UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Aplicación para el control de acceso en edificios

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Javier Alejandro Rocha Herrera

ASESOR DE INFORME

Dr. Edmundo Gabriel Rocha Cózatl

Ciudad Universitaria, Cd. Mx., 2024

Índice general

Capítulo 1

Introducción y objetivos

1.1. Introducción

La seguridad y la regulación de accesos en edificios es muy importante, dado a que permite autentificar y autorizar que los usuarios puedan ingresar a ciertas áreas, durante y dentro de un cierto intervalo de tiempo. También es posible llevar a cabo un monitoreo de zonas mediante el uso de cámaras, tener un registro de accesos, salidas, tiempo de estadía de los usuarios, etc. Todo ello permite que se tenga un mayor control y orden dentro de las construcciones y edificios, convirtiéndolos en lugares más seguros y confiables. Lo cual es de suma importancia, ya que ayuda a prevenir siniestros en instituciones como bancos o edificios de gobierno, así como también, facilita la logística de accesos en aeropuertos. El control de acceso está relacionado con las políticas y mecanismos que permiten o prohíben el uso de algún recurso o la capacidad de realizar/desempeñar una actividad [34].

El sistema básico de control de acceso consta de los siguientes elementos:

- Dispositivo de entrada (Lectora, Keypad o biométrico): Decodifica la información de entrada del sistema y la envía traducida al panel. [35]
- Panel (controlador): Tiene la funcionalidad de autentificar a los usuarios. Permite o prohíbe el acceso a los usuarios a ciertas áreas mediante la comparación de la información recibida de los dispositivos de entrada con respecto con los datos guardados en su memoria obtenidos de la base de datos del servidor. [35]
- Servidor: Contiene toda la información relativa al control de acceso como nombres de usuarios, identificadores ID, números de tarjeta, fechas de ingreso y salida de los usuarios, etc. [35]
- Dispositivo de bloqueo/liberación de la puerta: Puede ser un electroimán o una cerradura electromagnética. Su función principal es asegurar o liberar la puerta, para que ésta pueda ser abierta o permanecer cerrada. [35]
- Sensor: Permite obtener la información del estado de la puerta. [35]
- Dispositivo de salida (Lectora, sensor, botón): Es un dispositivo de entrada de información al sistema, manda información al panel, para que un usuario pueda salir de un área. [35]

Durante mi participación profesional tuve la oportunidad de solucionar problemas, implementar mejoras y darle mantenimiento, mediante el uso de la programación, a una aplicación que tenía como objetivo la administración del control de accesos a edificios. Trabajar con ésta solución, me dio la oportunidad de desempeñar actividades tanto en el área de software, como en la de hardware. Esto me permitió aprender más a profundidad programación orientada a objetos, lenguajes para el gestionamiento de bases de datos, redes y circuitos electrónicos.

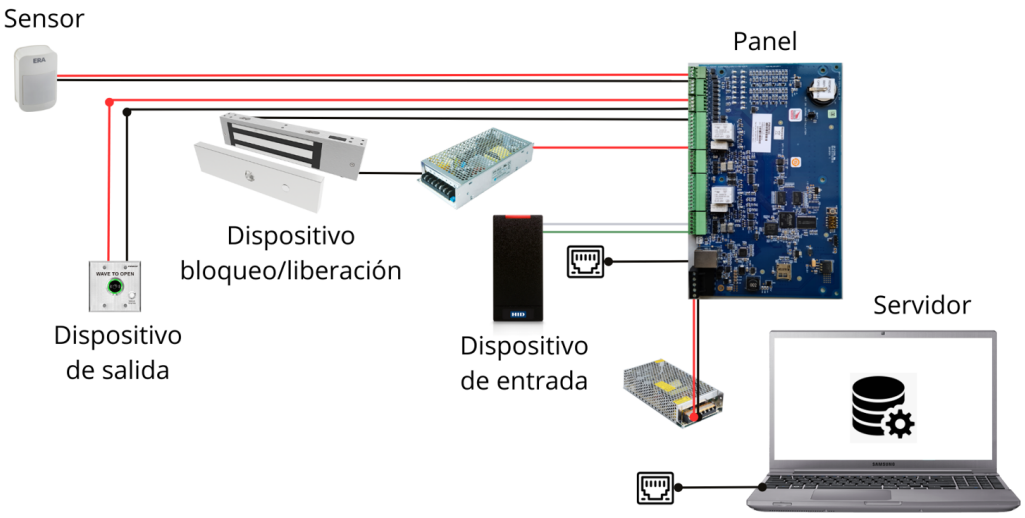

Figura 1.1: Diagrama de las conexiones de los elementos básicos del sistema para el control de acceso. [29]

La finalidad de éste trabajo, es exponer como las actividades realizadas durante mi participación en el entorno laboral contribuyeron en mi formación profesional y exponer como los conocimientos adquiridos, durante mi formación como ingeniero en la Facultad de Ingeniería, me brindaron las bases teóricas y prácticas necesarias para desempeñarme de la mejor manera posible dentro de un entorno de trabajo, y de ésta forma, poder resolver problemas y contribuir en la solución de las áreas de oportunidad dentro de mi equipo.

1.2. Objetivos

- Desarrollar un programa, que ayude a decodificar la información de las tarjetas que los usuarios deslizan a través una lectora. Con el fin de obtener el número de sitio y de tarjeta, propios de cada una de ellas; para así poder imprimir y visualizar dicha información en una pantalla.
- Desarrollar un programa que ayudara a automatizar las pruebas del deslizamiento de tarjetas sobre las lectoras, que sea controlado de forma remota.

Capítulo 2

Descripción de la empresa

2.1. Historia de la empresa

Los inicios de Honeywell se remontan al año de 1885, cuando Albert Butz funda una pequeña empresa llamada "Butz Thermo-Electric Regulator C.O.", tras inventar y patentar su Regulador de horno y alarma (predecesor de los termostatos modernos).
Posteriormente, con el tiempo, el pequeño negocio de Butz fue absorbido por la compañía "Temperature Controlling C.O.", y en el año de 1893 cambio de nombre a " Electric Heat Regulator C.O".
En 1916, cinco años después, " Electric Heat Regulator C.O"fue comprada y renombrada por W.R Sweatt como "Minneapolis Heat Regulator company".Esta nueva compañía expandió sus líneas de producción y patentó el primer motor eléctrico aprobado por " Underwriters Laboratories".
Mientras eso sucedía, en 1906 el ingeniero Mark Honeywell había fundado una empresa dedicada a hacer generadores para calentar agua llamada "Honeywell Heating Specialty Co.".
Varios años después, en 1927 " Minneapolis Heat Regulator Company " y " Honeywell Heating Specialty Co."se fusionaron y se formó la empresa " Minneapolis-Honeywell Regulator Co". Ésta nueva empresa creció rápidamente e incursionó en otros campos de la ingeniería, como en la industria de control industrial e indicadores, convirtiéndose en la productora más grande de relojes enjoyados de alta calidad.
En el año 1969, cuando estaba en auge la " Carrera Espacial", Honeywell contribuyó con la fabricación de 16,000 diferentes partes para misión del Apollo 11, las cuales se encontraban distribuidas entre 14 diferentes dispositivos electrónicos que hacían posible la estabilización y el sistema de control (SCS).
En el año 1985 las compañías llamadas " Allied" y " Signal Companies"se fusionaron dando pie al surgimiento de la empresa " AlliedSignal", el principal giro de dichas empresas estaba enfocado en el sector aerospacial, automotor e ingeniería en materiales. Con el tiempo, en 1990 Minneapolis-Honeywell fue absorbida por ésta nueva empresa, AlliedSignal, quien decidió conservar el nombre de Honeywell por el prestigio que había ganado la compañía. Juntas ambas empresas compartieron una gran cantidad de conocimiento e intereses en distintos negocios como el aerospacial, productos químicos, partes de automóviles y controladores de edificios. Para cuando el siguiente siglo llegó, Allied ya había adquirido las compañías de " Union Texas Natural Gas " y la empresa de giro automotriz y aerospacial "Bendix Corp".
Desde inicios del siglo XXI, Honeywell sigue creciendo y expandiéndose adquiriendo diferentes negocios a través de diferentes industrias. En los últimos años Honeywell se a fusionado con varias empresas. Por ejemplo, en 2005 " Universal Oil Products"fue adquirida, la cual era una empresa internacional líder en suplir petróleo refinado, procesador de gas, producción petroquímica, y una gran empresa manufacturera. Tres años después, se adquirió la empresa de " Metrologic Instruments", que era encargada de hacer láseres, imágenes móviles (ultrasonidos, X-ray y electrocardiogramas) y scanners remotos.
En 2016 Honeywell adquirió Intelligrated, cuya empresa se encarga de brindar soluciones de automatización para centros de distribución y depósitos industriales.
En 2018 Honeywell segrego a su compañía Resideo como una compañía independiente que cotiza en la bolsa, pero ésta aún es la encargada de crear productos de la marca registrada Honeywell, de la misma manera su negocio en sistemas de transporte se segrego como Garret Motion. Finalmente, en el año 2019, Honeywell lanzó su plataforma de Software " Honeywell Forge"para la administración y

el manejo de los negocios de la empresa.[1]
Actualmente Honeywell tiene 5 unidades de negocios principales:

- Aeroespacial
- Honeywell Building Technologies (HBT)
- Safety and Productivity Solutions (SPS)
- Performance Materials and Technology (PMT)
- Honeywell Connected Enterprise(HCE)

Figura 2.1: Primer regulador de horno inventado por Albert Butz. [2]

2.2. Descripción del puesto de trabajo

Dentro de mis actividades laborales tuve la oportunidad de desempeñar y mejorar mis habilidades dentro de las áreas de conocimiento de la programación y la electrónica, en el departamento de seguridad de edificios (Control de acceso). Mis principales tareas consistieron en aplicar mis conocimientos de programación en C++, C# y SQL junto con mi equipo para darle suporte y mantenimiento a la aplicación de seguridad de la empresa, así como también utilizar mis conocimientos para realizar mejoras y nuevas implementaciones. Dado a que muchas veces era necesario replicar los ambientes de los clientes para realizar pruebas, también me fue posible poder tener contacto con servidores, redes, así como a los sistemas embebidos con placas STM32. Algunas actividades de desarrollo en las que participé fueron en un decodificador de números de tarjetas de acceso con una IDE de programación. En dónde aprendí el funcionamiento del protocolo de comunicación Wiegand; en el desarrollo de un emulador IoT embebido, con ayuda de una placa STM32, con el fin de automatizar pruebas remotamente y como última actividad, realicé documentación y colaboré en el establecimiento de nuestro laboratorio de pruebas.

2.3. Organigrama

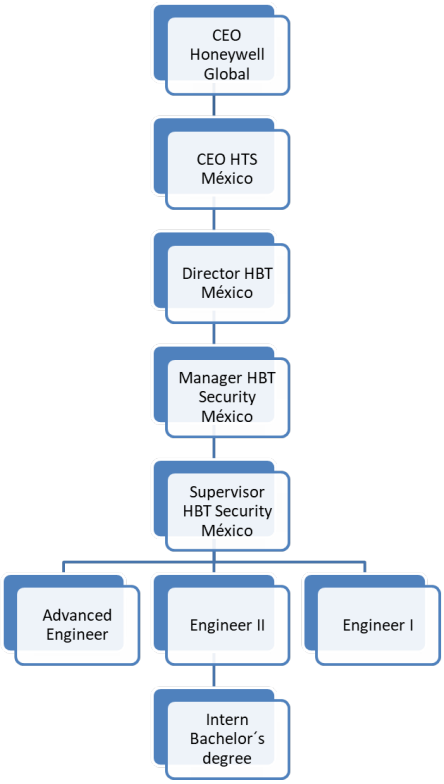

Figura 2.2: Organigrama de los puestos de trabajo dentro de la empresa. [36]

Capítulo 3

Antecedentes

Mediante el uso de la programación es posible establecer procedimientos lógicos en forma instrucciones para poder definir, controlar el comportamiento de dispositivos o procesos bajo ciertas condiciones o eventos, así como, manejar grandes cantidades de información, analizarla, procesarla y utilizarla al beneficio de la humanidad. Sin embargo, para desempeñar dichas actividades es necesario cumplir con normas y principios de seguridad para proteger la integridad de las personas en todo momento, así como la información de los usuarios.
Por otro lado, la electrónica, nos ayuda a transmitir, recibir y ejecutar dichas instrucciones mediante el envío de señales y el uso de protocolos de comunicación. Ésta rama de la ingeniería fue muy valiosa dentro del proyecto, dado a que con ella fue posible establecer un vínculo con el mundo físico, para poder llevar a cabo toda la lógica e instrucciones programadas. De la misma forma hizo posible obtener entradas de información para nuestro sistema mediante el uso de sensores y dispositivos periféricos, así como, ejecutar acciones con la ayuda de actuadores. En este proyecto el Hardware se encontraba altamente relacionado con el software, y gracias a ambos nos era posible cumplir con el objetivo primario del control de accesos.
Gracias al crecimiento acelerado de la tecnología y diversos factores, actualmente han surgido nuevos campos de conocimiento derivados de varias ramas de la ingeniería. Dichos campos han servido para maximizar, mejorar la solución de problemas y obtener mejores resultados, tal es el caso de la tecnología IoT, el uso y almacenamiento en la nube, así como la inteligencia artificial. En mi experiencia laboral tuve un acercamiento a cada uno de ellos en diferentes proporciones, sin embargo, en éste trabajo se hablará más acerca de la tecnología IoT.

3.1. Antecedentes teóricos del decodificador Wiegand

3.1.1. Protocolo Wiegand

Mediante el protocolo Wigand es posible trasmitir Bits de información. Para ello, se utilizan dos cables de datos "data 0 " y "data 1".
Para trasmitir información, los cables deben ponerse en estado bajo (LOW). Es decir:

A) Para la trasmisión de un uno: Se debe poner en estado bajo (LOW) el cable de datos correspondiente "data 1", mientras que el cable de datos "data 0"permanece en estado Alto (HIGH).

B) Para la trasmisión de un cero: Se debe poner en estado bajo (LOW) el cable de datos "data 0", mientras que el cable de datos "data 1"permanece en estado Alto (HIGH).

Cuando se desliza una tarjeta de acceso a través de una lectora, esta trasmite una serie de 1 y 0 a través de sus dos cables, esto genera una palabra (Conjunto de Bits que pueden ser 1 o 0). Dicha palabra es resultado de la decodificación de la información de la tarjeta (esta información puede utilizarse como un identificador del usuario dueño de la tarjeta, ya que contiene el número de sitio (Facility Code), el número de tarjeta (Card Number) y los bits de paridad). Aunque a través de este protocolo es posible trasmitir la cantidad de Bits que se deseen, existe un consenso en la cantidad de Bits a trasmitir que van desde los 26, 32, 35, 44 o 128. Siendo la trasmisión más común el protocolo estándar de 26 Bits.
En la transmisión de los datos del protocolo Wiegand, el tiempo de separación entre cada pulso es de

2 [mS] (milisegundos) y la duración de cada pulso es de 50 [μs] (microsegundos).

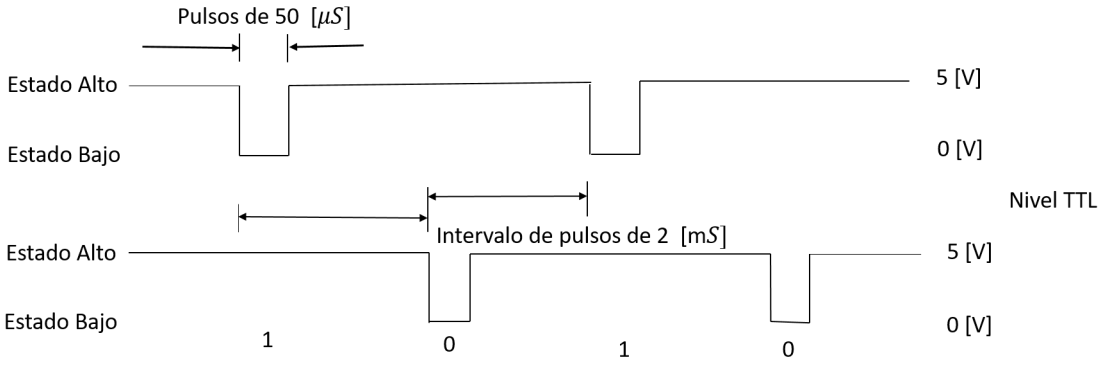

Figura 3.1: Representación gráfica de la transmisión de bits mediante pulsos Wiegand. [17]

Formato del protocolo Wiegand.
Como se mencionó anteriormente la cantidad de bits enviados en el protocolo Wiegand cambia, según el formato que se utilice y cada tarjeta de acceso es " configurada"para funcionar bajo cierto formato.

El único protocolo que se encuentra estandarizado es el de 26 Bits, esto quiere decir que su formato (la cantidad y la posición de bits correspondientes al número de tarjeta, Facility Code, y bits de paridad dentro de la trama) se encuentra definido universalmente.
Su representación se muestra a continuación:

Figura 3.2: Representación gráfica del formato de 26 bits en el protocolo Wiegand. [10]

A continuación se presentan el significado de los datos que fueron mencionados dentro del formato de la figura 3.2:

- PE (Even Parity): es el bit de paridad par, que se encuentra relacionado al bloque de bits que representan al Facility Code
- F (Facility Code): son el conjunto de bits que describen al identificador o número de sitio (Facility Code).
- N (Card Number): son todos los bits que corresponden al número de tarjeta.
- PO (Odd Parity): es el bit de paridad impar, que se encuentra relacionado al bloque de bits que representan al Número de tarjeta.

