UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE ELECTRÓNICA EN CONTROL Y AUTOMATISMO

TEMA:

Diseño y ensamble de un prototipo de alimentador automático ecológico para piscinas de cría de camarón

AUTOR:

Jaramillo Abril Manuel Alejandro

Trabajo de Titulación previo a la obtención del título de

INGENIERO ELECTRÓNICO EN CONTROL Y AUTOMATISMO

TUTOR:

M. Sc. Suárez Murillo, Efraín Oswaldo

Guayaquil, Ecuador

8 de marzo del 2021

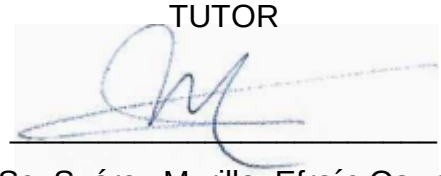
UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO
CARRERA DE ELECTRÓNICA EN CONTROL Y AUTOMATISMO

CERTIFICACIÓN

Certifico que el presente trabajo fue realizado en su totalidad por el Sr. **Jaramillo Abril, Manuel Alejandro** como requerimiento para la obtención del título de **INGENIERO ELECTRÓNICO EN CONTROL Y AUTOMATISMO**.

TUTOR



M. Sc. Suárez Murillo, Efraín Oswaldo

DIRECTOR DE CARRERA

M. Sc. Heras Sánchez, Miguel Armando

Guayaquil, a los 8 días del mes de marzo del año 2021

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO
CARRERA DE ELECTRÓNICA EN CONTROL Y AUTOMATISMO

DECLARACIÓN DE RESPONSABILIDAD

Yo, **Jaramillo Abril, Manuel Alejandro**

DECLARÓ QUE:

El trabajo de titulación **"Diseño y ensamble de un prototipo de alimentador automático ecológico para piscinas de cría de camarón"** previo a la obtención del Título de **Ingeniero Electrónico en Control y Automatismo**, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Guayaquil, a los 8 días del mes de marzo del año 2021

EL AUTOR

JARAMILLO ABRIL, MANUEL ALEJANDRO

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO
CARRERA DE ELECTRÓNICA EN CONTROL Y AUTOMATISMO

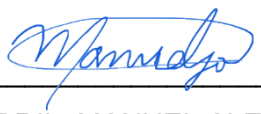
AUTORIZACIÓN

Yo, **Jaramillo Abril, Manuel Alejandro**

Autorizo a la Universidad Católica de Santiago de Guayaquil, la publicación, en la biblioteca de la institución del Trabajo de Titulación: **"Diseño y ensamble de un prototipo de alimentador automático ecológico para piscinas de cría de camarón"**, cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y total autoría.

Guayaquil, a los 8 días del mes de marzo del año 2021

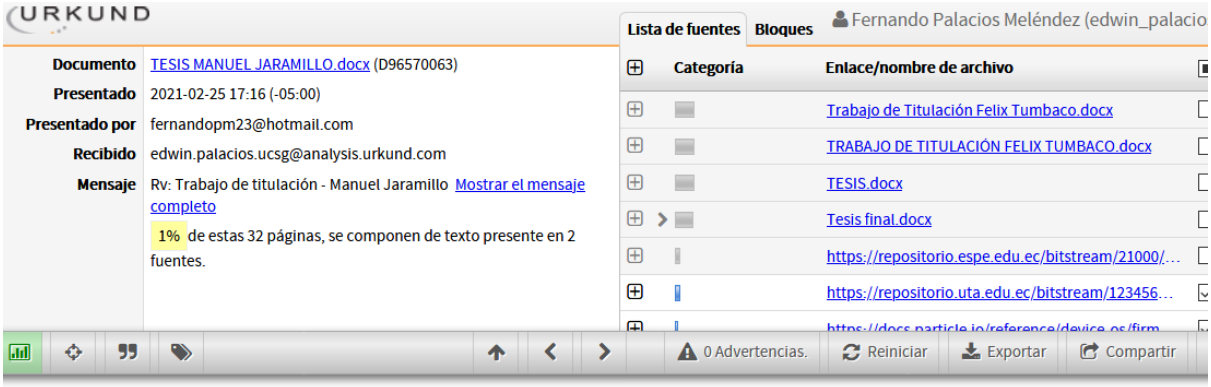
EL AUTOR

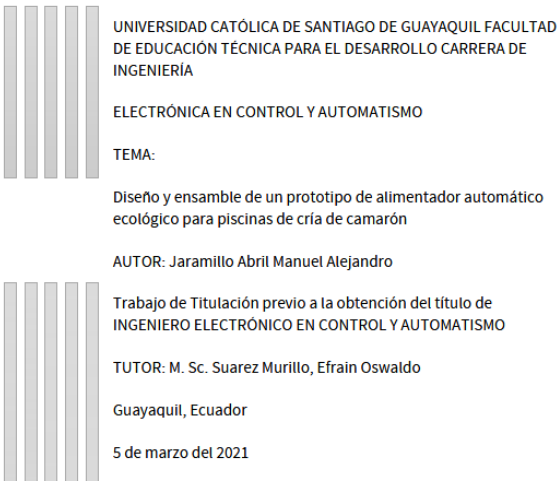


JARAMILLO ABRIL, MANUEL ALEJANDRO

REPORTE DE URKUND

Informe del Trabajo de Titulación de la Carrera de Ingeniería ELECTRÓNICA EN CONTROL Y AUTOMATISMO, con 1% de coincidencias perteneciente al estudiante, MANUEL ALEJANDRO JARAMILLO ABRIL.





Atte.

M. Sc. EFRAÍN OSWALDO SUÁREZ MURILLO

TUTOR TRABAJO DE TITULACIÓN

DEDICATORIA

Dedico este trabajo principalmente a Dios, por haberme dado la vida y permitirme el haber llegado hasta este momento tan importante de mi formación profesional.

A mis padres: Pablo Eduardo Jaramillo Rodríguez y Tania María Abril Mera, por ser los pilares más importantes y por demostrarme siempre su cariño y apoyo incondicional sin importar nuestras diferencias de opiniones.

EL AUTOR

JARAMILLO ABRIL, MANUEL ALEJANDRO

AGRADECIMIENTO

Agradezco a Dios por bendecirme, por guiarme a lo largo de mi existencia, ser el apoyo y fortaleza en aquellos momentos de dificultad y de debilidad.

Gracias a mi padre y a mi madre, por ser los principales promotores de mis sueños, por confiar y creer en mis expectativas, por los consejos, valores y principios que me han inculcado.

Agradezco a los docentes de la Facultad Técnica para el Desarrollo de la Universidad Católica de Santiago de Guayaquil, por haber compartido sus conocimientos a lo largo de la preparación académica para mi profesión, de manera especial, al Ing. Efraín Suarez Murillo tutor de mi proyecto de investigación quien me ha guiado con su paciencia, y su rectitud como docente.

EL AUTOR

JARAMILLO ABRIL, MANUEL ALEJANDRO

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE ELECTRÓNICA EN CONTROL Y AUTOMATISMO

TRIBUNAL DE SUSTENTACIÓN

f. 

M. Sc. ROMERO PAZ, MANUEL DE JESUS

DECANO

f. 

M. Sc. PALACIOS MELÉNDEZ, EDWIN FERNANDO

COORDINADOR DEL ÁREA

f. 

M. Sc. PHILCO ASQUI, LUIS ORLANDO

OPONENTE

ÍNDICE GENERAL

Contenido Pág.

ÍNDICE DE FIGURAS

Capítulo 2

Capítulo 3

ÍNDICE DE TABLAS

Capítulo 2

RESUMEN

En el presente trabajo se realiza el diseño y ensamble de un alimentador automático ecológico para piscinas de camarón. Se empieza explicando antecedentes importantes de la cría de camarón en Ecuador y se definen los objetivos que se desea cumplir en este trabajo, esto se encuentra dentro del primer capítulo. Para entender mejor el contexto de que trata un alimentador automático ecológico se da una fundamentación teórica en el capítulo 2, en el cual se explican conceptos importantes como la automatización, tipo de cultivo, la camaronicultura sustentable, etc. Luego de repasar la fundamentación teórica, se explica todo el diseño y ensamble de la máquina en el capítulo 3, en el cual también se indican mediciones, circuitos esquemáticos en Proteus, programación en Particle IDE, diseño de la app en Blynk, etc. En cada una de las secciones del capítulo 3 se explica a detalle cada paso, conexión y diseño para poder ensamblar el alimentador automático ecológico en su totalidad. Para el mayor entendimiento posible de cada proceso se elaboraron diferentes diagramas, ilustraciones y modelos 3d, los cuales cubren cada detalle del diseño de la estructura y circuitos del alimentador automático.

Palabras claves: Programación, automatización, camaronicultura, ecológico, esquemático, Proteus, Blynk y Particle IDE.

ABSTRACT

In this document we carry out the design and assembly of an ecological automatic feeder for shrimp pools. It begins by explaining important antecedents of shrimp farming in Ecuador and defines the objectives to be met in this work, this is within the first chapter. To better understand the context of an ecological automatic feeder, a theoretical foundation is given in Chapter 2, in which important concepts such as automation, type of cultivation, sustainable shrimp farming, etc. are explained. After reviewing the theoretical foundation, the entire design and assembly of the machine is explained in chapter 3, which also indicates measurements, schematic circuits in Proteus, programming in Particle IDE, app design in Blynk, etc. In each of the sections of chapter 3, each step, connection and design are explained in detail to be able to assemble the ecological automatic feeder in its entirety. For the best possible understanding of each process, different diagrams, illustrations and 3D models were elaborated, which cover every detail of the design of the structure and circuits of the automatic feeder.

Key Words: Programming, automation, sustainable shrimp farming, ecologic, schematic, Proteus, Blynk and Particle IDE.

Capítulo 1: Descripción general del trabajo de titulación

1.1. Introducción.

El cultivo de camarón ha contribuido que la economía de Ecuador siga desarrollándose, especialmente en períodos muy difíciles para las exportaciones del país (la caída de los precios del petróleo). Es por eso el deseo de producir aún más eficientemente de lo que ofrece el sector industrial, reduciendo costos significativos en la línea de producción del camarón, directamente en el campo de la alimentación. Ya que, durante muchos años, se han mantenido los métodos tradicionales de alimentación.

Hoy en día una de las mayores exportaciones del Ecuador se debe gracias al cultivo de camarones. El cultivo de camarón ha contribuido que la economía de Ecuador siga desarrollándose, especialmente en períodos muy difíciles para las exportaciones del país (la caída de los precios del petróleo). Incluso este año 2020 la exportación de este producto ha superado al Banano representando un 18% del total de las exportaciones no petroleras, convirtiéndose en el primer producto exportable del Ecuador. Es por eso el deseo de producir aún más eficientemente de lo que ofrece el sector industrial, reduciendo costos significativos en la línea de producción del camarón, directamente en el campo de la alimentación. Ya que, durante muchos años, se han mantenido los métodos tradicionales de alimentación.

El presidente de la Cámara Nacional de Acuicultura (CNA), José Antonio Camposano explicó que el volumen de exportación en los últimos 7 años se ha incrementado con una tasa de crecimiento promedio del cultivo de camarón del 12% y el 15% anuales, lo que representa el 11% de la comercialización global. Además, la industria camaronera ha creado, de forma directa e indirecta, más de 200.000 plazas de trabajo en el país. Sin embargo, estos porcentajes podrían ser mayores si se implementan sistemas automatizados para aumentar la producción en las diferentes camaroneras existentes, ya que no todas poseen de un proceso de alimentación óptimo para las larvas de camarón. En el presente trabajo se propone una solución para este problema,

el cual consiste en diseñar un dispositivo automatizado capaz de distribuir de mejor manera el alimento de larvas en proporciones e intervalos de tiempo adecuados para optimizar el crecimiento y cultivo del camarón. Este dispositivo no solo aumentará la producción del camarón, sino que permitirá un ahorro en la mano de obra de las camaroneras y un ahorro en la compra del alimento.

1.2. Antecedentes.

En Ecuador, las actividades de producción de camarón de estanque comenzaron en la provincia de El Oro a fines de la década de 1960. Cuando un grupo de empresarios notó el pico de marea en los llamados aguajes, los camarones ingresaban por el estero y se quedaban en la laguna que formaban durante esos aguajes. Al limitar el espacio con muros hechos de tierra, se forma un estanque donde podía crecer las semillas de camarones silvestres. Lo que siguió fue la idea de construir una laguna artificial, permitiendo que la marea fluyera durante el proceso de aguaje, luego confinar el agua conteniendo las semillas.

A las pocas semanas, los camarones eran capturados mediante una técnica de pesca llamada trasmallo. La industria camaronera en Ecuador ha experimentado el surgimiento de patologías propias del cultivo, la situación política y económica interna del país, cambios en la demanda internacional, competencia feroz y bajos precios en el mercado mundial, y actualmente es una industria competitiva contra los grandes productores asiáticos y, además, aumentando el rendimiento.

1.3. Definición del Problema.

El problema a solucionar en este trabajo es la distribución no adecuada del alimento en las piscinas de cría de camarón. Problema el cual surge de un método rústico de alimentación a mano por trabajadores en canoas, de esta manera racionando el balanceado de una manera poco precisa, malgastando alimento y distribuyendo en horarios no optimizados.

1.4. Justificación del Problema.

En muchas camaroneras del país, especialmente aquellas con áreas más grandes, el método tradicional de alimentación en el estanque de engorde conlleva muchas dificultades. Los trabajadores del área de alimentación tienen problemas porque a menudo no pueden cumplir la alimentación asignada a un 100% por motivos de tiempo y operatividad. Se observa que en ocasiones los encargados de alimentación cometen estos errores por completar rápidamente su trabajo de esta manera incumpliendo el total de piscinas asignadas, desperdiciando la cantidad total de raciones en una sola área, omitiendo el correcto esparcimiento del alimento en las áreas específicas, y en ocasiones ignorando algunas piscinas dejándolas sin alimento.

Además de las desventajas de utilizar los métodos de alimentación tradicionales, también ocurren otros problemas provocados por este método rústico. Entre estos está el desperdicio de comida, que se mal gasta al no considerar el tiempo preciso o la actividad alimenticia del camarón, cuando se dispersa el alimento bruscamente en la piscina. También se presenta el problema de que los alimentos no utilizados se acumulan en el fondo de las piscinas, lo que genera sedimentos y una alta densidad, provocando el deterioro la calidad del agua.

1.5. Objetivos del Problema de Investigación.

1.5.1. Objetivo General.

Diseñar un dispositivo automatizado para optimizar la alimentación en las piscinas de cría de camarón.

1.5.2. Objetivos Específicos.

- Programar un software y una App que permitan controlar el tiempo de alimentación.
- Emplear una fuente de energía renovable.
- Crear una estructura capaz de soportar climas extremos.

1.6. Hipótesis.

A través de la programación de un código en el microcontrolador Particle Photon utilizando la plataforma Particle IDE, una App en Blynk y un diseño de estructura resistente, se crea el alimentador automático con todas sus características funcionales controladas desde la app como son el control remoto por wifi, la programación de un temporizador para que se active cada cierto tiempo que el usuario crea necesario, y la programación del tiempo de duración de encendido del alimentador según lo crea necesario el usuario.

