



GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL



VNIVERSITAT
DE VALÈNCIA

TRABAJO DE FIN DE GRADO

KIT DE MONITORIZACIÓN Y CONTROL APLICADO AL MANTENIMIENTO DE VIVEROS.

AUTOR:
RUBÉN GARRIDO ALONSO

TUTORÍA:
SILVIA CASANS BERGA
EDITH NAVARRO ANTÓN

SEPTIEMBRE, 2020





GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL

TRABAJO DE FIN DE GRADO

KIT DE MONITORIZACIÓN Y CONTROL APLICADO AL MANTENIMIENTO DE VIVEROS.

AUTOR:
RUBÉN GARRIDO ALONSO

TUTORÍA:
SILVIA CASANS BERGA
EDITH NAVARRO ANTÓN

TRIBUNAL:

PRESIDENTE/PRESIDENTA: VOCAL 1:

VOCAL 2: FECHA DE DEFENSA:

CALIFICACIÓN:





Índice

Resumen del proyecto.....	5
Abstract.	6
Capítulo 1. Introducción.....	7
1.1 Motivación del trabajo.....	7
1.2 Objetivo.....	7
Capítulo 2. Sistema de monitorización y medida.	8
2.1 Hardware.....	8
2.1.1 Sensores.....	8
2.1.2 Accionadores.	10
2.1.3 Circuitos acondicionadores.....	12
2.1.4 Arduino.	19
2.1.5 Módulos Xbee.....	20
2.2 Software	22
2.2.1. Herramientas de programación.....	22
2.2.2 Programación en Arduino.....	23
2.2.3 Interfaz de usuario.....	30
2.2.4 Programación en LabView.	33
2.2.5 Comunicación con módulos XBee.....	37
Capítulo 3. Resultados experimentales.	40
3.1 Implementación sistema de adquisición.	40
3.1.1 Placa de alimentación.....	41
3.1.2 Placa de adquisición.	42
3.2. Puesta en marcha (simulación “física”)	44
3.2.1 Desarrollo de prototipos.....	44
3.2.2 Simulación de la respuesta del sensor de pH (hard/software).....	47
3.2.3 Simulación del módulo.	49
Capítulo 4. Conclusión y futuras mejoras.....	56
Bibliografía.....	57
Anexos.....	58



ÍNDICE DE FIGURAS

Figura 1. Sensor DHT 22.....	9
Figura 2. Pinout sensores DHT22 y DHT21	9
Figura 3. Electrodo de pH.....	9
Figura 4. Esquema de un electrodo de pH.....	10
Figura 5. Ejemplo de relé accionado a 12 VDC.....	11
Figura 6. Cilindro simple efecto neumático.....	11
Figura 7. Válvula de tres vías y dos posiciones.....	11
Figura 8. Rango medición electrodo de pH	12
Figura 9. Etapa de acople de impedancias en el circuito acondicionador de electrodo de pH.	13
Figura 10. Etapa de amplificación y ajuste de offset en circuito acondicionador electrodo de pH.	14
Figura 11. Filtro Sallen-Key	14
Figura 12. Esquema conversión 24 V - 12 V.....	15
Figura 13. Esquema conversion 12 V - 5V.....	15
Figura 14. Esquema conversión 5 V - 3.3 V.....	16
Figura 15. Esquema obtención -5 V.....	16
Figura 16. Referencia de voltaje AD580H.	16
Figura 17. Adaptación circuito sensado humedad y temperatura.	17
Figura 18. Circuito equivalente de cada salida del driver TBD62083A.	17
Figura 19. Conexionado botonera física.....	18
Figura 20. Placa controladora Arduino Fio.	19
Figura 21. Módulo XBee.....	20
Figura 22. Conexionado de XBee.....	21
Figura 23. Conexionado transmisión XBee.	21
Figura 24. Esquema software del proyecto.	22
Figura 25. Esquema proyecto.....	23
Figura 26. Comportamiento del electrodo de pH a diferentes temperaturas....	25
Figura 27. Interfaz de usuario.....	30
Figura 28. Bloque de interruptores.	31
Figura 29. Indicadores de la interfaz de usuario.....	31
Figura 30. Pulsador introducción de parámetros.....	32
Figura 31. SubVI Introducción de parámetros.	32
Figura 32. Inicio de comunicación por puerto serie.	33
Figura 33. Ejemplo lectura en LabView.	34
Figura 34. Encendido de pilotos en LabView.	34
Figura 35. Escritura de datos en LabView.....	35
Figura 36. Variables globales en LabView.	35
Figura 37. SubVI introducción de parámetros.	36
Figura 38. Concatenación de datos para transmisión puerto serie.	36
Figura 39. Módulos XBee utilizados.	37
Figura 40. Configuración XBee router.	37
Figura 41. Configuración XBee coordinador.....	38
Figura 42. Prueba envío y recepción de tramas en XBee 1.....	38
Figura 43. Prueba envío y recepción de tramas XBee 2.	38
Figura 44. PCB cara "top" completa.	40
Figura 45. PCB de alimentación (top/bottom).....	41
Figura 46. Convertidor AC DC.....	41



Figura 47. Conector PTR.....	42
Figura 48. Cara “top” PCB de adquisición.....	42
Figura 49. Prototipo en protoboard.....	44
Figura 50. Placa PCB prototipo	45
Figura 51. Transferencia tóner a placa de cobre.....	46
Figura 52. Pistas de placa PCB prototipo.....	46
Figura 53. Placa PCB prototipo ensamblada.....	47
Figura 54. Circuito simulación acondicionamiento sensor de pH.....	47
Figura 55. Placa PCB prototipo	49
Figura 56. Comprobación software botones físicos.....	50
Figura 57. Activación botones físicos	50
Figura 58. Ventana Probe LabView.....	51
Figura 59. Leds placa PCB prototipo.....	51
Figura 60. Comprobación de envío parámetros objetivo.....	52
Figura 61. Conexionado físico para la medición de pH.....	53
Figura 62. Simulación medición de pH.....	53
Figura 63. Conexionado sensor DHT22	54
Figura 64. Cadena de datos con información de temperatura y humedad.....	54
Figura 65. Lectura de temperatura y humedad.....	55
Figura 66. Código del programa principal.....	63



ÍNDICE DE TABLAS

Tabla 1. Características DHT22.	8
Tabla 2. Simulación software pH.	48



Resumen del proyecto

Cada día que pasa, el desarrollo de procesos automatizados se abre camino a gran ritmo en entornos industriales, es por ello que cada día se consigue reducir tiempos de producción y con ello, también el coste de producción de muchos servicios y productos.

Siguiendo esa tendencia, en este proyecto se ha desarrollado un prototipo que permite automatizar la producción de planta tanto ornamental como frutal. Dicho módulo puede implementarse en diferentes entornos de trabajo, tanto en entornos industriales con un elevado volumen de producción como en entornos no industriales.

Para conseguir esto, se deben monitorizar los parámetros ambientales más importantes garantizando el mantenimiento de los diferentes tipos de planta y actuar en el entorno para que las condiciones ambientales siempre sean las ideales para la variedad de planta que se este trabajando.

El proyecto que se presenta implementa un sistema de monitorización y control de todos los parámetros, para ello, se utiliza un microcontrolador que se encarga de gestionar los diferentes sensores y actuadores, así como la comunicación con el ordenador de control.

Los diferentes capítulos del documento describen como se ha llevado a cabo tanto el diseño como la implementación de cada una de las etapas distinguiendo entre hardware y software.



Abstract.

Day to day, the development of automated processes is cutting a path in industrial environments, this favor to decrease production times, in consequence, the costs of many services and products is also reduced.

Following this trend, this project tries to develop a module that allows automating the production of both ornamental and fruit plants, both in industrial environments with a high production volume and in non-industrial environments with not very high volumes. So that it can be implemented in different work environments.

To achieve this, it is necessary to monitor the most important environmental parameters for the maintenance of the different types of plant and act in the environment so that the environmental conditions are always ideal for the variety of plant that is being worked on.

During the project, a monitoring and control system of all the parameters is implemented, for this, a microcontroller will be used that will be in charge of managing the different sensors and actuators, as well as communication with the control computer.

Finally, during the progress of this project, it will be described how both the design and the implementation of each of the sections within the software and the hardware have been carried out.



Capítulo 1. Introducción

1.1 Motivación del trabajo

La motivación del proyecto surge como necesidad de implementar mejoras de producción de plantas ornamentales y no ornamentales en todas aquellas instalaciones que quieran un módulo de monitorización y control de fácil instalación con pequeño-medio presupuesto.

Con el fin de plantear un desarrollo software-hardware consultando con trabajadores de viveros y aficionados al cultivo de diferentes tipos de planta, se concluyó que con relativa sencillez se puede realizar un control automatizado, cubriendo la necesidad real que contribuye a mejorar de la calidad de planta que se quiera cultivar. De este modo, disponiendo de un sistema automatizado efectivo, se reduciría la carga de trabajo del operario de vivero.

Tras recoger información sobre el funcionamiento y la dinámica de la producción, nace la idea de diseñar un modulo de fácil acceso para cualquier empresa-particular que permita una sencilla instalación y que proporcione el máximo control sobre las variables que afectan a la producción. También se ha buscado tener acceso a la información necesaria para saber en qué condiciones se está desarrollando la actividad.

1.2 Objetivo.

Como se ha indicado anteriormente, el objetivo es diseñar e implementar un módulo de control que permita la automatización del proceso de cultivo en habitaciones de vivero. De forma que, se consiga desarrollar un sistema de bajo coste para automatizar una habitación de vivero con unas dimensiones máximas de 30 m². Además el módulo a desarrollar debe ser controlado desde un ordenador que permitirá la monitorización de los parámetro de interés (temperatura, pH y humedad relativa).

Como el objetivo es plantear una aplicación lo más polivalente posible, el módulo debe poder adaptarse a toda instalación independientemente de las diferentes variedades de producción. Para ello también se ha propuesto como objetivo el poder controlar diferentes habitaciones desde un mismo punto de control, elegir qué funciones se desean automatizar y también en qué condiciones deberán de funcionar.

En resumen el objeto de este proyecto es diseñar un módulo accesible, fácil de utilizar, polivalente y que por parte del desarrollador sea sencillo adaptarlo a las necesidades de la persona que quiera disponer del módulo en su instalación.



Capítulo 2. Sistema de monitorización y medida.

2.1 Hardware

El sistema de control implementado trabaja siempre bajo los parámetros establecidos por el usuario. Para ello, previa puesta en funcionamiento, se deben establecer estos parámetros.

Dentro de la versatilidad que puede llegar a ofrecer la implantación del módulo siempre se deben cumplir las condiciones necesarias para poder alimentar los diferentes sensores y actuadores.

El módulo se ha diseñado para trabajar en habitaciones de vivero, por lo cual, la base del diseño reside en un rango relativamente amplio tanto de temperatura, pH y humedad. Para las condiciones de diseño se ha tomado un rango de temperaturas de 30-50 °C, un rango de pH de 5.5 a 8.5 y el rango de humedad puede fluctuar entre 20-80 % de humedad relativa.

El dispositivo encargado de procesar la información y comunicarse con el centro de control va a ser Arduino Fio.

2.1.1 Sensores

Para poder capturar y trabajar sobre todas las variables necesarias y garantizar un correcto funcionamiento se ha tenido que adquirir información sobre la temperatura de la habitación, la humedad y también el PH del agua utilizada para el riego. Dicha información se procesa a tiempo real con la finalidad de poder actuar sobre las variables cuando se requiera.

El primero de los sensores utilizados es el sensor DHT22, este sensor es usado para medir la temperatura y la humedad en la habitación. Se ha escogido éste ya que es un sensor que proporciona sensibilidad en su medición adecuada para los parámetros que se desean alcanzar. La medición de temperatura está en el rango de -40 – 80 °C mientras que la humedad se puede medir en el rango de 0-100 %RH, sus características técnicas se indican en la Tabla 1:

Model	DHT22	
Power supply	3.3-6V DC	
Output signal	digital signal via single-bus	
Sensing element	Polymer capacitor	
Operating range	humidity 0-100%RH; temperature -40-80Celsius	
Accuracy	humidity +2%RH(Max +5%RH); temperature <+0.5Celsius	
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius	
Repeatability	humidity +/-1%RH; temperature +/-0.2Celsius	
Humidity hysteresis	+0.3%RH	
Long-term Stability	+0.5%RH/year	
Sensing period	Average: 2s	
Interchangeability	fully interchangeable	
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm	

Tabla 1. Características DHT22.



Los siguientes parámetros a tener en cuenta son la resolución y la repetitividad, en nuestro caso, una sensibilidad de 0.5 % RH para la humedad y de 0.5 °C es mas que suficiente y como se observa en la Tabla 1, el valor de la resolución es de 0.1 % RH y 0.1 °C. Existe una desviación en la medida del 2 %RH a 5 %RH y un error máximo de 0.5 °C en la medida de la temperatura, para nuestra aplicación, según estos datos, el sensor es idóneo para el módulo. El sensor DHT 22 se muestra en la Figura 1.



Figura 1. Sensor DHT 22.

El método de conexionado con el microcontrolador ha sido otro punto a favor a la hora de escoger el sensor, Figura 2, esto se debe a que la conexión del sensor con el microcontrolador se establece mediante un protocolo digital, lo cual conlleva una mejora en la simplicidad del diseño.

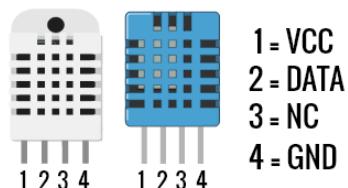


Figura 2. Pinout sensores DHT22 y DHT21

Para realizar la medida de pH se utiliza un electrodo de pH el cual irá acompañado de su circuito de acondicionamiento y se conectará a la placa de adquisición de datos mediante un conector BNC. A continuación se detallará el comportamiento de un electrodo de pH para comprender como se ha podido interpretar información a través de él. En la Figura 3 se muestra un ejemplo de electrodo de pH.



Figura 3. Electrodo de pH.



Un electrodo de pH es un dispositivo capaz de medir el PH de forma electroquímica, es decir, dependiendo del medio en el que se encuentre la parte sensible de este electrodo proporcionará un determinado rango de tensión a través del cable coaxial. Su construcción es relativamente simple y su eficiencia en este proyecto es bastante alta, es decir, con un diseño simple y económico se puede obtener una medida de pH en líquidos con un porcentaje de error bajo.

La parte más importante del electrodo es la membrana sensible, Figura 4 p.7, ésta membrana es una de las partes que debe de estar en contacto con el medio a la hora de tomar la medida.

Una vez se entra en contacto con el medio en el cual se debe de efectuar la medición el electrodo genera una diferencia de potencial que un circuito externo al electrodo se encarga de transformar a un rango de tensión interpretable por el microcontrolador.

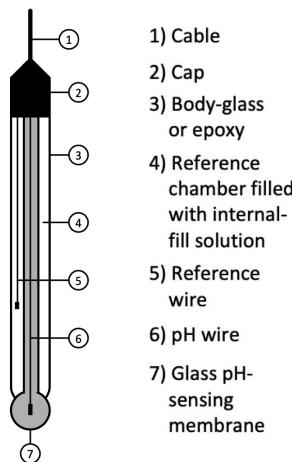


Figura 4. Esquema de un electrodo de pH.

2.1.2 Accionadores.

Para poder automatizar la gestión de la habitación de vivero donde se ha implantado el módulo se ha hecho uso de diversos accionadores. En todo momento el propósito ha sido conseguir la mejor adaptación del módulo con el entorno donde se instala, por ello se han diseñado diferentes conexiones para estos accionadores.

Para monitorizar la temperatura se realizan dos tareas de control, en primer lugar, si la temperatura está por debajo del rango idóneo, se eleva ésta utilizando un calefactor, cuya activación y desactivación se realiza a través de un relé SPDT, Figura 5, de forma que al enviar un pulso de 12 VDC o bien 24 VDC se acciona el relé que permite el paso de corriente hacia el calefactor.



Figura 5. Ejemplo de relé accionado a 12 VDC.

En el caso que la temperatura medida se encuentre fuera del rango idóneo, se procede a la apertura de dos o más ventanas, dependiendo de las disponibilidades de la habitación. De esa forma se logra una corriente de aire que provoca una disminución de la temperatura.

Para automatizar la apertura de las ventanas, se utilizan cilindros de simple efecto accionados por neumática, Figura 6. La elección de accionamiento se debe a que prácticamente en todos los entornos industriales se dispone conexiones neumáticas a lo largo de las instalaciones. En el caso de que no se disponga, sería necesaria la instalación de un compresor de aire, no necesariamente de categoría industrial, para el accionamiento de los cilindros de simple efecto.



Figura 6. Cilindro simple efecto neumático.

El accionamiento de estos cilindros se lleva a cabo mediante la apertura de una válvula de tres vías y dos posiciones accionada por 12VDC o 24VDC dependiendo de que solenoide disponga cada válvula. Es por esto por lo que como se ha mencionado anteriormente, el módulo diseñado cuenta con diferentes niveles de salida para accionar los diferentes actuadores.

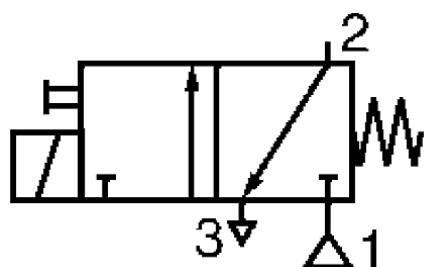


Figura 7. Válvula de tres vías y dos posiciones.



Como se puede observar en la Figura 7, al accionar el solenoide, el paso de aire sería directo de 1 a 2 y de esta forma abriríamos la ventana, mientras que una vez se deja de aplicar corriente al solenoide, el aire que queda encerrado en el cilindro en la posición de ventana abierta se descarga de 2 a 3. Para evitar los movimiento bruscos se pueden utilizar válvulas de estrangulamiento, que limitan el caudal de aire que circula y evitan de esa forma los movimientos bruscos de apertura o bien de cerrado.

La regulación de pH se realiza mediante dos electroválvulas que corregirán el pH del agua de riego, de forma que, una vez se monitorice el nivel de pH y este se encuentre fuera de rango se accionará una electroválvula que dejará pasar el fluido necesario para la regulación de pH.

2.1.3 Circuitos acondicionadores.

Si bien se ha mencionado que el sensado de temperatura y humedad no requiere de ningún circuito auxiliar para acondicionar la lectura, no ocurre lo mismo para el sensor de pH. No es suficiente con leer el voltaje que proporciona el electrodo a través de conector BNC. Son diversos los motivos por los que existe una necesidad de diseñar un circuito que pueda acondicionar la señal.

En primer lugar, se ha de conocer el rango de valores que proporciona el electrodo de pH para saber como interpretar la diferencia de voltaje proporcionada, a su vez, hay que tener en cuenta la impedancia de salida del sensor ya que una impedancia de salida muy alta falsearía la lectura.

Debido a la fina membrana del bulbo, la impedancia de salida del electrodo de pH puede oscilar entre $10\text{ M}\Omega$ y $1000\text{ M}\Omega$ por lo que se hace imprescindible utilizar un circuito que adapte esta impedancia y la convierta a una mucho más pequeña para poder realizar una lectura fiable.

El rango de voltaje que puede ofrecer un electrodo de pH va de 414 mV a -414 mV como se puede observar en la Figura 8. En este caso, el microcontrolador utilizado no puede leer voltajes negativos, por lo que, en el circuito de acondicionamiento a diseñar se deberá modificar este rango para que los valores se comprendan entre 0 V y 3.3 V , correspondiendo -414 mV a 0 V y 414 mV a 3.3 V .

