Universitat Oberta de Catalunya

uoc.edu

Sistema de control de Riego para plantas “AgroLOURDES”

Milton Andrés Campoverde Rosales
Plan de Estudios del Estudiante
Área de Electrónica

Aleix Lopez Anton
Nombre Profesor/a responsable de la asignatura

Enero del 2020

Licencias alternativas (elegir alguna de las siguientes y sustituir la de la página anterior)

A) Creative Commons:

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-CompartirIgual 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-CompartirIgual 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento 3.0 España de Creative Commons

B) GNU Free Documentation License (GNU FDL)

C) Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	*AgroLOURDES*
Nombre del autor:	*Milton Campoverde Rosales*
Nombre del consultor/a:	*Aleix Lopez Anton*
Nombre del PRA:	*Nombre y dos apellidos*
Fecha de entrega (mm/aaaa):	01/2020
Titulación::	*Máster universitario en Ingeniería de Telecomunicaciones*
Área del Trabajo Final:	*Electrónica*
Idioma del trabajo:	*Español*
Palabras clave	*Riego, IOT, Agricultura*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

La agricultura es con diferencia el mayor consumidor de agua a nivel global. El 70% del consumo de agua del mundo es para riegos de cultivos. En países en vías de desarrollo, el agua destinada al riego de cultivos representa el 95% del agua consumida, jugando un papel clave dentro de la producción de alimentos.

Además, los países en vías de desarrollo poseen grandes cantidades de reservas de tierra arable y de agua, que pueden satisfacer las necesidades de los alimentos del mundo en el futuro.

Por esta razón nace el proyecto *AgroLOURDES*, porque intenta optimizar el crecimiento del cultivo gracias al registro de parámetros ambientales de las plantas e intentando reducir en lo posible el mal consumo de agua.

Está compuesto por una serie de sensores que registran magnitudes ambientales, un sistema de control que actúa en función de los datos adquiridos a través de los sensores, una interface web donde se puede obtener la información almacenada en la base de datos con la información recogida a través de los sensores.

Este proyecto tiene la finalidad de operar o bien en pequeña escala, o gran escala, tales como: huertos en ciudades, jardines, viveros o grandes extensiones de cultivo.

Además, este sistema debe de ser económico, ya que está enfocado para que se implante preferiblemente en países en vías de desarrollo, mejorando así la monitorización del los cultivos y por ende los beneficios del usuario final.

Abstract (in English, 250 words or less):

Agriculture is by far the largest consumer of water globally. 70% of the world's water consumption is for crop irrigation. In developing countries, water for crop irrigation accounts for 95% of the water consumed, playing a key role in food production.

In addition, developing countries have large amounts of arable land and water reserves, which can meet the world's food needs in the future.

For this reason, AgroLOURDES project is born, because it tries to modify the growth of the crop thanks to the registration of environmental parameters of the plants and try to reduce as much as possible the bad consumption of water.

It consists of a series of sensors that record environmental magnitudes, a control system that acts on the basis of the data acquired through the sensors, a web interface where you can obtain the information stored in the database with the information collected to through the sensors.

This project has the transformation of operating either small-scale or large-scale stories such as: orchards in cities, gardens, nurseries or large areas of cultivation.

In addition, this system must be economical, since it is focused so that it can be affected in developing countries, thus improving the monitoring of crops and therefore the benefits of the end user.

Índice

Lista de figuras

1. Introducción

AgroLOURDES es un sistema automático de control de riego, basado en IOT “Internet of Things” donde permite al usuario monotorizar parámetros ambientales de las plantas.

Este sistema está enfocado para un uso en pequeña escala (jardines, huertos, pequeños invernaderos,etc…) o gran escala(fincas, hectares de cultivo,etc…)

La información recogida de los sensores, es almacenada en una base de datos donde posteriormente el cliente puede acceder y recopilar información obteniendo sus propias conslusiones.

1.1 Contexto y justificación del Trabajo

Como seres vivos las plantas necesitan agua para vivir. Los cultivos absorben los nutrientes del suelo y realizan varias funciones fisiológicas en presencia de agua. Cuando el suelo no tiene la cantidad suficiente de agua o no es oportuna su disponibilidad por medio de la lluvia o de fuentes naturales, se hace necesario el riego. El sistema de riego depende del tipo de suelo, del cultivo, de la cantidad de agua necesaria, de la mano de obra disponible y de los recursos económicos, ya que un sistema de riego supone una inversión considerable en el negocio agropecuario. El riego no solo implica el costo de su instalación sino también el de su mantenimiento.
Un adecuado sistema de riego suministra la cantidad necesaria de agua en el momento que se necesita, humedeciendo el suelo hasta la profundidad que requiera el cultivo.
Los cultivos tienen momentos críticos para sus necesidades de agua que, si no se subsanan, son traducidos en pérdidas en rendimiento o por falta de germinación.
Existen diferentes tipos de riego, como por ejemplo el riego por gravedad que surte el agua por tuberías suministrando agua al terreno. Este método es similar al riego por inundación que se utiliza en los campos de arroz.
Tenemos también el riego por goteo, que puede realizarse tanto en superficie como en subterráneo mediante mangueras y goteros ubicados estratégicamente en la zona de absorción por parte de las plantas.
El sistema que se diseña en este trabajo está enfocado en el control de parámetros ambientales y vitales para la supervivencia del cultivo.
La idea es la creación de un sistema funcional y económico para que en lugares donde no se tenga la disponibilidad económica puedan acceder al producto y así aumentar el cuidado y rendimiento de sus plantaciones, concretamente se especifica para un mercado de países en vías de desarrollo, ya que éstos poseen grandes reservas de tierra arable y agua, y pueden satisfacer las necesidades de alimentos del mundo y del futuro.

1.2 Objetivos del Trabajo

En este trabajo se propone un sistema que permite automatizar el riego en invernaderos para plantas de distintas especies haciendo un uso eficiente de los recursos para la supervivencia de la planta.
Esta solución se consigue, gracias a la implantación de sensores tales como:

- Temperatura
- Medidor de Luz
- Humedad Relativa.

Además, el sistema también constara de:

- Bomba peristáltica, para dosificar los nutrientes para el cultivo
- Electroválvulas para tener un control del caudal del agua.
- Ventilador para regular el nivel de temperatura de la zona.
- Pantalla a LCD
- Escudo de comunicación por WiFi.
- Microcontrolador

La información recogida será enviada a un servidor donde estará alojada una BBDD “base de datos” para poder obtener estadísticas de los datos observados mediante sensores.

Cuando el proyecto finalice el usuario será capaz de:

- Obtener la información de los sensores accediendo al servidor y actualizar la información almacenada en la base de datos.
- Activar de forma automática la apertura y cierre de la electroválvula
- Controlar el flujo del caudal de la bomba peristáltica
- Controlar la velocidad de o de los ventiladores para regular la temperatura ambiental.

1.3 Enfoque y método seguido

Actualmente existen diferentes sistemas de riego en donde el sistema de riego por goteo es muy significativo y un referente en todas sus variantes. El futuro va a estar marcado por el uso racional en el consumo y utilización del agua, y aquí, el riego por goteo va a estar presentes incluso en cultivos en los que hoy en día no están planteados; y es que el ahorro de agua, su tratamiento y el óptimo aprovechamiento de ésta, van a llevarnos a hacer extensibles si uso en casi todos los ámbitos de la producción vegetal.
Además de otros motivos por esta razón se ha diseñado este dispositivo, con la idea de riego por goteo, de bajo coste y fácil adquisición permitiendo la posibilidad de mejoras a corto y largo plazo. Una de las características relevantes del proyecto es que trabaja con el estándar IEEE 802.11 Wi-Fi, permitiendo al usuario una comunicación inalámbrica con el dispositivo, de esta forma se reducen costes por cableado, posible conectividad desde cualquier lugar y posibilidad de la elección entre varias señales libres o con seguridad.

1.4 Motivación

La principal motivación del autor de este documento para realizar este proyecto es la de poder realizar un trabajo de investigación y desarrollo con una finalidad en concreto, en este caso la monitorización y riego de plantas.
Gracias al avance de la tecnología, se pueden dar soluciones para alternativas que antes eran inimaginables y costosas.

La posibilidad de crear más proyectos de la misma índole es una motivación extra para el autor debido a que existen mercados tales como en países en vías de desarrollo donde los sistemas de riego, cuidado, monitorización, etc… son económicamente inalcanzables para un gran sector de la población y ofrecerles una solución económica, fiable y robusta es una gran motivación.

Por último, el autor considera que a realización práctica de este proyecto le va aportar experiencia que va a usar de manera inmediata una vez finalizado el proyecto.

1.5 Planificación del trabajo

Las etapas del proyecto se han ido resolviendo según las fechas límites de cada PEC.

En la siguiente Figura: 1 podemos ver el tiempo de cada etapa.

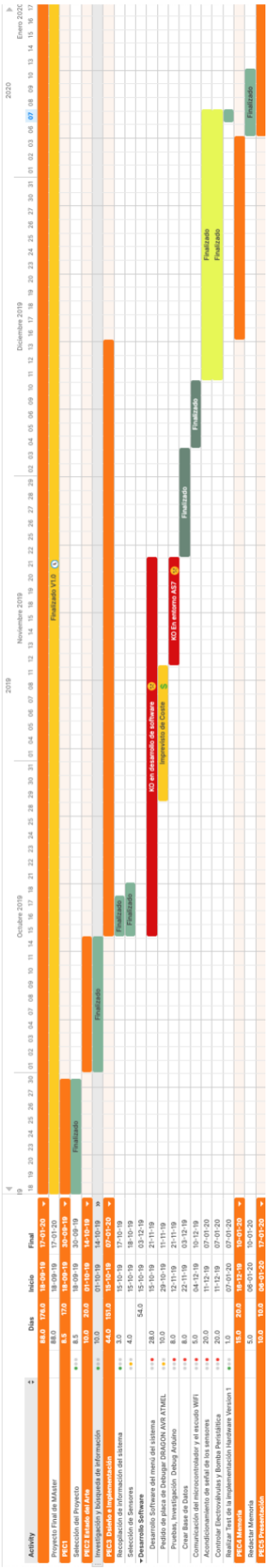


Figura: 1Diagrama de Gantt del proyecto

1.6 Breve sumario de productos obtenidos

En la siguiente Figura: 2, mostramos los productos en formato WBS (Work Breakdown Structure).

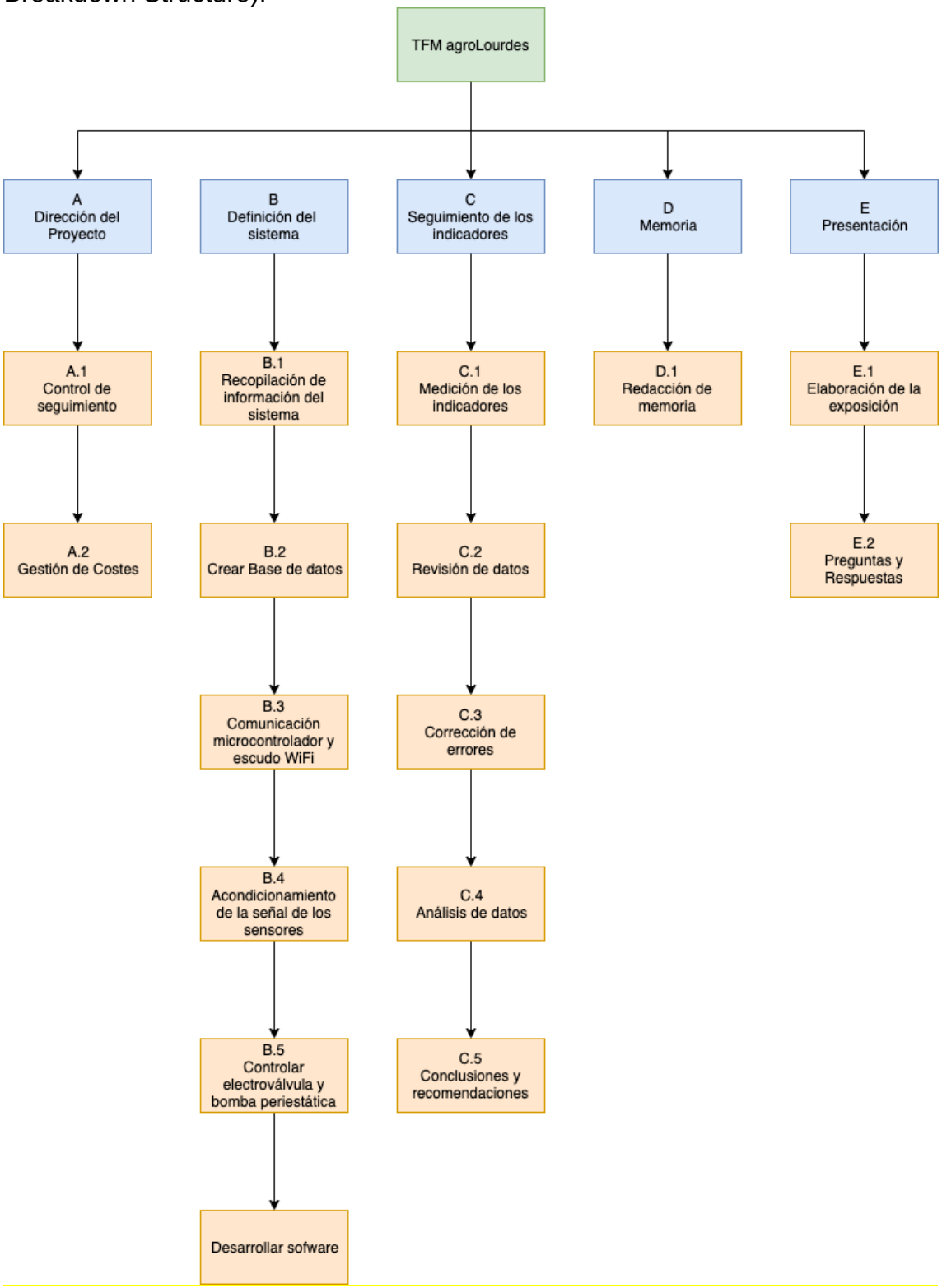


Figura: 2Productos obtenidos en formato WBS

1.7 Breve descripción de los otros capítulos de la memoria

La memoria se encuentra estructurada en 4 capítulos, cada uno con sus respectivos apartados, se incluye un glosario, bibliografía y anexos.

- Introducción: Contiene una introducción del proyecto, con su descripción del problema y trabajos relacionados. Se expresa la motivación, objetivos y planificación del proyecto con un diagrama de Gantt.
- Estado del Arte: Se crea el estado del arte, donde se explica las diferentes versiones de placas de desarrollo que existen en el mercado, Arduino, Raspberry PI, etc… Se incluyen ejemplos de sistemas relacionados con el trabajo.
- Descripción del sistema: Se describe el sistema Hardware y Software del proyecto, comentando acondicionamiento de sensores, prácticas físicas, simulaciones teóricas y cálculos matemáticos.
- Implementación del sistema: Se describe el funcionamiento y resultados conseguidos. Además, se exponen conclusiones sobre el proyecto.

2. Estado del arte

Se ha efectuado una investigación sobre proyectos y sistemas similares al que se desarrolla en este TFM. Lo más relevante luego de la investigación ha sido:

- IOT “Internet de las cosas”: El término IOT se acuñó por primera vez en los años 80 en el entorno militar , pero ha sido en el transcurso de los últimos años en donde se ha extendido de manera espectacular su uso y aplicaciones.
  Es por ello por lo que muchos expertos se refieren al IOT como la siguiente revolución industrial, que cambiará la manera en que las personas se comunican, trabajan, viajan, etc… y también cambiará la forma en las que las empresas interactúan con el mundo.

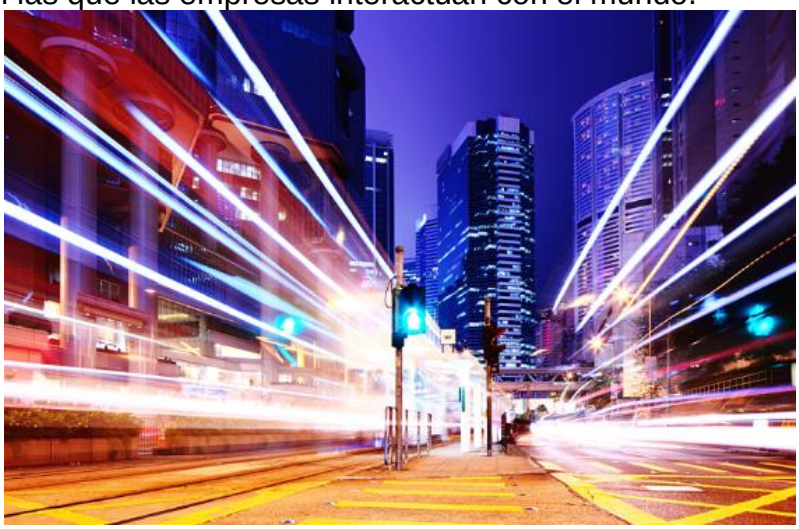


Figura: 3 IoT Internet de las coses

Cuando hablamos de IoT nos referimos a la interconcexión de dispositives físicos de diferentes naturalezas: vehículos, sensores, edificios… y se caracterizan por tener la capacidad de recolectar e intercambiar datos.
IoT tiene prácticamente todos los sectores y entornos, donde destacamos los siguientes:

1. **Transporte:** Coches conectados, siendo estos unos de lso dispositivos IoT más sofisticados, ya son una realidad y se estima que en 2020 habrá más de 220 millones.
2. **Agricultura:** En este caso los dispositivos son fundamentalmente sensores que toman medidas de niveles de acidez, temperaturas y otras variables que ayudan a mejorar cultivos.
3. **Banca:** Existen más de 3 millones d ecajeros automáticos distribuidos por el mundo, los cuales comienzan a incorporar capacidades de IoT como video en streaming para mejorar la atención del cliente.
4. **Energía:** Las aplicaciones en este campo sin enormes, desde sensores qye tomen métricas en los sistemas de generación y distribución de energía hasta contadores inteligentes en la cada de los clientes.
5. **Hogar conectado:** se estima que en 2030 la mayoría de los dispositivos domésticos están conectados, esto abre un gran abanico en nuevas aplicaciones.
6. **Salud:** Los dispositivos aplicados al campo de la salud pueden recoger datos y automatizar procesos que permitan un mejor diagnóstico y mejoren los tratamientos de los pacientes.
7. **Turismo:** más del 31% de los hoteles ya usa puertas inteligentes en sus habitaciones e incorpora sensores que mejora el confort y la experiencia de los clientes.
8. **Ciudades Inteligentes:** La IoT es la clave para recolectar información y la gestión del conocimiento.

Figura: 4Cloud IoT

IoT tiene un impacto brutal en los diferentes aspectos a tener en cuenta en sistemas complejo, en los próximos años la innovación se va a focalizar en las siguientes 6 etapas:

1. **Conectividad**
2. **Seguridad**
3. **Almacenamiento de datos**
4. **Integración de Sistemas**
5. **Dispositivos hardware**
6. **Desarrollo de aplicaciones**

Big Data va a estar presente en todo este proceso ya que le volumen de datos insertados y recolectados va a crecer de forma exponencial y el número de dispositivos y orígenes de datos alcanzarán cifras hasta ahora inimaginables.

Las tecnologías Big Data deben de dar respuesta a los desafíos técnicos que el internet de las cosas pone sobre la mesa:

- Procesos de ingestión de datos en streaming o en tiempo real
- Almacenamiento ordenado de datos de diversa naturaleza y estructura
- Procesamiento eficiente de grandes volúmenes de información de forma distribuida
- Algoritmos inteligentes y analítica avanzada que habrá que aplicar sobre los datos para obtener valor de los mismo

- **Tecnologías y Plataformas:**
  Sin duda, el hardware libre y de bajo coste ha abierto la puerta a la proliferación de dispositivos asequibles y su aplicación a nuevos campos. Proyectos como Arduino y Raspberry Pi han dinamizado el mercado del hardware IoT y han supuesto un verdadero driver para la innovación, siguiendo los principios y filosofía del software libre, pero aplicado en el mundo del diseño hardware.

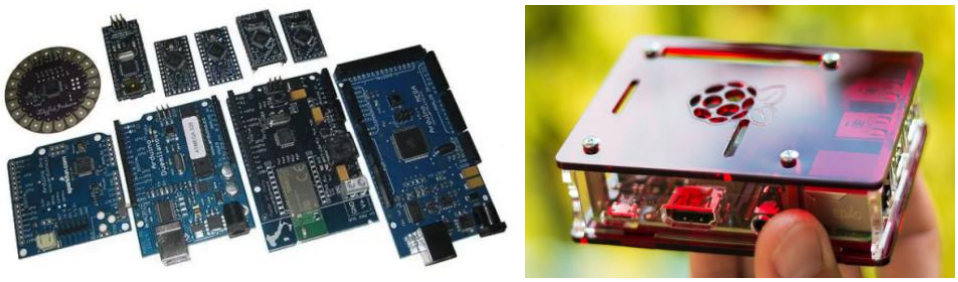


Figura: 5 Família Arduino y Raspberry Pi

Arduinos oficiales y no oficiales

- **Oficiales:** son aquellas placas oficiales manufacturadas por la compañía italiana Smart Projects y algunas han sido diseñadas por la empresa estadounidense SparkFun Electronics (SFE) o por la también estadounidense Gravitech. Incluso el gigante Intel ha colaborado en el

diseño de una de estas placas… Arduino Pro, Pro Mini y LilyPad son las manufacturadas por SFE y Arduino Nano por Gravitech, el resto se crean en Italia. Estas placas son las reconocidas oficialmente, incluyen el logo y son las únicas que pueden llevar la marca registrada de Arduino.

Figura: 6 Entorno de desarrollo Arduino

- **No oficiales o compatibles:** son placas compatibles con Arduino pero no pueden estar registradas bajo el nombre de Arduino. Por supuesto son diseñadas y fabricadas por otras compañías ajenas. Estos desarrollos no aportan nada al desarrollo propio de Arduino, sino que son derivados que han salido para cubrir otras necesidades. Estas frecuentemente utilizan un nombre que integra el sufijo “duino” para identificarlas, como por ejemplo Freeduino

Tipo de Placas Oficiales

- **Arduino UNO:** Se basa en un microcontrolador Atmel ATmega320 de 8 bits a 16Mhz que funciona a 5v. 32KB son correspondientes a la memoria flash (0,5KB reservados para el bootloader), 2KB de SRAM y 1KB de EEPROM. En cuanto a memoria es una de las placas más limitadas, pero no por ello resulta insuficiente para casi todos los proyectos que rondan la red.Los pines pueden trabajar con intensidades de corriente de hasta 40mA.

Figura: 7 Placa Arduino UNO

- **Arduino TRE:** Primera placa Arduino fabricada en Estados Unidos. Integra un procesador Texas Instrument Sitara AM335x de 1Ghz basado en ARM Cortex A8 con 512MB de DDR3L, lo que le da hasta 100 veces más rendimiento comparado con otras placas como Leonardo y Uno. Esto abre las puertas a más aplicaciones avanzadas y soporte para sistemas basados en Linux. Por un lado sigue contando

con el microcontrolador Atmel ATMega32u4 de 16Mhz y 32KB de flash y 2.5KB de SRAM, junto al 1KB de EEPROM. Tiene 14 pines digitales, 7 PWM, 6 analógicos multiplexados, y su parte SBC cuenta con HDMI, USB, microSD, soporte para conector LCD, etc… Todo compatible con los escudos de Arduino y con Arduino IDE.

