ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN PROTOTIPO DE LAVAMANOS INTELIGENTE PORTATIL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES

ALEXANDER JAVIER CHUQUITARCO CHIDA

javier.chuquitarco@epn.edu.ec

DARIO FERNANDO TRUJILLO BALLAGÁN

dario.trujillo@epn.edu.ec

DIRECTOR: ING. LEANDRO PAZMIÑO ORTIZ, MSc.

leandro.pazmino@epn.edu.ec

CODIRECTORA: ING. MONICA VINUEZA RHOR, MSc.

monica.vinueza@epn.edu.ec

QUITO, ENERO 2021

DECLARACIÓN

Nosotros, Alexander Javier Chuquitarco Chida y Darío Fernando Trujillo Ballagán, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Sin perjuicio de los derechos reconocidos en el primer párrafo del artículo 114 del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación -COESC-, somos titulares de la obra en mención y otorgamos una licencia gratuita, intransferible y no exclusiva de uso con fines académicos a la Escuela Politécnica Nacional. Entregaremos toda la información técnica pertinente. En el caso de que hubiese una explotación comercial de la obra por parte de la EPN, se negociará los porcentajes de los beneficios conforme lo establece la normativa nacional vigente".

Alexander Javier Chuquitarco Chida
C.I 1725128381

Darío Fernando Trujillo Ballagán
C.I 1721531463

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por Alexander Javier Chuquitarco Chida y Darío Fernando Trujillo Ballagán bajo nuestra supervisión.

Ing. Leandro Pazmiño Ortiz, MSc.
DIRECTORA DEL PROYECTO

Ing. Mónica Vinueza Rhor, MSc.
CODIRECTOR DEL PROYECTO

DEDICATORIA

Este documento está dedicado principalmente a nuestros padres ya que sin su apoyo incondicional día a día no hubiéramos logrado alcanzar este gran objetivo, también a toda nuestra familia por sus alientos y consejos para seguir en la lucha y alcanzar nuestros sueños y finalmente a nuestros compañeros de aula por el apoyo incondicional y por compartir nuestros conocimientos todos los días en las aulas.

Muchas gracias.

AGRADECIMIENTO

Un agradecimiento a todos nuestros mentores por las enseñanzas impartidas en todo este ciclo universitario. También damos gracias a muebles metálicos Cardel por el apoyo brindado en nuestro proyecto de titulación y, finalmente un agradecimiento a nuestro tutor por el apoyo brindado para la elaboración de este proyecto.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En este documento se desarrolla la teoría acerca del diseño, implementación y pruebas de funcionamiento del prototipo de lavamanos inteligente portátil.

En la primera sección del documento se plantea la introducción, en la cual se justifica la creación del proyecto y se señala aspectos generales del prototipo. Posteriormente, se aborda algunos conceptos teóricos relacionados con sensores, microcontroladores, servidores, entre otros; que sustentan el contenido desarrollado en este trabajo.

En la segunda sección trata de la metodología, en la cual se explica el proceso realizado para la elaboración del proyecto desde el diseño esquemático hasta su implementación, además se indica los elementos que lo conforman.

La tercera sección se refiere a resultados y discusión, donde se desarrolla: los requerimientos del sistema, *hardware* y *software* necesarios para la implementación del prototipo, diseño del sistema de automatización, diseño del circuito impreso, la estructura del prototipo y su respectiva implementación. Además, se detalla las pruebas de funcionamiento y el costo total del proyecto.

Finalmente van las conclusiones y recomendaciones, donde se recogen las experiencias adquiridas durante la elaboración del proyecto. Adicionalmente se presenta las referencias bibliográficas, que respaldan el marco teórico del proyecto y también se anexa la documentación necesaria como: diagramas de flujo, manuales de mantenimiento, código fuente, entre otros; con el fin de facilitar el uso y el manejo del prototipo.

Palabras clave: Portátil, sensor, automatización, implementación, lavamanos.

ABSTRACT

This document develops the theory about the design, implementation and functional tests of the portable smart hand basin prototype.

The first section of the document presents the introduction, in which the creation of the project is justified and general aspects of the prototype are indicated. Subsequently, some theoretical concepts related to sensors, microcontrollers, servers, among others, are addressed; that support the content developed in this work.

The second section deals with the methodology, which explains the process carried out for the elaboration of the project from the schematic design to its implementation, in addition to indicating the elements that comprise it.

The third section refers to results and discussion, where it is developed: the requirements of the system, hardware and software necessary for the implementation of the prototype, design of the automation system, design of the printed circuit, the structure of the prototype and its respective implementation. Finally, the functional tests and the total cost of the project are detailed.

Finally there are the conclusions and recommendations, where the experiences acquired during the preparation of the project are collected. Additionally, the bibliographic references are presented, which support the theoretical framework of the project and the necessary documentation is also attached, such as: flow diagrams, maintenance manuals, source code, among others; in order to facilitate the use and handling of the prototype.

Keywords: *Hand-held, sensor, automation, implementation, handwash.*

1. INTRODUCCIÓN

Ante la emergencia sanitaria generada a nivel mundial por el COVID-19 la OMS (Organización Mundial de la Salud), propone algunas normas de aseo personal, especialmente enfocándose en el lavado de manos para evitar contagios.

Un lavado de manos regular y bien realizado requiere de veinte segundos aproximadamente en donde el proceso es el siguiente: abrir la llave de agua, humedecer las manos, luego colocar jabón y frotar las manos entre ellas, para finalmente enjuagar con más agua. En este proceso la persona tiene contacto por lo menos dos veces con objetos necesarios para el lavado de manos como son la llave de grifo y el jabón o también un dispensador de jabón. Por estos motivos se realizó este proyecto basándose principalmente en dos aspectos importantes por la OMS, los cuales son: el lavado constante de manos y evitar el contacto con cualquier tipo de objeto.

Dado que en Ecuador no se ha planteado un prototipo similar, ni tampoco se ha realizado ningún estudio referente al tema, se espera que el prototipo tenga acogida tanto para las oficinas, supermercados y lugares de gran movilidad de personas, como también principalmente para los establecimientos educativos enfocándose en los establecimientos de primaria, para generar una cultura de buena higiene en las manos a los niños.

Para tener una idea clara del diseño del prototipo es necesario establecer las necesidades y requerimientos del sistema. Una vez identificados los requerimientos, se procede a determinar los componentes tanto para la parte del *software* y *hardware*.

Una vez determinado el *hardware* y software que cumpla con los requerimientos del proyecto, se procede a diseñar el prototipo. Posteriormente, se realiza la implementación y sus respectivas pruebas de funcionamiento, esto con el fin de garantizar su correcto funcionamiento y durabilidad.

1.1 Marco teórico

- ***Hardware***

  - Módulo RFID-RC522 RF

La tecnología RFID se basa en la lectura de *tags* de manera inalámbrica, los módulos RFID presentan compatibilidad de trabajar con microcontroladores como Arduino. El lector RFID RC522 es un lector – grabador de 13.56 MHz, la comunicación se realiza mediante el estándar SPI. Trabaja mediante un sistema de modulación y demodulación de cualquier componente pasivo RFID de 13.65 MHz. La distancia de lectura que presenta el lector es de alrededor 5-7cm, el módulo se observa en la Figura 1.1 [1] .

Los *tags* (ver Figura 1.1), están conformadas por 64 bloques de memoria, los cuales se emplean para la lectura o escritura, cada bloque almacena hasta 16 *bytes*. Además, el número de cada *tag* consiste en 5 números hexadecimales. No requiere de una fuente energía, presenta un mecanismo de control de cifrado y un circuito de comunicación. El periodo para la retención de datos que presentan los *tags* es de alrededor de 10 años, además la lectura de datos no tiene límites [1].

El lector funciona como un emisor-receptor, el cual envía una señal para comenzar la comunicación con las *tags*. La señal es receptada por los *tags* para posteriormente transmitir su código hexadecimal, finalmente esta información es captada y decodificada por el lector RFID [1].

Figura 1.1: Módulo RFID-RC522 RF [1]

A continuación, se presenta una tabla comparativa (Tabla 1.1) entre las tecnologías de auto identificación RFID y código de barras.

Tabla 1.1: Comparativa ente tecnologías RFID y código de barras [2]

Características	Código de barras	RFID
Precisión	Requiere intervención humana.	100% automático.
Modelo de lectura	Superficie.	A través de los materiales.
Flexibilidad	Para la lectura se necesita línea de visión.	No necesita línea de visión para lectura.
Capacidad	Espacio limitado.	Mayor capacidad de almacenamiento.
Lectura	Una lectura a la vez.	Lectura / escritura simultánea.

- Módulo relé de dos canales

Este módulo está formado por 2 canales, es ideal para trabajar con microcontroladores, ya que se programa de manera sencilla, sin utilizar librerías para su control, sólo necesita un voltaje de salida (TTL - 5 V) para controlar su funcionamiento. Presenta un *led* el cual indica el estado de cada canal [3].

Para activar su salida normalmente abierta se envía un "0" lógico y para desactivar la salida un "1" lógico. Para trabajar con las placas de Arduino, se recomienda utilizar la función “millis”, ya que de esta manera el sistema continúa trabajando mientras se activa/desactiva el relé. Dicho módulo se visualiza en la Figura 1.2 [3].

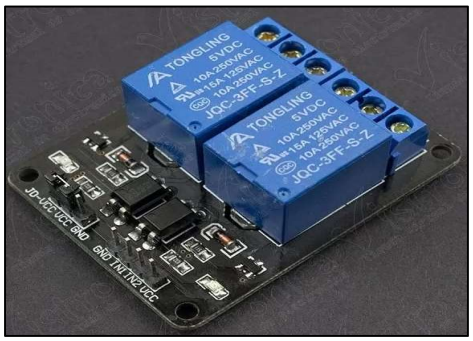

Figura 1.2: Módulo relé de dos canales [3]

- *Buzzer* activo

El *buzzer* activo (zumbador), es un dispositivo que emite un sonido de una frecuencia fija, cuando se encuentra energizado. Para producir el sonido, utiliza un zumbador piezoeléctrico activo, el cual emite un sonido de alrededor de 2.5 KHz E esto durante la señal de control recibida se encuentra en alto. Este módulo consta de la electrónica necesaria, por tanto, resulta sencillo de conectar y controlar. Adicionalmente, para el microcontrolador no supone una carga significativa, ya que el mismo emite una onda eléctrica que se convertirá en sonido. El *buzzer* se muestra en la figura 1.3 [4].

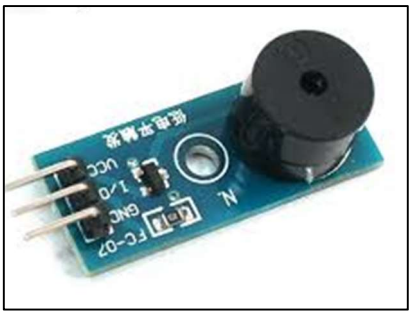

Figura 1.3: *Buzzer* activo [4]

- Bomba de diafragma R385

Bomba que presenta desplazamiento positivo, es decir el empuje de las paredes elásticas ya sean de membranas o diafragmas, es provocado mediante el aumento de presión que varía el volumen de la cámara, ya sea aumentándolo o disminuyéndolo. Además, posee válvulas de retención, las cuales permiten el bombeo del líquido [5].

Este tipo bomba de diafragma son usadas en dispensadores de agua como teteras, hervidores eléctricos, etc. Esta versión ha sido mejorada para un mayor rendimiento, en comparación con versiones anteriores, esta presenta un soporte fijo, aumentando la vida útil en gran medida. Dicha bomba se visualiza en la Figura 1.4 [5].

Figura 1.4: Bomba de agua R385 [5]

- **Pantalla *Nextion* HMI 3.5” NX4832T035**

La pantalla inteligente *Nextion* NX4832T035 ver Figura 1.5, presenta un interfaz de control amigable con el usuario. La pantalla es de la serie de placas de TFT y el *software* pertenece a la empresa *Nextion*, para comunicarse con el microcontrolador utiliza un puerto serie. Es fácil de incorporar a cualquier proyecto, ya que utiliza las librerías oficiales [6].

Figura 1.5: Pantalla *Nextion* 3.5" [6]

- Sensor Infrarrojo

El sensor infrarrojo (Figura 1.6), están compuestos por un *LED* infrarrojo y un fototransistor. El *LED* funciona como emisor y el fototransistor como receptor. Este sensor cuenta con 3 pines de conexión, los pines 5V y GND se utilizan para alimentación y el pin *OUT* para enviar la señal a la placa Arduino. Este último se utiliza para indicar si está llegando la señal del *LED* al fototransistor [7].

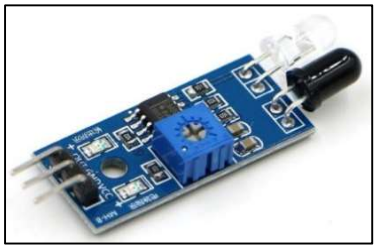

Figura 1.6: Sensor infrarrojo [7]

- Sensor Ultrasónico HC-SR04

El módulo HC-SR04 (Figura 1.7), realiza la medición de proximidad por ultrasonidos, detecta objetos en un rango de 2-450 cm. El sensor realiza la medición mediante el envío y recepción de ultrasonidos, la medición se basa en enviar un pulso de arranque y cuantificar la anchura del pulso de retorno. Este módulo se caracteriza por su bajo consumo de energía y gran precisión [8].

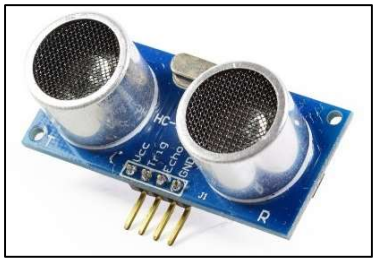

Figura 1.7: Ultrasónico HC-SR04 [8]

- Sensor MLX90614

El sensor MLX90614 ver Figura 1.8, se caracteriza por realizar mediciones de temperatura a objetos sin contacto. El sensor es sensible a la radiación infrarroja emitida por un objeto a distancia, además presenta un rango de trabajo que va desde los - 70°C - 380°C, con una precisión de 0.5°C [9].

La salida del sensor es lineal, en la cual se compensa las variaciones de la temperatura ambiente. Permite una fácil conexión con la placa Arduino, además incluye un amplificador de bajo ruido. Está conformado con un *chip* de silicio con una fina membrana micromecanizada la cual es sensible a la radiación infrarroja, además cuenta con los elementos necesarios para amplificar, digitalizar la señal y calcular la temperatura [9].

Su funcionamiento reside en la ley de *Stefan-Boltzmann*, la cual indica que "Todo objeto por encima del cero absoluto (ºK) emite radiación cuyo espectro es proporcional a su temperatura". El sensor realiza la lectura de esta radiación, obteniendo a su salida una señal eléctrica la cual es proporcional a la temperatura de todos los objetos en su rango de visión [9].

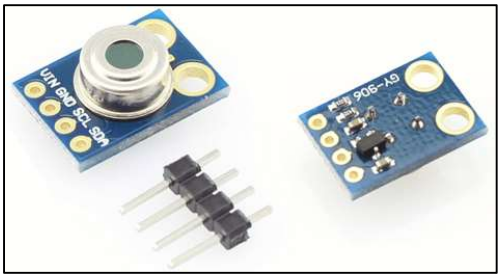

Figura 1.8: Sensor MLX90614 *[9]*

- **Plataformas de desarrollo**

  - Arduino

Se trata de una plataforma de código abierto, la cual se controla mediante un conjunto de instrucciones a su microcontrolador, para hacerlo, se utiliza el lenguaje de programación Arduino y el *Software* Arduino (IDE) [10].

  - Placa Arduino MEGA 2560

El Arduino Mega se trata de un microcontrolador más completo del ecosistema Arduino, ya que presenta 16 entradas análogas, un cristal oscilador de 16 MHz, un botón de *reset*, una

entrada para la alimentación de la placa y cuenta con 54 pines digitales las cuales trabajan como entradas/salidas [11].

Cuando se alimenta la placa con una fuente externa, es necesario el uso de un convertidor AC/DC y regular la alimentación en el rango permitido para esta placa. De preferencia, el voltaje suministrado de estar alrededor de los 7V - 12V [11].

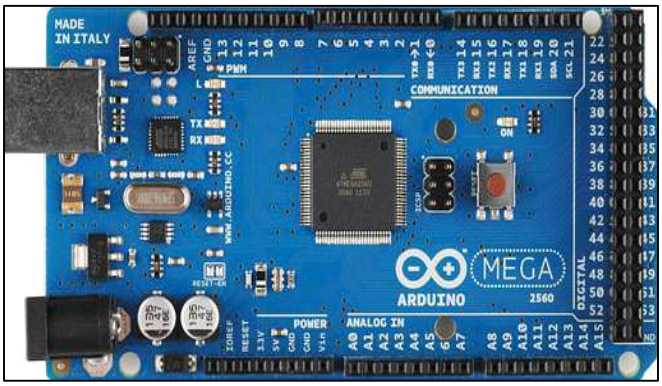

Figura 1.9: Placa Arduino MEGA 2560 [11]

- Módulo *NodeMcu* V3 ESP8266 *Wi-Fi*.

El módulo *Wi-Fi* ESP8266 tiene integrado el protocolo TCP/IP y es capaz de dar acceso *Wi-Fi* a cualquier microcontrolador. Se encuentra preprogramado con un *firmware* AT, por tanto, se conecta a un microcontrolador para tener las mismas capacidades *Wi-Fi* que con un *shield* [12].

El módulo *NodeMCU* ver Figura 1.10, se basa en el chip ESP8266. Presenta algunas modificaciones en comparación de versiones anteriores como la integración de un puerto USB, la incorporación de un regulador de tensión integrado y se puede programar mediante el IDE de Arduino. El módulo cuenta con un pin de salida de 5V, ya que utiliza el conversor CH340 [12].

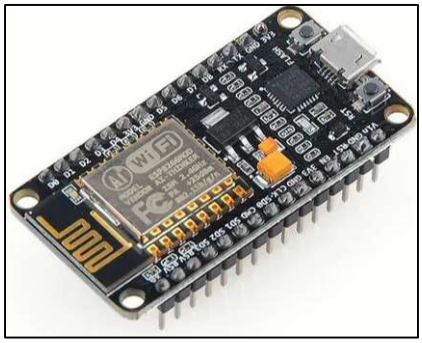

Figura 1.10: Módulo *NodeMcu* V3 ESP8266 *Wi-Fi* [12]

- *Raspberry* Pi

Raspberry Pi se trata de una computadora de bajo costo, permite a las personas conocer la informática ya que mediante los lenguajes como *Scratch* y *Python* se aprende a programar. Comparada con una computadora de escritorio esta placa puede realizar las mismas tareas, desde navegar por Internet y reproducir videos de alta definición, hasta crear hojas de cálculo, etc [13].

A continuación, se presenta una tabla comparativa (Tabla 1.2) entre los ecosistemas más usados actualmente en el medio electrónico: Arduino y *Raspberry* Pi [13].

Tabla 1.2: Comparación *Raspberry* Pi Vs Arduino [13].

Arduino	***Raspberry* Pi**
Ejecuta un programa una y otra vez.	Ejecuta múltiples programas al mismo tiempo.
De manera integrada funcionan sus componentes y sensores.	Para utilizar sensores y otros componentes necesita instalar librerías y *software*.
Más económico en comparación con *Raspberry* Pi.	Menos económico en relación con Arduino.
Está pensado para funcionar con batería.	Complicado hacerlo funcionar con batería.
Para conectarse a internet requiere *hardware* externo.	Con su puerto RJ-45 se conecta a Internet.
Presenta un USB tipo B hembra para conectarlo a un PC	Presenta 4 puertos USB para conectarlo a distintos dispositivos.
Utiliza un procesador de familia AVR.	Utiliza procesadores ARM.
Es un dispositivo *plug* and *play.*	Para evitar la corrupción de archivos, se debe apagarlo adecuadamente.
Utiliza como lenguaje de programación C/C++, Arduino IDE.	*Python* es su lenguaje de programación.

- ***Software***

  - IDE Arduino

Se trata de un *software* el cual cuenta con un depurador, compilador y una interfaz gráfica (GUI). El entorno IDE contiene principalmente dos partes básicas: el editor y el compilador, donde el primero se usa para escribir el código requerido y el segundo para compilar y cargar el código en el módulo Arduino. Además, este entorno es compatible con los lenguajes C y C ++ [14].

- *Proteus*

Proteus es una herramienta de *software* de simulación y diseño desarrollada por *Labcenter Electronics* para sistemas eléctricos y diseño de circuitos electrónicos. Se usa para la creación de proyectos electrónicos en todas sus etapas desde el diseño del esquema electrónico hasta la construcción. Dentro de sus principales aplicaciones tenemos las siguientes [15]:

- **ISIS** se utiliza para diseñar esquemas y simular los circuitos en tiempo real [15].
- **ARES** se utiliza para el diseño de PCB. Presenta la función vista 3D, el cual no permite visualizar el resultado del PCB diseñado junto con los componentes [15].

