ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"CONTROL DE TEMPERATURA DEL SISTEMA AUXILIAR DE UN PROCESO SOLAR TÉRMICO PARA LA PRODUCCIÓN DE AGUA CALIENTE SANITARIA."

INFORME DE PROYECTO INTEGRADOR

Previo a la obtención del Título de:

INGENIERO EN ELECTRICIDAD, ELECTRÓNICA Y AUTOMATIZACIÓN INDUSTRIAL

DAN BRUCE CABRERA GÓMEZ

JOSÉ ANTONIO SANTANA CEDEÑO

GUAYAQUIL – ECUADOR

AÑO: 2018

AGRADECIMIENTOS

Agradezco a mi familia por ser el cimiento principal de mi vida, por su constancia en momentos difíciles, por depositar su confianza en mí y alentarme cuando todo estaba cuesta arriba, por ser fuente de inspiración y esfuerzo para tener visión hacia un futuro mejor. Gracias por demostrarme con pequeños detalles que jamás estuve solo y que el camino siempre es mejor con ellos a mi lado. Gracias por enseñarme a pensar en grande y demostrarme que nunca es suficiente.

A Margarita García, Ramón Macías, Lissette Macías y Ronny Macías, mi segunda familia. Gracias por ser ejemplo de dedicación y solidaridad. Los llevo a todos lados conmigo.

A aquellos amigos que entre bancas fuimos creando lazos de fraternidad y que muchas veces hasta con un led o un par de resistencias podían salvar el día. Gracias por la ayuda, por compartir conocimientos, por las bromas y las risas. De igual manera, a todos quienes fueron aportando con su ayuda a lo largo de esta carrera universitaria.

A mi compañero de proyecto Dan Bruce Cabrera Gómez, por el empeño, persistencia y firmeza para lograr el objetivo final. Por último, y no menos importante, a la tutora de este proyecto de titulación, MSc. Carolina Godoy, por guiarnos en el proceso con entusiasmo y compromiso, siempre dispuesta a colaborar.

A todos ustedes. ¡Gracias totales!

José Antonio Santana Cedeño

Agradezco infinitamente a Jehová Dios durante toda mi vida, por guiarme, mostrarme su amor y darme las fuerzas para hacer frente a todas las circunstancias de la vida.

A mi Padre Angel Efrén Cabrera Barrera por su ejemplo, su cariño, su amor, y por sobretodo su ayuda a lo largo de mi vida y más cuando me animaba a seguir dando lo mejor de mi académicamente a ser mejor y no conformarme con poco en cualquier cosa que tenga que realizar, sin duda él ha sido uno de los pilares más grandes para mí.

A mi madre Rosa Elba Gómez Lindao por su fiel amor, aliento y fuerzas que suministraba a lo largo de mis estudios universitarios. Por transmitirme su perseverancia en mis proyectos y metas por ayudarme a realizar correctamente las cosas y por transmitirme que el respeto por los demás es primordial.

Dan Bruce Cabrera Gómez

DEDICATORIA

El presente proyecto lo dedico a mi madre, Orlanda Cedeño Cevallos, por ser la insignia de fortaleza en mi familia, por siempre estar a mi lado para recordarme que puedo conseguir todo aquello que me proponga en la vida. A mi padre, Adolfo Santana Bermúdez, por ser símbolo de responsabilidad y esmero durante toda su vida para ser el sostén del hogar. A mi hermano, Christian Santana Cedeño, por su compañía y apoyo incondicional a lo largo de estos años. A mi hermana, Krystel Santana Cedeño, por ser ejemplo de tenacidad para lograr los objetivos sin importar qué tan empinado sea el camino. A mi segunda familia, por su apoyo en los momentos adversos, por su afecto y consideración, y por jamás ausentarse en cada paso que he dado en mi vida. Esta alegría es fruto y trabajo de todos ustedes.

Quiero que sepan que son los mejores y que este triunfo es más suyo que mío, sin su ayuda todo esto no hubiese sido posible. Su aporte con cada consejo y los valores inculcados fueron esenciales para mi formación personal y académica. Fue un largo camino con muchas pruebas, derrotas y victorias, pero llegamos a la meta. Les prometo que el esfuerzo no fue en vano y que días mejores están por venir.

Hoy puedo decir que no fue fácil, pero al fin lo logramos. De todo corazón, mil gracias por acompañarme en este viaje. Esto recién empieza.

José Antonio Santana Cedeño

El presente proyecto lo dedico a Dios por haberme permitido la oportunidad de poder culminar una etapa muy importante en mi vida, de demostrarme que siempre estuvo apoyándome a mí y toda mi familia en los momentos donde parecía que todo era imposible.

A mis queridos y amados padres, siendo ellos la mayor de mis fuerzas, quienes me dieron todo su apoyo en todo momento de mi vida, por inculcarme valores éticos y morales para ser una persona correcta y respetuosa, por ayudarme a alcanzar este gran logro, son mi mayor ejemplo de perseverancia y demostrarme que con cada esfuerzo hay un gran logro, gracias infinitamente gracias a ellos por no dejar de creer en mí.

A todos los amigos que forme a lo largo de mi vida universitaria y que aportaron en mi vida con conocimientos y alegrías y momentos inolvidables. En especial a mi compañero de proyecto José Antonio Santana Cedeño por su gran apoyo y ayuda a lo largo de este proyecto por demostrarme que con un gran esfuerzo y perseverancia se logra el objetivo deseado, aprecio muchísimo haber contado con su compañía y colaboración, éxitos en todos los planes para tu vida estimado amigo.

A nuestros profesores encargados en nuestro proyecto, nuestra tutora MSc. Janeth Carolina Godoy Ortega y nuestro colaborador PhD. Wilton Edixon Gálvez Agila por su apoyo, guía y confianza depositado en nosotros a lo largo de este proyecto, por aportar parte de su experiencia en mi formación profesional.

Dan Bruce Cabrera Gómez

TRIBUNAL DE EVALUACIÓN

MSc. Carolina Godoy	**PhD. Wilton Agila**
PROFESOR DE MATERIA INTEGRADORA	TUTOR ACADEMICO

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, me(nos) corresponde exclusivamente; y doy(damos) mi(nuestro) consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

.....

Dan Bruce Cabrera Gómez

.....

José Antonio Santana Cedeño

RESUMEN

Las energías renovables en los últimos años han tenido grandes avances a nivel mundial basados en el diseño y desarrollo de equipos capaces de suplantar el uso de sistemas habituales que afectan al medio ambiente, o que consumen gran cantidad de electricidad y recursos. Tal es el caso de los sistemas solares térmicos aplicados para la generación de energía eléctrica o producción de agua caliente sanitaria mediante el principio de transferencia de calor haciendo uso de la radiación solar como su principal fuente de energía.

El presente proyecto consiste en desarrollar un control de temperatura de un sistema de apoyo con su respectivo tablero de control, incluyendo un prototipo de acumulador auxiliar que estará conectado en serie al sistema principal antes mencionado para aumentar la eficiencia y rendimiento de la producción. Este proceso agrupa el uso de la plataforma de desarrollo de Arduino, una etapa de circuitos electrónicos para la activación del actuador y un colector soltar térmico de placa plana por termosifón facilitado por la Facultad de Ingeniería Mecánica y Ciencias de la Producción (FIMCP) de la Escuela Superior Politécnica del Litoral (ESPOL).

Para el desarrollo, el presente proyecto de titulación fue realizado en cuatro capítulos. El capítulo 1 consiste en el planteamiento del problema junto con los objetivos a abarcar durante el proceso, además de una breve descripción de la solución. En el capítulo 2 se seleccionan las herramientas, equipos y recursos a usar, para después elaborar un esquemático acerca de la interacción de todos los componentes del sistema. El capítulo 3 se enfoca en la metodología efectuada para obtener un proceso autónomo cumpliendo con las validaciones respectivas dentro de la programación y las medidas de seguridad eléctricas pertinentes. Por último, en el capítulo 4 se implementan los subsistemas del proceso para proceder a validar la operación y presentar un prototipo del sistema auxiliar con el tablero de control incorporado; también se realiza un análisis entre los tipos de control empleados con la finalidad de elegir el más apropiado y eficiente para la producción.

ÍNDICE GENERAL

ABREVIATURAS

SST	Sistema Solar Térmico
HMI	Human Machine Interface
TFT	Thin-Film-Transistor
SSR	Solid State Relay
PCB	Printed Circuit Board
PI	Proportional Integrat

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. DELIMITACIÓN DEL PROBLEMA

1.1 Planteamiento del problema.

En la Región Interandina del Ecuador la temperatura es variante de forma apreciable y en algunas ocasiones impredecible, pudiendo alcanzar valores menores a 10 °C según la zona [1].

Si bien es cierto, los climas fríos suelen ser reconfortantes, pero la situación se torna desagradable cuando la temperatura alcanza valores por debajo de lo habitual. Por otro lado, aunque se considere el comportamiento y la condición extrema de esta variable, es probable que no resulte perjudicial tomar una ducha por la noche o por la mañana, o tal vez en ambos casos, pero existe la posibilidad de que se manifiesten condiciones negativas en la salud y comportamiento del cuerpo con el paso del tiempo.

Como solución a este percance, un porcentaje de la población prefiere los sistemas de calefacción basados en suministro eléctrico a pesar de que se produce un aumento en el consumo de energía, contribuyendo al incremento de las emisiones de CO_2 asociadas a la generación de energía eléctrica. Estas emisiones son producidas en unidades de generación basadas en la quema de combustibles fósiles y, junto a otros gases de efecto invernadero, son las principales causas de la intensificación de este fenómeno natural, dando paso a la contaminación del medio ambiente y calentamiento global. Otra solución técnica al problema de calentamiento de agua son los calentadores a gas, sin embargo, estos necesitan una constante revisión en las instalaciones, y además son potencialmente peligrosos debido a la combustión.

Con el propósito de eliminar o disminuir las desventajas relacionadas con los sistemas anteriores, los colectores solares surgen como una posible alternativa, siendo los Sistemas por termosifón los más comunes en las viviendas. A pesar de que el sistema ofrece la reducción de las emisiones de CO_2 y no se trata de un sistema basado en combustión, el resultado no cumple con todas las expectativas dado que no dispone de un sistema que controle el flujo de agua

caliente con o sin la dependencia de la radiación solar. Dicho brevemente, el sistema es eficiente cuando la radiación solar es capaz de calentar la superficie del colector para aumentar su temperatura y de esta forma iniciar el proceso de calefacción.

1.2 Objetivos.

1.2.1 Objetivo General

Diseñar un sistema de control de temperatura basado en la plataforma de Arduino para un prototipo de Sistema Solar Térmico.

1.2.2 Objetivos Específicos

- Programar una interfaz gráfica con la finalidad de adquirir, mostrar, parametrizar y controlar datos.
- Desarrollar una aplicación que permita un control móvil, sencillo y eficaz de un SST relacionado con las variables temperatura y hora parametrizadas por el usuario.
- Construir un sistema para la producción de agua caliente sanitaria acoplando un colector solar y un sistema auxiliar eléctrico con la intención de demostrar su eficiencia y beneficio con respecto al uso de los sistemas convencionales de calefacción, además de mejorar el sistema solar térmico común.

1.3 Justificación.

Ante la temática expuesta previamente se propone como solución el diseño de un sistema de control para un prototipo de sistema de calefacción mediante el cual se estima conseguir un ahorro energético y obtener un proceso automático para abastecer de agua caliente sanitaria.

El objetivo de aplicar el sistema en mención es suministrar agua caliente durante las 24 horas del día haciendo uso de recursos naturales y no solo de la energía otorgada por el sistema eléctrico. El colector solar es de fácil montaje y es ubicado en el tejado de la vivienda, funcionando independientemente de una intervención manual. Por otro lado, el sistema auxiliar de calefacción será colocado en serie al Sistema Solar Térmico y su operación dependerá de la temperatura que

proporcione el colector al agua almacenada. De esta manera se garantiza una reducción en el consumo eléctrico, así como la disminución de emisiones de CO_2 debido a la generación de energía eléctrica dado que al intervenir únicamente el colector solar se dejan de emitir contaminantes a la atmósfera.

El sistema de control es diseñado con la finalidad de condicionar el uso excesivo del sistema auxiliar. Así, por ejemplo, la operación del sistema eléctrico basado en una resistencia calefactora sería innecesaria en horas que la radiación del sol hace predominar el captador solar, lo que implica un menor consumo de energía eléctrica. De este modo se busca que el sistema de calefacción sea integral y pueda ser empleado en cualquier instante.

1.4 Alcance.

Se requiere un sistema que por medio de colectores solares se use la energía captada para calentar agua. El proceso de control toma el valor de temperatura necesaria para la generación de agua caliente sanitaria en el hogar. Es necesario saber en qué instantes la temperatura es idónea para tratar el agua o, si las condiciones no son las apropiadas, realizarlo mediante un sistema auxiliar de apoyo considerando el sistema eléctrico como su fuente principal de energía.

El proyecto será dividido en dos fases:

Fase 1: Estudio del sistema

- Analizar el área de trabajo.
- Listar o reunir todas las variables que involucra el proceso.
- Conocer las características de los equipos o el sistema en su conjunto.
- Detallar las funciones del sistema, así como los subsistemas que lo componen junto a sus especificaciones.
- Parametrización de las condiciones óptimas de operación.
- Realizar las conexiones correspondientes de los módulos y sensores con la tarjeta electrónica.

Fase 2: Desarrollo de aplicación

- Desarrollar una aplicación móvil para la lectura de los datos obtenidos mediante sensores y controlar la temperatura del proceso. Estos datos deben estar en un rango de acuerdo con las características propias de su uso. Si las condiciones térmicas no son las adecuadas para el funcionamiento óptimo del colector solar, se habilitará el sistema auxiliar de tal forma que el proceso siga su funcionamiento cuando se requiera.

CAPÍTULO 2

2. ESTADO DEL ARTE

En el presente capítulo se abordarán conceptos de energías que evitan o disminuyen el uso de recursos fósiles mediante la aplicación de un captador solar, así como la implementación de tecnología para el monitoreo del sistema y el uso de módulos destinados a la comunicación inalámbrica para el envío y recepción de señales y variables del proceso.

2.1 Antecedentes

Se conoce que la radiación solar es una fuente de energía limpia y existen varias formas de utilizarla. Por esta razón, ciertos fabricantes se han dado la tarea de crear sistemas que se encarguen de aprovechar al máximo la energía solar y convertirla en energía térmica. [2]

Abordando el tema de energía solar térmica destinada al área residencial, los captadores de baja temperatura (35-150 °C) son idóneos para proveer a las instalaciones de agua caliente sanitaria, además de ser aplicados en procesos industriales. Este tipo de captadores se clasifican en: captador solar plano y panel de tubos de vacío. Los captadores solares planos son los más utilizados en sistemas domésticos gracias a su fácil adaptación y a la buena relación coste/efectividad que ofrecen. [3]

La Facultad de Mecánica y Ciencias de Producción de la Escuela Superior Politécnica del Litoral dispone de un captador solar plano situado a las afueras del Laboratorio de Metalurgia. El colector en mención es un ejemplar diseñado artesanalmente por un estudiante de la carrera de Ingeniería Mecánica tomando en consideración cálculos específicos, relación entre el colector y el tanque con respecto al dimensionamiento, normas de energías renovables, datos de radiación solar, ubicación y otros parámetros significativos con el propósito de crear un prototipo de buena eficiencia lo más aproximado a los comerciales.

Actualmente el colector solar es usado para pruebas y estudios; además, se le da mantenimiento periódico para cuidar la calidad de sus componentes puesto que, al estar expuesto a la rugosidad y demás factores ambientales, se puede

producir corrosión y otros efectos negativos que afectarían al rendimiento del colector solar.

Para un mejor uso se necesita implementar un sistema auxiliar y automatizarlo para potenciar la funcionalidad y aumentar la vida útil de los elementos que lo van a componer como es el caso de actuadores y sensores. A través de este proceso se pondrán en práctica conocimientos de programación, energías renovables, control de procesos industriales y automatización industrial.

2.2 Marco Teórico

La energía solar es una fuente renovable que puede aprovecharse en diversos campos como la generación de electricidad, industria agrícola, aprovechamiento térmico, entre otras. Sin embargo, el mayor desarrollo que ha tenido son los sistemas de generación eléctrica mediante la implementación de dispositivos de captación junto a equipos electrónicos que permiten un control preciso de la energía entregada a la red. [4]

No obstante, el aprovechamiento térmico también ha tenido un crecimiento notable, diseñando, previo a cálculos y bajo normas estandarizadas, colectores térmicos que permiten captar la radiación solar de manera eficiente para procesos de calefacción y la producción de vapor. Existen tres tipos de captadores: de baja temperatura, de media temperatura y de alta temperatura. Dentro de estas clasificaciones existen diversidades de colectores solares como: captadores solares de cilindro parabólico, captadores solares de tubos de vacío y captadores solares planos.

Los colectores predominantes en instalaciones domésticas son los colectores solares planos y pertenecen al grupo de captadores de baja temperatura como fue mencionado en el apartado anterior. Estos colectores usan el efecto invernadero y están construidos por una fina capa de vidrio con un aislamiento térmico. Por el contrario, los colectores de tubos de vacío son más eficientes ya que se evidencia un aumento en el poder de captación solar debido a su construcción con tubos cilíndricos, formados por un absorbedor selectivo y revestidos por una superficie captadora de vidrio. [5]

A continuación, se describirá cada elemento del sistema y su importancia:

2.2.1 Descripción del sistema

El Sistema en estudio se trata de un proceso de calefacción conformado por un Sistema Solar Térmico y un Sistema Auxiliar que permitirá el calentamiento de agua en condiciones climáticas adversas. Se trata de un proceso inalámbrico, pues la parte de control y monitoreo estará ubicada a cierta distancia del sistema de calefacción. El control será realizado mediante la placa Arduino Mega y el monitoreo del sistema será visualizado por medio de una pantalla TFT de Nextion. Además, cuenta con una aplicación móvil que permite parametrizar el sistema de forma remota.

El sistema de calefacción está conformado por un colector solar por termosifón, el mismo que se encarga del calentamiento aprovechando la radiación solar. El agua depositada en el tanque del sistema solar térmico alcanza distintos valores de temperatura dependientes de la radiación que incide sobre el colector solar; si dicha temperatura no es suficiente para satisfacer los requisitos del consumidor, el sistema auxiliar, controlado por la tarjeta de desarrollo Arduino Mega, entra en operación hasta alcanzar el valor de consigna ajustado. Por otra parte, la aplicación remota será capaz de monitorear y parametrizar la temperatura de consigna, además de crear una histórico de la temperatura del agua en el acumulador auxiliar.

A continuación, en la Figura 2.1 se muestra la secuencia de interacción entre los componentes del sistema.

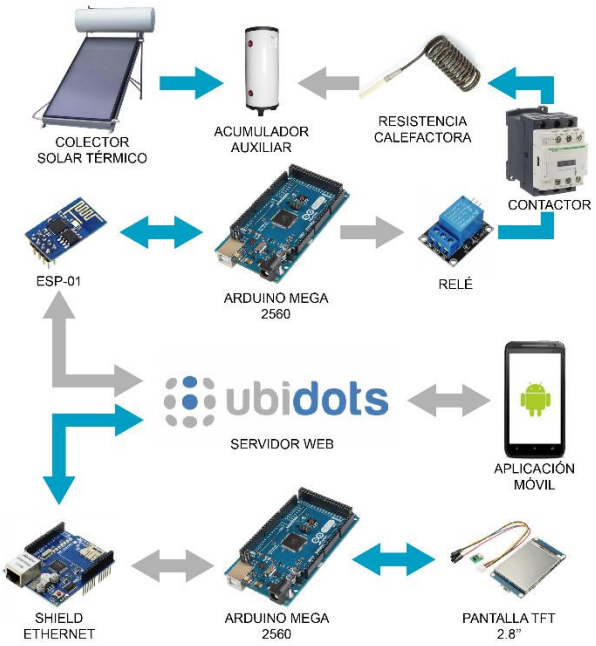


Figura 2.1: Sistema de calefacción

2.2.2 Sistema Solar Térmico

Permite aprovechar la radiación solar efectuando una conversión de energía con el fin proveer de agua caliente a una vivienda. Existen dos tipos de sistemas: por circulación forzada y por termosifón, también llamados sistemas de circulación natural o sistemas pasivos.

El efecto termosifón es el flujo del agua a causa de las diferencias de temperatura entre el agua caliente del colector y el agua fría del depósito. La radiación solar calienta el agua dentro del captador, disminuyendo su densidad y asciende hasta la parte superior del colector solar en tanto que el agua con menor temperatura se ubica en la parte inferior [6]. En la Figura 2.2 se pude apreciar un Sistema Solar Térmico por termosifón.

Figura 2.2: Sistema Solar Térmico por termosifón

A continuación, se listan los componentes más importantes de un SST: [7]

Colector Solar Térmico

Este equipo es el encargado de absorber, mediante materiales metálicos con revestimiento negro, la radiación solar y convertirla en energía térmica para posteriormente transferirla a un fluido de trabajo que circula por su interior. Se pueden clasificar en dos tipos principales: Colectores Solares Planos y Colectores Solares de Tubos al Vacío.

Intercambiador de calor

Existen de diversos tipos, pero para colectores por termosifón el más habitual es el de tipo serpentín. Son los encargados de transmitir el calor hacia el agua almacenada en el tanque gracias a un fluido caloportador que circula internamente; por esta razón, es importante que sea resistente a altas temperaturas.

