CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SECCIÓN DE BIOELECTRÓNICA

"Desarrollo y construcción de un sistema de medición y control de los parámetros de una cámara de incubación para el crecimiento de monocapas celulares *in vitro*"

T E S IS

Que presenta

Luis Manuel Martínez Méndez

Para obtener el Grado de

Maestro en Ciencias

en la Especialidad de

Ingeniería Eléctrica

Director de Tesis:
Dr. Pablo Rogelio Hernández Rodríguez

México, D.F. Febrero 2016

DEDICATORIAS.

A mis padres.

Por estar siempre a mi lado en los buenos momentos y en los no tan buenos, por ser esa inspiración en mi vida, esa fuerza que hace que me levante cada vez que caigo, por escucharme cada vez que necesito un consejo, por ayudarme cada vez que necesito apoyo.

A mis hermanos.

Por ser cómplices de cada una de las etapas de mi vida, permanecer en etapas difíciles de mi vida, por tanto apoyo que me han brindado. Y a ti en especial, dónde quiera que te encuentres, sé que estás en un mejor lugar, gracias por aquellas palabras que me regalaste cuando más las necesitaba, te llevaré siempre en mi corazón.

Luis Manuel Martínez Méndez

Autor

AGRADECIMIENTOS.

A mi director de tesis, Dr. Pablo Rogelio Hernández Rodríguez, por las enseñanzas dadas, por haberme recibido en su laboratorio, y por el apoyo en la duración del proyecto.

Al ingeniero Eladio Cardiel Pérez, por la paciencia a lo largo del proyecto, por sus consejos y orientación parte y fuera de la maestría, gracias Inge.

Mi agradecimiento también es para el Dr. Javier Hernández del departamento de Genética, para el Dr. Luis Reyes y su auxiliar de investigación Elsa Irene Sánchez del departamento de Fisiología, por permitirme usar sus sistemas de incubación para células y por tener todas las veces que se necesitó hacer pruebas de verificación del sistema desarrollado.

A Gonzalo por ser parte de este proyecto, por toda la ayuda brindada cuando por algún tipo de circunstancia las ideas no fluyen, por ser consejero en las buenas y en las no tan buenas, gracias por toda la ayuda que me brindaste fuera del proyecto Gonzo.

A Sandrita Arias y Toño, porque desde que los conocí me brindaron su ayuda y todo su apoyo, y siempre me aportaban ideas buenas que pudieran servir dentro y fuera del proyecto.

Gracias a Liz Hernández, Juan De Dios Gómez López y Juan Carlos Cifuentes, por dejarme formar parte de su familia, y permitirme estar compartiendo el mismo techo con ustedes amigos, por ser el enlace con el departamento de Genética y por toda la confianza depositada, fue muy grato estar conviviendo con ustedes, por escucharme y siempre darme consejos.

Y por último y no por ello menos importante a mis amigos de que me acompañaron en todo el camino, Anais, Gaspar, Juan (Cory), Hayde, Esmeralda Fonseca, Salmita,

Chinita Nelly… y todos los que tocaron mi vida de una manera u otra, que formaron parte de esta etapa, muchas gracias por formar parte.

CONTENIDO

LISTA DE FIGURAS.

LISTA DE TABLAS.

RESUMEN

Los cultivos celulares son esenciales en la investigación científica. Para lograr su adecuado crecimiento es necesario mantenerlas bajo condiciones adecuadas de temperatura, humedad, y dióxido de carbono.

En el presente trabajo se ha implementado un sistema para el monitoreo y control de dióxido de carbono [CO_2], así como para mediciones y registro de temperatura y humedad dentro de una cámara de incubación, con el objetivo de facilitar el buen crecimiento de monocapas celulares que serán sometidas a diferentes pruebas de investigación.

El sistema se basó en el dispositivo K-33 BLG para la medición de [CO_2] y en el DHT22 para el monitoreo de la temperatura y humedad relativa. Cuenta con una interfaz gráfica para el control y monitoreo gráfico de [CO_2], para la presentación de los valores de temperatura y humedad relativa, configuración de referencias (set points) de los parámetros y para alertar en casos de superación de umbrales. El sistema es autónomo, utiliza como dispositivo de procesamiento de información, control y almacenamiento, un sistema de microcontroladores y se puede adaptar en diversos tipos de incubadoras.

Se elaboraron protocolos de prueba para los dispositivos y el sistema. La validación de las mediciones se realizó tomando como referencia los medidores de tres incubadoras comerciales.

Los resultados de las pruebas de funcionamiento y validación indican un buen desempeño del sistema desarrollado y marcan el inicio de su uso en procesos experimentales.

ABSTRACT

Cell cultures are essential in research. They need to be under appropriate conditions of temperature, humidity, and carbon dioxide in order to promote their growth.

A system for monitoring and controlling the CO_2 concentration as well as for measuring and recording, both temperature and humidity parameters inside of an incubation chamber, has been developed in this work. This system was designed to support the development of healthy growth of monolayer cultures and their survival.

The system was based on the device K-33 BLG for CO_2 concentration measurements and the sensor DHT22 for monitoring temperature and relative humidity. A graphical interface was developed for the control and monitoring of CO_2, exhibition of temperature and relative humidity values, definition of set points parameter, and for alerting when thresholds have been exceeded. The system is autonomous and can be adapted in several types of incubators. It uses a microcontrollers array for information processing, control and storage.

Some protocols were designed in this work for assessing and characterizing the devices, processing software and the complete system. The validation was performed using three commercial incubators as reference.

According to the good results in performance and validation, this system can be recommended to be used in experimental procedures.

CAPÍTULO 1. INTRODUCCIÓN.

Las áreas de las ciencias médico-biológicas que requieren hacer estudios con cultivos celulares son, entre otras: la fisiología celular, toxicología y virología, la ingeniería de tejidos, la producción de tejidos biológicos, citogenética y transgénica. Estos cultivos celulares deben tener condiciones apropiadas para desarrollarse y multiplicarse.

Entre las condiciones que se desean para un buen desarrollo y crecimiento de monocapas celulares encontramos que el laboratorio debe tener ciertas características y requerimientos: un mantenimiento de la asepsia necesaria, libres de cualquier tipo de contaminación principalmente microbiana y fúngica, y desde luego condiciones ambientales necesarias de [CO_2], temperatura, humedad, así como bajos niveles energéticos luminosos, eléctricos y magnéticos perturbadores [2].

En consecuencia, resulta de fundamental importancia tener un estricto control en las variables que influyen en el crecimiento celular, particularmente cuando son sometidos a experimentación.

Existe una gran variedad de equipos que son utilizados en laboratorios de cultivos celulares, algunos de ellos son muy específicos y otros muy generales. Así, tenemos cabinas de flujo laminar, incubadoras de CO_2, instrumentos ópticos de observación del cultivo como lupas de microcirugía, microscopios invertidos y/o convencionales y equipos fotográficos, neveras y congeladores, equipos de esterilización como los autoclaves, baños con temperatura controlada, centrífugas, contadores electrónicos de células, equipos de purificación de agua, balanzas, pH-metro, pipeteadores, etc. [2].

El ambiente apropiado para un cultivo celular se caracteriza por tres parámetros fundamentales:

1. Temperatura: La temperatura es muy importante dentro del cultivo celular, la mayor parte de las células que se cultivan dentro de un laboratorio son de animales mamíferos y las temperaturas en las que estas células crecen y se desarrollan son de alrededor de 37 ^{0}C, con una variación de ± 0.5 ^{0}C [3].
2. Dióxido de carbono (CO_2): El suplemento de una proporción adecuada (habitualmente un 5%) de CO2 a la mezcla de gases del interior de la incubadora es esencial para el mantenimiento del pH en el medio de cultivo en niveles a la neutralidad, balanceando el tampón carbonato-bicarbonato disuelto en el medio del cultivo y, por lo tanto, en los valores necesarios para el cultivo [2].
3. Humedad: La humedad tan elevada (95% relativa) que se maneja dentro de los cultivos es para evitar la deshidratación del medio donde se desarrollan las células, pudiendo llegar a comprometer una presión osmótica correcta del medio de cultivo.

Una mención especial la merece la cámara de ambiente controlado (incubadora), toda vez que las condiciones de asepsia resultan de suma importancia se requiere que el material del que están hechas estas cámaras sea de materiales no corrosivos [2].

1.1. Planteamiento del Problema.

El trabajo con cultivos celulares tiene un costo muy elevado, ya que se requiere cumplir con protocolos rigurosos de laboratorio y contar con equipos muy específicos para propiciar un adecuado desarrollo de los mismos. Cabe destacar que estos equipos no permiten la realización de experimentos in situ, ya que generalmente se requiere extraer los cultivos. Este procedimiento involucra factores que pueden modificar importantemente el experimento y los resultados.

Los equipos de incubación para el desarrollo de monocapas celulares, tienen que ser completamente herméticos para evitar que los cultivos celulares puedan estar expuestos a contaminantes, en consecuencia no se puede introducir algún sistema electrónico que requiera de cableado para su alimentación y monitoreo, esto desde luego genera limitaciones para el investigador que desee trabajar con monocapas en un estado de crecimiento

En tal virtud se propone desarrollar un sistema de ambiente controlado, manejando los parámetros necesarios para el desarrollo de monocapas celulares (MDCK), donde se pueda medir y controlar la temperatura, la humedad, el dióxido de carbono, mediante una interfaz de usuario.

El proyecto que se reporta en esta tesis se refiere al desarrollo de una cámara de ambiente controlado que asegure las condiciones de crecimiento de monocapas celulares, confluencia (maduración) y sobrevivencia, durante procesos experimentales de exposición a campos magnéticos.

El equipo incluye una interfaz gráfica de usuario para la configuración de valores de la concentración de CO_2 y para la medición de temperatura y humedad. Se ha incorporado un generador de campos magnéticos en el interior de la cámara, cuyo protocolo de activación se configura en la interfaz.

1.2. Objetivos.

1.2.1. **Objetivo General.**

Desarrollar un sistema de medición y control de parámetros de una cámara de incubación para el crecimiento de monocapas celulares in vitro para la medición de su impedancia eléctrica transepitelial expuestas a campos magnéticos.

1.2.2. **Objetivos Particulares.**

- Caracterizar los sensores de humedad, temperatura y dióxido de carbono.
- Desarrollar los sistemas de control que permitan mantener la concentración de dióxido de carbono requerida.
- Desarrollar y construir un sistema de monitoreo de temperatura y humedad.
- Desarrollar y construir un sistema de almacenamiento de datos en una memoria microSD.
- Validar del medidor de [CO_2].
- Diseñar una Interfaz gráfica de usuario.
- Acondicionar la estructura que servirá como base para la cámara de ambiente controlado.
- Pruebas

1.3. Estructura de la Tesis.

El presente trabajo está dividido en 6 capítulos. En el primer capítulo se da una introducción al tema de los parámetros que se manejan para el crecimiento de monocapa celulares en los equipos existentes, planteando el problema que existe con estos equipos el cual no permiten hacer experimentación in situ. Se menciona el objetivo general y los objetivos particulares que busca el presente trabajo, como son la caracterización de los sensores de humedad relativa, temperatura y dióxido de carbono, desarrollar un sistema de control que permita mantener la concentración de dióxido de carbono requerida, el desarrollo y construcción de un sistema de monitoreo de temperatura y humedad, así como el almacenamiento de los distintos parámetros en una memoria para cuando sean requeridos extraer la memoria y acceder a estos registros.

En el segundo capítulo se presentan los antecedentes necesarios para poder abordar de una cámara de ambiente controlado en las monocapas celulares. En este capítulo

también se hace un estudio del estado del arte con las distintas maneras que trabajan las incubadoras.

En el tercer capítulo se presenta el desarrollo de la solución propuesta, describiendo cada parte de los sistemas con los que se trabajo, desde interfaces para usuario en el sistema para una computadora, pasando por el diseño y construcción de un sistema autónomo que sea capaz de registrar los tres parámetros ya mencionados y controlando los niveles de dióxido de carbono en el interior.

El cuarto capítulo se presenta las pruebas que se llegaron a realizar con los dos sistemas implementados bajo ciertos protocolos establecidos, así como la validación de los sensores que estuvieran funcionando.

En el quinto capítulo se presentan los resultados que se obtuvieron del funcionamiento de los sensores. Se presenta también los resultados y discusiones en la implementación de los protocolos para validación de los dos sistemas que se trabajó.

En el sexto capítulo se presentan las conclusiones del trabajo realizado a través de esta tesis, también se presentan las perspectivas de este trabajo que se puede llegar a realizar por la manera en que fue abordado el problema.

CAPÍTULO 2. ANTECEDENTES Y ESTADO DEL ARTE.

En la actualidad se emplean diferentes técnicas para conservar los cultivos de células bajo características y ambientes controlados con el objetivo de garantizar sus propiedades fisiológicas, hormonas, factores de crecimiento, densidad celular, etc. y físico- químicas, pH, temperatura, presión osmótica, presión parcial de O2 y CO_2, y con ello lograr su reproducción.

Existen cámaras o estufas, con motores de baja velocidad en su interior [2]. Estas cámaras sirven para mantener cultivos en botellas cerradas que no necesitan CO_2. Utilizan botellas con la finalidad de que exista una gran cantidad de adhesión de las células para que se pueda dar una mayor proliferación.

Otras cámaras utilizan un mejor control de la temperatura, son equipadas con camisas de agua (“water jacket”) que incrementan notablemente la inercia térmica; como consecuencia, se tiene el inconveniente de que la estabilización es tardada [2]. Algunas otras están equipadas con termo-resistencias para poder aumentar y estabilizar la temperatura dentro de la cámara en menos tiempo.

Los dispositivos de inyección de una mezcla de aire y CO_2 dan la proporción deseada entre el 4 y el 7%. Dentro de estos sistemas existen dispositivos electrónicos IRGA (“infra- red gas analyzer”) para regular esa proporción. Sin embargo, estos dispositivos suelen descalibrarse después de un cierto tiempo de uso [2].

Los dispositivos para mantener húmedo el ambiente suelen ser bandejas de agua en el fondo del sistema. Esto puede ser peligroso para los cultivos por ser fuentes potenciales de contaminación, convirtiendo en caldo de cultivos a los pocos días [2].

2.1. Temperatura.

Algunos sistemas mantienen las condiciones adecuadas de temperatura dentro de la incubadora, basando su principio de funcionamiento en una o varias resistencias eléctricas que pueden ser operadas por medio de controles microprocesados provistos de sensores de temperatura; otra opción común son los termostatos [2]. Otras cámaras de ambiente controlado manejan camisas de agua que a su vez son calentadas por termostatos para poder hacer una transferencia de calor hacia todo el ambiente dentro de la cámara, esto lo podemos ver en la Figura 4.1.

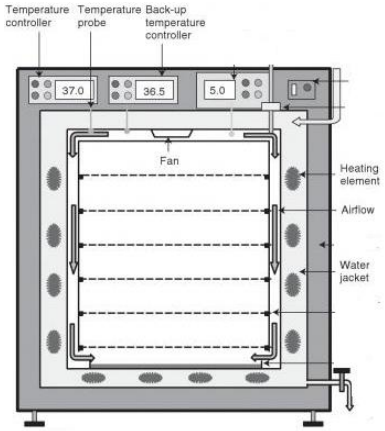


Figura 2.1 Cámara con camisa de agua y un ventilador para circulación de aire.

En cuanto a la transferencia del calor en la cámara interna de las incubadoras se utiliza sistemas de convección natural o forzada siendo estos los más usuales [2].

- **Convección Natural:** Es el movimiento del fluido (calor) por causas naturales, como el efecto de flotación, donde el fluido caliente sube y el fluido frío baja (Figura 4.2 (a)).
- **Convección Forzada:** El fluido (calor) es obligado a desplazarse en la cámara por medios externos como ventiladores o turbinas (Figura 4.2 (b) [2].

Estos dos métodos de propagación de calor los podemos ver en la Figura 4.2, y como es que se da en el interior de las cámaras.

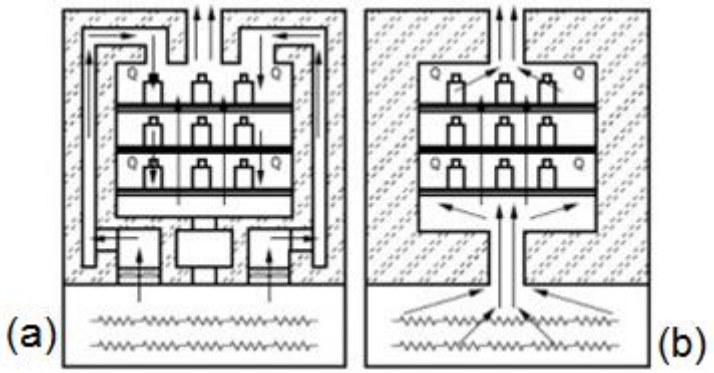


Figura 2.2. Propagación de calor dentro de la cámara, (a) convección natural, (b) convección forzada.

En cualquiera de estos sistemas de transferencia de calor, la resistencia que produce éste se encuentra ubicada en la parte inferior de la incubadora, el aire caliente circula hacia arriba, transfiriendo el calor a las muestras que se encuentran dentro de la cámara y sale por un desfogue ubicado en la parte superior de esta. La homogeneidad en el interior de la cámara depende del sistema de convección que se aplique [2].

2.2. Humedad.

Existen sistemas que manejan los niveles de humedad por medio de charolas o bandejas (Figura 4.3) de tal forma que se garantice una humedad relativa del 95% en el interior de la cámara. Con este valor se evita deshidratación celular y cambios en la presión osmótica del cultivo. Estos sistemas pueden resultar contraproducentes si no se tiene la asepsia debida, ya que puede crearse cultivos bacterianos en estas charolas.

Figura 2.3. Cámara Galaxy 170R, en el interior se muestra la charola contendora de agua.

Actualmente existen sistemas que en lugar de utilizar una charola o bandeja donde se deposite el agua esterilizada, manejan un sistema de alimentación de humedad por medio de inyección (Figura 2.4).

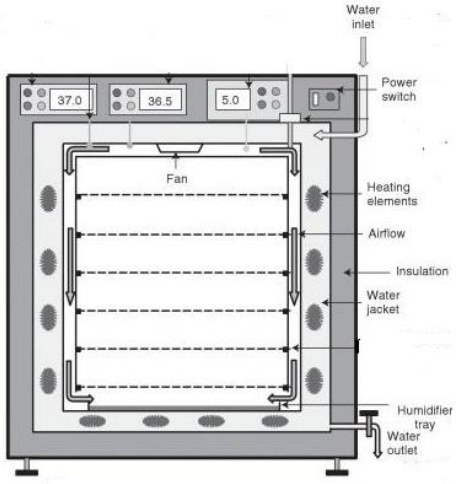


Figura 2.4. Cámara con sistema de control de humedad por inyección.

2.3. Dióxido de Carbono CO_2.

Las cámaras de ambiente controlado que manejan la mezcla de oxígeno y dióxido de carbono, la realizan por medio de inyección regulada para lograr una proporción entre el 4 y 7%, (Figura 2.5). El CO_2 de elevada pureza se suministra en botellas presurizadas y se mezcla en el dispositivo de inyección. El control de la mezcla se

realiza fundamentalmente mediante un dispositivo IRGA (“infra-red gas analyzer”) (Figura 2.6).

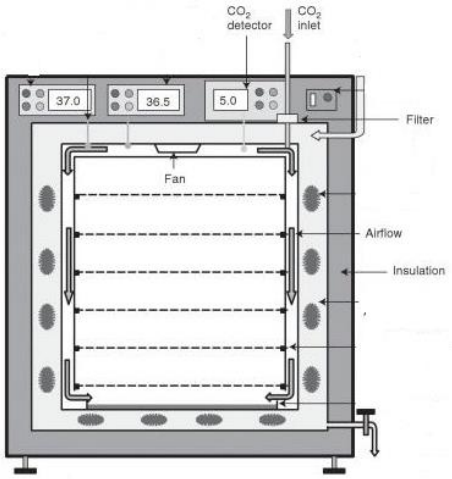


Figura 2.5 Sistema de inyección de mezcla de oxígeno y dióxido de carbono, con filtro HEPA en la entra para evitar contaminantes en el ambiente.

.

En las cámaras recientes se registra constantemente el porcentaje de CO_2 y realiza continuamente una autocalibración ya que un problema usual de estos dispositivos es la pérdida de la calibración (Figura 2.6).

Figura 2.6 Sistemas de medición de dióxido de carbono IRGA con autocalibración.

CAPÍTULO 3. DESARROLLO.

Para la medición y control de los parámetros dentro de la cámara, dióxido de carbono, temperatura y humedad, inicialmente se trabajó con el software propio del sistema con la intención de corroborar el desempeño de los sensores, de acuerdo a las recomendaciones dadas por el fabricante, posteriormente se desarrollo algunos algoritmos que se implementaron en un dispositivo programable para poder registrar y controlar las condiciones del CO_2 , así como para efectuar las mediciones de temperatura y humedad dentro de la cámara.

3.1. Sistema de medición de CO_2 a través de una computadora.

Se utilizará el módulo K-33 BLG de la empresa CO2meter el cual está diseñado para medir porcentaje de dióxido de carbono hasta 30%, así como la temperatura y la humedad. Esto lo hace un excelente sensor para aplicaciones encerradas como lo son las incubadoras.

Las mediciones que se pueden a llegar a realizar con el sistema van desde 0.04% (400 ppm) de dióxido de carbono, que son los niveles que se encuentran normalmente en el ambiente[4], hasta un máximo de 30% de concentración

Este módulo cuenta con un algoritmo interno denominado corrección automática de línea basal (ABC, por sus siglas en ingles), por lo que para aplicaciones en ambientes abiertos el sensor de dióxido de carbono es libre de mantenimiento. El algoritmo ABC permite que el sensor de CO2 cambie dinámicamente su lectura de CO2 por una constante. Es decir, el algoritmo registra la lectura más baja que se haya dado durante un periodo previamente establecido por el usuario, y suponiendo que este valor es menor o igual a un valor conocido (400 ppm), la salida nos dará el valor de la nueva lectura de CO_2 que corresponderá a la suma de la última lectura de CO_2 más el valor de línea basal [5].

.

$$ABC\ Offset = 400 - Lectura\ baja$$

$$Nueva\ Lectura\ de\ CO2 = Lectura\ pasada\ de\ CO2 + ABC$$

Se recomienda deshabilitar el algoritmo en ambientes en donde sus concentraciones sean inestable[5]. Por otro lado debido a que el sistema se utilizara en ambientes con un grado alto de concentración de dióxido de carbono, la calibración del módulo K-33 BLG se debe realizar por medio del software proporcionado por el fabricante a través de la computadora.

Los bloques que conforman el sistema para efectuar los registros y/o calibración del sensor de CO_2, a través de la computadora se muestran en la figura 3.1.

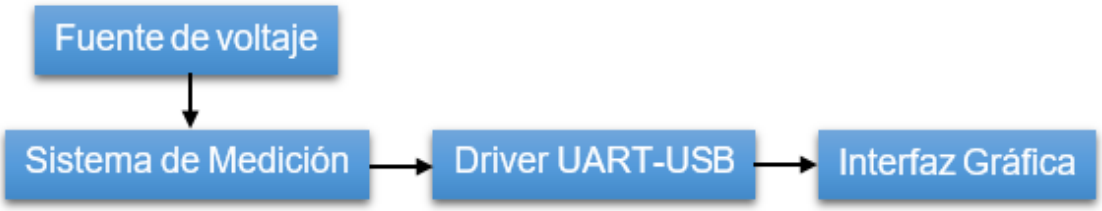


Figura 3.1 Diagrama a bloques del sistema de mediciones de temperatura, humedad y dióxido de carbono, para la PC.

3.1.1. **Sistema de Medición de dióxido de carbono.**

Las mediciones que se harán mediante el dispositivo K-33 BLG serán exclusivamente de dióxido de carbono, no obstante que éste es capaz de entregar información de humedad y temperatura como se documentó previamente. Para estos fines el módulo tiene incorporado un microcontrolador que procesa la señal del sensor de luz infrarroja (IR) directamente [5]. En la figura 3.2 se puede observar parte del módulo de medición en la que se aprecia el microcontrolador mencionado así como el sensor de luz infrarroja (IR).

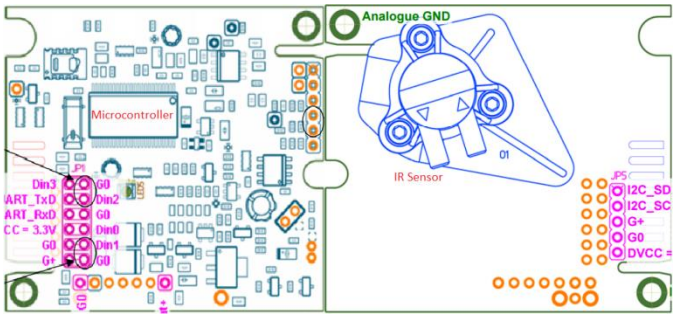


Figura 3.2 Módulo de medición K33 BLG tomado de la hoja de datos de la compañía CO2meter, a la izquierda se observa el microcontrolador; en tanto que a la derecha se puede apreciar el sensor analógico de luz infrarroja (IR).

El sensor de humedad relativa (RH) y temperatura que se encuentra instalado en el módulo K-33 BLG, es un sensor SHT11 (Sensirion Co., Switzerland)[6]. Es un sensor que fue calibrado desde su fabricación en una cámara húmeda de precisión Los coeficientes de calibración se programan en una memoria OTP en el chip, éstos coeficientes se utilizan para calibrar internamente las señales de los sensores. Este dispositivo entrega una salida digital de 14 bits, y que por tanto es capaz de ofrecer una resolución de 16,384[6].En la figura 3.3 podemos observar al módulo de medición y la localización del sensor SHT11.

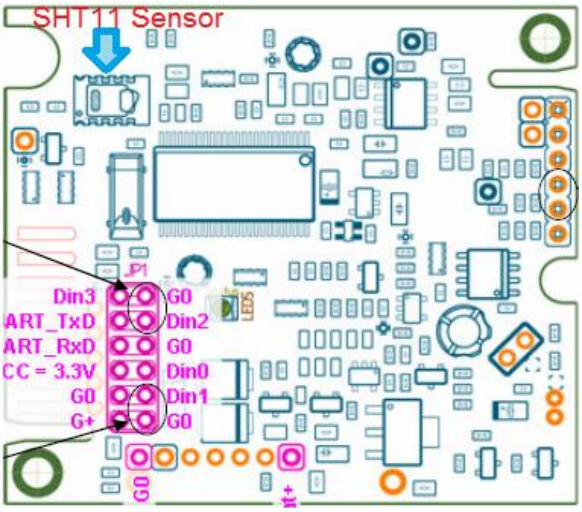


Figura 3.3 Modulo de medición K33 BLG, sensor de temperatura y humedad relativa (RH) SHT11 marca Sensirion.

La función que realiza el módulo de medición K-33 BLG es almacenar los datos que entrega el sensor SHT11 en una memoria Flash.

3.1.2. **Driver UART-USB.**

El módulo de medición K-33.BLG tiene dos protocolos de comunicación con los cuales se puede tener acceso a los datos que se vayan registrando, el primer protocolo de comunicación I^2C, Inter-Integrated Circuit, será descrito en la sección 3.2.9 referente al microcontrolador maestro, el segundo protocolo de comunicación UART, Transmisor-Receptor Asíncrono Universal,[7]. Será descrito en la sección 3.2.9. Siendo ambos protocolos de comunicación seriales. En ambos casos, los protocolos de comunicación se emplean como modelos de comunicación maestro-esclavo.

Para efectuar los registros por medio de la computadora de las magnitudes de temperatura, humedad relativa y dióxido de carbono, se necesita tener un driver que traduzca el protocolo de comunicación UART a USB de manera dúplex, ya que se requerirán registros de las distintas magnitudes y a su vez la PC necesita comunicarse con el módulo de medición.

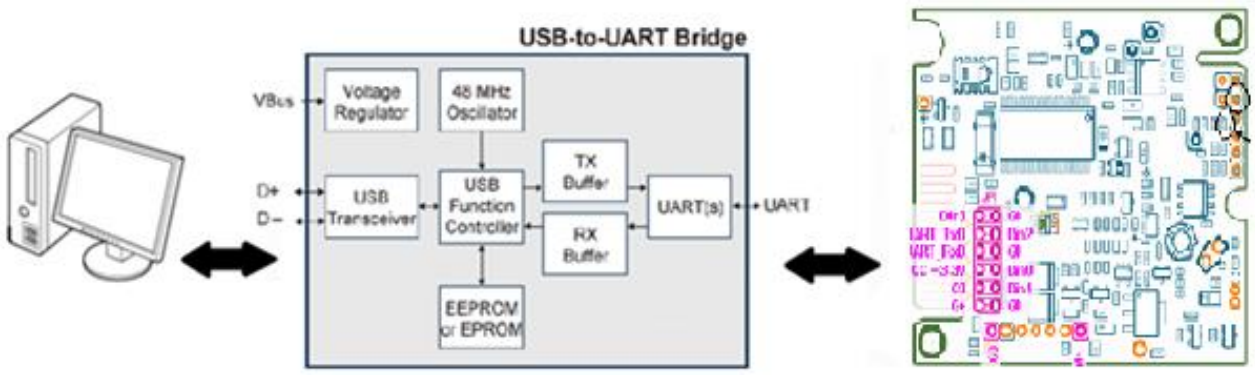


Figura 3.4 Diagrama a bloques del driver de conversión entre los protocolos de comunicación USB-UART.

En la figura 3.4 se observa un diagrama a bloques del driver utilizado para hacer la traducción entre los dos protocolos de comunicación USB para la PC y el UART del módulo de medición.