1.7. Metodología de Investigación.

El presente trabajo de investigación es de tipo aplicada, ya que se centra específicamente en como se pueden llevar a la práctica las teorías generales y su motivación va hacia la resolución de problemas que se plantea en un momento dado; lo cual se evidencia en el conjunto de actividades metódicas y técnicas para el diseño de un alimentador automático desde cero. También cuenta con un enfoque cuantitativo al tener la posibilidad de controlar las variables de diseño y construcción del mismo. Esto implica que se realizará el diseño de la estructura (Distribuidor de alimentos, flotadores, conexiones) y software (código del microcontrolador). También se hará una aplicación de fuente de energía renovable en la cual se contará con un panel solar conectado a una batería de alimentación. De esta manera se logra que el dispositivo sea auto sustentable y facilita la aplicación en la industria camaronera.

Capítulo 2: Fundamentación teórica

2.1. Automatización Aplicada en la Industria Camaronera.

La automatización es un conjunto de elementos o procesos informáticos, mecánicos y electromecánicos que operan con poca o ninguna intervención humana. Suelen utilizarse para optimizar y mejorar el funcionamiento de las fábricas, pero la automatización también se puede utilizar en estadios, granjas e incluso infraestructura urbana (Logicbus, 2019). Hoy en día, la robótica y la tecnología informática permiten ampliar el alcance de la automatización. En innumerables campos industriales se utilizan máquinas que permiten la automatización de procesos. La automatización tiene varias ventajas: además de ahorrar tiempo, generalmente contribuye a la precisión del desarrollo de tareas.

La automatización es un pilar fundamental en el proyecto propuesto a aplicarse a la industria camaronera, ya que el proyecto se basa en eliminar un proceso que es hecho por la mano del hombre para reemplazarlo con la exactitud de una maquina automatizada capaz de tener un mejor control sobre la correcta alimentación del camarón. Con la base de la automatización se puede generar grandes mejoras en la producción, ya que hay estudios que demuestran las mejoras del alimentador automático sobre el crecimiento del camarón y el ahorro del alimento, comparados con el método de alimentación en canoas por la mano de trabajadores.

La idea principal es hacer que los trabajos más complicados o peligrosos no sean realizados por trabajadores sino por máquinas, mejora y aumenta significativamente la producción, permitiendo que las tareas que no pueden ser realizadas por el ser humano por su meticulosidad o complejidad sean realizadas por máquinas. Al mismo tiempo, la automatización ayuda a mejorar la seguridad de los empleados y minimiza la carga de trabajo humana.

Un sistema de automatización consta de dos partes principales: la parte operativa, que funciona directamente en la máquina. Son los elementos que

hacen que la máquina se mueva y realice las operaciones requeridas. Los elementos que componen la parte operativa son los actuadores de la máquina, como motores, cilindros, compresores y sensores, como fotodiodos y finales de carrera. La otra parte es la de control, habitualmente un autómata programable (tecnología de programación), aunque hasta hace poco se utilizaban relés electromagnéticos, tarjetas electrónicas o módulos lógicos neumáticos (tecnología cableada). En el sistema de fabricación automatizado, el controlador programable está ubicado en el centro del sistema. Debe poder comunicarse con todos los componentes del sistema de automatización (sc.ehu, 2001).

2.1.1. Tecnologías IoT.

La definición de Internet de las cosas puede ser la agrupación e interconexión de dispositivos y objetos a través de una red (red privada o Internet, red de red), donde todos estos dispositivos y objetos son visibles y pueden interactuar. En cuanto al tipo de objetos o equipos, pueden ser cualquiera, desde sensores y dispositivos mecánicos hasta objetos cotidianos (como neveras, zapatos o ropa). Todo lo imaginable se puede conectar a Internet e interactuar sin intervención humana, por lo que el objetivo es la interacción de máquina a máquina, o la interacción M2M (máquina a máquina) o el equipo M2M.

El rápido desarrollo de Internet ha hecho que Internet de las cosas sea una realidad, no solo una visión para el futuro. La fama de esta tecnología se debe principalmente a que brinda todas las aplicaciones y posibilidades para mejorar la vida diaria de las personas y el entorno empresarial, y esta tecnología se viene implementando desde hace algún tiempo. Estas aplicaciones son casi infinitas, pero se describirán algunos ejemplos para hacerlas visibles tanto en la vida diaria como en los entornos empresariales:

En un supuesto caso en el que los alimentos se almacenan en el refrigerador de una casa y que los alimentos tienen fecha de caducidad los frigoríficos se pueden conectar a Internet para notificar a los usuarios a través de sus teléfonos móviles, por ejemplo, cuando caducan los alimentos, la

temperatura baja por un mal funcionamiento, los alimentos se agotan o el consumo de energía solo se calcula en función del número de veces que se abre la puerta del frigorífico.

Otra situación podría ser la domótica, donde ya hay muchos dispositivos conectados a Internet para facilitar la vida de las personas, por ejemplo, ver Dispositivos de control por voz, pidiéndoles que reproduzcan canciones de la biblioteca de almacenamiento. Internet, o dispositivos y aplicaciones que permitan el control de todos los parámetros del agua en el acuario, o incluso los sistemas de alarma de las casas conectadas a centrales eléctricas. Cuando alguien ingrese a una casa un sistema de seguridad conectado a la red podrá notificar al propietario de manera instantánea o un dispositivo que permita encender la calefacción con el teléfono móvil.

Si se considera aplicaciones industriales, IoT ya se utiliza en muchas plantas de producción, donde los dispositivos y sensores conectados a la red permiten analizar los datos, y se generan y envían alertas y mensajes a diferentes usuarios para que puedan tomar las medidas necesarias, incluso no se requiere de intervención manual para tomar automáticamente planes de acción para corregir o hacer frente a estas alarmas.

Como se puede apreciar, el Internet de las Cosas ya existe y se ha convertido en una realidad, su alcance es muy amplio, y cada día aparecen más dispositivos para hacer posible esta tecnología. Esta tecnología relacionada con el Internet de las Cosas permite recopilar datos y enviarlos a la red para su análisis, o incluso realizar análisis previos antes de enviarlos a la red.

El Internet de las Cosas se ha desarrollado en este proceso de comunicación, porque uno de los obstáculos a superar es el tipo de protocolo con el que se comunican estos dispositivos (es decir, el "lenguaje" que se utiliza entre estos). Actualmente, existen dispositivos o sensores muy nuevos que pueden comunicarse y conectarse a Internet de manera fácil y directa, pero también hay muchos otros dispositivos antiguos no estándar cuyos

protocolos de comunicación y conexión no son simples. Además, cada fabricante o "proveedor" tiene su propio protocolo de comunicación, lo que significa que no todos los dispositivos son compatibles. Uno de los mecanismos que se ha intentado establecer es la creación de un protocolo abierto y estándar (propuesto por IBM) denominado MQTT (Message Queueing Telemetry Transmission), que permite a todos los fabricantes participar y soportar el protocolo, facilitando así la comunicación entre diferentes dispositivos. Diferentes fabricantes.

Por otro lado, si buscas dispositivos para el Internet de las Cosas, debes considerar varios aspectos, como el bajo consumo de energía y el pequeño tamaño, por lo que SoC (SoC, System on Chip) es una parte importante de estos dispositivos. Un SoC es un circuito integrado que contiene todos o la mayoría de los módulos que tendrá una computadora (por ejemplo, un SoC se puede encontrar en un teléfono móvil). Como ejemplos de grandes fabricantes se tiene a ARM e Intel, aunque no son los únicos fabricantes, existen nuevos fabricantes como MediaTek, Qualcomm o Samsung. Además, para todos los usuarios, existen alternativas muy asequibles, como Arduino, que permite a los usuarios montar sus propios dispositivos y circuitos de control del hogar.

Otra parte importante de los dispositivos IoT son los sensores, procesadores y plataformas que se encargan de administrar la información, pero esta debe provenir de sensores. En este sentido, Arduino pone esta tecnología a disposición de todos los usuarios. Además, los proveedores que brindan servicios en la nube también proporcionan kits de herramientas equipados con varios sensores y pueden conectarse fácilmente con los servicios.

Finalmente, otro componente técnicamente importante de Internet de las cosas es la tecnología utilizada para comunicarse entre varios dispositivos (es decir, redes de comunicación) que no están cerca unos de otros. Por ejemplo, en esta sección, puede discutir la comunicación a través de una red "WiFi", que permite altas velocidades de transmisión, pero consume mucho dinero y

tiene un alcance reducido. Otro ejemplo muy conocido es la red móvil (3G, 4G o 5G en el futuro), que tendrá mayor alcance y menor consumo. Además, existen otros tipos de redes de IoT dedicadas, como Sigfox (que tiene una amplia cobertura en EE. UU. Y Europa) o LoRa.

Como se mencionó anteriormente, Internet de las cosas tiene un significado muy amplio en aplicaciones ilimitadas. Sin embargo, no es nuevo que los dispositivos estén conectados o tengan un cierto grado de inteligencia. La clave es cómo Internet de las cosas ahora recomienda cómo hacer esto. La idea es conectarlos a Internet, y directamente a Internet si es posible, tienen la capacidad de recopilar datos e información y transmitirla a otros dispositivos, y pueden guardar y analizar la información para mejorar el dispositivo en sí o mejorar otros equipos (Gracia, 2018).

2.1.2. Camaronicultura Sustentable.

El desarrollo de la tecnología pesquera ha promovido la investigación y la fabricación de productos para respaldar el éxito del cultivo de camarón. Uno de estos es la alimentación automática. Una máquina que se puede ajustar de acuerdo con el nivel de requisitos de alimentación que se han desarrollado previamente.

El principal desafío del cultivo científico de camarón es proporcionar la cantidad adecuada de alimento artificial en el momento adecuado según sea necesario. El mayor costo operativo de la acuicultura es el costo del alimento, que puede representar el 50% o más, esto es un hecho. El desarrollo del alimento y métodos de alimentación de alta calidad es importante porque afectan la cantidad total del alimento consumido, la calidad del suelo y el agua del estanque y, en última instancia, el éxito del cultivo.

Los alimentadores automáticos y los sistemas de alimentación juegan un papel importante en el éxito de las granjas acuícolas mundiales y, en un futuro cercano, se harán necesarios sistemas de acuicultura intensiva con densidades de población más altas. En estanques camaroneros con operaciones de cultivo intensivo, se han probado varios alimentadores

automáticos diseñados principalmente para peces. Comercialmente, muchos modelos están disponibles en países asiáticos como Tailandia, China, Malasia y Vietnam, y tienen opiniones diferentes sobre sus aplicaciones y rendimiento.

La idea propuesta en el presente trabajo para la automatización en las camaroneras se trata de un alimentador automático y autónomo capaz de alimentar a las larvas de camarón de manera que se puedan programar horarios y tiempos personalizados para la necesidad del propietario. Además, que sea capaz de operar bajo condiciones climáticas extremas como se da en la aplicación real de las piscinas de larva de camarón.

Siguiendo los conceptos de la automatización y las tecnologías Iot se puede llevar a cabo el diseño de este producto a la mano de los conocimientos en ingeniería electrónica en control y automatismo.

2.1.3. Energía renovable.

La energía renovable es una fuente de energía que se obtiene de los recursos naturales y produce energía de forma indefinida. La energía solar, mareomotriz y eólica son claros ejemplos de cuáles son las fuentes de energía renovable. En este proyecto se utiliza la energía solar como fuente de energía renovable, ya que se utiliza un panel solar de 20W en la elaboración del alimentador automático, de esta manera mereciendo el nombre “Ecológico” al ser amigable con el medio ambiente (Lineaverdehuelva, 2018). La figura 2.1 muestra un sistema de paneles fotovoltaicos para aprovechar la energía que se genera a través del sol.

Figura 2. 1: Energía renovable con panel solar
Fuente: (Tatoma, 2019).

Aparte del deseo de poder ser amigables con el medio ambiente, también se aplicó este método de alimentación por energía renovable debido a que se presenta un problema muy grande a la hora de tratar de energizar el alimentador automático con otro tipo de fuente cableada, ya que el alimentador automático se va a encontrar en su mayoría del tiempo flotando en el agua y es muy difícil resolver este problema de una manera práctica.

2.2. Especificaciones del cultivo.

2.2.1. Especie de cultivo.

Las camaroneras de Ecuador se concentran principalmente en la cría del camarón blanco Litopenaeus Vannnamei. En Ecuador, este tipo de camarón se cría en ciertas áreas de estanques de agua salobre, zonas continentales e insulares pertenecientes a cinco provincias costeras: Guayas, Esmeraldas, El Oro, Manabí y Santa Elena. El ciclo de producción puede ser de 80, 100, 120 o incluso algunos Más de 200 días de tiempo de siembra, según el método utilizado por cada camaronera. Cuanto mayor sea el tamaño de los camarones durante la cosecha, más grande será el precio de venta (Schwarz, 2005). La figura 2.2 muestra un espécimen de camarón Litopenaeus Vannamei.

Figura 2. 2: Litopenaeus Vannamei espécimen.
Fuente: **(Zambritisa, 2018)**.

2.2.2. Tipo de cultivo.

El cultivo de camarón se divide en tres sistemas de engorde o tipos:

- Sistema Extensivo
- Sistema Semi Intensivo
- Sistema Intensivo

Tabla 2. 1: Características de los tipos de cultivo de camarón

Sistema	Principales Características
Extensivo	Densidades bajas: 10 000 – 15 000 /ha
	No se alimenta con dietas formuladas
	Producción promedio: 600 lb/ha/año
Semi Intensivo	Densidades medias: 15 000 – 120 000 /ha
	Se alimenta con dietas formuladas
	Producción promedio: 1000 - 5000 lb/ha/año
Intensivo	Densidades altas: más de 120 000 /ha
	Se alimenta con dietas formuladas
	Producción promedio: mayores a 5000 lb/ha/año

Fuente: **(Crespi & New, 2009)**.

Teniendo el conocimiento de los tipos de cultivos de camarón que existen se puede focalizar mejor el área de aplicación del alimentador automático. Con esto se puede definir que el alimentador será más utilizado en los tipos de cultivo Semi intensivo e Intensivo. Esto no significa que no se pueda aplicar para el tipo de cultivo Extensivo, ya que el uso del dispositivo también dependerá de las necesidades del propietario, ya que el alimentador cumpliría de igual manera su función de control y distribución del alimento.

2.2.3. Estanques de Cultivo.

Los estanques de cría o piscinas de engorde son lugares donde los camarones son ubicados luego de pasar por pre criaderos y se utilizan para incrementar la biomasa de los crustáceos hasta alcanzar el tamaño requerido para su comercialización. En los inicios del cultivo de camarón, el tamaño de estos estanques excedía las 100 hectáreas, lo cual era difícil de controlar y mantener, actualmente el tamaño de estos estanques es menor a las 20 hectáreas facilitando los controles y mantenimientos (Maquilón, 2017). La figura 2.3 muestra un ejemplo de las piscinas o estanques de cultivo de camarón.

Figura 2. 3: Piscinas de cultivo de camarón.
Fuente: **(Piedrahita, 2018)**.