Figura: 8 Placa Arduino TRE

- **Arduino/Genuino 101**

Se trata de una placa que se conoce como Arduino 101 en América y Genuino 101 fuera de Estados Unidos. Esta nueva placa ha sido presentada en el Opening Conference at Maker Faire de Roma, y su precio ronda los 30$ (27€). Genuino 101 sigue la misma filosofía de las placas oficiales de Arduino, pero llama la atención su módulo Intel Curie, un módulo de dimensiones reducidas y bajo consumo potenciados por el SoC Intel Quark de 32 bits. Se trata de un SoC que contiene un microcontrolador x86 (una oportunidad única de programar en una plataforma x86, alejándose de los ATMega y los ARM), 80KB de SRAM (24KB disponible para sketches), 384 KB de memoria flash, DSP, Bluetooth, sensores acelerómetros y giroscopio, etc… Por el resto de las características, como conexiones y tamaño, es igual a Arduino UNO y compatibles con sus shields.

Figura: 9 Placa Arduino/Genuino 101

- **Arduino Mega:** Su nombre proviene del microcontrolador que lo maneja, un ATmega2560. Este chip trabaja a 16Mhz y con un voltaje de 5v. Sus capacidades son superiores al ATmega320 del Arduino UNO, aunque no tan superiores como las soluciones basadas en ARM. Este microcontrolador de 8 bits trabaja conjuntamente con una SRAM de 8KB, 4KB de EEPROM y 256KB de flash (8KB para el bootloader). Como puedes apreciar, las facultades de esta placa se asemejan al Due, pero basadas en arquitectura AVR en vez de ARM. En cuanto a características electrónicas es bastante similar a los anteriores, sobre todo al UNO. Pero como se puede apreciar a simple vista, el número de pines es parecido al Arduino Due: 54 pines digitales (15 de ellos PWM) y 16 pines analógicos. Esta placa es idónea para quien necesita más pines y potencia de la que aporta UNO, pero el rendimiento necesario no hace necesario acudir a los ARM-based.

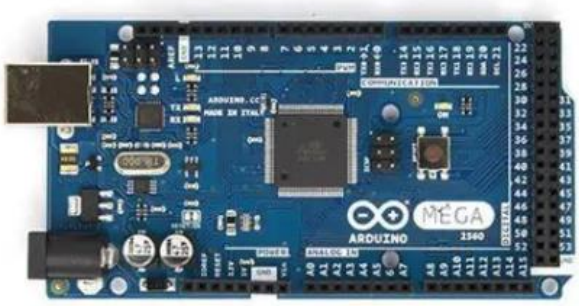


Figura: 10 Placa Arduino Mega

- **Arduino Primo:** Apuesta fuertemente por la conectividad gracias a Nordic nRF52. Lo que significa que Arduino Primo está pensado para el IoT o el Internet de las cosas. Contará con Bluetooth LE (Low Energy), WiFi e incluso tecnologías como NFC o IR. Gran trabajo de desarrollo realizado por Arduino Project para adaptarse a las nuevas necesidades tecnológicas.

Figura: 11 Placa Arduino Primo

Tipo de placas no oficiales

Existen centenares de ellas, por lo que redactaré las más famosas. Hay que decir que siempre es mejor trabajar con las oficiales por cuestiones de soporte y de comunidad de desarrolladores que resultan obvias, pero hay que reconocer que algunas placas compatibles son ciertamente interesantes:

- **Almond PCB:** OpenBIonics nos trae Almond PCB, una placa similar a las de Arduino oficiales. Incluye un microcontrolador Atmega 2560, 11 SALIDAS, 9 pines E/S configurables digitales, 2 pines ADC, 256KB de flash, 4KB de EEPROM, USB, I2C, UART, SPI, etc…

Figura: 12 Placa Almond PCB

- **AVR.duino U+:** SlicMicro es el creador de esta placa compatible, tanto en hardware como en software, con Arduino UNO Rev3. Esta plataforma de hardware open source añade características frente al oficial. Para poder pasar tu código desde Arduino IDE debes seleccionar la opción Arduino UNO Rev3 como si ésta fuese tu placa y el código cargará sin problemas. Las características adicionales que integra (SlicBus Port, un LED adicional, potenciómetro, pulsador). El resto es igual al Arduino, incluido su ATmega328 que comparte con algunas versiones oficiales. Esta placa es interesante para aquellos que buscan las características combinadas de Arduino UNO y de Esplora, aunque más limitada en gadgets onboard que esta última. Lo que si es una ventaja es su puerto SlicBus que permite conectar módulos especiales fabricados por SlicMicro.

Figura: 13 AVR.duino U+

- **Bq ZUM BT-328:** Es una de las mejores placas compatibles con Arduino que ha creado la compañía española bq. La placa tiene algunas novedades frente a Arduino UNO oficial, como la inclusión del set de tres pines (para conectar sin hacer empalmes), botón de encendido y apagado, Bluetooth, soporta más conexiones gracias a sus 3.2A frente a los 0.8A de la oficial, tiene una conexión microUSB, etc. Además, para su programación puedes usar una plataforma IDE web que se denomina bitbloq. Y todo por unos 35 €.

Figura: 14 Placa Bq Zum BT-328

- **Goldilocks:** Thin Layer Embedded diseñó esta placa basada en un FPGA (Altera Cyclone IV, con RAM DDR2, SRAM, flash, oscilador y un Atmel ATSHA204 Authentication IC/EEPROM) para ofrecer una flexibilidad extrema. Es compatible con los shields de Arduino, pero en este caso no solo se puede programar a nivel de software, sino también a nivel de hardware gracias a su FPGA. Esta misma compañía también tiene otro modelo muy similar denominado Breadstick con unos pines macho especialmente pensados para insertarlo en una protoboard.

Figura: 15 Placa Goldilocks

- **Zigduino:** Logos Electromechanical ha creado este kit que es más que una simple placa. Integra un microcontrolador ATmega128RFA1, un ZigBee para conexiones a red inalámbrica (IEEE 802.15.4). También incluye un jack externo RPSMA y es totalmente compatible con Arduino Duemilanove. Al implementar estas funcionalidades de red sin necesidad de shields externos, el precio de la placa supera los 50 euros.

Figura: 16 Pla Zigduino

Lenguajes de programación

El lenguaje de programación de Arduino está basado en C++.

Figura: 17 LEnguaje de programación C++

Características de C:

- Es el lenguaje de programación de propósito general asociado al sistema operativo UNIX
- Es un lenguaje de medio nivel. Trata con objetos básicos como caracteres, números, etc… también con bits y direcciones de memoria.
- Posee una gran portabilidad.
- Se utiliza para la programación de sistemas: construcción de intérpretes, compiladores, editores de texto,etc.

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.
Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma. Actualmente existe un estándar, denominado ISO C++,
C# es un lenguaje propietario de Microsoft que mezcla las características básicas de C++ (no las avanzadas) simplificándolas al estilo Java y ofreciendo un framework. C# forma parte de la plataforma .NET

Existe una guía de estilo para escribir código claro de Arduino y que sea fácil de entender. No es obligatorio, pero es una recomendación:

- Documentar al máximo
- Usar esquemas
- Predominar la facilidad de lectura sobre la eficiencia del código
- Poner el setup() y loop() al principio del programa
- Usar variables descriptivas
- Explicar el código al principio
- Usar identación

Lenguajes de programación para Arduino Alternativos

Existen alternativas de lenguaje para desarrollar software con Arduino.

- **.Net:** Muchos usuarios tienen experiencias con el conjunto de lenguajes del framework de Microsoft .NET y es que Microsoft, está apostando por la creación de wearables (Dispositivos llevables) con su propia versión del sdk (Software Development Kit) para Intel Galileo. Si no tenemos esta versión del intel galileo, podemos utilizar una pequeña placa compatible con Arduino UNO llamada netduino. La cual nos permite programar nuestro Arduino utilizando el entorno de desarrollo Visual C#(o Visual Studio) con el lenguaje de programación C#.
- **Python:** se puede utilizar la librería *pyfirmdata* para poder controlar el arduino de forma remota. Por un lado, necesitaremos poner un programa especifico en nuestro Arduino y así podemos mandar las ordenes desde nuestro equipo usando python.
- **Node.js:** la librería *jhony-five* para node JS nos permite programar nuestro arduino de igual manera que lo hecho hecho con python.
- **Java:** Con la librería *RXTXcommons* y la librería *Arduino* para Java podemos trabajar con este lenguaje.

Depuradores de Código en Arduino

Cuando trabajamos en proyectos, donde se debe la extensión del código software toma grandes extensiones de líneas de código. Es inevitable que aparezcan errores en donde no podemos encontrar fácilmente el error. Por esta razón se necesitan es necesario la depuración de código.

- **The Atmel-ICE Debugger**:Es una herrmienta de desarrollo de gran alcance de Microchip para depurar y programar los microcontroladores Microchip SAM y AVR basados en Microchip ARM Cortex-M con capacidad de depuración en el chip. Es compatible con las siguientes interfaces JTAG, SWD, PDI, TPI, aWire, SPI y debugWIRE. Hasta 128 puntos de interrupción de software.

Figura: 18 Placa Atmel-ICE Debugger

- **AVR Dragon:** Herramienta de desarrollo de bajo coste para dispositivos AVR de 8 y 32 bits con capacidad de depuración en chip (OCD). Puede realizar una depuración con SPI, JTAG, PDI. Programación en serie de alto voltaje, programación paralela y modos aWire, y admite depuración mediante SPI, JTAG, PDI.

Figura: 19 AVR Dragon

Aplicaciones de Arduino

- **Invernadero electrónico:** Regular la temperatura dentro del invernadero gracias al uso controlado de lámparas y con sistema servomotores para abrir las ventanas y encender ventiladores.
  Mantener constante el nivel de humedad del suelo con un sistema de riego con bomba de agua y recibiendo los datos de temepratura ambiente, humedad del aire y luminosidad.

Figura: 20 Prototipo Invernadero electrónico

- **Nduino:** Medidor de bajo coste basado en Arduino para la agricultura de precisión. Este aparato está destinado a determinar los requerimientos de nitrógeno de las plantas(cereales, fundamentalmente) directamente en ele campo, con el fin de afinar la fertilización nitrogenada.
  Mide el contenido en clorofila de la hoja, y como este está relacionado con el estado de nutrición de la planta, nos permite conocer el estado del sembrio.

Figura: 21 Prototipo Nduino

- **Garduino:** Huerto controlado con Arduino, creando un sistema controlado de irrigación, iluminación y temperatura.

Figura: 22 Prototipo Garduino

- **Vinduino:** Proyecto de ahorro de agua de un viticultor. Monitoreo de la humedad del suelo a doferentes profundidades para determinar cuándo regar, y más importante la cantidad de agua que necesita. Se consiguen ahorros de hasta un 25%.

Figura: 23 Prototipo Vinduino

3. Descripción del Sistema

3.1 Herramientas utilizadas Sofware

El proyecto se ha desarrollado con la ayuda de 3 programas:

1. **IDE "Entorno de Desarrollo Integrado" Software Arduino**
2. **PSpice Circuit Simulation Technology**
3. **Paquete XAMPP**

IDE ARDUINO: es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse exclusivamente a un solo lenguaje de programación o bien puede utilizarse para varios.
El IDE es un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Además, en el caso de Arduino incorpora las herramientas para cargar el programa ya compilado en la memoria flash del hardware.
Los programas de Arduino están compuestos por un solo fichero con extensión "ino", aunque es posible organizarlo en varios ficheros. El fichero principal debe de estar en una carpeta con el mismo nombre que el fichero.

Figura: 24 Entorno de desarrollo integrado Arduino

PSpice Circuit Simulation Technology: Pspice es un programa para la simulación de circuitos analógicos y digitales que permite, además de comprobar su funcionamiento, realizar el análisis y el estudio de los mismos. El programa PSPICE ha ido evolucionando desde sus inicios hasta la versión actual en entornos WINDOWS.
Las verisones para WINDOWS incluye un programa denominado SCHEMATICS que permite crear un dibujo del esquema de un circuito, así como capturarlo. El dibujo puede componerse de símbolos, atributos, conexiones y texto. El conjunto de herramientas que permiten dibujar, capturar, simular y analizar circuitos analógicos y digitales se engloban bajo el nombre de PSPICE

Figura: 25 PSPICE STUDENT

XAMPP: Es una distribución de Apache que incluye software libre. El nombre es un acrónimo compuesto por las iniciales de los programas que los constituyen: el servidor web Apache, los sistemas relacionalesde administración de base de datos MySQL y MariaDB, así como los lenguajes de programación Perl y PHP,

la inicial X se usa para representar a los sistemas operativos Linux, Windows y Mac OS X.

Figura: 26 XAMPP Sofware libre

3.2 Herramientas utilizadas Hardware

En la siguiente Figura: 27 se muestra los materiales utilizados:

1. Generador de Frecuencias rango de frecuencias [0-1]MHz
2. Osciloscopio Promax 60MHz
3. Fuente de tensión 0-30V 0-5ª
4. 2 Multímetros
5. Depurador AVR Dragon

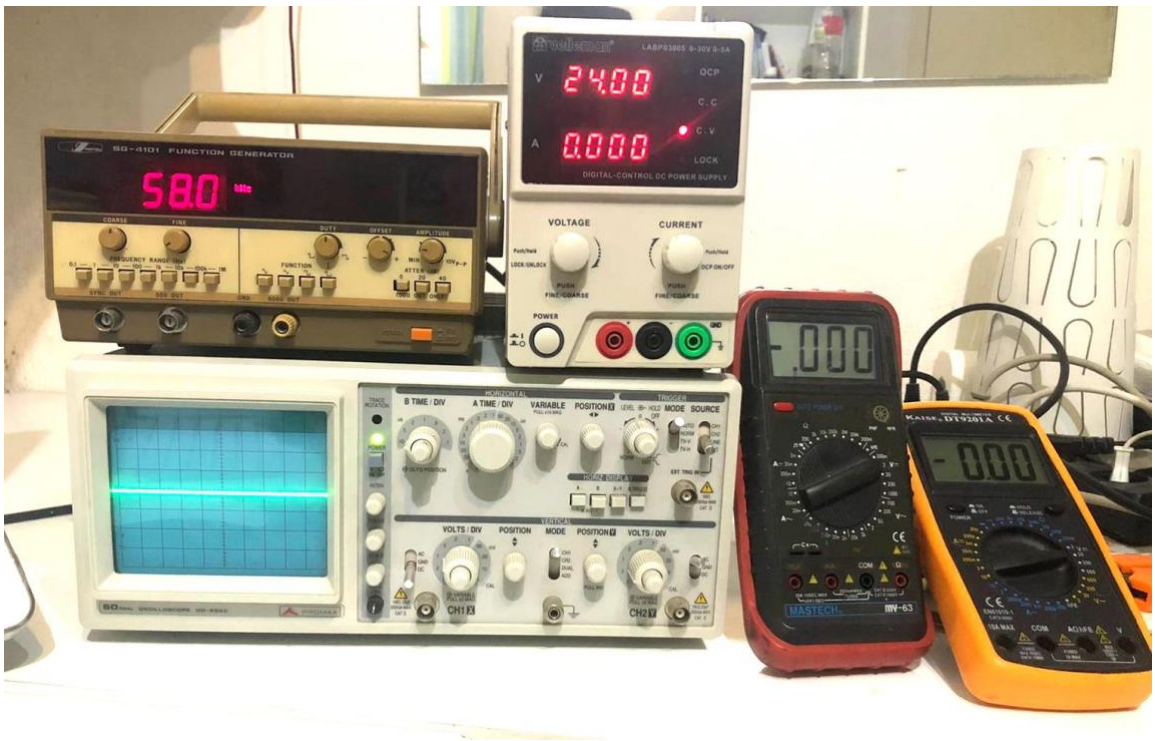


Figura: 27 Herramientas utilizadas en el proyecto

3.3 Sistema Hardware

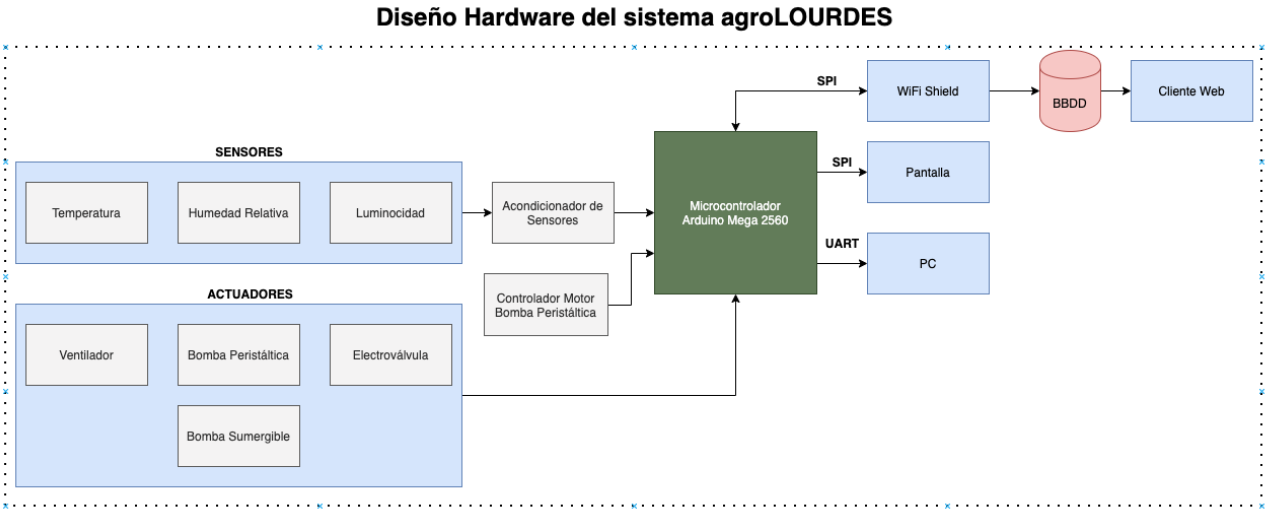


Figura: 28 Diagrama de bloques de la arquitectura del sistema

3.3.0 Microcontrolador

- Arduino Mega 2560 R3: Se encarga de recoger la información de los sensores y permite la comunicación con el servidor donde se almacenan los datos.
  Sus características principales son:
  1. Microcontrolador con memoria de 8KB
  2. 54 entradas/salidas digitales y 14 de éstas pueden utilizarse para salidas PWM
  3. 16 entradas analógicas, UARTs(puerto serials), un oscilador de 16MHz, conexión USB, conector de alimentación , un header ICSP y un pulsador RESET.

Figura: 29 Ardino Mega2560 R3

3.3.1 Comunicación

- Arduino WiFi Shield SD: Este módulo se encuentra conectado vía bus SPI con el objetivo de hacer un puente entre Arduino y la base de datos. El objetivo principal es la conexión con un AP(Punto de acceso) y pasar los datos a una dirección web.
  Característica principal:

  1. Alimentación de 5V
  2. Conectividad 802.11b/g
  3. Encriptación WEP y WAP2
  4. Comunicación por protocolo SPI

Figura: 30 Shield WiFi

3.3.2 Sensores

Para la selección de los sensores, se han tenido en cuenta las siguientes pautas:

1. Campo de medida
2. Intercambiabilidad
3. Estabilidad
4. Robustez
5. Tamaño

- Temperatura LM35:
  Se optó por este sensor, debido a la precisión de 10mV/ºC, su baja tensión de alimentación 5V, rango de temperatura de operación [-55;150]ºC y facilidad de implementación. No necesita acondicionamiento de la señal de la salida de este sensor. Sin embargo, en este sistema será necesario porque mi ADC tiene una tensión de referencia de 2.56V.

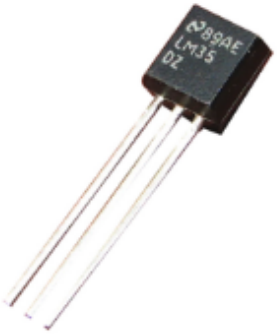


Figura: 31 Sensor de Temperatura LM35

- Humedad Relativa HIH-5030-001
  Se optó por este sensor por su salida lineal vs HR. Su diseño en baja potencia y estabilidad. Tiene una alimentación de 2.7v.

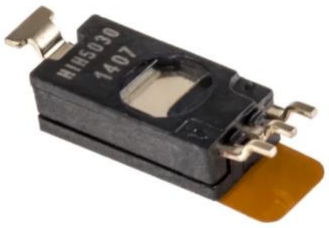


Figura: 32 Humedad Relativa HIH-5030-001

- LDR NSL-19M51: Se optó por este sensor debido a su valor económico encontrado en el punto de distribución RS-ONLINE.
  El fabricante expone pocos detalles sobre el sensor, así que los resultados serán obtenidos de forma experimental.

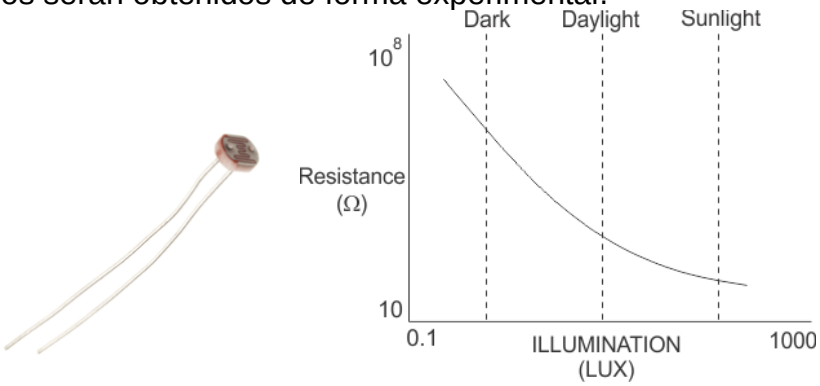


Figura: 33 NSL-19M51

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNITS
Light Resistance	10 lux., 2854°K[3]	20	-	100	KΩ
	100 lux., 2854°K[3]	-	5	-	
Dark Resistance	10 sec after removal of test light.	20	-	-	MΩ
Spectral Peak	-	-	550	-	nm
Gamma	1-10 Lux	-	0.7	-	-
Gamma	10-100 Lux	-	0.7	-	-

NOTE:
3. Cells light adapted at 30 to 50 Ftc for 16 hrs minimum prior to electrical tests.

Figura: 34 Parámetros del fabricante

3.3.3 Actuadores

- **Bomba Peristáltica Nema17 Stepper Motor:**
  Se escogió este motor para que pueda realizar el trabajo de bomba de precisión, ya que según el fabricante este motor tiene un paso de ángulo de 1.875º permitiendo una dosificación precisa.

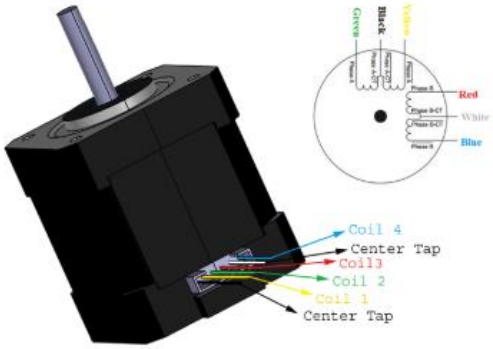


Figura: 35 NEMA 17 Stepper Motor

Para poder realizar el control del motor se utiliza el módulo **L298N**:
Este dispositivo proporciona la corriente necesaria para el control del motor sea DC o Paso-Paso.