La compañía CO2meter provee este driver para hacer los registros entre su módulo de medición y la computadora. En la figura 3.5 vemos la conexión en el módulo K-33 BLG y el driver con la conexión USB.

Figura 3.5 Módulo de medición de K-33 BLG conectado al driver UART-USB.

Ambos protocolos (UART y USB) realizan el manejo de datos de manera serial, solo que las tazas de velocidades en la transmisión son diferentes así como el orden y la manera en que se hace la petición de datos, de ahí la importancia del uso de éste driver del protocolo de comunicación.

3.1.3. **Fuente de Voltaje.**

Existen diferentes módulos de medición de dióxido de carbono que maneja la compañía de CO2meter. Estos diferentes módulos manejan diferentes rangos de medición, están pensados para diferentes aplicaciones que se puedan dar con la concentración del gas, las distintas alimentaciones de estos módulos van desde un 3.3 volts hasta 12 volts.

El módulo que utilizamos, K-33 BLG, de acuerdo a sus hojas de información, maneja un voltaje de 4.75 a 12 volts, con dos posibles vías de alimentación. La primera vía de alimentación consiste en una protección para el módulo de medición en caso de un sobre voltaje, esta protección se hace por medio de unos diodos zener que vienen integrados en la placa del sistema, haciendo la función de limitar el voltaje en el posible caso de que exista un pico de voltaje alto [4].

En la figura 3.6 se muestran los diodos zener de protección en serie con una resistencia de 3.3 ohms, la referencia para toda la placa será G0, y la alimentación positiva para el circuito de protección se llamará Vbat+, con un voltaje de alimentación del 5.5 a 12 volts[7].

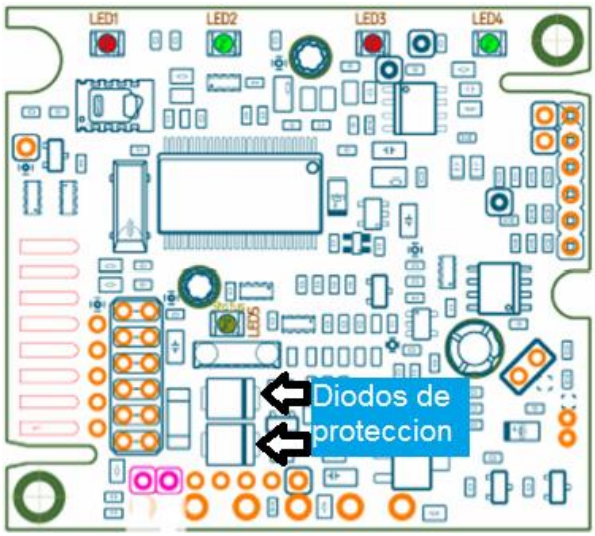


Figura 3.6 Módulo K-33 BLG, alimentación y protección a posibles sobre voltajes.

La segunda manera de poder alimentar el modulo es de manera directa y sin ninguna protección con un rango de voltaje que va desde 4.5 a 9 volts [7]. Pudiendo alimentarla a través de los pines que están cerca las conexiones de transmisión de UART o de las conexiones de transmisión de I^2C.

En la figura 3.7 se observan estas dos conexiones cerca de las áreas de comunicación antes mencionadas.

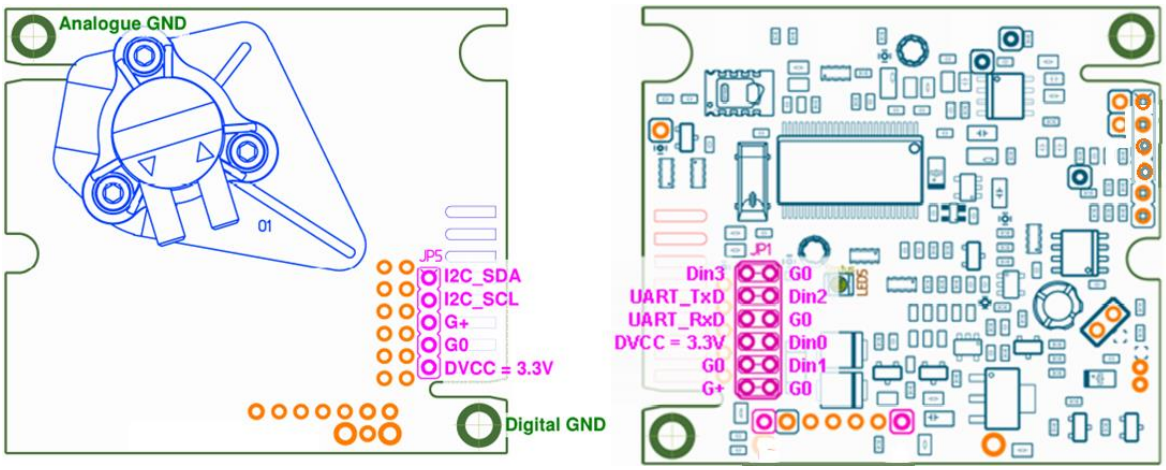


Figura 3.7 Módulo K-33 BLG, pines y forma de alimentación de la placa sin protección a posibles sobre voltajes.

La conexión G+ es el pin positivo en la alimentación en tanto que la referencia es G0, para ambos casos. Para toda la placa la referencia digital será G0.

La manera en que se trabajará el módulo es con protección como lo recomienda el fabricante, para evitar algún daño.

Debido a que el módulo K-33 BLG se estará comunicando con una computadora y se harán pruebas por un medio de un puerto USB, para proteger el puerto de la computadora y la placa de un sobre voltaje o alguna conexión errónea entre el sistema de medición y el puerto USB de la computadora. Se alimentara al módulo con un juego de pilas de 1.5 volts, se requiere un voltaje de 5.5 a 12 volts para las entradas en las alimentaciones que tienen protección, por lo tanto es necesario 4 pilas de 1.5 volts conectadas en serie para poder alcanzar 6 volts en la salida y sea necesario para el funcionamiento del módulo.

3.1.4. **Interfaz Gráfica para pruebas.**

Con la intención de realizar pruebas de reconocimiento y calibración del módulo de medición K-33 BLG del sistema, se comenzó a trabajar con el software llamado

GasLab que pertenece a la misma compañía CO2meter, en la figura 3.8 podemos observar la interfaz general del software.

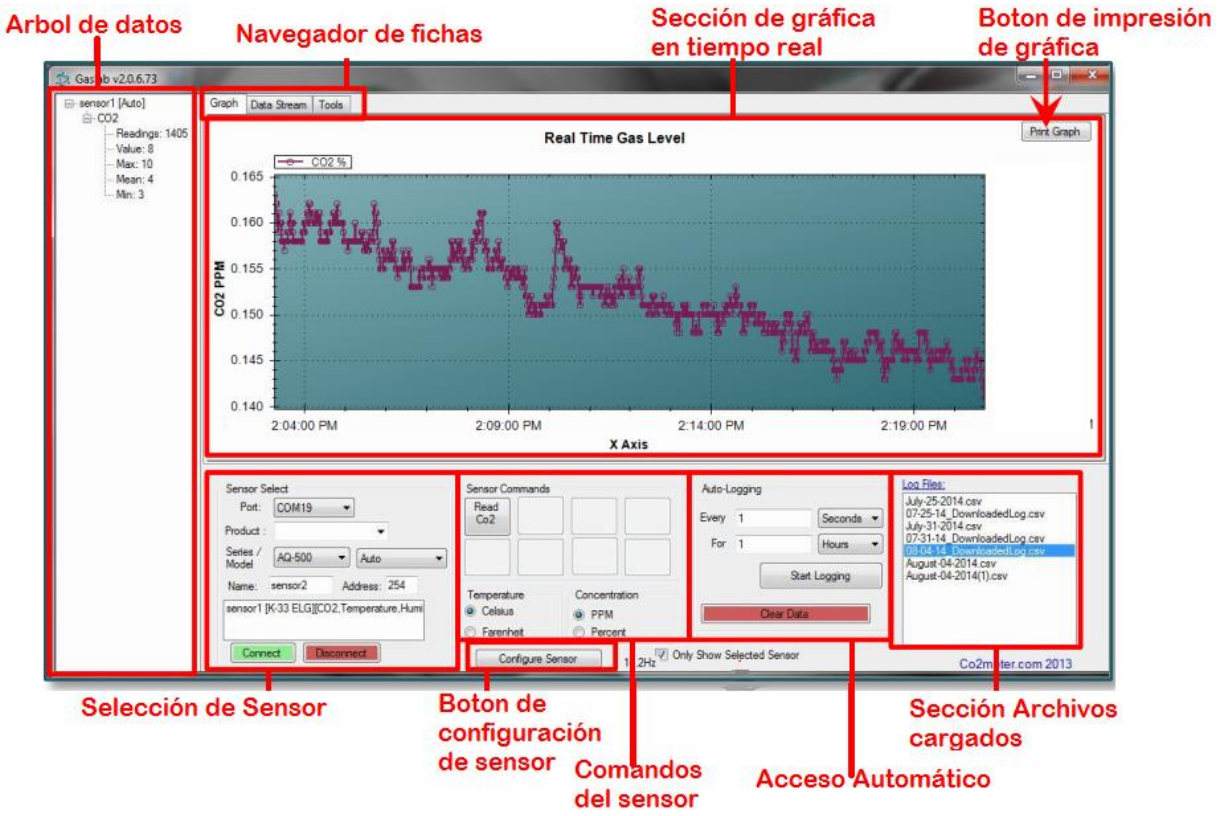


Figura 3.8 Pantalla principal GasLab.

Este software nos permite seleccionar entre los diferentes sistemas desarrollados por la compañía, así como el puerto por donde se conectara el sistema. Además se pueden seleccionar las magnitudes que se desean mostrar en tiempo real en la gráfica, así como las unidades en que aparecerán éstas, haciendo ajustes automáticos para cada parámetro.

Permite configurar los intervalos de tiempo en los que se efectua el muestreo para el sensor de dióxido de carbono[8], debido a que éste necesita un tiempo de recuperación del sensor de luz infrarroja para evitar sobrecalentamiento.

En la interfaz del software se pueden hacer respaldos de la información de las mediciones que se están efectuando en tiempo real, de las tres distintas magnitudes, de manera individual o las tres en conjunto en un solo archivo, también permite cargar archivos previos almacenados en la computadora.

La aplicación admite hacer una configuración rápida del sensor de CO_2, permitiendo calibrar a éste usando como referencia 400ppm que existen en un ambiente al aire libre, permite activar y desactivar el algoritmo de autocalibración, permite ver estados de entradas y salidas que tiene el módulo de medición, así como poder ver el estado de la alimentación que tiene la tarjeta.

No obstante que el software GasLab nos brinda la posibilidad de trabajar con el módulo K-33 BLG de una manera sencilla y amigable, tiene el inconveniente de que la configuración determinada previamente por el usuario se mantiene indefinida no importando que el modulo esté o no comunicado con la computadora. Por tal razón se decidió trabajar con la otra aplicación que ofrece el fabricante denominada DAS.

Esta interfaz permite hacer una comunicación directa con el módulo K-33 BLG, sin el inconveniente que presenta el software GasLab[8]. Cada vez que se conecte o desconecte el módulo o incluso que se presente un mal funcionamiento en éste, la interfaz detecta que el modulo K-33 BLG no se encuentra y automáticamente deshabilita todas las funciones que habían sido programadas, esto permite detectar fallos en la comunicación. En la figura 3.9 se observa la pantalla general del software DAS.

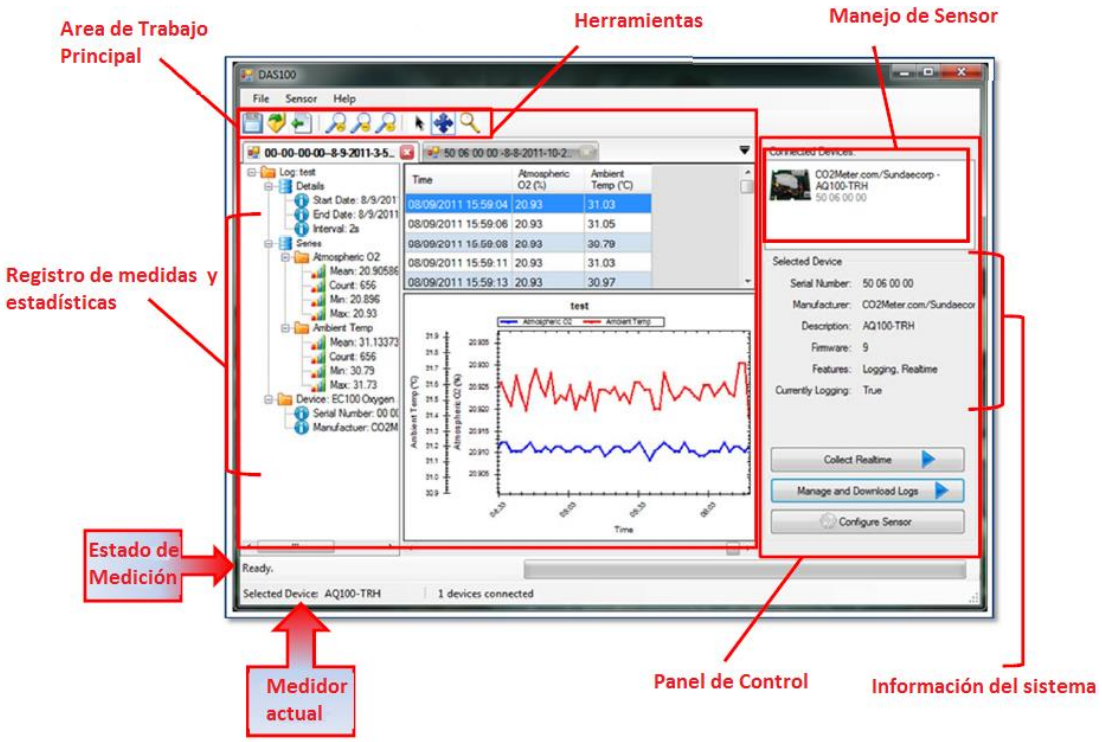


Figura 3.9 Pantalla principal DAS.

El software DAS es muy similar al GasLab, manejo de señales haciendo registros con las magnitudes medidas de manera independiente o en conjunto, el software DAS detecta automáticamente el sensor por ello no es necesario indicar la serie y el modelo. También maneja las gráficas en tiempo real y va haciendo autoescalas en sus ejes “x” y “y”, cada vez que se realiza una lectura de las distintas magnitudes se van graficando.

Permite también trabajar con el logaritmo de autocalibración (ABC), además de que da la opción de poder calibrar el sensor cuando se desee tomando como base 400ppm que se encuentran en el ambiente. Haciendo calibración automáticamente en la pantalla de configuración entrega un valor de dióxido de carbono que se encuentra en el ambiente, además de la humedad relativa (RH), temperatura y el nivel de carga que está alimentando al módulo.

Una parte importante que tiene el software DAS comparado con el software GasLab, es que esta aplicación tiene dentro del menú de configuraciones, la opción de restablecer el módulo a sus parámetros de fábrica.

3.2. Sistema de medición e inyección de dióxido de carbono en una cámara.

Una vez que se tuvo el conocimiento necesario para el funcionamiento y manejo del módulo de medición K-33 BLG, nos propusimos desarrollar un sistema que fuera totalmente autónomo de una computadora, que fuera capaz de medir el dióxido de carbono, temperatura y humedad dentro de una cámara de incubación para células que serán expuestas a campos magnéticos, registrando las distintas magnitudes, así como controlando la inyección del gas, además el sistema avisará al usuario cuando algún parámetro no esté en los rangos que se han especificado.

La figura 3.10 muestra el diagrama a bloques del sistema autónomo desarrollado para la medición de temperatura, humedad relativa y el manejo del dióxido de carbono dentro de una cámara.

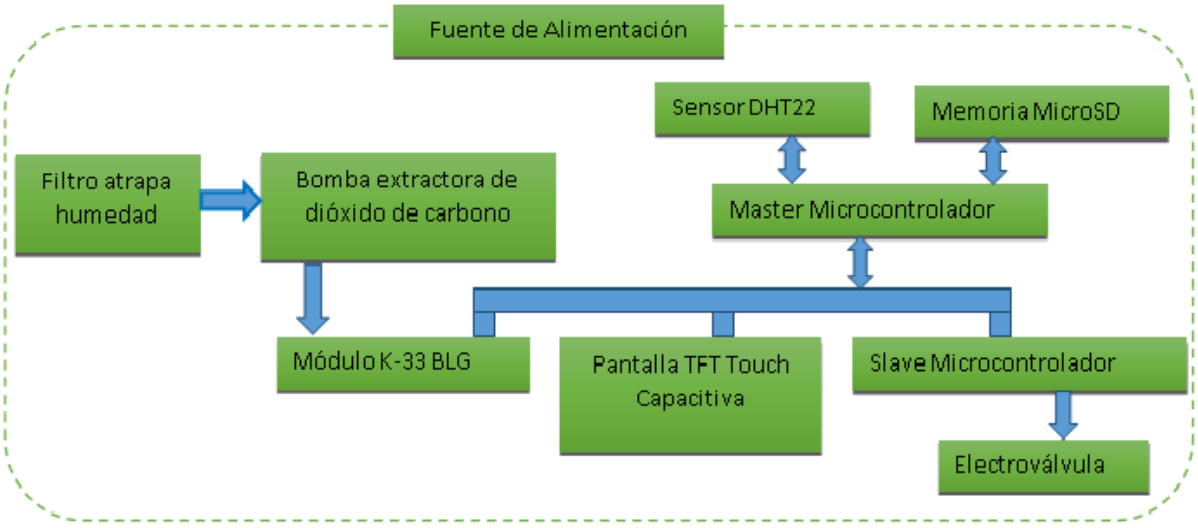


Figura 3.10 Diagrama a bloques del sistema autónomo.

Debido a que el módulo K-33 BLG maneja dos tipos de comunicación, I^2C y el UART (descritos en la sección 3.2.9), y la pantalla TFT Touch que se emplea también necesita de dos protocolos de comunicación seriales para su funcionamiento siendo estos I^2C y SPI (descritos en la sección 3.2.9), se decidió dar una solución mediante un sistema Maestro-Esclavo a la etapa de control en el manejo del gas de dióxido de carbono.

3.2.1. **Filtro de humedad.**

Se necesitara un filtro que atrape la humedad, ya que la extracción de dióxido de carbono que se estará haciendo, será de una cámara con ambientes de humedad a niveles altos, de 95% a 100%, por lo que se necesita tener un filtro que pueda atrapar la humedad para que ésta no llegue al módulo de medición y vaya a dañar el sistema electrónico de módulo.

En la figura 3.11 se observa el tipo de filtro que se usará para atrapar la humedad que acompañará al dióxido de carbono en el momento en que se esté extrayendo de la cámara.

Figura 3.11 Filtro CM-0112 de la compañía CO2meter.

El filtro diseñado contiene en su interior un material que permite absorber la humedad, el material considerado para hacer ésta función fue mediante hilos de estambre, se consideró este material por el contenido en lana que presenta en su composición.

3.2.2. **Bomba extractora de dióxido de carbono.**

Debido a que se estará controlando el dióxido de carbono del interior de una cámara, es necesario tener un extractor del gas para efectuar la medición de la concentración del interior. En la figura 3.12 se muestra una micro-bomba que actúa como extractor del gas, el modelo es CM-0111 de la compañía CO2meter.

Figura 3.12 Micro-bomba CM-0111 de la compañía CO2meter.

La bomba CM-0111 brinda una alimentación máxima de 500ml/min, con una presión de 380 a 400 mbar [9].

3.2.3. **Almacenamiento de datos de los parámetros del medio.**

Una de las características importantes del sistema es el almacenamiento de los datos de temperatura, humedad relativa (RH) y de la concentración de dióxido de carbono en un archivo *.TXT. El almacenamiento se realiza en una memoria µSD con una capacidad de 2 GB.

La tarjeta para la memoria µSD emplea el protocolo de comunicación SPI, consta cuatro líneas de comunicación y dos líneas de alimentación. Este protocolo será implementado a través de un sistema Arduino mega, que será el maestro de todo el sistema.

El módulo de la memoria µSD poseen 6 pines de conexión, de los cuales, al trabajar con el protocolo SPI 4 líneas son de comunicación: *Data In* (MOSI), *Data Out* (MISO), *Chip Select* (CS) y *clock* (CLK); 1 es línea de alimentación, 1 para tierra, tabla 3.1. El rango de voltaje de operación permitido es de 4.5 ~ 5.5V [10].

PIN	Nombre	Descripción
1	CS	Activación de tarjeta
2	SCLK	Reloj
3	MOSI	Comandos de datos desde el host
4	MISO	Datos hacia el host
5	VDD	Alimentación (5V)
6	VSS	GND

Tabla 3.1 Pines de conexión de la memoria µSD mediante comunicación SPI.

Para la escritura y lectura de la memoria µSD se adquirió el módulo de evaluación, figura 3.13, el cual cuenta con todas sus salidas en header de 6 pines, cuenta además con el socket para la memoria.

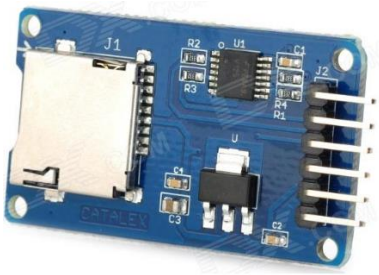


Figura 3.13 Módulo de lectura y escritura para la memoria µSD.

Para generar un archivo tipo *.txt es necesario trabajar en el formato fat16 en la memoria. La librería que trabaja con este formato se obtuvo de la página del proveedor del módulo de la memoria µSD [10].

3.2.4. **Sensor de temperatura y humedad relativa DHT22.**

DHT22 Se tomo la decisión de utilizar este sensor por la estabilidad sobresaliente que muestra a largo plazo, un bajo consumo de energía y que es completamente intercambiable en caso de una eventual falla.

La señal de salida digital de este sensor esta calibrada, asegurando que en la aplicación que se desarrollará será estable y en consecuencia confiable. Para el que el sensor pueda realizar su trabajo se necesita ser conectado a microcontroladores que manejen sus registros de 8 bits [1].

Cada uno de los sensores de este modelo esta compensado en temperatura y calibrado en una cámara de precisión, los coeficientes de calibración son guardados y programados en una memoria OTP (One Time Programmable), cuando el sensor está realizando la detección de la temperatura, cita los coeficientes que están en la memoria [1].

El sensor cuenta con 4 pines, uno para su alimentación, otro será para la referencia, un tercero será el bus datos de salida, y el último pin no se conecta.

En la tabla 3.2 se presenta las especificaciones técnicas del sensor.

Modelo	DHT22
Alimentación	3.3-5.5V DC
Señal de salida	Señal de salida digital vía 1-bus de conexión
Elemento de censado	Condensador polímero de humedad
Rango de operación	Humedad 0-100%RH; Temperatura -40~80 Celsius
Precisión	**Humedad ±2%RH**(Max ±5%RH); Temperatura ±0.5Celsius
Resolución o sensibilidad	Humedad 0.1%RH; Temperatura 0.1 Celsius
Repetitibilidad	Humedad ±1%RH; Temperatura ±0.2Celsius

Tabla 3.2 Tabla de especificaciones del sensor DHT22.

En la figura 3.14 se muestra el diagrama eléctrico del sensor DHT22 y el microcontrolador maestro.

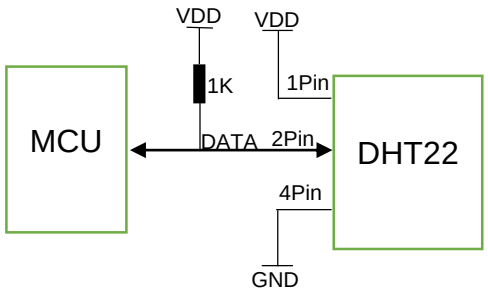


Figura 3.14 Diagrama eléctrico DHT22.

La comunicación entre el microcontrolador y el sensor se hace por medio de un solo pin, tal como se ve en la figura 3.14, el protocolo de que se debe realizar entre el microcontrolador y el sensor DHT22 es el siguiente:

El microcontrolador manda una señal de comienzo al sensor, el sensor cambia de un estado de “standby” a un estado de comienzo. Cuando el microcontrolador finaliza la señal de comienzo, el sensor mandará inmediatamente una señal de 40-bits, reflejando la humedad relativa y la temperatura al microcontrolador. Sin una señal de comienzo por parte del microcontrolador, el DHT22 no mandara ningún tipo de señal. El sensor DHT22 cambiará al estado “standby” cuando terminé de mandar todos los bits al microcontrolador, el sensor esperara una nueva petición desde el microcontrolador [1].

Los datos de salida son de 40 bits, la descripción es estos bits es descrita en el siguiente ejemplo.

El microcontrolador recibirá 40 bits desde el sensor DHT22;

0000 0010 1000 1100 0000 0001 0101 1111 1110 1110

16 bits son datos RH 16 bits son datos T 8 bits son una suma de prueba

Se convierten los primeros 16 bits de RH desde un sistema binario a un sistema decimal.

0000 0010 1000 1100 → 652

Posteriormente se divide entre 10 el valor obtenido a partir del sistema decimal y eso dará como resultado la humedad relativa que entrego el sensor.

RH=652/10=65.2%RH

Lo mismo se hace para los siguientes 16 bits pertenecientes a los datos de la temperatura que es entregada en grados Celsius.

0000 0001 0101 1111 → 351

T=351/10=35.1℃

En el caso de la temperatura, cuando el bit más significativo se encuentra en uno, indica que la temperatura se encuentra por debajo de los 0 grados Celsius.

1000 0000 0110 0101, T= minus 10.1℃

En el caso de los últimos 8 bit de comprobación se suman los datos de la humedad y la temperatura, siendo divididos en tramas de 8 bits y dando el resultado en una variable extra, corroborando los datos obtenidos [1].

0000 0010+1000 1100+0000 0001+0101 1111=1110 1110

La librería que permite implementar el protocolo antes mencionado sobre el sensor está disponible tanto en la página web del proveedor[11], como en la página web de la placa Arduino [12].

3.2.5. **Sistema de medición de dióxido de carbono para el sistema autónomo.**

El sistema de medición de dióxido de carbono K-33 BLG será descrito en el modo en que fue utilizado para el desarrollo del sistema general independiente de la computadora.

Anteriormente se estuvo trabajando bajo el protocolo de comunicación UART entre la computadora y el módulo K-33 BLG, para el trabajo que se decidio hacer en el sistema autónomo, que es un sistema maestro – esclavo (Master-Slave), con la comunicación UART no se podía realizar tal trabajo, el modulo cuenta con otro protocolo de comunicación, I^2C, que permite desarrollar tal sistema (Master-Slave).

En la figura 3.15 se muestra el área dentro del módulo de medición de CO_2 donde están localizados los pines correspondientes para poder realizar el protocolo de comunicación I^2C.

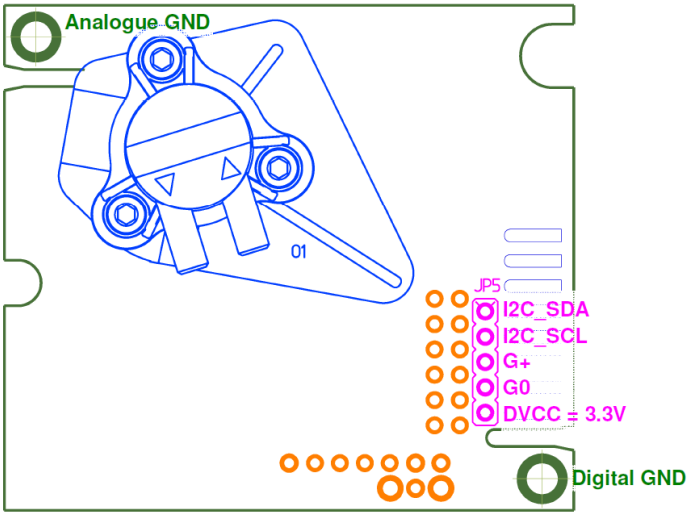


Figura 3.15 Módulo K-33 BLG, pines de protocolo I^2C.

El sensor opera en una tasa de 100kBit/sec y está reservado para actuar como esclavo [7], la dirección a la que responde como esclavo es 0x68 [13].

El protocolo de comunicación I^2C (descrito en la sección 3.2.9), sólo necesita dos vías para poder realizar su comunicación, la primer vía es SCL (figura 3.14 I2C_SCL), que es la vía de reloj, esta vía es generada por el maestro, y todos los esclavos están sincronizados por ella. La segunda vía es SDA (figura 3.14 I2C_SDA), esta vía es bidireccional, los datos tanto del maestro como los del esclavo son enviados por esta vía, los esclavos solamente pueden mandar cuando el maestro se los ordene. El protocolo será descrito en la sección 3.3.9 de éste mismo apartado.

3.2.6. **Interfaz de usuario, pantalla TFT touch capacitiva.**

La pantalla que se adquirió para ser la interfaz de usuario, es una pantalla de 2.8 pulgadas, modelo TFT touch capacitiva, ya que existe la versión resistiva, la gran ventaja que tiene la versión capacitiva se debe a que la capacitiva no necesita ser calibrada, la pantalla es la compañía Adafruit.

Esta desarrollada para trabajar con placas Arduino y con placas Raspberry pi, maneja 18 bits para colores, lo que da 262,000 tonos diferentes, manejando 240x320 pixeles cada uno de manera individual, puede detectar la presión del dedo en cualquier lugar de la pantalla [14].

La pantalla maneja dos protocolos de comunicación, el primero es el protocolo de comunicación serial SPI, este protocolo sirve para mandar cualquier objeto que se dese pintar en la pantalla, el segundo protocolo que usa la pantalla es el I^2C (descrito en la sección 3.2.9), este protocolo sirve para estar leyendo los datos de la pantalla, es decir se leen las coordenadas de donde se presionó en la pantalla por medio de este protocolo, la dirección de la pantalla para I^2C es 0x38 [14].

