DESARROLLO DE UNA SOLUCIÓN COMERCIAL TECNOLÓGICA BASADA EN SOFTWARE Y HARDWARE LIBRES

TESIS

PARA OBTENER EL GRADO DE

MAESTRO EN
SISTEMAS INTELIGENTES MULTIMEDIA

PRESENTA

ING. HÉCTOR EDMUNDO RAMÍREZ GÓMEZ
ASESOR: DR. ROGELIO ÁLVAREZ VARGAS

GUADALAJARA, JALISCO, JULIO 2020

GOBIERNO DE MÉXICO

CONACYT Consejo Nacional de Ciencia y Tecnología

CIATEQ

Querétaro, Qro., 10 de julio del 2020.

Mtro. Geovany González Carlos
Coordinador Académico de Posgrado
CIATEQ, A.C.

Los abajo firmantes, miembros del Comité Tutorial del Ing. Héctor Edmundo Ramírez Gómez, una vez revisado su Proyecto Terminal de tesis/tesina, titulado "Desarrollo de una solución comercial tecnológica basada en software y hardware libres" **autorizo** que el citado trabajo sea presentado por el alumno para su revisión, con el fin de alcanzar el grado de **Maestro en Sistemas Inteligentes Multimedia**

Sin otro particular por el momento, agradezco la atención prestada.

Firma

Dr. Regelio Alvarez Vargas
Asesor Académico

Firma

Dr. José Ramón Morales Avalos
Asesor en Planta

GOBIERNO DE MÉXICO

CONACYT Consejo Nacional de Ciencia y Tecnología

CIATEQ

Zapopan, Jalisco, 16 de Julio del 2020.

Dra. María Guadalupe Navarro Rojero
Directora de Posgrado
CIATEQ, A.C.

Por medio de la presente me dirijo a usted en calidad de Revisor del proyecto terminal del alumno **HÉCTOR EDMUNDO RAMÍREZ GÓMEZ**, cuyo título es:

"DESARROLLO DE UNA SOLUCIÓN COMERCIAL TECNOLÓGICA BASADA EN SOFTWARE Y HADWARE LIBRES"

Después de haberlo leído, corregido e intercambiado información con el alumno, y realizado los cambios que le fueron sugeridos, puede ser autorizada su impresión, a fin de que se inicien los trámites correspondientes para su defensa.

Sin otro particular por el momento, y en espera de que mis sugerencias sean tomadas en cuenta en beneficio del estudiante y la Institución, agradezco la atención prestada.

Atentamente,

Dr. Nary Delgadillo Checa

AGRADECIMIENTOS

Para quienes son los responsables de tener lo que tengo y de ser quien soy; mi familia.

RAMGO

RESUMEN

El presente documento muestra el procedimiento sobre cómo se logra diseñar y construir una solución tecnológica que se adapta a instalaciones preexistentes de control de acceso en una empresa cliente, basado en la lectura de tarjetas NFC y conexión inalámbrica al sistema de administración principal. Se limita a usar tecnologías open-source y aunque es diseñado específicamente para el escenario solicitado se configura bajo una estructura modular que permite su adaptación a otras aplicaciones. Se desarrolla con base en metodologías de desarrollo de proyectos de hardware y de emprendimiento ágil, al mismo tiempo que el trabajo experimental se complementa con iteraciones para validar el funcionamiento e integración de sus componentes y se genera documentación de todo el proceso. A partir de la integración de distintas tecnologías se logra generar un sistema funcional con capacidad de mantenimiento remoto, logrando aumentar la eficiencia de algunos de los principales procesos de negocio de la empresa que compra la implementación del proyecto. A consecuencia se concluye que la investigación científica aplicada tiene un alto potencial para la generación de soluciones tecnológicas de utilidad para la industria, demostrando que esta formalidad favorece la comercialización de proyectos tecnológicos y genera oportunidades de consultoría profesional. Todo el trabajo desarrollado es liberado a la comunidad bajo un licenciamiento abierto con el objetivo de que pueda ser utilizado como plataforma para el desarrollo de soluciones tecnológicas en las oportunidades de aplicación donde se identifique utilidad.

Palabras clave: Ingeniería y tecnología, Ingeniería de los ordenadores, Sistemas de control del entorno, Tecnología electrónica, Diseño de circuitos, Open source.

ABSTRACT

This document shows the procedure on how to design and build a technological solution that adapts to pre-existing control installations in a client company, based on the reading of NFC cards and wireless connection to the main administration system. It is limited to using open-source technologies and although it is specifically designed for the requested scenario, it is configured under a modular structure that allows its adaptation to other applications. It is developed based on hardware project development and agile entrepreneurship methodologies, while the experimental work is complemented by iterations to validate the operation and integration of its components and documentation of the entire process is generated. From the integration of different technologies it is possible to generate a functional system with remote maintenance capacity, managing to increase the efficiency of some of the main business processes of the company that purchases the implementation of the project. As a result, it is concluded that applied scientific research has a high potential for the generation of useful technological solutions for the industry, demonstrating that this formality favors the commercialization of technological projects and generates professional consulting opportunities. All the work developed is released to the community under an open licensing so that it can be used as a platform for the development of technological solutions in application opportunities where usefulness is identified.

Keywords: Engineering and technology, Computers engineering, Environment control systems, Electronics engineering, Circuit design, Open source.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

GLOSARIO

API: Significa "Application Programming Interface". Una API es un conjunto de comandos, funciones, protocolos y objetos que los programadores pueden usar para crear software o interactuar con un sistema externo.

Bluetooth: Esta tecnología inalámbrica permite la comunicación entre dispositivos compatibles con Bluetooth. Se usa para conexiones de corto alcance entre computadoras de escritorio y portátiles, PDA (como el Palm Pilot o Handspring Visor), cámaras digitales, escáneres, teléfonos celulares e impresoras.

BMC: Business Model Canvas es una plantilla de gestión estratégica y Lean Startup para desarrollar nuevos modelos de negocios o documentarlos. Es un modelo visual con elementos que describen la propuesta de valor, la infraestructura, los clientes y las finanzas de una empresa o producto.

Business Plan: Un plan de negocios es un documento que resume los objetivos operativos y financieros de una empresa y contiene los planes y presupuestos detallados que muestran cómo se deben cumplir los objetivos. Es la hoja de ruta para el éxito de su negocio. Para cualquiera que inicie un negocio, es un primer paso vital.

CNC: CNC significa Control Numérico por Computadora. Esto significa que una computadora convierte el diseño producido por el software Computer Aided Design (CAD) en números. Los números pueden considerarse como las coordenadas de un gráfico y controlan el movimiento del cortador. De esta manera, la computadora controla el corte y la conformación del material.

DFM: Design for manufacturing (DFM) es una técnica de diseño para facilitar la fabricación de una variedad de piezas que constituirían el producto final después del ensamblaje. El diseño para la manufactura se enfoca en minimizar las complejidades involucradas en las operaciones de manufactura, así como en reducir el costo total de producción de las partes.

Firmware: El firmware es un programa de software o un conjunto de instrucciones programadas en un dispositivo de hardware. Proporciona las instrucciones necesarias sobre cómo el dispositivo se comunica con el otro hardware de la computadora.

Form-factor: Un factor de forma es el diseño y funcionalidad general del hardware electrónico, que generalmente se destaca por una característica, como un teclado QWERTY, una pantalla táctil o la forma en que se abre y se cierra el dispositivo. El término se usa para especificar el tamaño, la configuración o la disposición física y las especificaciones del hardware del dispositivo, junto con un enfoque en sus componentes internos.

GPIO: Significa "entrada / salida de propósito general". GPIO es un tipo de pin que se encuentra en un circuito integrado que no tiene una función específica. Si bien la mayoría de los pines tienen un propósito específico, como enviar una señal a un determinado componente, la función de un pin GPIO es configurable y puede ser controlada por software.

HTTP: Significa "Protocolo de transferencia de hipertexto". HTTP es el protocolo utilizado para transferir datos a través de la web. Forma parte del conjunto de protocolos de Internet y define los comandos y servicios utilizados para transmitir datos de páginas web.

IDE: Un software IDE es una aplicación que los desarrolladores utilizan para crear programas informáticos. En este caso, "integrado" se refiere a la forma en que se combinan múltiples herramientas de desarrollo en un solo programa

IoT: El Internet de las cosas, comúnmente abreviado "IoT", es un término general que se refiere a cualquier cosa conectada a Internet. Incluye dispositivos informáticos tradicionales, como computadoras portátiles, tabletas y teléfonos inteligentes, pero también incluye una lista creciente de otros dispositivos que recientemente se han habilitado para Internet. Los ejemplos incluyen electrodomésticos, automóviles, aparatos electrónicos portátiles, cámaras de seguridad y muchas otras cosas.

JSON: JavaScript Object Notation, es un formato de texto sencillo para el intercambio de datos. Comúnmente usado en comunicaciones a través de internet por medio de llamadas HTTP, además de ser usado también en archivos de configuración de software.

kBd: Significa "kilobaudios", haciendo referencia a mil baudios. Los baudios son usados para describir el rango máximo de oscilación de una señal electrónica, y hace referencia a la velocidad de transmisión de datos.

MAC: Significa "Dirección de control de acceso a medios" y no, no está relacionada con las computadoras Apple Macintosh. Una dirección MAC es un número de identificación de hardware que identifica de forma única a cada dispositivo en una red. La dirección MAC se fabrica en cada tarjeta de red, como una tarjeta Ethernet o una tarjeta Wi-Fi, y por lo tanto no se puede cambiar.

Machine Learning: El aprendizaje automático es una aplicación de inteligencia artificial (IA) que proporciona a los sistemas la capacidad de aprender y mejorar automáticamente a partir de la experiencia sin ser programado explícitamente. El

aprendizaje automático se centra en el desarrollo de programas informáticos que pueden acceder a los datos y utilizarlos para aprender por sí mismos.

MCU: Un solo chip que contiene el procesador (CPU), memoria no volátil (memoria flash o ROM) para el programa, memoria volátil (RAM) para procesar los datos, un reloj y una unidad de control de E / S. Las unidades de microcontroladores (MCU) están disponibles en numerosos tamaños y arquitecturas

NFC: Son las siglas para "Near Field Communication", o comunicación de corto rango. Esta tecnología habilita una comunicación segura y simple entre dispositivos electrónicos. Su rango de comunicación es relativamente corto, de apenas 10 cm, lo cual es intencional para los casos de aplicación específicos donde se emplea.

OBD: El diagnóstico a bordo (OBD) es un término automotriz que se refiere a la capacidad de autodiagnóstico e informe de un vehículo. Los sistemas OBD le dan al propietario del vehículo o al técnico de reparación acceso al estado de los diversos subsistemas del vehículo.

OTA: Las señales OTA que se utilizan para programar, actualizar, configurar y aprovisionar dispositivos de hardware se conocen colectivamente como OTAP (programación por aire)

PCB: Significa "placa de circuito impreso". Una PCB es una placa delgada hecha de fibra de vidrio, epoxi compuesto u otro material laminado. Las vías de conducción se graban o "imprimen" en la placa, conectando diferentes componentes en la PCB, como transistores, resistencias y circuitos integrados.

PoC: Una prueba de concepto (PoC) es una demostración, cuyo propósito es verificar que ciertos conceptos o teorías tengan el potencial para una aplicación en el mundo real. Por lo tanto, PoC es un prototipo que está diseñado para determinar la viabilidad, pero no representa productos entregables.

PRD: Un documento de requisitos del producto (PRD) es un documento que contiene todos los requisitos para un determinado producto. Está escrito para permitir que las personas entiendan lo que debe hacer un producto. Sin embargo, un PRD generalmente debe evitar anticipar o definir cómo lo hará el producto para luego permitir a los diseñadores e ingenieros de la interfaz usar su experiencia para brindar la solución óptima a los requisitos.

SCADA: El Control de Supervisión y Adquisición de Datos (SCADA) se refiere a los sistemas de control industrial (ICS) que se emplean para controlar y hacer un seguimiento de los equipos o una planta en industrias como el control de desechos y agua, telecomunicaciones, energía, transporte y refinación de petróleo y gas. SCADA es un sistema informático utilizado para recopilar y analizar datos en tiempo real.

Scope Creep: Se refiere a un proyecto que ha visto crecer sus objetivos originales mientras está en progreso. Como sugiere el término, el alcance del proyecto es un proceso sutil que comienza con pequeños ajustes y termina en proyectos que demoran más en completarse o incluso fallan antes de que se terminen. Incluso si se completa el proyecto, el scope creep puede dar resultados finales que no se parecen en nada a lo que originalmente se había previsto.

SoC: Significa "Sistema en un chip". Un SoC (pronunciado "S-O-C") es un circuito integrado que contiene todos los circuitos y componentes necesarios de un sistema electrónico en un solo chip.

USD: El USD es la abreviatura del dólar estadounidense en el mundo del comercio de divisas.

WiFi: Wi-Fi es una tecnología de red inalámbrica que permite a las computadoras y otros dispositivos comunicarse a través de una señal inalámbrica. Describe los componentes de red que se basan en uno de los estándares 802.11 desarrollados por el IEEE y adoptados por la Wi-Fi Alliance.

WPS: WPS significa configuración protegida de Wi-Fi. Es un estándar de seguridad de redes inalámbricas que intenta que las conexiones entre un enrutador y los dispositivos inalámbricos sean más rápidas y fáciles.

VPN: *Virtual Private Network*, o Red Privada Virtual. Se refiere al proceso de establecer una conexión privada y segura entre uno o más dispositivos de red en ubicaciones remotas, donde no comparten una misma red local.

CAPÍTULO 1: INTRODUCCIÓN

En la industria y el mercado tecnológicos existe la coexistencia de sistemas de diferentes proveedores que interactúan en un mismo entorno, pero estos sistemas no siempre tienen una compatibilidad absoluta o presentan fallas en su funcionamiento debido a instalaciones y configuraciones equivocadas. Aunque existen estándares que buscan establecer una colaboración fluida y sin complicaciones, en algunos casos es necesario el diseño de sistemas dedicados para aplicaciones específicas.

Es por eso que la especialización técnica de los profesionistas les permite desarrollar soluciones que se adapten a condiciones preexistentes tomando en cuenta las características y limitantes que definen el alcance y funcionamiento de nuevos sistemas a diseñar, además de evaluar su usabilidad y factibilidad comercial.

Este desarrollo de soluciones tecnológicas refleja el resultado de la especialización académica del profesionista que busca aplicar y experimentar con nuevas habilidades adquiridas en busca de desarrollos innovadores con aplicaciones reales.

1.1 ANTECEDENTES

Las instalaciones de una empresa cuentan con un sistema de control de acceso y administración de usuarios que genera complicaciones y tiempos muertos por fallas como la actualización de una base de datos completa por la simple operación de un nuevo registro de usuario. Se había contratado a GUAO Studio [77], empresa desarrolladora de software, para tomar acciones a petición de la gerencia y desarrollar un nuevo sistema de administración confiable, rápido y robusto. Este sistema fue desarrollado, pero se encontraron con una limitación al buscar acoplar este nuevo diseño al sistema físico de control de accesos que está basado en torniquetes giratorios.

Este sistema de control de acceso físico en los torniquetes está basado en el acoplamiento de aparatos comerciales de reconocimiento facial que además de no tener conectividad alguna al sistema de administración también presenta fallas en la detección de rostros y mala experiencia en los usuarios.

Fue entonces que se invitó al autor del presente documento para desarrollar un nuevo método de control de acceso en los torniquetes que se adaptara a las condiciones previamente mencionadas, con el requerimiento inicial de parte de la administración de que se usaran tarjetas para el acceso y se registraran las entradas de los usuarios en el sistema.

Conforme el desarrollo del proyecto ha avanzado se ha despertado el interés en la comercialización de este producto bajo el licenciamiento abierto tanto para software como para hardware, buscando seguir metodologías formales de desarrollo de productos e inteligencia de negocio.

1.2 DEFINICIÓN DEL PROBLEMA

El diseño de este nuevo sistema de control de acceso en los torniquetes contempla más condiciones que sólo el uso de tarjetas, inherentes al estado actual de la infraestructura total de las instalaciones de la empresa.

Una de las primeras limitantes es la necesidad de adaptación a los torniquetes instalados contemplando dimensiones y cuestiones técnicas como mecanismos internos de funcionamiento mecánicos y sistemas eléctricos, así como el espacio disponible para el montaje de un nuevo sistema de control, que se ha definido por el desarrollo de un dispositivo de hardware.

Una segunda limitante es lo referente a la conectividad al nuevo sistema de administración desarrollado por la empresa de software, donde es necesario acordar el protocolo de comunicación para la correcta coexistencia de los subsistemas.

En la búsqueda de la comercialización de productos tecnológicos como el que se desarrolla para resolver la problemática identificada se deben analizar cuestiones comerciales, como el uso y desarrollo de propiedad intelectual abierta y privada.

1.3 JUSTIFICACIÓN

Al desarrollar este subsistema no se busca únicamente solucionar una problemática existente con base en el diseño, prueba y validación de un sistema embebido. Se busca también elevar el grado de especialización del autor en el desarrollo de soluciones electrónicas en aplicaciones reales que implican limitantes preexistentes y que tienen un potencial de comercialización.

Hablando de cuestiones técnicas se pretende profundizar en el proceso de desarrollo de una solución tecnológica aplicada incluyendo diseño de circuitos y tarjetas electrónicas, modelado y validación tanto del sistema de hardware como de software, además de la correcta integración con el resto del sistema. Es decir, pasar por todo un proceso de desarrollo de producto.

Se busca agregar un grado de innovación, por lo que además del proceso ya antes mencionado se buscará la programación remota del firmware en el embebido y la agregación de una capa de respaldo de la base de datos de usuarios para brindar robustez en caso de fallas de conectividad a internet.

1.4 OBJETIVOS

Diseñar y construir un producto tecnológico que se adapte a instalaciones preexistentes de control de acceso, basado en lectura de tarjetas NFC y conexión inalámbrica al sistema de administración principal bajo las siguientes metas:

1. El producto deberá ser funcional y confiable, lo que se validará con pruebas sistemáticas.
2. Se contará con una capa de respaldo de registro de usuarios para evitar problemas por falta de conexión a internet.
3. Se podrá actualizar el firmware del sistema de manera remota.
4. Se evaluarán las mejores prácticas para buscar la comercialización del producto bajo licenciamiento abierto.

1.5 HIPÓTESIS

La investigación científica aplicada formalmente para solucionar la problemática identificada lleva a seguir un proceso de desarrollo de un producto de base tecnológica eficiente, y esta rigurosidad abre la posibilidad de desarrollar nuevos productos comerciales.

CAPÍTULO 2: MARCO TEÓRICO

2.1 INTRODUCCIÓN Y DEFINICIONES

El proyecto que este documento describe se basa en la ideología del *Open Source Software* para el desarrollo de los componentes necesarios para un producto comercializable, tanto de software como de hardware, por lo que es importante comenzar esta sección con la definición del término.

El *Open Source Software* "es aquel que puede ser accesado libremente, usado, cambiado y compartido (en su forma original o modificada) por cualquier persona. Es creado por muchas personas (la comunidad) y distribuido bajo licencias que cumplen con la *Open Source Definition" [1]*.

Este concepto de *Open Source Software* es más utilizado en las actividades cotidianas hoy en día de lo que en realidad conocemos. Uno de los ejemplos más claros está en que el 64% de los servidores de internet corren sobre Apache, y el 70% del correo electrónico es enviado a través de Sendmail [2]. Ambos sistemas, Apache y Sendmail, son totalmente *Open Source*.

Los párrafos anteriores hacen mención a la definición y penetración del *Open Source* en la industria hoy en día, sin embargo, es importante conectar cómo estos conceptos aplican en el campo de lo tangible; el hardware. El término *Open Hardware* se refiere a diseños de hardware que pueden ser reproducidos por cualquiera [3], más detalladamente, a las especificaciones de diseño de un objeto físico que está licenciado de tal manera que dicho objeto puede ser estudiado, modificado, recreado y distribuido [4].

Todo el open hardware debe ir acompañado de documentación, incluyendo los archivos de diseño y código fuente. La licencia que lo acompañe debe permitir la modificación y distribución de los archivos de diseño [4]. Esto hace posible e incentiva el estudio, resolución de problemas, modificación y manejo del open hardware.

Ambas iniciativas han dado pie en años recientes a un nuevo movimiento de hardware, la comunidad *maker*, principalmente por la creciente colaboración para desarrollar productos basados en hardware y por nuevas herramientas de desarrollo y prototipado rápido; impresoras 3D, máquinas CNC, *MCUs*, y la adopción de lenguajes de programación de alto nivel para la programación de sistemas embebidos [5].

A su vez, este movimiento de hardware en conjunto con tecnologías cuya adopción ha crecido exponencialmente (Inteligencia Artificial, Machine Learning y Análisis de Datos) han acelerado un nuevo movimiento; el de IoT (Internet of Things). El IoT está impulsado por tres ejes principales; la conectividad ubicua, el procesamiento de datos, y el hardware barato [5].

Dentro del hardware accesible, uno de los principales jugadores es el Arduino [6] cuyas prestaciones son ideales para el prototipado rápido de aplicaciones del IoT. Otra plataforma que ha resaltado en los últimos años es la NodeMCU [7] la cual está basada en el SoC ESP8266. La ventaja principal de la NodeMCU sobre el popular Arduino es su capacidad de conectividad WiFi integrada [8].

El IoT y el nuevo movimiento de hardware son ambas ideas transformadoras, por lo que es importante entenderlas ahora así como lo era entender el Internet en 1995. Es por ello que se ha impulsado una serie de investigaciones sobre cómo el open hardware y el análisis de los datos generados (IoT) puede insertar valor agregado en diferentes aplicaciones.

2.2 INVESTIGACIONES PREVIAS

Partiendo de las investigaciones básicas, se ha usado la NodeMCU para fines demostrativos en maquetas sobre la factibilidad del desarrollo de soluciones de domótica [9] en donde además del control digital se aprovecha la capacidad de la NodeMCU para habilitar un servidor HTTP y proveer páginas web cuyo código e interacciones están vinculadas a acciones lógicas en los puertos GPIO para controlar actuadores. Al estar basada la NodeMCU en el SoC ESP8266, éste último requiere ejecutar un firmware que habilite sus funcionalidades. Elías [10] agrega un módulo de conectividad WPS al código fuente del firmware antes de compilar para después cargarlo al ESP8266 y tener habilitada esta nueva funcionalidad en la tarjeta NodeMCU, demostrando que la accesibilidad al código fuente por ser una plataforma open hardware permite hacer mejoras de funcionalidad y expandir el rango de posibles aplicaciones de estos dispositivos.

Jafee [11] explora la variedad de aplicaciones de la NodeMCU en un espectro más amplio, sacando provecho del acceso a tecnologías Open Source para generar un valor adicional a través de la adquisición e interpretación de datos, por lo que en sus recomendaciones menciona la importancia de los protocolos de seguridad en las comunicaciones.

Dentro de las aplicaciones industriales y la experimentación de nuevas soluciones, varios autores han trabajado con estas plataformas Open Hardware para validar la utilidad de aplicaciones a la medida, tal es el caso de Shkurti et al. [12] quienes implementan una red de sensores para obtener distintas mediciones como la calidad ambiental, argumentando la influencia de ésta en la calidad de vida de los habitantes de una ciudad. En entornos de manufactura se han desarrollado sistemas de monitoreo de humedad y temperatura con el objetivo de extender la vida útil de la maquinaria [13]. De igual manera implementan un sistema de monitoreo de temperatura en un cuarto de servidores (*Data Center*) donde la temperatura y la humedad, al igual que en las aplicaciones mencionadas previamente, son variables críticas en los sistemas [8].

Calderón et al. [13] exploran la industria alimenticia mediante el desarrollo e implementación de un sistema de monitoreo de temperatura para contenedores de leche en un proceso de pasteurización. Los mismos autores hacen una comparación con implementaciones previas de sistemas SCADA basadas en LabVIEW [14] y dispositivos de adquisición de datos (DAQ), destacando la amplia diferencia de costos.

En la industria de la salud y servicios derivados, donde la información es aún más crítica, Skraba et al. [15] desarrollan un prototipo de monitoreo de pulso cardíaco basado en sensores de costo accesible. Dicho prototipo toma ventaja de la capacidad de identificar cada NodeMCU por medio de su dirección MAC para identificar a los pacientes que están bajo observación. Este prototipo deja en claro que la complejidad y sensibilidad de las aplicaciones desarrolladas con Open Hardware puede variar ampliamente, pues los autores refieren que la parte más complicada de su experimentación fue el desarrollo del software para analizar e interpretar los datos de los pacientes.

