# ESCUELA POLITÉCNICA NACIONAL

## FACULTAD DE INGENIERÍA ELECTRÍCA Y ELECTRÓNICA

## SISTEMA DE RIEGO PARA INVERNADEROS CON INTERFAZ WEB UTILIZANDO UN ARDUINO YUN

## ADQUISICIÓN DE DATOS CON ARDUINO YUN

**TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO/A EN TELECOMUNICACIONES**

**FRANCISCO XAVIER VALDEZ LOVATO**

**francisco.valdez@epn.edu.ec**

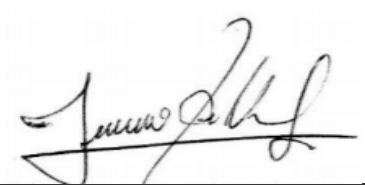
**DIRECTOR: RAMIRO E. MOREJÓN T.**

**ramiro.morejon@epn.edu.ec**

**DMQ, Octubre 2022**

# CERTIFICACIONES

Yo, FRANCISCO XAVIER VALDEZ LOVATO declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.



**FRANCISCO XAVIER VALDEZ LOVATO**

Certifico que el presente trabajo de integración curricular fue desarrollado por FRANCISCO VALDEZ, bajo mi supervisión.

**RAMIRO E. MOREJON T.**

**DIRECTOR**

# DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

FRANCISCO XAVIER VALDEZ LOVATO

RAMIRO E. MOREJON T.

# DEDICATORIA

*A mis padres que con su infinito amor y confianza me han permitido lograr todos mis objetivos incluido este.*

*A mi hermana que ha sido mi compañera incondicional desde siempre.*

*A mi abuelita Isabel que con su amor incondicional me ha ayudado a seguir adelante y nunca rendirme.*

Francisco

# AGRADECIMIENTO

A mi madre, que ha sido mi mayor fuente de inspiración, ha sido la persona que siempre ha estado para mí; a pesar de que había momentos difíciles sabía que ella estaba ahí ayudándome en todo lo que estuvo a su alcance, cuidándome, preocupándose, aconsejándome y motivándome para seguir el mejor camino y lograr siempre objetivos cada vez más altos, nunca dudo que pueda lograrlos y le agradezco el infinito amor y confianza que siempre ha puesto sobre mí.

A mi padre, que me ha brindado su apoyo incondicional y las herramientas necesarias para que pueda lograr este objetivo, ha sido una persona importante en toda esta travesía siempre confiando que lo lograría, estando pendiente de cada paso que daba y recordándome que siempre podía contar con él.

A mi hermana, que ha sido un impulso en los momentos mas difíciles, ha sido quien a estado conmigo largas noches haciéndome compañía y me recordaba que rendirse no era una opción, le agradezco por todo su cariño y por aguantar en ocasiones mis malos humores, ha sido mi compañera durante toda mi vida y en especial en esta etapa de la universidad.

A mi abuelita Isabel, que ha sido la persona que me ha brindado el amor mas incondicional de todos, le agradezco por haberme cuidado siempre, por haberme apoyado en todo y por siempre tener una palabra de aliento o un abrazo que me hacia olvidar cualquier problema que pudiese tener y que no podía no brindarle esta satisfacción.

A mis amigos de la universidad Xavier, Cristian y Alejandra que han sido las personas que han sufrido conmigo cada momento difícil de la universidad y que siempre nos hemos apoyado para que podamos cumplir todos con este objetivo.

A mis amigos de toda la vida Esteban, Francisco, Juan, María, Juan Carlos, Patricio y Andrés les agradezco a todos por el apoyo incondicional y confianza que siempre han puesto sobre mí, por las palabras de aliento y por los momentos que hicieron que la vida universitaria fuera una experiencia inolvidable.

A mi otra hermana, mi pequeña Pau quien me brindo su apoyo y cariño desde siempre, motivándome para que cumpla este y todos mis objetivos y demostrándome que una persona desconocida puede convertirse en una hermana siendo una razón más para no rendirme.

A Luis, que en los momentos de mayor estrés me brindaba una mano en todo lo que podía permitiéndome enfocarme en la universidad para que pueda cumplir este objetivo siendo muchas veces la persona que me hacía compañía en días de arduo trabajo.

A toda mi familia y amigos que no nombre pero que han sido parte muy importante de este viaje, que con una palabra de apoyo o una muestra de confianza me llenaban de fuerzas y ganas de seguir adelante y poder ahora decirles gracias y esto también es para ustedes.

Finalmente, a la Escuela Politécnica Nacional y en especial al Ing. Ramiro Morejón por su correcta orientación y apoyo durante el desarrollo de este proyecto permitiéndome concluir mi etapa en esta prestigiosa universidad de la mejor manera posible.

# ÍNDICE DE CONTENIDO

# RESUMEN

Este trabajo integrador trata sobre un sistema de riego para invernaderos con interfaz web utilizando la plataforma Arduino YUN. Este componente en concreto se centra en la adquisición de los parámetros ambientales que influyen dentro del invernadero como son temperatura, humedad y luminosidad. Para este propósito se utilizó la plataforma Arduino UNO en conjunto con el Dragino YUN shield, además de un arreglo de sensores que miden los parámetros ambientales, el módulo sensor de corriente/voltaje INA3221 y el multiplexor TCA9548A, estos dos últimos trabajan con el protocolo I2C.

Se presentan conceptos referentes a sistemas de riego, la plataforma Arduino, los sensores utilizados y en que consiste el protocolo I2C. Además, se presenta el diseñó de un sistema que integra los sensores y el módulo INA3221 en un solo dispositivo. También se presenta el diseño del algoritmo que maneja la adquisición y organización de los datos obtenidos del sistema de sensores que se evaluara dentro de un ambiente controlado, con el propósito de observar variaciones en los parámetros ambientales y probar su funcionamiento.

Con el propósito de verificar la correcta adquisición de los datos, se utilizaron equipos especializados en medir temperatura, humedad y luminosidad para calibrar los sensores utilizados y comparar los datos obtenidos con el modelo final propuesto. Se hizo uso del protocolo I2C en conjunto con un módulo sensor de corriente/voltaje INA3221 y el multiplexor TCA9548A que trabajan con este protocolo para hacer uso eficiente de los pines del Arduino y que este sea un sistema escalable en cuanto al número de sensores posibles.

**PALABRAS CLAVE:** sensores, Arduino Yun, sistema de riego, I2C.

# ABSTRACT

This final degree project deals with an irrigation system for greenhouses with a web interface using Arduino YUN platform. This component focuses specifically on the acquisition of the environmental parameters that influence the greenhouse such as temperature, humidity, and luminosity. For this, the Arduino UNO platform in conjunction with Dragino YUN shield were used, also sensors array that measures the environmental parameters, current/voltage meter modules INA3221 and the multiplexer TCA9548A. These last two work with I2C protocol.

Concepts related to irrigation system, Arduino platform and sensors used are presented and what I2C protocol consist of. Moreover, the design of a system that integrates the sensors with the INA3221 module in a single device is presented. Also, an algorithm that manage the acquisition and organization of the data obtained from the sensors system to be evaluated within a controlled environment to observe the variation in the acquired parameters and probe its performance.

To verify the correct acquisition of the data, specialized equipment to measure temperature, humidity and luminosity were used to calibrate the sensor used in this project and compare the data obtained with the proposed final model. Besides, I2C protocol was used in conjunction with a current/voltage meter modules INA3221 and multiplexer TCA9548A that work with this protocol to improve the efficiency use of the Arduino pins and make it a scalable system for the number of sensors



**KEYWORDS:** sensors, Arduino Yun, irrigation system, I2C.

# 1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Un sistema de riego se define como un sistema conformado por múltiples componentes estructurados de tal forma que permiten abastecer de agua a los cultivos y controlar los parámetros ambientales del invernadero donde estos habitan con el fin de brindar un ambiente óptimo para su crecimiento [1].

Actualmente, la automatización de un sistema de riego en un invernadero es casi indispensable para cualquier cultivo brindándole el entorno adecuado para su crecimiento independiente de la estación en la que se encuentre fuera del invernadero, protegiendo a los cultivos de heladas, vientos fuertes, plagas, etc [2]. Dentro del invernadero se obtiene un microclima artificial que permite obtener la mayor productividad de los cultivos en el menor tiempo posible. La automatización de un sistema de riego en un invernadero consiste en monitorear temperatura, humedad y cantidad de luz solar principalmente, ya que, estos son factores que afectan directamente en el crecimiento de los cultivos y si alguno de estos parámetros sale del rango de valores ideales, accionar un sistema que los regule [1].

Un dispositivo que permite la automatización de un sistema de riego automático es la plataforma Arduino. Esta es una potente plataforma de desarrollo de código abierto que permite controlar todo un sistema automático de riego de manera sencilla e interactiva [3]. Existen diferentes tipos de Arduino con características propias de cada uno y para este proyecto se previó utilizar el Arduino Yun, no obstante, la empresa fabricante a postergado la entrega de esta plataforma por lo que el componente se desarrolló utilizando el Arduino Uno en conjunto con el shield Dragino Yun.

El Arduino UNO en conjunto con el Shield Dragino YUN es una plataforma de desarrollo que se basa en un sistema Linux. Su principal ventaja es la posibilidad de conectarlo de manera sencilla a una red inalámbrica o cableada gracias al panel web Yún; este panel permite configurar al Arduino y cargar las diferentes actualizaciones del algoritmo desarrollado sin necesidad de conectarlo directamente a la PC [4].

A pesar de que el Arduino Yun es una plataforma bastante potente, tiene como limitación el número de pines que dispone, esto limita el número de sensores que se pueden utilizar [4]. Para solucionar este inconveniente se optó por el uso del sensor de corriente/voltaje INA3221 que incorpora 3 canales de censado y permite obtener 3 mediciones diferentes de manera simultánea, y el multiplexor TCA9548A que permite expandir aún más el número de sensores posibles; ambos trabajan con el protocolo I2C que permite utilizar únicamente 2 pines para la comunicación con la plataforma Arduino.

El protocolo I2C permite comunicar dos o más dispositivos a través de dos cables por su arquitectura maestro-esclavo, donde el maestro es quien dirige la comunicación solicitando o enviando datos según requiera; además permite obtener la confirmación de datos recibidos dentro de la misma trama lo que lo diferencia de la comunicación clásica serial. Es posible tener hasta 127 esclavos siempre y cuando tengan una dirección diferente, ya que, no se puede manejar dos dispositivos con la misma dirección en un mismo puerto I2C [5].

Los módulos INA3221 permiten utilizar sensores que no sean compatibles con I2C y en su lugar sensores más comerciales que basen su funcionamiento en una variación de voltaje según varía el parámetro ambiental que se esté midiendo. En la mayoría de los casos estos sensores son menos costosos que los que incorporan la tecnología I2C y poseen una precisión similar [6].

Cabe mencionar que dentro del plan de trabajo para este proyecto se definió un conjunto de variables que se utilizarían dentro de cada uno de los componentes que lo conforman, estas variables son las que almacenan los datos de luminosidad, temperatura, humedad relativa, humedad del suelo y estampa de tiempo. Dentro de este trabajo se detallan las variables que se manejan en este componente y como se almacenan los datos en cada una de estas.

Por lo antes mencionado, este proyecto se enfoca en la implementación de un sistema de riego automático tradicional, pero con funciones de nueva generación. Este permitirá obtener datos en tiempo real, manejarlos desde servidores locales y desde la nube. Este componente en concreto se concentra en la adquisición y validación de datos de parámetros ambientales tales como temperatura, humedad y luminosidad, haciendo uso de la plataforma Arduino UNO con conjunto con el shield Dragino YUN, un arreglo de sensores, el módulo INA3221 y el multiplexor TCA9548A que trabajan con el protocolo I2C.

## 1.1 Objetivo general

Desarrollar un subsistema de adquisición de datos para un invernadero utilizando la plataforma de desarrollo Arduino Yun.

## 1.2 Objetivos específicos

1. Estudiar la estructura de la plataforma Arduino Yun y sus herramientas de desarrollo.

2. Definir una estructura de variables para almacenar los parámetros físicos obtenidos de los sensores y permitir la interacción con otros módulos.
3. Elaborar las funciones que permitan realizar la adquisición de los datos de los sensores.
4. Establecer el mecanismo de adquisición y validación de datos.

## 1.3 Alcance

La plataforma Arduino YUN está constituida por dos procesadores, el Atmega32u4 y el Atheros AR9331; para este componente se utilizó el Atmega32u4 para las tares de adquisición de datos con la ayuda de sensores ambientales, el módulo INA3221, el multiplexor TCA9548A y el protocolo I2C; estos tres últimos permiten optimizar el uso de los pines del Arduino brindando la posibilidad de utilizar un número mayor de sensores.

Para poder hacer uso de los sensores en conjunto con el sensor INA3221 y la plataforma Arduino se realizó la calibración de los sensores con la ayuda de equipos especializados en medir los parámetros ambientales de interés.

Para la adquisición de datos se desarrolló un algoritmo con varias funciones que permiten adquirir y organizar los datos obtenidos para poder presentarlos en una pantalla LCD y contrastarlos con los datos obtenidos con equipos utilizados para la calibración previa.

## 1.4 Marco teórico

### 1.4.1 Sistema de riego

Las plantas en los invernaderos necesitan absorber agua del suelo para crecer y desarrollarse; al no tener un sistema especializado que monitorice el nivel de humedad del suelo, este podría saturarse lo que dificultaría la absorción de agua por parte de las plantas en los invernaderos [7]. Para este propósito se emplean sistemas de riego, que son un conjunto de estructuras que permiten que en un área determinada se pueda cultivar y se tenga el suministro de agua necesario [8].

Los sistemas de riego se utilizan tanto para cultivos que se encuentran al aire libre como para cultivos que se encuentran en un ambiente controlado, en donde es necesario limitar la cantidad de agua que se suministra a cada cultivo al ser un espacio reducido. A estos ambientes controlados se los llama invernaderos.

### 1.4.2 Invernadero

Un invernadero es un lugar cerrado, que generalmente cuenta con una cubierta translucida de vidrio o plástico, en su interior se encuentran diferentes tipos de cultivos que habitan en lo que se denomina microambiente. Un microambiente se define como un entorno en donde se tiene control sobre los parámetros ambientales que influyen directamente sobre los cultivos como son temperatura, humedad y luminosidad [2]. Dentro de los invernaderos generalmente se implementan sistemas de riego con el fin de tener un ambiente totalmente automatizado y controlado.

**Figura 1.1** Invernadero tradicional [2].

#### 1.4.2.1 Sistemas de riego en invernaderos

Aparte de tener un ambiente controlado, uno de los factores clave dentro de un invernadero es el sistema de riego. Actualmente se dispone de varios tipos de sistemas de riego que se adaptan a cualquier tipo de cultivo brindando una solución específica para cada uno de estos. Es necesario conocer todos los detalles sobre el invernadero y lo que se cultivara dentro del mismo para saber con exactitud cuál es el sistema de riego que más conviene implementar [9].

#### 1.4.2.2 Tipos de sistema de riego en invernaderos

##### 1.4.2.2.1 Riego por goteo

Se denomina también como riego localizado o gota a gota y permite aplicar agua, abonos o agroquímicos de una manera óptima a los cultivos dentro de invernaderos haciendo uso diferentes tipos de emisores o goteros según la necesidad [9]. Dentro de los principales tipos de emisores se tiene:

- **Goteros auto compensantes**

Este tipo de emisores ofrecen un caudal de agua fijo con un nivel de presión dentro de un rango medio alto. Su característica principal es que todos los emisores dentro de la misma línea de riego tienen la misma presión evitando que los últimos emisores tengan un nivel de presión más bajo a los que se encuentran al inicio debido a la caída de la presión en la tubería [10].

- **Goteros anti drenantes**

Este tipo de goteros se cierran de forma automática al detectar que el nivel de presión en el sistema de riego disminuye, esto evita el vaciado de la línea de riego una vez concluido este. Esto evita que el sistema se llene de aire y optimiza el uso de la bomba de riego debido a que no debe iniciar el sistema para empezar a funcionar [10].

- **Goteros regulables**

Este tipo de goteros permiten regular el caudal de agua a través de un mando mecánico y su principal ventaja es que se adapta fácilmente a cualquier superficie, reduce las pérdidas y desperdicio de agua al ser un sistema totalmente controlado; esto permite mantener un nivel constante de humedad en el suelo sin generar charcos ni estancamientos de agua [9] [10].

### 1.4.2.2.2 Riego por microaspersión

Este tipo de sistema de riego es una variación del sistema de riego por aspersión que consiste en aplicar al cultivo agua en forma de llovizna mediante el uso de aspersores [11]. La diferencia entre riego por microaspersión y el riego por aspersión es el alcance que cubren los aspersores, al ser el invernadero un area limitada requiere que el alcance sea menor y lo compensa con un mayor tamaño de gota para que se pueda abastecer de suficiente agua a todos los cultivos [12]. Su alcance es de aproximadamente 2 metros, pero depende directamente de la presión de agua que se tenga y la boquilla que se utiliza. Cuenta con un rotor o deflactor giratorio, que permite aumentar el diámetro de cobertura [9]. Este tipo de riego es ideal para cultivos de baja, mediana altura y que sean de bajo volumen.

### 1.4.2.2.3 Riego con difusores

Este tipo de riego emplea difusores, que a diferencia de los aspersores y micro aspersores posee un deflector fijo por lo que su diámetro de cobertura es más reducido; generalmente se lo emplea para aumentar la humedad relativa de sectores específicos dentro del

invernadero como por ejemplo los pasillos [9]. La presión de agua que aplica depende de la presión que se le permita, ya que, cuenta con un sistema de regulación mecánico que permite aumentar o disminuir la presión de agua. Además, existe en el mercado una gran variedad de ángulos de salida de agua con el cual se puede diseñar el sistema de riego para que cubra todos los ángulos dentro del invernadero [9].

#### 1.4.2.2.4 Riego subterráneo

Este tipo de riego consiste en una línea de riego perforada que va enterrada a una cierta profundidad, la cual, depende del tipo de cultivo y del tipo de suelo que se esté utilizando. Este método garantiza que el agua llegue directamente a las raíces de los cultivos, pero se debe hacer un estudio previo para calcular correctamente la profundidad [9].

#### 1.4.2.2.5 Riego con manguera

Este tipo de riego es el tradicional que se utiliza en la mayoría de los cultivos artesanales al aire libre. Consiste en utilizar una manguera que es manipulada por un operario lo que resulta ineficiente para un invernadero, ya que, lo que se busca es tener un sistema totalmente automatizado, además es muy difícil conseguir que el riego sea uniforme por el tipo de salida de agua que provee la manguera y por el factor humano [9].

#### 1.4.2.2.6 Riego hidropónico

Este sistema de riego se caracteriza por combinar varios sistemas de riego y complementarlos para proveer a los cultivos de soluciones nutritivas que son balanceadas de acuerdo con su tipo, lo que ayuda en su crecimiento [9]. Estos cultivos pueden crecer directamente dentro de una solución mineral, un sustrato o un medio inerte.

Existen diferentes tipos de sistemas hidropónicos de acuerdo con el medio en donde crecen los cultivos, y estos son:

- **Sistemas hidropónicos en medio líquido**

Utilizando este sistema los cultivos crecen directamente sobre el agua mezclada con soluciones minerales necesarias para su crecimiento. Se pueden utilizar diferentes sistemas portadores cuando se utiliza este tipo de riego como el sistema de bandejas flotantes, sistema de hidroponía de flujo profundo y sistemas por lámina de agua [13].

- **Sistemas hidropónicos en sustrato**

En este sistema se utilizan sustratos inertes en conjunto con un riego por goteo o subirrigacion [13]. Estos sustratos poseen minerales que el suelo al natural no posee y ayuda a un mejor crecimiento de los cultivos.

Entre los sustratos más comunes se puede encontrar a la perlita, la lana de roca, la fibra de coco y la turba. Además, para este tipo de sistema se tienen diferentes métodos portadores como son en bancadas o surcos, en saco, en contenedores individuales o canales y en superficie, que comúnmente son enarenados [13].

- **Sistemas aeropónicos**

En este sistema la raíz de los cultivos se encuentra al aire libre dentro de un contenedor que la mantiene en la oscuridad, donde se le aplica una solución nutritiva en estado gaseoso que se asemeja a una niebla [13]. Este tipo de sistemas generalmente se utilizan para cultivos de alta rentabilidad y que necesitan un elevado nivel de control del proceso de producción, como por ejemplo cultivos de tomate, pimiento o fresas.

#### 1.4.2.2.7 Riego por nebulización

Este sistema de riego es el más utilizado en invernaderos y semilleros. Consiste en la aplicación de agua a través de un sistema de nebulizadores que poseen orificios de muy corto diámetro lo que ocasiona que, al pasar el agua a través de estos, choque con las paredes y se expulse en forma nebulizada; para conseguir esto, el sistema de riego utiliza una presión de agua relativamente alta de entre 2-4 bares [9]. Este tipo de sistemas además de utilizarse para riego también son un mecanismo para aumentar la humedad relativa del invernadero, lo que permite regular la temperatura y en conjunto con un sistema de ventilación refrigerar el invernadero de ser necesario [14]. Además, mediante este método se puede aplicar abonos solubles en agua o fitosanitarios.

### 1.4.2.3 Principales variables ambientales dentro de un invernadero

#### 1.4.2.3.1 Temperatura relativa

La temperatura se define como una propiedad que tienen los sistemas para determinar si estos se encuentran en equilibrio térmico [15]. También se la define como una magnitud que mide la cantidad de energía térmica que tiene un cuerpo [16]. Pero, como una definición más coloquial se puede decir que la temperatura es una magnitud que nos permite saber que tan frio o que tan caliente se encuentra un sistema.

Una vez definido lo que es temperatura, se entiende como una cantidad relativa a una magnitud que fue descrita tomando como referencia una medición o suceso, en el caso de la temperatura se toma en cuenta el punto de congelación y ebullición de una sustancia que generalmente es el agua para establecer un rango [17].

Tomando en cuenta lo antes mencionado, la temperatura relativa dentro de un invernadero nos permite conocer que tan frio o caliente se encuentra este.