- *Nextion Editor*

Nextion Editor, es un *software* de desarrollo que se utiliza para la construcción visual de la interfaz gráfica de usuario para dispositivos integrados con uso intensivo de GUI (*Graphical User Interface*) con varios modelos de pantallas TFT y paneles táctiles [16].

- AutoCAD

Se trata de un *software* para realizar modelos o construcciones en 3D. El programa permite manipular las propiedades de los objetos como la ubicación, tamaño y la forma. Además, permite colocar o agrupar los mismos en capas [17].

- Plataforma IFTTT (*IF This Then That*)

IFTTT se trata de una plataforma de *software* que conecta aplicaciones, dispositivos y servicios de diferentes desarrolladores para activar una o más automatizaciones que involucran aplicaciones, dispositivos y servicios [18].

Las automatizaciones se logran a través de *applets*, que son macros que conectan múltiples aplicaciones para ejecutar tareas automatizadas. Se puede activar o desactivar un subprograma utilizando el sitio *web* de IFTTT o las aplicaciones móviles. También, se pueden crear subprogramas o hacer variaciones de los existentes a través de la interfaz sencilla y fácil de usar de IFTTT. Los subprogramas FTTT pueden usar un filtrado avanzado y otras herramientas para crear nuevas interacciones, ya que la plataforma tiene soporte para *JavaScript* [18].

- *Inkscape*

Inkscape, es un potente y gratuito *software* de dibujo vectorial para *Windows* y Mac que permite crear gráficos de forma similar a Adobe Illustrator. Se trata de un *software* de código abierto el cual se utiliza para diseñar imágenes SVG (*Scalable Vector Graphics).* SVG es un estándar basado en XML para describir un dibujo. La característica principal de este estándar es la libertad de escala, es decir, se puede acercar o alejar a una escala arbitraria sin perder detalles o crear perfiles irregulares [19].

Las principales características de este *software* son la creación y alteración de rutas, estilos de trazos, degradados, compatibilidad con archivos Adobe AI y archivos de gráficos vectoriales SVG. Además, *Inkscape* permite compatibilidad con archivos de imagen estándar, incluidos JPG y PNG para ortoscopia [19].

- **Redes**

Una red es un sistema de comunicación que interconecta varios dispositivos informáticos autónomos mediante enlaces, sean estos alámbricos o inalámbricos; permite compartir información, dispositivos, recursos y servicios. Se clasifican según su extensión en: redes *LAN, MAN, PAN, WAN* e Internet [20].

- Redes *WLAN*

Significa "Red de área local inalámbrica" ver Figura 1.11, la red permite que los dispositivos se comuniquen de forma inalámbrica. Los dispositivos de una *WLAN* se comunican a través de *Wi-Fi*, en contraparte, con una red cableada tradicional, en la cual se comunican a través de cables *Ethernet* [20].

Si bien una *WLAN* puede verse diferente a una *LAN* tradicional, funciona de la misma manera. Por lo general, los dispositivos nuevos se agregan y configuran mediante *DHCP*. Pueden comunicarse con otros dispositivos en la red de la misma manera que lo harían en una red cableada. La principal diferencia es cómo se transmiten los datos, en una *LAN*, los datos se transmiten a través de cables físicos en una serie de paquetes *Ethernet* que contienen; mientras que en una *WLAN* los datos se transmiten por aire utilizando uno de los protocolos IEEE 802.11 [20].

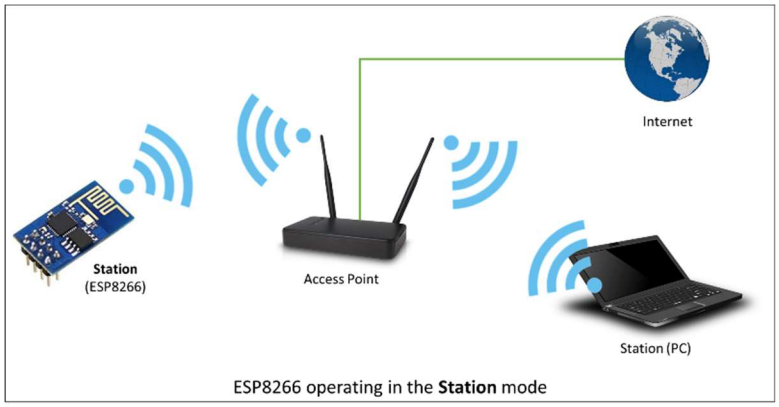

Figura 1.11: Redes *WLAN* [20]

- Estándar IEEE 802.11

El estándar se desarrolló para redes *WLAN*, lo denominaron 802.11, el cual presentaba una velocidad de transmisión de 2 Mbps como máximo, demasiado lento para la mayoría de las aplicaciones [21].

El estándar IEEE 802.11 está constituido por dos capas, la capa MAC (*Media Access Control*) y la capa física. Permite que distintos medios de transmisión puedan trabajar con un solo protocolo [21].

Como el proyecto necesita de una conexión *WLAN*, los estándares que soporta la placa *NodeMcu* son los siguientes [21]:

- **IEEE 802.11b:** Trabaja en la banda de 2.4 GHz, con una velocidad de transmisión de11 Mbps [21].
- **IEEE 802.11g:** Presenta velocidad de transmisión alrededor de los 54 Mbps. Trabaja en la banda de 2.4 GHz [21].
- **IEE 802.11n:** Presenta velocidades de transmisión de 600 Mbps, además trabaja en las bandas de 2.4 GHz y 5 GHz [21].

- Protocolo de Transferencia de Hipertexto (*HTTP)*.

Se trata de un protocoló a nivel de la capa aplicación. Básicamente, *HTTP* se trata de un protocolo basado en *TCP / IP*, dicho protocolo se emplea para la entrega de datos en la *World Wide Web*. El puerto utilizado de manera predeterminada es el TCP 80, el protocolo especifica cómo se construyen y envían los datos de solicitud de los clientes al servidor, y cómo los servidores responden a estas solicitudes. Básicamente, *HTTP* trabaja entre un cliente y un servidor como un protocolo de solicitud-respuesta (*request-response*) [22].

GET y *POST* son dos tipos diferentes de solicitudes *HTTP*. *GET* se usa para ver algo, sin cambiarlo, mientras que *POST* se usa para cambiar algo. Básicamente, *GET* se usa para recuperar datos remotos y *POST* se usa para insertar / actualizar datos remotos [22].

- ***GET***: obtiene información del servidor para enviarlas al cliente (una página WEB almacenada, un archivo, información de bases de datos, etc.) [22].
- ***POST*:** envía información desde el cliente al servidor para que la procese, agregue o actualice [22].

En lo relativo a los datos, como, por ejemplo, al rellenar formularios con nombres de usuario y contraseñas, el método *POST* ofrece mucha discreción. Los datos no se muestran en el caché ni tampoco en el historial de navegación. La flexibilidad del método *POST* también resulta muy útil, no solo se pueden enviar textos cortos, sino también otros tipos de información, como fotos o vídeos [22].

La mayor desventaja del método *GET* es su débil protección de los datos. Los parámetros *URL* que se envían quedan visibles en la barra de direcciones y son accesibles sin clave en el historial de navegación, en el *caché* y en el *log* de los servidores [22].

2. METODOLOGÍA

Para tener una idea clara del diseño del prototipo implementado, fue necesario establecer las necesidades y requerimientos de las personas. La elaboración del proyecto consistió en varias etapas: análisis, diseño, programación, implementación y verificación de funcionamiento.

Una vez identificados los requerimientos, se procedió a determinar los componentes tanto para la parte del *software* y *hardware*. En este caso se usó Arduino por la simplicidad, costos y su capacidad de incluir módulos externos, esto hace que Arduino sea capaz de trabajar con cualquier tipo de sensor, siendo la mejor elección para el proyecto.

Una vez identificado el *hardware* para el desarrollo del prototipo, se procedió a la elección del *software*, como se optó por el ecosistema de Arduino la interfaz de programación utilizada es Arduino IDE.

El control del dispensador de sustancias se realizó mediante una bomba de agua, esto mediante el encendido y apagado de relés. También para que las personas realicen un correcto lavado de manos se optó por el uso de una pantalla TFT. Además, para realizar la mediación de la temperatura de las personas se empleó un sensor de infrarrojo, ya que estos permiten realizarlo a distancia. En el caso de que la persona presente una temperatura mayor a 37 grados centígrados se visualiza una notificación en la pantalla TFT.

Para la creación del diseño esquemático del prototipo, una vez determinado el *hardware* para que cumpla con los requerimientos del proyecto, se creó el circuito esquemático del prototipo. Para la parte del *software*, se desarrolló el código fuente para la programación del sistema del microcontrolador.

Una vez finalizado el diseño del prototipo, se imprimió y transfirió a la baquelita de cobre, tras lo cual se plancho, quemo y perforó la baquelita con el fin de obtener los recorridos y puntos donde se colocaron y soldaron cada uno de los componentes del sistema. Finalmente, el código fuente del programa se grabó en el microcontrolador.

Una vez instalados los elementos en el circuito impreso, se procedió a verificar fallas y errores, ya sea provocados al momento del armado o de algún daño físico que haya sufrido algún dispositivo. Después, se procedió a instalar el sistema en el lavamanos con las protecciones necesarias para el dispositivo, esto tomó tiempo ya que se tuvo que fabricar un lavamanos que cumpla con estándares de calidad y que se adapte a los dispositivos electrónicos.

Finalmente se realizó pruebas del prototipo para la comprobación de su funcionamiento y se verificó detalladamente que el dispositivo cumpla con todo lo estipulado anteriormente. Finalmente, se procedió a realizar la corrección de errores y ajustes necesarios de tal manera que el dispositivo sea lo más efectivo posible y de esta forma se garantizó el funcionamiento adecuado del prototipo.

3. RESULTADOS Y DISCUSIÓN

3.1 Requerimientos del sistema

Debido a la pandemia por el Covid-19 el prototipo requiere brindar propuestas a las personas para que su uso no tenga ningún tipo de dificultad. Por tanto, se realizó un análisis completo sobre las necesidades del proyecto, esto con la finalidad de determinar la construcción y elección de los componentes que conformarán el prototipo.

Para este análisis, en base a la observación del proceso de lavado de manos se identificaron los siguientes requerimientos:

- Control y registro de usuarios.
- Detección de las manos de los usuarios.
- Control de las bombas de agua.
- Medición de la temperatura corporal de los usuarios.
- Envió de mensaje de correo electrónico.

- Interfaz de usuario.
- Automatización de procesos.
- Alimentación del prototipo.
- Diseño del circuito impreso.
- Diseño de la estructura del prototipo.

- **Control y registro de usuarios**

El prototipo debe presentar un sistema de control y registro de usuarios de forma inalámbrica al momento de iniciar el proceso de lavado de manos, esto con la finalidad que en el caso de que algún usuario llegara a presentar fiebre sea sencillo de identificarlo.

- **Detección de las manos de los usuarios**

El prototipo debe tener un sistema que permita la detección de las manos del usuario a una distancia entre 15-20 cm, esto con la finalidad de minimizar el contacto de los usuarios con la superficie del prototipo. Adicionalmente, dicho sistema debe ser gestionado mediante un microcontrolador, todo esto con la finalidad de tener un control del sistema de manera automatizada.

- **Control de las bombas de agua**

Para realizar el control tanto del agua como del jabón antibacterial es necesario el uso de dos bombas de agua, además el prototipo debe tener un componente electrónico que permita el control de dichas bombas desde un microcontrolador, esto con la finalidad de automatizar dicha tarea.

- **Medición de la temperatura corporal**

Como una funcionalidad extra del proyecto es la medición sin contacto de la temperatura corporal del usuario. Es necesario la utilización de un sensor que permita medir la temperatura corporal del usuario a una distancia adecuada, adicionalmente el rango de error que debe tener dicho sensor no debe ser mayor al 1 °C, esto con la finalidad que dicha medición sea lo más fiable posible.

- **Sistema para el envío de notificaciones**

El envió de una alarma en forma de correo electrónico es necesario para el proyecto, esto con la finalidad de notificar en el caso de que algún usuario llegara a presentar una temperatura corporal mayor a 37 °C. Además, se tiene que tener en cuenta el tipo de servidor que se va a utilizar en el prototipo y los componentes que permitan realizar dicha tarea.

- **Interfaz de usuario**

Para que el usuario pueda interactuar y utilizar el prototipo de manera sencilla, el proyecto debe contar con una interfaz gráfica. La misma debe ser controlada desde un microcontrolador y el diseño de la interfaz gráfica tiene que ir acorde a los distintos procesos que debe realizar el proyecto.

- **Automatización de procesos**

El proyecto debe tener un microcontrolador que sea capaz de gestionar y controlar los distintos sensores y componentes electrónicos que va a tener el prototipo. Adicionalmente, se debe solventar el acceso a internet y el manejo de una interfaz gráfica.

- **Alimentación del prototipo**

Se tiene que tener en cuenta los distintos niveles de voltaje requeridos por los componentes del proyecto, por lo que es necesario el uso de una fuente de energía que sea capaz de cumplir con dichos requerimientos de energía.

- **Diseño del circuito impreso**

Una vez determinado el *hardware* y *software* que cumpla con los requerimientos del proyecto, es necesario crear el circuito esquemático del prototipo, para posteriormente proceder a imprimir el diseño y trasladarlo a una placa de cobre, posterior a esto se debe realizar el planchado, quemado y perforado de la placa de cobre para obtener las rutas y puntos en los cuales se colocarán cada uno de los elementos.

- **Estructura del proyecto**

Para que los elementos del prototipo se mantengan funcionando correctamente se necesita un armazón, esto con la finalidad de brindar protección a los distintos componentes electrónicos y además que facilite el uso del prototipo para los usuarios. Por tanto, es necesario el uso de un *software* el cual permita el diseño digital de la estructura con el fin de cumplir este requerimiento.

3.2 *Hardware* y *software* necesarios para implementar el prototipo.

Culminando el proceso de análisis de los requerimientos del proyecto, se procedió a la elección de los componentes necesarios para la construcción del prototipo.

- **Control y registro de usuarios**

El proyecto debe contar con un sistema de registro de usuarios al momento de iniciar el proceso de lavado de manos. Por tanto, se realizó una comparativa, revisar Tabla 1.1, entre las tecnologías de código de barras y RFID [2].

Teniendo en cuenta las características de cada tecnología, se optó por el uso de la tecnología RFID, debido a su facilidad de uso y funcionalidades cumpliendo con lo requerido para el prototipo. Específicamente para la identificación de cada usuario se seleccionó los *tags* pasivos y para realizar la lectura de dichos *tags* el módulo RC522.

Se selección la etiqueta o *tag* pasiva RFID (ver Figura 3.1), ya que la distancia de lectura y escritura de los *tags* es de alrededor 5-7 cm, además los *tags* cuentan con 64 bloques de memoria (0-63), los cuales son usados para lectura o escritura. Las principales características se describen en la Tabla 3.1, teniendo en cuenta dichas características se seleccionó los *tags* RFID ya que cumplen con los requerimientos del proyecto [1].

Tabla 3.1: Características de la Etiqueta o *tag* RFID 13.56MHz [1].

CARACTERISTICAS	DESCRIPCIÓN
Chip	Philips Mifare 1k S50
Capacidad	8Kbit EEPROM
Número de serie	32 bits
Temperatura de operación	-20 ℃ ~ 50 ℃
Frecuencia de operación	13.56MHz

Figura 3.1: *Tag* RFID 13.56 MHz [1]

El módulo RC522 ver la Figura 3.2, es un lector – grabador RFID de 13.56 MHz, se seleccionó dicho módulo, ya que permite trabajar de manera sencilla con la plataforma de Arduino, además presenta un mecanismo de modulación y demodulación para cualquier componente pasivo RFID de 13.65 MHz. Las principales características del módulo se las describe en la Tabla 3.2 [1].

Tabla 3.2: Características Módulo RFID RC522 [1].

CARACTERISTICAS	DESCRIPCIÓN
Frecuencia de trabajo	13.56 MHz
Velocidad de transmisión	10 Mbps
Consumo de energía	13-26 mA a 3.3 V
Distancia de lectura	0 a 6 cm
Máxima velocidad de SPI	10 Mbps
Estándar de comunicación	SPI
Temperatura de trabajo	-20 a 80ºC

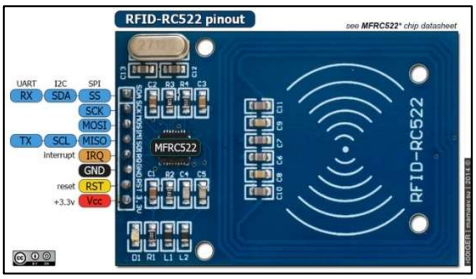

Figura 3.2: Módulo RFID RC522 [1]

- **Detección de las manos de los usuarios**

Para poder realizar la detección de las manos del usuario al iniciar el proceso de lavado, se procedió a utilizar el sensor ultrasónico HC-SR04 ver Figura 3.3, porque presenta un rango de detección alrededor de 2-450 cm, cumpliendo con lo requerido para el prototipo. El sensor realiza la medición mediante el envío y recepción de ultrasonidos, por tanto, emite un pulso de arranque y cuantifica la anchura del pulso de retorno. Además, permite la conexión y manejo de manera sencilla con un microcontrolador. Las principales características se observan en la Tabla 3.3 [8].

Tabla 3.3: Características del Sensor HC-SR04 [8].

CARACTERISTICAS	DESCRIPCIÓN
Señal de entrada	Pulso 10 µs TTL
Señal de salida (Echo)	Señal PWL de TTL
Resolución	0.3 cm
Distancia de funcionamiento	2 cm – 450 cm
Ángulo de trabajo	<15º
Frecuencia de la señal emitida	40 KHz
Voltaje y corriente de trabajo	5 VDC / 15 mA

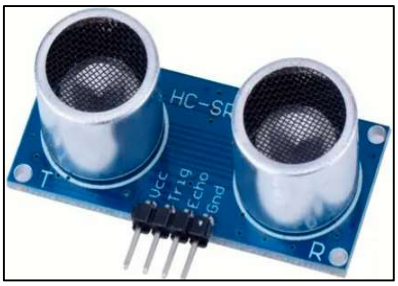

Figura 3.3: Ultrasónico HC-SRR04 [8]

Además, para verificar la presencia del usuario al momento de realizar la medición de la temperatura corporal, se utilizó un sensor infrarrojo (ver Figura 3.4). Se selección dicho sensor principalmente por su rango de detección que es de 2-30 cm y también por su tamaño, ya que este es adecuado para el prototipo.

El sensor infrarrojo, además presenta dos pines para alimentación 5V y GND y para enviar la señal al microcontrolador el pin denominado *OUT*, el cual indica si se recepto el reflejo del *LED* al fototransistor. Las principales características se las visualiza en la Tabla 3.4 [7].

Tabla 3.4: Características Sensor Infrarrojo [7].

CARACTERISTICAS	DESCRIPCIÓN
Modelo	FC-51
Angulo de trabajo	35 °
Voltaje de alimentación	3–6 VDC
Rango de funcionamiento	2–30 cm
Dimensiones	3,1 cm x 1,4 cm
Señal de salida	Salida nivel bajo cuando se detecta el obstáculo.

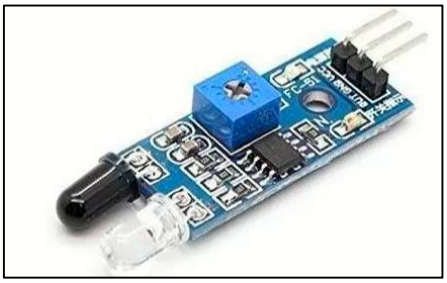

Figura 3.4: Sensor infrarrojo [7]

Para que el usuario pueda identificar que los procesos de detección son exitosos, se optó por colocar un *buzzer* activo ver Figura 3.5. Dicho componente se seleccionó ya que este dispositivo emite una frecuencia determinada y fija cuando, presenta un zumbador piezoeléctrico activo, el cual produce un sonido alrededor de 2,5 kHz. Además, no supone una carga para el microcontrolador ya que no requiere producir la señal eléctrica (*PWM*) para posteriormente convertirla en sonido, motivo por el cual este dispositivo fue seleccionado para emplearse en el prototipo. Las principales características se observan en la Tabla 3.5 [4].

Tabla 3.5: Características *Buzzer* [4].