Como característica común entre los trabajos mencionados hasta este punto se encuentra el requerimiento de grabar el código desarrollado en el hardware. Comúnmente este proceso se lleva a cabo de manera manual y con una conexión física de un cable de conexión serial (USB) de la computadora a la tarjeta NodeMCU. Este método deja de ser eficiente en el momento en que se requiere hacer un cambio a la programación y se tiene la complicación de que los sensores y dispositivos están desplegados e instalados, por ejemplo, en la parte superior de una nave industrial o en una luminaria de alumbrado público. Para tal escenario se puede contar con el método de programación OTA (Over the Air).

Gore [16] hace un trabajo de investigación para comprobar la confiabilidad de este método de programación argumentando que el método clásico de carga de programa (cableado) demanda mucho tiempo y es ineficiente en ambientes de producción y finalmente logra comprobar que a través de la programación OTA es posible actualizar

el software de una red de NodeMCUs sin interacción física con los dispositivos y sin interrumpir su conexión a la red. Bajo la misma directriz, Kodali y Yerroju [17] analizan cómo el IoT puede ayudar a ahorrar hasta 42 billones de dólares al año en consumo eléctrico en India por medio de sistemas "inteligentes" de control de alumbrado público. Para esto desarrollan un dispositivo basado en NodeMCU para controlar los horarios de alumbrado público y la intensidad con la que las luminarias operan, tomando ventaja de la capacidad del ESP8266 (SoC en la NodeMCU) de funcionar como red de nodos, y no sólo como clientes.

Perdomo [78] identifica una problemática en la industria del transporte público y decide desarrollar una solución tecnológica para digitalizar el proceso de cobro a los pasajeros. Su trabajo está basado también en NodeMCU e integra otros componentes para producir un diseño específico; integra una HMI y lector de NFC para poder usar tarjetas como medio de pago. En su trabajo detalla cómo las prestaciones de la tecnología NFC, como lectura de corto alcance y bajos costos, son idóneos para el caso de aplicación en el que trabaja.

Todos los trabajos de investigación mencionados previamente tienen la capacidad de convertirse en productos comerciales, y al estar basados en investigación científica formal su probabilidad de ser exitosos aumenta, pues tienen comprobada su factibilidad técnica. Más adelante en esta sección se revisarán otros elementos necesarios para poder llevar trabajos de investigación al mercado, lo cual es uno de los objetivos específicos para el proyecto de este documento.

Un patrón común entre las investigaciones referidas es el uso de Arduino IDE (*Integrated Development Environment*) como entorno de desarrollo para el software que es grabado en las NodeMCU. De acuerdo con la documentación oficial disponible en la página principal de Arduino [18], este IDE es una herramienta de desarrollo que contiene un editor de texto para escribir código y un conjunto de herramientas de gran utilidad en el proceso de desarrollo de un programa. Se puede usar no sólo para las plataformas

de Arduino, sino con una amplia variedad de otras plataformas Open Hardware como la NodeMCU con tan sólo usar la librería indicada. A partir de este punto, la creatividad del desarrollador y el *expertise* en codificación establecen el límite de lo que se puede crear.

2.3 PRODUCTOS EXISTENTES

En el mercado internacional es posible encontrar diversos productos basados en tecnologías Open Source, tanto en hardware como en software. Tal es el caso de Bolt [19], una plataforma que pareciera el "Lego del IoT" al ser un conjunto de bloques que se pueden interconectar para trabajar en distintas aplicaciones como automatización de casas e incluso puede ser conectado a algoritmos de *Machine Learning*, todo a través de conectividad WiFi y con un precio base de $17.00 USD.

Por la naturaleza Open Hardware que tiene la NodeMCU (en conjunto con su SoC ESP8266) podemos encontrar diferentes versiones de esta tarjeta fabricada y comercializada por otros agentes. Knewron [20] comercializa un dispositivo muy similar a la NodeMCU que brinda también conectividad WiFi y control de puertos GPIO sólo que en un *form-factor* diferente. Zmote es otro ejemplo del desarrollo de productos comerciales basados en Open Hardware [21], éste es un pequeño dispositivo que permite controlar a través de una aplicación web electrodomésticos cuya interacción está basada en comunicación infrarroja.

Pero las oportunidades de soluciones comerciales que abre el Open Hardware no se dan únicamente en el ámbito del IoT y automatización casera e industrial. OpenXC [22] ha desarrollado una serie dispositivos para medir variables de rendimiento en automóviles a través del puerto OBD, y se puede adquirir a partir de $130.00 USD. Estos dispositivos abren el acceso a la información que un vehículo genera al funcionar, y a través de

conectividad inalámbrica Bluetooth y una API los desarrolladores pueden crear aplicaciones específicas para el caso de uso que necesiten.

2.4 PERSPECTIVA DE NEGOCIO

Dentro de los 4 modelos de negocio que la *Open Source Initiative* reconoce en el desarrollo de productos Open Hardware podemos encontrar el *Widget Frosting* [23] que se refiere al caso en donde una compañía de hardware (para la que el desarrollo de software es más un gasto que una ganancia) abre su plataforma para obtener mejores controladores e interfaces de una manera más barata. Tal es el caso de Raspberry Pi, cuyos diagramas esquemáticos están liberados pero la plataforma no es Open Source, pues depende de los ingresos por la venta de las tarjetas para financiar las actividades de su fundación [24], Este mismo modelo aplica para los productos mencionados en párrafos previos puesto que sus ingresos están en la venta del hardware que cubre una necesidad específica y abierta para el desarrollo del software necesario para crear soluciones.

Un segundo método en que los productos de Open Hardware generan ingresos es por medio de la consultoría y prestación de servicios necesarios para su puesta en marcha, configuración e integración de la solución [23]. Estos dos modelos, *Widget Frosting* y consultoría, son de especial interés para el proyecto ya que son los aplicables a su posible comercialización.

2.5 DESARROLLO DE PRODUCTOS

Cuando se está desarrollando un proyecto es común que si el grupo o individuo a cargo del desarrollo tiene un perfil ingenieril no se tomen en cuenta detalles importantes desde la perspectiva de negocio y se entre en un ciclo de perfeccionamiento técnico agregando nuevas funcionalidades que no han sido validadas en cuanto al valor agregado que aporta al propósito final del proyecto [3]. Este fenómeno es conocido en las disciplinas de administración de proyectos como *scope creep*.

Una de las principales diferencias entre el desarrollo de proyectos de hardware contra los de software, sean Open Source o no, es la inversión monetaria necesaria en materiales físicos y procesos, como la adquisición de materiales y la manufactura de circuitos impresos.

Por ello, es importante seguir un proceso de documentación y desarrollo que ayude a mantener el desarrollo del proyecto en curso y con un objetivo claro. El PRD (*Product Requirements Document*) es el documento de referencia cuando se requiere identificar aspectos específicos del proyecto en desarrollo, tales como el propósito, requerimientos funcionales, requerimientos de usabilidad, limitantes, suposiciones y métricas de desempeño [25]. Incluso este documento sirve de referencia cuando es necesario priorizar funcionalidades de acuerdo a su importancia (y así se evita caer en *scope creep)*.

Cuando se busca comercializar un proyecto de hardware, autores como Chamdiramani [25] recomiendan tener un *Business Model Canvas (BMC)* ya que brinda un panorama general sobre las directrices y estrategias de negocio como quiénes son los clientes y cómo se generan ganancias. A diferencia de un Business Plan (Plan de negocios), el BMC expone la información clave en cada sección de un plan de negocios, sin tener que entrar en detalle y hacer modificaciones conforme el desarrollo y comercialización del producto se va iterando y validando.

Existe un proceso de desarrollo de productos de hardware propuesto por Einstein [26] que contempla cuatro fases: ideación, diseño, ingeniería de desarrollo e ingeniería de producto. La figura muestra este proceso de manera general.

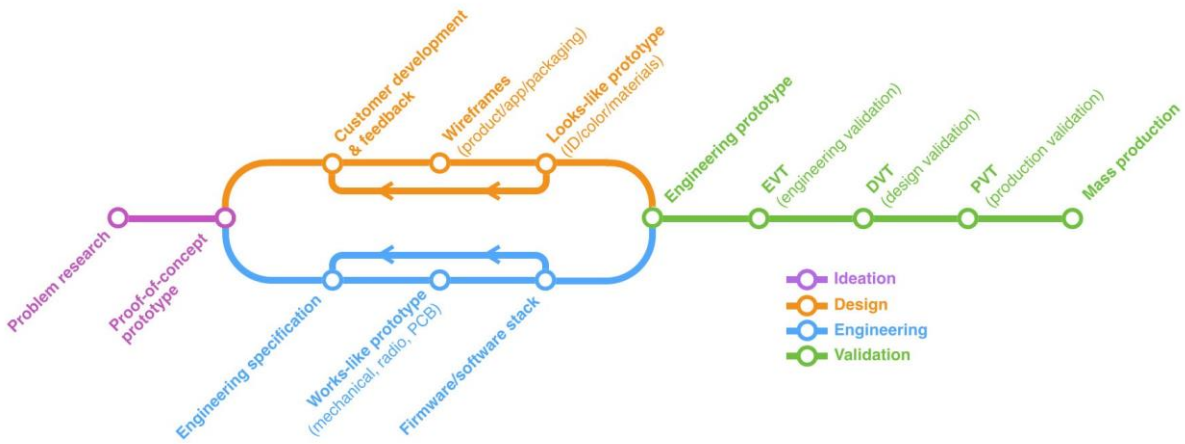

Figura 1: Proceso de desarrollo de hardware [26]

La primera etapa, la ideación, comienza por establecer una clara definición del alcance del proyecto, para esto durante la investigación del problema se recomienda primero entender el entorno de aplicación del proyecto teniendo entrevistas con el cliente para posteriormente construir un prototipo de prueba de concepto (PoC).

En la segunda fase, el diseño, el autor recomienda enfocarse en retroalimentación del cliente. Esto marca la diferencia entre la probabilidad de lograr tener un producto exitoso contra sólo hacer un desarrollo ingenieril. Para lograr esto se debe tener especial atención a la experiencia de usuario, la cual debe ser resaltada y considerada. Seguido a tener una conceptualización clara y un PoC validado es tiempo de proceder a la esencia del desarrollo de un proyecto; la ingeniería de desarrollo. En esta etapa, Einstein [26] refiere que uno de los mejores indicadores de un producto bien hecho es la rigurosidad de las especificaciones del PRD (*Product Requirements Document*). Además, menciona que si el proyecto tiene una placa de circuito impreso (PCB), por lo general toma de 5 a 10 revisiones del mismo antes de estar listo para su manufactura en volumen.

La parte final dentro del mismo proceso es la ingeniería de prototipo, es el primer punto de contacto en el ciclo de desarrollo en que el diseño y la ingeniería se encuentran. Aquí se unen los componentes técnicos con los de diseño y presentación del producto. Se

genera un diseño para manufactura (DFM, *desing for manufacturing)*, el cuál es el diseño de las partes plásticas y metálicas para su optimización en manufactura. Aquí se llevan a cabo las pruebas de validación y se comprueba que el producto cubre todos los requerimientos funcionales especificados.

2.6 TECHNOLOGY READINESS LEVEL (TRL)

Es importante dedicar un apartado de esta sección para hablar del modelo de los niveles de maduración tecnológica usado por la NASA [27]. Dicho modelo es "un sistema de medición usado para evaluar el nivel de maduración de una tecnología en particular". Se compone de 9 niveles, en donde cada uno establece un conjunto de parámetros a evaluar. A continuación, se describen cada uno de estos niveles de acuerdo con CONACYT [28]:

1. Investigación aplicada - tecnológica básica

   Se define como la observación y reporte de principios básicos. Aquí se identifican las maneras de resolver una problemática con las soluciones tecnológicas existentes. Se comienza con la transición a investigación aplicada identificando qué tecnologías o mezcla de ellas pueden conformar una solución, a nivel conceptual.

   Desde una perspectiva de negocios, se han evaluado cualitativamente los conceptos básicos de la solución con base en la experiencia, para entonces comenzar a explorar las nuevas tecnologías para el autor y evaluar su conveniencia para el desarrollo del proyecto.

2. Validación conceptual

   Se define como la conceptualización de la tecnología o formulación de la aplicación. Con base en la investigación previa se comienza una etapa especulativa a nivel conceptual de cómo las tecnologías disponibles conformarán los distintos componentes del proyecto, identificando también la

forma correcta de su integración basada en investigaciones y trabajos publicados.

Esta es una etapa meramente especulativa, sin pruebas de concepto funcionales a nivel laboratorio.

3. Pruebas de concepto analíticas y experimentales de funciones críticas

   Se comienzan a construir prototipos para validar el funcionamiento de las tecnologías que se encontraron previamente en la investigación básica. Estas pruebas de concepto van validando una a una las tecnologías a usar y descartando aquellas que no son funcionales o compatibles.

   El objetivo principal de esta etapa es verificar que el concepto funciona como lo esperado

4. Desarrollo tecnológico

   Este nivel es definido por la validación de componentes o sistema en ambiente de laboratorio. Se comienza a trabajar en la integración de los componentes del sistema, aún no constituido como un ente independiente y robusto, sino que cuenta con conexiones a equipo externo e instrumentación de medición y monitoreo. El objetivo principal es lograr la interacción entre los componentes.

   Desde una perspectiva de negocios, se comienzan a identificar nuevas oportunidades de aplicación del proyecto como resultado de las observaciones en las pruebas de concepto e integraciones previas. A su vez, se comienzan a identificar los requisitos de desarrollo y fabricación de un concepto de producto final, el cual comienza a ser definido por las restricciones y requisitos de interacción entre los componentes, además del escenario de aplicación.

5. Validación de componentes en un ambiente relevante

   Este nivel se identifica al tener los componentes integrados de manera que la configuración del sistema sea similar a su aplicación final, pero su operatividad es aún a nivel laboratorio. Representa el puente de conexión entre la investigación y la ingeniería, puesto que se comienzan a realizar calibraciones y modificaciones para aumentar la fidelidad de los componentes en sus funciones básicas y en su interacción como sistema. Estas modificaciones son el resultado del aprendizaje obtenido en las pruebas de concepto individuales hechas en etapas previas.

   Hasta este punto, la solución ha sido validada a través de pruebas en el entorno previsto; simulado o real. El nuevo hardware, a nivel prototipo, está listo para comenzarse a usar.

6. Demostración tecnológica

   Este nivel está definido por la validación del sistema de ingeniería en condiciones relevantes a las reales operativas, aún a nivel prototipo. Aquí se cuenta ya con un prototipo capaz de realizar todas las funciones requeridas por el sistema final. Aquí es cuando se tiene una demostración de mercado bajo el concepto *Early Adopters* [29].

7. Comisionamiento de sistemas

   Este nivel está definido por un prototipo completo demostrado en un ambiente relevante, es decir, se cuenta con un prototipo cuyo sistema operativo es completamente funcional.

   Desde una perspectiva de negocios se ha demostrado que la tecnología funciona y opera a escala pre-comercial. Se han identificado cuestiones de fabricación y operación finales y se han resuelto cuestiones tecnológicas menores. Se hace una primer corrida piloto y pruebas funcionales reales.

8. Sistema completo y funcional para su aplicación final

   A este nivel, la tecnología ha sido probada en su forma final y bajo condiciones supuestas. En muchos casos significa el final del desarrollo del sistema.

   Todas las cuestiones operativas y de fabricación han sido resueltas. Se han elaborado documentos para su utilización y mantenimiento del producto, y se ha demostrado que la tecnología funciona a nivel comercial a través de una aplicación a gran escala.

9. Operación del sistema

   La tecnología se encuentra en su forma final y operable en un sin número de condiciones operativas, ya se habla de un producto completamente desarrollado y disponible para la sociedad. Se puede entregar el producto para una producción en serie y comercialización.

La Figura 15: Niveles del modelo TRL . Fuente: muestra gráficamente estos distintos niveles del modelo TRL. En el capítulo 3 (Procedimiento de investigación) se estará usando la secuencia de este modelo para redactar el avance y construcción del proyecto.

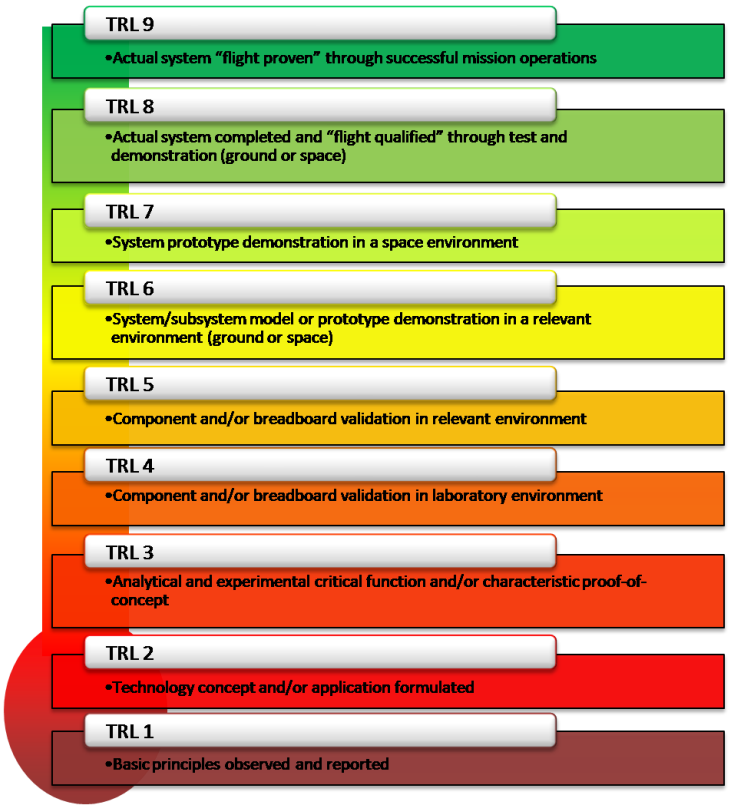

Figura 2: Niveles del modelo TRL [27]

2.7 RELACIÓN CON EL PROYECTO

El conjunto de referencias presentadas en esta sección tales como las investigaciones previas, los productos comerciales existentes, las metodologías de desarrollo de proyectos y las herramientas de desarrollo técnico son tomadas como base para el desarrollo del proyecto que este documento presenta.

Como se detallará en secciones subsecuentes, la naturaleza del caso de uso específico del proyecto requiere que sea habilitada la posibilidad de hacer actualizaciones remotas, es por esto que se profundiza en la implementación de la programación OTA, para poder hacer modificaciones necesarias según surja la necesidad.

Se ha confirmado que el entorno de desarrollo Arduino IDE es la herramienta indicada para elaborar la codificación del software necesario para que el hardware cumpla con su propósito específico, pues brinda la apertura de usar recursos abiertos para una correcta integración entre la programación y el ESP-8266, además de tener la libertad de profundizar en cuestiones técnicas tanto como sea necesario.

Es importante resaltar el hecho de que el proyecto desarrollado no entra directamente en la definición del IoT, pues no genera datos que son analizados con fines de inteligencia y mejora posterior. Es un proyecto específicamente de desarrollo de hardware que cubre una necesidad específica en una oportunidad de mercado.

Por lo anterior, se toman como referencia las modelos de negocio y procesos de desarrollo de hardware, en las subsecciones que son de utilidad, con el fin de agregar valor de desarrollo profesional al estudiar la integración de distintas disciplinas y el aprendizaje de nuevas áreas, como las propias del desarrollo de estrategias de negocio.

CAPÍTULO 3: PROCEDIMIENTO DE INVESTIGACIÓN

3.1 INTRODUCCIÓN

Esta sección redacta el principal trabajo de investigación aplicada realizado durante el desarrollo de este proyecto. Dicho proceso se describe como una historia que inicia con una idea, hasta el cálculo de costos una vez instalado el sistema para presentar una cotización y lograr la venta del proyecto. Todo esto se redacta progresivamente conforme al modelo de Nivel de Maduración Tecnológica (TRL, Technology Readiness Level) de la NASA y adoptado por el CONACYT como referencia para sus convocatorias del Fondo de Innovación Tecnológica.

Previo a esta redacción progresiva se plantean las descripciones técnicas de los componentes que integran al proyecto, lo cual es necesario para poder llevar el hilo en el desarrollo de la experimentación y ensamble.

Bajo el mismo principio, se presentan los fundamentos necesarios de las herramientas utilizadas para el proyecto, tales como entornos de desarrollo y herramientas de diseño electrónico.

Finalmente se presenta una sección dedicada al proceso de instalación y otra al cálculo de costos y definición del modelo de negocio. Ambas etapas son dignas de una descripción detallada para poder sustentar las conclusiones de este trabajo.

Antes de continuar, es importante describir la arquitectura del proyecto, la cual se ilustra con la figura 3 a continuación:

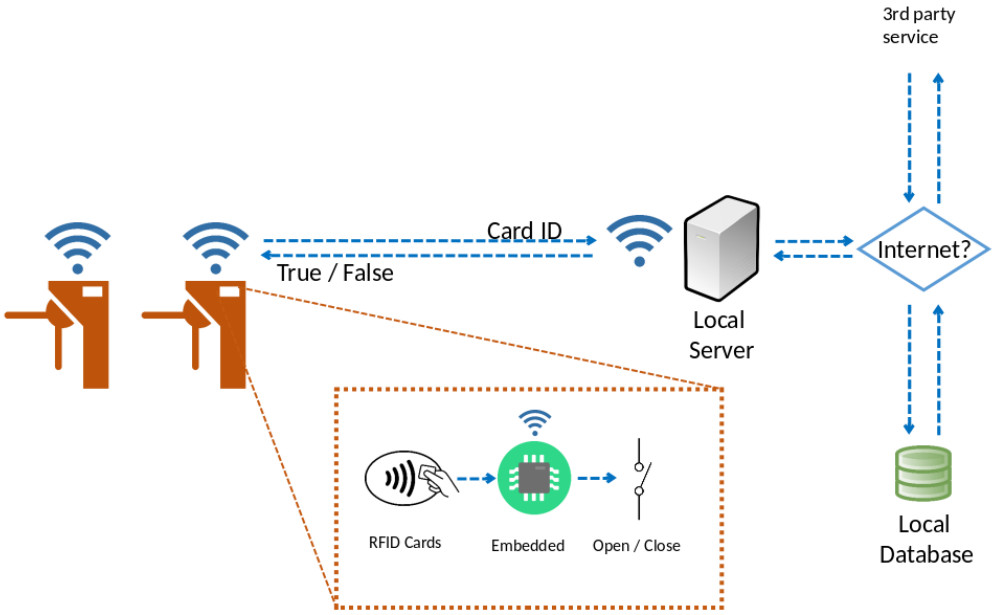

Figura 3: Arquitectura general del sistema. Fuente: Elaboración propia

Los componentes de principal interés en esta descripción introductoria son el dispositivo de hardware embebido que integra distintos componentes, al cual llamaremos en delante – dispositivo –. Este está representado por el recuadro punteado en color naranja, acercado en la siguiente figura:

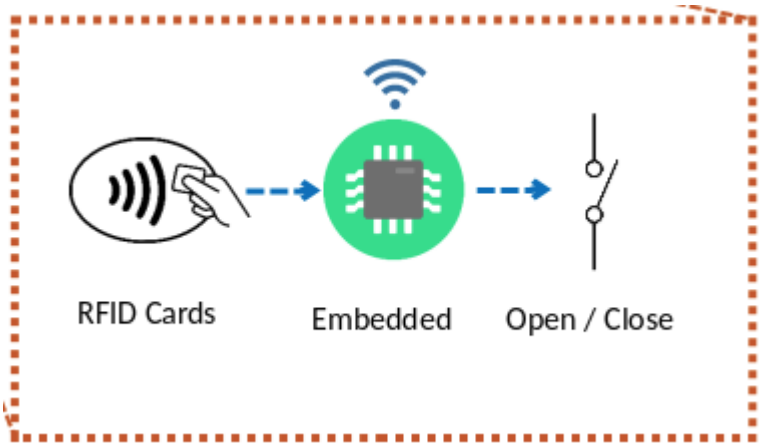

Figura 4: Diagrama conceptual del dispositivo. Fuente: Elaboración propia

El segundo componente de principal interés es el servidor local, al cual identificaremos a partir de este punto como – micro servidor –, el cual podemos identificar en la figura 3

como un componente intermedio entre el sistema local y el exterior. La figura 5 hace un acercamiento al mismo.

Figura 5: Representación del micro servidor. Fuente: Elaboración propia

A lo largo del desarrollo de este capítulo se estarán describiendo los detalles y estructura de estos dos componentes principales, que son pilares del sistema.