Acumulador

En su defecto, se ubica a una distancia por encima del Colector Solar con la precaución de no producir efectos de sombra al captador ya que la eficiencia se vería reducida y los resultados no serían favorables. Está

revestido con aislante térmico para evitar las pérdidas de calor y garantizar el equilibrio de temperatura dentro del tanque.

2.2.3 Sistema Auxiliar

Es un sistema de apoyo controlado en caso de que la temperatura del agua almacenada en el acumulador del SST no sea la idónea para el consumidor. Existen variedades, entre los más reconocidos y usados en las viviendas son los sistemas a gas y sistemas alimentados por la red eléctrica.

En este caso, el sistema de apoyo estará compuesto por una resistencia eléctrica además de un módulo o circuito de activación que aísla la etapa de control y la etapa de potencia para proteger la tarjeta de desarrollo y demás dispositivos presentes.

2.2.4 Termistor

El sistema contará con dos sensores de temperatura DS18B20. Este tipo de sensor es encapsulado en una sonda metálica por ser frágiles a vibraciones y se caracteriza por su variabilidad de resistencia eléctrica proporcional al incremento de temperatura [8]. Para más información, véase el Tabla 10 del Anexo 1 correspondiente a la hoja de datos característicos del equipo.

2.2.5 Modos de control

Para controlar un sistema es usual utilizar el principio de lazo cerrado o feedback. Este principio indica que, cuando la variable del proceso es menor al valor de consigna (error mayor a cero), la variable manipulada incrementará con el objetivo de disminuir el error hasta hacerlo nulo. En general, para sistemas térmicos hay la posibilidad de implementar dos tipos de control que serán detallados a continuación: [9]

Control On/Off

Es el más simple de todos por ser un control de dos estados. Dicho de otra manera, enciende o apaga completamente el actuador dependiendo

si la variable manipulada es menor o mayor al valor de consigna. Tiene un funcionamiento exitoso cuando la variable del proceso se mantiene cercana al punto de operación, pero en muchos casos la implementación de este control no es eficiente y da como resultado un sistema inestable con oscilaciones no deseadas.

Control Proporcional Integral

Surge como solución al limitado desempeño del Control Proporcional y su función principal es eliminar totalmente el error de estado estacionario después de un intervalo de tiempo asegurando que la variable del proceso es igual al punto de referencia. En conclusión, es el resultado de la suma del término proporcional, que actúa de manera instantánea en presencia de un error, y el término integral.

En un sistema térmico, este tipo de control es capaz de evitar que la temperatura exceda el punto de referencia en presencia del más mínimo cambio de temperatura en el acumulador aislado actuando de manera sofisticada y segura.

2.2.6 Arduino MEGA 2560

Se trata de una PCB (Printed Circuit Board) reprogramable conformada por un puerto USB, un microcontrolador ATMega2560, y puertos donde se pueden conectar directamente placas de expansión y demás componentes permitiendo aprovechar todas las características que ofrece la placa de desarrollo. Los proyectos en Arduino se clasifican de dos formas: autónomos y no autónomos. Los primeros se caracterizan por ser independientes al computador una vez que el microcontrolador haya sido programado, en tanto que los no autónomos la tarjeta de desarrollo debe estar necesariamente conectad a la computadora durante la ejecución para la adquisición y transferencia de datos. [10]

En la Tabla 1 se listan las características más importantes de Arduino Mega: [11]

ESPECIFICACIONES TÉCNICAS	
Microcontrolador	ATmega2560
Voltaje de operación	5 V
Voltaje de entrada (recomendado)	7-12 V
Voltaje de entrada (límite)	6-20 V
Puertos E/S Digitales	54 (15 proporcionan salidas PWM)
Puertos Entradas Analógicas	16 (cada uno con 10 bits de resolución)
Corriente DC por puerto E/S	20 mA
Corriente DC para puerto 3.3 V	50 mA
Memoria Flash	256 KB (8 KB usados para bootloader)
SRAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz
LED	13
Tamaño	101.52 mm
Ancho	53.3 mm
Peso	237 g

Tabla 1: Características de Arduino Mega

La mayor ventaja de Arduino es su fácil integración para proyectos multidisciplinarios, facilitando la programación y fomentando el aprendizaje y aplicación de la electrónica. Toda la plataforma de Arduino cuenta con licencia de código abierto, lo que conlleva a que todos los usuarios puedan realizar mejoras facilitando la depuración de proyectos.

2.2.7 Módulos de comunicación

En cuanto a la comunicación inalámbrica, es necesario hacer uso de módulos que permitan la recepción y emisión de señales que active o desactive el actuador a implementar en el proceso. A continuación, se detallan los módulos que serán usados en el sistema:

Shield Ethernet

Es un módulo compatible con el IDE de Arduino basado en el chip Ethernet Wiznet W5100 TCP/IP [12]. Consta de un puerto RJ45 que permite el acceso a internet, puertos SPI (Serial Peripheral Interface) para la conexión a Arduino. Opera con un voltaje de 5v otorgado por la placa de desarrollo Arduino Mega. La placa Shield Ethernet implementa protocolos TCP, lo que hace posible tener una capa física y enlazar datos por el estándar IEEE802.3. Véase la Figura 2.3.

Figura 2.3: Módulo Shield Ethernet

WIFI WIRELESS ESP-01

Es un módulo diseñado e implementado por Espressif [13] y muestra un gran contraste al módulo anterior. Su característica principal es conectarse a una red Wifi local o remota; sin embargo, conectarse a una red remota es complicado ya que trabaja con protocolos TCP/IP, HTTP incluyendo IPv4. Su voltaje de alimentación es de 3.3v y se programa con comandos AT para su configuración. Cuenta con su propio microcontrolador al que se puede cargar un programa y, haciendo uso de

las librerías correctas, implementar el dispositivo sin necesidad de estar conectado a una placa de desarrollo como Arduino. Véase la Figura 2.4.

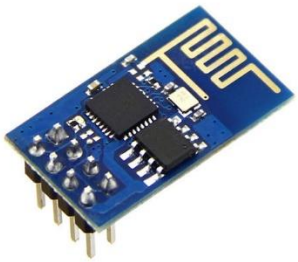


Figura 2.4: Módulo ESP-01

2.2.8 Pantalla TFT de 2,8'' HMI Nextion

Es un dispositivo HMI (Human Machine Interface) que ofrece visualización y control entre el operador y un proceso o máquina. Esta tecnología se presenta como una mejor alternativa a la pantalla LCD estándar, es utilizada por consumidores del campo electrónico y, en muchos casos, aplicada para IoT (Internet of Things). [14]

Las pantallas Nextion se clasifican en tres tipos: básicas, mejoradas e inteligentes. Para propósitos de este proyecto, se hará uso de una pantalla básica de 2,8'' modelo NX3224T028_011R, véase la Figura 2.5, ya que es un dispositivo táctil adecuado para la interfaz gráfica del sistema. Es de alto rendimiento y funciona a 5v con una corriente de 65mA. Entre sus elementos importantes se encuentran: memoria Flash, sensor táctil, GPU, ranura para tarjeta SD e interfaz UART. La ranura para la tarjeta SD es de gran utilidad para cargar el programa a la memoria Flash en caso de que el computador no detecte la pantalla por medio del convertidor USB a TTL.

Figura 2.5: Pantalla TFT 2.8”

2.2.9 Softwares de desarrollo

En el mercado existe todo tipo de tarjetas electrónicas que permiten realizar varias funciones, desde las más básicas con sus versiones educacionales, hasta aplicaciones más complejas como controlar un proceso industrial o comercial. Arduino, Raspberry Pi, FPGA y CPLD son claros ejemplares contando con un software de programación como el entorno de programación de Arduino (IDE), Raspbian y Quartus, respectivamente. Actualmente Arduino es una de las plataformas más utilizadas debido a su fácil uso y la integración que ofrece entre hardware y software.

Por otro lado, la pantalla Itead de Nextion tiene un software de desarrollo llamado Nextion Editor que permite programar y simular el comportamiento del dispositivo. Además, brinda la ventaja de realizar un diseño personalizado de acuerdo con las exigencias del usuario.

Arduino Integrated Development Environment

También llamado Arduino Software IDE, fue desarrollado en Java, y está basado en el entorno y lenguaje de desarrollo libre: Processing. Es un entorno de programación formado por un conjunto de múltiples herramientas tales como editor de código, consola de texto y demás, que permiten construir una interfaz, compilar, depurar y cargar el programa del computador a la memoria Flash de la placa usando el puerto USB como canal de comunicación. [15]

El IDE de Arduino dispone de una diversidad de librerías para los periféricos y shields, las mismas que son creadas o modificadas de manera gratuita por los usuarios con la finalidad de brindar soluciones a problemas específicos y mejorar la experiencia de interacción entre el programador y el entorno interactivo de desarrollo.

Nextion Editor

Es un software destinado al diseño y programación de la pantalla TFT, conformado por varios componentes tales como texto, botones, sliders, etc.; estas herramientas son de gran ayuda para la creación de la interfaz gráfica, facilitando el monitoreo de las variables del proceso. [16]

Su interfaz principal está divida en ocho áreas:

- Menú Principal
- Componentes
- Librería de imágenes
- Área de display
- Área de página
- Área de edición de atributos
- Ventana de salida del compilador
- Área de eventos

En el área de eventos se programa la TFT con comandos respectivos de Nextion Editor. Una vez desarrollada la programación de la pantalla, se puede iniciar un proceso de prueba haciendo uso de la herramienta de depuración que permite simular el proyecto en tiempo real de modo que el operador puede examinar el funcionamiento de la pantalla y si esta trabaja según las expectativas.

APP Inventor

Es una plataforma gratuita en línea creada por MIT (Massachusetts Institute of Technology) destinada al desarrollo de aplicaciones de índole profesional, comercial y estudiantil para teléfonos móviles y tabletas.

Maneja una programación mediante bloques, fomentando una mejor interacción entre el programador y la interfaz de desarrollo.

Ubidots

Es una nube destinada a proyectos IoT en industrias. Al disponer de una cuenta, el usuario puede crear un dispositivo con sus respectivas credenciales llamadas Token, Device Name y API ID, donde se almacenan los datos. [17]

2.2.10 Lenguajes de programación

La programación de Arduino es libre y está basada en lenguaje C/C++. Dicho de otra manera, el lenguaje Arduino no es exactamente un lenguaje, más bien se trata de un conjunto de instrucciones que simplifican el desarrollo de aplicaciones, posibilitando una experiencia de programación amigable para todo tipo de usuario. Los paquetes como avr-libc, avr-gcc y binutils son de suma importancia para la programación y compilación de los microcontroladores AVR de Atmel; de igual modo, según la aplicación, las librerías de alta calidad existentes en la web o en la página oficial de Arduino son un plus para obtener un sistema óptimo y confiable. [15]

Aspectos como la familiarización con el entorno y disposición de una serie de funcionalidades que cubren de la mejor manera las necesidades del proyecto son motivos suficientes para considerar el IDE oficial de Arduino como la herramienta principal para la programación a implementar en la placa Arduino Mega.

A su vez, la consola de programación de Nextion será usada para ciertas funcionalidades que resultan más sencillas programarlas bajo esta plataforma, con el beneficio de conseguir una programación menos extensa en el software de Arduino.

Lenguaje Arduino

Dispone de múltiples librerías que posibilitan la creación de sistemas simples y complejos, manteniendo la eficiencia y rapidez que ofrecen los programas en lenguaje C y C++.

A los programas se les llaman Sketch y están constituidos por tres secciones importantes: [18]

- **Declaración de librerías y variables globales**
  En esta sección se definen los puertos que se van a necesitar, se declaran las variables y se llaman a todas las librerías de la programación.
- **void setup()**
  En esta parte del programa se determinan todas las configuraciones iniciales del proceso. Su ejecución es precedente a void loop() y se da una sola vez al energizar la placa por primera vez, o cuando se produce un reset de la misma.
- **void loop()**
  Realiza una ejecución iterativa de las instrucciones que se encuentran dentro de esta sección. Se debe agregar que, las tabulaciones en la programación no son necesarias ni representan un problema en la compilación; sin embargo, se recomienda el uso de estas para llevar un orden con el fin de que la lectura, interpretación y búsqueda de errores en el código sea más sencilla.

Lenguaje Nextion

El lenguaje de Nextion es simplemente un conjunto de instrucciones no compatible con ningún lenguaje de programación perteneciente únicamente a Nextion Editor.

CAPÍTULO 3

3. METODOLOGÍA DE TRABAJO.

En este capítulo se puntualizan las técnicas utilizadas para el desarrollo del proyecto. Como servidor web se hará uso de la plataforma Ubidots. Es oportuno mencionar que se hizo uso de librerías que se irán mencionando y detallando en el transcurso del capítulo.

3.1 Planteamiento del proceso

Antes de empezar con el desarrollo de la programación, es necesario realizar el reconocimiento de la planta para relacionarse con el proceso y los elementos que actúan en el sistema con la finalidad de determinar la carencia de recursos que limitan el producto final. El control del proceso se desarrolla bajo el lenguaje de programación C/C++ propio de Arduino que no son más que conjuntos de instrucciones que simplifican el desarrollo de aplicaciones.

Inicialmente el Sistema Solar Térmico está comprendido por un tanque totalmente aislado de las condiciones climáticas, un colector solar plano y termocuplas encargadas de receptar la temperatura interna del tanque y la propia del colector solar. El sistema fue construido con el objetivo de producir agua caliente sanitaria, proceso que es completamente eficiente en días soleados con una radiación solar directa alcanzando temperaturas mayores a 50 °C según los datos obtenidos [19]. El colector es de tipo termosifón creando un efecto invernadero en el área formada entre la plancha de vidrio y la base de la caja metálica aislada en su defecto; es decir, trabaja bajo el principio de diferencia de presión donde el fluido caloportador absorbe calor mientras circula dentro de los tubos desde la parte inferior a la superior de la malla formada por la interconexión de las tuberías. Parte del calor adquirido es transferido al agua depositada en el reservorio del SST mediante un serpentín instalado internamente en el tanque. Es necesario aclarar que el fluido de calefacción y el agua a consumir son dos sistemas independientes entre sí por motivos relacionados a la salud e higiene.

En definitiva, se trata de un proceso totalmente mecánico que al considerar condiciones climáticas adversas tales como días nublados o una incidencia difusa

sobre la placa de vidrio resulta poco eficiente y confiable debido a las bajas temperaturas que podría alcanzar, sin mencionar las pérdidas de calor que se presentan durante la noche por la ausencia de radiación solar.

Por consiguiente, con el propósito de suprimir las deficiencias presentadas por el SST, se instala en paralelo un sistema de calefacción eléctrico controlado en determinados horarios con la intención de proveer de agua caliente sanitaria a una vivienda durante las horas de mayor consumo. Además, se hace uso de sensores digitales de temperatura por una adquisición de datos más eficaz y facilidad de acople a la tarjeta de desarrollo de Arduino. A su vez, dado que la planta se encontrará en el exterior de la vivienda, la tarjeta de desarrollo de campo se comunicará inalámbricamente a la nube de Ubidots para alojar los datos a gestionar. Los datos pueden ser gestionados por medio de una aplicación móvil o una interfaz gráfica instalada dentro de la vivienda que será lo más cercano a una estación de control destinada a la visualización de temperaturas, configuración de horarios de control y cambios para la temperatura de consigna. En la Figura 3.1 se muestra el sistema auxiliar de calefacción con sus respectivos componentes como la tarjeta de control, contactor y módulo Wifi.

Figura 3.1: Sistema de calefacción

3.2 Configuración de Pantalla HMI

La pantalla HMI tiene su propio software de desarrollo conocido como Nextion Editor, y es usado para la configuración y programación del equipo. Como ya se expuso anteriormente, está constituido por un área de diseño con sus propias herramientas y una consola de programación.

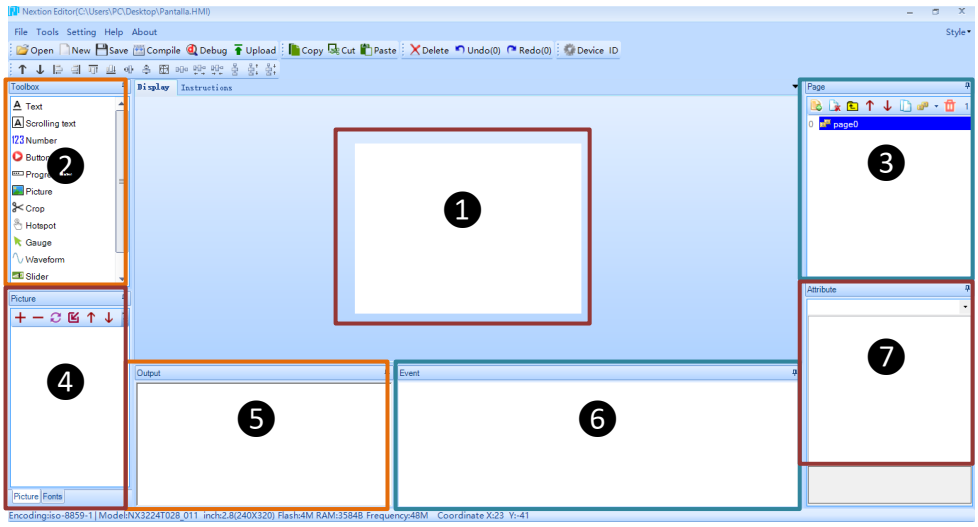


Figura 3.2: Interfaz de Nextion Editor

En la Figura 3.2 se pueden observar claramente las secciones que componen la interfaz de Nextion Editor. La sección 1 es el área de trabajo donde se diseña la parte visual de la pantalla usando las herramientas ofrecidas en la sección 2; las propiedades de los objetos añadidos pueden ser vistas y configuradas en la sección 7. Se debe agregar que una de las propiedades de los objetos está relacionada con la fuente de texto, aspecto que también puede ser configurado por medio de la sección 4 donde se almacenarán los estilos de fuente creados, así como imágenes que se deseen agregar en la pantalla para brindar una presentación más personalizada. A medida que se van añadiendo páginas a la pantalla, la sección 3 se irá actualizando con la finalidad de mostrar la página en la que se está trabajando. La sección 6 es la consola de programación donde se le dará una acción después de cada evento táctil. Por último, en la sección 5 se indicarán los errores existentes, después de cada compilación.

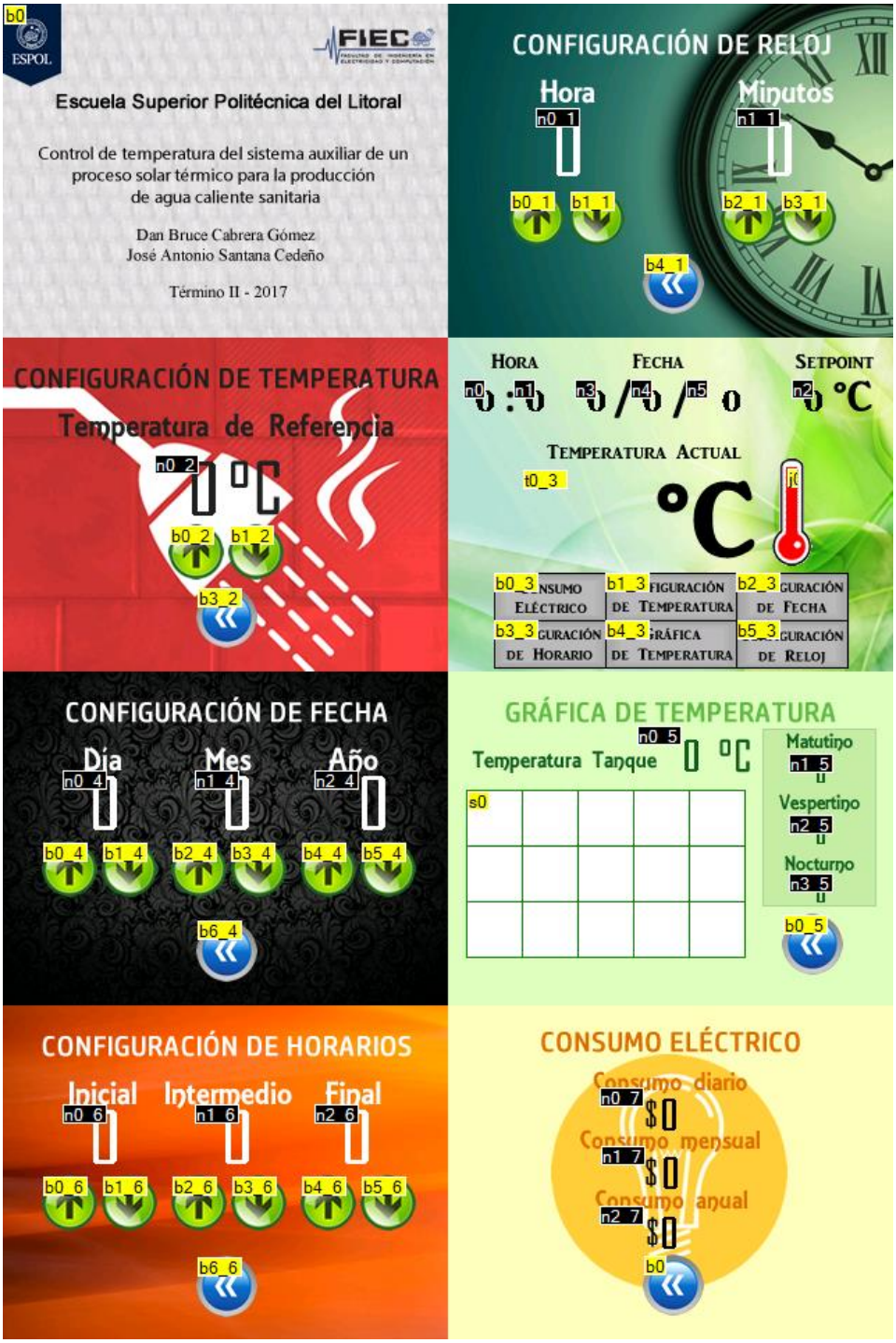


Figura 3.3: Diseño de la interfaz gráfica

En la Figura 3.3 se recopilan todas las páginas que conforman la interfaz gráfica del sistema en estudio. La programación cargada para cada botón es básica

presentando ligeros cambios según el propósito de cada uno. En el caso de los botones de configuración en general, la programación está basada en sentencias if que deberán cumplir las condiciones pertinentes para efectuar cada operación respectiva al evento y de esta manera actualizar los valores en cada variable numérica. Por otra parte, los botones de cambio de página tienen una configuración más sencilla, puesto que solo se necesita una instrucción en específico para ser dirigidos a la página correspondiente. La información de cada objeto se transfiere a la placa de desarrollo por medio de uno de los puertos seriales de Arduino Mega y será leída en el microcontrolador mediante librerías.