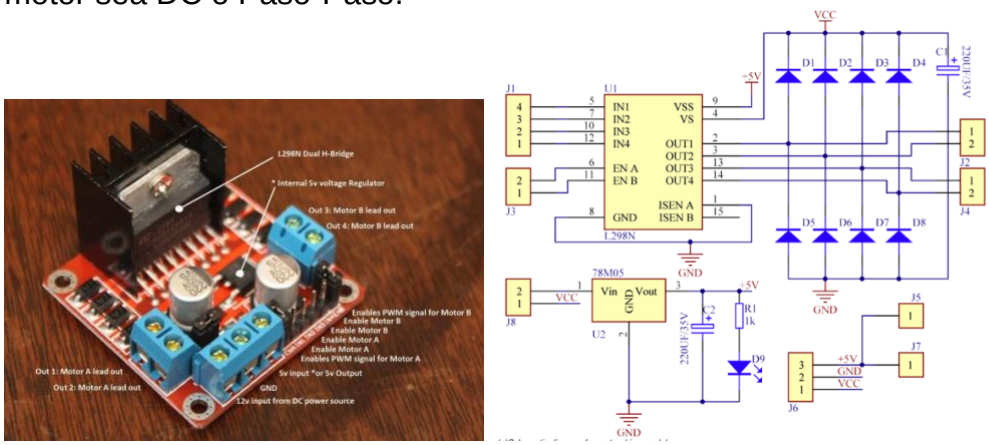


Figura: 36 Driver L298N

- **Ventilador NMB-MAT BG0703-B042-000:**
  Se optó por este ventilador por su bajo consumo 0.16A y tensión de operación 12v.

Figura: 37 Ventilador NMB-MAT BG0703-B041-000

- **Electroválvula KSD 24V DC:**
  El fabricante expone poca información sobre este actuador, por lo que sus características técnicas son obtenidas a través de la práctica real en el laboratorio.

Figura: 38 KSD 24V DC

- **Bomba de agua Sumergible DC 12V**

  Se ha escogido esta bomba debido a la tensión que trabaja 12V.

3.3.4 Circuitos integrados utilizados:

- **Driver ULN2003**
  Está compuesto por 7 Drivers idénticos e independientes entre si. Donde cada driver está constituido por 2 transistores en configuración Darlington. Esta configuración consiste en conectar dos transistores bipolares en cascada obteniendo así una ganancia muy elevada debido a la multiplicación de la ganancia de cada uno de los dos transistores.
  Es útil porque permite el control de un microcontrolador a pequeños relés, motores DC, motores paso a paso, luces de baja tensión o tira de leds.
  Para obtener mayor corriente de salida, se tiene la posibilidad de conectar más de un canal en paralelo teniendo en cuenta que la tierra debe ser común a ambos. Se ha utilizado este driver principalmente porque trabaja bien con tensiones de control de 5V y de 3.3V. Tiene una corriente máxima por canal de 500mA.

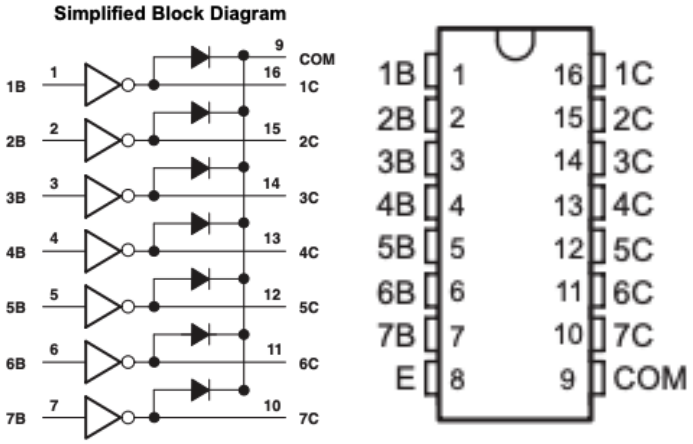


Figura: 39 Diagrama de Bloques ULN2003

- **Amplificador operacional LM324:**
  Este amplificador requiere una alimentación de Vcc+1.5V, debido a la oscilación de voltaje de salida.
  Esto limita la amplificación de la lectura de la tensión de salida de los sensores, porque como máximo se amplificará a 3.5V, si tiene como entrada una tensión de 5V. Por este motivo, se ha modificado la tensión de referencia de nuestro ADC. Además, este AO tiene un condensador de compensación para que sea más estable, y de esta forma evitar las oscilaciones.
  Características principales:
  1. Número de operacionales 4
  2. Voltaje de operación: [3-32]Vdc
  3. Bajo consumo de potencia
  4. Compensado en frecuencia interna

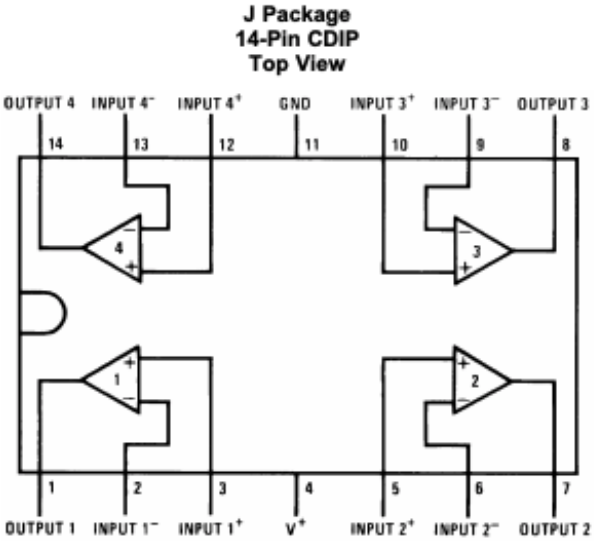


Figura: 40 Operacional LM324

3.4 Acondicionamiento de la señal en los sensores

Debido a que el sensor de temperatura únicamente puede entregar como máximo un valor de 1.5V en su salida, se ha procedido a modificar la tensión de referencia del ADC de la placa ATMEGA2560
Esta modificación se ha realizado mediante software, con la ayuda de la función analogReference() de ARDUINO.

```
analogReference(INTERNAL2V56) //Activa referencia de 2.56V
```

3.4.1 Acondicionamiento LM35Z

Para este acondicionamiento se ha utilizado el amplificador no inversor.

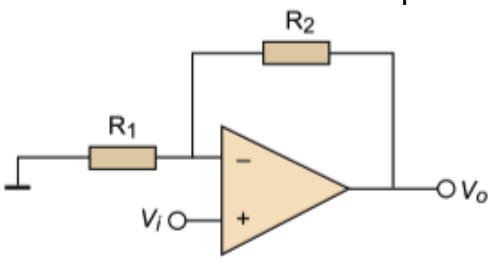


Figura: 41 Amplificador No Inversor

Donde la salida teórica es:

$$V_o = V_{in} \cdot \frac{R_1 + R_2}{R_1}$$

Por lo que a partir de esta ecuación obtenemos el valor de las resistencias R1 y R2.
El requisito que se ha impuesto, es que el ADC leerá 2,56V cuando el LM35Z tenga una salida de 0.5V, lo que representará una temperatura de 50ºC. De esta forma conseguimos aprovechar la resolución máxima de la tensión de fondo de escala del ADC.

El valor de la ganancia G es:

$$G = \frac{V_o}{V_{in}}$$

$$G = \frac{2.56V}{0.5V} = 5.12$$

A partir de estos valores se obtiene el valor de R1 y R2.

$$R_2 = R_1 \cdot 4.12$$

El resultado de la resistencia, de R2 es 4.12 veces mayor que R1.

3.4.1.2 Simulación teórica PSPICE LM35Z

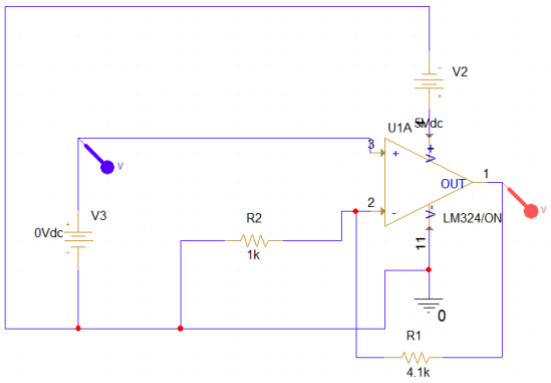
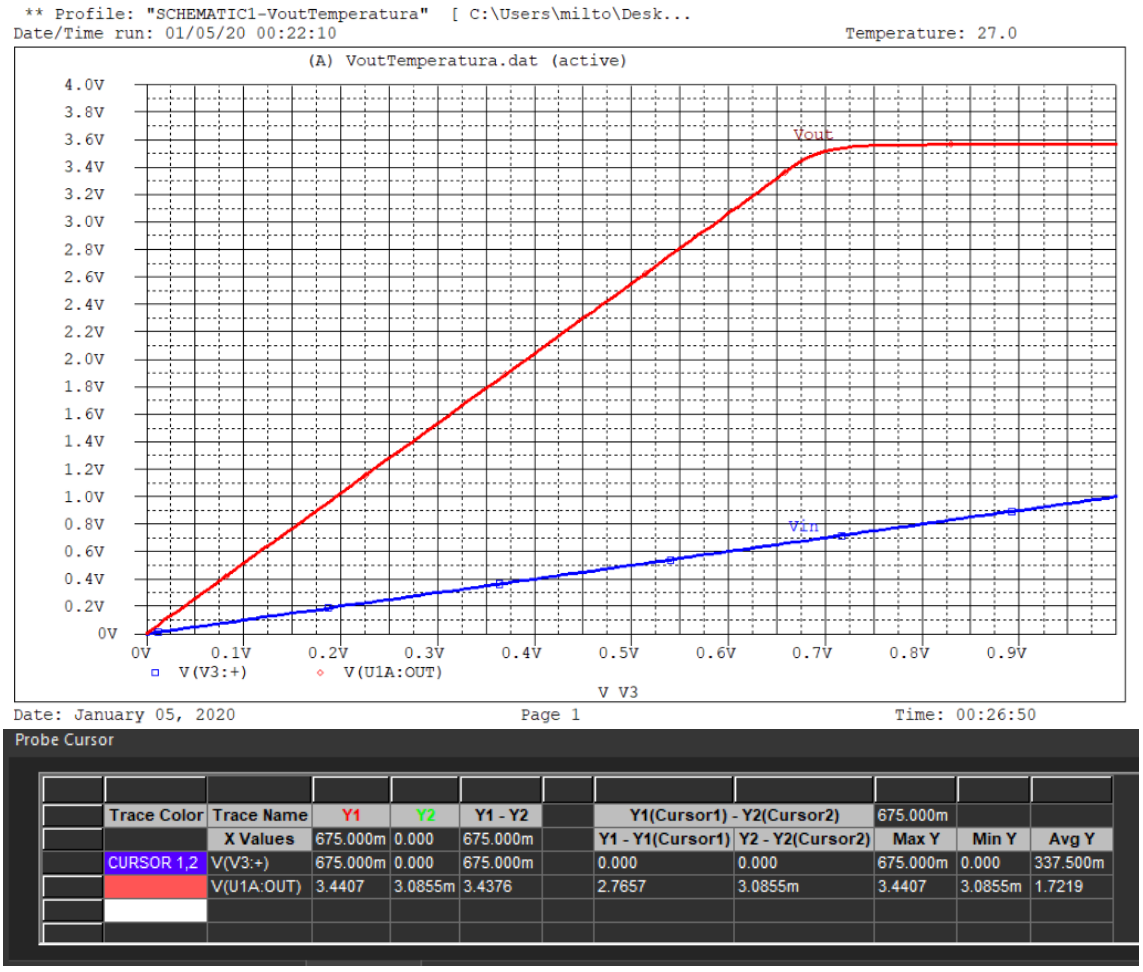


Figura: 42 Circuito del Acondicionamiento LM35Z PSPICE

En el programa se ha fijado una resistencia de R1= 1K y una R2=4.12K.



Probe Cursor

	Trace Color	Trace Name	Y1	Y2	Y1 - Y2		Y1(Cursor1) - Y2(Cursor2)		675.000m		
		X Values	675.000m	0.000	675.000m		Y1 - Y1(Cursor1)	Y2 - Y2(Cursor2)	Max Y	Min Y	Avg Y
	CURSOR 1,2	V(V3:+)	675.000m	0.000	675.000m		0.000	0.000	675.000m	0.000	337.500m
		V(U1A:OUT)	3.4407	3.0855m	3.4376		2.7657	3.0855m	3.4407	3.0855m	1.7219

Figura: 43 Salida Circuito acondicionador sensor LM35Z

En la gráfica se aprecia que la salida máxima del sensor que podrá ser leída por el operacional hasta antes que se sature es 675mV, a partir de este valor no se

verá ningún cambio en la salida. Es decir que como máximo se podrá leer 67.5ºC.

3.4.1.3 Cálculo del valor correcto en software del LM35Z

En la siguiente Figura 42 se demuestra como se obtiene el valor de conversión del ADC y como se debe de convertir mediante software para realizar una correcta lectura del sensor de temperatura, cuyo valor es de 30ºC.

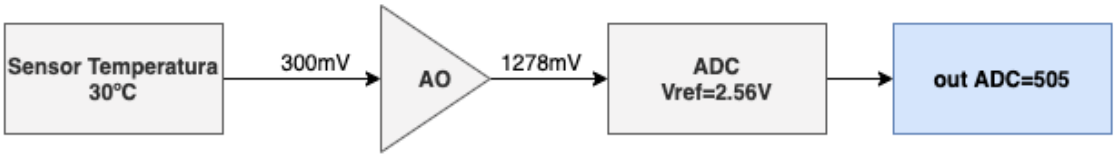


Figura: 44 Diagrama de bloques de la palabra convertida

$$Palabra = \frac{outADC}{TºC} = 16.83$$

$$Temperatura\ real\ = \frac{outADC}{16.83}$$

Con este cálculo realizado, cualquier valor que el ADC convierta será divido entre 16.83 dando lugar al valor correcto de lectura de temperatura por la pantalla LCD.

3.4.2 Acondicionamiento HR

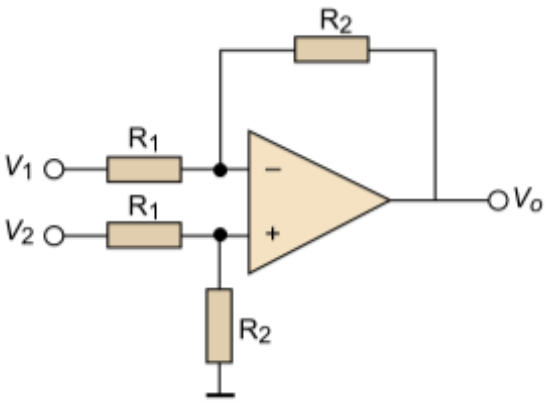


Figura: 45 Amplificador Diferencial

Donde la salida es:

$$V_O = \frac{R_2}{R_1} \cdot (V_2 - V_1)$$

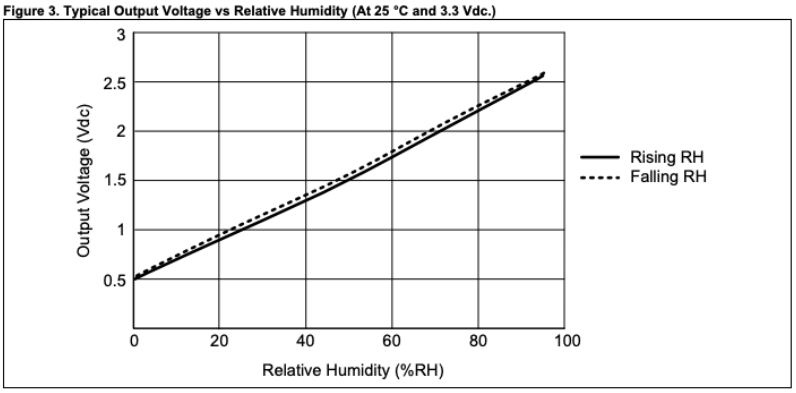


Figura: 46 Salida en tensión vs HR en %

Según el datasheet del fabricante Figura: 46 del sensor tenemos que a una salida de 0.5V, siendo este valor como 0% de HR, mientras que a 2.6V tenemos una salida de aproximadamente un 93% de HR.
Con esta información procedemos a encontrar el valor en tensión para el 100% de HR.

$$V_{out_{100\%HR}} = \frac{2.6V \cdot 100\%}{93\%} = 2.8V$$

Por lo tanto, cuando el sensor marca la tensión de 0.5V el circuito acondicionador tendrá como salida un 0V, mientras cuando el circuito acondicionador tenga una salida de 2.8V representará un 100% de HR.
Para trabaja en el rango de [0-2.56]V, creamos el siguiente circuito acondicionador.

1. **Divisor de tensión:**

$$V_{sensor0\%HR} = V_{in3.3} \cdot \frac{R_2}{R_1 + R_2}$$

Aplicando $V_{sensor0\%HR} = 0.5$ y $V_{in3.3} = 3.3V$ encontramos el valor de las resistencias.

Tenemos que para esta etapa R1=5.7·R2.

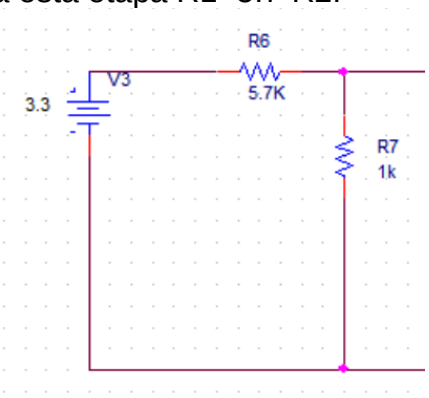


Figura: 47 Divisor de tensión

2. **Seguidor de tensión en serie**
Se coloca un seguidor de tensión en serie para que la carga no afecte a la salida del divisor de tensión.

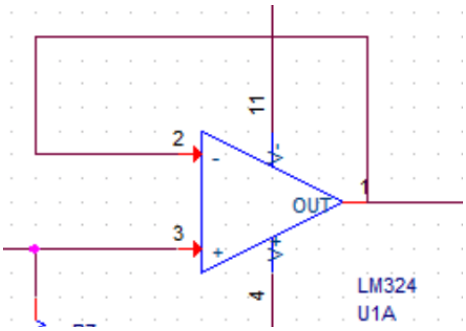


Figura: 48 Segudior de Tensión

3. Amplificador diferencial

En la salida del circuito se obtienen el siguiente cálculo:

$$V_O = (V_{outSensor} - V_{sensor0\%HR}) \cdot \frac{R_4}{R_3}$$

$$G = \frac{V_O}{V_{outSensor} - V_{sensor0\%HR}} = \frac{2.56}{2.8 - 0.5} = 128/115 = 1.11$$

La relación de resistencia nos queda R4= R3·1.11.

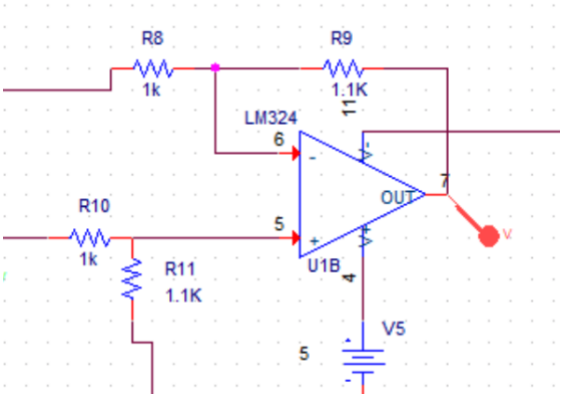


Figura: 49 Amplificador Diferencial

La información obtenida se dirigirá a uno de los canales del multiplexor del microcontrolador para poder conectarse al mismo ADC donde se encuentra conectado el sensor de temperatura.

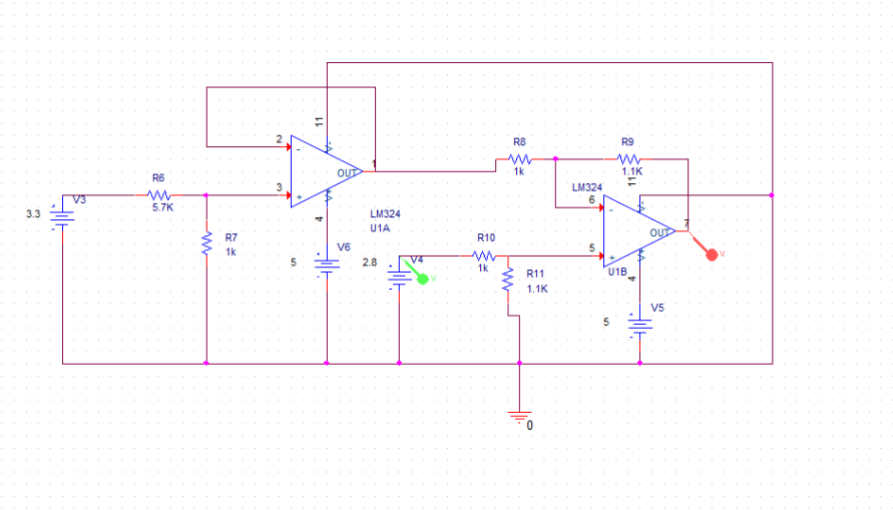


Figura: 50 Circuito del Acondicionamiento HIH PSPICE

3.4.2.1 Resultados simulación sensor HR

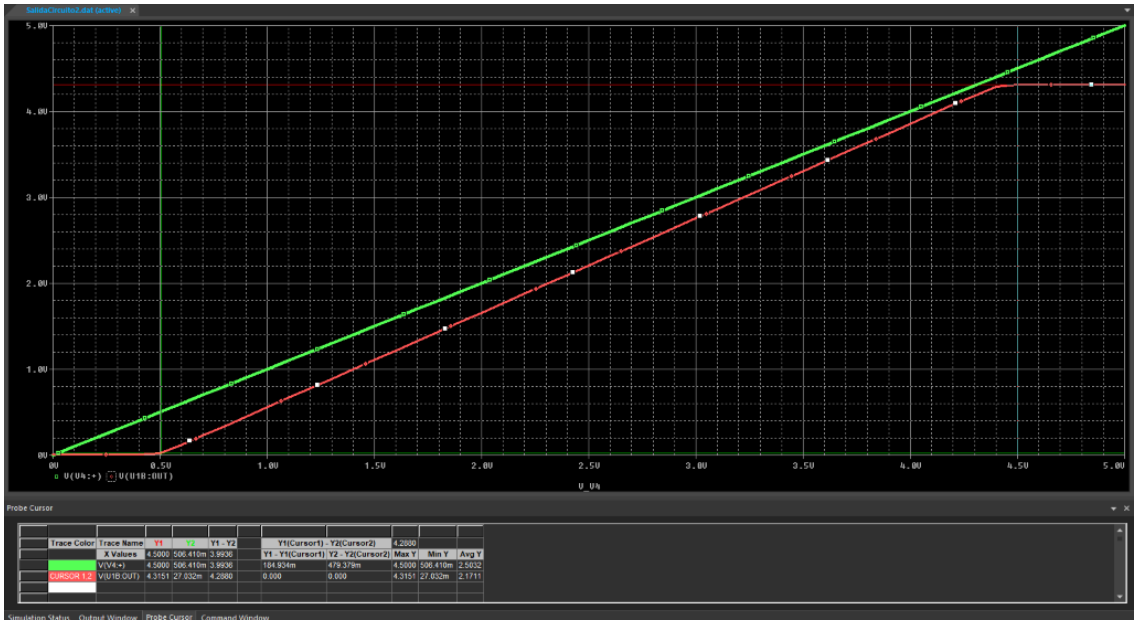


Figura: 51 Simulación Acondicionador de señal HR

En la Figura: 51 vemos que el operacional LM324N no puede dar más tensión de salida a partir de 4.3V.

El fabricante indica que el LM324N tiene una salida Vout=Vcc-1.5V, cosa que se comprueba en la práctica cuando se realiza el montaje del circuito.
Es importante tener en cuenta las especificaciones del fabricante, porque en este caso la simulación no es fiable al 100%, por lo que siempre es recomendable antes realizar pruebas físicas.

3.4.2.2 Resultados prácticos del sensor HR

Cuando se ha intentado reproducir el circuito físicamente, han existido dificultades debido a la no exactitud de los componentes teniendo siempre una

tensión a la salida del circuito acondicionador. Cuando debería de leer un 0V. Aparece un valor de 400mV a la salida del circuito acondicionador, teniendo el sensor de temperatura una salida de hasta 900mV, para que el circuito comience a trabajar. Esto es erróneo, ya que en los cálculos matemáticos realizados se observa como a partir de 0.5V de la salida del sensor el acondicionador debe variar su salida.
Este error es debido al voltaje de offset o voltaje de desvío del operacional que se sitúa entorno de 20 a 26mV.

