UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

SEDE VIÑA DEL MAR - JOSÉ MIGUEL CARRERA

ESTUDIO DE APLICACIÓN DEL PUERTO I2C POR MEDIO DE ARDUINO

Trabajo de Titulación para optar al Título de
TÉCNICO UNIVERSITARIO EN ELECTRÓNICA

Alumno:
Franco Antonio Pizarro Helo

Profesor Guía:
Ing. José Llantén Álvarez

Profesor Correferente:
Ing. Sergio Riquelme Bravo

2022

RESUMEN

KEYWORDS: SENSORES, ARDUINO, COMUNICACIÓN, PROGRAMACIÓN.

En el presente informe de proyecto de título, se expondrá la comunicación I2C en forma detallada de su funcionamiento, además, se hará una aplicación simulada en Proteus, para entender el protocolo y la programación, gracias a la plataforma Arduino. Cabe destacar que puede servir para variados sistemas periféricos que no necesiten tener rápidas respuestas.

Se hará un estudio de costo de la mano de obra, de los componentes y materiales, exponiendo finalmente el costo total de la aplicación que se realizará, exponiendo los problemas que se obtuvieron en la programación e implementación, además, se plantearán los objetivos generales y específicos de este proyecto.

ÍNDICE

ÍNDICE FIGURAS

TABLAS

INTRODUCCIÓN

Circuito Inter Integrado (inter integrated circuit o I2C), también conocido como TWI (interfaz de dos hilos), es muy utilizado en las industrias para comunicar microcontroladores y sus periféricos en sistemas integrados, debido que ocupa solo dos cables de forma sincrónica y en serie, que conectan a sus esclavos (dispositivo que puede ser receptor o transmisor) con los maestros (dispositivo que controla todo el programa).

Normalmente es ocupado para la transmisión de datos de control y configuración, por ejemplo, para relojes a tiempo real, control de volumen, conversor de señal analógica-digital o digital-analógica con baja tasa de frecuencia de muestreo, pequeños espacios de memoria o conmutadores bidireccionales, multiplexores. También se puede encontrar componentes y módulos que soportan dicha comunicación, por ejemplo, pantallas OLED basadas en el controlador SSD1306, sensor de temperatura y humedad HDC 1080, convertidor analógico digital ADS1115, MLX90614 (termómetro infrarrojo), lector RFID RC522, reloj RTC como el DS3231 o DS1307.

El protocolo de comunicación I2C con Arduino es ampliamente utilizado por una gran cantidad de sensores y actuadores, esto es por la llegada del internet de las cosas o loT (internet of things) y por el aprovechamiento de los pines, haciendo que este protocolo vuelva a ser usado, su creación fue en 1982.

Esta investigación y aplicación quiere llegar a las futuras generaciones, para que sea un manual de alumnos de la carrera electrónica, para cuando quieran estudiar esta comunicación I2C no empiecen de 0 y tengan una investigación ya hecha.

CAPÍTULO 1: COMUNICACIÓN I2C

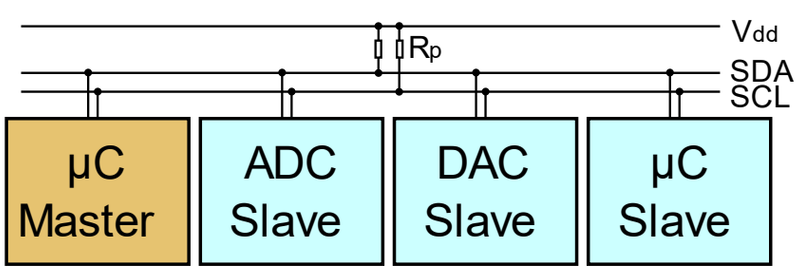
1 COMUNICACIÓN I2C

En este capítulo se aprenderá el protocolo de comunicación I2C, sobre sus conexiones y cómo se logra comunicar con sus dispositivos, además una breve historia. Se explicará cómo se puede ocupar la comunicación I2C por medio de Arduino para que finalmente se pueda entender la aplicación a desarrollar.

Por último, se dará a conocer qué componentes se ocuparán, para poder revelar los objetivos que se tienen en mente.

1.1 CONEXIÓN AL I2C

Como se habló anteriormente la comunicación I2C tiene dos líneas para transmitir información, como se puede apreciar en la figura 1-1, SDA es la línea donde los dispositivos podrán transferir los datos, y SCL es la línea que sincroniza el sistema por medio de pulsos de reloj. Existen 2 líneas de alimentación que ocupan todos los dispositivos, Vcc (fuente de voltaje) y la de referencia “Vss” (masa o GND). Las líneas SDA y SDL son de drenaje abierto, por lo tanto, necesitan resistencia de pull-up para asegurar un nivel alto cuando no hay dispositivos conectados al bus, con esto se podrá conectar en paralelo múltiples entradas y salidas. Los valores más comunes de los resistores de polarización son entre 1.8KΩ Y 10KΩ. Se pueden conectar hartos dispositivos, pero hay que tener en cuenta, que la capacidad máxima sumada de todos los dispositivos no supere los 400pF. La velocidad estándar es de 100Kbps.



Fuente: https://aprendiendoarduino.wordpress.com/2017/07/09/i2c/

Figura 1-1. Diagrama conexiones.

1.1.1 Historia

El puerto I2C fue introducido por una empresa de tecnología más grande e importante del mundo, Philips Semiconductors (hoy NXP semiconductores), en el año 1982 queriendo comunicar internamente circuitos integrados con un simple bus bidireccional de dos hilos de una forma eficiente en una CPU y sus dispositivos periféricos. Creando la familia de microcontroladores MAB8400 con un controlador de bus I2C. Sin embargo, con el paso del tiempo otros fabricantes comenzaron a adoptarlo hasta convertirse en el estándar del mercado mundial que es hoy.

1.2 COMUNICACIÓN

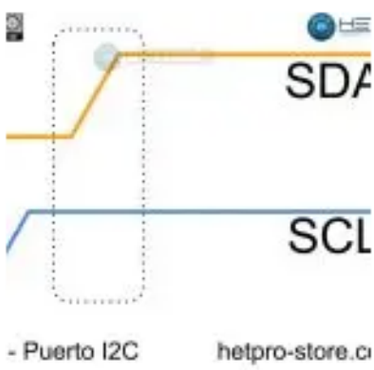
Existen dos términos importantes en la comunicación I2C: maestro y esclavo. Los maestros son los únicos que pueden iniciar una transferencia direccionando la información, además genera la señal de reloj, que es esencial para el orden de comunicación de dispositivos. Se pueden conectar varios maestros a un mismo bus denominado bus multimaestro, pero solo uno puede funcionar a la vez. Los esclavos, por su parte, no tienen la capacidad de generar pulsaciones de reloj, pueden actuar de dos formas: esclavo – transmisor o esclavo – receptor.

La comunicación puede dar inicio cuando el bus está disponible, el maestro pone en estado bajo la SDA, pero dejando en alto SCL, como se aprecia en la figura 1-2, estableciendo la condición de inicio (start). El primer byte que se transmite después de la condición inicial tiene siete bits de direccionamiento del dispositivo que se quiere comunicar, el octavo bit de menor peso corresponde con la operación que se quiere realizar en el esclavo, bit de lectura/escritura(R/W). Si el bit de R/W es alto (lectura), el dispositivo maestro genera pulsos de reloj para que el dispositivo esclavo pueda enviar los datos. Ahora si el bit de R/W es bajo (escritura), el dispositivo maestro envía datos al esclavo. Una vez que el maestro ha enviado la dirección del dispositivo y se haya conectado con el esclavo que quiere conectarse, puede enviar byte de datos. Cada byte leído/escrito por el maestro debe ser obligatoriamente reconocido por un bite de ACK, que confirma la recepción de datos por el dispositivo maestro o esclavo, dependiendo de quien está recibiendo, si se genera una confirmación de no recepción se mandará el bit de NACK. La comunicación finaliza cuando el maestro transmite la condición de parada (stop), se genera cuando se pone en estado alto SCL y luego SDA, ver figura 1-3. Si se quiere seguir transmitiendo, el dispositivo maestro genera otra condición de inicio en lugar de la parada, esta nueva condición de inicio se llama “inicio reiterado”.

Fuente: https://hetpro-store.com/TUTORIALES/i2c/

Figura 1-2. Condición de inicio.

En la figura 1-3 se expone la condición de parada.



Fuente: https://hetpro-store.com/TUTORIALES/i2c/

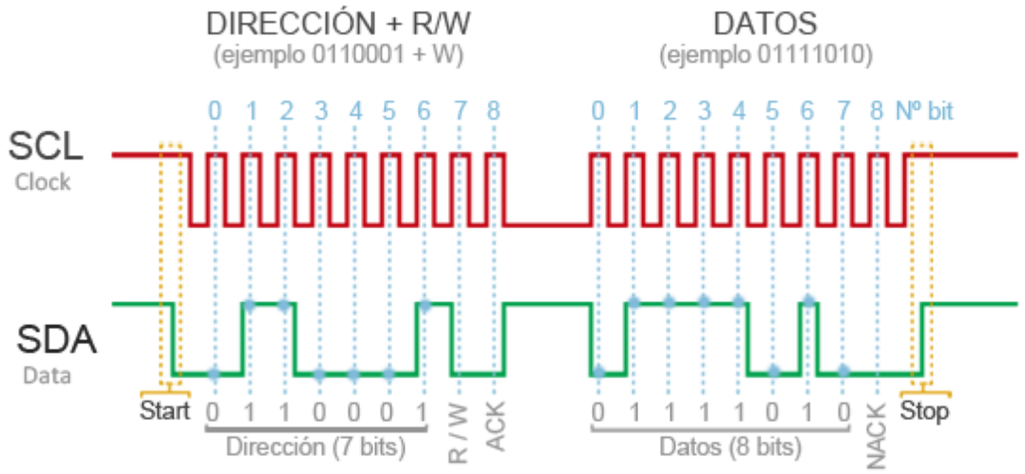
Figura 1-3. Condición de parada.

1.2.1 Escritura de dato

Una vez que el maestro encontró al esclavo con quien esperaba comunicarse, pueden enviar byte de datos, el dispositivo maestro puede seguir mandando bytes al esclavo, que normalmente serán puestos en registros con direcciones sucesivas, ya que el esclavo incrementa automáticamente la dirección del registro interno después de recibir cada byte. Se expone la figura 1-4 para comprender el formato de dirección y dato.