La programación en Nextion se ejecuta para eliminar de la programación principal de Arduino los comandos encargados de realizar los mismos eventos que pueden se configurados en la pantalla TFT, ya que esto demandaría la creación de funciones y declaración de múltiples instancias haciendo uso de librerías especiales para la correcta lectura de datos en el puerto serial, y consecuentemente ocupar de forma inadecuada un cierto porcentaje de la memoria del microcontrolador.

3.3 Conexión a Internet y gestión de datos

Como se mencionó en el capítulo 2, para la conexión a internet se usaron el módulo Wifi ESP-01 y la Shield Ethernet de Arduino, dispositivos en los que usualmente son necesarios comandos AT para su respectiva configuración. No obstante, se debe considerar que existen librerías capaces de ejecutar las mismas acciones que puede realizar una programación más extensa basada en dichos comandos, pero de forma mucho más sencilla. Sin embargo, es de mayor ventaja utilizar únicamente la librería para la Shield Ethernet y no para el módulo ESP826 debido a que las librerías destinadas para este módulo presentan problemas actualmente.

En las siguientes líneas se especificarán, de forma precisa, las características que diferencian la Shield Ethernet del módulo ESP-01, así como los pasos realizados para conectar a Internet cada uno de los dispositivos y una forma de intercambiar datos con la web.

3.3.1 Shield Ethernet

Este hardware permite establecer una conexión a Internet y acceder a servidores web; sin embargo, presenta la gran desventaja de no ser un equipo inalámbrico, sino más bien necesitar de una conexión física a un router para la adquisición y envío de datos. Su montaje a la placa electrónica de Arduino es sencillo y sin importar su limitación, la aplicación de este equipo es idónea, por ejemplo, en estaciones de trabajo donde se requiera acceder a internet y establecer una comunicación bidireccional en procesos no autónomos donde la placa de Arduino necesariamente debe encontrarse conectado a un ordenador para la adquisición de información, y su eventual procesamiento y análisis en una base de datos.

Dicho lo cual, para lograr este propósito y entrando en detalle al tema de interés, es necesario hacer uso de la librería Ubidots-Ethernet [20]. Una vez instalada la librería, según las especificaciones del desarrollador, es necesario comprobar que la placa Shield funciona correctamente en conjunto con la placa Arduino Mega. Para ello, la librería nos provee de varios códigos en la sección de 'Ejemplos' y basta con cargar "WebClient" al Arduino Mega y comprobar que el dispositivo trabaja apropiadamente.

Realizada la prueba y establecida la primera conexión, se deben buscar las programaciones correspondientes a la emisión y recepción entre los ejemplos predeterminados. En la Figura 3.4 se puede observar que la librería otorga 6 diferentes ejemplos o posibilidades de programación donde se hace uso de distintas instrucciones que cumplen con las funciones de los comandos GET y POST. Para poder obtener los datos alojados en Ubidots se elige el código UbidotsGetValue, donde hay que definir ciertas cadenas de caracteres para hacer un uso apropiado de la programación.

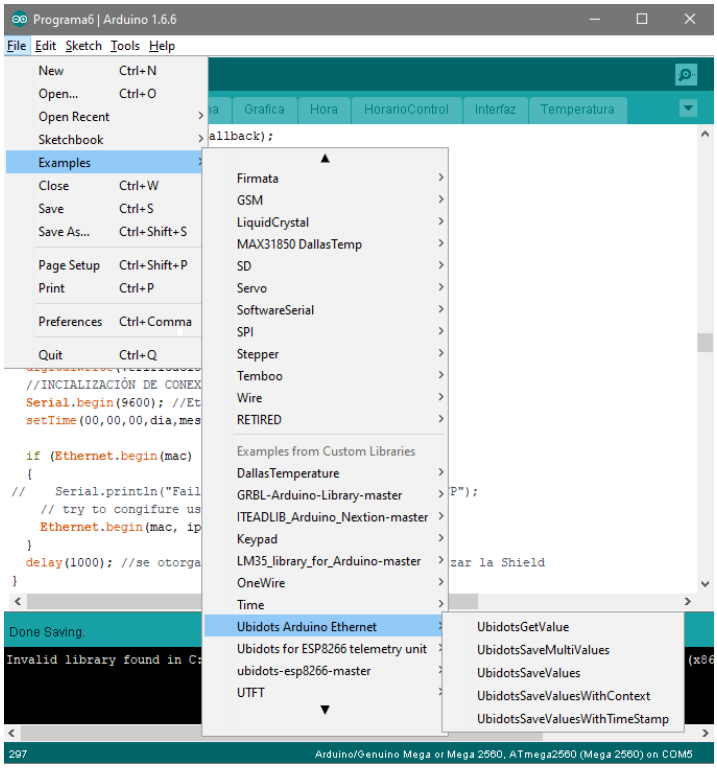


Figura 3.4: Búsqueda de códigos en la IDE de Arduino

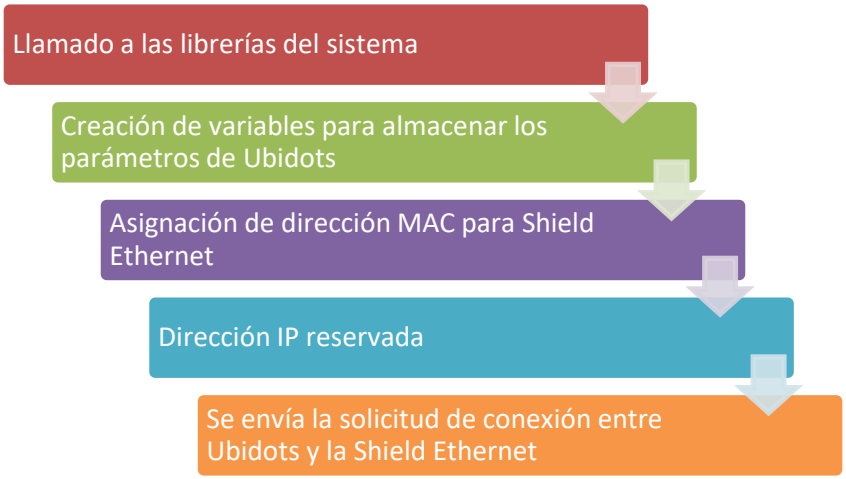


Figura 3.5 Proceso de inicialización de librerías

En la Figura 3.5 se muestra brevemente el proceso de inicialización de librerías y definición de variables que debe realizarse al inicio del Sketch

de Arduino. Como se mencionó en el Capítulo 2, un Sketch se divide en tres partes fundamentales, siendo esta la primera y donde se hace el llamado de las librerías, así como la definición de las variables y cadenas de caracteres necesarios para establecer la conexión a Internet.

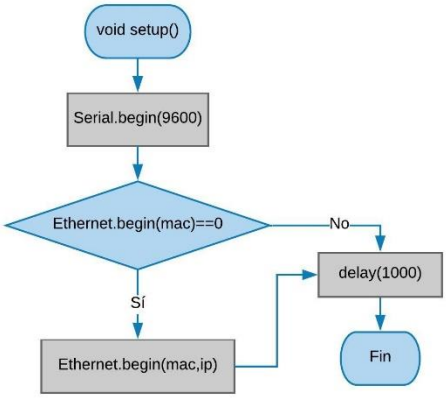


Figura 3.6: Diagrama de flujo para conexión a Internet

En la Figura 3.6 se presenta el algoritmo de conexión a Internet. Dentro de la función void setup() se usan instrucciones pertenecientes a la librería Ethernet.h encargada de darle una dirección MAC y una dirección IP a la Shield Ethernet para su configuración.

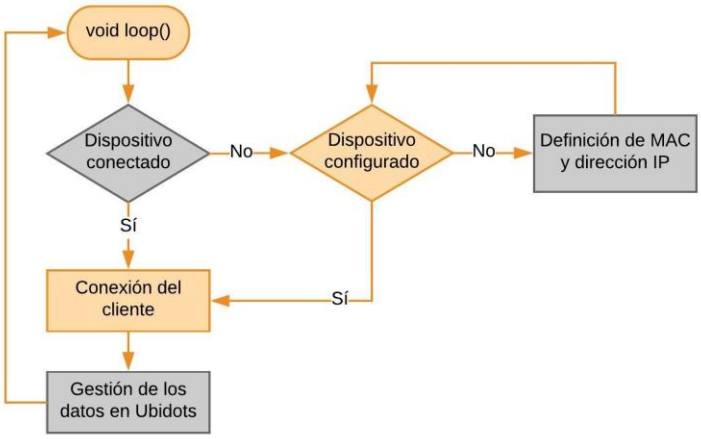


Figura 3.7: Gestión de datos entre Arduino y Ubidots

Para terminar, en la Figura 3.7 se define un condicional en caso de que la conexión a Internet no se haya establecido, iniciándose un ciclo repetitivo hasta que el hardware se configure correctamente y, en consecuencia, se obtenga una conexión exitosa. Seguido de esto, se declara la función client.getValue() que tiene como argumento el nombre del dispositivo y el nombre de la variable que se quiere obtener el dato.

3.3.2 Módulo ESP-01

Por lo que se refiere a la conexión a Internet inalámbricamente, como ya se mencionó, el uso de las librerías existentes no suele ser la solución más eficiente ya que es posible que estas no estén bien desarrolladas y presenten inconsistencias en cuanto a su funcionamiento. Por esta razón se realizó una programación específica haciendo uso de comandos AT que permita tanto la conexión a internet, así como la adquisición y envío de datos a la nube. Para ello es recomendable no alterar los ajustes de fábrica del dispositivo debido a que dicha configuración predeterminada se trata de un Firmware basado en los comandos antes mencionados, cuyo propósito es lograr que el módulo funcione únicamente como un puente de conexión a la red.

En la Tabla 2 se presentan los comandos AT usados en la programación con la finalidad de comprender cada uno de ellos y su importancia. En particular, la conexión a Internet mediante comandos AT se realiza manualmente escribiendo las instrucciones a través del monitor serial de la IDE de Arduino; sin embargo, es necesario que la configuración sea insertada en la programación para que el proceso sea automático en cada reset o energización de la placa de desarrollo.

Es de buena práctica evitar conflictos de comunicación con el módulo Wifi, razón por la cual inicialmente se produce un reset del dispositivo y posterior a ello se envían las instrucciones correspondientes para conectarse a la red wifi de interés; por tanto, el módulo es configurado como estación ya que será usado como cliente para gestionar el intercambio de información entre el controlador y la nube. Para una mejor

comprensión, el algoritmo que se presenta en la Figura 3.8 detalla de manera ordenada los pasos imprescindibles para establecer la conexión inalámbrica.

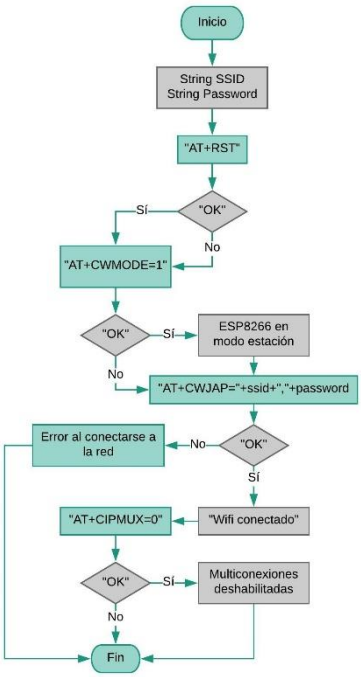


Figura 3.8: Configuración de módulo ESP-01

Para la manipulación de datos, es importante crear tramas de envío que alojen en ellas las identificaciones de la variable a la que se desee emitir el mensaje. Estas variables son creadas en la plataforma Ubidots y están almacenadas en un dispositivo reconocido por el nombre, la ID y sus respectivas credenciales API. A continuación, se muestra la creación de las peticiones para la gestión de datos, y la manera en cómo se involucra en la programación de la tarjeta de control.

Comandos	Parámetros	Descripción
AT+RST	___	Reinicia el módulo
AT+CWMODE	1 = estación	Configuración en modo wifi
	2 = Access point	
	3 = ambas	
AT+CWLAP	___	Busca las redes wifi cercanas
AT+CWJAP	Ssid, pwd	Conexión a la red wifi elegida
AT+CIPMUX	0 = conexión simple	Configura las conexiones múltiples
	1 = múltiples conexiones	
AT+CIPSERVER	0 = cierra el modo servidor	Configuración como servidor
	1 = abre el modo servidor	
AT+CIPSTART	Id = 0-4, addr = dirección IP, port = puerto	Configura la conexión TCP o UDP
AT+CIPCLOSE	___	Cierra la conexión TCP o UDP

Tabla 2: Comandos AT

Peticiones POST y GET

Para realizar la tarea de envío de datos a la nube, se adaptó una programación compartida por un usuario en línea realizando ciertos cambios referentes al formulario con la intención de acoplar el código [21].

En la Figura 3.9 y Figura 3.10 se representan gráficamente, a través de diagramas de flujos, los algoritmos implementados para cada petición. Los procesos son muy parecidos, con diferencias en la estructura de los formularios que se emiten para las solicitudes GET y POST. Los formularios de las peticiones son construidos en base al formato JSON, capaz de gestionar datos de la misma manera que un formulario en formato XML, presentando la ventaja de ser aplicables para cualquier lenguaje de programación además de ser más sencillos de escribir [22].

Para simplificar, el formulario de las peticiones difiere según el tipo y la cantidad de información que se quiera obtener [23]. En el caso de la petición POST, el formulario está compuesto por las IDs de las variables a las cuales se desea escribir una información; no obstante, solo es posible enviar hasta dos datos a la vez. Para ilustrar mejor véase la Figura 3.11.

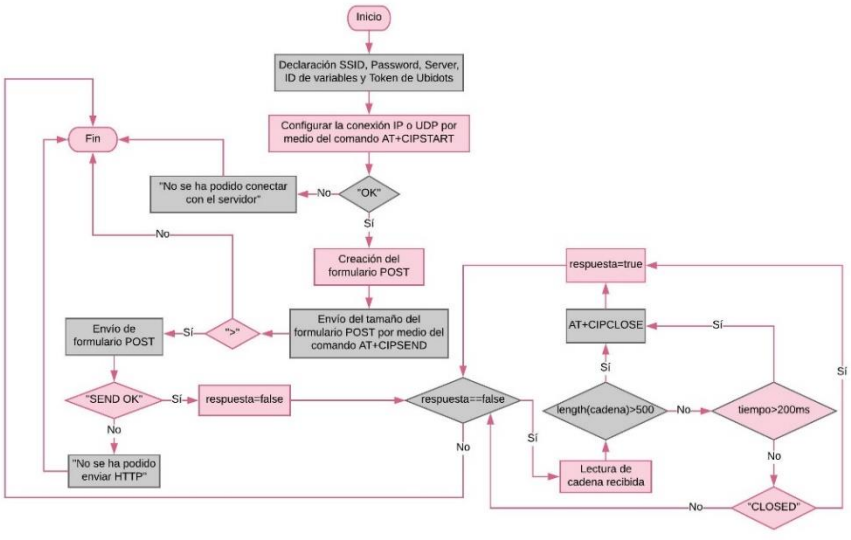


Figura 3.9: Diagrama de flujo para envío de datos

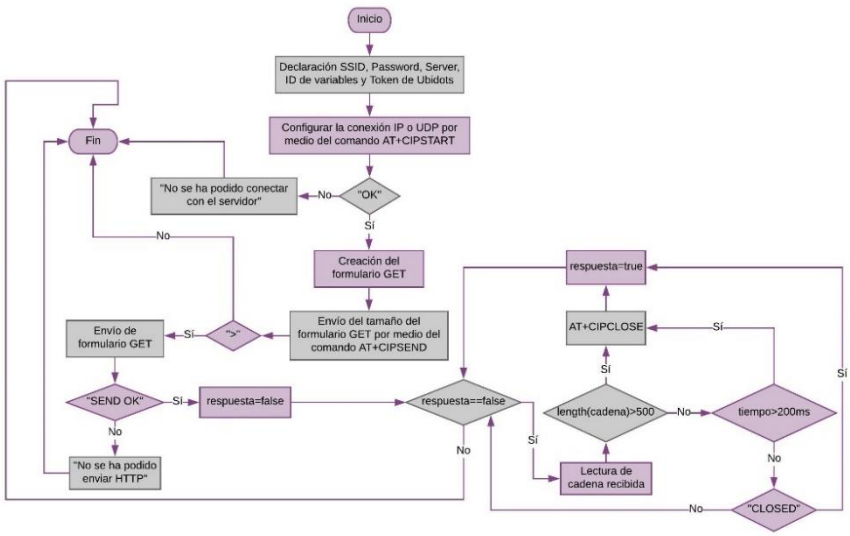


Figura 3.10: Diagrama de flujo para recepción de datos

```
POST /api/v1.6/collections/values/?token={TOKEN} HTTP/1.1
Host: things.ubidots.com
Content-Type: application/json
Content-Length: 112
```

Figura 3.11: Formulario de petición POST

La petición GET únicamente puede entregar la información de un solo dato dependiente del formulario enviado; para este caso en particular, véase la Figura 3.12. En este caso, la petición responde con una extensa trama de caracteres y dentro de ella está almacenada la información solicitada de modo que resulta necesario recorrer la trama para extraer el dato y hacer uso de él; sin embargo, la tarea se vuelve molesta cuando se trata de dos o más tramas de la recepción teniendo en cuenta que se debería introducir una y otra vez las mismas líneas de programación después de cada petición GET. Dicho lo anterior, se crea una función que podrá ser llamada en una sola línea, teniendo como argumento la cadena recibida y retornando el valor de la variable solicitada. Este hecho, además de la ventaja declarada, implícitamente facilita la lectura de la programación para futuras modificaciones de forma rápida y eficaz, dado que con un solo cambio afectaría el comportamiento de todo el código.

```
GET /api/v1.6/devices/{LABEL_DEVICE}/{VARIABLE_LABEL}/lv?token=
{TOKEN} HTTP/1.1
Host: things.ubidots.com
```

Figura 3.12: Formulario de petición GET

3.4 Interfaz gráfica

Por lo que se refiere a la interfaz gráfica del proceso, esta estará compuesta por la placa Arduino Mega 2560, una placa Shield Ethernet y una pantalla TFT de 2.8”; equipos conectados entre sí usando los puertos seriales de la placa de desarrollo con un enlace a Ethernet a través de un par trenzado entre el router y la Shield. Su función principal consiste en la interacción constante con la plataforma Ubidots para el envío, adquisición y actualización de las variables que gobiernan el sistema de control. El comportamiento del equipo en cuestión dependerá de la red a la cual se encuentra conectado para que opere en óptimas condiciones, siendo necesaria una previa conexión para dar inicio al sistema.

El código fue separado en funciones con la intención de que la programación principal sea más ligera visualmente y para evitar que ocurran eventos repetitivos en cuanto a la actualización constante de objetos ajenos a la página actual, pertenecientes a las demás páginas que estructuran la interfaz.

Explorando un poco más la idea de la comunicación entre los dispositivos, en el caso de la pantalla táctil no se trata sólo de conectar a uno de los puertos de Arduino, sino más bien, se necesita de alguna especie de puente tipo software capaz de establecer una comunicación entre ambos dispositivos y que de esta manera puedan interpretar la información que emiten entre sí. Como solución a esta problemática se hizo uso de la librería Nextion.h [24] obteniendo resultados favorables ya que no se presentó una pérdida de conexión en ningún instante. Ahora bien, para que la placa de desarrollo reconozca y actúe sobre un objeto es necesario identificarlo de manera anticipada. Para esto, la librería cuenta con instrucciones específicas que reciben como argumentos el número de página, la ID y el nombre del elemento. La Figura sirve como ejemplo para ilustrar lo planteado.

Por otro lado, la reacción del microcontrolador ante un evento táctil depende del comando nexLoop() que recibe como parámetro una lista de punteros para advertir que la pantalla fue pulsada y así efectuar las tareas configuradas. En esta ocasión, en cuanto a eventos táctiles concierne, Arduino fue programado para detectar únicamente los botones de cambio de páginas y los eventos de configuración de fecha, hora, entre otros, fueron cargados directamente a la memoria Flash de la pantalla HMI.