#### 1.4.2.3.2 Humedad relativa

La humedad relativa se define como la cantidad de vapor de agua que se encuentra en el aire y esta expresada en porcentaje con relación a la cantidad de vapor necesaria para saturar el aire a una determinada temperatura [18].

Dentro del invernadero este es un parámetro climático fundamental y es indispensable su control para que exista un intercambio gaseoso y que se pueda llevar a cabo el proceso de fotosíntesis [19].

#### 1.4.2.3.3 Luminosidad

La luminosidad se define como la cantidad de energía radiante proveniente del sol, esta no se puede almacenar, pero si se puede disponer de ella durante el día y los cultivos lo utilizan como fuente de energía para realizar el proceso de fotosíntesis [20]. Con el fin de asegurar el mayor aprovechamiento de energía radiante se propone diseñar el invernadero con materiales translucidos. Debido a que la luz natural solamente se puede aprovechar durante el día y en condiciones despejadas, existen dispositivos emisores de luz artificial que se pueden instalar en los invernaderos con el fin de proveer de la cantidad de luz necesaria a los cultivos independiente del entorno fuera del invernadero [20].

### 1.4.2.4 Parámetros ambientales ideales para un invernadero

- **Temperatura**

Para que los cultivos puedan crecer adecuadamente se debe proveer de una temperatura especifica de acuerdo con el tipo de cultivo que se esté trabajando; generalmente esta temperatura varia de entre 10-15° centígrados hasta los 30° centígrados [21]. Se debe en lo posible asegurar una variación de entre 5 a 7° centígrados entre la temperatura diurna y nocturna del invernadero con el fin de obtener los máximos beneficios de los cultivos [21].

- **Humedad relativa**

Lo valores de humedad relativa idóneos dependen del tipo de cultivo, pero los más favorables se encuentran entre el 50 al 75%, si se sale de este rango se pueden presentar efectos adversos como la disminución de la absorción de nutrientes si es que existe una humedad relativa alta o que los cultivos sufran de deshidratación si es que la humedad relativa es baja [19].

- **Luminosidad**

Técnicamente el valor ideal de luminosidad que requieren los cultivos al día es de entre 30000 a 100000 luxes por alrededor de 8 a 10 horas, esta es la energía que produce el sol al medio día y cuando el cielo está totalmente despejado [22]. Cabe mencionar que no todas las plantas requieren la misma cantidad de luz al día para desarrollarse adecuadamente y este es el caso del tomate que requiere alrededor de 10000-40000 luxes por alrededor de 8-10 horas [23]. Se debe tomar en cuenta que existirán días nublados en donde el aporte de energía radiante por parte del sol es escaso; cuando existan estas situaciones se debe compensar con lámparas de luz artificial [24].

### 1.4.3 Arduino

Arduino es una plataforma de código abierto que se basa en el concepto de hardware y software fáciles de utilizar. Esta plataforma es capaz de leer como entrada los datos enviados por sensores, la pulsación de un botón o un mensaje de Twitter y procesarlas con el fin de tener una salida que puede activar un motor, encender un LED o publicar algo online. Para que realice cualquier acción, esta plataforma integra un microprocesador que se puede programar mediante un set de instrucciones escrito en un lenguaje de programación propio de Arduino que está basado en Wiring, utilizando como interfaz para esto el IDE de Arduino [3].

Existen varios de tipos de Arduino desarrollados para satisfacer las necesidades que tenga cada usuario. Cada Arduino se diferencia de otro por las capacidades que cada uno posea. Para este proyecto se hizo uso del Arduino UNO, el cual se detalla en la siguiente sección

#### 1.4.3.1 Arduino UNO

Esta placa es la más utilizada y documentada de todas las que conforman la familia Arduino. Esto debido a la robustes que muestra tanto como para proyectos iniciales como proyectos más avanzados. Está formado por un microprocesador ATmega328P, este consta de 14 pines I/O digitales, de los cuales 6 pueden ser utilizados como salidas PWM,

además cuenta con 6 entradas analógicas, un resonador cerámico de 16 MHz, un puerto USB, un conector para alimentación, un cabezal ICSP y un botón de reinicio [25].

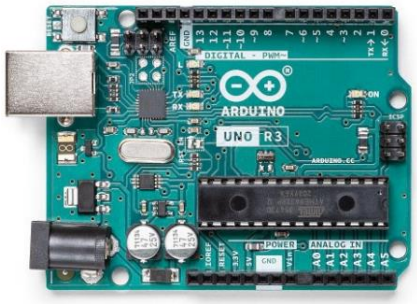



**Figura 1.2.** Arduino UNO [25]**.**

En conjunto con el Arduino UNO se utilizó el shield Dragino YUN para potenciar las capacidades de conectividad de este.

### 1.4.3.2 Yun Shield

Los shield para Arduino se definen como una placa externa que se conecta a Arduino directamente y permiten ampliar sus capacidades. Generalmente se utilizan los shield con las placas Arduino UNO o MEGA [26].

En el caso de este proyecto se hizo uso del shield Yun que permite a la placa Arduino extender sus capacidades de conectividad con el poder de un sistema operativo basado en Linux, el cual, habilitas conexiones avanzadas de red y sus aplicaciones. Entre algunas de sus características se tiene la posibilidad de conectarse a una red inalámbrica o alámbrica, esto gracias a su interfaz Web llamada Panel Yun; este permite administrar y configurar la placa Arduino a través de la red sin la necesidad de conectarla directamente al computador, en el caso de este componente se utilizó esta herramienta para actualizar las diferentes versiones del código conforme se fue modificando [27].


**Figura 1.3.** Dragino Yun Shield *[27].*

### 1.4.4 Módulo INA3221

Este módulo es un detector de corriente/voltaje que cuenta con tres canales de medición independientes. Este módulo es compatible con protocolos de comunicación I2C y SMBUS. Mide el voltaje dentro de un rango de 0-26 V y hasta 3.2 A por cada canal; se alimenta con un único suministro de voltaje de entre 2.2 a 5.5 V y cuenta con 4 direcciones programables, es decir, para nuestro modelo será posible tener hasta 4 sensores por cada parámetro ambiental antes definido [28]. A pesar de que permite un número mayor de sensores en comparación a si se hiciese uso de la comunicación serial con Arduino, uno de los objetivos de este componente es tener un sistema escalable, por lo que se optó por hacer uso de un multiplexor I2C TCA9548A.

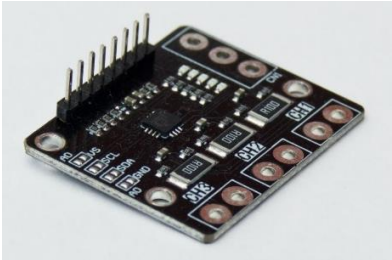


**Figura 1.4.** Módulo INA3221 [29].

### 1.4.5 Multiplexor TCA9548A

Este dispositivo permite seleccionar diferentes entradas de acuerdo con un sistema de control, esto permite expandir el número de entradas que es posible tener en un sistema. Este dispositivo cuenta con 8 direcciones posibles programables y posee 8 pares de entradas SCL y SDA. Trabaja con un voltaje de entre 3.3 a 5 V y todos los pines trabajan con un voltaje de hasta 5V [30].

Para este componente, este dispositivo permite expandir el número de sensores posibles debido a que no trabaja con las direcciones directamente sino con el canal al cual están conectados. Así es posible escalar este modelo hasta 32 sensores por cada parámetro ambiental definido en un inicio.

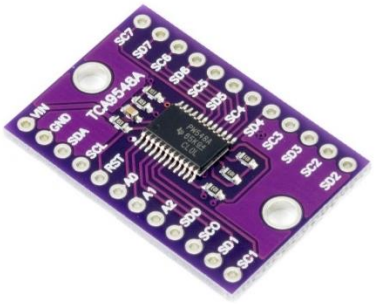



**Figura 1.5.** Multiplexor TCA9548A [31].

### 1.4.6 Protocolo I2C

I2C es un protocolo de comunicación serial, conocido también como TWI, en donde se define la trama de datos y las conexiones físicas para que pueda existir transferencia de información (bits) entre dos o más dispositivos digitales haciendo uso únicamente de dos pines de comunicación SDA, System Data que es la línea por donde se transfieren los datos entre los dispositivos, y SCL, System Clock que es la línea de los pulsos de reloj que controlan la comunicación [5].

#### 1.4.6.1 Características

Este protocolo maneja un esquema de comunicación maestro-esclavo. El maestro será quien controla la comunicación mientras que el/los esclavos serán los que reciban o envíen información según el maestro lo solicite. Con este protocolo se alcanza una velocidad de transmision de datos de 100Kbits por segundo, aunque en casos especiales se podría llegar hasta 3.4 MHz [32].

Esta comunicación es posible gracias a que se le asigna una dirección a cada dispositivo, así se puede reconocer a estos dentro del sistema. El maestro es quien inicia la comunicación, este decide con que dispositivo desea conectarse para enviar o recibir datos y así mismo terminar con la comunicación cuando la transferencia de datos termine. Se hace uso de dos resistencias de pull-up para asegurar un nivel lógico alto cuando no exista ningún dispositivo conectado al maestro [33].

La trama de un mensaje enviado sobre el protocolo I2C está conformado por los siguientes campos [5]:

- Bit de inicio
- Bit de parada
- Bit ACK o de confirmación

- Bit NACK o no confirmación
- Bit de lectura escritura R/W
- 7 o 10 bits para la dirección del dispositivo maestro/esclavo
- 8 bits de dirección (en algunos casos pueden ser 16 bits)
- 8 bits de datos

Este conjunto de parámetros puede variar de acuerdo con el modo de comunicación que se emplee, este puede ser Maestro-Transmisor y Esclavo-Receptor o viceversa.

### 1.4.6.2 Aplicaciones

Este protocolo se utiliza generalmente cuando se trabaja con sensores compatibles con este o que se comuniquen a través de una interfaz que maneje I2C. Además de esto, se lo utiliza en aplicaciones multimedia que hagan uso de equipos sincronizadores RF, codificadores y decodificadores de video y procesadores de audio [34].

### 1.4.6.3 I2C con Arduino

I2C podría aparentar ser un protocolo nuevo, pero la realidad es que yace desde el año 1982 que comenzaron a utilizarlo, actualmente se ha vuelto nuevamente popular su uso gracias a la plataforma Arduino y la llegada del internet de las cosas (IoT) debido al aprovechamiento de pines que permite [35].

#### 1.4.6.3.1 Librería Wire Arduino I2C

Esta librería es una herramienta indispensable para el uso de Arduino con el protocolo I2C, ya que, hace posible una interacción simple e intuitiva entre la placa y el protocolo, además esta librería viene instalada por defecto en el software Arduino IDE por lo que no es necesario hacer uso de un gestor de librerías [35].

Su característica principal y lo que lo hace tan simple e intuitivo es que no se necesitan crear instancias u objetos de la clase, por el contrario, se llama directamente a las funciones que se quiera utilizar, similar a lo que sería con una librería de comunicación Serial.

##### 1.4.6.3.1.1 Funciones principales

- **Wire.begin()**

Esta función inicializa a la librería Wire y permite que la placa Arduino se conecte con el bus I2C. Esta función puede o no tener un parámetro de entrada y esto depende si se

quiere utilizar a la placa como maestro o esclavo, si fuese el segundo caso se debe utilizar el parámetro address, que es la dirección que utiliza la placa para unirse al bus, como parámetro de entrada [35].

- **Wire.requestFrom()**

Esta función permite que el maestro pida datos a cualquier dispositivo esclavo pidiendo como argumento de entrada la dirección y cantidad de bits que se requiera[35].

En ambos casos esta función solicita datos a un esclavo los cuales serán leídos por la función Wire.read().

- **Wire.beginTransmission()**

Esta función inicia la transmision de datos hacia un esclavo con una dirección especifica [35].

- **Wire.write()**

Esta función permite enviar datos y se puede utilizar en dos escenarios. El primero cuando se quiere enviar datos de un esclavo a un maestro después de recibir una solicitud de datos por parte de este y el segundo escenario es cuando se quiere enviar datos de un maestro a un esclavo una vez ejecutada la función Wire.beginTransmission() [35].

Es posible utilizar como alternativa Wire.print() y Wire.println(), funciones utilizadas en la comunicación serial.

- **Wire.endTransmission()**

Esta función termina la comunicación que fue iniciada por la función Wire.beginTransmission() y transmite los bytes almacenados en el buffer mediante el uso de la función Wire.write() [35].

Esta función no requiere ningún parámetro de entrada, pero necesita que antes se haya utilizado el comando Wire.beginTransmission().

- **Wire.available()**

Esta función retorna el número de bits que están disponibles en el buffer que pueden ser leídos por el comando Wire.read() después de haber utilizado el comando Wire.requestFrom() en el caso de utilizarse en un maestro o después de recibir una petición de datos si se usa en un esclavo [35].

- **Wire.read()**

Esta función permite retornar un byte leído. Cuando se utiliza en un maestro debe ir después de utilizar la función Wire.requestFrom() mientras que si se utiliza en un esclavo esta debe estar dentro de la función de recepción de datos [35].

- **Wire.setClock()**

Esta función permite setear la velocidad de comunicación I2C. Estos valores pueden ir desde 100000 para el modo estándar hasta 400000 para el modo rápido [35].

- **Wire.onReceive()**

Este método permite establecer que función se ejecutara el momento en que se reciban los datos enviados desde un maestro a un esclavo [35].

- **Wire.onRequest()**

Este método permite establecer la función que se ejecutara en el esclavo cuando el maestro le solicite él envió de datos [35].

### 1.4.7 Sensores

Un sensor es todo aquel dispositivo que posee una propiedad que sea sensible a la variación de una magnitud física en un determinado medio, es decir, que cuando esta magnitud varie la propiedad del dispositivo muestre una variación en su intensidad o voltaje proporcional a la de la magnitud lo que hace posible medirla [36].

#### 1.4.7.1 Sensor de temperatura Chore-Tronics 40741

Este es un termistor que es un sensor de tipo resistivo que basa su funcionamiento en la variación del valor de su resistencia electrica a medida que varía la temperatura. Estos puedes ser de dos tipos, los de coeficiente de temperatura negativo o NTC por sus siglas en inglés, que a medida que la temperatura aumenta el valor de su resistencia electrica disminuye y los de coeficiente de temperatura positiva o PTC por sus siglas en inglés, que funcionan de la manera contraria a los NTC. [37]

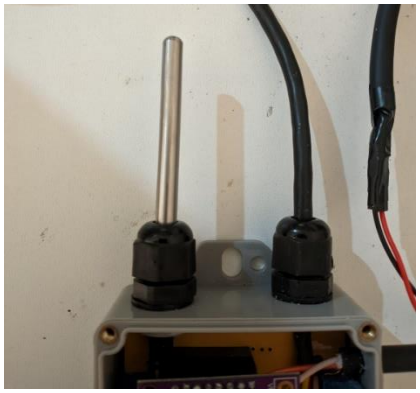


**Figura 1.6.** Sensor de temperatura Chore-Tronics 70741 [Autor].

A continuación, se muestra una lista de las especificaciones técnicas y rangos de medición del módulo [38]:

- **Voltaje de operación:** 12 V
- **Rango de medición de temperatura:** -30 – 50 ° C
- **Precisión en el sensor de temperatura:** +- 0.07° F

### 1.4.7.2 Sensor de Humedad Big Herdsman HTV597

El sensor de humedad HTV597 fabricado por la empresa “Big Herdsman” forma parte de la línea de equipos para la implementación de sistemas de ambiente controlado que fabrica la empresa. El funcionamiento de estos sensores se basa la variación del voltaje que entregan de 0 a 10 V donde la variación es proporcional a la humedad relativa dentro de un rango de 0 a 100 % con una precisión de +- 5%. Al ser un instrumento utilizado industrialmente en escenarios donde se necesita de una gran precisión no es necesario realizar una calibración previa. [39]

**Figura 1.7.** Sensor de Humedad relativa Big Herdsman [Autor].

### 1.4.7.3 Sensor de luz tipo celda/panel RY6-427

Este tipo de sensor es de tipo panel solar, que proporciona voltajes de 0 a 3V de acuerdo con el nivel de luminosidad al que este expuesto. Este está compuesto de celdas policristalinas de alta eficiencia que están montadas sobre una base de PCB y cubiertas por resina epoxi como protección para las celdas sin reducir su eficiencia. [40]

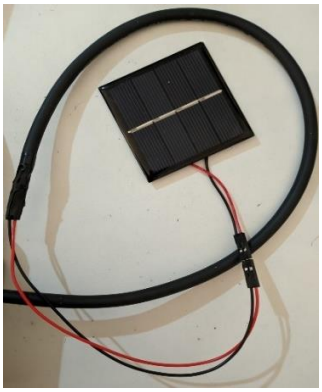


**Figura 1.8.** Sensor de luz tipo celda/panel RY6-427 [Autor].

### 1.4.8 Pantalla LCD

La LCD es una pantalla plana de cristal líquido que utiliza las propiedades de modulación de luz de este material en conjunto con polarizadores para producir imágenes en color o monocromáticas [41]. En el caso de las LCD utilizadas con Arduino son los modelos más básicos de LCD en los cuales se puede presentar caracteres básicos como son números, letras y algunos especiales. Se basa en un comportamiento de filas y columnas con dimensiones de 16x2 generalmente, es decir, se podrán mostrar 16 caracteres simultáneamente en cada una de las 2 filas [42].

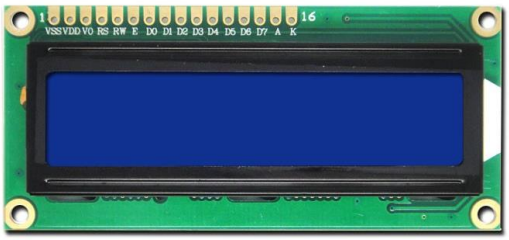



**Figura 1.9.** Pantalla LCD 16x2 [43].

# 2 METODOLOGÍA

Este trabajo se enfoca en la elaboración de un arreglo de sensores que será controlado a través de un programa desarrollado en el IDE de la plataforma Arduino. Su enfoque es

mixto; cualitativo debido a que para el análisis de resultados se hizo uso de un arreglo de sensores, definido en esta sección, y un sensor INA3221 extra simulando ser otro arreglo de sensores con la diferencia de que sus entradas son divisores de voltaje y no sensores, esto debido a que se priorizó el correcto funcionamiento del proyecto y la corrección de errores que se presentaron durante la implementación de este. Por otro lado, este trabajo también tiene un enfoque cuantitativo, ya que, no está desarrollado para manejar únicamente dos arreglos de sensores, sino que es posible utilizar hasta 32 de manera concurrente.

El protocolo I2C está concebido para reconocer un total de 127 direcciones, lo cual, implica que se podrían conectar hasta 127 módulos sensores, no obstante, el hardware asociado con los sensores no tiene la versatilidad de ser configurado en cualquiera de las 127 direcciones, sino que las direcciones disponibles están asociadas al tipo de modulo utilizado por lo que en la práctica solo se podría conectar un numero de módulos igual al número de direcciones configurables en dicho dispositivo.

Con el propósito de extender a un número mayor de arreglos de sensores se utilizó el multiplexor TCA9548a, ya que, existe una condición a cumplir para el manejo de más de uno de estos dispositivos; debido a que cada arreglo de sensores tiene como dispositivo principal al módulo INA3221 y este cuenta con 4 direcciones posibles, únicamente se podrían colocar 4 módulos al puerto I2C del Arduino, y es en este punto en donde el multiplexor TCA9548a cobra importancia y nos permite tener 4 módulos por cada uno de sus 8 canales, expandiendo el número de arreglos de sensores posibles hasta 32.

Por otro lado, este trabajo es de tipo experimental debido a que el dispositivo antes mencionado y el algoritmo de control fueron desarrollados desde cero con el propósito de cumplir con los objetivos propuestos en este trabajo y cubrir su alcance.

El tipo de recolección de datos utilizado fue el de pregunta secuencial, el cual, se explica detalladamente en la sección del algoritmo de adquisición de datos.

En la etapa de formulación del proyecto general se estableció un arreglo de variable, las cuales, se utilizarán dentro del proyecto y en cada uno de sus componentes; estas variables son las que almacenan datos de luminosidad, temperatura, humedad, humedad del suelo, estampa de tiempo. Dentro de este componente se obtienen las variables de luminosidad temperatura y humedad relativa en un principio, en la sección de conclusiones se detalla que gracias a las prestaciones del proyecto realizado se pudo implementar el hardware y software necesario para obtener los datos de la humedad del suelo sin realizar modificaciones de consideración.

Este proyecto se divide en cuatro secciones: primero, la calibración de los sensores con la ayuda de instrumentos especializados en medir temperatura y luminosidad. En el caso del sensor de humedad no se realizó calibración debido a la precisión que tiene de fábrica y a su funcionamiento. En segundo lugar, se presenta el diseño y construcción del arreglo de sensores que integra los tres sensores que medirán cada uno de los parámetros ambientales establecidos en conjunto con el módulo INA3221. Como tercer punto se tiene el diseño y construcción de la interfaz que comunica a arreglos de sensores con la plataforma Arduino y finalmente se presenta el desarrollo del algoritmo que permite almacenar, organizar, validar y presentar los datos obtenidos por el dispositivo antes mencionado.

## 2.1 Calibración de los sensores

Como se planteó anteriormente la calibración se realizado únicamente a los sensores de temperatura y luminosidad debido a que el sensor de humedad es bastante preciso y no necesita de esta. El principio para calibrar estos sensores se basó en utilizar un instrumento de medida específico para ambos parámetros ambientales y comparar el voltaje que los sensores emitían con el parámetro medido por estos instrumentos. Una vez obtenido el banco de medidas se realizado el cálculo de la media de estos valores cuando era necesario, esto debido a que en las diferentes pruebas realizadas se obtenían valores de voltaje ligeramente distintos para un mismo valor de temperatura o luminosidad.

Para realizar el cálculo de la media se utilizó la Ecuación 2.1. Se escogió este método para calcular el valor medio o promedio debido a que los valores de voltaje obtenidos con las mediciones eran bastante similares cuando se tenía un mismo valor de parámetro.

$$\boldsymbol{A = \frac{1}{n}\sum_{i=1}^{n} a_1 = \frac{a_1 + a_2 + \cdots + a_n}{n}}$$

**Ecuación 2.1** Cálculo de la media aritmética de un conjunto de datos [44] .

Utilizando la Ecuación 2.1 se pudo reducir el banco de datos para proceder a generar la curva de calibración y posteriormente a obtener la expresión matemática que represente a dicha curva y a su vez que represente al comportamiento del sensor.