Las transacciones en el bus I2C tienen este formato guiándose por la figura 1-4:

| start | 0 1 1 0 0 0 1 R/W | ACK | 0 1 1 1 1 0 1 0 | NACK | stop |



Fuente: https://aprendiendoarduino.wordpress.com/category/i2c/

Figura 1-4. Formato de dirección y dato.

1.2.2 Direccionamiento

Normalmente los bits de direccionamiento son de 7 bits, existen casos que tienen dispositivos de 10 bits, aunque rara vez. Cuando se ocupa una dirección de 7 bits implica que se puede conectar 128 dispositivos, desde 0 hasta 127. Cabe recordar que serán 8 bits por el bit de lectura/escritura. El número de dispositivos que se puede conectar al bus pueden ser los 128, pero se debe observar que la capacitancia máxima sumada entre todos los dispositivos no supere a los 400pF.

1.3 LIBRERÍA Y FUNCIONES I2C EN ARDUINO

Arduino dispone de plataforma I2C por hardware vinculado físicamente a ciertos pines, para poder utilizar la comunicación se utilizará la librería “Wire.h” y también la librería “LiquidCrystal_I2C.h”, que permitirá trabajar fácilmente una pantalla LCD con los dos pines para la comunicación. La librería Wire es la herramienta indispensable para interactuar con el bus I2C con Arduino, de forma simple e intuitiva a todos los dispositivos conectados. Directamente se escribe la clase (Wire) y luego se llama a la función correspondiente que se ocupará. Algunas funciones de la librería Wire que se utilizará para programar Arduino son:

- begin() = Esta función sirve para iniciar la librería Wire y hacer que la placa se una al bus I2C. Se puede ocupar de dos formas diferentes dependiendo del número de parámetros. El primero no admite parámetros, vinculando la placa al I2C para actuar como un dispositivo maestro. El segundo admite un único parámetro que es la dirección, especificando que el dispositivo es un esclavo.
- requestFrom() = Con esta función el maestro le puede solicitar datos al esclavo. Se puede ocupar de dos formas. La primera admite dos argumentos: la dirección y la cantidad, esta función solicita al esclavo una cantidad especifica de bytes y libera el bus, los bytes solicitados son leídos utilizando la función "Wire.read()". La segunda admite tres parámetros: la dirección, cantidad y parada, si parada es true (verdadero) al terminar se liberar el bus, en cambio sí es false (falso) no se enviará condición de parada.
- beginTransmission() = Comienza la transmisión con el esclavo con la dirección indicada.
- endTransmission() = Finaliza la transmisión que comenzó con el esclavo y transmite los bytes que estaban en espera.
- write() = El maestro escribe datos desde un esclavo o al revés y pone en cola la transmisión.
- read() = Esta función retorna un byte leído. En un maestro esta función debe ser ejecutada después de una llamada a Wire.requestFrom(), en un esclavo debe ser ejecutada dentro de la función de recepción.

1.4 APLICACIÓN A DESARROLLAR

Ahora que se tienen los conocimientos previos para utilizar la comunicación I2C, se podrá entender la aplicación que se va a realizar, consiste en 4 sensores que estarán sometidos a diferentes temperaturas para poder verificar que realmente funciona, la lectura de estos se verá reflejada en el Display LCD, esto será controlado con solo dos cables que corresponde al bus I2C.

Esta aplicación puede ser utilizada en un invernadero industrial, donde dichos sensores estén en diferentes sectores para conocer cuál es la temperatura media, además, se tendrá mayor control en el sector que se debe intervenir para subir o bajar la temperatura. Esta aplicación puede ser muy útil para el agrónomo, debido a que existen algunas plantas que necesitan una temperatura específica y teniendo una gran área de cultivo le será muy difícil controlar la temperatura. También puede ser ocupado en empresas de alimentos o bebidas, donde se necesita un monitoreo constante de temperatura, si no hay monitoreo podría ser fatal para la producción o para el consumidor.

1.5 VENTAJAS Y DESVENTAJAS DE LA COMUNICACIÓN

Sus ventajas son:

- Sencillez al solo ocupar dos líneas.
- Bajo costo.
- Cuenta con mecanismo para saber si la señal ha llegado frente a otros protocolos.
- Es muy útil para muchas aplicaciones, ya que la velocidad de transferencia es mucho mayor de lo que el caso requiere.

Las desventajas son:

- Velocidad de transmisión baja, en aplicaciones que se necesita una actualización casi inmediata.
- Es una comunicación half dúplex, es decir, solo puede transmitir en una dirección a la vez bidireccionalmente, pero no de forma simultánea.
- Susceptible a las interferencias.

1.6 COMPONENTES A UTILIZAR

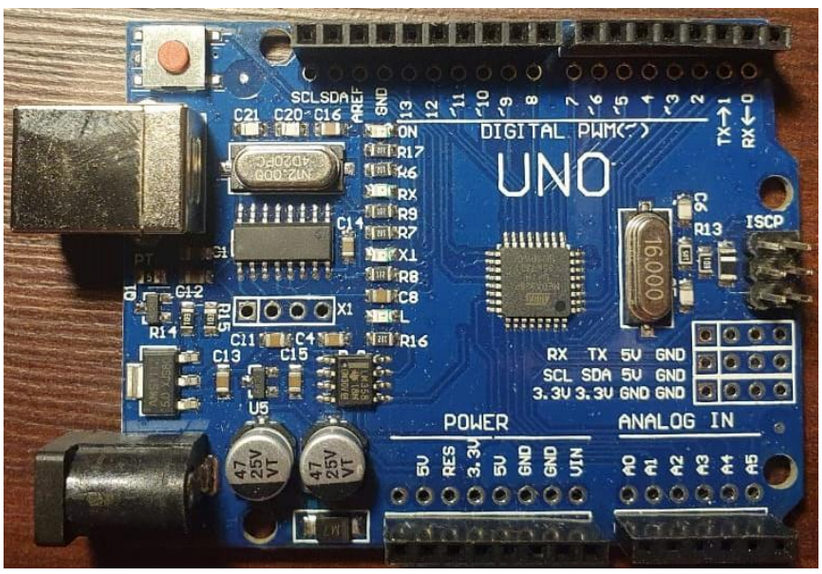
A continuación, se presentarán los materiales que se ocuparán para realizar esta aplicación, además de una breve descripción de su funcionamiento y sus características técnicas.

1.6.1 Arduino UNO

Dispositivo electrónico reconocido a nivel mundial por su hardware y software flexibles al momento de programar, es una placa de microcontrolador, es capaz de automatizar, monitorear y controlar procesos industriales. Puede ser alimentado por cable USB o batería externa de 7 a 20 voltios, aunque normalmente se usa a través de una computadora para poder grabar el programa. Arduino esta publicado como herramienta de código abierto, donde dispone extensiones por programas experimentados. La plataforma Arduino se programa utilizando una mezcla del lenguaje C y C++.

Posee un microcontrolador ATmega328, 14 pines de entradas/salidas digitales, de los cuales 6 proporcionan salidas PWM, 6 entradas analógicas, frecuencia del reloj de 16Mhz. Su voltaje de operación es de 5V, corriente máxima que soporta individualmente por pin entrada/salida 40mA, total de corriente alimentado por USB de 500mA.

En este modelo los pines que se ocupan para la comunicación I2C se encuentran al lado izquierdo de “AREF”, ver figura 1-5. Algunos modelos disponen de dicha plataforma en los pines A4 y A5. Este componente hará la función de maestro.



Fuente: Foto sacada por el alumno.

Figura 1-5. Arduino uno.

1.6.2 Display LCD

Un LCD, Liquid Crystal Display o pantalla de cristal líquido, es un dispositivo muy utilizado en programación, por su versatilidad de desplegar información, se compone de dos placas de vidrio paralelas con una separación de unos micrones, se logra visualizar información o contenido variado, gracias a estas placas que tienen unos electrodos especiales que definen con su forma, los símbolos, caracteres, números, etc. El LCD tiene que estar programado por medio de un microcontrolador, para que éste pueda dirigir todo su funcionamiento.

El LCD que se presenta en la figura 1-6, tiene 3 pines de control que son, R/W(Read/Write), EN, RS. El primero permite leer datos desde la pantalla si está en estado alto o escribir datos si se encuentra en estado bajo. El segundo cumple la función de habilitar la pantalla si está en estado alto o deshabilitar la pantalla si se encuentra en estado bajo del LCD. El tercero permitirá el

registro de datos si está en estado alto, además, consta de 8 pines bidireccionales de bus de datos, comenzando desde DB0 al DB7. Cabe mencionar que faltan los pines de alimentación, donde VDD es conectado a 5V, el pin VSS a tierra GND y VEE se puede ajustar el contraste/brillo del LCD por medio de una variación entre 0-5V.

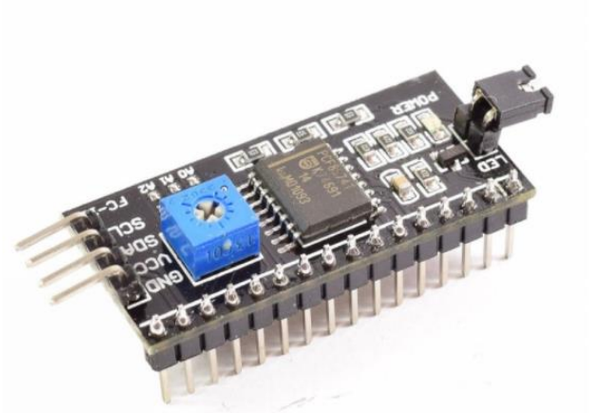
Fuente: https://articulo.mercadolibre.cl/MLC-552877770-lcd-display-16x2-_JM

Figura 1-6. Pantalla LCD.

1.6.3 Módulo adaptador I2C

Debido que el LCD trabaja con muchos pines y el puerto I2C utiliza dos cables, se ocupará el módulo de interfaz serial que permitirá visualizar los datos a través de dos hilos ahorrando un número significativo de pines. Finalmente son cuatro cables para conectar, dos de datos, uno de VDD y el último de tierra o GND. Esta placa tiene un chip I2C PCF8574 que convierte los datos en paralelo a datos serie para la pantalla LCD, con alimentación 5V, para conectar solo hay que soldar las entradas con las del LCD o implementar en protoboard y poner los mismos pines a las líneas.