3.2 DESCRIPCIONES TÉCNICAS

Esta sección plantea las bases necesarias en cuestiones técnicas de cada uno de los componentes de los dos pilares del proyecto (micro servidor y dispositivo). Es una sección de transición hacia la descripción de desarrollo y evolución del proyecto de investigación. Se divide en dos partes; las descripciones técnicas propias del dispositivo y aquellas inherentes al micro servidor.

3.2.1 De los componentes del dispositivo

El dispositivo final está montado en un circuito impreso (PCB) que sirve como plataforma de integración de los componentes electrónicos. Estos componentes por sí solos no realizan ninguna función de utilidad si no se controlan a través de código embebido en

el microcontrolador. Para fines prácticos llamaremos –firmware-- a este código de control embebido.

Revisemos entonces cada uno de los componentes del dispositivo:

3.2.1.1 Node MCU

Su página de referencia principal la define como una tarjeta de desarrollo *Open-Source*, interactiva, programable, de bajo costo, simple, inteligente y con Wi-Fi integrado [7]. Esta tarjeta de desarrollo, basada en el SoC ESP8266, integra funcionalidades como GPIO, PWM, IIC, 1-Wire y ADC en la misma tarjeta. Su objetivo principal es acelerar el desarrollo de proyectos, y es justo el caso de uso para este trabajo de investigación. La siguiente figura muestra la tarjeta de desarrollo:

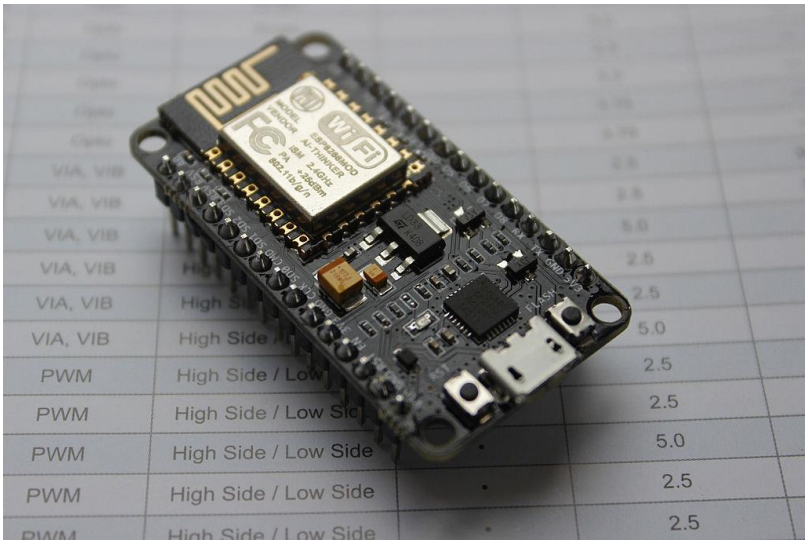

Figura 6: NodeMCU v3 [68]

La figura 6 muestra la tarjeta de desarrollo NodeMCU en su versión 3. El recuadro metálico en la parte superior trasera es el SoC ESP8266, el cual cuenta con un microcontrolador Tensilica de 32 bits. La placa base es el módulo ESP-12E. Todo el dispositivo integra 4MB de memoria flash, un reloj de 80 MHz y alrededor de 50Kb de RAM disponible [30]. Todo esto con un transmisor-receptor de Wi-Fi. La siguiente figura muestra la configuración y funcionalidades de sus pines.

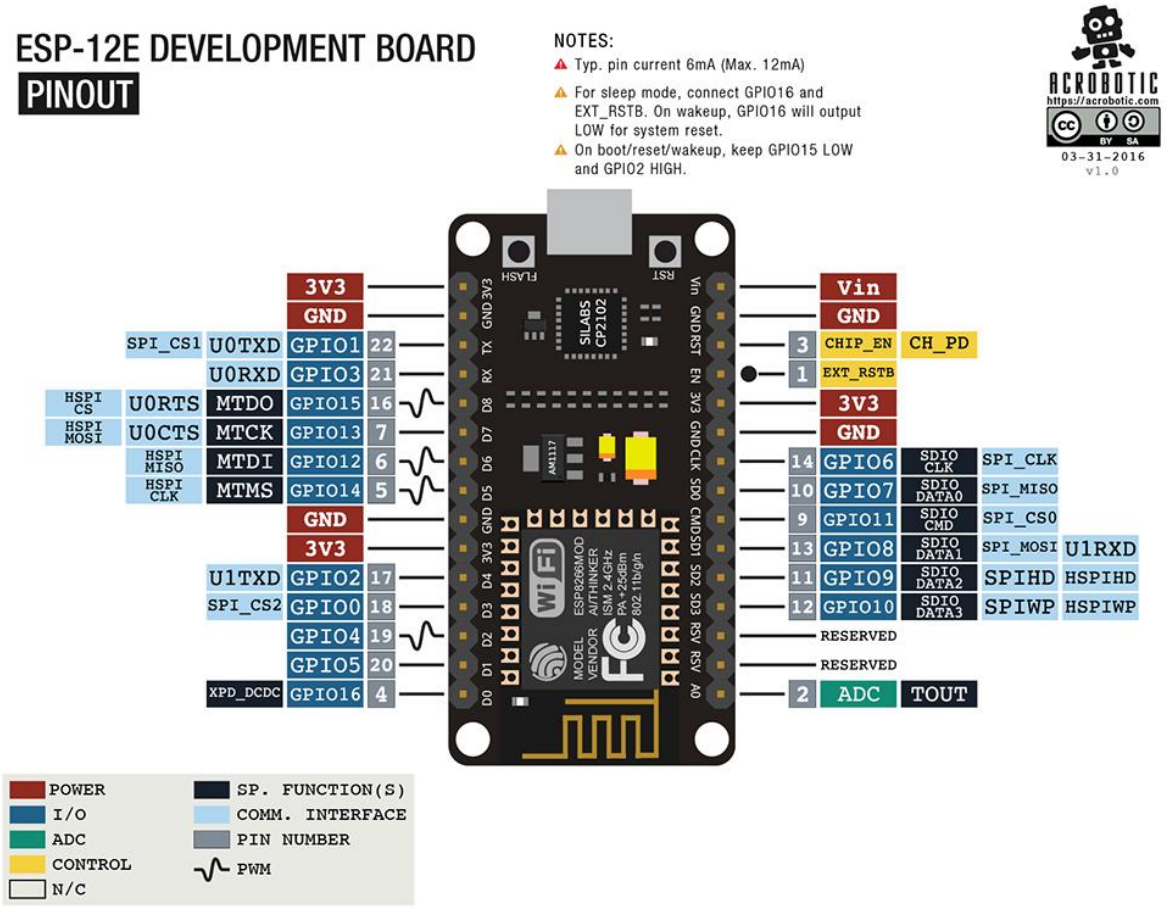

Figura 7: Configuración de pines del NodeMCU [76]

El SoC ESP8266 cuenta con dos certificaciones importantes; la FCC (Federal Communications Comission) de los Estados Unidos de Norteamérica, y la CE (Conformité Européenne) del Área Económica Europea [32]. La certificación CE es un símbolo de libre comercialización en Europa, mientras que la certificación FCC es un requisito para ciertos dispositivos eléctricos para poder ser comercializados en los Estados Unidos de Norteamérica. Debido a esto, es común encontrar este SoC dentro de productos comerciales, especialmente del campo de la automatización, domótica e internet de las cosas.

De acuerdo con la comunidad [33] si un módulo de comunicación por radiofrecuencia como el ESP8266 previamente certificado es integrado en el desarrollo de un nuevo

proyecto, como es el caso de este trabajo, ya no es necesario hacer más que proveer el mismo identificador de certificación del módulo en el proyecto gracias a un cambio en las reglas institucionales [34]. Lo anterior es viable siempre y cuando no se modifique el medio físico de comunicación inalámbrica, como la antena, en el nuevo dispositivo. De lo contrario es necesario pasar por todo un proceso de certificación con agentes autorizados, lo cual puede representar una inversión del orden de los miles de dólares [33].

México tiene un MRA (*Mutual Recognition Agreement*) [35], o acuerdo de reconocimiento mutuo, en el que se establece que tanto las certificaciones emitidas por la FCC como por el IFT (Instituto Federal de Telecomunicaciones) serán reconocidas de manera bilateral con fines comerciales dentro del marco del TLCAN (Tratado de Libre Comercio de América del Norte) o NAFTA (*North America Free Trade Agreement*) [36].

Los párrafos anteriores son relevantes para este trabajo ya que no sólo es un trabajo de investigación para aplicación tecnológica, sino que también lleva un esfuerzo de exploración comercial. Por lo tanto, el conocer el entorno de las regulaciones comerciales es clave para una correcta planeación estratégica de comercialización. Hasta este punto lo más relevante es demostrar que el elemento que se está usando como núcleo del dispositivo, el SoC ESP8266, cuenta con respaldo regulatorio internacional.

Volviendo a temas más técnicos, hay una funcionalidad crítica del NodeMCU que no ha sido mencionada; la programación OTA (Over *the Air)*. Esta funcionalidad se refiere a que con las debidas configuraciones necesarias es posible hacer actualizaciones de firmware en el embebido de manera remota, a través de una conexión de red, sin necesidad de conexiones físicas por cables entre una computadora y el hardware. Gore et al. [16] revisan esta metodología de programación concluyendo que gracias a esto se pueden alcanzar metas de flexibilidad y escalabilidad, cuestiones que son necesarias para este trabajo ya que la instalación final del sistema se encuentra no sólo en un lugar

difícil de alcanzar, requiere herramientas especiales y detener las operaciones del negocio para poder conectar un cable al dispositivo y hacer actualizaciones de firmware, sino que también se encuentra a varios cientos de kilómetros del lugar de residencia del autor. Es importante mencionar también que la instalación final del sistema consta de 5 dispositivos instalados, y gracias a la programación OTA cualquier actualización de firmware es hecha con tan solo unos cuantos comandos desde cualquier parte del mundo, con ayuda de una VPN (Virtual *Private Network)* de la cual se hablará más adelante.

3.2.1.2 Lector NFC

El proceso de la empresa definido para el acceso de los clientes indica que se debe hacer uso de una credencial que los identifique como usuarios activos, para lo que se opta por usar credenciales con tecnología NFC, las cuales pueden ser impresas con la información de los usuarios y además facilitan la identificación discreta de los mismos a través del código identificador único de cada tarjeta.

Como lector de las tarjetas NFC se utiliza el dispositivo RFID-RC522, el cual utiliza el circuito integrado MFRC522 fabricado por NXP, y de acuerdo con su hoja técnica [37] es un lector/escritor de alta integración para la comunicación sin contacto a 13.56 MHz, compatible con los estándares ISO/IEC14443A/MIFARE y NTAG.

El transmisor interno del MFRC522 puede manejar una antena de lector / escritor diseñada para comunicarse con tarjetas ISO / IEC 14443 A / MIFARE y transpondedores sin circuito adicional activo. El módulo receptor proporciona una implementación robusta y eficiente para señales de demodulación y decodificación, gestiona el encuadre completo ISO / IEC 14443 A y cuenta con funcionalidad de detección de errores (paridad y CRC). Admite comunicación sin contacto y utiliza velocidades de transferencia MIFARE de hasta 848 kBd en ambas direcciones.

La siguiente figura muestra el aspecto físico del dispositivo RFID-RC522 en su versión comercial.

Figura 8: Aspecto físico del RFID-RC522 [75]

Este dispositivo se comunica por medio de ocho vías a la tarjeta NodeMCU a través del protocolo SPI y con ayuda de las librerías "SPI.h" y "MFRC522.h" en el código embebido [38]. La siguiente figura muestra en *pinout* de conexión entre el RFID-522 y el NodeMCU.

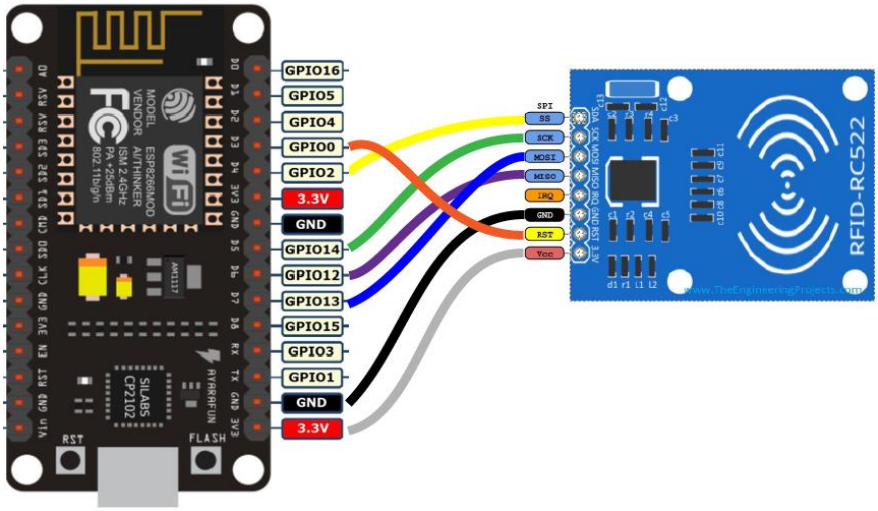

Figura 9: Conexión entre NodeMCU y RFID-RC522 [69]

La configuración que muestra la figura 9 es uno de los principales puntos de interés en el proyecto desarrollado, dado que expande las capacidades del NodeMCU como elemento central y habilita las prestaciones de un lector de tecnología RFID, específicamente NFC para el caso de uso en el que se aplica.

3.2.1.3 Regulador de voltaje

Todo el circuito impreso producido como resultado de este trabajo tiene líneas de distribución de voltaje que ofrece dos opciones controladas por una configuración de interruptores. La siguiente figura muestra esto a detalle.

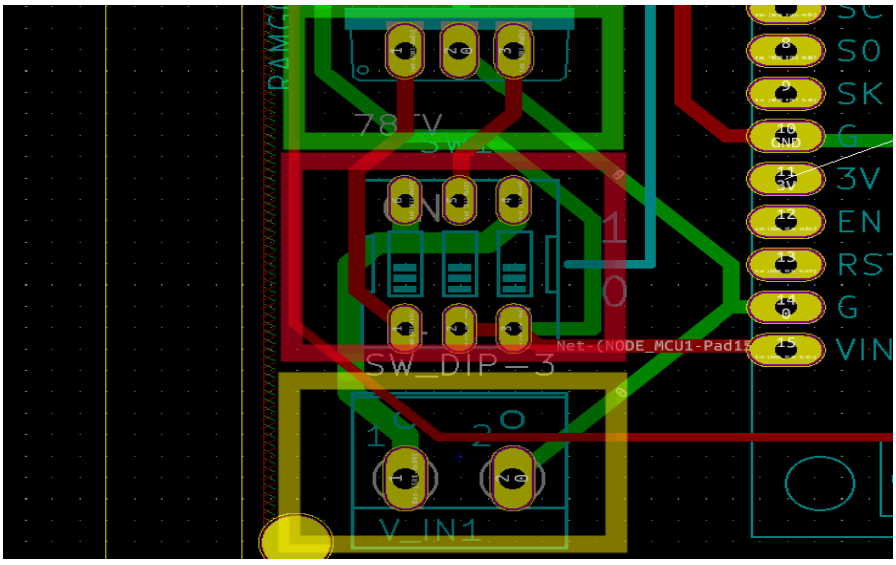

Figura 10: Etapa de distribución eléctrica. Fuente: Elaboración propia

En la figura 10 podemos apreciar tres recuadros:

- En la parte inferior el recuadro amarillo muestra los pines de entrada de voltaje, que puede ser desde 6 hasta 12 VCD (corriente directa).
- En la parte intermedia el recuadro rojo muestra el arreglo de interruptores, bajo una configuración de dip-switch3, que controlan el modo de alimentación del circuito. En configuración 110b el regulador de voltaje entra en funcionamiento, y en configuración 001b la alimentación eléctrica del circuito se toma directamente de la fuente de alimentación externa que se utilice sin pasar por el regulador.
- La parte superior muestra dentro del recuadro verde la posición en la que se coloca un regulador de voltaje LM7805

Tanto la NodeMCU como el RFID-RC522 funcionan a 3.3 VCD; el primero cuenta con su propio regulador interno, a diferencia del segundo que toma su alimentación directamente de la NodeMCU como se muestra en la figura 9. Debido a que los demás elementos del circuito operan a 5 VCD, éste es el voltaje que se suministra a todos los elementos del circuito.

El LM-7805, de acuerdo a su hoja técnica [39], es un regulador de voltaje positivo de 3 terminales con capacidad de hasta 1 ampere con un rango de voltaje de entrada de hasta 35 VCD y una salida fija de 5 VCD. Con estas características se cubren perfectamente las demandas eléctricas del proyecto.

Es importante mencionar que la temperatura de operación de este regulador es directamente proporcional al voltaje de entrada que se le suministre. Para el caso de aplicación de este proyecto se usa una fuente externa de 12 VCD provocando que el regulador alcance una temperatura considerablemente molesta al contacto directo. Es por esto que en el montaje final se auxilia de un disipador de calor metálico como se muestra en la siguiente figura.

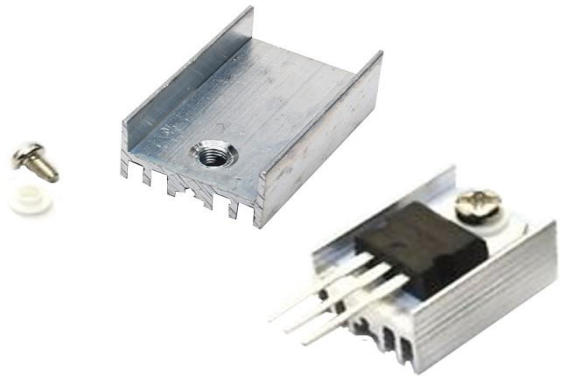

Figura 11: Muestra de un disipador de calor montado en el regulador de voltaje [71]

La configuración de conexiones de un regulador de voltaje como el LM-7805 es relativamente sencilla tal como la siguiente figura lo ilustra

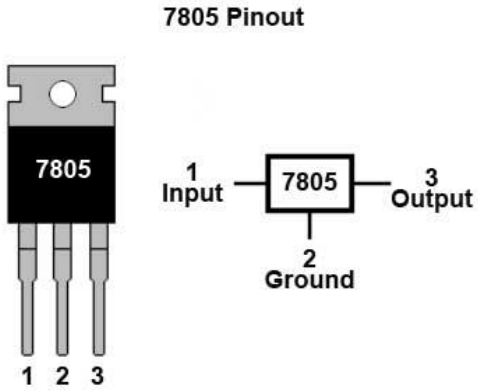

Figura 12: Configuración de conexión del regulador de voltaje [70]

3.2.1.4 Driver de relevadores

Esta etapa de desarrollo del proyecto fue la más problemática de todo el proceso. Un error en la configuración de conexiones de los pines del CI usado como driver entre el NodeMCU y los relevadores causaba un sobrecalentamiento en el regulador de voltaje y la falla total del sistema. Después de varias horas invertidas en investigación e identificación del error se logró solucionar el problema, sin embargo, el primer lote de tarjetas PCB quedó obsoleto y fue necesario manufacturar una segunda versión.

El escenario anterior fue causado por omitir la etapa de prototipado necesaria específicamente para la conexión entre relevadores y microcontrolador, obviando su correcto funcionamiento y como efecto generando un retraso considerable en el desarrollo del proyecto.

Sin embargo, esta experiencia ha dejado una importante lección en el desarrollo de proyectos electrónicos: prototipar siempre antes de fabricar tarjetas de circuito impreso.

El CI involucrado en lo descrito en párrafos anteriores es el ULN-2003, cuya hoja técnica [40] los describe como un circuito integrado compuesto internamente por 7 drivers idénticos e independientes entre sí que permiten controlar por medio de un

microcontrolador relevadores, motores y luces de baja tensión. La siguiente figura muestra dicho CI.

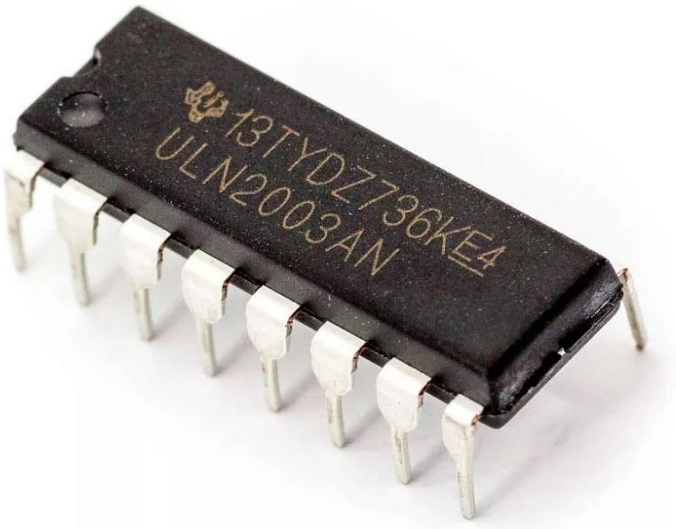

Figura 13: Aspecto físico del CI ULN2003 [67]

Podemos apreciar que su configuración física no es más impresionante que cualquier otro empaquetado de 16 pines. Lo realmente interesante está en el interior, que está ilustrado a continuación.

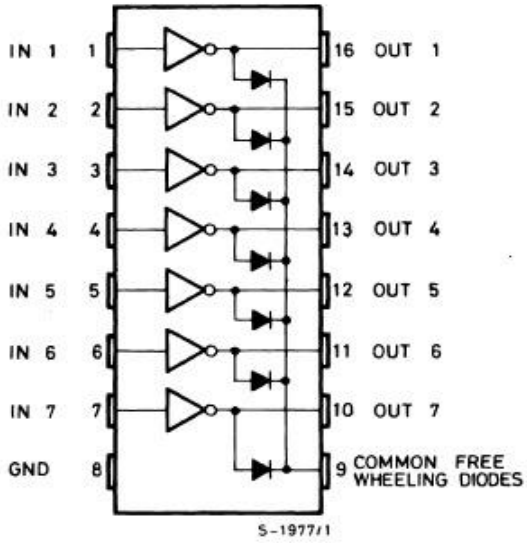

Figura 14: Configuración interna y asignación de pines de ULN2003 [40]

La figura 13 muestra el arreglo de sus 7 drivers. Cada uno está constituido por dos transistores en configuración Darlington, la cual consiste en conectar dos transistores bipolares en cascada para obtener una elevada ganancia, teniendo como resultado la capacidad de poder controlar cargas de cierta potencia con corrientes de entrada pequeñas. Cada uno de estos drivers cuenta internamente con una resistencia de entrada en serie, por lo que se puede conectar el CI directamente al microcontrolador en uso.

Se puede observar también que a la salida de cada driver se tiene un diodo, que sirve para eliminar la tensión de apertura o "rebote" cuando se usan cargas inductivas como los relevadores. Sin embargo, para que estos diodos trabajen es necesario conectar el pin 9 al positivo de las cargas y que las tierras de todo el circuito sean comunes.

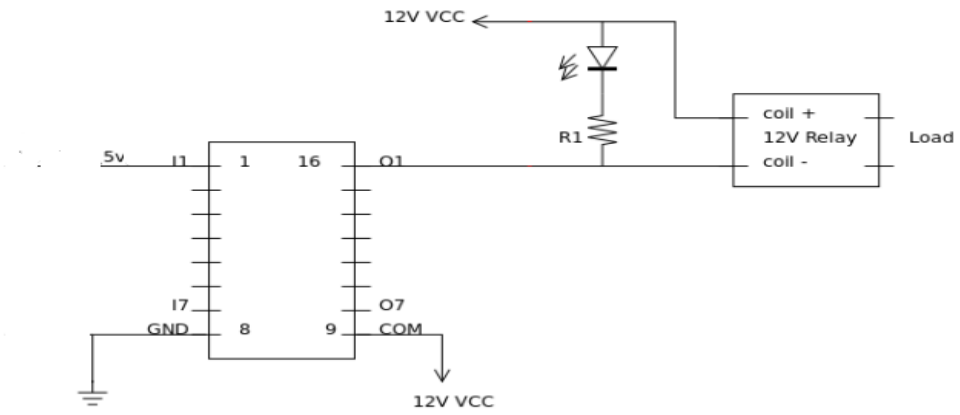

Figura 15: Diagrama básico de conexión de un CI ULN-2003 y un relevador [72]

La siguiente figura muestra un recorte del diseño del circuito impreso en donde el recuadro amarillo resalta la posición del ULN2003 y los dos recuadros azules indican la posición de los dos relevadores usados en el proyecto.