### 2.1.1 Calibración del sensor de temperatura

En [38] se puede observar las especificaciones de este sensor en donde se indica que a una temperatura de 25 ºC se tiene una resistencia de 10 [Kohm]. Con este dato se pudo

conocer el valor de resistencia que se debe colocar a la salida del sensor y obtener las diferentes variaciones de voltaje de acuerdo con la variación de temperatura.

Este sensor trabaja en un amplio rango de -30 ºC hasta los 120 ºC, pero la calibración se hizo tomando en cuenta un rango de valores que van desde 5 ºC hasta los 87 ºC, esto se debe a que en [21] se puede observar que la temperatura del aire dentro del invernadero no debería bajar de los 10 ºC y no debería superar los 35 ºC, a pesar de esto y con el fin de obtener una curva de la calibración más precisa se optó por ampliar los límites del rango de calibración. Por otra parte, para realizar la calibración de este sensor se utilizó como medio de medida agua a diferentes temperaturas debido a que es más sencillo variar la temperatura de esta a variar la temperatura del aire en un amplio rango, por este motivo los valores máximos que se pudieron obtener trabajando con agua fueron de 5 ºC como temperatura mínima y 87 ºC como temperatura máxima.

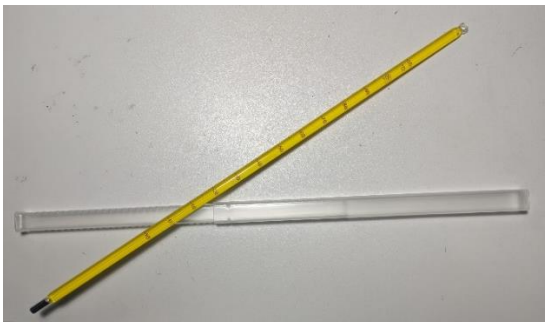


**Figura 2.1.** Termómetro de mercurio de amplio rango [Autor].

El método de calibración consistió en utilizar un termómetro de mercurio de amplio rango (de -10 ºC a 120 ºC) mostrado en la figura 2.1, para medir la temperatura del agua y relacionarlo con el voltaje emitido por el sensor Chore -Tronics 40741 a dicha temperatura. Basado en este principio se procedió a tomar diferentes medidas de agua a diferentes temperaturas durante dos días; en ambos días se tomaron la misma cantidad de medidas 100 exactamente**,** teniendo al final de los dos días 200 mediciones. Después de aplicar la Ecuación 2.1 a estos datos se obtuvo una tabla con 36 mediciones mostradas en el Anexo 1, con las cuales se formó la curva de calibración como se muestra en la figura 2.2.

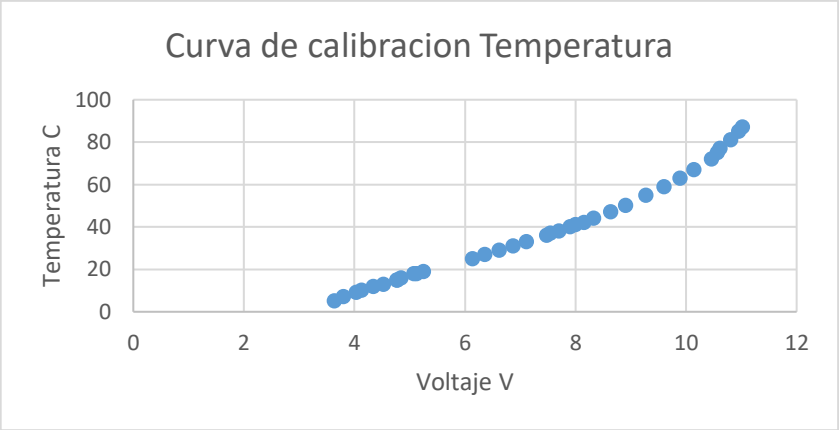



**Figura 2.2.** Curva de calibración del sensor de temperatura [Autor].

Como se puede observar en la figura 2.2, el comportamiento del sensor no es lineal por lo que asumir que la variación de voltaje es proporcional a la variación de temperatura sería erróneo. Por esta razón, se hizo uso del software Excel para general una línea de tendencia, la cual, podía ser de diferentes tipos tanto lineal, logarítmica, exponencial, entre otras, así como se puede observar en la figura 2.3.

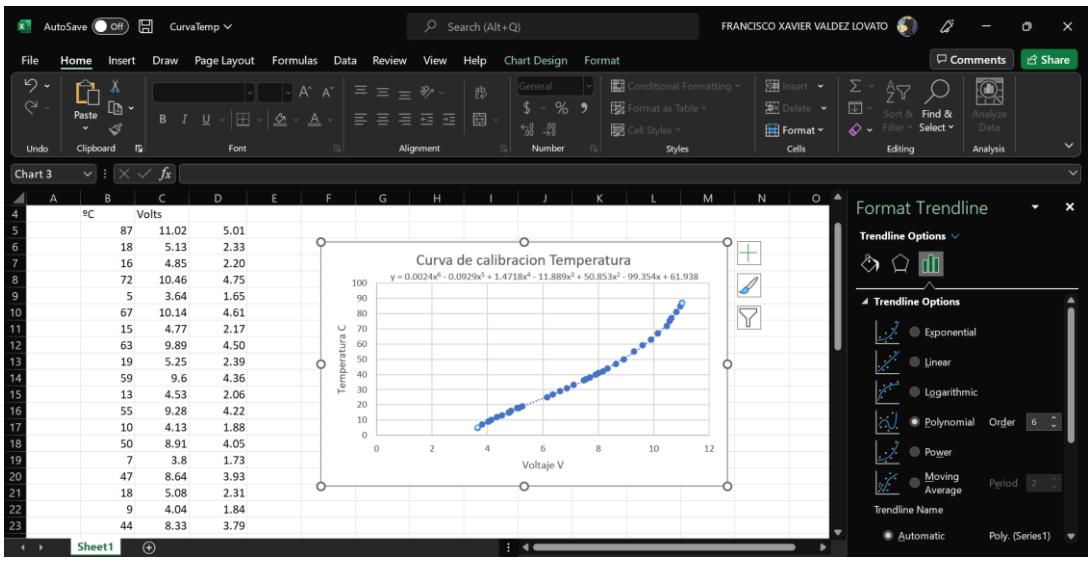



**Figura 2.3.** Generación de línea de tendencia con el software Excel [Autor].

Con la ayuda de esta herramienta se pudo generar una curva que se acople a la tendencia de los datos obtenidos y para este caso fue una curva polinómica de orden 6. En la figura 2.4 se puede observar la línea de tendencia por debajo de la curva de calibración del sensor de temperatura. Gracias a esta línea de tendencia fue posible generar una expresión matemática que la represente, y que a su vez representa el comportamiento del sensor de temperatura. La expresión obtenida se muestra en la Ecuación 2.2

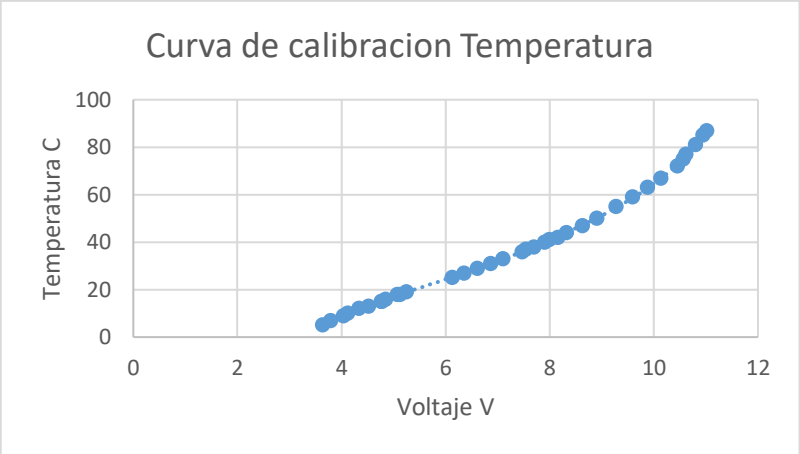



**Figura 2.4.** Línea de tendencia de la curva de calibración del sensor de temperatura [Autor].

$$y = 0.0024x^6 - 0.0929x^5 + 1.4718x^4 - 11.889x^3 + 50.853x^2 - 99.354x + 61.938$$

**Ecuación 2.2** Expresión del comportamiento del sensor de temperatura [Autor].

En la Ecuación 2.2 la variable "*y*" representa a la temperatura y "*x*" al voltaje. Gracias a esta expresión se pudo codificar el comportamiento del sensor de temperatura, así al momento de medir el voltaje obtenido de este, el código permite transformarlo en el valor de temperatura que ese nivel de voltaje represente.

### 2.1.1 Calibración del sensor de luminosidad.

Para la calibración de este sensor se utilizó el mismo método que para la calibración del sensor de temperatura, pero con algunas diferencias puntuales. En primer término, el instrumento utilizado para hacer la recolección de medidas fue un luxómetro digital tal y como se muestra en la figura 2.5. Al ser un sensor tipo celda, se realizó la adquisición de datos bajo dos ambientes: con luz natural y con luz artificial; debido a esto se obtuvieron dos curvas de calibración, una para cada ambiente. Se optó por separar los ambientes, ya que, el comportamiento del sensor era muy diferente en cada uno de estos. Esta diferencia es producida por la presencia de luz infrarroja en la luz natural.

**Figura 2.5.** Luxómetro digital [Autor].

Se recolectaron las mediciones por 2 días, se pudieron tomar 150 medidas con luz natural y 100 con luz artificial por cada día, dando un total de 300 mediciones con luz natural y 200 con luz artificial al final de los dos días. El número de medidas cambia con el ambiente debido a que al trabajar con luz natural el nivel de luminosidad es muy superior al que se tiene con luz artificial debido a la presencia de luz infrarroja; con luz natural fácilmente se sobrepasa los 100000 luxes en un día medianamente despejado, mientras que con luz artificial es bastante complicado sobrepasar los 40000 luxes, por ende, el rango de medida disminuye considerablemente.

Tomando en cuenta lo antes mencionado, se optó por un rango de medición de 0 - 150000 luxes para luz natural y de 0 - 40000 luxes para luz artificial. Una vez obtenidas las mediciones, se aplicó la ecuación 2.1 y se obtuvieron dos tablas de datos, una para cada ambiente. El número de datos de la tabla que representa al ambiente con luz natural se redujo a 59 datos y para la tabla que representa al ambiente con luz artificial se redujo a 37 datos, las dos tablas se muestran en el Anexo 2. Con estas tablas se obtuvieron las curvas de comportamiento del sensor para los ambientes analizados tal y como se muestra en las figuras 2.6 y 2.7 respectivamente.

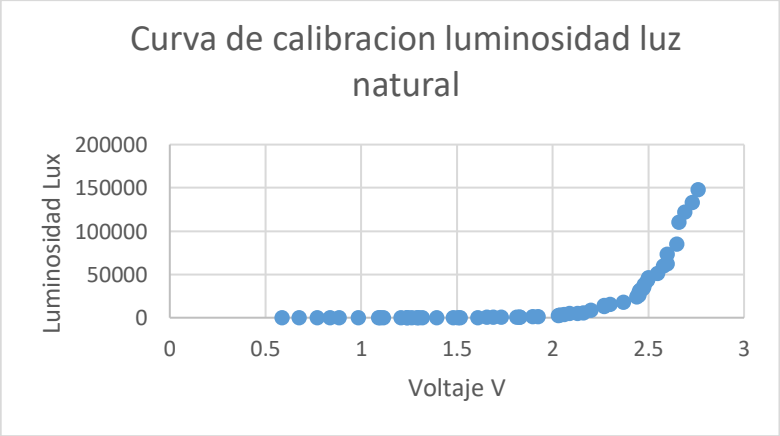



**Figura 2.6.** Curva de calibración con luz natural [Autor].

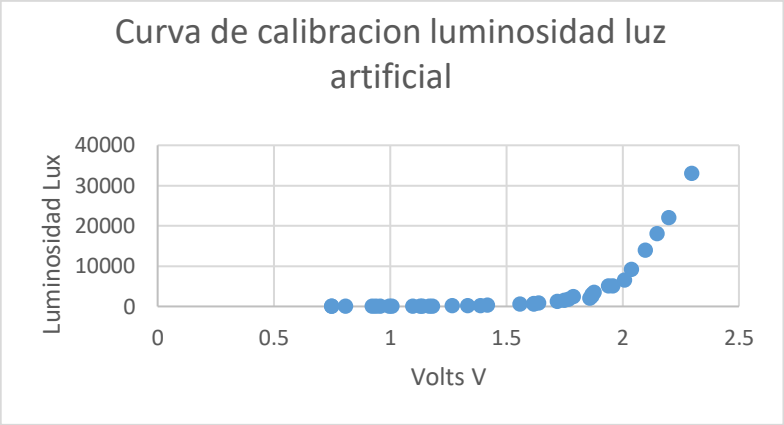



**Figura 2.7.** Curva de calibración con luz artificial [Autor].

Una vez obtenidas las gráficas 2.6 y 2.7 se obtuvo la línea de tendencia que para este caso fue de tipo exponencial por el comportamiento que mostraba el sensor tanto con luz natural como con luz artificial. En las figuras 2.8 y 2.9 se pueden observar las líneas de tendencia debajo de la gráfica de cada ambiente y además las ecuaciones 2.3 y 2.4 son las expresiones obtenidas que representan el comportamiento del sensor de luminosidad para cada uno de los ambientes.

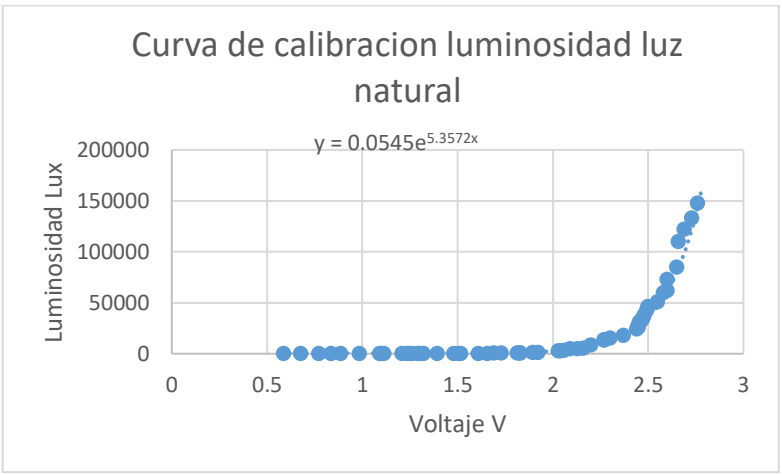



**Figura 2.8.** Línea de tendencia para la curva de calibración con luz natural [Autor].

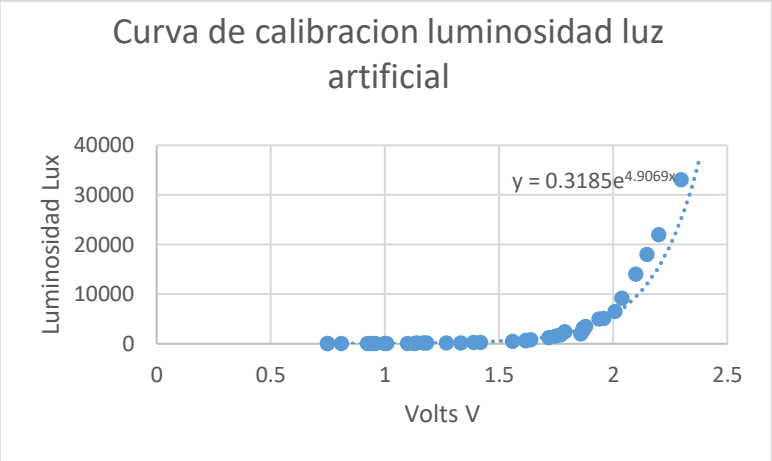



**Figura 2.9.** Línea de tendencia de la curva de calibración con luz artificial [Autor].

$$y = 0.0545e^{5.3572x}$$

**Ecuación 2.3** Expresión del comportamiento del sensor de temperatura con luz natural [Autor].

$$y = 0.3185e^{4.9069x}$$

**Ecuación 2.4** Expresión del comportamiento del sensor de temperatura con luz artificial [Autor].

A pesar de haber obtenido estas expresiones que representan el comportamiento del sensor para los dos ambientes, el rango de variación de voltaje del sensor, de 0 a 3V, es pequeño en comparación con el rango de valores de luminosidad posibles; debido a esto las expresiones encontradas no fueron tan precisas. Por esta razón se hizo un análisis de las expresiones con el fin de corregir el margen de error que se generaba y una vez corregido integrar las expresiones en el algoritmo. Dentro del algoritmo que se detallará más adelante, se puede observar la solución propuesta para corregir el margen de error en las expresiones.

## 2.2 Diseño del arreglo de sensores

El arreglo de sensores se define como la integración de todos los sensores en un dispositivo único capaz de comunicarse con la plataforma Arduino a través de dos terminales (SDA y SCL). En esta etapa se hizo uso del software Proteus para el diseño de la placa de control de este dispositivo. Esta placa se conforma por el módulo INA3221 y los sensores de humedad, temperatura y luminosidad.

Con el objetivo de que cualquier elemento de la placa pueda ser fácilmente reemplazado, no se integró ningún dispositivo directamente, sino que se hizo uso de borneras para los sensores, comunicación I2C y alimentación tanto de los sensores como la del módulo INA3221, y además se hizo uso de un header hembra de 8 pines para la conexión de este último.

Dado que en Proteus no existe un paquete que modele a la bornera se creó uno manualmente de acuerdo con las medidas comerciales de esta. En la figura 2.10 se puede observar el modelo resultante de la placa que integrara las conexiones para los sensores, el módulo INA3221 y las conexiones entre estos.

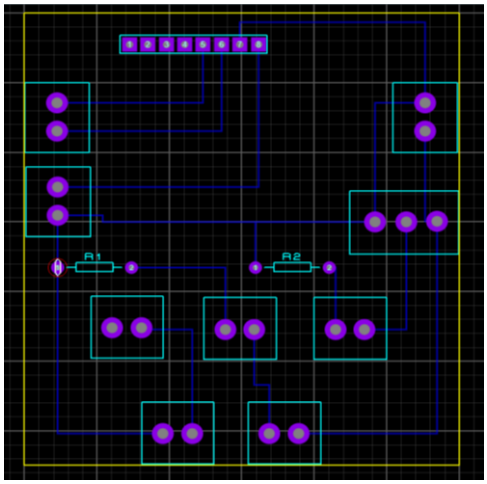



**Figura 2.10.** Diseño de la placa para el sistema de control del arreglo de sensores [Autor].

En la figura 2.11 y 2.12, se puede observar un modelo 3D tanto en la parte superior como inferior de la placa.

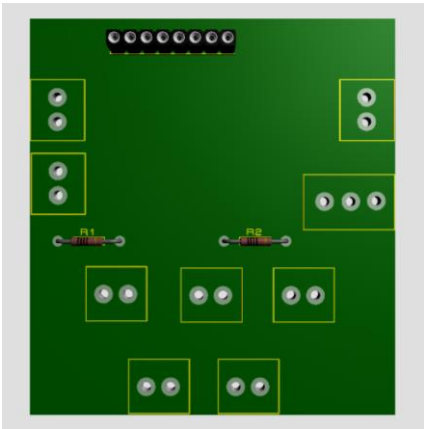


**Figura 2.11.** Parte superior de la placa de control del sistema de control del arreglo de sensores [Autor].

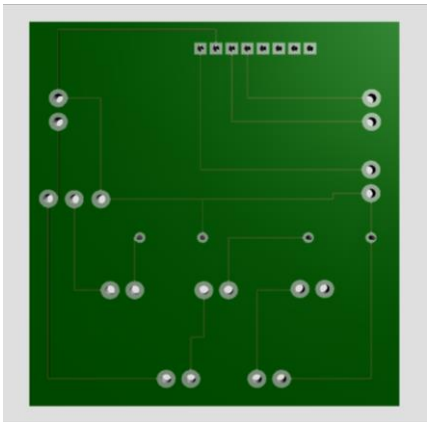


**Figura 2.12.** Parte posterior de la placa de control del sistema de control del arreglo de sensores [Autor].

Las medidas de la placa PCB se establecieron con el objetivo de integrar todo en la carcasa del sensor de temperatura, la cual, se puede observar en la figura 2.13.


**Figura 2.13.** Carcasa del sensor de temperatura [Autor].

## 2.3 Diseño y construcción de la interfaz Arduino – Arreglo de sensores

La interfaz de comunicación entre el arreglo de sensores y la plataforma Arduino está conformada por el multiplexor TCA9548a y todas las conexiones necesarias para comunicarse tanto con la plataforma Arduino como con los arreglos de sensores, tanta comunicación I2C como alimentación. Finalmente, se incorpora el circuito de selección del tipo de luz y el disparador de la interrupción.

Así como en el punto anterior se hizo uso del software Proteus para el diseño de esta interfaz. Otro punto en común es que la mayoría de los elementos no se integran directamente, sino que se conectan a través de borneras.

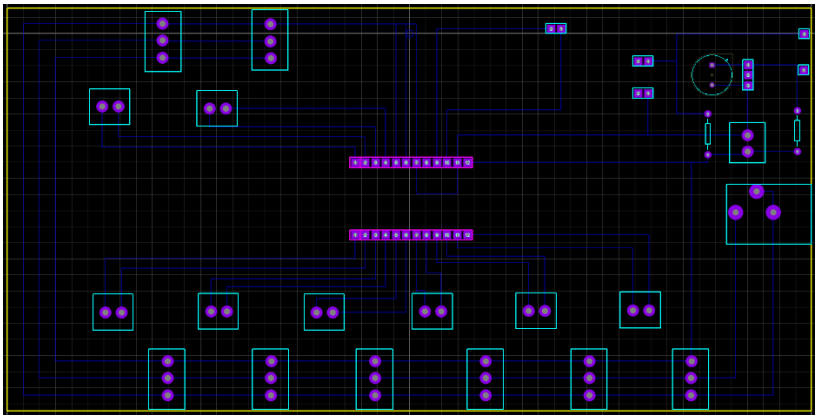


**Figura 2.14.** Interfaz de comunicación Arduino – Arreglo de sensores [Autor].

En la figura 2.14 se puede observar el diseño de la interfaz de comunicación con todos los elementos que la integran y las conexiones entre los mismos. Como en el caso del arreglo de sensores se modelo las borneras de forma manual con las medidas comerciales de esta al no existir un paquete que contenga este elemento.