El módulo que se expone en la figura 1-7, puede cambiar las direcciones con los pines A0, A1 y A2, esto es bastante útil, ya que, si se obtiene un dispositivo con la misma dirección, se podrá cambiar a una dirección que no se esté ocupando, además, comienza desde 0x20 hasta 0x27 (estos valores pueden variar dependiendo del modelo), que es un total de 8 Display LCD, por lo tanto, se pueden conectar 8 LCD en el mismo bus I2C, teniendo cada módulo una dirección diferente.



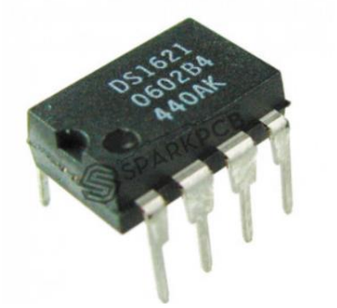
Fuente: https://naylampmechatronics.com/blog/35_tutorial-lcd-con-i2c-controla-un-lcd-con-solo-dos-pines.html

Figura 1-7. Módulo adaptador.

1.6.4 Sensor de temperatura digital DS1621

Sensor que convierte la temperatura en una palabra digital, su rango de medición es de -55 °C hasta 125 °C en incrementos de 0.5 °C. Es equivalente en grados Fahrenheit entre -67 °F a 257 °F en incrementos de 0.9 °F. Proporciona lecturas de temperatura de 9 bits, por lo tanto, la transferencia será de 2 bytes, que indicará la temperatura del dispositivo.

El sensor DS1621 consta de 8 pines, de los cuales 2 son de alimentación: VDD y GND, se puede alimentar desde 2,7V hasta 5,5V. Otros 2 pines para la comunicación I2C: SDA, SCL. Tres pines que servirán para configurar la dirección del dispositivo: A0, A1, A2, si un pin está conectado a VCC éste tomará el valor de “1”, en cambio, sí está sin conectar o conectado a GND tomará el valor de “0”. La salida de alarma térmica, Tour, se activa cuando supera la temperatura definida por el usuario, los ajustes de éste se almacenan en una memoria no volátil. Figura 1-8 se aprecia el sensor DS1621.



https://www.sparkpcb.com/ic-transistors/ds1621-digital-thermometer-and-thermostat.html

Figura 1-8. sensor de temperatura DS1621

1.7 OBJETIVOS DEL PROYECTO

A continuación, se describirán los objetivos generales y específicos de este trabajo, describiendo de forma resumida los puntos que se quieren lograr.

1.7.1 Objetivo general

Diseñar e implementar la aplicación, para entender de mejor manera del puerto I2C.

1.7.2 Objetivos específicos

- Aprender el protocolo I2C.
- Exponer las funciones que se ocupan para la plataforma Arduino
- Programar un código, para realizar conexión I2C en Arduino.
- Realizar un estudio de costos de la aplicación.
- Realizar pruebas.

CAPÍTULO 2: DESARROLLO Y PROGRAMCIÓN

2 DESARROLLO Y PROGRAMACIÓN

En este capítulo se explicará de forma detallada el desarrollo de la aplicación, sus conexiones, algunos problemas que se obtuvieron al ir avanzando y sus soluciones, se proporcionará mayor cantidad de información respecto a los componentes planteados en la parte final del Capítulo 1. Se mostrará la simulación completa de la aplicación y su programación, la cual se explicará brevemente.

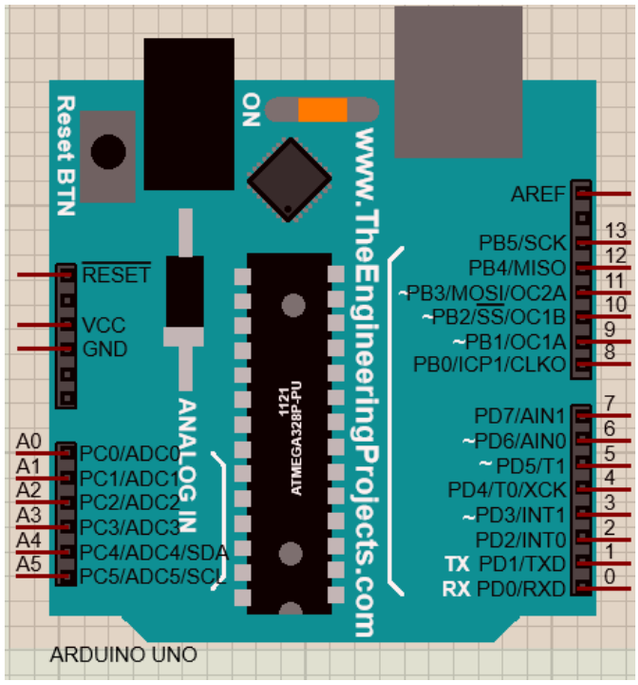
2.1 ELECCIÓN Y FUNCIÓN DE LOS COMPONENTES

Para el avance de la aplicación se debe determinar la funcionalidad de cada dispositivo, justificar la razón por la cual se utilizaron los componentes descritos en el capítulo 1.6 y sus debidas conexiones para así avanzar en la programación.

2.1.1 Controlador

En esta oportunidad se utilizó un Arduino UNO, aunque no es la única solución para realizar esta aplicación, puede servir otros controladores, pero se debe tener presente que opere en la plataforma I2C. Se utilizará esta placa de desarrollo debido a que dispone de plataforma I2C por hardware vinculado, es libre y extensible, lo que significa, que cualquiera puede ampliar y mejorar el diseño de hardware de las placas como el entorno de desarrollo. Esto permite que se puedan utilizar librerías de software de terceros, que puedan adaptase mejor a las necesidades, gracias a sus cualidades existe una comunidad extensa trabajando con esta plataforma que pueden hacer esto posible, generando una cantidad de documentación bastante amplia, por ende, abarca casi cualquier necesidad. Su programación es multiplataforma, es decir, se puede ocupar en sistemas operativos Windows, Mac OS y Linux. Es reutilizable, ya que, al terminar el proyecto es fácil de desmontar los componentes externos de la placa y se puede subir la programación sobrescribiendo el programa anterior.

El Arduino que se utilizará, gracias a la plataforma Proteus, es el que se encuentra en la figura 2-1, los pines que se encuentran en A4 y A5, serán para la comunicación I2C, SDA y SDC respectivamente. Dicho dispositivo será el maestro como se mencionó en el subcapítulo 1.6.1.



Fuente: Foto sacada por el Alumno de Proteus.

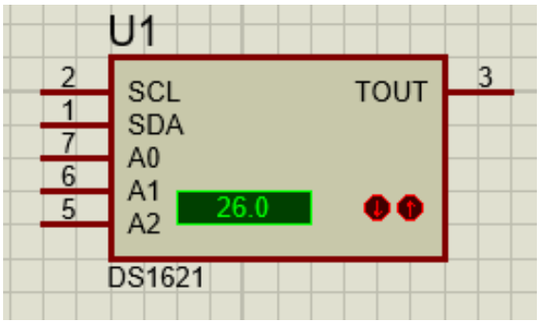
Figura 2-1. Arduino UNO, Proteus.

2.1.2 Temperatura

El componente que se ocupará para medir la temperatura es el ya mencionado en el subcapítulo 1.6.4, el sensor DS1621. Se eligió este dispositivo ya que es un sensor digital que envía el valor de la temperatura en una palabra digital reconocible para el Arduino, no es necesario hacer ninguna conversión por el usuario, en cambio, si se utiliza un sensor analógico debe realizarse, además, viene con protocolo I2C, esto quiere decir que no se ocupará un convertidor I2C ahorrándose el costo de éste. Fue complicado elegir este componente, ya que, al principio se iba a ocupar el sensor DS18B20, se descartó por el hecho de ocupar el protocolo 1-wire, es decir que tiene solo un pin de dato, para hacer funcionar dicho dispositivo se tenía que utilizar otra librería adicional, programar la comunicación 1-wire en la tarjeta lo que se haría más extenso y complicado, además, se tenía que ocupar un módulo de 1-wire a I2C.

El sensor que se encuentra en la plataforma Proteus es el que se muestra en la figura 2-2, con la posibilidad de poder ir variando la temperatura con las flechas rojas, obteniéndose un realismo de temperatura variando. Debido a que se ocupará 4 sensores y no pueden tener la misma dirección, se cambiará con los pines A2, A1, A0, empezando con el más significativo. El primer sensor tendrá estos pines sin conectar, ósea estará en estado “000”, el segundo sensor tendrá el pin A0 en estado alto obteniendo “001”, el tercer sensor tendrá el pin A1 en estado alto

resultando “010”, el cuarto sensor tendrá dos piense en estado alto A1 y A0 obteniendo “011” como se presenta en la figura 2-5.



Fuente: Foto sacada por el Alumno de Proteus.

Figura 2-2. sensor DS1621, Proteus.

2.1.2.1 Comandos del sensor

Se observa en la documentación del dispositivo que se deben ocupar algunos comandos que se expondrán en la tabla 2-1. Los dos primeros comandos son los que se utilizarán, ya que se podrá programar el DS1621 para que realice de forma continua lecturas de temperatura.

Tabla 2-1. Comando de temperatura

Descripción	Comandos
Leer temperatura convertida	AAh
Iniciar la conversión de la temperatura	EEh
Leer pendiente	A9h
Detiene la conversión de temperatura	22h

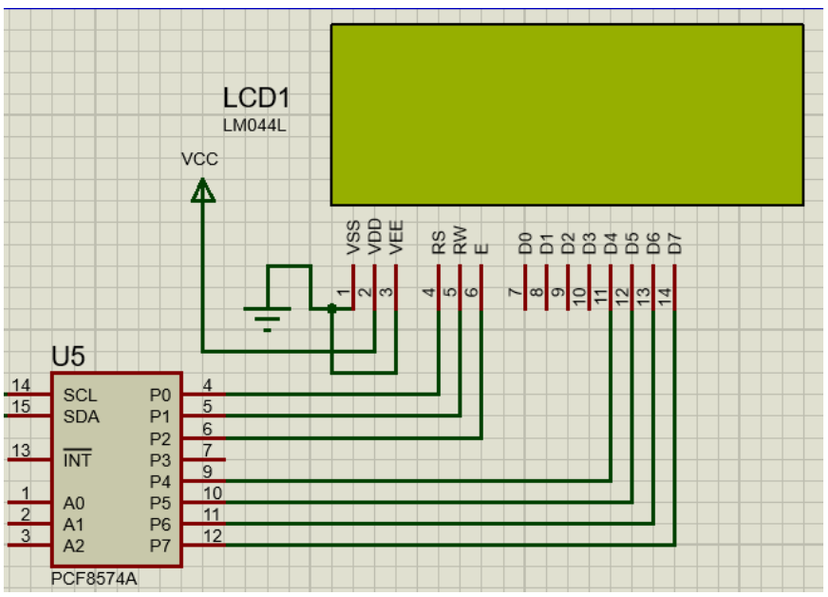
Fuente: Tabla creada por el Alumno.