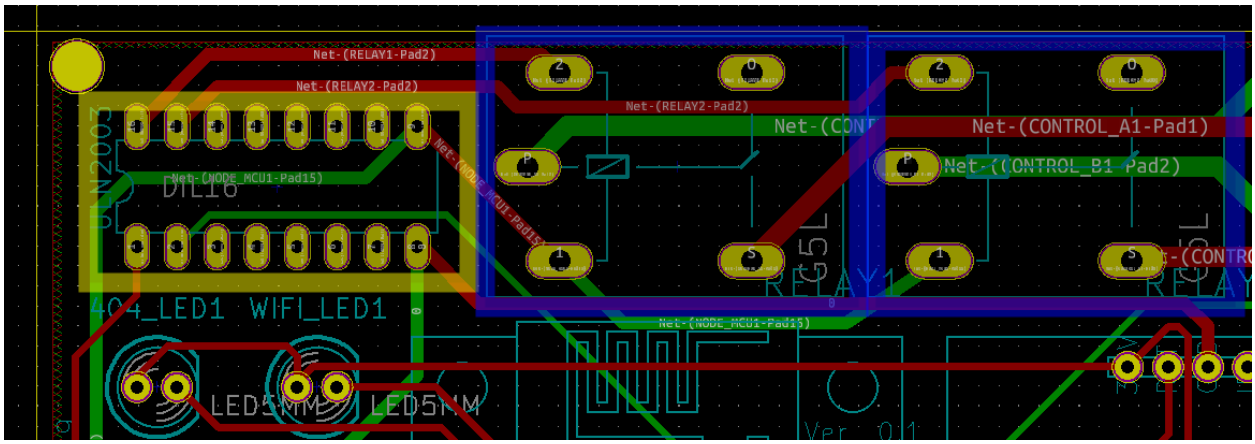

Figura 16: Montaje del driver ULN2003 y dos relevadores. Fuente: elaboración propia

3.2.1.5 Relevadores

Se usan dos relevadores comunes como etapa de salida y control de potencia en el circuito del proyecto. Estos relevadores son controlados por el driver ULN-2003 descrito

en la sección anterior. Este driver recibe las señales digitales de los pines GPIO del NodeMCU.

Un relevador funciona como interruptor para controlar cargas de un voltaje considerablemente mayor a la señal de control. En el desarrollo de este proyecto se usan relevadores de 1 polo y 1 tiro, con activación de 5 VCD y capacidad de control de hasta 250 VCA. Nótese que la señal de control es de corriente continua y la controlada es corriente alterna. Estos relevadores basan su funcionamiento en un par de bobinas que al ser alimentadas generan un campo electromagnético que actúa sobre un mecanismo metálico que actúa como interruptor al contacto. El aspecto físico de estos se muestra a continuación.

Figura 17: Relevador a 5 VCD 1 polo 1 tiro [73]

3.2.1.6 Tarjeta de circuito impreso (PCB)

Las tarjetas de circuito impreso, mejor conocidas por sus siglas en inglés como PCB (*Printed Circuit Board*), son la base de toda la industria de productos electrónicos hasta ahora. Existe una amplia variedad en características de fabricación de estas tarjetas; desde el material del que están hechas que puede ser rígido o flexible, las capas de conexiones disponibles, el color del recubrimiento, calibre de las guías metálicas conductoras, el recubrimiento de acabado aplicado, entre muchas otras. Todo lo descrito en este párrafo es conocimiento nuevo para el autor, adquirido durante el desarrollo del proyecto.

De igual manera, las herramientas disponibles para diseñar estas tarjetas abundan en la industria. Básicamente todas son herramientas de software CAD (*Computer Aided Design*) como los empleados en diseños arquitectónicos. El diferenciador aquí es que los detalles del diseño son específicos para la fabricación de PCBs. Esta última etapa de fabricación requiere de archivos propios de la manufactura computarizada (CAM, *Computer Aided Manufacture*), tales como archivos tipo "gerber" y un conjunto de otros que especifican las rutas de las guías en cada capa, las interconexiones entre las mismas y los diámetros de los orificios cuando se trabaja con diseños "*through hole*" o perforados.

Para el desarrollo del PCB de este proyecto se usó el software libre KiCAD [41] del que se habla más a detalle en la sección "Herramientas utilizadas" de este documento.

Cabe mencionar que la etapa de diseño de PCB de este proyecto fue una de las más demandantes, pues la curva de aprendizaje para adquirir las habilidades necesarias en cuanto a requerimientos de diseño y uso del software demandó una inversión considerable de tiempo y experimentación. Sin embargo, a la conclusión del proyecto se cuenta con los conocimientos suficientes para poder diseñar y fabricar este tipo de tarjetas en un periodo mucho más corto de tiempo.

En lo que concierne a la fabricación de las tarjetas PCB, el mercado global ofrece una variedad de fabricantes que sólo necesitan los archivos de diseño de la tarjeta y el respectivo pago del costo de fabricación para recibir en la puerta de cualquier domicilio los PCB listos para ser ensamblados con sus componentes electrónicos. Para este proyecto se optó por usar los servicios de manufactura de la compañía JLPCB [42] que opera en Hong Kong, China.

Antes de entrar en detalles de cómo fue diseñado y fabricado la tarjeta PCB es mandatorio mencionar que el diseño de la tarjeta impresa depende del diseño del

circuito electrónico esquemático del proyecto. Es un paso previo que, aunque no es imprescindible es de gran utilidad en el momento de posicionar las "huellas" de los componentes en el diseño del PCB, pues con base en las conexiones definidas en el diagrama esquemático la misma herramienta de software sugiere rutas de conexión entre los pines de los componentes, y adicionalmente es la base de la revisión de calidad y comprobación de errores en las conexiones del mismo. Se muestra a continuación el diagrama esquemático del circuito de este proyecto.

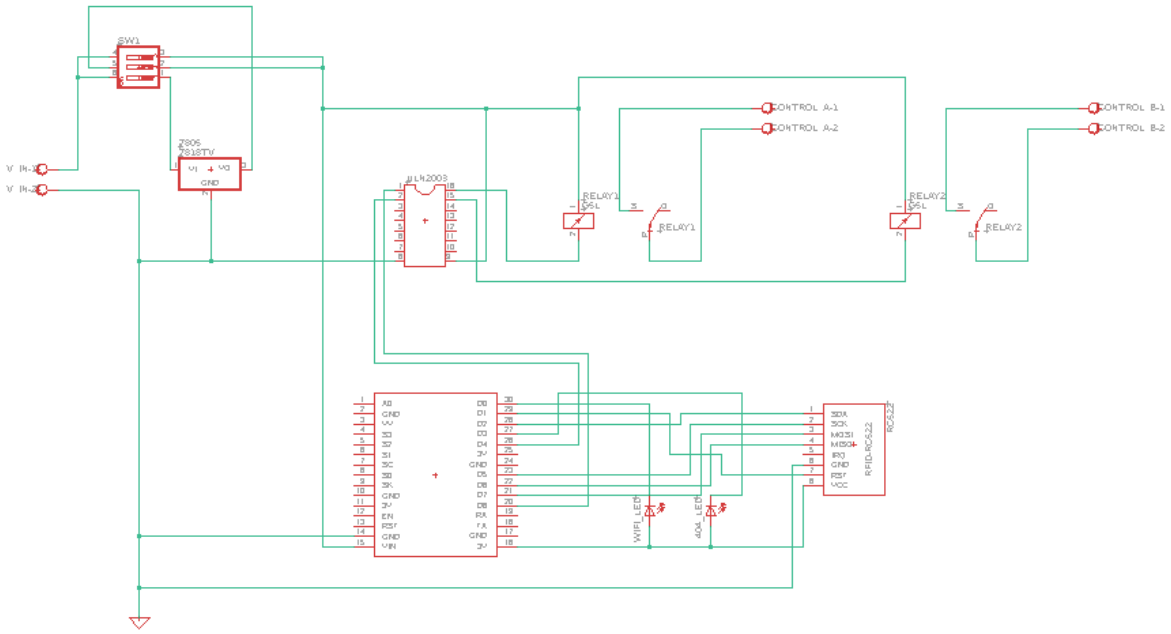

Figura 18: Diagrama esquemático completo del circuito del proyecto. Fuente: Elaboración propia

El diagrama de la figura anterior muestra a un nivel conceptual claro y preciso cada uno de los componentes electrónicos que forman parte del dispositivo y las conexiones entre ellos. Como veremos más adelante, este diagrama no define la posición de los componentes en el PCB. La siguiente figura muestra el diseño completo del PCB.

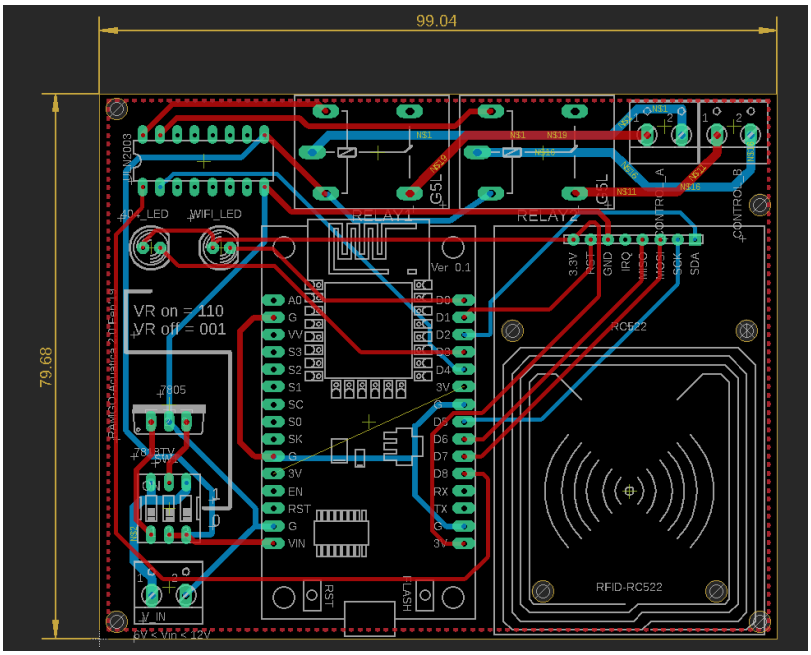

Figura 19: Diseño del PCB completo. Fuente: Elaboración propia

En la figura "Diseño del PCB completo" se puede apreciar que se tienen distintos colores entre las partes del circuito;

- Las marcas en verde representan áreas donde las terminales de los componentes traspasan la tarjeta para ser soldados. A esta metodología de montaje se le conoce como "de tarjeta perforada" o comúnmente en inglés como "*through hole*". Estos puntos tienen una terminación metálica en la superficie de ambas caras de la tarjeta, lo que hace que el trabajo de soldadura sea sencillo. El resto de la tarjeta tiene un recubrimiento protector que evita que la soldadura de estaño se adhiera.

- Las líneas rojas representan las guías de conexión en la capa superior de la tarjeta PCB, lo que significa que en este proyecto se tiene una tarjeta de dos capas perforada. El contar con dos capas para poder interconectar los componentes facilita el trabajo de trazado de las guías, pues la complejidad del diseño aumenta con el número de guías a menor número de capas. La siguiente figura muestra las guías de la capa superior.

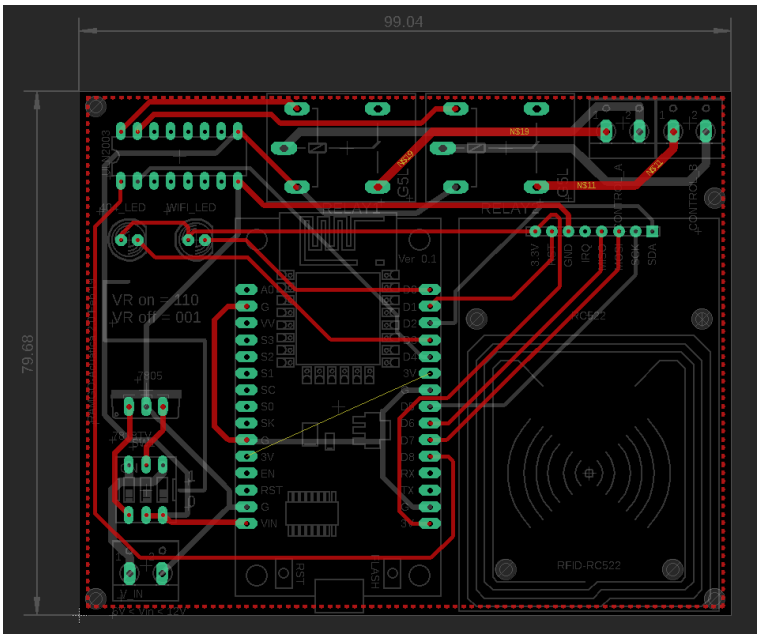

Figura 20: Guías de conexión en la capa superior del PCB.
Fuente: Elaboración propia

- La capa inferior está representada por las guías en color azul. Como se menciona en el párrafo anterior, las guías de cada capa en suma forman el circuito de interconexión total entre los componentes electrónicos del sistema. Podemos apreciar el detalle de la capa inferior en la siguiente figura.

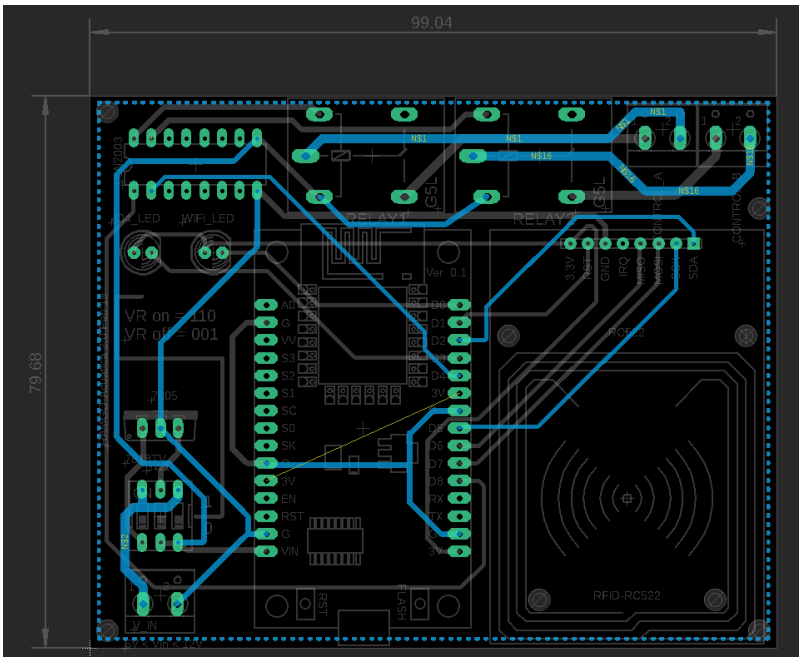

Figura 21: Guías de interconexión en la capa inferior del PCB.
Fuente: Elaboración propia

- Tenemos finalmente, pero no de menor importancia, un último color indicativo en la capa superior del PCB. Los trazos de color blanco son figuras e indicaciones que representan la posición de los componentes que van montados en el PCB. De esta manera el proceso de ensamblado se vuelve fácil y a prueba de errores, pues incluso indica la posición correcta de los elementos. Sirve también para anotaciones indicativas en la tarjeta, como la configuración de los pines del *dip-switch* que determina la fuente de alimentación del sistema.

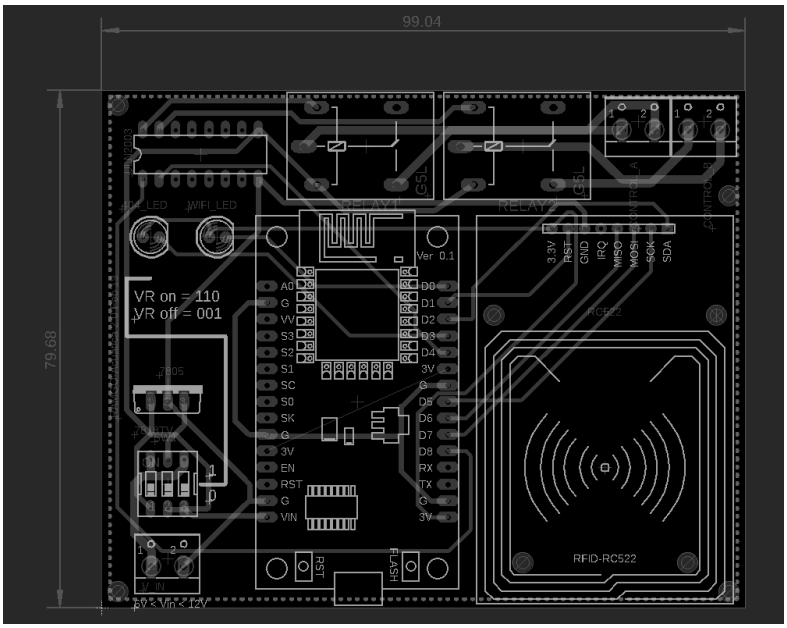

Figura 22: Marcas indicativas impresas sobre el PCB. Fuente: Elaboración propia

Una vez que se contó con los diseños digitales de la tarjeta se procedió a comparar proveedores para su manufactura y se optó por JLPCB como se menciona en párrafos anteriores de esta sección. Este proveedor ofrece diversas opciones en cuanto a acabados, número de piezas y tiempos de entrega. Dentro de su proceso de cotización en su página web cuentan con un visualizador de archivos "gerber" (los usados en el proceso de manufactura) que te muestra una visualización realista del aspecto físico de la tarjeta. De esta manera es posible contar con una última revisión a detalle de los acabados de la tarjeta antes de ordenar su fabricación.

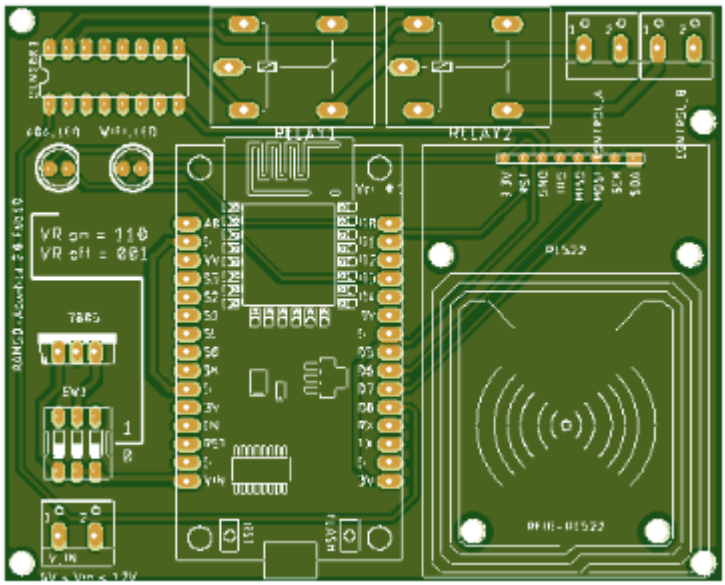

Figura 23: Visualización de aspecto final de la capa superior del PCB. Fuente: Elaboración propia

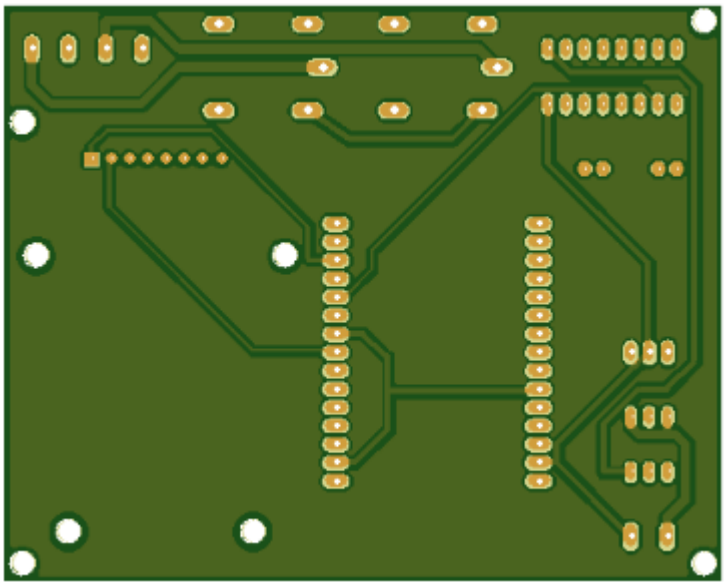

Figura 24: Visualización de aspecto final de la capa inferior del PCB. Fuente: Elaboración propia

Las dos figuras anteriores muestran el aspecto realista de la tarjeta antes de ser ordenada su fabricación.

La fabricación de 10 tarjetas tuvo un costo de $19.95 USD, de los cuáles únicamente $2.00 USD fueron por concepto de fabricación, el resto fue el costo de ordenar una fabricación rápida debido a la urgencia de contar con estas tarjetas para poder avanzar con el desarrollo del proyecto. Bajo este esquema se contó con las tarjetas físicas en menos de una semana, en lugar de 3 semanas que tomaría normalmente.

3.2.2 Microservidor (Raspberry)

De acuerdo con el diagrama de infraestructura del sistema presentado en la Figura 22: Arquitectura general del sistema. Fuente: Elaboración propia, el proyecto cuenta con un componente "Local Server" al cual se le ha denominado como microservidor para fines de esta redacción. Este microservidor tiene una función específica: mantener funcionando el sistema en caso de falla de conexión a internet.

Gracias a esta funcionalidad de respaldo el usuario final no tiene una experiencia negativa al no permitirle el acceso a las instalaciones de la empresa simplemente porque no se cuenta con conexión a internet en ese momento. Obsérvese en Figura 22: Arquitectura general del sistema. Fuente: Elaboración propia cómo el flujo de datos siempre pasa por el microservidor, y en ningún momento se tiene una comunicación directa entre los dispositivos y el servicio del tercero. De esta manera y por medio de la programación del microservidor, éste registra todos los identificadores de usuario leídos y guarda su respectiva respuesta de acceso; concedido o negado. Así, cuando no se cuenta con una conexión activa a internet que permita la comunicación con el servicio del tercero el microservidor actuará como punto de verificación para los accesos de los usuarios.

Otra función de suma importancia que justifica el uso de un microservidor local es que este mismo actúa como central de control de los dispositivos, puesto que a través de una conexión de red privada virtual (VPN) es posible hacer programaciones remotas (*OTA*) en caso de mantenimiento o cambios requeridos por el cliente. Al actuar como central de control también es posible verificar si los dispositivos están activos o no.

En la sección siguiente se detalla la configuración del microservidor, así como los servicios habilitados en el mismo para habilitar sus prestaciones.

3.2.2.1 Tarjeta Raspberry Pi

La tarjeta de desarrollo Raspberry Pi es una conocida plataforma que brinda un poder de cómputo bastante decente en un hardware del tamaño de una tarjeta de crédito. Existen distintas versiones como es costumbre en el mundo de la tecnología, sin embargo, se presentan sus principales características técnicas de acuerdo con la asociación *Open Source* [24].

Su principal característica es que su costo es de tan sólo $35 USD, lo que es un precio accesible para el desarrollo de proyectos que requieren un poder de cómputo más allá de un microcontrolador tipo Arduino. La Raspberry Pi corre sistemas operativos basados en Linux y ofrece conexión de pines GPIO para control electrónico. Tiene conectividad de red a través de puerto Ethernet y Wi-Fi, además de Bluetooth y conexión USB.

Para el desarrollo de este proyecto se usa la versión 3 de la Raspberry Pi, que se muestra en la figura siguiente.

Figura 25: Raspberry Pi 3, comparación de tamaño [74]

3.2.2.2 Configuración de la imagen

Toda tarjeta Raspberry Pi debe tener una "imagen", es decir, un sistema operativo corriendo sobre la arquitectura del hardware. Al ser capaz de correr distribuciones de Linux sobre arquitectura ARM las opciones de sistemas operativos disponibles para usar son prácticamente infinitas. Incluso es posible "cocinar" la propia imagen por medio de la configuración de "recetas" usando herramientas como *Yocto* [43], sin embargo, el diseño de un sistema operativo a la medida está fuera de los alcances de este proyecto.

En su lugar, se opta por usar la distribución más reconocida y estable para correr sobre una Raspberry Pi; *Raspbian*. De acuerdo a su sitio oficial [44], se trata de un sistema operativo libre, basado en *Debian* y optimizado para el hardware de la Raspberry Pi. Al tratarse de un producto Open Source, Raspbian se actualiza constantemente y se

ofrecen nuevas distribuciones. Generalmente se dispone de tres versiones de estas distribuciones [45]; con entorno gráfico y software recomendado, sólo con entorno gráfico, y la versión mínima sin entorno gráfico (sólo consola). Para este proyecto se ha seleccionado la versión mínima puesto que no se requiere de un entorno gráfico para su uso y configuración; todo se hace a través de conexión remota y control por línea de comandos.