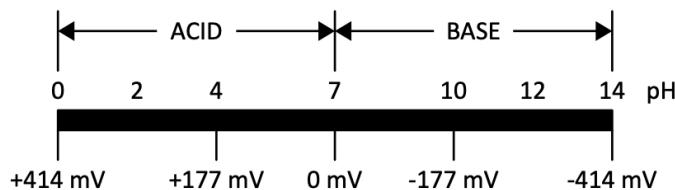


Figura 8. Rango medición electrodo de pH.



Una vez conocidas y estudiadas las características técnicas del sensor, como se ha ido mostrando, se debe realizar el diseño electrónico del circuito de acondicionamiento, el cual realice las siguientes funciones:

- Adaptar impedancia de salida.
- Añadir offset para realizar lecturas de voltaje en un rango de valores positivos.
- Amplificar la medida para realizar una lectura de 0 V a 3.3 V.

Para adaptar la impedancia que va a recibir el microcontrolador, se ha utilizado un seguidor de voltaje realizado con un amplificador operacional. Los amplificadores operacionales proporcionan una impedancia de entrada alta y una impedancia de salida baja, por lo que se ha decidido utilizar el amplificador operacional OPA129, ya que cuenta con una impedancia de entrada de $10^{13} \Omega$ en modo diferencial y de $10^{15} \Omega$ en modo común, lo suficientemente alta como para ser óptimo para la aplicación.

Dado que el rango de voltaje proporcionado por el sensor contiene valores negativos, la tensión de alimentación de este circuito integrado será de 5 V para la alimentación positiva Vcc y -5 V para la alimentación negativa -Vcc. El diseño resultante de esta etapa es el que se muestra en la Figura 9.

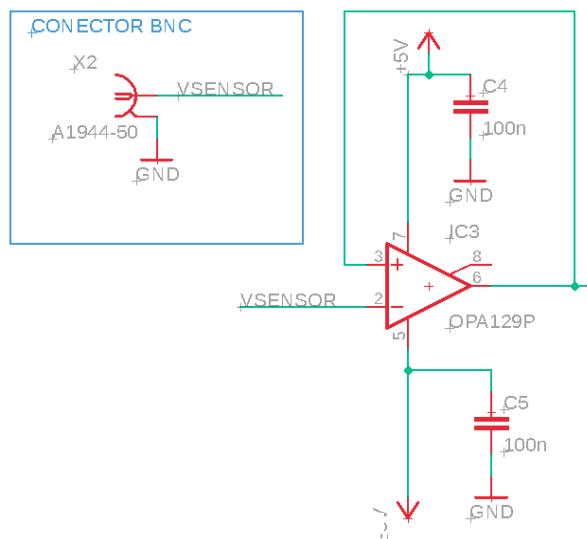


Figura 9. Etapa de acoplamiento de impedancias en el circuito acondicionador de electrodo de pH.

Como se menciona con anterioridad, se ha de elevar el nivel de señal para que únicamente se proporcionen valores positivos al microcontrolador, así como amplificar la medida a un rango interpretable por el microcontrolador, en este caso 0-3.3 V. Para ello se ha utilizado otro amplificador operacional, donde además de amplificar, se proporcionará un offset que se deberá de regular en su calibración.

Para esta etapa, se ha de tener en cuenta que debido a la tolerancia que puedan tener los componentes utilizados así como las derivas térmicas de estos, es conveniente utilizar un potenciómetro trimmer para realizar una calibración que sirva de ajuste para adaptar tanto el offset como la ganancia exacta que necesita el circuito para garantizar una correcta lectura.

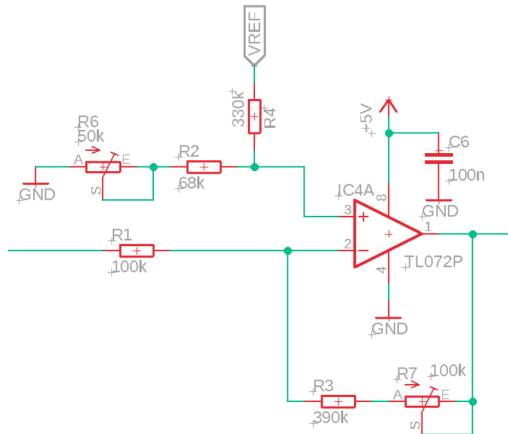


Figura 10. Etapa de amplificación y ajuste de offset en circuito acondicionador electrodo de pH.

De esta forma a través del PIN3 que corresponde al pin positivo del amplificador operacional, se introduce un voltaje que corresponderá al offset que se deba de proporcionar, el valor de voltaje Vref viene dado por la referencia de tensión AD580 la cual entrega 2.5 V, R4 junto a R2 y R6 generan un divisor resistivo ajustable desde el potenciómetro R6.

$$V_{pin3} = V_{ref} \times \frac{R2+R6}{R2+R6+R4} \quad (\text{ec.1})$$

La ganancia es fijada por el lazo de realimentación negativo cuyo valor se obtiene con la siguiente ecuación:

$$V_{pin1} = (-V_{sensor}) \times \frac{R3+R7}{R1} + \left(1 + \frac{R3+R7}{R1}\right) \times V_{pin3} \quad (\text{ec.2})$$

La última etapa del circuito corresponde a un filtro paso bajo de segundo orden con topología Sallen-Key, Figura 11, con el propósito de filtrar las posibles oscilaciones indeseables que se puedan producir en la medida de tensión.

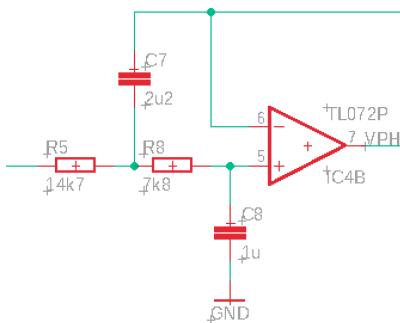


Figura 11. Filtro Sallen-Key



Esta última etapa atenuará 40 dB/dec toda aquella frecuencia inferior a la frecuencia de corte que se calcula con la siguiente expresión:

$$F_c = \frac{1}{2\pi \times \sqrt{R_5 \cdot R_8 \cdot C_7 \cdot C_8}} \quad (\text{ec.3})$$

Como se puede observar en el esquemático del anexo 3, se utilizan diferentes tensiones de alimentación para cubrir toda las necesidades de los diferentes circuitos integrados. La alimentación de entrada principal al circuito es de 24 V DC y desde este valor de tensión se obtendrán 12 V DC, 5 V DC, 3.3 V DC y -3.3 V DC. A continuación se muestra cómo se han diseñado los diferentes circuitos para conseguir todas las alimentaciones.

Partiendo de una tensión de 24 V DC y mediante el uso de reguladores de tensión, se realiza una disminución en el nivel de tensión de la siguiente forma:

- Se convierte 24 V DC a 12 V DC con el integrado LM7812, Figura 12

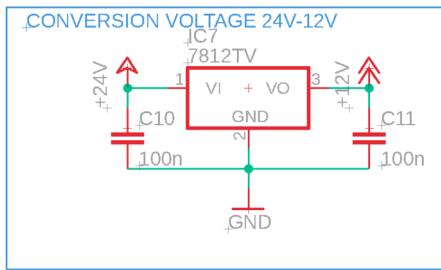


Figura 12. Esquema conversión 24 V - 12 V.

- Desde 12 V DC mediante el integrado LM7805, se convierte a 5 V DC, Figura 13.

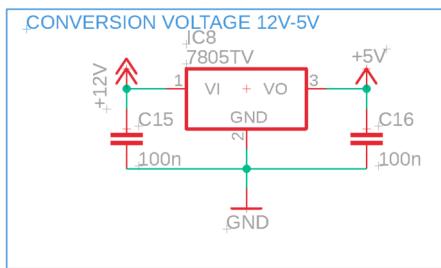


Figura 13. Esquema conversión 12 V - 5V.



- Desde 5V DC utilizando el regulador LM7833 se convierte a 3.3V DC, Figura 14.

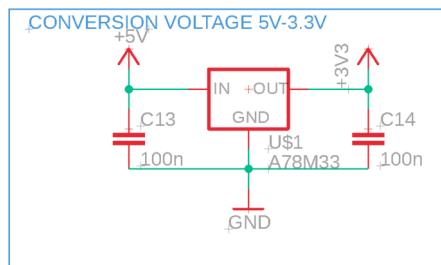


Figura 14. Esquema conversión 5 V - 3.3 V.

Para poder obtener el voltaje negativo necesario para el correcto funcionamiento del seguidor de voltaje de la primera etapa se ha utilizado el circuito integrado MAX1044, el cual si se alimenta a 5 V puede configurarse como inversor de voltaje, el esquema se ha obtenido de la hoja de datos del mismo circuito integrado y se muestra en la Figura 15.

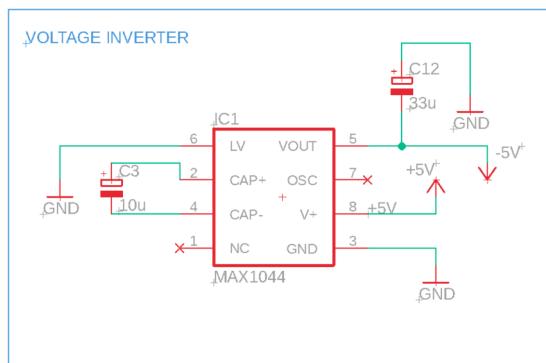


Figura 15. Esquema obtención -5 V.

En cuanto a la obtención del voltaje de referencia (Vref) utilizado en la segunda etapa del circuito acondicionador de pH, el diseño para conseguir los 2.5 V, tal y como se muestra en la Figura 16.

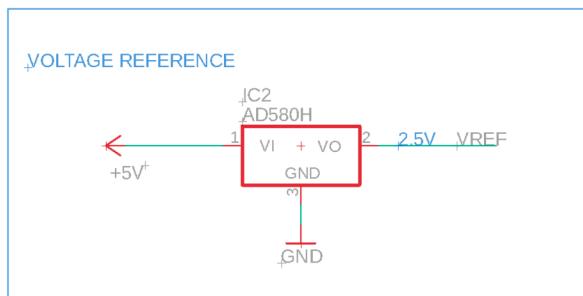


Figura 16. Referencia de voltaje AD580H.



En cuanto al sensor de temperatura y humedad, lo único que precisa el montaje es de un resistencia de Pull Up para poder transmitir los datos, el circuito en cuestión se muestra en la Figura 17.

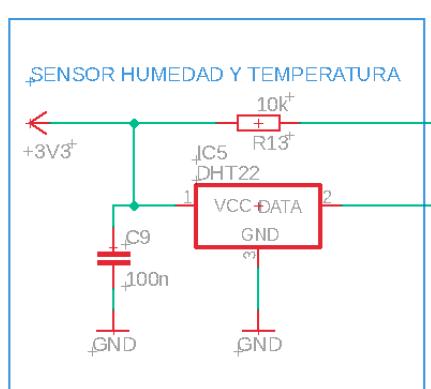


Figura 17. Adaptación circuito sensado humedad y temperatura.

En el apartado anterior se ha indicado la necesidad de utilizar diferentes accionadores que permitirán realizar las tareas de automatizado tales como la activación de calefactores y electroválvulas, para poder activar estos dispositivos, se ha diseñado un circuito de potencia formado por drivers capaces de transformar los pulsos del microcontrolador en señales de 12 V o 24 V según el dispositivo a activar. Para conseguirlo, se ha utilizado el circuito integrado TBD62083A que mediante la tecnología MOSFET consigue acondicionar la señal para poder accionar las diferentes salidas. Como se puede observar en el esquema interno mostrado en la Figura 18, el voltaje al cual se alimenta el integrado será el voltaje de salida, de forma que uno de los drivers se alimentará a 12 V mientras que otro integrado exactamente igual y con la misma configuración de entradas y salidas se alimenta a 24 V.

Equivalent circuit (each driver)

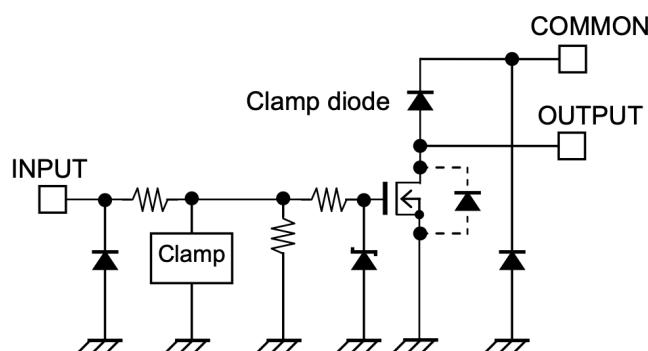


Figura 18. Circuito equivalente de cada salida del driver TBD62083A.



Para las entradas físicas se ha instalado una botonera de forma que cuando se presiona uno de los botones se envía una señal de 3.3 V al PIN del microcontrolador correspondiente, mientras no se esté enviando un estado alto al PIN del microcontrolador se conecta el PIN a tierra a través de una resistencia de 10 kΩ, el esquema realizado se muestra en la Figura 19.

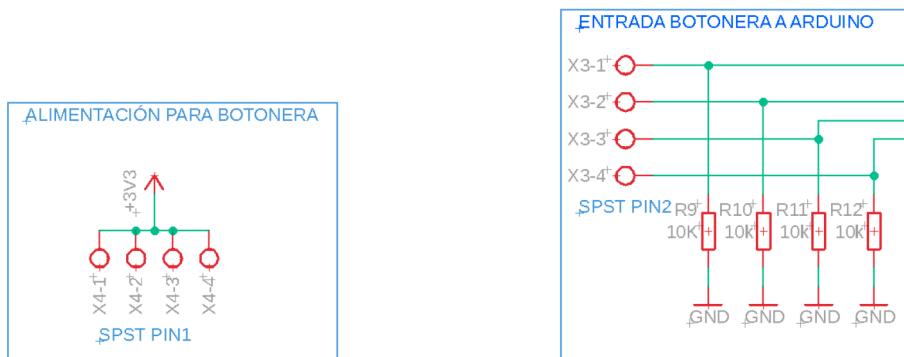


Figura 19. Conexionado botonera física.



2.1.4 Arduino.

Para programar todas las instrucciones necesarias y poder procesar todas las señales a tiempo real así como mantener la comunicación con LabView se utiliza un Arduino Fio, Figura 20.

Arduino Fio es una placa para microcontrolador que utiliza el integrado ATMEGA328P, cuenta con 14 pines de E/S digitales y 8 entradas analógicas. La característica que hace que hace mas interesante a Arduino Fio frente a otras placas, es que, esta versión de Arduino, en su reverso, cuenta con un zócalo para módulos XBee, los cuales permiten establecer la comunicación de manera inalámbrica entre diferentes módulos.

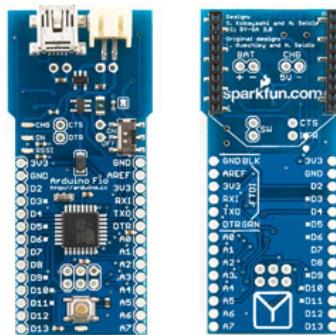


Figura 20. Placa controladora Arduino Fio.

Para este proyecto es necesario utilizar las diferentes características que Arduino Fio proporciona. En primer lugar, se ha valorado positivamente la optimización de espacio que se lleva a cabo en la placa de adquisición del proyecto gracias al reducido espacio que Arduino Fio ocupa. Sin embargo la principal característica por la que se ha escogido Arduino Fio frente a otras placas de programación es la posibilidad de incorporar los módulos XBee para establecer una comunicación inalámbrica entre los diferentes módulos y la estación desde donde se controle el sistema.

La configuración de entradas y salidas necesarias para el proyecto no requieren una placa de control compleja ni con unas características que se salgan de los parámetros estandarizados en las tarjetas de adquisición de Arduino.

Así pues, la placa Arduino Fio se adapta a la perfección al proyecto por diferentes motivos, la posibilidad de implementar módulos XBee, la facilidad a la hora de alimentar el dispositivo y su configuración de entradas y salidas.



2.1.5 Módulos Xbee

Los módulos Xbee son los encargados de establecer la comunicación entre los módulos instalados en las habitaciones de vivero y la estación de control del proyecto. Esta comunicación se realiza de manera inalámbrica de forma que incrementa la versatilidad a la hora de implementar los módulos de monitorización y control, ya que se adaptan prácticamente a cualquier distribución de vivero.

XBee incorpora un transmisor - receptor ZigBee (Figura 21), el cual permite una fácil conexión y versatilidad en su uso. De esta manera un dispositivo electrónico de bajo consumo como puede ser Arduino Fio puede establecer comunicaciones inalámbricas sin necesidad de realizar una amplia programación.

La transmisión y recepción se realiza en la banda de 2.4 GHz mediante el protocolo ZigBee. Pese a la simplicidad de uso, XBee es un dispositivo con mucho potencial, su versión más básica hace que podamos establecer comunicaciones con una distancia máxima de 100 metros en exteriores, a una velocidad de 256 kbps.

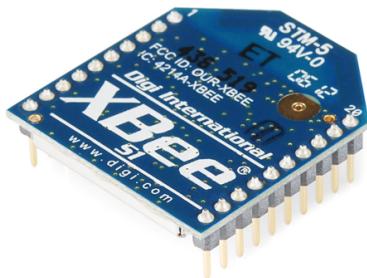


Figura 21. Módulo XBee.

Las características técnicas de XBee nos dicen que, en teoría, se pueden conectar hasta 65535 módulos XBee a una misma red, aunque en la realidad el número de dispositivos es menor. Aún así cabe destacar la versatilidad y adaptación que puede brindar XBee a un proyecto como el que se describe en este documento. Si bien todas estas características hacen que XBee encaje a la perfección en el proyecto, también lo hace su bajo consumo, el cual según los datos proporcionados por XBee es menor de 50 mA en funcionamiento y menor de 5 µA en el modo sleep.

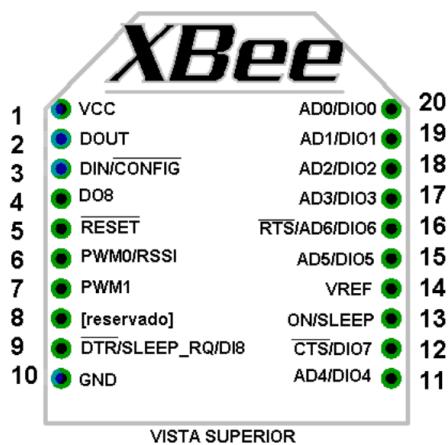


Figura 22. Conexionado de XBee.

En la Figura 22 se puede observar la distribución de pines que se presenta el módulo XBee, si se compara con la distribución de Arduino Fio, se observa que es la misma distribución, lo que garantiza un correcto funcionamiento.