2.1.3 Pantalla

Debido a que las mediciones tomadas por el sensor serán enviadas por medio de un Arduino, se quiere lograr que el usuario sea capaz de ver la temperatura que se está midiendo en los diferentes sensores de cada sector, para esto se ocupará una pantalla LCD Display 20x4, 20 caracteres y 4 filas. Se utilizó este tamaño para que se pueda ver al mismo tiempo las 4 temperaturas de cada sector, por lo tanto, en cada fila tendrá un sector “x” y su debida temperatura en Celsius. Las características de este dispositivo fueron explicadas en el subcapítulo 1.6.2 con gran detalle, el LCD que se utilizará es LM044L y las conexiones que se harán se encuentra en la figura 2-3.

2.1.3.1 Conexión al módulo adaptador I2C

Como se explicó anteriormente el LCD tiene demasiados pines para ser conectado directamente al Arduino, se tiene que ocupar un módulo adaptador para I2C, para controlar el LCD con dos pines, por lo tanto, se utilizará el PCF8574A que se encuentra en Proteus. Las conexiones que se hará en la simulación, al LCD y al módulo serán conectadas como muestra la figura 2-3.



Fuente: Foto sacada por el Alumno de Proteus.

Figura 2-3. Conexión al LCD y módulo adaptador I2C, Proteus.

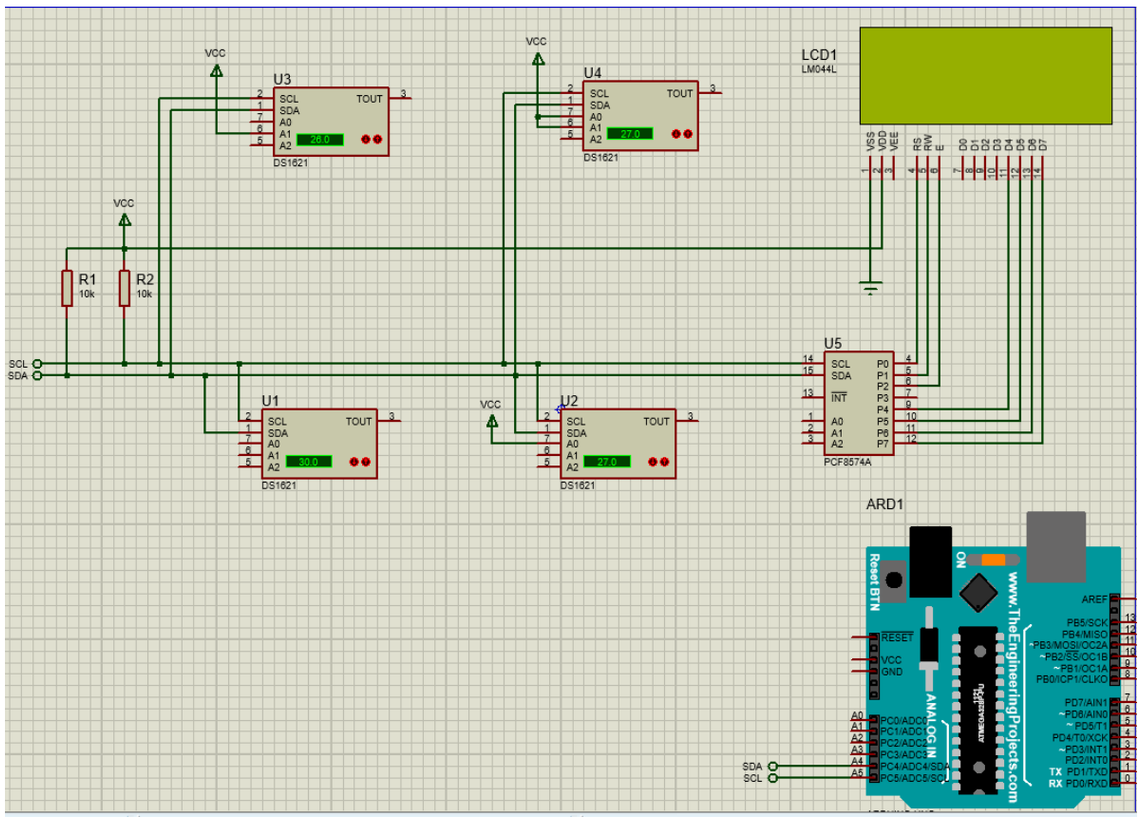
Normalmente cuando se implementa en Protoboard, el adaptador va soldado a los pines del LCD, ahorrando espacio en éste, como se presenta en la figura 2-4.

Fuente: https://naylampmechatronics.com/blog/35_tutorial-lcd-con-i2c-controla-un-lcd-con-solo-dos-pines.html

Figura 2-4. Conexión al LCD y adaptador modulador I2C, soldados.

2.2 CIRCUITO GENERAL EN PROTEUS

En esta sección se mostrará la simulación en Proteus con los componentes y todas sus conexiones para la aplicación. Los valores de las resistencias pull-up que se utilizaron son 10KΩ, como se presenta en la figura 2-5.



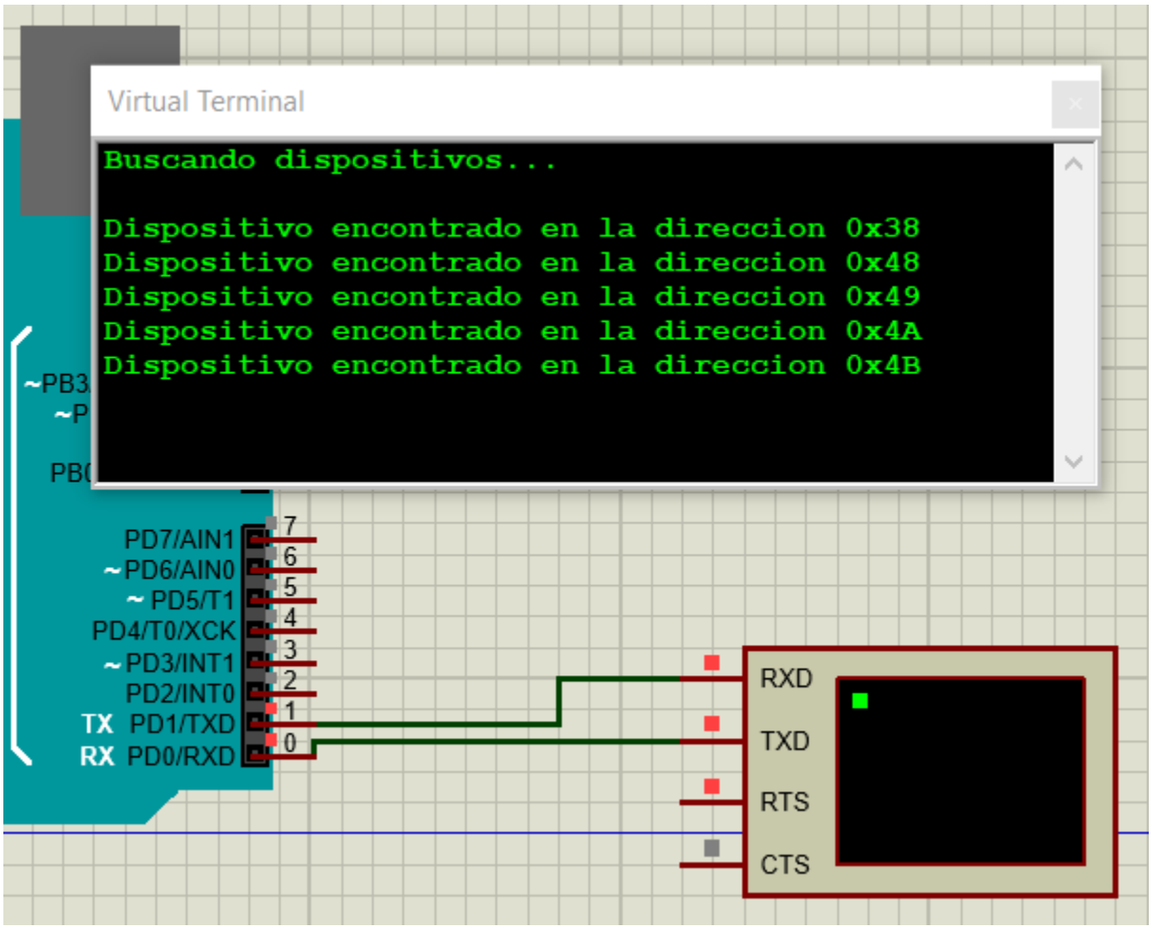
Fuente: Foto sacada por el Alumno de Proteus.

Figura 2-5. Circuito completo

2.3 DIRECCIÓN DE LOS COMPONENTES

Como indica la bibliografía, todos los dispositivos que se comunican a través del protocolo I2C poseen una dirección hexadecimal única, cada mensaje y orden que transmite al bus lleva anexa esta dirección, indicando cuál de los dispositivos será el receptor. Lo normal es leer el datasheet y comprobar la información técnica del fabricante del componente, aquí se puede encontrar la dirección por defecto. Hay algunos dispositivos que no tienen documentación, para solucionar tal problema existe un programa para Arduino que informará las direcciones de los dispositivos que se tienen conectados en el bus. Se utilizará un Sketch llamado “I2C Scanner” encontrado en la página oficial de Arduino, esta prueba con todas las direcciones posibles y en caso de encontrar algún dispositivo informa la dirección de los componentes conectados. Cabe destacar que se tiene que conectar todos los dispositivos que se ocuparán al mismo bus, se subirá este Sketch a la simulación que se encuentra en la figura 2-5, agregando el terminal virtual a los terminales del Arduino como muestra la figura 2-6, para poder observar los resultados.

El enlace donde se encontró el sketch es https://playground.arduino.cc/Main/I2cScanner/, se modificó un poco el programa cambiando las palabras en español.



Fuente: Foto sacada por el Alumno de Proteus.

