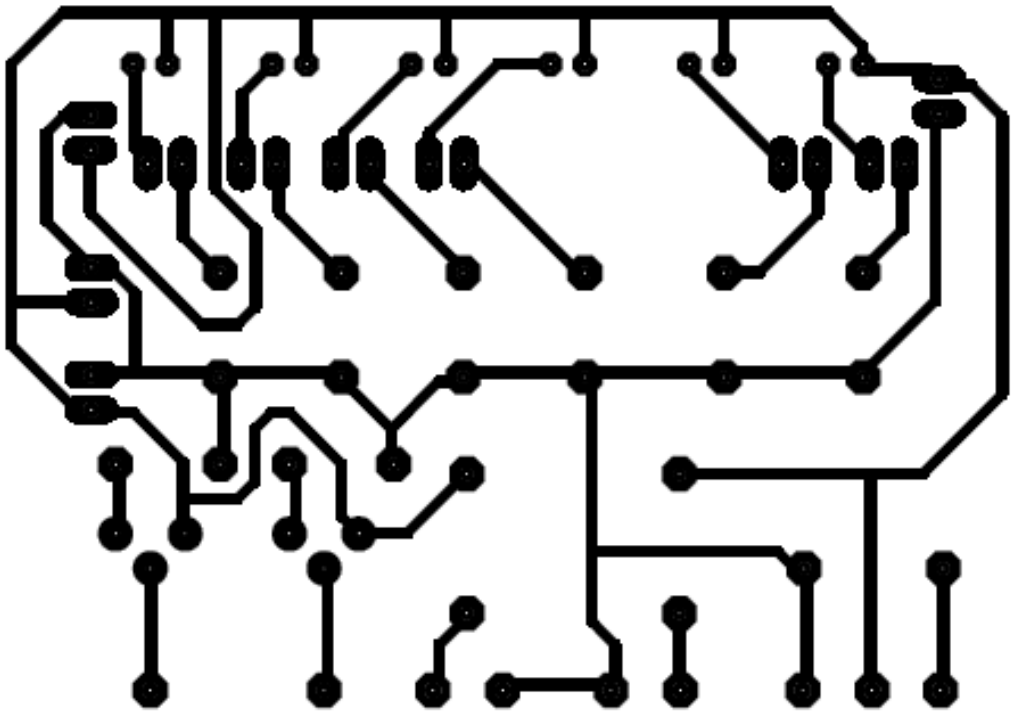


Figura 6-46. Diseño layout del circuito mostrando solo vías y pads

Ya que los componentes utilizados no son demasiados, basta con el uso de una sola capa para su diseño, capa

bottom. Para facilitar el proceso de fabricación del PCB y que el diseño sea más robusto, se añade un plano de tierra, quedando de la siguiente manera.

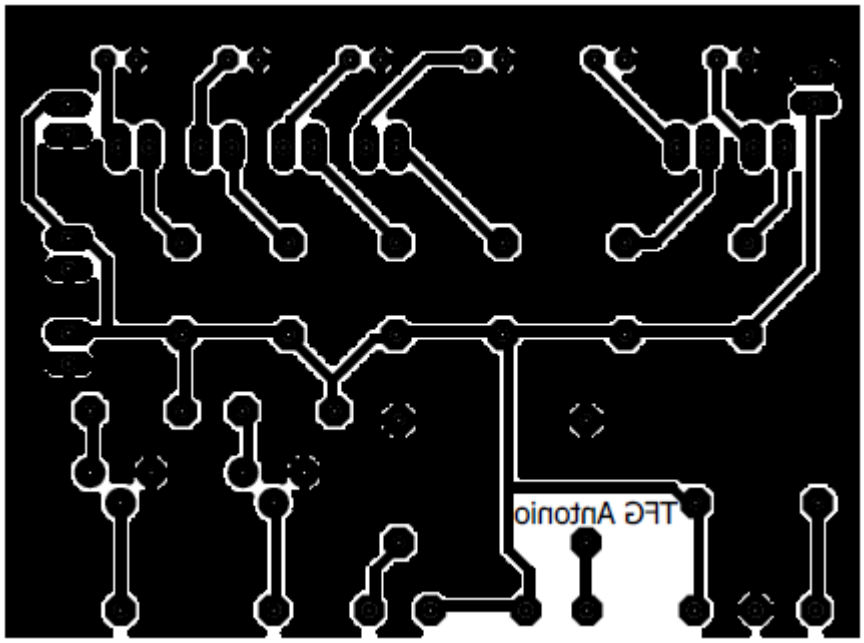


Figura 6-47. Diseño layout del circuito con el plano de tierra añadido

6.10.1 Proceso de fabricación del PCB

El método utilizado para la fabricación del PCB es la fotolitografía. Los pasos de este método de fabricación son los siguientes:

1. Se imprime el PCB en papel de acetato, en nuestro caso la cara bottom.
2. El siguiente paso es preparar las disoluciones para el revelado de la placa: para preparar el líquido revelador se disuelve una cucharada de bicarbonato de sodio en un litro de agua; para el líquido atacante la proporción será 100cc de agua fuerte, 100cc de agua oxigenada y 100cc de agua.
3. Después de tener listas las disoluciones, se comienza a revelar el PCB. Se hace uso de una insoladora de rayos UVA que elimina la resina de aquellas zonas expuestas directamente a la luz. En un ambiente lúgubre, se despega la cubierta adhesiva de la placa para dejar al descubierto su cara fotosensible, colocada hacia arriba, superponiendo el papel de acetato con el circuito impreso, teniendo cuidado de colocarlo de la forma correcta. En la placa aparece el circuito que ve directamente el observador. Después, se hace el vacío y durante aproximadamente 3 minutos se deja actuar la radiación, este proceso se vuelve a repetir para la otra cara de la placa con cuidado de no mover el papel de acetato para que los pasos de cara coincidan.
4. Una vez sacada la placa, ésta se sumerge en el líquido revelador, que actuará de capa protectora sobre las zonas de la placa que no han sido insoladas; tras varios minutos se saca de la disolución y se enjuaga.
5. Justo después se introduce en el líquido atacante, tras varios minutos se enjuaga y se seca. El líquido revelador es reutilizado, pero el líquido atacante pierde su poder corrosivo por lo que se deposita en un recipiente para su correspondiente tratado.

6.10.2 Montaje de los componentes

Antes de taladrar la placa hay que retirar la resina que protege al cobre de su oxidación, para ello se utiliza

acetona. Esto facilita que el estaño se adhiera correctamente. Después de esto, se taladra la placa para conseguir soldar los componentes THD, que en nuestro montaje son todos.

La soldadura que se realizará será manual, con estaño para facilitar la soldadura de los componentes a la placa. Las soldaduras se realizarán de menor a mayor tamaño, evitando que los componentes más grandes dificulten las demás soldaduras.

Una vez fabricada la placa PCB, bastaría con sustituirla por la placa de pruebas mostrada en la ***Figura 6-43***. De esta forma se consigue más robustez en el diseño.

7 DESARROLLO SOFTWARE

7.1 IDE Arduino

Las siglas IDE significan entorno de desarrollo integrado, y puede definirse como la herramienta que nos permite desarrollar nuestras aplicaciones de una manera cómoda, ofreciendo ayudas en cuanto a la sintaxis, plantillas y opciones para depurar.

Un IDE debe tener una serie de características, entre las que destacan: multiplataforma, soporte para diversos lenguajes de programación, reconocimiento de sintaxis y capacidad de importar y exportar proyectos.

A continuación se hará un pequeño resumen del IDE estándar de Arduino, aunque existen otras muchas alternativas posibles, como Fritzing, a la que se hará referencia después.

El IDE oficial de Arduino posee una interfaz muy sencilla e intuitiva, en la cual se pueden apreciar diversas zonas: [17]