La pantalla maneja 7 líneas para comunicación y dos de alimentación. De las 7 líneas de comunicación 5 son para el protocolo SPI (descrito en la sección 3.2.9), y dos para el protocolo I^2C en su versión capacitiva, en la tabla 3.3 se encuentra la descripción de los pines.

ICSP SCLK, Arduino Mega	Reloj SPI
ICSP MISO, Arduino Mega	Entrada Maestro-Esclavo salida SPI
ICSP MOSI, Arduino Mega	Salida Maestro-Esclavo salida SPI
CS, Pin digital #10 Arduino Mega	Selección de chip
DC, Pin digital # 9 Arduino Mega	Selección de comandos para los datos
SDA, Pin analógico 4 Arduino Mega	Datos de I^2C
SCL, Pin analógico 5 Arduino Mega	Reloj de I^2C
Reset, Pin Arduino Mega	Resetea pantalla y Arduino Mega
VCC, Pin 5VCC Arduino Mega	La pantalla se alimenta con 5 volts directo de la placa Arduino Mega
GND, Pin GND Arduino Mega	Referencia de la placa Arduino Mega

Tabla 3.3 Tabla descriptiva de pines en pantalla touch.

Para desarrollar el sistema autónomo utilizamos la placa arduino mega, ésta placa tiene dos formas de poder hacer la comunicación SPI, el primer modo de conexión es por medio de los pines 50, 51, 52 y 53, y el segundo modo de conexión es a través de un bus exclusivo para esta comunicación este bus se de comunicación se llama ICSP.

En la figura 3.16 se muestra la localización de los modos para realizar el protocolo de comunicación SPI, y en donde se efectuarán las conexiones de la pantalla, para los dos protocolos de comunicación.

Figura 3.16 Pines donde llegaría la pantalla hacer sus protocolos de comunicación, I^2C y SPI.

La pantalla necesita de diferentes librerías para poder funcionar, estas librerías las proporciona la compañía Adafruit, tiene librerías de ejemplos para comenzar con el uso de la pantalla, esta librería se llama Adafruit_ILI9341. Tiene otras dos librerías que fueron las que se emplearon en el manejo de la pantalla, una de ellas nos permite hacer gráficos dentro de la pantalla, letras, líneas, círculos, cuadrados, etc., esta librería se llama Adafruit_GFX. Otra librería que se uso fue la librería que es capaz de controlar la parte táctil de la pantalla en su modo capacitivo, esta librería se llama Adafruit_FT6206, las librerías se pueden encontrar en la página de Adafruit [14].

La librería Adafruit_GFX permite imprimir letras dentro de la pantalla, pero fue necesario generar nuestras propias librarias para diferentes tamaños de letra y distinto tipo de letras, ya que la librería Adafruit _GFX no permite usar otro tipo de letras y maneja solo algunos tamaños, por eso se tuvieron que desarrollar distintos tipos de letras y tamaños.

Se programarón distintas pantallas para la interfaz de usuario, en la figura 3.16, se observa la pantalla principal de la interfaz de usuario.

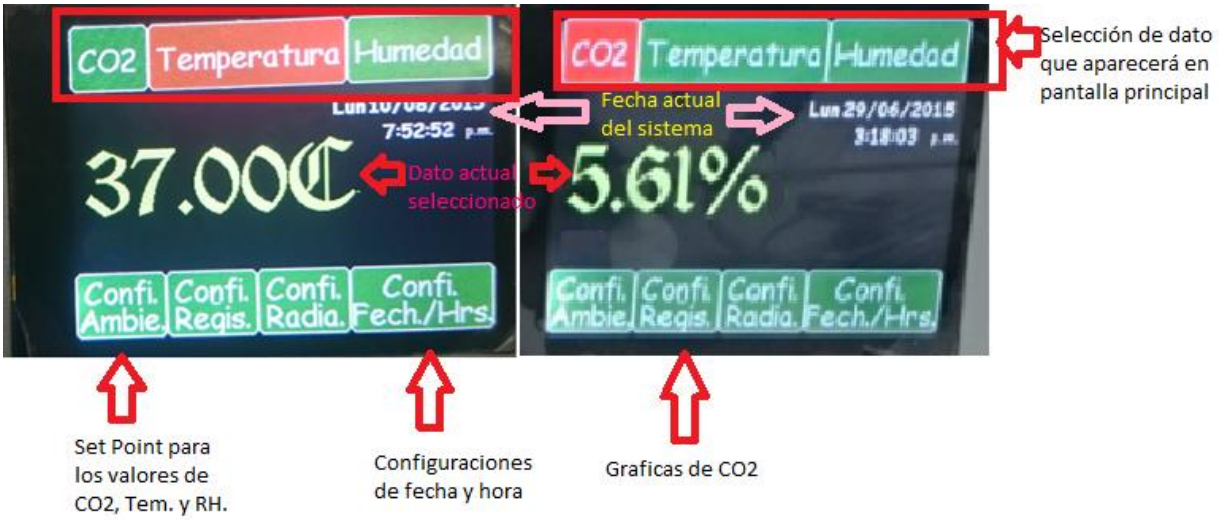


Figura 3.17 Pantalla principal de la interfaz de usuario.

Las pestañas de arriba en la figura 3.17, son para seleccionar el dato que se desea visualizar en la pantalla principal, se tiene también la última magnitud leída por el sistema dependiendo del datos que se haya seleccionado en la pestaña de arriba. Por otro lado se tiene la fecha actual del sistema, se puede configurar la fecha y hora en el botón que se encuentra en la parte inferior de la derecha, mandándonos a otras pantallas donde se podrá hacer ésta configuración.

El sistema también se puede configurar para emitir una alarma de advertencia cuando las magnitudes, temperatura y humedad relativa no se encuentren en el valor que el usuario desea. En el mismo botón de la parte inferior también se puede hacer la configuración de los niveles de dióxido de carbono que se desean dentro de la cámara, siendo ésta única magnitud controlada por el sistema desde la interfaz, y cuando los niveles de dióxido de carbono se encuentren por debajo del nivel deseado por el usuario después de 6 minutos, el sistema emitirá una señal de alarma para indicar al usuario lo que sucede dentro de la cámara.

La interfaz es capaz de graficar el parámetro del CO2 que ha estado registrando el sistema dentro de la cámara, siendo la gráfica una aproximación ya que no se tiene la resolución necesaria para poder desplegar los datos exactamente, pero si el usuario

está interesado en la exactitud de los datos, puede adquirirlos de la memoria µSD mencionada en el apartado 2.2.3.

3.2.7. **Mecanismo de control.**

Para poder controlar el paso del dióxido de carbono, se usó una electroválvula de la compañía Festo, modelo MSZD-3 de 24 Volts, a 1W de potencia, es una electroválvula monoestable.

La electroválvula será accionada por el microcontrolador esclavo, el microcontrolador maestro habrá enviado el nivel de dióxido de carbono que el usuario desea tener dentro de la cámara y también estará enviando los niveles de dióxido de carbono que hay dentro de la cámara en esos momentos, esto provocará que el microcontrolador esclavo accione o no accione la electroválvula para el control del gas. La activación que realiza el microcontrolador esclavo a la electroválvula se hace por medio del optoacoplador 4N25 [15] , utilizado este como elemento de aislamiento y protección del sistema da control.

En la figura 3.18 se ve el modelo de electroválvula que se utilizará, y será colocada en la parte posterior de la cámara entre el tanque contenedor del gas y la cámara.

Figura 3.18 Modelo de la electroválvula y parte posterior de la cámara donde se colocará la electroválvula.

3.2.8. **Fuente de alimentación del sistema.**

El manejo de distintos dispositivos, que se alimentan con diferentes niveles de voltaje generó la necesidad de diseñar una fuente de voltaje para alimentar todo el sistema.

Se requieren salidas de voltaje de 9 Volts y 400 mA para los microcontroladores, la pantalla y el sensor DHT22 que se alimentan por medio de los microcontroladores, una salida de voltaje de 7 Volts y 100 mA para la alimentación del módulo de medición de dióxido de carbono, una salida de 5 Volts y 100 mA para la bomba que extrae el gas de la cámara, y una salida de 24 Volts y 40 mA para la activación de la electroválvula.

Se recomienda tener voltajes de tres volts por arriba del voltaje que se desea regular mediante el circuito LM317 [16]. Sumando los 3 Volts a cada una de las salidas,

tendremos tres devanados en el secundario, uno será de 12 Volts que alimentara a los microcontroladores, pantalla, sensor DHT22 y el módulo de medición de dióxido de carbono con una corriente de 600 mA, da una potencia de 7.2 Watts. El otro devanado que tendrá una salida en el secundario es de 9 Volts, para la alimentación de la bomba extractora de gas, con una corriente de salida de 100 mA, con una potencia de 0.9 Watts. Y el último devanado será de 30 Volts con una corriente de 100 mA para la activación de la electroválvula, dando una potencia de 3 Watts. Dando una potencia total de 11.1 Watts. Para el diseño del transformador se redondeó a 12 Watts. En el primario será la misma potencia, por lo tanto se obtuvo la corriente que se consumirá para poder obtener el valor del fusible, con la siguiente formula:

$$V * I = W \rightarrow 120 * I = 12\ Watts\ \rightarrow I = \frac{12\ Watts}{120\ Volts} = 100\ mA$$

Siendo el valor del fusible de 0.1 A.

En la figura 3.19 se observa el diseño total de la fuente.

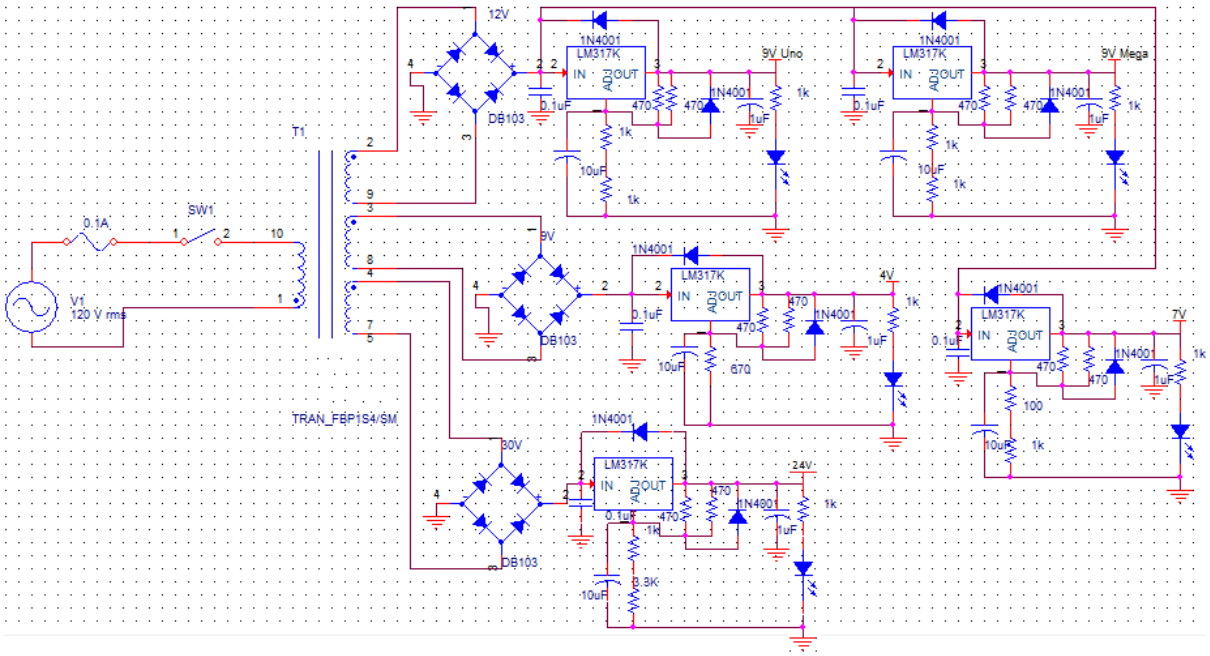


Figura 3.19 Esquemático de la fuente de alimentación del sistema.

Los puentes de diodos que se utilizaron para el diseño de la fuente son DB103 soportan un 1 Amper en su salida [17].

Los capacitores de 0.1uF que están en la entrada de los reguladores LM317 así como los capacitores de 1uF que están en la salida son para limitar el rizado, y estos valores los sugiere el proveedor del circuito [16].

Los diodos 1N4001, tanto los que van de los pines 3 a 2 del regulador LM317, así como los que van de las resistencias al pin 3 del regulador, son diodos de protección para las cargas que se vayan a colocar a la salida de cada uno de los reguladores, este diseño se obtuvo de un foro de diseño de fuentes[18].

3.2.9. **Microcontrolador Maestro.**

El control de todo el sistema es llevado a cabo por la placa de desarrollo Arduino Mega con un microcontrolador de Atmega 1280 de la compañía Atmel, siendo un microcontrolador de 8 bits en sus registros, tiene puertos de comunicación UART, SPI, ICSP, I^2C, tiene 54 entradas/salidas digitales, 14 PWM entradas/salidas, 16 entradas analógicas de 10 bits de resolución, una memoria Flash de 256 kB, un reloj de 16Mhz [19, 20]. Se decidió trabajar con este placa Arduino Mega por la cantidad de memoria Flash que contiene, el manejo de los distintos puertos de comunicación que maneja , el manejo de sus registros de 8 bits, y la cantidad de librerías que tiene desarrolladas la plataforma Arduino.

En la figura 3.20 muestra el diagrama eléctrico a bloques de todo el sistema desarrollado que maneja el microcontrolador maestro, las conexiones entre el módulo de medición de CO2, el sensor DHT22, la pantalla TFT touch capacitiva, la memoria µSD y el microcontrolador esclavo.

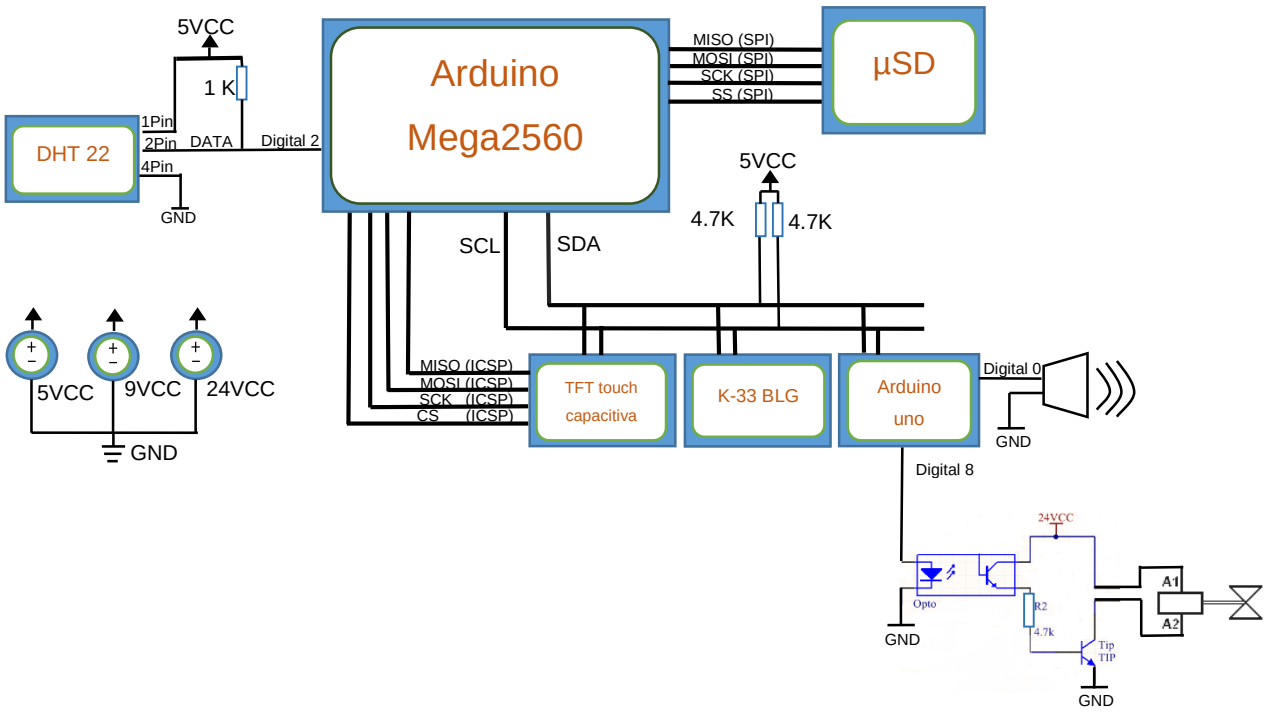


Figura 3.20 Diagrama eléctrico a bloques del sistema general.

En la figura 3.20 se observa las distintas conexiones que se dieron con el microcontrolador maestro para poder implementar los protocolos de comunicación con los diferentes dispositivos electrónicos.

El protocolo de comunicación I^2C fue desarrollado hace 23 años por la compañía Philips Semiconductor con el propósito de conectar una gran cantidad de dispositivos usando una menor cantidad de cables en la conexión de su bus con otros dispositivos, teniendo un dispositivo como maestro y los demás como esclavos. El dispositivo maestro manda el dato dirección a través de un solo cable, bit por bit, siendo de 8 bits la dirección a mandar, por lo tanto se puede tener $2^8 = 256$ dispositivos conectado, sin que exista una misma dirección entre ellos, seleccionando uno a la vez. Una de la reglas es que debe ser toda la comunicación síncrona entre los dispositivos esclavos

y el dispositivo maestro, siendo generada la comunicación síncrona por el dispositivo maestro[21].

La taza de velocidad con la que trabaja I^2C en el envió de datos va desde 100 kHz (‘Industrial’ y SMBus), I^2C 400 kHz y I^2C (‘High Speed mode’) 3.4 MHz[21]. Para la aplicación que estaremos trabajando será de 100 kHz (100 kbps).

Se ha mencionado anteriormente el bus maneja dos líneas que son requeridas para hacer la comunicación, siendo estas las de datos (SDA) bi-direcional y la del reloj (SCL) que permitirá sincronizar todos los dispositivos conectados en bus. En la figura 3.21 se muestra el modo que debe conectarse el bus de I2C de acuerdo a las especificaciones de Philips Semiconductor.

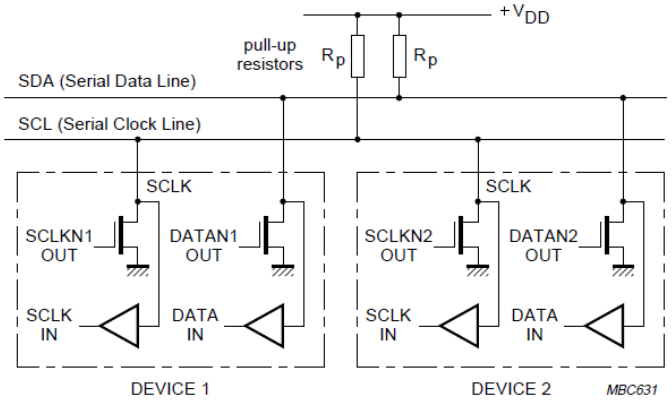


Figura 3.21 Conexión del bus de comunicación I2C.

Se puede observar en la figura 3.21 que el bus siempre estará alto activo, esto indicara que el bus esta libre y que se podrá hacer la comunicación entre los dispositivos maestro-esclavo.

La resistencias que son utilizadas como pull up en la conexión fueron calculadas con la fórmula que proporciona Philips Semiconductor, suponiendo que la salida de transistor maneje una corriente de 3mA, dara una resistencia mínima de pull-up de:

$$R = \frac{V_{DDmin} - 0.4V}{3\ mA} \qquad V_{DDmin} = 5V\ (\min 4.5\ V), R_{min} = 1.3\ k\Omega$$

El protocolo de comunicación I^2C tiene sus condiciones para transmitir los datos, y se describen a continuación:

- Condición **Start y Stop**: son condiciones que genera el maestro a los esclavos. La condición Start se dará cuando la línea de SDA pasa de un estado alto a un estado bajo, en cuanto que la línea SCL se encuentre en un estado alto. La condición Stop se dará cuando se pase de un estado bajo a un estado alto en la línea de SDA mientras que SCL esté en alto. Cuando se genera la condición de Start indicará a los dispositivos esclavos que el bus estará ocupado, y cuando se genera la condición de Stop es un aviso de que el bus estará liberado. En la figura 3.22 se observa las condiciones Start y Stop generadas por el maestro.
-

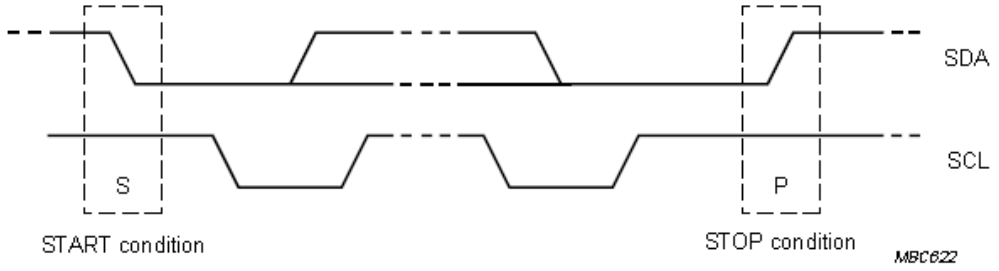


Figura 3.22 Condiciones Start y Stop en el protocolo de comunicación I2C.

- Los datos válidos serán durante el periodo que la línea SDA se encuentre estable y la línea de SCL correspondiente al reloj, se encuentre en un estado alto. El cambio entre de un estado a otro en la línea de datos SDA, solo se podrá dar en el momento que la línea de reloj SCL este en estado bajo. En la figura 3.23 se muestra las condiciones para la trasmisión de datos.

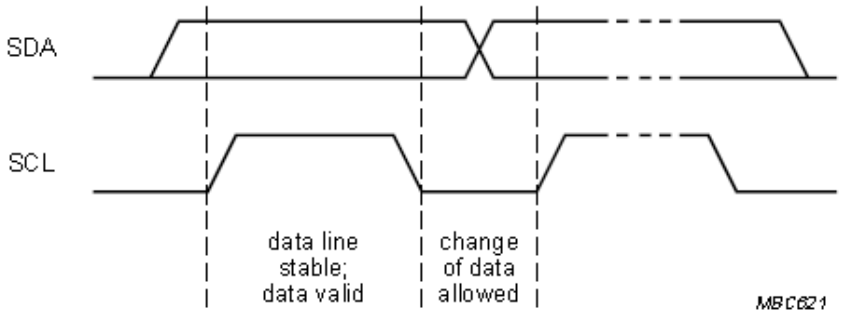


Figura 3.23 Condiciones para la transmisión de datos por el protocolo I2C.

- Cada byte que se ponga en la línea SDA deberá tener una longitud de 8-bits. Este número de byte a transmitir no está restringido, cada byte que se transmita estará seguido de un bit llamado “acknowledge”, para poder diferenciar entre los diferentes bytes, el bit acknowledge en SDA deberá ser bajo, la línea del reloj SCL deberá estar en alto. Los bits que se transmiten de un byte van del más significativo (MSB) al menos significativo (LSB). Si los esclavos se ven interrumpidos (interrupción interna) durante el proceso de transmisión, mantienen la línea SCL en bajo forzando al maestro a que esté en condición de espera. En la figura 3.24 se observa las condiciones para transmisión de bytes.

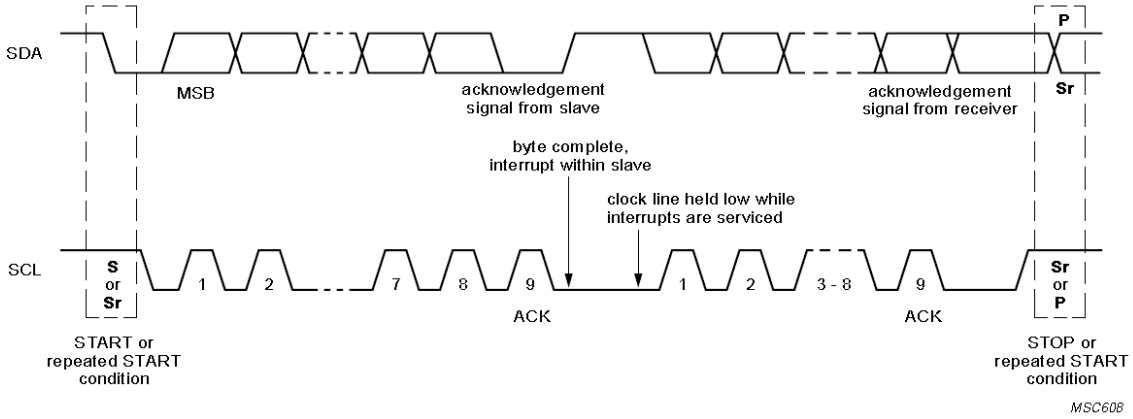


Figura 3.24 Transmisión de diferentes bytes en el protocolo de comunicación I2C.

- Después de haberse generado la condición de Start a uno de los esclavos siendo esta de 7-bits, seguida de un bit de lectura/escritura ($R/\overline{W}$) indicando un

nivel bajo activo para escritura y un nivel alto activo para lectura, después viene el dato correspondiente al acknowledge de la dirección, seguido de los datos que llegaran al dispositivo que contenga la dirección, previamente mencionada finalizando con un acknowledge después de cada byte mandado. Al finalizar un dato trasferido siempre será precedido de un Stop. El dispositivo maestro puede generar varias condiciones Start sin liberar el bus I^2C a otros dispositivos esclavos, incluso puede generar condiciones de lectura/escritura a dispositivos esclavos que mantiene una actual comunicación. En la figura 3.25 se observa todas estas condiciones que se pueden generar entre el dispositivo maestro y esclavo[21].

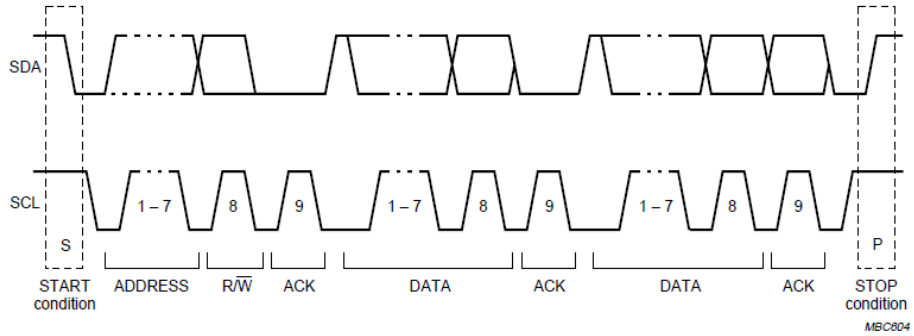


Figura 3.25 Condiciones para transmisión de dirección y datos en el protocolo I2C.

El protocolo de comunicación SPI, fue desarrollado por la compañía Motorola. SPI (Serial Peripheral Interface) es un protocolo de comunicación full dúplex serial síncrono que cuenta con cuatro cables de conexión llamados: SCLK (Serial Clock) reloj que será generado por el dispositivo maestro de manera serial, MOSI (Master On Slave In) dato que va desde el maestro hasta el esclavo, MISO (Master In Slave On) dato que va desde el esclavo hasta el maestro, SS (Slave Select) permite seleccionar el esclavo deseado[21].

Para la conexión entre el microcontrolador maestro y los dispositivos esclavos que se desean conectar a este protocolo se necesitaran “3+n” hilos de conexión, donde “n” son la cantidad de dispositivos a conectar. En la figura 3.26 se muestra la cantidad de

cables que se necesitan para poder realizar las conexiones entre el maestro y tres dispositivos esclavo[22].

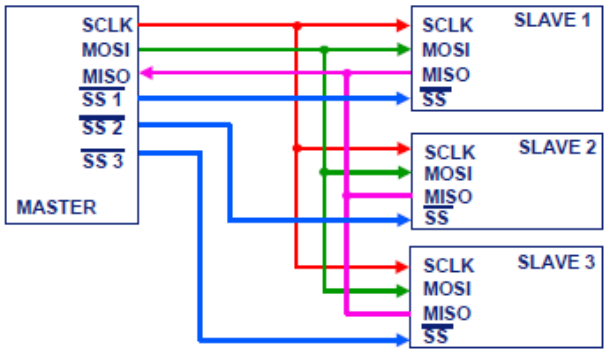


Figura 3.26 Conexiones en el protocolo SPI entre un maestro y tres esclavos.