Figura 2-6 Resultado de las direcciones.

Lo que se pudo obtener en la figura 2-6 son los direccionamientos de los dispositivos, en la tabla 2-2 se podrán apreciar los direccionamientos de cada dispositivo.

Tabla 2-2. Dirección de cada dispositivo.

Dispositivos	Dirección
Módulo adaptador I2C	0x38
Sensor U1	0x48
Sensor U2	0x49
Sensor U3	0x4A
Sensor U4	0x4B

Fuente: Tabla creada por el Alumno.

2.3.1 Sketch utilizado

En esta sección se presentará el programa que se utilizó para obtener las direcciones. Como se expuso, el sketch fue obtenido en la página Arduino, se demostrará la programación, debido a una posibilidad de que podría borrarse la página o no encontrar la programación, lo que se muestra en figura 2-7.

Se ocupa un "wire.begin" para inicializar la librería Wire y un "serial.begin" para indicar al Arduino que inicie la comunicación al dispositivo que está conectado a los pines RX Y TX que sería el virtual terminal de la figura 2-6, con una velocidad de comunicación serial de 9600 bits por segundo. Se declaran las variables que se utilizarán, luego se utiliza una estructura de control "for" para que se repita una cantidad de veces determina, cuya finalidad es para probar con todas las direcciones posibles, "wire.beginTransmission" comienza la transmisión con el esclavo y "wire.endTransmission" finaliza la transmisión con el esclavo entregando a la variable "variableDeRespuesta" un bits de confirmación que vendría siendo el ACK, si tal bit es "0" indica que se encontró un dispositivo, si el bit es 4 representa que hay un error en la dirección del esclavo.

```
#include <Wire.h>

void setup() {
  Wire.begin();
  Serial.begin(9600);
}

void loop() {
  byte codigoDeRespuesta, direccion;
  int dispositivosEncontrados = 0;
  Serial.println("\n\nBuscando dispositivos...");

  for (direccion = 1; direccion < 127; direccion++ ) {
    Wire.beginTransmission(direccion);
    codigoDeRespuesta = Wire.endTransmission();

    if (codigoDeRespuesta == 0) {
      Serial.print("\r");
      Serial.print("Dispositivo encontrado en la direccion 0x");
      if (direccion < 16)
        Serial.print("0");
      Serial.print(direccion, HEX);

      dispositivosEncontrados++;
    }else if (codigoDeRespuesta == 4) {
      Serial.print("\r");
      Serial.print("Error desconocido en la direccion 0x");
      if (direccion < 16)
        Serial.print("0");
      Serial.println(direccion, HEX);
    }
  }
  if (dispositivosEncontrados == 0) Serial.println("No se encontró ningún dispositivo"
  delay(1000);
}
```

Fuente: Foto sacada por el Alumno de Arduino.

Figura 2-7. Sketch para encontrar los dispositivos.

2.4 PROGRAMACIÓN

En esta sección se revelará la programación que se utilizó en la aplicación. Dado que en el subcapítulo 1.5 se expuso con detalle cada función, solo se mostrará la aplicación explicando a grandes rasgos el funcionamiento, será por partes ya que el programa es bastante extenso. Se empezará por las librerías y el void setup, cabe destacar que liquidCrystal_I2C es para definir la dirección del módulo adaptador y el tamaño del Display LCD, que se está utilizando, luego declaramos 8 (ocho) variables, 4 (cuatro) para almacenar el valor de la temperatura de cada sensor y los restantes son las direcciones de los dispositivos, como muestra la figura 2-8.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x38,20,4);
int U1=0x48, U2=0x49, U3=0x4A, U4=0x4B;
int tempS1, tempS2, tempS3, tempS4;
void setup() {
  Wire.begin();         //activo lógica de control al bus I2C como master
  lcd.begin(20,4);
  lcd.init();          // Inicializa el LCD
}
```

Fuente: Foto sacada por el Alumno de Arduino.

Figura 2-8. Programación parte 1.

En la segunda parte viene el void loop donde contiene todo el programa, seguido de un "wire.beginTranmission" con esta función comienza la transmisión con el esclavo seleccionado, se escribirá con la función "wire.write" el comando de conversión de temperatura "EE" a cada sensor, luego vendría "wire.endTransmission" que terminaría la comunicación con cada sensor. Lo que se muestra en la figura 2-9.

```
void loop() {
  lcd.print("Espera, tomando");
  lcd.setCursor(0,1);
  lcd.print("mediciones");
  //  REGISTRO DE COMIENZO DE CONVERSIÓN
  Wire.beginTransmission(U1); //Condición de INICIO
  Wire.write(0xEE); //Se envió comando para que dé comienzo la conversión
  Wire.endTransmission(); //Condición de STOP
  Wire.beginTransmission(U2);
  Wire.write(0xEE); //Se envió comando para que dé comienzo la conversión
  Wire.endTransmission();
  Wire.beginTransmission(U3);
  Wire.write(0xEE); //Se envió comando para que dé comienzo la conversión
  Wire.endTransmission();
  Wire.beginTransmission(U4);
  Wire.write(0xEE); //Se envió comando para que dé comienzo la conversión
  Wire.endTransmission();
```

Fuente: Foto sacada por el Alumno de Arduino.

Figura 2-9. Programación parte 2.

Para la tercera etapa del programa, se utilizó un "while" para que sea un ciclo infinito, es decir, que termina de leer el programa que se encuentra en el "while" espera un 1s y comienza nuevamente, esto es para que la medición se actualice por el tiempo ya indicado en el Display

LCD. Se utiliza el comando de “AA” para leer la última conversión realizada por el sensor y luego se le pide un byte al esclavo seleccionado para enviar información al maestro, luego esta información de temperatura queda guardada en la variable ya declarada. Para que el programa muestre las mediciones en negativo se utiliza un “if” donde la condición tiene que ser mayor que 128, al entrar a esta estructura de control tiene que restar 256, como muestra la figura 2-10, sin esta resta la temperatura al llegar a -1 °C marcaria 255 °C, esto se repite a cada sensor que se utilizará.

```
while (true){
delay(1000);
Wire.beginTransmission(U1) ;     //comenzar a comunicar con sensor U1
Wire.write(0xAA);                //Se envió comando para leer la última conversión
Wire.endTransmission();          //finalizar comunicación
Wire.requestFrom(U1, 1);    // Solicitar 1 byte al sensor U1
tempS1=Wire.read();
//sí es mayor que 128, es decir superior a 10000000, debe realizar el complemento
//a 2 para la temperatura cuyo valor es negativo
 if(tempS1 > 128){
    tempS1 = tempS1 - 256;
 }
Wire.beginTransmission(U2) ;     //comenzar a comunicar con sensor U2
Wire.write(0xAA);                //Se envió comando para leer la ultima conversión
Wire.endTransmission();          //finalizar comunicación
Wire.requestFrom(U2, 1);    // Solicitar 1 byte al sensor U2
tempS2=Wire.read();
//sí es mayor que 128, es decir superior a 10000000, debe realizar el complemento
//a 2 para la temperatura cuyo valor es negativo
 if(tempS2 > 128){
    tempS2 = tempS2 - 256;
 }