El procedimiento para "montar la imagen" en una tarjeta de memoria micro SD y su uso en una tarjeta Raspberry Pi se encuentra detallado en el sitio oficial de la fundación [46] y se omiten en la presente redacción para mantener el enfoque del desarrollo del proyecto.

En los párrafos siguientes se detallará el proceso de configuración de la imagen a partir del punto en el que la misma ha sido iniciada por la tarjeta, se ha conectado a una red WiFi y se ha actualizado su paquetería.

- **SSH**

  Dado que se está usando una imagen mínima, toda manipulación del sistema operativo se hace a través de línea de comandos. Para lograr esto desde otra computadora es necesario accesar a través de una conexión SSH [47]. Este tipo de conexión remota se considera segura ya que ofrece encriptación punto a punto de 128 bits. Una vez que se ha habilitado la conectividad SSH en la Raspberry Pi, es posible iniciar sesión de manera remota usando credenciales existentes, las cuales por configuración predeterminada de Raspbian son usuario:pi, contraseña:raspberry. Para fines de redacción se usarán estas credenciales.

  Por configuración de fábrica, Raspbian ya incluye el paquete necesario para usar la conexión ssh; *openssh*. Únicamente es necesario habilitar esta función como se indica en la documentación oficial [48]

- **Python**

  Python es un lenguaje de programación de alto nivel que ha ganado fuertemente popularidad en los últimos años debido a su facilidad de uso y potencial para desarrollar proyectos de una manera rápida, sin descartar el poder de procesamiento y sus prestaciones para profundizar en la elaboración de complejos algoritmos [49].

  El microservidor de este proyecto utiliza Python para su componente de software encargado de procesar las peticiones de los dispositivos de acceso y las respuestas del servicio del tercero. Al igual que ssh, las paqueterías de Python vienen instaladas de fábrica en Raspbian por medio de los paquetes P*ython3.6*. (A la fecha de redacción de este texto, la versión actual de Python es la 3.6, y cambia constantemente).

  En esta sección sólo se está detallando los aspectos de configuración de la imagen para soportar las funcionalidades del microservidor, en secciones subsecuentes se detalla el algoritmo de procesamiento de llamadas.

- **MongoDB**

  Como todo proyecto de procesamiento de transacciones en donde existe un flujo de datos que deberá ser consultado en algún momento es necesario contar con una base de datos que almacene y gestione esta información. Para este proyecto se emplea MongoDB. De acuerdo con su sitio oficial [50] es una base de datos NoSQL basada en documentos, es decir, almacena datos en forma de documentos tipo JSON.

  A pesar de que el proveedor ofrece un servicio de almacenamiento en la nube, para este proyecto se usa una base de datos local, instalada y configurada en el

sistema operativo del microserver. Para esto, se usa la paquetería *mongodb*, y para ser compatible con el código del programa principal escrito en Python es necesario instalar el paquete *pymongo*.

Debido a que las versiones recientes de MongoDB no son compatibles con arquitecturas de 32 bits, como es el caso de la Raspberry Pi 3, es necesario instalar una versión anterior de MongoDB. La secuencia de comandos que se siguió para lograr una base de datos de mongo instalada y compatible con el código escrito en Python es la siguiente:

```
sudo apt install mongodb (installs version 2.4.14)
sudo apt install python-pip
sudo pip install pymongo==3.4.0 (to be compatible with old version of mongo previously installed)
```

Código 1: Instalación de MongoDB en Raspbian. Fuente: Elaboración propia

Una vez ejecutados los comandos anteriores la imagen del microserver cuenta con la paquetería necesaria para prestar el servicio de una base de datos local de MongoDB. Esta base de datos será manipulada por el script de código en Python que forma parte del backend a instalar posteriormente.

- **VPN**

Para habilitar la posibilidad de brindar mantenimiento remoto al sistema se configura una red privada virtual (VPN, red privada virtual). De acuerdo con [51] una VPN permite crear una red local sin necesidad de que sus integrantes estén físicamente conectados entre sí, sino a través de internet. En términos simples, una VPN permite el acceso remoto al microserver sin necesidad de tenerlo conectado a la misma red en la que se esté trabajando. Basta con unas cuantas sencillas configuraciones de un servicio para poder habilitar esta funcionalidad.

Se usa el servicio de Zero Tier [52] para esta configuración. Una vez creada una cuenta en este servicio, se puede crear una red privada con un identificador único. Este identificador es utilizado para agregar dispositivos como computadoras personales y/o la raspberry en uso. Siguiendo las instrucciones del proveedor [53] para configurar tanto el microserver como la computadora personal a usar para la configuración del sistema, se puede tener una comunicación directa y lista para proporcionar acceso por ssh. La siguiente figura muestra una captura del panel de control de Zero Tier con la VPN habilitada para el proyecto.

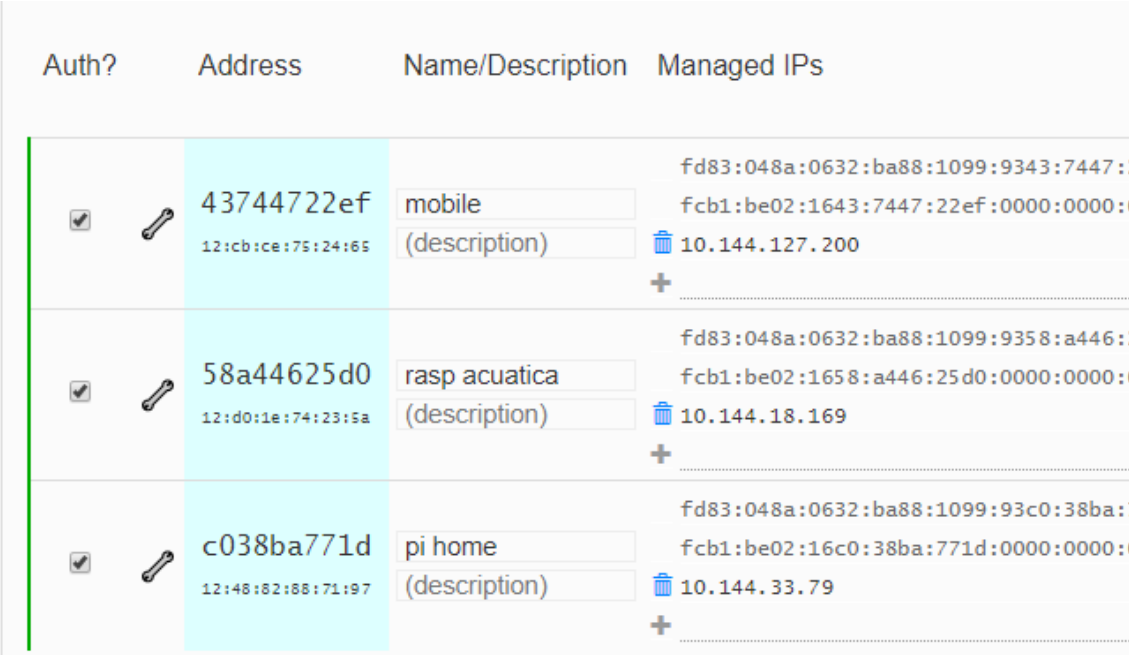

Figura 26: Dispositivos activados en VPN. Fuente: Elaboración propia

Como se puede observar en la Figura 183: Dispositivos activados en VPN. Fuente: Elaboración propia) esta configuración de VPN cuenta con tres dispositivos:

- mobile: el teléfono celular del autor. Se usa para poder accesar al microserver desde prácticamente cualquier ubicación siempre y cuando se cuente con una conexión de datos

- rasp acuática: el microserver ubicado en las instalaciones de la empresa, listo para establecer una conexión y ofrecer las herramientas de mantenimiento.
- pi home: la computadora de uso personal del autor.

Hasta este punto se cuenta con una tarjeta Raspberry Pi configurada y lista para funcionar como microservidor del sistema. Antes de hablar sobre los códigos de software implementados tanto en los dispositivos como en el microservidor se presentan las herramientas utilizadas para el desarrollo del proyecto.

3.2.3 Herramientas utilizadas

En esta sección se detallarán esas herramientas de diseño y desarrollo necesarias para este proyecto.

3.2.3.1 Arduino IDE

Como se ha confirmado en la sección 2.7 (Relación con el proyecto) este IDE es el más conveniente a usar para la programación de los dispositivos. De acuerdo a su sitio oficial [18] es una herramienta que contiene distintas prestaciones, como un editor de texto y monitor serial, para cargar código a dispositivos embebidos compatibles (como el NodeMCU). Para contar con esta herramienta, basta con seguir sus instrucciones de instalación y con unos cuantos pasos estará listo para usarse.

Se requiere una configuración adicional para poder programar la NodeMCU con el Arduino IDE. Basta con agregar un archivo json a las configuraciones para que el IDE pueda comunicarse con el ESP8266, como se detalla en la documentación oficial [54].

3.2.3.2 PyCharm

Es un IDE especializado para el desarrollo de software escrito en lenguaje de programación Python. Aunque el software puede ser escrito en cualquier editor de texto simple, usar un IDE especializado proporciona herramientas de gran utilidad para

acelerar el proceso de desarrollo, validación y solución de errores. Incluso cuenta con un verificador de PEPs (*Python Enhacement Proposals*), que es el estándar mundial para verificar la calidad de código escrito en este lenguaje [55]. Para el desarrollo de este proyecto se usa la versión *community* del IDE, cuyas instrucciones de instalación se encuentran en su documentación oficial [56].

3.2.3.3 Postman

Debido a que en el proyecto se usan llamadas a *APIs* a través de *HTTP*, es necesario contar con una herramienta que permita hacer pruebas de conexión para analizar la estructura de los datos que se envían y se reciben por estas llamadas. Para lograr esto se usa *Postman* [57], que permite direccionar los *get requests* a la dirección web que sea definida en el servidor, poder estructurar los datos de envío y observar la respuesta. De igual forma, permite analizar el tiempo de respuesta de nuestro *microserver*, lo cual es una característica importante al pensar en la experiencia del usuario final de este proyecto. *Postman* permite realizar las iteraciones necesarias de una manera rápida y ágil para ir probando con cambios en el código del lado del servidor y observar las respuestas en los dispositivos.

3.2.3.4 KiCAD

Para el diseño de las tarjetas PCB del proyecto se optó por usar el software KiCAD [41] que es una herramienta *Open Source* para el diseño electrónico. Se usan dos funciones principales, las cuales están directamente relacionadas: diseño esquemático del circuito electrónico y diseño de *PCB* con su respectiva generación de archivos *Gerber* para su fabricación. KiCAD es un software libre bajo la licencia GNU GPL v3 [58]. Es importante mencionar que dentro de los principales patrocinadores que soportan el desarrollo de esta herramienta es posible encontrar a la fundación *Raspberry Pi* [59] y a *Arduino LLC*, lo que demuestra que el ecosistema de desarrollo *Open Source* es amplio e interconectado, listo para brindar las herramientas necesarias para el desarrollo de cualquier tipo de proyectos.

3.3 ETAPAS DE MADURACIÓN DEL PROYECTO

Hasta este punto se cuenta ya con los antecedentes, marco teórico, definición de objetivos, descripciones técnicas tanto para hardware y software, así como las herramientas necesarias para construir el proyecto objeto del presente documento. Con base en esto, en la presente sección se avanzará progresivamente sobre los niveles del modelo *TRL* [28] haciendo referencia al desarrollo del proyecto y se detallarán las acciones necesarias y suficientes para poder recrear el trabajo de investigación.

TRL 1: Investigación aplicada

Como se definió en el marco teórico, este nivel está identificado por una investigación aplicada/tecnológica básica. Una vez identificada la problemática, se identificaron las posibles maneras de resolverla con las soluciones tecnológicas existentes. Este primer acercamiento surge del diálogo con los interesados que en su momento buscaban cómo digitalizar sus principales procesos de negocio. Específicamente surge el área de oportunidad para la aplicación de un sistema tecnológico de donde surge el proyecto aquí presentado.

De manera paralela, la conversación toma un aire de consultoría. Con base en la experiencia se diseña una arquitectura y se inicia uno de los principales retos del autor; la definición del modelo de negocio, iniciando por una cotización y cálculo del tiempo de desarrollo, así como comparar el trabajo con lo que se encuentra en el mercado.

TRL 2: Validación conceptual

Esta etapa toma como base la anterior, al igual que en toda la secuencia de la escala *TRL*. Se define como la conceptualización de la tecnología o formulación de la aplicación y consiste en identificar cómo las tecnologías disponibles conformarían los distintos componentes del proyecto, identificando también la forma correcta de su integración con base en investigaciones y trabajos publicados. Debido a la naturaleza del proyecto, la disponibilidad de información informal sobre proyectos relacionados es

amplia en la red. Esta es una especial área de oportunidad ya que es sólo un punto de partida, pero se puede llegar a documentos oficiales e indexados como a los que se ha hecho referencia a lo largo del documento.

TRL 3: Pruebas de concepto analíticas y experimentales de funciones críticas

En esta etapa se comenzaron a construir prototipos para validar el funcionamiento de las tecnologías que se encontraron previamente en la investigación básica. Estas pruebas de concepto fueron validando una a una las tecnologías a usar y descartando aquellas que no eran funcionales o compatibles entre sí. Como estamos hablando de un proyecto de desarrollo de hardware, que lleva software embebido, los prototipos consistieron en probar y documentar las aplicaciones básicas de cada uno de los módulos que conformarían, hasta ese punto experimental, la integración del proyecto. Aquí el objetivo fue verificar que el concepto funciona, bajo un esquema modular.

Al avanzar con las pruebas de funcionamiento de cada módulo experimental se fue diseñando el plan de desarrollo y se iniciaron las pruebas de integración entre los módulos. Aquí se denota el inicio de la siguiente etapa del modelo.

TRL 4: Desarrollo tecnológico

Este nivel está definido por la validación de componentes o sistema en un ambiente de laboratorio. Aquí, se comenzó a trabajar en la integración de los componentes del sistema, aún no constituido como un ente independiente y robusto, sino que cuenta con conexiones a equipo externo e instrumentación de medición y monitoreo. El objetivo aquí es lograr la interacción entre los componentes.

Desde una perspectiva de negocios, se comienzan a identificar nuevas oportunidades de aplicación del proyecto como resultado de la observación en las pruebas de concepto e integraciones previas. A su vez, se comienzan a identificar los requerimientos para el desarrollo y fabricación de un concepto de producto final, el cual comienza a

ser definido por las restricciones y requisito se interacción entre los componentes, además del escenario de aplicación.

TRL 5: Validación de componentes en un ambiente relevante

Este nivel se identifica al tener los componentes integrados a manera que la configuración del sistema sea similar a su aplicación final, pero su operatividad es aún a nivel laboratorio.

Representa el puente entre la investigación y la ingeniería, ya que se comienzan a hacer calibraciones y modificaciones para aumentar la fidelidad de los componentes en sus funciones básicas y en su interacción como sistema, Estas modificaciones son resultado del aprendizaje obtenido en las pruebas de concepto modulares hechas en etapas previas.

La solución ha sido validada a través de pruebas en el entorno previsto, simulado o real. El nuevo hardware, a nivel prototipo, está listo para comenzar a ser usado.

TRL 6: Demostración tecnológica

Este nivel está definido por la validación del sistema de ingeniería en condiciones relevantes similares a las operativas, pero aún a nivel prototipo.

Aquí se contaba ya con un prototipo capaz de desarrollar todas las funciones requeridas por el sistema final. Es importante mencionar que la metodología de *Lean Startup [60]* identifica este momento como la ejecución de un *MVP (Minimum Viable Product),* pues, aunque se tenga la creencia de que el prototipo cubre ya todas las necesidades de la solución, es muy probable que siempre surjan nuevos requerimientos o sean necesarias algunas modificaciones para adaptarse a estos cambios.

En esta etapa de desarrollo se hicieron pruebas funcionales en el sitio final de instalación del proyecto, aun cuando tan sólo se tenía un prototipo. Dichas pruebas consistieron en llevar el prototipo del sistema, aún montado en un *protoboard* a las instalaciones de la empresa para hacer pruebas de comunicación e instalación:

- Se hicieron pruebas para verificar que el alcance del lector NFC, que de acuerdo al fabricante es de hasta 50mm [37], fuera suficiente para leer las tarjetas a través del *housing* en donde quedaría finalmente instalado el dispositivo.
- Se realizó una exploración de la tarjeta madre que controla los torniquetes con el objetivo de identificar los puntos de conexión para ejecutar la apertura de los mismos, así como la identificación de las opciones de toma de corriente para alimentar el dispositivo. Esto se logró y fue determinante para continuar con el desarrollo del proyecto, ya que determinó instrucciones específicas en el código de programación del embebido y configuración de conexiones en el diseño del PCB.
- Se hizo una observación de la infraestructura de red disponible para identificar el punto de conexión para el microserver. Esto resultó en un nuevo requerimiento para instalar un *access-point* que levantara una red local exclusiva para su uso con el sistema de control de acceso. De esta manera, se aumentó la seguridad del sistema al estar cerrado y ser independiente, a nivel de configuración de red, del resto de la infraestructura. Es importante mencionar que el punto de acceso a internet siguió siendo la red local de la empresa.

TRL 7: Comisionamiento de sistemas

Después de la identificación de modificaciones necesarias que se descubrieron en la prueba de campo del nivel anterior se continuó trabajando en el refinamiento del diseño tanto del dispositivo como de la infraestructura necesaria para la operación de esta solución.

Gracias a este trabajo de rediseño, en este nivel se logró tener un prototipo final con su respectivo sistema operativo funcional al 100%. Sin embargo, como parte del proceso se han identificado las cuestiones referentes a la fabricación y operaciones finales. Entre otras cosas, se han resuelto complicaciones tecnológicas menores. Previo a solicitar la producción de las tarjetas electrónicas se hace una primera corrida de sistema completo, monitoreando en todo momento el comportamiento del mismo, y ejecutando pruebas con base en las especificaciones finales.

TRL 8: Sistema completo y funcional para su aplicación final.

Este nivel se caracteriza por tener un sistema final completo que ha sido evaluado a través de pruebas y demostraciones. Al haber pasado satisfactoriamente las pruebas, significa que el desarrollo del sistema ha concluido.

En este punto se elabora la respectiva documentación para su uso y mantenimiento, tomando como base los registros de todos los trabajos realizados durante el proceso del proyecto.

TRL 9: Operación del sistema.

Una vez que el desarrollo del sistema ha concluido es momento de proceder a la instalación del mismo en la ubicación final. Aquí se procedió a llevar tanto el microserver como los 4 dispositivos a las instalaciones de la empresa. La siguiente imagen muestra uno de los dispositivos terminados.

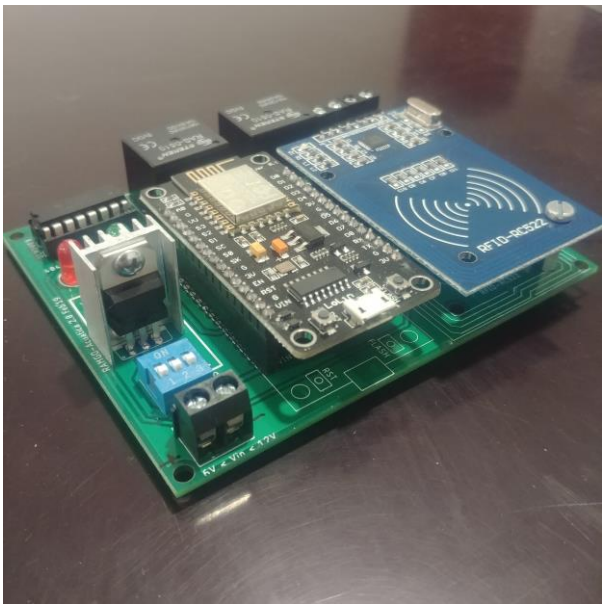

Figura 27: Uno de los dispositivos terminado y listo para ser instalado. Fuente: Elaboración propia

En resumen, las principales tareas a ejecutar para el proceso de instalación fueron:

- Configuración de la red de uso exclusivo del sistema
- Conexión del microserver y dispositivos a la red exclusiva
- Asignación de direcciones IP estáticas a cada dispositivo del sistema
- Instalación del microserver en una ubicación oculta y segura para evitar manipulaciones
- Instalación de dispositivos en cada uno de los puntos de control de acceso
- Prueba y calibración de tiempos de apertura
- Conexión del sistema al *endpoint* del servidor externo para conexión al sistema de administración
- Configuración de la VPN del sistema para acceso remoto
- Pruebas con usuarios finales

Todas estas tareas tomaron aproximadamente 24 horas de trabajo, en 3 intervenciones. Es importante mencionar que más allá del profundo aprendizaje logrado durante el desarrollo del proyecto, el mayor reto es asegurarse de que la instalación del sistema sea lo suficientemente robusta para evitar comportamientos inesperados.

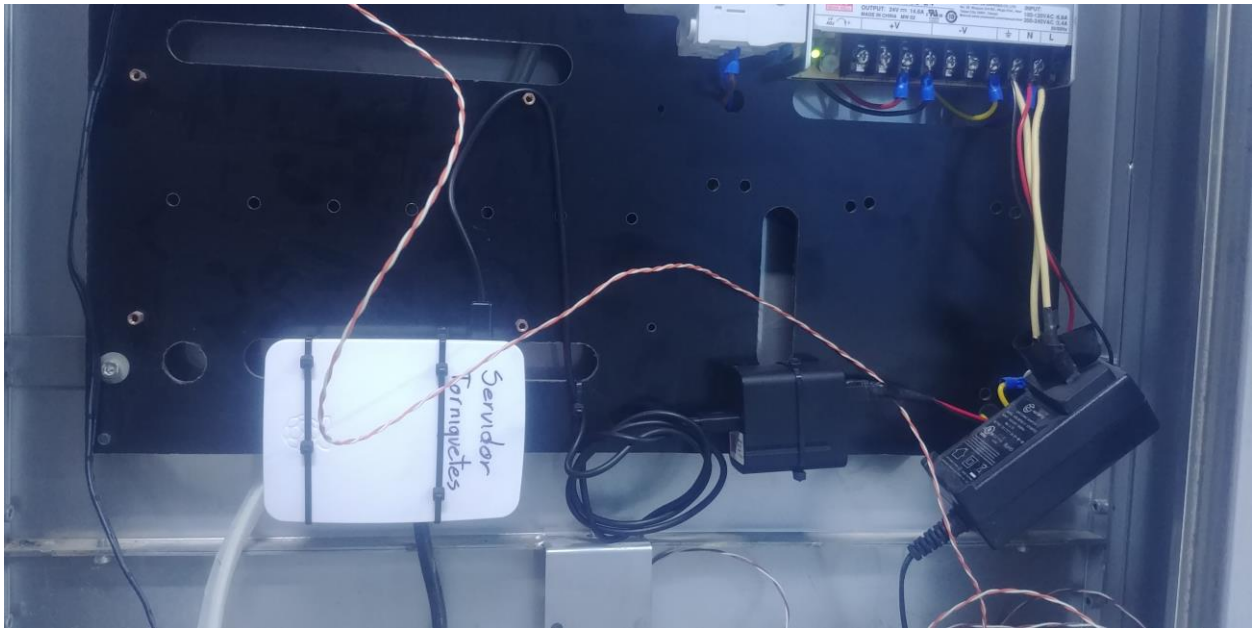

Figura 28: Instalación del microserver. Fuente: Elaboración propia

La figura de la parte superior muestra la instalación del microserver. Éste funciona como el nodo central de todas las operaciones del sistema.

Figura 29: Housing de los dispositivos. Fuente: Elaboración propia

Cada dispositivo quedó protegido por un housing que lo deja instalado fijamente en la posición correcta para ejecutar su funcionamiento.

Una de las etapas que más tiempo demanda en el proceso de instalación es el de calibración de tiempos y pruebas del sistema. Aquí es común encontrar comportamientos no esperados ni considerados en las etapas de diseño y desarrollo, las cuales están relacionadas con el entorno de instalación del sistema. Por ejemplo, se descubrió que el usar cierto tipo de tornillos para fijar el dispositivo distorsionaba el campo electromagnético del lector NFC provocando un comportamiento erróneo y aleatorio. Afortunadamente, el autor se percató de dicha causa raíz y se solucionó a tiempo. La siguiente imagen muestra uno de los momentos en los que el sistema funciona correctamente.