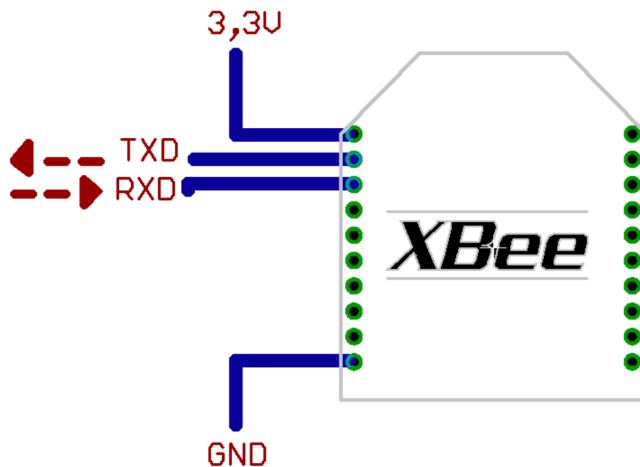


Figura 23. Conexionado transmisión XBee.

En la Figura 23, se muestra como se realiza la alimentación y transmisión a través de un módulo XBee. Esta misma distribución y conexión es la que se utiliza en el proyecto.



2.2 Software

2.2.1. Herramientas de programación

Para la realización del proyecto y su correcta implementación, ha sido necesario llevar a cabo la programación de una parte digital que gobierne los accionadores y que sepa interpretar la información que continuamente está recibiendo de cada uno de los sensores, de forma que en todo momento esté pendiente de controlar todos los procesos.

El objetivo es, que desde una sala de control, se puedan dar ordenes y establecer los parámetros ideales para el tipo de cultivo que se lleve a cabo. Esta información se pasa a Arduino mediante el módulo XBee que transmite la información por puerto serie.

Aun así, una vez establecidos los parámetros ideales para el cultivo, el módulo de control instalado en el vivero deberá de ser capaz de actuar por si mismo, es decir, pese a estar en constante comunicación con el centro de control, puede actuar de manera autónoma sin que ningún ordenador este conectado. Así se evita que sea necesaria la constante comunicación del centro de control con los diferentes módulos para poder llevar a cabo los diferentes mecanismos de auto regulación.

Por tanto, aunque el cerebro del kit de monitorización y control sea Arduino Fio, desde el software LabView se lleva a cabo un procesamiento a tiempo real de todos los datos recogidos en el vivero.

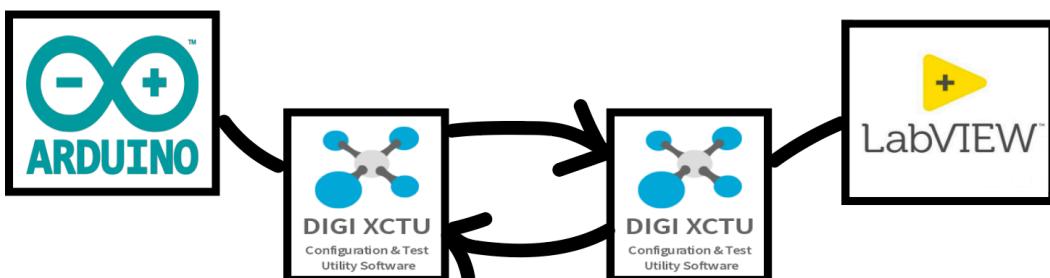


Figura 24. Esquema software del proyecto.

Como se puede observar en la Figura 24, se han tenido que utilizar diferentes herramientas de programación. La herramienta principal y el componente imprescindible del proyecto y la encargada de gobernar el sistema es Arduino Fio.

Todos aquellos sensores y actuadores descritos en el apartado de hardware se conectan directamente a la placa de control de Arduino Fio. En total van conectados 6 sensores y 6 actuadores, por lo que, para entender como el kit de control y monitorización procesa la información a nivel de software, se debe conocer como es el esquema de distribución mostrado en la Figura 25.

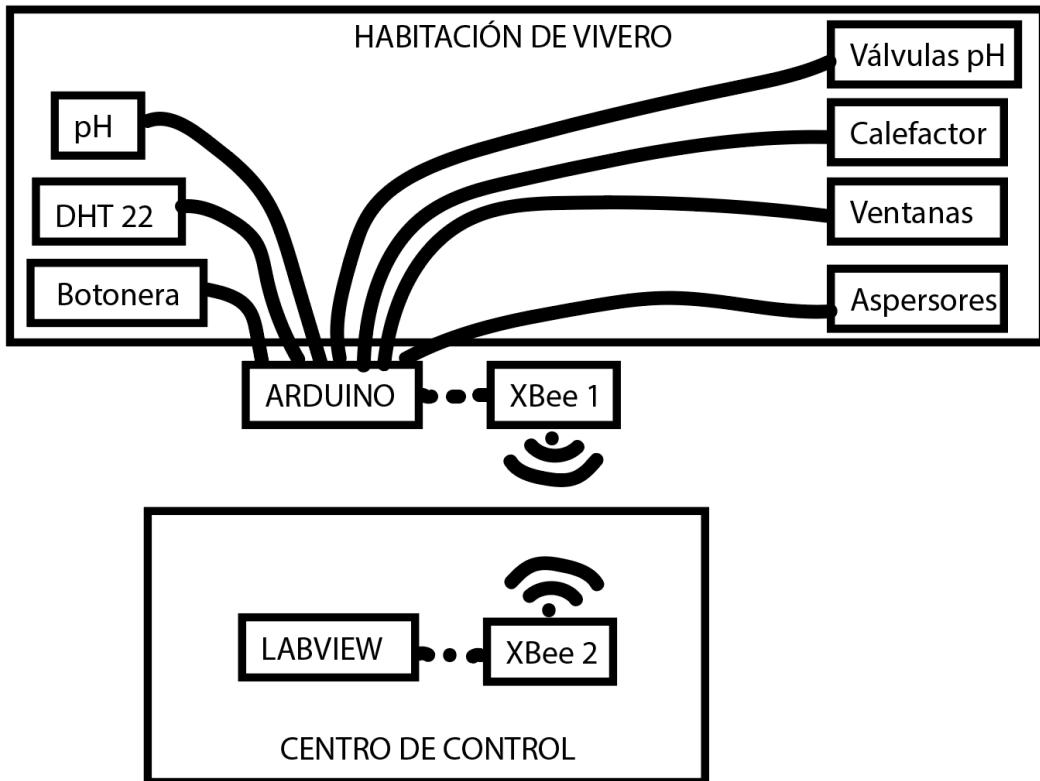


Figura 25. Esquema proyecto.

2.2.2 Programación en Arduino

La programación de Arduino se divide en dos partes, o mejor dicho, dos modos de funcionamiento, el primer modo de funcionamiento se lleva a cabo cuando Arduino esta comunicándose con el centro de control, mientras que el segundo modo de funcionamiento permite que Arduino funcione sin necesidad de estar transmitiendo datos con LabView.

Para funcionar de forma automatizada, es imprescindible que Arduino haya recibido los datos introducidos por LabView, es decir, que tenga los parámetros de temperatura, humedad y pH para poder llevar a cabo el control.

La primera instrucción que realiza Arduino en cada ciclo es comprobar si está comunicando por puerto serie con LabView, de esa forma, Arduino selecciona el modo de funcionamiento a establecer. Si Arduino se está comunicando con LabView, realizará las instrucciones a la vez que enviará los datos al centro de control.

Se ha mostrado la parte de Hardware que atañe al circuito acondicionador de pH así como de humedad y temperatura, no obstante no se ha mostrado como preparar a nivel software esta información que reciben los sensores.



En cuanto al sensor de humedad y temperatura, se cuenta con la ventaja de estar diseñado para comunicar con Arduino, de forma que existe una librería preparada para este tipo de sensores, mas concretamente, la librería <DHT.h>.

El protocolo que utiliza este sensor relativamente simple, la comunicación se realiza de la siguiente forma:

Mientras estamos en estado de reposo o en estado IDLE, el Arduino esta leyendo un estado alto continuo, una vez el MCU está preparado para leer información, coloca el PIN asignado al sensor en estado bajo por 18us y posteriormente lo coloca en estado alto entre 20 μ s y 40 μ s esto indica que el MCU esta preparado para recibir información, por lo que el sensor envía el estado alto por 80us para indicar que esta listo, una vez se vuelve a poner en estado bajo significa que ya esta transmitiendo los bits de información para que el MCU los interprete. En resumen la secuencia es la siguiente:

- Pin asignado a DHT en estado alto en Modo IDLE.
- El Arduino coloca el PIN asignado en estado bajo por 18us.
- Arduino lo vuelve a colocar en estado alto de 20us a 40us esperando respuesta del sensor.
- El sensor responde poniendo el PIN en estado bajo 80us.
- El sensor pone el PIN en estado alto 80us para indicar que esta listo para transmitir.
- El sensor pone el PIN a estado bajo 50us antes de transmitir cada BIT.

La trama de datos queda estructurada de la siguiente manera:

- 1º byte: Parte entera de la humedad relativa.
- 2º byte: parte decimal de la humedad relativa.
- 3º byte: Parte entera del valor de la temperatura.
- 4º byte: Parte decimal de la temperatura.
- 5º byte: Envía la Checksum, que son los últimos 8 bits de la suma de los 4 primeros bytes.

Las siguientes líneas de código muestran la inicialización del sensor DHT22.

```
//Añadimos librerias necesarias para DHT22
#include <DHT.h>
#define DHTTYPE DHT22
const int DHTPin = 5;
```

Una vez mostrado el proceso de lectura de humedad y temperatura, se procede a estudiar cómo interpretar la información que proviene del circuito de acondicionamiento de pH. Para llevar a cabo la lectura de este parámetro, se ha utilizado la entrada analógica A0 de Arduino Fio, la cual es capaz de leer un voltaje de 0 a 3.3 V y pasarlo a una cuenta de 0 a 1023, por lo que, Arduino no nos proporciona de manera directa el voltaje proveniente del circuito de alimentación de PH, sino que se ha de traducir esta cuenta a una información de



utilidad. Se puede hacer de forma sencilla ya que es una conversión lineal. A continuación se indica una forma simple de convertirlo con Arduino.

```

PH_medido = analogRead(A0); //Leemos el PH de 0 a 1023
PH_volt = map(PH_medido, 0, 1023, 0, 5);
float m_ph = (0.3333 + (0.0012201*temp))*(-1);
float b_ph = 2.5 - (m_ph * 7);
PH_convertido = (PH_volt - b_ph) / m_ph;
if (PH_convertido < 10.0) {PH_LV=PH_convertido*1000;}
else {PH_LV=PH_convertido*100;}

```

En las líneas de código anteriores, se puede observar la conversión a nivel de código de la lectura de tensión leída en la entrada analógica A0 a un valor de pH sin decimales que se puede enviar por puerto serie. Nótese que la conversión se ha realizado a 5 V debido a que las pruebas se han realizado en placa de Arduino Uno.

Estas líneas de código se deben ajustar al intervalo de pH que se quiera calcular, en el ejemplo se ha calculado para una ganancia en el circuito acondicionador de 6.15, lo que corresponde aproximadamente a un pH de 5.5 a 8.5, lo cual sería un rango válido para nuestras mediciones, aunque no tendría porque serlo para todo tipo de cultivos, por lo que esta característica es reprogramable.

Como todos los electrodos de PH, tienen variaciones en la medida conforme lo hace la temperatura del medio en el que se encuentra, se debe de realizar una interpolación lineal atendiendo a la temperatura del medio.

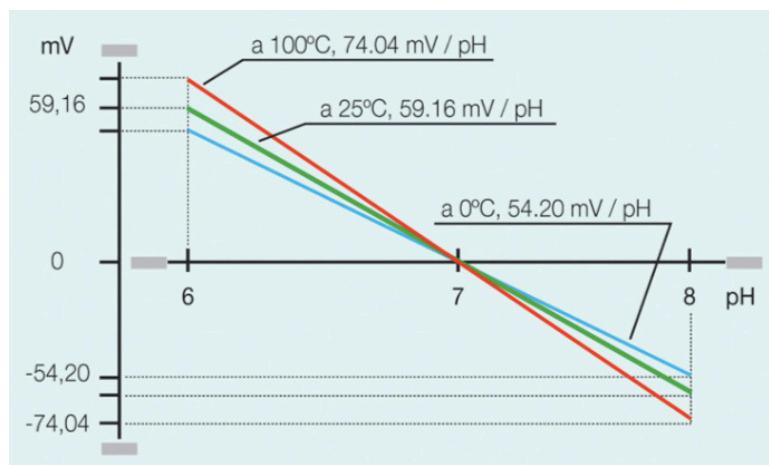


Figura 26. Comportamiento del electrodo de pH a diferentes temperaturas.

En la Figura 26 se puede observar como la relación voltaje – pH varía dependiendo de la temperatura del medio, para ello es necesario que cada vez que se realice una medición de pH se tenga en cuenta la temperatura a la que se está realizando la medida.

De esta forma, la manera de calcular el pH se realiza a través de diferentes cálculos. El electrodo describe un comportamiento lineal a una misma temperatura, por lo que, se calcula el pH en base a la ecuación de una recta. En



primer lugar, se debe de conocer el factor de ganancia que tiene nuestro circuito de acondicionamiento, en este caso los cálculos se han realizado para una ganancia 6.15.

Realizando los cálculos necesarios se concluye que, cada grado que aumente la temperatura, la sensibilidad del sensor aumenta $1,22 * 10^{-3} \frac{V}{pH}$, por lo que, en la lectura que proporciona el circuito acondicionador de pH, para 0 °C, la respuesta del circuito es de $0.333 \frac{V}{pH}$, por lo que, la pendiente de la recta en base a la temperatura medida se puede calcular utilizando ec.4

$$\text{Pendiente} = (0.333 + 0.0012201 * \text{temperatura medida}) * (-1) \quad (\text{ec.4})$$

La ecuación de la recta que relaciona el pH con la tensión se corresponde con ec.5

$$\text{Voltaje} = m * pH + b \quad (\text{ec.5})$$

Siendo "m" la pendiente calculada y siendo "b" la ordenada en el origen calculada utilizando ec.6.

$$b = 2.5 V - (\text{pendiente} * 7). \quad (\text{ec.6})$$

Como se puede observar en la gráfica de la Figura 26, para un voltaje de 0 V en el electrodo (lo que corresponde a 2.5 V a la salida del circuito de acondicionamiento) el pH toma un valor de 7.

Una vez conocidas la pendiente y la ordenada en el origen, se calcula el correspondiente valor de pH utilizando (ec.4) y (ec.5).

```
float m_ph = (0.3333 + (0.0012201*temp))*(-1);
float b_ph = 2.5 - (m_ph * 7);
PH_convertido = (PH_volt - b_ph) / m_ph;
```

El cálculo de temperatura y de pH se realiza de forma directa con las instrucciones mostradas en las siguientes líneas de código:

```
hum = dht.readHumidity(); // Leemos la humedad relativa
temp = dht.readTemperature(); // Leemos la temperatura en grados centígrados (por defecto)
int hume= 100*hum;
int tempe= 100*temp;
```

Las líneas de código indicadas se deben procesar a cada ciclo del programa para garantizar un correcto funcionamiento, posteriormente, como indica el diagrama de flujo, se procede a comparar los valores sensados con los valores objetivo para proceder, si es necesario, a la activación de algún actuador.



```

if (datoslabview[0]=='a') {digitalWrite(s_ventanas,HIGH);ventana=1;}//Oertura manual de ventanas desde labview
else if ((hum > hum_INT+ 1000) && (func_manual==0)) {digitalWrite(s_ventanas,HIGH);ventana=1;} //Si la humedad es
else if (digitalRead(b_ventanas)==HIGH) {digitalWrite(s_ventanas,HIGH);ventana=1;}//Si presionamos el boton fisic
else {digitalWrite(s_ventanas,LOW);ventana=0;}

if (datoslabview[1]=='b') {digitalWrite(s_calefactor,HIGH);calefactor=1;}//Enciendo el calefactor de forma manual
else if ((temp < temp_INT-300) && (func_manual==0)) {digitalWrite(s_calefactor,HIGH);calefactor=1;}//El calefactor
else if (digitalRead(b_calefactor)==HIGH) {digitalWrite(s_calefactor,HIGH);calefactor=1;}
else {digitalWrite(s_calefactor,LOW);calefactor=0;}

```

Como se observa en el código anterior, existen diferentes condiciones por las que un actuador se acciona, hasta tres condiciones pueden llevarse a cabo para que un accionador se active.

El resultado de diferentes comparaciones proporcionará el valor que se le debe de dar al actuador, la primera de estas comparaciones tiene como base un accionamiento “manual del activador” desde LabView, donde mediante puerto serie se transmite un string de datos donde cada posición del string contiene un carácter, dependiendo del carácter que se envié por LabView se dará un valor u otro al actuador.

La segunda condición para activar el actuador viene dada por el resultante de la comparación del parámetro que hayamos leído y el valor objetivo que desde LabView se haya indicado. Para conseguir activar el actuador en esta segunda condición, además se debe de cumplir que la entrada digital de “funcionamiento manual” que se activa desde LabView esté desactivada.

El funcionamiento manual activable desde LabView funciona de tal manera que una vez activado, el sistema de control deja de gobernar los actuadores en base a los parámetros que haya enviado LabView, esto puede ser realmente útil para poder hacer una cierta puesta en marcha o bien para realizar algún movimiento en el vivero, donde se desea seguir registrando los diferentes parámetros sensados en el registro de actividad pero se quiere que los actuadores se accionen automáticamente por algún motivo, en la siguiente línea de código se muestra como se realiza la activación de esta funcionalidad.

```
if (datoslabview[6]=='g'){func_manual=1;} //Leemos de LabView si estamos funcionando de forma manual
```

La tercera de las comparaciones considera el estado de los botones físicos instalados en el vivero, que pueden servir para accionar una ventana o un aspersor de forma manual cuando se necesite.

Cada vez que uno de los actuadores toma el valor “1” se transmite a LabView, donde en todo momento, mediante una interfaz muy intuitiva, se pueden monitorizar los valores de los sensores y a la vez cuál es el estado de cada actuador.



```
for (int i=8;i<12;i++)//Aqui cargo los dc
{temperatura += datoslabview[i];//siempre
 humedad += datoslabview[i+3];//siempre v
 PH += datoslabview[i+6];}//siempre va a
```

Como se observa en las líneas de código anteriores, se realiza una lectura de los parámetros objetivo que el usuario introduce en LabView, de forma que, para poder realizar las comparaciones necesarias y determinar el estado de los actuadores, se ha de conocer en todo momento los valores introducidos en LabView. Esta operación se realiza leyendo un string de datos que constantemente se está enviando desde LabView a Arduino, de forma que cada uno de los bits de dicho string representa un dato que Arduino interpreta.

En las líneas de código mostradas anteriormente se puede observar como en cada una de las variables de temperatura, pH y humedad se cargan cuatro bits, en total, los bits que contienen la información para estas variables atan a los bits 8-17 de la cadena de datos que envía LabView a Arduino.