Wire.beginTransmission(U3) ;     //comenzar a comunicar con sensor U3
Wire.write(0xAA);                //Se envió comando para leer la ultima conversión
Wire.endTransmission();          //finalizar comunicación
Wire.requestFrom(U3, 1);    // Solicitar 1 byte al sensor U3
tempS3=Wire.read();
//sí es mayor que 128, es decir superior a 10000000, debe realizar el complemento
//a 2 para la temperatura cuyo valor es negativo
 if(tempS3 > 128){
    tempS3 = tempS3 - 256;
 }
Wire.beginTransmission(U4) ;     //comenzar a comunicar con sensor U4
Wire.write(0xAA);                //Se envió comando para leer la ultima conversión
Wire.endTransmission();          //finalizar comunicación
Wire.requestFrom(U4, 1);    // Solicitar 1 byte al sensor U4
tempS4=Wire.read();
//sí es mayor que 128, es decir superior a 10000000, debe realizar el complemento
//a 2 para la temperatura cuyo valor es negativo
 if(tempS4 > 128){
    tempS4 = tempS4 - 256;
 }
```

Fuente: Foto sacada por el Alumno de Arduino.

Figura 2-10. Programación parte 3.

En la cuarta etapa del código, se incluye lo último, que sería la configuración del Display LCD para que muestre los valores medios de cada sensor, que seria las variables que se guardaron anteriormente, lo que se muestra en la figura 2-11.

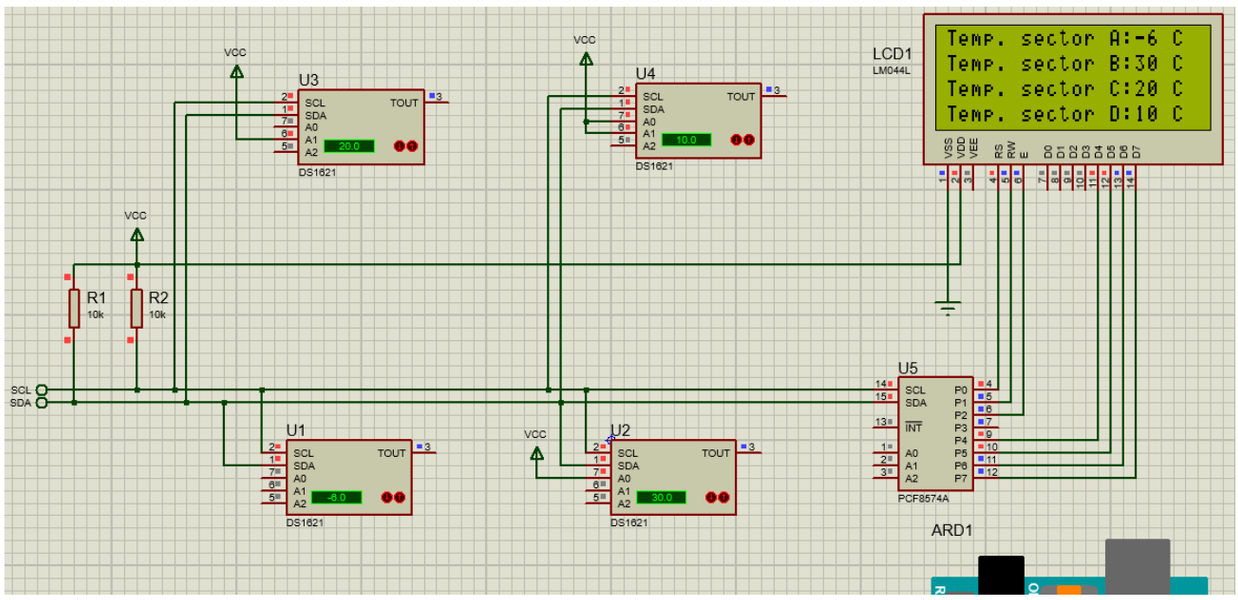
```
  lcd.clear(); //limpiar la pantalla del LCD cada iteración
  lcd.setCursor(0, 0); //fijar el cursor de escritura en la línea 0 y columna 0
  lcd.print("Temp. sector A:");
  lcd.print(tempS1);
  lcd.print(" C");
  lcd.setCursor(0, 1); //fijar el cursor de escritura en la línea 0 y columna 1
  lcd.print("Temp. sector B:");
  lcd.print(tempS2);
  lcd.print(" C");
  lcd.setCursor(0, 2); //fijar el cursor de escritura en la línea 0 y columna 2
  lcd.print("Temp. sector C:");
  lcd.print(tempS3);
  lcd.print(" C");
  lcd.setCursor(0, 3); //fijar el cursor de escritura en la línea 0 y columna 3
  lcd.print("Temp. sector D:");
  lcd.print(tempS4);
  lcd.print(" C");
}
}
```

Fuente: Foto sacada por el Alumno de Arduino.

Figura 2-11. Programación parte 4.

2.5 APLICACIÓN FUNCIONANDO

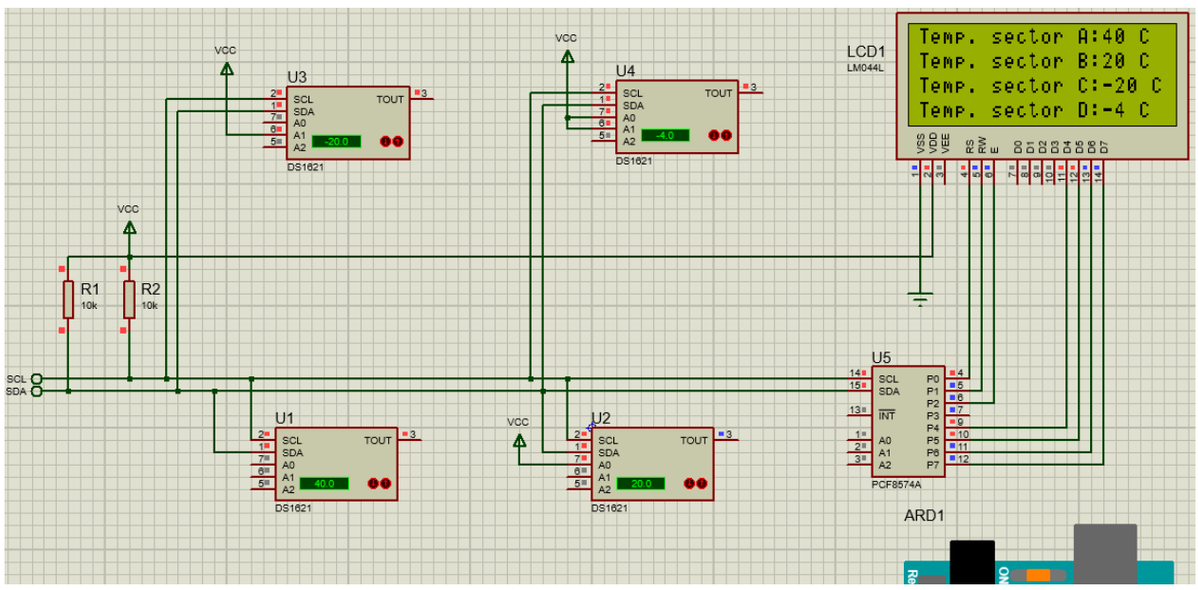
Para finalizar se evidenciará que el programa funciona correctamente en la simulación realizada en Proteus, haciendo 2 pruebas con distintas mediciones. Ver figura 2-12



Fuente: Foto sacada por el Alumno de Proteus.

Figura 2-12. Aplicación funcionando. Prueba 1.

Segunda prueba de funcionamiento, lo que se muestra la figura 2-13.



Fuente: Foto sacada por el Alumno de Proteus.

Figura 2-13. Aplicación funcionando. Prueba 2.

CAPÍTULO 3: RESULTADOS DE IMPLEMENTACIÓN Y COSTOS DEL PROYECTO

3 RESULTADOS DE IMPLEMENTACIÓN Y COSTOS DEL PROYECTO

En los capítulos anteriores se expuso acerca de los dispositivos que se utilizarán, el diseño del circuito, la programación y se hizo un estudio de la comunicación I2C. En el siguiente capítulo a tratar se expondrá acerca de los costos de los materiales para la implementación física de la aplicación y el resultado de ésta, donde se mencionarán los pasos que se siguieron para la implementación del circuito tratados de forma específica en el capítulo 2, exponiendo los problemas obtenidos a lo largo de este proceso y sus soluciones, algunos imprevistos requiriendo algún cambio que no estaban considerados previamente.

3.1 RESULTADOS DE IMPLEMENTACIÓN

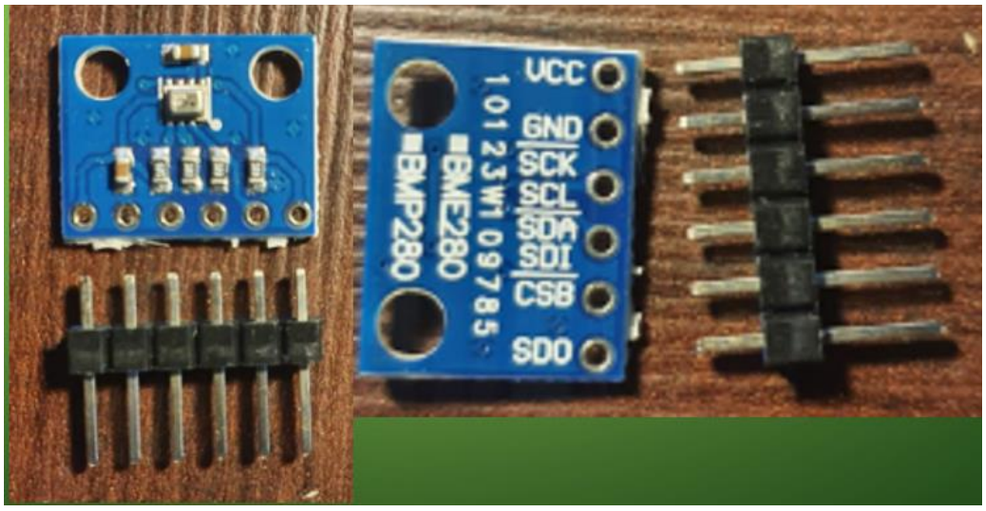
En esta sección se expondrá el resultado de la implementación de la aplicación, el cual lamentablemente no se pudo encontrar el sensor que se iba a ocupar, DS1621, debido a factores ligados a la dificultad de stock y movilización por la actual pandemia en el mundo, que ha hecho difícil la obtención de los materiales y tiempo de espera extenso de envío de los productos desde sus respectivos proveedores, por esta razón se cambió el sensor a uno existente en el mercado nacional. Se expondrán sus características técnicas y una breve explicación de su funcionamiento.

3.1.1 Nuevo sensor

Por las razones expuestas, se cambió a un sensor digital de presión barométrica absoluta y temperatura, BMP280, lo que se muestra en la figura 3-1. Funciona con un ruido más bajo, admite nuevos modos de filtro y una interfaz SPI e I2C. El módulo proporciona una medición precisa de presión barométrica y temperatura en el medio ambiente; debido a su alta precisión en la medición de la presión, y los cambios de presión con la altitud, es posible calcular la altitud con precisión que hace que sea un altímetro exacto. Con este sensor también es posible medir la humedad relativa. Su rango de medición de temperatura se extiende desde -40 °C hasta 85 °C, es equivalente en grados Fahrenheit entre -40 °F a 185 °F, con una precisión de ± 1. La tarjeta se puede alimentar con 3.3 V, valor que la tarjeta microcontrolador tiene disponible, una fuente de 3.3 V añadida en su tarjeta. La dirección del dispositivo se puede cambiar con el pin SDO, que es el bit menos significativo, debido a esto se podrán ocupar solo dos sensores para la aplicación, como se expuso en el capítulo 1 no se pueden ocupar dos direcciones idénticas para la

comunicación I2C. El pin CSB tiene que estar conectado a VCC para poder ocuparlo en la interfaz I2C, si se conecta a tierra el sensor va a ser ocupado para interfaz SPI. Para poder operar este sensor se tuvo que utilizar la librería “Adafruit_BMP280.h”.

La figura 3-1 representa el sensor de temperatura BMP280.



Fuente: Foto sacado por el alumno

Figura 3-1. Sensor de temperatura BMP280.

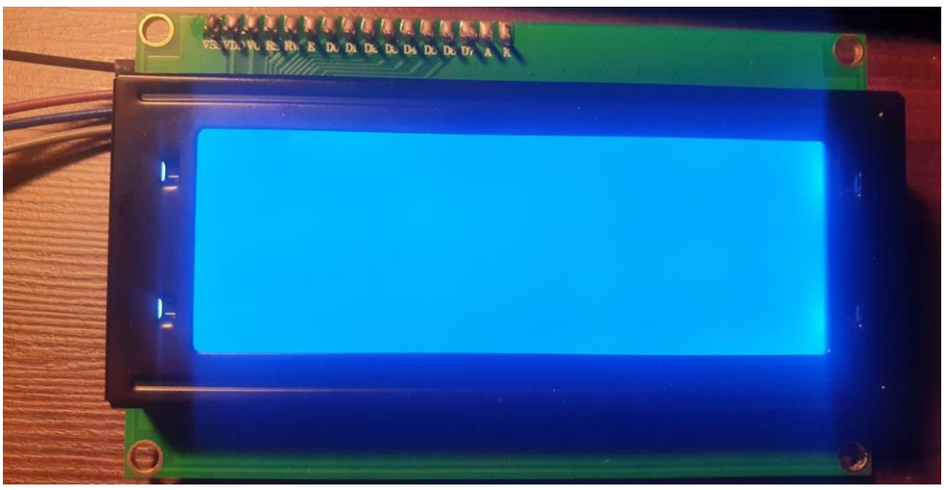
Los pines se soldaron con estaño para que quedara fijo y poder utilizarlo en el protoboard, lo que se muestra figura 3-2.

Fuente: Foto sacado por el alumno

Figura 3-2. Soldado de pines.

3.1.2 Configuración de contraste

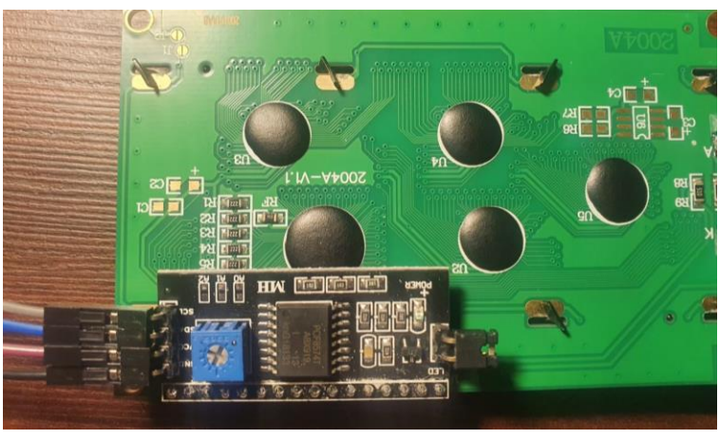
El Display LCD que se compró fue el ya mencionado en el capítulo 1, al implementar no mostraba ninguna letra ni caracteres, solo se prendía la pantalla, se pensó que el módulo I2C estaba malo, pero finalmente el problema era que tenía demasiado nivel de brillo, lo que se muestra la figura 3-3.



Fuente: Foto sacado por el alumno

Figura 3-3. LCD con mucho brillo.