2.2.4. Factor de conversión alimenticia.

El factor de conversión alimenticia de un animal se considera por la relación entre el alimento consumido durante el engorde y la ganancia de peso que produce. A través de esta medición, se puede estimar el valor nutricional de la cantidad de alimento proporcionado y utilizado en el proceso de cría de esta especie. Dado que los animales comen más alimentos mientras más bajo sea su potencial nutricional (Saúl, 2019).

2.2.5. Métodos de Alimentación Tradicionales.

En cuanto a la forma de proporcionar alimento a las criaturas del cultivo, las dos formas más habituales hasta ahora son: la alimentación al voleo, que

consiste en distribuirla en el estanque y distribuirla de la forma más uniforme posible; y la suplementación en bandejas de alimentación. En cuanto a la primera forma, esta se puede realizar de forma manual o mecánica. Para el primer caso, se puede hacer desde la orilla del estanque utilizando una carretilla o con carros, o se puede utilizar un bote pequeño para formar una ruta en zigzag por todo el estanque y cambiar la ruta en el siguiente suministro para cubrir la mayor superficie posible del estanque, evitando así que se acumule en determinadas zonas. En el segundo caso, el suministro se realiza desde la orilla mediante un carro con un alimentador que pueda enviar y dispersar la comida a una distancia considerable (Maquilón, 2017). La figura 2.4 muestra el método de alimentación tradicional al voleo en camaroneras.

Figura 2. 4: Método de alimentación tradicional.
Fuente: **(Gutiérrez, 2019)**.

Capítulo 3: Diseño y ensamble

3.1. Componentes.

Para la construcción del alimentador automático se necesitó de varios componentes, tanto en materiales para la estructura como los componentes electrónicos para la parte de control. Algunos componentes electrónicos fueron diseñados desde 0 usando la plataforma Proteus, la cual sirve para diseños de placas y circuitos electrónicos. Los componentes electrónicos usados en el alimentador automático son:

- Batería 12V.
- Panel solar 20w.
- Controlador del panel solar.
- Motor de 12V.
- Controlador Particle Photon.
- Módulo de relé.
- Convertidor reductor de 12V a 5V (Buck).
- Cableado.

Los componentes y materiales usados para la estructura del alimentador automático son:

- Pletina de 1 pulgada y media por 1/8 (6 metros).
- Tubo de una pulgada.
- Tubo pvc ¾.
- Ángulo 45 grados pvc ¾.
- Tanque para alimento.
- Embudo.
- Ángulo perimetral.
- Fibra de vidrio.
- Rodamiento metálico.
- Adaptador para final de motor.
- Soldadura.
- Poliestireno expandido dos bloques de 200x40x20 cm.
- Caja metálica para proteger componentes electrónicos.

3.1.1. Diseño del módulo de relé.

Entre los componentes esenciales para el alimentador automático se encuentra el módulo de relé, el cual es indispensable para el funcionamiento de la máquina. El módulo cumple la función de activar el motor enviándole los 12v de la batería cuando recibe la señal del microcontrolador en la placa Particle Photon. Sin este módulo sería imposible el control del motor debido a que la placa Particle Photon por sí sola no puede enviar 12v en la salida de sus pines, en cambio con la ayuda del relé del módulo que se conecta directamente a la batería de 12v si es posible activar el motor.

El módulo de relé tiene por un extremo la conexión hacia los pines de la placa Particle para que esta le envíe pulsos cuando sea necesario activar el motor, y por el otro extremo tiene la conexión para alimentar el relé directamente desde la batería. En resumen, este módulo funciona como un switch para activar o desactivar el motor por medio de un relé integrado que es controlado por la placa Particle Photon.

Para el diseño de la placa de este módulo se utilizó la plataforma Proteus, en la cual se hizo un circuito esquemático, un diseño de conexiones en la PCB y un modelo 3D (Robots-argentina, 2020). La figura 3.1 muestra el módulo relé utilizado en el proyecto.

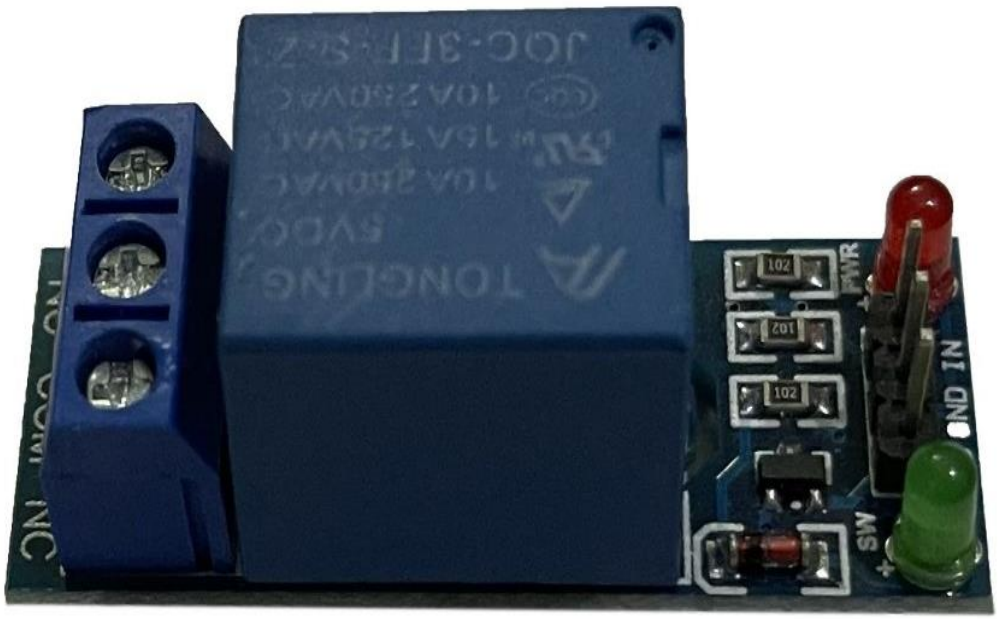


Figura 3. 1: Módulo de relé.
Elaborado por: Autor.

3.1.1.1. Diseño del circuito esquemático.

Para el circuito esquemático del módulo de relé se requirieron varios componentes que ofrece la plataforma Proteus para el correcto funcionamiento del circuito. Es importante tener en cuenta al momento de elegir los elementos que estos tengan un diseño para las conexiones en el PCB y la visualización 3D, ya que si no lo tienen se debe crear uno manualmente o será imposible visualizar dichos elementos al momento de diseñar la placa PCB (DIYmechanics, 2017). La figura 3.2 muestra el diseño del circuito esquemático del módulo de relé en el programa Proteus. Los elementos utilizados en el circuito esquemático fueron:

- Un relé (En Proteus se lo encuentra como “RELAY”)
- Un transistor pnp (En Proteus se lo encuentra como “2SB716”)
- Un diodo Scotchsky (En Proteus se lo encuentra como “DIODE-SC”)
- Dos leds (En Proteus se los encuentra como “LED”)
- 3 resistencias (En Proteus se las encuentra como “RES”)
- Una bornera de 3 bornes (En Proteus se la encuentra como “TBLOCK-I3”)
- 3 pines sobresalientes (En Proteus se los encuentra como “25630301RP2”)

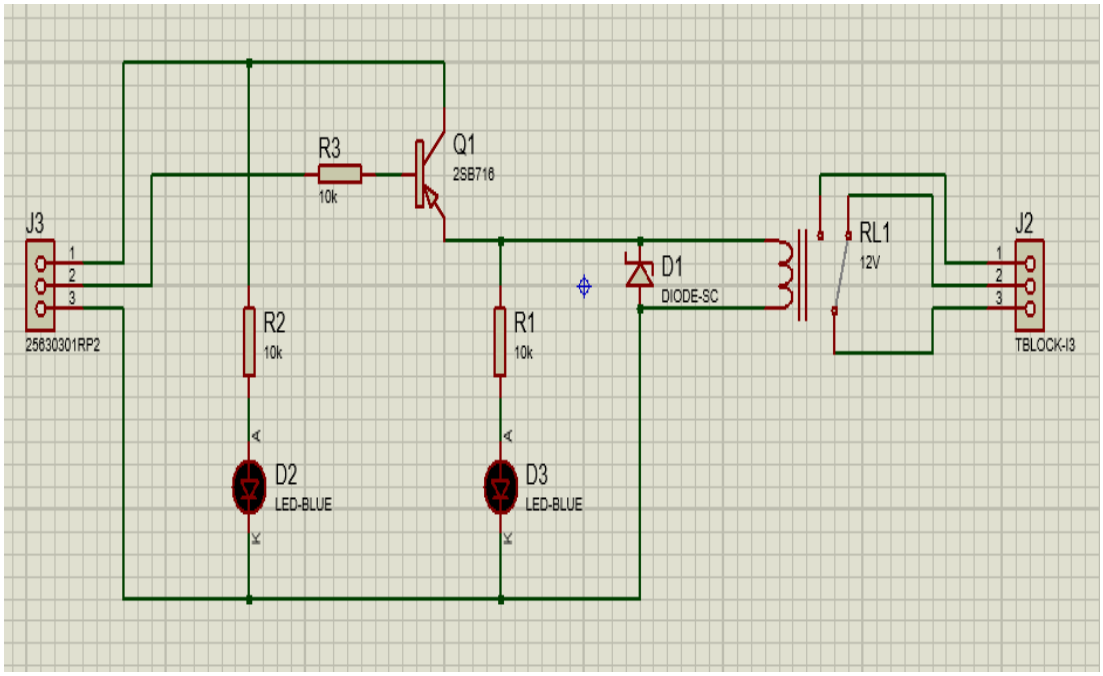


Figura 3. 2: Circuito esquemático del módulo de relé.
Elaborado por: Autor.

3.1.1.2. Conexiones en el PCB.

El siguiente paso para realizar el diseño del módulo de relé es colocar todos los elementos del circuito esquemático en el interfaz de diseño PCB, con sus respectivos símbolos para la placa. Estos símbolos deben de tener las medidas exactas de separación de los pines, para que al momento de colocar los elementos físicos en la PCB estos no se descuadren con los canales de las pistas y puedan atravesar correctamente la placa.

Luego de colocar los elementos del esquemático al gusto se debe trazar las pistas, conectando los pines de los elementos tal y como se realizó en el circuito esquemático. Estas pistas pueden ser trazadas en las dos caras de la placa, es decir, el diseñador puede cruzar pistas de una cara hacia la otra con fines estéticos o funcionales. Las pistas también deben ser diseñadas según el amperaje que cruce por cada pista. En el diseño de esta placa se utilizó en su mayoría pistas de 15 micras y para las pistas que van conectadas al COM, NO y NC del relay se utilizó unas de 30 micras. La figura 3.3 muestra las pistas de conexiones del módulo de relé elaboradas en Proteus (Ares).

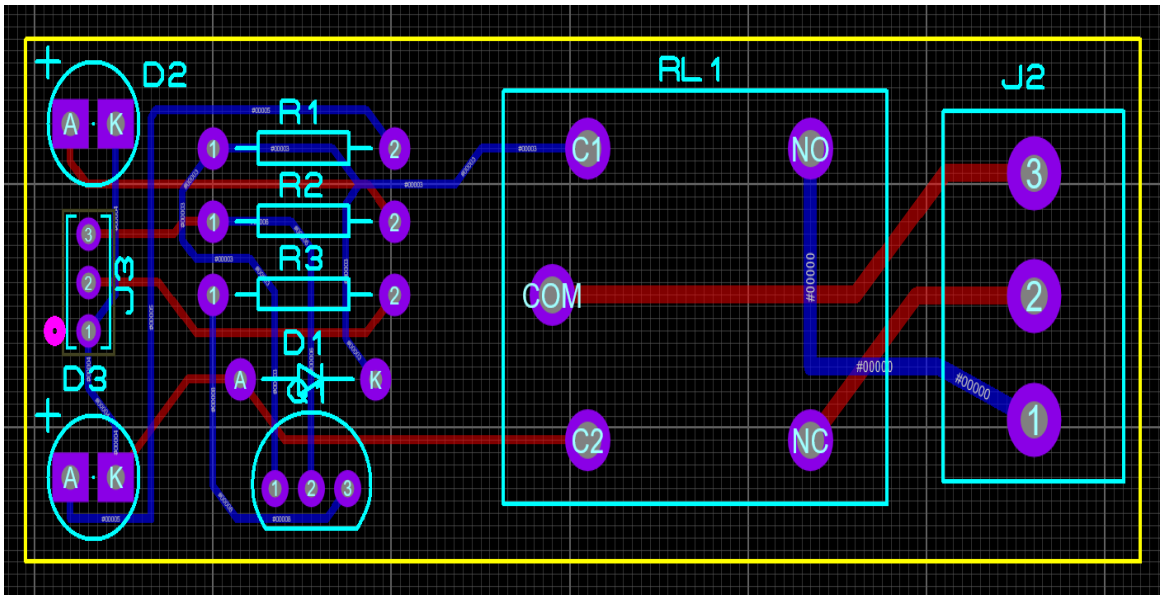


Figura 3. 3: Conexiones en el PCB del módulo de relé
Elaborado por: Autor.

3.1.1.3. Diseño 3D en PCB.

Luego de haber realizado el circuito esquemático y el diseño de la placa PCB hay que generar el modelo 3D de la placa realizada. Este modelo se puede apreciar si se hace click en la opción “visualizador 3D” que se encuentra

en la barra de herramientas, luego de seleccionar esta opción se generará automáticamente un diseño 3D de la placa PCB con las conexiones realizadas. La placa se la puede personalizar al gusto cambiándole el color y los elementos también se pueden detallar en mayor escala si se realiza un buen diseño manual de cada componente en la librería del programa.

Para este diseño se creó nuevos modelos 3D para el relé y la bornera en el programa CAD, ya que no existen estos diseños en Proteus. Para cada diseño 3D se tiene que importar un archivo tipo STEP, el cual se coloca dentro de la opción “modelo 3D” que se encuentra en las herramientas de Proteus. Luego de subir este archivo se tiene que alinear el diseño 3D con los pines del diseño en Ares y ya se podría visualizar en el diseño 3D de la placa. La figura 3.4 muestra el diseño 3D del módulo de relé elaborado en Proteus.