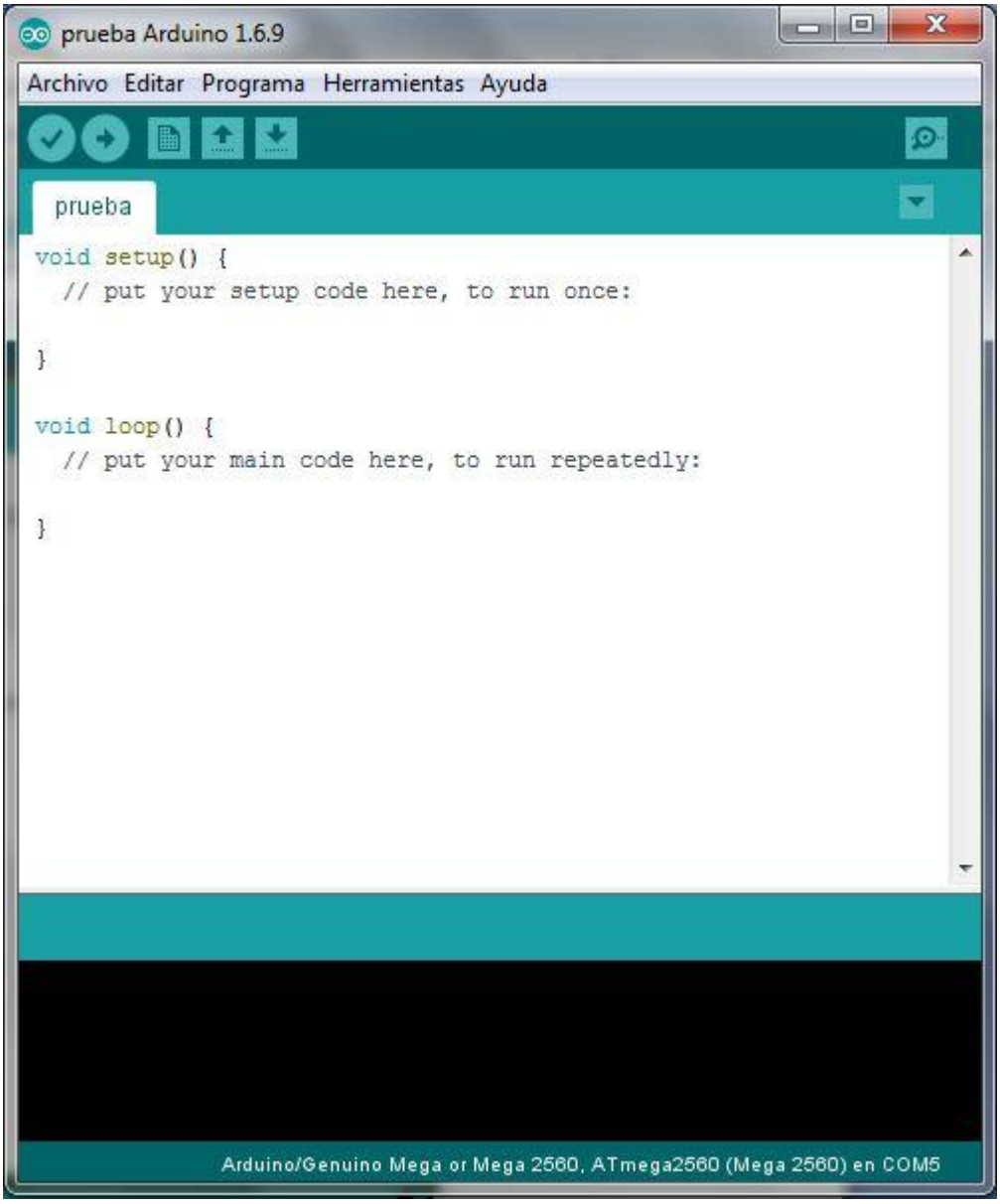


Figura 7-1. Interfaz gráfica del IDE Arduino

1. Menú. En esta barra se encuentran muchas funciones como la carga de un fichero, las librerías, ejemplos, selección de la placa usada y el puerto por el que se comunica etc.

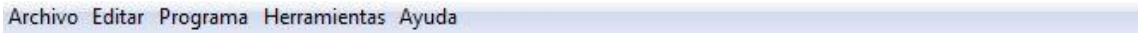


Figura 7-2. Menú del IDE de Arduino

2. Menú de acceso rápido. Botones con las funciones más usadas del IDE. Según el orden en el que aparecen en la imagen son:
   i. Verificar. Comprueba la sintaxis del programa y si es posible su funcionamiento.
   ii. Subir. Carga el programa en la placa.
   iii. Nuevo. Crea un nuevo sketch.
   iv. Abrir. Abre el archivo deseado.
   v. Salvar. Guarda el código en la ruta especificada.
   vi. Monitor serie. Abre una ventana donde aparece la comunicación con la placa Arduino. Muy útil a la hora de depurar código.

Figura 7-3. Menú de acceso rápido del IDE de Arduino

3. Editor de texto. En este cuadro se puede editar el código.

Figura 7-4. Editor de texto del IDE de Arduino.

4. Panel de mensajes. Aparecen mensajes como el proceso de compilación o si ha ocurrido algún error. En la imagen anterior se ha incluido la línea “led” para observar como devuelve el fallo al intentar su compilación.

Figura 7-5. Mensaje de compilación del IDE de Arduino

Figura 7-6. Mensaje de error del IDE de Arduino

5. Consola. Muestra la misma información que la zona de mensajes pero con más detalle. En la imagen puede verse el error que produce la línea "led".

Figura 7-7. Consola del IDE de Arduino

7.2 Programa de control utilizado

En esta sección se explicará sin entrar en demasiado detalle el programa usado, diferenciando las funciones creadas específicamente para este proyecto, y las que se han tomado de librerías predefinidas o creadas previamente por otros diseñadores.

7.2.1 Funciones creadas

En el código pueden diferenciarse cuatro tipos de funciones que se explicarán a continuación:

- Inicialización.
- Manejo de los menús (display).
- Funciones usadas para fijar parámetros.
- Lectura de sensores.
- Comunicación con el puerto serie.

Antes de comenzar a explicar a grandes rasgos el funcionamiento del programa, hay que destacar que se apoyará en el uso de un Display LCD y un teclado 4x4 para así dar la posibilidad de interactuar al usuario, esto hay que tenerlo en cuenta para la comprensión de lo que se explicará a continuación.

7.2.1.1 Inicialización y menús

La estructura de programación de Arduino consta de dos funciones imprescindibles para el funcionamiento de este. Se trata de la función setup y la función loop. La primera de ellas inicializa el programa, se ejecuta una sola vez al comenzar el programa. La segunda será donde se encuentre el código a ejecutar y se repetirá de manera cíclica, de ahí su nombre.

En nuestra función setup se encontrará la inicialización de la comunicación con el display LCD, con el puerto serie (usado principalmente para la depuración del código) y con el sensor de temperatura DS18B20. También se fijarán los pines del Arduino como entrada o salida según vayan a usarse. Por último muestra en pantalla un mensaje de bienvenida y llama a la función "menuInicio", donde entraría automáticamente al modo lectura, donde continuamente leería los parámetros que son de interés (se explicará más adelante en detalle). De este modo de lectura se saldría pulsando la tecla A. Una vez pulsada, se mostraría el menú inicial ofreciendo la posibilidad de modificar los parámetros, visualizarlos, o bien entrar en modo lectura de nuevo.

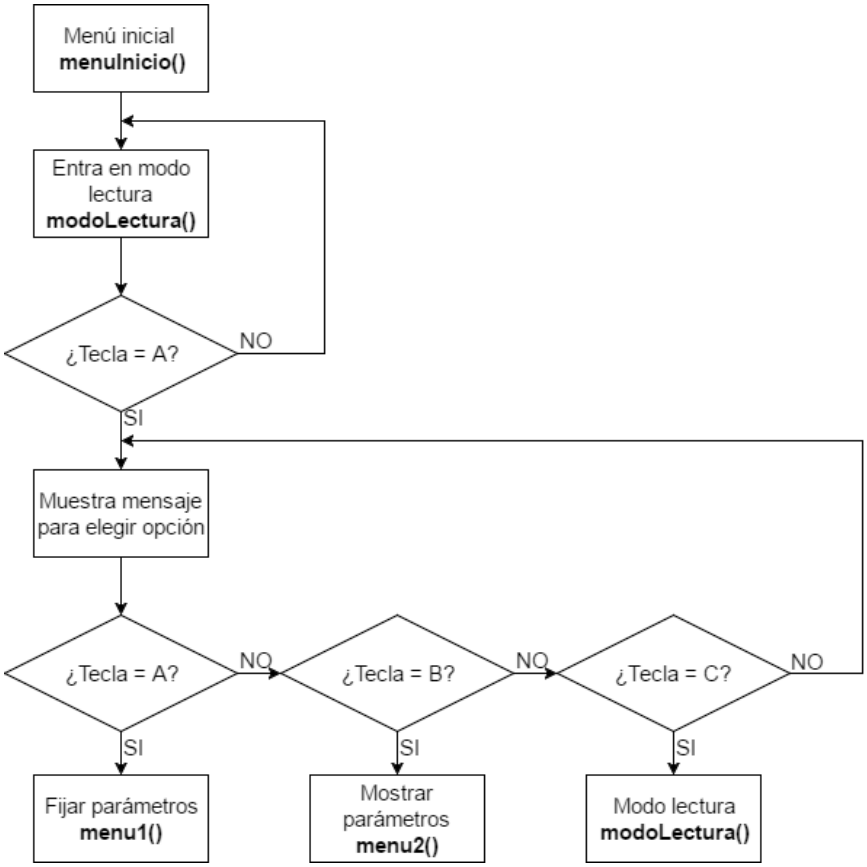


Figura 7-8. Diagrama de flujo del menú de inicio

Al arrancar nuestro sistema, se entraría automáticamente en el modo lectura donde, como se ha comentado antes, se leen los parámetros de importancia. Esta lectura se realiza en bucle hasta que se pulse la tecla A.