Figura 30: Primer dispositivo funcionando correctamente. Fuente: Elaboración propia

3.4 DESARROLLO TÉCNICO

En esta sección se presentan los distintos módulos de software desarrollados y las configuraciones pertinentes tanto para el funcionamiento del microserver como del dispositivo. Todos los códigos aquí mostrados se encuentran en un repositorio de github:

https://github.com/edmundormz/aquatica

3.4.1 Microserver

Como se ha mencionado previamente, el microserver del proyecto funciona como un gestor del sistema y puerta de acceso a la red para los mensajes entre los dispositivos y el endpoint del sistema de administración externo.

La siguiente figura muestra un diagrama de flujo que representa el proceso que se sigue en el microserver al recibir una petición del dispositivo con un ID de usuario.

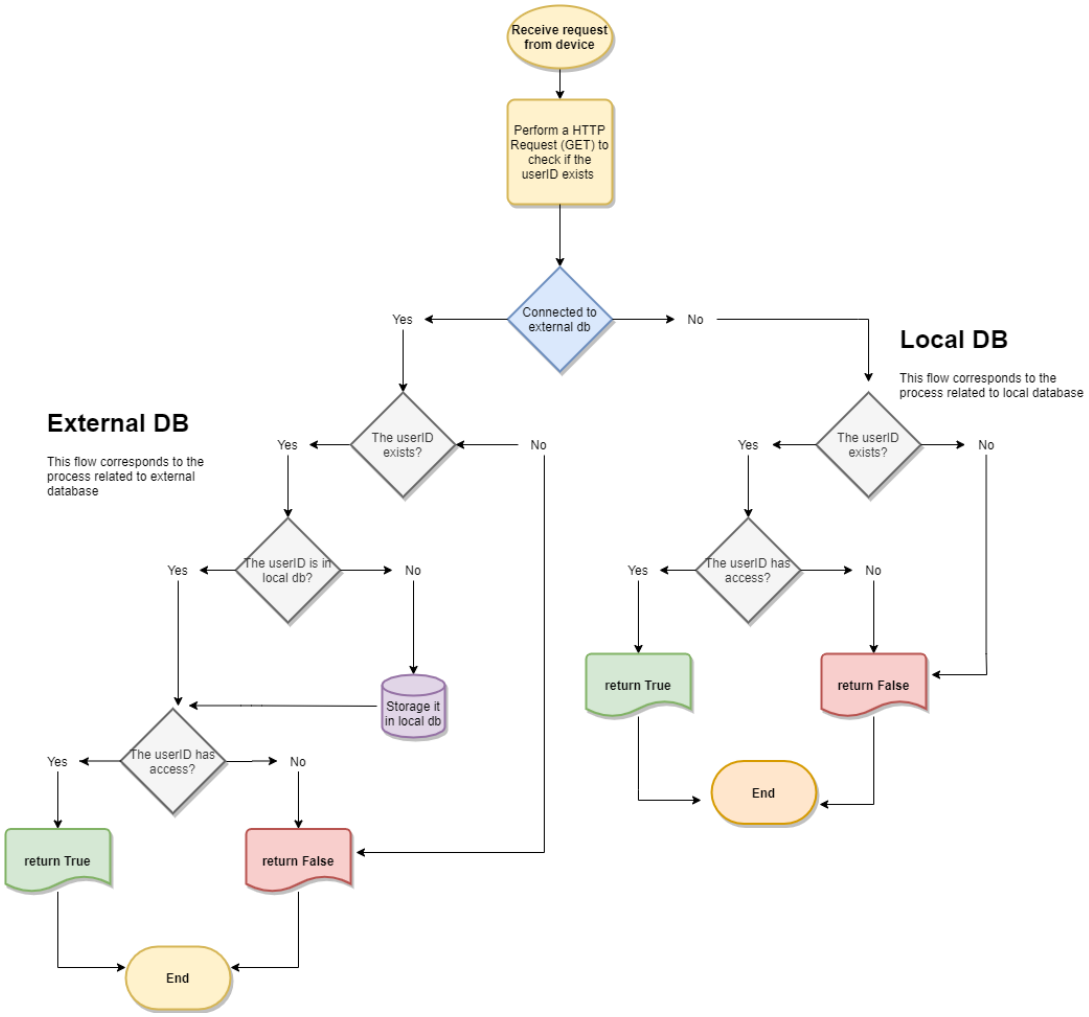

Figura 31: Diagrama de flujo del proceso ejecutado n el microserver. Fuente: Elaboración propia

El evento de inicio es cuando se recibe una petición desde cualquiera de los dispositivos (cuando se lee una tarjeta NFC). Esta petición lleva como argumento un identificador (id) que será usado durante toda la secuencia del proceso. El microserver tiene un servidor http corriendo durante todo el tiempo que éste está encendido, escuchando por peticiones de los dispositivos.

Se invoca una función de llamada HTTP para revisar si el *id* está registrado. El primer elemento condicional del diagrama representa una verificación para detectar si se tiene conexión a internet, específicamente a la base de datos en la nube a la que se tiene acceso a través del endpoint del sistema de administración en la nube.

El siguiente paso es una verificación para saber si el id está previamente registrado en el sistema. En caso de que no sea así, se termina el proceso con una respuesta negativa a la solicitud de acceso. Si el id sí está registrado en el sistema, se revisa si el mismo está registrado también en la base de datos local del microserver. Todo id que pasa por el proceso queda almacenado en la base de datos local con su respectiva respuesta de autorización de acceso, esto para tener un respaldo en caso de falla de conexión a internet.

Seguido a esto, se verifica si el id tiene permitido el acceso en el momento de la solicitud, pudiendo aquí ser una respuesta positiva o negativa y terminando el proceso enviando la misma al dispositivo.

Regresando al primer condicional del proceso, en caso de no tener una conexión al endpoint externo, se continúa el mismo proceso de identificación y verificación de acceso del id en proceso con base en los registros de la base de datos local.

3.4.1.1 Código

El proceso mostrado en el diagrama de flujo previamente presentado se implementó en un código de software en lenguaje Python. Está conformado por tres scripts colaborativos:

- hello.py

    - Es la función principal, está corriendo en segundo plano escuchando por peticiones http de los dispositivos. Este mismo invoca a los siguientes dos scripts dependiendo del camino a tomar en el proceso.

- external.py
    - Este script gestiona la conexión a la base de datos externa (endpoint del sistema de administración de la empresa).
- local.py
    - Este script se encarga de ejecutar las operaciones CRUD [61] en la base de datos de mongo instalada en el microserver.

Estos scripts se muestran a continuación

hello.py

```
#!/usr/bin/env python
# Remember to:
# export FLASK_APP=hello.py (this file name)
# To make server externally visible use: --host=0.0.0.0 (flask run --host=0.0.0.0)

from __future__ import print_function
from flask import Flask

import mongo.databases.local as internal_db
import mongo.databases.external as external_db

app = Flask(__name__)

@app.route('/check_user_id/<user_id>', methods=['GET'])
def external_check_user_id(user_id):
    """Check if a userID exists in external database
    :return
    - True: if the userID has access
    - False: if the userID has not access
    """

    db = external_db.Database(user_id)
    response = db.get_response()  # contains a dict
    local_db = internal_db.Database(user_id)

    if response['status'] == 'connected':
        print('(info) - connected to external database, checking userID')
        if response['response'] == 'true':
            local_db.insert_or_update_user_id(response['response'])
            return 'true'
        elif response['response'] == 'false':
            local_db.insert_or_update_user_id(response['response'])
            return 'false'
        elif local_db.is_authorized():
            return 'true'
        else:
            return 'false'
    else:
        print('(warning) - could not connected to external database, checking userID from internal database')
        # Check if the user exists and is authorized in local database
        if local_db.check_user_id() and local_db.is_authorized():
            return 'true'
        elif local_db.check_user_id() and not local_db.is_authorized():
            return 'false'
        elif not local_db.check_user_id():
            print('(info) - userID: ({}): does not has any record in local database'.format(user_id))
            print('(info) - TIP: the next time that connected to external database (if the userID exists) it will '
                  'be inserted in local database to consult it')
            return 'false'

if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=80)
```

local.py

```
"""Services to handle a local mongodb connection"""

import pymongo


class Database(object):

    INTERNAL_URI = 'mongodb://localhost:27017'
    MAX_SER_SEL_DELAY = 3  # connection delay in milliseconds

    def __init__(self, user_id):
        """Perform the connection to local database

        Once this class is instanciated, the connection will perform to be used in the class functions

        :param user_id: the user id to check
        """
        self.user_id = user_id

        try:
            self.client = pymongo.MongoClient(Database.INTERNAL_URI, Database.MAX_SER_SEL_DELAY)
            self.client.server_info()  # force the connection request
        except pymongo.errors.ServerSelectionTimeoutError as err:
            print('could not connected to local db because: {}'.format(err))

        self.db = self.client['aquatica']
        self.collection = self.db['users']

    def check_user_id(self):
        """Check an user id

        Performs a query into a specific collection returning specific fields

        :return:
        - True: If the userID exists
        - False: if the userID does not exists
        """
        #user_id = self.collection.find_one({'userID': self.user_id}, {'userID': 1})  # returns the userID (if any)
        user_id = self.collection.find_one({'userID': self.user_id})  # returns the userID (if any)
        user_exist = True if user_id else False

        return user_exist
```

```
def is_authorized(self):
    """Check if an user id is authorized to access

    Perform a query into a specific collection returning specific fields

    :return:
    - True: If the userID is authorized to access
    - False: if the userID is not authorized to access
    """
    #authorize = self.collection.find_one({'userID': self.user_id}, {'authorized': 'true'})  # returns authorize dict
    #return True if authorize else False
    authorize = self.collection.find_one({'userID': self.user_id}, {'authorized': 1})  # returns authorize dict
    return True if authorize['authorized'] == 'true' else False

  def insert_or_update_user_id(self, authorized):
    """Insert or update an userID

    If the user does not exist this will insert a new record, otherwise if the user exists this will check
    if there is some new value and it will be updated.

    Note: this function has concurrency allowing to update one by one the existing records in the local database
        avoiding duplicates
    """
    _id = self.collection.find_one({'userID': self.user_id}, {'_id': 1})  # returns the _id (if any)
    data_to_insert = {'userID': self.user_id, 'authorized': authorized}  # data example

    if not _id:
        # the record does not exists in db
        self.collection.insert_one(data_to_insert)
        print('(info) - record successfully inserted: {}'.format(data_to_insert))
    else:
        # the record already exists in db
        self.collection.update_one(_id, {'$set': data_to_insert}, upsert=True)
```

external.py

```
class Database(object):

    def __init__(self, user_id):
        """Set the api-endpoint

        :param user_id: the userID to check in the external database
        """
        # api-endpoint
        self.api_endpoint = 'http://142.93.93.25/api/Students/canPass?nfcId={}'.format(user_id)

    def get_response(self):
        """Get response from external server

        The external server can returns the following values:
        - String:true: => when a userID exists in the database and has access
        - String:false => when a userID exists in the database and has not access

        :return:
            - a dictionary that contains if the connection was established or not and the response received from
              the external server
        """
        timeout = 3  # timeout to close connection with the database (unit:seconds)

        try:
            # sending get request and saving the response as response object
            response = requests.get(url=self.api_endpoint, timeout=timeout)
            data = {'status': 'connected', 'response': 'false'} if response.text == 'false' \
                else {'status': 'connected', 'response': response.text}
            return data
        except requests.ConnectTimeout as err:
            print('(error) - failed to connect to external database due to: {}'.format(err))
            return {'status': 'disconnected', 'response': 'false'}
```

3.4.2 Dispositivo

Previamente se han mencionado detalles técnicos en lo que refiere al hardware del dispositivo. Recordando, se compone por una NodeMCU como elemento principal, montado en una tarjeta de circuito impreso (PCB), la cual funciona como plataforma para la interacción con los demás componentes electrónicos (lector NFC, relevadores, etc.).

En esta sección se describe el código desarrollado para controlar la electrónica del dispositivo, este código será referido como firmware a partir de este punto.

A continuación, se mostrará el código fuente desarrollado y embebido en el NodeMCU. Se presenta una descripción general y se profundiza en las funcionalidades que cada bloque del código habilita en las prestaciones del proyecto.

```
#include <ESP8266WiFi.h>              //Basic ESP8266 library
#include <ESP8266WiFiMulti.h>         //To connect to different networks
#include <ESP8266HTTPClient.h>        //For making HTTP requests
#include <ESP8266WebServer.h>         //To enable web service and upload scripts
#include <ESP8266mDNS.h>              //To enable multicast DNS service
#include <ESP8266HTTPUpdateServer.h>  //To manage script updating via web
#include <SPI.h>                      //Required to communicate with NFC reader
#include <MFRC522.h>                  //To manage NFC reader

#define SS_PIN 4                      //D2 (NFC reader required)
#define RST_PIN 5                     //D1 (NFC reader required)
#define relay1 15                     //D8
#define relay2 2                      //D4
#define led_404 0                     //D3
#define wifi_led 16                   //D0 BUILTINLED
#define access_relay_timing 500       //Time to leave the relays closed
#define block_relay_timing 1000

ESP8266WiFiMulti wifiMulti;           //WiFi Multi object
MFRC522 mfrc522(SS_PIN, RST_PIN);     //MFRC522 object
HTTPClient http;                      //HTTPClient object
ESP8266WebServer httpServer(80);      //Opens web server on port 80
ESP8266HTTPUpdateServer httpUpdater;  //UpdateServer object

//MRFC522 Variables
String content;
byte letter;

//Network variables
String node_name = "node_" + String(ESP.getChipId());
String server_addresses[2] = {"http://192.192.192.192/check_user_id/",
"http://142.142.142.142/api/Students/canPass?nfcId="};
String client_id = "";
String request = "";
```

El bloque de código en la página anterior muestra las librerías que se incluyen en el script, la definición de pines GPIO, la declaración de instancias de los objetos necesarios y declaración de variables que serán usadas en las funciones subsecuentes. Todo el código del script está comentado de manera que sea fácil de interpretar por el lector. Esta es una práctica que ayuda incluso al autor cuando se desee consultar o modificar el código en un futuro.

```
void setup(void)
{
 //System initialization
 SPI.begin();
 mfrc522.PCD_Init();
 pinMode(led_404, OUTPUT);
 pinMode(relay1, OUTPUT);
 pinMode(relay2, OUTPUT);
 pinMode(wifi_led, OUTPUT);

 //Turn off outputs
 digitalWrite(relay1, LOW);
 digitalWrite(relay2, LOW);
 digitalWrite(wifi_led, HIGH);
 digitalWrite(led_404, HIGH);
}


void connectWiFi(){
 WiFi.mode(WIFI_AP_STA);
 wifiMulti.addAP("SSID", "Password");
 wifiMulti.addAP("HecMundo", "password");
 while (wifiMulti.run() != WL_CONNECTED) {
  digitalWrite(wifi_led, LOW);
  delay(100);
  digitalWrite(wifi_led, HIGH);
  delay(100);
 }
 WiFi.hostname(node_name);
 MDNS.begin(node_name);
 httpUpdater.setup(&httpServer);
 httpServer.begin();
 MDNS.addService("http", "tcp", 80);
 return;
```

Este bloque de código muestra dos funciones; *setup* y *connectWiFi*. La primera es necesaria como parte de las reglas de programación en el IDE de Arduino. Esta función es la encargada de, como su nombre lo dice, configurar el hardware en cuanto a sus pines de entrada/salida se refiere.

La segunda función, *connectWiFi* es la encargada de conectar la tarjeta NodeMCU a una red WiFi. A manera más detallada, se da la instrucción de usar el ESP8266 en modalidad *access-point* y *station*; esto quiere decir que operará como servidor web y como cliente. La funcionalidad como servidor es necesaria para poder habilitar la funcionalidad de hacer actualizaciones de firmware OTA (*Over the air),* es decir, de manera remota y a través de un servicio web. Esto habilita que el autor no necesite desplazarse 500 Km para hacer una actualización de firmware si esto fuera requerido. El modo cliente es simplemente para interactuar con el microserver, a través de una red local.

En esta misma función se le indica al script qué redes inalámbricas debe buscar para intentar una conexión, y en este caso se están usando dos redes para robustecer el funcionamiento del sistema y evitar interrupciones del servicio en caso de que una red deje de funcionar. Por razones de privacidad se usan nombres de redes y credenciales ficticias.

```
void processRFID(){
 for(byte i = 0; i < mfrc522.uid.size; i++){
  content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : ""));
  content.concat(String(mfrc522.uid.uidByte[i], HEX));
 }
 client_id = content;
 content = "";
 return;
}

void blink_led(bool access){
 if (access == true){
     digitalWrite(relay2, HIGH);
     digitalWrite(wifi_led, LOW);
     delay(access_relay_timing);
     digitalWrite(relay2, LOW);
     digitalWrite(wifi_led, HIGH);
    }
    else{
     digitalWrite(relay1, HIGH);
     digitalWrite(led_404, LOW);
     delay(block_relay_timing);
     digitalWrite(relay1, LOW);
     digitalWrite(led_404, HIGH);
    }
 return;
}
```

Las siguientes dos funciones son *processRFID* y *blink_LED*. La primera se encarga de procesar la lectura generada por el lector de tarjetas NFC, almacenando su respectivo valor en una variable global que es usada en el resto del código.

La segunda función, *blink_LED* es la encargada de controlar los relevadores y LEDs indicadores. Es invocada por otras funciones con un parámetro booleano, y dependiendo del valor del mismo ejecuta una rutina de cambio de estados en el respectivo conjunto de relevador y LED indicador. Las variables *access_relay_timing* y *block_relay_timing* son clave para la calibración del sistema una vez instalado, puesto que estos tiempos de pulso son importantes para su integración con los sistemas preexistentes.

```
void webRequest(){
  for (int i=0; i<2; i++){
    request =  server_addresses[i] + client_id;
    http.begin(request);                //Specify request destination
    http.setTimeout(1000);              //Waits 1 sec for server response
    int httpCode = http.GET();          //Send the request
    if (httpCode > 0) {                 //Check the returning code
      String payload = http.getString();   //Get the request response payload
      if (payload == "true"){
        http.end();
        blink_led(true);  //Access granted
      }
      else{
        http.end();
        blink_led(false); //Access denied
      }
      break;
    }
    else{
      http.end(); //Server not found, close connection
    }
  }
 return;
}
```

La función aquí presentada, *webRequest*, es la encargada de ejecutar la comunicación al microserver y/o servicio tercero a través de una petición web. En pro de robustecer el funcionamiento del sistema, esta función recorre las direcciones de servidores disponibles, almacenadas en la variable *server_addresses* al principio del código. De esta manera, si el primer servidor no responde, inmediatamente llama al segundo. Dependiendo de la respuesta obtenida por el servidor, se invoca a la función *blink_led*, presentada en párrafos anteriores.

Finalmente, la función *loop* es la que está corriendo todo el tiempo que el dispositivo está encendido. En orden de importancia, se encarga de distintas funcionalidades esenciales para brindar el servicio del dispositivo como parte del sistema; está siempre escuchando por una petición web que le instruya actualizar el firmware, se encarga de asegurar que se cuente siempre con una conexión WiFi y está esperando a procesar el identificador de una tarjeta NFC en caso de que esta se presente en el lector de tarjetas. Las dos sentencias condicionales de la detección de tarjetas son necesarias para elegir sólo una en caso de que se presenten varias en el lector.

La instrucción *return* dentro de estos condicionales evita que las funciones *processRFID* y *webRequest* sean invocadas constantemente. Éstas son invocadas únicamente si se acerca una tarjeta NFC al lector y su identificador es interpretado.

```
void loop(){
 httpServer.handleClient();
 MDNS.update();
 //Check WiFi connection
 if(WiFi.status() != WL_CONNECTED){
  connectWiFi();
 }

 //RFID Card detection
 if(!mfrc522.PICC_IsNewCardPresent()){   //Look for cards
  return;
 }
 if(!mfrc522.PICC_ReadCardSerial()){     //Select one of the cards
  return;
 }

 processRFID();
 webRequest();
 }
```

La suma de los bloques de código presentados en esta sección integra el código embebido en el NodeMCU y que controla todas las funcionalidades del dispositivo en su interacción con el sistema.

3.5 COSTOS Y MODELO DE NEGOCIO

Esta sección revisará los costos del desarrollo y el modelo de negocio asociado para la rentabilidad del proyecto.

En la sección de "objetivos específicos" se plantea que uno de ellos es evaluar las mejores prácticas para buscar la comercialización de producto bajo licenciamiento abierto. El desarrollo de esta sección trabaja sobre este tema.

De igual manera, en la sección de "perspectiva de negocio" se menciona el *widget frosting*, pero este término pasa a segundo plano. En su lugar entra el servicio de

consultoría especializada para el diseño e integración de la solución tecnológica que representa el proyecto aquí desarrollado.

3.5.1 Modelo de negocio

Cuando se desarrolla un nuevo negocio, siempre es importante identificar el nicho de mercado al que queremos ofrecer nuestra propuesta de valor, sin embargo, lo que importa realmente es tener la solución a la necesidad del cliente, a su tarea u obtener un valor agregado que el mismo previamente haya manifestado [62].

En el caso de este proyecto se contó con la apertura de la empresa cliente para que el autor tuviera una ventana de tiempo suficiente para la etapa de diagnóstico e investigación, redactada al principio de este documento. Así fue entonces que se logró desarrollar una solución muy específica a su necesidad. Es importante mencionar que se diseñó pensando en la modularidad del sistema para tener la flexibilidad necesaria en caso de requerir modificaciones posteriores.

Al inicio del desarrollo del proyecto se consideraba que la principal fuente de ingreso sería la venta de activos, pero no fue así. Como se presentará en las secciones siguientes, el proyecto pivoteó a una consultoría integral. Es por esto que se omite el análisis de un modelo de negocio, o *business model canvas*, debido a que no se trabajó en el diseño de un emprendimiento, sino en un proyecto de consultoría.

Bajo este mismo paradigma, es importante mencionar que, si bien este proyecto no es un desarrollo que produzca grandes volúmenes para consumidores finales, como un commodity, el cliente final ha sido una empresa con un importante presupuesto destinado a la mejora de sus instalaciones. Este indica que el canal de ventas resultó funcionar bajo un esquema B2B (*Business to business*), donde con base en la plataforma tecnológica desarrollada se pueden ofrecer soluciones similares a pocos clientes, o empresas, pero que representan un importante ingreso.

Recordemos que todo este proyecto ha sido desarrollado bajo una ideología *Open Source*, por lo que vale la pena mencionar un modelo de negocio [23] conocido como *Hardware Integration Model*. En este modelo, las compañías de hardware integran software *open source* en su producto. El software es gratuito, el cliente únicamente compra el dispositivo que lo ejecuta, por ejemplo, los teléfonos Android. Esto permite a los desarrolladores de hardware reducir significativamente los costos de sus productos.

El modelo *Hardware Integration Model* es mencionado en esta sección debido a que presenta una referencia importante para la comercialización de este proyecto, puesto que en el acto de venta el cliente adquiere el hardware desarrollado por el autor (el dispositivo) y únicamente se integra el firmware que está liberado en la comunidad como código abierto.

Existe otra vertical interesante en los modelos de negocio *Open Source*; el desarrollo como consultor independiente. Surge a partir de que un creciente número de desarrolladores en la comunidad ofrecen sus servicios para desarrollar, instalar, mantener o mejorar software libre. Este proyecto tuvo un pivoteo interesante con respecto a su planteamiento inicial; si bien se imaginaba como el desarrollo de un producto o servicio de una empresa en nacimiento, la realidad es que se convirtió en un entrenamiento patrocinado donde el autor desarrolló habilidades necesarias para poder ofrecer al día en que estas palabras están siendo redactadas servicios de consultoría. Estas habilidades incluyen estimación de tiempos, costos y ventas, entre otras. Mismas que serán presentadas en adelante.

Al analizar la implementación de *open source* en los negocios se identifican ventajas muy características para la rentabilidad de los mismos. Entre ellas podemos encontrar:

- Flexibilidad
    - Existe una amplia variedad de componentes compatibles entre sí, tanto en tarjetas electrónicas como en lenguajes de programación. En este punto del

documento podemos constatar que se ha tomado ventaja de esta flexibilidad en el diseño e integración del proyecto; se han identificado componentes electrónicos compatibles y se han empleado distintos lenguajes de programación.

- Fiabilidad
    - Existe una comunidad global que soporta el desarrollo de estas plataformas y herramientas, por lo que al momento de tener complicaciones en la conexión o programación de la solución se puede recurrir a las comunidades en la red que colaboran para resolver problemas similares. En estos espacios se pueden encontrar soluciones confiables en una cantidad mínima de tiempo.