Declarada las funciones e instancias a ocupar en la programación, se inicializan dentro de void setup() además de fijar los baudios para la transmisión de datos en cada puerto serial. Simultáneamente, se configura el reloj interno del microcontrolador para que opere como un reloj por software encargado de ajustar la fecha y hora del sistema. La desventaja de este reloj es que obliga al operario a restablecer el calendario después de cada reset ya que, por defecto, siempre inicia con una fecha y hora predeterminada después de cada energización.

Finalizada la etapa de ajuste en el código, se procede a realizar la programación principal de la interfaz gráfica que se encuentra sintetizada en el diagrama de flujo visto en la Figura 3.13. El acceso a Ubidots ocurre cada 10 segundos para prevenir un colapso por parte de los componentes que acceden simultáneamente al puerto serial, pudiendo provocar una caída e inhibición del microcontrolador. Para simplificar, la función Control() ejecutará tareas básicas como la actualización de las variables encargadas de inicializar el proceso de calefacción; además, una vez que la temperatura del tanque alcance la temperatura de consigna, realiza el cálculo de consumo energético del sistema auxiliar según el valor de los contadores reajustados correspondiente al tiempo de operación del sistema de apoyo.

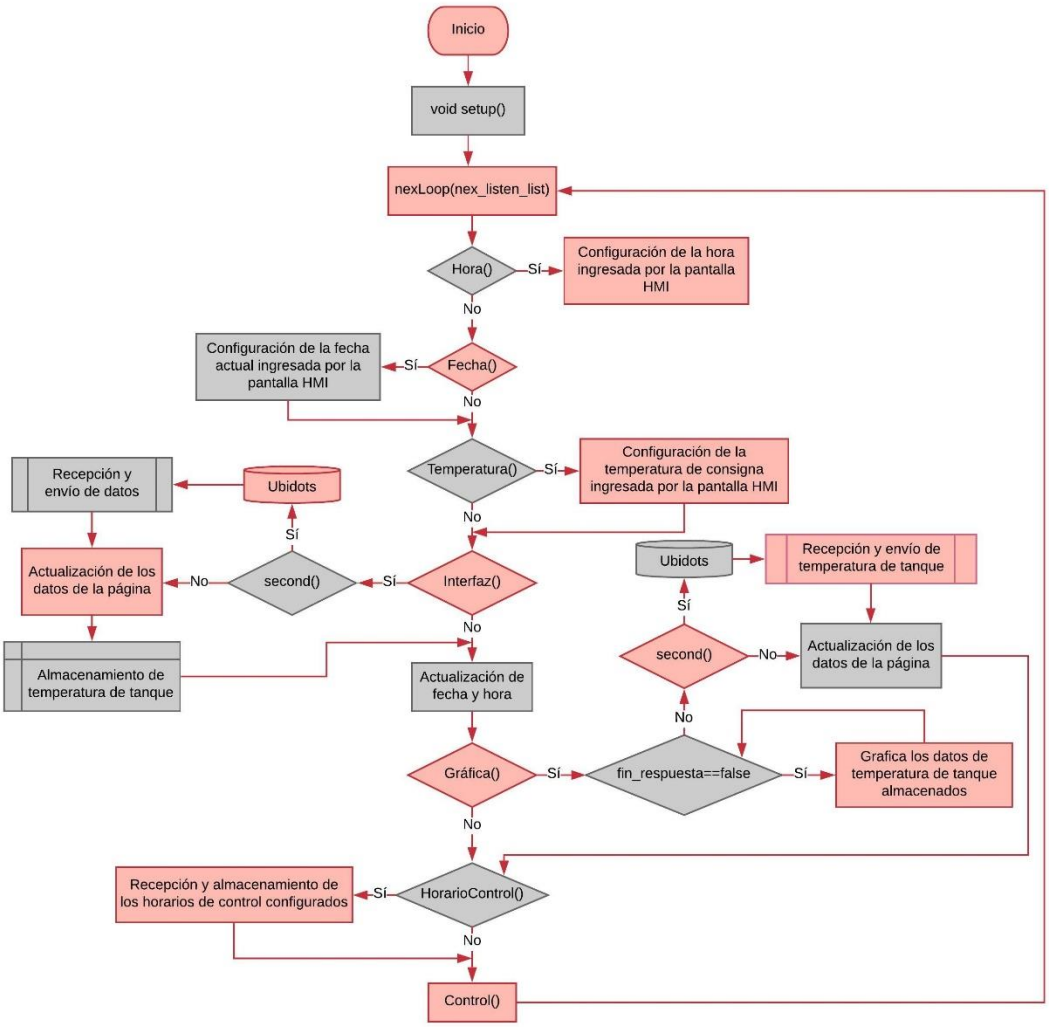


Figura 3.13: Comportamiento de la estación de monitoreo

La programación de la estación de monitoreo se encuentra en el Anexo 2.

3.5 Desarrollo de control

El control del proceso en estudio puede ser de varias formas debido a su comportamiento y por ser un sistema de primer orden; por lo tanto, se consideró aplicar un control ON/OFF y un control PI. En comparación, el control ON/OFF es más rústico carente de precisión dado que es de tipo todo o nada y apaga el sistema auxiliar una vez que la variable manipulada ha alcanzado o superado el valor de consigna. Por otra parte, el control PI a través de su término proporcional trabaja directamente sobre el error, que no es más que la diferencia entre el valor

de consigna y la variable manipulada, hasta reducirlo y con su término integral se pretende eliminar totalmente el error en estado estacionario para conseguir una estabilidad en el sistema.

Conforme a lo expresado y al tener la disponibilidad de una librería desarrollada para el control de sistemas, se maneja la librería PID [25] de Arduino que cumple con cada una de las cualidades que caracterizan a un control PID analógico, además de las causas que podrían comprometer la efectividad del control [26]. La ventaja de usar esta librería es que mediante una sentencia se pueden definir las constantes Kp, Ki y Kd, el valor de consigna, la variable manipulada y la variable de salida sin necesidad de escribir varias líneas de código. Definidos dichos parámetros, internamente la librería realizará todo lo referente a la teoría de control como es el cálculo del error, el término proporcional y el término integral para posteriormente obtener el resultado de la suma de ambos términos que modificará el estado de la variable de control que actuará sobre la resistencia.

Por último, para determinar las ganancias Kp, Ki y Kd no es necesario la identificación de la planta siempre que se aplique el método de Ziegler Nichols. Este método consiste en igualar a cero la ganancia derivativa e integrativa respectivamente, y aumentar la constante Kp hasta encontrar un valor que permita oscilar al sistema de forma estable al cual se la denominará ganancia crítica Kc. Posterior a ello, se hace uso de la Tabla 3 para calcular el valor de la constante Ki hasta obtener la estabilidad del sistema [27].

	Kp	**Ki**	**Kd**
P	0.50Kc		
PI	0.45Kc	0.54Kc/Tc	
ID	0.59Kc	1.18Kc/Tc	0.074KcTc

Tabla 3: Ajuste de ganancias para control PI

3.6 Desarrollo de aplicación móvil

La aplicación se realizó con la plataforma web App Inventor, descrita en el capítulo 2, con el fin de monitorear y controlar el sistema desde cualquier ubicación y no únicamente por medio de la interfaz gráfica instalada dentro de la

vivienda. Acerca de la programación, esta se encuentra basada en un lenguaje gráfico donde cada bloque representa una función en específico tales como la inserción de texto o diseño de la pantalla; por otro lado, los objetos empleados son básicamente botones, labels para los títulos, registros y casillas de texto para la escritura de datos como las temperaturas del tanque auxiliar y el colector solar medidos en grados Celsius. Las ventanas que componen la aplicación corresponden al inicio de sesión, visualización de datos e historial de temperatura respectivamente.

De manera concisa, en la primera ventana se dará la bienvenida y se solicitará el usuario y contraseña de ingreso que disponga el beneficiario. Esta sección cuenta con todas las validaciones pertinentes puesto que, si los datos solicitados son incorrectos, el ingreso al sistema será inviable y se presentarán los avisos correspondientes. Bajo estas circunstancias se procura brindar cierto grado de seguridad para evitar que una persona ajena al proceso sea capaz de controlarlo. En la Figura 3.14 se ilustra el algoritmo de ingreso a la aplicación móvil.

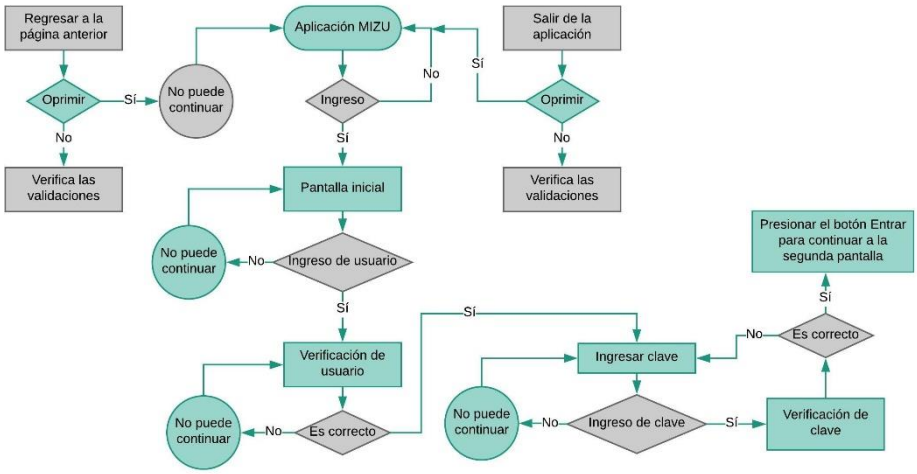


Figura 3.14: Verificación e ingreso a la aplicación móvil

Una vez verificada la cuenta y haber ingresado exitosamente, la aplicación móvil da paso a la segunda ventana. A través de ella se puede conocer la temperatura actual del tanque auxiliar, así como la temperatura de consigna, que a su vez puede ser modificada, y de esta manera inferir si el funcionamiento del sistema

solar térmico es óptimo para cumplir con el requisito de temperatura del consumidor o si será necesario que el sistema auxiliar entre en operación para lo cual se dispone de un botón de activación en caso de que el consumidor desee que el proceso de calefacción inicie fuera de los horarios de control; bajo esta conjetura, la activación se ejecuta al inicializar una variable tipo bandera para indicar al controlador la petición del consumidor. En consonancia con lo antes mencionado, el cambio de la temperatura de referencia puede ser alterado siempre que el valor a ingresar no sea menor que la temperatura del tanque auxiliar. Además, dado el supuesto caso que el usuario esté interesado en saber la variación de temperatura del tanque, la ventana de visualización dispone de un botón que direcciona a la última ventana de la aplicación. Véase el algoritmo de la Figura 3.15.

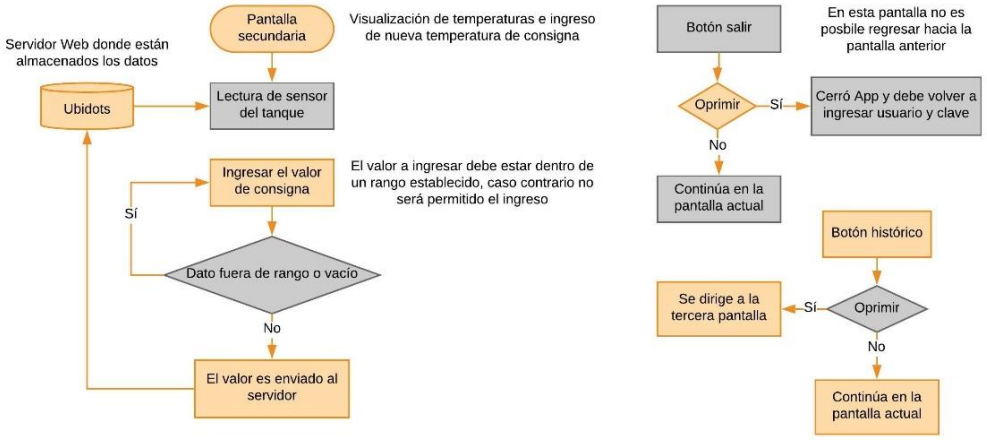


Figura 3.15: Ventana de monitoreo y ajuste de temperatura

De acuerdo con lo indicado, en la última ventana el usuario tiene la posibilidad de acceder a un historial de temperatura correspondiente al tanque auxiliar. La adquisición de datos se da cada 1000 milisegundos según lo programado en la aplicación y posterior a ello el dato se almacena en un registro que se irá actualizando periódicamente. En la

Figura 3.16 se observa la secuencia realizada para la adquisición de datos.

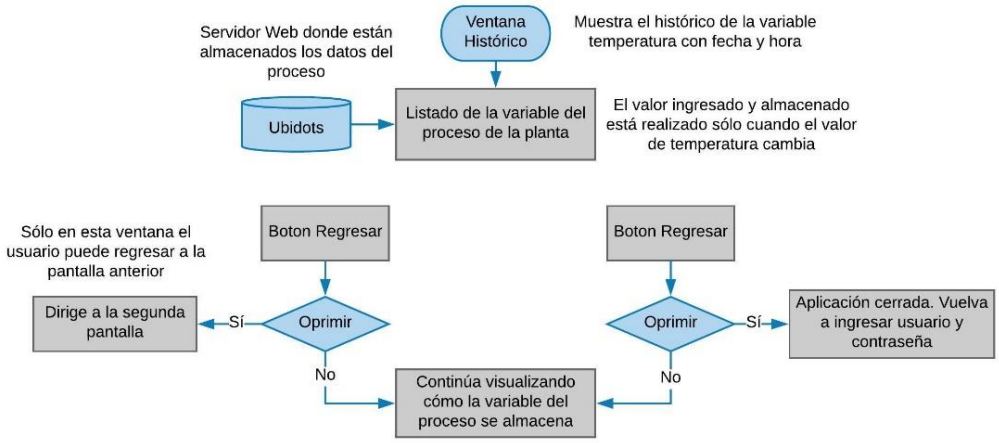


Figura 3.16: Creación de historial de temperatura

Para una mejor noción acerca de la programación de la aplicación remota, véase el Anexo 4.

3.7 Diseño de circuitos

Considerando que los equipos de campo estarán alejados de un suministro de voltaje DC y que además se requieren de tarjetas de aislamiento para las distintas etapas y acondicionamiento para los sensores, se realizaron esquemáticos y simulaciones de circuitos electrónicos haciendo uso del software Proteus que además incorpora la herramienta ARES para el desarrollo y diseño de PCB.

Para empezar, la tarjeta de desarrollo junto con los demás equipos de campo es alimentados con una fuente de alimentación DC variable de 0 – 24 Voltios. Como fuente principal de voltaje alterno se implementó un transformador con una relación de 110/120-12 Vac, donde su devanado secundario es conectado a un puente de diodos para rectificar el voltaje. El voltaje rectificado ingresa por uno de los terminales del integrado LM317 y mediante un potenciómetro se consigue la variación de voltaje DC en el terminal de salida con una corriente de aproximadamente 1 Amperio. Con la finalidad de estabilizar las señales de voltaje, se conectan capacitores electrolíticos en los terminales del integrado y así evitar las fluctuaciones, además tiene un diodo de protección contra retornos entre los terminales de entrada y salida. En la Figura 3.17 y Figura 3.18 se puede

observar tanto el esquemático de la fuente de voltaje DC variable como el diseño en PCB realizado en ARES.

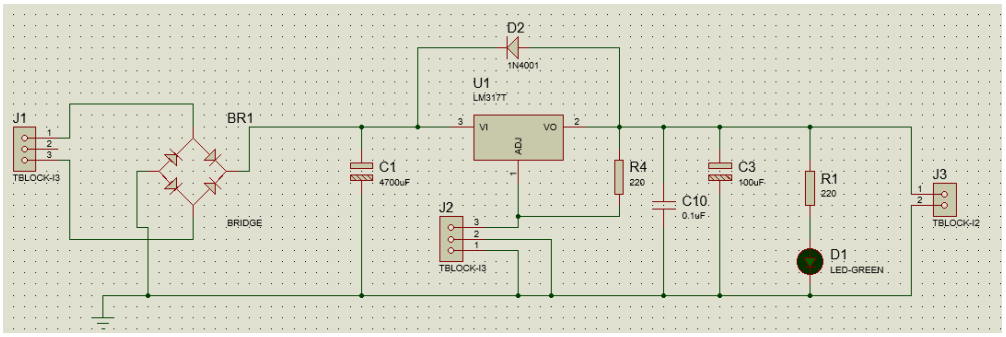


Figura 3.17: Esquema de fuente variable

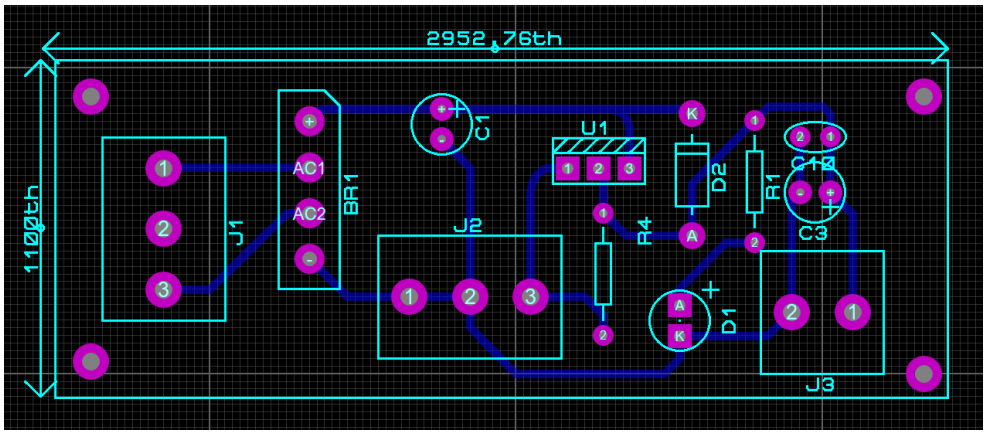


Figura 3.18: Diseño en PCB de fuente variable

En segunda instancia, para los sensores de temperatura DS18B20 es necesario el acondicionamiento de señal. En contraste con otros sensores, este es de tipo digital y, según las recomendaciones del fabricante, requiere de una resistencia de 4.7 kΩ entre el terminal de alimentación y el terminal de dato para su funcionamiento. Es importante mencionar que en la tarjeta de desarrollo basta con definir un solo puerto digital como entrada para la conexión de dos o más sensores. En la Figura 3.19 y Figura 3.20 se puede observar el diseño final de la tarjeta de acondicionamiento.

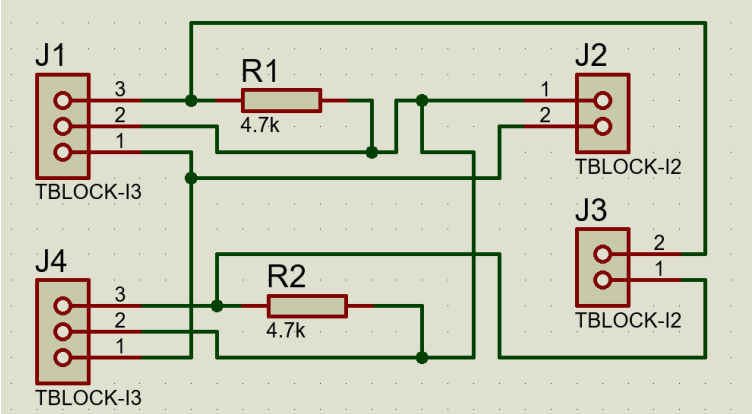


Figura 3.19: Esquema de placa para acondicionamiento de señal

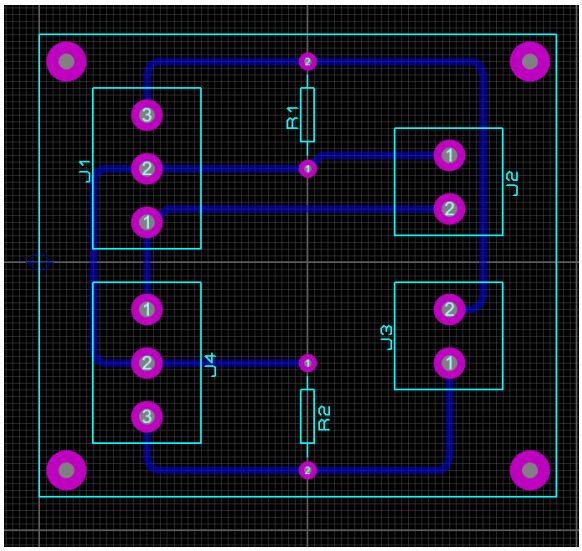


Figura 3.20: Diseño en PCB de placa para acondicionamiento de señal

En la actualidad, el diseño del control para un proceso puede ser ejecutado de diversas maneras, ya sea mediante circuitos electrónicos analógicos o de manera digital basados en una programación bien estructurada. Para el desarrollo del proyecto, la etapa de control fue realizada por medio de una programación que enviará constantemente señales para activar y desactivar el sistema auxiliar, siendo necesario una etapa de aislamiento que garantice una protección de todos los dispositivos evitando que los picos de corriente y sobretensiones producidas en la etapa de potencia destruyan la etapa de control.