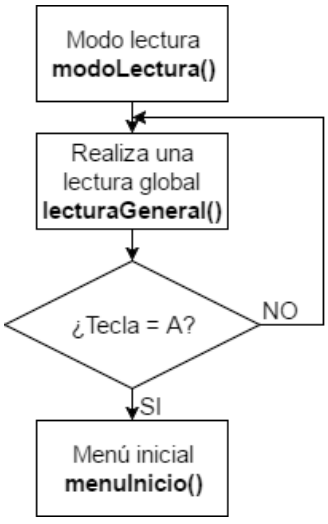


Figura 7-9. Diagrama de flujo del "modo lectura"

Los parámetros de importancia son:

- Lectura de la temperatura a través del sensor DS18B20.
- Lectura del nivel de llenado mediante las boyas de nivel máximo y mínimo.
- Lectura del pH a través del potenciómetro que simula el sensor.

- Lectura de la conductividad a través del potenciómetro que simula el sensor (y posterior conversión para saber el nivel de dureza).
- Lectura del puerto serie. A través del monitor serie pueden visualizarse los parámetros deseados, introduciendo la tecla correspondiente a cada parámetro.
    - Tecla ‘t’. Se muestra la temperatura medida y la temperatura fijada.
    - Tecla ‘p’. Se muestra el nivel de pH medido.
    - Tecla ‘d’. Se muestra el nivel de conductividad que proporciona el potenciómetro y el nivel de dureza en distintos sistemas de medida.
    - Tecla ‘n’. Se muestra si el nivel de agua es el correcto, es superior al máximo o inferior al mínimo.
    - Tecla ‘l’. Se muestra el número de horas fijadas para la iluminación.
    - Tecla ‘c’. Se muestra el número de horas fijadas para la electroválvula de CO_2.
- Lectura del tiempo transcurrido para la luz.
- Lectura del tiempo transcurrido para la electroválvula.

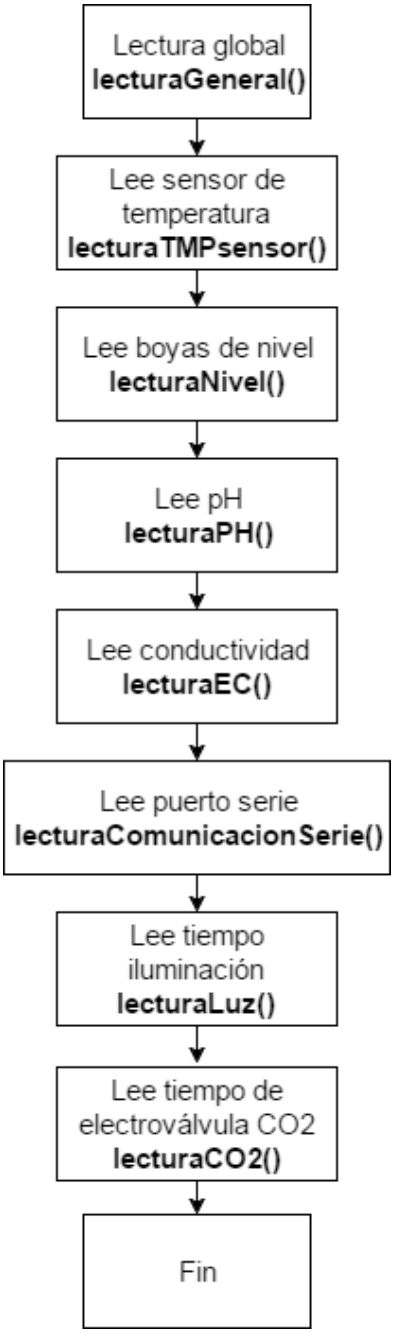


Figura 7-10. Diagrama de flujode la función "lecturaGeneral"

Si se escoge la opción de modificación de parámetros, se podrán modificar tres parámetros, la temperatura, las horas de iluminación y las horas de activación de la electroválvula de CO_2. Si no se desea modificar ningún parámetro se ofrece también la posibilidad de volver al menú de inicio.

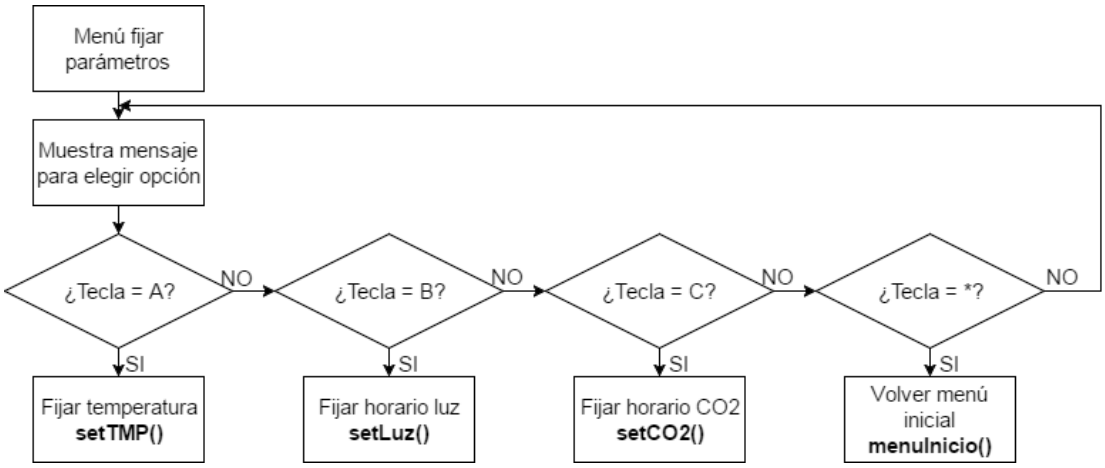


Figura 7-11. Diagrama de flujo del menú "fijar parámetros"

Si se selecciona la opción de mostrar parámetros, se dirigirá otra pantalla donde, una a una con una separación de cuatro segundos, se mostrarán los valores de los parámetros que son de interés para el control del acuario. Una vez mostrados, se da la opción de volver a mostrarlos o bien volver al menú de inicio.

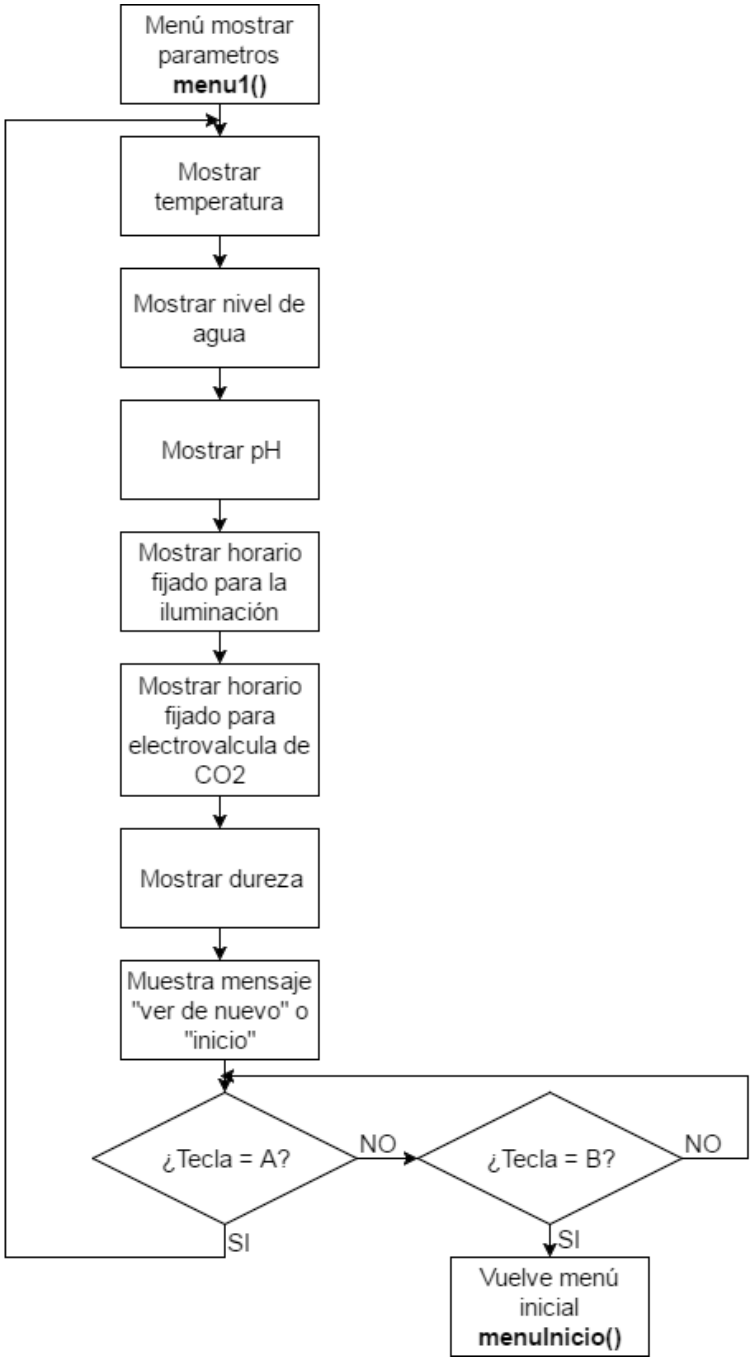


Figura 7-12. Diagrama de flujo del menú "mostrar parámetros"

7.2.1.2 Funciones usadas para fijar parámetros

Se ofrece la posibilidad de fijar la temperatura, el tiempo que estará iluminado nuestro acuario y el tiempo que estará activa la electroválvula de CO_2.