Mediante un bucle “for” se van cargando los valores en las variables que corresponda, estas variables son de tipo de dato **char** y se deben de transformar a enteros mediante las siguientes instrucciones.

```
temp_INT=temperatura.toInt();//Aqui lo que hace es
hum_INT=humedad.toInt();
PH_INT=PH.toInt();

temperatura="";//Reseteamos las string para que
humedad="";
PH="";
```

También se han de resetear todas las variables para que al volver a iniciar el ciclo no se concaténen y evitar cualquier tipo de fallo. El paso de datos de Arduino a Labview en cambio se realiza con la función **serial.print**, la cual envía de la siguiente forma los datos mediante puerto serie.



```

Serial.print(PH_LV); //Valor PH
Serial.print('\t');
Serial.print(chume); //Valor humedad
Serial.print('\t');
Serial.print(temp); //Valor temperatura
Serial.print('\t');
Serial.print(ventana); //Estado Ventanas
Serial.print('\t');
Serial.print(calefactor); //Estado Calefactor
Serial.print('\t');
Serial.print(phpos); // Estado Valvula PH+
Serial.print('\t');
Serial.print(phneg); // Estado Valvula PH-
Serial.print('\t');
Serial.println(aspensor); // Estado aspersor
delay(50);

```

Recapitulando, en la programación de Arduino, se distinguen dos modos de funcionamiento para que Arduino gobierne el vivero, la que se ha detallado hasta ahora, es decir, cuando Arduino trabaja juntamente con LabView y existe un intercambio de datos en tiempo real, o bien cuando Arduino debe de trabajar de forma independiente a la interfaz de usuario (LabView), como se observa en las siguientes líneas de código.

```

else{
    if (digitalRead(b_ventanas)==HIGH) {digitalWrite(s_ventanas,HIGH);}
    else if (hum > hum_INT+ 1000) {digitalWrite(s_ventanas,HIGH);}
    else {digitalWrite(s_ventanas,LOW);}

    if (temp < temp_INT-300) {digitalWrite(s_calefactor,HIGH);}
    else if (digitalRead(b_calefactor)==HIGH) {digitalWrite(s_calefactor,HIGH);}
    else {digitalWrite(s_calefactor,LOW);}

    if (PH_LV > PH_INT){digitalWrite(s_phpos,HIGH)}//La valvula de nivelado se abre para nivelar el PH
    else {digitalWrite(s_phpos,LOW);}

    if (PH_LV > PH_INT){digitalWrite(s_phneg,HIGH)}//La valvula de nivelado se abre para nivelar el PH
    else {digitalWrite(s_phneg,LOW);}

    if (hum < hum_INT - 1000){digitalWrite(s_aspensor,HIGH)}//Si la humedad es muy baja un aspersor se activa
    else if ( digitalRead(b_aspensor)==HIGH ){digitalWrite(s_aspensor,HIGH);}
    else {digitalWrite(s_aspensor,LOW);}
}

```

Una vez se comprueba que no está disponible el puerto serie, Arduino toma la alternativa y hace las comparaciones necesarias para seguir sensando y comparando con los valores objetivos, de forma que, no se transmite ni envía ningún tipo de dato al centro de control.

Realmente el comportamiento es idéntico al que realizaría con LabView, con la diferencia de que no se recopila ningún tipo de información. Es imprescindible contar con un modo de funcionamiento alternativo al que se lleva a cabo comunicando con LabView. Cuando por cualquier motivo no se pueda establecer comunicación, cierto es que todo entorno industrial requiere de un cierto mantenimiento periódico, por lo que la intención de realizar esta alternativa al funcionamiento normal es intentar seguir teniendo el vivero completamente automatizado, aunque con ciertas limitaciones, pero al fin y al cabo se puede seguir trabajando.



2.2.3 Interfaz de usuario.

Se ha mencionado diferentes veces que el objetivo principal de este proyecto es ofrecer versatilidad y un diseño adaptable al sitio donde se vaya a implementar y desarrollar, es por lo que la programación siempre quedará abierta a posibles modificaciones para poder adaptarse a las diferentes condiciones de funcionamiento que se puedan encontrar. Para diseñar la interfaz de usuario, se ha buscado la máxima simplicidad y sencillez de uso facilitando de este modo el trabajo al operario que la manipule.

Se sabe que cualquier operario que trabaje en un vivero en algún momento necesitará consultar el estado de este mismo o bien manipular los parámetros objetivo para poder introducir nuevos valores, por lo que dicha interfaz debe de ser muy intuitiva.

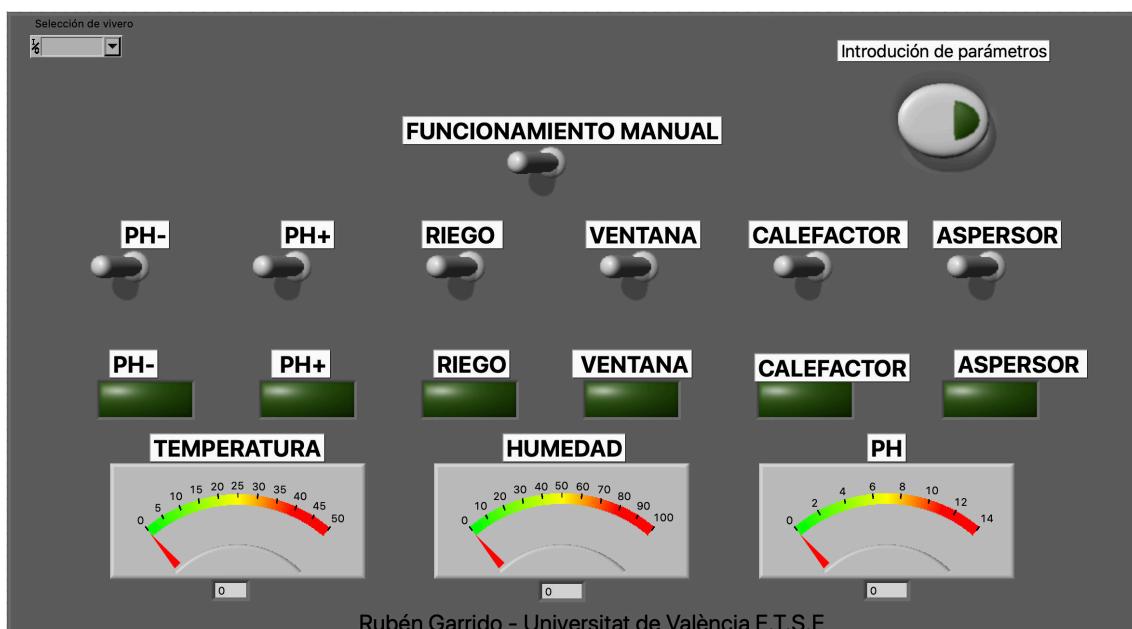


Figura 27. Interfaz de usuario.

Como se puede observar en la Figura 27, la interfaz cuenta con dos bloques principales, un primer bloque de interruptores y un segundo bloque de indicadores donde se puede ver que actuadores están funcionando en cada caso y cuales son los parámetros sensados. Además, en la esquina superior izquierda se encuentra el selector de habitación, en el desplegable se encontrarían las diferentes habitaciones que se pueden monitorizar y controlar.

Por último, en la esquina superior derecha se encuentra un botón cuya función es abrir un VI complementario para realizar la introducción de parámetros. La introducción de parámetros se debe de realizar siempre al inicializar el control de una habitación con cultivo nuevo o bien para realizar algún ajuste si el avance del cultivo no es adecuado.



Figura 28. Bloque de interruptores.

Como es de esperar, cada uno de los interruptores se encarga de activar uno de los accionadores del vivero, es decir, si se pulsa cualquiera de los interruptores se activará la salida correspondiente al interruptor seleccionado y a la misma vez, se encenderá el piloto indicador, tal y como se muestra en la Figura 28, en la que se ha activado manualmente la salida de riego y el calefactor.

Estos interruptores funcionan aunque no se tenga el interruptor de funcionamiento manual pulsado, es decir, si se acciona cualquier interruptor sin activar previamente el funcionamiento manual, el actuador en cuestión se activará. Sin embargo, la regulación automática de temperatura, pH y humedad que realiza Arduino seguirá llevándose a cabo mientras que, si previamente se ha activado el modo de funcionamiento manual, cesarán por completo las labores de autorregulación.



Rubén Garrido - Universitat de València E.T.S.E

Figura 29. Indicadores de la interfaz de usuario.

En el bloque inferior referente a los indicadores, se pueden observar los pilotos asociados a cada una de las salidas, tal y como muestra la Figura 29. Los indicadores de temperatura, pH y humedad que cuentan con un visor tipo analógico y un visor numérico que indican las condiciones a las que se encuentra el vivero a tiempo real en el vivero.

Por último, para introducir los parámetros objetivo, se ha colocado en la mitad superior un botón que despliega una nueva pestaña perteneciente al SubVI de la introducción de parámetros, de forma que al pulsar este botón, automáticamente el usuario será conducido al desplegable donde se presentarán las instrucciones para introducir los parámetros.



Figura 30. Pulsador introducción de parámetros.

En la Figura 30 se muestra el botón de introducción de parámetros que conduce al desplegable con las indicaciones para realizar la inicialización del vivero o el reajuste de parámetros. Estas indicaciones se muestran en la Figura 31.



Figura 31. SubVI Introducción de parámetros.

La interfaz de usuario se ha diseñado de tal forma que simplemente se deben de seguir las instrucciones proporcionadas por la interfaz, no será necesario tener que cerrar de manera manual este desplegable y volver a buscar la interfaz original, simplemente haciendo click en los botones de inicializado y finalizado correspondientes, se abrirán o cerrarán los diferentes bloques. De esta forma se facilita al máximo la labor de usuario.



2.2.4 Programación en LabView.

Llegados a este punto, se ha mostrado la programación del microcontrolador de la placa de Arduino y el funcionamiento de la interfaz de usuario programada. A continuación se estudiará mas a fondo como se ha programado el código de LabView para hacer que funcione juntamente con Arduino.

Se ha mencionado repetidamente que el modo de comunicación es a través de puerto serie. Por la sencillez de los datos a transmitir, este método es el mejor, ya que abarata mucho los costes y reduce mucho la complejidad de la programación. Tanto Arduino como LabView envían cadenas de datos, cuyos valores son comparados con los datos que se esperan recibir para realizar cambios o no en las variables de programa que a su vez pueden afectar a alguna de las salidas.

Igual que Arduino, LabView se encarga de recibir esa información y compararla con la información que espera recibir para llegar a realizar o no algún tipo de interacción.

Así como Arduino era capaz de funcionar sin tener que estar conectado a LabView, las funciones que pueda realizar LabView carecen de sentido si no están conectadas a Arduino, de forma que no es posible arrancar el programa si no se está recibiendo ningún tipo de dato por puerto serie.

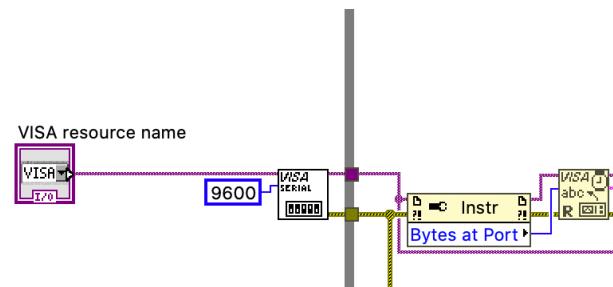


Figura 32. Inicio de comunicación por puerto serie.

Como se observa en la Figura 32, la comunicación por puerto serie se establece a 9600 baudios. La programación de LabView se divide en dos partes al igual que la de Arduino, un bloque de lectura de datos y otro bloque de escritura de datos.

El bloque de lectura de datos consta de diferentes apartados, como se ha explicado con anterioridad, la comunicación se realiza a través de strings o cadenas de datos, las cuales transportan en cada una de sus posiciones un valor interpretable por LabView.

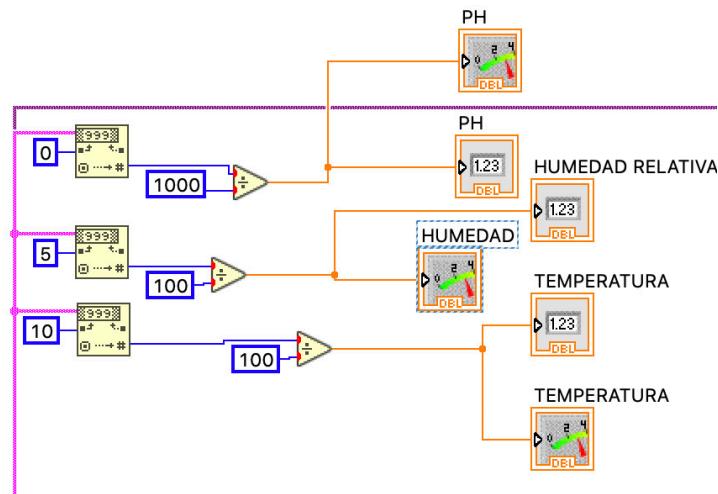


Figura 33. Ejemplo lectura en LabView.

Como se muestra en la Figura 33, los datos que se envían a LabView desde Arduino se van desglosando para que sean interpretables por el usuario. Cada variable tiene un tamaño determinado, luego es LabView el que interpreta las variables mediante las operaciones necesarias. La función del código es mostrar el valor de pH, temperatura y humedad sensado.

No obstante, no sucede lo mismo para las demás variables, donde las funciones de comparación que se llegan a realizar en LabView simplemente se utilizan para iluminar un piloto en la interfaz de usuario, de esta forma, solo con mirar la interfaz, el usuario puede hacerse una idea de como está trabajando el vivero y cuales son las condiciones a las que éste está trabajando. Se muestra como se ha realizado esta parte de la programación en la Figura 34.

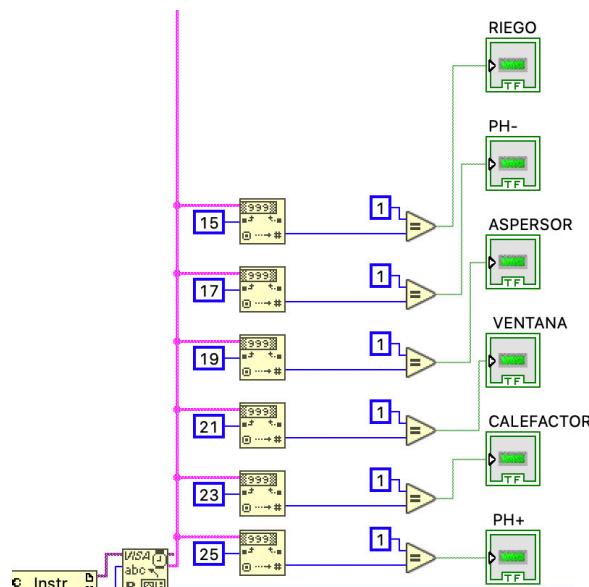


Figura 34. Encendido de pilotos en LabView.



En cuanto a la escritura de datos, la programación sigue un camino bastante parecido, se ha mostrado que dentro del entorno Arduino, al realizar una sentencia comparativa para activar algún actuador, cada variable leída de LabView se compara con un letra, y es que, según está hecho el programa de LabView, cada una de las diferentes variables genera un carácter, este carácter puede adoptar el valor “x” cuando no queremos activar una salida y un valor diferente de “x” para poner en marcha el actuador correspondiente. La construcción de este string de datos se puede observar en la Figura 35.

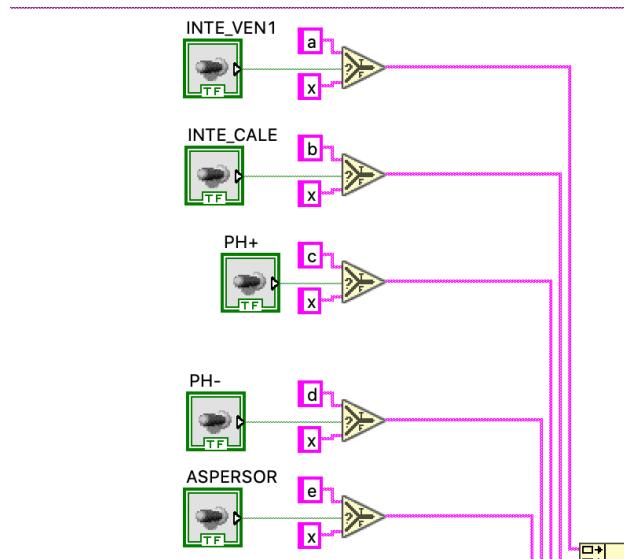


Figura 35. Escritura de datos en LabView.

Tal y como ocurre con la lectura de las variables numéricas de la temperatura, humedad y pH, para realizar la escritura de estas variables en la cadena de datos que LabView envía a Arduino se reservan unos determinados bits (8-17). La lectura de estos datos por parte de Arduino se ha mostrado en apartado anterior.

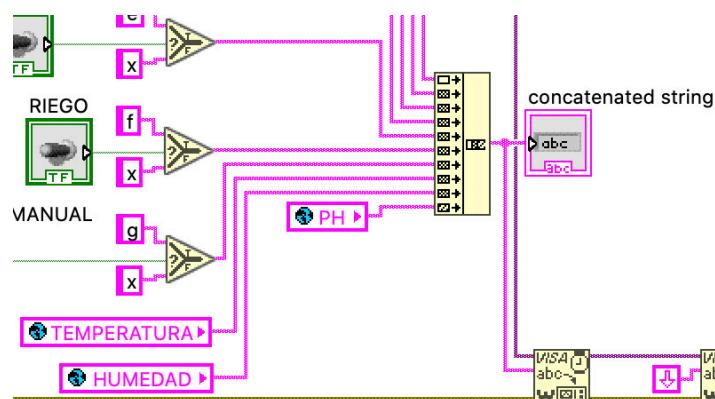


Figura 36. Variables globales en LabView.

Como se muestra en la Figura 36, para los parámetros de pH, temperatura y humedad se ha almacenan en variables globales. La utilización de un SubVI específico para la introducción de parámetros es el motivo por el que estos



parámetros se deben de almacenar en una variable que este sincronizada entre ambos VIs.

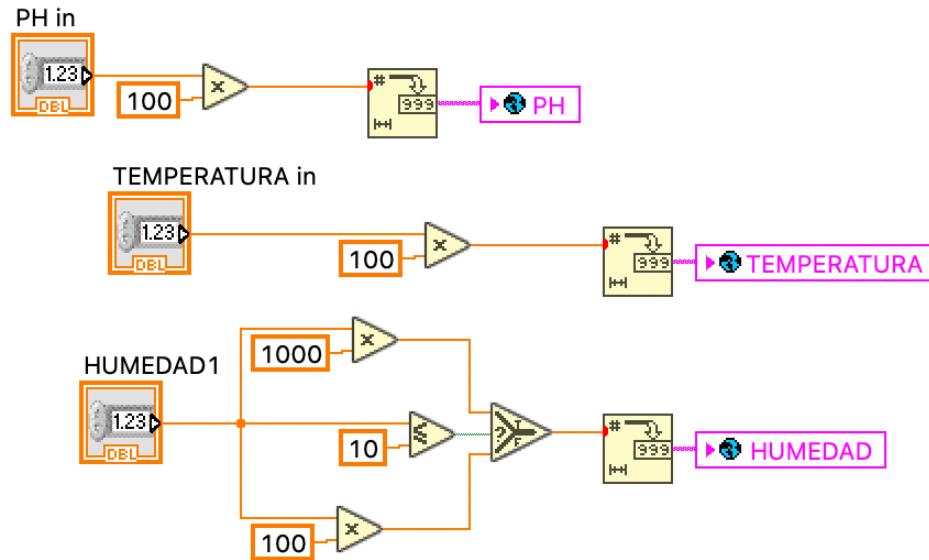


Figura 37. SubVI introducción de parámetros.

El SubVI de introducción de parámetros se estructura de forma simple, como se observa en la Figura 37, se introduce el parámetro indicado siguiendo las indicaciones mostradas en la interfaz. Para poder enviarlo a Arduino, se ha de asegurar que envía un número entero, es por ello por lo que todos los tipos de datos de carácter no booleano están sujetos a multiplicaciones o divisiones para poder convertir el valor a un parámetro que pueda interpretar el entorno de programación donde estemos y que la trama que se envíe de un entorno a otro carezca de errores.