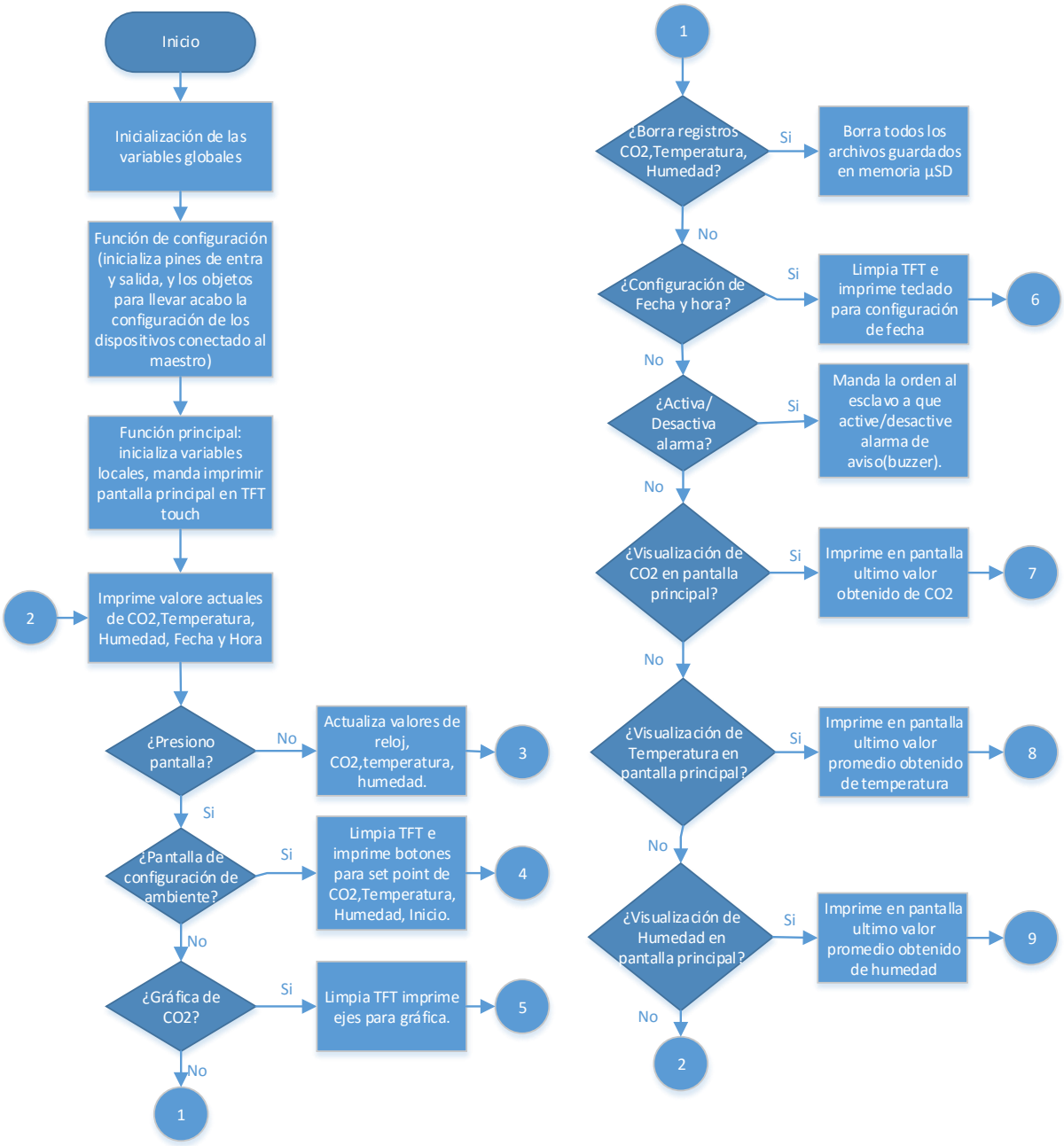
En este protocolo de comunicación se transmiten de 8 a 16 bit de manera serial. Tres de las líneas antes mencionadas llevan datos al bus. Cada dispositivo puede actuar en la transmisión y recepción de datos. Dos de las tres líneas pueden transmitir datos (en una sola dirección) y la tercera es el reloj serial. Algunos dispositivos pueden transmitir mientras que otros se encuentran recibiendo. Generalmente los dispositivos que son capaces de transmitir datos también son capaces de recibir datos[22].

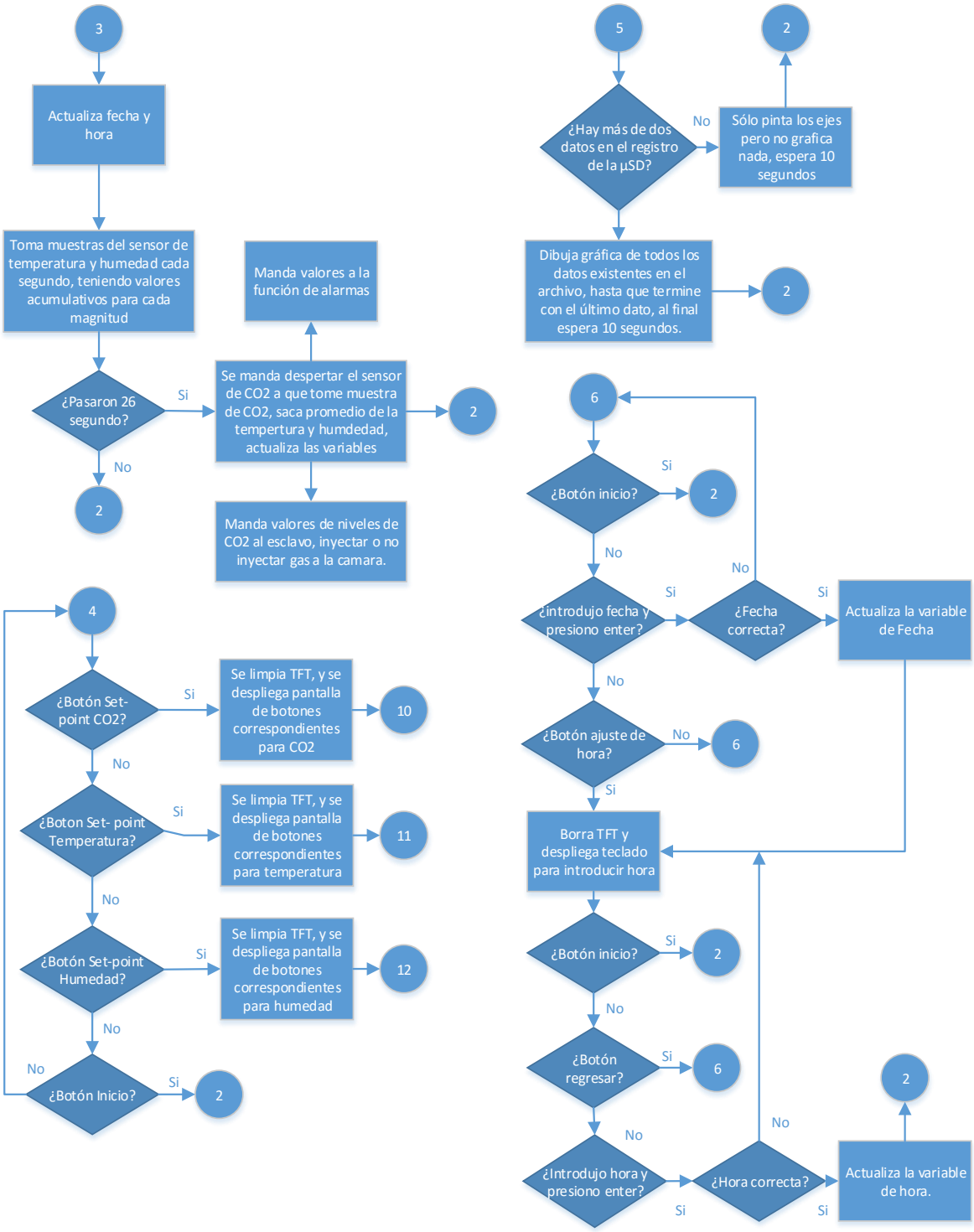
La comunicación y selección entre el dispositivo maestro y el dispositivo esclavo se da mediante una activación generada por el maestro a través de Slave Select (SS), los datos serán transmitidos por el bus, pero solo el esclavo que este seleccionado podrá ver dichos datos[22].

Su taza de transferencia máxima de datos es 1Mbits por segundo siendo transmitidos por la línea MOSI donde el maestro envía y el esclavo recibe, o por la línea MISO donde el maestro recibe y el esclavo envía. Los datos son trasmitidos de manera síncrona, cada bit se transmite en un ciclo de reloj[22].

Una vez descrito los dos protocolos de comunicación que se utilizaron para el desarrollo del sistema, agregando las librerías de estos dos protocolos de comunicación al programa que se desarrolló. A continuación se describe el programa

que se desarrolló a través de un diagrama de flujos que se muestra a través de la figura 3.27. En el Apéndice B encontraremos el código correspondiente al microcontrolador maestro.




3
Actualiza fecha y hora
Toma muestras del sensor de temperatura y humedad cada segundo, teniendo valores acumulativos para cada magnitud
¿Pasaron 26 segundo?
Si
No
2
Se manda despertar el sensor de CO2 a que tome muestra de CO2, saca promedio de la tempertura y humdedad, actualiza las variables
Manda valores a la función de alarmas
Manda valores de niveles de CO2 al esclavo, inyectar o no inyectar gas a la camara.
2
4
¿Botón Set-point CO2?
Si
No
Se limpia TFT, y se despliega pantalla de botones correspondientes para CO2
10
¿Boton Set- point Temperatura?
Si
No
Se limpia TFT, y se despliega pantalla de botones correspondientes para temperatura
11
¿Botón Set-point Humedad?
Si
No
Se limpia TFT, y se despliega pantalla de botones correspondientes para humedad
12
¿Botón Inicio?
Si
No
2
5
¿Hay más de dos datos en el registro de la µSD?
No
Si
Sólo pinta los ejes pero no grafica nada, espera 10 segundos
2
Dibuja gráfica de todos los datos existentes en el archivo, hasta que termine con el último dato, al final espera 10 segundos.
2
6
¿Botón inicio?
Si
No
2
¿introdujo fecha y presiono enter?
Si
No
¿Fecha correcta?
Si
No
Actualiza la variable de Fecha
¿Botón ajuste de hora?
No
Si
6
Borra TFT y despliega teclado para introducir hora
¿Botón inicio?
Si
No
2
¿Botón regresar?
Si
No
6
¿Introdujo hora y presiono enter?
Si
No
¿Hora correcta?
Si
No
Actualiza la variable de hora.
2

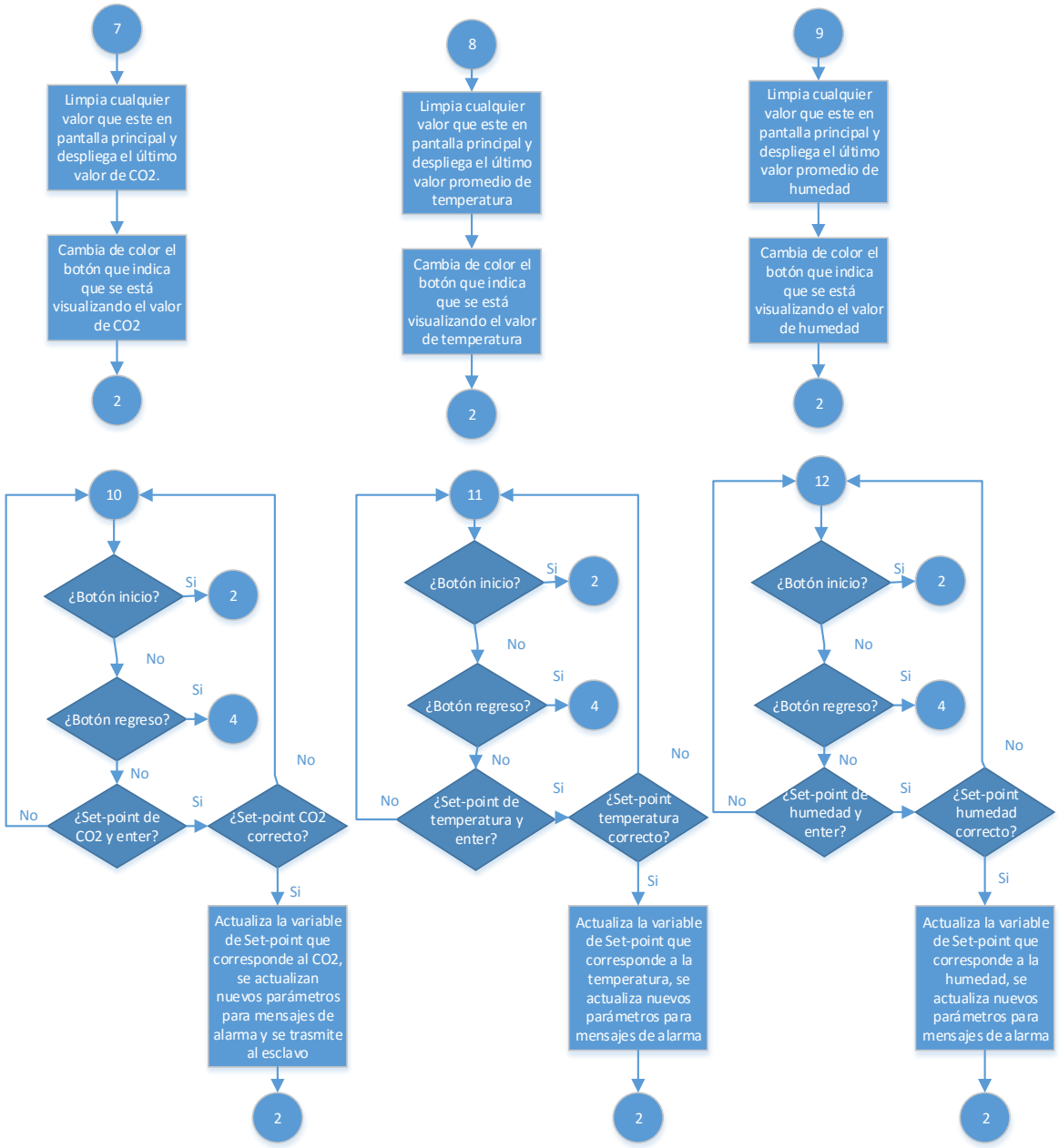


Figura 3.27 Diagrama de flujos del programa en el dispositivo maestro para el sistema desarrollado.

3.2.10. **Microcontrolador Esclavo.**

El microcontrolador que está llevando el control del actuador y la activación de alarmas es la placa de desarrollo Arduino Uno. Esta placa tiene un microcontrolador ATmega328P de la compañía Atmel. Es un microcontrolador que contiene 14 entradas/salidas digitales de las cuales pueden ser 6 utilizadas como salidas PWM,

también tiene 6 entradas analógicas con una resolución de 10 bits. Tiene 2 KB en memoria RAM, 1KB para la memoria EEPROM, y 32 KB en memoria Flash. Además también tiene pines para los protocolos de comunicación SPI. Los pines de salida para el protocolo de comunicación I^2C. Además permite también trabajar con el protocolo comunicación UART. Es un microcontrolador que tiene registros de 8 bits[12, 20].

En la figura 3.20 se muestra el diagrama eléctrico a bloques de todo el sistema maestro-esclavo, de una manera general. En la figura 3.28 se mostrará de manera más detallada el sistema del microcontrolador esclavo.

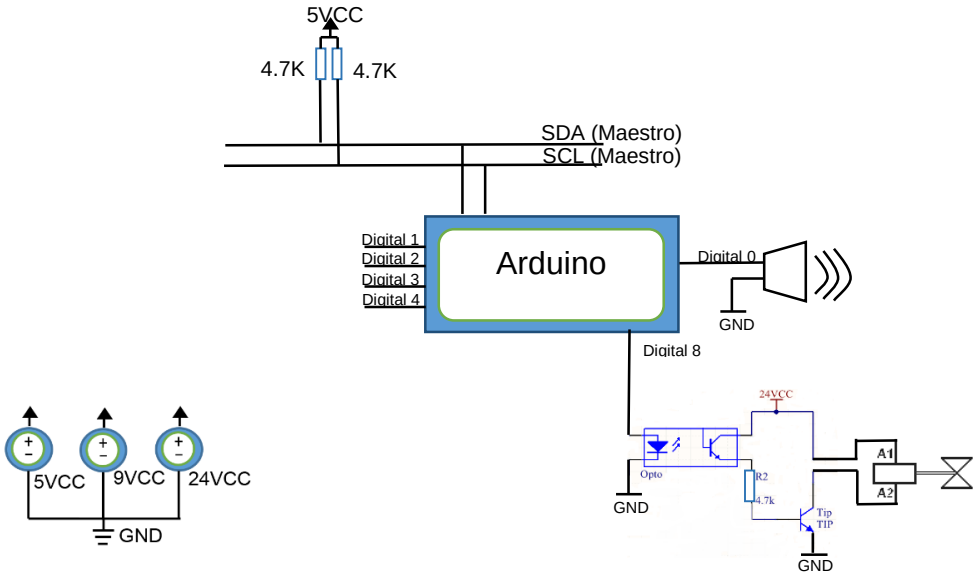


Figura 3.28 Diagrama eléctrico a bloques del sistema microcontrolador esclavo.

En la figura 3.28 se observa la conexión que existe entre dispositivo maestro y el dispositivo esclavo por medio del bus de comunicación I^2C, también se aprecian los dispositivos que están conectados al esclavo.

El control de inyección de gas entre la electroválvula MSZD-3 de 24 Volts y el dispositivo esclavo de control se hace a través del optoacoplador 4N25 y un transistor de potencia TIP31.

Cabe señalar que el dispositivo esclavo tiene cuatro puertos de salida que quedaran disponibles en la placa desarrollada.

En la figura 3.29 se presenta la descripción a bloques del programa que se desarrolló para el dispositivo esclavo. En el protocolo de comunicación I^2C, los dispositivos deben tener una dirección, en el caso de este dispositivo esclavo se le asigno la dirección 0xF5. En el Apéndice C encontraremos el código correspondiente al microcontrolador esclavo.

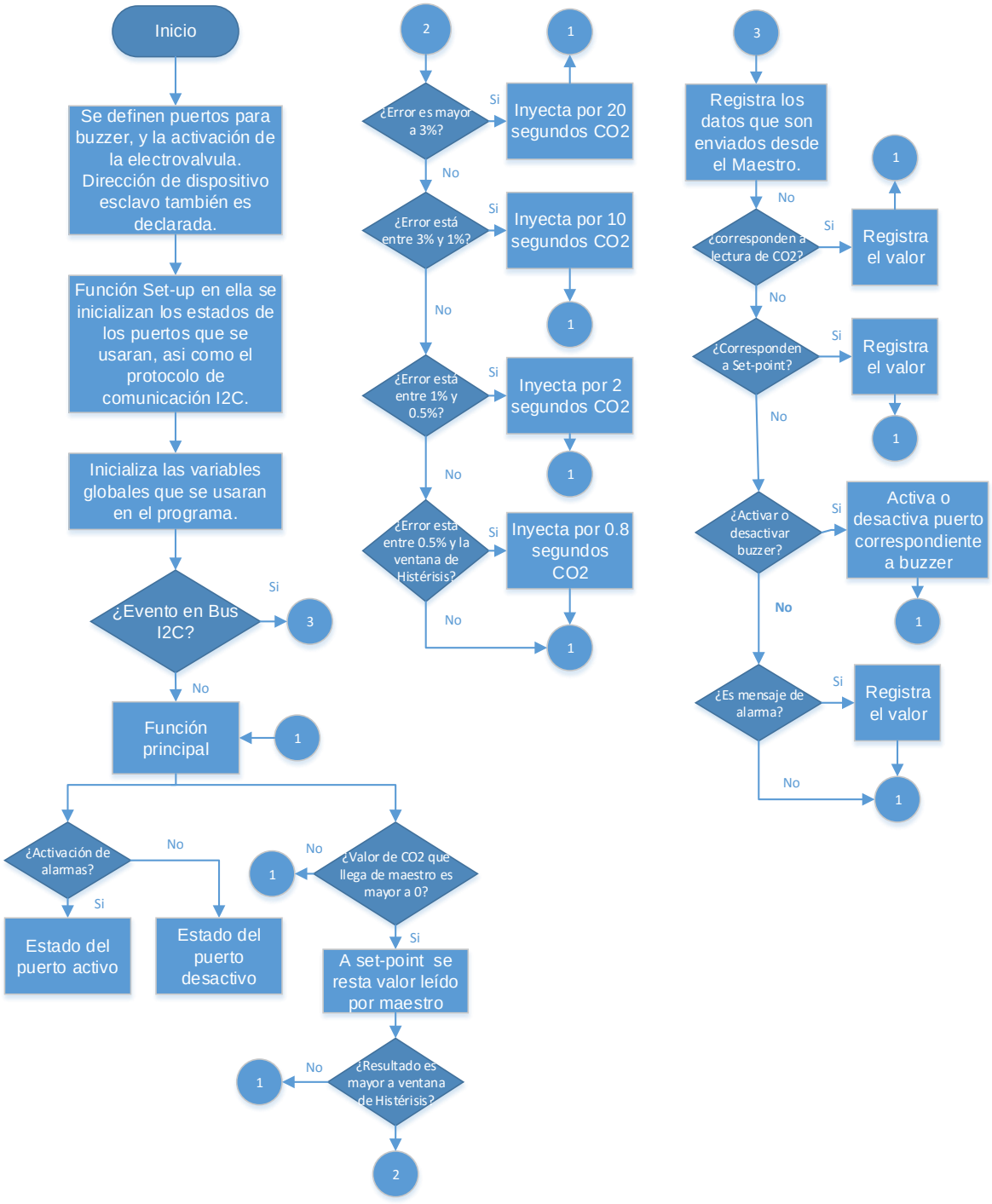


Figura 3.29 Diagrama a bloques del programa que se desarrolló para el dispositivo Esclavo.

CAPITULO 4. PRUEBAS.

El proyecto tiene como objetivo controlar la concentración de dióxido de carbono, así como medir las condiciones de humedad y temperatura que existen dentro de una cámara para el crecimiento y desarrollo de cultivos celulares.

Para que exista una confluencia de células o maduración de los cultivos que provienen de animales mamíferos es necesario que la concentración de dióxido de carbono alcance un nivel de 5% para poder conseguir un pH ligeramente alcalino de 7.4 en el medio en el que se desarrollan las células. Esta condición debe conjuntarse, por un lado con niveles de humedad que deberán ser de entre 95% a 100% en la cámara, para evitar que el medio donde están depositadas las células se modifique y una temperatura mayor a 32 y menor a 37.5 °C [3].

La estructura final de sensores se integró por el sensor K33-BLG para la medición de la concentración de CO_2 y por el sensor SHT22 para temperatura y humedad relativa.

Se realizaron tres tipos de pruebas:

a) Para evaluar la comunicación entre dispositivos y garantizar la veracidad de la información transmitida y almacenada.
b) Para caracterizar y validar el buen funcionamiento de los sensores respecto a lo reportado por el fabricante. Se utilizaron las aplicaciones comerciales de software GasLab y DAS (ambos de empresa CO2 Meter, USA), para el registro y análisis de la información proporcionada por los sensores.
c) Para validar la operación del sistema desarrollado en sus funciones de medición y control en la incubadora, se utilizó un termómetro y tres incubadoras comerciales como referencia.

4.1. Equipos de referencia y mediciones basales

Debido a que el sensor DHT22 está calibrado por el fabricante, temperatura y humedad relativa, se realizaron pruebas para confirmar el buen funcionamiento del dispositivo. Este proceso se realizó utilizando como referencia un multímetro de la marca Mastech modelo MS8228. El rango de temperaturas que pueden ser medidas con este aparato comercial van desde 0 ^{0}C hasta 40 ^{0}C con resolución de 0.1°C; el rango de medición de humedad relativa va de 20 % hasta 95%, con una resolución de 0.1% RH [23].

El equipo que se empleó para medición y validación de la concentración del dióxido de carbono, fueron tres cámaras de incubación comerciales: dos de marca Thermo Scientific de la serie 8000WJ, con triple pared que incluye camisa de agua para mantener las temperaturas por largos periodos en el rango de 5 ^{0}C hasta 55 ^{0}C. Tiene en su interior filtros HEPA para la mantener el ambiente interior libre de partículas y polvo. El rango de dióxido de carbono que maneja es de 0 % hasta 20%, usando sensores en la medición del gas de tecnología infrarroja, con sensibilidad de 0.1 %. Tiene un volumen interno de 187 157.868 cm^3 [24].

El segundo tipo de incubadora que se utilizó fue de la marca NUAIRE modelo NU-4750, con un volumen de 188 litros en su interior. Cuenta con camisas de agua para mantener las temperaturas deseadas. Las temperaturas que maneja la cámara en su interior van de 18 ^{0}C hasta 55 ^{0}C, con exactitud de ±0.0125 ^{0}C, con un set-point programado de 37 ^{0}C. La cámara cuenta con el control de dióxido de carbono con un rango de medición desde 0% hasta 20 %, con una exactitud ±0.1%. El set-point definido para el dióxido de carbono es de 5% de concentración[25].

Se midió el nivel de dióxido de carbono en un ambiente abierto con el sistema desarrollado, cuyo resultado fue de 0.04%, el cual coincide con lo comúnmente reportado y usado como referencia.

Los registros se hicieron durante 20 minutos, en tres días consecutivos, a la misma hora del día, utilizando las aplicaciones comerciales GasLab y DAS.

4.2. Protocolo de Medición de CO_2 para validación.

Una vez que se tuvo el sistema autónomo de medición para el dióxido de carbono, se requería tener la certeza de que funcionaba correctamente. Para esto se utilizó una cámara de incubación de la marca Thermo Scientific serie 8000WJ como referencia (Incubadora2), que no tuviera código de seguridad para poder cambiar los parámetros en el dióxido de carbono y así poder diseñar un protocolo de medición.

El protocolo establecido para la validación fue el siguiente:

1) Hacer el experimento a la misma hora, en tres días diferentes consecutivos.
2) Registrar el nivel de dióxido de carbono que se encuentra dentro de la cámara de la incubadora2.
3) Registrar los valores de [CO_2] en el rango de 1 a 6% cada 0.5 %, una vez que las lecturas de cada valor no cambien en más de 0.05%
4) Una vez estabilizada, se saca todo el dióxido de carbono que había dentro de ella y se configura a un nuevo valor que alcanzará la incuabadora2 en los niveles de dióxido de carbono, el valor mínimo que se pudo establecer es 1%, ya que en el cuarto de cultivo existen concentraciones significativamente altas que no permiten bajar más los niveles del gas.

El incremento se hará una vez que este estable el gas tanto en la cámara de incubación, como en varias lecturas del sistema para poder obtener un valor promedio en cada incremento.

En la figura 4.1 se muestra el arreglo utilizado para las pruebas de validación del sistema autónomo desarrollado y la incubadora 2.

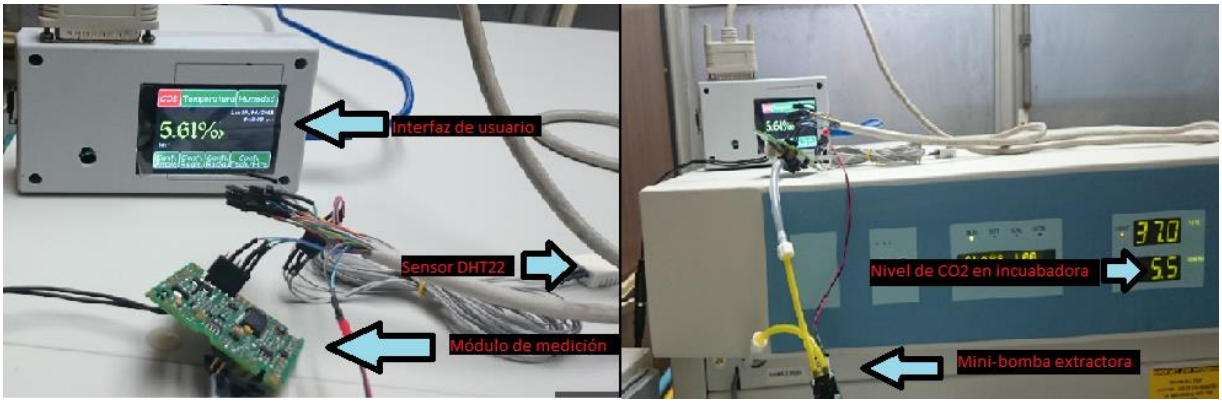


Figura 4.1 Sistema autónomo desarrollado y la incubadora 2 de la marca Thermo Scientific serie 8000WJ.

4.3. Prueba de medición de los tres parámetros y control de CO_2.

Implementado el protocolo anterior (Sección 4.2), y comprobado el funcionamiento de medición del sistema se dio paso a la implementación del control para el dióxido de carbono.

Montado el sistema de control en la estructura que se estaría trabajando, se procedio a diseñar un protocolo para la medición de los tres parámetros y el control de CO_2 dentro de la estructura.

El protocolo establecido para la medición de los tres parámetros y el control fue el siguiente:

1) Hacer el experimento a la misma hora en tres días diferentes consecutivos.
2) Se dejara la incubadora encendida una noche anterior a una temperatura interior de 37 ^{0}C con una charola de vidrio que contenga 1 litro de agua destilada.
3) El día de la medición, se abrirá la puerta de la incubadora por 2 minutos para dejar escapar posibles concentraciones de dióxido de carbono que puedan existir.
4) Se cerrará la cámara sin hacer ningún registro durante 2 minutos.

5) Se encenderá el sistema autónomo registrando los tres parámetros durante 2 minutos para, para una medición basal.
6) Después de los dos minutos, se configurará el sistema para una concentración de 5% de dióxido de carbono.
7) Se abrirá tanque que contiene el CO_2.
8) Se registrará durante 20 minutos para una estabilización del CO_2.
9) Posteriormente se abrirá la puerta de la cámara por 2 minutos.
10) Al finalizar se cierra la puerta y se dejará al sistema registrando durante 3 horas consecutivas.

CAPÍTULO 5. RESULTADOS Y DISCUSIÓN.

5.1. Resultados de la medición de temperatura y humedad obtenidos fuera de la incubadora

Se realizaron los registros de temperatura y humedad, se llevaron a cabo en un ambiente fuera de la cámara durante cinco minutos en un ambiente fuera de la cámara; los resultados obtenidos se compararon con la referencia comercial (Sección 4.1). En la figura 5.1 se observan el comportamiento de la temperatura registrada con el sensor DHT22, la media calculada de los registros obtenidos, y la referencia comercial.