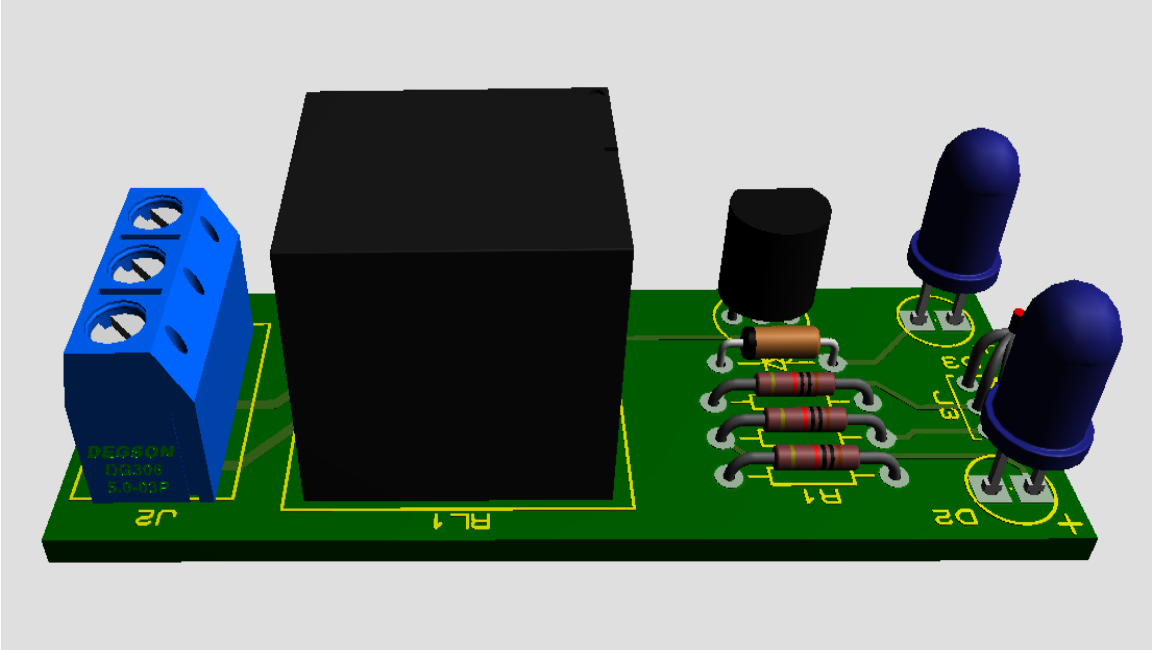


Figura 3. 4: Diseño 3D del módulo de relé.
Elaborado por: Autor.

3.1.2. Diseño del convertidor reductor.

Otro componente esencial para el alimentador automático es el convertidor reductor de 12V a 5V DC, o también conocido como “Buck Converter”. Este convertidor reductor se basa en una reducción por switcheo que consta de un inductor controlado por dos dispositivos semiconductores los cuales alternan la conexión del inductor bien a la fuente de alimentación o bien a la carga. Se utiliza este tipo de convertidor ya que los simples

reductores lineales, como por ejemplo el LM7805, son muy propensos a generar mucho recalentamiento, en cambio un convertidor tipo Buck, gracias a su sistema de switcheo, puede disipar todo ese recalentamiento producido. También se utiliza un potenciómetro para regular el voltaje de salida en caso de que se necesiten menos o más de 5v.

El Buck converter tiene dos estados, los cuales son el ON y OFF. En el estado ON, la fuente de alimentación se encuentra conectada al circuito cerrado cargando el inductor y el capacitor normalmente en una malla producida por el blockeo del diodo Schottky. Luego en el estado OFF, la fuente de alimentación queda en circuito abierto sin poder alimentar a los demás componentes, es aquí donde el inductor usa su energía almacenada y carga el capacitor quedando en un circuito cerrado producido por el diodo Schottky. De esta manera el circuito no permite que existan recalentamientos en los elementos que lo componen (Argos, 2015). La figura 3.5 muestra el convertidor reductor utilizado en este proyecto.

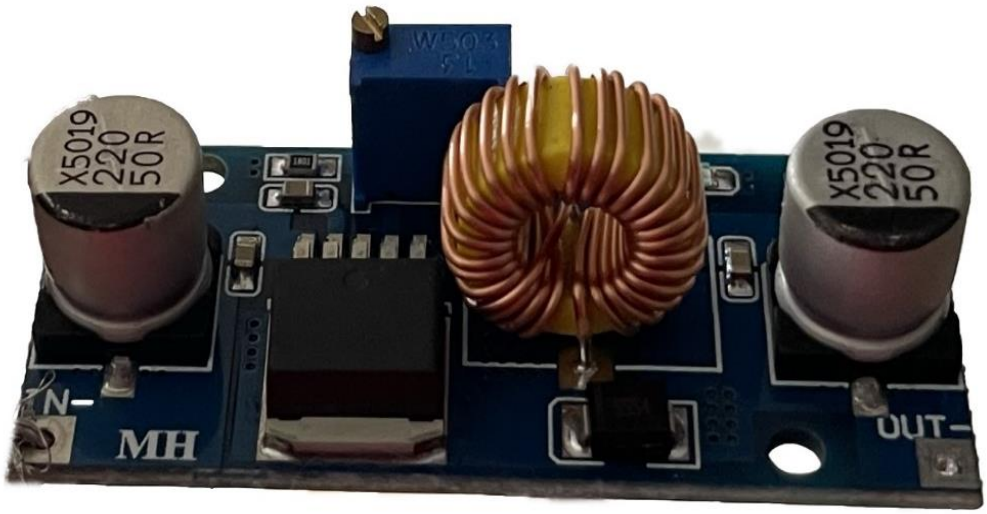


Figura 3. 5: Convertidor reductor.
Elaborado por: Autor.

3.1.2.1. Diseño del circuito esquemático.

Para el circuito esquemático del Buck converter se requirieron varios componentes que ofrece la plataforma Proteus para el correcto funcionamiento del circuito. Es importante tener en cuenta al momento de elegir los elementos que estos tengan un diseño para las conexiones en el PCB y la visualización 3D, ya que si no lo tienen se debe crear uno

manualmente o será imposible visualizar dichos elementos al momento de diseñar la placa PCB (Hons, 2020). La figura 3.6 muestra el diseño del circuito esquemático del convertidor reductor en el programa Proteus. Los elementos utilizados en el circuito esquemático fueron:

- Un LM2576 el cual se tiene que diseñar manualmente, ya que no se encuentra en la librería de Proteus.
- Dos capacitores electrolíticos de 100uF y 1000uF (En Proteus se los encuentra como “CAP-ELEC”).
- Un inductor de 100uH (En Proteus se lo encuentra como “INDUCTOR”).
- Un diodo Schottky (En Proteus se lo encuentra como “DIODE-SC”).
- Un potenciómetro o resistencia variable para regular el voltaje de salida. (En Proteus se lo encuentra como “POT”)
- Cuatro pines para los voltajes de entrada y salida (En Proteus se los encuentra como “PIN”).

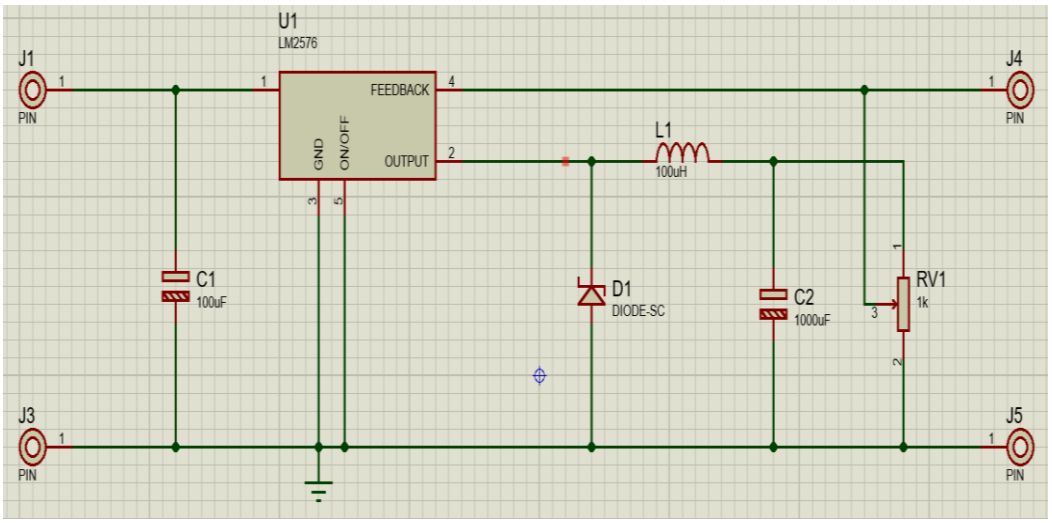


Figura 3. 6: Circuito esquemático del convertidor reductor.
Elaborado por: Autor.

Para este circuito se creó un LM2576 como componente en Proteus, ya que este no consta en las librerías pre instaladas de Proteus. Para esto se tuvo que diseñar el componente en el circuito esquemático, Ares y en el modelo 3D. En el circuito esquemático primero se diseñó la forma del componente, luego se colocaron pines y se los nombró y enumeró según su función, después se creó el componente con la herramienta “Make Device”, en la cual se especificaron más características del componente como el

nombre y tipo de componente, y finalmente se lo conectó a los demás componentes. El diseño en Ares y el modelo 3D se explican más adelante. La figura 3.7 muestra el componente LM2576 creado manualmente en la librería del programa Proteus.

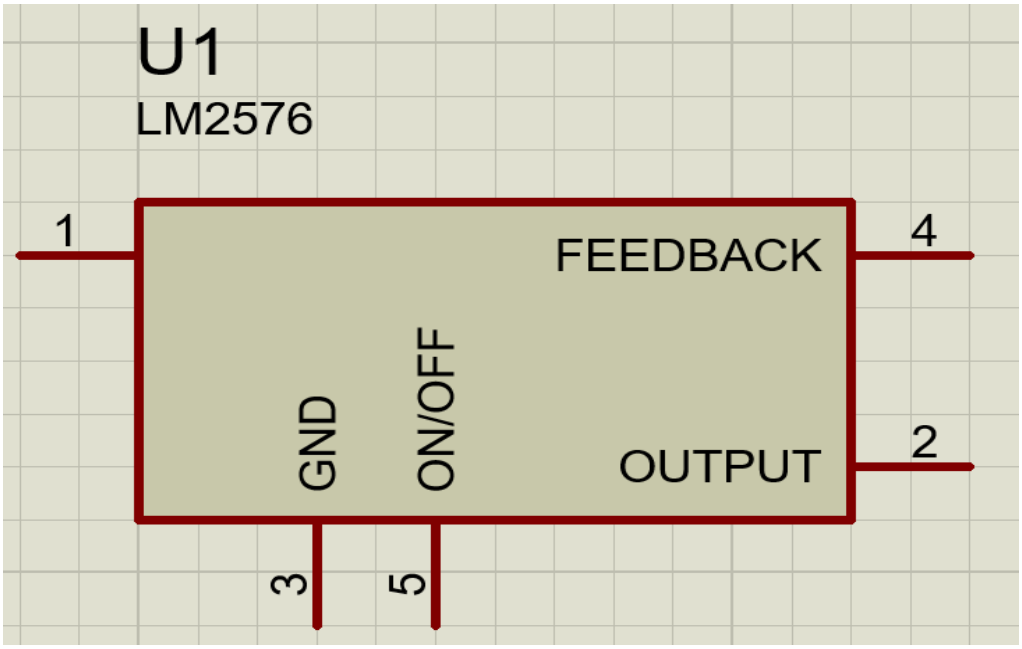


Figura 3. 7: Diseño de un LM2576 en circuito esquemático.
Elaborado por: Autor.

3.1.2.2. Conexiones en el PCB.

El siguiente paso para realizar el diseño del Buck converter es colocar todos los elementos del circuito esquemático en el interfaz de diseño PCB, con sus respectivos símbolos para la placa. Estos símbolos deben de tener las medidas exactas de separación de los pines, para que al momento de colocar los elementos físicos en la PCB estos no se descuadren con los canales de las pistas y puedan atravesar correctamente la placa.

Luego de colocar los elementos del esquemático al gusto se debe trazar las pistas, conectando los pines de los elementos tal y como se realizó en el circuito esquemático. Estas pistas pueden ser trazadas en las dos caras de la placa, es decir, el diseñador puede cruzar pistas de una cara hacia la otra con fines estéticos o funcionales. Las pistas también deben ser diseñadas según el amperaje que cruce por cada pista. En el diseño de esta placa se utilizó en su mayoría pistas de 15 micras y para las pistas que van conectadas directamente a la fuente de 12v se utilizó pistas de 25 micras.

Para el diseño del LM2576 en Ares primero se creó la forma con las medidas reales del componente, luego se utilizó la herramienta “Make Package” para crear el paquete en la librería y asignarlo al componente en el circuito esquemático. Este proceso se repitió en el diseño del inductor y del potenciómetro, ya que estos tampoco constaban en la librería de Arduino. La figura 3.8 muestra las pistas de conexiones del convertidor reductor elaboradas en Proteus (Ares).

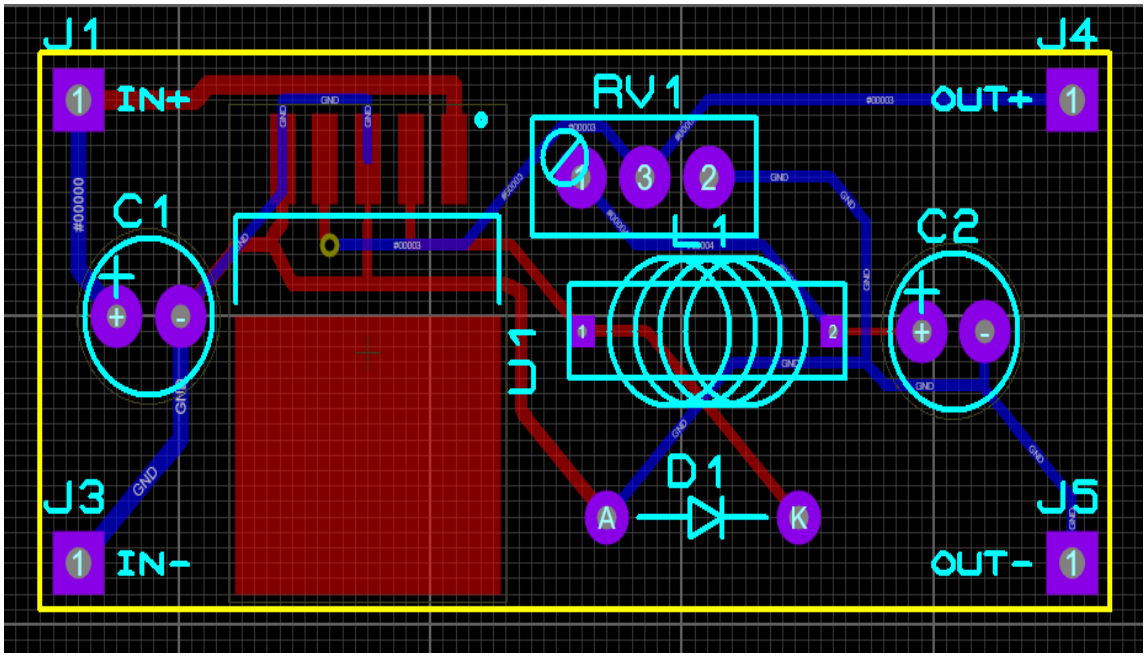


Figura 3. 8: Conexiones en PCB del convertidor reductor.
Elaborado por: Autor.

3.1.2.3. Diseño 3D en PCB.

Luego de haber realizado el circuito esquemático y el diseño de la placa PCB hay que generar el modelo 3D de la placa realizada. Este modelo se puede apreciar si se hace click en la opción “visualizador 3D” que se encuentra en la barra de herramientas, luego de seleccionar esta opción se generará automáticamente un diseño 3D de la placa PCB con las conexiones realizadas. La placa se la puede personalizar al gusto cambiándole el color y los elementos también se pueden detallar en mayor escala si se realiza un buen diseño manual de cada componente en la librería del programa.