En las figuras 2.15 y 2.16 se puede observar el diseño en 3D.

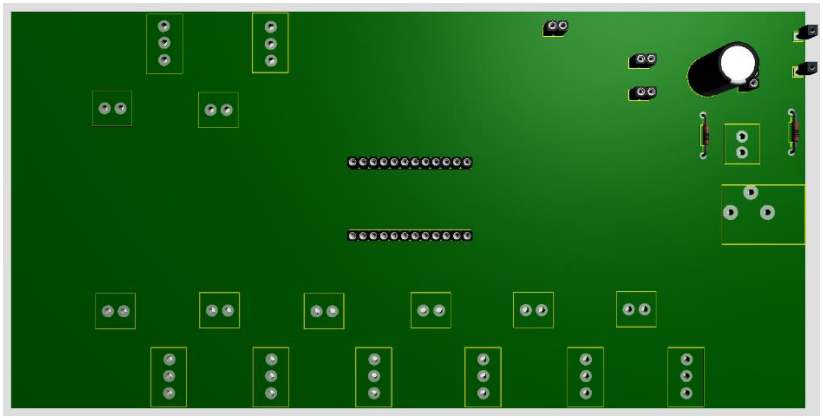


**Figura 2.15.** Modelo 3D de la parte frontal de la interfaz.

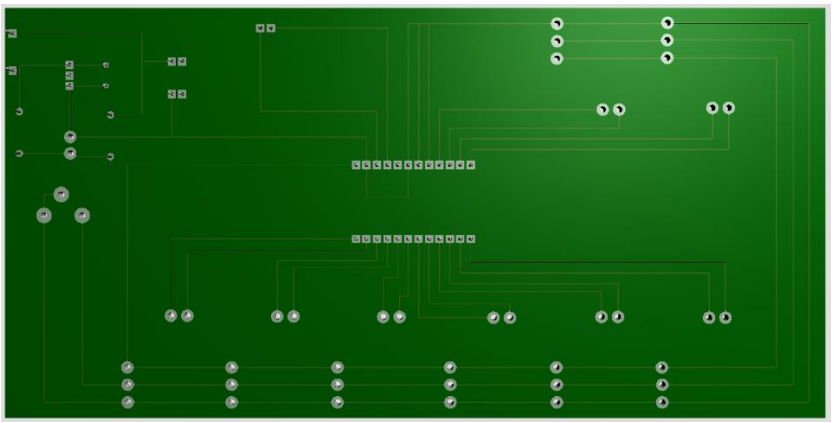


**Figura 2.16.** Modelo 3D de la parte posterior de la interfaz.

## 2.4 Desarrollo del algoritmo de adquisición de datos

Para esta etapa del proyecto se hizo uso el IDE de Arduino para desarrollar el algoritmo de control del arreglo de sensores y la interfaz de comunicación. Dentro de esta sección se explica que algunas acciones se ejecutan tomando en cuenta el número de módulos conectados y el orden en el cual se conectaron. El algoritmo organiza los sensores tomando en cuenta el canal del multiplexor al cual están conectados, es decir, los módulos conectados en el canal 0 se definen antes de los que estén en otros canales; una vez establecido el canal se da prioridad al módulo que tenga la dirección menor; los módulos pueden tener 4 direcciones que van de la 64 a la 67. En resumen, el orden de los módulos se establece tomando en cuenta el canal del multiplexor al cual están conectados, así como la dirección del módulo.

A continuación, se detallará cada una de las secciones del algoritmo desarrollado explicando su funcionamiento.

### 2.4.1 Inclusión de librerías

```
#include <Wire.h>
#include "SDL_Arduino_INA3221.h"
#include <LiquidCrystal_I2C.h>
```

**Figura 2.17.** Inclusión de librerías [Autor].

En la figura 2.17 se puede observar la declaración de 3 librerías que fueron las utilizadas para el desarrollo de este algoritmo. La librería *Wire.h* es la que permite utilizar la comunicación I2C de Arduino. Las siguientes dos librerías están desarrolladas bajo la librería *Wire.h*, por este motivo es indispensable que se la incluya. La librería *SDL_Arduino_INA3221.h* permite manejar los módulos INA3221 y obtener los datos de cada uno de los canales de censado que posee. Finalmente, la librería *LiquidCrystal_I2C.h* permite manejar la interfaz de comunicación I2C entre Arduino y una LCD 16x2.

### 2.4.2 Declaración de variables globales y objetos.

En esta sección se detallan las variables globales y objetos definidos dentro del algoritmo, además se explica el propósito que tiene cada una de estas. Dentro de esta sección se denomina módulo a cada INA3221 que forma parte del arreglo de sensores antes mencionado, esto con el fin de no causar confusión con la estructura del texto.

```
#define C_Luminosidad 1
#define C_Temperatura 2
#define C_Humedad 3
#define S_Lum 7

SDL_Arduino_INA3221 modulo_1 (64);
SDL_Arduino_INA3221 modulo_2 (65);
SDL_Arduino_INA3221 modulo_3 (66);
SDL_Arduino_INA3221 modulo_4 (67);


LiquidCrystal_I2C lcd(0x27, 16, 2);
LiquidCrystal_I2C lcd_1(0x21, 16, 2);

int Direcciones_Modulos[] = {64, 65, 66, 67};
int N_Modulos_Ch [] = {0, 0, 0, 0, 0, 0, 0, 0};
int NT_M = 0;
int Dir_Mod_Av_Ch[8][4];
boolean T_Luz = false;
int intPin = 2;
int select_LCD = 0;
String Tipo_Luz = " ";
int timeThreshold = 150;
long startTime = 0;
float Medidas_Final [3];
```

**Figura 2.18.** Declaración de variables globales y objetos [Autor].

En la figura 2.18 se puede observar la declaración de variables globales que se encuentra dividida en 4 secciones. Primero se tiene 3 variables que definen el canal del módulo INA3221 en donde se encuentra conectado el sensor que mide dicho parámetro, es decir, en el canal 1 se tiene el sensor de Luminosidad, en el canal 2 el sensor de Temperatura y en el canal 3 el sensor de Humedad; además se define una cuarta variable que representa el puerto de la placa Arduino en donde se conectara el circuito de selección del tipo de luz que se esté censando, ya sea natural o artificial en este caso se escogió el puerto 7.

En la segunda sección se definen cuatro objetos del tipo INA3221, estos permiten utilizar las funciones de adquisición de datos de la librería del módulo INA3221. Se diferencian entre sí por la dirección que se utilice; más adelante en la sección principal del algoritmo se detalla el uso que se da a cada uno de estos 4 objetos.

En la tercera sección se definen dos objetos del tipo LiquidCrystal_I2C, los cuales, permiten utilizar las funciones que vienen incluidas en la librería LiquidCrystal_I2C.h para el manejo de las LCD a través el protocolo I2C. Para este proyecto se hizo uso de dos LCD de 16x2, por lo tanto, se declararon dos objetos de este tipo con diferentes direcciones cada uno.

En la última sección de este código se definen las variables globales que contienen información necesaria para el desarrollo de todo el algoritmo. La primera variable, *Direcciones_Modulos*, es de tipo arreglo de enteros y contiene las direcciones posibles del módulo INA3221. La segunda variable, *N_Modulos_Ch*, es de tipo arreglo de enteros y se

inicializa como un arreglo de 1x8 con valor de 0 en todas sus posiciones; en esta variable se almacenará el número de módulos conectados por cada canal del multiplexor TCA9548a. La tercera variable, *NT_M*, es de tipo entero y se inicializa con un valor de 0; en esta variable se almacenará el número total de módulos conectados al multiplexor TCA9548a, teniendo en cuenta todos sus canales. La cuarta variable, *Dir_Mod_Av_Ch*, es de tipo arreglo de enteros y se inicializa como un arreglo de 8x4; en esta variable se almacenará la dirección o direcciones del o los módulos que estén conectados en cada uno de los canales del multiplexor TCA9548a; el tamaño de esta variable se define debido a que se tienen 8 canales en el multiplexor TCA9548a y por cada uno de estos se puede tener hasta 4 módulos INA3221, cada uno con una de las 4 direcciones posibles. La quinta variable se definió como *T_Luz* de tipo booleana, esta variable almacena un estado que cambia de acuerdo con el circuito seleccionador del tipo de luz que se esté censando, esto permitirá utilizar una de las dos expresiones obtenidas en la sección de calibración de sensores. La sexta variable, *intPin,* es de tipo entero y se inicializo con un valor de 2; esta variable hace referencia al pin de la plataforma Arduino que se utilizara para el manejo de interrupciones. En la plataforma Arduino UNO se puede manejar interrupciones a través de los pines 2 y 3. La séptima variable, *select_LCD*, es de tipo entero y se inicializo con un valor de cero; esta variable cambiará conforme se produzca una interrupción y de acuerdo con esto se modificará una determinada acción que se detallará más adelante. La octava variable, *Tipo_Luz,* es de tipo String y se utiliza para almacenar el tipo de luz que se está censando, siendo posible luz natural o luz artificial, este mensaje se muestra a través de una de las dos LCDs. La novena variable, *timeThreshold,* es de tipo entero y junto a la décima variable, *startTime*, se utilizan para evitar rebotes ocasionados por un pulsador, el cual, fue utilizado para manejar el disparo de la interrupción. La decimo primera y última variable, *Medidas_Final,* es de tipo arreglo de flotantes; en esta variable se almacenarán los valores promedio de todos los datos obtenidos por todos los arreglos de sensores que estén conectados al multiplexor.

### 2.4.3 Función Setup

```
void setup() {

  pinMode(intPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(intPin), blink, LOW);

  pinMode(S_Lum, INPUT);

  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Tesis Final");
  lcd.setCursor(0,1);
  lcd.print("Francisco Valdez");
  delay(2000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("T:");
  lcd.setCursor(9,0);
  lcd.print("H:");
  lcd.setCursor(0,1);
  lcd.print("L:");
  lcd.setCursor(7,0);
  lcd.print("C");
  lcd.setCursor(15,0);
  lcd.print("%");
```

**Figura 2.19.** Función setup parte 1 [Autor].

```
lcd_1.init();
lcd_1.backlight();
lcd_1.clear();
lcd_1.setCursor(0,0);
lcd_1.print("Sistema de");
lcd_1.setCursor(0,1);
lcd_1.print("censado de datos");
delay(2000);

Wire.begin();

Detectar_Modulos();

delay(1000);
```

**Figura 2.20.** Función setup parte 2 [Autor].

La función setup es la primera que se ejecuta dentro del programa; en esta se definen los criterios y funciones que se requiere se ejecuten una sola vez previas a la ejecución del cuerpo principal del programa. Para su explicación, esta sección de código está dividida en dos partes.

En la primera parte se define el modo en el que el pin destinado al manejo de interrupciones, en este caso el pin 2, trabajará; este se establece como entrada con la activación de la resistencia de pull-up que incorpora. Seguido se definen los parámetros de la interrupción y se establece que función debe ejecutarse cuando salte la interrupción,

también se define el modo en el cual trabajará el pin en donde estará conectado el circuito de selección del tipo de luz, en este caso se lo define como entrada. Además, se inicializa y configura la primera LCD, a continuación, se escribe un mensaje de bienvenida seguido de las iniciales de los parámetros ambientales que se medirán y las unidades de estos.

En la segunda parte del setup se inicializa y configura la segunda LCD, también se muestra un mensaje con el tema general de este proyecto. Para poder visualizar este mensaje se establece un tiempo de espera de 2 segundos. A continuación, se inicializa la comunicación I2C a través del comando *Wire.begin(),* seguido se invoca a la función *Detectar_Modulos,* la cual, permite conocer el número de módulos conectados a cada uno de los canales, así como el número total de módulos en general. Mas adelante se detallará el principio de funcionamiento de esta función y las variables involucradas. Finalmente se establece un tiempo de espera de un segundo antes de comenzar con el programa principal.

### 2.4.4 Funciones complementarias

Antes de explicar el programa principal se procederá a detallar las funciones complementarias a este. Estas funciones realizan acciones puntuales que sirven como apoyo para la estructura del programa principal.

#### 2.4.4.1 Función detectar módulos

```
void Detectar_Modulos (){

  int cont = 0;
  int var = 0;
  int aux = 0;

  for(int j = 0; j<8; j = j + 1){
     TCA9548A(j, 112);
     for(int k = 0; k<4; k = k + 1){
       Wire.beginTransmission(Direcciones_Modulos[k]);
       byte resultado = Wire.endTransmission();
       if(resultado == 0){
        Dir_Mod_Av_Ch[j][aux] = Direcciones_Modulos[k];
        cont = cont + 1;
        aux = aux + 1;
        delay(500);
       }
     }
     NT_M = NT_M + cont;
     //Serial.println(NT_M);
     N_Modulos_Ch[j] = cont;
     cont = 0;
     aux = 0;
   }
}
```

**Figura 2.21.** Función Detectar_Modulos [Autor].

En la figura 2.21 se puede observar la estructura de la función Detectar_Modulos. Esta función hace uso de la librería *Wire.h* para detectar los módulos conectados en cada uno

de los canales del multiplexor TCA9548a. Para esta función se hace uso de las direcciones de los módulos, establecidas en una variable global, con el propósito de iniciar la comunicación con el módulo al que pertenezca cada una de estas direcciones; si el resultado de esta comunicación es 0 significa que fue exitosa, por lo tanto, existe un módulo con esa dirección conectado a un canal. En esta función se modifican las variables que almacenan las direcciones de los módulos hábiles, la variable que contiene el número total de módulos y la que contiene el número de módulos por canal.

### 2.4.4.2 Función TCA9548A

```
void TCA9548A(uint8_t bus, int address){
  Wire.beginTransmission(address);  // TCA9548A address
  Wire.write(1 << bus);             // send byte to select bus
  Wire.endTransmission();
  //Serial.println(bus);
}
```

**Figura 2.22.** Función TCA9548A [Autor].

En la figura 2.22 se puede observar la estructura de la función *TCA9548A*. Esta función permite escoger el canal del multiplexor TCA9548A con el cual se trabajará. Para esto se tiene como argumento de entrada una variable tipo uint8_t que representa el número de canal siendo los valores posibles de 1 a 8 y la dirección del TCA9548A, en este proyecto se hace uso de un único multiplexor, pero la función esta desarrollada con el fin de ser genérica y que se pueda utilizar para el número de multiplexores necesario. Esta función inicia la comunicación I2C con el multiplexor que corresponda con la dirección enviada, posteriormente se envía un byte de datos; este se compone por 8 bits que estarán recorridos a la izquierda el número de posiciones representado por el canal al que se quiere acceder, así el multiplexor conoce con que canal establecer la comunicación.

### 2.4.4.3 Función Conv_Lum

```
float Conv_Lum(float x, boolean val){
  float Lum;

  if (val){
    Lum = 0.3185*exp(4.9069*x);
    if (x<2){
    Lum = Lum + (0.05*Lum);
    }else{
    Lum = Lum + (0.3*Lum);
    }
  }else{
    Lum = 0.0545*exp(5.3572*x);
    if (x<1){
      Lum = Lum+4;
    }else if (x>1 && x<1.3){
      Lum = Lum - Lum*0.2;
    }else if (x>1.3 && x<1.5){
      Lum = Lum - Lum*0.28;
    }else if (x>1.5 && x<1.8){
      Lum = Lum - Lum*0.30;
    }else if (x>1.8 && x<2){
      Lum = Lum - Lum*0.15;
    }else if (x>2 && x<2.2){
      Lum = Lum + Lum*0.25;
    }else if (x>2.2 && x<2.5){
      Lum = Lum + Lum*0.1;
```

**Figura 2.23.** Función Conv_Lum (1/2) [Autor].

```
    }else{
      Lum = Lum + Lum*0.03;
    }
  }
 return Lum;
}
```

**Figura 2.24**. Función Conv_Lum (2/2) [Autor].

En la figura 2.23 y 2.24 se puede observar la estructura de la función *Conv_Lum*. Esta función permite transformar el voltaje obtenido del arreglo de sensores, exactamente del canal 1 que es donde está conectado el sensor de Luminosidad, al nivel de Luminosidad correspondiente; este se realiza con las expresiones obtenidas en la etapa de calibración de sensores.

Debido a que se tienen dos expresiones que representan el comportamiento del sensor cuando está expuesto a luz natural o luz artificial, se debe controlar que expresión utilizar; con este propósito se pide como argumentos de entrada el valor de voltaje censado y una variable de tipo booleana que servirá para escoger el tipo de expresión a utilizar. Dentro de esta función también se corrige el error que existe con estas expresiones, debido al error explicado en la etapa de calibración; esto se corrige sumando o restando una cantidad de acuerdo con el comportamiento del sensor en los diferentes rangos de voltaje. Esta

corrección se realiza para ambas expresiones, siendo más complejo cuando se trabaja con luz natural debido a que los valores de luminosidad tienen una mayor variabilidad y generan un margen de error mayor al momento de la calibración.

### 2.4.4.4 Función Conv_Temp

```
float Conv_Temp(float x){
  float Temp;
  Temp = 0.0024*pow(x, 6) - 0.0929*pow(x, 5) + 1.4718*pow(x, 4) - 11.889*pow(x, 3) + 50.853*pow(x, 2) - 99.354*x + 61.938;
  return Temp;
}
```

**Figura 2.25.** Función Conv_Temp [Autor].

En la figura 2.25 se tiene la estructura de la función *Conv_Temp*. Esta función permite transformar el nivel de voltaje obtenido por el arreglo de sensores, exactamente del canal 2, en donde está conectado el sensor de temperatura. Con este propósito se utiliza la expresión encontrada en la etapa de calibración; la línea de tendencia de este sensor presenta un error mínimo del 2% razón por la cual no es necesario realizar una corrección. Debido a esto el único argumento de entrada que la función solicita es el nivel de voltaje censado.

### 2.4.4.5 Función Conv_Hum

```
float Conv_Hum(float x){
  float Hum;
  Hum = 10*x;
  return Hum;
}
```

**Figura 2.26.** Función Conv_Hum.

En la figura 2.26 se puede observar la estructura de la función *Conv_Hum*. Esta función permite transformar el nivel de voltaje censado por el arreglo de sensores, exactamente por el canal 3 que es donde se encuentra conectado el sensor de humedad. Debido a que este sensor es bastante preciso y su nivel de voltaje de proporcional al nivel de Humedad relativa no fue necesaria una calibración, y para transformar el voltaje producido por el sensor al nivel de humedad que corresponda se tiene que multiplicar el valor por 10, ya que, el nivel de voltaje que emite el sensor va de 0 a 10V que representa un nivel de humedad de 0 al 100%.

### 2.4.4.6 Función MostrarDatos

```
void MostrarDatos(float Lum, float Temp, float Hum){

  lcd.setCursor(2,0);
  lcd.print(Temp);

  lcd.setCursor(11,0);
  lcd.print(Hum);
  lcd.setCursor(15,0);
  lcd.print("%");

  lcd.setCursor(2,1);
  lcd.print(Lum);

}
```

**Figura 2.27.** Función MostrarDatos [Autor].

En la figura 2.27 se puede observar la estructura de la función MostrarDatos. Esta función permite configurar la LCD para que muestre los valores de los parámetros ambientales censados y procesados previamente.

### 2.4.4.7 Función Calcular_Prom

```
float Calcular_Prom(float Medidas[], int Long){

  float Promedio = 0;

  for(int i = 0 ; i<Long ; i = i + 1){
    Promedio = Promedio + Medidas[i];
  }

  Promedio = Promedio / Long;
  return Promedio;
}
```

**Figura 2.28.** Función Calcular_Prom [Autor].

En la figura 2.28 se puede observar la estructura de la función Calcular_Prom. Esta función permite calcular el promedio de las medidas obtenidas por todos los arreglos de sensores conectados al multiplexor TCA9548a. El principio de funcionamiento es el cálculo promedio tradicional que consiste en sumar todos los valores y dividirlos para el número total de valores. Como argumento de entrada se pide un arreglo de los valores a calcular el promedio y también la cantidad de valores dentro de este arreglo.

### 2.4.4.8 Función blink

```
void blink() {
  if(millis() - startTime > timeThreshold){
    if(select_LCD >= NT_M){
    select_LCD = 0;
    startTime = millis();
  }else{
    select_LCD = select_LCD + 1;
    startTime = millis();
  }
  }

}
```

**Figura 2.29.** Función blink [Autor].

En la figura 2.29 se puede observar la estructura de la función blink. Esta es la función que se ejecutara cuando se produzca la interrupción definida en el algoritmo. Esta función detecta cuantas veces se ha entrado a la función de atención de la interrupción y de acuerdo con esto modifica la variable *select_LCD*. Esta variable se utilizará en la función *Mostrar_LCD_2* que se define a continuación. Dentro de esta función se utiliza una estructura tal que el efecto causado los rebotes al utilizar pulsadores no afecte; los rebotes son causados por el principio mecánico de los pulsadores, estos causan que la interrupción se active varias veces con solo una pulsación, situación que no es conveniente. Para manejar esto se implementa un condicional que evalúa el tiempo que dura la pulsación; para esto se utiliza la función *millis* que nos brinda la información del tiempo que ha pasado desde que se entró a la interrupción y se resta el tiempo de inicio y el resultado de esta operación debe ser mayor al tiempo mínimo que debe durar un pulso, este tiempo se definió en la sección de definición de variables globales y objetos. Si el tiempo es mayor se ejecuta la acción dentro de la interrupción y sino no realiza ninguna acción, se debe modificar el valor del tiempo de inicio con el valor generado por la función *millis*.