Para poder ver las letras y caracteres se tuvo que ajustar con un destornillador en el módulo I2C, el potenciómetro de contraste, moverlo hasta encontrar el brillo preciso, lo que se muestra figura 3-4.

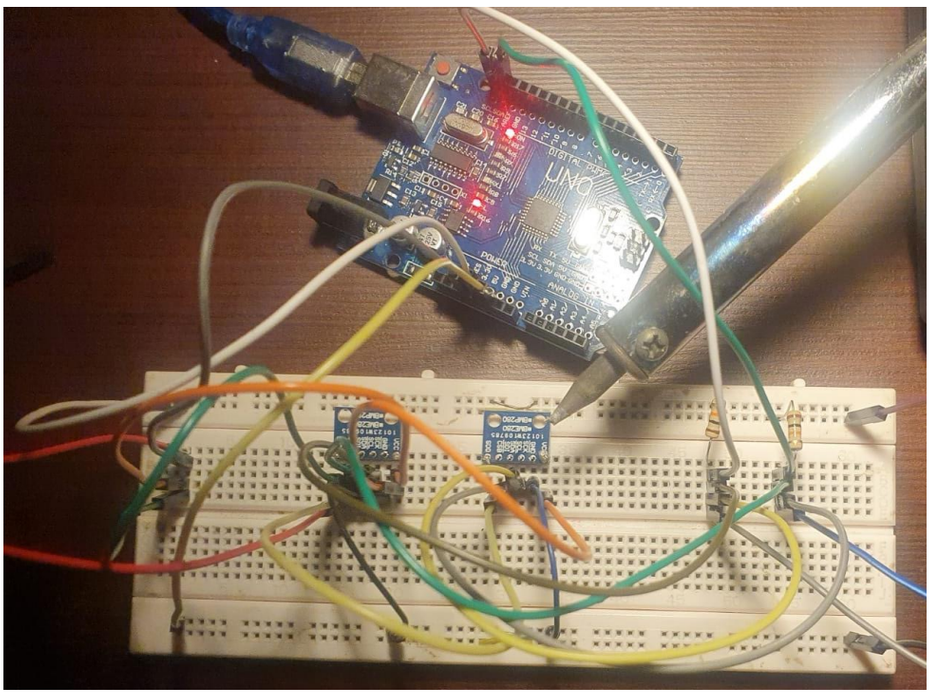


Fuente: Foto sacado por el alumno

Figura 3-4. Potenciómetro de contraste.

3.1.3 Circuito implementado

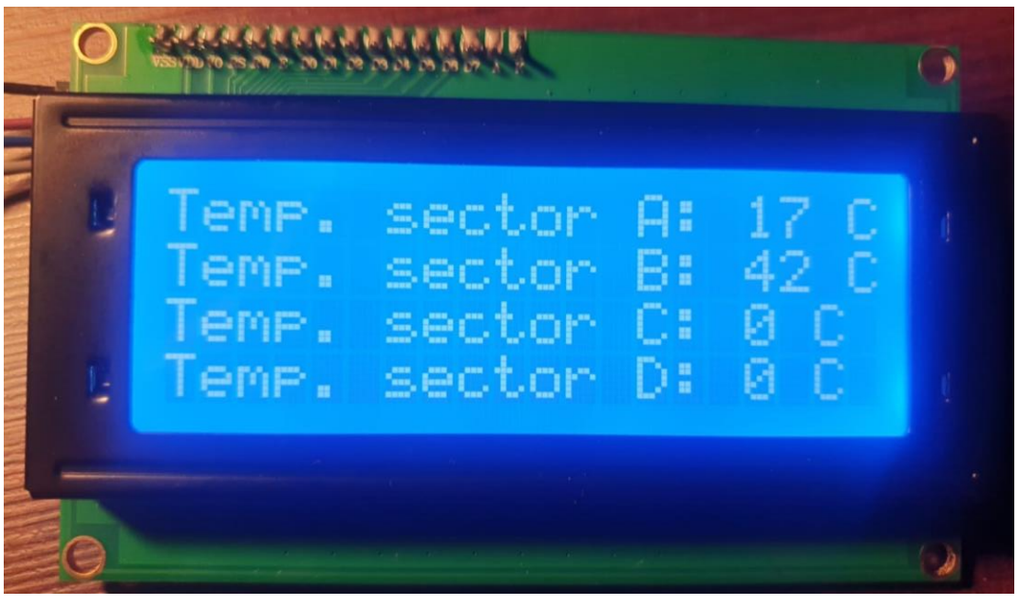
En esta sección se expondrá la implementación del circuito completo, haciendo dos pruebas para su correcto funcionamiento. En la figura 3-5 se muestran los sensores, el que está primero de izquierda a derecha es el que mide el sector A y el segundo es el sector B, los otros sectores marcan 0, ya que no están disponibles. El sector A esta en temperatura ambiente y el sector B tiene cerca un Cautín tibio, lo que se muestra figura 3-5.



Fuente: Foto sacado por el alumno

Figura 3-5. Circuito final.

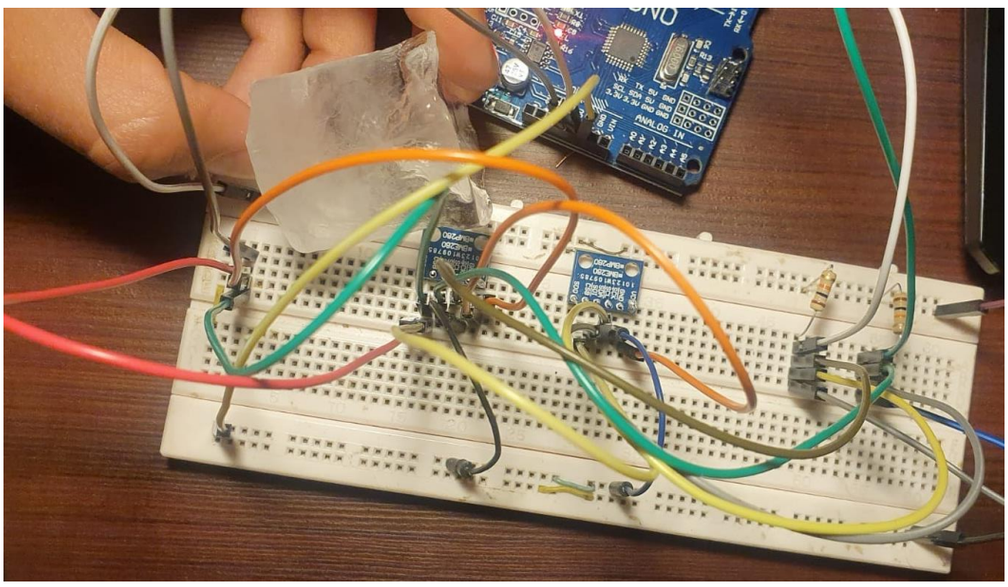
La prueba de la figura 3-5 indica los valores que se presentan en el Display LCD, figura 3-6.



Fuente: Foto sacada por el alumno

Figura 3-6. Resultados de prueba.

Para la segunda prueba se pondrá un hielo cerca del sensor que marca la temperatura del sector A y el sector B estará con temperatura ambiente, lo que se muestra figura 3-7.



Fuente: Foto sacada por el alumno

Figura 3-7. Circuito final. Prueba 2.

Resultados de segunda prueba del circuito de la figura 3-7. Ver figura 3-8.

Fuente: Foto sacada por el alumno

Figura 3-8. Resultado de prueba 2.

3.2 COSTOS DE LA APLICACIÓN

A continuación, se expondrán los valores de costo de los componentes y los materiales utilizados para realizar esta aplicación, además de la mano de obra, que corresponde a una estimación de un costo fijo mensual ligado a la remuneración de un técnico electrónico jornada completa y luego el cálculo de las horas-hombre. Cabe destacar que los valores en UF y dólares se realizaron considerando el valor equivalente a pesos chilenos en la fecha de 08 de agosto de 2021, en donde 1 UF es equivalente a 29.765,32 CLP y 1 USD corresponde a 788,25 CLP.

3.2.1 Componentes

En este aparatado se expondrá de forma detallada el estudio de los componente y materiales que se utilizaron para que esta aplicación operara correctamente, se expondrán 2 proveedores, para luego elegir la oferta más conveniente, para así poder hacer un estudio más completo de productos y conseguir los mismos componentes, pero a un precio más conveniente, lo que se muestra la tabla 3-1. Cabe destacar que se hará el estudio de los costos con los dos sensores utilizados en este trabajo de título.

Tabla 3-1. Proveedores de componentes y material.

Componente/Material	Proveedor	Costo (CLP)	Costo (USD)
Arduino Uno R3	Global Chile electrónica	$9.950	12,62
	Makerschile	$10.490	13,31
Sensor DS1621	AliExpress	$2.197	2,79
	eBay	$3.521	4,47
Sensor BMP280	Altronics	$2.891	3,67
	Global Chile electrónica	$1.980	2,51
Módulo adaptador I2C	Global Chile electrónica	$1.500	1,90
	MercadoLibre	$1.990	2,52
Display LCD	MercadoLibre	$7.000	8,88
	Global Chile electrónica	$5.500	6,98
Protoboard	Global Chile electrónica	$6.000	7,69
Resistencia 10k	MercadoLibre	$127	0,16
Cables dupont	Global Chile electrónica	$30	0,038

Fuente: Tabla creada por el alumno

3.2.2 Costo de mano de obra

Debido que la aplicación se realiza por medio de componentes y materiales, la mano de obra es fundamental para poner en marcha la aplicación, además del tiempo utilizado en realizar la investigación en todo el informe, por ende, se investigó datos respecto al salario que gana un técnico electrónico en Chile, según la página neuvoo.cl el promedio es de $5.400.000 anuales o $2.769 por hora, cabe destacar que para profesionales con más experiencia pueden llegar a ganar mucho más.

Gracias a esta información se puede determinar valores aproximados para poder calcular la mano de obra utilizada en la aplicación, se estima que para construir esta aplicación 100% funcional se requiere un tiempo alrededor de 5 horas, en donde se hicieron las debidas conexiones y las pruebas correspondientes. En la tabla 3-2 se presenta el costo de la mano de obra para implementar esta aplicación.

Tabla 3-2. Costo de mano de obra, implementación.

Horas dedicadas	Costo hora - hombre (CLP)	Costo hora – hombre (UF)	Costo total por mano de obra (CLP)	Costo total por mano de obra (UF)
5	$2.769	0,093	$13.845	0,47

Fuente: Tabla creada por el alumno

Si se considera las horas en donde se programó y se investigó toda la información de este informe sería alrededor de 90 horas, lo que se muestra la tabla 3-3.

Tabla 3-3. Costo de mano de obra considerando investigación y programación.

Horas dedicadas	Costo hora - hombre (CLP)	Costo hora – hombre (UF)	Costo total por mano de obra (CLP)	Costo total por mano de obra (UF)
90	$2.769	0,093	$249.210	8,37

Fuente: Tabla creada por el alumno

3.2.3 Costo final del proyecto

En esta sección se expondrá la elección de los componentes más económicos de los proveedores antes expuestos, generando una tabla final de los componentes y materiales mostrando la cantidad adquirida, ver tabla 3-4. Cabe destacar que no se pondrá el sensor DS1621 ya que no fue un gasto, debido que no se pudo comprar y se hicieron las pruebas con el sensor BMP280.

Tabla 3-4. Proveedores de componentes y material.

Componente/Material	Proveedor	Costo (USD)	Cantidad	Costo final (USD)
Arduino Uno R3	Global Chile electrónica	12,62	1	12,62
Sensor BMP280	Global Chile electrónica	2,51	2	5,02
Módulo adaptador I2C	Global Chile electrónica	1,90	1	1,90
Display LCD	Global Chile electrónica	6,98	1	6,98
Resistencia 10k	MercadoLibre	0,16	2	0,32
Protoboard	Global Chile electrónica	7,69	1	7,69
Cables dupont	Global Chile electrónica	0,038	10	0,38
			Costo total	34,91

Fuente: Tabla creada por el alumno

El costo total de los componentes y materias es de 34,91 (USD) equivalente a pesos chilenos de 27.195 (CLP). Si se consideran todos los valores obtenidos, ya sea costo de mano de obra y costo final de los componentes y materiales que se necesitaron para probar la aplicación, se puede calcular el valor final de éste, teniendo presente que se tendrá dos tablas considerando el tiempo invertido en la programación e investigación y por otra parte solo la implementación, en este trabajo de título, lo que se muestra la tabla 3-5.