Por lo tanto para corregir este error alimentamos con una tensión de -5V, que alimenta la parte negativa del operacional. El objetivo es que se cumpla, que cuando la salida del sensor es 0.5V, la salida del circuito acondicionador debe situarse en 0V.

Figura: 52 Testing HR =0.5V Vout=0V

Se comprueba que cuando la salida del sensor es 0V, la salida del circuito acondicionador es -0.5V.

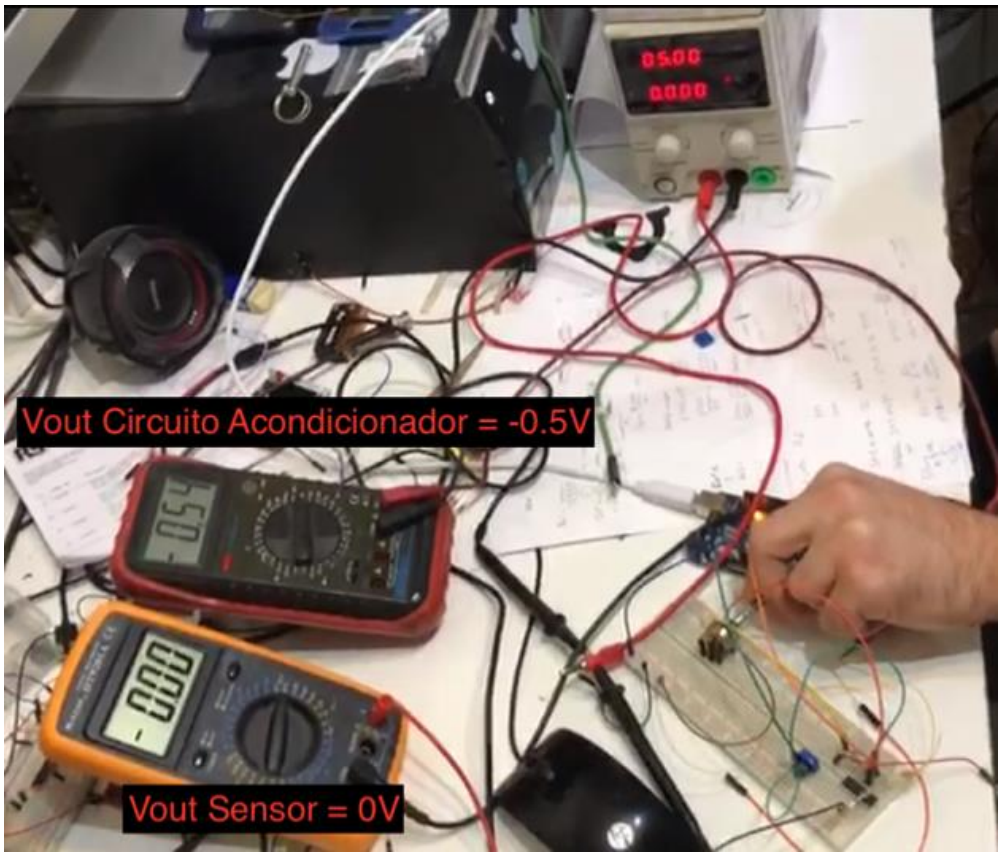


Figura: 53 Testing HR =0V Vout=-0.5V

La máxima salida del sensor es de 2.8V, por lo tanto, la máxima salida del circuito acondicionador debe de ser 2.56V que es la tensión de referencia del ADC.
Nos ayudamos de un potenciómetro para simular los posibles valores del sensor de humedad relativa.

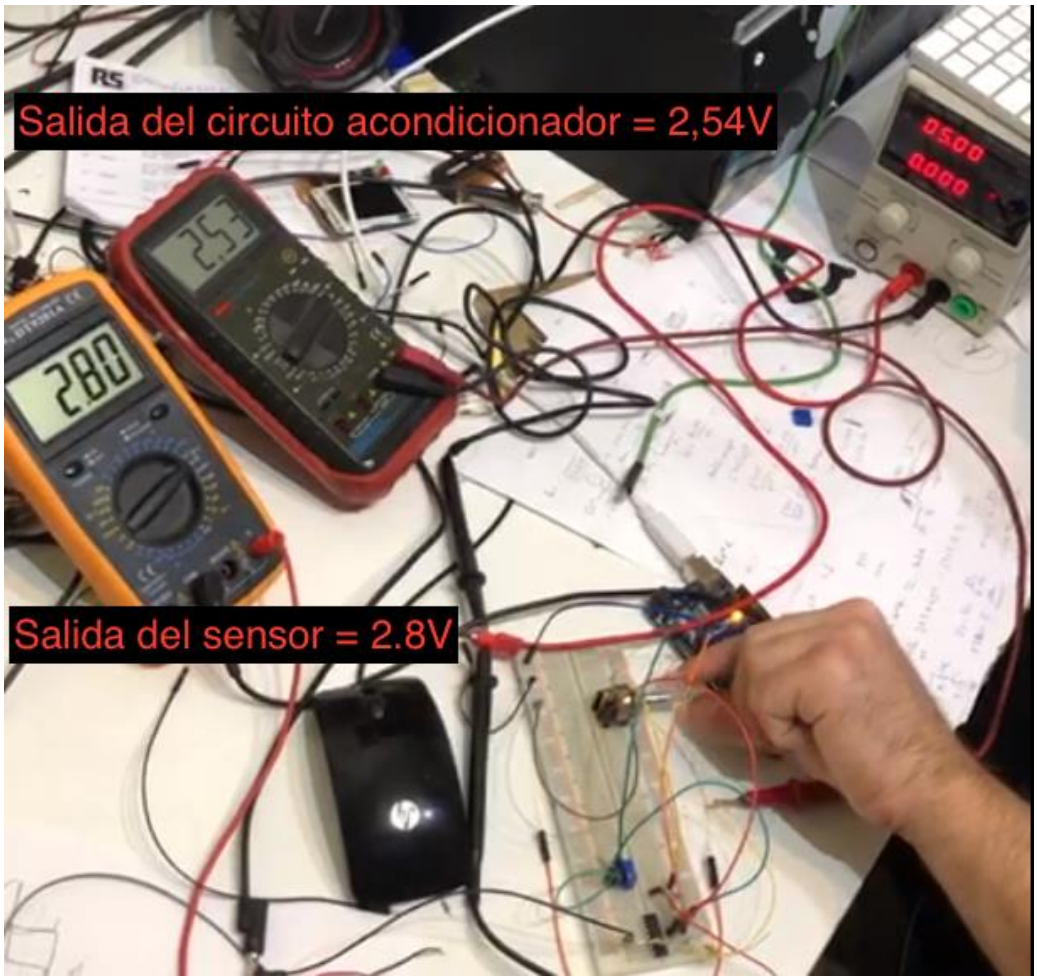


Figura: 54Testing HR =2.8V Vout=2.53V

Por lo tanto, se comprueba que para que funcione el circuito en unas condiciones aceptables, se debe de alimentar con una tensión negativa el LM324N, cosa que en las simulaciones de PSPICE no sucedía.

Por otra parte, comentar para poder realizar la calibración de este sensor, nos hemos basado en las ecuaciones que nos brinda el fabricantes y como referencia la humedad relativa que nos ofrece la web del clima atmosférico de la zona en donde se están realizando las pruebas.

$$V_{out} = (VSUPPLY) \cdot (0.00636(sensor\ RH) + 0.1515)$$

$$CompensaciónTemperaturaTrueRH = \frac{SensorRH}{1.0546 - 0.00216 \cdot T\text{ºC}}$$

Tabla 1 Resultado teórico y práctico sensor HR

Vout	True RH	Vssup	Sensor RH	Temepratura
2,07405	**74,955**	3,3V	75%	25ºC

Además, una vez obtenido este resultado se ha colocado un seguidor de tensión adicional a la salida del sensor HR debido a que la carga del circuito afecta a la salida del sensor.

3.4.3 Acondicionamiento LDR NSN-19M51

Para poder medir valores con el sensor LDR se ha realizado un divisor de tensión, con la siguiente conexión pull-down:

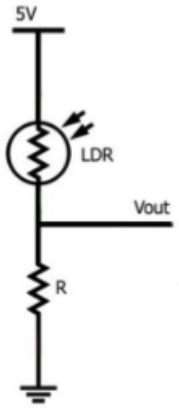


Figura: 55 Configuración Pull-Down

A la salida del divisor de tensión se ha colocado un condensador de 100uF con la finalidad de realizar un filtrado del ruido. Con esto se consiguió la estabilización la lectura del sensor.

3.4.5 Esquemático de la etapa de acondicionamiento de sensores

En la siguiente figura, se observa el circuito acondicionador con las modificaciones finales tras realizar las pruebas teóricas y físicas en laboratorio. Hay que destacar que se ha agregado un seguidor de tensión a la salida del sensor HR, ya que en laboratorio se comprueba que la carga de las resistencias afecta a la tensión de salida del sensor, provocando errores de lectura.

Hay que comentar que al no encontrarse el dispositivo exacto en PSPICE del sensor LDR, ya que éste debe ser creado desde cero. Se ha procedido a colocar uno a modo ilustrativo otro sensor denominado **LDRdummy**.

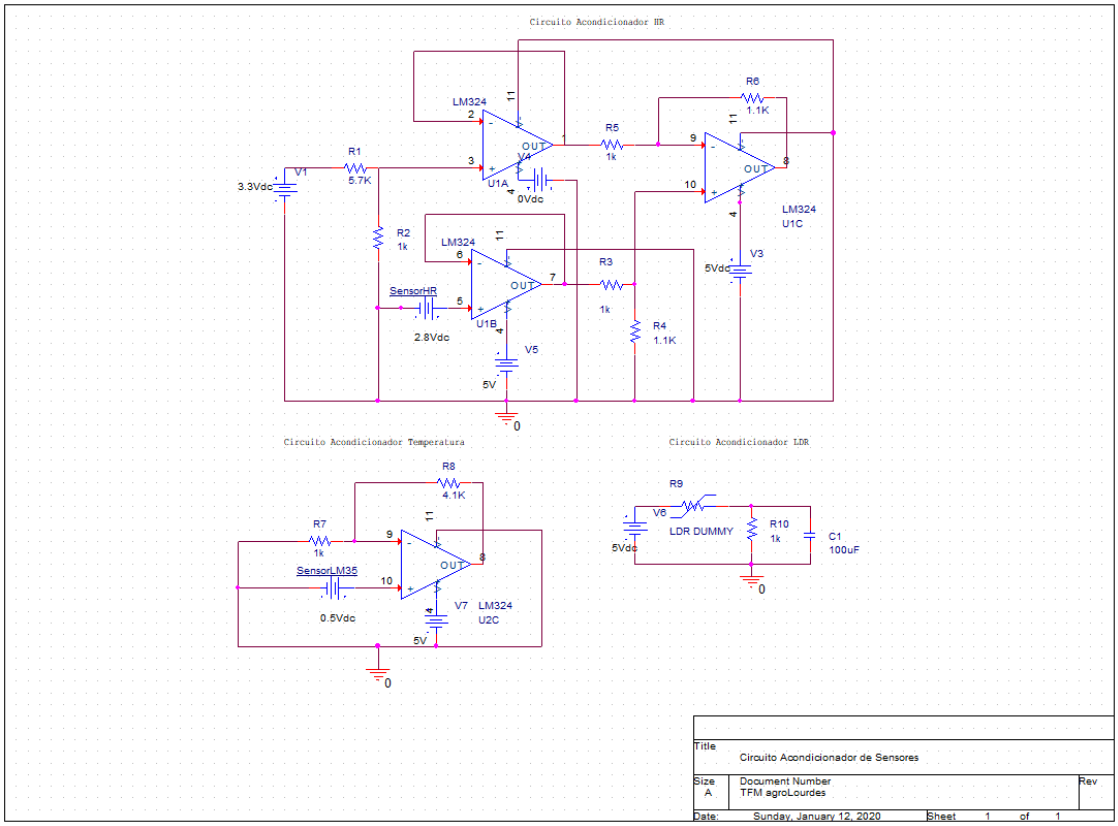


Figura: 56 Circuito de acondicionamiento de los sensores

3.5 Simulación del comportamiento del actuador Electroválvula

3.5.1 Evaluación del impacto de la electroválvula en el driver ULN2003A

Para poder realizar esta simulación, se ha tenido que obtener los valores de la resistencia interna de la electroválvula junto a su inductancia de manera práctica ya que no se ha podido localizar el datasheet del fabricante.

Hemos obtenido la resistencia parásita conectando la fuente de alimentación a 24 voltios donde se observó un consumo de 0.276 A.

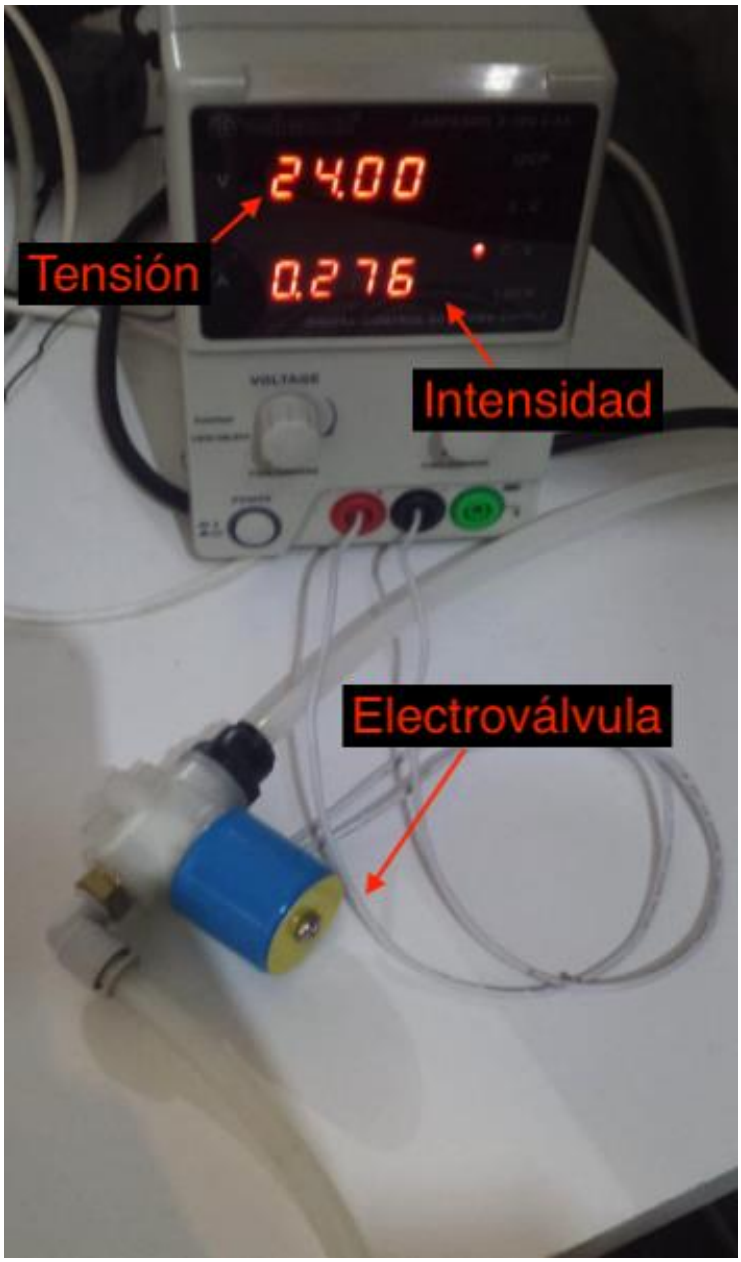


Figura: 57Ley de Ohm en electroválcula

Con este valor hemos aplicado la ley de Ohm:

$$V = R \cdot I$$

Donde R es:

$$R = \frac{V}{I} = \frac{24}{0.276} = 87.0\ \Omega$$

Para encontrar el valor de la inductancia se utilizó la fórmula de los filtros RL:

$$F_c = \frac{R}{2\pi \cdot L}$$

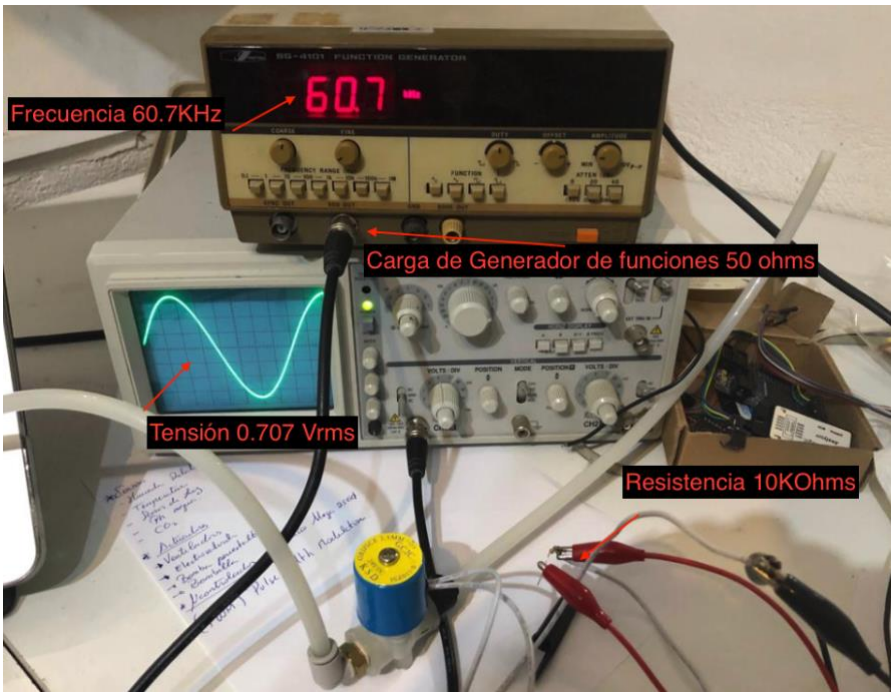


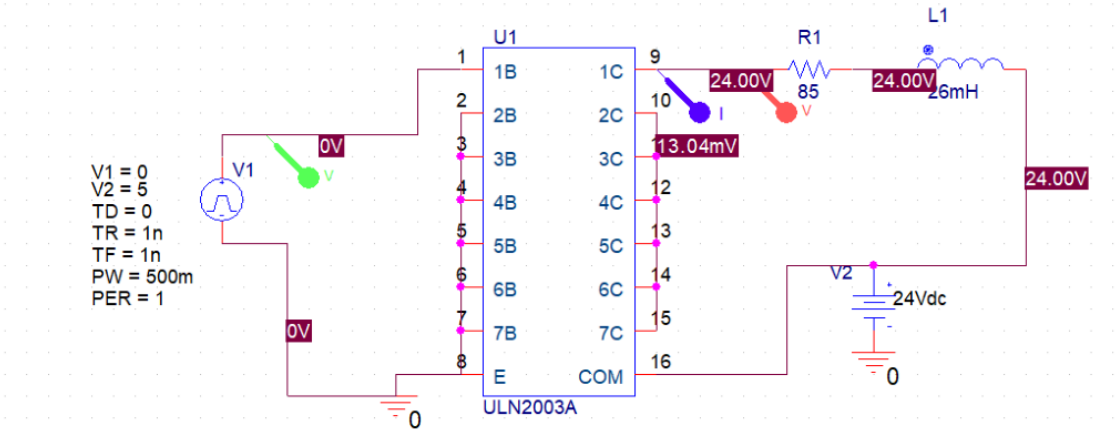
Figura: 58 Cálculo del valor de la inductancia con la frecuencia de corte

Se ha utilizado una resistencia de 10KOhms con la finalidad de despreciar la resistencia de 50 Ohms del generador de funciones y la parásita de 87 Ohms.

En el generador de funciones se ha modificado la tensión a 1v y variado su frecuencia hasta obtener el en osciloscopio una tensión $\sqrt{2}/2$, ya que esta es la frecuencia de corte donde cae 3dB.
De esta forma se ha obtenido la inductancia, cuyo valor es 26mH.

$$L = \frac{R}{2\pi \cdot L} = \frac{50 + 87 + 10000}{2\pi \cdot 60.7 \cdot 10^3} = 26.6mH$$

3.5.1.1 Resultado de la simulación teórica Electroválvula



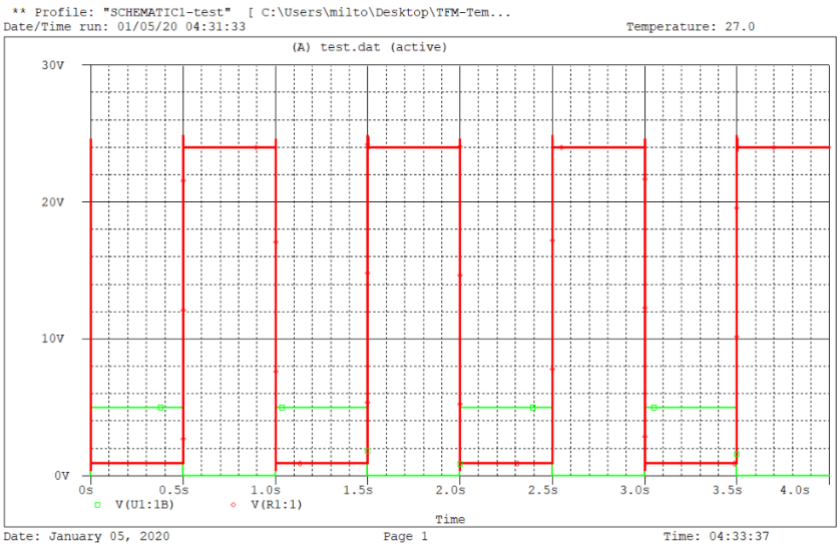
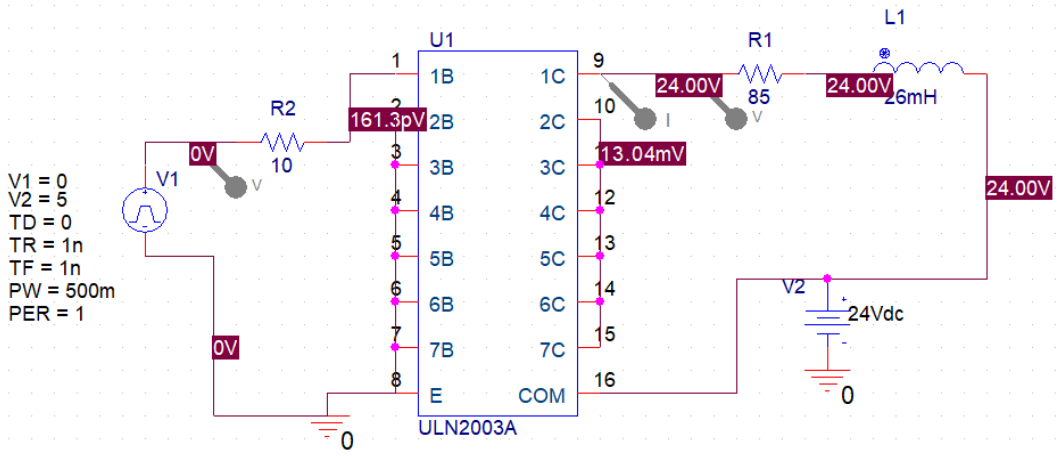


Figura: 59 Salida en tensión Driver ULN2003A

En la Figura: 59 se aprecia la caída de tensión del colector emisor de la celda es de 1v con una intensidad de 300mA, esto implica una potencia disipada de 0.3W, siendo un valor pequeño. Con esto comprobamos que el circuito no tendrá problemas de sobrecalentamiento, ya que la electroválvula es el dispositivo con mayor consumo del sistema.