### 2.4.4.9 Función Mostrar_LCD_2

```
void Mostrar_LCD_2 (float Temperatura [], float Luminosidad [], float Humedad []){
  if(select_LCD == 0){
    lcd_1.clear();
    lcd_1.setCursor(0, 0);
    lcd_1.print("Num Tot Mod: ");
    lcd_1.setCursor(14, 0);
    lcd_1.print(NT_M);
    lcd_1.setCursor(0, 1);
    lcd_1.print(Tipo_Luz);

  }else{
    lcd_1.clear();
    lcd_1.setCursor(0,0);
    lcd_1.print("T");
    lcd_1.setCursor(1,0);
    lcd_1.print(select_LCD);
    lcd_1.setCursor(2,0);
    lcd_1.print(":");
    lcd_1.setCursor(9,0);
    lcd_1.print("H");
    lcd_1.setCursor(10,0);
    lcd_1.print(select_LCD);
    lcd_1.setCursor(11,0);
    lcd_1.print(":");
    lcd_1.setCursor(0,1);
```

**Figura 2.30.** Función Mostrar_LCD_2 (1/2) [Autor].

```
    lcd_1.print("L");
    lcd_1.setCursor(1,1);
    lcd_1.print(select_LCD);
    lcd_1.setCursor(2,1);
    lcd_1.print(":");
    lcd_1.setCursor(3,0);
    lcd_1.print(Temperatura[select_LCD - 1]);
    lcd_1.setCursor(12,0);
    lcd_1.print(Humedad[select_LCD - 1]);
    lcd_1.setCursor(3,1);
    lcd_1.print(Luminosidad[select_LCD - 1]);
  }

}
```

**Figura2.31.** Función Mostrar_LCD_2 (2/2) [Autor].

En las figuras 2.30 y 2.31 se puede observar la estructura de la función *Mostrar_LCD_2*. En esta función se configura la segunda LCD para que muestre diferentes mensajes de acuerdo con el valor de la variable *select_LCD*. Si el valor es menor al número total de módulos conectados al multiplexor se muestran los valores de cada uno de los módulos, es decir, al disparar la interrupción por primera vez se muestran los valores de las variables ambientales captados por el módulo uno; al presionarlo por segunda vez se muestran los valores del segundo modulo y esto se repite con el número de sensores conectados. Cuando el valor de la variable *select_LCD* supere el número total de sensores se muestra un mensaje mostrando el número total de sensores conectados y el tipo de luz que se está censando, este último mensaje además es el que se muestra por defecto al iniciar el programa.

### 2.4.5 Función loop

Dentro del entorno de Arduino esta función es la que se repetirá constantemente dentro de todo el algoritmo, en otras palabras, está en la función principal de todo el programa. Para facilitar la explicación del código se lo dividirá en varias partes. Cabe mencionar que se hace referencia a funciones y variables antes expuestas.

#### 2.4.5.1 Declaración de variables locales

```
void loop() {

  float Mediciones[NT_M*3];
  float Luminosidad [NT_M];
  float Temperatura [NT_M];
  float Humedad [NT_M];

  float shuntvoltage = 0;
  float busvoltage = 0;

  float Lum_Prom = 0;
  float Temp_Prom = 0;
  float Hum_Prom = 0;

  int aux = 0;
```

**Figura 2.32.** Declaración de variables locales en la función loop [Autor].

En la figura 2.32 se puede observar la declaración de variables locales que se utilizan dentro de la función loop. La variable *Mediciones* es de tipo arreglo de flotantes y es en esta en donde se almacenarán todas las medidas tomadas por cada uno de los módulos conectados al multiplexor TCA9548a; estas están ordenadas de la siguiente manera: primero las medidas de Luminosidad, seguido de las de temperatura y tercero las de humedad, además dentro de cada grupo de medidas se prioriza al número de sensor al cual pertenece esa medida, es decir, las medidas del sensor 1 serán las primeras en cada grupo de medidas y así sucesivamente. El método para determinar el número de sensor se explicó al inicio de la sección de metodología. Las variables *Luminosidad, Temperatura* y *Humedad* son del tipo arreglo de flotantes y almacenan los valores de las medidas del parámetro que su nombre indica, esto de cada uno de los módulos conectados al multiplexor TCA9548a, sigue la misma lógica de organización que la variable *Mediciones*. Las variables *busvoltage* y *shuntvoltage* son de tipo flotante; en estas se almacenan los datos obtenidos del módulo INA3221, *busvoltage* almacena el voltaje en el punto donde se conectan los puertos de cada canal y *shuntvoltage* almacena el dato del voltaje que cae sobre la resistencia shunt que está conectada en paralelo con cada uno de los tres canales del módulo, a partir de estas dos mediciones se puede obtener el voltaje que cae sobre la

carga en este caso la resistencia de 10K utilizada para censar la variación de voltaje que producen los sensores. Las variables *Lum_Prom, Temp_Prom* y *Hum_Prom* son de tipo flotante y están inicializadas en 0; en estas variables se almacenan los valores promedio de todas las medidas tomadas por todos los módulos conectados al multiplexor TCA9548a correspondiente a cada uno de los parámetros ambientales de interés que son Luminosidad, Temperatura y Humedad respectivamente. Finalmente, la variable *aux* es de tipo entero y esta inicializada en cero; en esta la variable se setea la posición en la cual se almacena un dato censado, esta variable cobra sentido más adelante en el código.

### 2.4.5.2 Algoritmo de control, adquisición y organización de datos

Este es el algoritmo principal del programa debido a que aquí se propone un modelo que permite conectar cualquier número de sensores de entre 1 hasta 32 sin tener que modificar nada de código. Además, dentro de este se adquieren los datos de los arreglos de sensores y se los almacena en las variables correspondientes.

```
for(int i = 0; i<8; i = i + 1){
  for(int j = 0; j<N_Modulos_Ch[i]; j = j + 1){

    TCA9548A(i, 112);
```

**Figura 2.33.** Algoritmo de control, adquisición y organización de datos (1/4) [Autor].

En la figura 2.33 se observa la primera parte de este algoritmo; esta sección de código define dos lazos anidados dentro de los cuales se encuentra el resto del código. El primero lazo nos permite tener una referencia del canal del multiplexor TCA9548a con el cual se está trabajando, tanto para la adquisición de datos, como para su almacenamiento. El segundo lazo nos permite tener la referencia del número de módulos que están conectados por cada uno de los canales del multiplexor TCA9548a. Finalmente, se invoca a la función TCA9548A, la cual, nos pide como parámetros de entrada el canal con el que se desea comunicar y la dirección del multiplexor TCA9548a; la referencia del canal se tiene con el primer lazo representado por la letra *i* y la dirección enviada es la 112, este valor es fijo al utilizar únicamente un multiplexor.

```
switch(Dir_Mod_Av_Ch[i][j]){

case 64:
```

**Figura 2.34.** Algoritmo de control, adquisición y organización de datos (2/4) [Autor].

En la figura 2.34 se puede observar la estructura que se encuentra dentro de los lazos anidados antes detallados; como se puede apreciar, se basa en la función *switch*, se

escogió esta solución debido a que se analiza las direcciones posibles de los módulos INA3221, siendo estas utilizadas para definir los 4 casos dentro de la función *switch*. Esta función admite un parámetro de entrada el cual selecciona el caso que corresponda; este parámetro se lo obtiene de la variable *Dir_Mod_Av_Ch* la cual es un arreglo de enteros en donde se almacenaron las direcciones de los módulos conectados por cada canal del multiplexor, así podemos utilizar la referencia *i* para acceder a la información del número de canal y la referencia *j* para acceder a la dirección de cada uno de los módulos disponibles por canal, además se tiene control sobre cuantas veces se ejecutara esta sentencia de código sin que se hagan ejecuciones innecesarias.

```
case 64:

  //Medicion Voltaje canal Luminosidad
  busvoltage = modulo_1.getBusVoltage_V(C_Luminosidad);
  shuntvoltage = modulo_1.getShuntVoltage_mV(C_Luminosidad);

  // Transformacion a Luxes
  if(digitalRead(S_Lum) == HIGH){
    T_Luz = false;
    Tipo_Luz = "Luz Natural";
  }else{
    T_Luz= true;
    Tipo_Luz = "Luz Artificial";
  }
  Luminosidad [aux] = Conv_Lum((busvoltage + (shuntvoltage / 1000)), T_Luz);

  Mediciones [aux] = Luminosidad [aux];


  //Medicion Voltaje canal Temperatura
  busvoltage = modulo_1.getBusVoltage_V(C_Temperatura);
  shuntvoltage = modulo_1.getShuntVoltage_mV(C_Temperatura);

  // Transformacion a grados Celsius
  Temperatura [aux] = Conv_Temp(busvoltage + (shuntvoltage / 1000));
```

**Figura 2.35.** Algoritmo de control, adquisición y organización de datos (3/4) [Autor].

```
    Mediciones [aux+N_Modulos_Ch[i]] = Temperatura [aux];


    //Medicion Voltaje canal Humedad
    busvoltage = modulo_1.getBusVoltage_V(C_Humedad);
    shuntvoltage = modulo_1.getShuntVoltage_mV(C_Humedad);

    //Transformacion a % de Humedad
    Humedad [aux] = Conv_Hum(busvoltage + (shuntvoltage / 1000));

    Mediciones [j+2*N_Modulos_Ch[i]] = Humedad [aux];

    aux = aux + 1;

    break;

  case 65:
```

**Figura 2.36.** Algoritmo de control, adquisición y organización de datos (4/4) [Autor].

En las figuras 2.35 y 2.36 se puede observar el segmento de código que se encuentra en cada uno de los casos de la función switch. Este código se divide en tres partes correspondientes al manejo de los datos de cada parámetro ambiental que se están censando. En la figura 2.35 podemos observar que primero se hace la adquisición de los datos a través de las funciones *modulo.getBusVoltage* y *modulo.getShuntVoltage*, estas funciones son parte de la librería que maneja el módulo INA3221 y requieren como argumento de entrada el canal que se desea censar; este argumento de entrada fue definido al inicio del código. Una vez obtenido el voltaje emitido por el sensor se procede a transformar ese nivel de voltaje en el nivel de luminosidad que corresponda, pero antes se valida el valor de la variable *S_Lum* que nos indica el tipo de luz para así seleccionar la expresión que se utilizara para la transformación; además, se setea el valor de la variable *Tipo_Luz* que es utilizada para mostrar que tipo de luz se está censando en la segunda LCD. Para la transformación del voltaje en nivel de luminosidad se utiliza la función *Conv_Lum;* a esta se le envía como parámetros de entrada el voltaje del sensor, definido como la suma del voltaje de la resistencia shunt y el voltaje del bus y, además se envía la variable *S_Lum* que determina el tipo de luz. Una vez transformado el voltaje en nivel de luminosidad se guarda el dato en la variable *Luminosidad* y en la variable *Mediciones;* para este propósito se utiliza la variable *aux* que modificara su valor cada vez que se guarde un grupo de datos, correspondiente a las tres variables ambientales, gracias a esto se tiene control sobre los datos obtenidos y se los puede organizar de acuerdo con lo mencionado en un inicio. Para el caso de la temperatura y la humedad se realiza el mismo proceso que

para la Luminosidad con la diferencia que para estos dos casos no se debe validar ninguna variable, ya que, no se tienen diferentes ambientes.

Para el resto de los casos al algoritmo es el mismo, lo único que cambia es el objeto que se utiliza para adquirir los datos con las funciones *modulo.getBusVoltage* y *modulo.getShuntVoltage;* como se puede observar en la figura 2.35, el objeto utilizado se denomina modulo1 ya que este fue definido en un inicio con la dirección correspondiente al caso 64 de la función *switch.* Como antes se mencionada los casos se definen de acuerdo con la dirección del módulo y de acuerdo con esto también se define el objeto a utilizar. Estos objetos se describen en la sección de definición de variables globales y objetos.

### 2.4.5.3 Cálculo del promedio, visualización y almacenamiento de datos

```
int L_L = sizeof(Luminosidad)/sizeof(Luminosidad[0]);
int L_T = sizeof(Temperatura)/sizeof(Temperatura[0]);
int L_H = sizeof(Humedad)/sizeof(Humedad[0]);

Lum_Prom = Calcular_Prom(Luminosidad, L_L);
Temp_Prom = Calcular_Prom(Temperatura, L_T);
Hum_Prom = Calcular_Prom(Humedad, L_H);

Medidas_Final[0] = Lum_Prom;
Medidas_Final[1] = Temp_Prom;
Medidas_Final[2] = Hum_Prom;

MostrarDatos(Lum_Prom, Temp_Prom, Hum_Prom);
Mostrar_LCD_2(Temperatura, Luminosidad, Humedad);

delay(5000);
```

**Figura 2.37.** Cálculo del promedio, visualización y almacenamiento de datos [Autor].

Esta es la sección final de la función loop. En este segmento de código se definen tres variables locales que almacenan el tamaño de las variables Luminosidad, Temperatura y Humedad, estas variables se utilizan como argumento de entrada para el cálculo del promedio de los valores obtenidos. A continuación, se llama a la función *Calcular_Prom* que permite calcular el promedio; se envía como parámetros de entrada el arreglo que contiene los datos de cada parámetro ambiental, en este caso *Luminosidad, Temperatura* y *Humedad*, y la longitud de este. El proceso se realiza por cada parámetro ambiental y el resultado de cada uno se almacena en las variables *Lum_Prom, Temp_Prom* y *Hum_Prom* respectivamente. Una vez obtenido el promedio se procede a almacenarlo en un solo arreglo, *Medidas_Final,* organizado como se explicó al inicio de la sección de metodología. Finalmente, se procede a mostrar los valores promedio de cada parámetro ambiental en la primera LCD con la función *MostrarDatos* y se llama a la función *Mostrar_LCD_2* a la cual

se envía como parámetros de entrada los arreglos que contienen los datos de todos los módulos dividido por parámetro ambiental. Cabe mencionar que esta última función modifica su comportamiento de acuerdo con la interrupción programada.

## 2.5 Circuito de selección del tipo de luz y disparo de interrupción

Para seleccionar el tipo de luz que se está censando se hizo uso un pin digital de la plataforma Arduino, a este se conecta un circuito con una configuración pull up tal y como se muestra en la figura 2.35. Este se compone de un switch que va conectado a tierra por un extremo y a una resistencia de pull up que va a VCC en el otro. Entre la resistencia y el switch se tiene la salida que va hacia el pin digital del Arduino. Para leer este pin no se hizo uso de interrupciones al ser más efectivo una lectura constante dentro de la función loop.

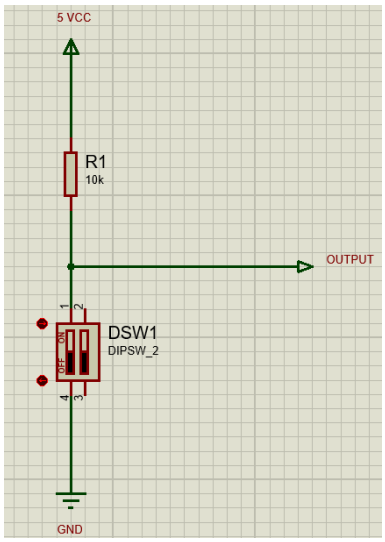



**Figura 2.38.** Circuito en configuración pull-up para selección de tipo de luz [Autor].

En la figura 2.38 se puede observar la estructura del circuito de selección del tipo de luz. Este circuito envía un estado lógico a un pin digital del Arduino; si el switch se encuentra apagado se envía un nivel lógico HIGH mientras que al activar el switch se envía un estado lógico LOW. Estos dos estados se manejan dentro del código de la siguiente manera: si el valor leído por el pin es HIGH la luz censada es natural mientras que si el valor leído por el pin es LOW la luz censada es artificial, de acuerdo con esto se manejan las expresiones de transformación y el mensaje mostrado en la LCD.

En el caso de la interrupción también se optó por un circuito en configuración pull up, pero a diferencia del circuito de selección del tipo de luz este se compone de una parte definida por hardware y una por software. Dentro de la función *blink* se detalla el segmento de código utilizado para este propósito y en este apartado se detalla la solución por hardware; esto se implementa para mitigar el efecto producido por los rebotes que aparecen por el

uso de pulsadores. Para este circuito si es necesario el uso de un pulsador por sobre un switch debido a que este será el que se encargue de disparar la interrupción.

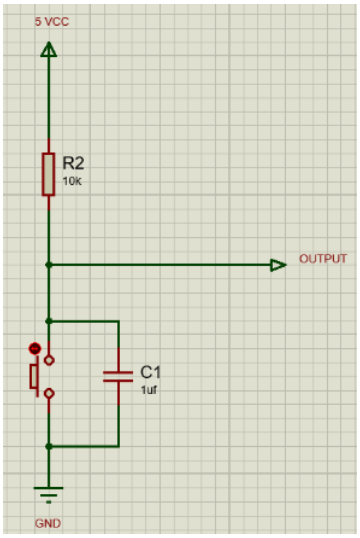



**Figura 2.39.** Circuito para el manejo de interrupciones [Autor].

En la figura 2.39 se puede observar la estructura del circuito para el manejo de la interrupción definida en el algoritmo. Consiste en un circuito con configuración pull up, al cual, se agregó un condensador de 1 uF que ayuda a mitigar el efecto de los rebotes. Al presionar el pulsador se cambiará el estado de la salida del circuito, pero al dejar de presionarlo, el condensador en conjunto con la resistencia de pull up forman un circuito de retardo anti-rebote por el principio de tiempo de carga y descarga del condensador.

# 3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

## 3.1 Resultados

La presentación de resultados se dividió en cuatro partes. En la primera se muestra la implementación del dispositivo censado que unifica los sensores y el módulo INA3221, cuyo diseño se tiene en las figuras 2.11 y 2.12. En la segunda parte se muestra los resultados de la implementación de la interfaz de comunicación Arduino – Arreglo de sensores cuyo diseño se puede observar en las figuras 2.15 y 2.16. A continuación, se muestran los resultados del funcionamiento de todo el proyecto en conjunto, utilizando un arreglo de sensores solamente para medir las variables ambientales Luminosidad, Temperatura y Humedad y compararlas con las medidas obtenidas por los equipos utilizados en la calibración de los sensores que miden los dos primeros parámetros mencionados y en el caso de la Humedad se muestra el parámetro medido. Finalmente, se hace uso de un módulo INA3221 extra con un arreglo de resistencias que simulen otro arreglo de sensores para probar la posibilidad de utilizar más de un arreglo de sensores.

### 3.1.1 Implementación del arreglo de sensores

En esta sección se muestra el resultado de la elaboración del arreglo de sensores. Este está conformado por un sistema de control, sensores y los terminales de comunicación I2C y alimentación.

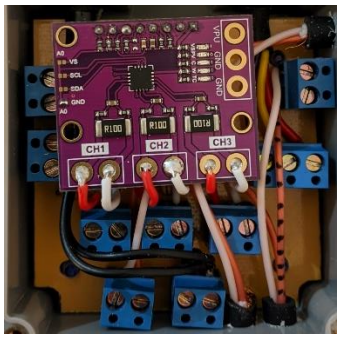



**Figura 3.1.** Sistema de control [Autor].

En la figura 3.1 se puede apreciar el resultado final de la elaboración del sistema de control del arreglo de sensores en una placa PCB la cual contiene al módulo INA3221 y las conexiones para los diferentes sensores, además de las conexiones para la alimentación tanto del módulo como de los sensores.

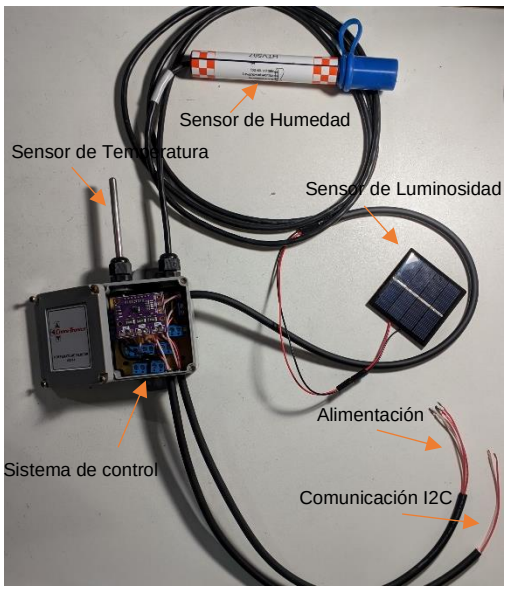



**Figura 3.2.** Arreglo de sensores completo [Autor].

En la figura 3.2 se puede observar el arreglo de sensores completo con todos los sensores, el sistema de control y los puertos de alimentación y comunicación I2C. Se puede observar que ningún elemento está directamente integrado en el sistema de control, se conectan a través de borneras para el caso de la alimentación, sensores y comunicación I2C y a través

de un header hembra para el módulo INA3221, así es sencillo reemplazar cualquiera de estos elementos.

### 3.1.2 Implementación de la interfaz de comunicación

En esta sección se muestran los resultados de la elaboración de la interfaz de comunicación entre la plataforma Arduino y los arreglos de sensores.