Para fijar el nivel de temperatura se pide que se escriba por teclado el nivel fijado, comprobando siempre si la tecla pulsada es un número. Si no fuese un número habría que comprobar si se ha pulsado la tecla "aceptar", correspondiente a la letra D, o la tecla "cancelar", correspondiente a la letra C. Una vez que se ha fijado el número y se ha pulsado la tecla "aceptar" deben convertirse las teclas pulsadas a un valor tipo entero, ya que la función de lectura por teclado convierte a caracteres.

En cuanto al horario de iluminación o de activación de la electroválvula de CO_2, la función usada la misma,

con la diferencia de que nos piden un número de horas.

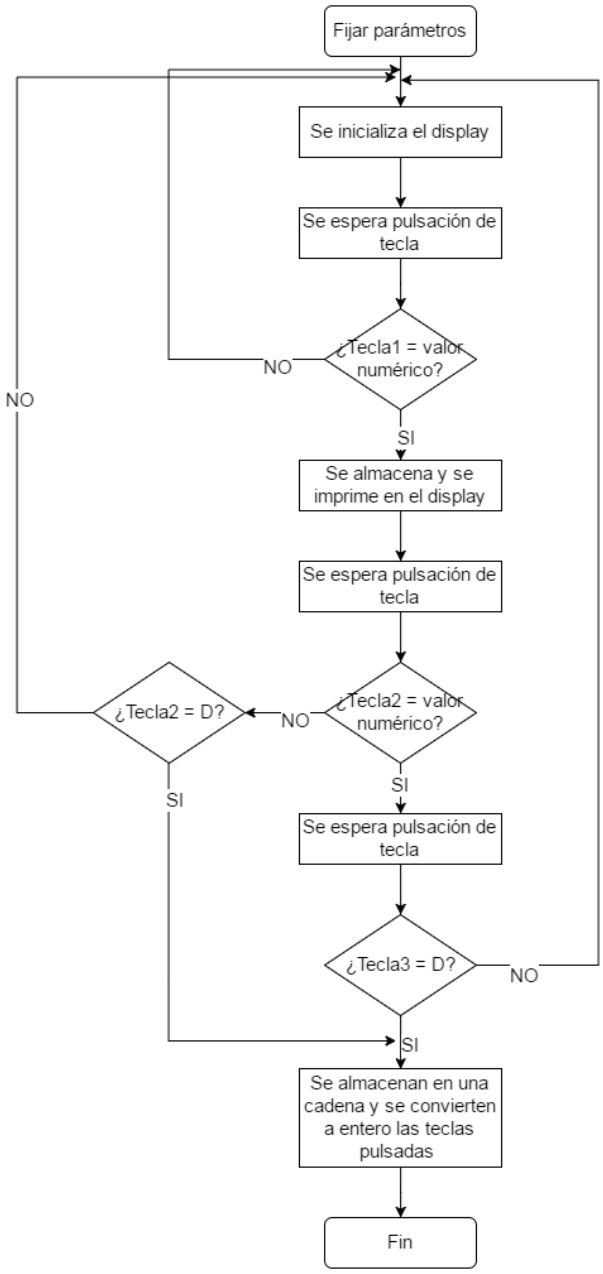


Figura 7-13. Diagrama de flujo genérico para fijar parámetros

Para trabajar con tiempos, entre las funciones elementales que proporciona Arduino, encontramos la función "millis" [18], la cual proporciona el tiempo en milisegundos desde que se arrancó el Arduino. Por lo que el funcionamiento sería muy sencillo, se llama periódicamente a la función millis y se compara ese tiempo con el rango de horas fijado, y dependiendo si se ha pasado ya el tiempo fijado o no se procederá al encendido o apagado de la iluminación o de la electroválvula.

Una vez se recibe las horas de encendido deseadas, se pasan a milisegundos para poder compararse con el valor devuelto por la función millis.

7.2.1.3 Lectura de sensores

Nuestro sistema simulará los sensores de pH y conductividad mediante el uso de potenciómetros. El posterior estudio del voltaje suministrado determinará los niveles reales. A lo largo del código se realizarán lecturas de estos valores para evitar grandes tiempos entre una lectura y otra. Si estos niveles medidos están fuera de lo

establecido se activaría un actuador, que en nuestro caso consiste en la activación de un relé que enciende un LED.

En el siguiente diagrama se puede apreciar cómo se realizaría la lectura analógica de estos sensores (potenciómetros).

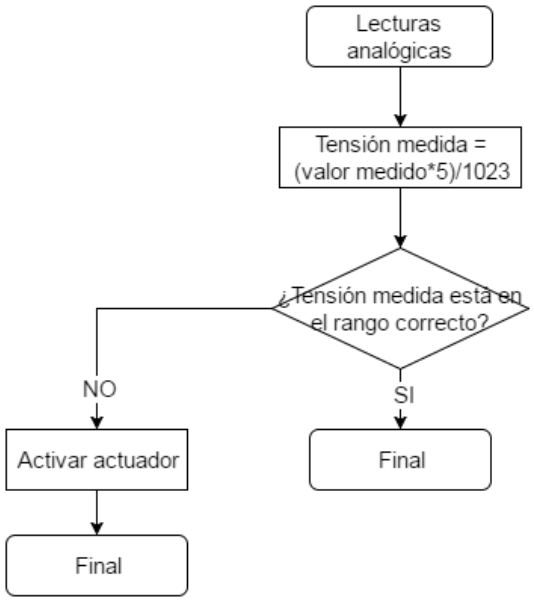


Figura 7-14. Diagrama de flujo para la lectura de un potenciómetro

Para la medida del nivel de agua usaremos dos boyas de nivel colocadas de manera que fijen el rango adecuado de llenado del acuario. De manera que cuando el nivel de agua sobrepase uno de estos dos límites se mande una señal que active la bomba de llenado o de vaciado, según si se ha activado la boya que fija el nivel mínimo o la que fija el nivel máximo.

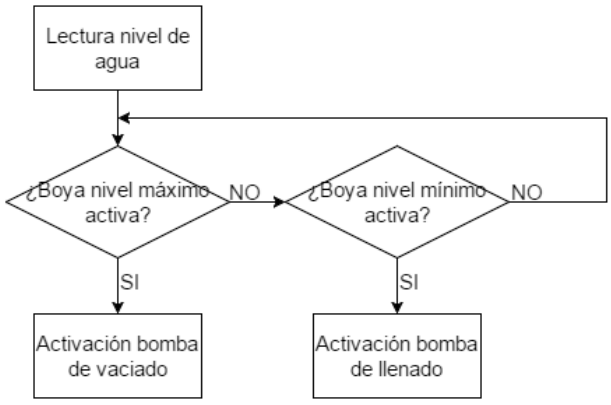


Figura 7-15. Diagrama de flujo para la lectura de las boyas de nivel

Para la lectura de la temperatura, a diferencia de la conductividad y el pH, si se usará un sensor. Este sensor, como ya se ha comentado en capítulos anteriores, es el sensor DS18B20 de Dallas.

Para poder hacer uso de este sensor se necesitan dos librerías, OneWire y DallasTemperature. A continuación se muestra un pequeño código de ejemplo donde se muestra su funcionamiento básico y se explica mediante el uso de comentarios.

```
#include <OneWire.h> //Se importan las librerías
#include <DallasTemperature.h>

#define Pin 2 //Se declara el pin donde se conectará la DATA

OneWire ourWire(Pin); //Se establece el pin declarado como bus para la comunicación OneWire

DallasTemperature sensors(&ourWire); //Se instancia la librería DallasTemperature

void setup() {
delay(1000);
Serial.begin(9600);
sensors.begin(); //Se inician los sensores
}

void loop() {
sensors.requestTemperatures(); //Prepara el sensor para la lectura

Serial.print(sensors.getTempCByIndex(0)); //Se lee e imprime la temperatura en grados Celsius
Serial.println(" grados Centigrados");
Serial.print(sensors.getTempFByIndex(0)); //Se lee e imprime la temperatura en grados Fahrenheit
Serial.println(" grados Fahrenheit");

delay(1000); //Se provoca un lapso de 1 segundo antes de la próxima lectura

}
```

Figura 7-16. Código de ejemplo del sensor de temperatura DS18B20

7.2.1.4 Comunicación con el puerto serie

Por último hay que destacar el uso de la comunicación con el puerto serie para la visualización de los parámetros deseados.

En un principio, durante el desarrollo del proyecto, para mostrar los parámetros por el monitor serie sin tener que recurrir al uso del display, se colocaron una serie de sentencias *Serial.print("Parametro = valor")* el el momento de realizar las lecturas. Al realizar las lecturas de una manera periódica, resultaba incómodo a la vista ya que aparecían los mensajes de todos los parámetros del proyecto por el monitor serie de una manera periódica y con poco tiempo de espera entre un mensaje y otro. [19]

Hasta ahora la comunicación por el puerto serie únicamente se está realizando en un sentido, del Arduino al puerto serie. Para solucionar el problema comentado, se procede a realizar una lectura del monitor serie, y dependiendo de esta, se muestra un parámetro u otro.