Para este diseño se creó nuevos modelos 3D para el LM2576, el inductor y el potenciómetro en el programa CAD, ya que no existen estos diseños en Proteus. Para cada diseño 3D se tiene que importar un archivo tipo STEP, el

cual se coloca dentro de la opción “modelo 3D” que se encuentra en las herramientas de Proteus. Luego de subir este archivo se tiene que alinear el diseño 3D con los pines del diseño en Ares y ya se podría visualizar en el diseño 3D de la placa. La figura 3.9 muestra el diseño 3D del convertidor reductor elaborado en Proteus.

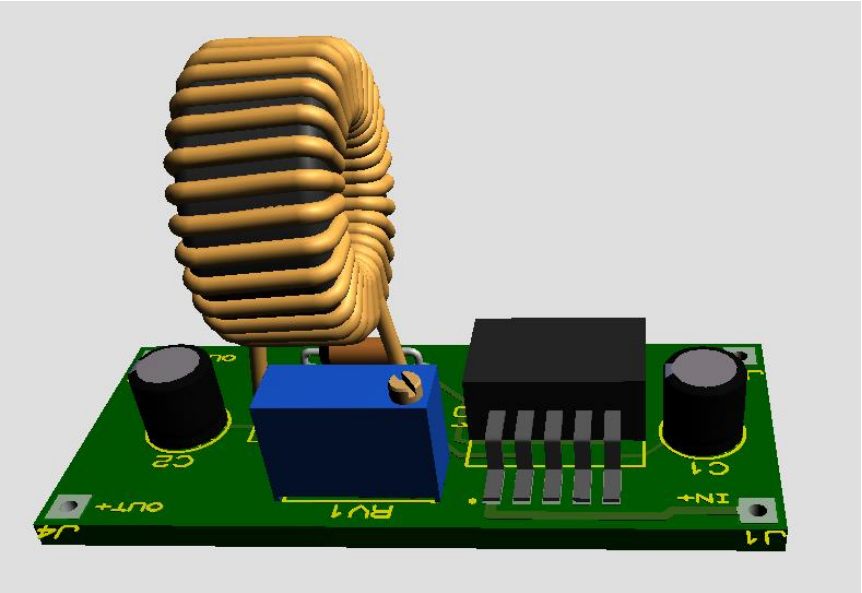


Figura 3. 9: Diseño 3D del convertidor reductor.
Elaborado por: Autor.

3.1.3. Controlador Particle Photon.

Particle Photon tiene un potente microcontrolador STM32 ARM Cortex M3 como cerebro y un chip Broadcom BCM43362 Wi-Fi como conexión a Internet. Con 18 pines GPIO mixtos y un IDE basado en web similar al IDE de Arduino, se puede conectar fácilmente el proyecto. La placa Particle Photon tiene un LED RGB y dos botones “Configuración” y “Reinicio”, que se pueden cambiar entre diferentes modos para ayudar a depurar el proyecto. Particle es una plataforma de código abierto, por lo que se puede acceder a todos sus esquemas y código para mejorar fácilmente el producto. SparkFun aprovechó esto y creó un Photon RedBoard, que le da a Photon el tamaño de un Arduino, para que pueda usar su escudo Arduino favorito con el servicio Particle.

Photon en sí es solo una placa de conexión para el módulo Wi-Fi P0, que contiene un microcontrolador y un chip Wi-Fi, pero sin antena. RedBoard usa el módulo P1, que es P0 con antena incorporada. Ambos están preinstalados con firmware de Particle y pueden acceder a sus servicios en la

nube, pero requieren una placa de conexión personalizada para acceder a todos los pines (Pancoast, 2016).

Gracias al Particle Photon y su plataforma web se pudo crear el programa que controlará el motor del alimentador automático por medio de wifi. También se tiene la ventaja que se puede reescribir el código sin necesidad de desconectar el controlador, simplemente se carga el código mediante wifi y el controlador se reiniciará con el nuevo programa. Este controlador también es muy conveniente por su tamaño reducido el cual ayuda a limitar el tamaño de la caja de electrónicos, lo que conlleva a que se reduce el peso del producto final. La figura 3.10 muestra el microcontrolador Particle Photon utilizado en este proyecto.

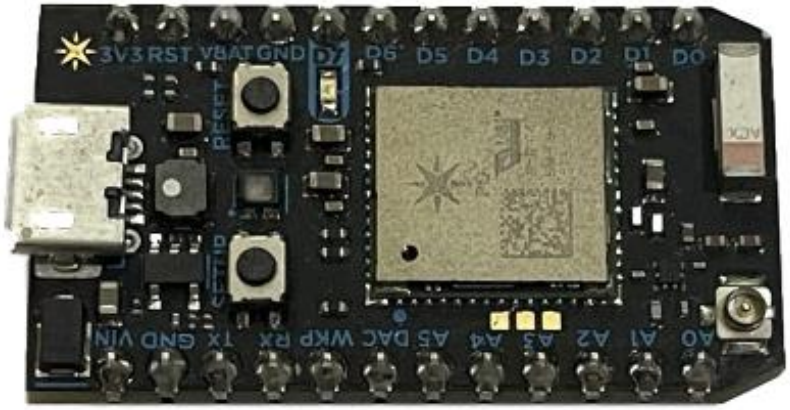


Figura 3. 10: Controlador Particle Photon.
Elaborado por: Autor.

3.1.4. Panel solar y Controlador del panel solar.

Uno de los objetivos de este proyecto es aplicar una fuente de energía renovable con el propósito de que el alimentador automático sea amigable con el medio ambiente, pero también se requiere de esta fuente de energía renovable dado que el alimentador estará ubicado la mayoría del tiempo en el agua, de esta manera complicando el uso de una fuente de energía conectada a alguna línea de energía no renovable. En cambio, se optó por utilizar una fuente de energía que permita cargar la batería que alimenta el motor sin necesidad de un cableado submarino.

Para lograr este objetivo se utilizó un panel solar de 20W, el cual contiene 36 células solares conectadas en serie para alcanzar la potencia

deseada. Este panel está compuesto por celular solares monocristalinas las cuales mejoran la eficiencia del panel solar, permitiendo al panel llegar a un voltaje máximo de salida de 18V. Las dimensiones del panel son de 420x335x17 milímetros, su peso es de 2.0 kilogramos, y está construido con cristal temperado, anti reflectante, de alta transparencia y con un marco de aluminio resistente capaz de soportar todo tipo de condiciones climáticas (SunEnergise, 2021). La figura 3.11 muestra el panel solar marca Sun Energise utilizado en este proyecto.

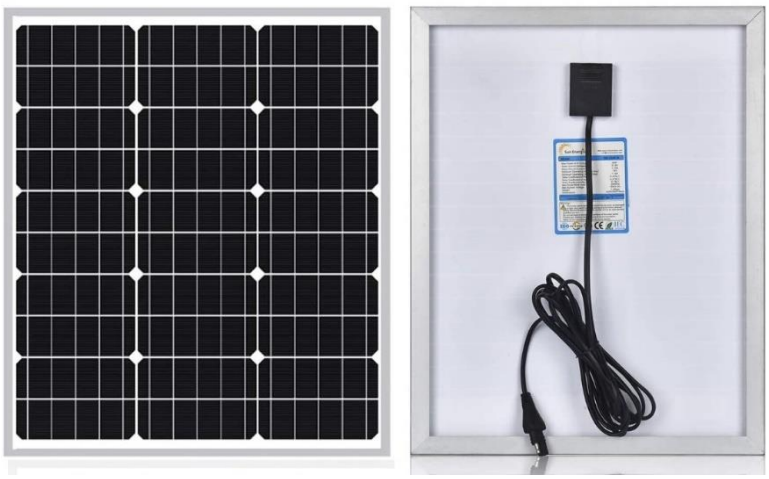


Figura 3. 11: Panel solar monolítico 20W.
Fuente: **(SunEnergise, 2021)**.

También se utilizó un control de carga inteligente para mejorar la eficiencia del panel solar. Este controlador de carga inteligente tiene un chip MCU inteligente integrado y un algoritmo de carga de tres etapas mejorado para regular toda la carga y proteger la batería. También tiene funciones como la protección inteligente contra polaridad inversa, sobrecarga, cortocircuito y corriente inversa, sobrecarga, sobre descarga, etc. Todas estas funciones de protección inteligente aseguran la protección completa de la batería que alimentará al motor y los circuitos de alimentador automático.

El controlador tiene un interfaz simple que contiene la corriente de carga digital PV y la pantalla de voltaje de la batería. Con esta interfaz se puede controlar el estado de funcionamiento del panel solar y comprender mejor las condiciones de la batería que debe estar bien protegida. También incluye un ajuste de horas de trabajo de acuerdo con los requisitos personales del usuario de esta manera haciendo la el uso del panel práctico y útil

(SunEnergise, 2021). La figura 3.12 muestra el controlador del panel solar marca Sun Energise utilizado en este proyecto.

Figura 3. 12: Controlador del panel solar.
Fuente: **(SunEnergise, 2021)**.

3.1.5. Diagrama de conexiones.

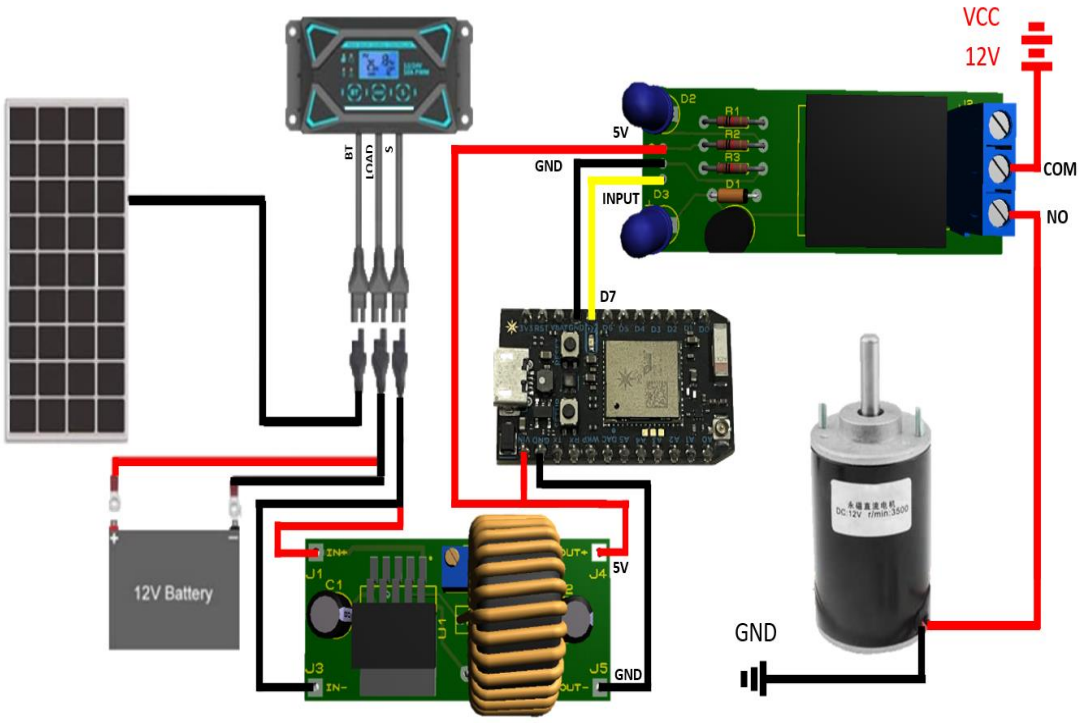


Figura 3. 13: Diagrama de conexiones.
Elaborado por: Autor.

En la figura 3.13 se muestra un diagrama en el cual se pueden apreciar las conexiones de todos los elementos necesarios para el control de la máquina. Se puede simplificar estas conexiones haciendo una tarjeta general en la que se encuentren todos los componentes juntos, esto es una idea a futuro después de realizar algunas pruebas.

Se empieza conectando la batería de 12v al controlador del panel en el conector nombrado “Load”, luego se conecta el panel al controlador en el conector nombrado “BT”. Una vez conectados estos componentes, se continúa conectando el convertidor reductor al conector del controlador nombrado “S” en las entradas IN+ e IN-, después se conecta las salidas OUT+ y OUT- al Vin y GND del Particle Photon respectivamente, y también se conectan 5v al módulo relé en el pin de 5v, luego se conecta a tierra el pin GND del módulo de relé y el pin Input se lo conecta al pin D7 del Particle Photon. Se continúa conectando el borne COM (Común) del relé a la fuente de 12V, luego se conecta el borne NO (Normalmente abierto) al positivo del motor y finalmente se conecta el negativo del motor a tierra.

3.1.6. Tanque para alimento.

La función principal del tanque es el almacenamiento del balanceado para camarón, el cual debe estar apropiadamente hermetizado para que no se genere humedad dentro del mismo, pero también cumple otra función la cual viene de la propiedad de su color para refractar la luz del día y así conservar mejor el alimento. La refracción es un fenómeno en el cual se da que, al pasar de un medio material a otro medio material, la luz que se propaga en forma de ondas cambia su velocidad. Este fenómeno es aprovechado usando un color claro en el tanque de alimento para que mantenga fresco el balanceado. La figura 3.14 muestra el tanque para almacenar el balanceado de camarón utilizado en este proyecto.

Figura 3. 14: Tanque para el balanceado.
Elaborado por: Autor.

3.1.7. Sistema de aspersión de alimento

Figura 3. 15: Sistema de aspersión de alimento.
Elaborado por: Autor.

En la figura 3.15 se muestra el diseño del sistema de aspersión de balanceado del alimentador automático ecológico. Este sistema está compuesto por 3 componentes, los cuales son el embudo, el tubo de aspersión y el acoplamiento para que el tubo se junte con el motor. Primero se utiliza un embudo que está conectado al tanque de alimento utilizando fibra de vidrio en la unión, este embudo tiene la función de controlar el flujo de alimento que va hacia el tubo de aspersión, ya que si el flujo es muy grande el alimento se quedará atascado y no podrá llegar al siguiente punto. El tubo de aspersión está compuesto por tubos pvc de ¾ de pulgada y uno de media pulgada. Los tubos de ¾ de pulgada se utilizan para darle dirección de vuelo al alimento, se utilizan dos tubos rectos y dos tubos con ángulo para darle más distancia de alcance al alimento. Finalmente se une el tubo de media pulgada con un acoplamiento que permite unir el eje rotatorio del motor con el tubo de aspersión, este acoplamiento varía con respecto a las medidas del eje rotatorio del motor que usemos.

3.2. Diseño de la estructura.

Figura 3. 16: Estructura del alimentador automático.
Elaborado por: Autor.

Para que el alimentador automático pueda soportar todas las condiciones climáticas y cualquier tipo de golpes fue necesario la construcción y diseño de una estructura metálica capaz de soportar tales condiciones. En la construcción de la estructura se utilizaron varios materiales rígidos como el tubo de una pulgada, pletina de 1 pulgada y media por 1/8, tubo pvc ¾, ángulo 45 grados pvc ¾, fibra de vidrio, ángulo perimetral, rodamiento metálico, adaptador para final de motor y soldadura. La figura 3.16 muestra la estructura del alimentador automático ecológico ensamblada.