En general, para la trama estándar de 26 bits, los 8 bits más cercanos al bit más significativo serán asignados a la variable " Facility Code"(Del Bit 18 al 25), y los 16 Bits más cercanos al bit menos significativos (del bit 2 al 17) serán asignados a la variable " Card Code". El Bit 1 y 26 serán reservados para ser los Bits de paridad de cada bloque.
Para el protocolo Wiegand de 26 bytes se pueden generar como máximo 255 números de sitio diferentes y cerca de 65,535 IDs de tarjetas (Número de tarjetas, Card Number) para cada identificador de sitio (Facility Code). Por lo que el número de tarjetas totales que se pueden crear es de 16,711,425.[36]
Por otra parte, si se trabaja con otra cantidad de bits dentro del protocolo Wiegand (32, 34, 35, 37, 44, 128 bits) el formato usado puede cambiar.

Por ejemplo, se puede utilizar un protocolo que utiliza 34 bits con estos dos formatos:

- Para el primer formato del protocolo de 34 bits se pueden utilizar 8 bits para representar el campo para el " Identificador o número de sitio"(Facility code) y puede empezar desde la posición 2 de la trama.

- Para otro formato que utilice el protocolo de 34 bits el " Identificador o número de sitio"(Facility Code) puede ser representado con 12 bits empezando desde la posición 21 dentro de la trama.

Los formatos definidos dependen del Hardware utilizado y deben ser adaptados a éste, para que sea posible la decodificar la información.
Es importante aclarar que en la industria de seguridad el cambio de formato es importante, dado a que esto ayuda a volver menos vulnerable al sistema de control de accesos.

En general, todos los formatos contienen tres elementos importantes: bits de paridad, un conjunto de bits para representar el "Facility Code" y otro más para el número de tarjeta " Card Number", cada uno de ellos cumple una funcionalidad.

- Facility Code: Identifica la instalación o aplicación específica donde se usará la tarjeta
- Número de tarjeta: Es el número que será el identificador asignado a cada usuario, éste número debe ser único.
- Bits de paridad: Es utilizado para corroborar que los datos binarios se hayan trasmitido correctamente. El diseñador del formato del programa decidirá si cada bit de paridad empleado debe ser par o impar. Los bits de paridad son asociados a bloques o palabras, en éste caso un bit es asignado al grupo de bits correspondientes para el Facility Code y otro al grupo de bits para representar el Número de tarjeta. Para verificar que los bits se hayan transmitido correctamente el número de bits de cada bloque deben resultar en par o impar de acuerdo a lo que se haya definido en el diseño inicial del formato.

  A continuación, se presentan unos ejemplos de la aplicación de los bits de paridad, planteando casos hipotéticos:
  Bajo una paridad par (PE), en el caso hipotético que los números del 1 al 5 en binario representarán el número de sitio (Facility Code), el valor del bit de paridad par quedaría de la siguiente forma:

Representación Facility Code en decimal	Representación Facility Code en binario								PE
1	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0	1
3	0	0	0	0	0	0	1	1	0
4	0	0	0	0	0	1	0	0	1
5	0	0	0	0	0	1	0	1	0

Tabla 3.1: Ejemplo de la asignación del bit de paridad par, para cada uno de los cinco números de sitio (Facility Code) considerados.

  Como se aprecia en la Tabla 3.1, cuando se trabaja con paridad par, el valor del bit de paridad toma el valor de 1 en los casos en los cuales la palabra correspondiente al Facility Code tiene un número impar de unos, en éste ejemplo éste hecho se presenta en los casos cuando el Facility Code vale 1,2 y 4. Por el contrario, el bit de paridad adquiere el valor de 0, cuando la cantidad de unos en la palabra del Facility Code es par. Como es en el caso cuando el Facility Code vale 3 y 5.

  Por el contrario, si ahora se trabaja bajo una paridad impar (PO) para el mismo rango de números, pero ahora estos representan el número de tarjeta con paridad impar quedaría de la siguiente forma.

Representación número de tarjeta en decimal	Representación número de tarjeta en binario																PO
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

Tabla 3.2: Ejemplo de la asignación del bit de paridad impar, para cada uno de los cinco números de tarjeta (card number), considerados.

De la misma forma como se aprecia en la tabla 3.2, cuando se trabaja con paridad impar, el valor del bit de paridad toma el valor de 1 en los casos en los cuales la palabra correspondiente al número de la tarjeta tiene un número par de unos, como en los casos cuando el número de tarjeta vale 3 y 5. Por el contrario, el bit de paridad adquiere el valor de 0, cuando la cantidad de unos en la palabra correspondiente al número de tarjeta es impar. Como en el caso cuando el número de tarjeta vale 1, 2 y 4.

3.2. Antecedentes teóricos del automatizador embebido IoT

3.2.1. Hardware utilizado

La placa seleccionada para la realización de éste proyecto tiene el número de parte B-L475E-IOT01A. Ésta placa fue escogida dado a que nos proporcionaba la posibilidad de poder realizar una aplicación IoT. En específico nos ayudaba a satisfacer la necesidades de crear un emulador capaz generar eventos (deslizar tarjetas a través de las lectoras) y ser controlado remotamente. Algunas de las características y ventajas que nos proporciona esta placa se mencionan a continuación:

- Un microcontrolador de ultra bajo consumo energético perteneciente a la familia STM32L4 con un núcleo ARM Cortex M4 con un 1 Mbyte de memoria flash y una memoria SRAM de 128 Kbytes.
- Un módulo de internet Inventek System (ISM43362-M3G-L44) con el estándar 802.11 b/g/n
- Memoria flash de 64-Mbit Quad-SPI (Macronix).
- Módulo de Bluetooth (SPBTLE-RF).
- Módulo RF programable de bajo consumo energético (SPSGRF-868 or SPSGRF-915)
- Un NFC dinámico basado en el modelo M24SR, con una antena NFC
- Sensor capacitivo digital de humedad relativa y temperatura. (HTS221)
- Un magnetómetro de alto desempeño de tres ejes (LIS3MDL).
- Un giroscopio y un acelerómetro de tres dimensiones (LSM6DSL).

El módulo ISM43362-M3G-L44 es un dispositivo WiFi Serial Embebido (eS-WiFi) para conectarse a internet. El módulo de WiFi consiste en un procesador Cortex, una antena integrada y un dispositivo WiFi Cypress. Éste módulo provee interfaces UART, USB y SPI permitiendo la conexión en un sistema embebido. Éste módulo no requiere de un sistema operativo y en él se llevan a cabo todas las capas de la pila TCP/IP (aplicación, transporte, red o Internet y enlace o acceso a la red) las cuales sólo requieren de IWIN (Inventek Wireless Interoperability Network), que son comandos AT para establecer la conectividad para el producto inalámbrico, minimizando el tiempo de desarrollo, rutinas de prueba y certificación.[4]

3.2.2. Protocolo SPI

Por sus siglas en inglés SPI (Serial Peripheral Interface) es un protocolo de comunicación comúnmente utilizado por circuitos integrados para establecer una comunicación con sus dispositivos periféricos embebidos dentro de una placa. Éste protocolo fue creado por la compañía Motorola, y fue

introducido el primer microcontrolador que conservaba la misma arquitectura que el famoso microprocesador Motorola 68000 (El cual fue el mejor microprocesador de su época y fue utilizado dentro de la computadora Apple Macintosh en 1984).[2]

Este protocolo de comunicación necesita de 4 líneas de comunicación.

- Señal de reloj (SCLK): Es una señal enviada desde el maestro hacia todos los esclavos. Todas las señales SPI se encuentran sincronizadas con ésta señal de reloj.
- Señal de selección de esclavo (SS_n): Ésta señal es utilizada para indicar con que esclavo se comunicará el maestro.
- MOSI (Master Out-Slave in): Es una línea de comunicación desde el maestro hacia el esclavo.
- MISO (Master in Slave-Out): Es una línea de comunicación desde el esclavo al maestro.

En el protocolo de comunicación SPI un dispositivo central (Maestro) es el encargado de inicializar toda la comunicación con los esclavos. Cuando éste dispositivo " Maestro" requiere mandar o recibir información, éste selecciona alguno de los dispositivos esclavos poniendo en nivel bajo su línea de comunicación (SS_n) correspondiente. Posteriormente el maestro activa la señal de reloj a una frecuencia funcional para el maestro y el esclavo, finalmente el maestro puede mandar información mediante la línea de comunicación MOSI o recibir mediante la línea de comunicación MISO. Dentro de éste protocolo existen cuatro modos de comunicación (Modo 0,1,2,3), dentro de los cuales se definen los siguientes parámetros:

- El flanco de reloj en el cual la línea MOSI conmuta
- El flanco de reloj en el cual el maestro revisa la línea MISO
- La señal de reloj cuando éste está en reposo (Estado de la señal de reloj cuando no está activo Alto o bajo).

El maestro y los esclavos tienen que tener los mismos parámetros en cuanto a su frecuencia de reloj, polaridad de reloj (CPOL) y en su polaridad de fase (CPHA) para que la comunicación sea posible.
El bit CPOL define la polaridad de la señal del reloj cuando éste está inactivo. El estado inactivo se define en el periodo de tiempo cuando el canal SS_n pasa de alto a bajo en el comienzo de la transmisión, así como también, cuando el SS_n conmuta de un estado bajo a alto, al final de la transmisión. En otras palabras, es la polaridad que tendrá el reloj al inicio (cuando se abra) o al final (cuando se cierre) el canal de comunicación con el esclavo.[19]
Por otra parte, el bit CPHA define la fase del reloj. Es decir, dependiendo del bit CPHA, durante el flanco ascendente o descendente del reloj se usará para recibir y/o enviar los datos por los canales de comunicación MOSI y MISO.[19]
El protocolo SPI no define un rango máximo de datos o un esquema en particular. Éste protocolo no tiene conocimiento de algún mecanismo para la confirmación de recepción de información y no ofrece ningún control de flujo.[2]

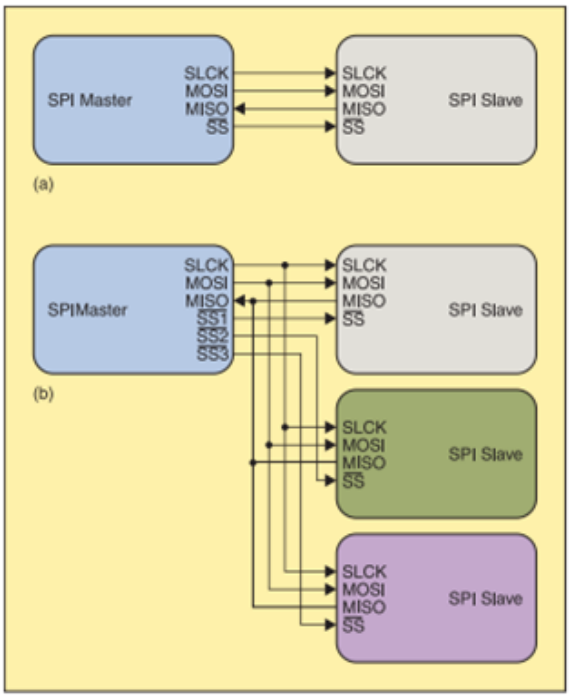

Figura 3.3: Diagrama del protocolo SPI. (a) Protocolo SPI con un sólo esclavo. (b) Protocolo SPI con múltiples esclavos. [2]

3.2.3. Comandos AT

Los comandos AT son comandos de lenguaje de texto corto. En el año de 1980 Hayes Microcomputer Products, INC. Fue uno de las primeras empresas, que manufacturaba modems, en utilizar un tipo de comando 'AT' para controlar las operaciones de comunicación entre sus dispositivos mediante POTS (Plain Old Telephone Service). Desde entonces una gran cantidad de productos fueron desarrollados para utilizar este tipo de comunicación.
'AT' es un acrónimo de 'AT'tention, y es usado para tener la atención del dispositivo que lo utiliza para poder controlar todas sus funcionalidades. Normalmente al comando AT se le suele acompañar por otras letras y números, los cuales ayudan a controlar funciones asociadas a ese comando. Por ejemplo: 'ATDT1234567' significa " ATtention módem habilite una línea para marcar el número que sigue al comando", en este caso 1234567. Actualmente en algunos dispositivos es posible acortar el comando 'AT' por el comando funcional, tomando el ejemplo anterior el comando sería DT1234567.
En el caso específico de éste proyecto se utilizaron los Comandos AT del sistema Inventek llamados IWIN. En estos comandos se adoptó la topología funcional de los comandos AT, mencionada anteriormente, para el control del módulo es-WiFi y adicionalmente, el Sistema Inventek agrego un signo de '=' a su sintaxis para delimitar los comandos AT de los valores.
Por ejemplo: el comando AT para configurar la dirección IP del módulo eS-WiFi sería 'C6=127.0.0.1' en vez de 'ATC6=127.0.0.1'[3]

3.2.4. Dirección MAC

La dirección MAC (Media Access Control) es un identificador único que es asignado a cada dispositivo que se puede conectar a la red. Es una combinación de 12 dígitos hexadecimales, que se encuentran divididos cada dos dígitos por un punto y coma o un guion. Por ejemplo, la dirección MAC 7a040722r2h3, se presenta como 7a:04:07:22:r2:h3 o como 7a-04-07-22-r2-h3. A diferencia de la dirección IP, la dirección MAC nunca cambia.

3.2.5. Modelo OSI

Es un modelo teórico de referencia que define el proceso conceptual que se debe de seguir para establecer una comunicación y conexión a través de una red. Fue oficialmente adaptado como un estándar ISO, en 1979. Este modelo consta de 7 capas, en donde cada una de ellas se manipula la información de forma diferente (Bits, frames, paquetes). La unidad en la cual cierta capa maneja dicha información, se llama Unidades de datos de protocolo (Protocol Data Unit-PDU). [22]
Adicionalmente, en algunas de las capas se agregan datos específicos en la información que se está trasmitiendo. Dichos datos, son agregados en forma de cabeceras (Headers), trailer (se agrega al final de la información transmitida) y/o ambas, dicha información son datos de control o instrucciones que deben ser consideradas en cada capa correspondiente del proceso [22].
Aunque no se ahonda a detalle en la operación y los procesos que se sigue en cada capa, se dará una idea generar del modelo OSI, a continuación.
"La idea de modelar del sistema de red, no tiene como finalidad definir cómo opera, pero define los elementos y las funciones que componen a la red, de tal forma que es posible clasificarlos en las distintas capas del modelo"[22].
Las capas del modelo OSI se muestran en la figura 3.4.