Ahora procedemos a realizar la evaluación del impacto de la electroválvula en el driver ULN2003A con una resistencia en la entrada R2=10 Ohm.



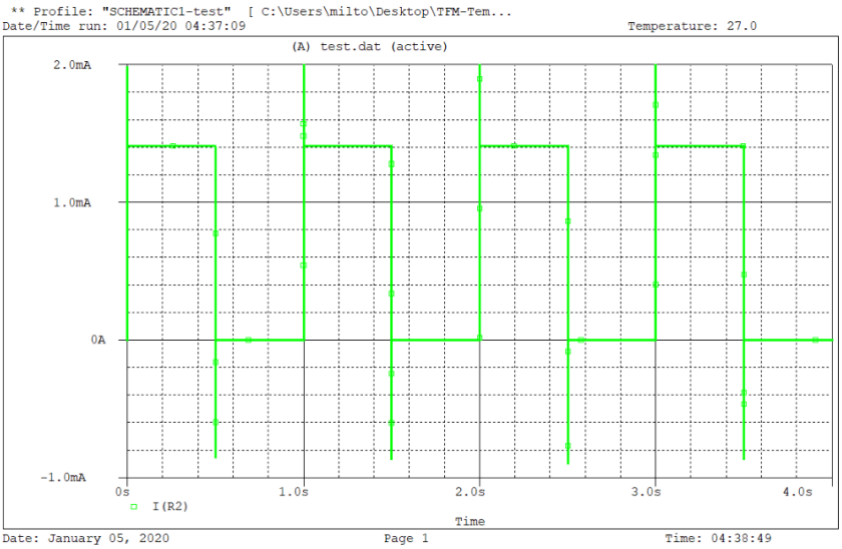


Figura: 60 Intensidad en Carga R=10 Ohms

Se comprueba que la intensidad que cruza por el cable a una carga de 10 ohms es de 2mA, siendo este valor muy por debajo del máximo tolerado por el pin de entrada/salida del microcontrolador 40mA según las especificaciones del datasheet del fabricante del microcontrolador ATMEGA2560.

31. Electrical Characteristics

Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0mA
DC Current V_{CC} and GND Pins	200.0mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Figura: 61 Intensidad máxima tolerada en pin E/S Arduino según fabricante

3.5.1.2 Justificación de la inductancia de la electroválcula en PSPICE

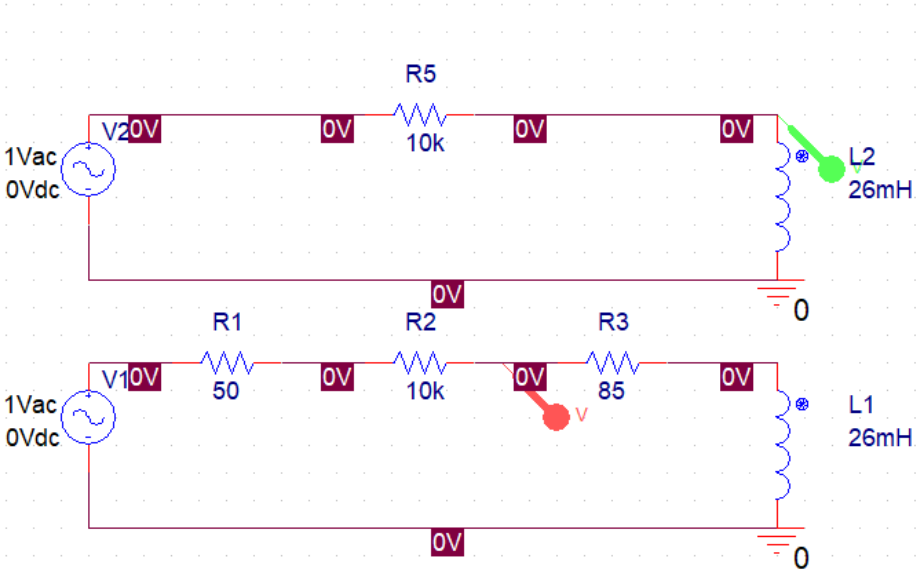


Figura: 62 Evaluación de 2 tipos de cargas

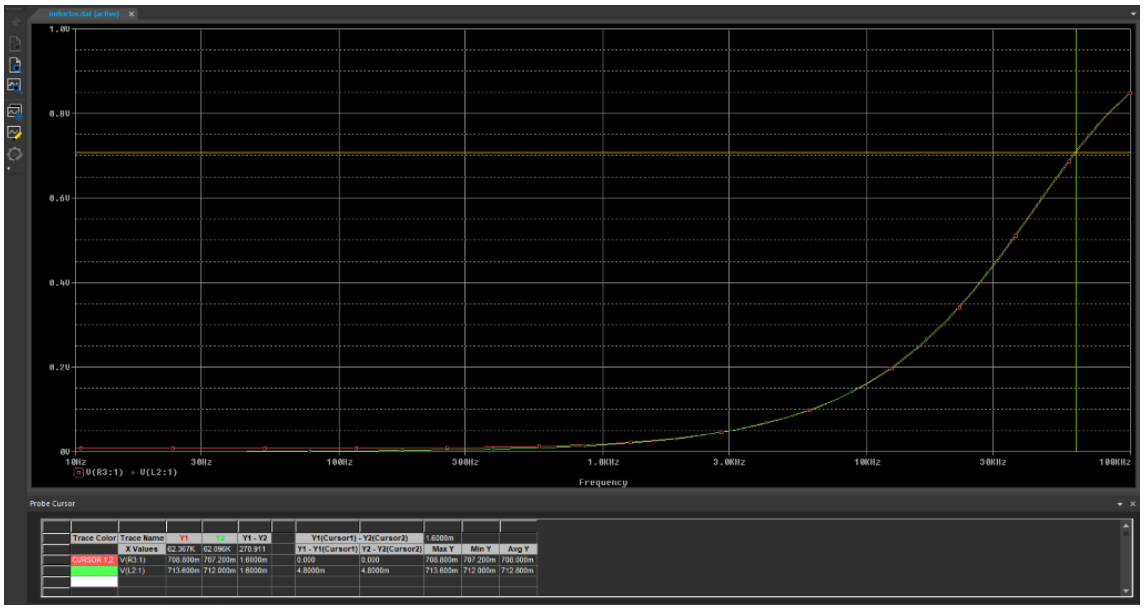


Figura: 63 Salida en tensión en 2 casos: 1) Resistencia de 10K en serie con 50 y 85 Ohms. 2) Resistencia de 10K

Tras haber realizado los cálculos en la práctica, hemos procedido a realizar simulaciones teóricas teniendo en cuenta 2 escenarios:

1. Resistencia 10K en serio con 50 y 85
2. Resistencia de 10K

Comprobamos que el resultado práctico realizado, es coherente con el teórico. Tanto en el caso 1 como en el 2 tenemos resultados muy similares Figura: 63, pero debido a los errores de lectura en la parte práctica, en la simulación teórica tenemos una frecuencia de corte de 62.47KHz para el primer caso y luego una

frecuencia de corte 62.09KHz para el segundo caso. Pero en definitiva la inductancia calculada es prácticamente la misma de 26mH.

3.6 Diseño software y Librerias

La siguiente Figura: 64 muestra el diagrama de flujo del sistema.

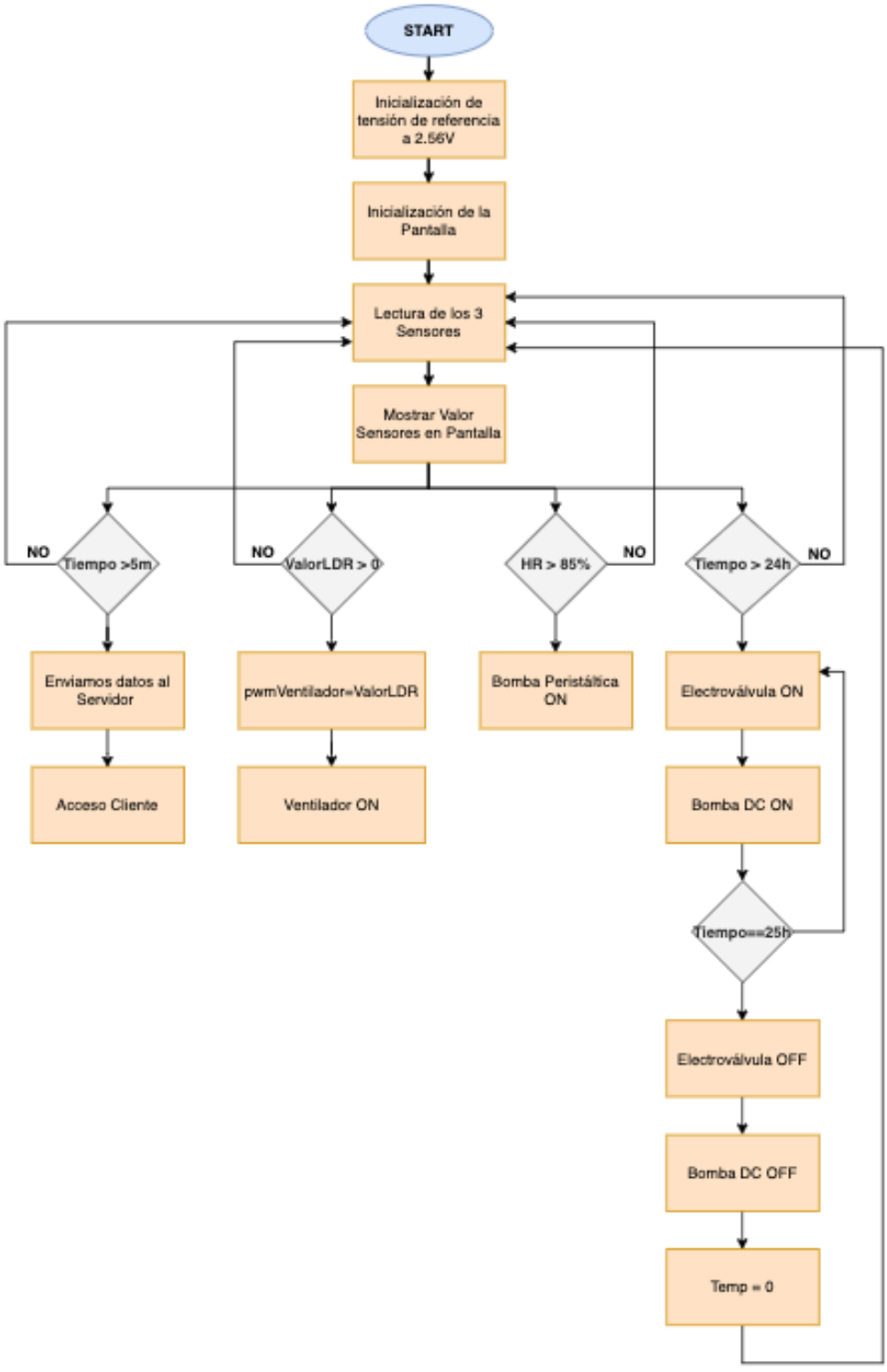


Figura: 64 Diagrama de Flujo del software del sistema

El desarrollo del código se ha divido en 3 partes:

1.- Firmware de los sensores y escudo WiFi con lenguae C++.

2.- Software para almacenar la informaicón en la base de datos, con lenguaje .php y SQL.

3.- Software para la visualización de los datos en una tabla y gráfica con HTML, PHP,JSON**.**

3.6.1 Firmware de los sensores y escudo WiFi con lenguae C++

En la siguiente Figura: 65 tenemos el diagrama UML “Lenguaje Unificado Modelado” de clases utilizadas en el firmware del microcontrolador.

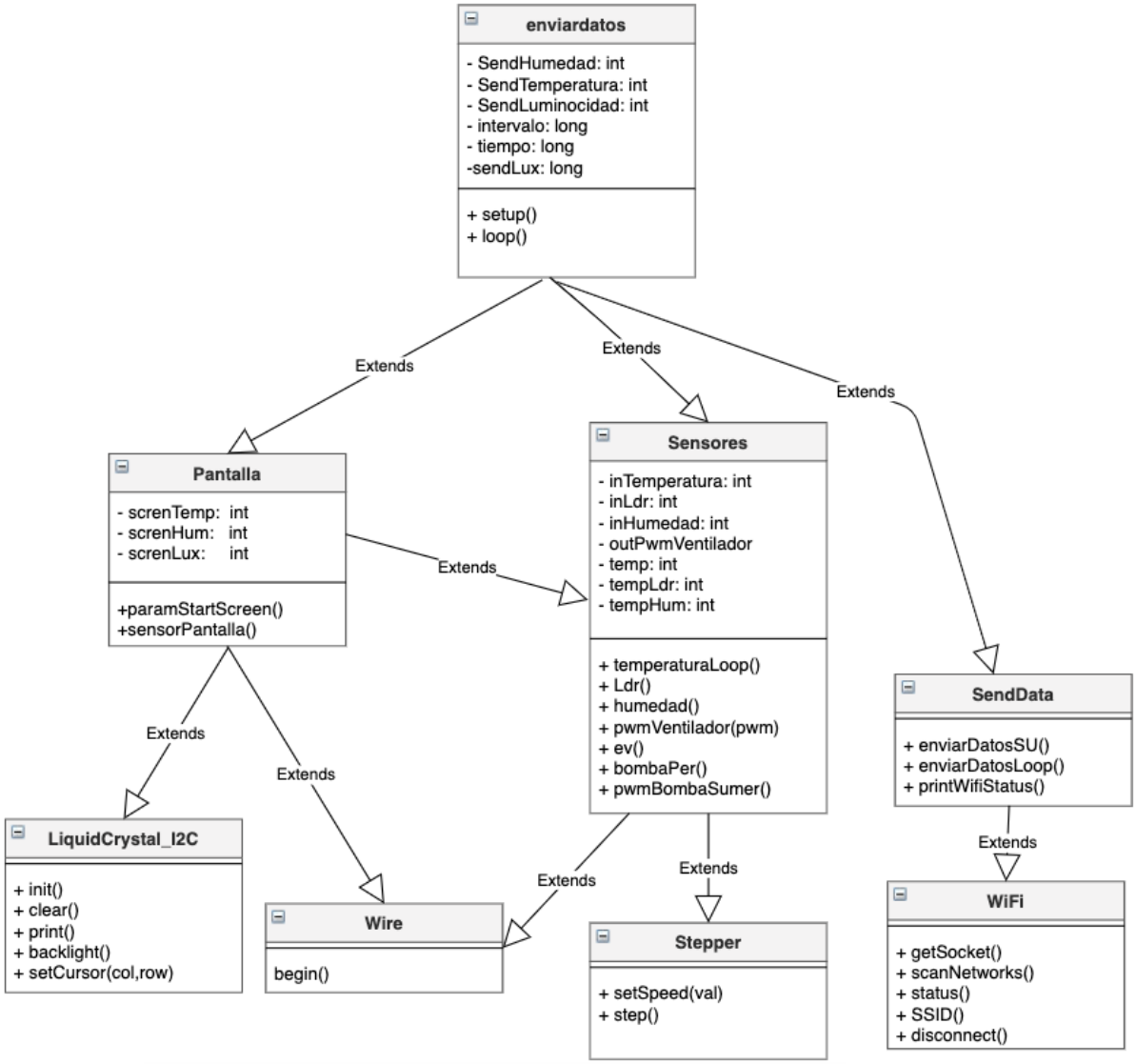


Figura: 65 Diagrama UML de clases de los ficheros de comunicaicones y sensores

A continuación, se detallan algunas de las funciones más relevantes del proyecto:

1. **analogReference(INTERNAL2V56)**

Esta línea de código se encarga de cambiar la tensión de referencia del ADC a un valor de 2.56V, y se encuentra en la clase pricipal del proyecto enviardatos, dentro del método **setup().**

2. **Float EnviarDatosSU(int Temperatura, int Humedad, int Luminocidad)**
   Este método es el encargado de enviar la información a la BBDD luego de una espera de 5m. Es llamado dentro de la clase principal del proyecto **enviardatos**, dentro del método **loop().**
3. **Millis()**
   Es una función quese utiliza para medir tiempo. Esta instrucción da el tiempo en milisegundos desde que se enciende por primera vez la placa Arduino Mega.
   Se utiliza dentro de la clase principal **enviardatos** y se encuentra dentro del método **loop().**

3.6.2 Software para alamcenar la infomración en la base de datos con .php y SQL

En la siguiente Figura: 66 tenemos el diagrama UML de la única clase en .php que se encarga de realizar la conexión a la base de datos y almacenar los datos recibidos de los sensores por el método **enviarDatosSU()** de la clase **SendData** del microcontrolador .

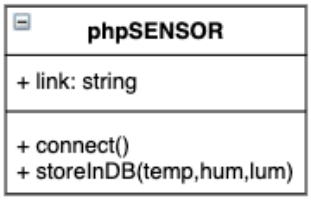


Figura: 66 Clase principal en .php para envío de información a la BBDD

A continuación se comenta las dos funciones,:

1. **Connect()**
   Dentro de este método se encuentra la función **mysqli_connect()** que permite abrir una conexión con el servidor MySQL.
2. **storeInDB()**
   Dentro de este método se encuentra **mysqli_query()** que se encarga de realizar un insert en la base de datos con la información de los sensores.

Previamente, en el servidor Xampp se ha creado una tabla con las diferentes columnas en la base de datos MySQL, en donde se almacena los datos recogido de los sensores.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	ID	bigint(20)			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
2	humidity	float			Sí	*NULL*			Cambiar Eliminar Más
3	temperature	float			Sí	*NULL*			Cambiar Eliminar Más
4	Luminocidad	int(11)			No	*Ninguna*			Cambiar Eliminar Más
5	DATE	timestamp			No	current_timestamp()			Cambiar Eliminar Más

Figura: 67 Tabla BBDD que almacena la información de los sensores

La información se almacena en la BBDD mediante el localizador de recursos uniformes “URL”.
Esto se consigue gracias al método GET donde el símbolo ‘?’ indican los parámetros que se reciben desde el formulario que ha enviado datos al servidor. Luego del símbolo ‘?’ aparece el valor de los parámetros y el símbolo ‘&’ que concatena los parámetros.

Un posible ejemplo es el siguiente:

http://localhost/testcode/dht.php?temperature=12&humidity=10&Luminocidad=120

3.6.3 Software para la visualización de los datos en una tabla y gráfica con HTML, PHP y JSON.

Una vez almacenado la información en la BBDD, se crea una vista con HTML y PHP, patra que el cliente pueda ver la información en una tabla.
El código se adjunta en el anexo del informe.

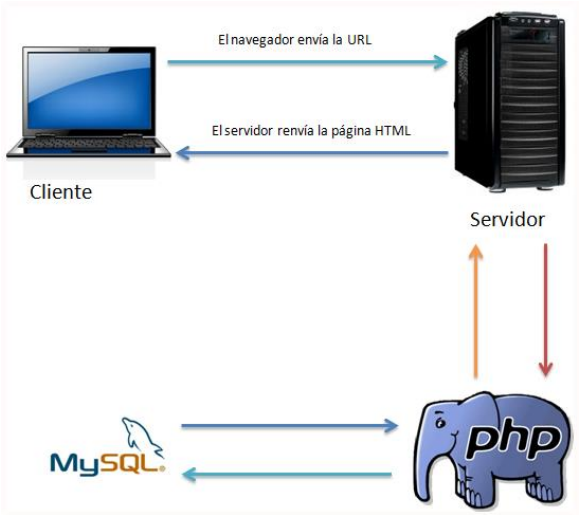
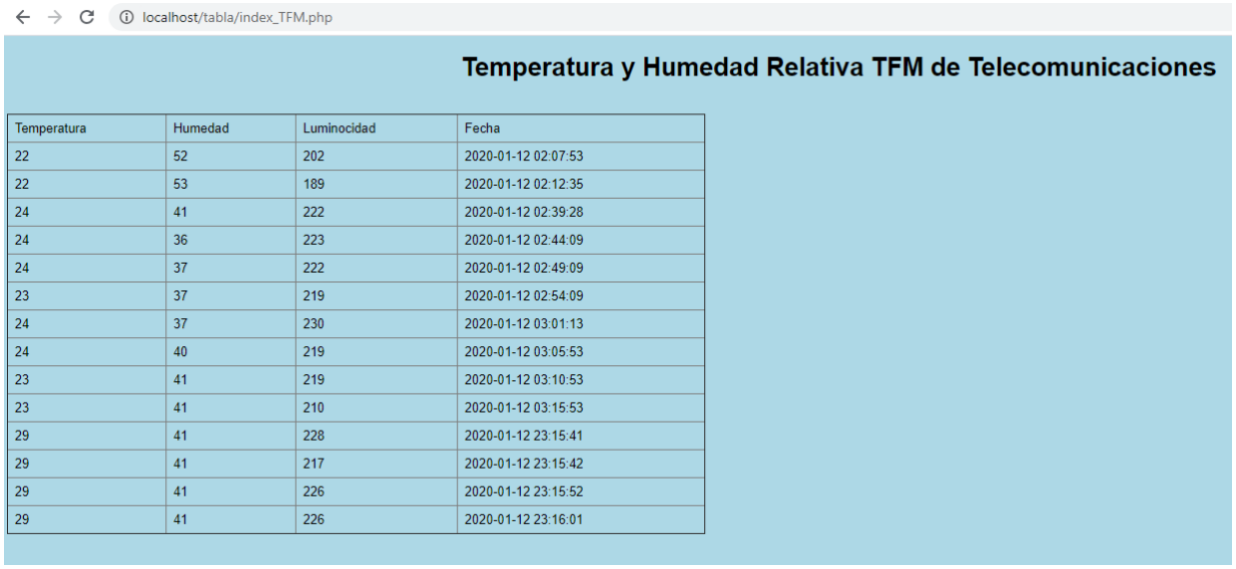


Figura: 68 Configuración BBDD y sitio web



Temperatura	Humedad	Luminocidad	Fecha
22	52	202	2020-01-12 02:07:53
22	53	189	2020-01-12 02:12:35
24	41	222	2020-01-12 02:39:28
24	36	223	2020-01-12 02:44:09
24	37	222	2020-01-12 02:49:09
23	37	219	2020-01-12 02:54:09
24	37	230	2020-01-12 03:01:13
24	40	219	2020-01-12 03:05:53
23	41	219	2020-01-12 03:10:53
23	41	210	2020-01-12 03:15:53
29	41	228	2020-01-12 23:15:41
29	41	217	2020-01-12 23:15:42
29	41	226	2020-01-12 23:15:52
29	41	226	2020-01-12 23:16:01

Figura: 69 Tabla Temepratura y Humedad Relativa

Con la ayuda de las librerías Plotly.js se ha creado otra vista que permite graficar los datos almacenados de la BBDD.