CARACTERISTICAS	DESCRIPCIÓN
Voltaje de entrada	4V – 7V
Diámetro	12 mm
Separación Pines	7,6 mm
Corriente máxima	<30 mA
Intensidad de sonido	85 dB A 10 CM

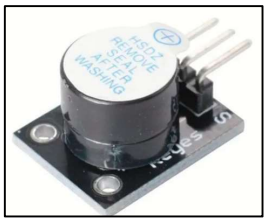

Figura 3.5: *Buzzer* activo [4]

- **Control de bombas de agua**

Para solventar el requerimiento de realizar el control tanto del agua y el jabón antibacterial, se optó por el uso de bombas de diafragma R385. Se selección dichas bombas ya que son de desplazamiento positivo, por lo que el empuje de las paredes elásticas ya sean de membranas o diafragmas, es provocado mediante el aumento de presión que varía el volumen de la cámara, ya sea aumentándolo o disminuyéndolo. Además, posee válvulas de retención, las cuales permiten el bombeo del líquido [5].

Este tipo bomba de diafragma es empleada en dispensadores de agua como teteras, hervidores eléctricos, etc. En comparación con versiones anteriores, esta ha modificado el soporte fijo, aumentando la vida útil en gran medida. Dicha bomba se visualiza en la Figura 3.6, además sus principales características de detallan en la Tabla 3.6. Para este proyecto se necesitan dos bombas, la primera será encargada de transportar el agua y la segunda transportará el jabón antibacterial, este tipo de bombas cumplen con los requerimientos del proyecto [5].

Tabla 3.6: Características bomba R385 [5].

CARACTERISTICAS	DESCRIPCIÓN
Voltaje de funcionamiento	6-12 VDC
Corriente de alimentación	0.5-0.7A
Caudal máximo	1-3L / Min
rango de aspiración máxima	2 m
Vida útil	2500H
Tamaño	86 x 43 mm

Figura 3.6: Bomba de diafragma R385 [5]

En conjunto con las bombas R385, se optó para realizar el control de las mismas el módulo relé 5V de dos canales (Figura 3.7), dicho módulo se activa la salida normalmente abierta al receptar un "0" lógico y se desactiva con un "1" lógico. Las principales características del módulo se las detalla en la Tabla 3.7, teniendo en cuenta dichas características el módulo cumple con los requerimientos del proyecto [3].

Tabla 3.7: Características del Módulo Relé de 2 canales [3].

CARACTERISTICAS	DESCRIPCIÓN
Tensión máxima	30 V-10 A por cada canal
Canales optoacoplados	2
Alimentación	5 VDC
Corriente de trabajo	20 mA cada canal

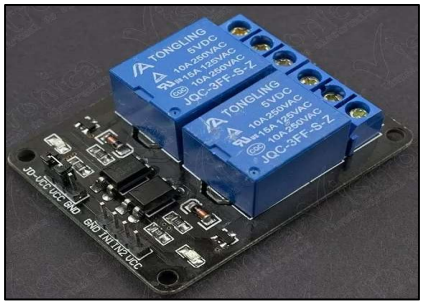

Figura 3.7: Modulo relé de dos canales [3]

- **Medición de la temperatura corporal de los usuarios**

Para cumplir con este requerimiento se optó por el uso del termómetro infrarrojo MLX90614 ver Figura 3.8, se seleccionó el sensor MLX90614, ya que presenta sensibilidad a la radiación infrarroja emitida por un objeto a distancia, además cuenta con un rango de temperatura de trabajo desde los -70°C hasta 380°C, con una precisión de 0.5°C [9].

La salida que presenta el sensor es lineal, la cual se compensa de acuerdo con las alteraciones de la temperatura ambiente, se encuentra integrado en la placa GY-906, lo que facilita la conexión con plataformas como Arduino. Las principales características se detallan

en la Tabla 3.8, tomando en cuenta dichas características este sensor cumple con los requerimientos del proyecto [9].

Tabla 3.8: Características del Termómetro infrarrojo MLX90614 [9].

CARACTERISTICAS	DESCRIPCIÓN
Protocolo de comunicación	SMBUS (I2C)
ADC	17 bits
Salida PWM	10 bits
Rango de temperatura (Ambiente)	-40 ℃ - 170 ℃
Rango de temperatura (Objeto)	-70 ℃ -380 ℃
Precisión	±0.5°C
Modelo	MLX90614
Voltaje de trabajo	5 VDC

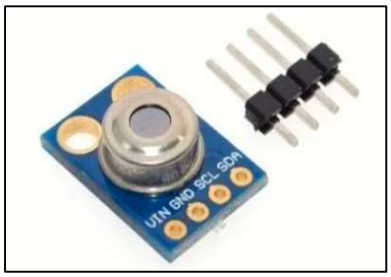

Figura 3.8: Sensor MLX90614 *[9]*

- **Envío de notificaciones**

Para solventar esta necesidad, se optó por el utilizar el módulo *NodeMCU* ESP8266 V3.0, ya que este módulo permite tener una conexión a Internet de forma inalámbrica, además presenta facilidad al momento de interactuar con plataformas como arduino mediante librerías [12].

El módulo *NodeMCU* ver Figura 3.9, está montado alrededor del chip ESP8266, esta versión cuenta con todos sus pines en los laterales. Presenta un regulador de tensión, además cuenta puerto USB, presenta la posibilidad de programar la palca mediante el IDE de Arduino [12].

Esta versión presenta una salida de 5V ya que utiliza conversor CH340, además sus pines permiten la posibilidad que la placa sea montada en un protoboard. Las características técnicas se detallan en la Tabla 3.9, teniendo en cuenta dichas características se seleccionó el módulo ya que cumple con los requerimientos del prototipo [12].

Tabla 3.9: Características del módulo *NodeMcu* V3 ESP8266 *Wi-Fi* [12].

CARACTERISTICAS	DESCRIPCIÓN
Voltaje Int	3.0 VDC – 5.0 VDC
Nivel de señal	TTL USB
Módulo	WI-FI ESP8266-12E
Antena	PCB
Numero de PWMs	12
Protocolos soportados	I2C, SPI, Serie, 1-Wire
Wi-Fi	802.11 b/g/n.

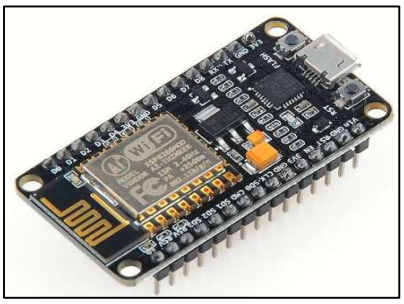

Figura 3.9: *NodeMcu* ESP8266 [12]

Para realizar el envió de notificaciones en forma de correos electrónicos, se utilizó la plataforma IFTTT, se optó por esta plataforma ya que se encarga de conectar aplicaciones, dispositivos y servicios de diferentes desarrolladores para activar una o más automatizaciones de manera gratuita. Además, dicha plataforma presenta facilidad de trabajar con la placa *NodeMCU* mediante el uso de librerías.

Las automatizaciones se logran a través de applets, que son macros que conectan múltiples aplicaciones para ejecutar tareas automatizadas. Se puede activar o desactivar un subprograma utilizando el sitio web de IFTTT o las aplicaciones móviles, en el proyecto dicha plataforma fue empleada para realizar el envío de correros electrónicos [18].

- **Interfaz de usuario**

Para solucionar la necesidad de una interacción entre el usuario y los distintos procesos del proyecto sea de manera sencilla, se empleó una pantalla TFT de la empresa *Nextion*.

Específicamente se seleccionó la pantalla inteligente TFT 3.5” *Nextion* NX4832T035 ver Figura 3.10, ya que cuenta con una interfaz de control interactiva con el usuario, además utiliza un único puerto serie para realizar la comunicación con el microcontrolador. Es fácil de incorporar a cualquier proyecto basado en Arduino, ya que se puede hacer uso de las librerías oficiales. Para programarla se utiliza el *software* proporcionado por el fabricante *Nextion Editor*, además que para quemar el programa en la pantalla se utiliza el protocolo de comunicación TTL o por medio de la entrada micro-SD. Las principales características técnicas de dicha pantalla se detallan en la Tabla 3.10 [6].

Tabla 3.10: Características Pantalla *Nextion* NX4832T035 [6].

CARACTERISTICAS	DESCRIPCÓN
Resolución	480 x 320
Interfaz serie TTL	4 pines
Memoria *Flash*	16 MB
Voltaje de alimentación	5 VDC
Corriente	500 mA
Consumo	5V – 145 mA

Figura 3.10: Pantalla *Nextion* NX4832T035 [6]

Para el diseño de la interfaz gráfica, se seleccionaron dos *softwares,* el primero *Nextion Editor*, porque es un *software* de desarrollo que se utiliza para la construcción visual de la interfaz gráfica de usuario para dispositivos integrados con uso intensivo de GUI con varios tipos de pantallas TFT y paneles táctiles.

Adicionalmente, para dimensionar las imágenes en función a la resolución de la pantalla 480 x320 pixeles se optó por el *software Inkscape*, ya que permite la edición de vectores gráficos de calidad.

- **Automatización de procesos**

Para la automatización de los diferentes procesos que debe realizar el proyecto, es necesario el uso de un microcontrolador. Por tanto, se realizó una tabla comparativa (Tabla 1.2) entre los ecosistemas más usados actualmente en el medio electrónico: Arduino y *Raspberry* Pi. Esto con la finalidad de solventar los requerimientos de los distintos procesos detallados anteriormente.

Si bien el ecosistema de *Raspberry* Pi presenta mayores ventajas en comparación con Arduino. La simplicidad, costos y su capacidad de incluir módulos externos, hace que Arduino nos permita trabajar con cualquier tipo de sensor, siendo la mejor elección para el proyecto.

Una vez seleccionado el ecosistema, se procedió a la elección de la placa de la plataforma de Arduino, en función al número de pines necesarios para el manejo de sensores (15 pines digitales) y puertos seriales (3 puertos seriales) que necesita el proyecto, por tanto, se optó por el Arduino Mega2560.

El Arduino Mega 2560 se lo visualiza en la Figura 3.11, se trata de una placa de desarrollo basada en el microcontrolador ATmega2560. Cuenta con 16 entradas analógicas, 4 UARTs, un cristal de 16Mhz, además presenta 54 pines digitales (15 pueden ser usadas como salidas PWM). Las principales características del Arduino MEGA2560 se pueden visualizar en la Tabla 3.11 [11].

Tabla 3.11: Características Arduino MEGA 2560 [11].

CARACTERISTICAS	DESCRIPCÓN
Corriente de cada Pin	40 mA
Frecuencia de reloj	16 MHz
Voltaje de trabajo	5 VDC
Alimentación externa	7 – 12 VDC
Pines digitales	54 (15 PWM)
Pines analógicos	16 (A0 – A15)
Microcontrolador	ATMega2560
Voltaje de entrada límite	7 VDC – 20 VDC

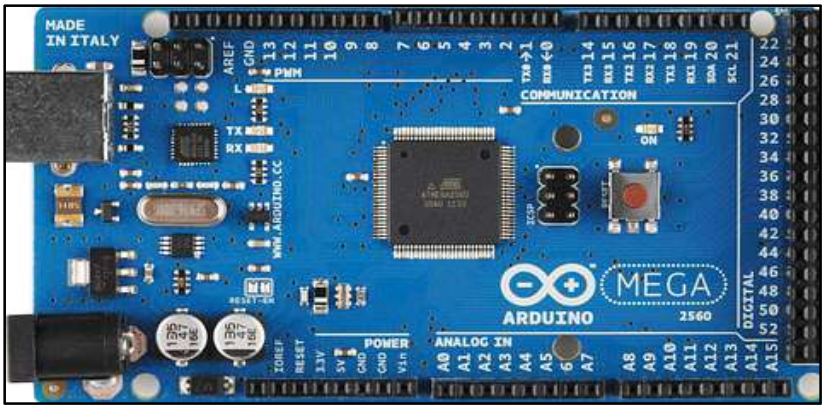

Figura 3.11: Arduino Mega 2560 [11]

- **Alimentación del proyecto**

Como el prototipo hace uso de diferentes componentes electrónicos los cuales necesitan distintos niveles de voltaje y corriente. En la Tabla 3.12 se puede visualizar los distintos niveles

de voltaje y corriente de cada componente, para ello se tomó lo valores ideales proporcionados por los fabricantes.

Tabla 3.12: Corriente nominal de cada componente.

Componentes	**Corriente**
Arduino Mega 2560	600 mA
Pantalla *Nextion* NX4832T035	500 mA
Módulos Relé de 2 canales	80 mA
Módulo RFID RC522	26mA
Sensor ultrasónico HC-SR04	15 mA
Sensor infrarrojo	23 mA
Buzzer Activo	30 mA
Termómetro infrarrojo MLX90614	30 mA
Módulo *NodeMcu* V3 ESP8266 Wi-Fi	90 mA
2 Bombas R385	1.4 A
TOTAL	2.72 A

Teniendo en cuenta los distintos niveles de voltaje y corriente para alimentar al prototipo, se optó por el uso de la fuente de alimentación de la marca *ColorsIt* modelo EN60950 (ver Figura 3.12). Las características de dicha fuente se detallan en la Tabla 3.13, con lo cual satisface la demanda de energía de los distintos componentes del proyecto [23].

Tabla 3.13: Características Fuente de poder *COLORS IT* EN60950 480W [23].

CARACTERISTICAS	**DESCRIPCIÓN**
Fabricante	ColorsIt
Modelo	ATX12V P4 / EN60950
Módulo	WI-FI ESP8266-12E
Potencia	480 watt
Voltaje Entrada	230/115 VAC 50-60 Hz
Voltajes de Salida	+3.3V - 1.4 A +5 V - 28 A +12 v - 18 A

Figura 3.12: Fuente de Poder *ColorsIt* [23]

3.3 Diseño del sistema de automatización

- **Esquema general del proyecto**

En el esquema general se detalla el funcionamiento del proyecto y la manera que interactúan entre si los elementos. El prototipo se desarrolló con dos sistemas de control el uno encargado de interactuar con los elementos electrónicos y otro cumpliendo la tarea de enviar y recibir información mediante la conexión a internet.

En la Figura 3.13 las guías azules simbolizan la comunicación para que la placa de Arduino reciba datos de los elementos electrónicos, en cambio las guías naranjas representan como la placa de Arduino envía datos a los mismos. De igual forma, las líneas naranjas y azules entrecortadas representan el envío y recepción de datos entre el módulo ESP8266 y la plataforma de *software* IFTTT mediante una conexión a Internet. Además, la comunicación serial entre las dos placas se representa con una guía en forma de flecha doble.

El proceso inicia con el usuario pasando el *tag* por el lector RFID, el programa identifica al usuario y manda una señal de onda para que el *buzzer* emita un sonido indicando que se produjo la lectura del *tag*, después presenta el nombre del usuario mediante la pantalla *Nextion*.

Posteriormente, se indica que el usuario debe colocar las manos debajo del grifo, el sensor ultrasónico detecta que el usuario colocó las manos debajo del mismo y emite un sonido mediante el *buzzer* e inicia el proceso del lavado de manos. Posteriormente, se muestra mediante la interfaz gráfica el proceso del correcto lavado de manos, teniendo en cuenta el tiempo que debe durar dicho proceso. Posteriormente, se procede con la activación de las bombas R385 tanto del agua como la del jabón.

Para la medición de temperatura corporal, se le indica al usuario mediante la interfaz gráfica que coloque la muñeca en el sensor MLX90614, el sensor infrarrojo detecta la muñeca del usuario emite un sonido para indicar que se detectó la misma e inicia con la medición, el valor de la temperatura se muestra al usuario mediante la pantalla. En caso de que la temperatura sea mayor a 38 °C, el Arduino enviará al módulo *WI-FI* ESP82666 mediante comunicación serial el nombre del usuario con su respectiva temperatura.

Para realizar el envío de la alarma en forma de correo electrónico, se procede a recibir los datos que envía la placa Arduino mediante comunicación serial, se procesa los datos y se envía a la plataforma IFTTT mediante el método POST, esta a su vez enviará un correo electrónico a la cuenta de la persona encargada de los usuarios, con el fin de tomar medidas adecuadas para evitar posibles contagios. Posteriormente, se indica a la placa de Arduino que él envió del correo fue exitoso.

La placa Arduino recibe el mensaje que el correo fue enviado y le indica al usuario que espere instrucciones mediante la pantalla *Nextion*. En caso de que la temperatura del usuario sea normal, se indicara al mismo que el proceso ha terminado sin ninguna complicación y el sistema esperará a que otro usuario acerca su *tag* al lector e iniciara de nuevo el proceso.

Como se detalló en párrafos anteriores, el proyecto a desarrollar presenta dos mecanismos de control que trabajan en conjunto para cumplir con los requerimientos de los usuarios, teniendo en cuenta que los mismos tengan un contacto mínimo con cualquier superficie del prototipo. Para finalizar, a continuación se observa el esquema del prototipo, donde se indica cómo interactúan entre sí los elementos del proyecto.

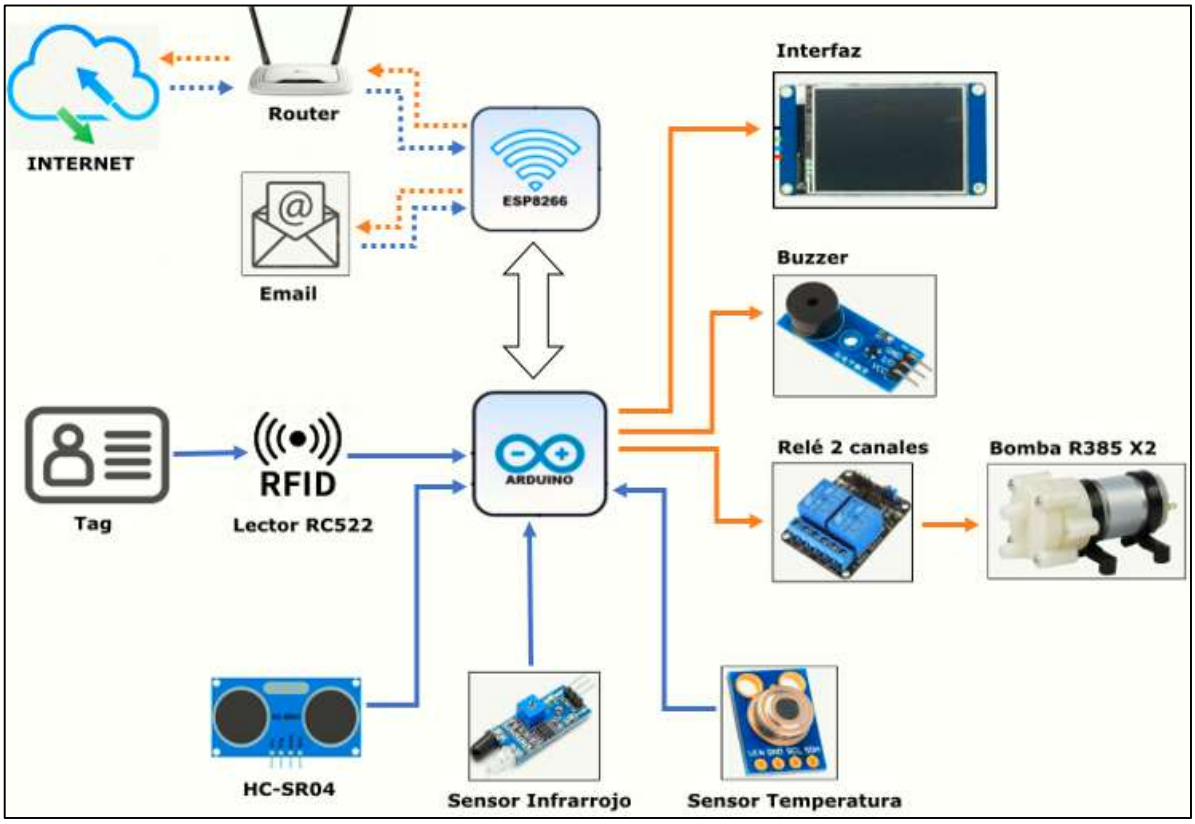

Figura 3.13: Esquema general del funcionamiento del proyecto

- **Diseño de la interfaz gráfica**

Para comenzar, se procedió al diseño de las imágenes en las cuales se mostrará las diferentes etapas del sistema. Esto se lo realizó con ayuda del programa *Inkscape*, teniendo en cuenta que la pantalla *Nextion* utilizada en el prototipo presenta una resolución de 480x320. El diseño de las imágenes en dicho programa se puede visualizar en la Figura 3.14.