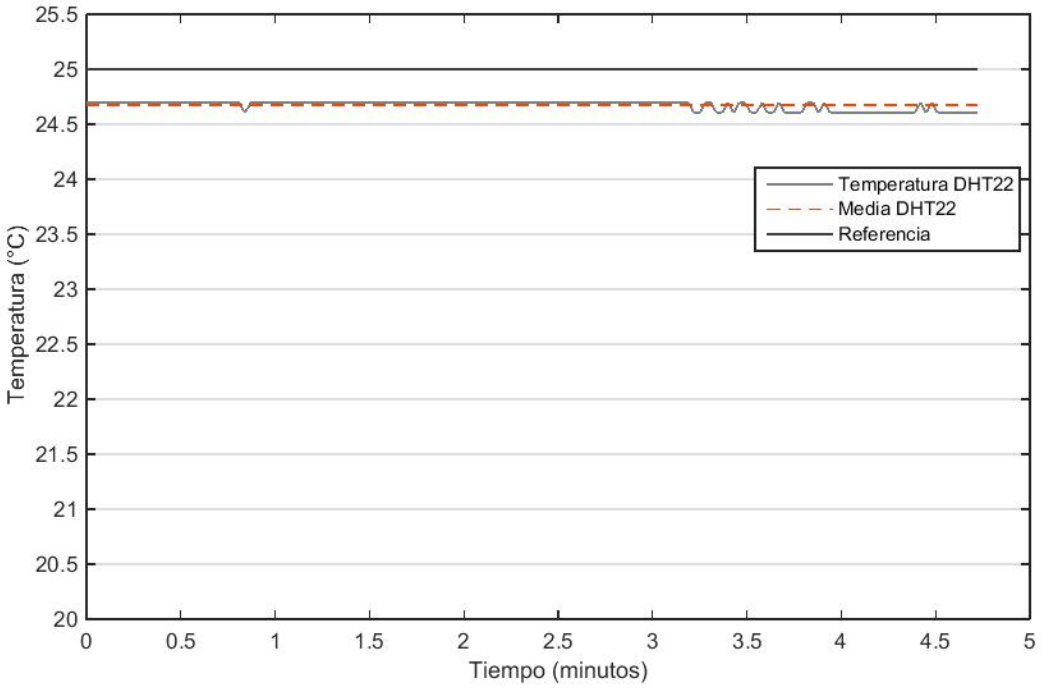


Figura 5.1 Registros obtenidos del sensor DHT22 para la variable de temperatura.

La diferencia entre la media de los datos obtenidos del sensor DHT22 respecto al valor de la referencia comercial fue de 0.33 ^{0}C a 25 ^{0}C.

Paralelamente se llevó acabo un registro de la humedad relativa durante cinco minutos, en la Figura 5.2 se observa el comportamiento de la humedad relativa

registrada con el sensor DHT22, la media calculada de los registros obtenidos, y la referencia comercial.

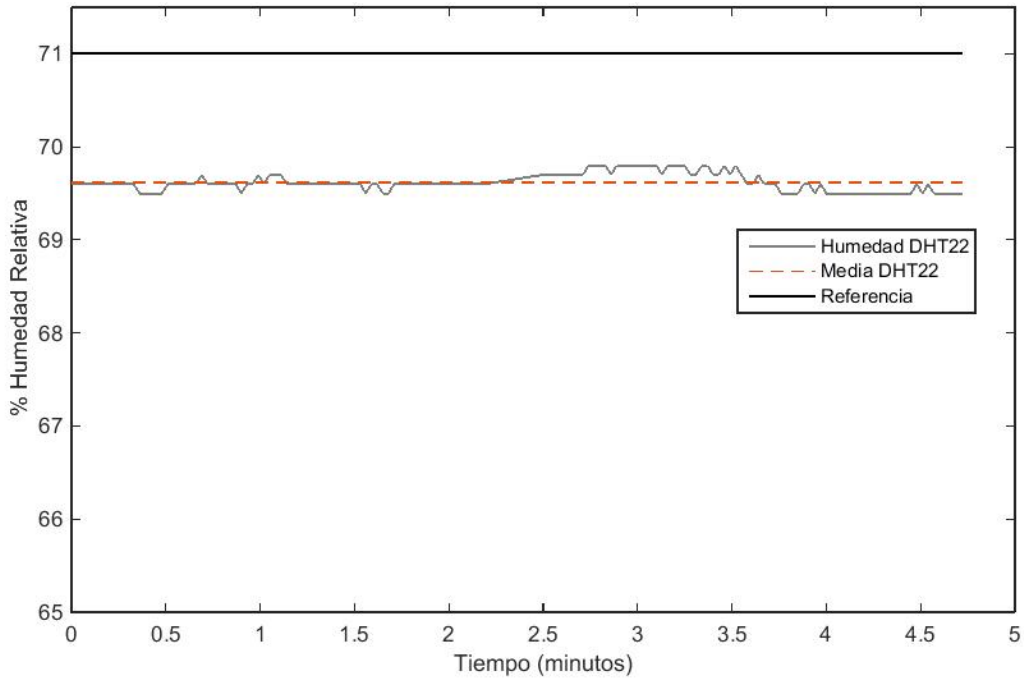


Figura 5.2 Registros de la humedad relativa obtenidos con el sensor DHT22.

La diferencia entre la media de los datos obtenidos del sensor DHT22 respecto al valor de la referencia comercial fue de 1.4% RH a 71% RH. El sensor arrojó una respuesta acorde con los parámetros reportados por el proveedor de ±2%.

5.2. Resultados de las mediciones de temperatura y humedad obtenidos dentro de una incubadora.

Los resultados arrojados por el sensor DHT22 dentro la incubadora Thermo Scientific denominada incubadora2, con una temperatura promedio en el interior de 37 ^{0}C, se muestran en la figura 5.3:

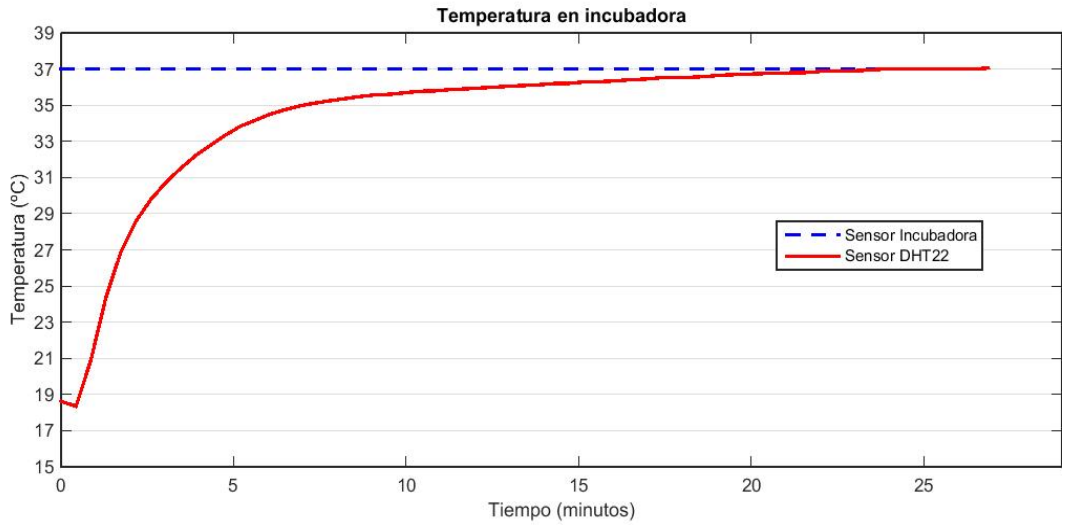


Figura 5.3 Registros de sensor DHT22, parámetro de temperatura de una cámara de ambiente controlado.

El nivel mínimo de la temperatura registrado con el sensor DHT22 que se muestra en un inicio corresponde al momento en que el sensor estaba fuera de la cámara como se puede apreciar en la gráfica, una vez que se introduce el sensor, la respuesta de éste, después de haber cerrado la puerta tiende a alcanzar el valor de 37 ^{0}C definidos para la incubadora en alrededor de 13 minutos. Cabe hacer el comentario que este tiempo de respuesta no es estrictamente del sensor, toda vez que al introducir el sensor al interior de la cámara, la temperatura de 37 ^{0}C definida, desciende y poco a poco comienza a recuperarse hasta los 37 ^{0}C.

Para efectuar los registros de humedad relativa dentro de la incubadora, cabe hacer el comentario que se realizaron de manera paralela con la toma de datos de la temperatura. La figura 5.4 muestra los resultados de la humedad relativa obtenidos dentro de la incubadora2.

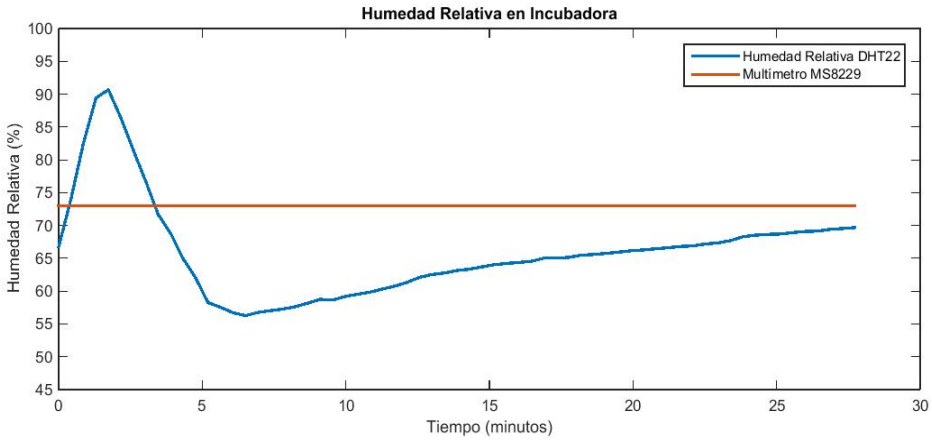


Figura 5.4 Gráfica de sensor DHT22, parámetro de humedad relativa dentro de la incubadora2, usando el multímetro de la marca MASTECH como referencia.

Debido a que la incubadora2 no contaba con un indicador para el parámetro de humedad que nos permitiera corroborar los datos obtenidos por el sensor DHT22, se requirió utilizar una referencia externa, la referencia que se empleo fue el multímetro (Sección 4.1). Cabe destacar que el sobretiro que se observa al inicio del registro corresponde a la humedad a que fue expuesto el sensor con la asepsia mediante alcohol.

5.3. Resultados de las mediciones de Dióxido de Carbono (CO_2) obtenidos mediante el software DAS, fuera de la incubadora.

Del módulo K33-BLG, se obtuvieron nuevos registros de los 3 parámetros ya mencionados. En la figura 5.5 se observan los resultados adquiridos por medio del software DAS de la compañía CO2meter para ello empleamos el protocolo de comunicación UART integrado en el módulo K33-BLG y la interfaz del software, ésta nos permite generar archivos con los registros de cada parámetro.

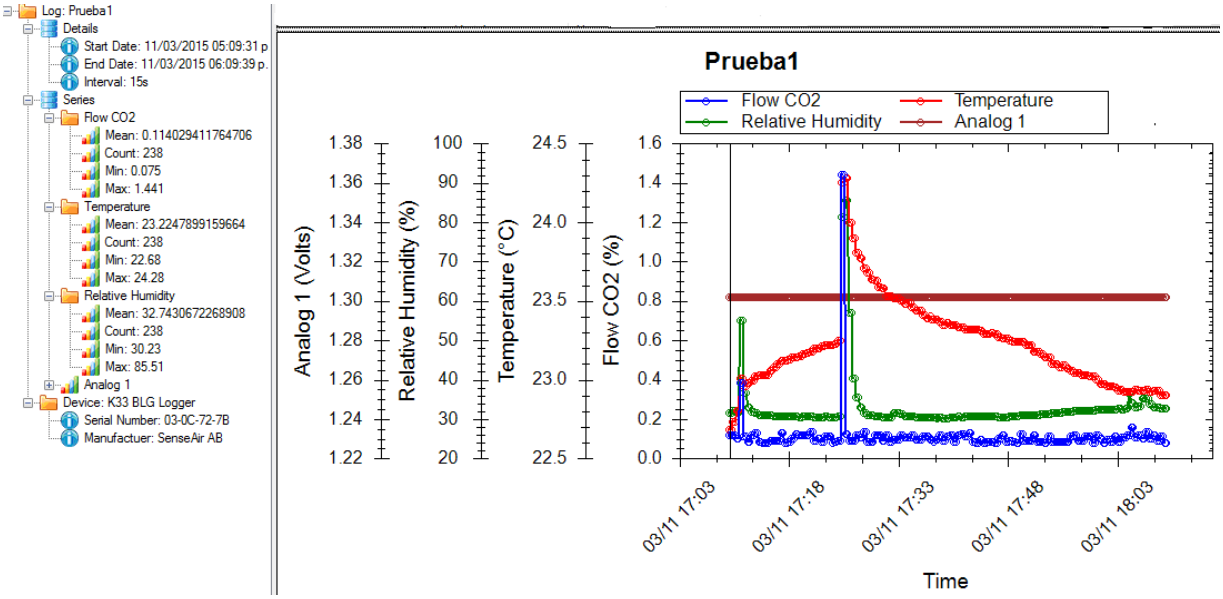


Figura 5.5 Registros obtenidos por el módulo K33-BLG y el software DAS, en un espacio abierto.

5.4. Resultados de las mediciones de dióxido de carbono (CO_2) obtenidos dentro de una incubadora.

En la figura 5.6 se muestran los resultados de las mediciones de altas concentraciones de CO_2 en espacios cerrados. Todos los registros se realizarón con el software DAS. Los resultados mostrados corresponden a las concentraciones del gas en la incubadora de marca NUAIRE

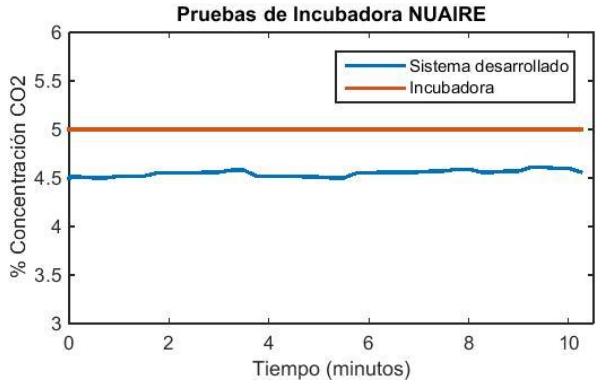


Figura 5.6 Gráfica obtenida en una cámara de incubación por medio del módulo K33-BLG y el software DAS.

En la figura 5.6 anterior se pueden apreciar los valores de la concentración de CO_2 registrados por medio del módulo y la computadora. El valor máximo registrado fue de 4.6%, en tanto que la media de todo el registro fue de 4.25%.

Estos resultados permitieron corroborar que el sistema tiene capacidad de leer altas concentraciones de CO_2.

En la Figura 5.7 se muestra la resultados de los registros hechos por el software DAS en la incubadora Thermo Scientific denominada incubadora1.

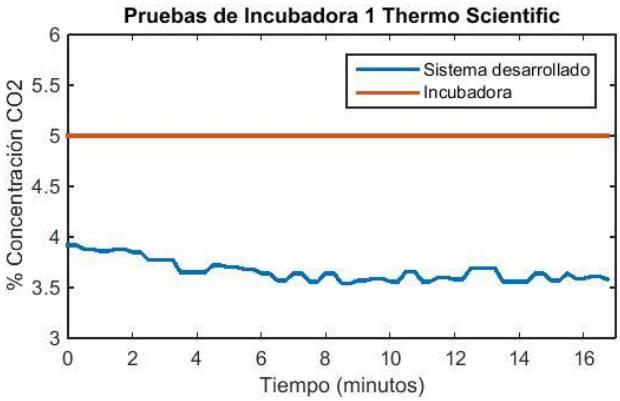


Figura 5.7 Gráficas comparativas de los registros obtenidos con el módulo K-33 BLG y el sensor interno de la incubadora Thermo Scientific denominada incubadora1.

En la Figura 5.7 se muestra los registros que se hicieron a la incubadora1, ésta muestra la inestabilidad del CO_2 en su interior. La cámara indicaba un porcentaje de 5% de CO_2 y el módulo K33-BLG en conjunto con el software DAS presentaban una media de 3.66 % de CO_2.

La diferencia de 1.34% que existe entre el sensor de la incubadora y el módulo K-33 BLG sugiere que el sistema de medición de CO_2 de la incubadora1 presenta alguna anomalía en su sensor, ésto, porque existían reportes de fallas en el equipo.

5.4.1. **Resultados de las mediciones efectuadas con el sistema autónomo desarrollado.**

Los registros de la concentración de CO_2 que fueron obtenidos con el modulo K-33 BLG interfazado con el arduino mega mediante el protocolo I^2C, correspondientes al día 1, se muestran en la Tabla 5.1.

	Sistema Desarrollado	Incubadora2	Error
1	1.19	1	0.19
2	1.78	1.5	0.28
3	2.22	2	0.22
4	2.71	2.5	0.21
5	3.20	3	0.20
6	3.71	3.5	0.21
7	4.13	4	0.13
8	4.66	4.5	0.16
9	5.12	5	0.12
10	5.59	5.5	0.09
11	6.04	6	0.04

Tabla 5.1. Registros del parámetro de dióxido de carbono, entre el sistema desarrollado y la cámara de incubación 2, primer día de registros.

En la Tabla 5.1 se aprecia que hay un error máximo de 0.28% de concentración de CO_2 entre los datos registrados por el sistema respecto a los establecidos en la incubadora. Siendo menor a lo especificado por el fabricante de la incubadora.

En la Figura 5.8 se muestra las dos curvas correspondientes a cada columna de los datos de CO_2 obtenidos por el sistema y la incubadora2, que corresponden al día uno.

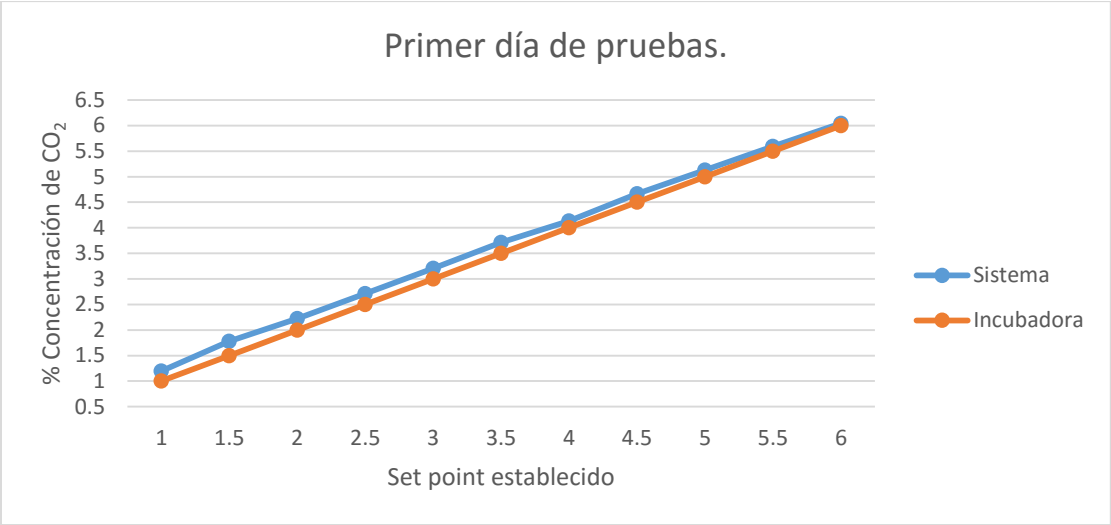


Figura 5.8 Gráfica de los registros obtenidos por el sistema desarrollado y la incubadora 2.

El coeficiente de correlación que hay entre el sistema desarrollado y la cámara de incubación resulto de 0.999.

La Tabla 5.2 muestra los datos correspondientes al segundo día de pruebas realizadas en el cuarto de cultivo entre la cámara de incubación 2 y el sistema desarrollado. Error máximo que se obtuvo entre los dos sistemas fue de 0.27%.

	Sistema Desarrollado	Incubadora2	Error
1	1.175	1	0.17
2	1.64625	1.5	0.14
3	2.1	2	0.10
4	2.57125	2.5	0.07
5	3.06	3	0.06
6	3.50375	3.5	0.00
7	3.997	4	0.00
8	4.458	4.5	0.04
9	4.81214286	5	0.18
10	5.40294118	5.5	0.09
11	5.7275	6	0.27

Tabla 5.2 Registros del parámetro de dióxido de carbono, entre el sistema desarrollado y la cámara de incubación 2, segundo día de registros.

En la Figura 5.9 se muestran los registros de concentración de CO_2 que corresponden a los datos de la tabla 5.2.

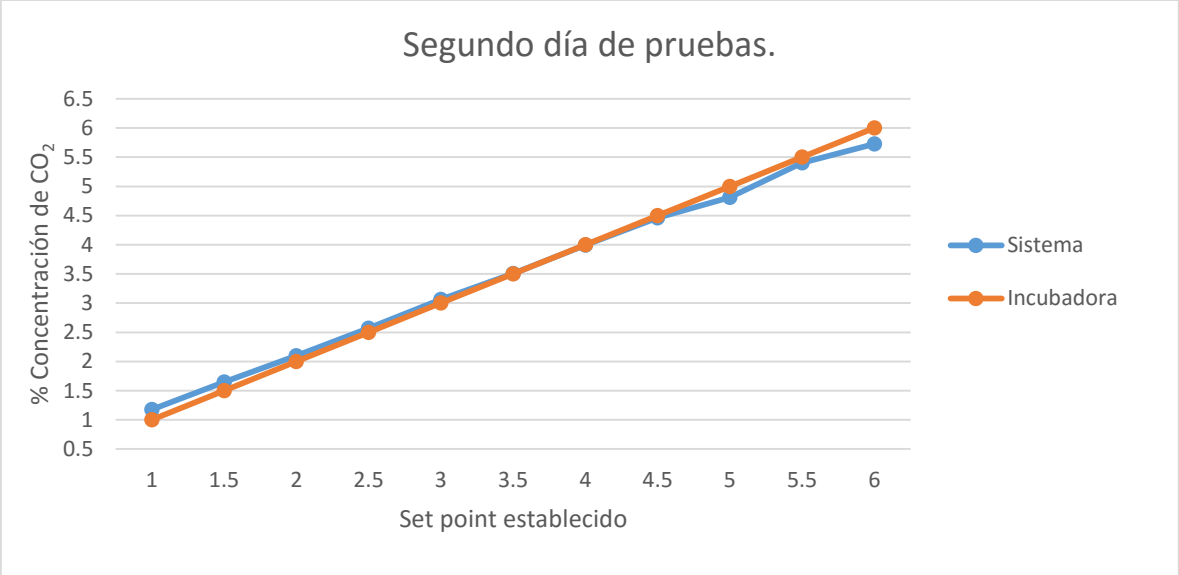


Figura 5.9 Gráfica de los registros obtenidos por el sistema desarrollado y la incubadora 2 de la marca, realizados en el segundo día de pruebas.

Para el segundo día de pruebas la correlación que hay entre el sistema desarrollado y la cámara de incubación resulto de 0.998.

La tabla 5.3 muestra los valores correspondientes a los registros realizados entre el sistema desarrollado y la incubadora2, durante el día 3. Cabe resaltar que los datos obtenidos en los tres días por el sistema, son datos promediados.

	Sistema Desarrollado	Incubadora2	Error
1	1.06	1	0.06
2	1.66	1.5	0.16
3	2.14	2	0.14
4	2.59	2.5	0.09
5	3.12	3	0.12
6	3.54	3.5	0.04
7	4.00	4	0.00
8	4.47	4.5	0.02
9	4.95	5	0.04
10	5.30	5.5	0.2
11	5.76	6	0.23

Tabla 5.3 Registros de la magnitud de dióxido de carbono, entre el sistema desarrollado y la cámara de incubación 2, tercer día de registros.

El comportamiento de los registros de CO_2 obtenidos por el sistema desarrollado del tercer día es similar al obtenido el segundo día, apenas se tiene una diferencia de 0.04% para el mayor de los casos,

En la Figura 5.10 se muestran los resultados de CO_2 obtenidos por el sistema desarrollado y la incubadora2 durante el tercer día.

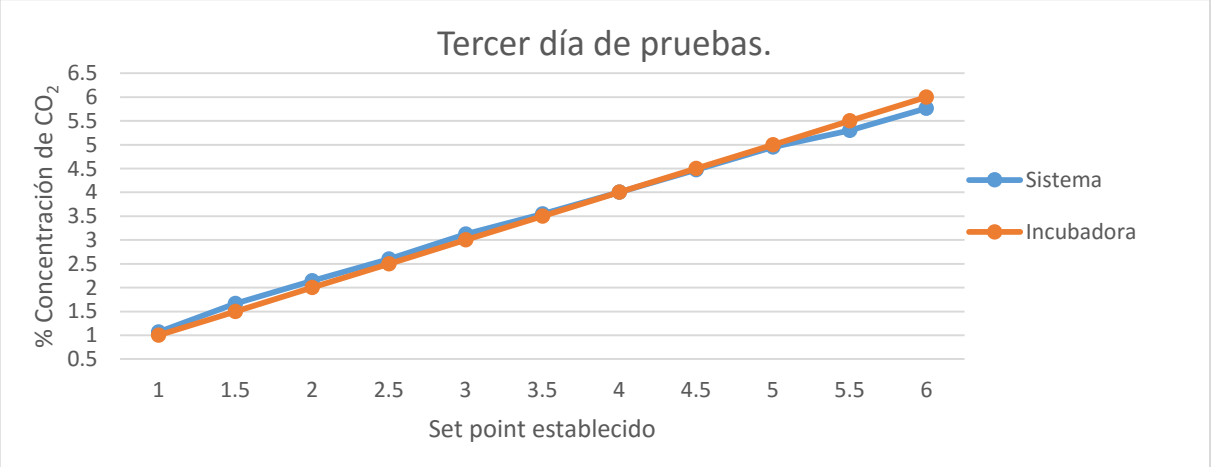


Figura 5.10 Gráfica de los registros obtenidos por el sistema desarrollado y la incubadora 2, realizados en el tercer día de pruebas.

En la figura 5.10 se puede observar que la tendencia entre las curvas es alta ya que la correlación es 0.997.

En la tabla 5.4 se muestra un comparativo de los resultados obtenidos de la concentración de CO_2 de los tres días en que los que se efectuaron los registros.

		Sistema Desarrollado		
	Incubadora 2 CO2%	**Día 1 CO2%**	**Día 2 CO2%**	**Día 3 CO2%**
1	1.0	1.19	1.17	1.06
2	1.5	1.78	1.64	1.66
3	2.0	2.22	2.10	2.14
4	2.5	2.71	2.57	2.59
5	3.0	3.20	3.06	3.12
6	3.5	3.71	3.50	3.54
7	4.0	4.13	3.99	4.00
8	4.5	4.66	4.45	4.47
9	5.0	5.12	4.81	4.95
10	5.5	5.59	5.40	5.30
11	6.0	6.04	5.72	5.76

Tabla 5.4 Registros de la magnitud de dióxido de carbono, entre el sistema desarrollado durante los tres días y la cámara de incubación 2.

Los datos de los tres días se introdujeron al software de GraphPad Prism para obtener la concordancia que existe entre el sistema desarrollado y la incubadora 2, obteniendo un coeficiente de Pearson de r=0.9998, lo que indica que la correlación entre las variables X (incubadora2 Thermo Scientific) y Y (Sistema desarrollado) es muy alta [26].

La estrecha relación que existe entre los sistemas comparados queda demostrada con el coeficiente de correlación (r^2)[26, 27], que mediante el programa GraphPad arrojo una r^2=0.9995, entre la incubadora 2 y las tres mediciones realizadas por el módulo en los diferentes días. Representándolo en porcentaje resulta de 99.95%.

La Figura 5.11, muestra el resultado correspondiente a la regresión lineal que fue generada a partir de software GraphPad con los registros de la incubadora 2 y los registros de los tres días mediante el sistema desarrollado.

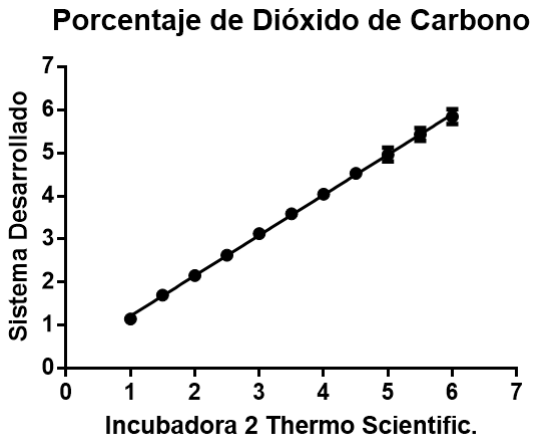


Figura 5.11 Gráfica que muestra la regresión lineal entre la incubadora 2 de la marca Thermo Scientific, y el sistema desarrollado.

5.5. Resultados de la acción del control del CO_2 y de las mediciones de T, RH y [CO_2].

Los resultados mostrados en las siguientes figuras corresponden a las pruebas efectuadas en la incubadora acondicionada para este proyecto

En la figura 5.12 se muestran los primeros 22 minutos de registro de la concentración de CO_2, los datos que inician antes del cero en el eje x, corresponden a una medición basal del gas en el interior de la cámara la cual se estuvo registrando durante dos minutos, la respuesta que se observa después del cero, corresponde al inicio del control por parte del sistema. Una vez iniciada esta acción la recuperación del nivel deseado (5%) tardó alrededor de 5 minutos, lo cual representa una respuesta buena con respecto a los sistemas comerciales que andan en alrededor de 10 minutos.