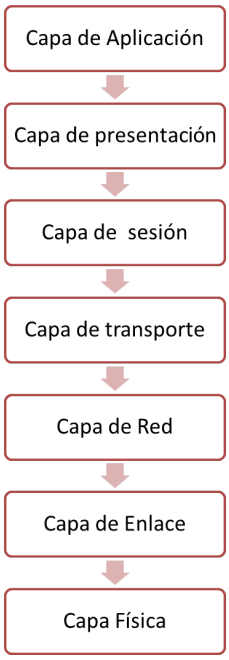

Figura 3.4: Capas teóricas del modelo OSI. [22]

Groso modo se hablará de cada capa a continuación:

- Capa física: en ésta capa la información se maneja en forma de palabras de bits, su función principal es proveer de todo lo necesario para poder generar una transmisión transparente de bits recibidos desde la capa de enlace del remitente, hasta la capa de enlace del destinatario o viceversa. Se define el medio físico a utilizar para poder transmitir la información, por ejemplo, se puede transmitir las señales mediante una diferencia de potencial eléctrica a través de un cable, mediante ondas a través de una fibra óptica o incluso mediante señales electromagnéticas a través del aire o algún otro medio.[22]
- Capa de enlace: En ésta capa la información se maneja en forma de frames (Un conjunto de cientos de Bytes hasta algunos miles de Bytes). Ésta capa es responsable del establecimiento y despliegue de las conexiones de enlace de datos entre las entidades que se comunicarán, un ejemplo de un parámetro a definir en ésta capa es la velocidad de tasa de transmisión (Data Rate). En ésta capa se asegura que los frames de datos sean recibidos en el mismo orden en el cual fueron enviados, y/o que al menos los frames puedan ser recomendados en el orden correcto.[22]

- Capa de red: La forma en la que se trata a la información en ésta capa es en forma de paquetes. Algunas de las tareas principales que se llevan a cabo es el ruteo de la información de una red a otra (elección de la mejor ruta desde el origen al destino). En ésta capa se lleva a cabo una segmentación de la información (fragmentación o reducción de tamaño de los PDUs) para facilitar la transferencia. En ésta capa se hace un mapeo completo para facilitar la transferencia de la información desde la fuente hasta la red del dispositivo, así como, desde la red del dispositivo al destinatario y de regreso.[22]

- Capa de transporte: En ésta capa la información es tratada como segmentos. En ésta etapa del proceso se garantiza la conexión entre las entidades que desean conectarse y de liberar la conexión cuando la transferencia de datos a finalizado. En la capa de transporte, la información es transformada en segmentos en el remitente y reconstruida en el destinatario, el mismo proceso sucede en dirección contraria. Además de ello, aquí también se controla la tasa de transferencia de los segmentos entre ambas partes.[22]

- Capa de sesión: Su propósito principal es regular y organizar la comunicación para que se puedan llevar a cabo múltiples transferencias de datos al mismo tiempo. La capa de sesión es responsable de empezar las sesiones de comunicación entre las dos entidades involucradas y cuando ésta finalice es responsable de liberar la comunicación.[22]

- Capa de presentación: En ésta capa se trabaja en la manera en la que la información será presentada en la aplicación.[22]

- Capa de aplicación: En ésta capa los servicios son presentados al usuario final.[22]

El flujo de información desde la capa de aplicación hasta la capa física se llama encapsulamiento. Esto se debe a que durante el paso de la información a través de las capas se le suele agregar información de control al inicio y al final (Trailer y Header respectivamente), lo que la hace parecer una especie de cápsula. Por el contrario, el flujo en dirección contraria, desde la capa física hasta la capa de aplicación, se le llama desencapsulamiento, lo que implica remover los headers y los trailers para devolver a la información a su versión original. [22]

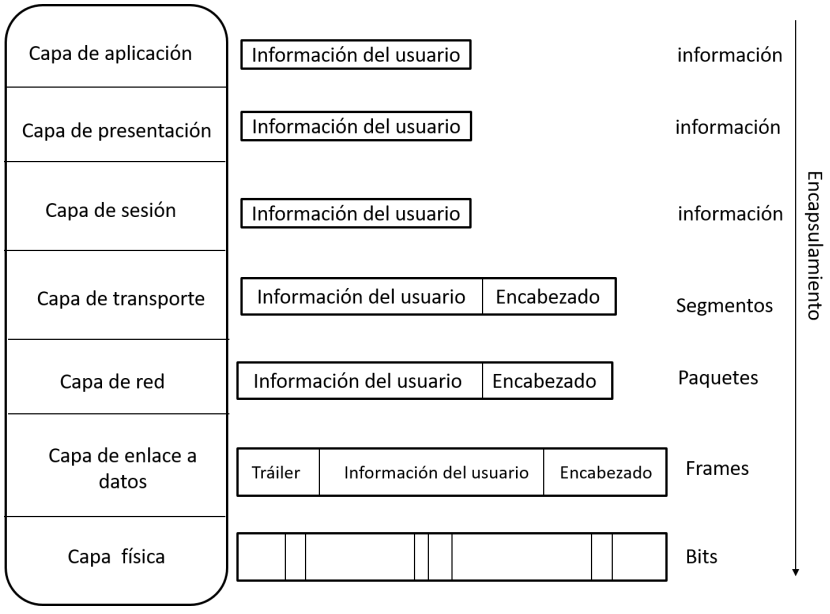

Figura 3.5: Estructura de datos a través de las siete capas del modelo OSI. [22]

3.2.6. Modelo TCP/IP

El nombre proviene de la combinación de dos protocolos de comunicación:

- Protocolo de Control de Transmisión (TCP-Transmission Control Protocol)
- Protocolo de Internet (IP-Internet Protocol)

En el protocolo TCP/IP los datos recibidos viajan a través de las capas que muestran en la figura 3.6.

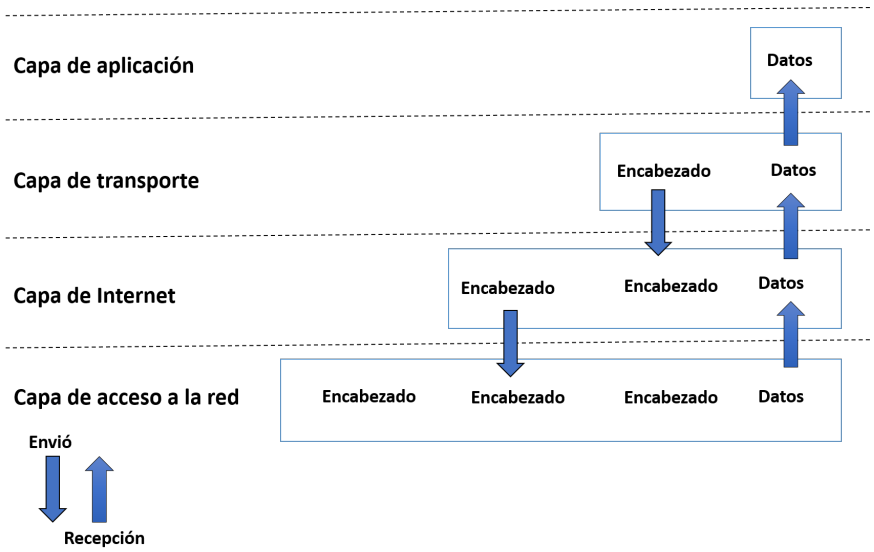

Figura 3.6: Estructura de los datos a través de las cuatro capas del modelo TCP/IP. [8]

En cada capa se agrega información de control a los datos que viajan a través de ellas para asegurar la correcta entrega. Ésta información de control, como se mencionó anteriormente, recibe el nombre de "Header", porque antecede a los datos que se están transmitiendo. A pesar de ello, cada capa trata toda la información que recibe de la capa superior como información, es decir, no se hace distinción entre los headers agregados por las capas superiores y la información original. Por otro lado, cuando la información es recibida sucede lo opuesto, antes de la que la información pueda ser enviada a la capa superior desde las inferiores, es necesario quitar los encabezados ("Headers") agregados por cada una de ellas. Cada capa tiene su propio e independiente estructura de datos, conceptualmente cada capa es ajena de la estructura de datos usadas por las propias superiores e inferiores. [8]
Aunque el modelo TCP/IP tiene en común algunas capas con el modelo OSI (Open System Interconnect), como se aprecia en la figura 3.7, éste consta de cuatro capas esencialmente:

1. Capa de acceso a red: En ésta capa se realizan tareas para gestionar la infraestructura física que permite a las computadoras establecer una comunicación entre sí por internet.[9] Las capas equivalentes en el modelo OSI son la física (Physical) y enlace (Data Link). Otra de las tareas importantes que se llevan a cabo en ésta capa es el mapeo de la dirección IP usadas en la capa de internet a direcciones físicas de Hardware (Como direcciones MAC).[8]

2. Capa de internet: El Protocolo de Internet (IP) es el más importante en ésta capa. La versión más utilizada actualmente es la IPv4, aunque también es posible encontrar la IPv6. En ésta capa se controla el flujo y el proceso de selección de la mejor ruta a través de la red para enviar la información para garantizar que los datos se envíen de forma rápida y correcta, también se lleva a cabo la recolección y unión de paquetes de datos en el destino. [9]
   Protocolo IP: Cada vez que se desea trasmitir información por Internet, ésta se divide en pequeños " paquetes"de datos que en su conjunto forman el mensaje original. Gracias al protocolo IP es posible asignarle a cada uno de estos " paquetes"la dirección IP del remitente y la del receptor, enrutar los mensajes a través de la red, es decir escoger la mejor ruta, Así como encontrar el servidor o la computadora a la que le corresponde cada IP requerida.

3. Capa de transporte: En ésta capa se asegura que se tenga una conexión de datos confiable entre los dispositivos de comunicación. "La capa de transporte divide los datos en segmentos, confirma

los segmentos que ha recibido el remitente y se asegura que el destinatario confirme los segmentos recibidos por su parte."[9] Esta función es realizada por el protocolo TCP sin embargo, es posible utilizar otro servicio de transporte en ésta capa llamado " User Datagram Protocol "(UDP) el cual es menos confiable, dado a que no garantiza que los datos se hayan trasmitido o recibido completamente.
Protocolo TCP: La principal tarea de éste protocolo es la entrega y el transporte de los datos de una manera fiable entre sistemas de Internet.[21]. TCP se apoya del protocolo de Internet para transportar los datagramas. El protocolo TCP asegura que los datos no se dañen, se pierdan, se dupliquen o se entreguen en malas condiciones al receptor durante su camino.[20]
Protocolo UDP: Utilizando éste protocolo las aplicaciones pueden mandar información muy rápido, ya que no se necesita establecer una conexión con el destinatario ni esperar por una confirmación del mismo " acknowledge". Sin embargo, no existe garantía de que los paquetes de información serán recibidos completamente y en el mismo orden en el que fueron enviados, además de que no se proporciona una protección de la información contra terceros que puedan manipular o accesar a dicha información que está siendo trasmitida. Algunas aplicaciones que utilizan éste protocolo son aquellas que necesitan transmitir valores de medida o ejecutar repetidamente las mismas órdenes.
Éste protocolo también permite conexiones IP de multicasteo. Si una aplicación permite el envío de paquetes eficiente y rápidamente a muchos destinatarios al mismo tiempo, UDP proporciona una base sólida. Por otro lado, dicho protocolo es apto como un protocolo de transporte para servicios que necesitan funcionar en tiempo real, como transmisiones de audio y video. Los cuales necesitan tener control de la transmisión, recepción y reproducción de la información por grandes periodos de tiempo. Aunque actualmente la mayoría de las aplicaciones de tiempo real utilizan el protocolo RTP (Real-time Transport Protocol)

4. Capa de aplicación: A ésta capa la conforman el grupo de aplicaciones que permiten al usuario acceder a la red. Es lo que el usuario final puede ver y con los que interactúa para recibir y enviar datos"[9].El protocolo HTTP es el protocolo más importante en ésta capa, aunque también existen otros protocolos que también se utilizan, como FTP, POP3, SMTP, entre otros. [24] Los elementos equivalentes dentro del modelo OSI para ésta capa son: " Sesión"(Session), " Presentación"(Presentation) y " Aplicación"(Aplication).

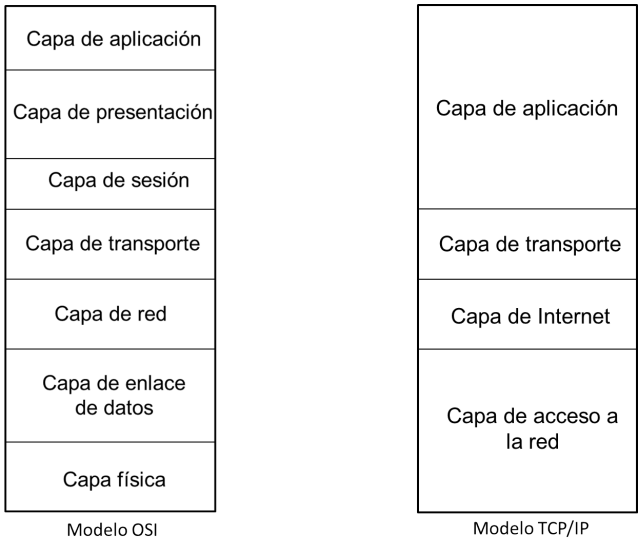

Figura 3.7: Equivalencia de capas entre el modelo OSI y el modelo TCP/IP. [22]

3.2.7. Protocolo HTTP

Dentro de éste protocolo existen dos conceptos importantes, en los cuales basa su funcionamiento.

- Cliente: Un cliente es cualquier proceso que solicita un servicio en específico a un proceso de un servidor.
- Servidor: Un servidor es un proceso que provee asistencia a lo que es solicitado por el cliente.

Por sus siglas en inglés Hyper Text Transfer protocol, es un protocolo para la transmisión y recepción de datos a través de archivos, usualmente HTML, en Internet[20].
Se trata de un protocolo de estructura cliente- servidor sin estado (Stateless protocol), es decir, es de estructura cliente-servidor porque las peticiones de datos son iniciadas por el cliente al servidor y " sin estado"porque la comunicación no es continua y sostenida entre las dos partes involucradas (Servidor-Cliente), como cuando se habla por teléfono. Por el contrario, funciona más como un sistema de mensajería, en donde las dos partes se comunican mediante el envío de " paquetes" o cartas, sólo cundo dichos paquetes son enviados o recibidos se abrirá el canal de comunicación y será cerrado tan pronto como se cumpla con lo anterior [6].
A la información solicitada por el cliente se le suele llamar Petición o " Request " y los datos proporcionados por el servidor se conocen como respuesta o " Response".
Los procesos de cliente y servidor pueden existir en la misma computadora o en diferentes computadoras unidos por una red. Cuando el proceso de cliente y el servidor residen en dos o más computadoras independientes dentro de una red, el servidor puede proveer su servicio para más de un cliente. Adicionalmente un cliente puede solicitar servicios de múltiples servidores dentro de la red sin importar la ubicación o las características físicas de la computadora en la cual el servidor resida. [5]
Cuando se visita un sitio Web, el navegador o cliente envía una petición al servidor Web para obtener datos o información, por ejemplo, es posible la obtención de la página web mediante la solicitud de un archivo HTML que contenga la información pertinente. Durante la comunicación entre el servidor y el cliente, se intercambian tanto datos reales como " Información meta". Toda ésta información se condensa en la cabecera HTTP. [6]
Por ejemplo: Para que el navegador (cliente) pueda visitar la página www.msaleh.co.cc, éste debe realizar un " request" al servidor web como el que se muestra en la parte superior de la figura. 3.8, en dónde mandan algunos datos de cabecera necesarios para desplegar la página, los cuales no son visibles para los usuarios.
De igual forma cuando el servidor web recibe el " request"del cliente, éste responde con información meta como la que se muestra en la parte inferior de la figura 3.8.

```
Request Header:
GET / HTTP/1.1
Host: www.msaleh.co.cc
User-Agent:Mozilla/5.0 (Windows; U; Windows
NT 5.1; en-US; rv:1.9.0.10)
Gecko/2009042316 Firefox/3.0.10
Accept: */*
Connection:close

Response Header:
HTTP/1.1 200 OK
Date: Sat, 09 May 2009 12:27:54 GMT
Server: Apache/2.2.11 (Unix)
Last-Modified: Thu, 12 Feb 2009 15:29:42
GMT
Etag: "c3b-462ba63a46580"-gzip
Cache-Control: max-age=1200, private,
proxy-revalidate, must-revalidate
Expires: Sat, 09 May 2009 12:47:54 GMT
Accept-Ranges: bytes
Content-Length: 976
Content-Type:text/html
```

Figura 3.8: Ejemplo de la metadata enviada y recibida a través de los Encabezados de peticiones y respuestas HTTP, para llevar a cabo la comunicación entre cliente y servidor. [7]

Una petición HTTP, está formado por los siguientes campos

- Método HTTP: Éste campo define la operación que el cliente quiere realizar[20].Existen diferentes acciones que el usuario puede seleccionar como: GET,POST,OPTIONS, HEAD.
- La dirección del recurso pedido (www.msaleh.co.cc).
- La versión del protocolo
- Cabeceras HTTP, que aportan información adicional al servidor
- Cuerpo de mensaje, mediante el método post se puede enviar más información para el servidor.