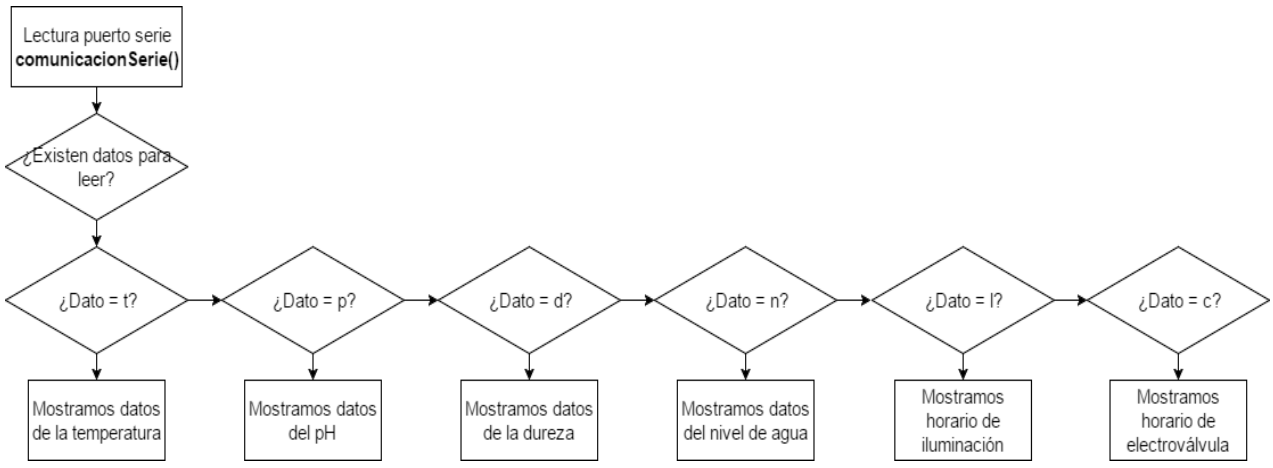


Figura 7-17. Diagrama de flujo para la lectura del monitor serie

7.2.2 Funciones usadas de las librerías

A continuación se describirán brevemente algunas de las librerías usadas a lo largo del código y sus funciones más importantes. Estas librerías ya habían sido creadas previamente por diseñadores, y únicamente se ha hecho uso de algunas de ellas cuando ha sido necesario.

7.2.2.1 Librería Wire.h

Esta librería te permite comunicarte con dispositivos que usan la comunicación I2C.

Tabla 7-1. Funciones de la librería Wire.h y su descripción

Funciones	Breve descripción

`void begin();`
`void begin(uint8_t);`
`void begin(int);`
	Inicializa la librería y enlaza el bus I2C como maestro o esclavo. Normalmente solo se realiza esta llamada una vez.

`uint8_t requestFrom(uint8_t, uint8_t);` `uint8_t requestFrom(int, int);`	Usado por el maestro para pedir bytes de un dispositivo esclavo. Tiene como parámetro la dirección y la cantidad de bytes.
`void beginTransmission(uint8_t);` `void beginTransmission(int);`	Comienza una transmisión con el esclavo pasándole su dirección como parámetro.
`uint8_t endTransmission(void);`	Finaliza la transmisión con un esclavo tras haber sido iniciada con beginTransmission.
`virtual size_t write(uint8_t);` `virtual size_t write(const uint8_t *, size_t);`	Escribe datos de un esclavo en respuesta a la llamada del maestro o
`virtual int available(void);`	Devuelve el número de bytes disponibles para recuperar con read.
`virtual int read(void);`	Lee un byte que fue transmitido de un esclavo al maestro después de la llamada de requestFrom.
`void onReceive( void (*)(int) );`	Registra una función a ser llamada cuando un esclavo recibe una transmisión del maestro.
`void onRequest( void (*)(void) );`	Registra una función a ser llamada cuando el maestro solicita datos del esclavo.

7.2.2.2 Librería LCD.h

Tabla 7-2. Funciones de la librería LCD.h y su descripción

Funciones	Breve descripción
`void setBacklight(boolean on, int minutesOn);`	Enciende o apaga la iluminación de fondo.

`void setBrightness(int brightness);`	Fija el brillo deseado para el LCD.
`void clearScreen();`	Deja la pantalla vacía.

`void drawPixel(int x, int y);`
`void drawRect(int color, int x1, int y1, int x2, int y2);`
`void fillRect(int color, int x1, int y1, int x2, int y2);`
`void drawLine(int x1, int y1, int x2, int y2);`
`void continueLine(int x, int y);`
`void drawText(int x, int y, String text);`
	Sirven todas para "dibujar" en el LCD, tanto píxeles, rectas, o texto.

`void resetCursor();`	Devuelve el cursor a la posición por defecto.
`void setCursor(int x, int y);`	Sitúa el cursor en una posición.

7.2.2.3 Librería LyquidCrystal_I2C.h

Tabla 7-3. Funciones de la librería LyquidCrystal_I2C.h y su descripción

Funciones	Breve descripción

`LiquidCrystal_I2C (const char *dev, uint8_t lcd_Addr);`
`LiquidCrystal_I2C(const char *dev, uint8_t lcd_Addr, uint8_t En, uint8_t Rw, uint8_t Rs);`
`LiquidCrystal_I2C(const char *dev, uint8_t lcd_Addr, uint8_t En, uint8_t Rw, uint8_t Rs, uint8_t d4, uint8_t d5, uint8_t d6, uint8_t d7 );`
	Inicializa las variables y define la dirección I2C del LCD.

`virtual void begin(uint8_t cols, uint8_t rows, uint8_t charsize = LCD_5x8DOTS);`	Inicializa el LCD para una dimensión fija.
`virtual void send(uint8_t value, uint8_t mode);`	Envía un valor particular al LCD para que se muestre o como comando.
`void setBacklightPin ( uint8_t value, t_backlighPol pol );`	Fija un pin en el dispositivo para controlar la luz de fondo.
`void setBacklight ( uint8_t value );`	Enciende o apaga la luz de fondo.

`int init();`	Enciende la clase LCD y el módulo IO.
`void config (const char *dev, uint8_t lcd_Addr, uint8_t En, uint8_t Rw, uint8_t Rs, uint8_t d4, uint8_t d5, uint8_t d6, uint8_t d7 );`	Para inicializar variables privadas.
`void write4bits(uint8_t value, uint8_t mode);`	Escribe 4 bits (los menos significativos) al control de líneas de datos del LCD.
`void pulseEnable(uint8_t);`	Envía un pulso de 1µs para el pin de activación para ejecutar un comando o escribir una operación.

7.2.2.4 Librería stdlib.h

Tabla 7-4. Funciones de la librería stdlib.h y su descripción

Funciones	Breve descripción
`int atoi(const char *str);`	Convierte una cadena de caracteres a un entero.

7.2.2.5 Librería Keypad.h

Tabla 7-5. Funciones de la librería Keypad.h y su descripción

Funciones	Breve descripción
`void begin(makeKeymap(userKeymap));`	Inicializa el mapa de teclas interno para ser igual que userKeymap.
`char waitForKey();`	Espera hasta que se pulse una tecla. Precaución: el código no se ejecuta hasta que no se presione alguna tecla.
`char getKey();`	Devuelve la tecla pulsada, si no se pulsa ninguna no devuelve.
`KeyState getState();`	Devuelve el estado de una tecla, es decir, si está pulsada o no.
`boolean KeyStateChanged();`	Permite conocer cuando una tecla cambia de estado.
`setHoldTime(unsigned int time);`	Fija el tiempo que el usuario debe mantener pulsada la tecla para activarse.
`setDebounceTime(unsigned int time);`	Fija el tiempo que el teclado esperará hasta una nueva pulsación.
`addEventListener(keypadEvent);`	Provoca un evento si el teclado es

	usado.

7.2.2.6 Librería OneWire.h

Tabla 7-6. Funciones de la librería OneWire.h y su descripción

Funciones	Breve descripción
`OneWire myWire(pin);`	Crea un objeto OneWire, usando un pin específico.
`myWire.search(addrArray);`	Busca el siguiente dispositivo. Devuelve la cadena de 8 bits y verdadero si se ha encontrado un dispositivo y falso si no ha encontrado.
`myWire.reset_search();`	Comienza una nueva búsqueda. El siguiente uso de search empezará con el primer dispositivo.
`myWire.reset();`	Resetea el bus OneWire, suele ser necesario antes de iniciar la comunicación con cualquier dispositivo.
`myWire.select(addrArray);`	Selecciona un dispositivo que tenga la dirección pasada como parámetro.
`myWire.skip();`	Salta la selección del dispositivo, se realiza cuando solo se tiene un dispositivo.
`myWire.write(num);` `myWire.write(num, 1);`	La primera instrucción escribe un byte. La segunda, después de escribir un byte, mantiene potencia aplicada al bus.
`myWire.read();`	Lee un bit.
`myWire.crc8(dataArray, length);`	Computa un código de redundancia cíclico en una cadena de datos.