- Costo
    - Cuando se usa *open source* para el negocio, se paga únicamente por el soporte que se necesita. Por ejemplo, cuando no se dispone del tiempo para la curva de aprendizaje necesaria para poner en marcha un sistema. Desde otra perspectiva, la del cliente, se abre una ventana de oportunidad para el desarrollador en el momento que se requiere consultoría especializada.

3.5.2 Costos

El tema de los costos fue uno de los más complicados durante el desarrollo de este proyecto, simplemente porque al involucrar desarrollo de hardware la inversión es significativamente más alta al compararla con un desarrollo que involucre exclusivamente software. Recordemos también que la calidad de un prototipo físico está directamente relacionada con la inversión que se le destine.

En el caso del desarrollo de este proyecto, la inversión en materiales fue de $4,784.00 MXN, que incluyen:

- Manufactura de 10 tarjetas PCB

- Componentes electrónicos
  - Relevadores
  - LEDs
  - DIP switches
  - Terminales de conexión
  - Reguladores de voltaje con disipadores de calor
  - Circuitos integrados ULN2003
  - Barras de pines
  - Lectores RFID-RC522
  - Tarjetas NodeMCU

No se está considerando el costo asociado a las horas hombre dedicadas al trabajo de este proyecto, puesto que son las mismas dedicadas a la construcción del proyecto de investigación del que es objeto este documento. El autor prefiere ver este tiempo como un entrenamiento patrocinado en donde se logran importantes avances profesionales.

De manera adicional, se buscó asesoría de parte de profesionales de la industria con mucha más experiencia en el desarrollo de proyectos como este, a quienes se les agradece y por confidencialidad no se mencionarán explícitamente.

Una de estas consultas fue relacionada con la fabricación de las 10 tarjetas PCB en una empresa local. Esta cotización se hizo por motivos de premura de tiempo para instalar el sistema, pues se tenía la creencia que sería más rápido que mandar a manufacturar las tarjetas a China. La cotización con la empresa local fue de $ 2,650.00 MXN por las 10 tarjetas PCB, con un tiempo de entrega de 2 a 3 semanas, el cual no era viable para la planeación de instalación del proyecto ni para los costos. En cambio, la cotización con el fabricante de China fue de $2.00 USD por las 10 piezas, más $17.95 USD por envío rápido. Así fue que se contó con las tarjetas PCB en menos de una semana, listas para ser ensambladas e instaladas.

Uno de los temas que mayor complicación generó en el desarrollo del proyecto fue la incertidumbre, por falta de experiencia del autor, de cuánto cobrar por el proyecto.

Para resolver esto se hizo una extensa investigación, dentro de la cual una de las principales aportaciones fue por parte de uno de los profesores del programa de la maestría para la que se está desarrollando este proyecto. Esta consulta estableció una referencia en la que un proyecto similar se valuaba en un aproximado de $40,000.00 MXN. Este proyecto era menos elaborado que el del autor; incluía además del microcontrolador, un sensor de humedad y temperatura, una pantalla LCD y una salida.

Esta referencia sirvió como base para hacer cálculos de honorarios "a la inversa" para estimar el valor de la ingeniería aplicada al proyecto. Sin embargo, en aras de poder generar los argumentos suficientes para establecer una negociación con el cliente, se hizo una investigación más profunda redactada en el apartado siguiente.

Como el lector se habrá dado cuenta en este punto, el proyecto se "vendió" antes de acordar un precio. Es por esto que se agradece la disposición del cliente a colaborar de esta manera inversa y por brindar la confianza y el espacio para experimentar en una aplicación real.

3.5.3 Compensación económica

Al tratarse del cálculo de la compensación económica a ser requerida al cliente por los productos y servicios entregados no se escatimó en realizar una profunda investigación de los estándares de la industria. Por esto, se tomó como referencia el estudio de salarios 2019 de Software Gurú [63], el cual representa la undécima edición del mismo y cuenta con el respaldo de instituciones como la AMITI (Asociación Mexicana de la Industria de Tecnologías de la Información), la CANIETI (Cámara Nacional de la Industria Electrónica, de Telecomunicaciones y Tecnologías de la Información), entre otras. Si bien el estudio hace referencia a la industria del software, presenta utilidad para los objetos del presente análisis dado a que al menos 50% de la funcionalidad del proyecto depende de la programación de software y firmware para soportar los productos y servicios entregados.

Este estudio obtiene sus indicadores a partir de una encuesta nacional, y sus cifras representan salarios brutos mensuales. Las cifras tomadas como referencia para este documento refieren la media de ingreso para una experiencia de 4.5 años, lo que se considera como un nivel *middle* en la industria del software. La media nacional se encuentra en un ingreso bruto mensual de $37,908.00 MXN. Guadalajara tiene una media más alta en comparación con la nacional, 11.75% superior, lo que la posiciona en un ingreso bruto mensual de $42,365.00 MXN.

Con base en la experiencia del autor al momento del diseño e integración del proyecto se consideró una compensación económica correspondiente a un nivel *junior*, que equivale a una experiencia de máximo 3 años y es 33% menor a la de un *middle,* por lo que en los párrafos siguientes se presenta el desglose de compensaciones medias en la industria a nivel nacional y para Guadalajara, correspondientes a los dos lenguajes de programación usados en la construcción del proyecto, así como para los roles ejecutados en las actividades complementarias que fueron necesarias para la exitosa implementación del mismo.

Para Python, la media nacional se encuentra en $39,918.00 MXN. El ajuste para Guadalajara resulta en $44,608.00 MXN, y el ajuste para el nivel *junior* en la misma ciudad es de $29,887.00 MXN.

Para C++, la media nacional es de $32,003.00 MXN. El ajuste para Guadalajara resulta en $35,763.00 MXN y el ajuste para el nivel *junior* en la misma ciudad es de $23,961.00 MXN.

Con estas cifras que representan el ingreso bruto mensual, ahora es posible calcular el costo por hora, dato necesario para cálculos siguientes. Se toma en cuenta un total de 21.72 días laborales por mes, y un total de 8 horas por día. Haciendo los cálculos correspondientes, se obtiene que:

- El costo por hora de desarrollo en Python para un nivel *junior* en Guadalajara es de $172.00 MXN.
- El costo por hora de desarrollo en C++ para un nivel *junior* en Guadalajara es de $137.90 MXN.

Estas cifras de costo por hora serán empleadas más adelante cuando se presente la cotización final.

Conforme se fue avanzando en el desarrollo del proyecto, generando aprendizajes a pesar de los imprevistos y progresos, el autor se fue involucrando en actividades "secundarias" necesarias para la realización del proyecto. Por lo anterior, adicionalmente se hizo una revisión de los salarios en el mismo estudio que no eran propias del desarrollo de software, sino de roles administrativos. Se sigue la misma lógica de análisis que en los párrafos anteriores, y se presentan los salarios brutos mensuales por rol, media nacional y para Guadalajara:

- Consultoría de negocio $48,957.00 MXN
- Arquitectura de soluciones $43,697.00 MXN
- Administración de proyectos $44,400.00 MXN
- Desarrollo *back-end* $35,309.00 MXN

Y al hacer los cálculos correspondientes promedio de los 4 roles, respecto al aumento para Guadalajara y el cálculo por hora, ajustado a nivel *junior* se obtuvo que la hora de consultoría que incluía las actividades mencionadas en la lista anterior tenía un costo de $185.96.00 MXN. Nuevamente, esta cifra es considerada como referencia en la cotización final presentada al cliente.

Antes de pasar a otro tema, este párrafo hará un resumen de los servicios incluidos hasta este punto en el proyecto:

- Servicios relacionados al hardware:
  - Levantamiento de requerimientos
  - Investigación y desarrollo
  - Diseño y construcción de prototipo
  - Diseño de circuito electrónico
  - Diseño de PCB
  - Ensamble de componentes
  - Pruebas de funcionamiento

- Servicios relacionados al software:
  - Programación de firmware
  - Programación de microserver

La lista aquí presentada resume y conceptualiza las actividades desarrolladas que produjeron el trabajo presentado en este documento para la realización del proyecto.

Como se ha mencionado ya, los productos ofrecidos resultado del proyecto se complementaron con un catálogo de servicios de consultoría necesarios. Ahora es importante mencionar que el trabajo de consultoría se extendió para solucionar una problemática del cliente no contemplada en los inicios del trabajo. El cliente requirió además contar con tarjetas de identificación para los usuarios de sus instalaciones, mismas que trabajarían con los dispositivos instalados en los controles de acceso instalados. Se trata de tarjetas de plástico con tecnología NFC, personalizadas con una impresión en un formato estandarizado que incluyera el nombre y fotografía de usuarios y personal.

Este nuevo requerimiento demandó que el autor iniciara un trabajo de investigación de mercado, cotización, selección de producto, pruebas y configuración del equipo necesario; impresora especializada para credenciales. Dentro de las actividades derivadas de este requerimiento se encontraron, por mencionar las principales:

- Cotización de impresora
- Diseño de las impresiones
- Pruebas de calidad de impresión buscando economizar consumibles
- Capacitación del personal
- Instalación del sistema
- Configuración de herramientas para asistencia remota

Si bien estas actividades no demandaron un desarrollo de software ni diseño de hardware, sí demandaron tiempo del autor y el mismo fue dedicado por tener clara la alta importancia que el cliente le relacionaba. En temas de costos se consultó en las mismas asesorías previamente mencionadas y se encontró que bajo este tipo de actividad la norma es cobrar un porcentaje del costo total del equipo, oscilando entre el 3% y el 10%. En la cotización final, presentada más adelante, se incluye este concepto con un porcentaje del 10%.

3.5.4 Cálculos finales

Una vez presentados en las páginas anteriores los fundamentos en los que se basa el autor para emitir una cotización argumentada, se presenta lo que fue una estimación de horas por actividad realizada. Esta estimación se genera en un momento en el que las curvas de aprendizaje ya han sido superadas y por lo tanto se cuenta con la experiencia necesaria para recrear el proyecto:

- Actividades relacionadas al hardware
    - Diseño esquemático 4 horas
    - Diseño de PCB 8 horas

- Ensamble de componentes 5 horas
- Programación de firmware 8 horas
- Pruebas y validación 4 horas
- Investigación y prototipado 20 horas

Generando un total de 49 horas, cada una valuada en $350.00 MXN, lo que representa un monto de $17,150.00 MXN.

- Actividades relacionadas al software
  - Configuración de sistema operativo 4 horas
  - Programación de software 8 horas
  - Pruebas y validación 4 horas
  - Integración de hardware 2 horas

Lo que suma 18 horas de trabajo, valuadas en $220.00 MXN cada una, que genera un monto de $3,960.00 MXN.

Además de estas actividades de carácter técnico, se considera la comisión por la instalación del sistema de impresión de credenciales y los honorarios por las horas de instalación, que se presentan a continuación en la cotización final entregada al cliente.

Concepto	Horas	Costo / hora	Monto
Desarrollo de hardware, firmware y software: 5 dispositivos de lectura NFC programados con capacidad de actualización remota para soporte, servidor local con aprendizaje de accesos como respaldo en caso de falla de conexión a internet y con acceso remoto para soporte	120	$350.00	$42,000.00
Credencialización: Búsqueda, cotización y comparación técnica de impresora, tarjetas y consumibles. Diseños para alumnos y empleados con base en requerimientos de contenido y ahorro de consumibles. Automatización del proceso de generación de credenciales y asignación de NFC ID a CRM. Instalación con lector adicional de refacción y capacitación a personal. (10% de comisión sobre costo de impresora y tarjetas)	N/A	N/A	$3,935.00
Instalación: Configuración, instalación y pruebas de funcionamiento in situ (Incluyendo seguimiento post-instalación y ajustes para optimizar funcionamiento)	24	$350.00	$8,400.00
	Monto Total		$54,335.00
	Descuento	35.00%	$19,017.25
	Subtotal		$35,317.75
	IVA		$5,650.84
	Total		$40,968.59

Figura 32: Cotización final entregada al cliente. Fuente: Elaboración propia

Nótese que, si bien las actividades inherentes al hardware y software suman apenas 67 horas, en la cotización se indica que fueron 120 horas. Esto se debe a dos razones; es una práctica común en la industria de la consultoría derivada de la necesidad de tener un *buffer* de respaldo en caso de imprevistos, y a que el presupuesto de la empresa cliente es significativamente mayor al de un cliente final estándar. Además, el valor que le da la empresa a esta solución no reside en hardware instalado, sino en el alivio de un cuello de botella que genera una experiencia de usuario negativa en sus instalaciones.

Otro punto importante a mencionar es el costo por hora, expresado en $350.00 MXN. Si bien el ejercicio de análisis de salarios en la industria arrojó cantidades menores tanto para software como para hardware, aquí no se hace esta distinción categórica y se homogeniza bajo el concepto de horas de desarrollo. Esto resulta de las mismas asesorías por especialistas ya mencionadas, quienes cotizan la hora de desarrollo incluso hasta en $500.00 MXN.

El concepto de credencialización se cobra con un porcentaje del 10% sobre el costo total del equipo instalado, y en el mismo se detallan las actividades desarrolladas.

Finalmente, en la tabla se menciona el concepto de instalación que hace referencia a los honorarios por las horas invertidas en la instalación del sistema en el edificio de la empresa. Su respectivo monto busca cubrir gastos de viáticos dado que fue necesario trasladarse a otro estado de la república.

Se puede observar también que se menciona un descuento del 35% sobre el total de los precios. Este tiene un propósito meramente de estrategia mercadológica para generar empatía con el cliente, argumentado como descuento por el extenso periodo de tiempo necesario para entregar la solución y como agradecimiento por la oportunidad de desarrollo y aprendizaje del autor para la materialización del proyecto. También tiene como objetivo compensar el alza en los precios por hora de trabajo, que se relaciona de nuevo con estrategias de venta, específicamente con aquella que dicta que el mercado atribuye una mayor calidad de producto a aquellos con un precio mayor.

Con el objetivo de que el autor estuviese preparado para una negociación cuando se entregara dicha cotización, se hizo una investigación de mercado de soluciones similares a las desarrolladas en este proyecto. Se encontró que el costo de la solución cotizada con un desarrollador especializado en la industria es de $60,000.00 MXN sin considerar los conceptos de instalación ni credencialización, y que el costo de la solución basada en productos comerciales, sin incluir la credencialización, instalación ni conexión al sistema ERP preexistente, es de $36,809.00.

Finalmente, con base en el análisis hecho, las asesorías y las estrategias de venta mencionadas, se logró que la empresa cliente aceptara la cotización, resultando en un primer ingreso en la carrera profesional del autor por un proyecto de consultoría.

En el caso de los componentes de hardware utilizados, principalmente la NodeMCU, debe mencionarse que no se están tomando los diseños del mismo como base para generar un nuevo componente, sino que se está diseñando un hardware desde cero en donde se montará el mismo sin sufrir modificación alguna. A modo de referencia es importante mencionar que el proyecto NodeMCU se encuentra bajo la licencia *MIT* [64], la cual es una licencia permisiva que sólo requiere la mención de la misma y preservación de *copyright*. Los trabajos bajo esta licencia, así como modificaciones y desarrollos extensivos pueden ser distribuidos bajo distintos términos y sin el código fuente. Permite la comercialización, modificación, distribución y uso privativo.

El proyecto aquí desarrollado es liberado bajo la licencia GNU *Lesser General Public License*, quedando abierto y a disposición de la comunidad en el repositorio de GitHub previamente mencionado. Esta decisión es hecha por el autor buscando retribuir a la comunidad. Sin embargo, en el caso de tener futuras implementaciones o ventas del mismo, a los clientes les será entregado únicamente el hardware, buscando así fomentar el modelo de negocio basado en la consultoría especializada y así poder monetizar futuras implementaciones.

Finalmente, uno de los principales incentivos para el desarrollo de este proyecto en sus inicios fue la creencia de que se podría generar un ingreso atractivo para el autor, aunque en realidad no fue así. Fue más valioso el ya mencionado entrenamiento patrocinado y la experiencia adquirida para el desarrollo de proyectos de esta naturaleza, lo cual abre importantes ventanas de oportunidad para el propio desarrollo profesional.

CAPÍTULO 4: RESULTADOS

Es momento de hacer un análisis de los resultados obtenidos después de la gran cantidad de trabajo realizado, las inversiones de tiempo hechas y las adaptaciones que las circunstancias han demandado en el desarrollo de este proyecto. Se presentan entonces en los siguientes párrafos esos logros y lecciones aprendidas durante el desarrollo de lo redactado en el presente documento.

A través de la integración de distintas tecnologías, con base en una variedad de herramientas, se ha logrado la entera satisfacción del autor al lograr poner en marcha una solución tecnológica funcional y que ha sido diseñada desde los niveles iniciales de la escala TRL. Así mismo, la empresa cliente ha manifestado su satisfacción con el mismo proyecto, puesto que ha resuelto una problemática que se tenía y ha aumentado la eficiencia de algunos de sus principales procesos de negocio.

El trabajo aquí presentado es liberado a la comunidad *open source* para que pueda ser utilizado por los colegas desarrolladores como una plataforma para desarrollos futuros si es que en determinado momento se identifican oportunidades de aplicación e incluso oportunidades de negocio. El fin último que se busca con esto es poder ver que nuevas soluciones se materialicen a partir del trabajo aquí desarrollado, y que estas nuevas soluciones sean más sofisticadas que el mismo proyecto de origen aquí liberado.

Se logró que la solución instalada cuente con un grado de robustez para evitar cuellos de botella en los procesos de negocio de la empresa y experiencias negativas en los usuarios. Esto gracias a dos funcionalidades específicas; el aprendizaje de registro de usuarios y a la capacidad de programación *OTA*. La primera asegura que el sistema siga cumpliendo con su tarea en caso de falla de conexión a internet, y la segunda permite el soporte remoto para ejecutar reconfiguraciones físicas y de conexión con los sistemas preexistentes. Estas características fueron resultado de la premisa constante y siempre presente por el autor de generar cierto grado de innovación tecnológica aplicada, que

en el mismo sentido brinda características diferenciadoras al proyecto contra soluciones comerciales.

No se podría hablar de resultados exitosos sin mencionar las lecciones aprendidas durante el proceso. La más importante en el desarrollo de este trabajo de investigación fue aquella en la que se presentaron fallas graves a nivel sistemático la primera vez que se montaron los componentes sobre una tarjeta PCB para poner en marcha el dispositivo. El haber omitido las pruebas de prototipo en la etapa de control de relevadores generó un calentamiento excesivo en el regulador de voltaje que alimenta todo el sistema, además de un funcionamiento totalmente erróneo del dispositivo. Aquí la lección fue no omitir procesos de prototipado antes de manufacturar componentes y no hacer compras sin validación previa. Lo importante aquí es aprovechar este suceso para resaltar la importancia de seguir metodologías de investigación y desarrollo, como las presentadas en el marco teórico, para evitar dar pasos en falso y perder tiempo y recursos.

Aún con todo el trabajo concretado y las lecciones aprendidas, se logró evitar caer en el conocido *project scope* o pérdida de objetividad durante el desarrollo del proyecto. Si bien es cierto que conforme el autor avanzaba en las investigaciones y descubrimiento de tecnologías y herramientas disponibles para sofisticar el trabajo de investigación, esta actitud de alerta y enfoque fue esencial al momento de tomar decisiones sobre el rumbo del trabajo de investigación. Al mismo tiempo, el hecho de tener la disposición por desarrollar aptitudes de negocio durante el proceso ha reforzado la importancia de mantener el enfoque no únicamente en los objetivos planteados, sino en cómo lograr estos objetivos cubriendo una necesidad del mercado.

Respecto al seguimiento de metodologías, como se ha mencionado ya, se tuvieron en el radar a dos principales; Lean Startup [60] y la de desarrollo de proyectos de hardware [26]. Si bien la primera aporta el componente de la importancia de la validación rápida para confirmar suposiciones, tanto de usabilidad como técnicas, la segunda aporta un

marco de referencia para ejecutar una correcta administración del proyecto. Sin embargo, no se debe pasar por alto el componente más importante de este trabajo de investigación; la escala *TRL*. Esta última aporta el diferenciador más relevante al proyecto, puesto que el avance progresivo en sus niveles ha considerado en el caso de este trabajo, no únicamente el componente científico, sino que se ha incorporado otro componente en temas de negocios y comercialización.

Bajo el mismo tenor de lo comercial y los negocios, se puede confirmar que este proyecto tiene fronteras abiertas en caso de querer ser implementado en otros países debido a dos razones; su núcleo, el ESP8266, cuenta con las regulaciones de radiofrecuencia ya mencionadas [32] y a que está liberado en la comunidad bajo una licencia permisiva [65].

Finalmente, pero no de menor importancia, uno de los grandes resultados es el irrefutable hecho de que el grado de especialización que el autor ha adquirido al desarrollar este proyecto ha mejorado significativamente, pues brinda la experiencia suficiente para futuros desarrollos de base tecnológica con potencial de comercialización, cumpliendo así la justificación presentada al inicio de este documento.

CONCLUSIONES

Este capítulo está dedicado a mencionar los hallazgos más importantes a lo largo del proceso de investigación que conllevó este proyecto, y a dar contexto a las conclusiones que serán presentadas bajo cuatro pilares; cómo es que se alcanzaron los objetivos, el cumplimiento de la hipótesis presentada, cómo los resultados se relacionan en comparación con las investigaciones del marco teórico, y cómo este proyecto destaca ante lo desarrollado previamente.

El hecho de haber tenido como referencia una metodología para el desarrollo de proyectos de hardware es fundamental para llevar un seguimiento sistematizado al avance que los desarrollos técnicos, desde las tareas más sencillas hasta la integración de componentes, fueron agregando a las etapas del proyecto, con la mira siempre en el cumplimiento de los objetivos. En suma, a esto, el marco de referencia aportado por el modelo *TRL* brindó la pauta de investigación científica y guía para un avance en los niveles del mismo, teniendo siempre en cuenta el progreso en paralelo de la estrategia comercial en sus indicadores. Es así entonces como se logran alcanzar satisfactoriamente los objetivos planteados.

Al retomar lo contenido en el marco teórico es evidente notar que, así como los trabajos ahí presentados, este trabajo de investigación tiene la capacidad de convertirse en un producto comercial, y aún más importante es la propia característica de que al estar basado en investigación científica formal, la probabilidad de éxito comercial aumenta debido a que su factibilidad técnica ha quedado no sólo demostrada, sino documentada desde sus diseños hasta su proceso de configuración.

En lo referente a la hipótesis, recordemos lo que textualmente dice: "*La investigación científica aplicada formalmente para solucionar la problemática identificada lleva a seguir un proceso de desarrollo de un producto de base tecnológica eficiente, y esta rigurosidad abre la posibilidad de desarrollar nuevos productos comerciales*". Es posible

entonces identificar tres elementos clave; la investigación científica aplicada, el desarrollo de un producto de base tecnológica, y la posibilidad de desarrollo comercial. Estos tres elementos quedan confirmados en la presentación de resultados y se concluye que el seguir una metodología como el método científico puede generar innovaciones, pero si se le suma una rigurosa búsqueda de aplicación industrial es posible lograr desarrollos tecnológicos comerciales. Lo anterior hace referencia a una conclusión en un nivel generalista, sin embargo, basta con revisar detalladamente los logros alcanzados en los resultados para reafirmar que el desarrollo de este proyecto es un claro ejemplo de cómo la hipótesis es cumplida.

El alcance de dicha meta trae consigo efectos colaterales en un sentido positivo, a partir del hecho de que además del trabajo esencial de desarrollo científico en desarrollo de proyectos como el de este trabajo de investigación, sienta las bases para el desarrollo de habilidades inherentes al arte de identificar oportunidades de negocio y diseñar soluciones desde un expertise tecnológico. Para alcanzar lo anterior, ha quedado claro que es importante conocer la industria y su comportamiento, especialmente los costos asociados para ser capaz de participar atractivamente con proyectos de consultoría y el diseño de soluciones de base tecnológica, las cuales siempre deben ser diseñadas pensando en la capacidad de respuesta a cambios no previstos en temas de instalación en sistemas preexistentes. Lo anterior puede ser alcanzado de una manera rápida, flexible y segura cuando las capacidades técnicas del profesional se enfocan en el desarrollo con base en tecnologías open source.