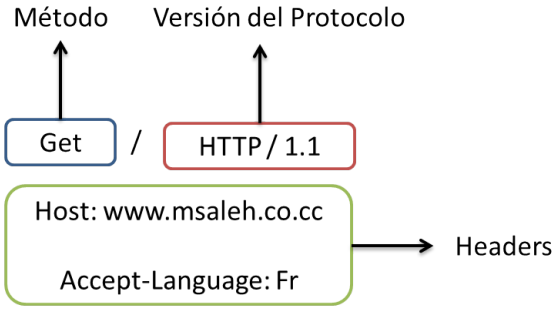

Figura 3.9: Estructura de la metadata de una petición HTTP, enviada como petición desde un cliente hacia un servidor. [20]

Por otro lado, la respuesta está formada por los siguientes campos:

- La versión del protocolo HTTP que se está usando.
- Un código de estado para informar si la petición fue exitosa o no. Cada código de estado tiene un significado.
- Un mensaje de estado, éste hace referencia a una breve descripción del código de estado. En este caso se retornó " OK"
- Cabeceras (Header) HTTP
- Algunas veces se puede agregar el recurso solicitado por el cliente. [20]

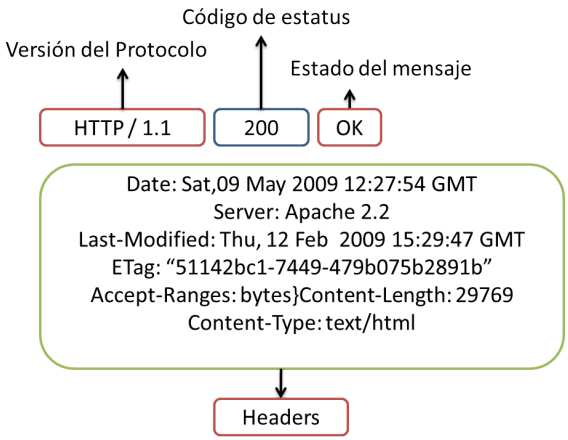

Figura 3.10: Estructura de la metadata de una respuesta HTTP, enviada desde el servidor hacia el cliente. [20]

3.2.8. PWM

"La Modulación por Ancho de Pulso (PWM) es una técnica de modulación que genera anchos de pulsos variables para representar la amplitud de entrada de una señal analógica". [11]
En ésta técnica se modifica una onda periódica (Cuadrada o senoidal), para transmitir información a través de un canal de comunicación o para controlar la cantidad de energía que se envía a una carga".[12]
Otra definición importante es el Ciclo de Trabajo de una señal periódica (Duty Circle), el cual puede ser definido como el ancho neto de la señal cuando está activa (En estado alto en el caso de una señal cuadrada) en relación con su periodo. [12]
Matemáticamente el ciclo de trabajo se expresa como:

$$CiclodeTrabajo = \frac{t}{T} \tag{3.1}$$

Donde:
t = tiempo de la señal activa
T = Periodo, tiempo total

Generar una señal PWM " consiste en activar una salida digital durante un tiempo para después apagarla en otro intervalo de tiempo. De tal forma que se puedan generar pulsos en estado alto que se repitan de manera constante. Al variar el ciclo de trabajo de la señal, se obtendrá un voltaje promedio diferente en la salida a lo largo del tiempo, el cual será equivalente al valor analógico deseado"[12]

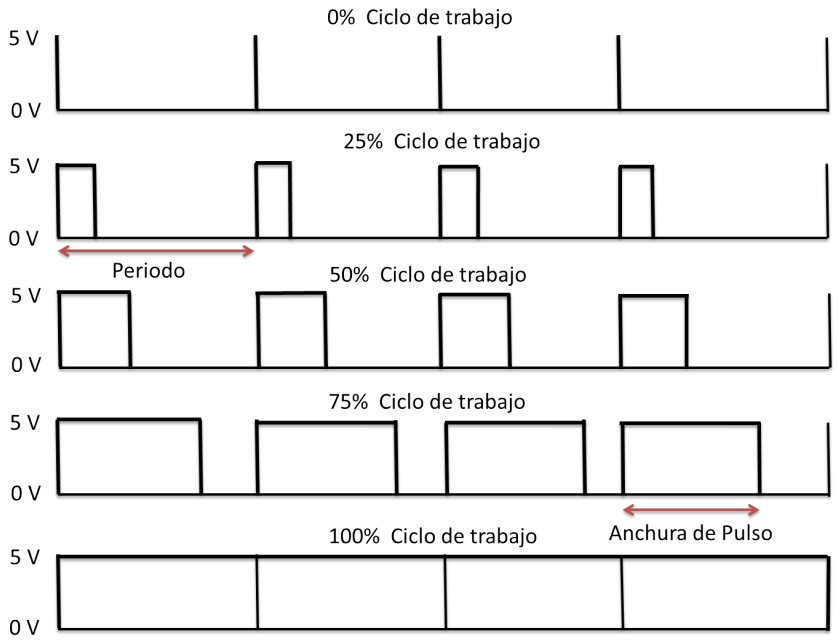

Figura 3.11: Representación gráfica de los diferentes ciclos de trabajo de una señal PWM. [12]

Con el fin de mover un servomotor una cierta cantidad de grados, este debe ser alimentado con un cierto nivel de voltaje. Para lograr esto con un microcontrolador, dado a que dichos dispositivos electrónicos basan su funcionamiento en señales digitales (5 [v] y 0 [V]), es necesario emular valores analógicos mediante la utilización de señales PWM (Pulse Width Modulation) o con un convertidor DAC. En este trabajo, se utilizó la primera opción (PWM). Con éste método, mientras mayores sean los ciclos de trabajo de las ondas cuadradas PWM, se obtendrá un mayor valor de voltaje.

3.2.9. Protocolo I^{2}C

Ésta basado en una relación de comunicación maestro-esclavo. En donde el microcontrolador (maestro) tiene un completo control cuando se establece una comunicación con los periféricos esclavos. Físicamente el bus I^{2}C está conformado por dos cables activos (SDA, SCL) y otro cable conectado a tierra. El protocolo I^{2}C especifica que el " circuito inter-integrado"(IC) que inicia la transferencia de datos en el bus será considerado el Bus Maestro y en consecuencia, todos los otros ICs serán considerados como esclavos. [2]

El bus de comunicación I^{2}C utiliza dos líneas de comunicación:

- Serial Data (SDA): Es una línea bidireccional de comunicación entre el microcontrolador y los dispositivos esclavos para leer y escribir datos. [2]
- Serial Clock (SCL): Ésta línea de reloj es utilizada para indicar cuando la información trasmitida a través de la línea de comunicación SDA es válida. Sirve para marcar un ritmo en la comunicación entre el microcontrolador y los esclavos. [2]

El proceso de comunicación I^{2}C, se enuncia continuación:

1. El maestro mandará la condición para empezar la comunicación. Éste impulso eléctrico actúa como una señal de " Atención"para todos los dispositivos conectados. [2]
2. Todos los dispositivos conectados estarán " Atentos " a la información que se trasmitirá a través del bus. [2]
3. El maestro enviará la dirección del dispositivo esclavo con el que establecerá comunicación, así como también enviará un bit para especificar la operación que se realizará (Lectura o escritura). [2]
4. Habiendo recibido la dirección enviada por el maestro, los dispositivos esclavos comparan dicha información con su propia dirección. Sí ésta no coincide, dichos dispositivos esperaran hasta que el bus de comunicación sea cerrado por la condición de finalización enviada por el maestro, para estar listos para otra petición. En caso que la dirección coincida con la propia de algún dispositivo esclavo, se producirá una respuesta " ACKNOWLEDGE". [2]
5. Una vez que el maestro recibe la señal de " ACKNOWLEDGE", éste puede empezar con la transmisión o recepción de información. [2]
6. Cuando todo está echo el maestro mandará la condición de finalización. Esto significa que el estado del bus de comunicación se ha cerrado y que los ICs conectados podrían recibir otra petición para el comienzo de la transmisión o recepción de información.[2]

Inicio	Dirección Esclavo	Rd/nWr	ACK	Data	ACK	Data	ACK	Finalización
1 bit	7 bits	1 bit	1 bit	8 bits	1 bit	8 bits	1 bit	1 bit

Tabla 3.3: Formato del protocolo I^{2}C. [2]

Las velocidades de transferencia de la información en el protocolo I^{2}C se encuentra entre los siguientes rangos:

- "Standar Mode"$\rightarrow$ 100 $[\frac{kb}{s}]$

- "Fast Mode"→ 400 $[\frac{kb}{s}]$
- "High speed Mode "→ 3.4 $[\frac{Mb}{s}]$

Algunas variantes de velocidad de transferencia de éste protocolo son:

- "Low speed mode"→ 10 $[\frac{kb}{s}]$
- "Fast Mode +"→ 1 $[\frac{Mb}{s}]$

3.2.10. Internet de las cosas (IoT)

Es una red de objetos físicos a los que se puede acceder a través de internet. Estos dispositivos contienen tecnología embebida para interactuar con estados internos o ambientes externos. Con ésta tecnología los objetos pueden hacer mediciones y comunicarse, para facilitar la toma de decisiones. La tecnología de Internet de las cosas reúne las características cuyas primeras letras forman el acrónimo S-E-N-S-E, las cuales vienen de las palabras Sensoriales (Sensorial), Eficiencia (Efficiency), conectados (Networked), especializados (Specialized), en cualquier parte (Everywhere).[13]
En actualidad, el auge de la tecnología IoT se vio impulsada y beneficiada por algunos factores como:

- Decremento en el costo de los sensores
- Menor costo en la banda ancha
- Menor costo en los procesadores
- Teléfonos inteligentes
- Mayor cobertura inalámbrica
- Big Data
- IPv6

Gracias al gran impulso y utilidad que a tenido la tecnología IoT, ha sido posible que su alcance llegue en diferentes sectores, tal es el caso del ámbito industrial. En dónde tuvo un gran auge y adoptó el nombre de Industrial Internet of Things o Internet de las cosas industrial (IIoT), la cual como su nombre lo sugiere, hace referencia a la aplicación de la tecnología IoT en entornos industriales en sectores como la instrumentación, control de sensores y dispositivos que utilizan tecnologías en la nube. Recientemente, las industrias han utilizado la comunicación máquina a máquina (M2M) para hacer posible la automatización y el control inalámbrico. La industria IIoT ha recibido el nombre de la cuarta ola de la revolución industrial, industria 4.0.[14] Algunas aplicaciones en donde se ha utilizado la IIoT son:

- Fabricación inteligente.
- Activos conectados y mantenimiento preventivo y predictivo.
- Redes eléctricas inteligentes.
- Ciudades inteligentes.
- Logística conectada.
- Cadena de suministro digitales inteligentes.

Capítulo 4

Contexto de la participación profesional

Durante mi periodo laboral, estuve dentro del área de operaciones y mantenimiento (O&M). Dicha área era encargada de atender y arreglar los requerimientos de los clientes, así como, apoyar a los ingenieros de campo. Por ello, aunque mis tareas principales estuvieron enfocadas a darle mantenimiento, arreglar problemas y desarrollar mejoras en la aplicación de control de accesos de la empresa, mediante el uso de la programación orientada a objetos. También tuve la oportunidad participar en dos proyectos que tenían la finalidad de solucionar las problemáticas que se tenían dentro del área de pruebas o "Testing"dentro de mi equipo de trabajo, las cuales eran:

- Necesidad de visualizar en una pantalla la información de las tarjetas de acceso de los usuarios que eran deslizadas a través de la lectoras, al momento de realizar pruebas.
- Necesidad de automatizar el deslizamiento de tarjetas a través de las lectoras, para realizar las pruebas.

4.1. Definición del problema

En el área de O&M a veces era necesario replicar el entorno o ambiente de nuestros clientes, para entender mejor el problema, analizarlo y trabajar en la solución.
De igual forma, después de aplicar los cambios que solucionaban el problema, era necesario simular el ambiente nuevamente, para observar el comportamiento obtenido y verificar que el conflicto estuviera resuelto. Sin embargo, algunas veces para replicar el problema y estudiar la solución, surgían algunas necesidades como:

- Deslizar repetidamente una tarjeta de acceso a través de las lectoras: Dicho proceso necesitaba realizarse manualmente, por lo que era necesario que una persona estuviera físicamente en el laboratorio y se encargara de ésta tarea repetidamente.
- Necesidad de visualizar los datos de la tarjeta que era deslizada por las lectoras: Se necesitaba tener instalada y abrir la aplicación para poder ver los datos de la tarjeta que había sido deslizada.

Con la finalidad de contribuir en la solución de éstas áreas de oportunidad. Colaboré en la implementación de dos proyectos, los cuales se mencionan a continuación:

A) Decodificador Wiegand de tarjetas de acceso: Éste proyecto tuvo como finalidad poder desarrollar un programa dentro del entorno de desarrollo integrado seleccionado (IDE), que fuera capaz de procesar y decodificar los binarios que enviaba una lectora tras el paso de una tarjeta de acceso, para que la información pudiera ser consultada, visualizada en una pantalla y ser interpretada por un usuario.

B) Desarrollo de un emulador IoT para automatizar las pruebas de deslizamiento de tarjetas con la placa B-L475E-IOT01A: Anteriormente el equipo de la India, ya contaba con un emulador

que realizaba ésta tarea. Sin embargo, para su elaboración se utilizaron dos microcontroladores. Por lo cual surgió como área de oportunidad la migración del código a un sólo microcontrolador. Esto presento muchas ventajas, como se sugiere la reducción en el número de microcontroladores, una mayor eficiencia de en ejecución de eventos, una mayor facilidad de ser conectado a la red, un microcontrolador con un mejor procesamiento y la necesidad de un menor consumo de espacio.

En éste proyecto tuve un acercamiento a los microcontroladores. Y sus objetivos principales eran dos:

- Migrar el código con el que se contaba a un microcontrolador STM32.
- Desarrollar, crear y habilitar el nuevo emulador que fuera capaz de simular y automatizar el paso de tarjetas a través de las lectoras de forma remota, controlando el dispositivo mediante el envío y recepción de información por internet.

Capítulo 5

Metodología utilizada

5.1. Decodificador de tarjetas Wiegand

Los pasos para la realización de éste proyecto fueron los siguientes:

- Habilitación de los pines del microcontrolador y recepción de información.
- Análisis y tratamiento de la información.
- Mostrar de la información al usuario.

5.1.1. Habilitación de los pines del microcontrolador y recepción de información

Para la realización de éste trabajo, la lectora de tarjetas fue conectada directamente al microcontrolador.
Como primer paso, se definieron algunas variables que nos ayudarían en el proceso de efectuar operaciones y almacenar valores. Tales como, un arreglo de dimensión constante de tamaño igual que el valor de la variable "totalBits", el cual sería utilizado para guardar los bits obtenidos mediante las interrupciones.

```
#define totalBits 100
unsigned char dataBits[totalBits];
```

Figura 5.1: Arreglo que contendrá las palabras correspondientes a la decodificación de cada tarjeta de usuario deslizada por la lectora conectada al microcontrolador. [28]

También se contará con una variable llamada " intervalTime", para definir un intervalo de tiempo de espera máximo entre cada pulso de información, el cual, si es superado, se sabrá que se ha acabado con la transmisión de datos. Es importante considerar que el tiempo de separación entre cada pulso es de 2 [mS] (milisegundos) y la duración de cada pulso es de 50 us (microsegundos).

```
#define intervalTime 5000
```

Figura 5.2: Definición de la variable que contendrá el intervalo de tiempo de espera para leer bits. [28]

De igual forma, se definió otro variable de tipo unsigned char, que nos ayudó a llevar un conteo del número de Bits leídos. Dicha variable será de utilidad para recorrer el arreglo " dataBits", ya que se utilizará de iterador para recorrer el arreglo y representará la posición en la cual se colocarán cada Bit dentro del mismo.

```
unsigned char bitCaptured;
```

Figura 5.3: Definición de variable para guardar la cantidad de bits leídos. [28]

Finalmente, se tendrán dos variables auxiliares llamadas " flag" y " counter". La primera tendrá la función de ser un indicador para saber cuándo se ha acabado de leer todos los Bits. En estado alto (HIGH) indicará que no hay más bits para leer, mientras que, cuando tenga un estado bajo (LOW), indicará que el proceso de Bits de lectura continúa.
Por otro lado, la variable " counter"será de tipo entera y fungirá como un contador, ya que el valor de dicha variable decrecerá dentro de un ciclo con cada iteración realizada. Posteriormente, se inicializaron

```
unsigned char flag;
int Counter;
```

Figura 5.4: Definición de variables auxiliares, para indicar que se han acabado de leer todos los bits (Flag) y hacer un conteo de los bits recibidos (Counter). [28]

y configuraron los puertos y los pines del microcontrolador, con el fin de habilitarlos para que éstos estuvieran " a la espera"de la información mandada por la lectora, mediante el uso de las interrupciones.

```
//Inicializando los pines del microcontrolador
pinMode(2,INPUT);
pinMode(3,INPUT);

//Habilidando las interrupciones en los pines 2 y 3
attachInterrupt(digitalPinToInterrupt(2),ISR_DATA_0,FALLING);
attachInterrupt(digitalPinToInterrupt(3),ISR_DATA_1,FALLING);
```

Figura 5.5: Inicializando y configurando los pines 2 y 3 del microcontrolador para captar interrupciones. [28]

Como se mencionó anteriormente, con ayuda de las interrupciones se ejecutaron las funciones ISR_DATA_0 e ISR_DATA_1, para poder capturar todos los bits trasmitidos a través de los cables Data 0 y Data 1 por la lectora.

```
//Función Interrupción para capturar los bits 0
void ISR_DATA_0()
{
  Serial.print("0");
  dataBits[bitCaptured]=0;
  bitCaptured++;
  flag = 0;
  Counter = intervalTime;
}

//Función Interrupción para capturar los bits 1
void ISR_DATA_1()
{
  Serial.print("1");
  dataBits[bitCaptured]=1;
  bitCaptured++;
  flag = 0;
  Counter = intervalTime;
}
```

Figura 5.6: Definiciones de las funciones para realizar interrupciones. [28]

5.1.2. Análisis y tratamiento de la información.

Dentro del programa se llevó acabo un procesamiento de la información para dividir y obtener los datos correspondientes al número de sitio (Facility Code) y al número de tarjeta (Card Number), así como, para poder leer distintos formatos (En éste caso formatos de 26 y 35 bits).

```
//Formato Wiegand 26 Bits
  if(bitCaptured == 26)
  {
    //Facility Code (Del Bit 25 al 18)
    for(i=1;i<9;i++)
    {
      facilityCode<<=1;
      facilityCode |= dataBits[i];
    }
    //Código de la tarjeta (Del bit 17 al 2)
    for(i=9;i<25;i++)
    {
      cardCode<<=1;
      cardCode |= dataBits[i];
    }
    //Función para imprimir el Facility Code
    //y Card Code
    printBits();
  }
```

```
//Formato Wiegand 35 Bits
  if(bitCaptured == 35)
  {
    //Facility Code (Del Bit 34 al 22)
    for(i=2;i<14; i++)
    {
      facilityCode<<=1;
      facilityCode |= dataBits[i];
    }
    //Código de la tarjeta (Del bit 21 al 2)
    for(i=14;i<34;i++)
    {
      cardCode<<=1;
      cardCode |= dataBits[i];
    }
     //Función para imprimir el Facility Code y el Card Code
    printBits();
  }
```

Figura 5.7: Lógica para la lectura de los formatos Wiegand 26 y 35 bits y extracción del número de tarjeta y número de sitio. [28]

Aquí se realizan algunas tareas como identificar el protocolo Wiegand utilizado, ordenar y dividir la información, así como, llamar a la función encargada de imprimir el número de sitio y el número de tarjeta.