Primero se diseñó un modelo 3D de la estructura para tener claras las medidas de cada material usando el programa SolidWorks, el cual permite usar medidas reales y apreciar el funcionamiento de la maquina con una simulación sin tener que gastar recursos físicos. Luego se cortaron y soldaron todas las piezas que se diseñaron en el modelo 3D a la medida. La figura 3.17 muestra la estructura del alimentador automático ecológico ensamblada en 3D. Se pueden apreciar todas las medidas exactas en los diagramas de los anexos 1,2 y 3.

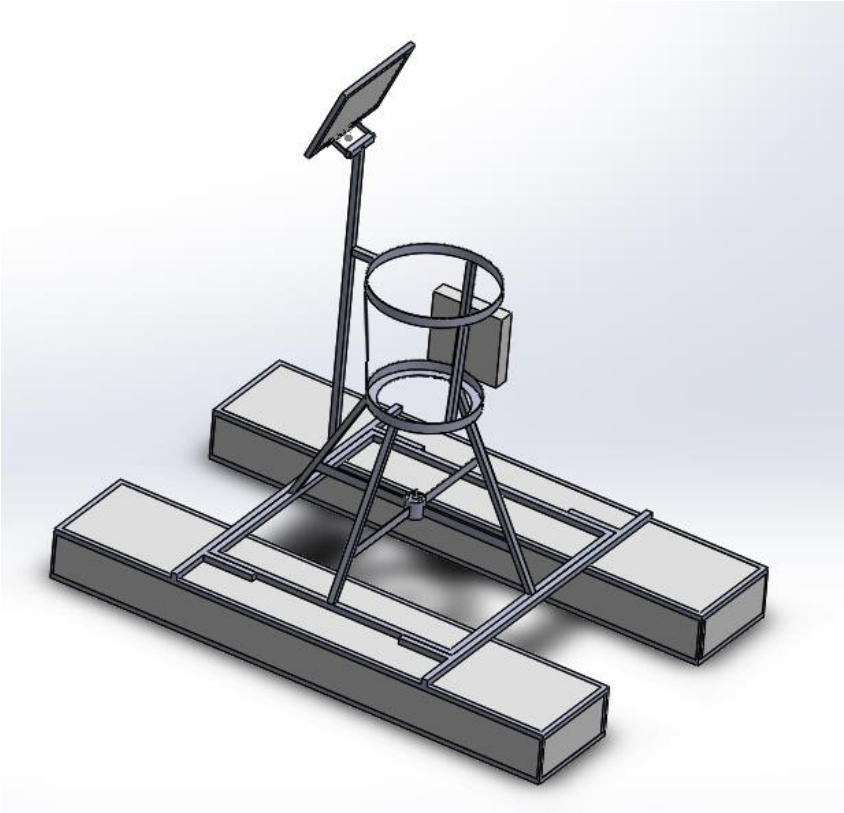


Figura 3. 17: Estructura del alimentador automático en 3D
Elaborado por: Autor.

Para la base del alimentador se utilizó tubo cuadrado de 1 pulgada, con el cual se hizo un cuadrado de 1m por cada lado. Luego con el mismo tubo de 1 pulgada, se hicieron los parantes que sostienen la estructura para el tanque, los cuales tienen un largo de 58cm. Una vez cortados los parantes, se continuó creando la base del motor, la cual está hecha de pletina de pulgada y media por 1/8 y tiene 40 cm de diámetro. Después se soldó los parantes a la base cuadrada de 1m y a la base del tanque.

Luego para sostener el tanque desde la parte de arriba se hizo un círculo de pletina de 47cm de diámetro, el cual es sostenido por un marco de tres soportes de 46cm de largo, el mismo que a su vez va soldado con la base de soporte inferior del tanque. Una vez terminado esto, se prosiguió a realizar los soportes del motor, los cuales tienen 38cm de largo y están hechas de la misma pletina. Estos soportes se juntan con un recubrimiento para el motor hecho con la misma pletina, el cual tiene 6cm de diámetro. La figura 3.18 muestra la estructura base del alimentador automático ecológico en 3D. Se puede apreciar un diagrama con todas las medidas en el anexo 1.

Figura 3. 18: Estructura base en 3D.
Elaborado por: Autor.

Teniendo la base de la estructura ya diseñada se puede dar paso a diseñar el soporte para el panel solar y la caja que contiene los electrónicos. La caja para los electrónicos está hecha de metal y es de 26x26 cm con 8cm de profundidad, esta misma se suelda a la estructura base con unas pletinas de 15cm de largo que sirven como soporte para la caja. Por otro lado, el soporte para el panel solar está hecho de tubo de 1 pulgada y tiene una altura de 1.10m, también lleva otro tubo horizontal de 14cm, el cual forma una T para el soporte del panel. Esta T lleva dos pernos soldados a los lados, junto con unas tuercas de presión y arandelas y también se le hizo un hueco de 1cm de diámetro en la mitad para pasar el cable de panel.

En el panel también se hizo un soporte que va a juntarse con la T de la estructura. Este soporte sirve para mover el panel solar al ángulo que el usuario crea apropiado por la posición del sol. El soporte tiene dos pletinas soldadas en un ángulo de 90 grados para juntarse mediante dos pernos al marco del panel. También tiene soldadas dos pletinas de 7cm de alto que se encuentran a 2.5cm de cada borde del soporte, estas pletinas tienen una forma de U en el tope para poder juntarse con el otro soporte encontrado en

la estructura base del alimentador. La figura 3.19 muestra el deseño del soporte para el panel solar del alimentador automático ecológico. Se puede apreciar un diagrama con las medidas de los dos soportes del panel en los anexos 1 y 2.

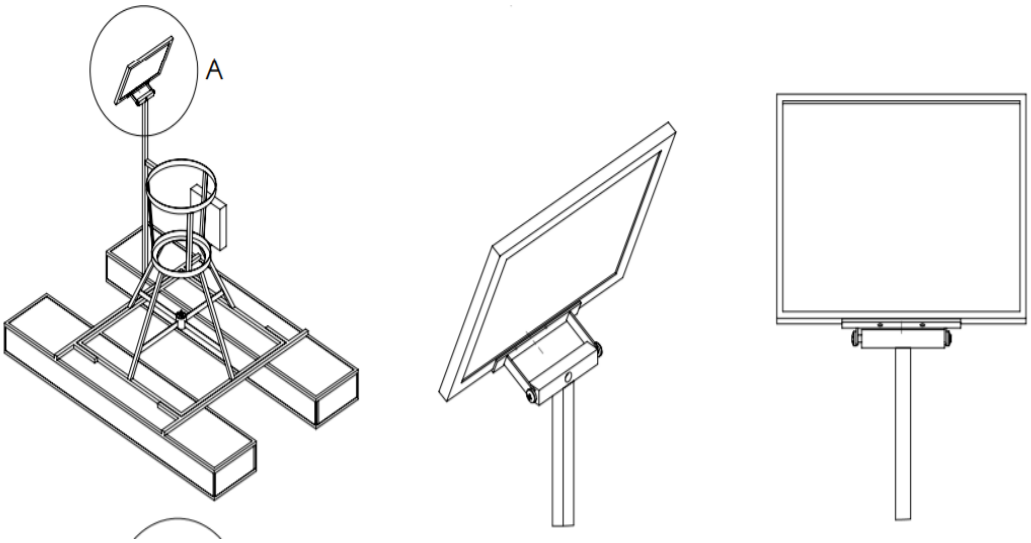


Figura 3. 19: Estructura del soporte para el panel.
Elaborado por: Autor.

Una vez completa la estructura base se puede ir al siguiente paso, el cual es diseñar la estructura para los flotadores. Para esta estructura se usaron dos bloques de poliestireno expandido de 200x40x20cm cada uno. Primero se hizo una cobertura para los bloques usando ángulo perimetral, por cada bloque se utilizaron 8 metros del ángulo perimetral para cubrir el largo de los cuatro lados (2m cada uno), también se utilizaron 4 cortes de 40cm cada uno para los bordes del ancho y por último se utilizaron 4 cortes más del ángulo perimetral para cubrir los bordes del alto del bloque. Esto se repitió para cada bloque y se juntaron los ángulos con remaches.

Luego se hizo una estructura de tubo cuadrado de 1 pulgada para que se junte con la estructura base hecha anteriormente. Primero se hicieron dos cortes largos de 1.37m para remacharlos a los extremos de la estructura de los bloques de poliestireno. Estos cortes se colocan horizontalmente dejando 1 metro de distancia entre sí y 0.5 metros de distancia con los extremos de largo. Finalmente se sueldan 4 cortes de 20cm del mismo tubo, los cuales van ubicados de cada lado en un ángulo de 90 grados dejando 1 metro de espacio entre sí para que la base quepa perfectamente. La figura 3.20 muestra el

diseño de los flotadores del alimentador automático ecológico en 3D. Se puede apreciar un diagrama con todas las medidas en el anexo 3.

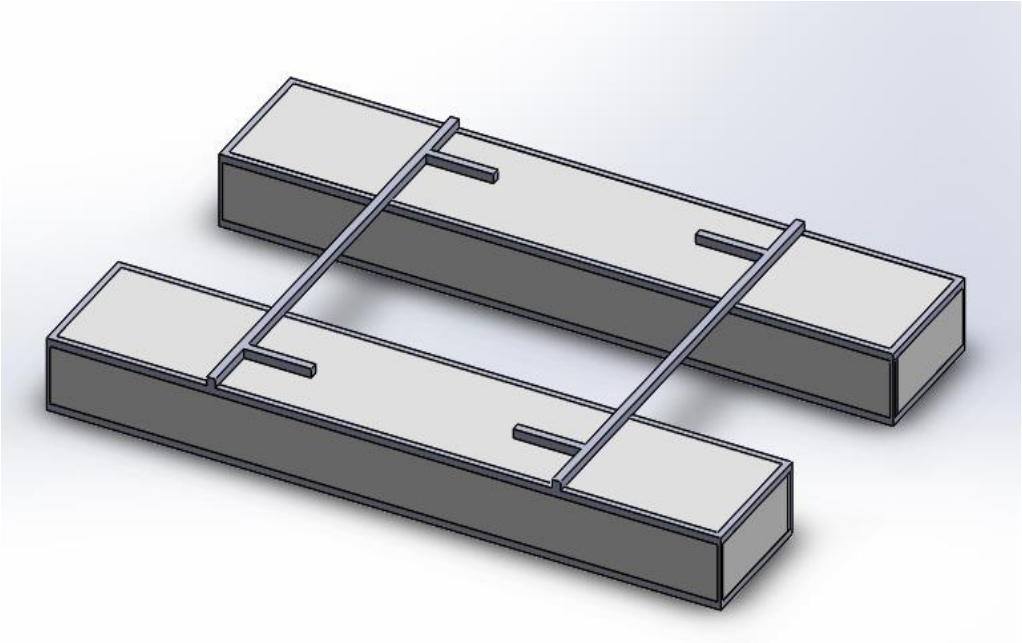


Figura 3. 20: Estructura de los flotadores en 3D
Elaborado por: Autor.

3.3. Código de programación y diseño de App.

El objetivo de este proyecto es construir un alimentador automático que se pueda controlar remotamente utilizando de wifi y por medio de una aplicación con una interfaz amigable con el usuario. Esto se logra programando un código el cual va cargado al microcontrolador del Particle Photon y este mismo permite la comunicación con la app.

El código tiene que ser capaz de permitir la programación por horas del alimentador automático, es decir, que permita elegir con qué frecuencia (horas) se deberá encender el alimentador automático, también debe usar una variable que pueda cambiar el tiempo de duración de encendido del motor, y permitir la impresión de valores en la interfaz de la app. También tiene que ser capaz de interactuar con la aplicación y que responda correctamente al interfaz del usuario. Por otro lado, la aplicación tiene que ser capaz de, mediante una interfaz amigable con el usuario, permitir al propietario elegir la frecuencia de horas y el tiempo que permanece encendido el alimentador cuando se cumple el horario de activación, y debe enviar notificaciones cuando el motor se encienda automáticamente.

3.3.1. Programación en Particle IDE.

Para la programación del código que gobierna al alimentador automático se utilizó la plataforma web Particle IDE. Esta plataforma permite, en conjunto con el controlador Particle Photon, enviar un código al microcontrolador de manera remota utilizando wifi, no como en la mayoría de los microcontroladores que se debe conectar directamente a una computadora para cargar el programa. A continuación, se explicará la función de los comandos utilizados en este programa.

```
1  SYSTEM_THREAD(ENABLED);
2
3  #include <blynk.h>
4
5  #define DEBUG_PRINT(...) { Particle.publish( "DEBUG", String::format(__VA_ARGS__) ); }
6
7  char auth[] = "a512duMvv7s2v25yhW6oeWgEqYIjStpABCD";
8
9  WidgetLED appPumpStatusLED(V4);
10 char debugBuffer[50];
11 bool remoteOnPb = 0;
12 bool PumpOnReq = 0;
13 bool autoPumpStartActivated = 0;
14 bool bypassSwitch;
15 Timer remotePbAutoOffTimer(120000, autoRemotePbOff, 1);
16
17 int startTimeHour = 6;
18 int startTimeMinute = 0;
19 int startTimeSecond = 0;
20 int stopTimeHour;
21 int stopTimeMinute;
22 int stopTimeSecond;
23
24 long runTimeDurationSeconds = 60;
25 long runTimeDurationSeconds1 = 0;
26
27 bool runHowOftenMode = 0;
28 int runHowOftenDaySelected = 1;
29 int runHowOftenHourSelected = 3;
30
31 int manualPushButton = D3;
32 int Pump = D7;
```

Figura 3. 21: Código parte 1.
Elaborado por: Autor.

En la figura 3.21 se muestra la primera parte del código en la cual se definió las variables a usar y las librerías. Se empieza con el comando “SYSTEM_THREAD(ENABLED)” el cual permite que cuando se inicie el Particle Photon, no se tenga que esperar a una conexión wifi antes de que el código pueda comenzar a ejecutarse. Luego se incluye la librería “blynk.h” la cual permitirá conectar el código con la app y se define un comando para poder depurar las variables. En la línea 7 se coloca el código “Auth” que

entrega Blynk al crear la aplicación para que solo el propietario la pueda controlar. Luego desde la línea 9 hasta la línea 32 se definen algunas variables explicadas a continuación.

- WidgetLED appPumpStatusLED(V4): Widget para ver el estado del LED en Blynk para el motor.
- Char debugBuffer[50]: Variable reusable para depurar tipo char.
- Bool remoteOnPb = 0: Variable tipo bool usada para activar el motor manualmente.
- Bool PumpOnReq = 0: Variable tipo bool usada para activar la salida al motor.
- Bool autoPumpStartActivated = 0: Variable tipo bool usada para reconocer si el motor fue activado de forma manual o automáticamente.
- Bool bypassSwitch: Variable tipo bool usada para implementar a futuro un botón de encendido manual en la estructura de la máquina.
- Timer remotePbAutoOffTimer(120000, autoRemotePbOff, 1): Temporizador usado para apagar automáticamente el motor.
- Int startTimeHour = 6: Variable tipo int usada para definir la hora en que el motor se activa.
- Int startTimeMinute = 0: Variable tipo int usada para definir el minuto en que el motor se activa.
- Int startTimeSecond = 0: Variable tipo int usada para definir el segundo en que el motor se activa.
- Int stopTimeHour: Variable tipo int usada para definir la hora en que el motor se desactiva.
- Int stopTimeMinute: Variable tipo int usada para definir el minuto en que el motor se desactiva.
- Int stopTimeSecond: Variable tipo int usada para definir el segundo en que el motor se desactiva.
- Long runTimeDurationSeconds = 60: Variable tipo long usada para definir la frecuencia de horas en las que se activa el motor.
- Long runTimeDurationSeconds1 = 0: Variable tipo long usada para definir cuanto tiempo dura encendido el motor.