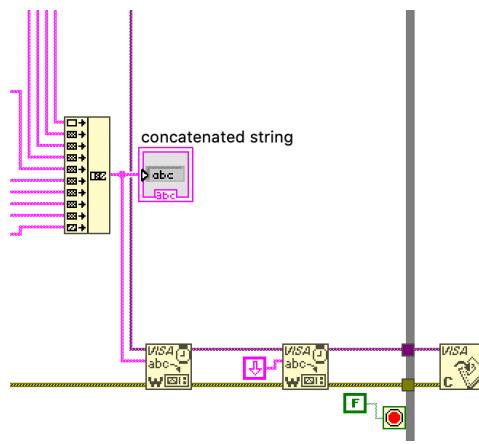


Figura 38. Concatenación de datos para transmisión puerto serie.

En la Figura 38 se muestra como se realiza la concatenación de todas las variables mostradas para poder generar el string y enviarlo a Arduino.



2.2.5 Comunicación con módulos XBee.

El último bloque de programación lo forman los módulos XBee. La programación de estos módulos es relativamente inmediata, para realizar la programación de estos dispositivos, se ha utilizado el software proporcionado por el fabricante Digi.

Debido a la situación excepcional durante la que se ha desarrollado el proyecto, resultaba difícil poner en marcha todas y cada una de las especificaciones, por lo que, al no contar con Arduino Fio, se han programado dos módulos XBee para poder comprobar que la comunicación entre ambos se podía llevar a cabo. Por ello se han conectado los dos módulos a dos puertos diferentes del ordenador y de este poder comunicarlos mediante el envío de tramas. Los módulos utilizados se muestran en la Figura 39.

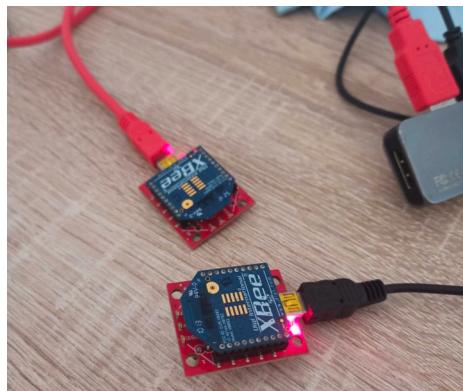


Figura 39. Módulos XBee utilizados.

La programación de ambos módulos se ha hecho mediante el programa XCTU y la configuración de cada uno de los módulos se puede observar en las Figuras 40 y 41.

Product family: XB24-ZB		Function set: ZigBee Router AT		Firmware version: 22A0
Networking Change networking settings				
<i>i</i> ID PAN ID	262	<i>i</i> SC Scan Channels	FFFF	Bitfield
<i>i</i> SD Scan Duration	3	<i>i</i> exponent		
<i>i</i> ZS ZigBee Stack Profile	0			
<i>i</i> NJ Node Join Time	FF	x 1 sec		
<i>i</i> NW Network Watchdog Timeout	0	x 1 minute		
<i>i</i> JV Channel Verification	Enabled [1]			
<i>i</i> JN Join Notification	Enabled [1]			
<i>i</i> OP Operating PAN ID	262			
<i>i</i> OI Operating 16-bit PAN ID	CDOC			
<i>i</i> CH Operating Channel	13			
<i>i</i> NC Number of Remaining Children	C			
Addressing Change addressing settings				
<i>i</i> SH Serial Number High	13A200	<i>i</i> SL Serial Number Low	40AA1972	
<i>i</i> MY 16-bit Network Address	81B7	<i>i</i> DH Destination Address High	0	
<i>i</i> DL Destination Address Low	FFFF	<i>i</i> NI Node Identifier		
<i>i</i> NH Maximum Hops	1E			

Figura 40. Configuración XBee router.



Product family: XB24-ZB Function set: ZigBee Coordinator AT Firmware version: 20A0

Networking
Change networking settings

i ID PAN ID	262
i SC Scan Channels	FFFF Bitfield
i SD Scan Duration	3 exponent
i ZS ZigBee Stack Profile	0
i NJ Node Join Time	FF x 1 sec
i OP Operating PAN ID	262
i OI Operating 16-bit PAN ID	CDDC
i CH Operating Channel	13
i NC Number of Remaining Children	A

Addressing
Change addressing settings

i SH Serial Number High	13A200
i SL Serial Number Low	40A8C2FE
i MY 16-bit Network Address	0
i DH Destination Address High	0
i DL Destination Address Low	FFFF
i NI Node Identifier	
i NH Maximum Hops	1E
i BH Broadcast Radius	0
i AR Many-to-One Route Broadcast Time	FF x 10 sec
i DD Device Type Identifier	30000

Figura 41. Configuración XBee coordinador.

Para las pruebas de funcionamiento se ha comprobado que ambos módulos eran capaces de enviar y recibir tramas, para ello se han enviado paquetes por puerto serie para comprobar su correcto funcionamiento.

- 0013A20040AA1972 - 0013A20040A8C2FE

Close Record Detach CTS CD DSR DTR RTS BRK Tx Bytes: 57 Rx Bytes: 60

Console log

```
Esto es una prueba para el TFG de
Ruben Garrido para ETSE UV
Esto es otra prueba del TFG de Ruben Garrido
para ETSE UV
45 73 74 6F 20 65 73 20 75 6E 61 20 70 72 75 65 62 61 20 70 61 72 61 20 65 6C
20 54 46 47 20 64 65 20 52 75 62 65 6E 20 47 61 72 72 69 64 6F 20 70 61 72 61
20 45 54 53 45 20 55 56 45 73 74 6F 20 65 73 20 6F 74 72 61 20 70 72 75 65 62
61 20 64 65 6C 20 54 46 47 20 64 65 20 52 75 62 65 6E 20 47 61 72 72 69 64 6F
20 70 61 72 61 20 45 54 53 45 20 55 56
```

Figura 42. Prueba envío y recepción de tramas en XBee 1.

- 0013A20040AA1972 - 0013A20040A8C2FE

Close Record Detach CTS CD DSR DTR RTS BRK Tx Bytes: 60 Rx Bytes: 57

Console log

```
Esto es una prueba para el
TFG de Ruben Garrido para
ETSE UV
Esto es otra prueba
del TFG de Ruben Garrido para
ETSE UV
45 73 74 6F 20 65 73 20 75 6E 61 20 70 72 75 65 62 61 20 70 61 72 61 20 65 6C 20 54
46 47 20 64 65 20 52 75 62 65 6E 20 47 61 72 72 69 64 6F 20 70 61 72 61 20 45 54 53
45 20 55 56 45 73 74 6F 20 65 73 20 6F 74 72 61 20 70 72 75 65 62 61 20 64 65 6C 20
54 46 47 20 64 65 20 52 75 62 65 6E 20 47 61 72 72 69 64 6F 20 70 61 72 61 20 45 54
53 45 20 55 56
```

Figura 43. Prueba envío y recepción de tramas XBee 2.



Como se puede observar en las Figuras 42 y 43, ambos módulos se han podido comunicar entre sí, ya que cada uno de ellos ha podido enviar y recibir la trama con éxito. De esta forma se ha determinado que los módulos utilizados para realizar estas pruebas se pueden utilizar con un funcionamiento exitoso en este proyecto.



Capítulo 3. Resultados experimentales.

Antes de dar por finalizada la programación se debe realizar una puesta en marcha física de los diferentes diseños realizados para validar su correcto funcionamiento.

Debido a la situación excepcional durante la que se ha desarrollado el proyecto, la puesta en marcha física no se ha podido efectuar de forma normal, haciendo uso de los laboratorios, y se ha tenido que adaptar a la escasez de equipo y materiales, pese a ello se ha intentado realizar de la forma mas parecida posible con la realidad.

A continuación se muestra el diseño final que no se ha podido fabricar, en cambio, en base a este diseño, se han fabricado diferentes prototipos con los que se han realizado las simulaciones físicas.

3.1 Implementación sistema de adquisición.

Para llevar a cabo la puesta en marcha del proyecto se ha diseñado una placa de circuito impreso donde van soldados todos los componentes necesarios para hacer que el sistema funcione. A lo largo de la memoria se han estudiado las diferentes partes a nivel de programación y a nivel de esquema electrónico. Si bien esta parte es de las mas importantes para el proyecto, a su vez también lo es un buen diseño físico que haga que el módulo a instalar sea robusto y fiable.

El diseño se ha llevado a cabo en dos placas diferentes pero complementarias, que se conectan entre si mediante headers, de forma que en una placa se han implementado las diferentes fuentes de alimentación para proporcionar todos los voltajes y corrientes necesarios para su correcto funcionamiento.

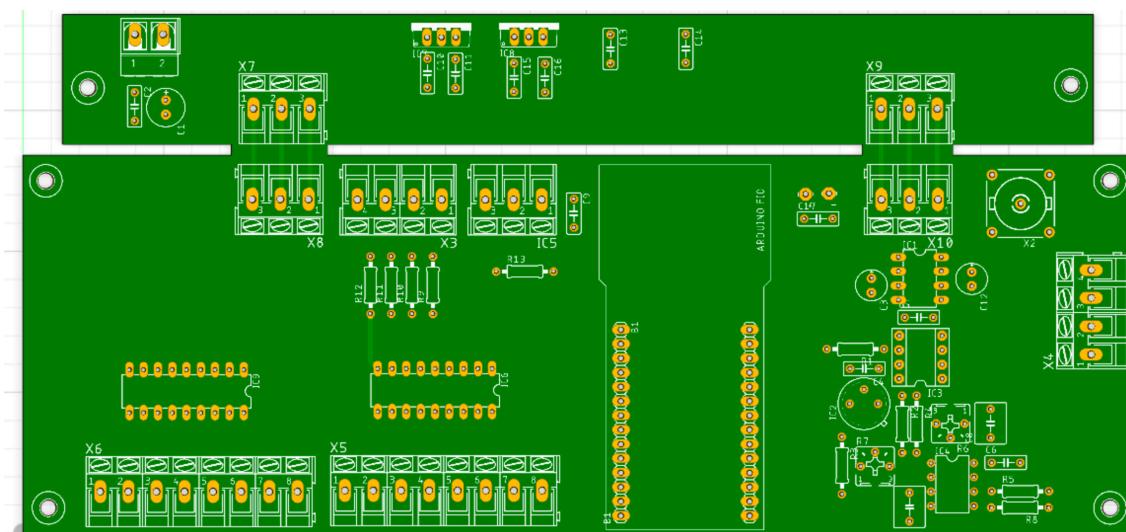


Figura 44. PCB cara “top” completa.



El diseño se ha llevado a cabo utilizando el Software Eagle de Autodesk, y el resultado se puede observar en la Figura 44. La PCB se ha diseñado en una sola cara para abaratar costes de fabricación, si bien es que contiene algún puente entre pistas, también se ha modificado el diseño para que sea posible su fabricación utilizando ambas caras para poder prescindir de éstos.

3.1.1 Placa de alimentación.

Para brindar mayor flexibilidad al proyecto, la placa donde se consiguen todas las tensiones de alimentación ha quedado separada del la placa de adquisición.

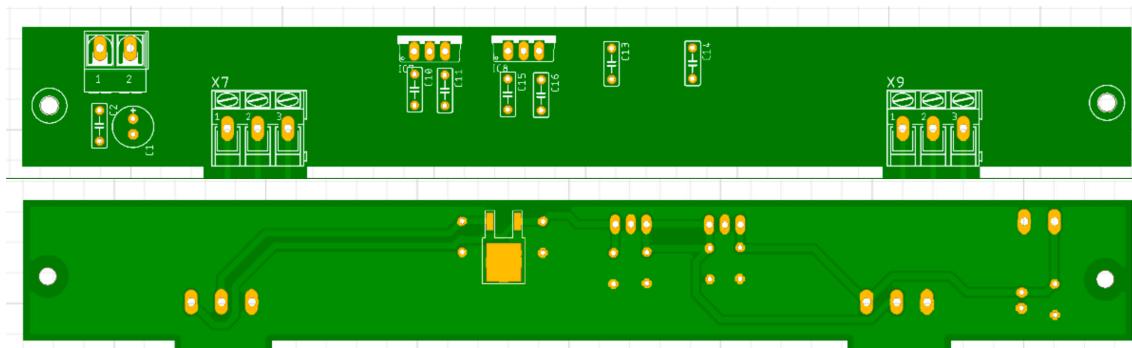


Figura 45. PCB de alimentación (top/bottom).

La PCB de alimentación cuenta únicamente con los componentes necesarios para servir de alimentación, tal y como se muestra en la Figura 45. Para ello se ha decidido utilizar el máximo de componentes de agujero pasante (PTH), aunque el regulador A78M33 se implementa en tecnología SMD.

La conexión de la alimentación principal de entrada es de 24 V DC y se conecta a través de un convertidor AC-DC como el que se muestra en la Figura 46.



Figura 46. Convertidor AC DC.

De ese modo, es necesario conectar los 230 V AC a la red y los 24 V DC a la entrada de la PCB. Tanto las entradas como la salida a la PCB de adquisición se pueden realizar directamente soldadas o bien mediante conectores tipo PTR como el de la Figura 47.



Figura 47. Conector PTR

La placa de alimentación es la más propensa a generar calor, ya que los reguladores lineales que incorpora generan calor al hacer su función, es por ello que los dos reguladores lineales de encapsulado TO-220 se han colocado a un extremo de la placa, de forma que, en caso de que un disipador sea necesario, se puede acoplar un único disipador para ambos encapsulados.

También se han incorporado dos agujeros a cada uno de los extremos para anclar la PCB a la caja o chasis sobre el que se monte, de forma que quede bien sujetada y evite cualquier tipo de cortocircuito. De este modo, como se observa en la Figura 45, se ha diseñado una placa con un diseño que permite disipar el calor de una forma correcta, capaz de proporcionar todas las tensiones de alimentación necesarias para un correcto funcionamiento y además que se puede implementar en un chasis o caja diferente a donde se encuentre la placa de adquisición que a continuación se detalla.

3.1.2 Placa de adquisición.

Si bien la placa de alimentación tenía un diseño sencillo, a la hora de diseñar la PCB de adquisición se han tenido en cuenta más factores. En primer lugar, se ha intentado distribuir los conectores de tal manera que favorezca una correcta colocación de los componentes y también de forma que favorezca el enrutado de las pistas. En la Figura 48 se muestra la cara top de la PCB de adquisición.

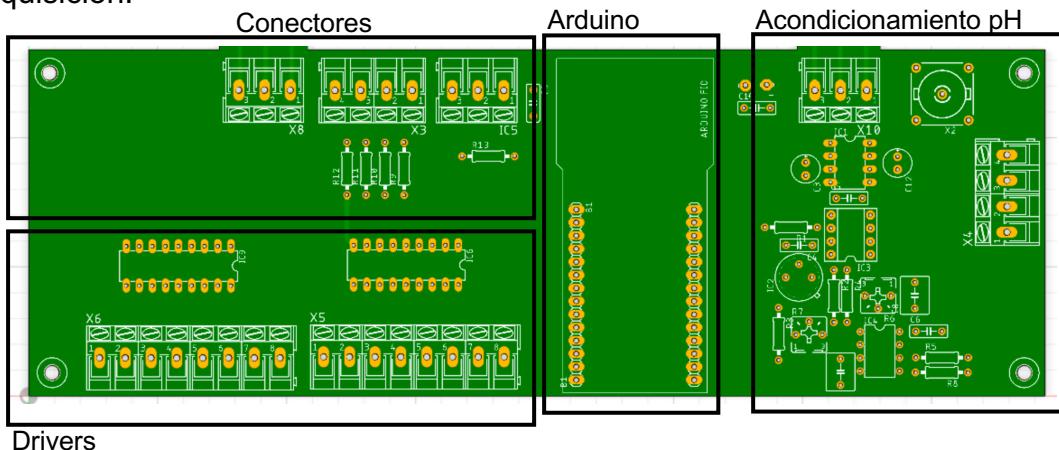


Figura 48. Cara “top” PCB de adquisición.

En la Figura 48 se puede observar que los conectores PTR de alimentación están situados de tal manera que queden enfrentados a los conectores de la placa de alimentación, para de esta forma, con un cable lo más corto posible poder conectarlos.



Los conectores X3 y X4 corresponden a la botonera. De ellos se tirará un cable hasta donde se vaya a instalar la botonera de forma física. Al igual que todos las entradas y salidas, se pueden colocar conectores o bien se puede soldar directamente el cable correspondiente.

El sensor DHT22 se puede colocar en la posición que se desee y luego cablear los tres pines hasta los conectores pertinentes, de esta manera, se evita que el sensor esté necesariamente conectado a la PCB, lo cual sería contraproducente, ya que la PCB de adquisición no debe de estar necesariamente en el mismo lugar donde se desee tener la referencia de temperatura y humedad relativa.

En cuanto a la medición de pH, el electrodo sensor entra a la placa mediante un conector BNC que se ha situado también en uno de los vértices de la PCB. Se puede observar en el vértice superior derecho de la PCB de la Figura 48.

Se ha buscado un diseño modular, que sea ajustable pero a su vez que sea robusto, es por ello que en muchos de los componentes se instalan en la PCB mediante zócalos, de forma que no es necesario que los integrados o el Arduino estén completamente soldados a la PCB, ya que si en un futuro se desea reprogramar alguna de sus funciones complicaría mucho esta labor, así que se ha decidido instalar zócalos a los componentes mas propensos a fallar o que necesitan cierto mantenimiento. En la parte central de la Figura 48 se muestra el “footprint” del Arduino Fio donde irían soldadas las tiras de pines que hacen de zócalo para insertar el dispositivo.

El circuito acondicionador de la medida de pH se encuentra en la mitad derecha de la PCB, este circuito contiene trimmers que servirán para el correcto ajuste del módulo, de forma que, deben de estar relativamente accesibles, por lo que habría que crear un acceso a través del chasis para ajustarlos, o bien ajustarlos con la tapa del chasis quitada.

En la parte izquierda de la PCB se encuentra el circuito de potencia, en el cual se han instalado los dos integrados correspondientes a los drivers para suministrar la energía necesario para el disparo de las electroválvulas.

Como se ha mostrado con anterioridad, se han diseñado dos salidas diferentes para las electroválvulas, de forma que en cada uno de los conectores se encuentran las misma señales pero a diferente nivel de tensión.



3.2. Puesta en marcha (simulación “física”)

3.2.1 Desarrollo de prototipos.

Con el fin de llevar a cabo una simulación lo más cercana posible al diseño y funcionamiento final, se han desarrollado dos prototipos para poder probar el diseño tanto de hardware como de software. Debido a la situación extraordinaria provocada por la pandemia, ha habido subcircuitos del esquema que no se han podido simular físicamente, dado que el acceso a diversos componentes así como aparatos de medición ha sido limitado.

El primero de los prototipos se ha implementado en protoboard, como se puede observar en la Figura 49. El desarrollo de este diseño se ha realizado juntamente con la programación en software, de forma que el diseño ha ido constantemente modificándose. Es decir, con las limitaciones existentes, en la medida de lo posible, se ha intentado realizar el trabajo que se hubiese tenido que realizar en el laboratorio.