5.1.3. Mostrar la información al usuario

Finalmente, en ésta sección se realiza el formato para mostrar la información de la mejor manera posible al usuario. En ésta sección se imprime por el puerto serie, tanto el número de sitio como el número de tarjeta decodificada en sistema decimal, mediante un casteo implícito.

```
void printBits()
{
  Serial.print("");
  Serial.print("FacilityCode = ");
  Serial.print(facilityCode);
  Serial.print(", CardCode = ");
  Serial.print(cardCode);
  Serial.print(" ");
  Serial.println("**********************************");
}
```

Figura 5.8: Función que imprime en número de tarjeta y el número de sitio, en base decimal. [28]

Como paso final se reinician las variables para preparar al programa para recibir otra entrada.

```
void clearEntry()
{
  bitCaptured = 0;
  facilityCode = 0;
  cardCode = 0;
  for(i=0; i<totalBits; i++)
  {
    dataBits[i] = 0;
  }
}
```

Figura 5.9: Función para limpiar variables necesarias, para la siguiente iteración. [28]

5.2. Emulador IoT de tarjetas Automatizado

Para realizar todas éstas tareas primero se necesitó conocer la arquitectura del microcontrolador. Para saber entender mejor la relación entre todos los periféricos y el microcontrolador, con ello se pudo empezar a programar cada sección. Para que el emulador pudiera cumplir con su funcionamiento primario de generar eventos automatizados programados por tiempo y/o por cantidad de eventos. Se vio en la necesidad de dividir la funcionalidad primaria en pequeñas tareas. Las cuales se enuncian a continuación:

- Conexión a Internet
- Generar un servidor WEB en la placa
- Mover un servomotor
- Mandar impulsos eléctricos a los relays
- Mostrar los eventos en la pantalla OLED

5.2.1. Conexión a internet

Dentro de éste proyecto para lograr que la placa B-L475E-IOT01A. se pudiera conectar a internet se siguieron los siguientes pasos:

- Inicializar la comunicación SPI y Declaración de variables útiles.
- Obtener la dirección MAC del dispositivo.
- Obtener y conectarse a Internet mediante las credenciales ingresadas por el usuario.

5.2.1.1. Inicializar la comunicación SPI y Declaración de variables útiles.

Como primer paso se configuró e inicializó la comunicación SPI entre el microcontrolador STM32L4 y el módulo de internet (ISM43362-M3G-L44).
Una vez habilitada la comunicación, dentro del programa se declararon variables para guardar datos e información relevante como la dirección MAC del dispositivo, la dirección IP, las credenciales del usuario (SSID, Contraseña), la información enviada, la información recibida, el tipo de acceso Wifi protegido, entre otras.
De igual forma en el código se escribieron instrucciones AT y éstas fueron enviadas al módulo de internet mediante el protocolo SPI para la configuración, inicialización y control del módulo.

5.2.1.2. Obtener la dirección MAC del dispositivo

Una vez que se configuró e inicializó el módulo, se ejecutó el comando AT para la obtención de su dirección MAC. Si es posible obtenerla, su valor es guardado en la variable correspondiente, de otra forma se retorna un estado de error.

5.2.1.3. Obtener y conectarse a Internet mediante las credenciales ingresadas por el usuario

Posteriormente, para conectar el módulo a internet y obtener la dirección IP, se escribieron y ejecutaron los comandos AT correspondientes.
En ésta sección es necesario que usuario ingrese sus credenciales y el tipo de acceso WiFi de seguridad por primera vez. Si los comandos AT se pueden ejecutar correctamente con la información recibida por el usuario, se guarda la información en las variables antes definidas. El diagrama de flujo que representa el proceso en grandes rasgos se muestra en la figura 5.10

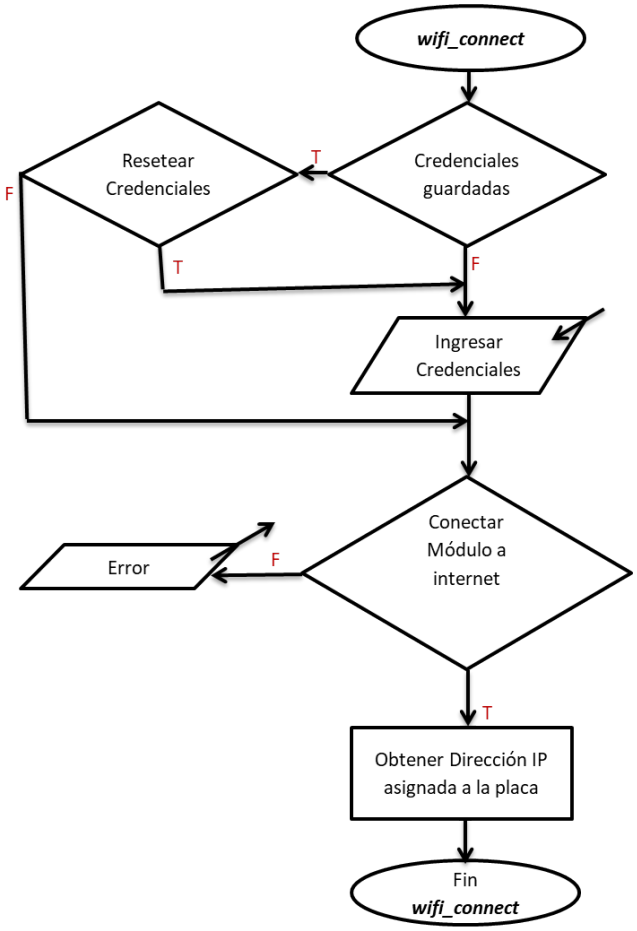

Figura 5.10: Diagrama de flujo que representa el algoritmo seguido para conectar el módulo de internet del microcontrolador a la red Wifi. [31]

5.2.2. Servidor Web embebido

" En general, para que los programas cliente-servidor que pertenecen a la capa de aplicación, puedan mandar y obtener información a través de la red, necesitan acceder a las capas inferiores, con el fin de que puedan enviar sus mensajes". [24]
"Para establecer la comunicación entre la capa de aplicación con la capa de transporte, dado a que los procesos de aplicación residen en el espacio de usuario, mientras que los procesos de la capa de transporte, forman parte del sistema operativo, se tiene disponible un mecanismo que nos permite establecer una comunicación entre ellos". [24] Dicho mecanismo, es una Interfaz de programación de aplicaciones (API). En particular, dentro del protocolo TCP/IP se utiliza el API de los Sockets. [24]
Los sockets están en los nodos finales de una comunicación entre dos procesos corriendo en una misma red y permite a las aplicaciones acceder a los servicios proporcionados por la capa de transporte del protocolo TCP/IP. [24] Éstos elementos necesitan ser creados mediante el uso de Software, posteriormente es posible utilizarlo para hacer operaciones de lectura y escritura. Cuando la aplicación ya no lo necesita es necesario cerrarlos. [24]. En resumen, se puede crear un socket mediante el uso de Software, y éste es una forma de conectar dos nodos de una red para que se puedan comunicar entre ellos. Esto se muestra en la figura 5.11

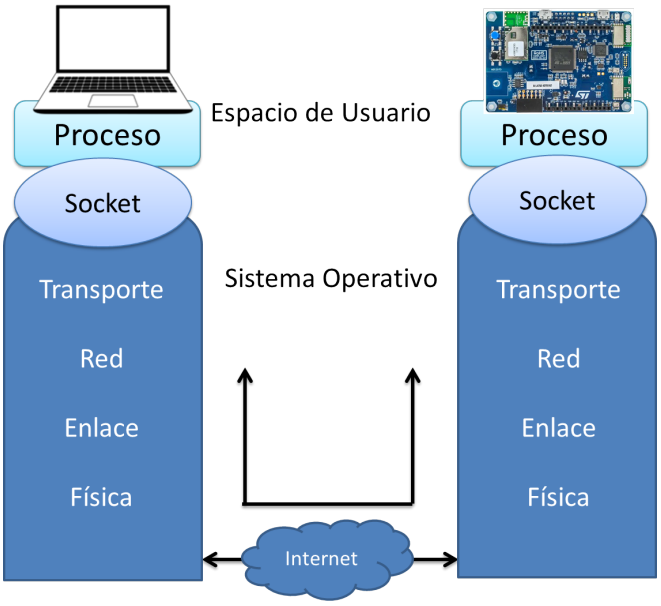

Figura 5.11: Representación abstracta de la comunicación entre procesos cliente-servidor. [24]

Un socket TCP conectado se identifica por cuatro valores: Las dos direcciones IP (cliente y servidor) y el número de puerto del cliente y el servidor.

Para preparar el servidor y trabajar mediante el uso del API de los sockets, se deben seguir los siguientes pasos que se aprecian en la figura 5.12

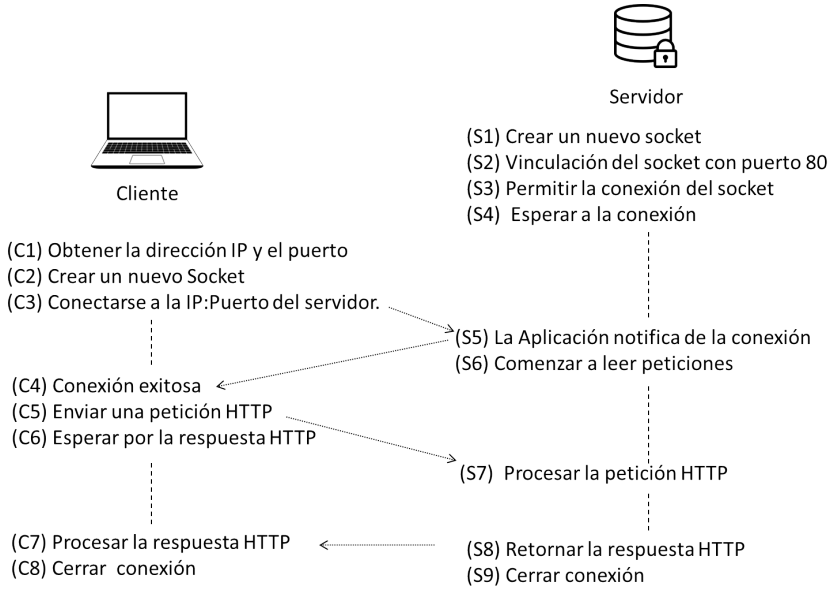

Figura 5.12: Pasos que siguen el cliente y el servidor para poder establecer una comunicación mediante el uso de una interfaz de sockets TCP. [23]

En el proyecto en el que se trabajó, para habilitar el servidor y establecer la comunicación cliente-servidor, se siguieron los siguientes pasos:

- Configurar e inicializar el servidor.
- Esperar la conexión y petición del cliente.
- Envío de la página Web.
- Cerrar la conexión actual Cliente-Servidor.
- En espera de una nueva petición del cliente.
- Recepción de petición del cliente para ejecutar eventos.
- Ejecución de eventos.
- Cerrar la conexión actual Servidor-Cliente.

5.2.2.1. Configurar e inicializar el servidor.

En ésta etapa del proceso se definió el socket a utilizar, el protocolo de transporte, la cantidad de peticiones para la conexión TCP, así como el puerto. Una vez que se tenía definida la información se ejecutaron los comandos AT correspondientes para inicializar el servidor con la configuración definida.

5.2.2.2. Esperar la conexión de un cliente en el servidor

Cuando se ingresa la dirección IP de Servidor en un navegador (el cual funge como cliente), se envía una solicitud al servidor para que éste mande la información para desplegar la página WEB. En ésta sección del código se verificó que la petición hecha por el cliente contara con la sintaxis adecuada para que pudieran ser ejecutados con los comandos AT y detectará cuando un cliente intentara abrir la comunicación con el servidor (Apertura de la página Web). Se obtuvo información como el puerto remoto, la IP remota y el puerto local para ser guardada en variables

5.2.2.3. Envío de la página Web

Una vez que el servidor detecta la petición hecha por el cliente éste envía los datos o la información del archivo HTML para que la página WEB puede ser visualizada en el navegador.

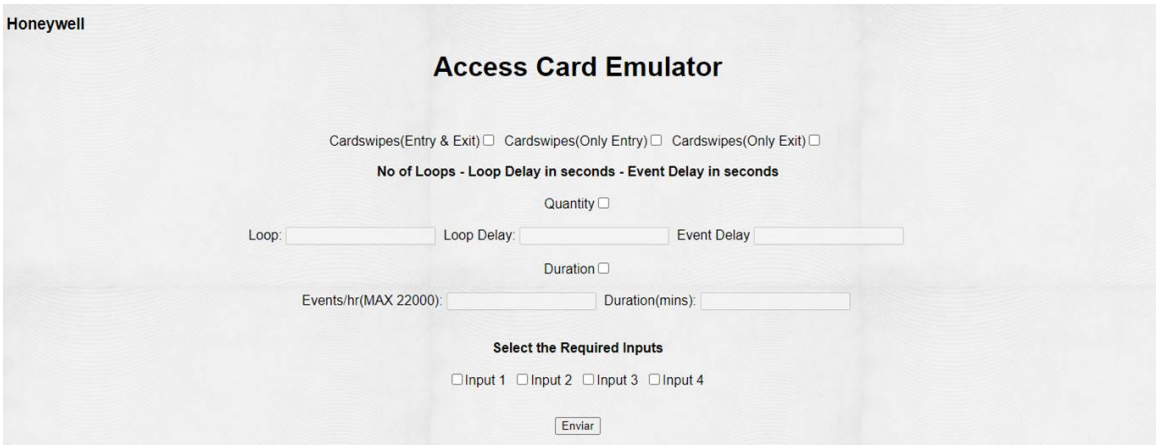

Figura 5.13: Página Web para que el usuario pudiera realizar la selección de eventos a realizar. Diseño realizado por el ingeniero S, Ashokkumar. [36]

5.2.2.4. Cerrar la conexión actual Cliente-Servidor

Una vez concluida la recepción y el envío de datos entre el cliente y el servidor, los canales de comunicación de cada uno deben ser cerrados para poder ser reabiertos cuando sea necesario (Con una nueva petición del cliente hacia el servidor). Para éste punto se ha desplegado la página Web y el cliente puede visualizarla en el navegador.

5.2.2.5. En espera de una nueva petición del cliente

El siguiente paso fue esperar que el cliente seleccionara los eventos que quería ejecutar dentro de la página web. Para que la información pudiera ser mandada mediante el comando post, se utilizó un botón que al ser presionada mandaba los datos seleccionados en cuerpo de la petición HTTP, mediante un comando post hacia el servidor.

5.2.2.6. Recepción de petición del cliente para ejecutar eventos

Una vez que se recibieron los eventos requeridos por el usuario, dentro del programa se llevó a cabo un análisis y filtrado de información para recopilar y ordenar los datos. Con el fin de analizar la petición del cliente y saber que eventos era necesario ejecutar.

5.2.2.7. Ejecución eventos

Una vez que se tuvieron los datos se procedió a ejecutar los eventos correspondientes. El emulador tenía dos formas de funcionar en base a un tiempo definido por el usuario y de acuerdo a una cierta cantidad de eventos seleccionados por el usuario.

- Modo cantidad de eventos. Cuando se seleccionaba ésta modalidad. Dentro de la página Web se tenía que definir la cantidad de loops que se llevarían a cabo, el retraso que tendría cada uno de ellos y la duración de cada uno de los eventos (Movimiento de servo y encendido de Relays). De la misma forma, se debía especificar que periféricos se controlarían, mediante la selección de cajas de texto. Las combinaciones posibles queda de la siguiente forma, una sola modalidad para controlar el servo (Only Entry , Exit and Entry, Only Exit) combinada o no con el encendido de Relays. La selección para operar estos últimos es libre, pudiendo escoger ninguno, uno o hasta cuatro (Input1, Input2, Input3 e Input4).

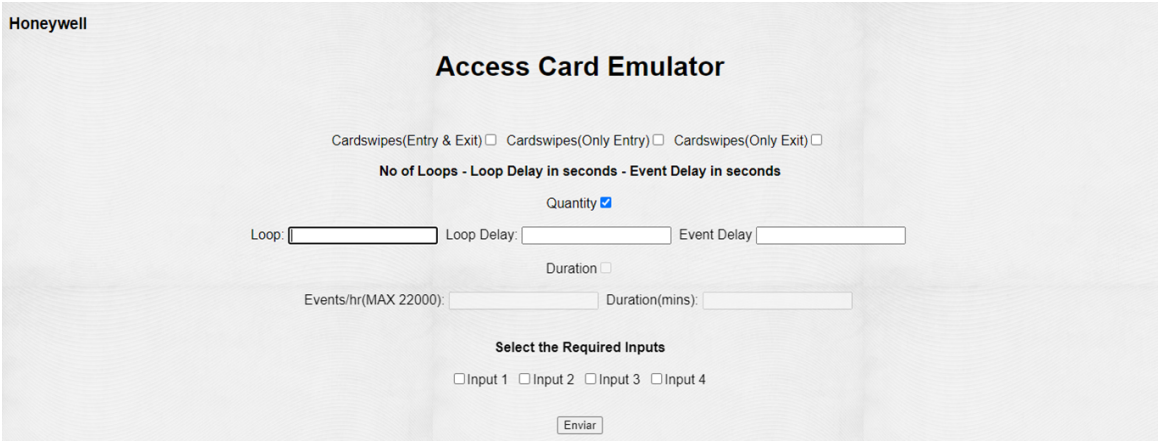

Figura 5.14: Casilla " Quantity "seleccionada por el usuario, lo que provocará una ejecución de eventos en la modalidad cantidad de loops. Diseño realizado por el Ingeniero S, Ashokkumar. [36]

- Modo Cantidad de tiempo: Cuando se escoge la modalidad tiempo, el usuario debe especificar como datos de entrada la velocidad de ejecución de eventos en el campo llamado EPH (Event per Hour), el tiempo de operación, así como los periféricos que serán activados.
  Para activar el servo sólo se debe seleccionar una casilla de entre las opciones Only entry, Entry & Exit, Only exit. Para activar los Relays se debe seleccionar aquellos que se desean controlar (Input 1, Input 2, Input 3, Input 4), aquí es posible seleccionar la cantidad de casillas deseadas. De igual forma, Es posible combinar la selección de Realys con la del Servomotor.