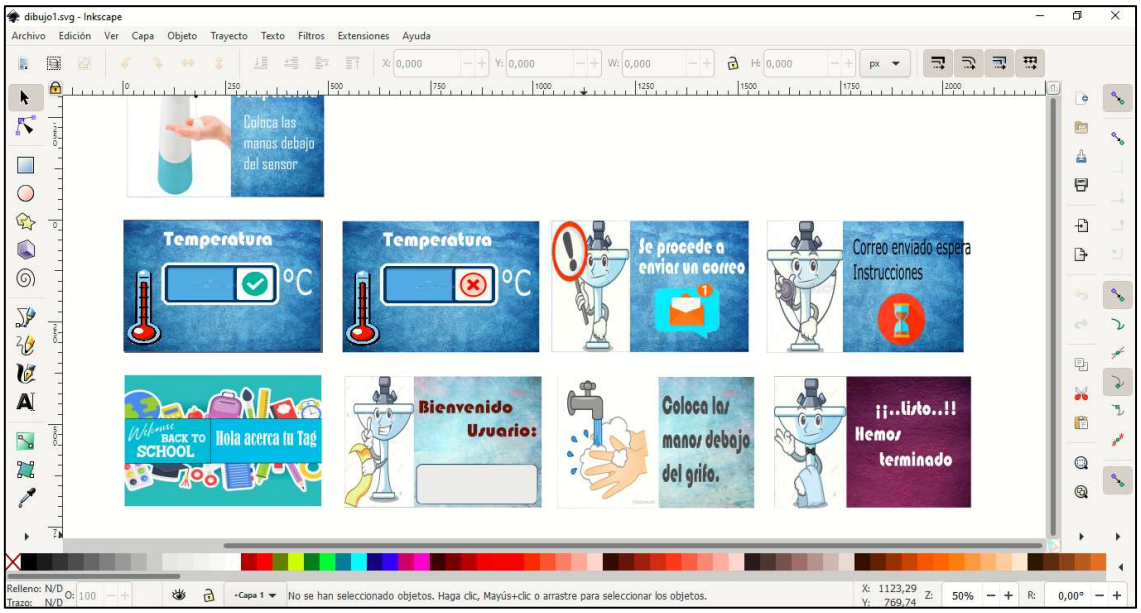

Figura 3.14: Diseño de imágenes 480X320

A continuación, se procedió a la creación de la interfaz para la pantalla *Nextion,* mediante el *software Nextion* editor. Para ello se realizaron los siguientes pasos.

- Primero, se seleccionó el tipo de pantalla, su resolución y la orientación, para el prototipo se seleccionó la pantalla <NX4832T035_011> con la orientación de 180 grados. Este proceso se visualiza en la Figura 3.15.

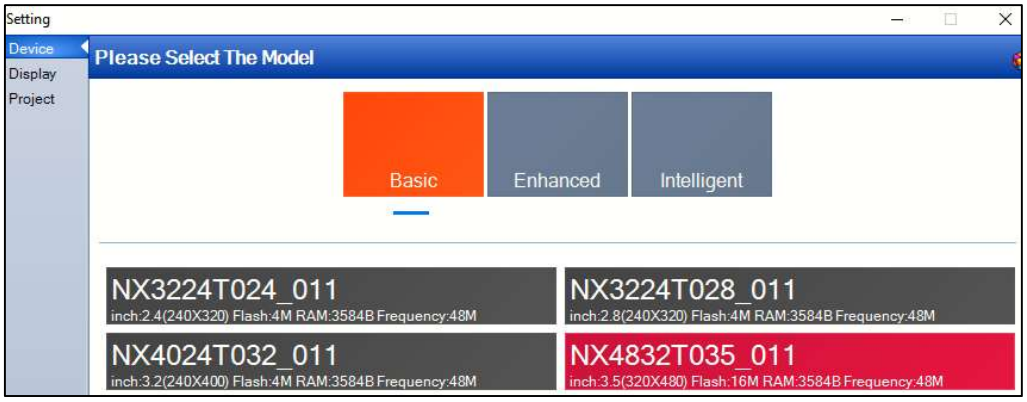

Figura 3.15: Configuración de la pantalla

- Después, se declararon los tipos de fuentes para el uso de las letras o números. Esto se visualiza en la Figura 3.16.

Figura 3.16: Creación de fuentes

- Posteriormente, se agregaron las imágenes que fueron creadas en el programa *Inskcape* considerando la resolución de la pantalla 480x320. Esto se observa en la Figura 3.17.

Figura 3.17: Selección de imágenes

- A continuación, para que la pantalla *Nextion* pueda ser controlada por Arduino, se crearon “páginas” en las cuales se insertaron las imágenes creadas anteriormente. Además, para que Arduino pueda identificarlas, se configuró el nombre de cada una, para el proyecto se requirió crear 17 páginas, para los distintos procesos del sistema. Esto se visualiza en la Figura 3.18.

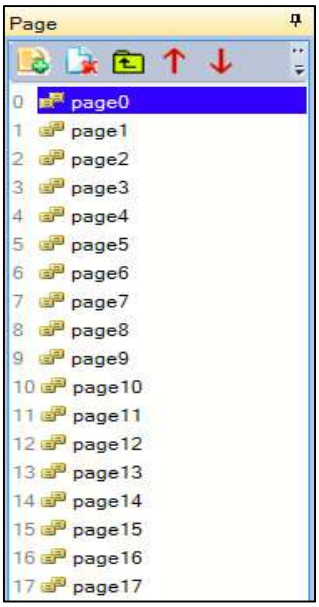

Figura 3.18: Creación de páginas.

- Para visualizar en la pantalla el nombre del usuario, se precedió a crear un botón de texto. Al mismo se lo denominó <nomusuario>, esto con la finalidad que el Arduino puede identificarlo y enviar datos de tipo *char*. Esto se observa en la Figura 3.19.

Figura 3.19: Creación botones de texto

- Para visualizar en la pantalla la temperatura del usuario, se procedió a crear dos páginas y en las mismas se declararon botones de texto. Para mostrar en la pantalla cuando la temperatura es normal, se creó la página 13 y en la misma se declaró el botón denominado <tcorrecta>, en cambio, cuando la temperatura es incorrecta se definió la página 14 y se declaró el botón de texto <tincorrecta>. Esto se observan en la Figuras 3.20 y 3.21 respectivamente.

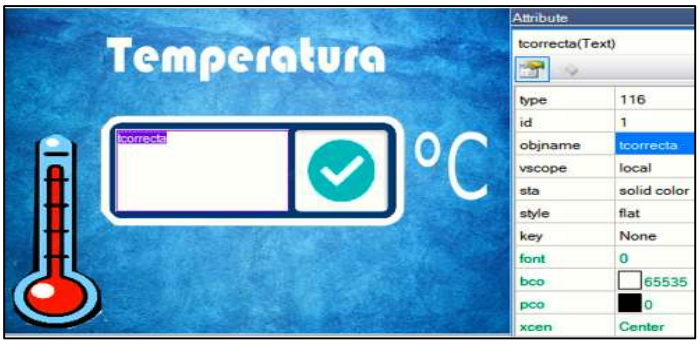

Figura 3.20: Botón de texto - Temperatura correcta

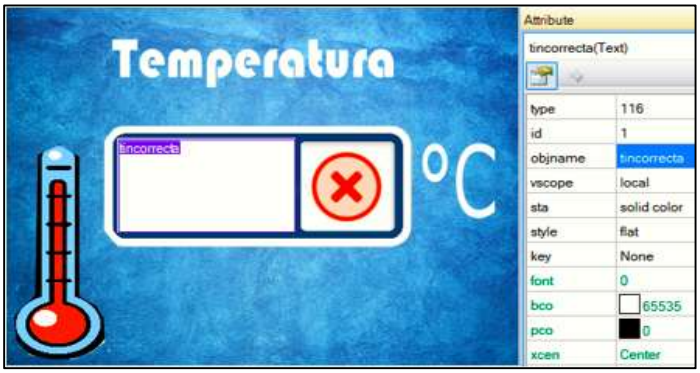

Figura 3.21: Botón de texto - Temperatura incorrecta

La forma en la que el Arduino gestiona el manejo de las páginas se describe más adelante. Finalmente, para grabar el programa en la pantalla *Nextion*, se utilizó una tarjeta micro-sd clase 10 en la cual se guardó el archivo "tft", y posteriormente se cargó el archivo en la pantalla.

- **Configuración plataforma IFTTT**

El correo electrónico utiliza como protocolo de comunicación SMTP, el cual permite enviar correos electrónicos a través de internet. Se utilizó la plataforma IFTTT en su versión gratuita, para evitar contratar un servicio de hosting. Este servicio nos permite programar acciones, ya sean llamadas o notificaciones con el fin de automatizar tareas mediante applets. Para la creación de applets, se debe configurar las condiciones "*This*" y "*That*", las mismas se visualiza en la Figura 3.22.

Figura 3.22: Condiciones - *This/That*

Para comenzar, se procedió a la configuración de la condición "*This".* En esta condición se configura el tipo de servicio que se necesita, en este caso para el envío de correos

electrónicos, se seleccionó el servicio *"Webhooks".* El cual nos permite desde un microcontrolador enviar un *"web request"* a través de un *trigger.* Esto se observa en la Figura 3.23.

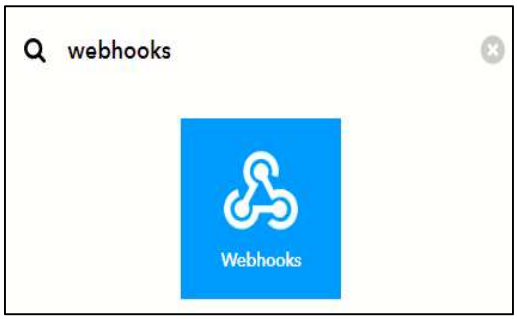

Figura 3.23: Selección del servicio

Después, se procedió a declarar el nombre del evento con el cual se activará el trigger. En este caso se lo denominó como <*email_alarm*>. Esto se muestra en la Figura 3.24.

Figura 3.24: Declaración del evento - *email_alarm*

Una vez, creado el *trigger* se configuró la condición *"That",* como el objetivo es que cada vez que un usuario presenté fiebre se envié un correo electrónico. Se selección el servicio <*Email*>. Esto se observa en la Figura 3.25.

Figura 3.25: Selección del servicio – *Email*

Seleccionado el servicio, la plataforma se encargará de enviar un *email* basado en HTML. El correo está conformado en dos partes el *subject* y el *body*. Para que la plataforma reciba información desde el microcontrolador hace uso de etiquetas las cuales son variables, en este caso se utilizó dos etiquetas una para almacenar el nombre <Value1> y para guardar el valor de temperatura <Value2>. Adicionalmente, se

configuró el *subject* como <ALARMA-ESTUDIANTE PRESENTA FIEBRE>, esto se observa en la Figura 3.26.

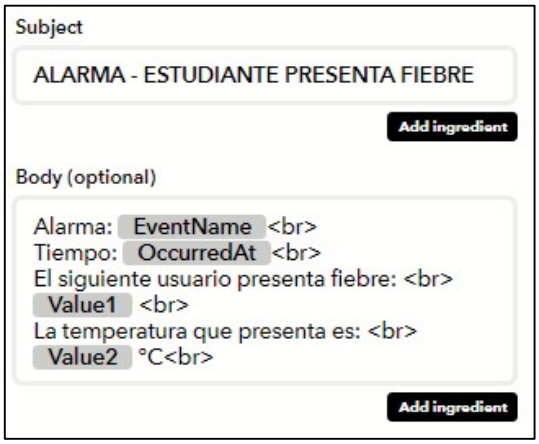

Figura 3.26: Diseño del correo electrónico

Posteriormente, se configuró la cuenta del correo electrónico a la que se enviará las alarmas, en el caso de que un usuario presente fiebre. En este caso se configuró el *email* profe.prueba.777@gmail.com. Esto se visualiza en la Figura 3.27.

Figura 3.27: Configuración del correo electrónico

Una vez creada los applets, se procedió a verificar el formato para activar el *trigger*, ya sea por medio del método *POST* o *GET*, en el proyecto se seleccionó el método *"POST".*

Se selección el método *POST* porque en lo relativo a los datos, como, por ejemplo, al rellenar formularios con nombres de usuario y contraseñas, el método *POST* ofrece mucha discreción. Los datos no se muestran en el caché ni tampoco en el historial de navegación. La flexibilidad del método *POST* también resulta muy útil, no solo se pueden enviar textos cortos, sino también otros tipos de información, como fotos o vídeos.

A continuación, se procedió a verificar la *"key"* del servicio de hosting, esta información se encuentra en la opción <*Documentation*> en la parte superior derecha de la plataforma ver Figura 3.28. A continuación se despliega una pestaña en la cual en la opción <*Your key is*> se encuentra la *key* de nuestro hosting (Ver Figura 3.29). Esta información es necesaria para configurar la petición que se va a realizar desde el módulo *Wi-Fi* hacia la plataforma IFTTT.

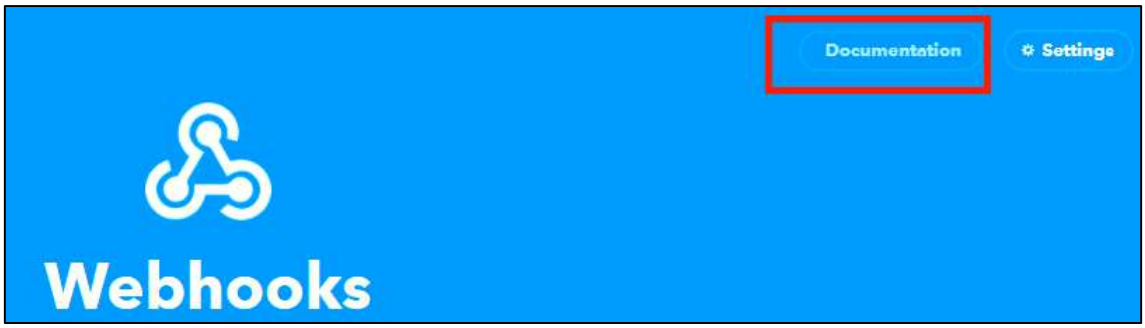

Figura 3.28: Opción <Documentation>

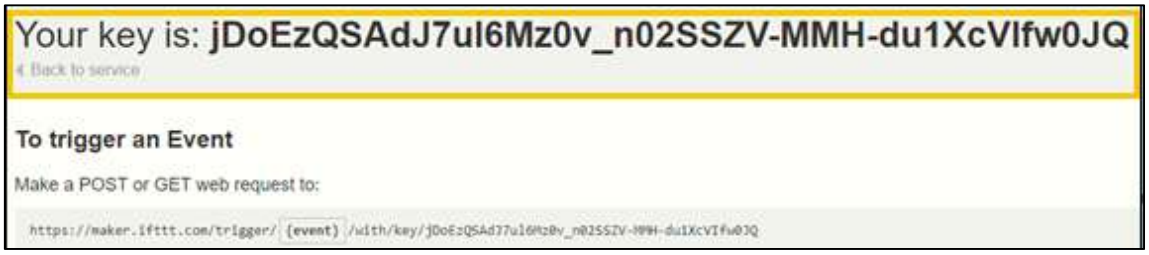

Figura 3.29: Llave del *applet* – *email_alarm*

- **Programación del módulo ESP8266 y Arduino Mega 2560**

La programación de los módulos de Arduino se lo realizó en lenguaje C y se compiló mediante el *software* Arduino IDE. La Figura 3.30 se visualiza el diagrama de flujo general de la programación del sistema. Arduino se encarga de enviar y recibir datos a los componentes, esto con el fin de gestionar y automatizar el sistema, mientras que el módulo *Wi-Fi* se encarga de enviar y recibir datos a la plataforma IFTT.

El diagrama de flujo general del programa presenta una introducción de la programación a desarrollar, al inicio de esta se muestra las librerías a utilizar como RFID, *WIFI*-ESP8266, *Nextion,* etc. A partir de este punto la programación se divide en dos códigos de programación.

El primero código, en la cual la placa de Arduino MEGA 2560 debe realizar diversas tareas: lectura de los sensores, manejo de la interfaz y la comunicación con el módulo *Wi-Fi.* Por tanto, se procedió a la creación de subrutinas con el fin de automatizar el sistema. El segundo, tiene que ver con el módulo *Wi-Fi* ESP8266 y la conexión con la plataforma IFTTT, la cual se encarga de enviar el correo electrónico a una cuenta predeterminada.

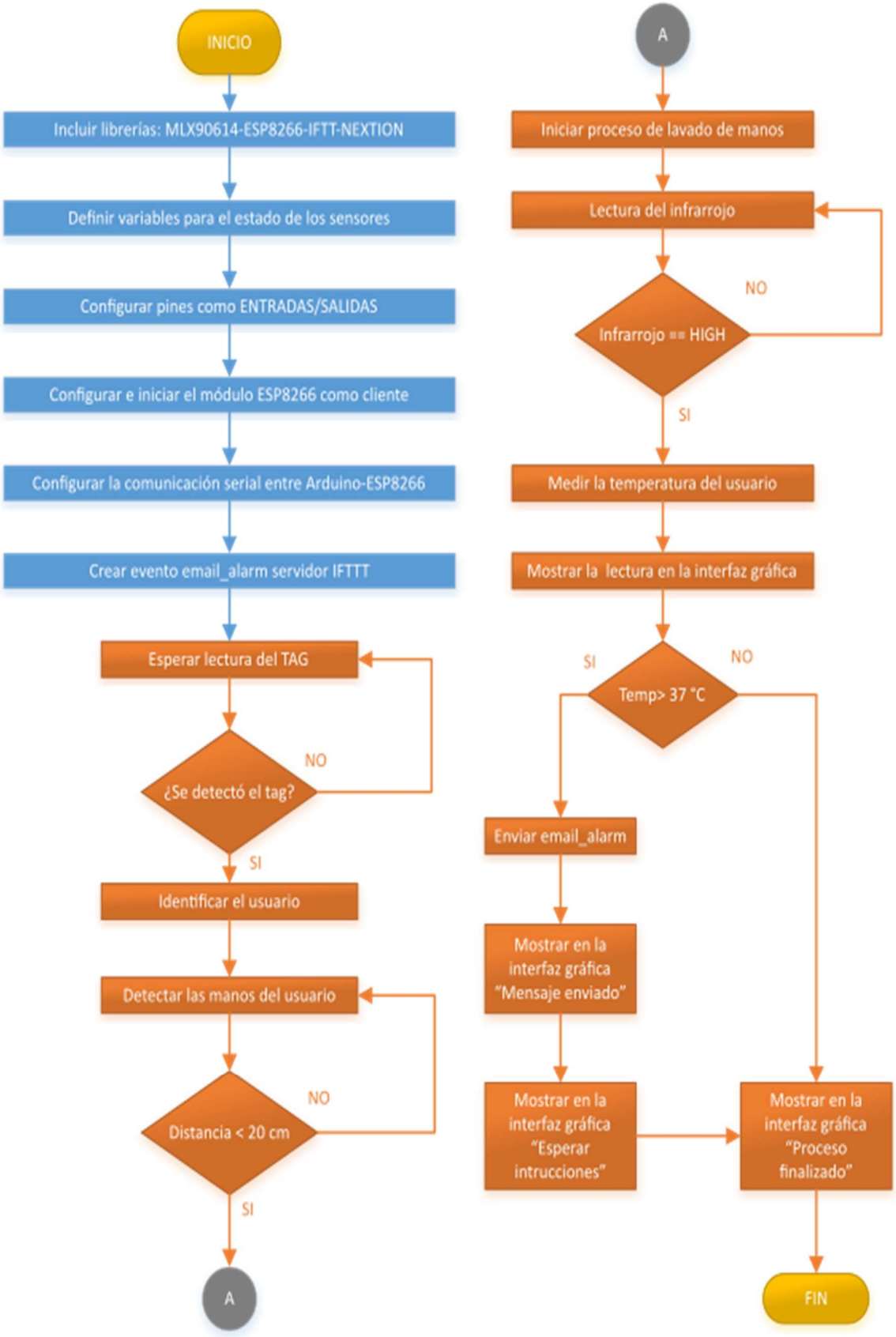

Figura 3.30: Diagrama general del prototipo

- **Programación del módulo - Arduino MEGA2560**

Para el desarrollo del código en la placa Arduino se tomaron en cuenta los siguientes aspectos.

- Inclusión librerías para el manejo de componentes.
- Creación de variables para realizar los procesos de automatización.
- Configuraciones en la parte del *setup.*
- Creación de funciones para la automatización del sistema.

Inclusión de las librerías para el manejo de los componentes electrónicos

Librería MFRC522 - Lector RFID

Para el manejo del lector RFID RC522, se agregaron las librerías <SPI.h> y <MRF522.h>. La primera se utiliza para que el lector se comunique con el Arduino por medio del protocolo de comunicación SPI, mientras que la segunda es la que se encarga de la lectura de los *tags* de cada usuario.