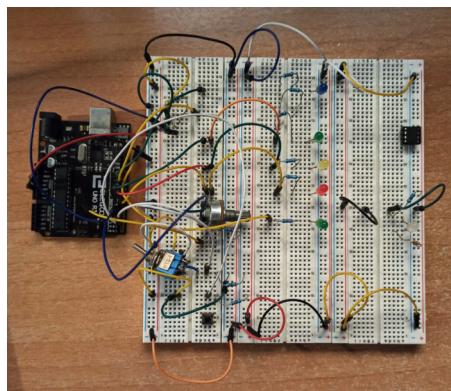


Figura 49. Prototipo en protoboard.

Debido a las limitaciones comentadas, los componentes usados se han reutilizado de otros proyectos o en base a la disponibilidad del laboratorio donde se ha implementado el diseño, es por ello que muchos de los componentes están sobredimensionados o se han utilizado distintos fabricantes para un mismo componente.

Una vez finalizado el proceso de diseño, se procedió al diseño de un placa PCB de prototipo. Esta placa de prototipo se diseño con el software Kicad, el diseño implementado se ha realizado en una cara, siempre teniendo en cuenta de que su proceso de fabricación estará limitado y se debe poder fabricar con materiales y máquinas de fácil obtención. En la Figura 50 se puede observar el resultado simulado de la placa en el software Kicad.

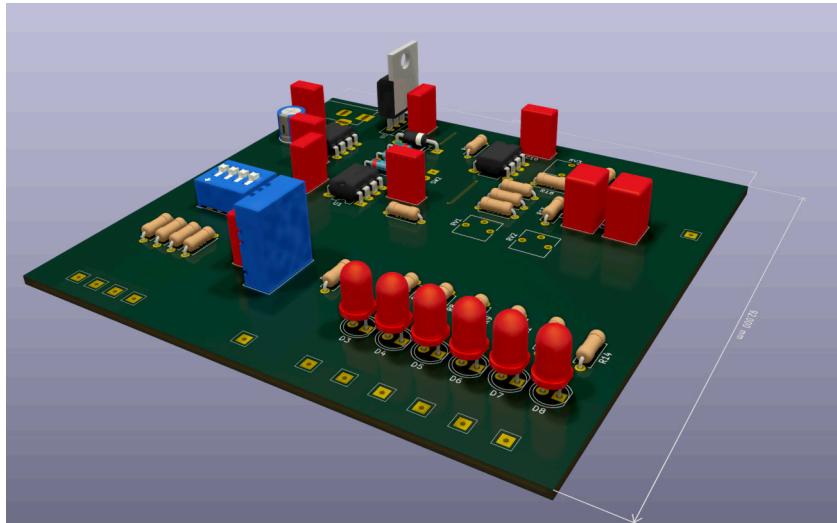


Figura 50. Placa PCB prototipo.

En cuanto a la simulación de los pulsadores físicos se han utilizado interruptores SPST, estos pulsadores están conectados a una tensión de 5 V en uno de sus polos mientras que el otro polo se conecta al microcontrolador. De este modo una vez se accione alguno de los interruptores, al microcontrolador llegará un estado alto en la entrada física correspondiente.

En cuanto al circuito de acondicionamiento de pH, se ha realizado un diseño de esquemático lo mas cercano al diseño teórico pero adaptándolo a la disponibilidad de componentes. Por lo que la funcionalidad es la misma pero la tensión de entrada se controla mediante un potenciómetro en lugar de un electrodo de pH.

El sensor DHT 22 se ha conectado mediante un socket a la PCB y luego desde un pin de esta se conecta la salida del sensor a Arduino.

Por último, la simulación de los diferentes accionadores con los que cuenta el kit se han utilizado leds, de forma que el subcircuito de potencia de los drivers se omite en este prototipo. Esto se debe a que la alimentación de esta placa se va a realizar mediante una pila de 9 V, lo cual dificulta la obtención de las tensiones de alimentación de los integrados que actúan como drivers. Estos leds se iluminan con una tensión de 5 V proporcionada por Arduino, de forma que cuando el microcontrolador pone alguna de las salidas a 1, se encenderá el led correspondiente.

El esquema de este prototipo se puede consultar en el anexo 4.

El proceso de fabricación se ha elaborado con el método de transferencia de toner. En las figuras 51, 52, 53 se muestran imágenes del proceso de fabricación de esta placa así como su resultado final.

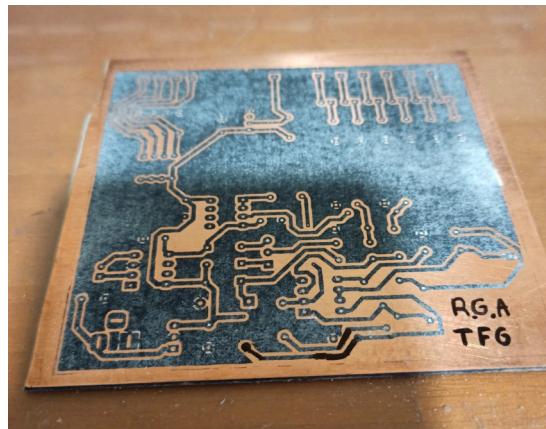


Figura 51. Transferencia tóner a placa de cobre.

En primer lugar, se transfiere el tóner por el método del planchado a la placa de cobre, se ha de eliminar el exceso de papel que quede pegado al cobre para que únicamente quede adherida la tinta al cobre, como se muestra en la Figura 51.

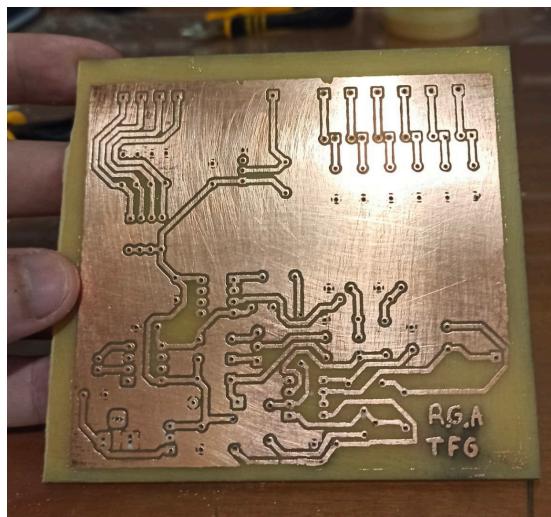


Figura 52. Pistas de placa PCB prototipo.

Una vez está el tóner adherido y preparado se introduce la PCB en ácido, en este caso el ácido se ha tenido que elaborar con elementos fáciles de encontrar. En este caso la proporción ha sido agua, agua oxigenada y sal fuman a partes iguales. Una vez introducida la placa en el ácido y eliminado el exceso de cobre, la placa tiene el aspecto que se muestra en la Figura 52.

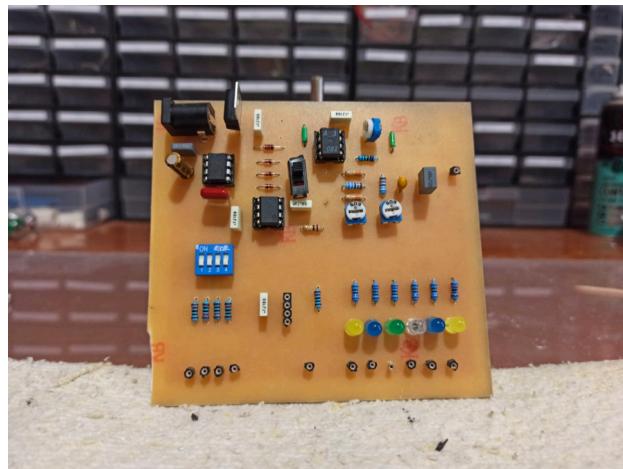


Figura 53. Placa PCB prototipo ensamblada.

Por último se han soldado todos los componentes necesarios para el montaje tal y como se puede ver en la Figura 53.

3.2.2 Simulación de la respuesta del sensor de pH (hard/software)

Como se ha indicado en el capítulo anterior, el electrodo de pH ofrece un rango de tensión que el circuito acondicionador debe convertir en un rango legible de tensión para el microcontrolador. Éste se encarga de enviar la información al PC para que pueda ser visualizada por el usuario mediante la interfaz desarrollada con el software LabView.

En primer lugar, se ha realizado una simulación software para confirmar que los escogidos para cada componente en el diseño teórico son adecuados. El circuito se puede observar en la Figura 54.

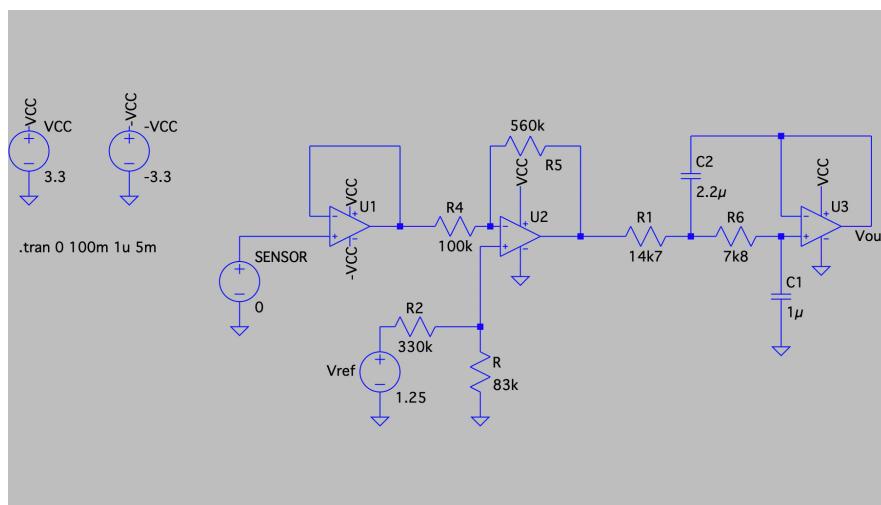


Figura 54. Circuito simulación acondicionamiento sensor de pH.



El objetivo de esta simulación fue comprobar la respuesta del circuito de acondicionamiento. Se simuló la respuesta de tensión del sensor para comprobar que el rango de respuesta del circuito diseñado teóricamente coincidía con lo esperado.

Por tanto, el diseño teórico se realizó para que el circuito mostrase una tensión de $V_{pH} = [0, 3.3]$ V, de forma que para el valor mas negativo de V_{sensor} , V_{pH} debía ser lo mas cercano a 0 V, mientras que para el valor mas positivo de V_{sensor} , V_{pH} deberá de ser lo mas cercano a 3.3 V. Siendo V_{sensor} la tensión proporcionada por en sensor y V_{pH} la tensión a la salida del circuito de acondicionamiento.

Ya que la respuesta del sensor tiene un rango $V_{sensor} = [-0.4, 0.4]$ se simuló un rango de valores de $V_{sensor} = [-0.236, 0.236]$. Este rango de valores corresponde a un rango de pH de 3-11 a 25 °C. También se buscó evitar trabajar en extremos del rango medible, ya que es donde mas dificultades se pueden presentar en el circuito de acondicionamiento.

En la Tabla 2 se muestran los resultados obtenidos en la simulación con el software LTSpice.

V_{sensor} (V)	V_{pH} (V)	pH correspondiente a 25°C
-0.236	3.267	11
-0.118	2.46	9
0	1.66	7
0.118	0.86	5
0.236	0.06	11

Tabla 2. Simulación software pH.

De este modo se ha determinado que el circuito de acondicionamiento de pH tiene un funcionamiento teórico que coincide con el esperado por los diseños teóricos previamente realizados.

Por otro lado, se realizó la simulación física. La simulación física de la respuesta del sensor se ha realizado a partir de la PCB prototipo, para comprobar el correcto funcionamiento del circuito, se midió la tensión a la salida del circuito de acondicionamiento. En la Figura 55 se observa el pin de salida que proporciona la tensión medida.

La simulación de la tensión proporcionada por el electrodo de pH se realizó mediante trimmers, mientras que mediante un switch incorporado en la PCB se fijó tensión negativa o positiva, simulando el rango de voltaje que puede llegar a ofrecer el electrodo de pH.



Figura 55. Placa PCB prototipo.

El objetivo de esta prueba fué determinar que el circuito implementado en la PCB, correspondiente al circuito de acondicionamiento de pH, era capaz de proporcionar un rango de voltajes que coincidiera con el rango legible del microcontrolador. Una vez realizadas las mediciones pertinentes, se determinó que el circuito implementado era capaz de ofrecer un rango de tensiones tal que $V_{pH} = [1.2, 4.3]$ V. Este rango de tensión correspondería a una variación de pH aproximada de 6-8, dependiendo de la temperatura y de la programación del microcontrolador.

Pese a que los resultados obtenidos no han sido los ideales, se puede utilizar el circuito para generar una tensión variable que es medible por el microcontrolador. Es por este motivo y por la dificultad de fabricar nuevos prototipos con componentes de mayor calidad o mas difíciles de encontrar lo que se decidió continuar con el proceso de simulación pese a no obtener un rango de valores de $V_{pH} = [0, 5]$ V.

3.2.3 Simulación del módulo.

Con objeto de validar el funcionamiento, utilizando la PCB descrita en el apartado anterior y la placa Arduino Uno directamente conectada al PC, se realizaron las comprobaciones necesarias para verificar el funcionamiento de la interfaz de monitorización y control desarrollada.

En primer lugar se comprobó el funcionamiento de los botones físicos que se instalarán en el vivero, siendo éstos emulados mediante los cuatro interruptores SPST descritos en el apartado anterior. Para comprobar el correcto funcionamiento, se activaron los interruptores correspondientes a la apertura de ventanas, al calefactor y al aspersor.



Figura 56. Comprobación software botones físicos.

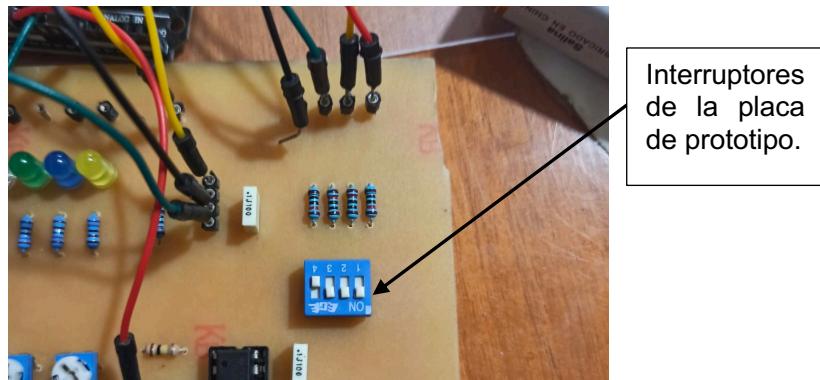


Figura 57. Activación botones físicos.

Tal y como se muestra en la Figura 55 y Figura 56, una vez activados los interruptores 1, 2 y 3 que equivalen a los botones que se encontrarían dentro del vivero, se puede observar como la interfaz enciende los pilotos correspondientes a los accionadores de apertura de ventanas, el calefactor y el aspersor. Realizando esta operación se comprobó el correcto funcionamiento de la programación del microcontrolador y del LabView referente a la lectura de datos.

Una vez comprobada la comunicación a través de los botones físicos alojados en la placa del prototipo, se comprobó que los botones de la interfaz de usuario tuvieran el mismo correcto funcionamiento que los botones físicos. De esta forma se garantiza un control manual del vivero tanto dentro del vivero como desde el centro de control donde esté alojado el PC.

Para realizar esta comprobación, se activaron los diferentes interruptores que la interfaz de usuario alberga. El primer paso para comprobar que cada una de las salidas del microcontrolador se encuentra en estado alto, es comprobar la cadena de datos que LabView está enviando a Arduino.

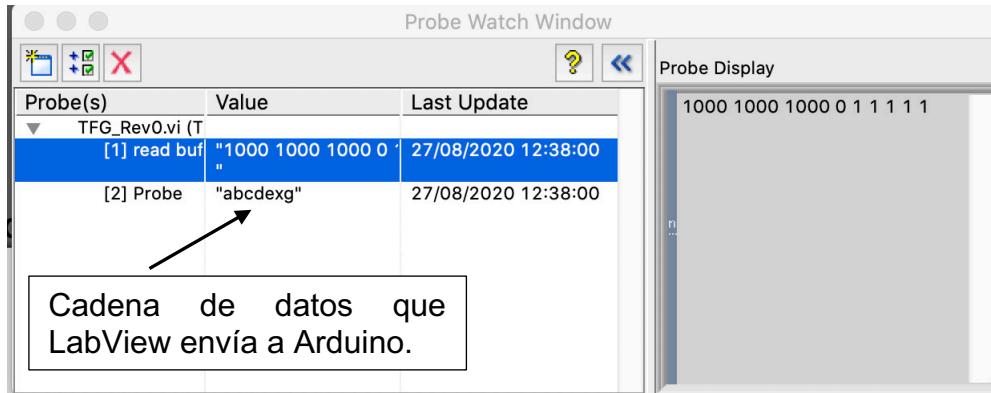


Figura 58. Ventana Probe LabView.

Como se observa en la Figura 58, al activar los diferentes interruptores de la interfaz de usuario, se envía una cierta cadena de datos por puerto serie, esta cadena de datos corresponde a “abcdexg”.

Como se ha descrito en el apartado de programación de LabView, cada vez que un interruptor de la interfaz, LabView envía un carácter diferente de “x”, es decir, al tener todos los interruptores apagados LabView envía por puerto serie “xxxxxx” en los siete primeros bits.

De ese modo al activar algún interruptor, el carácter “x” se sustituye por el que corresponda. En este caso, se activaron 6 interruptores diferentes, cinco de ellos correspondientes a accionadores (“abcde”) y el interruptor que corresponde al carácter “g” es el interruptor para indicarle a Arduino que el control se está haciendo de forma manual. Por tanto, de los primeros siete bits, seis de ellos contienen un carácter diferente a “x”.

Una vez comprobada la comunicación de LabView con Arduino, también se ha de comprobar que Arduino interpreta bien dicha información. Para ello, se conectaron las salidas del microcontrolador correspondientes a los actuadores a la placa de prototipo. Como se muestra en la Figura 59, cinco de los seis leds que forman la simulación de cada uno de los accionadores se iluminaron. Esto indicaría que el código de Arduino está interpretando correctamente la información recibida desde LabView.

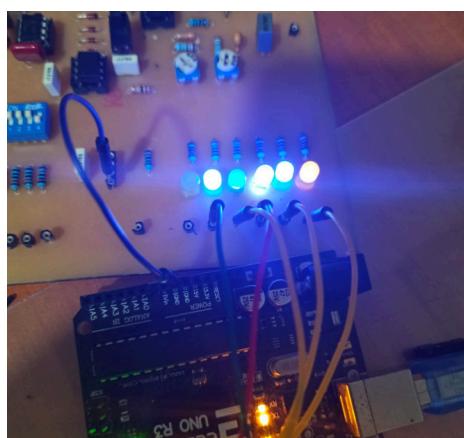


Figura 59. Leds placa PCB prototipo.



Previamente a comprobar el funcionamiento de cada uno de los sensores, se comprobó que el envío de los parámetros objetivo se realizara de forma adecuada para que el microcontrolador pudiera interpretarlo de forma óptima.

Para realizar estas comprobaciones se introdujeron los datos en LabView mediante la interfaz de introducción de parámetros, estos quedaban registrados y se enviaban por la cadena de datos pertinente hacia el microcontrolador. Como se puede observar en la Figura 60, el resultado fue positivo y la transferencia de datos se realizaba de forma correcta.