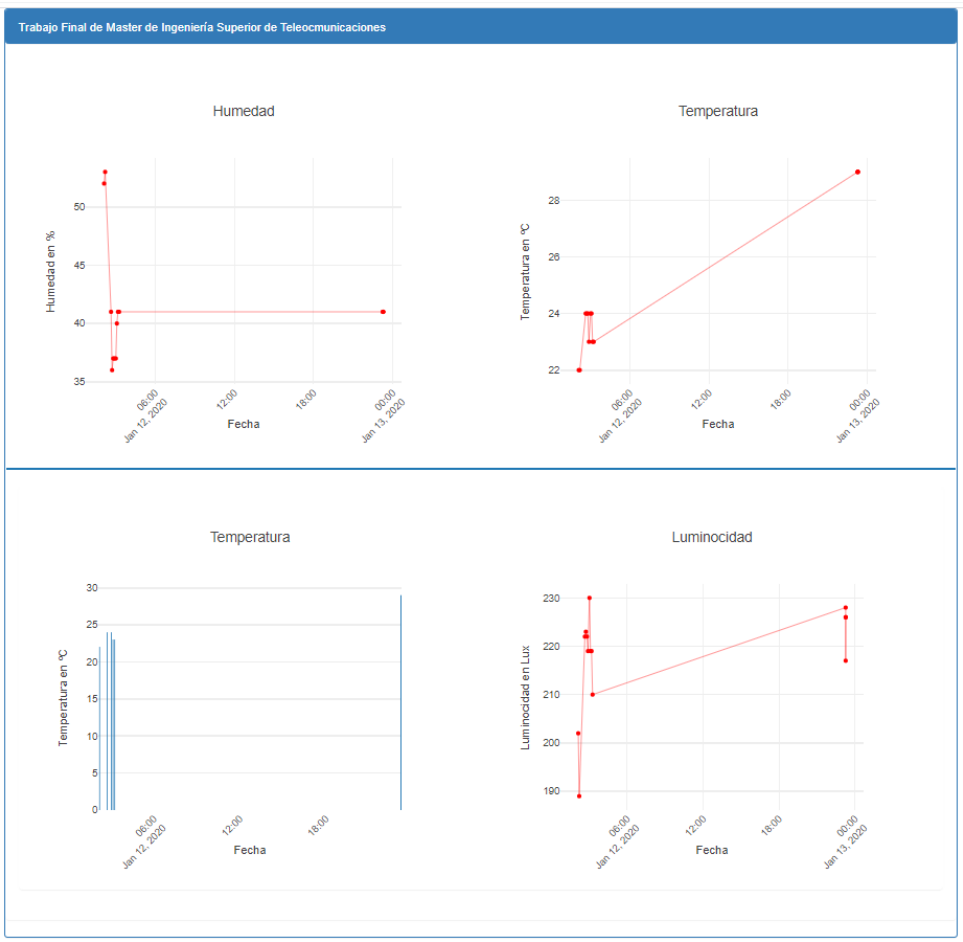


Figura: 70 Gráfico lineal y de Barras para los parámetros almacenados de la BBDD

En la Figura: 71 se observa el diagrama de clases en UML que permite realizar gráficas.

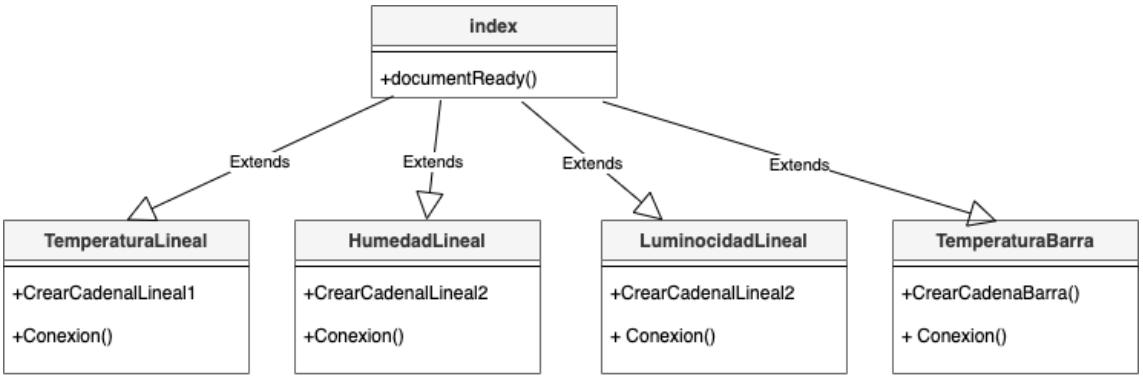


Figura: 71 Diagrama de clase UML PLOTLY.js

4 Implementación del sistema

En este capítulo se comenta la construcción del sistema tanto a nivel de software como hardware. Además, se realiza una prueba de evaluación del sistema para corroborar su correcto funcionamiento mediante una prueba cíclica de funcionamiento.

En la siguiente Figura: 72 se observa el sistema completo, donde se encuentra el circuito del microcontrolador alimentado a una tensión de 5V y la parte de los actuadores a una tensión de 12V.

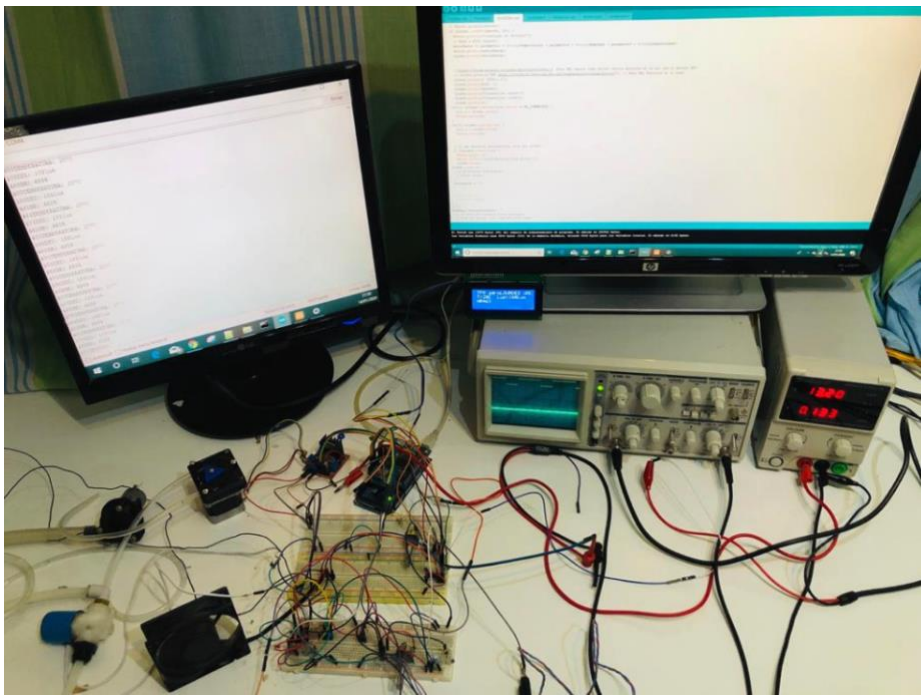


Figura: 72 Sistema Completo con Fuente de alimentación y osciloscopio

En la siguiente Figura: 73 se aprecian dos placas de desarrollo, en donde a la izquierda tenemos sensores y su circuito de acondicionamiento.
En la derecha se aprecia el driver ULN2003 que controla los actuadores.
Hay que comentar que, en los cálculos matemáticos para la etapa de acondicionamiento de las señales de salida de los sensores, existen resistencias que físicamente no se encuentran, por lo tanto, se han creado mediante la suma de resistencias en seria obteniendo la resistencia equivalente a la que se requería en los cálculos. Por ejemplo, en el caso de las resistencias de 4.1K y 1.1K, donde se ha utilizado 4 resistencias de 1k en serie con una resistencia de 100 Ohm y en el otro caso se ha utilizado una resistencia de 1K en serie con una resistencia de 100 Ohm.

Además, se ha colocado en la salida del sensor de luminosidad un condensador para estabilizar su señal, ya que de esta forma conseguimos reducir el ruido.

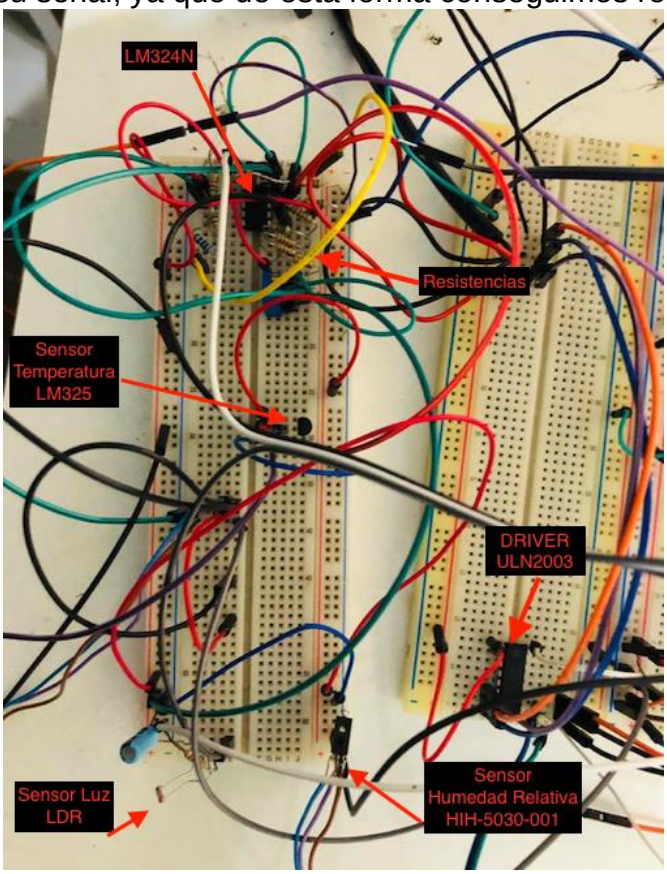


Figura: 73 Placa con sensores y Placa con Drive ULN2003

En la siguiente Figura: 74tenemos el microcontrolador junto con el escudo WiFi más el controlador del motor paso paso de la bomba peristáltica L298N.

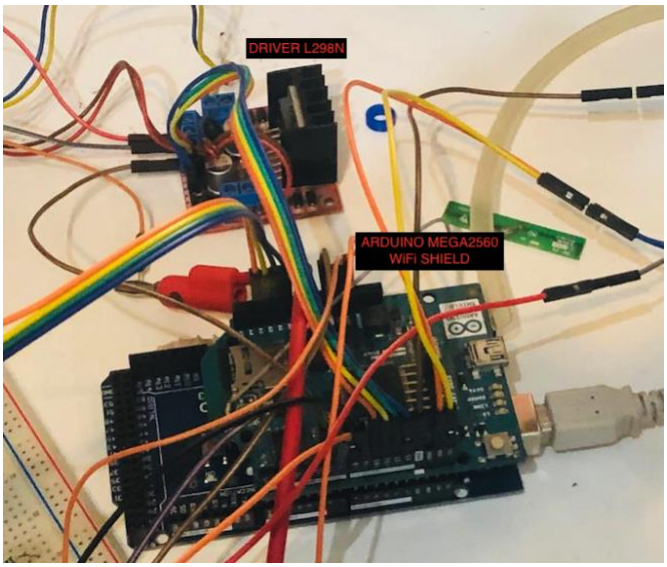


Figura: 74 ArduinoMega2560, escudo WiFi y driver L298N

En las siguientes Figura: 75 y Figura: 76se aprecian al sistema desde diferentes puntos des vista.

1. Vista Frontal inclinada superior
2. Vista Frontal elevada superior

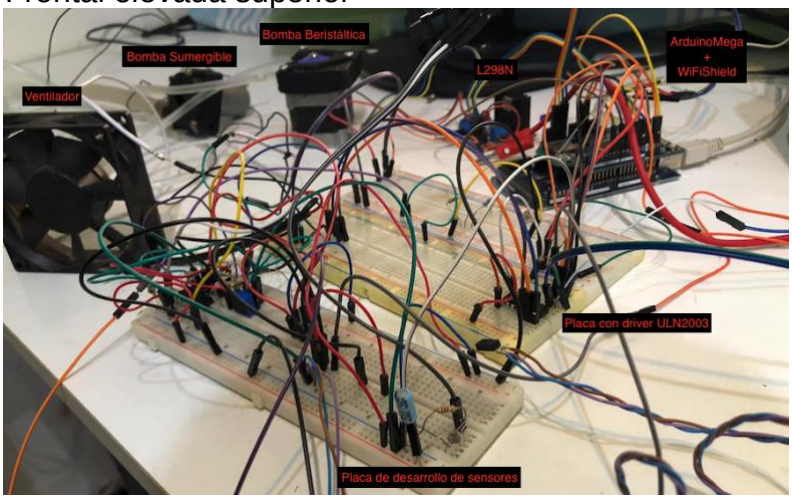


Figura: 75 Circuito vista inclinada superior

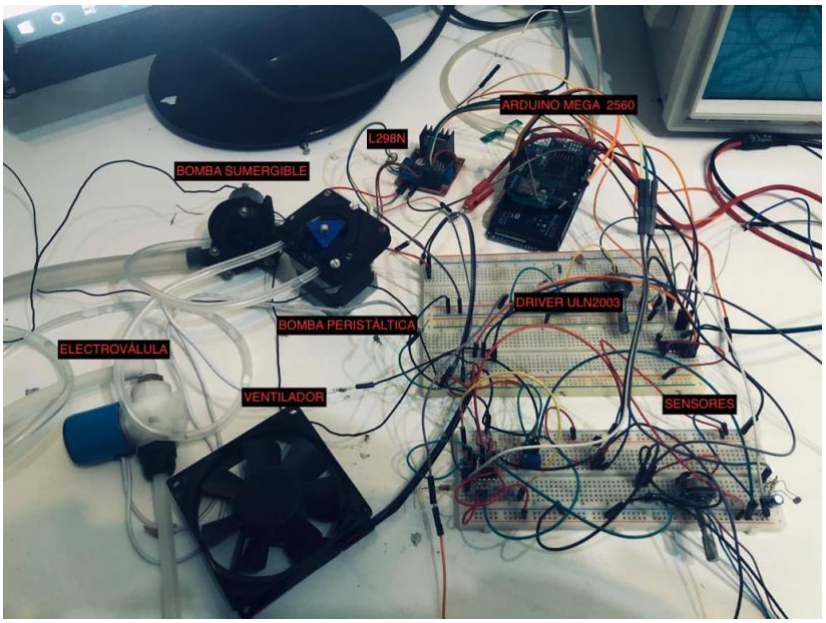


Figura: 76 Circuito Vista Frontal Superior

En la siguiente Figura: 77 se adjunta el esquemático de las conexiones realizadas del proyecto.
Este esquema indica los pines de conexión del microcontrolador con los actuadores y sensores.

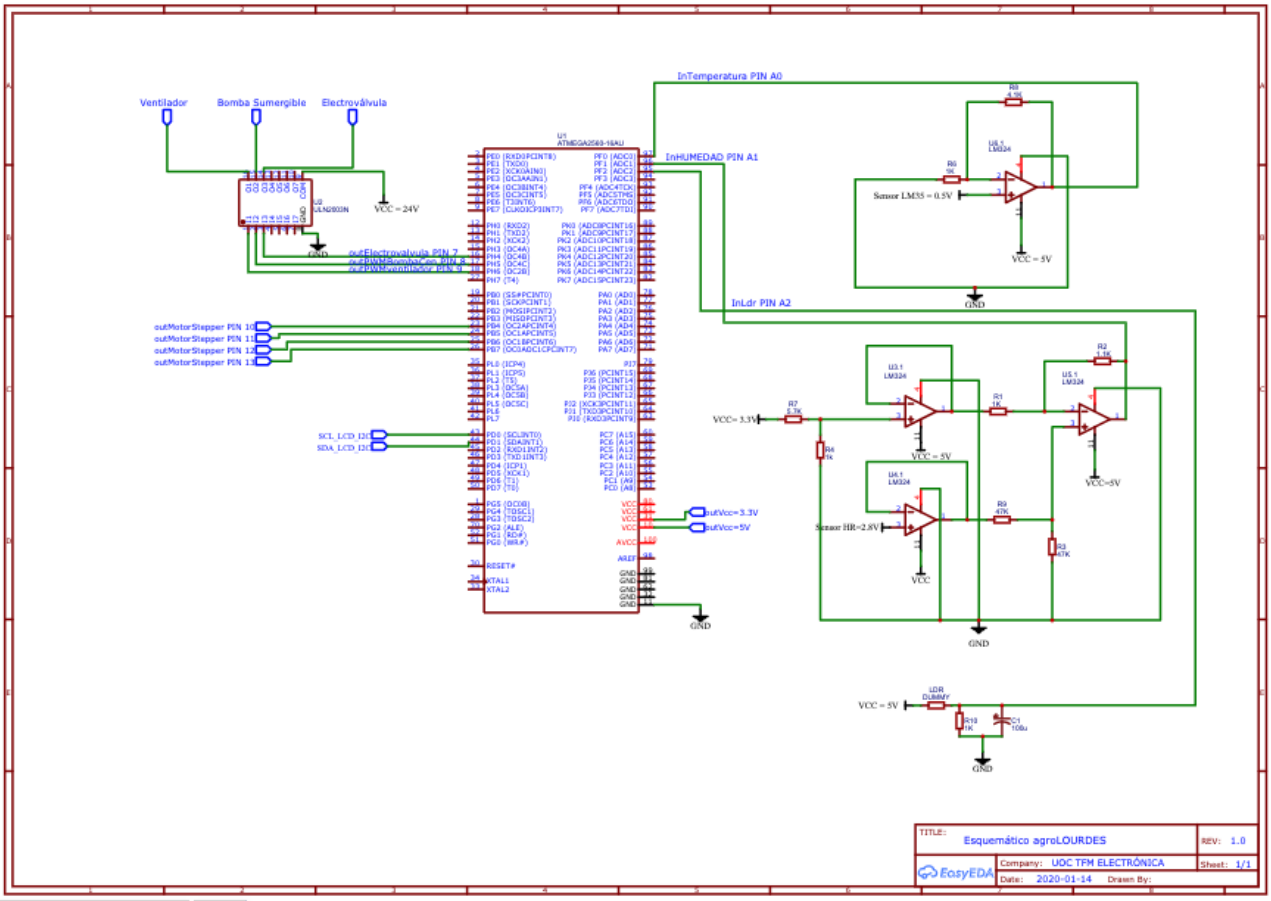


Figura: 77 Esquemático del circuito completo

4.1 Presupuesto

Tabla 2 Presuesto Dispositivos

Dispositivo	Cantidad	Precio €	Total €
Arduino Mega	1	34,50	34,50
Arduino WiFi Shield	1	87,10	87,10
LM35Z	1	1.95	1.95
LDR NSL-19M51	1	0,77	0,77
LM324N	1	0,23	0,23
ULN2003N	1	0.29	0.29
Electroválvula KSD 24V	1	8,41	8,41
Bomba Peristáltica	1	21,66	21,66
L298N	1	0,58	0,58
Bomba sumergible DC12v	1	6,79	6,79
Ventilador NMB-MAT BG0703-B042-0000	1	11,34	11,34
Pantalla LCD	1	9,18	9,18
Resistencias	15	0.44	0.44
Total			180,56€

Tabla 3 Presupuesto Ingeniero

Precio Trabajo Ing. Milton Campoverde Rosales		
Días trabajado	4h/día	8€/h
88	352h	2816€

El precio total del proyecto asciende a **2996,56€:**

$$Total = Dispositivos + Pago\ Ingeniero = 232{,}8 + 2816 = 3048{,}8€$$

4.2 Resultados

En este apartado se realiza un test cíclico para evaluar el funcionamiento del sistema. Se utiliza el osciloscopio para ver los ciclos de trabajo del PWM del ventilador y bomba sumergible.

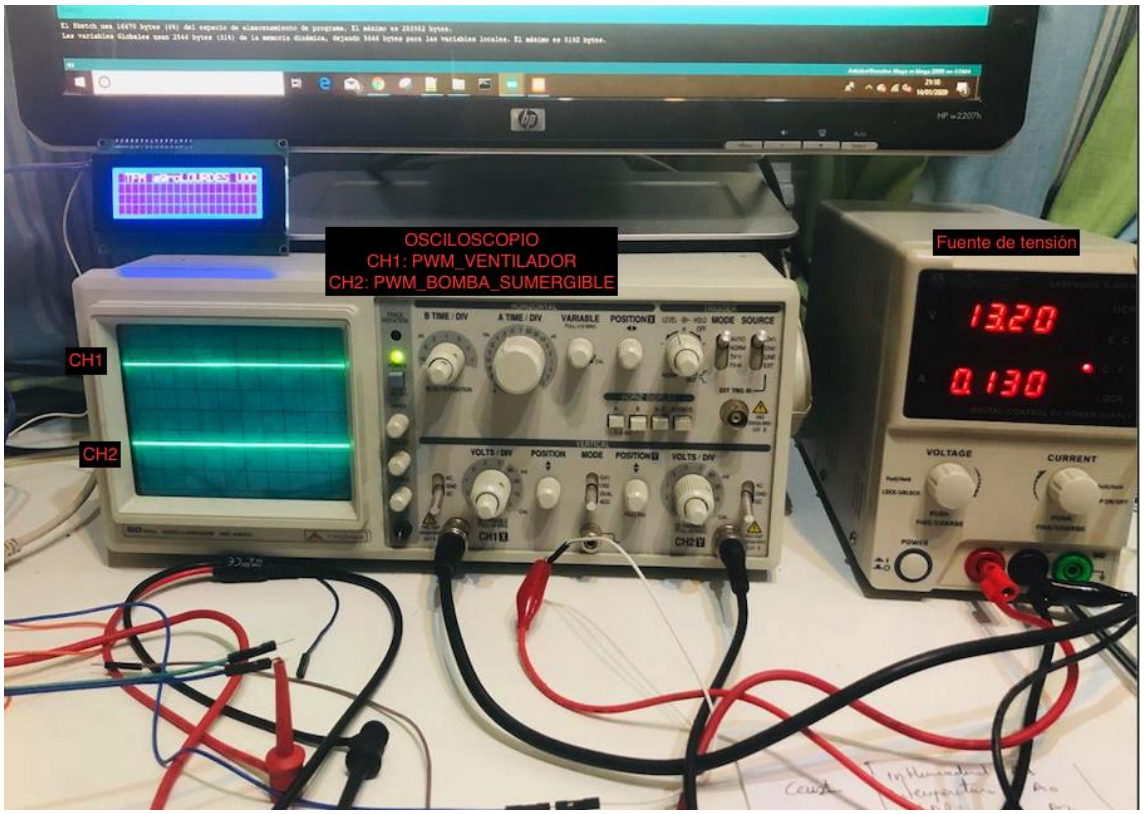


Figura: 78 Osciloscopio con 2 canales: CH1 Ventilador y CH2 Bomba sumergible

Antes del inicio del sistema agroLourdes se realizan 2 fase antes de que se ponga en marcha el sistema con las funcionalidades definidas finales.