Los pines MOSI, MISO y SCK, ya están preconfigurados para trabajar con el Arduino mega, por lo tanto, se configuraron los pines restantes RST y SDA, posteriormente se creó el objeto <MFRC522> para establecer las configuraciones de dichos pines. Esto se puede visualizar en la Figura 3.31.

```
                MFRC522      Arduino       Arduino   Arduino
                Reader/PCD   Uno/101       Mega      Nano v3
   Signal       Pin          Pin           Pin       Pin
   ---------------------------------------------------------
   RST/Reset    RST          9             48         D9
   SPI SS       SDA(SS)      10            53        D10
   SPI MOSI     MOSI         11 / ICSP-4   51        D11
   SPI MISO     MISO         12 / ICSP-1   50        D12
   SPI SCK      SCK          13 / ICSP-3   52        D13
*/
#include <SPI.h>        //Libreria para comunicar por el bus SPI
#include <MFRC522.h>
///Usuarios
#define RST_PIN 48      //Pin 9 para el reset del RC522
#define SS_PIN  53      //Pin 10 para el SLAVE SELECT (SDA) del RC522
MFRC522 mfrc522(SS_PIN, RST_PIN); //Creamos el objeto para el RC522
```

Figura 3.31: Librería lector RFID

Librería Sensor Adafruit_MLX90614.

Para la utilización del sensor de temperatura MLX90614, se incluyeron las siguientes librerías <Wire.h> y < Adafruit_MLX90614.h>. La primera permite la comunicación SMBus, el cual es un subconjunto de I2C, mientras que la segunda permite usar el sensor MLX90614. La respectiva programación se observa en la Figura 3.32.

```
#include <Wire.h>
#include <Adafruit_MLX90614.h>
Adafruit_MLX90614 mlx = Adafruit_MLX90614();
```

Figura 3.32: Librería Sensor MLX90614

Librería *JSON*

Para el envío y recepción de datos entre placa ESP8266 y el módulo Arduino, se utilizó la librería *JSON* mediante la instrucción <ArduinoJson.h>. Internamente, esta librería incorpora funciones para serializar y de deserializar objetos de forma sencilla. En este caso, se utilizó para enviar el nombre del usuario con su respectiva temperatura. La programación se observa en la Figura 3.33.

```
#include <ArduinoJson.h> Se incluye la librería JSON
```

Figura 3.33: Librería *JSON*

Librería *Nextion*

Para controlar la pantalla *Nextion* se utilizó la librería oficial proporcionada por el fabricante mediante la instrucción <*Nextion.h*>. Una vez instalada la librería se procedió a declarar las “páginas”, en las cuales se muestran las imágenes para indicar los procesos del sistema, dicha acción se realizó mediante la instrucción <*NexPage page#*>.

Además, para mostrar en la pantalla el nombre del usuario y su temperatura, se procedió a crear los objetos necesarios para mostrar en ellos los datos enviados por Arduino, para esto, se usó la instrucción <*NexText* “nombre del objeto”>. Hay que tener en cuenta que el nombre de las páginas y de los objetos se los configuró previamente en el editor de *Nextion*, dicho proceso se explicó en párrafos anteriores. La programación de dichas acciones se observa en la Figura 3.34.

```
#include "Nextion.h"
//////////Paginas nextion
NexPage page0     =   NexPage(0, 0, "page0"); // Bienvenido
NexPage page1     =   NexPage(1, 0, "page1"); // Usuario
NexPage page2     =   NexPage(2, 0, "page2"); // Colocar las manos bajo el grifo
NexPage page3     =   NexPage(3, 0, "page3"); // Proceso:1 Mojar las manos
NexPage page4     =   NexPage(4, 0, "page4"); // Proceso:2 Dispensor de jabon
NexPage page5     =   NexPage(5, 0, "page5"); // Proceso:3 Frotar las palmas
NexPage page6     =   NexPage(6, 0, "page6"); // Proceso:4 Entrelazar los dedos
NexPage page7     =   NexPage(7, 0, "page7"); // Proceso:5 Frotar manos-entrelazar
NexPage page8     =   NexPage(8, 0, "page8"); // Proceso:6 Frotar el dorso de los dedos
NexPage page9     =   NexPage(9, 0, "page9"); // Proceso:7 Movimiento de rotación
NexPage page10    =   NexPage(10, 0, "page10"); // Proceso 8: Frotar punta de los dedos
NexPage page11    =   NexPage(11, 0, "page11"); // Proceso 9: Enjuagar manos
NexPage page12    =   NexPage(12, 0, "page12"); // Temperatura: Deteción del sensor
NexPage page13    =   NexPage(13, 0, "page13"); // Temperatura: Correcta
NexPage page14    =   NexPage(14, 0, "page14"); // Temperatura: Incorrecta
NexPage page15    =   NexPage(15, 0, "page15"); // Envió e-mail
NexPage page16    =   NexPage(16, 0, "page16"); // Esperar indicaciones
NexPage page17    =   NexPage(17, 0, "page17"); // Fin del proceso
/////////Objetos de la pantalla
NexText   tu     =   NexText(1, 1, "nomusuario"); //Cuadro texto Usuario
NexText   tc     =   NexText(13, 1, "tcorrecta"); //Cuadro texto temperatura correcta
NexText   ti     =   NexText(14, 1, "tincorrecta"); //Cuadro texto temperatura Incorrecta
```

Figura 3.34: Librería Pantalla *Nextion*

Creación de variables para realizar los procesos de automatización

Para el funcionamiento del programa es necesario el uso de variables, las mismas se utilizaron en los distintos procesos del sistema.

Las variables usadas para almacenar la temperatura y el nombre de cada usuario son dos <usuario> y <temperatura_usuario>, la primera es de tipo *string* ya que en esta variable se almacenará el nombre del usuario en forma de caracteres y la segunda es del de tipo *float* porque se almacenará el valor decimal de la temperatura del usuario.

Adicionalmente se configuró la variable tipo *string* <mensaje>, esta última tiene como función enviar en forma de *string* la información de los usuarios. La respectiva programación se visualiza en la Figura 3.35.

```
String usuario; //Variable String almacenar el nombre del usuario
String mensaje = ""; //Variable String envió usuario-temperatura
float temperatura_usuario; //Variable float almacenar temperatura
```

Figura 3.35: Configuración de variables – Envío de datos

Para realizar la lectura de los *tags* RFID, se utilizaron 3 variables tipo *byte*, en la variable <LecturaUID> se guardan los valores del *tag* del usuario, mientras que en la variable <Usuario1> y <Usuario2> se almacena en la placa Arduino el número de serie único hexadecimal de cada *tag*, esto para poder realizar la comparación. Se seleccionó este tipo de variable por el tamaño y tipo de información almacenada en los *tags* pasivos. La programación se observa en la Figura 3.36.

```
byte LecturaUID[4];     //Creamos un array de 4 bits para guardar los valores de los TAGS
byte Usuario1[4] = {0x65, 0xDD, 0xFA, 0xC2}; // TAG del usuario 1
byte Usuario2[4] = {0xCD, 0xBF, 0x06, 0x85}; //TAG del usuario 2
```

Figura 3.36: Configuración de variables - RFID

Para la medición de la temperatura corporal de los usuarios, se procedió a crear tres variables tipo *float*, ya que el valor de la medición de la temperatura corporal del usuario es de tipo decimal. La primera variable <temp_persona> se encarga de guardar la medición realizada por el sensor MLX90614, la variable <temp_normal> se encarga de fijar el límite en el que se considera que una persona presenta una temperatura normal y la variable <temp_usuario> se encarga de almacenar el valor de la variable <temp_persona>, para posteriormente enviar la respectiva medición temperatura al módulo ESP8266. La programación se visualiza en la Figura 3.37.

```
float temp_persona;
float temp_normal = 37.00;
float temp_usuario;
```

Figura 3.37: Configuración de variables – Temperatura

Distribución de pines para los elementos electrónicos

Para utilizar los distintos componentes electrónicos del prototipo, es necesario definir que pines van a ser utilizados y además en qué forma van a ser configurados ya sea como entradas o salidas.

Para la configuración de los pines de entrada, se excluyeron los pines del lector RFID y del sensor de temperatura MLX90614, los cuales ya vienen previamente configurados por sus librerías. Para el resto de los componentes fue necesario definir los pines a ser utilizados.

Para el uso del sensor ultrasónico, se les asignó los pines 22 y 23. También, se definieron a los mismos como <echoPin> y <trigPin> respectivamente. Además, para utilizar el *buzzer*, se le asignó el pin 44 y fue definido como <*buzzer*>.

Además, para el control de las bombas de agua se les asignó los pines 24-25, los cuales fueron definidos como <grifo> <jabón> respectivamente. Finalmente, para el uso del sensor infrarrojo se le asignó el pin 26 y se lo definió como <sensor_infrarrojo>, esto con el fin de leer el estado de este. La programación de estas acciones se observa en Figura 3.38.

```
/// ENTRADAS
 int echoPin = 22;
 int sensor_infrarojo = 26;
//// SALIDAS
 int trigPin = 23;
 int buzzer = 44;
 int grifo = 24;
 int jabon = 25;
```

Figura 3.38: Distribución de pines

Configuración del *Setup*

En la parte del *setup*, se realizaron las configuraciones previo al arranque del sistema. Primero, se estableció la velocidad de transmisión entre el módulo Arduino y el módulo ESP8266, esto es de mucha importancia ya que la velocidad debe ser la misma en las dos placas para evitar datos erróneos. La programación se encuentra en la Figura 3.39.

```
Serial.begin(115200); //Configurar comunicación serial  del Arduino
Serial3.begin(115200); //Configurar comunicación serial con el ESP8266
```

Figura 3.39: Establecimiento de velocidad de transmisión

Después, se declararon los pines de cada componente del sistema como entradas y salidas. Esto se puede visualizar en la Figura 3.40.

```
pinMode(sensor_infrarrojo,INPUT); //Se declara como entrada pin infrarrojo
pinMode(echoPin, INPUT); //Se declara como entrada el pin echo ultrasónico
pinMode(trigPin, OUTPUT); //Se declara como entrada el pin trig ultrasónico
pinMode(buzzer,OUTPUT); //Se declara como salida el pin trig ultrasónico
pinMode(grifo,OUTPUT); //Se declara como salida el pin grifo
pinMode(jabon,OUTPUT); //Se declara como salida el pin jabón
```

Figura 3.40: Declaración de entradas y salidas

También, se configuró el arranque del protocolo de comunicación SPI, además se procede a iniciar las librerías necesarias para el uso del lector RFID, la pantalla *Nextion* y el sensor de temperatura. Finalmente, se configuró que al iniciar el programa se muestra en la pantalla *Nextion* la página 0. La programación se observa en la Figura 3.41.

```
SPI.begin(); //Se inicia protocolo de comunicación SPI
mfrc522.PCD_Init(); //Se inicia la librería MFRC522
nexInit();//Se inicia la librería Nextion
mlx.begin();//Se inicia la libreria sensor MLX90614
page0.show();//Se observa en la pantalla la página 0
```

Figura 3.41: Arranque de las librerías

Subrutinas del sistema

Para comenzar, se procedió a programar en la parte del *loop* las condiciones para iniciar con la lectura del *tag* e iniciar con el proceso de lavado de manos. Para lograrlo se colocaron dos condicionales que se deben cumplir para iniciar con el proceso de la lectura del *tag* de cada usuario.

El primer condicional se trata de la función <!mfrc522.PICC_IsNewCardPresent()>, cumple con la tarea de verificar si el lector detectó algún *tag*, en caso de que no se detecte retorna al *loop*. El segundo condicional, es la función <!mfrc522.PICC_ReadCardSerial()>, la cual se encarga de obtener los datos de la tarjeta y en caso de no hacerlo al igual que la anterior se retorna a la parte del *loop*.

Cuando las dos condiciones se cumplen, se inicia con la subrutina <Lectura_tarjeta()>. La programación de dichas funciones se observa en la Figura 3.42.

```
void loop() {
  if ( ! mfrc522.PICC_IsNewCardPresent()) //Verificar detección del TAG
    return;  // Retorna al loop esperando la lectura de un tag

  if ( ! mfrc522.PICC_ReadCardSerial())//Si no obtiene datos de la tarjeta
    return;  // Retorna al loop esperando por otra tag

  Lectura_tarjeta ();
```

Figura 3.42: Condicionales para iniciar lectura de los *tags* RFID

Subrutina <Lectura_tarjeta ()>

Para la lectura de los *tags* de los usuarios, se creó la subrutina denominada <Lectura_tarjeta ()>, la cual cumple con la tarea de comparar el numero único del *tag* del usuario con los registrados previamente en la memoria del Arduino. Posteriormente, se llama a la subrutina < comparaUID >, la cual inicia con la comparación del serial de los *tags*, se identifica al usuario y se pasa a la función <Bienvenido ()>. La programación se visualiza en la Figura 3.43.

```
void Lectura_tarjeta (){
Serial.print("UID:");
for (byte i = 0; i < mfrc522.uid.size; i++)
    {  // Bucle recorre de a un byte por vez
        if (mfrc522.uid.uidByte[i] < 0x10)
        {             // Si el byte leido es menor a 0x10
          Serial.print(" 0"); //Se imprime un espacio en blanco y numero cero
        }
        else
        {
          Serial.print(" "); //Se imprime un espacio en blanco
        }
          Serial.print(mfrc522.uid.uidByte[i], HEX);  // Imprime el byte del TAG leido en hexadecimal
          LecturaUID[i] = mfrc522.uid.uidByte[i]; //Almacena byte del TAG leido
    }
      Serial.print("\t");  //Se imprime un espacio de tabulacion
}
boolean comparaUID(byte lectura[], byte usuario[]) // Función comparaUID
{
  for (byte i = 0; i < mfrc522.uid.size; i++) { //Bucle recorre de a un byte por vez
    if (lectura[i] != usuario[i])  //byte leído es distinto al del usuario
      return (false); //Si los 4 bytes no coinciden  retorna falso
  }
  return (true);      //Si los 4 bytes coinciden retorna verdadero
}
```

Figura 3.43: Lectura de tarjeta e identificación del usuario

Subrutina <Bienvenido ( )>

Una vez identificado el usuario, se inicia con la subrutina <Bienvenido>, esta consiste en cambiar la variable <Usuario> de *string* a *char*, esto con la finalidad de enviar esta información a la pantalla *Nextion* mediante la función < tu.setText (name1)>. Esto se observa en la Figura 3.44.

```
void Bienvenido (){
   pagel.show(); //Se muestra la página 0
   pbuzzer (); //Se inicia función pbuzzer
   const char* namel ="";
   int namel_len = usuario.length() + 1; //Length (calcula la extencion de la cadena )
   char char_array_usuario [namel_len];//Se crea variable char con el numero de caracteres leído
   usuario.toCharArray(char_array_usuario,namel_len); //Se convierte los datos de char a array
   namel = char_array_usuario; //igualamas el valor char
   tu.setText(namel);//Se visualiza el nombre del usuario en la pantalla
   delay (3000);
}
```

Figura 3.44: Función Bienvenido ()

Subrutinas <Leer_ultrasonico ()>

Posteriormente, se procedió con la detección de las manos del usuario, esto con la finalidad de iniciar con el proceso de lavado de manos. Para ello se creó la subrutina <leer_ultrasonico>, dicha subrutina se compone del condicional "*do-while*".

En la parte del < do {} >, se inicia mostrando en la pantalla *Nextion* la página 2, esto se lo realizó mediante la instrucción < page2.show()>, después se procede con la lectura de los pines *trigPin* y *echoPin*. Estos pines se utilizaron para medir el retardo que presenta la onda que emite el sensor ultrasónico, este dato es almacenado en la variable <tiempo> para posteriormente realizar el cálculo de la distancia mediante la ecuación <distancia= (tiempo/2) /29.15> [24]. En el caso de que las manos del usuario se encuentren a una distancia menor a 20 cm se emite un sonido mediante el *buzzer* indicando que fueron detectadas. La programación de este proceso se puede ver en la Figura 3.45.

```
void leer_ultrasonico (){
   do {
   page2.show();
   digitalWrite(trigPin, LOW); //Recibimiento pulso.
   delayMicroseconds(5);
   digitalWrite(trigPin, HIGH); // Envio pulso.
   delayMicroseconds(10);
   digitalWrite(trigPin,LOW);
   tiempo = pulseIn(echoPin, HIGH); //Fórmula para medir pulso entrante
   distancia = (tiempo/2)/29.15; //Fórmula para calcular distancia
   }while (distancia >= 20); //Condición distancia es >= 20cm
   pbuzzer ();
   estado =1;
}
```

Figura 3.45: Función leer_ultrasonico ()

Subrutina Control_Circuito ()

Una vez realizada la subrutina < leer_ultrasonico () >, se inicia la función < Control Circuito () >. Dicha función está estructurada mediante la estructura de control *switch*. Esto se visualiza en la Figura 3.46.

```
void Control_Circuito (){
 switch (estado){

  case 1: /// Proceso de lavado
  Mojar_manos ();
  Dispensador_jabon ();
  Proceso_lavado ();
  Enjuagar ();
  Temperatura ();
  break;

  case 2: // temperatura correcta
  Temperatura_correcta ();
  break;

  case 3: // temperatura incorrecta
  Temperatura_incorrecta ();
  break;
 }
}
```

Figura 3.46: Función Control Circuito ()

Dicha función arranca con la lectura de la variable estado, para después iniciar con los distintos casos correspondientes. Como en la subrutina < leer_ultrasonico () > el valor de estado era iguala uno, se inicia el caso 1.

El caso 1 está conformado por 5 subrutinas las cuales se detallan a continuación:

- Subrutina <Mojar_manos () >, esta función se encarga de dos tareas, la primera es mostrar la pagina 3 en la pantalla *Nextion,* mediante el uso de la instrucción <page3.show() > y la segunda se trata de controlar el encendido y apagado de la bomba de agua R385, la bomba permanecerá encendida durante 3 segundos y después se apagará. Esto se visualiza en la Figura 3.47.

```
void Mojar_manos () {
  page3.show();
  digitalWrite (grifo, HIGH);
  delay (2000);
  digitalWrite (grifo,LOW);
  delay (1000);
}
```

Figura 3.47: Mojar_manos ()

- Subrutina <Dispensador_jabon ()> ver Figura 3.48, esta función se encarga igual que la anterior del control de una bomba de agua R385, específicamente la que distribuye el jabón antibacterial. La bomba se encenderá durante 1 segundo y se apagará. Además, se muestra en la pantalla *Nextion* la página 4 mediante la instrucción <page4.show() >.

```
void Dispensador_jabon () {
  page4.show();
  digitalWrite (jabon, HIGH);
  delay (1000);
  digitalWrite (jabon, LOW);
}
```

Figura 3.48: Dispensador_jabon ()

- Subrutina <Proceso_lavado ()>, esta función realiza la transición de imágenes, en la cuales se muestra el proceso de lavado de manos, dichas imágenes son mostradas en la pantalla *Nextion* mediante la instrucción <pageX. show>, además se coloca un retardo en función de la información que se visualiza en la pantalla, esto con la finalidad de que el usuario pueda tener el tiempo necesario para visualizar las imágenes. Esto se observa en la Figura 3.49.

```
void Proceso_lavado () {
  page5.show();
  delay (2500);
  page6.show();
  delay (4000);
  page7.show();
  delay (3000);
  page8.show();
  delay (4000);
  page9.show();
  delay (4000);
  page10.show();
  delay (4000);
}
```

Figura 3.49: Proceso_lavado ()

- Subrutina <Enjuagar ()>, básicamente esta función realiza el control de la bomba R385, encargada de manejar el flujo de agua. Esto con el objetivo de que el usuario termine con el proceso de lavado de manos. Esto se visualiza en la Figura 3.50.

```
void Enjuagar () {
  page11.show();
  digitalWrite (grifo, HIGH);
  delay (8000);
  digitalWrite (grifo,LOW);
}
```

Figura 3.50: Enjuagar ()

- La subrutina <Temperatura ()>, está formada por la estructura de control "*do-while*", una vez terminado el proceso de lavado de manos, se procede a medición de la temperatura corporal del usuario. Para ello primero se tiene que detectar la muñeca de este, por tanto, se utilizó un sensor infrarrojo para que detecte la

muñeca del usuario e inicie la medición mediante el sensor MLSX9014, posteriormente emite un sonido mediante el *buzzer* indicando que se realizó la medición. Después, se coloca la variable <estado> con el valor 2 si la temperatura del usuario es normal y 3 en el caso que la temperatura del usuario sea mayor a 37 °C. Dicho código se puede observar en la Figura 3.51.

```
void Temperatura () {
  do{
      page12.show(); //Detecta la muñeca del usuario
      infrarojo1=digitalRead(sensor_infrarojo);
  }while(infrarojo1==LOW);
  temp_persona = 6.45 + mlx.readObjectTempC();
  delay (1000);

  if (temp_persona <= temp_normal)
    {
      temp_usuario = temp_persona;
      pbuzzer ();
      estado = 2; //Estado 2 temp_correcta
      delay (1000);
    } else if (temp_persona >= temp_normal)
      {
      temp_usuario = temp_persona;
      pbuzzer ();
      estado = 3; //Estado 3 temp_incorrecta
      delay (1000);
      }
}
```

Figura 3.51: Subrutina Temperatura ()

Una vez realizada la medición de la temperatura al usuario, se procede a iniciar de nuevo la subrutina <Control_Circuito ()>. Después, se analiza el valor de la variable estado, en el caso de que estado sea igual a 2 se inicia con la función <Temperatura_correcta ()>. En cambio, sí estado es igual 3 se iniciará con la función <Temperatura_incorrecta ()>. Los funcionamientos de dichas funciones se describen a continuación.