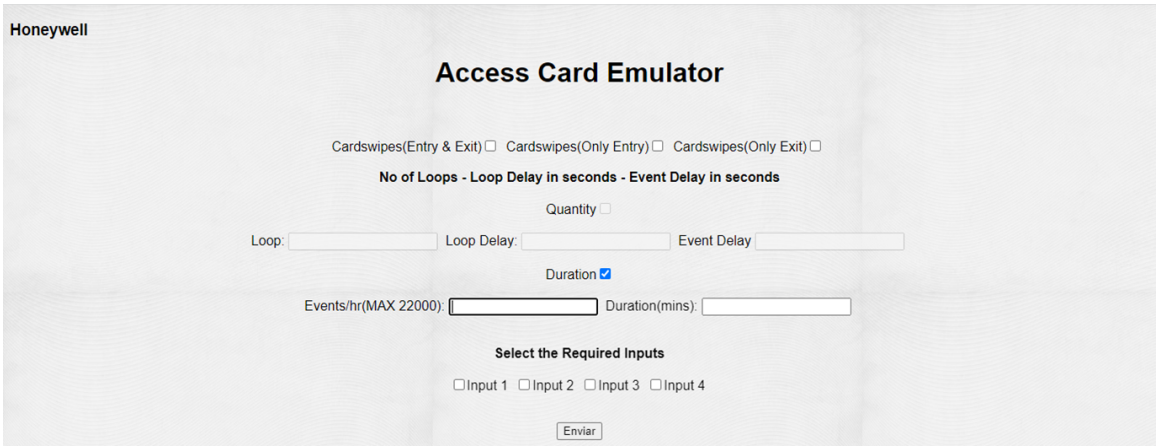

Figura 5.15: Casilla " Duration "seleccionada por el usuario, lo que provocará una ejecución de eventos en la modalidad cantidad de tiempo. Diseño realizado por el Ingeniero S, Ashokkumar. [36]

5.2.2.8. Cerrar la conexión actual Servidor-Cliente

Una vez que se realizó la transmisión de datos. Es necesario cerrar nuevamente el canal de comunicación entre el servidor y el cliente. El Diagrama de flujo del proceso a grandes rasgos se muestra en la figura 5.16

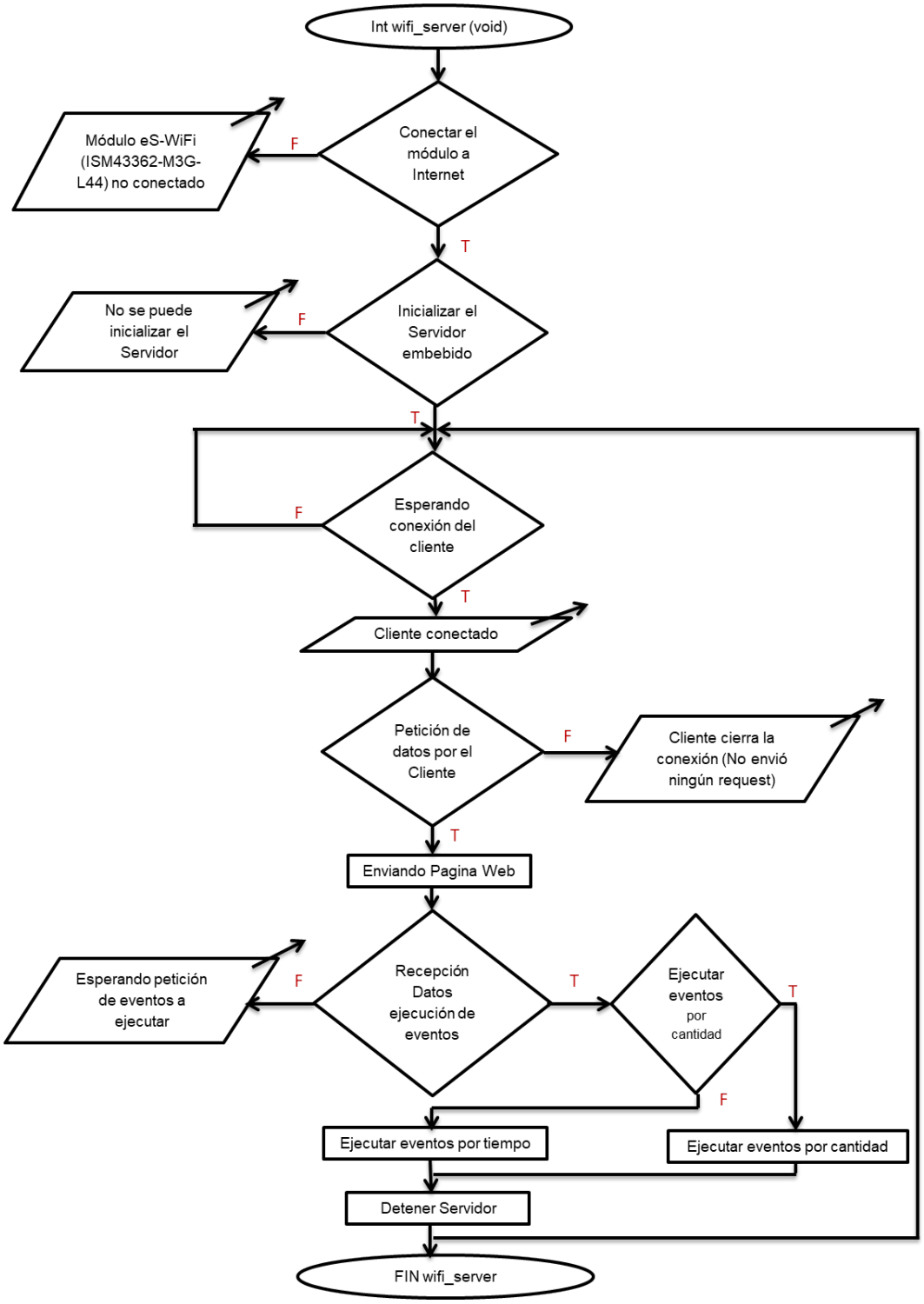

Figura 5.16: Diagrama de flujo que describe el proceso seguido para la instancia y uso del servidor embebido. [31]

5.2.3. Control del Servomotor

Dentro del programa se realizaron las siguientes funciones:

- Escoger y configurar el pin para generar una señal PWM
- Configurar la onda PWM
- Realizar funcionalidad, mover servomotor

5.2.3.1. Escoger y configurar el pin para generar una señal PWM

En ésta sección se configuro el reloj correspondiente del pin, se indicó el pin a utilizar y se asoció su reloj con el timer que sería configurado para que se pudiera general una señal PWM con diferentes anchos de pulsos y de esa forma fuera posible el movimiento del servomotor.
Es importante mencionar, que no en todos los pines de un microcontrolador es posible generar una señal PWM, por lo que se debe tener precaución de elegir alguno que proporcione ésta función. Estos datos pueden ser consultados en la hoja de datos de cada microcontrolador.

5.2.3.2. Configurar la onda PWM

En ésta sección se configuró y se inicializó el timer, para que éste estuviera en resonancia con la frecuencia de operación servomotor.
Para ello como primer paso se seleccionó un timer compatible con el puerto y pin que se iba utilizar. Como siguiente paso se buscó configurar el valor de su prescalizador para reducir su frecuencia de operación de 80 [MHZ] a 1 [MHZ]. El principal motivo de ello fue encontrar una frecuencia base que nos ayudará a generar la frecuencia que necesitaba el servomotor, la cual se encontró que era de 50 MHz (20 ms).

El modelo matemático que describe éste echo queda expresado de la siguiente forma:

$$f = CM * R \tag{5.1}$$

Donde:
f= Nueva frecuencia base del timer del microcontrolador ($\frac{[Ciclos]}{[Segundos]}$)
CM=Frecuencia inicial del microcontrolador (Conteos realizados por el reloj interno del microcontrolador durante cada segundo ($\frac{[Conteos]}{[Segundo]}$))
R= Resolución ($\frac{[Ciclos]}{[Conteos]}$)

La resolución se puede definir como la nueva cantidad de conteos que se realizarán en un ciclo, en los microcontroladores de la familia STM32 a esa cantidad de conteos se le suele llamar preescalizador y nos sirve para dividir la frecuencia base del micontrolador y de ese modo generar nuevas frecuencias. La ecuación matemática que lo describe es:

$$R = \frac{1\ [Ciclo]}{N[Conteos]} = \frac{1\ [Ciclo]}{PSC + 1[Conteos]} \tag{5.2}$$

Dónde:
PSC=prescalizador
N = Número / Cantidad
En la ecuación 5.2 el valor (+1) viene del hecho de que en programación la posición 0 también es considerada, por lo que debe tenerse en cuenta.
Sustituyendo la ecuación 5.2 en 5.1 se obtiene la ecuación:

$$f = CM\frac{[Conteos]}{[Segundos]} * \frac{1\ [Ciclo]}{PSC + 1[Conteos]} \tag{5.3}$$

Despejando el Prescalizador PSC de la ecuación 5.3, se obtiene:

$$PSC = \frac{CM}{f} - 1 \tag{5.4}$$

Sabiendo que el reloj interno del microcontrolador es capaz de contar80 000 000 veces cada segundo.

Es decir:

$$CM = (\frac{80000000}{1})[\frac{Conteos}{S}]. \tag{5.5}$$

Y que se desea que el Timer del Microcontrolador tenga una frecuencia de 1 [MHZ].

$$f = 1[MHZ] \tag{5.6}$$

Sustituyendo el valor de 5.5 y 5.6 en 5.3. Se obtienen que:

$$PSC = \frac{80\ [MHz]}{1\ [MHz]} - 1 \tag{5.7}$$

$$PSC = 79$$

Con éste valor del prescalizador fue posible obtener la nueva frecuencia base de 1 [MHz]. Sin embargo, aún era necesario obtener la frecuencia requerida por el Servo de 50[MHz].
Dado a que se necesitaba que el conteo que realizaba el Timer se reiniciara cada vez que se concluyera un ciclo completo, es decir cada 20 ms (para obtener la frecuencia de 50 [MHz]). Se calculó el valor que debía ser asignado al registro ARR (Auto-Relod Register) del microcontrolador para obtener dicha frecuencia. La expresión matemática que modela éste hecho es:

$$f_0 = \frac{f}{ARR} \tag{5.8}$$

Donde:
f_0=Frecuencia del servomotor
ARR= Valor del registro (Auto-Reload Register)

Considerando la frecuencia que necesitaba el servomotor como: $f_0 = 50[HZ]$

Despejando ARR, sustituyendo el valor de f y de f_0, se tiene

$$ARR = \frac{1[MHz]}{50[HZ]} = \frac{1000000[Hz]}{50[HZ]} \tag{5.9}$$

$$ARR = 20000$$

Por otro lado, gracias al valor asignado al registro llamado CCR (Capture Compare Register) fue posible definir el ciclo de trabajo de la onda PWM.

Es posible expresar lo antes planteado de la siguiente forma:

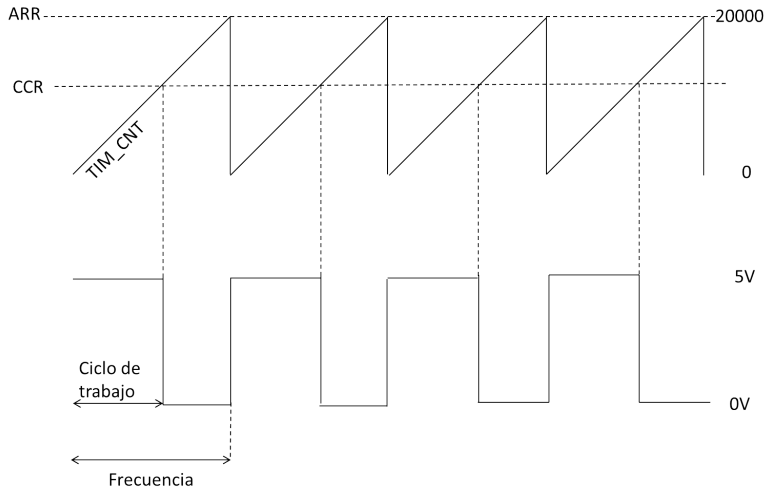

Figura 5.17: Representación gráfica de la generación de una señal PWM, mediante el uso de los diferentes registros que proporciona el microcontrolador utilizado. [27]

Como se puede apreciar en la figura 5.17 ,Si se incrementa el valor asignado al registro ARR, el periodo de la señal también lo hará. Por el contrario, si el valor del registro disminuye el periodo decrecerá.
Mediante el registro CCR, es posible definir el Ciclo de trabajo (Duty Circle) que tendrá la onda cuadrada PWM, dado a que con éste registro se puede definir el tiempo durante el cual la onda estará en estado alto.
El modo de operación de éste registro se enuncia a continuación: Cuando el valor del contador del timer sea menor que el valor establecido para el registro CCR, la onda PWM estará en estado alto (HIGH). Por el contrario, cuando el valor del contador supere el valor establecido en el registro CCR, la onda PWM cambiará a un estado bajo (LOW).
Mientras menor sea el valor asignado al registro CCR, se obtendrá un menor ciclo de trabajo, dado a que la señal PWM permanecerá más tiempo en estado bajo. Por el contrario, mientras mayor sea el valor asignado a éste registro se podrán tener ciclos de trabajo más grande.

Finalmente, es posible controlar la posición de los servomotores, mediante la variación de del ancho de pulso (ciclo de trabajo) de una onda cuadrada PWM. En general se sabe que si se mantiene un cierto periodo de tiempo un impulso en estado alto (5V), modificará su posición como se muestra a continuación:

- 1 milisegundo corresponde a 0°
- 1.5 milisegundos corresponde a 90 °
- 2 milisegundos corresponde a 180 °

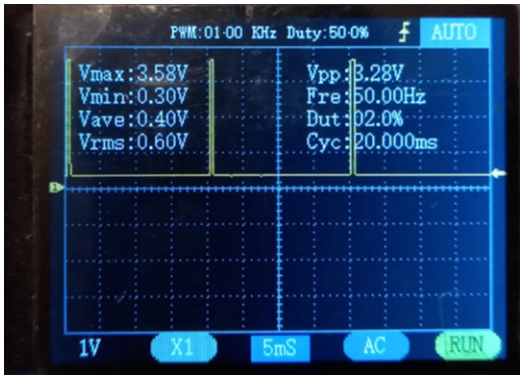

Figura 5.18: Medición en un osciloscopio de la señal PWM generada por el registro correspondiente en dónde se conectaría el servomotor, con un ciclo de trabajo de 2 %. Se mantiene el pulso de señal del servo en estado alto 1 milisegundo, lo que genera un movimiento del servo a la posición de 0 °. [30]

Figura 5.19: Medición en un osciloscopio de la señal PWM generada por el registro co- rrespondiente, en dónde se conectaría el servomotor, con un ciclo de trabajo de 7 %. Se mantiene el pulso de señal del servo en estado alto 1.5 milisegundos, lo que genera un movimiento del servo a la posición de 90 °. [30]

Figura 5.20: Medición en un osciloscopio de la señal PWM generada por el registro correspondiente, en dónde se conectaría el servomotor, con un ciclo de trabajo de 12.4 %. Se mantiene el pulso de señal del servo en estado alto 2 milisegundos, lo que genera un movimiento del servo a la posición de 180 °. [30]

5.2.3.3. Realizar funcionalidad, mover servomotor

Finalmente fue posible mover el servomotor, tras recibir la petición realizada por el usuario. Dentro de la página Web se contaba con tres opciones para el controlar el servomotor:

- Entry & Exit: Con ésta opción se simulaba que el usuario pasaba su tarjeta por la lectora de entrada y salida. Se realizaba un movimiento de 180°con el servomotor, lo que permitía que la tarjeta se deslizara por ambas lectoras ubicadas en los laterales del servomotor.
- Only Entry: Mediante esta alternativa se simulaba que el usuario deslizaba su tarjeta a través de la lectora de entrada. El Servomotor se posicionaba de 90°a 0°para deslizar la tarjeta por la lectora de entrada las veces necesarias.
- Only Exit: Con ésta selección se simulaba que el usuario deslizaba su tarjeta por la lectora de salida. Se realizaba un cambio de posición del servomotor de 90°a 180°, lo que permitía deslizar la tarjeta a través de la lectora de salida.