En la fase 1 realiza la siguiente validación:

1. Enciende y apaga la electroválvula (video)
2. Enciende la bomba sumergible (video)
3. Apaga la bomba Sumergible (video)
4. Ventilador encendido

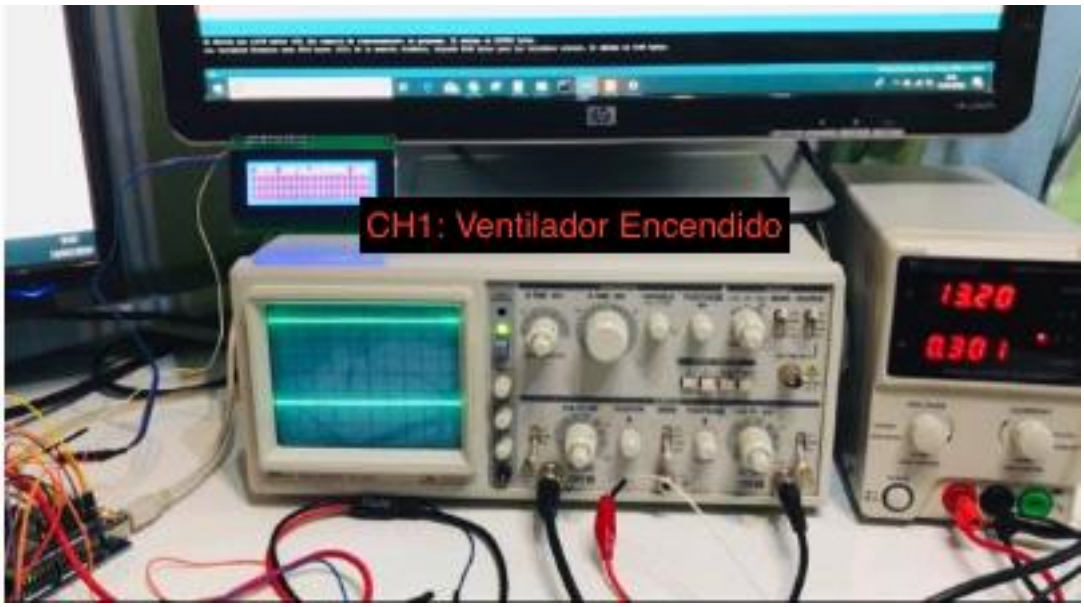


Figura: 79 Ventilador Encendido en Test Fase 1

5. Ventilador apagado

Figura: 80 Ventilador Apagado en Test Fase 1

6. Evalúa la bomba peristáltica con un giro completo en sentido horario (video)
7. Evalúa la bomba peristáltica con un giro completo en sentido antihorario (video)

```
Ol...c0KK0c...l0KKKKKKKKKKKKKKKKKKKKKKXW
0c.  :OKKO:  .c0K0kdollodk0KK000KKKKKXW
0c.  :OKKO:   c0x:..     .;d0K0KKK0KKXW
0c.  :OKKO:  .cx,  .,cc;.  'd00KKK000XW
0c.  ;OKKO;  .cl.  ,kKKO:  .c00Odl::;dX
Kd'  .:ll:.  'dl.  ;OKK0c.  :Od,.    ;K
K0d,.      .,d0l.  ;OKK0c.  :d,  .:odkN
KK0OxlcccclxO00l.  ,k0KO;  .cl.  ,kKKXW
K000KKKKKKKK000k;  .':::,.  'dl.  ;kKKXW
K0000K00KK000000kc'.    ..:x0l.  ;OKKXW
K0000K00KK000000K0OxdoooxO0K0d.  .dO0KN
K0000K00KK00000K00KKKKKKKKK0KOc.  .''lK
K0000K00KK0000KK0KKKK00000KK0KOd:'...:K
K0000K00KK0000KKKKK000000KKKKK0000kkk0N
K0000K00KK0000KKKKK00000KKKKK000000KKXW
************************************
***UNIVERSITAT OBERTA DE CATALUNYA***
*********TFM agroLOURDES*************
************************************
***Phase 1 initializing...***
Testing EV...
CEN_125
CEN_0
Centrifugal pumpe OK...
Ventilator test ...
Ventilator test OK...
Testing peristaltic pump...
clockwise
counterclockwise
Peristaltic pump OK...
***Phase 1 completed***
```

Figura: 81 Fase 1 finalizada con éxito

En la fase 2 realiza la siguiente validación:

1. Sitúa le PWM del ventilador al 100%

Figura: 82 CH1: PWM del ventilador al 100%

2. Sitúa el PWM del ventilador al 0%

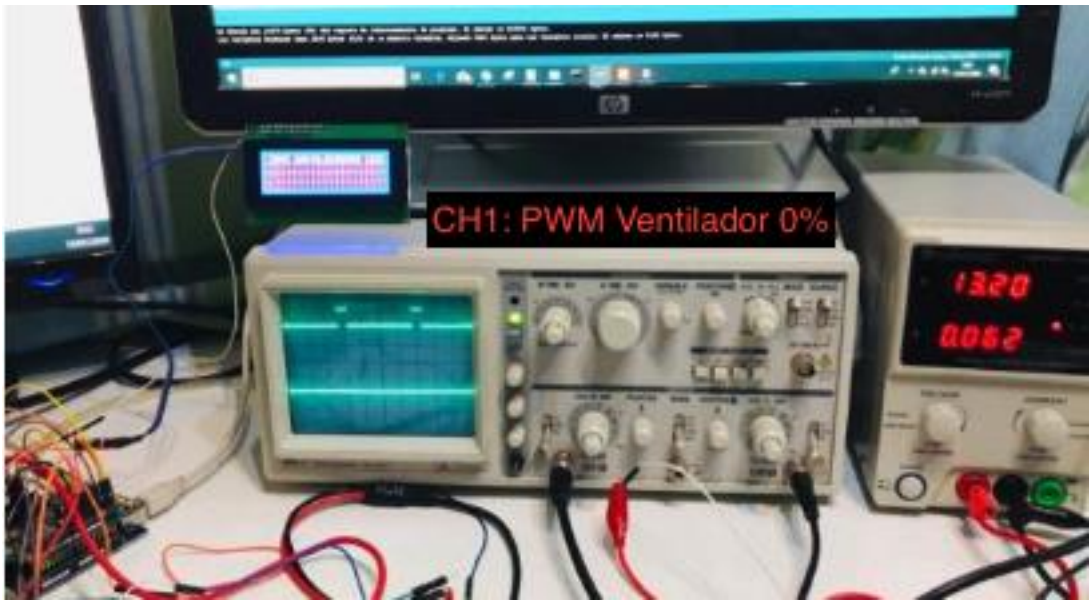


Figura: 83 PWM del ventilador al 0%

3. Enciende Electroválvula
4. Apaga Electroválvula
5. Sitúa le PWM de la bomba sumergible al 100%

Figura: 84 PWM de la bomba sumergible al 100%

6. Sitúa el PWM de la bomba sumergible al 0%

Figura: 85 PWM bomba sumergible al 0%

7. Bomba Peristáltica giro lento con sentido horario
8. Bomba peristáltica giro rápido con sentido antihorario

```
***Phase 2 initializing...***
Ventilator PWM test duty up...
Ventilator PWM test duty down...
Ventilator PWM off...
Electrovalve ON !!!
Electrovalve OFF !!!
Centrifugal pume PWM test duty up...
Centrifugal pume PWM test duty down...
Centrifugal pume PWM test OK...
Centrifugal pume PWM set to zero
Peristaltic pump clockwize slow test
clockwise
Peristaltic pump counterclockwize fast test
counterclockwise
Peristaltic pump Enable OFF
***Phase 2 completed***

******All test done succesfully******
```

Figura: 86 Fase finalizada con éxito

Una vez se ha validado el test, se procede con la lectura de los parámetros de temperatura, luminosidad y humedad relativa.
En esta fase se ha utilizado la pantalla LCD como herramienta de ayuda para poder mostrar qué información se envía al servidor.
Luego de enviar los parámetros vía wifi a la base de datos, el cliente podrá acceder finalmente a la información, de manera visual en una tabla y/o en una gráfica.

En la siguiente Figura vemos con se ha finalizado el test con éxito y muestra la información que será enviada al servidor.

```
******All test done succesfully******

STARTING MAIN PROCESS...
*************************
*************************
*************************
376TEMPERATURA: 22°C
204LUZ: 204lux
524HR: 524%
439TEMPERATURA: 25°C
25524HR: 524%
52205LUZ: 205lux
205437TEMPERATURA: 25°C
526HR: 526%
203LUZ: 203lux
Attempting to connect to SSID: ING_MILTON
Connected to wifi

Starting connection to server...
Conectado al Servidor
GET http://localhost/testcode/dht.php?&temperature=25&humidity=52&Luminocidad=203
```

Figura: 87 URL con información al servidor

En este caso se envía una lectura de:

- Temperatura = 25ºC
- Humedad = 52%
- Luminosidad 203 Lux

En la Figura: 88 se demuestra la llegada de la información al servidor ha sido correcta.
Para este test se envían datos cada 10 segundos.

+ Opciones

	ID	humidity	temperature	Luminocidad	DATE 1
Editar Copiar Borrar	574	53	25	193	2020-01-15 01:29:45
Editar Copiar Borrar	572	53	25	196	2020-01-15 01:29:35
Editar Copiar Borrar	573	52	25	203	2020-01-15 01:29:35

Figura: 88 Base de datos con información recogida de la prueba

Como se ha comentado, el cliente tiene dos opciones de visualización una por medio de la tabla, y otra opción es con una gráfica, donde además tiene la posibilidad de realizar acciones en la pantalla con la gráfica como por ejemplo realizando zoom de los valores, acotar valores, fechas, capturas de imágenes, etc…

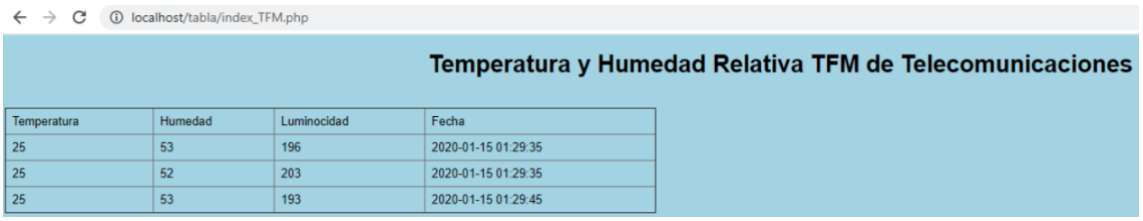


Figura: 89 Tabla que muestra datos de los sensores almacenado en la base de datos

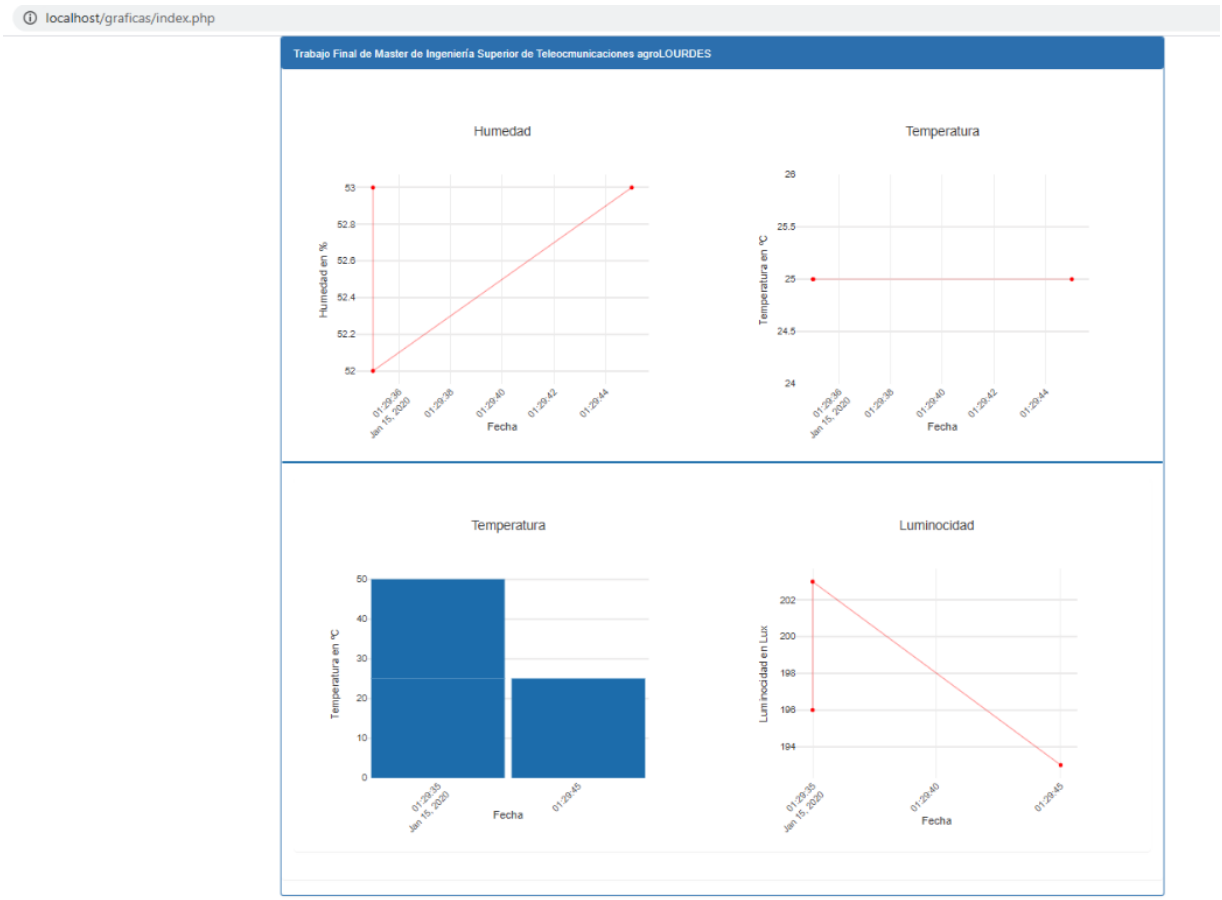


Figura: 90 Gráficas en Barra y Lineal de temperatura y lineal de luminosidad y humedad relativa

En la Figura 91 se aprecian 4 gráficas donde una de barras y una lineal corresponden a los valores de la temperatura, las otras dos corresponden a valores de la humedad y luminocidad.

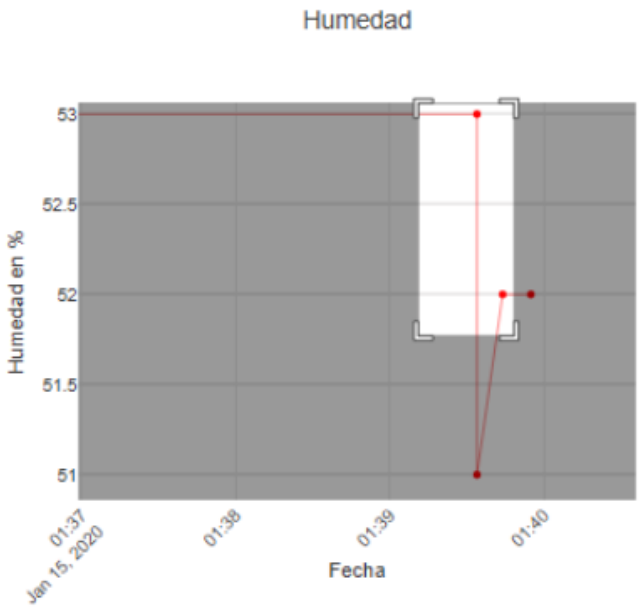


Figura: 91 Zoon en gráfica de Humedad Relativa

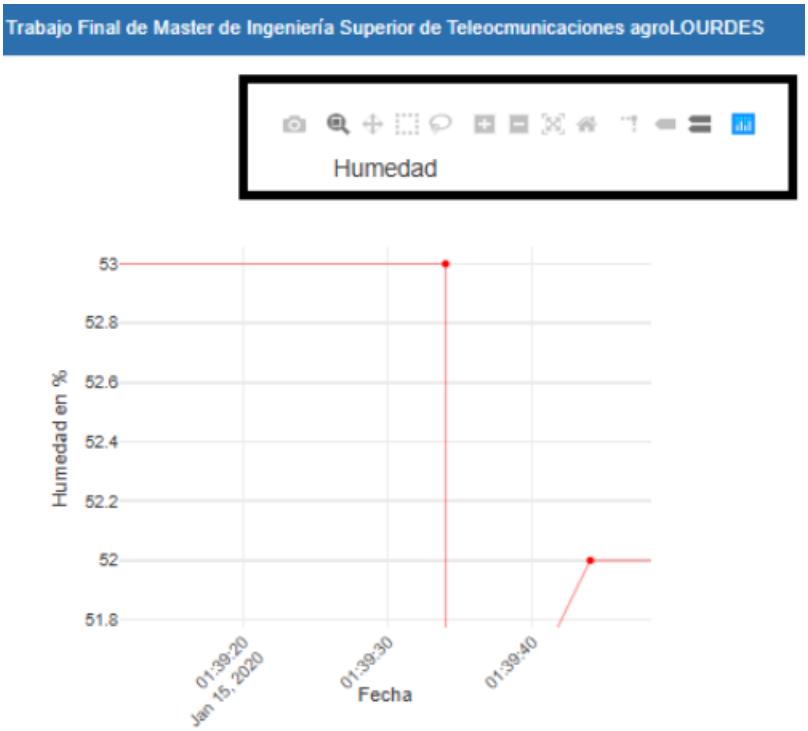


Figura: 92 Barra de herramientas con las diferentes opciones de gráfico

Luego de enviar la información al servidor por primera vez, se ha programado para que el servidor recibe la información cada 5m.

4.3 Problemas y soluciones

A la hora de trabajar con el operacional LM234 se han tenido dificultades porque en el datasheet del fabricante expone que tiene una tensión de salida de Vcc-1.5V, provocando que no se aproveche la máxima resolución del ADC de 10 bits del microcontrolador de ARDUINOMEGA2560. Como solución a este

inconveniente, se ha optado por modificar la tensión de referencia gracias al software a una tensión de 2.56V.

Se ha utilizado la herramienta de software PSPICE para poder validar los cálculos y realizar simulaciones para entender el funcionamiento del circuito.
Cuando se ha simulado la salida del operacional LM324 en la gráfica se aprecia que tiene una tensión máxima de 4,3V aproximadamente, lo cual en este caso el simulador falla ya que en ele datasheet si se alimenta a 5V como máximo su salida debería de ser 3.5V. Por lo que queda claro que las simulaciones no son fiables al 100% y por ello se recomiendo realizar pruebas físicas además de las teóricas.

Cuando se ha realizado el montaje del circuito acondicionador del sensor de humedad relativa, se han tenido problemas, debido a que la tensión de salida del circuito acondicionador en todo momento marcaba una tensión de 0.4V. independientemente de la señal de salida del sensor hasta los 0.9V, que comenzaba a variar. Luego de varias pruebas la tensión era provocada debido a las tolerancias de las resistencias y a la tensión de offset del operacional, lo que provocaba una ganancia a la salida errónea y no contemplada en los cálculos matemáticos.
La solución a este problema fue alimentar el pin GND con una tensión negativa para que pueda funcionar correctamente.

Por otra parte, la calibración del sensor de humedad relativa también ha dado dificultad, ya que en el laboratorio carecen de un higrómetro y no se tenía referencia. La solución a este problema fue la fórmula matemática que ofrece el fabricante en el datasheet y la consulta a la página de meteorología de la zona donde el autor está realizando las pruebas. Como nota importante, resaltar que el sensor es sensible a la luz y propenso a humedecerse por lo que se debe de secar con aire caliente para que las lecturas sean más precisas.

Cuando se ha procedido con el montaje del circuito del sensor LDR, su salida no era estable por lo que su valor de salida fluctuaba demasiado. La solución fue colocar un condensado para de esta forma filtrar el ruido y con esto se logró estabilizar la salía.

Respecto a la electroválvula, debido a que el fabricante no ofrece información técnica suficiente del dispositivo se han tenido que realizar pruebas para comprobar su inductancia y así realizar el estudio teórico y practico de su intensidad en el circuito. La solución utilizando como herramienta el generador de funciones y la fórmula de la frecuencia de corte.
Una vez se conoció el valor de su inductancia, se procedió a realizar simulaciones en PSPICE, y en esta ocasión los resultados prácticos y teóricos coincidía en un alto porcentaje.

Uno de los problemas fue la alimentación de los actuadores ya que la tensión de salida del microcontrolador oscila entre [3,3-5]V por lo que alimentar los actuadores que trabajan con una tensión [12-24]V es imposible. Entonces se optó por el driver ULN2003, que gracias a la configuración Darlignton permite

que se amplificara la corriente y con esto llegar a la tensión de operación de los actuadores.

4.4 Discusión de los resultados

En este proyecto, se ha intentado lograr precisión y economía en cuanto respecta la elección de sensores, por esta razón los sensores digitales se descartaron debido a su precio relativamente más elevado y además que arrastran un mayor error en comparación a los sensores analógicos.

Pero queda claro que al final existe un compromiso, y dependiendo de la aplicación en donde se necesiten este tipo de sensores ya sean digitales o analógicos, se debe de elegir entre abaratar costos o ser más precisos.

Además, todos los actuadores que se han utilizado en este proyecto han sido reciclado de electrodomésticos antiguos y obsoletos, lo que económicamente para este trabajo benefició al autor, pero en otras no, debido a la poca información que ofrece el fabricante para algunos de los actuadores.

Por otra parte, los resultados obtenidos hasta la fase que se ha llegado en el TFM (comunicación con el servidor, recogida de parámetros ambientales, precisión de los datos recogido, activación de actuadores, graficar datos) es satisfactorio en un 100%.

La próxima fase del proyecto sería validarlo en un entorno real, y posteriormente trabajar más en él para que logre operar a mayor escala (haciendas, cultivos con grandes extensiones de tierra, etc…) que es uno de los objetivos principales del proyecto.

Respecto al precio del proyecto, en total solo en componentes de prototipado suman un monto de 180,56€ más la mano de obra del Ingeniero suman un total de 3050€ aproximadamente. Pero es bien sabido que el prototipado de cualquier versión de un producto siempre es elevado, y en este proyecto ocurre lo mismo. Pero, el objetivo es que este proyecto no llegue a costar más de 30€ reduciendo su coste inicial en un 99%.

Por otra parte, una de las dificultades más complicadas en este proyecto ha sido el prototipar con Arduino y no tener la posibilidad de depurar código. En un inicio se logró conseguir una placa depuradora de código, pero se requiere un tiempo de aprendizaje y familiarizarse con el entorno AS7 “ATMEL STUDIO 7” para poder realizar pruebas, y por lo tanto esta opción quedó descartada ya que este proyecto tiene un tiempo límite de duración.

Resaltar también, otra dificultad fue un retraso de tiempo que existieron debido a la espera de algún componente, concretamente la placa de depuración AVR dragon de ATMEL, esto también fue un imprevisto en el proyecto a nivel económico.

Para finalizar, el autor destaca el aprendizaje adquirido realizando este TFM ya que en el futuro cercano puede poner en practica todo lo aprendido.