- Subrutina <Temperatura_correcta ()>, en esta función se inicia con la conversión de la variable <temp_usuaio> de *float* a *char*, esto se lo hizo con la finalidad de mostrar en la pantalla *Nextion* la temperatura del usuario (solo permite mostrar variables del tipo *char*). Para enviar la información a la pantalla se hace uso de la instrucción <page13.show()> y de la instrucción <ti.setText(conversión)>. La programación se observa en la Figura 3.52.

```
void Temperatura_correcta(){
   char conversion[10];
   dtostrf(temp_usuario, 5, 2, conversion);
   //Se convierte a un número entero con dos décimas
   page13.show();
   tc.setText(conversion);
   delay (5000);
}
```

Figura 3.52: Subrutina Temperatura_correcta ()

- Subrutina <Temperatura_incorrecta ()>, esta función es similar a la anterior con la diferencia que se inicia con la función <Enviar_email>. Dicha función se detalla más adelante. La programación se observa en la Figura 3.53.

```
void Temperatura_incorrecta (){
  char conversion[10];
  dtostrf(temp_usuario, 5, 2, conversion);
  //Se convierte a un número entero con dos décimas
  page14.show();
  ti.setText(conversion);
  delay (5000);
  Enviar_email ();
}
```

Figura 3.53: Subrutina Temperatura_incorrecta ()

Subrutina <Enviar_*email* ()>

En el caso que la variable estado sea igual a 2, se inicia la subrutina <Enviar_*email* ()>. Dicha subrutina consiste en enviar y recibir datos mediante la comunicación serial entre las placas Arduino y ESP8266.

Para el envío del mensaje con el nombre del usuario y su temperatura, se procedió a estructurar los datos en formato *JSON* mediante la instrucción < StaticJsonDocument<300> *doc*>. Por tanto, se declararon dos objetos, uno para almacenar la temperatura < temperatura_usuario> y el otro para guardar el nombre del usuario < mensaje>. Posteriormente, se visualiza en la pantalla *Nextion* la página 15, mediante la instrucción < page15.*show*()>.

Para la recepción del mensaje, el cual indica si el envío del mensaje fue exitoso, se procedió a usar la estructura de control "*while*". Esto con la finalidad de recibir datos mediante la comunicación serial, después se almacena la *string* recibida en la variable <*inBuffer*> y se procede a la deserialización. Finalmente se muestra en la pantalla *Nextion* la página 16 mediante la instrucción < page16.show()>, en la cual indica que el usuario debe esperar instrucciones. Esta programación se visualiza en la Figura 3.54.

```
void Enviar_email (){
   page15.show();
   String json;
   temperatura_usuario = temp_persona ;
   mensaje = usuario;

   StaticJsonDocument<300> doc;
   doc["Usuario"].set(mensaje);
   doc["Temperatura"].set(temperatura_usuario);;

   serializeJson(doc, json);
   Serial3.println(json);
   Serial.println(mensaje);
   page15.show();
   delay(5000);

while (Serial3.available()>0)
 {
   page16.show();
   String inBuffer = Serial3.readString();
   StaticJsonDocument<200> doc;
   DeserializationError error = deserializeJson(doc, inBuffer);
   if (error) { return; }
   String  respuesta = doc["Respuesta"];
   Serial.print("Estado del correo:");
   Serial.println(respuesta);
   page16.show();
   delay (5000);
 }
}
```

Figura 3.54: Comunicación con el módulo ESP8266

- **Programación del módulo de comunicación ESP8266**

Para la programación del módulo *Wi-Fi* se tomó en cuenta los siguientes puntos:

- Inclusión de librerías.
- Conexión a la red *LAN*.
- Recepción de datos de la placa Arduino.
- Envío de datos al servidor.
- Envío de datos a la placa Arduino

Inclusión de librerías

Para la comunicación con la placa de Arduino, primero se configuró la comunicación serial, esto se realizó mediante la librería <*SoftwareSerial.h*>. Adicionalmente, se procedió a configurar los pines D3 y D2 como transmisor y receptor respectivamente. Esto se visualiza en la figura 3.55.

```
#include <SoftwareSerial.h> // Se incluye libreria para poder usar puertos seriales digitales
SoftwareSerial NodeMCU(D3,D2); // Se establece comunicación serial mediante los pines D3(TX) Y D2 (RX)
```

Figura 3.55: Librería - Comunicación serial

Para utilizar el módulo *Wi-Fi* en modo cliente, se tuvo que incluir dos librerías. La primera <ESP8266WiFi.h>, permite utilizar el módulo ESP8266 y la librería <ESP8266HTTPClient.h>, la cual permite configurar en modo cliente el módulo *Wi-Fi*. Esto se visualiza en la Figura 3.56.

```
#include <ESP8266WiFi.h> // Se incluye libreria módulo ESP8266
#include <ESP8266HTTPClient.h> // Se incluye libreria para usar el modulo como cliente
```

Figura 3.56: Librería - Módulo ESP8266

Para el envío y recepción de información en forma de texto plano, se incluyó la librería <ArduinoJson.h>. Esta librería permite enviar peticiones al servidor, mediante el formato *JSON*. Esto se observa en la Figura 3.57.

```
#include <ArduinoJson.h> // Se incluye la libreria JSON para poder enviar y recibir datos en formato JSON
```

Figura 3.57: Librería – *JSON*

Conexión a la red *WLAN*

Para la conexión con la red *WLAN*, es necesario configurar las credenciales de acceso a esta. Por tanto, se crearon dos variables tipo *char,* <ssid> permite almacenar el nombre de la red a la cual se va a conectar y <*password*> almacena la contraseña de la misma. Esto se observa en la Figura 3.58.

```
const char* ssid = "VIRUS_GRATIS"; //"SSID_DE LA RED" a usar
const char* password = "201510130JAVI"; //"PASSWORD_ DE LA RED"
```

Figura 3.58: Credenciales de la red

En la parte del *setup,* se envían las credenciales para la conexión con la red, esto mediante la instrucción <WiFi.begin (ssid, password)>. Posteriormente, en el bucle *while* se verifica que la conexión fue establecida, una vez que la conexión es exitosa se continua con el programa. Esto se visualiza en la Figura 3.59.

```
WiFi.begin(ssid, password);   //WiFi connection
while (WiFi.status() != WL_CONNECTED) {  //Esperar a que se complete la conexión WiFI
delay(500);
Serial.println("Waiting for connection");
}
```

Figura 3.59: Conexión con la red

Recepción de datos de la placa Arduino

En la parte del *loop*, se procedió con la lectura de la comunicación serial, esto mediante la instrucción <*NodeMCU.readString* ()>, adicionalmente se almacenaron los datos recibidos en la variable tipo *string* <inBuffer>. Posteriormente, se declaró el objeto *JSON* denominado *"doc"* mediante la instrucción < *StaticJsonDocument*<200> doc.

Todo esto se lo realizó para realizar la desearilización de la *string* recibida mediante la instrucción <deserializeJson (doc, inBuffer)>. Finalmente, se procedió a almacenar los valores de la temperatura y nombre del usuario en las variables <nombre> y <temp>, respectivamente. En caso de que este proceso resultara erróneo, se retorna a la parte del *loop*. Esto se visualiza en la Figura 3.60.

```
String inBuffer = NodeMCU.readString(); // Se lee la trama del mensaje
StaticJsonDocument<200> doc; // Se establece longitud del mensaje que se recibe
DeserializationError error = deserializeJson(doc, inBuffer); //Deserialización
if (error) { return; }
String nombre = doc["Usuario"];//variable que contiene la información del usuario
float temp = doc["Temperatura"];//variable que contiene la información de la temperatura
```

Figura 3.60: Deserialización de la *string* recibida

Envío de datos al servidor

Para realizar la petición con la plataforma IFTTT, se creó cuatro variables. La variable <url>, la cual contiene el formato con toda la información necesaria para realizar la petición, mientras que los tres restantes, se encargan de almacenar la información del hosting: <host>, <eventName> y <apiKey>. Esto se observa en la Figura 3.61.

```
String apiKey = "jDoEzQSAdJ7ul6Mz0v_n02SSZV-MMH-dulXcVIfw0JQ"; // Clave de nuestro hosting IFTTT
String host = "http://maker.ifttt.com";
String eventName = "email_alarm";
String url = host + "/trigger/" + eventName + "/with/key/" + apiKey; // Concatenación de variables
```

Figura 3.61: Credenciales para la conexión con el servidor

Después, se declaró el objeto *JSON* doc, esto mediante la instrucción <StaticJsonDocument<300> doc>, dicho objeto se utilizó para almacenar los valores serializados de la temperatura y el nombre del usuario. Después, se establece la variable <nombre> y <temp>, para almacenar el nombre del usuario y el valor de su temperatura. Esto se observa en la Figura 3.62.

```
StaticJsonDocument<300> doc; // Se establece longitud del mensaje
doc["valuel"].set(nombre); // Se establece variabe nombre
doc["value2"].set(temp); // Se establece variabe temperatura
```

Figura 3.62: Variables para enviar la *string* en formato *JSON*

Posteriormente para enviar la información a la plataforma IFTT, se declaró la variable tipo *char* <OutputJSONMessageBuff [64] >, dicha variable se utilizó para establecer la longitud que va a tener la *string* con los valores serializados. Después, se procedió con la serialización teniendo en cuanta la longitud, esto se lo realizó mediante la instrucción <serializeJson (doc, OutputJSONMessageBuff, sizeof(OutputJSONMessageBuff))>. Esto se visualiza en la Figura 3.63.

```
char OutputJSONMessageBuff[64]; //Se establece el buffer de la trama a enviar
serializeJson(doc, OutputJSONMessageBuff, sizeof(OutputJSONMessageBuff));
// Se establece condiciones de la trama para la serializacion
Serial.println(OutputJSONMessageBuff); //Se imprime las variables a enviar
```

Figura 3.63: Serialización de la *string JSON*

Una vez serializados los datos, se estableció el destino para la conexión con la plataforma IFTTT. Para ello se declaró el objeto de clase HTTP cliente, luego se especificó el destino de solicitud mediante la instrucción < *http. Begin* (url)>, adicionalmente se configuró el encabezado que va a tener la petición mediante la instrucción <*http.addHeader* ("*Content-Type*", "*application/json*")>. Esto se observa en la Figura 3.64.

```
HTTPClient http;    //Se declarara objeto de clase HTTP Client
http.begin(url);     //Se especifica el destino de la solicitud
http.addHeader("Content-Type", "application/json"); // Se especifica el encabezado
```

Figura 3.64: Configuración con el servidor

Una vez establecido el destino, se procedió al envió del correo con los datos del usuario. Primero, se envió los datos mediante el método *POST*, esto se lo realizó mediante la instrucción <int httpCode = http.POST (OutputJSONMessageBuff)>. Después, se recibió la respuesta del servidor mediante el método *GET*, para ello se utilizó la instrucción < String payload = http. getString ()>. Finalmente, se cerró la conexión http mediante la instrucción <http.end ()>. Esto se visualiza en la Figura 3.65.

```
int httpCode = http.POST(OutputJSONMessageBuff);   //Se envia request
String payload = http.getString();                  //Se recibe el GET
Serial.println(httpCode);   //Se imprime código de retorno HTTP
Serial.println(payload);    //Respuesta del Request payload
http.end();  //Se cierra la conexión
```

Figura 3.65: Envió y recepción *POST – GET*

Envió de datos a la placa arduino

Una vez enviado el correo, se procedió a indicar a la placa de Arduino que el correo fue enviado, esto por medio del mensaje “Enviado”. Para esto se creó el objeto *JSON root*, mediante la instrucción < StaticJsonDocument<200> root>. Adicionalmente, se declaró

las variables tipo *string* <respuesta> y <respuesta_esp8266>. Una vez configurados los parámetros necesarios se procede a la serialización mediante la instrucción <serializeJson (root, respuesta_esp8266)>. Finalmente, se envía a la placa arduino por medio de la instrucción < *NodeMCU*.println(respuesta_esp8266)>. Esto se visualiza en la Figura 3.66.

```
String respuesta_esp8266; // Se establece variable tipo string para la respuesta
respuesta = "Enviado";
StaticJsonDocument<200> root; // Se establece languitus del mensaje tipo JSON
root["Respuesta"].set(respuesta); // Se estable las variables de la String Json
serializeJson(root, respuesta_esp8266); // Se procede a la serialización
NodeMCU.println(respuesta_esp8266);
Serial.println(respuesta_esp8266); // Se imprime el mensaje en el monitor serial
```

Figura 3.66: Serialización del mensaje "Enviado"

Tanto el diagrama de flujo de la placa Arduino como del módulo *Wi-Fi* ESP8266, se encuentran en la parte de Anexos como Anexo III.

- **Diseño del circuito impreso**

Para solventar este requerimiento, se optó por usar el programa *Proteus* ya que es una herramienta de *software* de simulación y diseño desarrollada *por Labcenter Electronics* para sistemas eléctricos y diseño de circuitos electrónicos.

Una vez seleccionados los componentes que forman parte del proyecto se procedió al diseño del PCB. El cual tiene como función facilitar la conexión de los distintos elementos electrónicos con la placa de Arduino, por tanto, se procedió al diseño del circuito impreso, teniendo en cuenta la distribución de pines del Arduino y los distintos niveles de voltaje que requieren los sensores.

Para iniciar, se procedió con el diseño de las conexiones mediante el *software Proteus* específicamente la herramienta ISIS, para ello se utilizaron las librerías de Arduino para simular la placa del Arduino MEGA 2560, esto con el fin de visualizar la posición de los pines. Esto se puede visualizar la Figura 3.67.

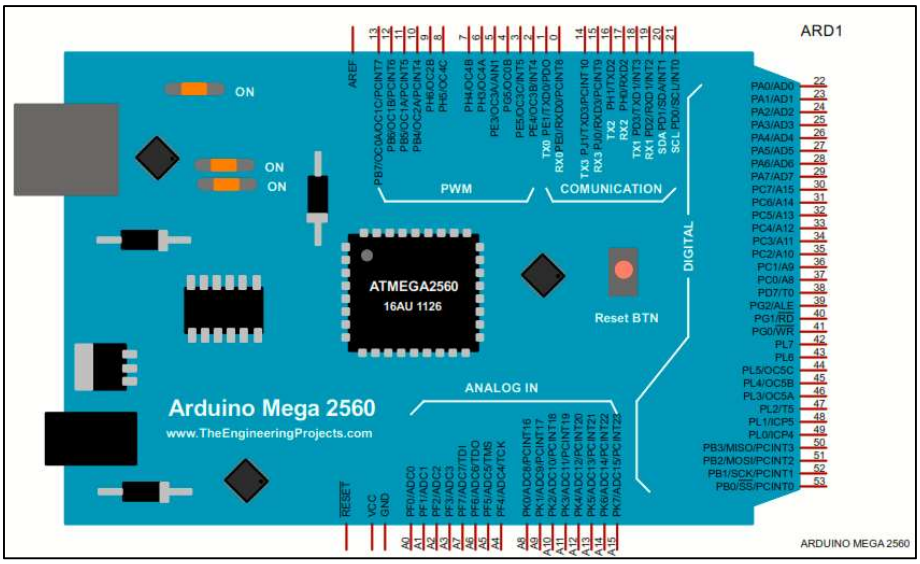

Figura 3.67: Simulación placa Arduino Mega 2560

Después, se procedió con la colocación de *jumpers* para la conexión de los sensores, esto con la finalidad de facilitar la instalación en la estructura del lavabo y realizar un correcto ruteo de cables, evitando que la placa Arduino se vea afectada por interferencias electromagnéticas. Esto se puede observar en la Figura 3.68.

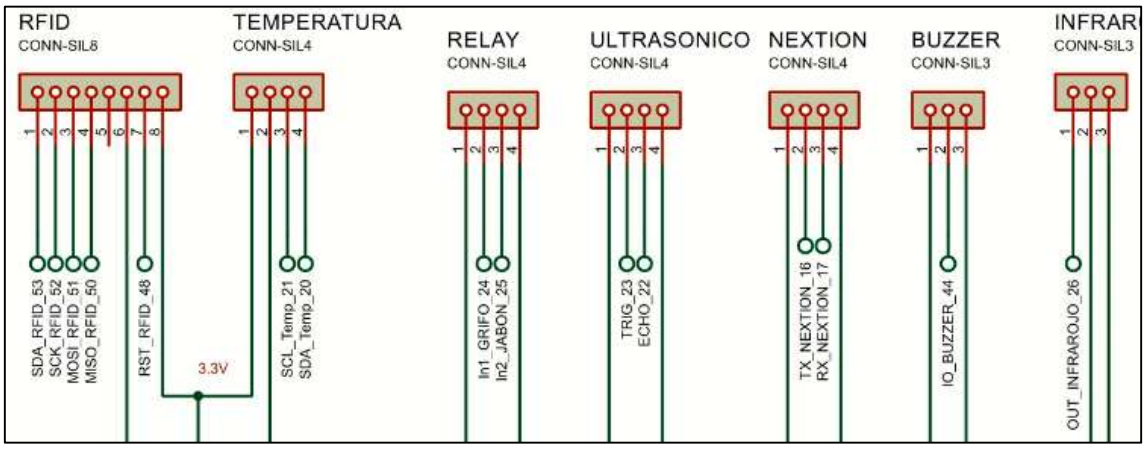

Figura 3.68: Borneras - Conexión de sensores

Posteriormente, para la parte de la alimentación del prototipo se utilizaron 8 borneras. Las cuales se distribuyeron de la siguiente manera, para la alimentación de las bombas de agua se usaron 4 borneras, en cambio para la alimentación del prototipo teniendo en cuenta los distintos niveles de voltaje (12v, 5v y 3v) se utilizaron 3 borneras más. Adicionalmente, para que todos los elementos se encuentren conectados a una tierra en común se utilizó una bornera extra y finalmente para alimentar a la placa de Arduino 2 *jumpers*. Dichas conexiones se pueden observar en la Figura 3.69.

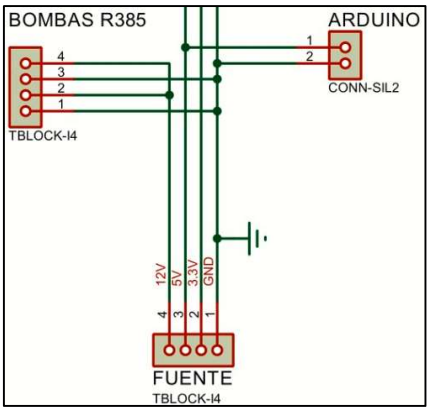

Figura 3.69: Jumpers de alimentación

Para que el módulo *Wi-Fi* pueda comunicarse se manera serial con la placa de Arduino, se utilizaron cuatro jumpers, dos destinados a la comunicación serial y las otras dos para alimentación del módulo *Wi-Fi*. Las borneras sobrantes tienen como finalidad que el módulo ESP8266 se mantenga fijo en el PCB, evitando un funcionamiento erróneo del mismo. Esto se observa en la Figura 3.70.

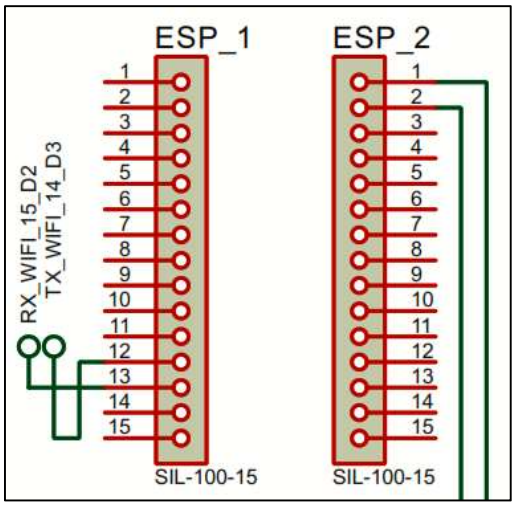

Figura 3.70: ESP8266 - Conexiones

Realizar la simulación que muestre la interacción entre todos los elementos no es posible a través del programa ISIS 8.5, ya que algunos de los elementos no se encuentran en el *software*. Por tal motivo las pruebas de funcionamiento fueron desarrolladas durante la marcha

Finalmente, para realizar la conexión entre el Arduino y el PCB se utilizó una bornera de 18 pines, el esquema lógico completo se lo visualiza a continuación en la Figura 3.71.