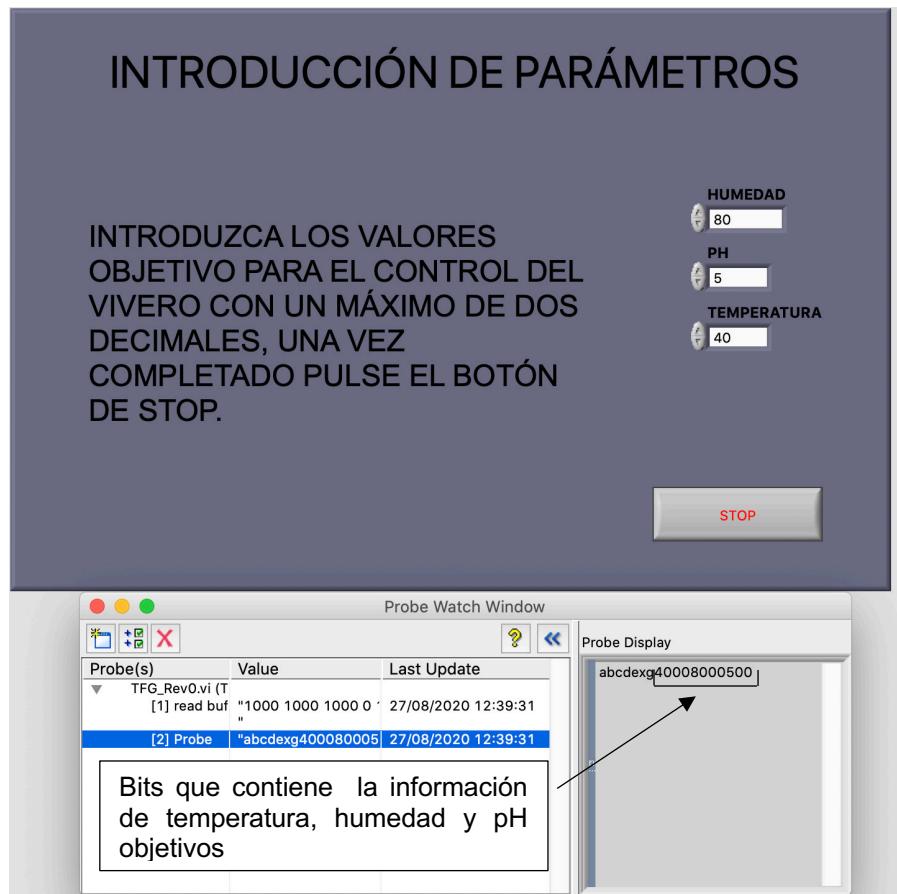


Figura 60. Comprobación de envío parámetros objetivo.

Una vez comprobado que la introducción y envío de los parámetros objetivo se realizaba tal y como se esperaba, se procedió a comprobar que todos los sensores funcionen de manera correcta.

En primer lugar se realizó la simulación del sensor de pH, en el cual, sabiendo que ya se ofrece un nivel de tensión medible por el microcontrolador, quedaba comprobar que éste era capaz de procesar dicha información y transferirla a LabView de forma correcta. Si todo funciona correctamente se debería de visualizar un valor de pH que se corresponde con el valor de tensión que se está proporcionando en la PCB prototipo (V_{pH}).



Para llevar a cabo el proceso de simulación, se fijó una temperatura de 40 °C, como se ha visto en capítulos anteriores, la temperatura ofrecerá la variable correspondiente para calcular el pH con la deriva térmica.

En la Figura 61 se muestra como se realizó el conexionado desde el prototipo a la placa de Arduino para, de este modo, proceder con la simulación.

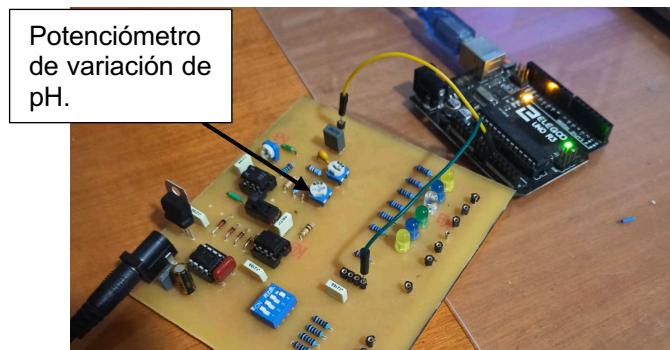


Figura 61. Conexión físico para la medición de pH.

Una vez conectado el pin de salida del circuito de acondicionamiento a la entrada analógica de Arduino A0, se comprobó que en la interfaz de usuario se estaba leyendo un determinado valor de pH.

En este caso, no resultó fácil entregar los diferentes valores de tensión dentro del rango que, como se ha mencionado con anterioridad, puede entregar la placa de prototipo. Aún así se pudo comprobar que para un determinado nivel de tensión, el microcontrolador era capaz de interpretar dicho valor y traducirlo a un valor legible por LabView, tal y como se muestra en la Figura 62.

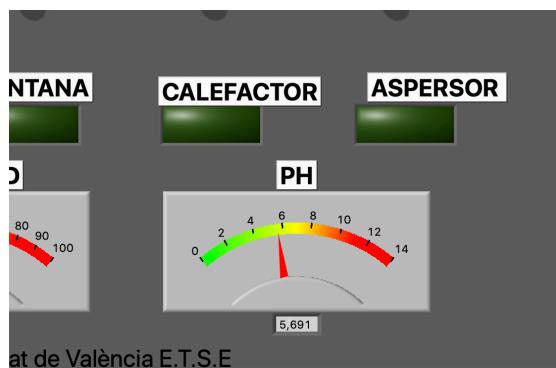


Figura 62. Simulación medición de pH.

Pese a las diferentes dificultades que se han presentado en la simulación de la respuesta de pH, los resultados fueron positivos. Es decir, el microcontrolador era capaz de interpretar la información que llegaba de la PCB prototipo. Es por eso por lo que se ha considerado que el funcionamiento a nivel de programación tanto de LabView como Arduino era correcto.

Por último, para terminar con la simulación, se comprobó el correcto funcionamiento del sensor DHT 22. Dicho sensor se encarga de proporcionar la



lectura tanto de temperatura como de humedad del ambiente, tal y como se ha mostrado en apartados anteriores.

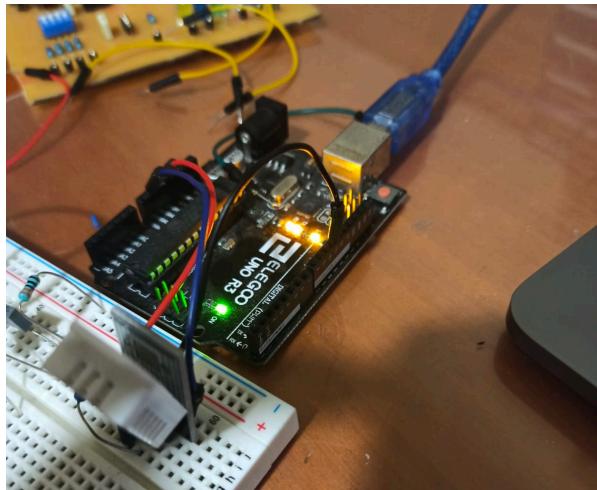


Figura 63. Conexionado sensor DHT22.

Como se ha descrito con anterioridad, el sensor DHT22 recoge los datos por un único pin, enviando la información en forma de pulsos, por lo que mediante las librerías cargadas en la programación del microcontrolador, la lectura se traduce de forma inmediata. En la Figura 63 se puede ver el conexionado que se ha llevado a cabo para conectar el sensor DHT 22.

Estas conexiones se realizaron en una placa protoboard siguiendo el mismo esquema que en la PCB prototipo. El motivo de no introducir el sensor en la placa prototipo es que no encajaba el zócalo soldado con los pines de dicho sensor. Es por ello que se introdujo en una protoboard y posteriormente se realizó la conexión desde el protoboard a la placa de prototipo.

En los apartados referentes a la programación tanto de LabView como de Arduino, se ha descrito como los datos una vez interpretados por los entornos de programación, se envían sin decimales para que no se pierda ningún dato.

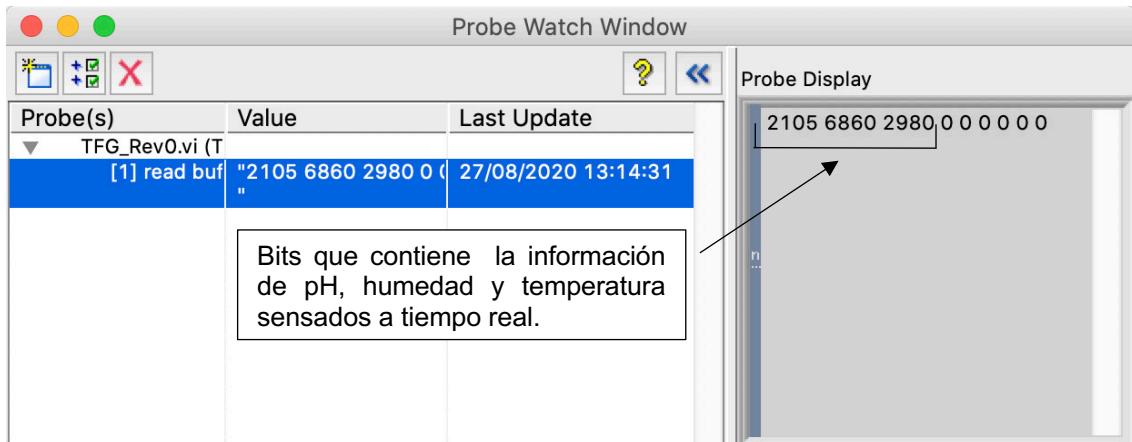


Figura 64. Cadena de datos con información de temperatura y humedad.



Una vez realizada la puesta en marcha del sistema con el sensor en funcionamiento, se comprobó que, como se muestra en la Figura 64, la información proporcionada por los sensores y convertida por Arduino, se transmitía de forma correcta al software LabView. De este modo, era de esperar que la interpretación de datos por parte de LabView se realizara de forma correcta tanto para los visores analógicos como para los visores numéricos, tal y como ocurría con el pH.

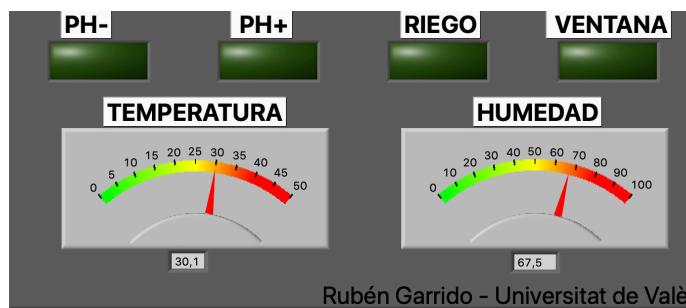


Figura 65. Lectura de temperatura y humedad.

En la Figura 65 se puede observar como el visionado se realizaba de forma correcta. Los valores de la Figura 64 y la Figura 65 no coinciden en exactitud ya que el sensado se realiza a tiempo real y se actualiza a cada segundo. Es por ello que en la diferencia de tiempo entre que se capturaron ambas imágenes hay un ligero desajuste en la temperatura y humedad relativa.



Capítulo 4. Conclusión y futuras mejoras.

Como conclusión, se ha obtenido un sistema cuya puesta en marcha ha resultado cumplir con los objetivos principales. Ciento es que no se ha podido implementar al completo en un entorno industrial más allá de las pruebas de funcionamiento que se han llevado a cabo en el laboratorio de pruebas.

Debido a la situación excepcional provocada por la pandemia se ha tenido que reestructurar el proceso de puesta en marcha adaptándose a las diferentes limitaciones. Los resultados obtenidos durante las simulaciones tanto de hardware como de software con el prototipo desarrollado han cumplido con los objetivos de tener un módulo de bajo coste y adaptable a cualquier tipo de instalación, haciendo posible la automatización de un vivero a bajo coste.

También la relación entre hardware y software ha cumplido las expectativas de funcionamiento modular, si se tiene en cuenta que, para un mismo hardware, se puede programar el software, de forma que, permita la adaptación a las necesidades del vivero donde se implante. Por tanto, por las razones mencionadas, se han cumplido los objetivos relacionados con la implantación de un sistema modular y versátil.

El desarrollo de los prototipos se ha realizado en un entorno completamente distinto al que se hubiera desarrollado en el laboratorio de la universidad, se ha tenido que realizar un esfuerzo extra para la fabricación de placas de circuito impreso con la finalidad de comprobar experimentalmente el funcionamiento del módulo diseñado.

En definitiva, para el diseño software no se ha tenido limitación en cuanto al desarrollo. Esto ha conllevado a que la implementación de esta parte sea la correcta. En cambio en el apartado de hardware si se ha tenido limitación en el desarrollo, lo cual ha interferido en el proceso de puesta en marcha y con ello en el proceso de simulación.

Como futuras mejoras, se puede incluir la mejora de la implementación a nivel de hardware. La implementación de un prototipo mas completo daría pie a mejorar el diseño tanto a nivel de hardware como a la hora de implementar alguna mejora en el software.



Bibliografía.

- [1] *Basics of Interfacing DHT11/DHT22 Humidity and Temperature Sensor with MCU.* OCFreaks!. (2020). Retrieved 28 August 2020, from <http://www.ocfreaks.com/basics-interfacing-dht11-dht22-humidity-temperature-sensor-mcu/>.
- [2] *Entradas y salidas digitales y analógicas.* Dfists.ua.es. (2020). Retrieved 28 August 2020, from http://dfists.ua.es/~jpomares/arduino/page_10.htm.
- [3] *Funcionamiento, cuidado y calibrado del medidor de pH.* Medidordeph.com. (2020). Retrieved 28 August 2020, from <http://medidordeph.com/funcionamiento-cuidado-calibrado-medidor-de-ph>.
- [4] Ruiz Gutiérrez, J. (2020). Implementación de Sistemas de Trasmisión de Datos y Sensores en Redes Inalámbricas con XBee integrado en la "Plataforma Open Hardware" Arduino [Ebook] (1st ed.). Retrieved 28 August 2020, from <https://www.docsity.com/es/programacion-arduino-xbee/5592806/>.
- [5] *TUTORIAL ARDUINO + LABVIEW.* Naylampmechatronics.com. (2020). Retrieved 4 September 2020, from https://naylampmechatronics.com/blog/23_TUTORIAL-ARDUINO-Y-LABVIEW.html.
- [6] Thayer Ojeda, L. (2010). *XBee Guia de Usuario* (2nd ed.). Eduard Martin.
- [7] Faludi, R. (2011). *Building Wireless Sensor Networks* (1st ed.). Brian Jepson.
- [8] *Using EAGLE: Board Layout - learn.sparkfun.com.* Learn.sparkfun.com. (2020). Retrieved 4 September 2020, from <https://learn.sparkfun.com/tutorials/using-eagle-board-layout/all>.
- [9] *Kicad: Techniques, Tips and Work-arounds – Inductive-Kickback.com.* Inductive-kickback.com. (2020). Retrieved 4 September 2020, from <https://inductive-kickback.com/2016/10/kicad-techniques-tips-and-work-arounds/>.



Anexos

ANEXO 1. CÓDIGO PROGRAMACIÓN ARDUNIO

```
//Añadimos librerias necesarias para DHT22 #include <DHT.h>
#define DHTTYPE DHT22
const int DHTPin = 5;

//Declaracion de variables

char data_labview; //Datos recibidos de Labview
int func_manual; //Cuando la variable está a 1 los accionadores solo funcionan
de forma manual
String datoslabview; //Data_labview agrupada en variable tipo string
String temperatura; //Para poder leer los datos del string y pasarlo a String
humedad; //las variables int, pero para ello tengo que utilizarToInt()
String PH;

float hum; float temp;

int temp_INT;//Utilizare estas variables de tipo entero para almacenar los
valores "meta"
int hum_INT;
int PH_INT;

//Entradas digitales
int b_aspersor=2; //Aspersor
int b_ventanas=3; //Abrir ventanas
int b_calefactor=4; //Enchufar calefactor

DHT dht(DHTPin, DHTTYPE);//inicializamos el sensor DHT22

//Salidas digitales
int s_ventanas=6; //Salida para las ventanas
int s_calefactor=7; //Salida para el calefactor int s_phpos=8; //Salida para
electrovalvula de PH+ int s_phneg=9; //Salida para electrovalvula de PH- int
s_aspersor=10; //Salida para el aspersor
int s_riego=11; //Salida para el riego

//Salidas PWM DE MOMENTO NO NECESITO NINGUNA

//Lecturas analogicas int PH_medido;
float PH_convertido; float PH_volt;

int PH_LV; //Lecturas digitales int boton_aspersor=0;

int boton_ventana=0; int boton_calefactor=0;
```



```
//Estado Salidas bool ventana=0; bool calefactor=0; bool phpos=0; bool
phneg=0; bool aspersor=0; bool riego=0;

void setup() {

Serial.begin(9600);
dht.begin();//empezamos a leer por el sensor pinMode(A0,INPUT);
pinMode(b_aspersor,INPUT); pinMode(b_ventanas,INPUT);
pinMode(b_calefactor,INPUT); pinMode(s_ventanas,OUTPUT);
pinMode(s_calefactor,OUTPUT); pinMode(s_phpos,OUTPUT);
pinMode(s_phneg,OUTPUT); pinMode(s_aspersor,OUTPUT);

}

void loop() {

PH_medido = analogRead(A0); //Leemos el PH de 0 a 1023 PH_volt =
map(PH_medido, 0, 1023, 0, 5);
float m_ph = (0.3333 + (0.0012201*temp))*(-1);
float b_ph = 2.5 - (m_ph * 7);

PH_convertido = (PH_volt - b_ph) / m_ph;
if (PH_convertido < 10.0) {PH_LV=PH_convertido*1000;} else
{PH_LV=PH_convertido*100;}

hum = dht.readHumidity();// Leemos la humedad relativa

temp = dht.readTemperature();// Leemos la temperatura en grados centígrados
(por defecto)

int hume= 100*hum;

int tempe= 100*temp; if (Serial.available()) {

data_labview=Serial.read(); //Los datos que envia Labview por puerto serie se
leen

datoslabview += data_labview; // Transformamos la "palabra" que envia
LabView en un string

if (datoslabview[6]=='g'){func_manual=1;} //Leemos de LabView si estamos
funcionando de forma manual

if(data_labview=='\n') {

if (datoslabview[0]=='a') {digitalWrite(s_ventanas,HIGH);ventana=1; }//Operatura
manuela de ventanas desde labview
}
}
```



```

else if ((hum > hum_INT+ 1000) && (func_manual=0))
{digitalWrite(s_ventanas,HIGH);ventana=1;} //Si la humedad es alta se abren
las ventanas para ventilar

else if (digitalRead(b_ventanas)==HIGH) {digitalWrite(s_ventanas,
HIGH);ventana=1;}//Si presionamos el boton fisico de abrir ventana

else {digitalWrite(s_ventanas,LOW);ventana=0;}

if (datoslabview[1]=='b') {digitalWrite(s_calefactor,HIGH);calefactor=1;
}//Enciendo el calefactor de forma manual

else if ((temp < temp_INT-300) && (func_manual=0))
{digitalWrite(s_calefactor,HIGH);calefactor=1;}//El calefactor se enciende si la
temperatura baja mas de 3 grados de la deseada

else if (digitalRead(b_calefactor)==HIGH) {digitalWrite(s_calefactor,
HIGH);calefactor=1;}

else {digitalWrite(s_calefactor,LOW);calefactor=0;}

if (datoslabview[2]=='c'){digitalWrite(s_phpos,HIGH);phpos=1;}//Operatura de
valvula para subir PH manual

else if ((PH_LV > PH_INT) && (func_manual=0)){digitalWrite(s_phpos,
HIGH);phpos=1;}//La valvula de nivelado se abre para nivelar el PH

else {digitalWrite(s_phpos,LOW);phpos=0;}

if (datoslabview[3]=='d'){digitalWrite(s_phneg,HIGH);phneg=1;}//Operatura de
valvula para bajar PH manual

else if ((PH_LV > PH_INT) && (func_manual=0)){digitalWrite(s_phneg,
HIGH);phneg=1;}//La valvula de nivelado se abre para nivelar el PH

else {digitalWrite(s_phneg,LOW);phneg=0;}

if (datoslabview[4]=='e'){digitalWrite(s_aspersor,HIGH);aspersor=1; }//Salida
aspersor manual

else if ((hum < hum_INT - 1000) &&
(func_manual=0)){digitalWrite(s_aspersor,HIGH);aspersor=1;}//Si la humedad
es muy baja un aspersor se activa

else if ( digitalRead(b_aspersor)==HIGH ){digitalWrite(s_aspersor,
HIGH);aspersor=1; }

else {digitalWrite(s_aspersor,LOW);aspersor=0;}