7.2.2.7 Librería DallasTemperature.h

Tabla 7-7. Funciones de la librería DallasTemperature.h y su descripción

Funciones	Breve descripción
`DallasTemperature(myWire);` `void begin(void);`	Inicializa el bus.
`uint8_t getDeviceCount(void);`	Devuelve el número de dispositivos en el bus.
`bool validAddress(const uint8_t*);`	Devuelve verdadero si la dirección es

	válida.
`bool validFamily(const uint8_t* deviceAddress);`	Devuelve verdadero si la dirección es de la familia de sensores soportados por la librería.
`bool getAddress(uint8_t*, uint8_t);`	Encuentra una dirección en un índice dado en el bus.
`bool isConnected(const uint8_t*);`	Determina si el dispositivo asociado a esa dirección está conectado.
`bool readScratchPad(const uint8_t*, uint8_t*);` `void writeScratchPad(const uint8_t*, const uint8_t*);`	Lee o escribe dispositivos en el listado de dispositivos.
`bool readPowerSupply(const uint8_t*);`	Lee los requisitos de potencia del dispositivo.
`uint8_t getResolution();` `void setResolution(uint8_t);`	Fija o devuelve la resolución global a 9, 10, 11, o 12 bits.
`uint8_t getResolution(const uint8_t*);` `bool setResolution(const uint8_t*, uint8_t, bool skipGlobalBitResolutionCalculation = false);`	Fija o devuelve la resolución de un sensor específico a 9, 10, 11, o 12 bits.
`void requestTemperatures(void);`	Envía un comando para todos los dispositivos para comenzar la conversión de temperatura.
`bool requestTemperaturesByAddress(const uint8_t*);`	Envía un comando para un dispositivo para comenzar la conversión de temperatura según la dirección.
`bool requestTemperaturesByIndex(uint8_t);`	Envía un comando para un dispositivo para comenzar la conversión de temperatura según el índice.

`int16_t getTemp(const uint8_t*);`
`float getTempC(const uint8_t*);`
`float getTempF(const uint8_t*);`
	Devuelve la temperatura en un entero de 12 bits, en grados C o grados F.

`float getTempCByIndex(uint8_t);` `float getTempFByIndex(uint8_t);`	Devuelve la temperatura en grados C o F, según el índice del dispositivo introducido como parámetro.
`bool isParasitePowerMode(void);`	Devuelve verdadero si el bus requiere alimentación parásita.
`bool isConversionAvailable(const uint8_t*);`	Devuelve verdadero si la conversión

`bool isConversionComplete(void);`	está disponible o está completada.

8 PRUEBAS REALIZADAS

En este capítulo se comentarán las pruebas más relevantes que se han realizado durante el desarrollo del proyecto, para comprobar su correcto funcionamiento.

Se explicará por separado tanto el funcionamiento del sensor de temperatura, de las boyas de nivel, y la simulación de los sensores de pH y conductividad mediante el uso de potenciómetros.

8.1 Prueba 1. Correcto encendido del sistema

8.1.1 Descripción general

Al conectar nuestro Arduino el sistema deberá arrancar de la manera esperada, esto se aprecia si en el display aparece un mensaje de bienvenida y posteriormente la opción de fijar parámetros o mostrarlos, quedando a la espera de escoger uno de estas opciones a través del teclado.

8.1.2 Procedimiento detallado para la comprobación

I. Realizar el montaje completo del sistema mostrado en el capítulo "desarrollo hardware".
II. Conectar la alimentación al Arduino.

8.1.3 Resultado

Se comprueba el correcto encendido del display visualizando de manera correcta el mensaje de bienvenida y, posteriormente, menú principal.

8.2 Prueba 2. Activación correcta de la iluminación

8.2.1 Descripción general

La activación de la iluminación implica introducir mediante el teclado el número de horas que estará encendida. La forma de comprobarlo será introduciendo por teclado el número de horas de encendido y comprobar que el LED que simula la iluminación se encienda y se apague durante ese rango horario.

8.2.2 Procedimiento detallado para la comprobación

I. Realizar el montaje completo del sistema mostrado en el capítulo "desarrollo hardware".
II. Conectar la alimentación al Arduino.
III. Navegar en el menú hasta la opción de fijar iluminación, dentro del menú fijar parámetros.
IV. Una vez seleccionada esta opción, fijar un número de horas para el encendido de la iluminación.

8.2.3 Resultado

Fijado el intervalo de tiempo para el encendido de la iluminación, basta comprobar que, pasado ese tiempo, el estado de la iluminación cambia de encendido a apagado.

8.3 Prueba 3. Activación correcta de la electroválvula de CO_2

8.3.1 Descripción general

Al igual que para la iluminación, la válvula de CO_2 se activará un periodo de tiempo introducido por el usuario mediante el teclado. Por tanto se comprobará de la misma manera que la activación de la iluminación.

8.3.2 Procedimiento detallado para la comprobación

I. Realizar el montaje completo del sistema mostrado en el capítulo "desarrollo hardware".
II. Conectar la alimentación al Arduino.
III. Navegar en el menú hasta la opción de fijar CO_2, dentro del menú fijar parámetros.
IV. Una vez seleccionada esta opción, fijar un número de horas para el encendido de la electroválvula.
V. Ya fijado este intervalo de tiempo para el encendido de la electroválvula de CO_2, basta comprobar que pasado ese tiempo, el estado de la electroválvula cambia de encendido a apagado.

8.3.3 Resultado

Fijado el intervalo de tiempo para el encendido de la electroválvula de CO_2, basta comprobar que, pasado ese tiempo, el estado de la electroválvula cambia de encendido a apagado.

8.4 Prueba 4. Lectura correcta de los potenciómetros que simulan el sensor de pH y el sensor de conductividad

8.4.1 Descripción general

Para comprobar que el potenciómetro que se usará para simular tanto el sensor analógico de pH como el de conductividad funcionan de manera correcta se mostrará por pantalla periódicamente el valor en tensión a la salida del potenciómetro, y su equivalente en la escala de pH o en mS/cm, comprobando con la tabla teórica que los valores son los correctos.

8.4.2 Procedimiento detallado para la comprobación

I. Conectar, tal y cómo se explica en el capítulo "desarrollo hardware", los potenciómetros que simularán los sensores de conductividad y pH.
II. Conectar la alimentación al Arduino.
III. Ejecutar el script de prueba de lectura de potenciómetro.
IV. Comprobar girando el potenciómetro que los valores medidos se corresponden con los esperados.

8.4.3 Resultado

Como las resistencias que se han usado para ponerlas en serie con los dos potenciómetros tienen un valor aproximado al valor exacto que debieran tener, es normal que a la hora de comprobar las salidas de ambos potenciómetros no salgan exactamente los mismos valores extremos.

Lo primero que hay que tener en cuenta es la resolución del Arduino utilizado, en nuestro caso es un Arduino Mega, que es de 10 bits. Esto quiere decir que las tensiones de 0 a 5V que entrarán a Arduino serán convertidas a valores enteros entre 0 y 1023, o lo que es lo mismo, podemos leer tensiones de 0 a 5V con saltos de $\frac{5V}{1024} = 0.049V = 4.9mV$.

Por lo que, en nuestro código, para hallar la tensión proporcionada por el potenciómetro, tendremos que realizar la siguiente conversión, tanto para el sensor de conductividad como el de pH.

$$Tensión\ leída = \frac{Salida\ potenciómetro \cdot 5}{1023}$$

Tras hallar esta tensión, hay que calcular el valor tanto en mS/cm como en la escala natural del pH realizando las siguientes operaciones.

Para la conductividad, como ya se ha comentado cuando se habló de este sensor, la relación que guarda el valor de la conductividad con la tensión proporcionada es una tensión lineal que sigue la siguiente relación:

$$k = 6.746411 \cdot 10^{-3} \cdot V_{out}\,, \qquad medida\ en\ S/cm$$

En nuestro código se llevará a cabo la siguiente conversión y se indicará que la medida está en mS/cm:

$$k = 6.746411 \cdot V_{out}\,, \qquad medida\ en\ mS/cm$$

Para el pH, realizaremos el siguiente cálculo para pasar de la tensión al valor correspondiente en la escala del pH:

$$pH = 14 - \frac{14 * V_{pH}}{828.24mV}$$

Después de la realización de estas conversiones, los valores leídos en los límtes de giro de nuestros potenciómetros son los siguientes:

- Nivel de pH máximo:
  - Tensión: 0.00V
  - Escala de pH: 14.00
- Nivel de pH mínimo:
  - Tensión: 0.89V
  - Escala de pH: -1.04
- Nivel de conductividad máximo:
  - Tensión: 2.98V
  - Valor en mS/cm: 20.11mS/cm
- Nivel de conductividad mínimo:
  - Tensión: 0.00V
  - Valor en mS/cm: 0.00mS/cm

Puede apreciarse como se produce un fallo con la medida del pH mínimo, pero este error no afecta a nuestro proyecto ya que valores tan bajos de pH no se dan.

8.5 Prueba 5. Activación o desactivación de la resistencia calefactora y del ventilador dependiendo de la temperature medida por el sensor DS18B20

8.5.1 Descripción general

Las pruebas realizadas con este sensor han sido la colocación de éste en un recipiente con agua fría, otro con agua caliente y por último, fuera de los recipientes, midiendo la temperatura ambiente.