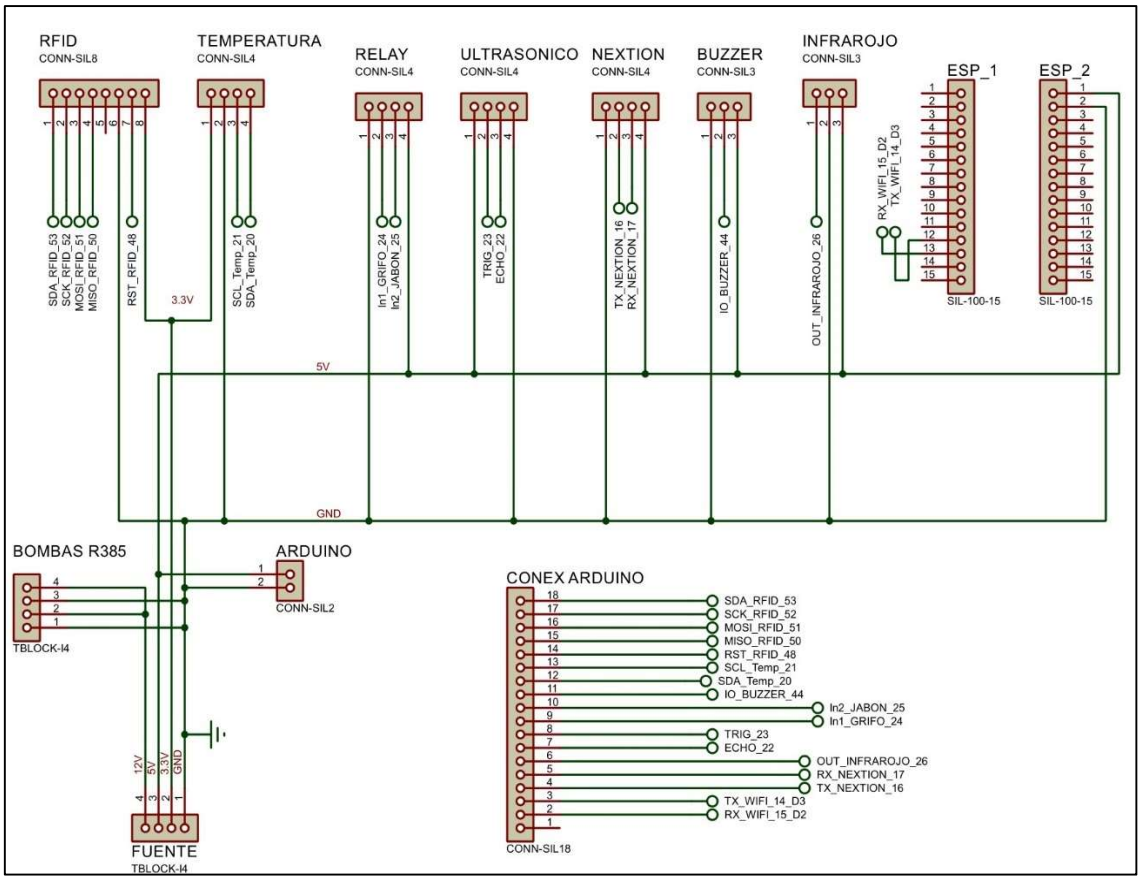

Figura 3.71: Esquema lógico completo del PCB

Diseño de la PCB con el uso de la herramienta ARES

Una vez finalizado el esquemático del circuito, se procede al diseño del circuito impreso, esta parte consiste en definir las dimensiones que va a tener la placa electrónica, además de ubicar los componentes teniendo en cuenta optimizar el espacio en la placa.

En primer lugar, se define la forma que va a tener el diseño de la PCB. En este caso hay que tener en cuenta que: el PCB va a ir instalado en un *case* de 18cmx45cm, y además la ubicación de los componentes del prototipo. Esto con el fin de conseguir un ruteo adecuado de los cables.

Adicionalmente, la ubicación de la fuente de alimentación en la estructura del prototipo es de mucha importancia, con la finalidad de evitar que esta provoque un mal funcionamiento en el prototipo. Por tanto, para ubicar los jumpers en el PCB encargado de la alimentación del prototipo, se tuvo en cuenta las consideraciones antes mencionadas. Finalmente, se procedió al ruteo de las pistas teniendo en cuenta no realizar ningún corto que pueda afectar el correcto funcionamiento del PCB. El diseño de las pistas se lo puede visualizar en la Figura 3.72, adicionalmente Ares permite tener

una vista 3D del PCB creado, esto fue de gran ayuda en el proceso de soldadura de los componentes.

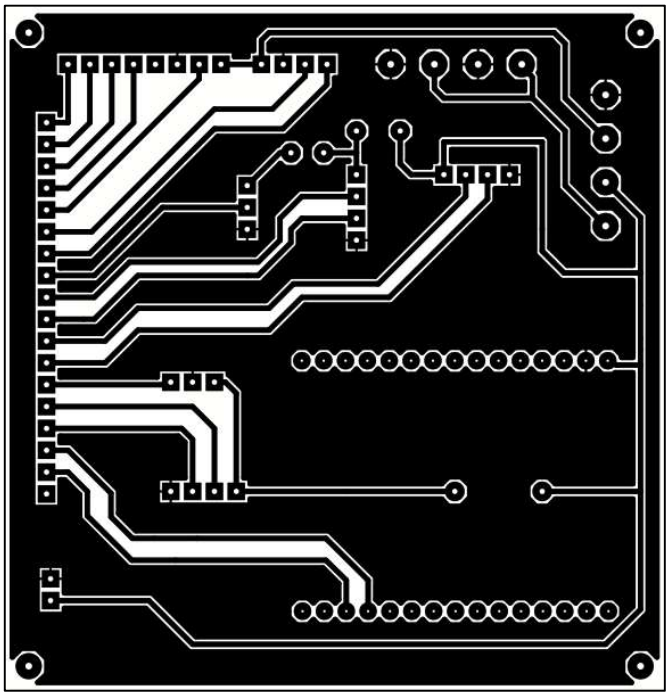

Figura 3.72: Diseño de las pistas del prototipo

Fabricación de la placa electrónica

Una vez obtenido el diseño del PCB, se procedió con la soldadura de los distintos componentes electrónicos que conforman la placa, para lo cual se realizaron los siguientes pasos:

- Se usó papel termotransferible para la impresión del diseño del PCB, se usó este tipo de material ya que transfiere las pistas a la superficie de la baquelita, aplicando calor sobre ella.
- Impreso el circuito, se ubicó el papel termotransferible sobre la baquelita y con la ayuda del calor de la plancha se transfiere el circuito impreso a la baquelita.
- Una vez las pistas se impregnaron a la baquelita, se procedió a sumergirla en percloruro férrico, ya que este líquido se encarga de remover el excedo del cobre.
- Se procede a limpiar la tinta del circuito con el objetivo de dejar las pistas de cobre al descubierto, además se precedió a verificar que no exista ningún tipo de defecto en las mismas y con un taladro se realizaron los agujeros para ubicar los componentes en la placa.

- Con los componentes colocados en su lugar, se procedió a soldar los mismos a la placa con ayuda de un cautín, estaño y pasta para soldar, esto último con la finalidad de que el proceso de soldar sea sencillo.
- Finalmente, se comprueba la continuidad de las pistas con ayuda de un multímetro.

En la Figura 3.73, se muestra la distribución de los distintos elementos su funcionalidad y descripción de los elementos.

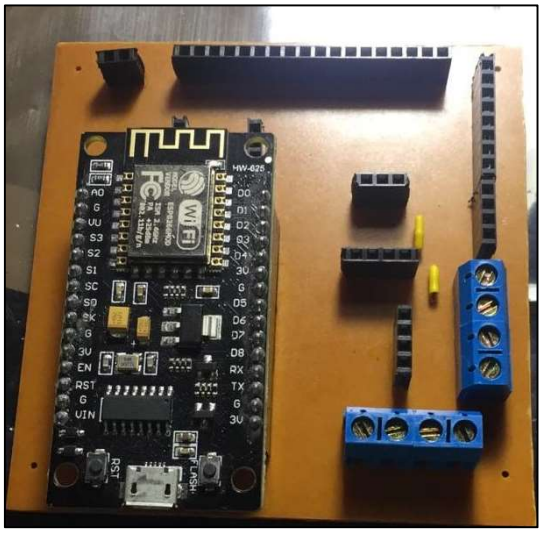

Figura 3.73: Circuito Impreso

3.4 Estructura del prototipo

Para el diseño de la estructura del proyecto se utilizó el *software* AutoCAD, el cual es una herramienta que maneja cualquier dibujo 2D. Para la elaboración de la estructura del lavamanos existe una norma establecida por el Servicio Ecuatoriano de Normalización (INEN), esta norma es la numero 3082 la cual fue modificada en 2018 y no ha sido alterada hasta la presente fecha. La norma está estipulada para la construcción de aparatos sanitarios en la cual se tiene algunos requisitos y métodos necesarios para la construcción. En esta norma también consta el estándar para la elaboración de un lavamanos, este varía según las necesidades del ocupante existen lavamanos extragrandes y otros minis, pero las medidas estándar son aproximadamente de 45-50 cm de fondo y 45-60 cm de ancho. En cuanto a la altura en que se debe colocar, la ideal es de 85 a 90 cm del suelo.

La estructura fue elaborada con láminas de tol y pintada con pintura electrostática, este procedimiento se usa en la elaboración de muebles de oficina. Esto garantiza que el lavamanos sea de excelente calidad y tenga una gran durabilidad en su estructura física.

Para evitar el contacto con el metal y que exista un cortocircuito en los dispositivos electrónicos, la base donde estos van colocados está hecha de madera la cual evita que exista algún tipo de contacto. La estructura fue diseñada con el estándar establecido por la norma del INEN y se tiene lo siguiente:

En la Figura 3.74, se puede observar que se cumple con el estándar INEN ya que la estructura tiene 50 cm de fondo y 45 cm de ancho. Es importante recalcar que el fondo en este caso es 55 cm ya que existen 5 cm de espacio para poder colocar el cableado necesario para alimentar el circuito y también las mangueras empleadas para extracción y expulsión de los desechos de agua.

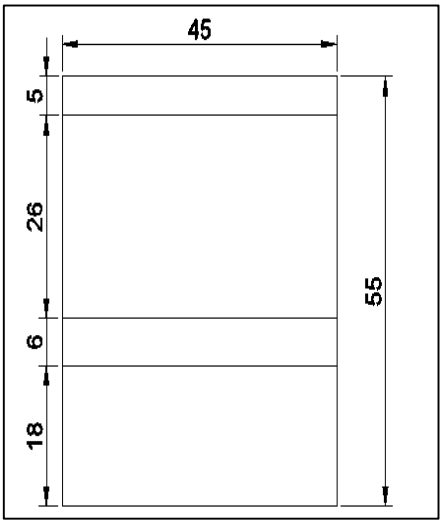

Figura 3.74: Estructura del prototipo - Vista superior

En la Figura 3.75, se puede observar la altura en la que se encuentra el depósito del lavamanos es 90 cm desde el suelo, por ello cumple con el estándar, el tamaño total de la estructura es de 140 cm ya que la parte electrónica está ubicada en la parte superior.

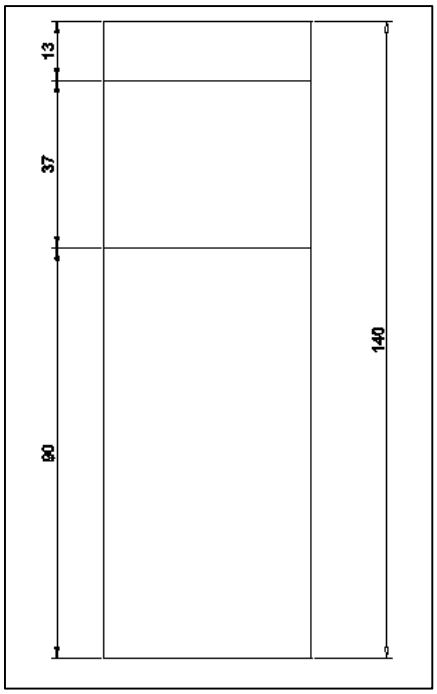

Figura 3.75: Estructura del prototipo - Vista frontal

En la Figura 3.76, se puede apreciar el diseño del prototipo, en la parte superior tiene una ligera inclinación la cual permite que se visualice la pantalla con mayor comodidad para la persona que esté usando el prototipo.

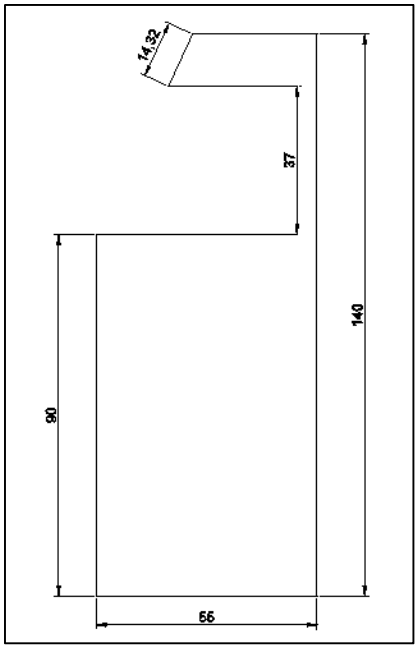

Figura 3.76: Estructura - Vista lateral

Finalmente, en la Figura 3.77 tiene la vista en una perspectiva isométrica donde se observa la estructura del lavamanos y el resultado final, se aprecia que es una estructura bastante sólida y segura.

En la parte superior se encuentran protegidos los circuitos electrónicos, y en la parte inferior se encuentra los depósitos de agua para abastecer el lavamanos.

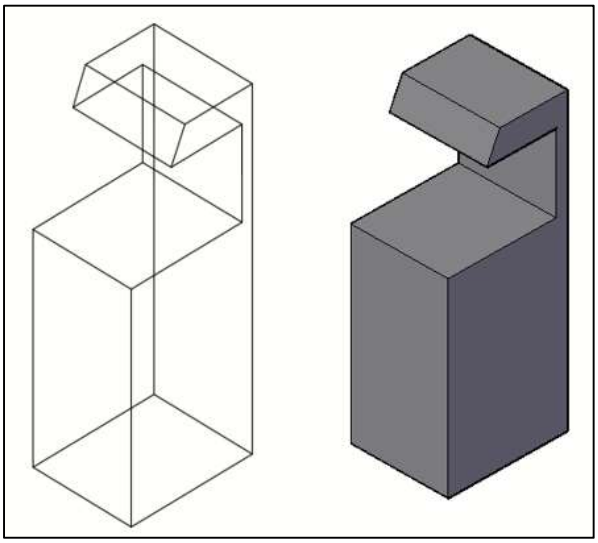

Figura 3.77: Estructura - Perspectiva isométrica

3.5 Implementación del Prototipo

Concluido el proceso de diseño, construcción y programación, se procede a la implementación del prototipo, teniendo en consideración lo siguiente:

- **Fabricación de la estructura**

Para la fabricación de la estructura se recibió la colaboración de la fábrica de muebles metálicos CARDEL quienes fabricaron la estructura con las siguientes características, ver Figura 3.78:

- **Estructura metálica a base de tol:** para este proceso se llevó a cabo el corte del metal, luego se realizó el proceso de doblado para dar forma al mueble, seguidamente se hicieron los cortes necesarios y finalmente se inició el proceso de soldadura para tener una estructura firme.
- **Pintura electrostática:** para este primero se realizó el lavado del mueble con desoxidante para quitar impurezas de la estructura, seguidamente se procedió a pintar con pintura en polvo, la cual mediante el proceso electrostático se adhiere al mueble para finalmente hornear el mueble y finalizar este proceso.

Figura 3.78: Estructura metálica del lavamanos

- **Instalación de los sensores**

Cabe recalcar que todo el mecanismo electrónico fue instalado en la parte superior del lavamanos. El sensor ultrasónico se instaló junto a las bombas de agua, esto con la finalidad de detectar las manos de los usuarios con facilidad. Esto se ubica en la Figura 3.79.

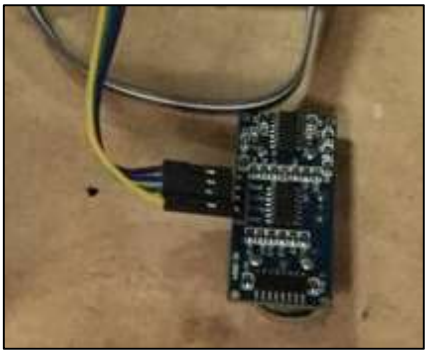

Figura 3.79: Instalación sensor ultrasónico

El lector RFID, fue ubicado en parte superior del tablero con la finalidad de que el usuario puede visualizarlo con facilidad. Esto se muestra en la Figura 3.80.

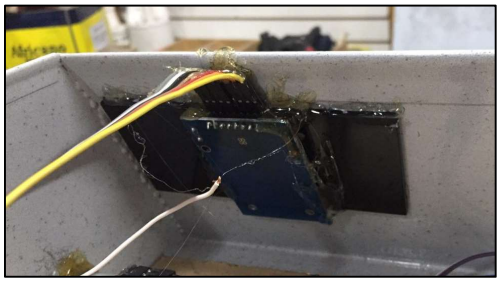

Figura 3.80: Instalación lector RFID

El sensor de temperatura fue colocado en la parte lateral derecha del lavamanos junto a él se ubicó el sensor infrarrojo. Esto se observa en las Figura 3.81.

Figura 3.81: Instalación sensor temperatura / infrarrojo

En la parte izquierda va instalado el *buzzer* para así facilitar los sonidos que este proporciona. Esto se visualiza en la Figura 3.82.

Figura 3.82: Instalación *buzzer*

- **Instalación de las bombas de agua.**

Las bombas de agua fueron instaladas separadas del mecanismo electrónico para evitar daños en las placas, adicionalmente con la ayuda del circuito impreso se conectaron las tierras de las bombas con las tierras de los demás elementos electrónicos. Esto se observa en la Figura 3.83, cabe recalcar que el ruteo de los cables se encuentra realizado más adelante.

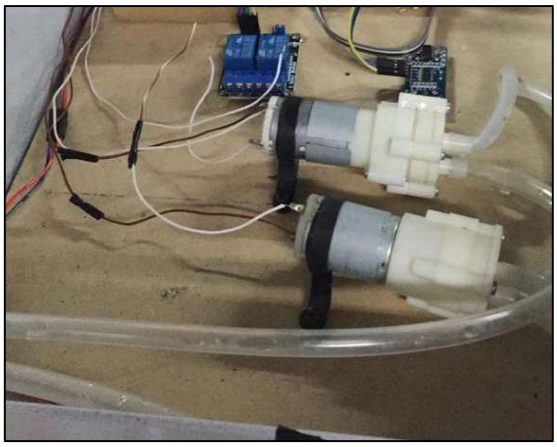

Figura 3.83: Instalación bombas de agua

- **Instalación de la pantalla *Nextion.***

La pantalla *Nextion* se instaló en la parte frontal del lavamanos, esto con la finalidad de que sea visible para el usuario, de esta manera permite observar las indicaciones a seguir para el proceso de lavado de manos. Esto se observa en la Figura 3.84.

Figura 3.84: Instalación pantalla *Nextion*

- **Instalación del mecanismo electrónico**

El circuito impreso, Arduino mega y los relés fueron instalados relativamente cerca entre ellos, esto con la finalidad de facilitar la conexión de cables de manera estética y protegerlo así de cualquier daño con agua o sobrecalentamiento de la fuente. Figura 3.85, cabe recalcar que el correcto ruteo de los cables se encuentra realizado más adelante.