De acuerdo con lo expresado en el párrafo precedente, para la etapa aisladora se hizo uso del optoacoplador MOC3021 porque su lado secundario es compatible con el voltaje de alimentación del sistema auxiliar; en otras palabras, este interruptor óptico es capaz de soportar un voltaje alterno de hasta 220 Vrms. Sin embargo, no hay que confundir conceptos ya que estas características del optoacoplador solo sirven para que funcione como intermediario entre la etapa de voltaje directo y la etapa de voltaje alterno. Ahora bien, si de activación del sistema de apoyo se trata, se tiene que aplicar un dispositivo semiconductor de potencia con la capacidad de soportar fuertes voltajes y altas corrientes como es el caso del TRIAC BT139-600E. A continuación, en la Figura 3.21 y Figura 3.22 se presenta el diseño del circuito implementado.

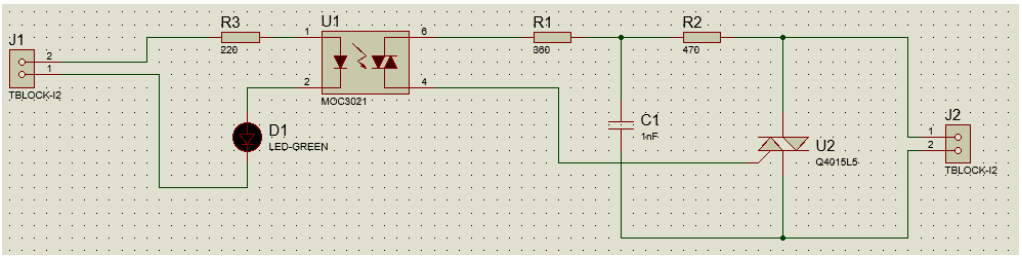


Figura 3.21: Esquema de placa para activación de etapa de potencia

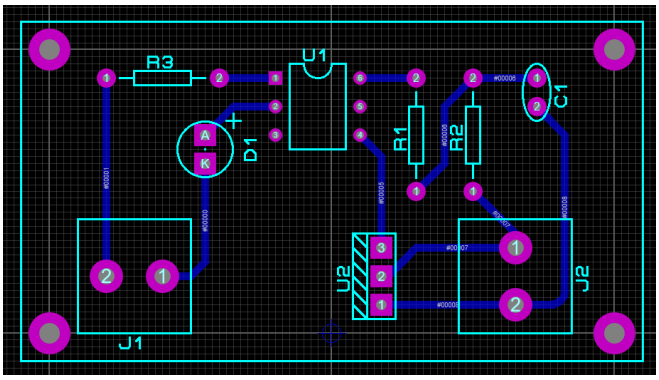


Figura 3.22: Diseño en PCB de placa para activación de etapa de potencia

Realizadas las pruebas y habiendo obtenido resultados favorables, se decidió usar un SSR (Solid State Relay), ya que presenta la ventaja de disponer en su circuitería un integrado de cruce por cero que protege la vida útil de la resistencia calefactora.

3.8 Montaje del sistema

Considerando ahora, que la programación ha sido concluida, y que los circuitos electrónicos fueron correctamente diseñados y han pasado las fases de prueba, se procede al montaje de cada etapa. En particular, el sistema se divide en dos partes compuestas por los distintos equipos electrónicos que tendrán las funcionalidades descritas a lo largo de este capítulo. De manera breve, la estación de monitoreo será integrada por los dispositivos que componen a la interfaz gráfica, y en campo todo lo relacionado al control y el circuito electrónico de potencia.

3.8.1 Estación de monitoreo

Cargada la programación del algoritmo visto en la Figura 3.13, y según el puerto serial usado para la comunicación entre la pantalla TFT y la placa Arduino Mega 2560, se realizan las conexiones convenientes. Para una mejor ilustración, véase la Figura 3.23.

Figura 3.23: Estación de monitoreo

3.8.2 Estación de campo

Como es de suponer, la etapa de control en conjunto con el circuito de potencia formará un solo sistema conectado a Ethernet de manera

inalámbrica debido a que su ubicación es en un área donde el acceso a Internet por medio de cable estructurado es muy complejo. En la Tabla 4 se especifica la conexión que se debe realizar entre la placa Arduino Mega y el módulo ESP-01.

Para activar la etapa de potencia, el lado primario del relé de estado sólido es conectado a un puerto con salida PWM de la placa de desarrollo que proveerá un tren de pulso para iniciar la operación del sistema auxiliar por medio del ancho de pulso, lo que deriva al aumento o disminución de la corriente que llega a la resistencia eléctrica según el control PI dependiente de la proximidad a la que esté la temperatura del tanque de la temperatura de consigna. El puerto elegido es el puerto 13, que en la placa Arduino Mega tiene una frecuencia de 980 Hz. El diseño del tablero puede ser visto en la Figura 3.24.

ESP-01	Arduino Mega 2560
RXD	TX2
GPIO0	___
GPIO2	___
GND	GND
VCC	3.3V
RST	___
CH_PD	3.3V
TXD	RX2

Tabla 4: Comunicación entre la placa Arduino Mega y TFT 2.8”

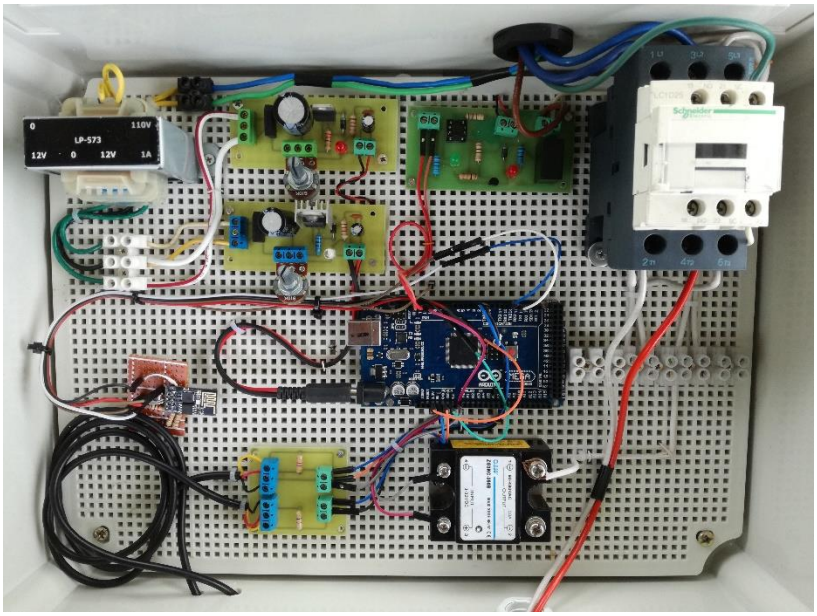


Figura 3.24: Tablero de control

CAPÍTULO 4

4. ANÁLISIS DE RESULTADOS.

En este capítulo se presentarán los resultados conseguidos durante el desarrollo del proyecto. Para detallar los resultados en las etapas más sustanciales del sistema, el capítulo fue dividido en subíndices que se extenderán si el caso lo amerita.

4.1 Funcionamiento del Sistema

Considerando que las condiciones del control básicamente se ajustan desde la estación de monitoreo, hay que tomar en cuenta que de igual manera el sistema de apoyo puede ser iniciado desde la aplicación móvil; es decir, es un sistema compuesto por dos maestros y un esclavo donde las señales de control serán enviadas por la red. Una vez que han sido ensambladas y energizadas las tarjetas de desarrollo en conjunto con los dispositivos, es prudente esperar por la inicialización de cada uno de ellos para proceder con las configuraciones respectivas.

Cuando el sistema ha iniciado, las variables de tipo bandera, almacenamiento y demás tienen un valor predeterminado por defecto. Razón por la cual en la estación de monitoreo se dispone de un menú de configuración para todas las variables que intervienen; de esta manera se puede configurar indefinidamente la temperatura de consigna y los horarios de control para un proceso independiente de una operación manual. Hay que hacer énfasis que el sistema auxiliar operará siempre que la temperatura del agua proveniente del tanque del colector sea menor que la temperatura de consigna y si la hora es igual a uno de los tres horarios ajustados, o si bien, el consumidor inició el proceso desde la aplicación móvil.

Indiscutiblemente el sistema dependerá plenamente del tipo de red a la que estén conectados los dispositivos, afectando el tiempo de respuesta de las variables y provocando un retardo. No obstante, la calidad de conexión infiere una relación con el control únicamente en el periodo de gestión de datos. Recibida la información de las variables y cumpliendo las condiciones necesarias, el sistema auxiliar inicia el proceso de calefacción con un apagado automático. Durante el

tiempo que esté activo el proceso, la gestión de datos solo se reanuda para el envío de temperatura de tanque y la recepción de la temperatura de consigna. Véase la Figura 4.1.

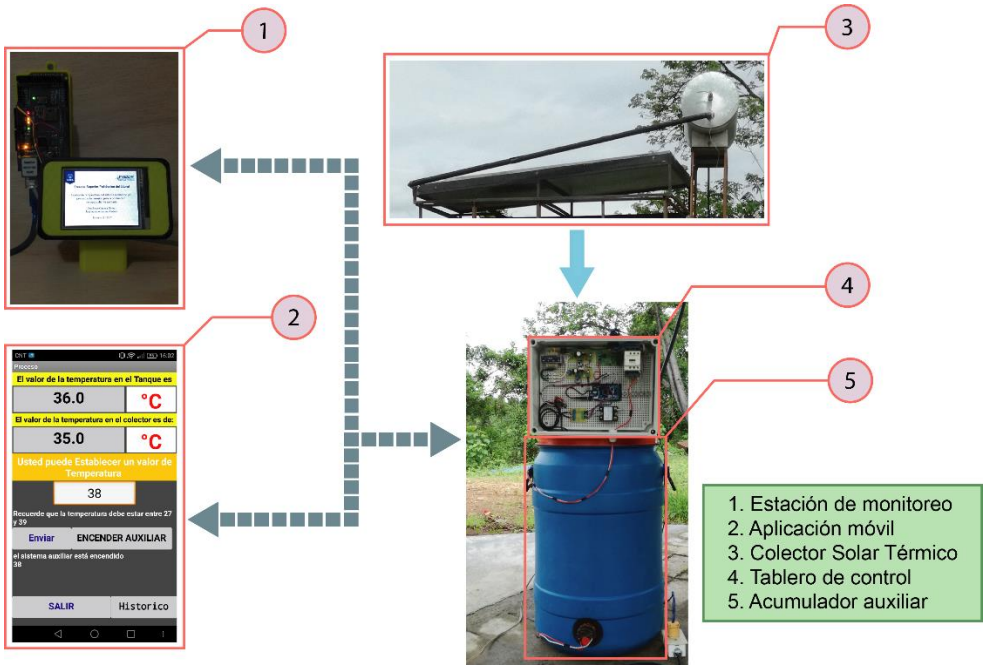


Figura 4.1: Esquema del proceso de calefacción

4.2 Comparación de los controles implementados

Los controles ON/OFF y PI fueron elegidos pensando en la naturaleza del sistema, ya que al ser un sistema lento de primer orden se necesita que el control actúe lo más rápido posible en el proceso. Por esta razón, el empleo de un control PID no es favorable; además gracias al término derivativo el tiempo de estabilización aumenta ocasionando que el proceso de calefacción tome más tiempo.

Para la producción de agua caliente sanitaria usando el sistema auxiliar eléctrico, se desarrollaron dos tipos de control con el objetivo de determinar cuál de ellos presenta mayor eficiencia. Antes de examinar los procesos de control, es oportuno mencionar que el reservorio actual tiene una capacidad de 120 litros; además, la temperatura ideal del agua para el consumo es de 37 °C, o máximo de 40 °C.

4.2.1 Control ON/OFF

Para las pruebas realizadas, se consideró la temperatura de consigna y la temperatura del agua depositada en el tanque, así como el horario al cual fue ajustado el control del sistema auxiliar.

Hora	Temperatura Acumulador Auxiliar [°C]
16:30	26
16:31	26,8
16:32	27,5
16:33	28,2
16:34	29
16:35	29,7
16:36	30,3
16:37	31,1
16:38	31,9
16:39	32,5
16:40	33,2
16:41	34
16:42	34,6
16:43	35,1
16:44	35,8
16:45	36,5
16:46	37,2
16:47	38
16:48	38,7
16:49	39,3
16:50	40
16:51	40,6
16:52	41,2

Tabla 5: Comportamiento de la temperatura del agua con control ON/OFF para un día parcialmente nublado

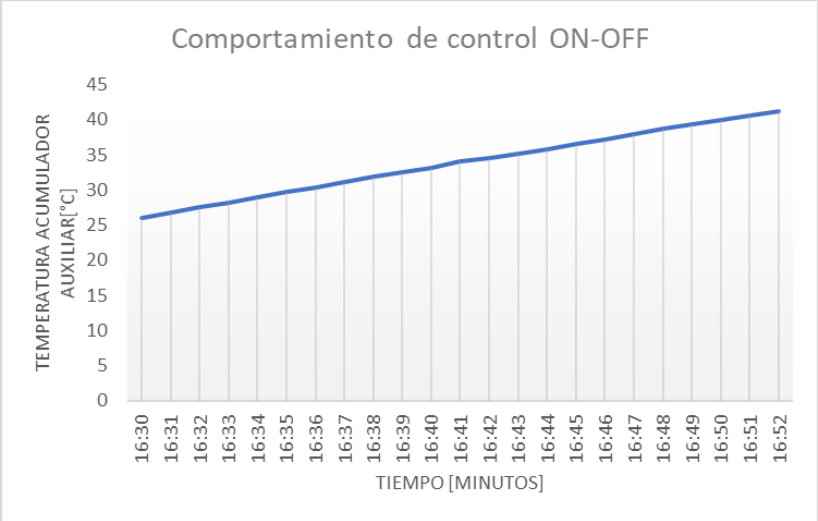


Figura 4.2: Gráfica de temperatura correspondiente al control ON/OFF para un día parcialmente nublado

En la Tabla 5 y Figura 4.2 se ilustra el comportamiento del sistema en un día parcialmente nublado, teniendo como temperatura de consigna un valor de 40 °C, en tanto que la temperatura del agua depositada en el reservorio auxiliar es de 26 a 27 °C. Tomando en cuenta que se trata de un control todo o nada, además del volumen de agua antes declarado, el proceso de calefacción duró de 20 a 22 minutos aproximadamente.

Por otra parte, las pruebas para un día nublado donde la radiación solar es prácticamente nula, el colector mantiene la temperatura del agua a 28 °C, es decir, no transfiere calor al acumulador, por ende, es necesaria la intervención del sistema auxiliar. Para este caso, se requiere alcanzar una temperatura de consigna de 36 °C; el comportamiento del proceso se presenta en la Tabla 6 y la Figura 4.3.

Hora	Temperatura Acumulador Auxiliar [°C]
11:44	28
11:45	28
11:46	28
11:47	28
11:48	28
11:49	28
11:50	29
11:51	29
11:52	29
11:53	30
11:54	30
11:55	31
11:56	31
11:57	31
11:58	31
11:59	32
12:00	32
12:01	33
12:02	33
12:03	34
12:04	34
12:05	34
12:06	34
12:07	35
12:08	35
12:09	35
12:10	35
12:11	36

Tabla 6: Comportamiento de la temperatura del agua con control ON/OFF para un día nublado

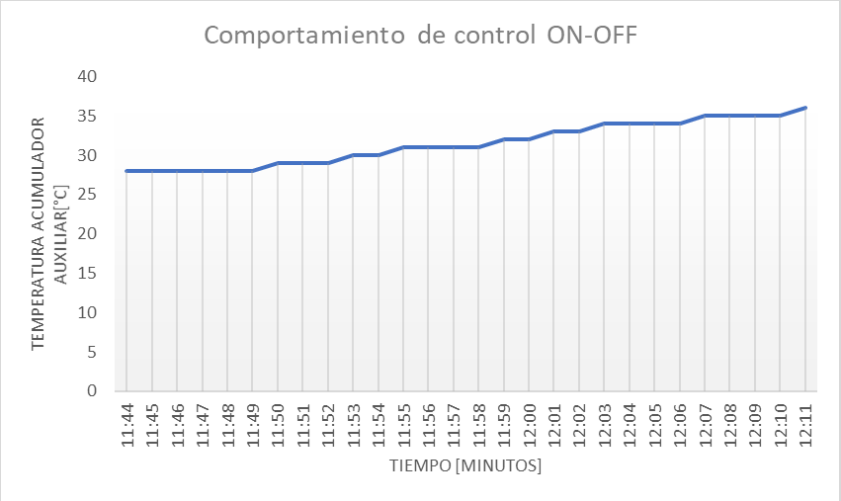


Figura 4.3: Gráfica de temperatura correspondiente al control ON/OFF para un día nublado

En otras palabras, aunque el sistema funcione adecuadamente, la lentitud del proceso está totalmente enlazada al volumen de agua que se pretende calentar, al clima y a la diferencia existente entre la temperatura del agua y la temperatura de referencia.

4.2.2 Control PI

Se trata de un control más exacto debido a las acciones provocadas por las constantes proporcional e integral que permiten regular la salida del controlador. Por consiguiente, siempre que la temperatura del agua esté por debajo de la temperatura de consigna, la salida del control permite que todo el voltaje alterno sea aplicado a la resistencia calefactora funcionando de igual manera que un control ON/FF; no obstante, a medida que la temperatura del agua se eleva y se acerca al valor de consigna, se disminuye el ancho de pulso a la salida de la tarjeta de control provocando el recorte de la señal alterna y, en consecuencia, presentando un suave aumento de temperatura para estabilizar el sistema y evitar que el agua supere el valor de consigna.

Hecha esta salvedad, con el agua almacenada en el reservorio del SST a una temperatura de 31 °C y ajustando una temperatura de consigna de

36 °C. A partir de 33 °C, el voltaje alterno que alimenta a la resistencia empieza a atenuarse puesto que el ancho de pulso de la señal de control, relacionado con la diferencia de temperatura, se reduce hasta llegar a cero. Sin embargo, a pesar de ser un control más preciso que el anterior, también resulta ser un proceso mucho más lento ya que la regulación del voltaje provoca una disminución de corriente directamente proporcional a la potencia de la resistencia calefactora. Véase la Tabla 7 y Figura 4.4.

De igual modo, en la Figura 4.5 y Tabla 8 se ilustra el comportamiento del sistema auxiliar para una temperatura de tanque de 34 °C con una temperatura de consigna de 40 °C para un día lluvioso.

Figura 4.4: Gráfica de temperatura del agua con control PI para un día parcialmente soleado

Hora	Temperatura Acumulador Auxiliar [°C]
11:15	31
11:16	31,2
11:17	31,35
11:18	31,6
11:19	31,8
11:20	31,95
11:21	32,04
11:22	32,15
11:23	32,4
11:24	32,7
11:25	32,89
11:26	33
11:27	33,1
11:28	33,25
11:29	33,45
11:30	33,7
11:31	33,92
11:32	34
11:33	34,15
11:34	34,4
11:35	34,66
11:36	34,83
11:37	35
11:38	35,07
11:39	35,1
11:40	35,15
11:41	35,2
11:42	35,3
11:43	35,43
11:44	35,57
11:45	35,66
11:46	35,77
11:47	35,83
11:48	35,9
11:49	35,94
11:50	36

Tabla 7: Comportamiento de la temperatura del agua con control PI para un día parcialmente soleado

Hora	Temperatura Acumulador Auxiliar [°C]
14:59	34
15:00	34
15:01	34
15:02	35
15:03	35
15:04	36
15:05	36
15:06	36
15:07	37
15:08	37
15:09	37
15:10	38
15:11	38
15:12	38
15:13	38
15:14	39
15:15	39
15:16	39
15:17	39
15:18	39
15:19	39
15:20	39
15:21	39
15:22	40
15:23	40
15:24	40
15:25	40
15:26	40
15:27	40
15:28	40

Tabla 8: Comportamiento de la temperatura del agua con control PI para un día parcialmente soleado

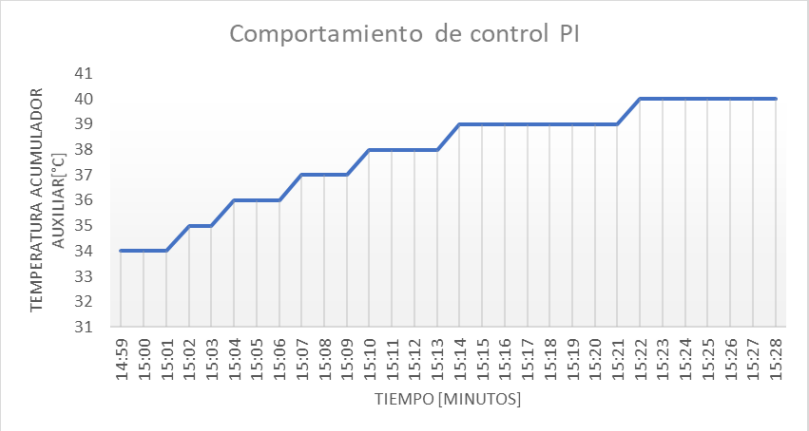


Figura 4.5: Comportamiento de la temperatura del agua con control PI para un día parcialmente soleado

Habiendo analizado las gráficas del sistema con control PI y reparando en que la diferencia de temperatura entre la variable manipulada y el punto de consigna es de aproximadamente 6 °C en ambas pruebas, se obtiene como resultado que el tiempo de operación de la resistencia calefactora es de 30 minutos aproximadamente.