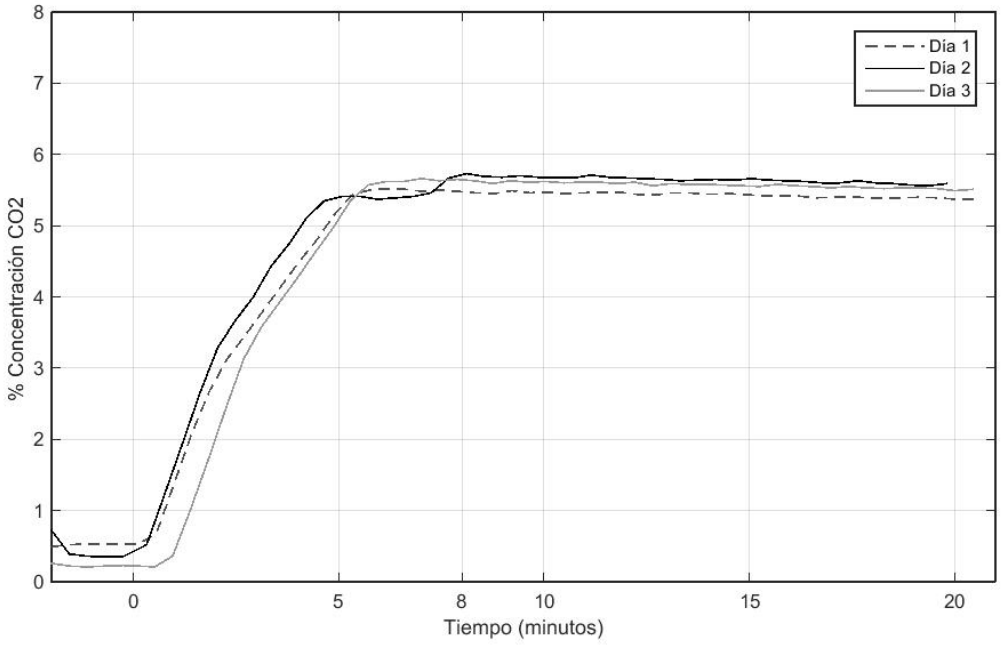


Figura 5.12 Resultados de la acción control de dióxido de carbono durante los primeros 22 minutos.

Resulta importante observar el tiempo de recuperación del nivel de la concentración de CO_2 a partir del momento en que se cierra la puerta, indicado como “P”, se observa cómo se lleva a cabo la recuperación de [CO_2] en el interior de la cámara, alcanzando el nivel deseado (5%) en un tiempo de 5 minutos. En la figura 5.13 se observa el comportamiento.

También podemos observar que una vez alcanzado el nivel de dióxido de carbono la concentración no desciende del 5% establecido como set point debido a la acción del sistema. Se excede con un porcentaje de alrededor de 0.5%, lo cual está dentro de los parámetros establecidos en el uso de incubadoras.

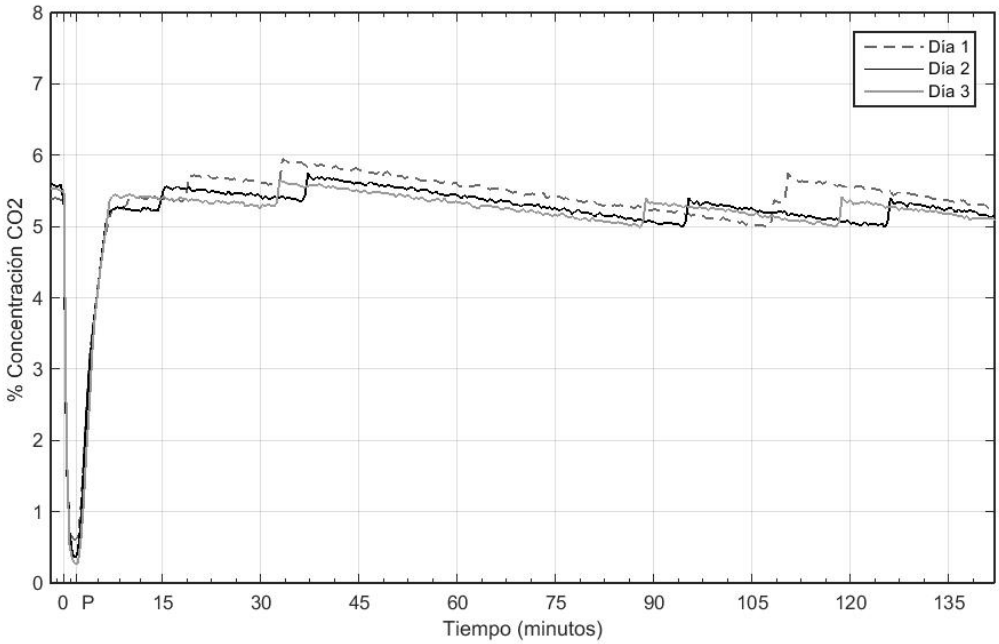


Figura 5.13 Resultados de protocolo de medición y control de dióxido de carbono durante 140 minutos.

En la figura 5.14 se muestra la medición del parámetro que corresponde a temperatura, en ella solo se muestra los registros que corresponden a los 140 minutos después de haber cerrado la puerta por última vez. También podemos observar que tarda alrededor de 30 minutos para poder alcanzar una temperatura promedio de 36.5 ^{0}C.

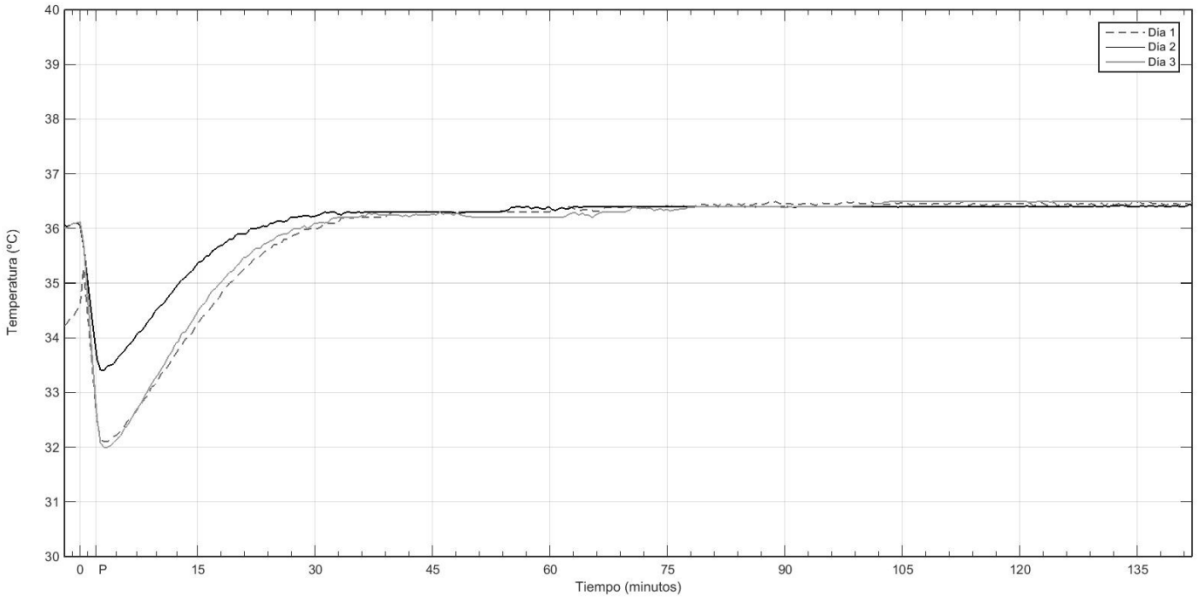


Figura 5.14 Resultados de protocolo de medición para el parámetro de temperatura durante 140 minutos.

En la figura 5.15 se muestran los registros de humedad relativa que se obtuvieron. En ella se puede apreciar que este parámetro alcanza el nivel deseado de 95% en un promedio de dos horas. Este tiempo largo se debe a acciones comunes y frecuentes de abrir y cerrar la puerta con las cuales se pierde humedad en la cámara.

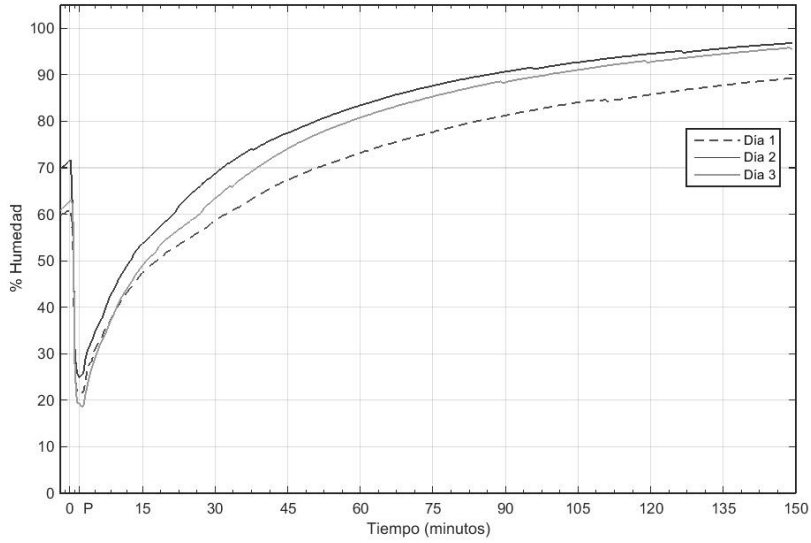


Figura 5.15 Resultados de protocolo de medición para el parámetro de húmedad relativa durante 150 minutos.

CAPITULO 6. CONCLUSIONES Y PERSPECTIVAS.

Se ha desarrollado un sistema para medición de temperatura, humedad y concentración de CO2, así como el control del CO2 en el interior de una cámara de incubación para el crecimiento de monocapas celulares in vitro. Este sistema cumple con las especificaciones determinadas para estos equipos.

Los protocolos diseñados para la medición de la concentración del dióxido de carbono, están siendo adoptados para su aplicación en el área de cultivos celulares del CINVESTAV.

El sistema autónomo desarrollado ha sido utilizado como instrumento de referencia y calibración en cámaras de incubación.

El sistema desarrollado tiene la capacidad de generar archivos de las mediciones de parámetros ambientales de la cámara de incubación a través de una interfaz gráfica para el seguimiento de experimentos.

El sistema desarrollado en conjunto con la cámara de incubación se encuentra en condiciones para ser utilizado en experimentos con cultivos celulares sometidos a la influencia de campos magnéticos.

No obstante, durante los trabajos realizados a lo largo del proyecto pudimos observar la necesidad de garantizar las condiciones adecuadas de temperatura a través de la medición de este parámetro en la proximidad de los cultivos control y experimental.

Mejorar el control de extracción del CO_2 para evitar un consumo excesivo de este gas durante el proceso de medición.

REFERENCIAS.

[1] https://www.adafruit.com/datasheets/Digital%20humidity%20and%20temperature%20sensor%20AM2302.pdf.

[2] F. C. Orejana and P. E. Gil-Loyzaga, "Estructura general de un laboratorio de cultivos celulares. Equipamiento esencial para las técnicas de cultivos celulares," *CULTIVO DE CÉLULAS ANIMALES Y HUMANAS. APLICACIONES EN MEDICINA REGENERATIVA,* p. 35, 2013.

[3] M. E. Castaño, "Cultivos celulares," *Editorial Biogenesis,* pp. 29-46, 2012.

[4] CO2meter, "K33-BLG/ELG Sensor module for environment logging Data Sheet and Manual ", CO2meter, Ed., ed, 2015.

[5] CO2meter, "http://cdn.shopify.com/s/files/1/0019/5952/files/ABC-calibration.pdf?1285616113."

[6] S. T. S. COMPANY, "Datasheet SHT1x (SHT10, SHT11, SHT15) Humidity and Temperature Sensor IC ", S. T. S. COMPANY, Ed., ed, 2011.

[7] CO2meter, "EM_CO2-Engine-BLG_ELG_description_Rev_1_00.doc," ed, 2010.

[8] CO2meter, "http://co2meters.com/Documentation/Manuals/Manual-GasLab.pdf," 2014.

[9] CO2meter, "Sensor Pump Kit For Sample Draw Sensors," CO2meter, Ed., ed, 2010.

[10] W. Greiman, "SD Library, sdfatlib ", https://github.com/jbeynon/sdfatlib, Ed., ed, 2013.

[11] Adafruit/DHT-sensor-library, "Arduino library for DHT11DHT22, etc Temp & Humidity Sensors," October 2015.

[12] https://www.arduino.cc/en/Main/ArduinoBoardUno.

[13] CO2meter, "AN113 – Using the K33 ELG/BLG Sensor with a Microcontroller," p. 7, April 2011.

[14] L. Ada, "Adafruit 2.8" TFT Touch Shield v2-Capacitive or Resistive," A. L. System, Ed., ed, 06/08/2015

[15] Vishay, "Optocoupler, Phototransistor Output, with Base Connection," V. Semiconductors, Ed., ed, 2012.

[16] T. Instruments, "LM317 3-Terminal Adjustable Regulator," 2014.

[17] D. C. Co., "TECHNICAL SPECIFICATIONS OF SINGLE-PHASE SILICON BRIDGE RECTIFIER."

[18] E. Pereda. *FUENTE DE ALIMENTACIÓN VARIABLE*. Available: http://www.aeromodelismocampoo.com/contefuente.php

[19] https://www.arduino.cc/en/Main/ArduinoBoardMega2560#.

[20] Ó. T. Artero, *Arduino: curso práctico de formación*: RC Libros, 2013.

[21] J. Irazabel and S. Blozis, "Philips Semiconductors,“I2C-Manual,” Application Note, ref," *AN10216-0, March,* vol. 24, 2003.

[22] G. Gridling and B. Weiss, "Introduction to microcontrollers," *Vienna University of Technology Institute of Computer Engineering Embedded Computing Systems Group,* 2007.

[23] http://www.electronicasannicolas.com.co/productos/index.php?id_product=9379&controller=product.

[24] T. F. Scientific. (2007, Thermo Scientific Series 8000 – Direct Heat and Water Jacket CO2 Incubators. Available: http://www.thermoscientific.com/content/dam/tfs/LPG/LED/LED%20Documents/Catalogs%20&%20Brochures/CO2%20Incubators/D12766~.pdf

[25] I. NuAire. (2008, Water-Jacketed, US Autoflow Automatic CO2 Incubator Models NU-4750, NU-4850, NU-4950 Available: https://www.labmakelaar.com/fjc_documents/c02incubator11.pdf

[26] H. Pedroza, L. Dicovskyi, R. Bocchetto, M. Carneiro, G. Sparovek, M. Castagna Molina*, et al.*, "Sistema de análisis estadístico con SPSS," IICA, Managua (Nicaragua). INTA, Managua (Nicaragua) 1992-4801, 2007.

[27] M. S. Alicia Vila, Ana López,Ángel A. Juan, "CORRELACIÓN LINEAL Y ANÁLISIS DE REGRESIÓN."