Sólo era posible escoger un modo de movimiento por evento.

5.2.4. Envío de pulsos eléctricos

Para cumplir el objetivo de prender los Relays. Se realizaron las siguientes tareas.

- Configuración de los puertos, en donde serían conectados los Relays.
- Inicializar los pines de la placa con la configuración realizada.
- Realizar una función para el encendido y apagado de los Relays.

5.2.4.1. Configuración de los puertos, en donde serían conectados los Relays.

Con el fin de enviar pulsos eléctricos a través de Relays fueron ocupados 4 pines. Por ésta razón, como primer paso fue necesario iniciar el reloj de cada uno de los puertos dónde estaban asociados los pines que se iban a utilizar.
Como siguiente paso se procedió a especificar los pines a utilizar, su configurar de los éstos como su modo de operación, activación, frecuencia de trabajo (Velocidad cambio de estado) y el modo de inicio de los pines (LOW/HIGH).

5.2.4.2. Inicializar los pines de la placa con la configuración realizada.

Posteriormente de realizar la configuración, se procedió a inicializar los pines con dichas especificaciones de operación

5.2.4.3. Realizar una función para el encendido y apagado de los Relays.

Una vez que se tenían los puertos y pines que se iban a utilizar, fue posible escribir los algoritmos para manda los impulsos y prender los relevadores, cuando el usuario así lo solicitara. Cundo se seleccionaban las casillas de la página web " Input1"," Input2"," Input3" e " Input4". Se mandaba un identificador asociado a los pines, a través de un comando post que al ser recibido por el servidor embebido éste decodificaba, analizaba y cumplía la tarea de prender los relevadores indicados.

5.2.5. Muestreo de eventos en la pantalla OLED

Para mostrar la información de ejecución en tiempo real como la cantidad de eventos seleccionado, el tiempo de ejecución, la cantidad de eventos realizados, el tiempo de transcurrido y la velocidad de eventos se utilizó una pantalla OLED que funcionaba mediante el protocolo de comunicación I^2C, ésta nos brindó muchas ventajas como una menor necesidad de cables para su funcionamiento, una mejor resolución de imagen y un menor consumo energético.

Capítulo 6

Resultados

6.0.1. Decodificador de tarjetas

Con éste proyecto pude tener mi primer acercamiento tecnología de seguridad, en específico de " Control de accesos", me dio la oportunidad de tener una mejor comprensión de algunos fundamentos, términos y funcionamiento de ésta tecnología.

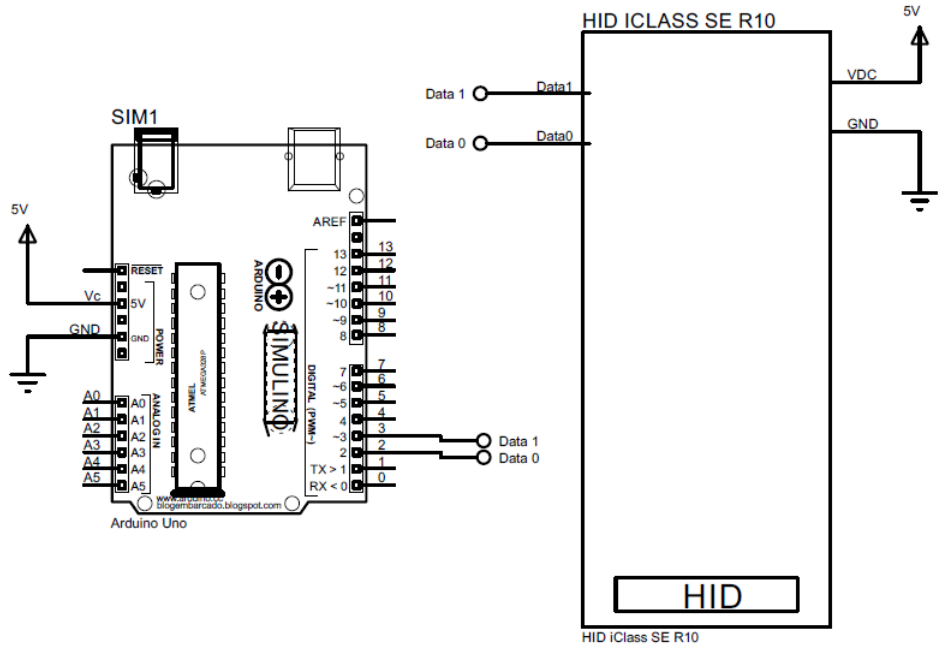

Figura 6.1: Diagrama de conexiones del decodificador. Conexión de la lectora con el microcontrolador. [28]

Adicionalmente éste proyecto se concluyó de manera exitosa, gracias a ello, se logró desarrollar un decodificador de tarjetas funcional que recibía la información decodificada realizada por la lectora tras el deslizamiento de una tarjeta sobre ella, imprimiendo en el puerto serial la información binaria recibida, el formato utilizado, el identificador de sitio (Facility Code) y el número de tarjeta (Card Number). Mediante este decodificador era posible identificar y obtener la información antes mencionadas para los formatos Wiegand de 35 y 26 bits, sin embargo, es posible agregar más formatos dentro del código del programa.
Como áreas de oportunidad dentro de éste proyecto existe la posibilidad de agregar más protocolos Wiegand de con formatos para aumentar la capacidad del dispositivo, así como también existe la oportunidad de mejorar la forma en la que éste dispositivo podrá ser instalado dentro del laboratorio.

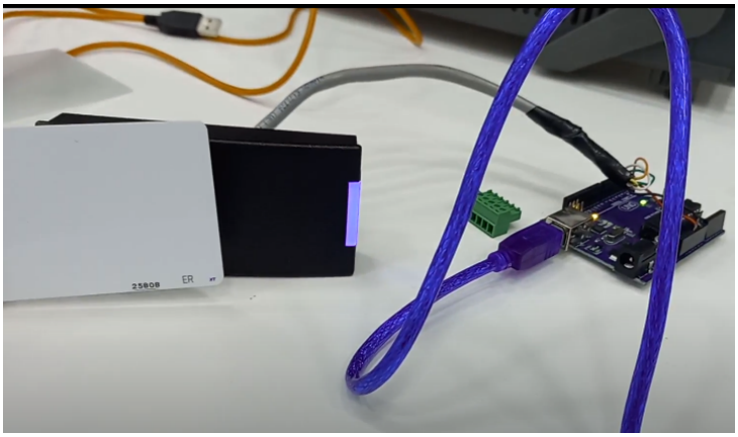

Figura 6.2: Pruebas de conexión y funcionalidad en el prototipo del decodificador. [28]

6.0.2. Emulador IoT embebido para automatizar pruebas

Gracias a éste proyecto me fue posible aplicar fundamentos y conceptos aprendidos durante la carrera de ingeniería mecatrónica como programación, control y automatización, electrónica básica, instrumentación, circuitos digitales. Así como también aprender y adentrarme a nuevas tecnologías, como los dispositivos embebidos, automatización y el Internet de las Cosas IoT. La realización éste proyectos fue muy enriquecedor para mi pues pude expandir mis conocimientos dentro del mundo de la ingeniería.

Por otra parte, gracias al desarrollo de éste emulador a mi equipo de trabajo le fue posible la automatización de pruebas remotamente de ambientes que necesitaban deslizar tarjetas por lectoras. Así mismo, éste dispositivo nos brindó la oportunidad de configurarlo y controlarlo en base al número de eventos o por cantidad de tiempo. Lo que hizo posible tener una mayor flexibilidad y beneficio para reproducir nuestras pruebas.

Por otra parte, se logró cumplir con los objetivos planteados inicialmente para éste proyecto. Donde se obtuvieron los siguientes resultados:

- Habilitar el nuevo microcontrolador, hacer pruebas e implementarlo dentro del espacio de trabajo. Tras llevar a cabo la tarea de migración del código, se tuvieron que adaptar algunas funcionalidades dentro del código y cambiar algunas otras, dado a que los componentes electrónicos de éste nuevo microcontrolador funcionaban con distintos comandos e instrucciones. Para ello, se estudiaron las hojas de datos de sus componentes, que proporcionaban las instrucciones y comandos necesarios para operar a éstos nuevos componentes electrónicos. Dentro del laboratorio, se instaló un prototipo para hacer pruebas y probar su funcionalidad. Dado a que se tenía una menor cantidad de cables y elementos electrónicos para su funcionamiento, fue más fácil adaptarlo al espacio de trabajo, los distintos prototipos se muestran en las figuras 6.3 y 6.4.

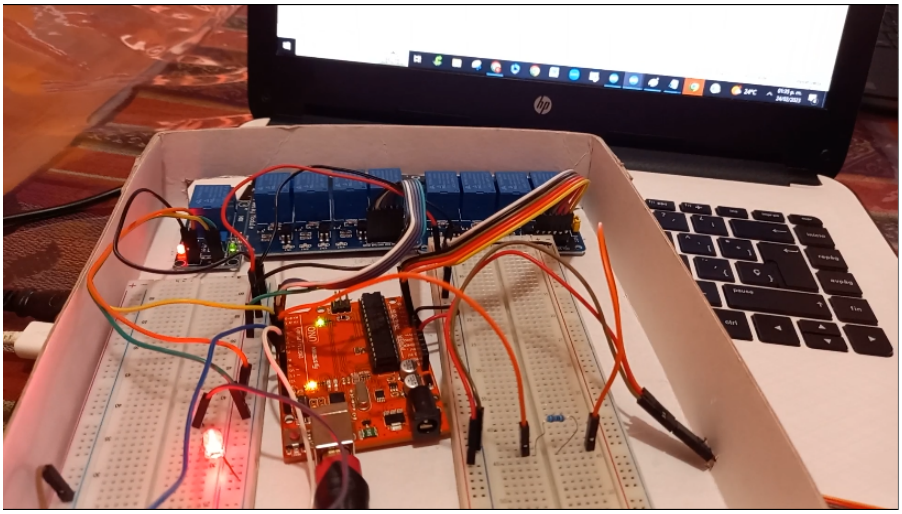

Figura 6.3: Prototipo del primer emulador IoT para automatizar el deslizamiento de tarjetas sobre las lectoras, antes de la migración del código fuente. [36]

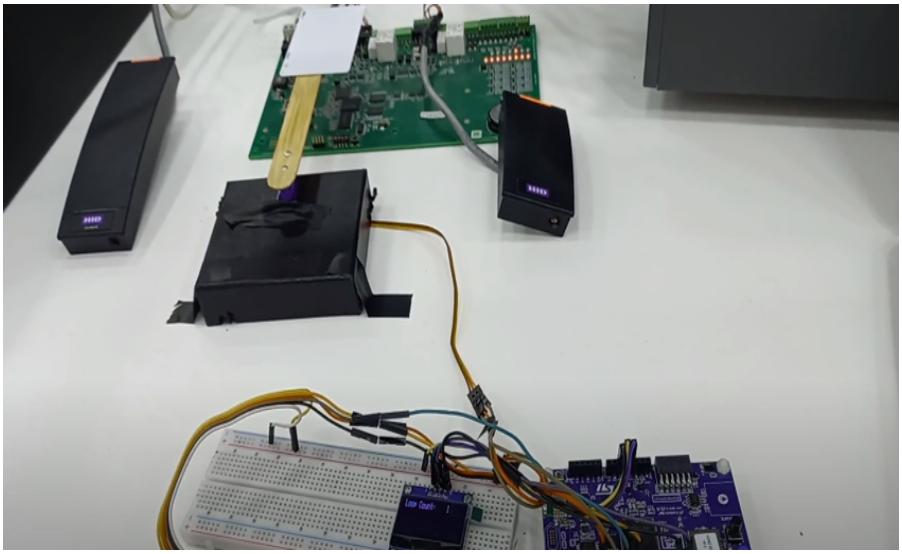

Figura 6.4: Prototipo del emulador IoT para la automatización del deslizamiento de tarjetas sobre las lectoras, después de la migración del código fuente. [32]

- Migración del código a un único y nuevo microcontrolador: Al llevar a cabo ésta tarea se logró tener ventajas adicionales tales como: Un mejor tiempo de respuesta para la realización de los eventos, dado a que en el emulador más reciente no necesitaba dos microcontroladores diferentes como en el primer emulador, para realizar las tareas de establecer la conexión a internet y para ejecutar los eventos. También se redujo la cantidad de espacio necesario para su instalación, así como, también una reducción en los elementos electrónicos para realizar los circuito, esto se muestra en los siguientes diagramas (figuras 6.5 y 6.6):

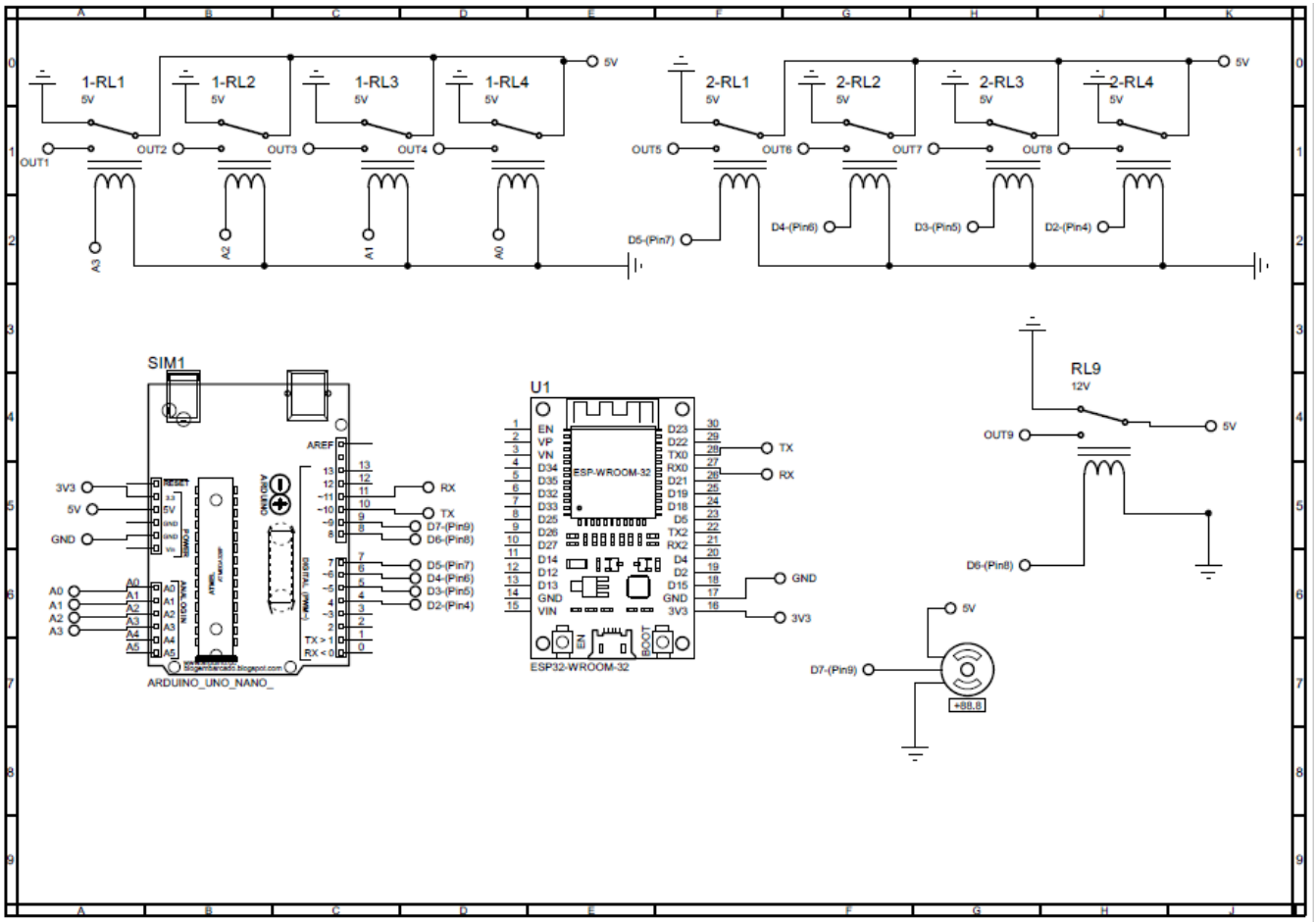

Figura 6.5: Diagrama del emulador original, para la realización de la tarea de automatizar el deslizamiento de tarjetas a través de las lectoras, antes de la migración del código fuente. [36]