Las implementaciones de varios tipos de control fueron realizadas con la finalidad de elegir aquel que tiene un mejor rendimiento conforme la naturaleza de cada uno. Para ello se consideraron los controles ON/OFF y PI; aunque inicialmente se tuvo en cuenta un control PID, pese a que la estabilidad y la disminución del error de estado estacionario es apreciable, este fue descartado ya que presenta un tiempo de estabilización, vinculado a su término derivativo, mucho mayor a los dos casos anteriores. Para solucionar el aspecto relacionado con el tiempo de estabilización, el desarrollo de un control PI es conveniente sin perder la efectividad del proceso, asegurando así un suave aumento de temperatura causado por la modulación de la señal de control evitando que el actuador opere durante todo el proceso en los límites de voltaje y corriente. Sin embargo, y habiendo dicho esto, un control ON/OFF es más apropiado para este tipo de sistema a pesar de trabajar en los límites de voltaje y corriente, pues el volumen de agua a calentar es considerable y

por ende no se produce un exceso de temperatura durante la calefacción, manteniendo el rendimiento, pero con una mayor eficacia.

4.3 Comunicación de dispositivos

En las dos estaciones que componen al proceso (monitoreo y campo), los dispositivos se conectan a los puertos se conectan a los puertos seriales preconfigurados en la programación para que el microcontrolador detecte la presencia del módulo o sensor y en consecuencia haya un flujo de datos bidireccional.

4.3.1 Estación de Monitoreo

La comunicación de los dispositivos en la fase de estación es justificada por la aplicación de librerías desarrolladas. Para la placa Shield Ethernet, el montaje sobre la placa Arduino no es el mayor de los problemas ya que será detectada fácilmente al ser un dispositivo perteneciente a la familia de Arduino. Sin embargo, no se puede manifestar lo mismo para el acceso a la plataforma Ubidots, pero con la incorporación de librería especializada al IDE de Arduino para el reconocimiento y manipulación de variables, además de la librería Ethernet para establecer la conexión a Internet, se consiguió resultados propicios para el proceso. Algo similar sucede con la pantalla TFT de Nextion que establece su comunicación una vez realizada todas las configuraciones con los comandos e instrucciones de la librería Nextion.h. La comprobación de las comunicaciones se observa en las respuestas recibidas a través del monitor serial, para ejemplificar la gestión de datos por medio de la Shield Ethernet véase la Figura 4.6.

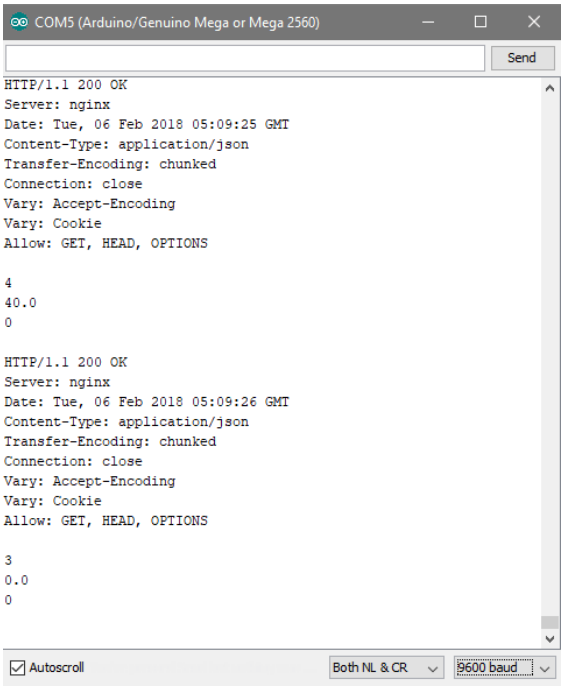


Figura 4.6: Verificación de gestión de datos

En el supuesto que el consumidor desee acceder a los datos de temperatura del tanque y analizar el comportamiento de temperatura del sistema en su totalidad, se puede retirar de la ranura de la Shield Ethernet, la tarjeta microSD encargada de almacenar los datos de interés junto a la fecha y hora de dicha adquisición.

4.3.2 Estación de Campo

Como se vio en el capítulo 3, el módulo ESP-01 no necesita de una librería ya que la programación para el manejo de datos que intervienen en el control se realizó mediante comandos AT, instrucciones en las que se basa el Firmware del módulo por defecto. Por el contrario, para la detección de sensor digital se requiere de la librería relacionada al dispositivo y escribir el comando de reconocimiento en la programación.

Para confirmar la gestión de datos y la modulación de la salida PWM para el control PI, véase la Figura 4.7.

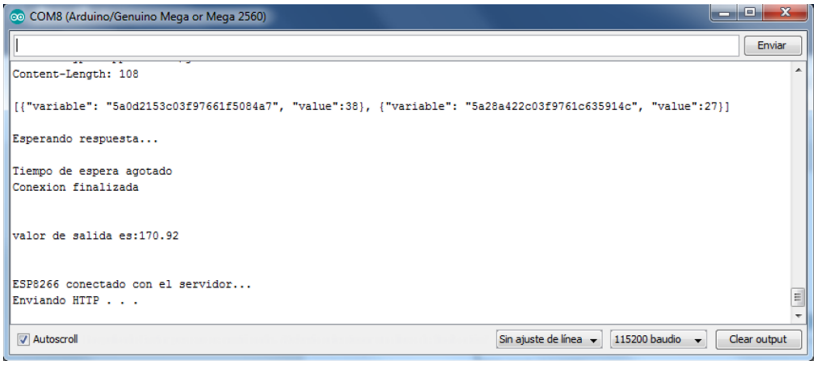


Figura 4.7: Verificación de datos y modulación de salida PWM

4.3.3 Aplicación remota

Con respecto a la aplicación remota, el consumidor puede a acceder a información de temperatura del tanque, así como el ajuste de un nuevo valor de consigna según sus intereses otorgando la ventaja de monitorear y controlar el sistema desde cualquier ubicación y no exclusivamente desde la estación de monitoreo instalada en la vivienda. Para que el usuario pueda ingresar, es necesario que disponga de una cuenta ya que de esta manera se ofrece un grado de seguridad para evitar que alguien externo al proceso altere el proceso de calefacción con valores no deseados de temperatura. En la Figura 4.8 se observa el diseño de la primera pantalla durante la programación de la aplicación; por otra parte, la ventana para la adquisición y muestreo de datos de temperatura y la configuración del valor de consigna es mostrada en la Figura 4.9.

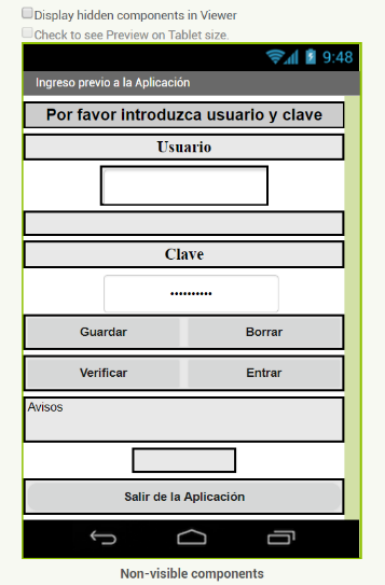


Figura 4.8: Diseño de ventana de ingreso a la aplicación

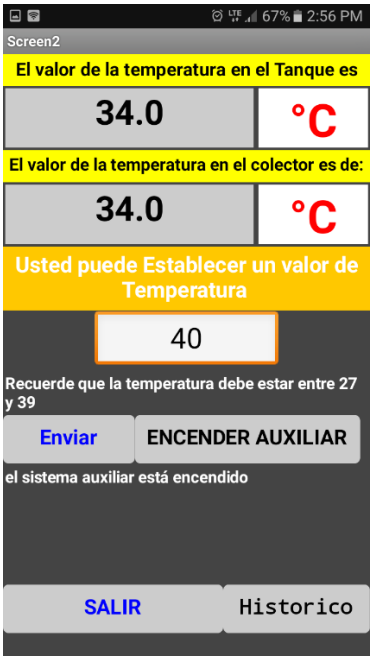


Figura 4.9: Ventana de monitoreo del sistema auxiliar

4.4 Problemas encontrados durante el desarrollo del proyecto

En el transcurso del proyecto se fueron presentando inconvenientes relacionados a la comunicación inalámbrica y al acoplamiento entre la tarjeta de potencia y la tarjeta de control.

4.4.1 Acoplamiento

Para efectuar un sistema de control confiable, es necesario una etapa de aislamiento entre la circuitería de control y los actuadores de potencia. Es por esta razón que el control de un sistema con voltaje alterno suele ser complicado, más aún cuando no se dispone de los recursos necesarios y se vuelve complicado conseguirlos.

En el caso de un control PI para un proceso térmico cuyo actuador es una resistencia calefactora con voltaje alterno de alimentación de 220/240 Voltios y un consumo máximo de 8 Amperios. inicialmente, para generar el control del sistema, a falta de un optoacoplador MOC3041, se utilizó un optoacoplador MOC3021 como etapa de aislamiento y un semiconductor de potencia TRIAC BT139-600E como etapa de activación. Los resultados obtenidos no fueron del todo favorables, ya que el proceso de calefacción fue ininterrumpido, pero la elevación de temperatura del semiconductor de potencia era considerable. La razón principal de la falla es el uso del optoacoplador MOC3021 ya que, al no contar con un circuito que detecte el cruce por cero a diferencia del MOC3041, los disparos en el semiconductor de potencia se efectúan en cualquier punto de la onda sinusoidal. Por este motivo, y al no disponer de un optoacoplador con dicha característica de detección, se implementó un relé de estado sólido, conocido también como SSR, que dispone de la circuitería especificada obteniendo un control más seguro sin comprometer la vida útil de los componentes del sistema. Para más información sobre el SSR implementado, véase el Anexo 5.

4.4.2 Comunicación módulo ESP-01

Como ya fue mencionado, el módulo ESP-01 tiene integrado un microcontrolador que viene preconfigurado con un Firmware determinado por el fabricante. A pesar de ello, la comunidad de Arduino se ha encargado de desarrollar distintos tipos de librerías con la finalidad de encontrar todos los recursos en un mismo paquete y simplificar el método de programación. Sin embargo, el uso de estas librerías no fue del todo

favorable pues, en algunos casos no satisfacían por completo las necesidades del proyecto, o simplemente presentaban errores en la compilación.

En consonancia con el párrafo anterior, los ajustes de fábrica fueron borrados en el momento que el primer código fue cargado al microcontrolador del módulo, razón por la cual es inútil el uso de comandos AT. No obstante, el restablecimiento del Firmware inicial es posible mediante el software ESP-01 Flash Downloader al cual se le carga el Firmware a grabar en la memoria del módulo. Sirva de ejemplo la Figura 4.10 para ilustrar el resultado final de la actualización de Firmware en el módulo ESP-01, y en la tabla Tabla 9 se indica la conexión entre la placa de desarrollo y el módulo ESP-01 para cargar el programa.

ESP-01	Arduino Mega 2560
RXD	RX
GPIO0	GND
GPIO2	___
GND	GND
VCC	3.3V
RST	___
CH_PD	3.3V
TXD	TX

Tabla 9: Conexión entre Arduino Mega y ESP-01

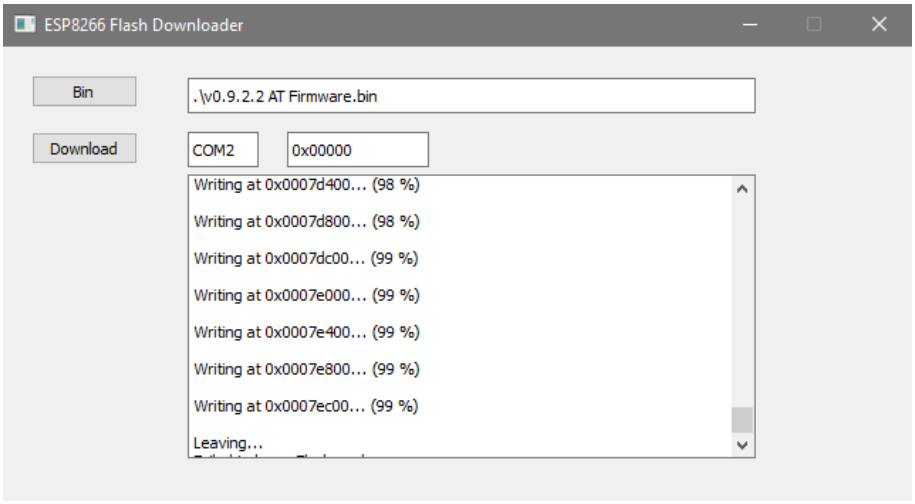


Figura 4.10: Actualización de Firmware

CONCLUSIONES Y RECOMENDACIONES

El presente proyecto tuvo como objetivo controlar el tiempo de operación de un sistema auxiliar para un Sistema Solar Térmico con la intención de obtener un proceso de calefacción amigable con el medio ambiente sin la dependencia neta de un sistema de corriente alterna para la producción de agua caliente sanitaria. Después de las pruebas realizadas con los dos tipos de control antes mencionados, se puede afirmar que la mejor alternativa para controlar la temperatura del proceso tratado es el uso de un control ON/OFF, ya que demanda un tiempo de operación mucho menor que un control PI al calentar un mismo nivel de agua con la misma diferencia de temperatura entre la temperatura manipulada y la temperatura de consigna. De esta manera, se reduce notablemente el consumo eléctrico en comparación con los sistemas convencionales destinados a la producción de agua caliente sanitaria suministrados por la red eléctrica.

El Sistema Solar Térmico presenta su mayor rendimiento en un día plenamente soleado con 5° de inclinación en dirección a la línea equinoccial; por lo tanto, cuando los rayos del sol inciden directamente sobre el colector, el agua almacenada se mantendrá a una temperatura equilibrada gracias a la envoltura aislante del acumulador y el material de fabricación, presentando pérdidas de calor poco relevantes. Es decir, el proceso de agua caliente sanitaria será obtenido únicamente por medio de energía natural sin un consumo eléctrico por parte del sistema auxiliar promoviendo la reducción de emisiones de CO_2, por lo que resulta de mayor ventaja que los sistemas habituales como la calefacción por sistemas a gas natural que infieren costos de mantenimiento elevados.

La conexión a Internet con el módulo wifi y Shield Ethernet con todo el proceso, permite lo que hoy por hoy es una tendencia de muchos sistemas y procesos, la adquisición y manejo de información disponible en cualquier lugar y de forma segura sin alterar proceso alguno. Por esta razón, como ya se estimaba, la comunicación con la red en todo momento entre sensores y controladores sería una de las partes importantes de la ejecución de todo. Podemos destacar que el servidor es un puente entre la planta y la estación de control y monitoreo. Por lo tanto, como se esperaba, esto permite que el sistema tenga un desarrollo ágil, versátil y confiable.

La conexión de los equipos a Internet es una parte esencial del sistema de control. Por esta razón, para un funcionamiento efectivo de la planta, se requiere de una red doméstica de alta calidad y que cada etapa del sistema sea instalada en un área donde los equipos puedan tener fácil acceso a la nube y así poder gestionar los datos que influyen directamente sobre el proceso en estudio. Dicha red, o redes, deben ser de gran velocidad además de estar libres de tráfico, ya que existe la posibilidad de una eventual recepción de datos basuras en caso de fallas en la red, comprometiendo el proceso de calefacción.

Con el propósito de mejorar el sistema en su totalidad, se sugiere el uso de una tarjeta de desarrollo más robusta que incluya módulos o circuitos integrados de múltiples funcionalidades, como módulo Wifi y puerto Ethernet, para evitar la compra o adquisición de dispositivos periféricos; así mismo, se precisan conocimientos de protocolos de comunicación para prevenir inconvenientes con el acceso a Internet. Por otra parte, considerando el volumen de agua al cual está sometido el proceso de calefacción, el uso de una resistencia de mayor potencia es indispensable con el fin de acelerar el proceso y mejorar el rendimiento del sistema para reducir los costos de operación.

BIBLIOGRAFÍA

[1] Instituto Nacional de Meteorología e Hidrología (INAMHI), “Anuario Meteorológico No. 52-2012” [Online]. Ecuador: INAMHI, 2015. Disponible en: http://www.serviciometeorologico.gob.ec/wp-content/uploads/anuarios/meteorologicos/Am%202012.pdf

[2] Pablo Kummetz, (2012, marzo 2) “La energía solar en América Latina: más que una promesa”. [Online]. Disponible en: http://www.dw.com/

[3] Jorge Pablo Díaz Velilla, “Tecnologías de aprovechamiento solar térmico” en Sistemas de energías renovables, 1ra Ed. España: Paraninfo, 2015, pp. 55-57.

[4] Melissa Echeverri, (2016, junio 11), “Empresas que apuestan por la energía solar” [Online]. Disponible en: https://www.larepublica.co

[5] Energía solar, (2015, septiembre 28). Captadores solares térmicos [Online]. Disponible en: https://solar-energia.net/energia-solar-termica/captadores-solares-termicos

[6] Carlos Hernández, Julieta C. Schallenberg, Ramón García, Gonzalo Piernavieja, Pedro Unamunzaga, Delia Cabrera, Mercedes Díaz, Javier Pardilla, Vicente Subiela, Gilberto Martel, “3. Energía solar térmica” en Energías renovables y eficiencia energética, 1ra Ed. España: Instituto Tecnológico de Canarias, 2008, pp. 54-55

[7] Camilo Lanata Giralt, “Manual de Sistemas Solares Térmicos”, Minvu, Santiago, Chile, ISBN (978-956-9432-04-0), 2014

[8] Omega, (2017, noviembre 7). Sensores RTD (Pt100) [Online]. Disponible en: https://www.omega.com/prodinfo/rtd.html

[9] K. Aström, T. Häglund, “PID Control” en PID Controllers: Theory, Design, and Tuning, 2nd Ed. United States of America: Instrument Society of America, 2015, pp. 60-69.

[10] Óscar Torrente Artero, “Capítulo 2: Hardware Arduino” en ARDUINO Curso práctico de información, 1ra Ed. México: Alfaomega, 2013, pp. 64-65.

[11] Arduino, (2017, noviembre 9). Arduino Mega Rev3 [Online]. Disponible en: https://store.arduino.cc/usa/arduino-mega-2560-rev3

[12] Arduino, (2017, noviembre 9). Arduino Ethernet Rev3 without POE [Online]. Disponible en: https://store.arduino.cc/usa/arduino-ethernet-rev3-without-poe

[13] Microchip, (2017, noviembre 9). ESP8266 Serial Esp-01 WIFI Wireless [Online]. Disponible en: http://www.microchip.ua

[14] Nextion, (2017, marzo 3). NX3224T028 [Online]. Disponible en: https://www.itead.cc/wiki/NX3224T028

[15] Óscar Torrente Artero, “Capítulo 3: Software Arduino” en ARDUINO Curso práctico de información, 1ra Ed. México: Alfaomega, 2013, pp. 129-154.

[16] Nextion (2017, junio 29). Nextion Editor Quick Start Guide [Online]. Disponible en: https://www.itead.cc/wiki/Nextion_Editor_Quick_Start_Guide#Debug.2C_online_simulator

[17] Ubidots, (2018, enero 23), “About Us” [Online]. Disponible en: https://ubidots.com/

[18] Jeremy Blum, “Capítuo 1: Getting Up and Blinking with the Arduino” en Exploring Arduino: Tools and Techniques for Enigneering Wizardry, 1ra Ed. Indianapolis: John Wiley & Sons, Inc., 2013, pp. 16-18.

[19] C. A. Larreta, “Diseño y construcción de un calentador de agua solar, económico y asequible a la clase media-baja del Ecuador”. FIMCP, Escuela Superior Politécnica del Litoral, Guayaquil, Ecuador, 2015.

[20] Gustavo Andrés Angulo. (2017, diciembre 13). Librería Ubidots-Ethernet [Online]. Disponible en: https://github.com/ubidots/ubidots-arduino-ethernet

[21] (2016, Agosto 23). Arduino y ESP8266 como cliente web [Online]. Disponible en: http://www.naylampmechatronics.com

[22] Alejandro Esquiva Rodríguez (2018, enero 17). JSON I - ¿Qué es y para qué sirve JSON? [Online]. Disponible en: https://geekytheory.com/json-i-que-es-y-para-que-sirve-json/

[23] Ubidots. (2018, febrero 6). REST API Reference [Online]. Disponible en: https://ubidots.com/docs/api/index.html#rest-api-reference

[24] (2017, octubre 25). ITEADLIB_Arduino_Nextion [Online]. Disponible en: https://github.com/itead/ITEADLIB_Arduino_Nextion

[25] (2017, diciembre 17). PID_Library [Online]. Disponible en: https://github.com/br3ttb/Arduino-PID-Library/

[26] Brett Beauregard. (2011, Abril 15). Improving the Beginner's PID [Online]. Disponible en: http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/

ANEXOS

ANEXO 1. Datasheet Termistor DS18B20

PRELIMINARY

DALLAS SEMICONDUCTOR

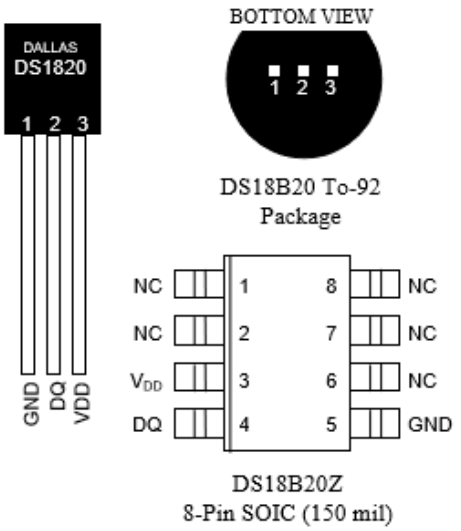
DS18B20
Programmable Resolution 1-Wire® Digital Thermometer

www.dalsemi.com

FEATURES

- Unique 1-Wire interface requires only one port pin for communication
- Multidrop capability simplifies distributed temperature sensing applications
- Requires no external components
- Can be powered from data line. Power supply range is 3.0V to 5.5V
- Zero standby power required
- Measures temperatures from -55°C to +125°C. Fahrenheit equivalent is -67°F to +257°F
- ±0.5°C accuracy from -10°C to +85°C
- Thermometer resolution is programmable from 9 to 12 bits
- Converts 12-bit temperature to digital word in 750 ms (max.)
- User-definable, nonvolatile temperature alarm settings
- Alarm search command identifies and addresses devices whose temperature is outside of programmed limits (temperature alarm condition)
- Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermally sensitive system

PIN ASSIGNMENT



PIN DESCRIPTION

GND - Ground
DQ - Data In/Out
V_{DD} - Power Supply Voltage
NC - No Connect

DESCRIPTION

The DS18B20 Digital Thermometer provides 9 to 12-bit (configurable) temperature readings which indicate the temperature of the device.