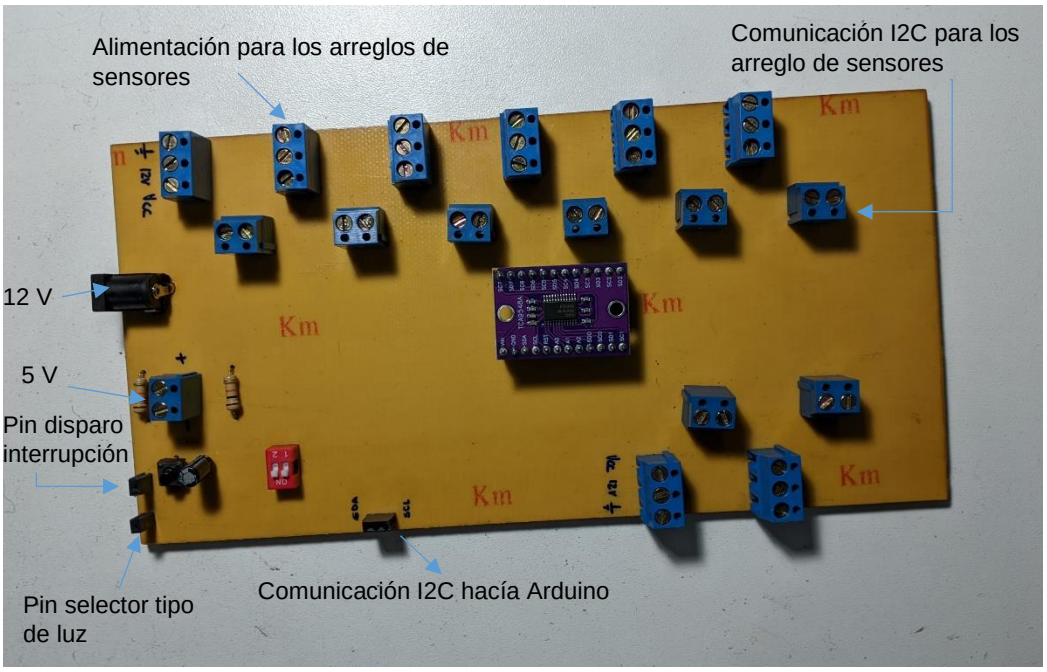



**Figura 3.3.** Interfaz de comunicación Arduino - Arreglos de sensores [Autor].

En la figura 3.3 se puede observar la implementación de la interfaz de comunicación entre la plataforma Arduino y los arreglos de sensores. Como se menciona en al apartado de diseño se utilizaron borneras y headers para las conexiones a los diferentes arreglos de sensores y para los terminales de comunicación y control de tipo de luz e interrupción que van hacia el Arduino. Además, cuenta con los terminales para la alimentación de los sensores que trabajan a 12 V y los módulos que conforman los arreglos de sensores y el multiplexor que trabajan a 5V.

Cada arreglo de sensores cuenta con 5 terminales, 3 de estos son para alimentación de sensores y del módulo INA3221, estos se conectan a las borneras de 3 entradas mientras que para la comunicación I2C se tienen las borneras de 2 entradas. De esta forma se tiene entradas para 8 dispositivos de control, pero de requerir más se puede conectar más de un arreglo de sensores en cada uno de estos puertos tomando en cuenta la dirección del módulo INA3221 del dispositivo.

### 3.1.3 Pruebas de funcionamiento del censado de parámetros ambientales

En esta sección se muestra el funcionamiento del proyecto en conjunto utilizando un arreglo de sensores para comparar los datos obtenidos por este con los obtenidos por los dispositivos utilizados para la calibración de los sensores. Se muestran los resultados de las mediciones para los tres parámetros ambientales Luminosidad, Temperatura y Humedad.

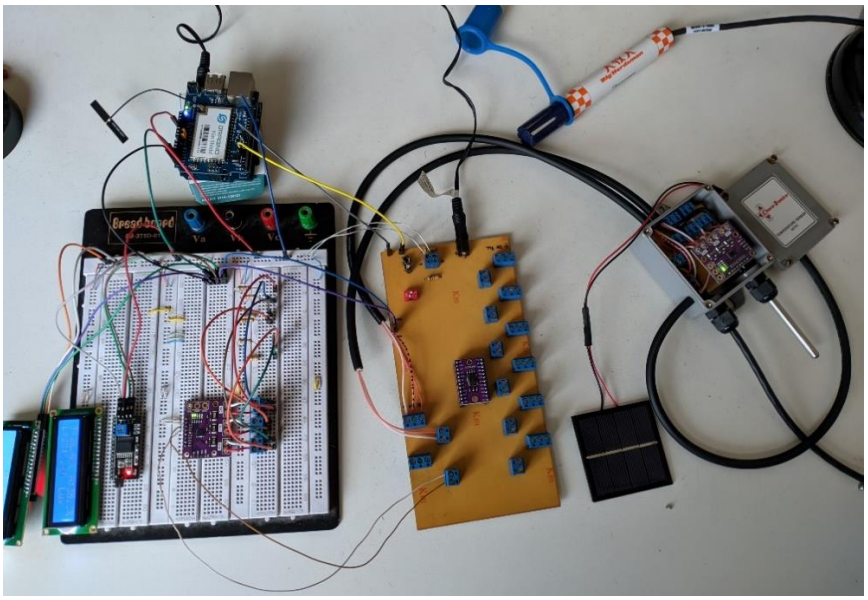


**Figura 3.4.** Sistema de adquisición de datos completo [Autor].

En la figura 3.4 se puede observar el sistema de adquisición de datos completo que está conformado por el arreglo de sensores, la interfaz de comunicación Arduino – Arreglo de sensores, dos LCD que muestran los datos adquiridos, un módulo INA3221 para comprobar funcionalidades de uso múltiple de arreglos de sensores y del Arduino UNO con el shield Dragino YUN. Como se puede apreciar el Arduino se alimenta por una fuente externa y no tiene comunicación alámbrica con la PC gracias a las funcionalidades provistas por el Shield Dragino Yun. Cabe mencionar que se hizo uso de un protoboard para alojar al módulo INA3221 y el arreglo de resistencias, ya que, esto se usó únicamente para probar funcionalidades.

### 3.1.3.1 Luminosidad

En este apartado se muestran los resultados obtenidos con el sensor de luminosidad y comparados con los resultados obtenidos por un luxómetro digital comercial. Estas pruebas se dividieron en dos partes al tener dos ambientes distintos.


**Figura 3.5.** Ejemplo de prueba 1 del sensor de luminosidad con luz natural [Autor].

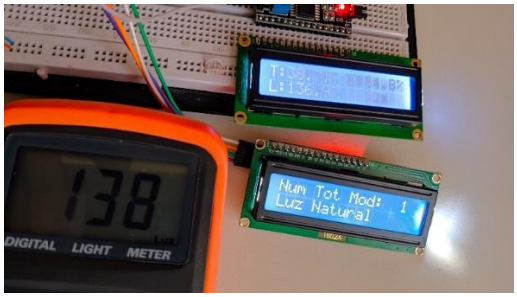



**Figura 3.6.** Ejemplo de prueba 2 del sensor de luminosidad con luz natural [Autor].

En la figura 3.5 y 3.6 se muestra un ejemplo de las pruebas realizadas con el sensor de Luminosidad en donde se puede observar los datos obtenidos tanto por el sistema de adquisición de datos implementado como los datos obtenidos por un luxómetro digital comercial, además se puede observar en el segundo LCD el mensaje que indica que el tipo de luz es natural. A continuación, en la tabla 3.1 se muestra una comparativa de los resultados obtenidos en las diferentes pruebas con el margen de error encontrado.

**Tabla 3.1.** Tabla de resultados del sensor de luminosidad con luz natural.

| No. Prueba | Sistema de adquisición implementado [luxes] | Luxómetro Digital [luxes] | Error [%] |
|---|---|---|---|
| 1 | 125.77 | 124 | 1.43 |
| 2 | 517.17 | 515 | 0.42 |

| | | | |
|---|---|---|---|
| 3 | 136.97 | 138 | 0.75 |
| 4 | 895.43 | 899 | 0.4 |
| 5 | 3568.18 | 3540 | 0.8 |
| 6 | 8950.57 | 8915 | 0.4 |
| 7 | 22430.09 | 22331 | 0.44 |
| 8 | 58957.86 | 58252 | 1.21 |
| 9 | 75645.18 | 74965 | 0.91 |
| 10 | 105178.11 | 104578 | 0.57 |

En la tabla 3.1 se muestran los resultados de las pruebas realizadas al sensor de Luminosidad del arreglo de sensores. Como se puede observar el error está en un rango de entre 0.4 a 1.5 % aproximadamente. A pesar de que el error es bastante bajo se debe mencionar que las medidas dependen mucho de la posición del sensor de luminosidad, así como la del luxómetro digital ya que una pequeña variación en su posición cambia la medida y aumenta o disminuye el error, esto debido a la forma del sensor principalmente que como se puede observar en el caso del luxómetro digital es circular y en el caso del arreglo de sensores es plano. Por esta razón se colocó a ambos sensores en una posición tal que la luz llegue de la manera más uniforme posible a los dos.

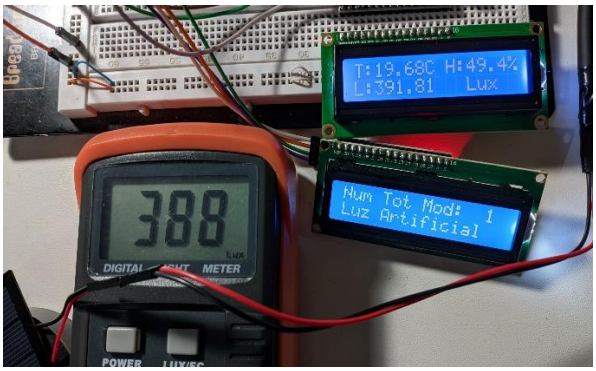



**Figura 3.7.** Ejemplo de prueba 1 sensor de luminosidad con luz artificial.

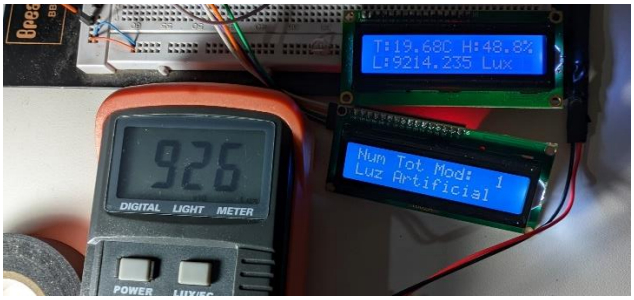



**Figura 3.8.** Ejemplo de prueba 2 del sensor de luminosidad con luz artificial.

En la figura 3.7 y 3.8 se muestran ejemplos de cómo se realizaron las pruebas del sensor de luminosidad al utilizar luz artificial; para este caso se utilizó la luz de una bombilla del comercial a diferentes distancias de los sensores para conseguir una variación de la luminosidad. Además, se puede observar en el segundo LCD un mensaje que indica que el tipo de luz que se está censando es artificial.

A continuación, en la tabla 3.2 se muestra una comparativa de los resultados obtenidos en las diferentes pruebas con el margen de error encontrado.

**Tabla 3.2.** Tabla de resultados del sensor de luminosidad con luz artificial.

| No. Prueba | Sistema de adquisición implementado [luxes] | Luxómetro Digital [luxes] | Error [%] |
|---|---|---|---|
| 1 | 78.36 | 79 | 0.81 |
| 2 | 81.50 | 81.60 | 0.1 |
| 3 | 84.47 | 84.43 | 0.05 |
| 4 | 391.81 | 388 | 0.98 |
| 5 | 1488.415 | 1480 | 0.57 |
| 6 | 2383.99 | 2290 | 4.1 |
| 7 | 3818.42 | 3860 | 1.07 |
| 8 | 5654.14 | 5640 | 0.25 |
| 9 | 5880.51 | 5940 | 1 |
| 10 | 9214.235 | 9260 | 0.5 |

En la tabla 3.2 se pueden observar los resultados obtenidos por el sensor de luminosidad trabajando con luz artificial. Al igual que trabajando con luz natural el error obtenido entre medidas es bajo que en la mayoría de los casos no supera el 2%; si bien se observa una medida en la cual el error sube hasta el 4% sigue estando de un rango adecuado por debajo del 5% de error.

### 3.1.3.2 Temperatura

En esta sección se muestran los resultados obtenidos con el sensor de temperatura del arreglo de sensores en comparación a los datos obtenido por un termómetro de mercurio. Para realizar las pruebas se utilizó, al igual que en la calibración, agua a diferentes temperaturas al ser un medio al cual se puede variar su temperatura de manera más sencilla que el aire.

**Figura 3.9.** Modelo utilizado para la prueba del sensor de temperatura [Autor].

En la figura 3.9 se puede observar el proceso que se realizó para obtener los datos y realizar las pruebas de censado. Se utilizo un recipiente de metal con el propósito de mantener la temperatura del agua estable.

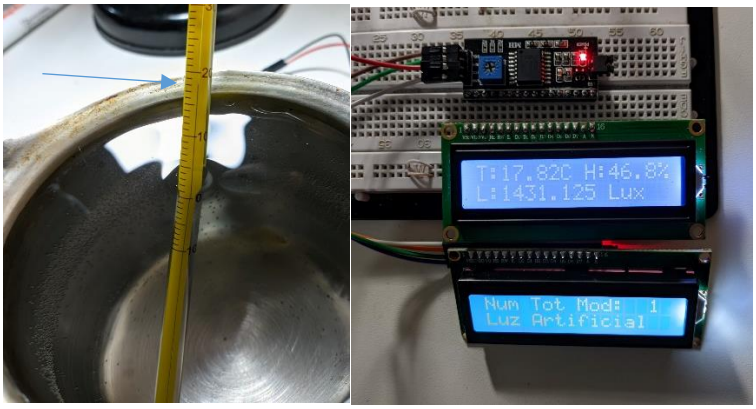



**Figura 3.10.** Ejemplo de prueba 1 para el sensor de temperatura.

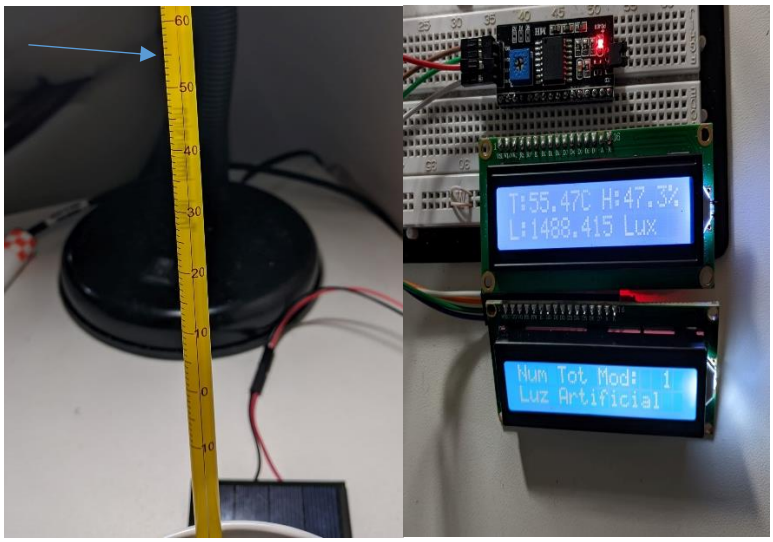



**Figura 3.11.** Ejemplo de prueba 2 para el sensor de temperatura.

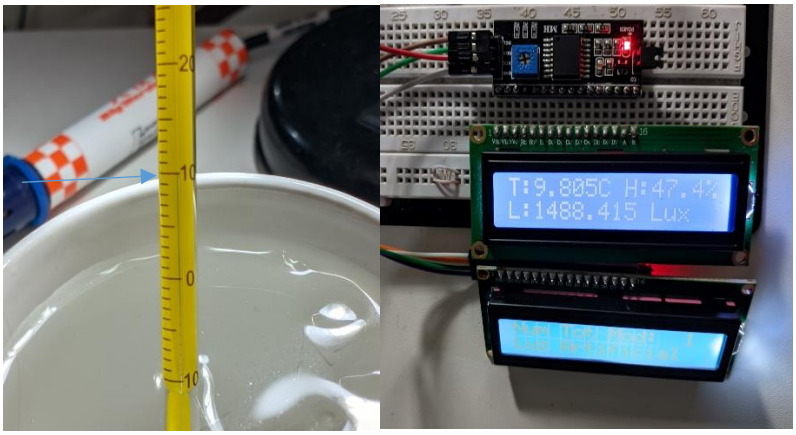



**Figura 3.12.** Ejemplo de prueba 3 para el sensor de temperatura.

En las figuras 3.10, 3.11 y 3.12 se muestran los ejemplos de cómo se realizaron las diferentes pruebas a diferentes temperaturas con el sensor de temperatura.

A continuación, en la tabla 3.3 se muestra el resultado de las medidas obtenidas por el sensor de temperatura del sistema implementado en comparación con las mediciones obtenidas por el termómetro de mercurio.

**Tabla 3.3.** Tabla de resultados del sensor de temperatura.

| No. Prueba | Sistema de adquisición implementado [ºC] | Termómetro de mercurio [ºC] | Error [%] |
|---|---|---|---|
| 1 | 5.12 | 5 | 2.4 |
| 2 | 6.91 | 7 | 1.29 |
| 3 | 9.80 | 10 | 2 |
| 4 | 14.79 | 15 | 1.4 |
| 5 | 17.82 | 18 | 1 |
| 6 | 23.53 | 23 | 2.3 |
| 7 | 36.65 | 37 | 0.95 |
| 8 | 55.47 | 55 | 0.85 |
| 9 | 67.94 | 68 | 0.09 |
| 10 | 75.78 | 76 | 0.29 |

En la tabla 3.3 se pueden observar los resultados obtenidos por el sensor de Temperatura. Como se puede apreciar el rango de error varia desde 0.09 hasta 2.4 % estando estos valores por debajo del 5% que se tiene como máximo. El error podría ser menor debido a que el termómetro posee una precisión de 1 grado mientras que el sistema de adquisición

implementado maneja una precisión de 0.01 grados por lo cual es bastante complicado tener un valor exacto en las mediciones utilizando este equipo. Sin embargo, la variación de temperatura que representa este margen de error no es relevante.

### 3.1.3.3 Humedad

En esta sección se muestran los resultados obtenidos con el sensor de humedad. Para este sensor se tomaron medidas variando el ambiente al que está expuesto este. Al ser un sensor bastante preciso no se necesitó de una calibración previa. Por este motivo se realizó una tabla con diferentes medidas tomadas probando que al variar el parámetro de humedad ambiental al que se expone, la lectura del sistema de adquisición también cambia.

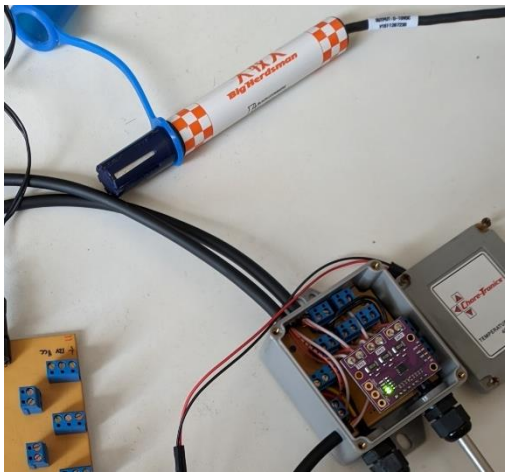


**Figura 3.13.** Modelo de prueba para el sensor de humedad.

En la tabla 3.4 se muestran los resultados obtenidos del sensor de humedad.

**Tabla 3.4.** Tabla de resultados del sensor de humedad.

| No. Prueba | Sistema de adquisición implementado [% de humedad] |
|---|---|
| 1 | 45 |
| 2 | 55 |
| 3 | 58 |
| 4 | 68 |
| 5 | 86 |

En la tabla 3.4 se pueden observar las distintas medidas obtenidas por el sensor de humedad. Como se puede apreciar los valores van desde 45 % hasta 79%, esto debido a

que es un parámetro que depende de la humedad del aire, siendo sencillo aumentarla, pero muy complicado disminuirla; el valor mínimo obtenido fue al aire libre en la noche y el máximo utilizando un equipo de vaporización en una habitación cerrada.

### 3.1.4 Pruebas de funcionamiento del sistema de adquisición de datos utilizando más de un arreglo de sensores.

En esta sección se muestran los resultados obtenidos cuando se maneja más de un arreglo de sensores; con el fin de simular otro de estos dispositivos, se utilizó un módulo INA3221 similar al que integran estos dispositivos y a los canales de censado de voltaje se conectó un arreglo de resistencias teniendo un valor fijo de voltaje. Los objetivos de estas pruebas son primeramente ver como el sistema de adquisición desarrollado funciona para más de un arreglo de sensores; aquí se incluye la muestra de datos promedio de ambos arreglos de sensores y a su vez la muestra de datos individuales de cada dispositivo.

En la figura 3.4 se puede observar el esquema de pruebas completo incluido al módulo INA3221 con el arreglo de resistencias en cada canal. En la figura 3.14 se puede observar esta adaptación de arreglo de sensores.

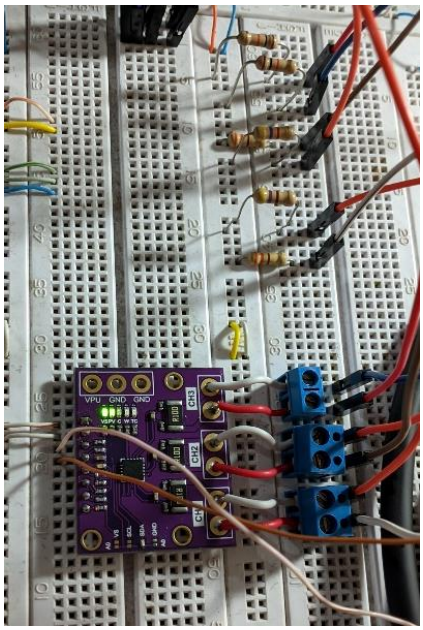



**Figura 3.14.** Adaptación de un arreglo de sensores.

El valor de las resistencias fue escogido a fin de tener un valor de cada parámetro en donde se pueda visualizar la variación al realizar el promedio, es decir, un valor no tan común.


**Figura 3.15.** LCDs al iniciar el sistema de adquisición.

En la figura 3.15 se puede observar los mensajes mostrado por las LCDs al iniciar el sistema de adquisición de datos implementado. En la pantalla superior se puede observar los valores promedio tomando en cuenta los dos arreglos de sensores y en la pantalla inferior se muestra el número de módulos conectados y el tipo de luz. Para poder mostrar los valores de cada módulo se debe disparar la interrupción y se debe tener un control sobre cuál es el orden de los arreglos de sensores conectados; este orden se lo define de acuerdo con lo explicado en la sección de metodología.


**Figura 3.16.** LCDs al disparar por primera vez la interrupción.

En la figura 3.16 se pueden observar los mensajes que se muestran en las pantallas LCD una vez disparada por primera vez la interrupción. En el caso de la pantalla superior no varía su contenido, es decir, continúa mostrando el valor promedio de los parámetros ambientales censados por los dos dispositivos conectados; pero en el caso de la segunda pantalla se puede apreciar el valor de los parámetros ambientales obtenidos por el primer arreglo de sensores, esto se aprecia observando el número alado de la letra que representa cada parámetro.