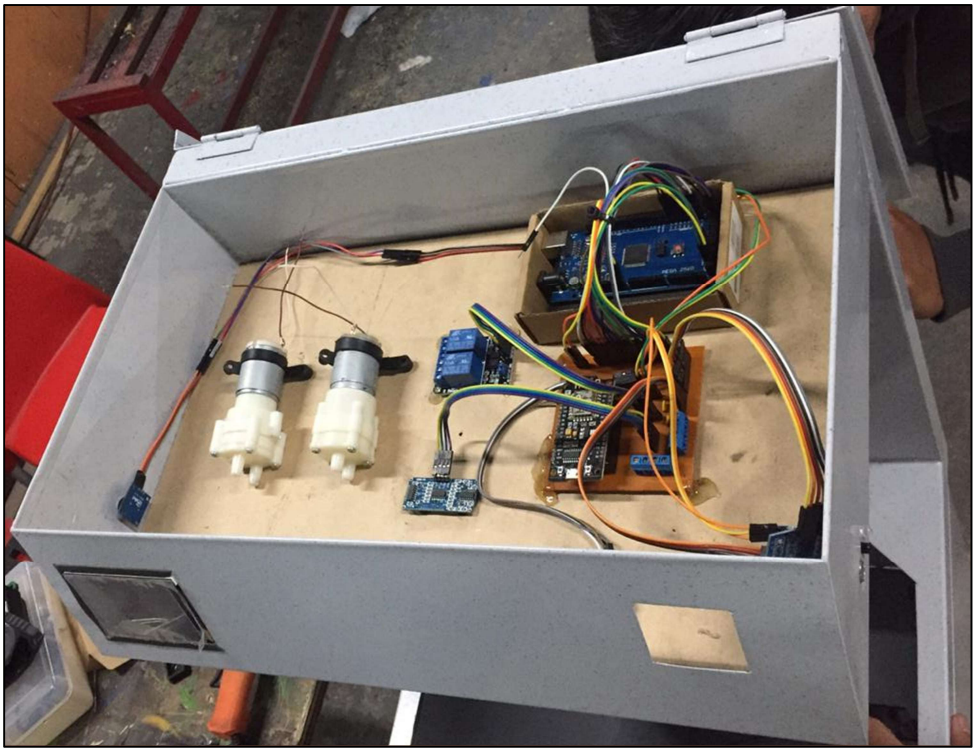

Figura 3.85: Instalación mecanismo electrónico

- **Instalación de la fuente de alimentación**

La fuente de alimentación fue instalada en la parte inferior de la estructura para evitar algún daño en el circuito electrónico y también permite distribuir de mejor manera el espacio de la estructura. Figura 3.86.

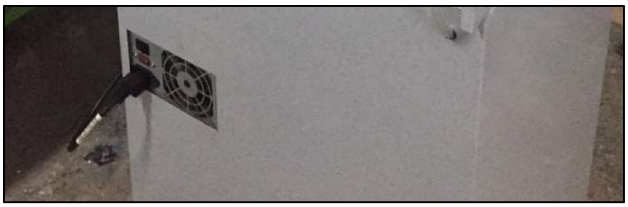

Figura 3.86: Instalación fuente de alimentación

3.6 Pruebas de funcionamiento

Para minimizar los posibles errores del prototipo, se recomienda realizar una inspección general del sistema instalado, esto con la finalidad de identificar averías en los elementos del prototipo.

La revisión se inició con las conexiones del circuito impreso, que no existan cortocircuitos en las pistas de la placa y que el cableado que se extiende desde la placa hacia los componentes electrónicos se encuentre en óptimas condiciones.

Adicionalmente, se verificó que exista conectividad con la red *WLAN*. Culminado la inspección, se realizaron pruebas de funcionamiento, estas pruebas constan de los siguientes puntos.

- Pruebas del sistema de automatización.
- Pruebas del envío de notificaciones.

- **Pruebas del sistema de automatización**

Con respecto al sistema de automatización, se realizaron pruebas a los distintos procesos que debe realizar el sistema entre las más importante se destacan:

Control de usuarios

Se realizó la lectura de los *tags* o etiquetas de cada usuario, obteniendo como resultados que el lector tenía problemas con la lectura de los *tags* ya que presentaba interferencias con la estructura del lavamanos. Para solucionarlo se modificó la abertura del espacio donde se encuentra instalado el lector RFID, para proteger el módulo RFID de la humedad se colocó sobre el mismo papel aislante. Esto se observa en la Figura 3.87.

Figura 3.87: Lector RFID

Control de las bombas de agua

Se realizó el control de las bombas R385, para esto se utilizó un relé de dos canales con el fin de activar y desactivar dichas bombas en intervalos cortos de tiempo, como resultado se obtuvo que al desactivar la primera bomba R385 encargada de distribuir el jabón antibacterial, esta producía que el Arduino Mega se reinicié.

Para solucionarlo se optó por cambiar los relés por dos módulos *MOSFET* LRF520N, ya que el problema que presentaba el relé se debía a la parte mecánica específicamente en su bobina ya que al momento de desactivar el relé este producía corrientes parasitas, las cuales afectaban el correcto funcionamiento del Arduino Mega.

Adicionalmente las conexiones que van desde la fuente de alimentación hacia las bombas fueron recubiertas con material aislante, esto con la finalidad de minimizar interferencias electromagnéticas, ver Figura 3.88.

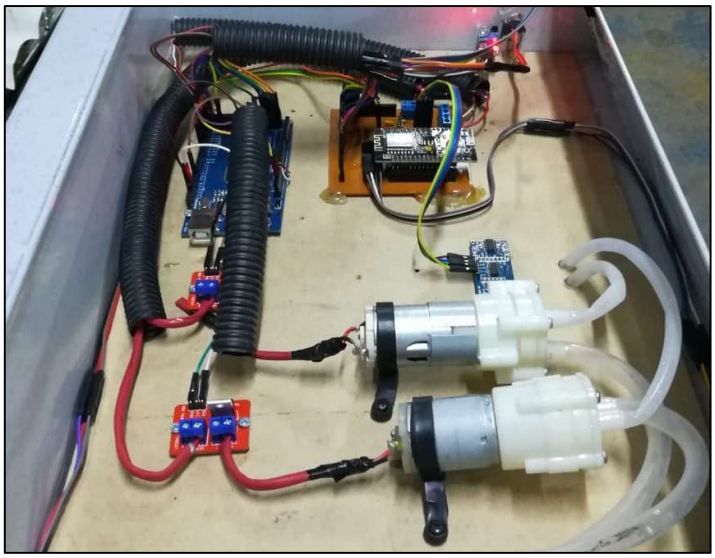

Figura 3.88: Bombas R385

Medición de temperatura corporal

Para verificar el correcto funcionamiento de la temperatura corporal, se realizó la comparación entre las lecturas tomadas por el sensor y las de un termómetro digital infrarrojo. Como resultado se obtuvo que el sensor presentaba errores de funcionamiento, ver Figura 3.89.

Para solucionarlo se procedió a quitar el protector ya que este provocaba lecturas erróneas del sensor. Realizado esto se volvió a comparar las lecturas del sensor con el termómetro infrarrojo, verificando que este presentaba una variación de +/- 0.2 °C, solventando el problema. Esto se puede observar en la Figura 3.90.

```
17:56:07.927 -> Connection established!
17:56:07.927 -> IP address:     192.168.43.132
17:56:57.308 -> Usuario encontrado: Dario Trujillo
17:56:57.308 -> Temperatura es: 1053.65
```

Figura 3.89: Lectura errónea del sensor de temperatura

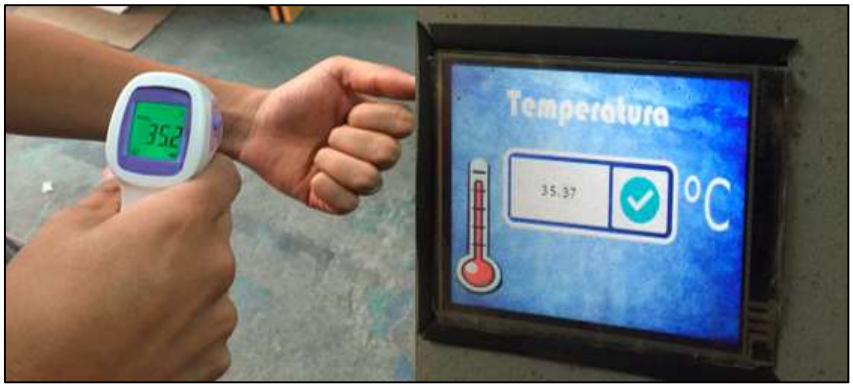

Figura 3.90: Calibración del sensor MLX90614

- **Pruebas del envío de notificaciones**

Envió y recepción del correo electrónico

Mediante el monitor serial de Arduino IDE y con la cuenta de correo electrónico, se verificó el envío y recepción de la alarma en forma de notificación, para simular el caso en el que un usuario presente fiebre se utilizó una pistola de calor para obtener una temperatura mayor a 37 °C.

Como resultado se obtuvo que el sistema encargado de enviar y recibir los correos electrónicos funcionó acorde lo esperado. Esto se observa en la Figura 3.91

Figura 3.91: Notificación de un usuario presenta fiebre

Una vez realizadas las distintas pruebas de funcionamiento del prototipo, se determinó que el prototipo se encuentra trabajando de manera correcta acorde a los requerimientos encontrados.

3.7 Costo del proyecto

En la Tabla 3.14 se observa el precio de cada uno de los componentes que se utilizaron para el desarrollo del proyecto. Dichos valores se extrajeron de la plataforma Mercado Libre.

Tabla 3.14: Costo del proyecto.

Detalle de material	Cantidad	Precio	Total
Módulo Arduino Mega	1	$20,00	$20,00
Módulo *Wi-Fi* ESP8266	1	$8,00	$8,00
Módulo *Mosfet* LRF520N	2	$3,00	$6,00
Panta *Nextion* 3.5''	1	$50,00	$50,00
Sensor MLX90614	1	$30,00	$30,00
Sensor HC-SR04	1	$2,50	$2,50
Módulo RFID	1	$3,00	$3,00
Etiqueta RFID	2	$1,00	$2,00
Bomba de agua R385	2	$9,00	$18,00
Buzzer	1	$2,00	$2,00
Estructura del prototipo	1	$150,00	$150,00
Espadines Hembra-macho	60	$3,00	$3,00
Fuente de poder	1	$15,00	$15,00
Mano de obra	1	$200,00	$200,00
Total			$509,50

4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- Con la elaboración del proyecto, se demuestra que es importante trabajar en medidas de prevención en este caso el correcto lavado de manos. Esta medida reduce el riesgo de contagio de cualquier tipo de enfermedad transmitida por contacto directo de las manos con cualquier tipo de superficie, razón por la cual se justifica la creación de este proyecto, puesto que, como objetivo primordial mejorar la calidad de vida de las personas.

- Una vez analizados los requerimientos del prototipo, se procedió a determinar los componentes tanto para la parte del *software* y *hardware*. Uno de los aspectos más importantes fue la elección del ecosistema Arduino, ya que, por la simplicidad, costos y su capacidad de incluir módulos externos, hizo que Arduino facilite trabajar con cualquier tipo de sensor, siendo la mejor elección para el proyecto en comparación con el ecosistema *Raspberry*.

- El funcionamiento del presente proyecto se basa en minimizar el contacto del usuario con superficies que pueden ser fuentes de contagio. Esto se logró mediante el uso de sensores ultrasónicos para detectar la presencia de las manos del usuario, válvulas para apertura/cierre del flujo de agua y para dispensar jabón, una interfaz

gráfica para mostrar mediante imágenes el correcto proceso de lavado de las manos. Además, incluye funcionalidades extra como medir la temperatura del usuario mediante el sensor infrarrojo y enviar una notificación por correo electrónico en caso de superar el rango de temperatura normal a través del módulo *Wi-Fi*.

- El presente proyecto cumple con el objetivo de construir un lavamanos inteligente portátil, dando como resultado que los usuarios realicen de manera correcta el proceso de lavado de manos, logrando mantener unos hábitos mínimos de higiene y así prevenir ciertas enfermedades de fácil transmisión originadas por virus y bacterias.

- Se concluyó que la programación tanto de la placa Arduino como del módulo Wi-Fi en base a subrutinas, es de gran ayuda al momento de realizar cualquier tipo de modificación en el código fuente, ya que, facilitan la lectura y comprensión de este. Además, se evita la repetición de líneas código y favorecen la reutilización de este, ya que, esta puede ser invocada varias veces dentro del código fuente.

- El proceso para la fabricación de la placa electrónica fue sencillo, debido a la utilización del *software Proteus*, puesto que, permitió la organización y simulación tanto física como lógica de los distintos elementos del proyecto.

- Se desarrolló la documentación necesaria de todo el proyecto, los diagramas de flujo del prototipo (Anexo III), el código fuente empleado (Anexo I y II) y el manual de mantenimiento de este (ANEXO IV), esto con la finalidad de facilitar la identificación de errores, el mantenimiento preventivo, correctivo y la ampliación e integración de futuras mejoras al prototipo implementado.

- Una vez realizado las pruebas de funcionamiento se optó por el uso de transistores *Mosfet* LRF520N en lugar del relé de dos canales, ya que estos no presentan elementos mecánicos que puedan causar fallas en el microcontrolador. Además, para evitar interferencias electromagnéticas se realizó un adecuado ruteo de cables.

4.2 RECOMENDACIONES

- Cuando se realizó la conexión del cableado de la fuente para alimentar a los dispositivos, se obtuvo un error el cual provocaba el reseteo del microcontrolador,

debido a que se conectaban todos los dispositivos a una misma línea de alimentación, por tanto, se recomienda alimentar el Arduino Mega por separado de los otros dispositivos así se garantiza el correcto funcionamiento del prototipo.

- Es recomendable aislar el módulo RFID RC522 de cualquier superficie metálica. Esto con la finalidad de evitar lecturas erróneas, puesto que, en el proyecto al instalar dicho módulo en la estructura se obtuvo interferencia con el metal que constituye gran parte del prototipo.

- Se recomienda separar la fuente de energía del resto de los dispositivos, ya que, esta puede provocar interferencias electromagnéticas llegando a causar daños en los dispositivos más delicados, por ello, se optó por colocar dicha fuente en la parte inferior del lavamanos.

- Cuando el módulo Wi-Fi tiene problemas al conectarse a la red, se puede deber a un mal ingreso de caracteres, es decir, se escribió mal el nombre de la red o su contraseña, debido a esto se recomienda verificar que tanto el nombre de la red como su contraseña estén escritos de manera correcta, así se evitara cualquier tipo de error de conexión.

5. REFERENCIAS BIBLIOGRAFICAS

[1] O. Heptro, «HEPTRO - Módulo RFID-RC522 RF con Arduino UNO SPI,» 25 Abril 2014. [En línea]. Available: https://hetpro-store.com/TUTORIALES/modulo-lector-rfid-rc522-rf-con-arduino/. [Último acceso: 19 Agosto 2020].

[2] Ingenima, «Evaluandoerp - Comparación entre tecnologías de RFID y código de barras.,» [En línea]. Available: https://www.evaluandoerp.com/comparacion-tecnologias-rfid-codigo-barras/#:~:text=Un%20código%20de%20barras%20no,una%20tecnología%20de%20solo%20lectura.&text=Esto%20significa%20que%20solo%20se,puede%20realizar%20múltiples%20lecturas%20simultáneas. [Último acceso: 05 Diciembre 2020].

[3] I. B. Herrero, «IEROBOTICS - Módulo Relé 5V 2 canales optoacoplados, salida 10A/250VAC,» Abril 2018. [En línea]. Available: https://www.iberobotics.com/producto/modulo-rele-5v-2-canales-optoacoplado/. [Último acceso: 19 Agosto 2020].

[4] L. LLAMAS, «ALARMA CON ARDUINO Y BUZZER ACTIVO (ZUMBADOR),» 01 Agosto 2016. [En línea]. Available: https://www.luisllamas.es/arduino-buzzer-activo/. [Último acceso: 19 Agosto 2020].

[5] A. Electronics, «AV Electronics - Mini Bomba de Agua de Membrana R385,» Agosto 2016. [En línea]. Available: https://avelectronics.cc/producto/mini-bomba-de-agua-sumergible-de-membrana-r385/#:~:text=Bomba%20de%20diafragma%20R385%2C%20es,permiten%20el%20bombeo%20del%20liquido.. [Último acceso: 19 Agosto 2020].

[6] A. Electronics, «AV Electronics - Pantalla Nextion 3.5,» Abril 2018. [En línea]. Available: https://avelectronics.cc/producto/pantalla-nextion-3-5/. [Último acceso: 19 Agosto 2020].

[7] Prometec, «PROMETEC - LOS SENSORES INFRARROJOS,» 2018. [En línea]. Available: https://www.prometec.net/siguelineas-ir/. [Último acceso: 19 Agosto 2020].

[8] L. LLAMAS, «MEDIR DISTANCIA CON ARDUINO Y SENSOR DE ULTRASONIDOS HC-SR04,» 16 Junio 2015. [En línea]. Available: https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/. [Último acceso: 19 Agosto 2020].

[9] L. LLAMAS, «ARDUINO Y EL TERMÓMETRO INFRARROJO A DISTANCIA MLX90614,» 29 Octubre 2016. [En línea]. Available:

https://www.luisllamas.es/arduino-y-el-termometro-infrarrojo-a-distancia-mlx90614/. [Último acceso: 19 Agosto 2020].

[10] Arduino.cl, «Arduino.cl - ¿Qué es Arduino?,» 2018. [En línea]. Available: https://arduino.cl/que-es-arduino/. [Último acceso: 19 Agosto 2020].

[11] Complubot, «Complubot - Guía para la selección de una controladora Arduino,» 26 Febrero 2015. [En línea]. Available: https://complubot.com/inicio/guia-para-la-seleccion-de-una-controladora-arduino/. [Último acceso: 19 Agosto 2020].

[12] Sandorobotics, «Módulo NodeMCU V3 ESP8266 – CH340,» 2019. [En línea]. Available: https://sandorobotics.com/producto/hr0126/. [Último acceso: 19 Agosto 2020].

[13] R. Alonso, «Raspberry Pi vs Arduino, ¿en qué se diferencian y para qué se usan?,» 03 Abril 2020. [En línea]. Available: https://hardzone.es/reportajes/comparativas/raspberry-pi-vs-arduino/. [Último acceso: 05 Diciembre 2020].

[14] A. Aqeel, «Introduction to Arduino IDE,» 03 Octubre 2018. [En línea]. Available: https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html. [Último acceso: 20 Agosto 2020].

[15] C. Today, «Circuits Today - Proteus PCB Design and Simulation Software – Introduction,» Octubre 2017. [En línea]. Available: https://www.circuitstoday.com/proteus-software-introduction. [Último acceso: 20 Agosto 2020].

[16] A. Patole, «Nextion Display with Arduino – Getting Started,» 08 Diciembre 2017. [En línea]. Available: https://randomnerdtutorials.com/nextion-display-with-arduino-getting-started/. [Último acceso: 20 Agosto 2020].

[17] Study.com, «Study.com - What Is AutoCAD?,» 28 Agosto 2019. [En línea]. Available: https://study.com/what_is_auto_cad.html. [Último acceso: 20 Agosto 2020].

[18] E. Oswald, «Digitaltrends - If this, then that: A beginner's guide to the wonderful world of IFTTT,» 30 Julio 2016. [En línea]. Available: https://www.digitaltrends.com/cool-tech/what-is-ifttt-and-how-does-it-work/. [Último acceso: 20 Agosto 2020].

[19] C. Hope, «Computer Hope - Inkscape,» 06 Abril 2019. [En línea]. Available: https://www.computerhope.com/jargon/i/inkscape.htm. [Último acceso: 30 Agosto 2020].

[20] L. G. S., «Teoría de Redes de Computadores,» Abril 2017. [En línea]. Available: https://www.oas.org/juridico/spanish/cyber/cyb29_computer_int_sp.pdf. [Último acceso: 20 Agosto 2020].

[21] S. Electronic, «WiFi Standards 802.11a/b/g/n,» 16 Julio 2014. [En línea]. Available: https://www.semiconductorstore.com/blog/2014/WiFi-standards-802-11a-b-g-n-vs-802-11ac-Which-is-Best/806/. [Último acceso: 30 Agosto 2020].

[22] D. No, «ESPloradores - Comunicaciones GET y POST,» 11 Mayo 2018. [En línea]. Available: https://www.esploradores.com/practica-15-comunicaciones-get-y-post/. [Último acceso: 30 Agosto 2020].

[23] R. d. einformatica, «Repuestosdeinformatica - COLORS IT EN60950 480W,» 2015. [En línea]. Available: https://repuestosdeinformatica.com/fuente-de-alimentacion/3234-colors-it-en60950-480w.html. [Último acceso: 10 Diciembre 2020].

[24] M. Dominguez, «Sensor de distancia - Fórmula,» 01 Diciembre 2013. [En línea]. Available: https://soloarduino.blogspot.com/2013/07/sensor-de-distancias.html. [Último acceso: 10 Diciembre 2020].

[25] O. M. d. l. Salud, «OMS,» OMS, 1 Octubre 2010. [En línea]. Available: https://www.who.int/gpsc/5may/tools/es/. [Último acceso: 1 Mayo 2020].

[26] L. CÓRDOBA, «Universidad de Cordoba,» 7 Julio 2015. [En línea]. Available: file:///C:/Users/Master/Downloads/[PDF]%20Documentacion%20Completa%20Grifo%20Automatizado_compress%20(1).pdf. [Último acceso: 1 Mayo 2020].