Information is sent to/from the DS18B20 over a 1-Wire interface, so that only one wire (and ground) needs to be connected from a central microprocessor to a DS18B20. Power for reading, writing, and performing temperature conversions can be derived from the data line itself with no need for an external power source.

Because each DS18B20 contains a unique silicon serial number, multiple DS18B20s can exist on the same 1-Wire bus. This allows for placing temperature sensors in many different places. Applications where this feature is useful include HVAC environmental controls, sensing temperatures inside buildings, equipment or machinery, and process monitoring and control.

PIN TO92	Símbolo	Descripción
1	GND	Tierra
2	DQ	Dato de entrada / Dato de salida
3	V_{DD}	Debe ser aterrizada en modo de operación parásita

Tabla 10: Descripción de puertos del sensor DS18B20

ANEXO 2. Programación de la Estación de Monitoreo

Programa Principal

```
//DECLARACIÓN DE LIBRERÍAS
//Librería para pantalla TFT
#include <Nextion.h>

//Librería para reloj y hora
#include <Time.h>
#include <TimeLib.h>

//Librería para Shield Ethernet
#include <Ethernet.h>
#include <SPI.h>
#include <UbidotsEthernet.h>

//Librería para lectura y escritura de tarjeta SD
#include <SD.h>
File Historico;

//DEFINICIÓN DE PARÁMETROS NECESARIOS PARA CONEXIÓN A ETHERNET Y ENLACE A UBIDOTS
//Parámetros para Ubidots
#define Shield  "arduino-ethernet" // Nombre del dispositivo donde se alojan las variables en Ubidots
#define TOKEN "A1E-Hmk2EbtdckW7sVBiqkz4GbyhZ6DNn2" // TOKEN de Ubidots
```

```
//Variables de envío de datos
#define ID_TempSetpoint "temperatura_setpoint"  // Temperatura requerida que se parametriza desde la pantalla TFT para enviar a Ubidots
#define ID_Bandera "bandera_control"  // Temperatura requerida que se parametriza desde la pantalla TFT para enviar a Ubidots
//Variables de recepción de datos
#define ID_TemperaturaTanque  "temperatura_tanque"  //Recepta el dato de temperatura del tanque
#define ID_TemperaturaColector  "temperatura_colector"   //Recepta el dato de temperatura del colector
#define ID_BanderaSistema  "bandera_sistema"   //Recepta el dato de temperatura del colector

//Parámetros para Shield Ethernet
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //Dirección MAC de Shield Ethernet
IPAddress ip(192, 168, 0, 108); //Cambia según la Red
Ubidots client(TOKEN);

//DECLARACIÓN DE VARIABLES GLOBALES
#define Verificacion 13
#define chipSelect 4 //Activa las comunicaciones SPI de la tarjeta SD

int PaginaActual, hora, minuto, dia, mes, anio, temperaturaSP, temptanquebarra, tempcolectorbarra, horain, minutoin , diain , mesin, anioin, horario1, horario2, horario3, horario1in, horario2in, horario3in, temperaturain, temptanqueshield, tempcolectorshield, temperaturatemp, tempSPshield, banderaControl, banderaSistema, tiempotemp, diatemp, mestemp, aniotemp;

float costo, contador, consumo, consumodiario, consumomensual, consumoanual, voltaje, corriente;
```

```
char textotemptanque[100], textotempcolector[100], textoconsumodia[100],
textoconsumomes[100], textoconsumoanio[100];

boolean fin_respuesta = false, condicion = false;

uint32_t temporalhora; //Almacenará temporalmente los valores ingresados para
hora proveniente de la TFT
uint32_t temporalminuto; //Almacenará temporalmente los valores ingresados para
minutos proveniente de la TFT
uint32_t temporaldia; //Almacenará temporalmente los valores ingresados para día
proveniente de la TFT
uint32_t temporalmes; //Almacenará temporalmente los valores ingresados para
mes proveniente de la TFT
uint32_t temporalanio; //Almacenará temporalmente los valores ingresados para año
proveniente de la TFT
uint32_t temporaltemp; //Almacenará temporalmente los valores ingresados para
temperatura proveniente de la TFT
uint32_t temporalhorario1; //Almacenará temporalmente los valores ingresados para
horario inicial proveniente de la TFT
uint32_t temporalhorario2; //Almacenará temporalmente los valores ingresados para
horario intermedio proveniente de la TFT
uint32_t temporalhorario3; //Almacenará temporalmente los valores ingresados para
horario final proveniente de la TFT

//DECLARACIÓN DE PÁGINAS
NexPage Presentacion = NexPage(0, 0, "Presentacion");
NexPage ConfigReloj = NexPage(1, 0, "ConfigReloj");
NexPage ConfigTemp = NexPage(2, 0, "ConfigTemp");
NexPage Principal = NexPage(3, 0, "Principal");
NexPage ConfigFecha = NexPage(4, 0, "ConfiFecha");
NexPage GraficaSensor = NexPage(5, 0, "GraficaSensor");
NexPage Horario = NexPage(6, 0, "Horario");
```

```
NexPage Consumo = NexPage(7, 0, "Consumo");

//DECLARACIÓN DE OBJETOS TFT
//Página de Interfaz
//Muestra en tiempo real los datos de hora y minuto
NexNumber n0 = NexNumber(3, 9, "n0"); //Variable numérica para la hora
NexNumber n1 = NexNumber(3, 1, "n1"); //Variable numérica para los minutos
//Almacena la temperatura de referencia
NexNumber n2 = NexNumber(3, 2, "n2"); //Variable numérica para la temperatura de consigna
//Muestra en tiempo real los datos de día, mes y año
NexNumber n3 = NexNumber(3, 11, "n3"); //Variable numérica para guardar el día
NexNumber n4 = NexNumber(3, 12, "n4"); //Variable numérica para guardar el mes
NexNumber n5 = NexNumber(3, 13, "n5"); //Variable numérica para guardar el año
//Muestra la temperatura en una barra
NexProgressBar j0_3 = NexProgressBar(3, 7, "j0_3"); //Variable de objeto para mostrar la temperatura de tanque
NexProgressBar j1_3 = NexProgressBar(3, 16, "j1_3"); //Variable de objeto para mostrar la temperatura de colector
//Muestra en tiempo real la temperatura del tanque y colector
NexText t0_3 = NexText(3, 8, "t0_3"); //Variable tipo texto de objeto para mostrar la temperatura del tanque
NexText t1_3 = NexText(3, 17, "t1_3"); //Variable tipo texto de objeto para mostrar la temperatura del colector

//Página de Configuración de Reloj
//Almacena los datos de hora y minuto
NexNumber n0_1 = NexNumber(1, 7, "n0_1"); //variable numérica para guardar la hora
NexNumber n1_1 = NexNumber(1, 8, "n1_1"); //variable numérica para guardar el minuto
```

```
//Página de Configuración de Temperatura
//Almacena el dato de temperatura de referencia
NexNumber n0_2 = NexNumber(2, 5, "n0_2"); //variable numérica para el día

//Página de Configuración de Fecha
//Almacena los datos de día, mes y año
NexNumber n0_4 = NexNumber(4, 7, "n0_4"); //variable numérica para el día
NexNumber n1_4 = NexNumber(4, 8, "n1_4"); //variable numérica para el mes
NexNumber n2_4 = NexNumber(4, 9, "n2_4"); //variable numérica para el año

//Página de Gráfica de Temperatura
//Almacena los datos de horarios a calentar y ahorro
NexNumber n0_5 = NexNumber(5, 3, "n0_5"); //variable numérica para mostrar la
temperatura del sensor
NexNumber n1_5 = NexNumber(5, 4, "n1_5"); //variable numérica para horario uno
NexNumber n2_5 = NexNumber(5, 5, "n2_5"); //variable numérica para horario dos
NexNumber n3_5 = NexNumber(5, 6, "n3_5"); //variable numérica para horario tres
NexWaveform s0 = NexWaveform(5, 7, "s0"); //gráfica del tanque

//Página de Configuración de Horario de Control
//Almacena los tres horarios a usar el sistema
NexNumber n0_6 = NexNumber(6, 1, "n0_6"); //variable numérica para horario 1
NexNumber n1_6 = NexNumber(6, 2, "n1_6"); //variable numérica para horario 2
NexNumber n2_6 = NexNumber(6, 3, "n2_6"); //variable numérica para horario 3

//Página de Consumo
//Almacena el consumo energético
NexText t0_7 = NexText(7, 2, "t0_7"); //variable numérica para ahorro energético
NexText t1_7 = NexText(7, 3, "t1_7"); //variable numérica para ahorro energético
NexText t2_7 = NexText(7, 4, "t2_7"); //variable numérica para ahorro energético
```

```
//LISTA DE PUNTEROS
NexTouch *nex_listen_list[] = {&Presentacion, &ConfigReloj, &ConfigTemp,
&Principal, &ConfigFecha, &GraficaSensor, &Horario, &Consumo, NULL};

//FUNCIONES PARA LLAMADO DE LAS PANTALLAS
void PresentacionPushCallBack(void *ptr) //Si la página 0 es cargada en la pantalla,
se ejecuta lo siguiente
{
  PaginaActual = 0; //Configura la variable como 0 y Arduino sabe que esa página
está cargada
}

void ConfigRelojPushCallBack(void *ptr) //Si la página 1 es cargada en la pantalla,
se ejecuta lo siguiente
{
  PaginaActual = 1; //Configura la variable como 1 y Arduino abe que esa página
está cargada
}

void ConfigTempPushCallBack(void *ptr) //Si la página 2 es cargada en la pantalla,
se ejecuta lo siguiente
{
  PaginaActual = 2;  //Configura la variable como 2 y Arduino sabe que esa página
está cargada
}

void PrincipalPushCallBack(void *ptr) //Si la página 3 es cargada en la pantalla, se
ejecuta lo siguiente
{
  PaginaActual = 3; //Configura la variable como 3 y Arduino sabe que esa página
está cargada
}
```

```
void ConfigFechaPushCallBack(void *ptr) //Si la página 4 es cargada en la pantalla, se ejecuta lo siguiente
{
  PaginaActual = 4; //Configura la variable como 4 y Arduino sabe que esa página está cargada
}

void GraficaSensorPushCallBack(void *ptr) // Si la página 5 es cargada en la pantalla, se ejecuta lo siguiente
{
  PaginaActual = 5; //Configura la variable como 5 y Arduino sabe que esa página está cargada
}

void HorarioPushCallBack(void *ptr) //Si la página 6 es cargada en la pantalla, se ejecuta lo siguiente
{
  PaginaActual = 6; //Configura la variable como 6 y Arduino sabe que esa página está cargada
}

void ConsumoPushCallBack(void *ptr) //Si la página 7 es cargada en la pantalla, se ejecuta lo siguiente
{
  PaginaActual = 7; //Configura la variable como 7 y Arduino abe que esa página está cargada
}

void setup()
{
  pinMode(Verificacion, OUTPUT);
  digitalWrite(Verificacion, LOW);
```

```
//Inicialización de la librería
nexInit();
Serial3.begin(9600); //TFT

//Llamado a funciones
Presentacion.attachPush(PresentacionPushCallBack);
ConfigReloj.attachPush(ConfigRelojPushCallBack);
ConfigTemp.attachPush(ConfigTempPushCallBack);
Principal.attachPush(PrincipalPushCallBack);
ConfigFecha.attachPush(ConfigFechaPushCallBack);
GraficaSensor.attachPush(GraficaSensorPushCallBack);
Horario.attachPush(HorarioPushCallBack);
Consumo.attachPush(ConsumoPushCallBack);

setTime(01, 00, 00, dia, mes, anio);

Serial.begin(9600); //Ethernet Shield

//Configuración SD
Serial.print("Inicalizando tarjeta SD...");
if (!SD.begin(chipSelect))
{
  Serial.println("Error en la inicialización");
}
else
{
  Serial.println("Inicialización completa");
}
SD.remove("datos.txt");
```

```
  //Configuración W5100
    if (Ethernet.begin(mac) == 0)
    {
      //Configuración usando la dirección IP en lugar de DHCP
      Ethernet.begin(mac, ip);
    }

  delay(1000); //1 segundo de espera para inicializar la Shield

  Serial.println("Configuración completa");
}

void loop()
{
  digitalWrite(Verificacion, HIGH);
  delay(800);

  nexLoop(nex_listen_list);

  n0_5.setValue(temptanqueshield);

  //Pregunta si se encuentra en la página de configuración de hora
  Hora();

  //Pregunta si se encuentra en la página de configuración de fecha
  Fecha();

  //Pregunta si se encuentra en la página de configuración de temperatura
  Temperatura();

  //Pregunta si se encuentra en la página principal de la interfaz
  Interfaz();
```

```
  //Actualiza fecha y hora
  hora = hour();
  minuto = minute();
  dia = day();
  mes = month();
  anio = year();

  //Pregunta si se encuentra en la página de ahorro energético
  Grafica();

  //Pregunta si se encuentra en la página de configuración de horario
  HorarioControl();

  //Control
  Control();

  //Pregunta si la pantalla TFT se encuentra en la página de consumo eléctrico
  ConsumoElectrico();
}
```

Función Interfaz()

```
void Interfaz()
{
  if(PaginaActual==3) //Página de control
  {
    fin_respuesta=false;
    digitalWrite(Verificacion,LOW);

    delay(500);
    temptanqueshield=(int)client.getValue(Shield,ID_TemperaturaTanque);
    tempcolectorshield=(int)client.getValue(Shield,ID_TemperaturaColector);
    tempSPshield=(int)client.getValue(Shield,ID_TempSetpoint);
```

```
    banderaSistema=(int)client.getValue(Shield,ID_BanderaSistema);
    client.add(ID_Bandera,banderaControl);
    client.sendAll();

    EscrituraSD();

    //Actualiza constantemente los valores en pantalla
    n0.setValue(hora);
    n1.setValue(minuto);
    n2.setValue(t1);
    n3.setValue(dia);
    n4.setValue(mes);
    n5.setValue(anio);
    t0_3.setText(textotemptanque);
    j0_3.setValue(temptanquebarra);
    t1_3.setText(textotempcolector);
    j1_3.setValue(tempcolectorbarra);

    memset(textotemptanque,0,sizeof(textotemptanque));
    itoa(t2,textotemptanque,10);
    temptanquebarra=map(t2,0,100,0,100);

    memset(textotempcolector,0,sizeof(textotempcolector));
    itoa(tempcolectorshield,textotempcolector,10);
    tempcolectorbarra=map(tempcolectorshield,0,100,0,100);
  }
}
```

Función Grafica()

```
void Grafica()
{
  if(PaginaActual==5) //Ahorro Energético
  {
    if(fin_respuesta==false)
    {
      Historico = SD.open("datos.txt");
      if (Historico)   // Si ha podido abrir el fichero
      {
        int tempgraf=0;
        while (Historico.available())
        {
          char c = Historico. read();
          Serial.println(c);
          Serial.println(int(c));
          if(int(c)>=48 && int(c)<=57)
          {
            tempgraf=tempgraf*10 + (int(c)-48);
            Serial.println(tempgraf);
          }

          if(tempgraf>=20 && tempgraf<=99)
          {
            s0.addValue(0,tempgraf);
            tempgraf=0;
          }
          delay(100);
        }
        Historico.close();
      }
```

```
      else
      {
        Serial.println("Error abriendo el archivo datos.txt");
      }
    }
    fin_respuesta=true;
    s0.addValue(0, temperaturaSP);
    digitalWrite(Verificacion,LOW);
    delay(500);
    temptanqueshield=client.getValue(Shield,ID_TemperaturaTanque);
    EscrituraSD();
    temptanqueshield=(int)temptanqueshield;
    n1_5.setValue(horario1);
    n2_5.setValue(horario2);
    n3_5.setValue(horario3);
    s0.addValue(0, temptanqueshield);
  }
}
```

Control()

```
void Control()
{
  if(hora==horario1 || hora==horario2 || hora==horario3)
  {
    bCtrl=1;
  }
  else
  {
    bCtr=0;
  }
```

```
if((bSst==1 || bCtr==1) && temptanqueshield<=tempSPshield)
{
  condicion=false;
  if((int)minute()==tiempotemp+1)
  {
    contador=contador+1;
  }
  tiempotemp=(int)minute();
}

if(temptanqueshield>=tempSPshield)
{
  if(condicion==false)
  {
    consumo=(voltaje*corriente/1000)*costo*(contador/60);
    consumodiario=consumodiario+consumo;
    consumomensual=consumomensual+consumodiario;
    consumoanual=consumoanual+consumomensual;
    contador=0;
    condicion=true;
  }
}

//Encera la variable de consumo una vez transcurridas las 24 horas
if((int)day()==diatemp+1)
{
  consumodiario=0;
  SD.remove("datos.txt");
}
diatemp=(int)day()+1;
```

```
  //Encera la variable de consumo cuando pasa un mes
  if((int)month()==mestemp+1)
  {
    consumomensual=0;
  }
  mestemp=(int)month()+1;

  //Encera la variable de consumo una vez que se cumple un año
  if((int)year()==aniotemp+1)
  {
    consumoanual=0;
  }
  aniotemp=(int)year()+1;
}
```

ANEXO 3. Programación de la Estación de Campo

Programación Principal

```
#include <OneWire.h>

#include <DallasTemperature.h>


String ssid ="SSD";

String password="PSSWRD";

String server = "things.ubidots.com";


String temperaturacolector="temperatura_colector";

String IDcolector="5a28a422c03f9761c635914c";

String temperaturatanque="temperatura_tanque";

String IDtanque="5a0d2153c03f97661f5084a7";
```

```
String temperaturasetpoint="temperatura_setpoint";
String IDsetpoint="5a1df620c03f974024534c5a";
String banderasistema="bandera_sistema";
String IDsistema="5a43c7f3c03f97641b8066e2";
String banderacontrol="bandera_control";
String IDcontrol="5a28a40fc03f976266599c55";
String cadena="";

//String URLGET = "/api/v1.6/variables/";
//String Token="?token=A1E-Hmk2EbtdckW7sVBiqkz4GbyhZ6DNn2";

String Token="?token=A1E-Hmk2EbtdckW7sVBiqkz4GbyhZ6DNn2";

int sensortanque, tanquetemporal, tempsetpoint, sensorcolector, colectortemporal,
    temperaturasp, tempsptemp, bSst, bCtr, bct, bst, tempt;

#define Sensor 2
#define Sensorcolector 3
#define SistAux 13

OneWire oneWireObjeto(Sensor);
DallasTemperature sensorDS18B20(&oneWireObjeto);

OneWire oneWireObjetocolector(Sensorcolector);
DallasTemperature sensorDS18B20colector(&oneWireObjetocolector);
```

```
void setup()
{
  Serial2.begin(115200);
  Serial.begin(115200);
  Reset();
  delay(50);
  ConexionWifi();
  pinMode(SistAux,OUTPUT);
  digitalWrite(SistAux,LOW);
}

void loop ()
{
  //SENSORES DE TEMPERATURA
  sensorDS18B20.requestTemperatures();
  sensortanque=sensorDS18B20.getTempCByIndex(0);
  sensorDS18B20colector.requestTemperatures();
  sensorcolector=sensorDS18B20colector.getTempCByIndex(0);
  delay(500);

  //ENVIO Y RECEPCIÓN DE DATOS
  EnvioDatos(IDtanque,IDcolector,sensortanque,sensorcolector);
  delay(500);

  cadena=RecepcionDato(banderasistema);
```

```
bsistema=Separacion(cadena);
delay(500);

if(bsistema!=0 && bsistema!=1)
{
  bsistema=bst;
}
bst=bsistema;
Serial.println(bsistema);

cadena=RecepcionDato(banderacontrol);
bcontrol=Separacion(cadena);
delay(500);

if(bcontrol!=0 && bcontrol!=1)
{
  bcontrol=bct;
}
bct=bcontrol;
Serial.println(bcontrol);

cadena=RecepcionDato(temperaturasetpoint);
temperaturasp=Separacion(cadena);
tempsptemp=temperaturasp;
```

```
if(temperaturasp<26 || temperaturasp>48)
{
  temperaturasp=0;
}

Serial.println(temperaturasp);

//CONTROL
while((bsistema==1||bcontrol==1) && sensortanque<temperaturasp)
{
  sensorDS18B20.requestTemperatures();
  sensortanque=sensorDS18B20.getTempCByIndex(0);
  sensorDS18B20colector.requestTemperatures();
  sensorcolector=sensorDS18B20colector.getTempCByIndex(0);
  //EnvioDato(IDtanque,sensortanque);
  EnvioDatos(IDtanque,IDcolector,sensortanque,sensorcolector);

  digitalWrite(SistAux,HIGH);

  delay(1000);
  cadena=RecepcionDato(temperaturasetpoint);
  tempt=Separacion(cadena);
```

```
    if(tempt>=26 && tempt<=48)
    {
      temperaturasp=tempt;
    }

    if(sensortanque>=temperaturasp)
    {
      bsistema=0;
      EnvioDatos(IDtanque,IDsistema,sensortanque,bsistema);
    }
  }
  digitalWrite(SistAux,LOW);

  tanquetemporal=sensortanque;
  colectortemporal=sensorcolector;
}
```