Tabla 3-5. Costo final implementación.

Costo total de componentes/materiales (CLP)	Mano de obra (CLP)	Total (CLP)	Total (UF)
$27.195	$13.845	$41.040	1,38

Fuente: Tabla creada por el alumno

El valor final de la implementación corresponde a $41.040 CLP equivalente a 52.06 USD. A continuación, se expone costos finales considerando el tiempo invertido en investigación lo que se muestra tabla 3-6.

Tabla 3-6. Costo final con investigación y programación.

Costo total de componentes/materiales (CLP)	Mano de obra (CLP)	Total (CLP)	Total (UF)
$27.195	$249.210	$276.405	9.29

Fuente: Tabla creada por el alumno

El valor final con programación e investigación corresponde a $276.405 CLP equivalente a 350.66 USD. Se logra observar que el valor final de la aplicación aumenta debido a la mano de obra ejercida, dependiendo de cuántas horas se utilizaron, donde uno tiene un valor más alto debido que se consideró el tiempo de investigación y de programación. Estos valores pueden cambiar en el tiempo, ya que se puede variar el salario de los técnicos electrónicos o también que varíen los valores de los componentes utilizados. Cabe recordar que si es un técnico electrónico con más experiencia el sueldo es bastante superior por lo cual los valores aumentarían considerablemente. Se hicieron dos tablas, ya que si se conoce la programación dada en el capítulo 2 solo hay que implementar, pero si se otorga crédito al creador del programa se tiene que considerar las horas de programación.

CONCLUSIONES

Al inicio del desarrollo de este informe se buscaba aprender sobre la comunicación I2C, como se establece conexión entre dispositivos y aprender las funciones que se utilizaron para programar la comunicación I2C por medio de la plataforma Arduino, para así poder implementarlo, se observa que cada dispositivo debe tener una dirección única, cosa que es muy importante debido que se generará un error al encontrar dos dispositivos con la misma dirección haciendo que el maestro no los reconozca. Cabe destacar que para cada aplicación que se requiera utilizar la comunicación I2C, se tiene que calcular la capacidad para no superar el máximo que es 400pF, si busca una aplicación de actualización casi inmediata no le servirá esta comunicación y tendrá que optar por otra.

Respecto al funcionamiento de la aplicación, se observa una respuesta favorable, cumpliendo con los objetivos planteados al inicio de este proyecto, pero al simular en Proteus ocurría una anomalía respecto a la aplicación, ya que al pasar un tiempo indeterminado se quedaba pegada la simulación. Investigando, se llegó a la conclusión de que era el Software Proteus que se tenía, ya que no era el original, por eso ocurría dicha anomalía.

Desgraciadamente no se pudo implementar con los 4 sensores de temperatura que se tenían en mente al inicio de este documento, debido a lo difícil de conseguir los materiales, pero se pudo comprobar la implementación con el sensor BMP280.

BIBLIOGRAFÍA

Aprendiendo Arduino [En línea]. 2017.

https://aprendiendoarduino.wordpress.com/2017/07/09/i2c/ >> Consulta: 04 de mayo 2021

HETRPRO. I2C [En línea]. 2018.

https://hetpro-store.com/TUTORIALES/i2c/ >> Consulta: 04 de mayo 2021

Descripción y funcionamiento del bus I2C. [En línea]. 2019.

http://robots-argentina.com.ar/didactica/descripcion-y-funcionamiento-del-bus-i2c/

>> Consulta: 04 de mayo 2021

Arduino UNO. [En línea]. 2017.

https://arduino.cl/que-es-arduino/ >> Consulta: 05 de mayo 2021

Wire Library. [En línea]. 2019.

https://www.arduino.cc/en/Reference/Wire >> Consulta: 06 de mayo 2021

Módulo I2C para pantalla LCD. [En línea]. 2019.

https://www.electroallweb.com/index.php/2019/12/13/modulo-i2c-para-pantallas-lcd/

>> Consulta: 08 de mayo 2021

1-wire to I2C. [En línea]. 2016.

https://picaro.mcielectronics.cl/ejemplo-16/ >> Consulta: 20 de mayo 2021

Maxim Integrated. [En linea]. 2015.

https://pdfserv.maximintegrated.com/en/ds/DS1621.pdf >> Consulta: 20 de mayo 2021

Servicio de impuestos internos. [En línea]. 2021.

https://www.sii.cl/valores_y_fechas/dolar/dolar2021.htm >> Consulta: 30 de julio 2021

Salario de técnico electrónico en Chile. [En línea]. 2021.

https://neuvoo.cl/remuneracion/?job=T%C3%A9cnico%20Electr%C3%B3nico#:~:text=El%20salario%20promedio%20de%20T%C3%A9cnico%20Electr%C3%B3nico%20en%20Chile%20es%20de,llegar%20a%20ganar%20hasta%20%249.180. >> Consulta: 30 de julio 2021

BMP280. [En línea]. 2020.

http://electropro.pe/image/data/imgProductos/005.%20Sensor%20De%20Presi%C3%B3n%20Barom%C3%A9trica%20%E2%80%93%20BMP180/BMP280 >> Consulta: 27 de julio 2021

BMP280 Digital Pressure Sensor. [En linea]. 2015.

https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf >> Consulta: 27 de julio 2021

GLOSARIO

MICROCONTROLADOR: Circuito integrado programable, capaz de ejecutar las ordenes grabadas en su memoria, es capaza de ejercer funciones de tipo industrial, de tamaño compacto, por lo que se le atribuye el nombre micro. Se utiliza para automatizar procesos y procesar información.

RESISTENCIAS DE PULL-UP: Resistencias conectadas a la implementación, generando un 1 lógico a la salida, hasta que se activa o presiona el interruptor que está conectado a tierra para cerrar el circuito resultando un 0 lógico. Evita falsos estados que se producen por el ruido.

I2C: Circuito Inter Integrado (inter integrated circuit o I2C), es un estándar que facilita la comunicación entre microcontroladores, estableciendo una comunicación de datos en serie y sincrónica a los dispositivos conectados.

MAESTRO: Son los dispositivos que inician y coordinan la comunicación, además son los únicos que pueden generar la señal de reloj y direccionar la información.

ESCLAVO: Dispositivos que están a la espera de que el maestro mande el primer byte de direccionamiento para comunicarse con el esclavo deseado. Pueden operar de dos formas; esclavo – transmisor o esclavo – receptor.

BYTE: Pronombre que se ocupa en informática, un byte es la unidad de información, compuesta a su vez por ocho dígitos binarios (0 y 1) o bits.