ANEXO A

```
#include <DHT.h>
#include <SPI.h>
#include <stdlib.h>
#include <ILI9341_due_gText.h>
#include <ILI9341_due.h>
#include <Wire.h>
#include <SD.h>
#include <Adafruit_FT6206.h>
#include "RTClib.h"
#include <Comic10.h>
#include <Comic15.h>
#include <Comic20.h>
#include <Comic23.h>
#include <Comic60.h>
#include <Old75.h>


// The FT6206 uses hardware I2C (SCL/SDA)
Adafruit_FT6206 ctp = Adafruit_FT6206();

// The display also uses hardware SPI, plus #9 & #10
#define TFT_CS 10
#define TFT_DC 9
#define DHTPIN 2
#define DHTTYPE DHT22

ILI9341_due tft = ILI9341_due(TFT_CS, TFT_DC);
RTC_Millis rtc;
ILI9341_due_gText t1(&tft);
```

```
DHT dht(DHTPIN, DHTTYPE);

// Color set
#define         BLACK           0x0000
#define         RED             0xF800
#define         GREEN           0x07E0
#define         BLUE            0x102E
#define CYAN            0x07FF
#define MAGENTA         0xF81F
#define YELLOW          0xFFE0
#define ORANGE          0xFD20
#define GREENYELLOW     0xAFE5
#define DARKGREEN       0x03E0
#define WHITE           0xFFFF

uint16_t  color;
uint16_t  colorFONDO=BLACK;

int co2Addr = 0x68;
int arduino = 0xF0;

double tempValuePro=0;
double tempValue=0;
double rhValuePro=0;
double rhValue=0;
double co2Value=0;
int tak=0;
uint8_t ala_co2=0, ala_tempe=0, ala_humed=0;
double a_co2=0.0,a_tempe=0.0,nivelCo2=0.0,nivelTemp1=0.0,nivelTemp2=0.0;
int a_humed=100,nivelHumedad=0;
```

```
char chractual[10]="";
char chranterior[10]="";
int chractuallon=0;
int primer=0;
int arduinoCo2Value=0;
int ardSetpoint=0;
long aaaa=0;
long mm=0;
long dd=0;
long hh=0;
long mn=0;
long ns=0;
char c_aaaa[5]="AAAA";
char c_mm[3]="MM";
char c_dd[3]="DD";
char c_hh[3]="HH";
char c_mn[3]="MM";
char c_ampm[5]="a.m.";
int ampm=3;
int hrs=0;
int c_date=0;
long oldd=70;
long oldm=70;
long olds=70;
long pmhours=0;
int radHrs=0;
int radMin=0;
String DiasSem[]={"Dom","Lun","Mar","Mie","Jue","Vie","Sab"};
String DDMMAA;
String HHMM;
```

```
String CO2;
String TEMPERATURA;
String HUMEDAD;

String auxiliar,s_radHrs,s_radMin;
boolean
modo_co2=true,modo_temperatura=false,modo_humedad=false,modo_alarma=true;
const int chipSelect = 53;


////////////////////////////////////////////////////////////////////////////Funcion de
Configuraión////////////////////////////////////////////////////////////////////////////////////////////
void setup(void)
{

  Serial.begin(9600);
  Wire.begin ();
  pinMode(13, OUTPUT);
  //Serial.println(F("Cap Touch Paint!"));
  tft.begin();
  tft.setRotation(iliRotation270);
  tft.fillScreen(colorFONDO);
  rtc.adjust(DateTime(__DATE__, __TIME__));
  dht.begin();
  if (! ctp.begin(40))
  {  // pass in 'sensitivity' coefficient
    Serial.println("Couldn't start FT6206 touchscreen controller");
    while (1);
  }
  if (startSDCard() == true){
```

```
    Serial.println("Inicio tarjeta SD correctamente");
  }
  //Serial.println("Capacitive touchscreen started");
  t1.defineArea(0, 0, 320, 240);
  t1.selectFont(Comic23);
}
void(* resetFunc) (void) = 0; //declare reset function @ address 0
////////////////////////////////////////////////////////////////////////Mensajes de las distintas alarmas que se
pueden dar////////////////////////////////////////////////////////////////////
void mensaje_alarma(int x1,int y1,char *mensajeAlarma)
{
  t1.setFontColor(YELLOW,BLACK);
  t1.selectFont(Comic23);
  t1.drawString(mensajeAlarma,x1,y1);
  t1.selectFont(Old75);
  t1.setFontColor(GREENYELLOW,BLACK);

}

////////////////////////////////////////////////////////////////////////////////Funcion                    de
Alarmas////////////////////////////////////////////////////////////////////////////
void alarmas(double ValueCO2,double ValueTemp,double ValueRh)
{
   DateTime T=rtc.now();
   if (oldm!=T.minute())
   {
     if (ala_co2<7 && ValueCO2<nivelCo2)
       {
         ala_co2++;
         if (ala_co2==6)
```

```
  {
      Wire.beginTransmission(arduino);
      Wire.write(0x4);
      Wire.endTransmission();
      mensaje_alarma(10,60,"Alarma de CO2");
  }
 }
else if (ValueCO2>nivelCo2)
{
 if (ala_co2==6)
 {
   Wire.beginTransmission(arduino);
   Wire.write(0x5);
   Wire.endTransmission();
   tft.fillRect(10,60,70,23,BLACK);
 }
 ala_co2=0;
}
if(ala_tempe<181 && ((ValueTemp<nivelTemp1)||(nivelTemp2<ValueTemp)))
{
  ala_tempe++;
  if (ala_tempe==180)
  {
      Wire.beginTransmission(arduino);
      Wire.write(0x4);
      Wire.endTransmission();
       mensaje_alarma(1,160,"Alarma Tempe.");
  }
}
else if (nivelTemp2>=ValueTemp>=nivelTemp1)
```

```
{
  if (ala_tempe==180)
  {
    Wire.beginTransmission(arduino);
    Wire.write(0x5);
    Wire.endTransmission();
    tft.fillRect(0,160,155,28,BLACK);
  }
  ala_tempe=0;
}
if (ala_humed<181 && ValueRh<nivelHumedad)
 {
   ala_humed++;
   if (ala_humed==180)
   {
        Wire.beginTransmission(arduino);
        Wire.write(0x4);
        Wire.endTransmission();
        mensaje_alarma(160,160,"Alarma Humeda.");
   }
 }
else if (ValueRh>nivelHumedad)
{
  if (ala_humed==180)
  {
    Wire.beginTransmission(arduino);
    Wire.write(0x5);
    Wire.endTransmission();
    tft.fillRect(160,160,160,23,BLACK);
  }
```

```
        ala_humed=0;
      }
   oldm=T.minute();
 }

}

////////////////////////////////////////////////////////////////////////////Función de Inicio y existencia de Tarjeta de Micro-SD//////////////////////////////////////////////////////////////////
boolean startSDCard(){
  boolean result=false;
  pinMode(53,OUTPUT);
  if(!SD.begin(chipSelect))
  {
    Serial.println("Fallo de tarjeta,o no se encuentra presente");
    result=false;
  }else{
    Serial.println("Tarjeta inicializada");
    result=true;
  }
  return result;
}
////////////////////////////////////////////////////////////////Función que genera los archivos para guardar datos de CO2, Temperatura y Humedad////////////////////////////////////////////////
void archMicroSD(int sen)
{
  DateTime T=rtc.now();
  if (T.hour()<10 && T.minute()<10 && T.second()<10) HHMM=("0"+String(T.hour())+":"+"0"+String(T.minute())+":"+"0"+String(T.second()));
```

```
 else if (T.hour()<10 && T.minute()<10 && T.second()>=10)
HHMM=("0"+String(T.hour())+":"+"0"+String(T.minute())+":"+String(T.second()));
 else if (T.hour()<10 && T.minute()>=10 && T.second()<10)
HHMM=("0"+String(T.hour())+":"+String(T.minute())+":"+"0"+String(T.second()));
 else if (T.hour()<10 && T.minute()>=10 && T.second()>=10)
HHMM=("0"+String(T.hour())+":"+String(T.minute())+":"+String(T.second()));
 else if (T.hour()>=10 && T.minute()<10 && T.second()<10)
HHMM=(String(T.hour())+":"+"0"+String(T.minute())+":"+"0"+String(T.second()));
 else if (T.hour()>=10 && T.minute()<10 && T.second()>=10)
HHMM=(String(T.hour())+":"+"0"+String(T.minute())+":"+String(T.second()));
 else if (T.hour()>=10 && T.minute()>=10 && T.second()<10)
HHMM=(String(T.hour())+":"+String(T.minute())+":"+"0"+String(T.second()));
 else HHMM=(String(T.hour())+":"+String(T.minute())+":"+String(T.second()));
 if (sen==1)
 {
  File dataCO2=SD.open("CO2.txt",FILE_WRITE);
  if (dataCO2)
  {
   dataCO2.print(co2Value);
   dataCO2.print("\t");
   dataCO2.println(HHMM);
   dataCO2.close();
  }
  else Serial.println("no se abrio el archivo CO2.txt");
 }

 if (sen==2)
 {
  File dataTem=SD.open("Temper.txt",FILE_WRITE);
  if (dataTem)
```

```
  {
    dataTem.print(tempValue);
    dataTem.print("\t");
    dataTem.println(HHMM);

    dataTem.close();
  }
  else Serial.println("no se abrio el archivo Temper.txt");
 }

 if (sen==3)
 {
  File dataHumed=SD.open("Humedad.txt",FILE_WRITE);
  if (dataHumed)
  {
    dataHumed.print(rhValue);
    dataHumed.print("\t");
    dataHumed.println(HHMM);
    dataHumed.close();
  }
  else Serial.println("no se abrio el archivo Humedad.txt");
 }

}

////////////////////////////////////////////////////////////////////////////Función que permite mandar a despertar el sensor de CO2////////////////////////////////////////////////////////////////////

void wakeSensor()
{
```

```
  TWCR &= ~(1<<TWEN);
  DDRD |= (1<<1);
  PORTD &= ~(1<<1);
  delay(1);
  PORTD |= (1<<1);
  TWCR |= (1<<TWEN);
  delay(1);
}
////////////////////////////////////////////////////////////////////////////Función para comprobar la existencia del sensor de CO2////////////////////////////////////////////////////////////////////////////
void initPoll()
{
  Wire.beginTransmission(co2Addr);
  Wire.write(0x11);
  Wire.write(0x00);
  Wire.write(0x60);
  Wire.write(0x35);
  Wire.write(0xA6);
  Wire.endTransmission();
  delay(20);
  Wire.requestFrom(co2Addr, 2);
  byte i = 0;
  byte buffer[2] = {0, 0};
  while(Wire.available())
  {
    buffer[i] = Wire.read();
    i++;
  }
}
```

```
////////////////////////////////////////////////////////////////////////////Función para leer los datos de CO2 obtenidos por el sensor////////////////////////////////////////////////////////
double readCo2()
{
  int co2_value = 0;
  digitalWrite(13, HIGH);
  Wire.beginTransmission(co2Addr);
  Wire.write(0x22);
  Wire.write(0x00);
  Wire.write(0x08);
  Wire.write(0x2A);
  Wire.endTransmission();
  delay(20);
  Wire.requestFrom(co2Addr, 4);
  byte i = 0;
  byte buffer[4] = {0, 0, 0, 0};
  while(Wire.available())
  {
    buffer[i] = Wire.read();
    i++;
  }

  co2_value = 0;
  co2_value |= buffer[1] & 0xFF;
  co2_value = co2_value << 8;
  co2_value |= buffer[2] & 0xFF;
  byte sum = 0;
  sum = buffer[0] + buffer[1] + buffer[2];
  if(sum == buffer[3])
  {
```

```
    digitalWrite(13, LOW);
    return ((double) co2_value);
  }else
  {
    digitalWrite(13, LOW);
    return (double) -1;
  }
}

////////////////////////////////////////////////////Función para para efecto que se presiono un boton de la pantalla principal////////////////////////////////////////////////////////////////////
void white_seccion(int x1,int y1,int x2,int y2)
{
  tft.fillRoundRect(x1,y1,x2,y2, 5,RED);
  while(ctp.touched())
    ctp.getPoint();
  tft.fillRoundRect(x1,y1,x2,y2, 5,WHITE);
}
////////////////////////////////////////////////////////////Función para obtener string apartir de numeros flotantes////////////////////////////////////////////////////////////////////
String getDecimal(float val)
{
 String decimales;
 int intPart = int(val);
 int decimal = 100*(val-intPart); //I am multiplying by 1000 assuming that the foat values will have a maximum of 3 decimal places
                                  //Change to match the number of decimal places you need
 if (decimal<10)decimales= String(intPart)+".0"+String(decimal);
 else decimales= String(intPart)+"."+String(decimal);
 return(decimales);
```

```
}

String getDecimal2(double value)
{
  String decimales;
  int intPart = int(value);
  int decimal = 1000*(value-intPart);

  if (decimal<10) decimales= String(intPart)+".00"+String(decimal);
  else if(decimal>=10 && decimal<=99) decimales=
String(intPart)+".0"+String(decimal);
  else decimales=String(intPart)+"."+String(decimal);
  return(decimales);
}
////////////////////////////////////////////////////////////////Función para pintar los distintos teclados en la
pantalla necesarios por el usuario////////////////////////////////////////////
void keyboard()
{

  t1.setFontColor(BLACK,ORANGE);
  t1.selectFont(Comic23);

  for(int xx=0;xx<5;xx++)
  {
    tft.fillRoundRect(xx*63+5,70,58,40,5,ORANGE);
    tft.drawRoundRect(xx*63+5,70,58,40,5,WHITE);
    auxiliar=String (xx+1);
    t1.drawString(auxiliar,xx*63+28,80);
  }
```

```
for(int xx=0;xx<5;xx++)
{
  tft.fillRoundRect(xx*63+5,115,58,40,5,ORANGE);
  tft.drawRoundRect(xx*63+5,115,58,40,5,WHITE);
  if (xx<4)auxiliar=String (xx+6);
  else auxiliar="0";
  t1.drawString(auxiliar,xx*63+28,125);
}

for(int xx=0;xx<5;xx++)
{
  tft.fillRoundRect(xx*63+5,160,58,40,5,ORANGE);
  tft.drawRoundRect(xx*63+5,160,58,40,5,WHITE);
  if ((xx==0)&&(ampm!=0)&&(ampm!=1))
  {
    t1.drawString(".",xx*63+28,170);

  }else if ((xx==0)&&(ampm==0))
  {
    t1.drawString("p.m.",xx*63+20,170);

  }else if((xx==0)&&(ampm==1))
  {
    t1.drawString("a.m.",xx*63+20,170);

  }
  if (xx==1)t1.drawString("Clr.",xx*63+18,170);
  if (xx==2)t1.drawString("Ent.",xx*63+15,170);
  if (hrs!=1)
  {
```

```
     if (xx==3)t1.drawString("Reg.",xx*63+15,170);
    }else if(xx==3)t1.drawString("Hrs.",xx*63+15,170);
    if (xx==4)t1.drawString("Ini.",xx*63+20,170);
  }
}
////////////////////////////////////////////////////////////////Funcion que despliega mensajes de error en caso de necesitarce////////////////////////////////////////////////////////////////
void error(int x1,int y1,char *error)
{
  t1.setFontColor(RED,BLACK);
  t1.selectFont(Comic23);
   for (int s=0;s<2;s++)
  {
     t1.drawString(error,x1,y1);
     delay(750);
     tft.fillRect(x1,y1,320-x1,240-y1,BLACK);
     delay(250);
  }
}
////////////////////////////////////////Funcion que valida los valores introducidos desde la pantalla del teclado para cualquier modo utilizado y muestra en//////////////////////////////////
//////////////////////////////////////////// pantalla el valor cuando es valido o muestra un mensaje de error cuando el parametro introducido es erroneo//////////////////////////////////////
void update_caracter(int tecla, int modos)
{

  if (modos==1)
  {
    t1.setFontColor(GREEN,BLACK);
    t1.selectFont(Comic23);
```

```
if (chractuallon<4)
{
  if (tecla>=48 && tecla<=54 && primer==0)
  {
    chractual[chractuallon]=tecla;
    chractual[chractuallon+1]='\0';
    chractuallon++;
    primer++;
    auxiliar=String(chractual);
    t1.drawString(auxiliar,10,210);

  }else if(tecla==11 && primer==0)
  {
    chractual[chractuallon]='0';
    chractual[chractuallon+1]='.';
    chractuallon=2;
    primer=2;
    auxiliar=String(chractual);
    t1.drawString(auxiliar,10,210);

  }else if(tecla==11 && primer==1)
  {
    chractual[chractuallon]='.';
    chractual[chractuallon+1]='\0';
    chractuallon++;
    primer++;
    auxiliar=String(chractual);
    t1.drawString(auxiliar,10,210);

  }else if(primer==2)
```

```
  {
    chractual[chractuallon]=tecla;
    chractual[chractuallon+1]='\0';
    chractuallon++;
    primer++;
    auxiliar=String(chractual);
    t1.drawString(auxiliar,10,210);

  }else if((primer==3)&& (chractual[2]!=48) )
  {
      chractual[chractuallon]=tecla;
      chractual[chractuallon+1]='\0';
      chractuallon++;
      primer++;
      auxiliar=String(chractual);
      t1.drawString(auxiliar,10,210);

  }else if((primer==3) && (chractual[2]==48) && tecla>=52)
  {

    chractual[chractuallon]=tecla;
    chractual[chractuallon+1]='\0';
    chractuallon++;
    primer++;
    auxiliar=String(chractual);
    t1.drawString(auxiliar,10,210);

  }
  else error(115,210,"Error de parametro");
}else error(165,210,"Fuera de rango");
```

```
 }
 if (modos==2)
 {
  t1.setFontColor(GREEN,BLACK);
  t1.selectFont(Comic23);
  if (chractuallon<4)
  {
   if (tecla>=49 && primer==0)
   {
    chractual[chractuallon]=tecla;
    chractual[chractuallon+1]='\0';
    chractuallon++;
    primer++;
    auxiliar=String(chractual);
    t1.drawString(auxiliar,10,210);

   }else if(tecla>=48 && tecla<=57 && primer==1 && chractual[0]>=49 &&
chractual[0]<=52 )
   {
    chractual[chractuallon]=tecla;
    chractual[chractuallon+1]='\0';
    chractuallon++;
    primer++;
    auxiliar=String(chractual);
    t1.drawString(auxiliar,10,210);

   }else if(chractual[0]==53 && primer==1 && tecla==48)
   {
    chractual[chractuallon]=tecla;
    chractual[chractuallon+1]='.';
```

```
  chractual[chractuallon+2]='0';
  chractual[chractuallon+3]='\0';
  chractuallon=4;
  primer++;
  auxiliar=String(chractual);
  t1.drawString(auxiliar,10,210);

}else if(tecla==11 && primer==1 && chractual[0]>=51 && chractual[0]<=57)
{
  chractual[chractuallon]='.';
  chractual[chractuallon+1]='\0';
  chractuallon++;
  primer++;
  auxiliar=String(chractual);
  t1.drawString(auxiliar,10,210);

}else if(tecla==11 && primer==2 && chractual[1]>=48 && chractual[1]<=57)
{
  chractual[chractuallon]='.';
  chractual[chractuallon+1]='\0';
  chractuallon++;
  primer++;
  auxiliar=String(chractual);
  t1.drawString(auxiliar,10,210);

}else if(tecla>=48 && tecla<=57 && primer==2 && chractual[1]=='.')
{
  chractual[chractuallon]=tecla;
  chractual[chractuallon+1]='\0';
  chractuallon=4;;
```

```
        primer++;
        auxiliar=String(chractual);
        t1.drawString(auxiliar,10,210);

      }else if(tecla>=48 && tecla<=57 && primer==3 && chractual[0]!=53 &&
chractual[1]!=48)
      {
        chractual[chractuallon]=tecla;
        chractual[chractuallon+1]='\0';
        chractuallon++;
        primer++;
        auxiliar=String(chractual);
        t1.drawString(auxiliar,10,210);

      }else if(tecla>=48 && tecla<=57 && primer==3 && chractual[0]!=53 &&
chractual[1]==48)
      {
        chractual[chractuallon]=tecla;
        chractual[chractuallon+1]='\0';
        chractuallon++;
        primer++;
        auxiliar=String(chractual);
        t1.drawString(auxiliar,10,210);


      }
      else error(115,210,"Error de parametro");
    }else error(165,210,"Fuera de rango");
  }
  if (modos==3)
  {
```

```
t1.setFontColor(GREEN,BLACK);
t1.selectFont(Comic23);
if (chractuallon<2)
{
  if(primer==0 && tecla==49)
  {
    chractual[chractuallon]=tecla;
    chractual[chractuallon+1]='0';
    chractual[chractuallon+2]='0';
    chractuallon=3;
    primer=3;
    auxiliar=String(chractual);
    t1.drawString(auxiliar,10,210);

  }else if(primer==0 && tecla==57)
  {
    chractual[chractuallon]=tecla;
    chractual[chractuallon+1]='\0';
    chractuallon++;
    primer++;
    auxiliar=String(chractual);
    t1.drawString(auxiliar,10,210);

  }else if(primer==1 && tecla>=53 && tecla<=57)
  {
    chractual[chractuallon]=tecla;
    chractual[chractuallon+1]='\0';
    chractuallon++;
    primer++;
    auxiliar=String(chractual);
```

```
      t1.drawString(auxiliar,10,210);

    }else error(115,210,"Error de parametro");
  }else error(165,210,"Fuera de rango");
}
if (modos==4)
{
  t1.setFontColor(GREEN,BLACK);
  t1.selectFont(Comic23);

  if (c_date==0)
  {
    if (chractuallon<2)
    {
      if ((tecla>=48)&&(tecla<=51)&&(primer==0))
      {
        chractual[chractuallon]=tecla;
        chractual[chractuallon+1]='\0';
        chractuallon++;
        primer++;
        auxiliar=String(chractual);
        t1.drawString(auxiliar,10,210);

      }else if ((tecla>=48)&&(tecla<=57)&&(primer==1)&&(chractual[0]!=51))
      {
        chractual[chractuallon]=tecla;
        chractual[chractuallon+1]='\0';
        chractuallon++;
        primer++;
        auxiliar=String(chractual);
```

```
      t1.drawString(auxiliar,10,210);

    }else if ((tecla>=48)&&(tecla<=49)&&(primer==1)&&(chractual[0]==51))
    {
      chractual[chractuallon]=tecla;
      chractual[chractuallon+1]='\0';
      chractuallon++;
      primer++;
      auxiliar=String(chractual);
      t1.drawString(auxiliar,10,210);

    }else error(115,210,"Error de parametro");
  }else error(165,210,"Fuera de rango");
}else if(c_date==1)
{
  if (chractuallon<2)
  {
    if ((tecla>=48)&&(tecla<=49)&&(primer==0))
    {
      chractual[chractuallon]=tecla;
      chractual[chractuallon+1]='\0';
      chractuallon++;
      primer++;
      auxiliar=String(chractual);
      t1.drawString(auxiliar,10,210);
   }else if((primer==1)&&(chractual[0]==48))
    {
      if (c_dd[0]==51)
      {
        if((c_dd[1]==49)&&(tecla==49||tecla==51||tecla==53||tecla==55||tecla==56))
```

```
          {
           chractual[chractuallon]=tecla;
           chractual[chractuallon+1]='\0';
           chractuallon++;
           primer++;
           auxiliar=String(chractual);
           t1.drawString(auxiliar,10,210);
          }else
if((c_dd[1]==48)&&(tecla==49||tecla==51||tecla==52||tecla==53||tecla==54||tecla==55|
|tecla==56||tecla==57))
          {
            chractual[chractuallon]=tecla;
            chractual[chractuallon+1]='\0';
            chractuallon++;
            primer++;
            auxiliar=String(chractual);
            t1.drawString(auxiliar,10,210);

           }else error(115,210,"Error de parametro");
         }else if (c_dd[0]==50)
         {
           if
((c_dd[1]==57)&&(tecla==49||tecla==51||tecla==52||tecla==53||tecla==54||tecla==55||t
ecla==56||tecla==57))
          {
            chractual[chractuallon]=tecla;
            chractual[chractuallon+1]='\0';
            chractuallon++;
            primer++;
            auxiliar=String(chractual);
```

```
      t1.drawString(auxiliar,10,210);

    }else if ((c_dd[1]<=56)&&(tecla>=49)&&(tecla<=57))
    {
      chractual[chractuallon]=tecla;
      chractual[chractuallon+1]='\0';
      chractuallon++;
      primer++;
      auxiliar=String(chractual);
      t1.drawString(auxiliar,10,210);

    }else error(115,210,"Error de parametro");
  }else
  {
      chractual[chractuallon]=tecla;
      chractual[chractuallon+1]='\0';
      chractuallon++;
      primer++;
      auxiliar=String(chractual);
      t1.drawString(auxiliar,10,210);

  }
}else if ((primer==1)&&(chractual[0]==49)&& (tecla>=48) && (tecla<=50))
{
  if (c_dd[0]==51)
  {
    if((c_dd[1]==49)&&(tecla==48||tecla==50))
    {
     chractual[chractuallon]=tecla;
     chractual[chractuallon+1]='\0';
```

```
          chractuallon++;
          primer++;
          auxiliar=String(chractual);
          t1.drawString(auxiliar,10,210);

        }else if((c_dd[1]==48)&&(tecla==48||tecla==49||tecla==50))
        {
          chractual[chractuallon]=tecla;
          chractual[chractuallon+1]='\0';
          chractuallon++;
          primer++;
          auxiliar=String(chractual);
          t1.drawString(auxiliar,10,210);

        }else error(115,210,"Error de parametro");
      }else
      {
          chractual[chractuallon]=tecla;
          chractual[chractuallon+1]='\0';
          chractuallon++;
          primer++;
          auxiliar=String(chractual);
          t1.drawString(auxiliar,10,210);

      }
    }else error(115,210,"Error de parametro");
 }else error(165,210,"Fuera de rango");
}else if (c_date==2)
{
  if (chractuallon<4)
```

```
{
  if ((primer==0)&& (tecla>=50)&& (tecla<=57))
   {
     chractual[chractuallon]=tecla;
     chractual[chractuallon+1]='\0';
     chractuallon++;
     primer++;
     auxiliar=String(chractual);
     t1.drawString(auxiliar,10,210);

   }else if (primer==1)
   {
     chractual[chractuallon]=tecla;
     chractual[chractuallon+1]='\0';
     chractuallon++;
     primer++;
     auxiliar=String(chractual);
     t1.drawString(auxiliar,10,210);

   }else if ((primer>=2)&&(primer<=3)&&(chractual[1]!=48))
   {
     chractual[chractuallon]=tecla;
     chractual[chractuallon+1]='\0';
     chractuallon++;
     primer++;
     auxiliar=String(chractual);
     t1.drawString(auxiliar,10,210);

   }else if ((chractual[1]==48)&&(primer==2)&&(tecla>=49)&&(tecla<=57))
   {
```

```
          chractual[chractuallon]=tecla;
          chractual[chractuallon+1]='\0';
          chractuallon++;
          primer++;
          auxiliar=String(chractual);
          t1.drawString(auxiliar,10,210);

        }else if
((chractual[1]==48)&&(chractual[2]==49)&&(primer==3)&&(tecla>=53)&&(tecla<=57))
        {
          chractual[chractuallon]=tecla;
          chractual[chractuallon+1]='\0';
          chractuallon++;
          primer++;
          auxiliar=String(chractual);
          t1.drawString(auxiliar,10,210);

        }else if
((chractual[1]==48)&&(chractual[2]!=49)&&(primer==3)&&(tecla>=48)&&(tecla<=57))
        {
          chractual[chractuallon]=tecla;
          chractual[chractuallon+1]='\0';
          chractuallon++;
          primer++;
          auxiliar=String(chractual);
          t1.drawString(auxiliar,10,210);

        }else error(115,210,"Error de parametro");
      }else error(165,210,"Fuera de rango");
    }
```

```
 }
 if (modos==5)
 {

  t1.setFontColor(GREEN,BLACK);
  t1.selectFont(Comic23);

  if (chractuallon<2)
  {
    if (c_date==0)
    {
      if ((tecla>=48) && (tecla<=49) && primer==0)
      {
        chractual[chractuallon]=tecla;
        chractual[chractuallon+1]='\0';
        chractuallon++;
        primer++;
        auxiliar=String(chractual);
        t1.drawString(auxiliar,10,210);

      }else if ((primer==0) && ((tecla>=50) && (tecla<=57))) error(115,210,"Error de parametro");
       else if ((primer==1) && (chractual[0]==48) && (tecla>=48) && (tecla<=57))
       {
         chractual[chractuallon]=tecla;
         chractual[chractuallon+1]='\0';
         chractuallon++;
         primer++;
         auxiliar=String(chractual);
         t1.drawString(auxiliar,10,210);
```

```
        }else if ((primer==1) && (chractual[0]==49) && (tecla>=48) && (tecla<=50))
        {
          chractual[chractuallon]=tecla;
          chractual[chractuallon+1]='\0';
          chractuallon++;
          primer++;
          auxiliar=String(chractual);
          t1.drawString(auxiliar,10,210);

        }else if ((primer==1) && (chractual[0]==49) && (tecla>=51) && (tecla<=57)) error(115,210,"Error de parametro");
      }else if (c_date==1)
      {
       if ((tecla>=48) && (tecla<=53) && primer==0)
       {
         chractual[chractuallon]=tecla;
         chractual[chractuallon+1]='\0';
         chractuallon++;
         primer++;
         auxiliar=String(chractual);
         t1.drawString(auxiliar,10,210);

       }else if ((primer==0) && ((tecla>=54) && (tecla<=57))) error(115,210,"Error de parametro");
        else if ((primer==1) && (tecla>=48) && (tecla<=57))
        {
          chractual[chractuallon]=tecla;
          chractual[chractuallon+1]='\0';
          chractuallon++;
```

```
        primer++;
        auxiliar=String(chractual);
        t1.drawString(auxiliar,10,210);
       }
    }
  }else error(165,210,"Fuera de rango");
}
if (modos==6)
{

  t1.setFontColor(GREEN,BLACK);
  t1.selectFont(Comic23);

  if (chractuallon<1)
  {
    if ((tecla>=48)&&(tecla<=53)&&(primer==0))
      {
        chractual[chractuallon]=tecla;
        chractual[chractuallon+1]='\0';
        chractuallon++;
        primer++;
        auxiliar=String(chractual);
        t1.drawString(auxiliar,10,210);

      }else error(115,210,"Error de parametro");
  }else error(165,210,"Fuera de rango");
}
if (modos==7)
{
  t1.setFontColor(GREEN,BLACK);
```

```
    t1.selectFont(Comic23);

    if (chractuallon<2)
    {
      if ((tecla>=48)&&(tecla<=53)&&(primer==0))
        {
          chractual[chractuallon]=tecla;
          chractual[chractuallon+1]='\0';
          chractuallon++;
          primer++;
          auxiliar=String(chractual);
          t1.drawString(auxiliar,10,210);

        }else if ((tecla>=48)&&(tecla<=57)&&(primer==1))
        {
          chractual[chractuallon]=tecla;
          chractual[chractuallon+1]='\0';
          chractuallon++;
          primer++;
          auxiliar=String(chractual);
          t1.drawString(auxiliar,10,210);

        }else error(115,210,"Error de parametro");
    }else error(165,210,"Fuera de rango");
  }
}
```