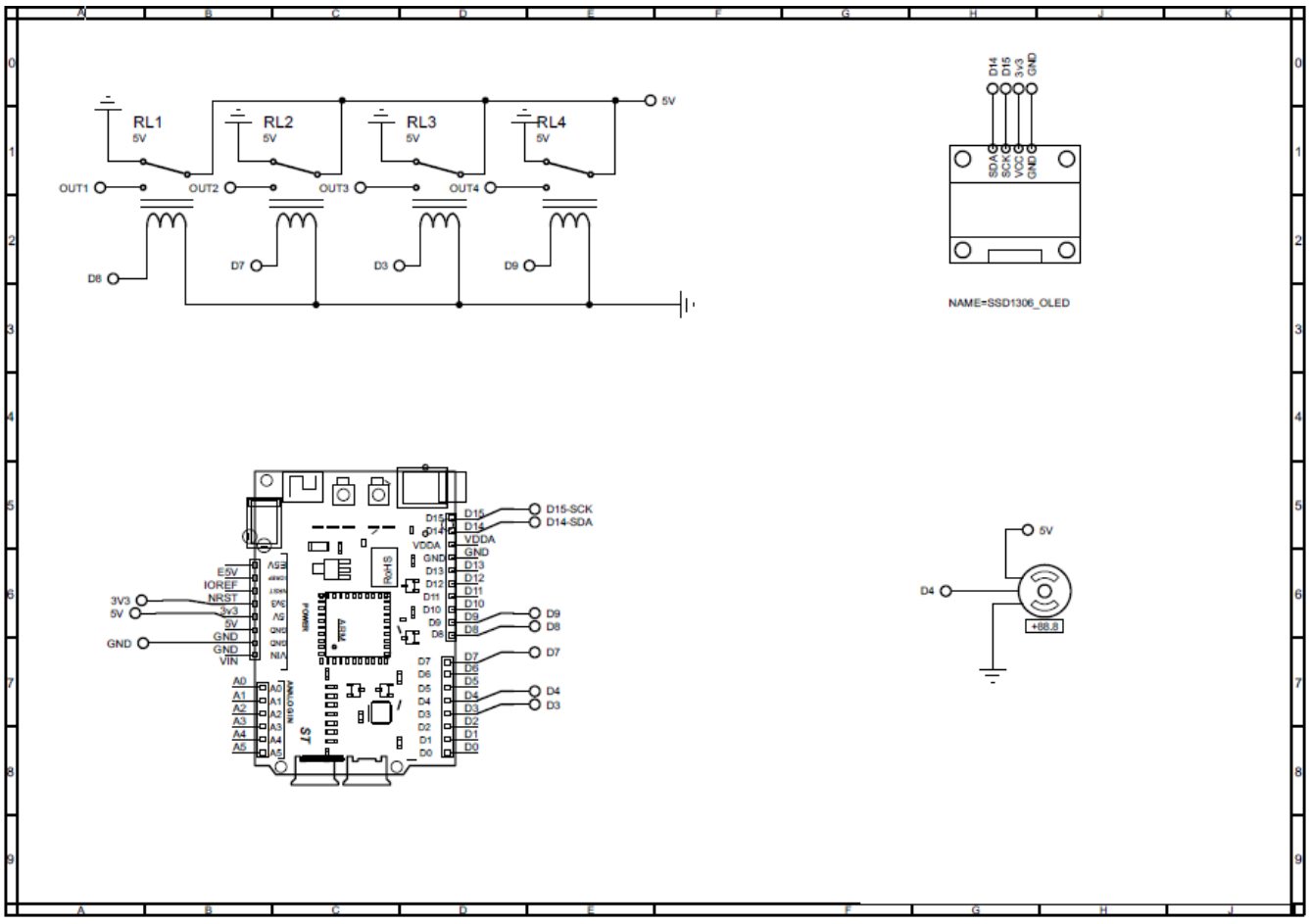

Figura 6.6: Diagrama del nuevo emulador IoT embebido para la automatización de pruebas, que consistían en el deslizamiento automático de tarjetas a través de las lectoras, después de la migración del código. [32]

Capítulo 7

Conocimientos aplicados

Durante mi desempeño laboral, pude fortalecer mis habilidades en la programación, específicamente en el paradigma orientada objetos. Dado a que dicho paradigma, nos permite representar elementos de la vida real dentro de la programación, en forma de " objetos", a los cuales se les pueden asignar características en forma de atributos y funcionalidades en forma de funciones. Lo que permite que la manipulación y uso de la información se vuelve más fácil y eficiente. Pues mediante una abstracción se pueden extraer la esencia de las cosas del mundo real para poder ser representadas virtualmente en forma de clases, las cuales no son más que plantillas o "blue prints"de los elementos representados. Particularmente, gracias a la programación orientada a objetos, se pudo realizar una abstracción de elementos como personas, lectoras, áreas, puertas, tarjetas, bases de datos, cámaras entre otras muchas cosas más; para poder ser representados dentro de la aplicación y poder trabajar con ellos.
Por otro lado, dado a que dentro de la aplicación de la empresa se necesitaba establecer una comunicación entre los dispositivos electrónicos (como paneles y lectoras) con el servidor embebido y con la base de datos, tuve la oportunidad de aprender y analizar más a profundidad la conversión de datos y como se llevaba a cabo la comunicación entre los dispositivos con la aplicación, así como también entre la aplicación con la base de datos. Debido a ello, pude aprender más acerca de las redes de Ethernet, el uso de procedimientos "Procedures"dentro del código, los cuales eran scripts con código SQL que al ser ejecutados efectuaban operaciones dentro de la base de datos.
De igual forma, con ayuda del algunas estrategias y aplicaciones aprendidas durante mi estancia laboral, se me facilitó la detección de errores, encontrar la solución a los problemas e incluso entender mejor a detalle en la ejecución del programa. Algunos ejemplos de ello, fue la aplicación de Git para gestionar, realizar un seguimiento de los cambios hechos dentro del código y mediante el debbugeo se pudo hacer un seguimiento del proceso seguido por los programas. De esa forma, pude entender mejor el comportamiento del programa, analizarlo y identificar los errores. Finalmente, pude cumplir el objetivo de programar un dispositivo que ayudara en la parte de testeo de pruebas. Y con ello, pude aprender a interpretar mejor la arquitectura de los microcontroladores, leer documentación, entender el funcionamiento de los componentes electrónicos y un poco de la programación de bajo nivel, los registros y relojs del microcontrolador STM32L4.
Adicionalmente, pude aprender de la existencia de los protocolos utilizados dentro del área de la seguridad para la transmisión de la información. (Wiegand y OSDP). Sin embargo, dado a que la mayoría de los dispositivos de seguridad aún siguen utilizando el protocolo Wiegand tuve un mayor acercamiento a dicho protocolo, entendí como funciona, como utilizarlo y que existen diferentes formatos.
De igual forma, pude conocer la existencia de los patrones de diseño y del principio SOLID. Dichos principios me ayudaron a seguir buenas prácticas de programación. Dado a que me permitieron tener una guía para que la escritura de mi código fuera más clara y a que tuviera más sentido para otros programadores.
También, pude aprender la metodología Agile SCRUM para la entrega de proyectos y tareas. La cuál consistía en hacer pequeños entregables constantes y reiterativos para lograr una mejora continua en el producto final, trabajando de la mano del cliente, considerando su feedback. Lo que permitía tener una mayor adaptabilidad y flexibilidad en el trabajo diario.

Durante mi trayectoria universitaria pude adquirir las bases y conocimientos necesarios para poder desempeñarme profesionalmente de la mejor manera posible, no sólo técnicamente, sino también humanamente.

Desde los conocimientos adquiridos en materias de tronco común, como álgebra lineal, matemáticas avanzadas, electricidad y magnetismo, por mencionar algunas, que me brindaron los aprendizajes básicos necesarios para explicar fenómenos, principios físicos y entender el trasfondo del funcionamiento de la tecnología utilizada.
Como también, con el aprendizaje adquirido en materias de ingeniería como electrónica básica, análisis de circuitos, circuitos digitales e instrumentación que me capacitaron para contar con las herramientas necesarias para poder interpretar y analizar los diseños de los distintos dispositivos utilizados, implementar instrumentos de medición, entender el funcionamiento de algunos sensores, así como, interpretar, entender la transformación de información de analógica a digital. Esto me proporción tener una mejor compresión del cómo funcionaban los dispositivos electrónicos internamente, los sensores, así como, interpretar las señales y resultados obtenidos. Y al final, pero no menos importante, las materias de humanidades me ayudaron a desarrollar mis habilidades de comunicación, liderazgo y creatividad, para tener un mejor desarrollo como ingeniero dentro de equipos de trabajo multidisciplinarios. Esto me concedió poder trasmitir mis ideas de forma clara, trabajar en equipo y hacerme responsable de liderar y organizar mis actividades asignadas.

Capítulo 8

Conclusión

En el mundo actual, en donde se tienen grandes cantidades de información y es necesario procesarla para realizar tareas, el uso de las computadoras y máquinas se ha vuelto necesaria para ayudar al ser humano a hacer más fácil algunas actividades, tales como: las representaciones esquemáticas de circuitos electrónicos, procesamiento de información, automatización de procesos, identificación de usuarios, monitoreo de zonas y control de accesos. Por ello, gracias a que la programación es el lenguaje por excelencia para poder comunicarse con las computadoras, se ha convertido en una herramienta esencial para poder realizar dichas tareas, al igual que para ejecutar procesos, manipular y obtener información. Durante mi colaboración en la solución ofrecida por ésta empresa, la programación fue de gran utilidad, porque nos permitió darle suporte técnico a los clientes de la aplicación, así como también poder darle mantenimiento a la aplicación como tal, encargada de gestionar usuarios, áreas, bases de datos y dispositivos electrónicos.

De igual forma, gracias a que se pudo elaborar el primer proyecto con éxito, se pudo tener una herramienta que nos ayudó a imprimir a través del puerto serie de la computadora el número de tarjeta y el número de sitio, cuando una tarjeta de usuario era deslizada a través de una lectora conectada al microcontrolador. Lo que permitió visualizar los números identificación de los usuarios en la computadora. Adicionalmente, mediante el uso del decodificador pude comprender más a profundidad el funcionamiento del protocolo de seguridad Wiegand.

Con ayuda del desarrollo del emulador de tarjetas automático, se pudo cumplir con una parte muy importante en el proceso de suporte, la cual es la validación experimental, mediante la generación de pruebas remotamente. Dicha validación se realizaba antes y después de aplicar los cambios en el código con la solución, para entender el problema, observar el comportamiento que se tenía y comprobar que funcionara correctamente después de haber aplicado los cambios.

Tener la posibilidad de realizar prácticas profesionales dentro de una empresa me ayudó a reforzar, comprender y aplicar los conocimientos aprendidos durante la carrera. Me dio la oportunidad tener un mayor panorama del mundo de la ingeniería, aprender nuevas cosas y entender muchas otras a mayor profundidad. Ésta experiencia me permitió desarrollar un crecimiento integral como ingeniero, ya que desarrollé tanto mis habilidades técnicas para aplicar mis conocimientos de Ingeniería, como también, mis habilidades blandas para trabajar en equipo, comunicarme para trasmitir mis ideas, así como a ser pro-activo para participar en proyectos. Cursar la carrera mecatrónica me dio todas las bases necesarias para poder entender problemas de ingeniería y analizarlo desde diferentes perspectivas y de ésta forma encontrar la mejor solución posible. Es decir, dado a que la carrera de mecatrónica es sinérgica, adquirí la habilidad de ver los problemas desde diferentes enfoques de conocimiento y proponer diferentes soluciones que involucren una o varias ramas de la ingeniería.

No me queda más que agradecer a mi amada universidad, profesores, familia amigos por todo el apoyo brindado. Y que, aunque mi camino apenas comienza y nunca se deja de aprender, gracias a todos estos factores tengo una base sólida en conocimientos de ingeniería, así como también todas las herramientas necesarias para culminar con éxito mi carrera y enfrentar todos los retos que vengan por delante.

Bibliografía

[1] «The History of Honeywell» , Honeywell [Online]. Available:https://www.honeywell.com/us/en/company/our-history.

[2] Leens, F, *Introduction to I^2C and SPI protocols*. IEEE Instrumentations and Measurement Magazine,2009.

[3] «eS-WiFi Module User Manual» Inventek Systems, [Online]. Available:http://www.inventeksys.com/iwin/wp-content/uploads/IWIN_Command_Set_Users_Manual.pdf.

[4] «ISM43362-M3G-L44 Product Specification» Inventek Systems, [Online]. Available:https://www.inventeksys.com/wp-content/uploads/ISM43362_M3G_L44_Functional_Spec.pdf

[5] Subhash Chandra, Y. Sanjay Kumar Singh, *An Introduction to Client/Derver computing*..New Dehli,Bangalore:NEW AGE INTERNATIONAL PUBLISHERS

[6] «Cabecera HTTP:Conceptos básicos para usuarios» Digital Guide IONOS, [Online]. Available:https://www.ionos.mx/digitalguide/hosting/cuestiones-tecnicas/cabecera-http/

[7] Mostafa E. Saleh,A. Abdel Nabi, A. Baith Mohamed,(2007) *Load Modeling for Power Flow and Transient Stability Computer Studies at BAKHTAR Network* .Egipto: ResearchGate.

[8] Loukides, M. , Cameron, D,(2002) *TCP/IP Network Administration.*Tercera edición. Estados Unidos: O´Reilly Media.

[9] «¿Qué es TCP/IP?» AVG, [Online]. Available:https://www.avg.com/es/signal/what-is-tcp-ip.

[10] «Understanding Card Data Formats» HID, [Online]. Available:https://www.idesco.com/files/articles/HID%20-%20Understanding%20card%20formats.pdf

[11] M. A. Laughton, D.F. Warne,(2002) *Electrical Engineer's Reference Book.*16va edición. Gran Bretaña. Newnes

[12] «¿Qué es PWM y cómo usarlo?» Solectro, [Online]. Available:https://solectroshop.com/es/blog/que-es-pwm-y-como-usarlo–n38

[13] Banafa, A.,(2023) *Introduction to Internet of Things (IoT)*. Dinamarca. River Publishers.

[14] «¿Qué es el IoT?» Oracle, [Online]. Available:https://www.oracle.com/mx/internet-of-things/what-is-iot/

[15] «Manual de referencia» ST, [Online]. Available:https://www.st.com/resource/en/reference_manual/rm0351-stm32l47xxx-stm32l48xxx-stm32l49xxx-and-stm32l4axxx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf

[16] «Hoja de datos» ST, [Online]. Available:https://www.st.com/resource/en/datasheet/stm32l471qe.pdf

[17] «EL PROTOCOLO WÍEGAND» Picmania, [Online]. Available: http://picmania.garcia-cuervo.net/conceptos_wiegand.php

[18] «Sistemas Numéricos» Universidad Don Bosco, [Online]. Available: https://www.udb.edu.sv/udb_files/recursos_guias/informatica-tecnologico/redes-de-comunicacion/2020/i/guia-1.pdf

[19] «Introduction to SPI Interface» ,Dhaker, P. [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html

[20] «Generalidades del protocolo HTTP» ,Mdn web docs_. [Online]. Available: https://developer.mozilla.org/es/docs/Web/HTTP/Overview

[21] «Protocolo de control de transmisiones (Transmission Control Protocol)»,IBM . [Online]. Available: https://www.ibm.com/docs/es/aix/7.1?topic=protocols-transmission-control-protocol

[22] M.Alani, M, *Guide to OSI and TCP/IP Models*.New York, Springer. 2014.

[23] David Gourley, Brian Totty, Marjorie Sayer, Anshu Aggarwal, Sailu Reddy , *HTTP: The Definitive Guide* . O´Reilly. 2002.

[24] «La interfaz de los sockets», Elvira Baydal. Universidad Politécnica de Valencia. [Online]. Available: https://www.youtube.com/watch?v=teCS58E5YMY.

[25] «UDP: What is the User Datagram Protocol?», Digital Guide IONOS. [Online]. Available: https://www.ionos.com/digitalguide/server/know-how/udp-user-datagram-protocol/.

[26] G. Blank.A,(2004) *TCP/IP Foundations*.Estados Unidos: San Francisco. SYBEX.

[27] «Stm32 Timers in PWM mode», Amaya, E. [Online]. Available: https://www.youtube.com/watch?v=qAZjdx71ePc .

[28] «Access control», Samir Tawfik. [Online]. Available: https://www.youtube.com/watch?v=zWKJIpcN9JU.

[29] «Pro-Watch 7000 Intelligent Controller Datasheet», Honeywell. [Online]. Available: https://prod-edam.honeywell.com/content/dam/honeywell-edam/hbt/en-us/documents/literature-and-specs/datasheets/HBT-Security-PW7K-Datasheet.pdf.

[30] «STM32 Guide 3: PWM + Timers», Mitch Davis. [Online]. Available: https://www.youtube.com/watch?v=AjN58ceQaF4list=RDCMUCtqVwrq2sogclCiqIvV-IUgindex=2.

[31] «STM32CubeL4», KORKAD y ASELSTM. [Online]. Available: https://github.com/STMicroelectronics/STM32CubeL4/blob/master/Projects/B-L4S5I-IOT01A/Applications/WiFi/WiFi_HTTP_Server/Src/main.c.

[32] «User Manual», ST. [Online]. Available: $https://www.st.com/resource/en/user_manual/um2153-discovery-kit-for-iot-node-multichannel-communication-with-stm32l4-stmicroelectronics.pdf$

[33] «Reference Manual», ST. [Online]. Available: $https://www.st.com/resource/en/reference_manual/dm00310109-stm32l4-series-advanced-armbased-32bit-mcus-stmicroelectronics.pdf$

[34] Ballad,B. Tricia,B. K. Banks Erin,(2011) *Access Control Authentication, and Public Key Infrastructure*. Canadá: Jones Bartlett LEARNING.

[35] «Access Control Training», SilmarElectronicsInc. [Online]. Available: $https://www.youtube.com/watch?v=SE5KFDPh8-M$

[36] Fuentes internas de la empresa.