- Bool runHowOftenMode = 0: Variable tipo bool usada para definir con qué frecuencia se activa el motor.
- int runHowOftenDaySelected = 1: Variable tipo Int usada para definir cada cuantos días se enciende el motor.
- Int runHowOftenHourSelected = 3: Variable tipo Int usada para definir la frecuencia de horas en las que se activa el motor.
- Int manualPushButton = D3: Variable tipo Int usada para definir el pin del botón manual de encendido.
- Int Pump = D7: Variable tipo Int usada para definir el pin que activa el módulo de relé.

```
33
34  BLYNK_WRITE(V1)
35  {
36   int pinValue = param.asInt();
37   if (pinValue == 1)
38   PumpManualRemotePb("on");
39   else
40   PumpManualRemotePb("off");
41  }
42
43  BLYNK_WRITE(V0) {
44   long startTimeInSecs = param[0].asLong();
45   TimeInputParam t(param);
46
47   if (t.hasStartTime())
48   {
49
50   int addr = 1;
51   uint8_t value = t.getStartHour();
52   EEPROM.put(addr, value);
53   startTimeHour = t.getStartHour();
54
55   addr = 2;
56   value = t.getStartMinute();
57   EEPROM.put(addr, value);
58   startTimeMinute = t.getStartMinute();
59
60   addr = 3;
61   value = t.getStartSecond();
62   EEPROM.put(addr, value);
63   startTimeSecond = t.getStartSecond();
64   }
65  }
```

Figura 3. 22: Código parte 2.
Elaborado por: Autor.

En la figura 3.22 se muestra la segunda sección del código en la cual se crean rutinas para los pines virtuales V0 y V1 (perteneciente al botón de control wifi en la app). En esta sección también se usan comandos para llamar a la EEPROM del Particle Photon y almacenar datos de hora, minuto y segundo, en caso de pérdida de energía. A continuación, se explican en detalle las líneas de código.

```
BLYNK_WRITE(V1)
{
 int pinValue = param.asInt();
 if (pinValue == 1)
 PumpManualRemotePb("on");
 else
 PumpManualRemotePb("off");
}
```

El comando “BLYNK_WRITE(V1)” crea una rutina para el pin virtual V1, el cual le pertenece al control manual por wifi en la app. Esta rutina empieza asignando un valor entrante al pin V1 con el comando pinValue = param.asInt, luego se utiliza una condicional “If” que plantea que si el valor del pin V1 está en alto (==1) se active el motor, y luego usa una condicional “else” que plantea que, si el “If” no se cumple, entonces el motor permanece desactivado.

```
BLYNK_WRITE(V0)
 {
 long startTimeInSecs = param[0].asLong();
 TimeInputParam t(param);
 if (t.hasStartTime())
 {
int addr = 1;
 uint8_t value = t.getStartHour();
 EEPROM.put(addr, value);
 startTimeHour = t.getStartHour();
 addr = 2;
 value = t.getStartMinute();
 EEPROM.put(addr, value);
 startTimeMinute = t.getStartMinute();
 addr = 3;
 value = t.getStartSecond();
 EEPROM.put(addr, value);
 startTimeSecond = t.getStartSecond();
```

Luego se crea una rutina para el pin virtual V0, la que consiste en leer la información del tiempo en que se activa el motor. Dentro de esta rutina se encuentra una condicional “If”, la que consiste en que, si hay valores seteados en la app de inicio para el motor, estos valores se guarden en la EEPROM del Particle Photon en caso de pérdida de energía. Primero se toma el valor de las horas con el comando t.getStartHour(), luego se lo almacena en la variable startTimeHour, la cual se va directamente a la EEPROM. Este proceso se repite con los minutos y segundos usando los comandos t.getStartMinute() y t.getStartSecond() y las variables startTimeMinute y startTimeSecond respectivamente (Particle, 2021).

```
BLYNK_WRITE(V5) {
 int durationPinValue = param.asInt();
 if (durationPinValue > 0) {
 runTimeDurationSeconds = durationPinValue;

 int addr = 4;
 uint8_t value = runTimeDurationSeconds;
 EEPROM.put(addr, value);
 }
 else {
 runTimeDurationSeconds = 60;
 }
}
BLYNK_WRITE(V7) {
 int durationPinValue1 = param.asInt();
 if (durationPinValue1 > 0) {
 runTimeDurationSeconds1 = durationPinValue1;

 int addr = 5;
 uint8_t value = runTimeDurationSeconds1;
 EEPROM.put(addr, value);
 }
 else {
 runTimeDurationSeconds1 = 60;
 }
}
```

Figura 3. 23: Código parte 3.
Elaborado por: Autor.

En la figura 3.23 se muestra tercera sección del código en la cual se crea una rutina para el pin virtual V5, que consiste en leer el valor ingresado en la app de la frecuencia de horas en que se activa el motor. También se crea una rutina para el pin virtual V7, la cual consiste en leer el valor ingresado en la app del tiempo de duración de encendido del motor. A continuación, se explican en detalle las líneas de código.

```
BLYNK_WRITE(V5) {
 int durationPinValue = param.asInt();
 if (durationPinValue > 0) {
 runTimeDurationSeconds = durationPinValue;
 int addr = 4;
 uint8_t value = runTimeDurationSeconds;
 EEPROM.put(addr, value);
 }
 else {
 runTimeDurationSeconds = 60;
 }
```

El comando “BLYNK_WRITE(V5)” crea una rutina para el pin virtual V5, el cual le pertenece al control del temporizador en la app. Esta rutina empieza asignando un valor entrante al pin V5 con el comando durationPinValue = param.asInt(), luego se utiliza una condicional “If” que plantea que, si el valor ingresando en el temporizador es mayor a 0, entonces se actualice el nuevo valor ingresado al temporizador. Finalmente se ingresa el valor escogido dentro de la app en la EEPROM del Particle Photon con el comando EEPROM.put(addr, value) para que en caso de una pérdida de energía se pueda recuperar este valor.

```
BLYNK_WRITE(V7) {
 int durationPinValue1 = param.asInt();
 if (durationPinValue1 > 0)
 {
 runTimeDurationSeconds1 = durationPinValue1;

 int addr = 5;
 uint8_t value = runTimeDurationSeconds1;
 EEPROM.put (addr, value);
 }
 else {
 runTimeDurationSeconds1 = 60;
```

Luego se crea una rutina para el pin virtual V7, el cual pertenece al control de tiempo de duración de encendido del motor encontrado en la app. Primero se asigna un valor entrante al pin V7 con el comando durationPinValue1 = param.asInt(), luego se crea una condicional “If”, la cual plantea que, si el valor ingresado en la app es mayor a 0, entonces se actualice el nuevo valor en el tiempo de duración de encendido del motor. Para finalizar esta rutina se ingresa el valor escogido en la app en la EEPROM del Particle Photon con el comando EEPROM.put(addr, value) para que en caso de una pérdida de energía se pueda recuperar este valor.

```
BlynkTimer timer;
unsigned int myServerTimeout = 3500;
unsigned int functionInterval = 100;
unsigned int blynkInterval = 25000;
void setup() {

 Time.zone(-5);
 pinMode(manualPushButton, INPUT_PULLDOWN);
 pinMode(Pump, OUTPUT);

 int addr = 1;
 uint8_t value;
 EEPROM.get(addr, value);
 if(value != 0xFFFF)
 startTimeHour = value;

 addr = 2;
 value;
 EEPROM.get(addr, value);
 if(value != 0xFFFF)
 startTimeMinute = value;

 addr = 3;
 value;
 EEPROM.get(addr, value);
 if(value != 0xFFFF)
 startTimeSecond = value;

 addr = 4;
 value;
 EEPROM.get(addr, value);
 if(value != 0xFFFF)
 runTimeDurationSeconds = value;

  addr = 5;
 value;
 EEPROM.get(addr, value);
 if(value != 0xFFFF)
 runTimeDurationSeconds1 = value;

 PumpOnReq = PumpControl("off");

 Particle.function("PumpRemotePb", PumpManualRemotePb);

 timer.setInterval(functionInterval, mainFunction);
 timer.setInterval(blynkInterval, checkBlynk);

 Blynk.config(auth);
}
```

Figura 3. 24: Código parte 4.
Elaborado por: Autor.

En la figura 3.24 se muestra la cuarta sección del código en la cual se definen comandos para el tiempo de espera de conexión wifi y la app Blynk, también se crea el void setup en el que se define la zona horaria, las señales de entrada y salida, se crean comandos para recuperar los tiempos de inicio y parada guardados de EEPROM, se definen funciones solo para cuando el

sistema se pone en marcha, se activan los timers y se configura el código Auth de la app. A continuación, se explican en detalle las líneas de código.

```
BlynkTimer timer;
unsigned int myServerTimeout = 3500;
unsigned int functionInterval = 100;
unsigned int blynkInterval = 25000;
```

Primero se usa el comando “BlynkTimer timer” que permite realizar acciones periódicas en el contexto del loop principal. Luego con la variable “myServerTimeout” se define el tiempo de espera de conexión del servidor a 3.5 segundos, después se define la frecuencia de llamada de función con la variable declarada como unsigned Int “functionInterval”, y en la siguiente línea se define la frecuencia del servidor a 25 segundos con la variable “blynkInterval”.

```
void setup() {

 Time.zone(-5);

 pinMode(manualPushButton, INPUT_PULLDOWN);
 pinMode(Pump, OUTPUT);

 int addr = 1;
 uint8_t value;
 EEPROM.get(addr, value);
 if(value != 0xFFFF)
 startTimeHour = value;

 addr = 2;
 value;
 EEPROM.get(addr, value);
 if(value != 0xFFFF)
 startTimeMinute = value;
```

```
 addr = 3;

value;
EEPROM.get(addr, value);
if(value != 0xFFFF)

startTimeSecond = value;

addr = 4;

value;
EEPROM.get(addr, value);
if(value != 0xFFFF)

runTimeDurationSeconds = value;

 addr = 5;

value;
EEPROM.get(addr, value);
if(value != 0xFFFF)

runTimeDurationSeconds1 = value;
```

En esta parte del código se define el void setup (), el cual es una rutina que se requiere que se ejecute solo cada vez que se ponga en marcha el Particle Photon, es decir, cada vez que se este se vuelva a energizar si llega a ocurrir una pérdida de suministro eléctrico.

Se empieza el void setup definiendo la zona horaria de la región con el comando Time.zone(-5), para que el reloj interno pueda estar con el horario correcto. Luego se definen las señales de entrada y salida que entrega o recepta el Particle Photon, como son el botón de encendido manual que se planea usar a futuro y la variable que da la señal para encender el motor. El

manualPushButton se define como INPUT ya que es una señal que viene de afuera para adentro, y la variable waterPump se define como OUTPUT ya que esta envía una señal de adentro hacia fuera para que se encienda el motor.

Luego de definir las señales de entrada y salida, se utilizan comandos para recuperar la información receptada en la EEPROM para que los valores ingresados anteriormente se reflejen. Se usan los mismos comandos para reflejar los datos de hora, minuto, segundo, temporizador y tiempo de duración. Primero se lama a cada addr, luego con el comando EEPROM.get(addr, value) se recupera el valor ingresado anteriormente, y finalmente se otorga el valor recuperado a cada variable utilizada para recopilar estos datos en la EEPROM escribiendo la variable utilizada seguida de “= value”.

```
 PumpOnReq = PumpControl("off");

 Particle.function("PumpRemotePb", PumpManualRemotePb);

 timer.setInterval(functionInterval, mainFunction);

 timer.setInterval(blynkInterval, checkBlynk);

 Blynk.config(auth);

}
```

Siguiendo con el void setup, se define que la variable “PumpOnReq” es igual a “PumpControl” y que esta variable estará en bajo cada vez que se vuelva a iniciar el programa. Luego se define la función para encender el motor remotamente con el comando Particle.function("waterPumpRemotePb", waterPumpManualRemotePb). Siguiente a esto se definen los timers utilizados para ayudar a prevenir el bloqueo de bucles si se pierde la conexión a Internet, y finalmente se ingresa el comando Blynk.config(auth) para reconocer el código Auth de la app.

```
void loop() {

 if (Blynk.connected()) {
 Blynk.run();
 }

 timer.run();
}

void mainFunction(){
 bypassSwitch = digitalRead(manualPushButton);

 if (Particle.connected())
 Particle.function("PumpRemotePb", PumpManualRemotePb);


 int currentHour = Time.hour();
 int currentMinute = Time.minute();
 int currentSecond = Time.second();

 if (Blynk.connected()) {

 Blynk.virtualWrite(V2, Time.format(Time.now(), "%I:%M:%S %p"));

 char amPM[3]; amPM[1] = 'M';
 if (Time.isAM((startTimeHour + 5) * 3600)) amPM[0] = 'A'; else amPM[0] = 'P';
 sprintf(debugBuffer, "%d Seg/s", runTimeDurationSeconds1);
 Blynk.virtualWrite(V8, debugBuffer);

 sprintf(debugBuffer, "%d Hora/s", runTimeDurationSeconds);
 Blynk.virtualWrite(V6, debugBuffer);
 }
```

Figura 3. 25: Código parte 5.
Elaborado por: Autor.

En la figura 3.25 se muestra la quinta sección del código en la cual se crean dos voids, los cuales son el void loop() y el void mainFunction(). En el void loop es muy breve y solo se definen comandos para iniciar la conexión con Blynk y encender el timer usado. Por otro lado, el void mainFunction es un poco más extenso por lo cual en esta sección se explica solo una parte de este void, la siguiente parte se explicará en la próxima sección. A continuación, se explican en detalle las líneas de código.

```
void loop() {
 if (Blynk.connected()) {
 Blynk.run();
 }
 timer.run();
}
```

El void loop es una rutina que se va a repetir siempre y en todo momento, es por esto que en este void se utiliza una condicional “If” que plantea que, si Blynk está conectado, entonces sean funcionales todo lo que se encuentra en

la app. Luego se pone en marcha el timer utilizado con el comando timer.run y se finaliza el void loop para empezar el siguiente void.

```
void mainFunction() {
 bypassSwitch = digitalRead(manualPushButton);

 if (Particle.connected())
 Particle.function("PumpRemotePb", PumpManualRemotePb);

 int currentHour = Time.hour();
 int currentMinute = Time.minute();
 int currentSecond = Time.second();

 if (Blynk.connected()) {

 Blynk.virtualWrite(V2, Time.format(Time.now (), "%I:%M:%S %p"));

 char amPM[3]; amPM[1] = 'M';
 if (Time.isAM((startTimeHour + 5) * 3600)) amPM[0] = 'A'; else amPM[0] = 'P';
 sprintf (debugBuffer, "%d Seg/s", runTimeDurationSeconds1);
 Blynk.virtualWrite(V8, debugBuffer);

 sprintf (debugBuffer, "%d Hora/s", runTimeDurationSeconds);
 Blynk.virtualWrite(V6, debugBuffer);
 }
```

Luego se crea el void mainFunction que es la rutina que se va a repetir siempre y en todo momento, la cual contiene la lógica del correcto funcionamiento del temporizador y el tiempo de duración del encendido del motor. También define el texto que aparece en los displays de la app y otros comandos que se explican a continuación.