Según la temperatura medida se encenderá el LED que indica la activación de la resistencia calefactora (la temperatura medida es menor que la temperatura fijada), el LED que indica la activación del ventilador (la temperatura medida es superior a la temperatura fijada), o no se activará ninguno (la temperatura está dentro del rango fijado).

Para estas pruebas, en el código se ha fijado el rango de temperatura deseado de 17 a 19 grados centígrados.

8.5.2 Procedimiento detallado para la comprobación

I. Conectar, tal y cómo se explica en el capítulo "desarrollo hardware", el sensor de temperatura DS18B20 y, únicamente, conectar los dos canales del modulo relé correspondientes a la resistencia calefactora y al ventilador.
II. Conectar la alimentación al Arduino.
III. Ejecutar el script para comprobar el funcionamiento correcto del sensor de temperatura DS18B20.

IV. Comprobar en diferentes recipientes con agua a distinta temperatura cómo, dependiendo de ésta, se enciende el LED que simula la resistencia calefactora, el LED que simula el ventilador, o ninguno de ellos.

8.5.3 Resultado

Dentro del recipiente de agua fría, por el puerto serie aparece que la temperatura es de 15.94ºC, valor inferior al nivel mínimo fijado de 17ºC. Por tanto, se enciende el LED que simula el encendido de la resistencia calefactora.

Dentro del vaso de agua caliente, por el puerto serie aparece que la temperatura es de 35.56ºC, valor superior al nivel máximo fijado de 19ºC. Por tanto, se enciende el LED que simula el encendido del ventilador.

Con el sensor midiendo la temperatura ambiente, por el puerto serie aparece que la temperatura es de 18.69ºC, valor que se encuentra dentro de los límites fijados. Por tanto, no se enciende ningún LED.

8.6 Prueba 6. Activación o desactivación de la bomba de llenado y de la bomba de vaciado dependiendo de las señales leídas por las boyas de nivel

8.6.1 Descripción general

Con las boyas de nivel se han realizado tres pruebas diferentes correspondientes a los tres estados que pueden producirse: nivel de agua por encima del nivel máximo (activándose el LED de activación de la bomba de vaciado), nivel de agua entre los niveles fijados (no activándose ningún LED), o nivel de agua por debajo del nivel mínimo (activándose el LED correspondiente a la activación de la bomba de llenado).

8.6.2 Procedimiento detallado para la comprobación

I. Conectar, tal y cómo se explica en el capítulo “desarrollo hardware”, las boyas de nivel máximo y mínimo y, únicamente, conectar los dos canales del modulo relé correspondientes a la bomba de llenado y bomba de vaciado.
II. Conectar la alimentación al Arduino.
III. Ejecutar el script que comprueba el correcto funcionamiento de las boyas de nivel.

8.6.3 Resultado

Se comprueba que, dependiendo de la posición de las dos boyas de nivel, se producen los tres estados posibles para el nivel de llenado (nivel correcto, nivel superior al máximo y nivel inferior al mínimo), activándose, si es necesario, el LED de la bomba de llenado o el de la bomba de vaciado.

8.7 Prueba 7. Visualización correcta del menú por el display LCD

8.7.1 Descripción general

Con esta prueba se comprobará que la navegación por el menú mostrado por el display es correcta y no se produce ningún bucle ni situaciones indeseadas. Para ello se pondrá en funcionamiento el sistema y se comprobará mediante la prueba de todas las combinaciones posibles que no existe ningún problema.

8.7.2 Procedimiento detallado para la comprobación

I. Realizar el montaje completo del sistema mostrado en el capítulo “desarrollo hardware”.
II. Conectar la alimentación al Arduino.
III. Ejecutar el programa completo del sistema.

8.7.3 Resultado

Navegando por el menú principal y por los submenús de fijar y mostrar parámetros que el funcionamiento es el esperado y no se produce ningún problema.

8.8 Prueba 8. Lectura correcta del teclado matricial 4x4 por parte de Arduino

8.8.1 Descripción general

Para comprobar que se lee por teclado de manera correcta se ejecutará un pequeño programa mediante el cual se muestre por pantalla las teclas pulsadas, comprobándose que son las mismas las mostradas por el display que las pulsadas el el teclado.

8.8.2 Procedimiento detallado para la comprobación

I. Conectar al Arduino Mega únicamente el teclado matricial 4x4.
II. Conectar la alimentación al Arduino.
III. Ejecutar el script de prueba de funcionamiento del teclado.

8.8.3 Resultado

Se comprueba, presionando aleatoriamente las teclas, que por el monitor serie aparecen las teclas pulsadas.

8.9 Cuadro resumen de las pruebas realizadas

Tabla 8-1. Tabla resumen de las pruebas realizadas

Prueba número	Breve descripción	Requerimiento verificado	Resultado	Observaciones
1	Encendido correcto del sistema	Requerimiento 1	OK	Ninguna
2	Activación correcta de la iluminación	Requerimiento 1 Requerimiento 7	OK	Ninguna
3	Activación correcta del CO_2	Requerimiento 1 Requerimiento 7	OK	Ninguna
4	Lectura de los potenciómetros que simulan los sensores de pH y conductividad	Requerimiento 3 Requerimiento 4 Requerimiento 6	OK	Se produce un fallo con la medida del pH mínimo, error que no afecta a nuestro proyecto ya que valores tan bajos de pH no se dan
5	Lectura del sensor DS18B20 y activación, si es necesario, de la resistencia calefactora o del ventilador	Requerimiento 1 Requerimiento 2 Requerimiento 8	OK	Ninguna

6	Lectura de las boyas de nivel máximo y mínimo y activación, si es necesario, de la bomba de llenado o de la bomba de vaciado	Requerimiento 5	OK	Ninguna
7	Visualización correcta del menú y correcta navegación por el mismo a través del display LCD 16x2	Requerimiento 6	OK	Ninguna
8	Lectura correcta del teclado matricial 4x4.	Requerimiento 7 Requerimiento 8	OK	Ninguna

9 CONCLUSIONES Y POSIBLES MEJORAS

9.1 Conclusiones

Se ha conseguido diseñar, tal y como se propuso en un principio, un controlador de parámetros de calidad de acuarios basado en la plataforma Arduino. Este controlador es capaz de monitorizar y modificar parámetros tales como la temperatura, el pH, la dureza o el nivel de agua.

Las conclusiones sacadas al realizar este proyecto son varias, destacando las numerosas posibilidades de uso que ofrece la plataforma Arduino, además de la gran cantidad de código existente en internet, que facilita en gran medida proyectos de este tipo.

En la actualidad, no existen demasiados dispositivos controladores de acuarios comerciales en el mercado, y los que existen tienen un precio bastante elevado. Hoy día sigue bastante extendido el control de acuarios tradicional, entendiéndose éste como el control de acuarios sin incluir el uso de elementos electrónicos. Por lo que, con el proyecto realizado se consigue un controlador similar a los comerciales a un precio mucho más económico.

A nivel personal, la experiencia desarrollando el trabajo ha sido bastante satisfactoria ya que, en un principio, no tenía experiencia ninguna con la plataforma Arduino y, abase de trabajo, constancia y organización he conseguido desarrollar el proyecto en su totalidad.

9.2 Posibles mejoras

Este proyecto sirve como base para futuros proyectos más completos ya que, tomando como eje el sistema creado, es bastante sencillo incluir nuevos sensores y actuadores.

A continuación numeraremos una serie de mejoras más concretas que se han ido descubriendo a lo largo del trabajo y que, por cuestiones de tiempo o presupuesto, no se han llevado a cabo.

- Ofrecer la posibilidad de fijar parámetros como el pH o la dureza y, aunque no puedan modificarse mediante el uso de actuadores, emitir una señal de alarma cuando estos niveles no son los deseados. Esta señal de alarma podría ser luminosa (haciendo uso de un LED) y/o sonora (haciendo uso de un "buzzer" o zumbador)
- Además de la posibilidad de interactuar a través del teclado matricial, el display o el monitor serie, se podría añadir la opción de poder interactuar a través de un móvil o un ordenador que no estuviera conectado al dispositivo, usando la comunicación WiFi o bluetooth. Así seria posible que, si en un momento uno de los parámetros no es el deseado y hubiera que actuar de manera manual sobre él, se recibiera una alarma que indicara del suceso en el teléfono móvil.
- Para unificar el diseño, se propone el diseño de un PCB situado sobre el Arduino que reúna todos los componentes y conexiones realizadas, para conseguir así simplificar en gran medida el diseño.