Función ConexionWifi()

```
void ConexionWifi()
{
  Serial2.println("AT");

  //-----Configuración de red-------//

  //ESP8266 en modo estación
  Serial2.println("AT+CWMODE=1");
```

```
  //Conexión a Red Wifi
  Serial2.println("AT+CWJAP="+ssid+","+password);
  Serial2.setTimeout(10000);
  if(Serial2.find("OK"))
  {
    Serial.println("Conectado a Red Wifi");
  }
  else
  {
    Serial.println("Error al conectarse en la red");
    Serial2.setTimeout(2000);
  }
  //Deshabilitar las conexiones multiples
  Serial2.println("AT+CIPMUX=0");
}
```

Función EnvioDatos()

```
void EnvioDatos(String ID1, String ID2, int data1, int data2)
{
  Serial2.println("AT+CIPSTART=\"TCP\",\""+server+"\",80");
  if( Serial2.find("OK"))
  {
    Serial.println("ESP8266 conectado con el servidor...");
```

```
    String valor="[{\"variable\": \""+ID1+"\", \"value\":"+data1+"},
"+"{\"variable\":\""+ID2+"\", \"value\":"+data2+"}]";
    String URL="/api/v1.6/collections";
    String postRequest ="POST "+URL+Token+" HTTP/1.1\r\n"+"Host: things.ubidots
\r\n"+"\r\n"+"Content-Length: "+valor.length()+"\r\n\n"+valor+"\r\n" ;

    Serial2.print("AT+CIPSEND=");
    Serial2.println(postRequest.length());

    if(Serial2.find(">")) // ">" se puede enviar petición HTTP
    {
      Serial.println("Enviando HTTP . . .");
      Serial2.println(postRequest);
      if( Serial2.find("SEND OK"))
      {
        Serial.println("Peticion HTTP enviada:");
        Serial.println("Esperando respuesta...");
        boolean fin=false;
        long tiempo=millis();
        cadena="";
        delay(50);
        while(fin==false)
        {
           while(Serial2.available()>0)
           {
```

```
  char c=Serial2.read();
  cadena.concat(c);  //guardar la respuesta en el string
}

if(cadena.length()>100)
{
  Serial.println("La respuesta excede la longitud maxima");
  Serial2.println("AT+CIPCLOSE");
  if(Serial2.find("OK"))
  {
    Serial.println("Conexion finalizada");
  }
  fin =true;
}

if((millis()-tiempo)>250)
{
  Serial.println("Tiempo agotado");
  Serial2.println("AT+CIPCLOSE");
  if( Serial2.find("OK"))
  {
    Serial.println("Conexion finalizada");
  }
  fin =true;
```

```
          if(cadena.indexOf("CLOSED")>0)
          {
            Serial.println();
            Serial.println("Cadena recibida");
            fin=true;
          }
        }
      }
      else
      {
        Serial.println("Error en HTTP");
      }
    }
  }
  else
  {
    Serial.println("Conexion con el servidor fallida");
  }
}
```

Función RecepcionDato()

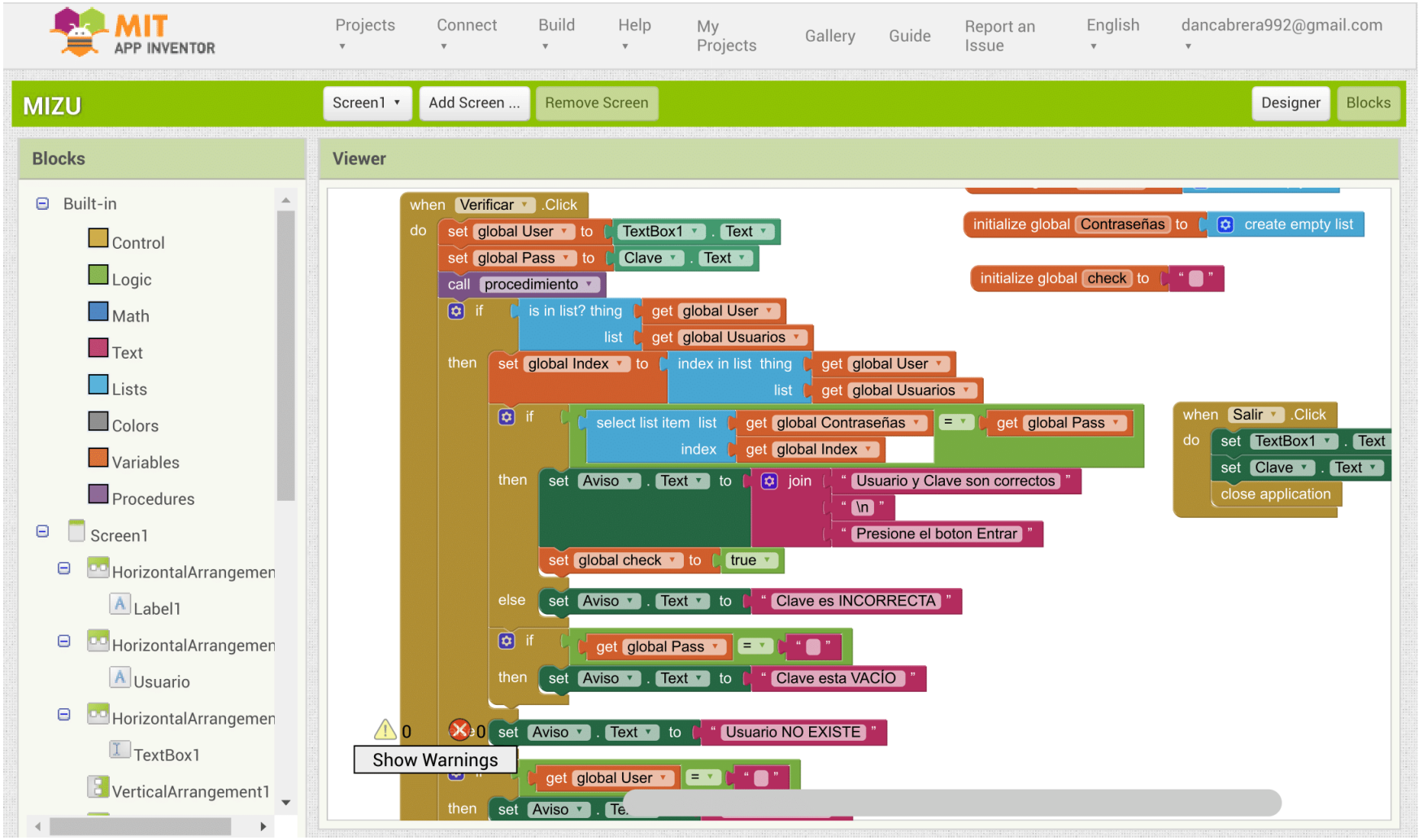
```
String RecepcionDato(String ID)
{
  String cadena="";

  Serial2.println("AT+CIPSTART=\"TCP\",\"" + server + "\",80");
```

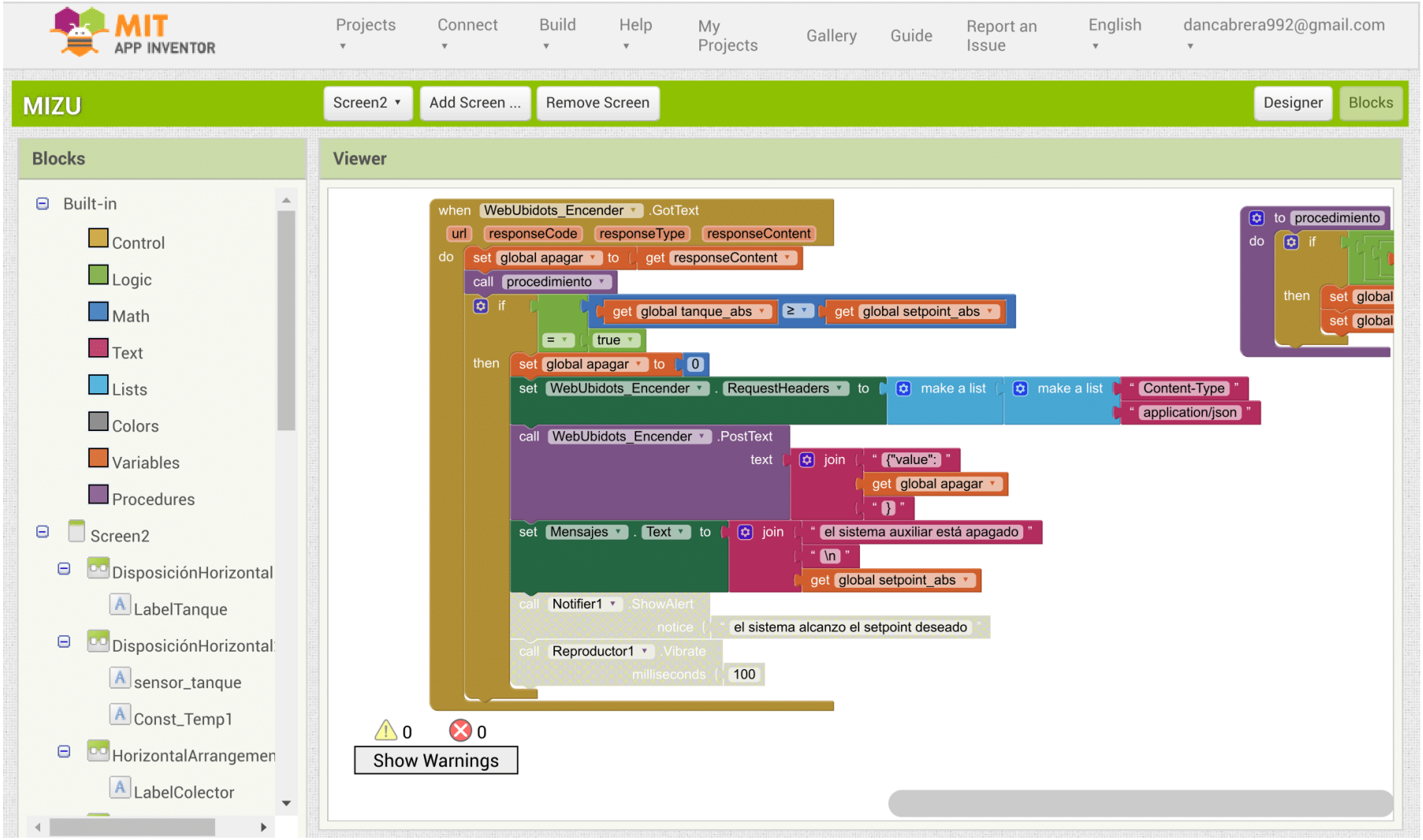
```
 if( Serial2.find("OK"))
 {
   Serial.println("ESP8266 conectado con el servidor...");

   //Armamos el encabezado de la peticion http

   String URL="/api/v1.6/devices/arduino-ethernet/";
   String getRequest="GET "+URL+ID+"/lv"+" HTTP/1.1\r\n"+"Host:
things.ubidots.com\r\n";
   getRequest=getRequest+"User-Agent: Arduino-Ethernet/2.0\r\n";
   getRequest=getRequest+"X-Auth-Token: A1E-
Hmk2EbtdckW7sVBiqkz4GbyhZ6DNn2\r\n";

   Serial2.print("AT+CIPSEND=");
   Serial2.println(getRequest.length());

   if(Serial2.find(">"))
   {
     Serial.println("Enviando HTTP . . .");
     Serial2.println(getRequest);
     if( Serial2.find("SEND OK"))
     {
       Serial.println("Peticion HTTP enviada:");
       Serial.println();
       //Serial.println(getRequest);
```

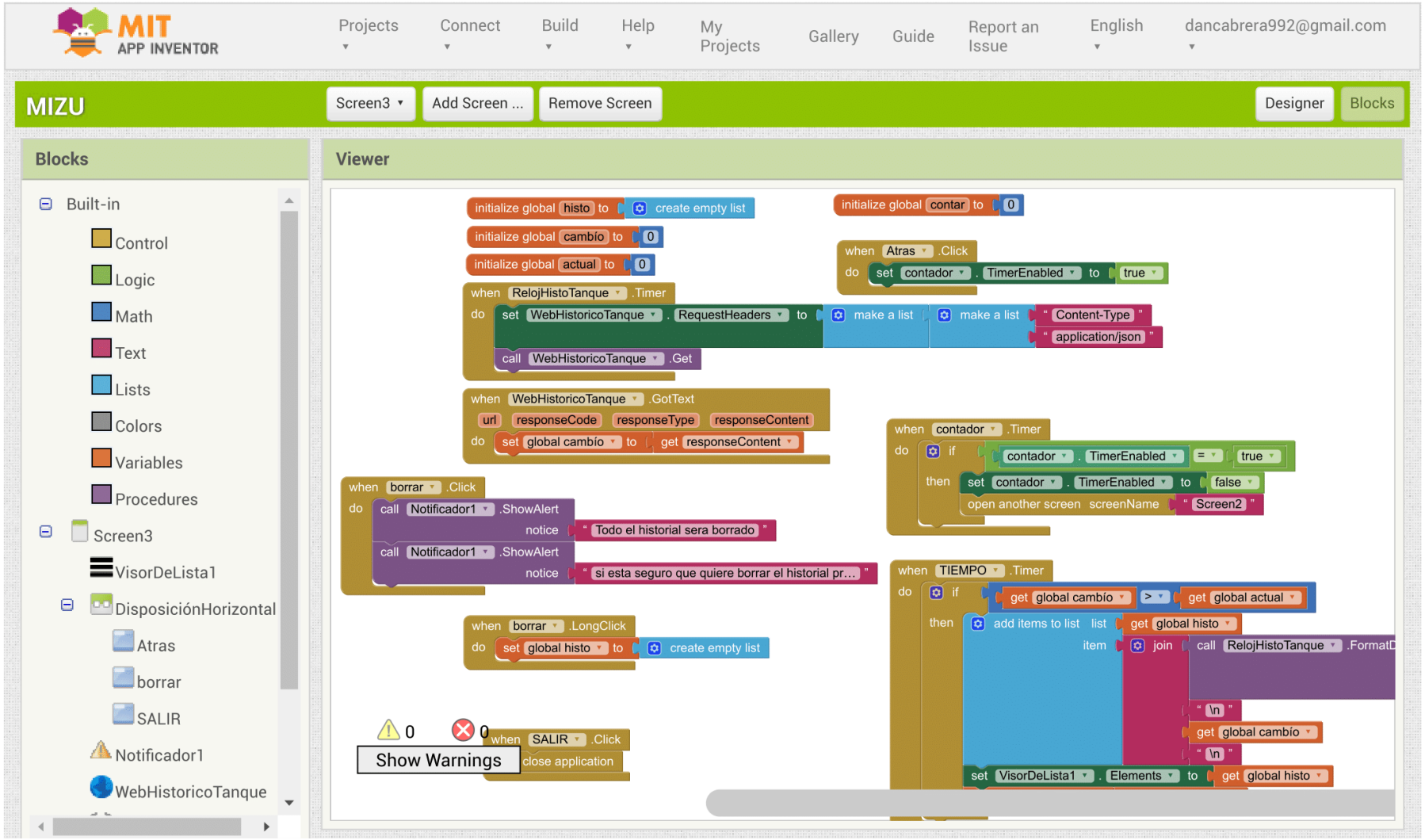
```
Serial.println("Esperando respuesta...")
boolean fin =false;
long tiempo =millis();
cadena="";
while(fin ==false)
{
   while(Serial2.available()>0)
   {
     char c=Serial2.read();
     cadena.concat(c);
   }
   if(cadena.length()>100)
   {
     Serial.println("La respuesta excede la longitud maxima ");
     Serial2.println("AT+CIPCLOSE");
     if(Serial2.find("OK"))
     {
       Serial.println("Conexion finalizada");
     }
     fin=true;
   }
   if((millis()-tiempo)>250)
   {
     Serial.println("Tiempo agotado");
     Serial2.println("AT+CIPCLOSE");
```

```
            if( Serial2.find("OK"))
            {
              Serial.println("Conexion finalizada");
            }
            fin=true;
          }
          if(cadena.indexOf("CLOSED")>0)
          {
            Serial.println();
            Serial.println("Cadena recibida");
            fin=true;
          }
        }
      }
      else
      {
        Serial.println("Error en HTTP");
      }
    }
  }
  else
  {
    Serial.println("Conexion con el servidor fallida");
  }
}
```

ANEXO 4. Programación de la aplicación móvil




MIT APP INVENTOR
Projects
Connect
Build
Help
My Projects
Gallery
Guide
Report an Issue
English
dancabrera992@gmail.com
MIZU
Screen2
Add Screen ...
Remove Screen
Designer
Blocks
Blocks
Built-in
Control
Logic
Math
Text
Lists
Colors
Variables
Procedures
Screen2
DisposiciónHorizontal
LabelTanque
DisposiciónHorizontal
sensor_tanque
Const_Temp1
HorizontalArrangemen
LabelColector
Viewer
when WebUbidots_Encender .GotText
url responseCode responseType responseContent
do set global apagar to get responseContent
call procedimiento
if get global tanque_abs ≥ get global setpoint_abs
= true
then set global apagar to 0
set WebUbidots_Encender . RequestHeaders to make a list make a list "Content-Type"
"application/json"
call WebUbidots_Encender .PostText
text join "{"value": "
get global apagar
"}"
set Mensajes . Text to join "el sistema auxiliar está apagado"
"\n"
get global setpoint_abs
call Notifier1 .ShowAlert
notice "el sistema alcanzo el setpoint deseado"
call Reproductor1 .Vibrate
milliseconds 100
to procedimiento
do if
then set global
set global
0
0
Show Warnings


MIT APP INVENTOR
Projects
Connect
Build
Help
My Projects
Gallery
Guide
Report an Issue
English
dancabrera992@gmail.com
MIZU
Screen3
Add Screen ...
Remove Screen
Designer
Blocks
Blocks
Viewer
Built-in
Control
Logic
Math
Text
Lists
Colors
Variables
Procedures
Screen3
VisorDeLista1
DisposiciónHorizontal
Atras
borrar
SALIR
Notificador1
WebHistoricoTanque
initialize global histo to create empty list
initialize global cambío to 0
initialize global actual to 0
initialize global contar to 0
when Atras .Click
do set contador . TimerEnabled to true
when RelojHistoTanque .Timer
do set WebHistoricoTanque . RequestHeaders to make a list make a list "Content-Type" "application/json"
call WebHistoricoTanque .Get
when WebHistoricoTanque .GotText
url responseCode responseType responseContent
do set global cambío to get responseContent
when contador .Timer
do if contador . TimerEnabled = true
then set contador . TimerEnabled to false
open another screen screenName "Screen2"
when borrar .Click
do call Notificador1 .ShowAlert notice "Todo el historial sera borrado"
call Notificador1 .ShowAlert notice "si esta seguro que quiere borrar el historial pr..."
when TIEMPO .Timer
do if get global cambío > get global actual
then add items to list list get global histo item join call RelojHistoTanque .FormatD
"\n"
get global cambío
"\n"
set VisorDeLista1 . Elements to get global histo
when borrar .LongClick
do set global histo to create empty list
0 0
Show Warnings
when SALIR .Click
close application

ANEXO 5. Datasheet SSR

Category		Plane installment type		3-phase AC solid state relay
Type		SSR	ZG3NC	ZG33
Outline and dimension ^mm∨		 60×45×24	 58×44×32	 31×16×15.5
Function		SSR Many kinds of input standards SSR With operation display		
Output	Insulation	Photoelectrical coupler		
	Load Voltage	30~240VAC/90~480VAC/12-65VDC		
	Load Current	10~40A	1~120A	
	Leakage current	· 100VAC 5mA · 200VAC 10mA		
	VoRmVceo(V)	1000		
	di/dt(A/us)	200		
	dv/dt(v/us)	400		
	$I^2t(A^2s)$	25-3000(Bigger current, bigger consumption)		
	Tj(℃)MAX	100		
Input	Input Voltage	3-32VDC/90-250VAC		
	Input Current	<40mA		
	Off	1VDC		
	On Voltage	3VDC		
	Off time	0.5ms DC/10msAC		
	On time	0.5ms DC/10msAC		
Dielectric strength		50Hz · 60S 2500VAC		
Ambient temperature		-30℃~80℃(Under non-ice/dewfall condition)		
Coefficient of safety of working current		Dissipative load is 60%/Inductive load is 40%		
Wiring installation diagram