5 Glosario

IOT	(Internet of Things) Internet de las cosas
PHP	Hypertext Preprocessor
Datasheet	Ficha técnica
PWM	(Pulse-Width modulation) Modulación de ancho de pulso
GND	(Ground)Toma de tierra
Arduino	Compaía de Software y Hardware
IDE	(Integrated Development Enviroment)Entorno de desarrollo integrado.
BBDD	Base de datos
Pspice	Programa de simulación de circuitos analógicos y digitales
SQL	(Structure Query Language)Lenguaje de consulta estructurada
I2C	Inter Integrated Circuit
JSON	JavaScript Object Notation
SCL	Serial Clock para I2C
SCK	Serial Clock para SPI
WBS	Work Breakdown Structure

6 Bibliografía

1.- Arduino, USA, 28/12/2019
https://www.arduino.cc/en/Guide/ArduinoWiFiShield

2.- ARDUINO, España, 10/11/2019
https://forum.arduino.cc/index.php?topic=475541.0

3.- Control automático de procesos industriales, Alfredo Roca

4.- Datasheet LM324N
http://www.ti.com/lit/ds/symlink/lm324-n.pdf

5.-Datasheet ULN2003A
http://www.ti.com/lit/ds/symlink/ulq2003a.pdf

6.-envatotuts+, Monty Shoken, 15/12/2019
https://code.tutsplus.com/es/tutorials/create-interactive-charts-using-plotlyjs-line-charts--cms-29201

7.- Instalar y configurar un servidor MySQL y PHPMyAdmin, Ruben Velasco,España, 5/11/2019.
https://www.redeszone.net/2017/03/18/instalar-configurar-servidor-mysql-phpmyadmin/

8.- Instrumentación Electrónica, Miguel Angel Perez Garcia, España

9.- Jecrespom, España 5/10/2019
https://aprendiendoarduino.wordpress.com/2016/11/12/wifi-en-arduino/

10.- Malvino, Alber Paul, Principios de Electrónica, España

11. - *PlatformIO IDE for Atom, Mexico, 02/10/2019*
http://docs.platformio.org/en/latest/ide/atom.html#installation

12.- Programación en C++, Luis Joyanes Aguilar

13.-Sistemas Electrónicos PROGRES, España, 25/09/2019
https://www.progres.es/es/productos/sensores-y-transmisores/planta

14.- Secmotic, España, Sevilla, 10/12/2019
https://secmotic.com/encender-bombilla-con-un-modulo-wifi-esp8266-arduino/

15.- *SysProgs,USA 2/10/2019*
https://visualgdb.com/tutorials/arduino/avr/debug/

16.- Válvula de control: Selección y Cálculo

7 Anexos

Configuración de Base de Datos

```
<?php
class dht11{
 public $link='';
 function __construct($temperature, $humidity){
  $this->connect();
  $this->storeInDB($temperature, $humidity);
 }

 function connect(){
  $this->link = mysqli_connect('localhost','root','') or die('Cannot connect to the
DB');

  mysqli_select_db($this->link,'temperatura') or die('Cannot select the DB');
 }

 function storeInDB($temperature, $humidity){
  $query = "insert into dht11 set humidity='".$humidity."',
temperature='".$temperature."'";
  $result = mysqli_query($this->link,$query) or die('Errant query:  '.$query);
 }

}
if(($_GET['temperature'] != '' and $_GET['humidity'] != '')){ /*La modificación es:
agregar paréntesis al IF*/
 $dht11=new dht11($_GET['temperature'],$_GET['humidity']);
}
/*http://localhost/testcode/dht.php?temperature=12&humidity=10*/
?>
```

Configuración de Pantalla LCD Arduino

```
#include "Pantalla.h"
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "Sensores.h"
Sensores sensor;
Pantalla menuP;
Pantalla::Pantalla() {

}
void
Pantalla::startScreen() {
  lcd.init();                    // initialize the lcd
```

```
  lcd.backlight();// Print a message to the LCD.
  lcd.setCursor(1, 0);
  lcd.print("TFM agroLOURDES UOC");
}
void
Pantalla::PantallaIncio() {
  screentemp = sensor.TemperaturaLoop();
  screenLux = sensor.LDR();
  screenHum = sensor.humedad();


  // Cursor en la primera posición de la primera fila
  //  lcd.setCursor(1, 0);
  //  lcd.print("TFM agroLOURDES UOC");
  // Cursor en la segunda posición de la segunda fila
  lcd.setCursor(1, 1);
  lcd.print("T:");
  lcd.print(screentemp);
  lcd.print("C ");
  //// Cursor en la octava posición de la segunda fila
  lcd.setCursor(8, 1);
  lcd.print("Luz:");
  lcd.print(screenLux);
  lcd.print("Lux   ");
  // Cursor en la primera posición de la tercera fila
  lcd.setCursor(1, 2);
  lcd.print("HR");
  lcd.print(screenHum);
  lcd.print("%");
  //vVentilador = ;

sensor.pwm1Ventilador(screenLux);
  //sensor.bombaPer();



}
/*void
Pantalla::SendData(){
    lcd.setCursor(1, 1);
  lcd.print("Send Data To Server");
}
*/
/*
void
Pantalla::Menu() {

  if ((digitalRead(53)) == HIGH) { //Condición si está activado el PIN A0
    lcd.clear();
    do{
```

```
      lcd.setCursor(1, 0);
  lcd.print("TFM agroLOURDES UOC");
      menuP.PantallaIncio();
    }while(digitalRead(51) == LOW);
   lcd.clear();
  }
  if (((digitalRead(53)) == LOW)) {
    lcd.setCursor(1, 0);
  lcd.print("TFM agroLOURDES UOC");
    lcd.setCursor(1, 1);
    lcd.print("Sensores:1");

    lcd.setCursor(1, 2);
    lcd.print("Ventilador:2");

  }
}*/


***Configuración Pantalla.h****

#ifndef PANTALLA_H
#define PANTALLA_H
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 20, 4);
class Pantalla {
  private:
    int screentemp = 0;
    int screenHum = 0;
    int screenLux = 0;
  public:
    Pantalla();
    void PantallaIncio();
    void startScreen();
    //void Menu();
   // void SendData();

};
#endif
***Condfiguración envio de datos****

#include "SendData.h"
#include <SPI.h>
#include <WiFi.h>
char ssid[] = "#######"; //  your network SSID (name)
char pass[] = "#######";

//Parámetros de entrada para enviar a la BBDD MySQL
#define parametro1 "&temperature="
```

```
#define parametro2 "&humidity="
#define parametro3 "&Luminocidad="


int status = WL_IDLE_STATUS;
// if you don't want to use DNS (and reduce your sketch size)
// use the numeric IP instead of the name for the server:
IPAddress server(192, 168, 1, 39); // numeric IP se busca en IPConfig y es el
del WiFi
WiFiClient client;

SendData::SendData() {

}


float
SendData::EnviarDatosSU(int Temperatura, int Humedad, int Luminocidad) {
  String envioDatos = "GET http://localhost/testcode/dht.php?";
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  // check for the presence of the shield:
  if (WiFi.status() == WL_NO_SHIELD) {
    Serial.println("WiFi shield not present");
    // don't continue:
    while (true);
  }

  String fv = WiFi.firmwareVersion();
  if (fv != "1.1.0") {
    Serial.println("Please upgrade the firmware");
  }

  // attempt to connect to Wifi network:
  while (status != WL_CONNECTED) {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    // Connect to WPA/WPA2 network. Change this line if using open or WEP
network:
    status = WiFi.begin(ssid, pass);

    // wait 10 seconds for connection:
    delay(10000);
  }
  Serial.println("Connected to wifi");
  //printWifiStatus2();

  Serial.println("\nStarting connection to server...");
```

```
  // if you get a connection, report back via serial:
  // Serial.println(server);
  if (client.connect(server, 80)) {
    Serial.println("Conectado al Servidor");
    // Make a HTTP request:
    envioDatos += parametro1 + String(Temperatura) + parametro2 +
String(Humedad) + parametro3 + String(Luminocidad);
    Serial.println(envioDatos);
    client.println(envioDatos);


client.println("GET
http://localhost/testcode/dht.php?temperature=12&humidity=10"); // Esta URL
Funciona es la base.
    client.println(" HTTP/1.1");
    client.print("Host: ");
    client.println(server);
    client.println("Connection: cancel");
    client.println("Connection: close");
    client.println();
  }while (client.available()&& status == WL_CONNECTED) {
    char c = client.read();
    Serial.write(c);
  }
  while (client.available()) {
    char c = client.read();
    Serial.write(c);
  }

  // if the server's disconnected, stop the client:
  if (!client.connected()) {
    Serial.println();
    Serial.println("disconnecting from server.");
    client.stop();
 client.flush ();
    // do nothing forevermore:
   // while (true);
  }
  envioDatos = "";


  /* client.stop();
     client.flush();*/
}
void
SendData::EnviarDatosLOOP() {
  // if there are incoming bytes available
  // from the server, read them and print them:
  while (client.available()) {
    char c = client.read();
```

```
    Serial.write(c);
  }

  // if the server's disconnected, stop the client:
  if (!client.connected()) {
    Serial.println();
    Serial.println("disconnecting from server.");
    client.stop();

    // do nothing forevermore:
    while (true);
  }
}
void
SendData::printWifiStatus2() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}
```

***Configuración Envio de Datos.h ***

```
#ifndef SENDATA_H
#define SENDATA_H
#include <SPI.h>
#include <WiFi.h>

class SendData {
  private:

  public:
    SendData();
    float EnviarDatosSU(int Temperatura, int humedad, int
Luminocidad);//Enviamos los datos a la BBDD
http://localhost/phpmyadmin/sql.php?server=1&db=temperatura&table=dht11
&pos=0
    void EnviarDatosLOOP();
    void printWifiStatus2();
```

```
};
#endif
****Sensores.CPP******

/*Autor: Milton Campoverde
  Empresa: Trabajo Final de Master UOC
  Funcion de la clase:



*/
#include <Stepper.h>
#include "Sensores.h"
const char stepsPerRevolution = 200;

Stepper myStepper(stepsPerRevolution, 10, 11 , 12 , 13);
Sensores::Sensores() {
  temp = 0;
  tempLdr = 0;
  tempHum = 0;
}

int
Sensores::LDR() {
  tempLdr = analogRead(inLdr);
  Serial.print(tempLdr);
  Serial.print("LUZ: ");
  Serial.print(tempLdr);
  Serial.print("lux");
  Serial.println();
  return tempLdr;
}
int
Sensores::pwm1Ventilador(int pwm) {
  analogWrite(outPwmVentilador, pwm);//80% de PWM
  //delay(2000);
  //analogWrite(outPwmVentilador, 0);//80% de PWM
}
void
Sensores::ev() {
  digitalWrite(7, HIGH);//80% de PWM
  delay(3000);
  digitalWrite(7, LOW);//80% de PWM;

}
int
Sensores::bombaPer() {

int x=1;
  // map it to a range from 0 to 100:
```

```
  int motorSpeed = map(1000, 0, 1023, 0, 100);
  // set the motor speed:
  Serial.println(motorSpeed);
  myStepper.setSpeed(1000);
  if (x==1){
   Serial.println("clockwise");
  myStepper.step(600);
  x--;
  }
  else if(x==0){
      Serial.println("counterclockwise");
  myStepper.step(-600);
  }




  /* if (motorSpeed > 0) {
     myStepper.setSpeed(motorSpeed);
     // step 1/100 of a revolution:
     myStepper.step(stepsPerRevolution / 100);
    }*/
}
int
Sensores::humedad() {
  tempHum = analogRead(inHumedad);
  Serial.print(tempHum);
  Serial.print("HR: ");
  Serial.print(tempHum);
  Serial.print("%");
  Serial.println();

  return tempHum / 10;
}

int
Sensores::pwm2bombaCen(int pwm2) {
  analogWrite(outPwmBombaCen, pwm2);//80% de PWM

}
int
Sensores::TemperaturaLoop() {
  temp = analogRead(inTemperatura);
  //test=analogRead(A0);
  //Serial.print(temp);
  Serial.print(temp);
  temp = temp * 50 / 852;
  Serial.print("TEMPERATURA: ");
  Serial.print(temp);
  Serial.println("ยบC");
```

```
  //  Serial.print(test);
  //  Serial.println();
  // // digitalWrite(3, LOW);
  //  //delay(1000);
  //  digitalWrite(2, HIGH);
  //  delay(1000);
  //  pinMode(4, OUTPUT);
  //  analogWrite(4,200);
  //   digitalWrite(2, LOW);
  //   delay(1000);
  // //  digitalWrite(3, HIGH);
  return temp;
}


***Sensores.h****

#ifndef SENSORES_H
#define SENSORES_H
#include <SPI.h>
#include <WiFi.h>
#include <Stepper.h>
class Sensores {
  private:
    int test;
    const int inTemperatura = A0; //Entrada Temperatura
    const int inLdr = A2;
    const int inHumedad = A1;


    //Salidas
    const int outPwmVentilador = 9; // Analog output pin
    const int outPwmBombaCen = 8; // Analog output pin
    byte Pwm = 0;        // valor del PWM
    const int en_step = 6; // Analog output pin
    int temp;
    int tempLdr;
    int tempHum;
  public:
    Sensores();//Constructor que incializa las variables
    int TemperaturaLoop(); //
    int LDR();
    int humedad();
    //Salidas
    int pwm1Ventilador(int pwm);
    void ev();
    int bombaPer();
    int pwm2bombaCen(int pwm2);
};
#endif
```

***Envio de datos ***

```
#include <Stepper.h>
#include <SPI.h>
#include <WiFi.h>
#include "SendData.h"
#include "Sensores.h"
#include "Pantalla.h"

Pantalla pantalla;
SendData sendData;
Sensores sensores;
int sendHumedad;
int sendTemperatura;
int sendLux;
const int stepsPerRevolution = 200;
Stepper myStepper2(stepsPerRevolution, 10, 11, 12, 13);
//Intervalo de tiempo para enviar datos al servidor cada 5 minutos
long intervalo = 10000; /// 300000;// es el tiempo de nuestro delay   5 minutos
10000 10 segundos
long tiempo = 0;
long tiempoAnterior = 0;
int x = 1;
int test=0;
int vVentilador = 0;
//long tSensorPantalla = 0;
//long tSensorPantallaTiempoAnterior=0;
//long intervaloSensorPantalla = 0;

void motor() {

  // map it to a range from 0 to 100:
  int motorSpeed = map(1000, 0, 1023, 0, 100);
  // set the motor speed:
  //Serial.println(motorSpeed);
 // myStepper2.setSpeed(50);
  if (x == 1) {
    Serial.println("clockwise");
    myStepper2.step(200);
    x--;
  }
  else if (x == 0) {
    Serial.println("counterclockwise");
    myStepper2.step(-200);
    x = 10;
  }
}
```

```
void setup() {
  //Initialize serial and wait for port to open:
  Serial.begin(9600);
pinMode(6, OUTPUT);
pinMode(7, OUTPUT);

  myStepper2.setSpeed(1000);
  analogReference(INTERNAL2V56);

  //keyIndex= sensores.TemperaturaLoop();

  //Serial.print(keyIndex);
  //  keyIndex2= sensores.humedad();
  //  Serial.print(keyIndex2);

  pantalla.startScreen();




  //  pinMode(2, OUTPUT);
  //  pinMode(3, OUTPUT);
  //  pinMode(4, OUTPUT);
  //sendData.EnviarDatosSU(keyIndex,80);
}

/*
  void loop(){
  // sensores.bombaPer();
  sensor.ev();
  }

*/
void loop() {
  //  sensores.pwm1Ventilador(100);
  //  pantalla.Menu();
  //  delay(500);
  // for(int i = 0; i<100;i++){

//test sequencial
if (test==0)
{
  sensores.ev();
  Serial.println("Testing EV...");
  delay(1000);
  sensores.pwm2bombaCen(125);
  Serial.println("CEN_125");
  delay(2000);
  sensores.pwm2bombaCen(0);
  Serial.println("CEN_0");
```

```
  Serial.println("Centrifugal pumpe OK...");
  //  sensores.bombaPer();
   Serial.println("Ventilator test ...");
  sensores.pwm1Ventilador(255);
  delay(3000);
  sensores.pwm1Ventilador(0);
  Serial.println("Ventilator test OK...");

  Serial.println("Testing peristaltic pump...");
  myStepper2.setSpeed(50);
  digitalWrite(6, HIGH);
  motor();
  //sensores.bombaPer();
  digitalWrite(6, LOW);
  delay(1000);
  digitalWrite(6, HIGH);
  motor();
  //sensores.bombaPer();
  digitalWrite(6, LOW);
  Serial.println("Peristaltic pump OK...");

  Serial.println();
  Serial.println();
  Serial.println("Phase 1 completed");
  //
//pwm test again
delay(1000);


///test fase 2
Serial.println("Phase 2 initializing...");
delay(250);
Serial.println("Ventilator PWM test duty up...");
for (int i=10;i<=255;i++)
    {
      sensores.pwm1Ventilador(i);
      delay(10);
      }
 Serial.println("Ventilator PWM test duty down...");
for (int y=254;y>0;y--)
    {
      sensores.pwm1Ventilador(y);
      delay(10);
      }
sensores.pwm1Ventilador(0);
 Serial.println("Ventilator PWM off...");
 delay(500);
 Serial.println("Electrovalve ON !!!");
sensores.ev();
sensores.ev();
```

```
Serial.println("Electrovalve OFF !!!");
 delay(500);
Serial.println("Centrifugal pume PWM test duty up...");
for (int i=10;i<=255;i++)
   {
     sensores.pwm2bombaCen(i);
     delay(25);
     }

Serial.println("Centrifugal pume PWM test duty down...");
for (int y=254;y>0;y--)
   {
     sensores.pwm2bombaCen(y);
     delay(25);
     }
     Serial.println("Centrifugal pume PWM test OK...");
     delay(750);
sensores.pwm2bombaCen(0);
Serial.println("Centrifugal pume PWM set to zero ");
x=1;
Serial.println("Peristaltic pump clockwize slow test ");
myStepper2.setSpeed(50);
  digitalWrite(6, HIGH);
  motor();
  Serial.println("Peristaltic pump counterclockwize fast test ");
myStepper2.setSpeed(100);
  motor();
  digitalWrite(6, LOW);
Serial.println("Peristaltic pump Enable OFF ");
test=1;
Serial.println("All test done succesfully");
delay(2000);
Serial.println("STARTING MAIN PROCESS...");
}

  pantalla.PantallaIncio();

    tiempo = millis();

    if (tiempo - tiempoAnterior > intervalo)
    {


      tiempoAnterior = tiempo;


      Serial.print(sensores.TemperaturaLoop());
      Serial.print(sensores.humedad());
      Serial.print(sensores.LDR());
      sendTemperatura = sensores.TemperaturaLoop();
```

```
    sendHumedad = sensores.humedad();
    sendLux = sensores. LDR();



    sendData.EnviarDatosSU(sendTemperatura, sendHumedad, sendLux);
    //sendData.EnviarDatosLOOP();
    // delay (10000);


 //delay(200);
 //  }

 //      if (sendTemperatura > 25)
 //    {
 //     if (sendTemperatura < 27) {
 //
 //      } else if (sendTemperatura > 28)
 //     {
 //        vVentilador = 250;
 //       sensores.pwm1Ventilador(vVentilador);
 //      } else
 //     { vVentilador = 0;
 //       sensores.pwm1Ventilador(vVentilador);
 //      }
 //    }


   }

}


****Barras.php***

<?php
        require_once "php/conexion.php";
?>


<div id="graficaBarras"></div>

<script type="text/javascript">

        var data = [
        {
                x: ['giraffes', 'orangutans', 'monkeys'],
                y: [20, 14, 23],
                type: 'bar'
        }
```

```
        ];

        Plotly.newPlot('graficaBarras', data);
</script>

***Idenx.php***

<!DOCTYPE html>
<html>

<head>
   <title>Gráficos con plotly</title>
   <link rel="stylesheet" type="text/css"
href="librerias/bootstrap/css/bootstrap.css">
   <script src="librerias/jquery-3.4.1.min.js"></script>
   <script src="librerias/plotly-latest.min.js"></script>
</head>

<body>
   <div class="container">
      <div class="row">
         <div class="col-sm-12">
            <div class="panel panel-primary">
               <div class="panel panel-heading">
                  Trabajo Final de Master de Ingeniería Superior de
Teleocmunicaciones
               </div>
               <div class="panel panel-body">
                  <div class="row" >

                     <div class="col-sm-6 ">
                        <div id="cargaLineal"></div>
                     </div>
                     <div class="col-sm-6">
                        <div id="cargaLineal2"></div>
                     </div>

                  </div>
                     <div class="row">
                      <style type="text/css">
                      hr.new1 {border: 1px solid #1F78B4}</style>
                      <hr class="new1">
                        <div class="col-sm-12">
                           <div class="panel panel-body">
                              <div class="row">
                                 <div class="col-sm-6">
                                    <div id="cargaBarras"></div>
                                 </div>
                                 <div class="col-sm-6">
                                    <div id="cargaBarras2"></div>
```

```
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>

</html>

<script type="text/javascript">
    $(document).ready(function() {
        $('#cargaLineal').load('lineal.php');
        $('#cargaLineal2').load('lineal2.php');
        $('#cargaBarras').load('barras.php');
        $('#cargaBarras2').load('barras.php');
    });
</script>


****Lineal.php***

<?php
require_once "php/conexion.php";
$conexion=conexion();
$sql="SELECT humidity,date from dht11";
$result=mysqli_query($conexion,$sql);

        $valoresY=array();//Humedad
        $valoresX=array();//Fecha
        while($ver=mysqli_fetch_row($result)){
                $valoresY[]=$ver[0];
                $valoresX[]=$ver[1];
        }

        $datosX=json_encode($valoresX);
        $datosY=json_encode($valoresY);

        ?>
        <div id="graficaLineal"></div>

        <script type="text/javascript">
                function crearCadenaLineal(json){
                        var parsed = JSON.parse(json);
                        var arr = [];
                        for(var x in parsed){
```

```
                        arr.push(parsed[x]);
                    }
                    return arr;
                }

        </script>

        <script type="text/javascript">

                datosX=crearCadenaLineal('<?php echo $datosX ?>');
                datosY=crearCadenaLineal('<?php echo $datosY ?>');

var data=[{
        x: datosX,
                    y: datosY,
                    type: 'scatter',
                    line:{
                        color: 'red',
                        width:0.5
                    },
                    marker: {
                        color: 'red',
                        size:5
                    }

}];

var layout = {
        title: 'Humedad',
        font:{
                family: 'Raleway, sans-serif'
        },
        xaxis: {
                tickangle:-45,
                title: 'Fecha'
        },
        yaxis:{
                zeroline:false,
                gridwidth:2,
                title: 'Humedad en %'
        },
        bargap :0.05

};
                /*var trace1 = {
                        x: datosX,
                        y: datosY,
                        type: 'scatter'
                };
/*
```

```
                var trace2 = {
                        x: [1, 2, 3, 4],
                        y: [16, 5, 11, 9],
                        type: 'scatter'
                };
*/
                //var data = [trace1];

                Plotly.newPlot('graficaLineal', data,layout);
        </script>
        ***Lienal.h****

        <?php
require_once "php/conexion.php";
$conexion=conexion();
$sql="SELECT temperature,date from dht11";
$result=mysqli_query($conexion,$sql);

        $valoresY=array();//Temperatura
        $valoresX=array();//Fecha
        while($ver=mysqli_fetch_row($result)){
                $valoresY[]=$ver[0];
                $valoresX[]=$ver[1];
        }

        $datosX=json_encode($valoresX);
        $datosY=json_encode($valoresY);

        ?>
        <div id="graficaLineal2"></div>

        <script type="text/javascript">
                function crearCadenaLineal2(json){
                        var parsed = JSON.parse(json);
                        var arr = [];
                        for(var x in parsed){
                                arr.push(parsed[x]);
                        }
                        return arr;
                }

        </script>

        <script type="text/javascript">

                datosX=crearCadenaLineal2('<?php echo $datosX ?>');
                datosY=crearCadenaLineal2('<?php echo $datosY ?>');

var data=[{
        x: datosX,
```

```
                        y: datosY,
                        type: 'scatter',
                        line:{
                                color: 'red',
                                width:0.5
                        },
                        marker: {
                                color: 'red',
                                size:5
                        }

}];

var layout = {
        title: 'Temperatura',
        font:{
                family: 'Raleway, sans-serif'
        },
        xaxis: {
                tickangle:-45,
                title: 'Fecha'
        },
        yaxis:{
                zeroline:false,
                gridwidth:2,
                title: 'Temperatura en ยนC'
        },
        bargap :0.05

};
                /*var trace1 = {
                        x: datosX,
                        y: datosY,
                        type: 'scatter'
                };
/*
                var trace2 = {
                        x: [1, 2, 3, 4],
                        y: [16, 5, 11, 9],
                        type: 'scatter'
                };
*/
                //var data = [trace1];

                Plotly.newPlot('graficaLineal2', data,layout);
        </script>



        ****Tabla-php***
```

```
	<?php
/*https://www.youtube.com/watch?v=nPAp-gT5gPI En este video está como se crea esta tabla conectando a la base de datos*/
/*El fichero .php se abre en explore directamente, peor en Chrome se abre así: http://localhost/tabla/index_TFM.php quitando la carpeta htdocs*/
	$conexion=mysqli_connect('localhost','root','','temperatura');

 ?>


<!DOCTYPE html>
<html>
<head>
	<meta charset = "UTF-8">
	<title>Mostrar Datos</title>
		<link rel="stylesheet" type="text/css"  href="estilo.css">
</head>
<body>
<div id="main-container">
<h1>Temperatura y Humedad Relativa TFM de Telecomunicaciones</h1>
<br>

	<table border="1" >
		<tr>
			<td>Temperatura</td>
			<td>Humedad</td>
			<td>Fecha</td>
		</tr>

		<?php
		$sql="SELECT * from dht11";
		$result=mysqli_query($conexion,$sql);

		while($mostrar=mysqli_fetch_array($result)){
		 ?>

		<tr>
			<td><?php echo $mostrar['temperature'] ?></td>
			<td><?php echo $mostrar['humidity'] ?></td>
			<td><?php echo $mostrar['DATE'] ?></td>
		</tr>
	<?php
	}
	 ?>
	</table>

</body>
</html>
```