**Figura 3.17.** LCDs al disparar por segunda vez la interrupción.

En la figura 3.17 se puede observar el contenido de las pantallas al disparar por segunda vez la interrupción. La pantalla inferior no cambia por la interrupción así que muestra el mismo contenido. Por otro lado, la pantalla superior muestra los datos obtenidos por el segundo arreglo de sensores y se evidencia por el numero alado de la letra que representa cada parámetro. Si se compara con la figura 3.16 los parámetros cambian entre arreglos de sensores como se esperaba y en la figura 3.15 se muestra el promedio de las mediciones de ambos dispositivos.

Finalmente, en la tabla 3.5 se presenta la estructura completa del arreglo en donde se almacenan los datos censados.

**Tabla 3.5.** Estructura del almacenamiento de datos.

| |
|---|
| Dato de luminosidad del sensor 1 |
| Dato de luminosidad del sensor 2 |
| .<br>.<br>. |
| Dato de luminosidad del sensor n |
| Dato de temperatura del sensor 1 |
| Dato de temperatura del sensor 2 |
| .<br>.<br>. |
| Dato de temperatura del sensor n |
| Dato de humedad del sensor 1 |
| Dato de humedad del sensor 2 |
| .<br>.<br>. |
| Dato de humedad del sensor n |

En la tabla 3.5 se puede apreciar la organización de los datos en el arreglo en donde se almacenan. Los datos se clasifican primeramente por la variable ambiental a la cual representan, pero además se toma en cuenta el arreglo de sensores del cual proviene cada dato. El primer grupo de datos pertenece a la luminosidad, seguido de la temperatura y finalmente la humedad. Tomando en cuenta esto se tiene la organización mostrada en la tabla 3.5 siendo *n* el número de dispositivos conectado pudiendo ser un máximo de 32.

## 3.2 Conclusiones

- El sistema de adquisición de datos implementado presenta resultados con un margen de error por debajo del 5% tomando en cuenta datos de equipos comerciales que miden específicamente los parámetros ambientales que conforman este sistema como son Luminosidad y Temperatura, esto debido a que dentro del proceso se tomaron en cuenta detalles como la calibración de los sensores previo a su implementación en un dispositivo único, permitiendo reducir el posible error que se presente. Además, el principio de funcionamiento del sistema se basa en la adquisición de una variación de voltaje producido por los sensores ambientales lo que permite manejar los errores y corregirlos de una manera sencilla y eficaz mediante el software.

- Al utilizar la plataforma Arduino, que es una plataforma de código abierto, es posible hacer uso de librerías específicas para cada uno de los dispositivos que integran este sistema de adquisición de datos y manejan la tecnología I2C, permitiendo la incorporación de estos de una manera más sencilla a que si se trabajase directamente con funciones genéricas de la comunicación I2C. Esto resulta en un algoritmo de adquisición de datos más intuitivo y que se enfoca más en presentar una solución que sea eficaz para todo el sistema y no para cada dispositivo por separado.

- El módulo INA3221 permite, gracias a sus 3 canales, incorporar en un solo arreglo de sensores tres tipos de sensores diferentes, ya sea, para medir un solo parámetro ambiental o como en este caso varios, esto facilita la implementación del arreglo de sensores y reduce el consumo del sistema completo, ya que, si se tuviese un módulo con un solo canal de censado para cada sensor resulta ser ineficiente en cuanto al consumo y desarrollo del sistema implementado.

- La solución presentada en este trabajo de integración se centra en el uso de módulos INA3221 y un multiplexor TCA9548a, la cual, integra la tecnología de

comunicación I2C, para poder manejar una gran cantidad de arreglos de sensores por medio de dos pines en conjunto con la plataforma Arduino, y sensores que no la poseen esta tecnología; esta integración permite solventar uno de los problemas que se tiene con la comunicación I2C y es que al utilizar dispositivos que manejan este tipo de tecnología se debe tomar en cuenta el limitante de las direcciones posibles para estos, es decir, si bien es cierto que los dispositivos I2C se manejan mediante dos pines, solamente se podrán utilizar el número de dispositivos que sus direcciones lo permitan, por este motivo esta solución hace posible que se multiplique el número de sensores posibles y el problema de direcciones no es tan crucial como si se trabajase directamente con sensores que trabajen con comunicación I2C. El módulo INA3221 es el encargado de integrar los sensores en un solo dispositivo, pero es el multiplexor TCA9548a el que realmente hace posible que se trabaje con un numero significativamente mayor de sensores, debido a que el módulo INA3221 es un dispositivo I2C que permite tener solamente 4 direcciones lo que resulta en el limitante mencionado anteriormente, y a pesar de que el multiplexor TCA9548a también es un dispositivo I2C, cuenta con 7 direcciones posibles, además cuenta con 8 canales por cada dispositivo y en cada uno de estos canales se puede conectar los cuatro módulos INA3221 cada uno con una dirección diferente, lo que hace que el sistema sea considerablemente escalable pudiéndose hacer uso de hasta 32 módulos INA3221 por cada multiplexor.

- El algoritmo de control de este sistema se desarrolló con el objetivo de tener una estructura organizada de los datos, tal que, permita acceder a estos de una manera sencilla y eficaz evitando que se pierda tiempo y recursos cuando se intente acceder a estos. Este principio se basa en clasificar los datos de acuerdo con la variable que corresponda y a su vez organizarlos de acuerdo con el número de arreglo de sensores que corresponda, por este motivo es importante tener un control sobre los datos adquiridos y sobre el dispositivo que los provee.

- Los sensores utilizados no manejen tecnología I2C lo que hizo posible realizar un sistema escalable, pero al ser sensores que basan su principio de funcionamiento en la variación de voltaje de acuerdo con la variación del parámetro ambiental que miden, presentan errores al momento de utilizarlos, ya que, dependen del voltaje que se les suministre y de la calibración previa necesaria para manejarlos. A pesar de tomar en cuenta estos puntos, el simple hecho que miden parámetros que varían constantemente hacen que su calibración resulte complicada, ya que, si bien se hace uso de equipos especializados en medir estos parámetros no siempre se tiene

una misma medida de voltaje para un mismo valor del parámetro que se esté midiendo y debido a esto se debe adquirir un número muy grande de datos para estimar un comportamiento cercano a lo ideal.

- En sensor de luminosidad utilizado fue de tipo celda fotovoltaica y fue el que generó más complicaciones al momento de implementarlo en el sistema. Esto debido a que su rango de variación de voltaje es demasiado corto de 0 a 3V, lo que genera conflicto debido a que la luminosidad es un parámetro que varía dentro de un amplio rango que va desde los 0 hasta los 150000 luxes como mínimo; por este motivo el representar tal rango de luminosidad para el corto rango de variación de voltaje del sensor fue un reto. Se presentaron muchos errores a pesar de la calibración y por el mismo hecho de que la forma del sensor utilizado comparado con el equipo utilizado para la calibración era muy distinta siendo este ultimo de forma circular lo que le permitía captar más luz en ciertas posiciones. Por este motivo para realizar tanto la calibración como las pruebas se tomó muy en cuenta la posición del sensor y del equipo de medición de tal manera que ambos estén expuesto a la misma cantidad de luz. Además, el hecho de que el rango en el que varía la luminosidad es muy extenso hace que un valor de 1000 luxes por ejemplo no represente ni el 1% de error tomando en cuenta valores muy grandes y gracias a esto se pudo corregir el comportamiento del sensor a tal punto que si bien existe un margen error sea casi imperceptible en todos los rangos.

- La plataforma Arduino en conjunto con el shield Dragino Yun permitió que el desarrollo de este proyecto sea más eficiente debido a que se pudo modificar las diferentes versiones del algoritmo de control mediante una red inalámbrica sin necesidad de que exista comunicación a través de un cable serial con la PC. El proyecto del cual este componente es parte no hace uso de las capacidades de la comunicación que posee el Shield según los objetivos propuestos, sim embargo ha sido de mucha utilidad ya que esto hizo más cómodo el desarrollo e implementación del sistema completo ya que no era necesario que se implemente en un lugar cercano a la PC.

- El esquema de escalabilidad previsto permitió que se agregue posterior al desarrollo de este proyecto sensores de humedad del suelo sin que se haya tenido que realizar cambios substanciales al hardware y a los módulos de software desarrollados.

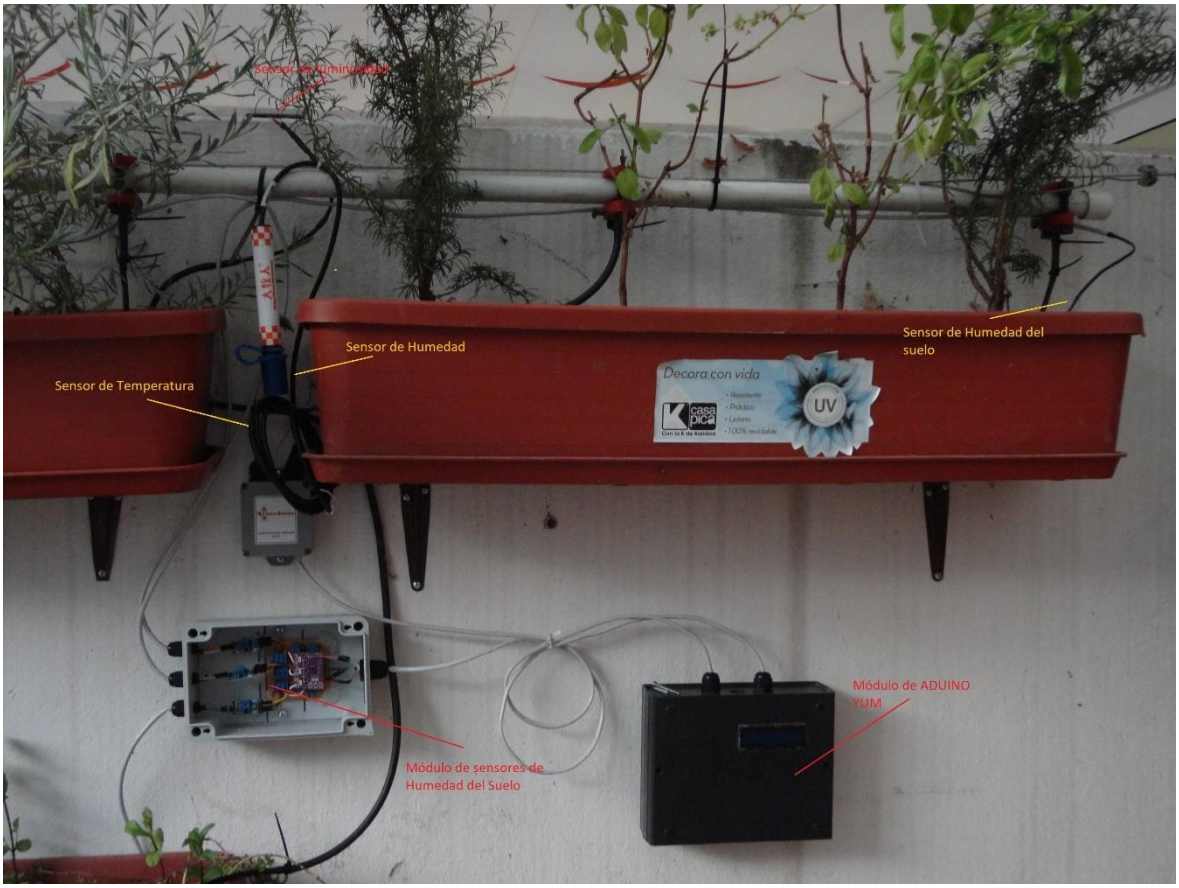



**Figura 3.18.** Sistema de adquisición de datos instalado en un jardín urbano [Autor].

- En la figura 3.18 se muestra el sistema de adquisición de datos instalado en un jardín urbano que cumple con las características de un pequeño invernadero, se muestran los módulos construidos como parte de este trabajo a saber: módulo del sistema embebido que contiene el Arduino YUN, módulo de sensores de temperatura, humedad del ambiente y luminosidad, módulo de sensores de humedad del suelo. Se puede concluir que un esquema estructurado de realización facilita el proceso de instalación y que las capacidades de crecimiento previstas han permitido incluir el módulo sensor de humedad del suelo. Este módulo se agregó de forma posterior sin que se haya realizado modificaciones importantes en el sistema o el código.

## 3.3 Recomendaciones

- Se recomienda utilizar el número de arreglos de sensores necesarios para cubrir el espacio designado para el invernadero en donde se desee implementar este sistema de adquisición de parámetros ambientales. El uso de un número mayor al

necesario de arreglos de sensores provocaría un consumo innecesario de energía y la precisión de los resultados no representaría ese consumo.

- Se debe tomar en cuenta que los sensores que trabajan con I2C no son la mejor solución para este tipo de aplicaciones. A pesar de que cuenten con una muy buena precisión y un fácil manejo, el monitoreo de parámetros ambientales para un invernadero no necesita de una precisión muy alta, debido a que la variación en cierto rango entre el valor real y el medido no afecta al desarrollo de los cultivos siempre y cuando este dentro de un margen, además los sensores que trabajan con I2C están encapsulados generalmente en módulos tipo placa electrónica, lo cual, resulta incómodo al momento de ubicarlos dentro de un invernadero, además que al ser sensores pequeños cubren una distancia muy corta.

- Se propone como disyuntiva para un futuro estudio, implementar un mecanismo de control del consumo de energía para este sistema con el fin saber si se es posible implementarlo junto con otros sistemas directamente o si debería considerarse una fuente de alimentación externa.

- Se recomienda que el sistema de adquisición de datos y el Arduino estén conectados a la misma referencia, esto debido a que el sensor de luminosidad y las pantallas LCD presentan errores al no cumplirse esta condición, así si es que se decidiese utilizar una fuente externa de alimentación el funcionamiento del sistema no se vea afectado.

- Se recomienda asegurar que la fuente de alimentación para los sensores sea de 12 V, debido a que el funcionamiento de estos se basa en la alimentación que se los provea para establecer el rango de variación de voltaje. Debido a que este sistema de adquisición de datos fue diseñado para una alimentación de 12 V el utilizar una fuente de alimentación diferente provocaría que los resultados presentados sean erróneos. Si se desease utilizar una fuente con un voltaje diferente se debe calibrar nuevamente los sensores y obtener las expresiones de su comportamiento para implementarlas dentro del algoritmo de adquisición de datos.

- Una vez realizadas las pruebas se pudo verificar que si bien es cierto las plantas requieren de luz para su proceso de desarrollo, sin embargo, al no ser una variable física que se pueda alterar fácilmente este parámetro debe ser considerado no crítico dentro del proceso.

- Se recomienda que el sistema desarrollado sea sometido a pruebas en condiciones reales con el propósito de evaluar su desempeño y permitir su interconexión con los otros componentes que forman parte de este proyecto.
- Otro elemento que podría ser de mucha utilidad para el seguimiento de los cultivos y el análisis de las condiciones meteorológicas del invernadero seria la inclusión de estampas de tiempo en el conjunto de lectura de manera que permitan este análisis. Se recomienda adicionalmente que la estampa de tiempo no este asociada con lecturas individuales sino con valores consolidados de las variables físicas registradas.
- Se recomienda que de la puesta en escenarios reales del prototipo se establezca un periodo adecuado de muestreo que permita implantar técnicas de ahorro de energía.

# REFERENCIAS BIBLIOGRÁFICAS


[1] Mungo Riego, "Mundo Riego," [Online]. Available: https://mundoriego.es/automatizacion-de-sistemas-de-riego/. [Accessed 25 05 2022].

[2] Instituto Nacional de Seguridad y Salud en el Trabajo, "¿Qué es un invernadero?," INSST, [Online]. Available: https://www.insst.es/-/-que-es-un-invernader-1. [Accessed 29 05 2022].

[3] Arduino, "Arduino CC: What is Arduino?," 05 02 2018. [Online]. Available: https://www.arduino.cc/. [Accessed 25 05 2022].

[4] Arduino CL, "Arduino Yun," [Online]. Available: https://arduino.cl/arduino-yun/#:~:text=El%20Arduino%20Yun%20es%20una,al%20panel%20web%20de%20Y%C3%BAn. [Accessed 25 05 2022].

[5] HetPro-Store, "I2C," [Online]. Available: https://hetpro-store.com/TUTORIALES/i2c/#:~:text=I2C%20es%20un%20puerto%20y,de%20comunicaci%C3%B3n%2C%20SDA%20y%20SCL.. [Accessed 07 06 2022].

[6] Programar Facil, "INA219 mide el consumo de tus proyectos con Arduino," [Online]. Available: https://programarfacil.com/blog/arduino-blog/sensor-ina219-arduino-esp8266-esp32/. [Accessed 25 06 2022].

[7] P. E. Demin, "Aportes para el mejoramiento del manejo de los sistemas de riego," 2014. [Online]. Available: https://inta.gob.ar/sites/default/files/inta_aportes_para_el_mejoramiento_del_manejo_de_los_sistemas_de_riego.pdf. [Accessed 29 05 2022].

[8] Wikipedia, "Sistema de riego," Wikipedia, 16 09 2021. [Online]. Available: https://es.wikipedia.org/wiki/Sistema_de_riego. [Accessed 29 05 2022].

[9] Sembralia, "Tipos de sistemas de riego en invernaderos," 2020 08 12. [Online]. Available: https://sembralia.com/blogs/blog/sistema-de-riego-en-invernaderos. [Accessed 29 05 2022].

[10] NOVAGRIC, "Riego por goteo," Novedades Agrícolas S.A., [Online]. Available: https://www.novagric.com/es/riego/sistemas-de-riego/riego-por-goteo. [Accessed 29 05 2022].

[11] Cenicana, "Riego por aspersion," 17 03 2015. [Online]. Available: https://www.cenicana.org/riego-por-aspersion/#:~:text=El%20riego%20por%20aspersi%C3%B3n%20consiste,el%20mantenimiento%20de%20los%20equipos.. [Accessed 2022 05 2022].

[12] NOVAGRIC, "Riego por Microaspersión," Novedades Agrícolas S.A., [Online]. Available: https://www.novagric.com/es/riego/sistemas-de-riego/riego-por-microaspersion. [Accessed 30 05 2022].

[13] NOVAGRIC, "Riego hidroponico," Novedades Agrícolas S.A., [Online]. Available: https://www.novagric.com/es/riego/sistemas-de-riego/riego-hidroponico. [Accessed 30 05 2022].

[14] NOVAGRIC, "Riego por nebulizacion," Novedades Agrícolas S.A., [Online]. Available: https://www.novagric.com/es/riego/sistemas-de-riego/riego-por-nebulizacion. [Accessed 30 05 2022].

[15] I. Bustos, Meteorología descriptiva, Santiago de Chile: Universitaria, 2019.

[16] Wikipedia, "Temperatura," 29 05 2022. [Online]. Available: https://es.wikipedia.org/wiki/Temperatura#Definici%C3%B3n. [Accessed 30 05 2022].

[17] Khan Academy, "Escalas de temperatura," [Online]. Available: https://es.khanacademy.org/science/fisica-pe-pre-u/x4594717deeb98bd3:energia-cinetica/x4594717deeb98bd3:calor-y-temperatura/a/643-escalas-de-temperatura. [Accessed 30 05 2022].

[18] Secoin, "HUMEDAD RELATIVA: QUÉ ES Y POR QUÉ ES IMPORTANTE CONTROLARLA," 25 01 2019. [Online]. Available: https://www.secoin.com.uy/blog/humedad-relativa-qu%C3%A9-es-y-por-qu%C3%A9-es-importante-controlarla. [Accessed 30 05 2022].

[19] M. Sonoma, "LA HUMEDAD RELATIVA EN INVERNADERO," Nutricontrol, 29 01 2020. [Online]. Available: https://nutricontrol.com/es/la-humedad-relativa-en-invernadero/#:~:text=La%20humedad%20relativa%20en%20un,se%20pueda%20llevar%20a%20cabo.. [Accessed 30 05 2022].

[20] ACEA, "Factores ambientales y su influencia en invernaderos. La luz dentro del invernadero," ACEA: Invernaderos para el mundo, [Online]. Available: https://acea.com.mx/articulos-tecnicos/60-los-factores-ambientales-y-su-influencia-en-invernaderos-313-la-luz-dentro-del-invernadero#:~:text=La%20luminosidad%20aumenta%20en%20los,a%20trav%C3%A9s%20de%20la%20cubierta.. [Accessed 30 05 2022].

[21] NOVAGRIC, "Clima de un Invernadero. ¿Cómo conseguir la Temperatura Ideal?," Novedades Agrícolas S.A., [Online]. Available:

https://www.novagric.com/es/blog/articulos/clima-invernadero-como-conseguir-temperatura-ideal#:~:text=Temperatura%20en%20invernaderos&text=Generalmente%2C%20la%20temperatura%20m%C3%ADnima%20requerida,resultar%20beneficiosa%20para%20las%20plantas.. [Accessed 30 05 2022].

[22] A. P. T. y. R. G. Lopez, "Medición de Luz Diaria Integrada en Invernaderos," *Purdue Extension,* vol. 1, p. 3, 2019.

[23] HYDRO Environment, "Guía: La luz en tus plantas," HYDRO Environment, 2022. [Online]. Available: https://www.hydroenv.com.mx/catalogo/index.php?main_page=page&id=221. [Accessed 08 2022].

[24] P. M. Moreno, A. N. Quercop, P. L. Murcia and V. N. Lidon, La iluminacion en los invernaderos, Orihuela: Limencop S.L., 2002.

[25] Arduino, "Arduino UNO R3," 2022. [Online]. Available: https://docs.arduino.cc/hardware/uno-rev3. [Accessed 16 06 2022].

[26] Arduino CL, "¿Que es un Shield?," MCI Electronics, [Online]. Available: https://arduino.cl/que-es-un-shield/. [Accessed 05 06 2022].

[27] ARDUINO, "Arduino Yún Shield," ARDUINO CC, [Online]. Available: https://docs.arduino.cc/retired/shields/genuino-yun-shield. [Accessed 05 06 2022].

[28] Texas Instrument, "INA3221 Triple-Channel, High-Side Measurement," 05 2016. [Online]. Available: https://www.ti.com/lit/ds/symlink/ina3221.pdf?HQS=TI-null-null-alldatasheets-df-pf-SEP-wwe&ts=1660521152972&ref_url=https%253A%252F%252Fwww.alldatasheet.com%252F. [Accessed 08 2022].