//Función para mandar los distintos modos y números presionados a la funcion update_caracter//////////////////////////////////////

```
//////////////////////////////////////////////////////////////y puedan ser validados, en esta función también se validan ciertas acciones//////////////////////////////////////////////////
void screen_keyboard(int modo){
  // Modo 1 CO2, 2 es temperatura, 3 Humedad , 4 Facha, 5 hora, 6 Exposición a la radiación hrs
  // 7 Exposición de radiación min

  int punto=0;
  tft.fillScreen(colorFONDO);

  t1.setFontColor(WHITE,BLACK);
  t1.selectFont(Comic23);
  auxiliar="";
  if(modo==1){
    t1.drawString("El parametro actual de",50,10);
    t1.drawString("CO2 es:",80,40);
    auxiliar=(getDecimal(a_co2)+"%");
    t1.drawString(auxiliar,165,40);

  }else if(modo==2){
    t1.drawString("El parametro actual de",50,10);
    t1.drawString("temperatura es:",50,40);
    auxiliar=(getDecimal(a_tempe)+"C");
    t1.drawString(auxiliar,230,40);

  }else if(modo==3){
    t1.drawString("El parametro actual de",50,10);
    t1.drawString("humedad es:",70,40);
    auxiliar=(String (a_humed)+" %");
    t1.drawString(auxiliar,195,40);
```

```
 }else if (modo==4){
  t1.drawString("Introducir  fecha  deseada:",30,10);
  auxiliar=(String (c_dd) + "/"+ String (c_mm) + "/"+ String (c_aaaa));
  t1.drawString(auxiliar,80,40);

 }else if (modo==5){
   t1.drawString("Introducir  hora  deseada:",50,10);
   auxiliar=(String (c_hh) + ":"+ String (c_mn) + " "+ String (c_ampm));
   t1.drawString(auxiliar,70,40);
 }

keyboard();
while (true)
 {
   if(ctp.touched())
   {
      TS_Point p = ctp.getPoint();
      int x = map(p.y, 0, 320, 0, 320);
      int y = map(p.x, 0, 240, 240, 0);
      if((y>=70)&&(y<=110))
      {
        if((x>=5)&&(x<=63))//boton 1
        {
         white_seccion(5,70,58,40);
         keyboard();
         update_caracter('1',modo);
        }
        if((x>=68)&&(x<=126))//boton 2
        {
```

```
    white_seccion(68,70,58,40);
    keyboard();
    update_caracter('2',modo);
   }
   if((x>=131)&&(x<=189))//boton 3
   {
    white_seccion(131,70,58,40);
    keyboard();
    update_caracter('3',modo);
   }
   if((x>=194)&&(x<=252))//boton 4
   {
    white_seccion(194,70,58,40);
    keyboard();
    update_caracter('4',modo);
   }
   if((x>=257)&&(x<=315))//boton 5
   {
    white_seccion(257,70,58,40);
    keyboard();
    update_caracter('5',modo);
   }
 }
if((y>=115)&&(y<=155))
{
  if((x>=5)&&(x<=63))//boton 6
   {
    white_seccion(5,115,58,40);
    keyboard();
    update_caracter('6',modo);
```

```
   }
   if((x>=68)&&(x<=126))//boton 7
   {
    white_seccion(68,115,58,40);
    keyboard();
    update_caracter('7',modo);
   }
   if((x>=131)&&(x<=189))//boton 8
   {
    white_seccion(131,115,58,40);
    keyboard();
    update_caracter('8',modo);
   }
   if((x>=194)&&(x<=252))//boton 9
   {
    white_seccion(194,115,58,40);
    keyboard();
    update_caracter('9',modo);
   }
   if((x>=257)&&(x<=315))//boton 0
   {
    white_seccion(257,115,58,40);
    keyboard();
    update_caracter('0',modo);
   }
}
if((y>=160)&&(y<=200))
{
  if((x>=5)&&(x<=63))//boton punto
   {
```

```
        if ((punto==0)&&((modo==1)||(modo==2)))
        {
          white_seccion(5,160,58,40);
          keyboard();
          punto=1;
          update_caracter(11,modo);
        }
        if((punto==1)&& (modo==2)&& chractual[0]>=49 && chractual[0]<=50 &&
chractual[2]==00)punto=0;
        if(modo==5){
          white_seccion(5,160,58,40);
          if (ampm==0)
          {
            ampm=1;
            c_ampm[0]='p';c_ampm[1]='.';c_ampm[2]='m';c_ampm[3]='.';c_ampm[4]='\0';
            t1.setFontColor(WHITE,BLACK);
            t1.selectFont(Comic23);
            auxiliar=(String (c_ampm));
            t1.drawString(auxiliar,150,40);
            keyboard();
            update_caracter(11,modo);
          }else //if(modo!=6 && modo!=7)
          {
            ampm=0;
            c_ampm[0]='a';c_ampm[1]='.';c_ampm[2]='m';c_ampm[3]='.';c_ampm[4]='\0';

            t1.setFontColor(WHITE,BLACK);
            t1.selectFont(Comic23);
            auxiliar=(String (c_ampm));
            t1.drawString(auxiliar,150,40);
```

```
      keyboard();
      update_caracter(12,modo);
    }
  }
}
if((x>=68)&&(x<=126))//boton Clr
{
    white_seccion(68,160,58,40);
    keyboard();
    for (int r=0;r<=5;r++) chractual[r]='\0';
    chractuallon=0;
    primer=0;
    punto=0;
    tft.fillRect(10,210,100,30,BLACK);
}
if((x>=131)&&(x<=189))//boton Ent
{
 white_seccion(131,160,58,40);
 keyboard();
 if (chractuallon>0)
 {
   t1.setFontColor(WHITE,BLACK);
   t1.selectFont(Comic23);

   if (modo==1)
   {
     tft.fillRect(10,210,100,30,BLACK);
     for (int s=0; s<chractuallon+1;s++)
     {
       chranterior[s]=chractual[s];
```

```
  }
  for (int r=0;r<=5;r++) chractual[r]='\0';
  chractuallon=0;
  primer=0;
  punto=0;
  t1.drawString("CO2  es:",80,40);
  auxiliar=chranterior;
  auxiliar=(auxiliar+"%");
  tft.fillRect(165,40,100,30,BLACK);
  t1.drawString(auxiliar,165,40);
  a_co2=atof(chranterior);
  if (a_co2>0.5) nivelCo2=a_co2-0.5;
  else nivelCo2=0;
  ardSetpoint=a_co2*100;
  Wire.beginTransmission(arduino);
  Wire.write(0x2);
  Wire.write(ardSetpoint>>8);
  Wire.write(ardSetpoint & 255);
  Wire.endTransmission();
}else if (modo==2)
{
  tft.fillRect(10,210,100,30,BLACK);
  for (int s=0; s<chractuallon+1;s++)
  {
    chranterior[s]=chractual[s];
  }
  for (int r=0;r<=5;r++) chractual[r]='\0';
  chractuallon=0;
  primer=0;
  punto=0;
```

```
  t1.drawString("temperatura  es:",50,40);
  auxiliar=chranterior;
  auxiliar=(auxiliar+" C");
  tft.fillRect(230,40,100,30,BLACK);
  t1.drawString(auxiliar,230,40);
  a_tempe=atof(chranterior);
  if (a_tempe>5)
  {
    nivelTemp1=a_tempe-5;
    nivelTemp2=a_tempe+2;
  }
}else if (modo==3)
{
  tft.fillRect(10,210,100,30,BLACK);
  for (int s=0; s<chractuallon+1;s++)
  {
    chranterior[s]=chractual[s];
  }
  for (int r=0;r<=5;r++) chractual[r]='\0';
  chractuallon=0;
  primer=0;
  punto=0;
  t1.drawString("humedad  es:",70,40);
  a_humed=atoi(chranterior);
  if (a_humed>=90)nivelHumedad=a_humed-5;
  auxiliar=(String (a_humed)+" %");
  tft.fillRect(195,40,100,30,BLACK);
  t1.drawString(auxiliar,195,40);
}else if (modo==4)
{
```

```
if (primer==2)
{
  tft.fillRect(10,210,100,30,BLACK);
  if (c_date==0) for (int s=0; s<chractuallon+1;s++) c_dd[s]=chractual[s];
  else if (c_date==1) for (int s=0; s<chractuallon+1;s++) c_mm[s]=chractual[s];
  for (int r=0;r<=5;r++) chractual[r]='\0';
  chractuallon=0;
  primer=0;
  c_date++;
  if (c_date==2)
  {
    dd=atoi(c_dd);
    mm=atoi(c_mm);
    DateTime T = rtc.now();
    rtc.adjust(DateTime(T.year(), mm, dd, T.hour(), T.minute(), T.second()));
  }
  auxiliar=(String (c_dd) + "/"+ String (c_mm) + "/"+ String (c_aaaa));
  tft.fillRect(70,40,200,30,BLACK);
  t1.drawString(auxiliar,70,40);
}else if (primer==4)
{
  tft.fillRect(10,210,100,30,BLACK);
  if (c_date==2) for (int s=0; s<chractuallon+1;s++) c_aaaa[s]=chractual[s];
  for (int r=0;r<=5;r++) chractual[r]='\0';
  chractuallon=0;
  primer=0;
  c_date++;
  aaaa=atoi(c_aaaa);
  DateTime T = rtc.now();
  rtc.adjust(DateTime(aaaa, mm, dd, T.hour(), T.minute(), T.second()));
```

```
      tft.fillRect(5,200,100,40,BLACK);
      auxiliar=(String (c_dd) + "/"+ String (c_mm) + "/"+ String (c_aaaa));
      tft.fillRect(70,40,200,30,BLACK);
      t1.drawString(auxiliar,70,40);
      if (c_date==3) screen_six();
    }else error(123,210,"Falta de parametro");
  }else if (modo==5)
  {
    if (primer==2)
    {
      if (c_date==0)for (int s=0; s<chractuallon+1;s++)c_hh[s]=chractual[s];
      else if (c_date==1) for (int s=0; s<chractuallon+1;s++) c_mn[s]=chractual[s];
      if (c_hh[0]==49 && c_hh[1]==50 && ampm==0)
      {
        c_hh[0]='0';
        c_hh[1]='0';
      }
      for (int r=0;r<=5;r++) chractual[r]='\0';
      chractuallon=0;
      primer=0;
      c_date++;
      tft.fillRect(10,210,100,30,BLACK);
      auxiliar=(String (c_hh) + ":"+ String (c_mn) + " "+ String (c_ampm));
      tft.fillRect(70,40,200,30,BLACK);
      t1.drawString(auxiliar,70,40);

    }
  }
 }
}
```

```
if((x>=194)&&(x<=252))//boton Reg
{
 if (modo<4)
 {
   white_seccion(194,160,58,40);
   keyboard();
   for (int r=0;r<=5;r++) chractual[r]='\0';
   chractuallon=0;
   primer=0;
   punto=0;
   screen_two();
 }else if ((modo==4)&&((c_date==0)||(c_date==2)))//boton Reg ahora se convierte en Hrs.
 {
   white_seccion(194,160,58,40);
   keyboard();
   for (int r=0;r<=5;r++) chractual[r]='\0';
   chractuallon=0;
   primer=0;
   punto=0;
   c_date=0;
   screen_six();
 }else if ((modo==4)&&(c_date==1))
 {
   white_seccion(194,160,58,40);
   keyboard();
   for (int r=0;r<=5;r++) chractual[r]='\0';
   chractuallon=0;
   primer=0;
   punto=0;
```

```
            DateTime T = rtc.now();
            if ((c_dd[0]<=50) && (c_dd[1]<=56))
            {
              dd=atoi(c_dd);
              rtc.adjust(DateTime(T.year(), T.month(), dd, T.hour(), T.minute(),
T.second()));
              screen_six();
            }else                                                              if
((((c_dd[0]==50)&&(c_dd[1]==57))||((c_dd[0]==51)&&(c_dd[1]==48)))&&(T.month()!=2
))
            {
              dd=atoi(c_dd);
              rtc.adjust(DateTime(T.year(), T.month(), dd, T.hour(), T.minute(),
T.second()));
              screen_six();
            }else
if((c_dd[0]==50)&&(c_dd[1]==57)&&((T.month()!=2)||(T.month()!=4)||(T.month()!=6)||(
T.month()!=9)||(T.month()!=11)))
            {
              dd=atoi(c_dd);
              rtc.adjust(DateTime(T.year(), T.month(), dd, T.hour(), T.minute(),
T.second()));
              screen_six();
            }else error(5,210,"Dia no corresponde a mes actual");
          }else if ((modo==5))
          {
            white_seccion(194,160,58,40);
            keyboard();
            for (int r=0;r<=5;r++) chractual[r]='\0';
            chractuallon=0;
```

```
          primer=0;
          punto=0;
          ampm=3;
          DateTime T = rtc.now();
          if (c_date==1)
          {
            hh=atoi(c_hh);
            if (ampm==1)
            {
              hh+=12;
            }
            rtc.adjust(DateTime(T.year(),      T.month(),      T.day(),      hh,      T.minute(),
T.second()));
          }else if (c_date==2)
          {
            hh=atoi(c_hh);
            mn=atoi(c_mn);
            if (ampm==1)
            {
              hh+=12;
            }
            rtc.adjust(DateTime(T.year(), T.month(), T.day(), hh, mn, T.second()));
          }
          screen_five();
        }
       }
       if((x>=257)&&(x<=315))//boton Ini
       {
         if (modo==4 && c_date==1)
        {
```

```
          DateTime T = rtc.now();
          if ((c_dd[0]<=50) && (c_dd[1]<=56))
          {
            dd=atoi(c_dd);
            rtc.adjust(DateTime(T.year(), T.month(), dd, T.hour(), T.minute(),
T.second()));
            white_seccion(257,160,58,40);
            keyboard();
            for (int r=0;r<=5;r++) chractual[r]='\0';
            chractuallon=0;
            primer=0;
            punto=0;
            tft.fillScreen(colorFONDO);
            screen_one();
          }else                                                                    if
((((c_dd[0]==50)&&(c_dd[1]==57))||((c_dd[0]==51)&&(c_dd[1]==48)))&&(T.month()!=2
))
          {
            dd=atoi(c_dd);
            rtc.adjust(DateTime(T.year(), T.month(), dd, T.hour(), T.minute(),
T.second()));
            white_seccion(257,160,58,40);
            keyboard();
            for (int r=0;r<=5;r++) chractual[r]='\0';
            chractuallon=0;
            primer=0;
            punto=0;
            tft.fillScreen(colorFONDO);
            screen_one();
```

```
          }else
if((c_dd[0]==50)&&(c_dd[1]==57)&&((T.month()!=2)||(T.month()!=4)||(T.month()!=6)||(
T.month()!=9)||(T.month()!=11)))
          {
            dd=atoi(c_dd);
            rtc.adjust(DateTime(T.year(), T.month(), dd, T.hour(), T.minute(),
T.second()));
            white_seccion(257,160,58,40);
            keyboard();
            for (int r=0;r<=5;r++) chractual[r]='\0';
            chractuallon=0;
            primer=0;
            punto=0;
            tft.fillScreen(colorFONDO);
            screen_one();
          }else error(5,210,"Dia no corresponde a mes actual");
        }else if (modo==5)
        {
           white_seccion(257,160,58,40);
           keyboard();
           for (int r=0;r<=5;r++) chractual[r]='\0';
           chractuallon=0;
           primer=0;
           punto=0;
           DateTime T = rtc.now();
           if (c_date==1)
           {
             hh=atoi(c_hh);
             if (ampm==1)
             {
```

```
            hh+=12;
           }
           rtc.adjust(DateTime(T.year(), T.month(), T.day(), hh, T.minute(),
T.second()));
          }else if (c_date==2)
          {
           hh=atoi(c_hh);
           mn=atoi(c_mn);
           if (ampm==1)
           {
            hh+=12;
           }
           rtc.adjust(DateTime(T.year(), T.month(), T.day(), hh, mn, T.second()));
          }
          tft.fillScreen(colorFONDO);
          screen_one();
        }
        else
        {
          white_seccion(257,160,58,40);
          keyboard();
          for (int r=0;r<=5;r++) chractual[r]='\0';
          chractuallon=0;
          primer=0;
          punto=0;
          tft.fillScreen(colorFONDO);
          screen_one();
        }
       }
    }
```

```
    }
  }
}

////////////////////////////////////////////////////////Función para pantalla dos y poder ir a configurar los distintos parámetros de CO2, temperaturay Humedad//////////////////////////////
void screen_two()
{
  int pulso_correcto=0;

  tft.fillScreen(colorFONDO);
  t1.selectFont(Comic23);
  t1.setFontColor(WHITE,BLACK);
  t1.drawString("Parametro a configurar:",35,20);

  t1.selectFont(Comic20);
  tft.fillRoundRect(5,80,60,50,5,GREEN);
  tft.drawRoundRect(5,80,60,50,5,WHITE);
  t1.setFontColor(BLACK,GREEN);
  t1.drawString("CO2",15,95);

  tft.fillRoundRect(75,80,130,50,5,GREEN);
  tft.drawRoundRect(75,80,130,50,5,WHITE);
  t1.setFontColor(BLACK,GREEN);
  t1.drawString("Temperatura",80,95);

  tft.fillRoundRect(215,80,100,50,5,GREEN);
  tft.drawRoundRect(215,80,100,50,5,WHITE);
  t1.setFontColor(BLACK,GREEN);
  t1.drawString("Humedad",225,95);
```

```
tft.fillRoundRect(5,140,310,50,5,GREEN);
tft.drawRoundRect(5,140,310,50,5,WHITE);
t1.setFontColor(BLACK,GREEN);
t1.drawString("Inicio",130,155);

while (pulso_correcto==0){
  if(ctp.touched())
  {
     TS_Point p = ctp.getPoint();
     int x = map(p.y, 0, 320, 0, 320);
     int y = map(p.x, 0, 240, 240, 0);
     if((y>=80)&&(y<=130))
     {
       if((x>=5)&&(x<=65)){
         pulso_correcto=1;
         white_seccion(5,80,60,50);
       }else if((x>=75)&&(x<=205)){
         pulso_correcto=2;
         white_seccion(75,80,130,50);
       }else if((x>=215)&&(x<=315)){
         pulso_correcto=3;
         white_seccion(215,80,100,50);
       }

     }else if((y>=140)&&(y<=190)){
       if((x>=5)&&(x<=315)){
         pulso_correcto=4;
         white_seccion(5,140,315,50);
       }
```

```
      }
    }
  }
  if(pulso_correcto==1)screen_keyboard(1);
  else if(pulso_correcto==2)screen_keyboard(2);
  else if(pulso_correcto==3)screen_keyboard(3);
  else {
    tft.fillScreen(colorFONDO);
    screen_one();
  }
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
void  screen_three(){

  int apuntador=0,desp=0;
  double incrementos=0,niveles=0;
  String nivelstr;
  float valores[17];
  float valormax=0;
  float valormin=100;
  float numero=0;
  char valor[5]="";
  char horanew[5]="";
  char horaold[5]="";
  int apun=0, inde=0;
  int lineasejeX[17];
  int lineasejeY[17];
  tft.fillScreen(colorFONDO);
  t1.selectFont(Comic15);
```

```
t1.setFontColor(WHITE,BLACK);


File dataCO2= SD.open("CO2.txt");
if (dataCO2){
  int carac=dataCO2.size();

  while (apuntador<carac)
  {
    tft.drawLine(42,0,42,190,GREEN);
    tft.drawLine(42,190,320,190,GREEN);
    inde=0;
    desp=0;
    while (inde<=17)
    {
      dataCO2.seek(apuntador);
      while (dataCO2.peek()!='\t')
      {
        valor[apun]=dataCO2.peek();
        apun++;
        apuntador++;
        dataCO2.seek(apuntador);
      }
      apun=0;
      while (dataCO2.peek()!='\n')
      {
        apuntador++;
        dataCO2.seek(apuntador);
        if ((dataCO2.peek()!='\n')&&(apun<5)) horanew[apun]=dataCO2.peek();
        apun++;
```

```
}
horanew[5]='\0';
apun=0;
apuntador++;
if (horanew[4]!=horaold[4])
{
  for (int i=0;i<5;i++)
  {
    horaold[i]=horanew[i];
  }
  horaold[5]='\0';
  uint8_t rotation=2;
  tft.setRotation((iliRotation)rotation);
  t1.drawString(horaold,5,44+desp);
  desp=desp+15;

  numero=atof(valor);
  if (numero>valormax) valormax=numero;
  if (numero<valormin) valormin=numero;
  valores[inde]=numero;
  inde++;
  Serial.print("La variable inde tiene:");
  Serial.println(inde);
}
if (inde==18)
{
  incrementos=(valormax-valormin)/10;
  tft.setRotation(iliRotation270);
  niveles=valormax;
```

```
    nivelstr=getDecimal2(valormax);
    t1.drawString(nivelstr,0,0);
    for (int j=1;j<11;j++)
    {
      niveles=niveles-incrementos;
      nivelstr=getDecimal2(niveles);
      t1.drawString(nivelstr,0,19*j);
    }
    nivelstr=getDecimal2(valormin);
    t1.drawString(nivelstr,0,190);
  }
  if (apuntador>=carac)
  {
    incrementos=(valormax-valormin)/10;
    tft.setRotation(iliRotation270);
    niveles=valormax;
    nivelstr=getDecimal2(valormax);
    t1.drawString(nivelstr,0,0);
    for (int j=1;j<11;j++)
    {
      niveles=niveles-incrementos;
      nivelstr=getDecimal2(niveles);
      t1.drawString(nivelstr,0,19*j);
    }
    nivelstr=getDecimal2(valormin);
    t1.drawString(nivelstr,0,190);
    break;
  }
}
if (inde==18) inde=inde-1;
```

```
    if (inde>1)
    {
      for (int t=0;t<inde;t++)
      {
        if (valores[t]>=valormin && valores[t]<=((valormin+incrementos)/2))
        {
          lineasejeX[t]=45+16*t;
          lineasejeY[t]=185;
        }
        else                    if(valores[t]>((valormin+incrementos)/2)                    &&
valores[t]<=(valormin+incrementos))
        {
          lineasejeX[t]=45+16*t;
          lineasejeY[t]=176;
        }
        else                      if(valores[t]>(valormin+incrementos)                    &&
valores[t]<=(((valormin+incrementos)/2)+incrementos*2))
        {
          lineasejeX[t]=45+16*t;
          lineasejeY[t]=165;
        }
        else          if(valores[t]>(((valormin+incrementos)/2)+incrementos*2)          &&
valores[t]<=(valormin+(incrementos*2)))
        {
          lineasejeX[t]=45+16*t;
          lineasejeY[t]=156;
        }
        else                    if(valores[t]>(valormin+(incrementos*2))                    &&
valores[t]<=(((valormin+incrementos)/2)+incrementos*3))
        {
```

```
        lineasejeX[t]=45+16*t;
        lineasejeY[t]=147;
      }
      else      if(valores[t]>(((valormin+incrementos)/2)+incrementos*3)      &&
valores[t]<=(valormin+(incrementos*3)))
      {
        lineasejeX[t]=45+16*t;
        lineasejeY[t]=138;
      }
      else            if(valores[t]>(valormin+(incrementos*3))            &&
valores[t]<=(((valormin+incrementos)/2)+incrementos*4))
      {
        lineasejeX[t]=45+16*t;
        lineasejeY[t]=129;
      }
      else      if(valores[t]>(((valormin+incrementos)/2)+incrementos*4)      &&
valores[t]<=(valormin+(incrementos*4)))
      {
        lineasejeX[t]=45+16*t;
        lineasejeY[t]=119;

      }
      else            if(valores[t]>(valormin+(incrementos*4))            &&
valores[t]<=(((valormin+incrementos)/2)+incrementos*5))
      {
        lineasejeX[t]=45+16*t;
        lineasejeY[t]=109;
      }
      else      if(valores[t]>(((valormin+incrementos)/2)+incrementos*5)      &&
valores[t]<=(valormin+(incrementos*5)))
```

```
        {
          lineasejeX[t]=45+16*t;
          lineasejeY[t]=100;
        }
        else                    if(valores[t]>(valormin+(incrementos*5))                 &&
valores[t]<=(((valormin+incrementos)/2)+incrementos*6))
        {
          lineasejeX[t]=45+16*t;
          lineasejeY[t]=90;
        }
        else          if(valores[t]>(((valormin+incrementos)/2)+incrementos*6)          &&
valores[t]<=(valormin+(incrementos*6)))
        {
          lineasejeX[t]=45+16*t;
          lineasejeY[t]=81;
        }
        else                    if(valores[t]>(valormin+(incrementos*6))                 &&
valores[t]<=(((valormin+incrementos)/2)+incrementos*7))
        {
          lineasejeX[t]=45+16*t;
          lineasejeY[t]=71;
        }
        else          if(valores[t]>(((valormin+incrementos)/2)+incrementos*7)          &&
valores[t]<=(valormin+(incrementos*7)))
        {
          lineasejeX[t]=45+16*t;
          lineasejeY[t]=62;
        }
        else                    if(valores[t]>(valormin+(incrementos*7))                 &&
valores[t]<=(((valormin+incrementos)/2)+incrementos*8))
```

```
        {
          lineasejeX[t]=45+16*t;
          lineasejeY[t]=52;
        }
        else        if(valores[t]>(((valormin+incrementos)/2+incrementos*8))        &&
valores[t]<=(valormin+(incrementos*8)))
        {
          lineasejeX[t]=45+16*t;
          lineasejeY[t]=43;
        }
        else                if(valores[t]>(valormin+(incrementos*8))                &&
valores[t]<=(((valormin+incrementos)/2)+incrementos*9))
        {
          lineasejeX[t]=45+16*t;
          lineasejeY[t]=33;
        }
        else        if(valores[t]>(((valormin+incrementos)/2)+incrementos*9)        &&
valores[t]<=(valormin+(incrementos*9)))
        {
          lineasejeX[t]=45+16*t;
          lineasejeY[t]=24;
        }
        else                if(valores[t]>(valormin+(incrementos*9))                &&
valores[t]<=(((valormin+incrementos)/2)+incrementos*10))
        {
          lineasejeX[t]=45+16*t;
          lineasejeY[t]=14;
        }
        else
        {
```

```
          lineasejeX[t]=45+16*t;
          lineasejeY[t]=4;
        }
       }
   for (int q=0;q<inde;q++) if (q+1<inde)tft.drawLine(lineasejeX[q],lineasejeY[q],lineasejeX[q+1],lineasejeY[q+1],WHITE);
   }

     delay(10000);
    tft.fillScreen(colorFONDO);
    t1.selectFont(Comic15);
    t1.setFontColor(WHITE,BLACK);
   }

     dataCO2.close();
     tft.setRotation(iliRotation270);
 }

 else{
   Serial.println("El archivo CO2.txt no se abrio correctamente");
 }
}
////////////////////////////////////////////////////////////Función para configurar la exposición de radiciaión dentro de la cámara de incubación//////////////////////////////////////////////////
void  screen_four(){

 SD.remove("CO2.txt");
 SD.remove("Temper.txt");
 SD.remove("Humedad.txt");
```

```
}
////////////////////////////////////////////////////////////////////////Función para configuración de hora la pantalla correspondiente//////////////////////////////////////////////////////////
void  screen_five(){

  //Hrs para configuración de hora en el teclado
  c_date=0;
  hrs=1;
  tft.fillScreen(colorFONDO);
  screen_keyboard(4);

}
////////////////////////////////////////////////////////////////////////Función que inicia las variables para minutos y horas y después invoca la función a configurar//////////////////////////////
void screen_six()
{

  c_date=0;
  hrs=0;
  ampm=0;
  for (int s=0;s<2;s++)
  {
    c_hh[s]='H';
    c_hh[s+1]='\0';
    c_mn[s]='M';
    c_mn[s+1]='\0';
  }
  c_ampm[0]='a';c_ampm[1]='.';c_ampm[2]='m';c_ampm[3]='.';c_ampm[4]='\0';
  tft.fillScreen(colorFONDO);
```

```
  screen_keyboard(5);
}

////////////////////////////////Función para pintar los botones de la pantalla principal de CO2, Temperatura y Humedad y saber que parámetro sera visualizado////////////////////////////////
void botonesinicio()
{
  t1.selectFont(Comic23);
  if(modo_co2)
  {
    tft.fillRoundRect(0,0,60,50,5,RED);
    t1.setFontColor(WHITE,RED);
  }
  else
  {
    tft.fillRoundRect(0,0,60,50,5,DARKGREEN);
    t1.setFontColor(WHITE,DARKGREEN);
  }
  tft.drawRoundRect(0,0,60,50,5,WHITE);
  t1.drawString("CO2",7,15);

  if(modo_temperatura)
  {
    tft.fillRoundRect(60,0,150,50,5,RED);
    t1.setFontColor(WHITE,RED);
  }
  else
  {
    tft.fillRoundRect(60,0,150,50,5,DARKGREEN);
    t1.setFontColor(WHITE,DARKGREEN);
```

```
  }
  tft.drawRoundRect(60,0,150,50,5,WHITE);
  t1.drawString("Temperatura",67,15);

  if(modo_humedad)
  {
    tft.fillRoundRect(210,0,110,50,5,RED);
    t1.setFontColor(WHITE,RED);
  }
  else
  {
    tft.fillRoundRect(210,0,110,50,5,DARKGREEN);
    t1.setFontColor(WHITE,DARKGREEN);
  }
  tft.drawRoundRect(210,0,110,50,5,WHITE);
  t1.drawString("Humedad",217,15);
}
////////////////////////////////////////////////////////////////////////////Boton de Alarma Activada(Buzzer)////////////////////////////////////////////////////////////////////////////
void botonAlarma()
{
  t1.selectFont(Comic23);
  if(modo_alarma)
  {
    tft.fillRoundRect(240,105,80,51,5,RED);
    t1.setFontColor(WHITE,RED);
    tft.drawRoundRect(240,105,80,51,5,WHITE);
    Wire.beginTransmission(arduino);
    Wire.write(0x3);
    Wire.endTransmission();
```

```
    t1.drawString("Alarma",245,108);
    t1.drawString("Activa",248,128);
  }
  else
  {
    tft.fillRoundRect(240,105,80,51,5,DARKGREEN);
    t1.setFontColor(WHITE,DARKGREEN);
    tft.drawRoundRect(240,105,80,51,5,WHITE);
    Wire.beginTransmission(arduino);
    Wire.write(0x6);
    Wire.endTransmission();
    t1.drawString("Alarma",245,108);
    t1.drawString("Desact",243,128);
  }
}
////////////////////////////////////////////////////////////Función que invoca las distintas funciones para obtener los datos de CO2,temperatura y humedad////////////////////////////////////
void sensor(int take)
{
  if (take<=16)
  {
    tempValuePro+=dht.readTemperature();
    rhValuePro+=dht.readHumidity();
  }
  if (take==0)
  {
    wakeSensor();
    initPoll();
  }else if (take==16)
  {
```

```
tempValue=tempValuePro/17;
rhValue=rhValuePro/17;
tempValuePro=0;
rhValuePro=0;
wakeSensor();
co2Value = readCo2();

if(co2Value >= 0)
{

  t1.selectFont(Old75);
  t1.setFontColor(GREENYELLOW,BLACK);
  co2Value*=0.001;
  arduinoCo2Value=co2Value*100;
  Wire.beginTransmission(arduino);
  Wire.write(0x1);
  Wire.write(arduinoCo2Value>>8);
  Wire.write(arduinoCo2Value & 255);
  Wire.endTransmission();
  archMicroSD(1);
  archMicroSD(2);
  archMicroSD(3);
  alarmas(co2Value,tempValue,rhValue);
  CO2=(getDecimal(co2Value)+"%");
  TEMPERATURA=(getDecimal(tempValue)+"C");
  HUMEDAD=(getDecimal(rhValue)+"%");

  if (modo_co2)t1.drawString(CO2,10,90);
  else if (modo_temperatura)t1.drawString(TEMPERATURA,10,90);
  else if (modo_humedad)t1.drawString(HUMEDAD,10,90);
```

```
    t1.selectFont(Comic15);
    t1.setFontColor(WHITE,BLACK);
    //t1.drawString(INDUCTOR,10,160);
   }else
   {
    tft.fillRect(10,90,220,70,BLACK);
    t1.selectFont(Comic60);
    t1.setFontColor(RED,BLACK);
    t1.drawString("Failed",10,90);
   }
  }
}
////////////////////////////////////////////////////////////Función que actualiza cada valor que se visualiza en la pantalla principal////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////CO2, temperatura, humedad, fecha y hora////////////////////////////////////////////////////////////////////////////
void updatevalue()
{

 DateTime T=rtc.now();
 if (oldd!=T.day())
    {
     t1.selectFont(Comic15);
     t1.setFontColor(WHITE,BLACK);
     if (T.day()>=10 && T.month()>=10)DDMMAA=(DiasSem[T.dayOfWeek()] + " "+String(T.day())+"/"+String(T.month())+"/"+String(T.year()));
     else if (T.day()<10 && T.month()>=10)DDMMAA=(DiasSem[T.dayOfWeek()] + " "+"0"+String(T.day())+"/"+String(T.month())+"/"+String(T.year()));
     else if (T.day()>=10 && T.month()<10)DDMMAA=(DiasSem[T.dayOfWeek()]+ " "+String(T.day())+"/"+"0"+String(T.month())+"/"+String(T.year()));
```

```
      else DDMMAA=(DiasSem[T.dayOfWeek()] + "
"+"0"+String(T.day())+"/"+"0"+String(T.month())+"/"+String(T.year()));
       t1.drawString(DDMMAA,200,60);
       oldd=T.day();
     }
  if (olds!=T.second())
     {
       sensor(tak);
       tak++;
       if (tak==26)tak=0;
       t1.selectFont(Comic15);
       t1.setFontColor(WHITE,BLACK);
       if (T.hour()<12)
       {
         if (T.minute()>=10 &&
T.second()>=10)HHMM=(String(T.hour())+":"+String(T.minute())+":"+String(T.second(
)));
        else if (T.minute()>=10 &&
T.second()<10)HHMM=(String(T.hour())+":"+String(T.minute())+":"+"0"+String(T.seco
nd()));
        else if (T.minute()<10 &&
T.second()>=10)HHMM=(String(T.hour())+":"+"0"+String(T.minute())+":"+String(T.sec
ond()));
        else
HHMM=(String(T.hour())+":"+"0"+String(T.minute())+":"+"0"+String(T.second()));
        t1.drawString(HHMM,237,80);
        tft.fillRect(300,80,20,20,BLACK);
        t1.selectFont(Comic10);
        t1.drawString("a.m.",300,80);
```

```
      }else if (T.hour()==12)
      {
        if (T.minute()>=10 &&
T.second()>=10)HHMM=(String(T.hour())+":"+String(T.minute())+":"+String(T.second(
)));
      else if (T.minute()>=10 &&
T.second()<10)HHMM=(String(T.hour())+":"+String(T.minute())+":"+"0"+String(T.seco
nd()));
      else if (T.minute()<10 &&
T.second()>=10)HHMM=(String(T.hour())+":"+"0"+String(T.minute())+":"+String(T.sec
ond()));
      else
HHMM=(String(T.hour())+":"+"0"+String(T.minute())+":"+"0"+String(T.second()));
      t1.drawString(HHMM,237,80);
      tft.fillRect(300,80,20,20,BLACK);
      t1.selectFont(Comic10);
      t1.drawString("p.m.",300,83);

     }else
     {
      pmhours=T.hour()-12;
      if (T.minute()>=10 &&
T.second()>=10)HHMM=(String(pmhours)+":"+String(T.minute())+":"+String(T.second
()));
      else if (T.minute()>=10 &&
T.second()<10)HHMM=(String(pmhours)+":"+String(T.minute())+":"+"0"+String(T.seco
nd()));
      else if (T.minute()<10 &&
T.second()>=10)HHMM=(String(pmhours)+":"+"0"+String(T.minute())+":"+String(T.sec
ond()));
```

```
        else
HHMM=(String(pmhours)+":"+"0"+String(T.minute())+":"+"0"+String(T.second()));
        t1.drawString(HHMM,237,80);
        t1.selectFont(Comic10);
        t1.drawString("p.m.",300,83);
       }
       olds=T.second();
     }
}

////////////////////////////////////////////////////////////////////////////Función de pantalla principal, incia variables y valores actuales seleccionados//////////////////////////////////////
void screen_one()
{
  int pulso_correcto=0;

  for (int s=0;s<4;s++)
  {
    c_aaaa[s]='A';
    c_aaaa[s+1]='\0';
  }
  for (int s=0;s<2;s++)
  {
    c_mm[s]='M';
    c_mm[s+1]='\0';
    c_dd[s]='D';
    c_dd[s+1]='\0';
    c_hh[s]='H';
    c_hh[s+1]='\0';
    c_mn[s]='M';
```

```
  c_mn[s+1]='\0';
}

c_date=0;
hrs=0;
ampm=3;
oldd=70;
botonesinicio();
botonAlarma();
t1.setFontColor(WHITE,DARKGREEN);
tft.fillRoundRect(0,190,70,50,5,DARKGREEN);
t1.drawString("Confi.",7,194);
t1.drawString("Ambie.",3,215);
tft.drawRoundRect(0,190,70,50,5,WHITE);

tft.fillRoundRect(70,190,70,50,5,DARKGREEN);
t1.drawString("Grafi.",77,194);
t1.drawString("CO2",83,215);
tft.drawRoundRect(70,190,70,50,5,WHITE);

tft.fillRoundRect(140,190,70,50,5,DARKGREEN);
t1.drawString("Borra",147,194);
t1.drawString("Memo.",144,215);
tft.drawRoundRect(140,190,70,50,5,WHITE);

tft.fillRoundRect(210,190,110,50,5,DARKGREEN);
t1.drawString("Confi.",235,194);
t1.drawString("Fech./Hrs.",212,215);
tft.drawRoundRect(210,190,110,50,5,WHITE);
```

```
t1.selectFont(Old75);
t1.setFontColor(GREENYELLOW,BLACK);

if (modo_co2)
{
  CO2=(getDecimal(co2Value)+" %");
  t1.drawString(CO2,10,90);
}else if (modo_temperatura)
{
  TEMPERATURA=(getDecimal(tempValue)+" C");
  t1.drawString(TEMPERATURA,10,90);
}else
{
  HUMEDAD=(getDecimal(rhValue)+" %");
  t1.drawString(HUMEDAD,10,90);
}

while (pulso_correcto==0)
{
  updatevalue();

  if(ctp.touched())
  {
    TS_Point p= ctp.getPoint();
    int x = map(p.y, 0, 320, 0, 320);
    int y = map(p.x, 0, 240, 240, 0);
    if((y>=0)&&(y<=50))
    {
      if((x>=0)&&(x<=60))
      {
```

```
  modo_co2=true;
  modo_temperatura=false;
  modo_humedad=false;
  white_seccion(0,0,60,50);
  botonesinicio();
  tft.fillRect(10,90,220,70,BLACK);
  t1.selectFont(Old75);
  t1.setFontColor(GREENYELLOW,BLACK);
  CO2=(getDecimal(co2Value)+" %");
  t1.drawString(CO2,10,90);
}
else if((x>=61)&&(x<=210))
{
  modo_co2=false;
  modo_temperatura=true;
  modo_humedad=false;
  white_seccion(60,0,150,50);
  botonesinicio();
  tft.fillRect(10,90,220,70,BLACK);
  t1.selectFont(Old75);
  t1.setFontColor(GREENYELLOW,BLACK);
  TEMPERATURA=(getDecimal(tempValue)+" C");
  t1.drawString(TEMPERATURA,10,90);

}
else if((x>=211)&&(x<=320))
{
  modo_co2=false;
  modo_temperatura=false;
  modo_humedad=true;
```

```
    white_seccion(210,0,110,50);
    botonesinicio();
    tft.fillRect(10,90,220,70,BLACK);
    t1.selectFont(Old75);
    t1.setFontColor(GREENYELLOW,BLACK);
    HUMEDAD=(getDecimal(rhValue)+" %");
    t1.drawString(HUMEDAD,10,90);

  }
}
else if((y>=105)&&(y<=156)&&(x>=240)&&(x<=320))
{
  if (modo_alarma)modo_alarma=false;
  else modo_alarma=true;
  white_seccion(240,105,80,51);
  botonAlarma();
}
else if((y>=190)&&(y<=240))
{
  if((x>=0)&&(x<=70))
  {
    pulso_correcto=1;
    white_seccion(0,190,70,50);
  }
  else if((x>=71)&&(x<=140))
  {
    pulso_correcto=2;
    white_seccion(70,190,70,50);
  }
  else if((x>=141)&&(x<=210))
```

```
        {
          pulso_correcto=3;
          white_seccion(140,190,70,50);
        }
        else if((x>=211)&&(x<=320))
        {
          pulso_correcto=4;
          white_seccion(210,190,320,50);
        }
      }
    }
  }
  if(pulso_correcto==1)screen_two();
  else if(pulso_correcto==2)screen_three();
  else if(pulso_correcto==3)screen_four();
  else screen_five();
}

void loop()
{

   screen_one();

}
```

ANEXO B

```
#include <Wire.h>

int Electro=8;
int direccion=0xF0;
int buzzerpin=2;
void setup()
{
  Wire.begin(direccion);
  Wire.onReceive(receiveEvent);
  Serial.begin(9600);
  pinMode(Electro,OUTPUT);
  pinMode(buzzerpin, OUTPUT);
  pinMode(13, OUTPUT); // We will use this pin as a read-indicator

}

float prueba=0.0;
float setpoint=0.0;
float co2Value=0;
float numeros=0.0;
int mayoruno=0;
int menoruno=0;
int menormedio=0;
unsigned char buzzer=1,alarma=0;
unsigned char estado=0;



void loop()
```

```
{
   if ((buzzer==1)&&(alarma==1))digitalWrite(buzzerpin, HIGH);
   else digitalWrite(buzzerpin, LOW);

   if(co2Value >= 0)
   {
    prueba=setpoint-co2Value;
    co2Value=-1;
    if (0.5<prueba)
    {
     if (3<prueba)
     {
      digitalWrite(Electro,HIGH);
      delay (20000);
      digitalWrite(Electro,LOW);
     }
     else if (1<prueba<=3)
     {
      digitalWrite(Electro,HIGH);
      delay (10000);
      digitalWrite(Electro,LOW);
     }
     else if (0.5<prueba<=1)
     {
      digitalWrite(Electro,HIGH);
      delay (2000);
      digitalWrite(Electro,LOW);
     }
     else
     {
```

```
          digitalWrite(Electro,HIGH);
          delay (800);
          digitalWrite(Electro,LOW);
        }
      }
    }

}

void receiveEvent(int howMany)
{
  int num=0;
  while( Wire.available()) //Leemos hasta que no haya datos. Teoricamente son 2.
  {
    estado= Wire.read();
    num = Wire.read()<<8; //Leemos los dos bytes
    num |= Wire.read();
  }
  numeros=(float)num/100.0;
   switch (estado){

     case 1:
       co2Value=numeros;
       break;

     case 2:
        setpoint=numeros;
        break;

     case 3:
```

```
      buzzer=1;
      break;

    case 4:
      alarma=1;
      break;

    case 5:
      alarma=0;
      break;

    case 6:
      buzzer=0;
      break;

  }
}
```