```



for (int i=8;i<12;i++)//Aqui cargo los datos que he pasado del PS del LabView al arduino y los cargo en las variables correspondientes

```
{temperatura += datoslabview[i];//siempre va a ocupar 3 bits
humedad += datoslabview[i+3];//siempre va a ocupar 3 bits
PH += datoslabview[i+6];//siempre va a ocupar 3 bits
temp_INT=temperatura.toInt();//Aqui lo que hago es pasar el string a un
entero hum_INT=humedad.toInt();
PH_INT=PH.toInt();

temperatura="";//Reseteamos las string para que a la siguiente ejecucion del
programa este vacia

humedad=""; PH="";
datoslabview="";//Se vacia el buffer //data_labview="";
Serial.print(PH_LV);//Valor PH Serial.print("\t"); Serial.print(hume);//Valor
temperatura Serial.print("\t"); Serial.print(tempe);//Valor humedad Serial.print("\t");
Serial.print(ventana);//Estado Ventanas Serial.print("\t");
Serial.print(calefactor);//Estado Calefactor Serial.print("\t");

Serial.print(phpos);// Estado Valvula PH+ Serial.print("\t");
Serial.print(phneg);// Estado Valvula PH- Serial.print("\t");
Serial.println(aspersor);// Estado aspersor delay(50);

} }

else{
if (digitalRead(b_ventanas)==HIGH) {digitalWrite(s_ventanas,HIGH);} else if
(hum > hum_INT+ 1000) {digitalWrite(s_ventanas,HIGH);} else
{digitalWrite(s_ventanas,LOW);}

if (temp < temp_INT-300) {digitalWrite(s_calefactor,HIGH);}

else if (digitalRead(b_calefactor)==HIGH) {digitalWrite(s_calefactor, HIGH);}

else {digitalWrite(s_calefactor,LOW);}

if (PH_LV > PH_INT){digitalWrite(s_phpos,HIGH);}//La valvula de nivelado se
abre para nivelar el PH

else {digitalWrite(s_phpos,LOW);}

if (PH_LV > PH_INT){digitalWrite(s_phneg,HIGH);}//La valvula de nivelado se
abre para nivelar el PH
```



```
else {digitalWrite(s_phneg,LOW);}

if (hum < hum_INT - 1000){digitalWrite(s_aspersor,HIGH);}//Si la humedad es
muy baja un aspersor se activa

else if ( digitalRead(b_aspersor)==HIGH ){digitalWrite(s_aspersor,HIGH);} else
{digitalWrite(s_aspersor,LOW);}

}
```



ANEXO 2. PROGRAMACIÓN EN LABVIEW

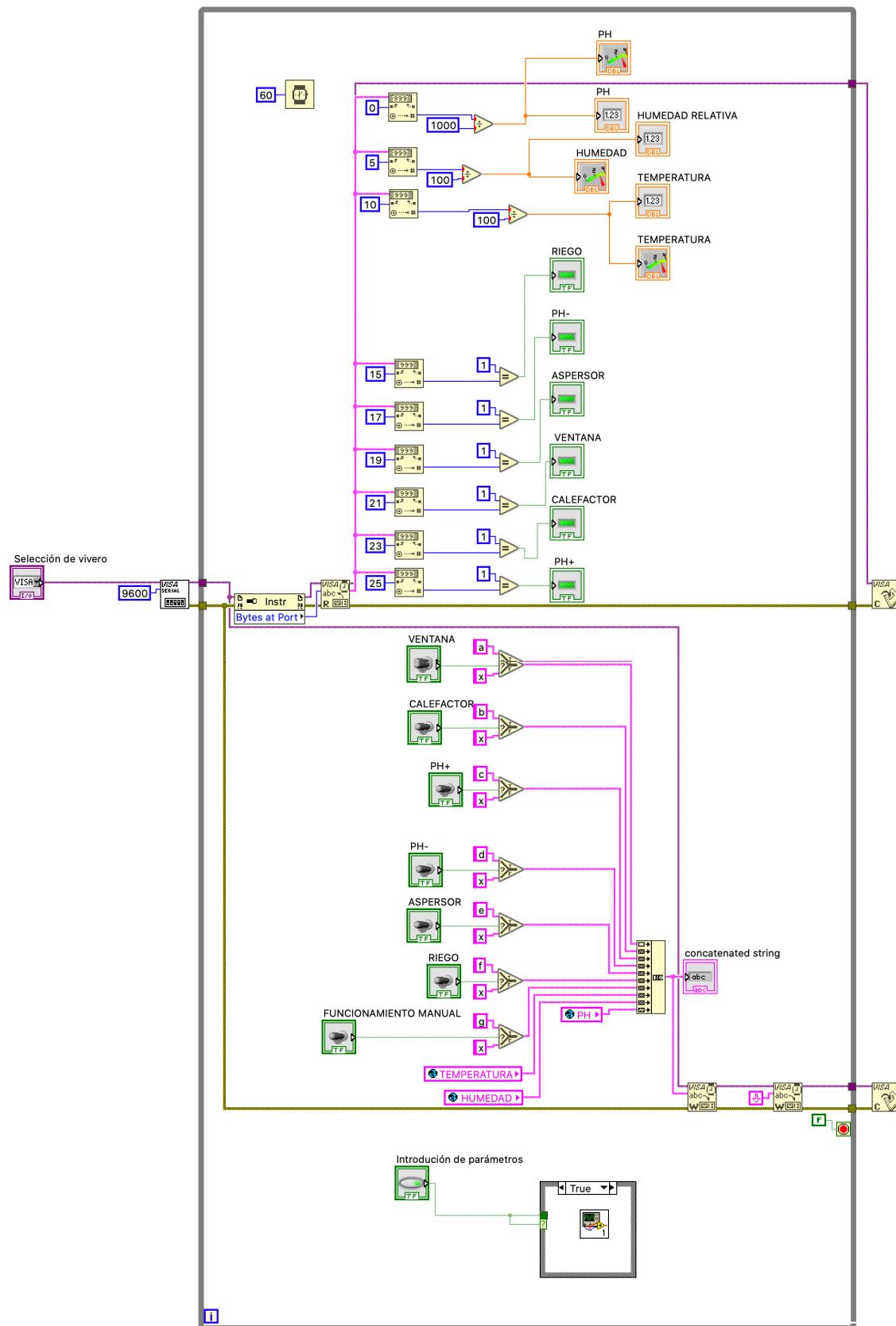
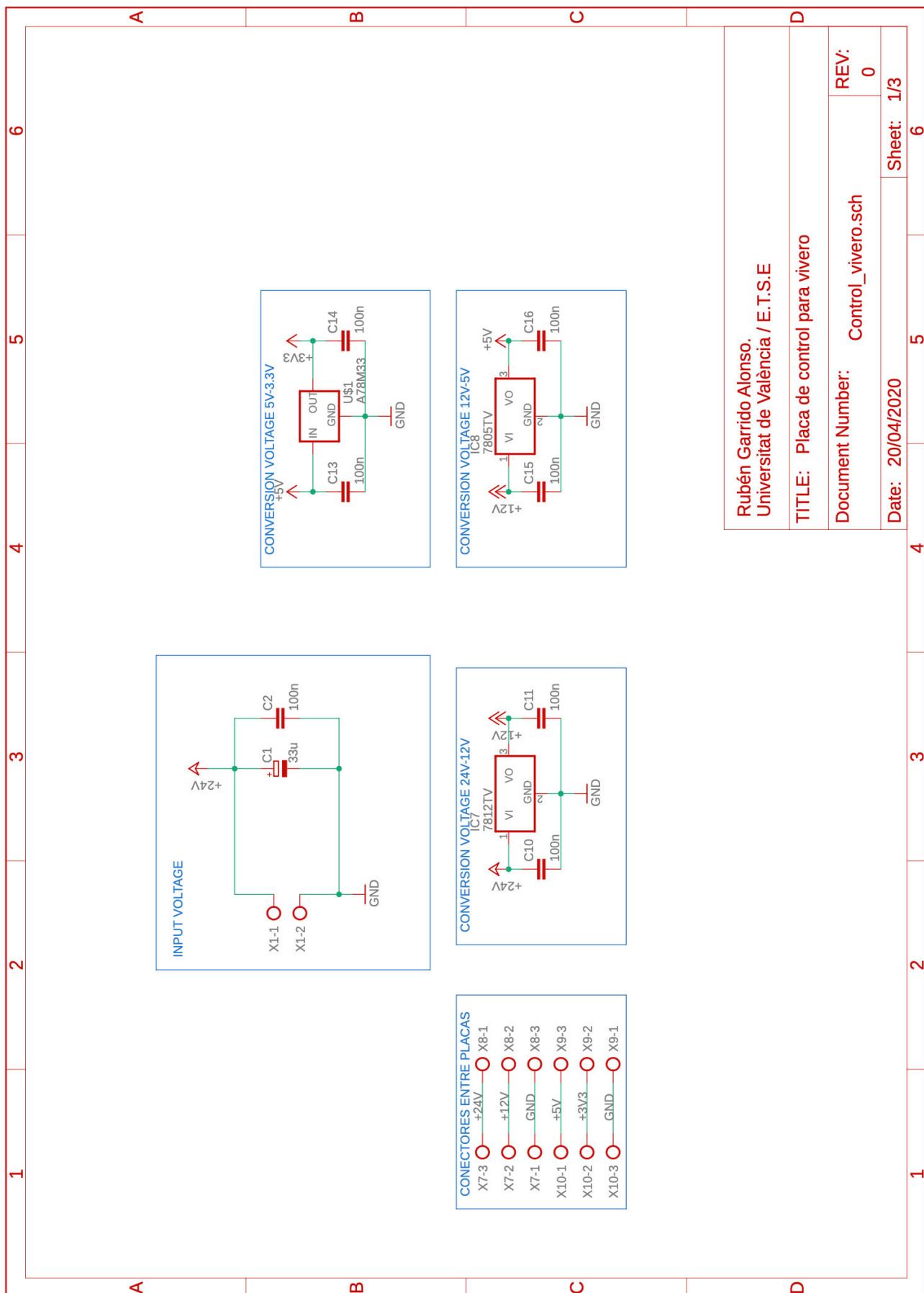


Figura 66. Código del programa principal.

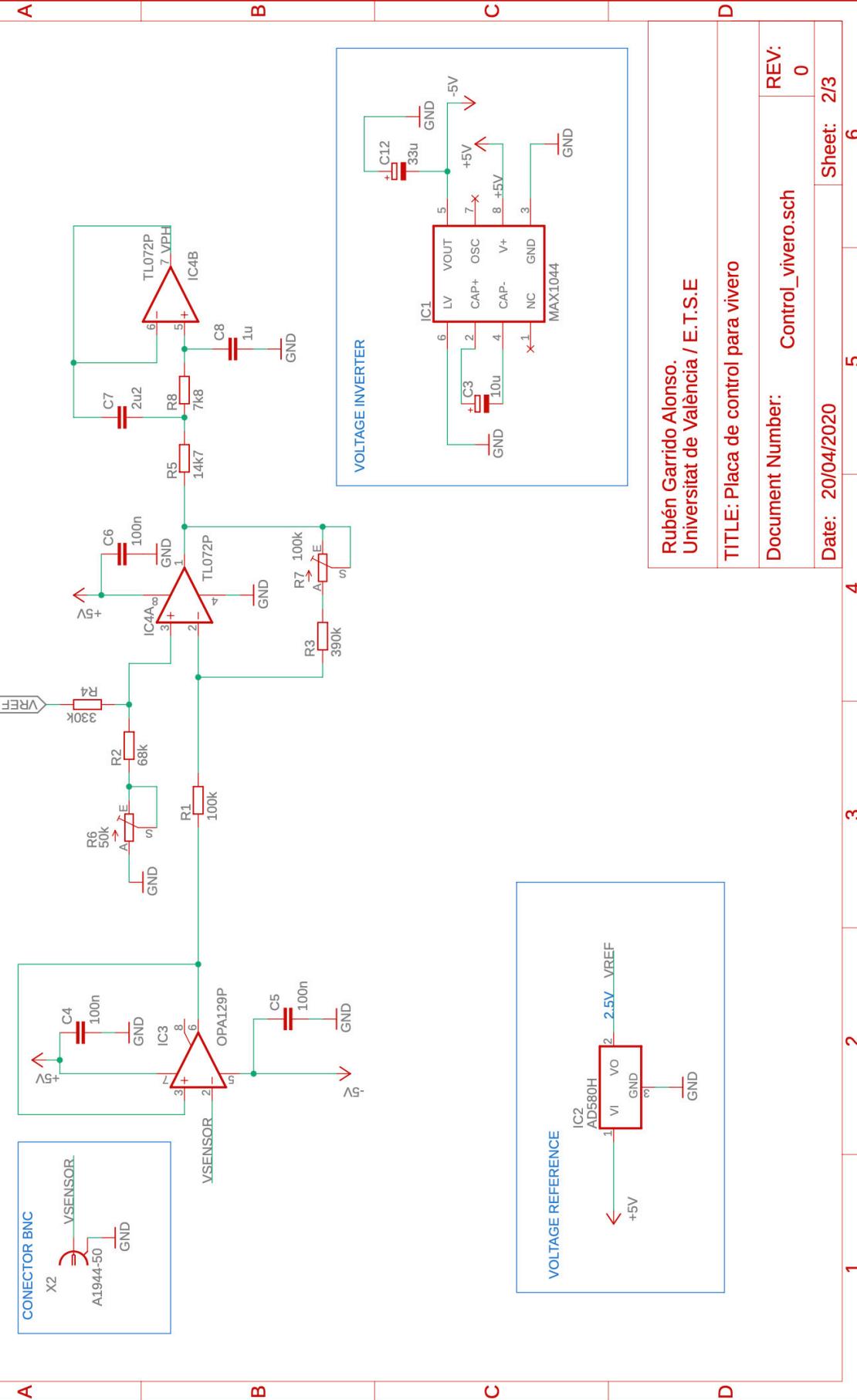


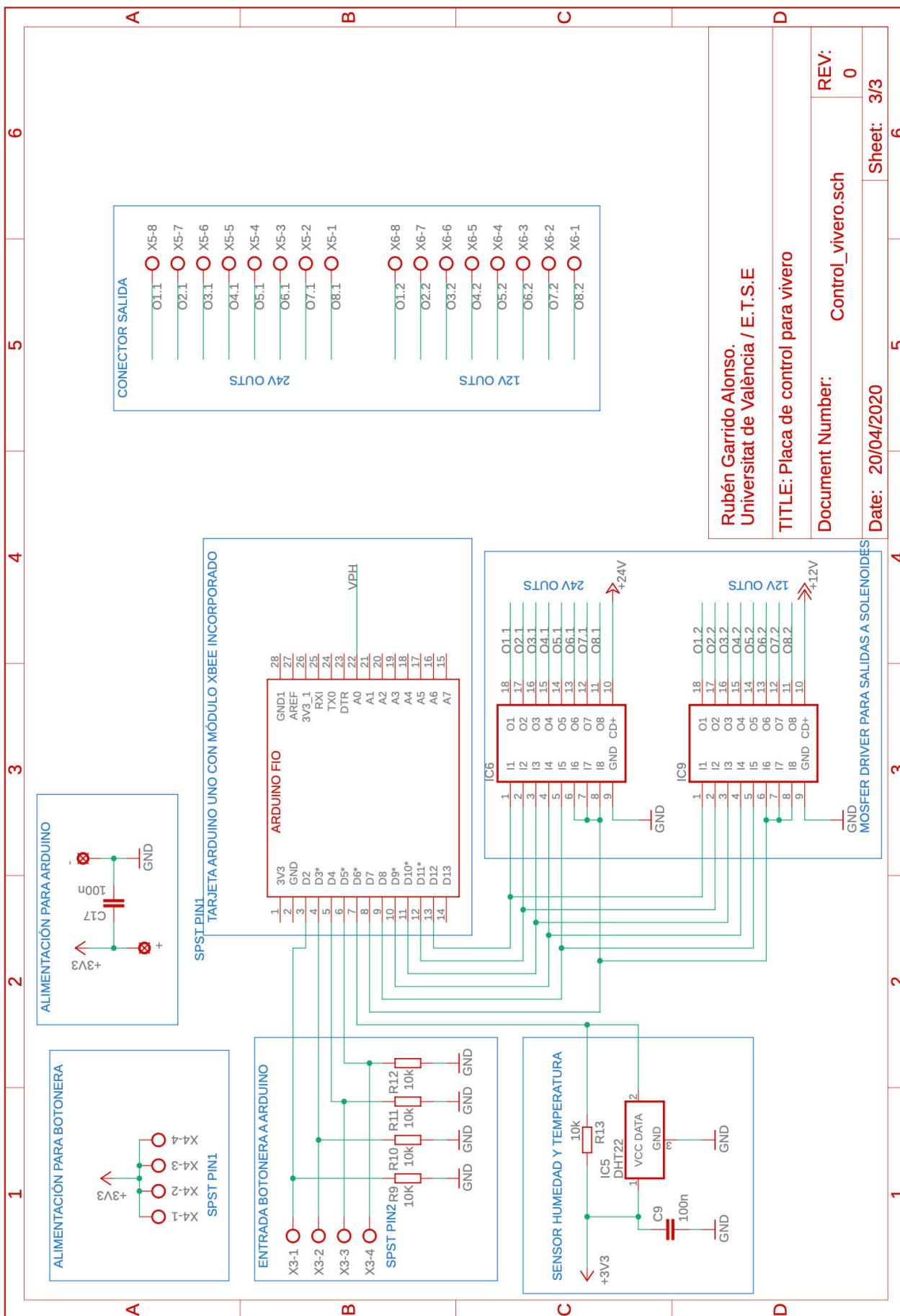
ANEXO 3. ESQUEMÁTICO PCB





CIRCUITO DE ACONDICIONAMIENTO PARA ELECTRODO DE PH





Rubén Garrido Alonso.
Universitat de València / E.T.S.E

TITLE: Placa de control para vivero

Document Number: Control_vivero.sch
Date: 20/04/2020
Sheet: 3/3
REV: 0

6



ANEXO 4. ESQUEMÁTICO PCB DE SIMULACIÓN