[29] ESPHome, "INA3221 3-Channel DC Current Sensor," [Online]. Available: https://esphome.io/components/sensor/ina3221.html. [Accessed 08 2022].

[30] Texas Instrument, "TCA9548A Low-Voltage 8-Channel I," 11 2019. [Online]. Available: https://www.ti.com/lit/ds/symlink/tca9548a.pdf?ts=1660523958399&ref_url=https%253A%252F%252Fwww.google.com%252F. [Accessed 08 2022].

[31] Naylamp Mechatronics, "MULTIPLEXOR I2C TCA9548A 8CH," 2020. [Online]. Available: https://naylampmechatronics.com/multiplexores/637-multiplexor-i2c-tca9548a-8ch.html. [Accessed 08 2022].

[32] RC Argentina, "Descipcion y funcionamiento del bus I2C," [Online]. Available: https://robots-argentina.com.ar/didactica/descripcion-y-funcionamiento-del-bus-i2c/. [Accessed 07 06 2022].

[33] S. A. Castano, "Comunicacion I2C," [Online]. Available: https://controlautomaticoeducacion.com/microcontroladores-pic/comunicacion-i2c/. [Accessed 06 06 2022].

[34] Dignal, "Bus I2C," 25 03 2015. [Online]. Available: https://dignal.com/el-bus-i2c/#:~:text=Por%20ejemplo%2C%20es%20com%C3%BAnmente%20usado,de%20dispositivos%20compatibles%20con%20I2C%20.. [Accessed 07 06 2022].

[35] J. G. Carmenate, "Comunicación I2C con Arduino lo mejor de dos mundos," Programar Facil, [Online]. Available: https://programarfacil.com/blog/arduino-blog/comunicacion-i2c-con-arduino/. [Accessed 07 06 2022].

[36] Wikipedia, "Sensor," 23 05 2022. [Online]. Available: https://es.wikipedia.org/wiki/Sensor. [Accessed 08 06 2022].

[37] A. P. Malvino, Principios de Electrónica, Mc Graw Hill, 2000.

[38] CTB, Inc., "Controles modelos 16-24. Manual de instrucciones del operador e instalación," 2018.

[39] I. R. E. M. Tobar, "DESARROLLO DE UN MODELO MATEMÁTICO PARA LA EVALUACIÓN DE LOS PARÁMETROS DE OPERACIÓN DE NAVES EN LA CRÍA DE POLLOS DE ENGORDE," 05 2017. [Online]. Available: https://docplayer.es/76487896-Escuela-politecnica-nacional.html. [Accessed 08 2022].

[40] Cytron, "Célula/panel solar 3 V 160 mA (0,48 W) Cable soldado," Cytron Marketplace, 2021. [Online]. Available: https://www.cytron.io/p-solar-cell-panel-3v-160ma-0.48w-wire-soldered. [Accessed 08 2022].

[41] Wikipedia, "LCD," 31 05 2022. [Online]. Available: https://en.wikipedia.org/wiki/Liquid-crystal_display. [Accessed 08 06 2022].

[42] L. d. V. Hernandez, "Texto en movimiento con un LCD y Arduino," ProgramarFacil, [Online]. Available: https://programarfacil.com/blog/arduino-blog/texto-en-movimiento-en-un-lcd-con-arduino/#LCD_Liquid_Crystal_Display. [Accessed 08 06 2022].

[43] NaylampMechatronics, "TUTORIAL LCD CON I2C, CONTROLA UN LCD CON SOLO DOS PINES," 2021. [Online]. Available: https://naylampmechatronics.com/blog/35_tutorial-lcd-con-i2c-controla-un-lcd-con-solo-dos-pines.html. [Accessed 08 06 2022].

# ANEXOS

ANEXO I. Tablas de calibración sensor de luminosidad

ANEXO II. Tabla de calibración sensor de temperatura

ANEXO III. Algoritmo de control completo

# ANEXO I

**Tablas de calibración sensor de luminosidad**

| Luminosidad luz natural | | Luminosidad luz artificial | |
|---|---|---|---|
| **Luxes** | **Volts** | **Lux** | **Volts** |
| 640 | 1.813 | 41 | 0.959 |
| 100 | 1.48 | 36 | 0.94 |
| 60 | 1.394 | 45 | 0.999 |
| 400 | 1.731 | 76 | 1.131 |
| 1300 | 1.924 | 88 | 1.17 |
| 5200 | 2.16 | 170 | 1.335 |
| 5100 | 2.13 | 500 | 1.56 |
| 8300 | 2.2 | 2400 | 1.79 |
| 394 | 1.69 | 14000 | 2.1 |
| 116 | 1.517 | 18000 | 2.15 |
| 52 | 1.321 | 1500 | 1.75 |
| 36 | 1.263 | 70 | 1.1 |
| 25 | 1.208 | 5000 | 1.94 |
| 30 | 1.238 | 90 | 1.184 |
| 15500 | 2.3 | 130 | 1.27 |
| 292 | 1.657 | 20 | 0.75 |
| 44 | 1.293 | 200 | 1.39 |
| 34 | 1.242 | 80 | 1.14 |
| 45 | 1.299 | 30 | 0.925 |
| 22 | 1.115 | 48 | 1.01 |
| 104 | 1.505 | 3000 | 1.87 |
| 2600 | 2.03 | 23 | 0.81 |
| 32000 | 2.46 | 600 | 1.62 |
| 148000 | 2.76 | 2600 | 1.87 |
| 33000 | 2.47 | 5100 | 1.96 |
| 2900 | 2.04 | 22000 | 2.2 |
| 133000 | 2.73 | 33000 | 2.3 |
| 4900 | 2.09 | 6500 | 2.01 |
| 3500 | 2.06 | 9200 | 2.04 |
| 1100 | 1.896 | 3500 | 1.88 |
| 217 | 1.61 | 2000 | 1.86 |
| 740 | 1.831 | 1700 | 1.77 |
| 670 | 1.827 | 1200 | 1.72 |
| 2 | 0.587 | 800 | 1.64 |
| 7 | 0.772 | 700 | 1.62 |
| 16 | 1.092 | 250 | 1.42 |
| 18 | 1.1 | 21 | 0.75 |
| 85000 | 2.65 | | |
| 73000 | 2.6 | | |

| | |
|---|---|
| 62000 | 2.6 |
| 60000 | 2.58 |
| 51000 | 2.55 |
| 46000 | 2.5 |
| 110000 | 2.66 |
| 122000 | 2.69 |
| 9 | 0.837 |
| 15 | 0.985 |
| 5 | 0.677 |
| 12 | 0.886 |
| 26000 | 2.45 |
| 43000 | 2.495 |
| 35000 | 2.475 |
| 38000 | 2.48 |
| 24000 | 2.44 |
| 13000 | 2.27 |
| 31000 | 2.455 |
| 27000 | 2.447 |
| 18000 | 2.37 |
| 14000 | 2.27 |

# ANEXO II

## Tabla de calibración sensor de temperatura

| Temperatura | |
|---|---|
| **ºC** | **Volts** |
| 87 | 11.02 |
| 18 | 5.13 |
| 16 | 4.85 |
| 72 | 10.46 |
| 5 | 3.64 |
| 67 | 10.14 |
| 15 | 4.77 |
| 63 | 9.89 |
| 19 | 5.25 |
| 59 | 9.6 |
| 13 | 4.53 |
| 55 | 9.28 |
| 10 | 4.13 |
| 50 | 8.91 |
| 7 | 3.8 |
| 47 | 8.64 |
| 18 | 5.08 |
| 9 | 4.04 |
| 44 | 8.33 |
| 36 | 7.48 |
| 42 | 8.16 |
| 31 | 6.87 |
| 41 | 8 |
| 12 | 4.34 |
| 40 | 7.91 |
| 29 | 6.62 |
| 38 | 7.7 |
| 27 | 6.36 |
| 37 | 7.55 |
| 25 | 6.14 |
| 33 | 7.11 |
| 15 | 4.78 |
| 77 | 10.62 |
| 85 | 10.95 |
| 81 | 10.81 |
| 75 | 10.57 |

# ANEXO III

## Algoritmo de control completo

```
#include <Wire.h>
#include "SDL_Arduino_INA3221.h"
#include <LiquidCrystal_I2C.h>

#define C_Luminosidad 1
#define C_Temperatura 2
#define C_Humedad 3
#define S_Lum 7

SDL_Arduino_INA3221 modulo_1 (64);
SDL_Arduino_INA3221 modulo_2 (65);
SDL_Arduino_INA3221 modulo_3 (66);
SDL_Arduino_INA3221 modulo_4 (67);


LiquidCrystal_I2C lcd(0x27, 16, 2);
LiquidCrystal_I2C lcd_1(0x26, 16, 2);

int Direcciones_Modulos[] = {64, 65, 66, 67};
int N_Modulos_Ch [] = {0, 0, 0, 0, 0, 0, 0, 0};
int NT_M = 0;
int Dir_Mod_Av_Ch[8][4];
boolean T_Luz = false;
int intPin = 2;
int select_LCD = 0;
String Tipo_Luz = " ";
int timeThreshold = 150;
long startTime = 0;
float Medidas_Final [3];

void setup() {

  pinMode(intPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(intPin), blink, LOW);

  pinMode(S_Lum, INPUT);

  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Tesis Final");
  lcd.setCursor(0,1);
  lcd.print("Francisco Valdez");
  delay(2000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("T:");
  lcd.setCursor(9,0);
  lcd.print("H:");
  lcd.setCursor(0,1);
  lcd.print("L:");
  lcd.setCursor(7,0);
```

```
  lcd.print("C");
  lcd.setCursor(15,0);
  lcd.print("%");
  lcd.setCursor(11,1);
  lcd.print("Lux");

  lcd_1.init();
  lcd_1.backlight();
  lcd_1.clear();
  lcd_1.setCursor(0,0);
  lcd_1.print("Sistema de");
  lcd_1.setCursor(0,1);
  lcd_1.print("censado de datos");
  delay(2000);

  Wire.begin();

  Detectar_Modulos();

  delay(1000);
}

void loop() {

  float Mediciones[NT_M*3];
  float Luminosidad [NT_M];
  float Temperatura [NT_M];
  float Humedad [NT_M];

  float shuntvoltage = 0;
  float busvoltage = 0;

  float Lum_Prom = 0;
  float Temp_Prom = 0;
  float Hum_Prom = 0;

  int aux = 0;

  for(int i = 0; i<8; i = i + 1){
    for(int j = 0; j<N_Modulos_Ch[i]; j = j + 1){

      TCA9548A(i, 112);

      switch(Dir_Mod_Av_Ch[i][j]){

      case 64:

        //Medición Voltaje canal Luminosidad
        busvoltage = modulo_1.getBusVoltage_V(C_Luminosidad);
        shuntvoltage = modulo_1.getShuntVoltage_mV(C_Luminosidad);

        // Transformación a Luxes
        if(digitalRead(S_Lum) == HIGH){
          T_Luz = false;
          Tipo_Luz = "Luz Natural";
        }else{
          T_Luz= true;
```

```
          Tipo_Luz = "Luz Artificial";
        }
        Luminosidad [aux] = Conv_Lum((busvoltage + (shuntvoltage /
1000)), T_Luz);

        Mediciones [aux] = Luminosidad [aux];


        //Medición Voltaje canal Temperatura
        busvoltage = modulo_1.getBusVoltage_V(C_Temperatura);
        shuntvoltage = modulo_1.getShuntVoltage_mV(C_Temperatura);

        // Transformación a grados Celsius
        Temperatura [aux] = Conv_Temp(busvoltage + (shuntvoltage /
1000));

        Mediciones [aux+N_Modulos_Ch[i]] = Temperatura [aux];


        //Medición Voltaje canal Humedad
        busvoltage = modulo_1.getBusVoltage_V(C_Humedad);
        shuntvoltage = modulo_1.getShuntVoltage_mV(C_Humedad);

        //Transformación a % de Humedad
        Humedad [aux] = Conv_Hum(busvoltage + (shuntvoltage / 1000));

        Mediciones [j+2*N_Modulos_Ch[i]] = Humedad [aux];

        aux = aux + 1;

        break;

      case 65:

        //Medición Voltaje canal Luminosidad
        busvoltage = modulo_2.getBusVoltage_V(C_Luminosidad);
        shuntvoltage = modulo_2.getShuntVoltage_mV(C_Luminosidad);

        // Transformación a Luxes
        if(digitalRead(S_Lum) == HIGH){
          T_Luz = false;
          Tipo_Luz = "Luz Natural";
        }else{
          T_Luz= true;
          Tipo_Luz = "Luz Artificial";
        }
        Luminosidad [aux] = Conv_Lum((busvoltage + (shuntvoltage /
1000)), T_Luz);

        Mediciones [aux] = Luminosidad [j];


        //Medición Voltaje canal Temperatura
        busvoltage = modulo_2.getBusVoltage_V(C_Temperatura);
        shuntvoltage = modulo_2.getShuntVoltage_mV(C_Temperatura);

        // Transformación a grados Celsius
```

```
        Temperatura [aux] = Conv_Temp(busvoltage + (shuntvoltage /
1000));

        Mediciones [aux+N_Modulos_Ch[i]] = Temperatura [aux];


        //Medición Voltaje canal Humedad
        busvoltage = modulo_2.getBusVoltage_V(C_Humedad);
        shuntvoltage = modulo_2.getShuntVoltage_mV(C_Humedad);

        //Transformación a % de Humedad
        Humedad [aux] = Conv_Hum(busvoltage + (shuntvoltage / 1000));

        Mediciones [aux+2*N_Modulos_Ch[i]] = Humedad [aux];

        aux = aux + 1;

        break;

      case 66:

        //Medición Voltaje canal Luminosidad
        busvoltage = modulo_3.getBusVoltage_V(C_Luminosidad);
        shuntvoltage = modulo_3.getShuntVoltage_mV(C_Luminosidad);

        // Transformación a Luxes
        if(digitalRead(S_Lum) == HIGH){
          T_Luz = false;
          Tipo_Luz = "Luz Natural";
        }else{
          T_Luz= true;
          Tipo_Luz = "Luz Artificial";
        }
        Luminosidad [aux] = Conv_Lum((busvoltage + (shuntvoltage /
1000)), T_Luz);

        Mediciones [aux] = Luminosidad [aux];


        //Medición Voltaje canal Temperatura
        busvoltage = modulo_3.getBusVoltage_V(C_Temperatura);
        shuntvoltage = modulo_3.getShuntVoltage_mV(C_Temperatura);

        // Transformación a grados Celsius
        Temperatura [aux] = Conv_Temp(busvoltage + (shuntvoltage /
1000));

        Mediciones [aux+N_Modulos_Ch[i]] = Temperatura [aux];


        //Medición Voltaje canal Humedad
        busvoltage = modulo_3.getBusVoltage_V(C_Humedad);
        shuntvoltage = modulo_3.getShuntVoltage_mV(C_Humedad);


        //Transformación a % de Humedad
        Humedad [aux] = Conv_Hum(busvoltage + (shuntvoltage / 1000));
```

```
        Mediciones [aux+2*N_Modulos_Ch[i]] = Humedad [aux];

        aux = aux + 1;

        break;

      case 67:

        //Medición Voltaje canal Luminosidad
        busvoltage = modulo_4.getBusVoltage_V(C_Luminosidad);
        shuntvoltage = modulo_4.getShuntVoltage_mV(C_Luminosidad);

        // Transformación a Luxes
        if(digitalRead(S_Lum) == HIGH){
          T_Luz = false;
          Tipo_Luz = "Luz Natural";
        }else{
          T_Luz= true;
          Tipo_Luz = "Luz Artificial";
        }
        Luminosidad [aux] = Conv_Lum((busvoltage + (shuntvoltage /
1000)), T_Luz);

        Mediciones [aux] = Luminosidad [aux];


        //Medición Voltaje canal Temperatura
        busvoltage = modulo_4.getBusVoltage_V(C_Temperatura);
        shuntvoltage = modulo_4.getShuntVoltage_mV(C_Temperatura);

        // Transformación a grados Celsius
        Temperatura [aux] = Conv_Temp(busvoltage + (shuntvoltage /
1000));

        Mediciones [aux+N_Modulos_Ch[i]] = Temperatura [aux];


        //Medición Voltaje canal Humedad
        busvoltage = modulo_4.getBusVoltage_V(C_Humedad);
        shuntvoltage = modulo_4.getShuntVoltage_mV(C_Humedad);

        //Transformación a % de Humedad
        Humedad [aux] = Conv_Hum(busvoltage + (shuntvoltage / 1000));

        Mediciones [aux+2*N_Modulos_Ch[i]] = Humedad [aux];

        aux = aux + 1;

        break;

      default:
        break;
      }
    }
  }
```

```
  int L_L = sizeof(Luminosidad)/sizeof(Luminosidad[0]);
  int L_T = sizeof(Temperatura)/sizeof(Temperatura[0]);
  int L_H = sizeof(Humedad)/sizeof(Humedad[0]);

  Lum_Prom = Calcular_Prom(Luminosidad, L_L);
  Temp_Prom = Calcular_Prom(Temperatura, L_T);
  Hum_Prom = Calcular_Prom(Humedad, L_H);

  Medidas_Final[0] = Lum_Prom;
  Medidas_Final[1] = Temp_Prom;
  Medidas_Final[2] = Hum_Prom;

  MostrarDatos(Lum_Prom, Temp_Prom, Hum_Prom);
  Mostrar_LCD_2(Temperatura, Luminosidad, Humedad);

  delay(5000);

}

void Detectar_Modulos (){

  int cont = 0;
  int var = 0;
  int aux = 0;

  for(int j = 0; j<8; j = j + 1){
     TCA9548A(j, 112);
     for(int k = 0; k<4; k = k + 1){
       Wire.beginTransmission(Direcciones_Modulos[k]);
       byte resultado = Wire.endTransmission();
       if(resultado == 0){
        Dir_Mod_Av_Ch[j][aux] = Direcciones_Modulos[k];
        cont = cont + 1;
        aux = aux + 1;
        delay(500);
       }
     }
     NT_M = NT_M + cont;
     //Serial.println(NT_M);
     N_Modulos_Ch[j] = cont;
     cont = 0;
     aux = 0;
   }
}

void TCA9548A(uint8_t bus, int address){
  Wire.beginTransmission(address);  // TCA9548A address
  Wire.write(1 << bus);          // send byte to select bus
  Wire.endTransmission();
  //Serial.println(bus);
}

float Conv_Lum(float x, boolean val){
  float Lum;

  if (val){
    Lum = 0.3185*exp(4.9069*x);
```

```
    if (x<2){
    Lum = Lum + (0.05*Lum);
    }else{
    Lum = Lum + (0.3*Lum);
    }
  }else{
    Lum = 0.0545*exp(5.3572*x);
    if (x<1){
      Lum = Lum+4;
    }else if (x>1 && x<1.3){
      Lum = Lum - Lum*0.2;
    }else if (x>1.3 && x<1.5){
      Lum = Lum - Lum*0.28;
    }else if (x>1.5 && x<1.8){
      Lum = Lum - Lum*0.30;
    }else if (x>1.8 && x<2){
      Lum = Lum - Lum*0.15;
    }else if (x>2 && x<2.2){
      Lum = Lum + Lum*0.25;
    }else if (x>2.2 && x<2.5){
      Lum = Lum + Lum*0.1;
    }else{
      Lum = Lum + Lum*0.03;
    }
  }
 return Lum;
}

float Conv_Temp(float x){
  float Temp;
  Temp = 0.0024*pow(x, 6) - 0.0929*pow(x, 5) + 1.4718*pow(x, 4) -
11.889*pow(x, 3) + 50.853*pow(x, 2) - 99.354*x + 61.938;

  return Temp;
}

float Conv_Hum(float x){
  float Hum;
  Hum = 10*x;
  return Hum;
}

void MostrarDatos(float Lum, float Temp, float Hum){

  lcd.setCursor(2,0);
  lcd.print(Temp);

  lcd.setCursor(11,0);
  lcd.print(Hum);
  lcd.setCursor(15,0);
  lcd.print("%");

  lcd.setCursor(2,1);
  lcd.print(Lum);

}
```

```
float Calcular_Prom(float Medidas[], int Long){

  float Promedio = 0;

  for(int i = 0 ; i<Long ; i = i + 1){
    Promedio = Promedio + Medidas[i];
  }

  Promedio = Promedio / Long;
  return Promedio;
}

void blink() {
  if(millis() - startTime > timeThreshold){
    if(select_LCD >= NT_M){
    select_LCD = 0;
    startTime = millis();
  }else{
    select_LCD = select_LCD + 1;
    startTime = millis();
  }
  }

}

void Mostrar_LCD_2 (float Temperatura [], float Luminosidad [], float
Humedad []){
  if(select_LCD == 0){
    lcd_1.clear();
    lcd_1.setCursor(0, 0);
    lcd_1.print("Num Tot Mod: ");
    lcd_1.setCursor(14, 0);
    lcd_1.print(NT_M);
    lcd_1.setCursor(0, 1);
    lcd_1.print(Tipo_Luz);

  }else{
    lcd_1.clear();
    lcd_1.setCursor(0,0);
    lcd_1.print("T");
    lcd_1.setCursor(1,0);
    lcd_1.print(select_LCD);
    lcd_1.setCursor(2,0);
    lcd_1.print(":");
    lcd_1.setCursor(9,0);
    lcd_1.print("H");
    lcd_1.setCursor(10,0);
    lcd_1.print(select_LCD);
    lcd_1.setCursor(11,0);
    lcd_1.print(":");
    lcd_1.setCursor(0,1);
    lcd_1.print("L");
    lcd_1.setCursor(1,1);
    lcd_1.print(select_LCD);
    lcd_1.setCursor(2,1);
    lcd_1.print(":");
    lcd_1.setCursor(3,0);
```

```
    lcd_1.print(Temperatura[select_LCD - 1]);
    lcd_1.setCursor(12,0);
    lcd_1.print(Humedad[select_LCD - 1]);
    lcd_1.setCursor(3,1);
    lcd_1.print(Luminosidad[select_LCD - 1]);
  }

}
```