REFERENCIAS

[1] Aqua Medic USA

http://www.aquamedicusa.com/products/at-control-system/

[2] ProfiLux 4

http://www.profilux.es/profilux-4/#

[3] Ferduino Aquarium Controller

http://ferduino.com/

[4] Jarduino

http://thingml.org/pmwiki.php?n=Main.JArduino

[5] Arduino

https://www.arduino.cc/en/Main/Products

[6] Sensor de temperatura DS18B20

https://www.maximintegrated.com/en/products/analog/sensors-and-sensor-interface/DS18B20.html

[7] Funcionamiento boya de nivel

https://tallerarduino.com/2012/10/26/sensor-de-nivel-de-liquido-y-arduino-o-pinguino-pic/

[8] Sensor de ultrasonido HC-SR04

https://www.sparkfun.com/products/13959

[9] Sensor de pH SE0161

https://www.dfrobot.com/index.php?route=product/product&product_id=1025

[10] Sensor de EC DFR0300

https://www.dfrobot.com/index.php?route=product/product&product_id=1123

[11] Teclado matricial 4x4

http://huborarduino.com/programacion/curso-programacion/34-leccion15.html

[12] Display LCD 16x2

https://www.arduino.cc/en/Tutorial/HelloWorld

[13] Acuario Plantado, actuadores necesarios

http://www.acuarioplantado.com/

[14] Relés

http://www.prometec.net/reles/

[15] Fritzing

http://fritzing.org/home/

[16] Eagle

http://www.autodesk.com/products/eagle/overview

[17] Software Arduino

https://www.arduino.cc/en/main/software

[18] Millis

https://www.arduino.cc/en/Reference/Millis

[19] Puerto serie

https://www.arduino.cc/en/reference/serial

ANEXOS

A. Comunicación I2C

El bus I2C es un estándar que facilita la comunicación entre microcontroladores, memorias y otros dispositivos, sólo requiere de dos líneas de señal. Fue diseñado por Philips en 1982 para permitir el intercambio de información entre sus artículos, aunque posteriormente fue extendiéndose a otros fabricantes hasta convertirse en un estándar de mercado. La metodología de comunicación de datos del bus I2C es en serie y sincrónica. Una de las señales del bus marca el tiempo (SCL) y la otra se utiliza para intercambiar datos (SDA).

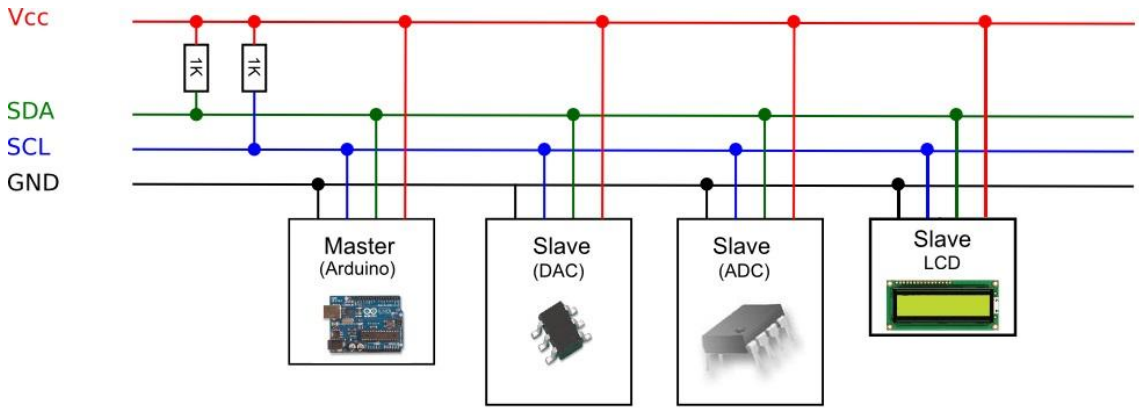


Figura A-1. Esquema funcionamiento del bus I2C

A continuación se explicarán algunas características del bus I2C:

- Cada dispositivo dispone de una **dirección única**, que se emplea para acceder a ellos de forma individual. Esta dirección puede ser fijada por hardware (se pueden modificar los últimos 3 bits mediante jumpers o interruptores) o totalmente por software.
- El bus I2C tiene una arquitectura de tipo **maestro-esclavo**. El dispositivo maestro inicia la comunicación con los esclavos, y puede mandar o recibir datos de los esclavos. Los esclavos no pueden iniciar la comunicación (el maestro tiene que preguntarles), ni hablar entre sí directamente. Es posible disponer de más de un maestro, pero solo uno puede ser el maestro cada vez. El cambio de maestro supone una alta complejidad, por lo que no es algo frecuente.
- **El bus I2C es síncrono**. El maestro proporciona una señal de reloj, que mantiene sincronizados a todos los dispositivos del bus. De esta forma, se elimina la necesidad de que cada dispositivo tenga su propio reloj, de tener que acordar una velocidad de transmisión y mecanismos para mantener la transmisión sincronizada.

Para poder realizar la comunicación con un solo cable de datos, el bus I2C emplea una trama amplia. La comunicación costa de:

- 7 bits a la dirección del dispositivo esclavo con el que queremos comunicar. Con estos bits podemos acceder a 112 dispositivos en un mismo bus, ya que 16 de las 128 direcciones posibles son direcciones especiales reservadas.
- Un bit restante indica si queremos enviar o recibir información.
- Un bit de validación.
- Uno o más bytes son los datos enviados o recibidos del esclavo.
- Un bit de validación.

En el siguiente ejemplo se puede ver el funcionamiento del bus I2C:

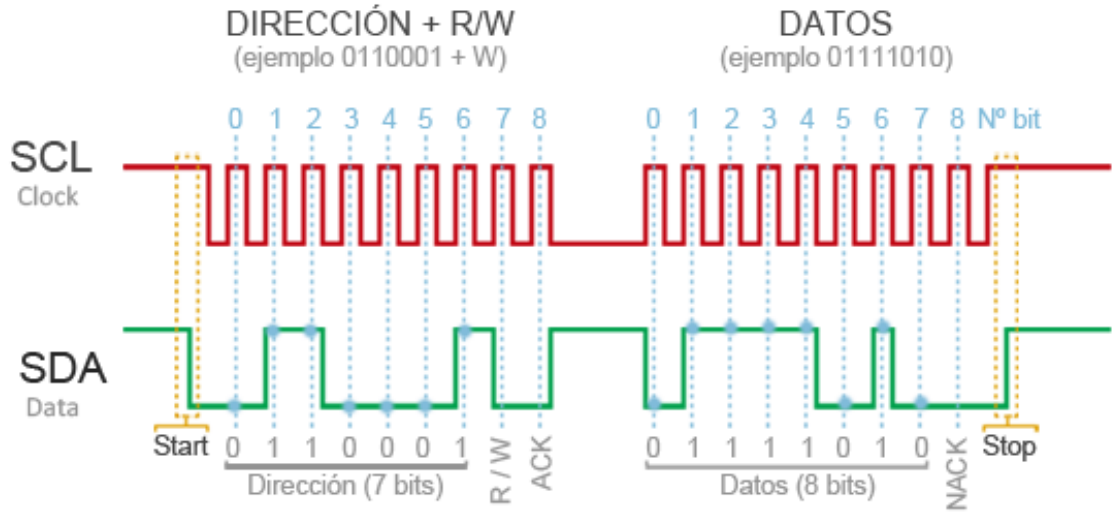


Figura A-2. Ejemplo de funcionamiento del bus I2C

La cantidad de datos enviados que puede apreciarse en el ejemplo (18 bits por cada 8 bits de datos) hace que la velocidad del bus I2C sea reducida por regla general. La velocidad estándar de transmisión es de 100MHz, con un modo de alta velocidad de 400MHz. También existen otros modos, como un envío de dirección de 8, 10 y 12 bits, o velocidades de transmisión de 1Mbit/s, 3.4Mbit/s y 5Mbit/s, aunque estos no suelen ser empleados en Arduino.

Como ventajas e inconvenientes a este tipo de comunicación se pueden comentar las siguientes:

Ventajas

- Requiere pocas conexiones.
- Dispone de mecanismos para comprobar si la señal ha llegado.

Inconvenientes

- Velocidad media-baja.
- No es full-duplex.
- No puede comprobarse si el contenido del mensaje es el correcto.

En el caso concreto de nuestro Arduino Mega, los pines que se usarán serán el 20 para SDA, y el 21 para SCL. Cabe señalar que, como Arduino es una plataforma de código abierto, existe un sketch con el cual podemos determinar la dirección de cada dispositivo, ya que en algunas ocasiones el fabricante no la proporciona, o no se tiene acceso a esa información.

Glosario

RAE: Real Academia Española

GH: dureza permanente

KH: dureza temporal

BNC: Bayonet Neill-Concelman, tipo de conector

LCD: pantalla de cristal líquido (Liquid Crystal Display)

IDE: entorno de desarrollo integrado (Integrated Development Environment)

IoT: internet de las cosas (Internet of Things)

MIT: Instituto Tecnológico de Massachusetts (Massachusetss Institute of Technology)

PWM: modulación por ancho de pulsos (pulse-width modulation)

DSP: procesador digital de señales (Digital Signal Processor)

TFT-LCD: pantalla de cristal líquido de transistores de película fina (Thin Film Transistor-Liquid Crystal Display)

USB: bus serie universal (Universal Serial Bus)

PCB: placa de circuito impreso (Printed Circuit Board)

I2C: circuito interintegrado (Inter-Integrated Circuit)

SDA: línea de datos (Serial Data)

SCL: línea de reloj (Serial Clock)

LED: diodo emisor de luz (light-emitting diode)