Adicionalmente, la experiencia adquirida durante este trabajo no marca el final de un proceso, sino que abre un nuevo reto que deberá ser bien recibido y explorado como oportunidad de crecimiento profesional. Este reto se refiere a la obligación que adquiere el autor para alcanzar niveles de desarrollo aún más sofisticados. Si bien ha quedado claro que se puede desarrollar un proyecto como este sin seguir una metodología ni documentación formales, el seguir el camino científico aporta ese carácter de

formalidad necesaria tanto para poder ser compartido y replicado por la comunidad académica, como para tomarse de referencia en futuras investigaciones.

Para cerrar esta sección con broche de oro es preciso destacar la importancia que tuvo el proceso formativo y los conocimientos adquiridos durante los cursos del programa de posgrado para el que se realiza este trabajo. Sentaron las bases imprescindibles para poder interpretar los resultados en cada paso del desarrollo del proyecto, además de fijar un rumbo basado en el conocimiento para determinar la herramienta adecuada en cada escenario. Cada una de las tareas y colaboraciones, cada proyecto final y cada asesoría por parte de los profesores forman parte de los cimientos que sostienen el trabajo aquí presentado.

Es importante mencionar algunos de estos cursos, sin pretender que los omitidos no sean relevantes. Gracias a -arquitectura de microprocesadores- se discriminó el hardware que no prestaba las funciones óptimas y suficientes, -programación en lenguaje ensamblador- consolidó las habilidades de programación fundamental, -compiladores y desarrollo de librerías- determinó el uso de librerías y el diseño estructural del código tanto en el embebido como en el microserver. La curiosidad por las tecnologías de RFID y NFC comenzó en -sistemas operativos en tiempo real- cuando gracias a las prácticas se logró diferenciar ambas opciones y seleccionar la adecuada para el caso de aplicación del proyecto. La infraestructura de red, la topología de los dispositivos, y los protocolos de comunicación seleccionados obtuvieron su justificación del curso de -protocolos de comunicación industriales y de redes-. Al momento de traducir los casos de uso del personal de la empresa con el sistema, las habilidades de abstracción aprendidas tanto en -ingeniería de software- como en -especificación de requerimientos- y -prueba y validación- fueron clave para que todo el proceso de codificación, prueba y despliegue del sistema fluyera de una manera ágil y exitosa.

Nótese entonces cómo la materialización de este proyecto no hubiera sido posible sin constantemente refinar la idea inicial durante cada uno de los cursos del programa, ni

tampoco habría sido posible el desarrollo de habilidades de negocio y consultoría sin las interacciones con los profesores y compañeros.

Conclusiones sobre costos y modelo de negocio

Dado que además de los aprendizajes y logros técnicos obtenidos en el desarrollo de este trabajo de investigación la experiencia en la cotización de proyectos como este y la adquisición de habilidades necesarias para ofrecer servicios de consultoría fueron de especial importancia, se dedica esta sección para destacar puntos importantes a manera de conclusiones relacionadas a los costos y modelo de negocio.

Se confirmaron importantes ventajas de usar *Open Source* para el desarrollo de negocios, como la significativa reducción de costos al adquirir el equipamiento tecnológico necesario para desarrollar este tipo de soluciones. Sin embargo, es claro que la curva de aprendizaje es mucho mayor cuando se opta por usar bases de este tipo, pero deja enorme satisfacción y el límite para la sofisticación de las soluciones creadas reside únicamente en el compromiso e inversión de tiempo que el profesional decida dedicar. Aunado a esto, una vez desarrollado el primer producto es mucho más sencillo atender proyectos posteriores dado que se tiene una base técnica importante ya desarrollada.

Es importante tocar el tema de licenciamiento, elemento clave en el ecosistema *open source*. Para el desarrollo de este proyecto se usan distintas herramientas tecnológicas, mas no se están usando desarrollos previos en cuyo caso, dependiendo de la licencia, se demandaría redistribuir el nuevo desarrollo con respectivo reconocimiento al desarrollo precursor. En el caso de las librerías implementadas en el software desarrollado, todas se encuentran bajo la licencia GNU *Lesser General Public License* [58] en su versión 2.1. Esta licencia requiere que los trabajos derivados contengan la misma licencia, pero los trabajos que sólo refieren al desarrollo bajo la misma no caen

en esta restricción, como es el caso del proyecto aquí desarrollado. Así mismo, esta licencia permite la comercialización, modificación, distribución y uso privativo.

APORTACIÓN DE LA TESIS

La presente tesis cuenta con diversos elementos que agregan valor tanto para el autor como para la comunidad, como la comprobación de que con base en los conocimientos adquiridos en el programa de posgrado y el seguimiento de una combinación de metodologías de desarrollo de proyectos es posible generar soluciones tecnológicas bajo un proceso de avance técnico progresivo al mismo tiempo que se comprueba si las funciones diseñadas del mismo son de utilidad para la aplicación final por medio de experimentos rápidos. Se demuestra cómo esta formalidad aumenta las posibilidades de comercialización de proyectos y se propone una metodología para la valuación de soluciones tecnológicas con base en el desarrollo de habilidades de consultoría tecnológica profesional. El proyecto es liberado en su totalidad bajo licenciamiento abierto para su implementación en futuras oportunidades.

APORTACIÓN SOCIAL DE LA TESIS

El trabajo aquí presentado lleva implícita una amplia ventana de aportación a la sociedad desde su principio fundamental del uso de tecnología *open-source*. Al estar liberado en la red a través de un repositorio público y con licenciamiento abierto puede ser utilizado por cualquier individuo con las habilidades técnicas suficientes para implementar su funcionalidad para la que fue originalmente desarrollado. La interpretación del proceso de diseño, validación, desarrollo y configuración está explícitamente detallada en el contenido del presente documento para poder ser recreado. Adicionalmente, al contar con todos los archivos fuente tanto del *software* como del *hardware* es posible hacer modificaciones si son necesarias para su adaptación a otras funciones específicas. En términos simples, acerca un desarrollo tecnológico a los profesionistas de industrias tecnológicas para generar nuevas soluciones. Una de las principales ventajas es que en cuanto a términos de inversión monetaria es significativamente menor a la de un proveedor con tecnología privativa, estableciendo entonces la posibilidad de poder ser aprovechado por pequeñas empresas en proceso de crecimiento para integrarse a su estrategia de transformación digital. Finalmente, se establece como un punto de partida para la exploración de su combinación con otras áreas del conocimiento donde este proyecto funcione como plataforma para profundizar en investigaciones sobre nuevas aplicaciones, tanto de utilidad industrial como académicas.

RECOMENDACIONES

Esta sección aborda recomendaciones del autor sobre tres ejes; qué se debe hacer con los resultados y dónde aplicarlos, trabajos futuros, y deseos surgidos para desarrollar nuevos estudios de este tipo. En el primero de estos tenores, ha de recordarse que el proyecto se encuentra disponible para ser usado como plataforma en cualquier aplicación donde se le encuentre utilidad. En este mismo escenario, es importante que si se requiere hacer modificaciones al diseño del hardware se deben completar pruebas en niveles básicos de prototipado antes de fabricar cualquier componente o tarjeta de circuito impreso. Lo importante es pivotear rápidamente para avanzar y no estancar la implementación de nuevas aplicaciones.

Respecto a trabajos futuros, se está pensando ya en una versión 3.0 del dispositivo (recordemos que la versión aquí presentada fue la segunda iteración del mismo). Este nuevo diseño estará pensado en la escalabilidad y miniaturización , con distintas características que le abonarán sofisticación y robustez, como la integración de la antena para NFC en el mismo PCB y el uso de componentes de soldadura superficial. Esta nueva versión buscará idealmente usar la actualización del dispositivo central de procesamiento; el ESP8266 deberá quedar descartado para usar el ESP32, que integra comunicación Bluetooth y una capacidad mayor de procesamiento [66].

Esta versión 3.0 incluye también una nueva arquitectura donde se prescinde de un microserver, donde los dispositivos actúen como red de nodos y sólo uno de ellos funcione como *gateway* para la conexión a internet. En caso de utilizar un microserver, se pretende contar con una imagen preconfigurada y lista para instalar. Esta misma versión deberá contar con una hoja de referencia técnica para construcción y configuración de las *APIs* necesarias, y se deberá indagar en una técnica de programación más sofisticada para optimizar el uso de recursos. Bajo el mismo contexto, se ha observado en las oportunidades de aplicación que esta nueva versión debería poder desacoplar un lector de NFC para facilidad de instalación discreta.

En el tercer eje, la premisa es la búsqueda de nuevas áreas de aplicación en diversas industrias. Durante el periodo de redacción de este documento el autor ha explorado sistemas de información gerencial (ERP, CRM) basados en software libre. Se ha identificado una nueva y posible línea de investigación en donde la intersección de las disciplinas de administración de negocios y el desarrollo tecnológico, abren un área de oportunidad para desarrollar soluciones innovadoras que sean atractivas para este mercado.

Bajo la misma frecuencia sobre posibles nuevos estudios, se pretende explorar qué penetración tiene el hardware de función específica, como el IoT, en sistemas de administración de negocios, por ejemplo, en manufactura inteligente e industria 4.0, con la mira en una integración con sistemas ERP preexistentes.

REFERENCIAS BIBLIOGRÁFICAS

[1] Frequently Answered [Questions | Open Source Initiative. [online]. [Accessed 18 May 2019]. Available from: https://opensource.org/faq#osd

[2] WILES, Frank. Using Open Source in Business. p. 8.

[3] BEMYAPP. Thinking Open-Source Hardware? Start Here. *BeMyApp* [online]. 24 May 2016. [Accessed 16 May 2019]. Available from: https://medium.com/@bemyapp/thinking-open-source-hardware-start-here-2606debb5092

[4] What is open hardware? *Opensource.com* [online]. [Accessed 16 May 2019]. Available from: https://opensource.com/resources/what-open-hardware

[5] BRUNER, Jon. How the New Hardware Movement is driving the Internet of Things. p. 8.

[6] What is an Arduino? *Opensource.com* [online]. [Accessed 15 May 2019]. Available from: https://opensource.com/resources/what-arduino

[7] NodeMcu -- An open-source firmware based on ESP8266 wifi-soc. [online]. [Accessed 18 May 2019]. Available from: https://www.nodemcu.com/index_en.html

[8] BENTO, D. IoT: NodeMCU 12 e X Arduino Uno, Results of an experimental and comparative survey. 2018.

[9] MONTER, Aitor Carricondo and FRAU, David Cuesta. Desarrollo de un sistema de monitorización domiciliaria basado en la plataforma NodeMCU V3. p. 105.

[10] ELÍAS, Julio Candelario. Implementación de WPS en el firmware NodeMCU para el ESP8266. p. 85.

[11] JAFFE, Samuel. Masters of Science in Electrical Engineering. p. 138.

[12] SHKURTI, Lamir, BAJRAMI, Xhevahir, CANHASI, Ercan, LIMANI, Besim, KRRABAJ, Samedin and HULAJ, Astrit. Development of ambient environmental monitoring system through wireless sensor network (WSN) using NodeMCU and "WSN monitoring." In : *2017 6th Mediterranean Conference on Embedded Computing (MECO)* [online]. Bar, Montenegro : IEEE, June 2017. p. 1–5. [Accessed 7 February 2019]. ISBN 978-1-5090-6742-8. Available from: http://ieeexplore.ieee.org/document/7977235/

[13] CALDERON-CORDOVA, Carlos, GONZAGA, Miguel, MORALES, Jose, MOROCHO, Miguel, TORRES, Bryan and RAMIREZ, Cristian. Prototype industrial IoT applied to temperature monitoring in storage silos of dairy products. In: *2018 13th Iberian Conference*

on Information Systems and Technologies (CISTI) [online]. Caceres: IEEE, June 2018. p. 1–6. [Accessed 7 February 2019]. ISBN 978-989-98434-8-6. Available from: https://ieeexplore.ieee.org/document/8399299/

[14] What is LabVIEW? - National Instruments. [online]. [Accessed 18 May 2019]. Available from: http://www.ni.com/en-us/shop/labview.html

[15] SKRABA, Andrej, KOLOZVARI, Andrej, KOFJAC, Davorin, STOJANOVIC, Radovan, STANOVOV, Vladimir and SEMENKIN, Eugene. Prototype of group heart rate monitoring with NODEMCU ESP8266. In: *2017 6th Mediterranean Conference on Embedded Computing (MECO)* [online]. Bar, Montenegro: IEEE, June 2017. p. 1–4. [Accessed 7 February 2019]. ISBN 978-1-5090-6742-8. Available from: http://ieeexplore.ieee.org/document/7977151/

[16] GORE, Shreya. Review on Programming ESP8266 with Over the Air Programming Capability. 2017. p. 3.

[17] KODALI, Ravikishore and YERROJU, Subbachary. Energy efficient smart street light. In: *2017 3rd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)* [online]. Tumkur: IEEE, December 2017. p. 190–193. [Accessed 7 February 2019]. ISBN 978-1-5386-1144-9. Available from: https://ieeexplore.ieee.org/document/8389131/

[18] Arduino - Environment. [online]. [Accessed 10 May 2019]. Available from: https://www.arduino.cc/en/Guide/Environment

[19] Bolt: Fully Integrated IoT Platform, Made for Machine Learning. *Bolt IoT* [online]. [Accessed 15 May 2019]. Available from: https://store.boltiot.com/

[20] ESP8266 based SmartWIFI Development Module by KNEWRON Technologies on Tindie. *Tindie* [online]. [Accessed 7 May 2019]. Available from: https://www.tindie.com/products/Knewron/esp8266-based-smartwifi-development-module/

[21] zmote: Wi-Fi Universal Remote for your home. [online]. [Accessed 15 May 2019]. Available from: http://www.zmote.io

[22] OpenXC. [online]. [Accessed 15 May 2019]. Available from: http://openxcplatform.com/

[23] Open Source Case for Business:Advocacy | Open Source Initiative. [online]. [Accessed 16 May 2019]. Available from: https://opensource.org/advocacy/case_for_business.php

[24] What is a Raspberry Pi? *Opensource.com* [online]. [Accessed 20 May 2019]. Available from: https://opensource.com/resources/raspberry-pi

[25] CHANDIRAMANI, Jiten. Ten Rules to Take Your Hardware Product from Concept to Market. *Medium* [online]. 20 December 2017. [Accessed 16 May 2019]. Available from: https://medium.com/techmates/ten-rules-to-take-your-hardware-product-from-concept-to-market-f2a8e8d9878e

[26] EINSTEIN, Ben. The Illustrated Guide to Product Development (Part 1: Ideation). p. 14.

[27] MAI, Thuy. Technology Readiness Level. *NASA* [online]. 6 May 2015. [Accessed 10 May 2019]. Available from: http://www.nasa.gov/directorates/heo/scan/engineering/technology/txt_accordion1.html

[28] TRL CONCAYT. [online]. [Accessed 19 September 2019]. Available from: https://www.conacyt.gob.mx/index.php/sni/convocatorias-conacyt/convocatorias-fondos-sectoriales-constituidos/convocatoria-se-conacyt-innovacion-tecnologica/convocatorias-cerradas-se-conacyt-innovacion-tecnologica/convocatoria-se-conacyt-innovacion-tecnologica-2015/9282-anexo-1-niveles-de-maduracion-tecnologica/file

[29] KENTON, Will. Why Early Adopters Matter. *Investopedia* [online]. [Accessed 2 March 2020]. Available from: https://www.investopedia.com/terms/e/early-adopter.asp

[30] NodeMCU v3 — Zerynth Docs documentation. [online]. [Accessed 20 September 2019]. Available from: https://docs.zerynth.com/latest/official/board.zerynth.nodemcu3/docs/index.html

[31] NodeMcu ESP8266. *Naylamp Mechatronics - Perú* [online]. [Accessed 20 September 2019]. Available from: https://naylampmechatronics.com/espressif-esp/153-nodemcu-esp8266.html

[32] ESP8266EX Wi-Fi SoC Has Been FCC & CE Certified - ESP8266 Developer Zone. [online]. [Accessed 20 September 2019]. Available from: https://bbs.espressif.com/viewtopic.php?f=9&t=59#p209

[33] bluetooth - Must FCC-certified modules be labelled with the correct FCC ID? *Electrical Engineering Stack Exchange* [online]. [Accessed 20 September 2019]. Available from: https://electronics.stackexchange.com/questions/165396/must-fcc-certified-modules-be-labelled-with-the-correct-fcc-id

[34] EGGERTON, John. FCC Loosens Device Labeling Restrictions. *Multichannel* [online]. [Accessed 20 September 2019]. Available from: https://www.multichannel.com/news/fcc-loosens-device-labeling-restrictions-375863

[35] Equipment Authorization - Mexico MRA. *Federal Communications Commission* [online]. 16 July 2015. [Accessed 15 May 2019]. Available from: https://www.fcc.gov/general/equipment-authorization-mexico-mra

[36] *us-mexico_telecom_mra_english_final.pdf* [online]. [Accessed 18 May 2019]. Available from: https://www.nist.gov/sites/default/files/documents/2017/07/07/us-mexico_telecom_mra_english_final.pdf

[37] MFRC522 Standard performance MIFARE and NTAG frontend. 2016. Vol. 2016, p. 95.

[38] Serial Peripheral Interface (SPI) - learn.sparkfun.com. [online]. [Accessed 29 October 2019]. Available from: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all

[39] LM7805.pdf. [online]. [Accessed 29 October 2019]. Available from: http://ee-classes.usc.edu/ee459/library/datasheets/LM7805.pdf

[40] uln2001.pdf. [online]. [Accessed 29 October 2019]. Available from: https://www.st.com/resource/en/datasheet/uln2001.pdf

[41] About KiCad. [online]. 9 April 2014. [Accessed 10 May 2019]. Available from: http://kicad-pcb.org/about/kicad/

[42] Who is JLCPCB? - JLCPCB: Help & Support. [online]. [Accessed 31 October 2019]. Available from: https://support.jlcpcb.com/article/13-who-is-jlcpcb

[43] About – Yocto Project. [online]. [Accessed 4 February 2020]. Available from: https://www.yoctoproject.org/about/

[44] FrontPage - Raspbian. [online]. [Accessed 11 November 2019]. Available from: https://www.raspbian.org/

[45] Download Raspbian for Raspberry Pi. [online]. [Accessed 4 February 2020]. Available from: https://www.raspberrypi.org/downloads/raspbian/

[46] Installing operating system images - Raspberry Pi Documentation. [online]. [Accessed 4 February 2020]. Available from: https://www.raspberrypi.org/documentation/installation/installing-images/README.md

[47] Protocolo SSH. [online]. [Accessed 4 February 2020]. Available from: https://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-ssh.html

[48] SSH (Secure Shell) - Raspberry Pi Documentation. [online]. [Accessed 4 February 2020]. Available from: https://www.raspberrypi.org/documentation/remote-access/ssh/README.md#3-enable-ssh-on-a-headless-raspberry-pi-add-file-to-sd-card-on-another-machine

[49] About Python™ | Python.org. [online]. [Accessed 4 February 2020]. Available from: https://www.python.org/about/

[50] La base de datos líder del mercado para aplicaciones modernas | MongoDB. [online]. [Accessed 4 February 2020]. Available from: https://www.mongodb.com/es

[51] RAMÍREZ, Iván. ¿Qué es una conexión VPN, para qué sirve y qué ventajas tiene? *Xataka* [online]. 2 August 2019. [Accessed 25 February 2020]. Available from: https://www.xataka.com/basics/que-es-una-conexion-vpn-para-que-sirve-y-que-ventajas-tiene

[52] About ZeroTier – ZeroTier. [online]. [Accessed 25 February 2020]. Available from: /about/

[53] Join a Network - ZeroTier Knowledgebase - Confluence. [online]. [Accessed 15 May 2019]. Available from: https://zerotier.atlassian.net/wiki/spaces/SD/pages/6848513/Join+a+Network

[54] Installing — ESP8266 Arduino Core documentation. [online]. [Accessed 25 February 2020]. Available from: https://arduino-esp8266.readthedocs.io/en/latest/installing.html

[55] PEP 0 -- Index of Python Enhancement Proposals (PEPs). *Python.org* [online]. [Accessed 25 February 2020]. Available from: https://www.python.org/dev/peps/

[56] PyCharm: el IDE de Python para desarrolladores profesionales, por JetBrains. *JetBrains* [online]. [Accessed 25 February 2020]. Available from: https://www.jetbrains.com/pycharm/

[57] Postman | The Collaboration Platform for API Development. *Postman* [online]. [Accessed 2 March 2020]. Available from: https://www.postman.com

[58] gnu.org. [online]. [Accessed 2 March 2020]. Available from: https://www.gnu.org/licenses/gpl-3.0.html

[59] Raspberry Pi Foundation - About Us. *Raspberry Pi* [online]. [Accessed 2 March 2020]. Available from: https://www.raspberrypi.org/about/

[60] The Lean Startup | Methodology. [online]. [Accessed 5 May 2020]. Available from: http://theleanstartup.com/principles

[61] What are CRUD Operations? Examples, Tutorials & More. *Stackify* [online]. 2 May 2017. [Accessed 22 May 2020]. Available from: https://stackify.com/what-are-crud-operations/

[62] ANDRADE, Rudy. *Emprende en México: Desarrolla y valida tu modelo de negocio en el entorno mexicano*. 1. 2019.

[63] Estudio de Salarios SG 2019. *SG Buzz* [online]. [Accessed 23 June 2020]. Available from: https://sg.com.mx/revista/58/estudio-de-salarios-sg-2019

[64] nodemcu/nodemcu-firmware. *GitHub* [online]. [Accessed 24 June 2020]. Available from: https://github.com/nodemcu/nodemcu-firmware

[65] *Snapshot* [online]. [Accessed 29 June 2020]. Available from: https://www.gnu.org/licenses/gpl-3.0.html

[66] ESP32 vs ESP8266 - Pros and Cons. *Maker Advisor* [online]. 18 May 2020. [Accessed 6 July 2020]. Available from: https://makeradvisor.com/esp32-vs-esp8266/

[67] Arreglo de 7 Transistores Darlington Tipo DIP de 16 pines ULN2003A. *Talos Electronics* [online]. [Accessed 7 July 2020]. Available from: https://www.taloselectronics.com/products/arreglo-de-7-transistores-darligton-tipo-dip-de-16-pines-uln2003a

[68] NodeMCU. *Wikipedia, la enciclopedia libre* [online]. 2020. [Accessed 7 July 2020]. Available from: https://es.wikipedia.org/w/index.php?title=NodeMCU&oldid=127152892Page Version ID: 127152892

[69] ALEX9UFO. alex9ufo 聰明人求知心切: NodeMCU + RFID + MQTT + Node-RED. *alex9ufo 聰明人求知心切* [online]. 9 August 2019. [Accessed 7 July 2020]. Available from: http://alex9ufoexploer.blogspot.com/2019/08/nodemcu-rfid-mqtt-node-red.html

[70] Teknomovo - teknomovo, regulador, voltaje, 5v, positivo, termica, 7805, encapsulado, TO-220, corriente. [online]. [Accessed 7 July 2020]. Available from: http://www.teknomovo.mx/productos/reguladoresdevoltaje/regulador-5v-positivo-7805.html

[71] Disipador Calor Para TO-220 Mosfet Transistores – DIPMECATRONICA. [online]. [Accessed 7 July 2020]. Available from: https://www.dipmecatronica.com.mx/product/disipador-calor-para-to-220-mosfet-transistores/

[72] Using ULN2003 to drive relays - first time schematic. [online]. [Accessed 7 July 2020]. Available from: https://forum.arduino.cc/index.php?topic=51342.0

[73] ESTRADA, Ruben. Relevador electrónico 5 Vcc. *HeTPro* [online]. [Accessed 7 July 2020]. Available from: https://hetpro-store.com/relevador-electronico-5-vcc/

[74] Pinterest. *Pinterest* [online]. [Accessed 7 July 2020]. Available from: https://www.pinterest.com.mx/duinocasescom/raspberry-pi-3-model-b/

[75] Modulo Rfid Mfrc522 Tarjeta Llavero Tag Arduino Mona. *Monarca Electrónica* [online]. [Accessed 7 July 2020]. Available from: https://monarcaelectronica.com.ar/productos/modulo-rfid-mfrc522-tarjeta-llavero-tag-arduino-mona/

[76] Módulo ESP8266 ESP-12E. *Naylamp Mechatronics - Perú* [online]. [Accessed 7 July 2020]. Available from: https://naylampmechatronics.com/espressif-esp/176-modulo-esp8266-esp-12e.html

[77] Guaostudio. Guaostudio [online]. [Accessed 15 July 2020]. Available from: https://guaostudio.com

[78] 1. PERDOMO, Ana Estefanía Mantilla. INGENIERA EN ELECTRÓNICA, CONTROL Y REDES INDUSTRIALES. P. 165.