Este void comienza chequeando si el botón de activación manual se encuentra en bajo o en alto, creando una variable “bypassSwitch” que lee el estado de este botón con el comando “= digitalRead(manualPushButton)”.

Una vez hecho esto, se crea una condicional “If” para verificar si el Photon está conectado al wifi para prevenir que se detenga la ejecución de este bucle y poder activar el motor remotamente. Después se crean 3 variables las cuales van a almacenar la hora, minuto y segundo actuales con los comandos Time.hour, Time.minute y Time.second respectivamente.

Luego se plantea otra condicional “If” que plantea que si Blynk está conectado envíe el formato de hora actual al display conectado al pin virtual V2 mediante el comando Blynk.virtualWrite. Dentro de esta condicional también se utilizan los comandos sprintf y Blynk.virtualWrite que envían los datos de temporizador y tiempo de duración de encendido del motor, los cuales se reflejan en los displays que se encuentran en la app configurados con los pines virtuales V5 y V8.

```
175
176   if (bypassSwitch == 0 && remoteOnPb == 0) {
177   if (PumpOnReq == 1) {
178
179   stopTimeHour = Time.hour((Time.now() + runTimeDurationSeconds));
180   stopTimeMinute = Time.minute((Time.now()+ runTimeDurationSeconds));
181   stopTimeSecond = Time.second((Time.now()+ runTimeDurationSeconds));
182
183
184   PumpOnReq = PumpControl("off");
185   autoPumpStartActivated = 1;
186
187
188   }
189   else {
190   if (autoPumpStartActivated) {
191   if (currentHour == stopTimeHour && currentMinute == stopTimeMinute && currentSecond == stopTimeSecond) {
192   PumpOnReq = PumpControl("on");
193   delay(runTimeDurationSeconds1*1000);
194    if (Blynk.connected())
195   Blynk.notify("Alimentador Automático Activado");
196
197   }
198   }
199   else {
200   PumpOnReq = PumpControl("off");
201   }
202   }
203   }
204   else if (bypassSwitch == 1) {
205
206   PumpOnReq = PumpControl("on");
207   autoPumpStartActivated = 0;
208
209   PumpManualRemotePb("off");
210   }
211   else if(remoteOnPb == 1) {
212
213   PumpOnReq = PumpControl("on");
214   autoPumpStartActivated = 0;
215   }
216   else {
217
218   PumpOnReq = PumpControl("off");
219   autoPumpStartActivated = 0;
220   }
221  }
```

Figura 3. 26: Código parte 6.
Elaborado por: Autor.

En la figura 3.26 se muestra la sexta sección del código la cual es la continuación del void mainFunction. Esta parte del void contiene los comandos y lógica matemática que va a regular el temporizador para que se active en la hora deseada y que el tiempo de duración también sea el indicado.

También se da solución a diferentes situaciones que podrían causar errores y, por lo tanto, impedir la ejecución del encendido del motor. A continuación, se explican en detalle las líneas de código.

```
if (bypassSwitch == 0 && remoteOnPb == 0) {
 if (PumpOnReq == 1) {

 stopTimeHour = Time.hour((Time.now () + runTimeDurationSeconds));
 stopTimeMinute = Time.minute((Time.now () + runTimeDurationSeconds));
 stopTimeSecond = Time.second((Time.now () + runTimeDurationSeconds));

 PumpOnReq = PumpControl("off");
 autoPumpStartActivated = 1;
 }

 else {
 if (autoPumpStartActivated) {
 if (currentHour == stopTimeHour && currentMinute == stopTimeMinute &&
currentSecond == stopTimeSecond) {
 PumpOnReq = PumpControl("on");
 delay(runTimeDurationSeconds1*1000);
  if (Blynk.connected())
 Blynk.notify("Alimentador Automático Activado");
 }
 }
 else {
 PumpOnReq = PumpControl("off");
 }
 }
 }
```

Primero se crea una condicional “If” la cual verifica si el motor está en modo manual o no, si no se encuentra en modo manual, entonces se da paso a la siguiente condicional “If”, en la cual se verifica si el motor está en modo

automático, si esto es real, entonces el motor no está encendido, así que se compara la hora actual con la hora de inicio ingresada. Después se utilizan las variables stopTimeHour, stopTimeMinute y stopTimeSecond, a las cuales se las define como la suma entre la hora actual y la multiplicación del tiempo ingresado del temporizador (El programa lo plantea en segundos) por 3600 (factor de conversión de segundos a horas). Esto significa que cuando la hora actual sea igual a la hora ingresada del temporizador, entonces se ponga en bajo la variable pumpOnReq utilizando waterPumpOnReq = waterPumpControl("off") y la variable autoPumpStartActivated sea ponga en alto, para dar paso al “else” que activa el motor del alimentador.

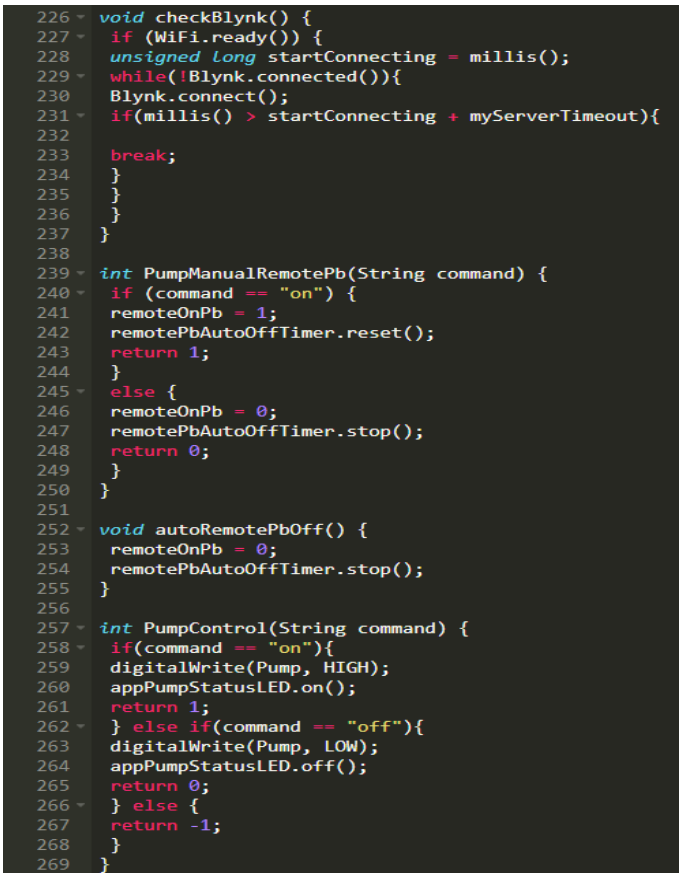
Luego se crea un “else” que se activa en el caso contrario al “If” anterior, es decir, cuando la variable pumpOnReq se encuentra en bajo. Dentro del else se crea otra condicional “If”, la cual verifica si la variable autoPumpStartActivated se encuentra en alto para dar paso a otra condicional “If” que se encarga de verificar si la hora actual es igual a la hora del temporizador. Si las dos horas son iguales, entonces se activa el motor. Luego se fija el tiempo que dura el motor encendido ingresado utilizando un delay definido por la variable runTimeDurationSeconds1 (Planteado en milisegundos por el programa) multiplicada por 1000 (Factor de conversión de milisegundos a segundos). Una vez que se haya cumplido el tiempo de duración, se envía una notificación emitida por la app del usuario con el mensaje “Alimentador Automático Activado” utilizando el comando Blynk.notify.

```
else if (bypassSwitch == 1)
{

 PumpOnReq = PumpControl("on");
 autoPumpStartActivated = 0;
 PumpManualRemotePb("off");
 }

 else if (remoteOnPb == 1)
```

```
 {
  PumpOnReq = PumpControl("on");

 autoPumpStartActivated = 0;
 }

 else
{
 PumpOnReq = PumpControl("off");

 autoPumpStartActivated = 0;
 }
}
```

En esta parte se da solución a problemas que podrían dar errores en caso de no ser definidos. Primero se crea un “else If” el cual es un comando que solo se activa si su expresión condicional es evaluada como real. En este primer else if se define que si el botón de activación manual se encuentra activo entonces se active el motor, pero además pone en bajo la variable autoPumpStartActivated para que no existan conflictos a la hora de ejecutar la activación automática del motor, y también se desactiva cualquier comando de activación remota que exista en ese momento, poniendo en bajo la variable PumpManualRemotePb.

Después se crea otro else If, el cual solo va a funcionar cuando el usuario desee activar el motor remotamente, poniendo en alto la variable PumpControl. Además, también pone en bajo la variable autoPumpStartActivated para que no existan conflictos a la hora de ejecutar la activación automática del motor. Luego para finalizar se crea un else, el cual solo sirve como medida de precaución, poniendo la variable waterPumpControl en bajo, en caso de que exista alguna falla al activar el motor (talvez nunca se dé esta situación), y también resetea la variable autoPumpStartActivated para que no existan conflictos a la hora de ejecutar la activación automática del motor.



```
226 void checkBlynk() {
227  if (WiFi.ready()) {
228  unsigned long startConnecting = millis();
229  while(!Blynk.connected()){
230  Blynk.connect();
231  if(millis() > startConnecting + myServerTimeout){
232
233  break;
234  }
235  }
236  }
237 }
238
239 int PumpManualRemotePb(String command) {
240  if (command == "on") {
241  remoteOnPb = 1;
242  remotePbAutoOffTimer.reset();
243  return 1;
244  }
245  else {
246  remoteOnPb = 0;
247  remotePbAutoOffTimer.stop();
248  return 0;
249  }
250 }
251
252 void autoRemotePbOff() {
253  remoteOnPb = 0;
254  remotePbAutoOffTimer.stop();
255 }
256
257 int PumpControl(String command) {
258  if(command == "on"){
259  digitalWrite(Pump, HIGH);
260  appPumpStatusLED.on();
261  return 1;
262  } else if(command == "off"){
263  digitalWrite(Pump, LOW);
264  appPumpStatusLED.off();
265  return 0;
266  } else {
267  return -1;
268  }
269 }
```

Figura 3. 27: Código parte 7.
Elaborado por: Autor.

En la figura 3.27 se muestra la séptima y última sección del código en la cual se crean dos voids breves que sirven para chequear la conexión a internet de Blynk y para para apagar automáticamente el botón de activación remota después de un cierto tiempo en caso de que se deje encendido. También se definen dos variables más tipo Int con string command, las cuales realizan varias acciones que sirve para controlar el encendido remoto del motor, la señal de salida por el pin D7 para activar el motor y los timers utilizados en el código. A continuación, se explican en detalle las líneas de código.

```
void checkBlynk() {
 if (WiFi.ready()) {
 unsigned long startConnecting = millis();
 while(!Blynk.connected()){
 Blynk.connect();
 if(millis() > startConnecting + myServerTimeout){
 break;
```

```
 }
 }
 }
}
int PumpManualRemotePb(String command) {
 if (command == "on") {
 remoteOnPb = 1;
 remotePbAutoOffTimer.reset();
 return 1;
 }
 else {
 remoteOnPb = 0;
 remotePbAutoOffTimer.stop();
 return 0;
 }
}
```

Primero se crea el void checkBlynk(), el cual sirve para verificar la conexión de internet de Blynk. Se crea una condicional “If” que verifica si el wifi está listo con el comando WiFi.ready y luego se empieza a conectar. También se crea otro If para reconectar al wifi en caso de que no se pueda conectar al servidor.

Luego se define la variable tipo Int “PumpManualRemotePb” la cual usa un String command para realizar varias acciones. Dentro de este string se crea una condicional “If” que plantea que, si el comando se utiliza, entonces se ponga en alto la variable remoteOnPb, la cual activa remotamente el motor. En este If también se resetea el timer usado con el comando remotePbAutoOffTimer.reset. Luego se crea un else en el caso contrario de que el comando string no se utilice, entonces se pone en bajo la variable y se detiene el temporizador de apagado automático del botón de activación remota.

```
void autoRemotePbOff() {
```

```
 remoteOnPb = 0;
 remotePbAutoOffTimer.stop();
}
int PumpControl(String command) {
 if(command == "on") {
 digitalWrite(Pump, HIGH);
 appPumpStatusLED.on();
 return 1;
 } else if(command == "off") {
 digitalWrite(Pump, LOW);
 appPumpStatusLED.off();
 return 0;
 }
else {
 return -1;
```

Se prosigue creando el void autoRemotePbOff(), en el que se define que cuando se use esta variable, se encienda el control remoto global en el botón para encender el motor y se detenga el temporizador de apagado automático del botón de activación remota, con los comandos remoteOnPb = 0 y remotePbAutoOffTimer.stop() respectivamente.

Luego se define la variable tipo Int “PumpControl”, la cual usa un String command para realizar varias acciones. Dentro de este string se crea una condicional “If” que plantea que, si el comando se utiliza, entonces se ponga en HIGH la variable Pump, la cual envía una señal de salida al pin D7 (Pump) para activar el motor. En este If también se usa el comando appPumpStatusLED.on() con el fin de encender el LED encontrado en la app para mostrar el estado del motor como encendido. Finalmente se crea un else If para que solo en el caso de que el comando del string esté en bajo, este envíe una señal de salida LOW al pin D7 (Pump) para desactivar el motor mediante el comando digitalWrite y también apague el LED encontrado la app mediante el comando appPumpStatusLED.off() para que se muestre como apagado (Particle, 2021).

3.3.2. App en Blynk.

Una vez completada la programación del código en Particle se requirió crear una interfaz fácil de usar, con la que el usuario pueda sentirse cómodo y pueda personalizar el tiempo de activación y de duración de encendido del alimentador automático. Esta interfaz no podía estar ubicada en la estructura del alimentador automático, ya que este se encontrará flotando en el agua la mayor parte del tiempo y se dificultará el acceso a la interfaz cada vez que se quiera cambiar los parámetros de alimentación, es por esto que se creó una interfaz usando una app que se pueda controlar remotamente por medio de wifi y que sea amigable con el usuario (Blynk, 2021).

Para la creación de la app se utilizó la plataforma de Blynk, la cual es una app que se puede encontrar de manera gratuita en la App Store para dispositivos Ios (Apple) o en la Play Store para dispositivos Android. Esta plataforma permite crear aplicaciones con diferentes controladores, entre estos el Particle Photon usado en este proyecto, y asigna un “TOKEN” único para que solo el propietario del alimentador automático lo pueda controlar, de esta manera incrementando la seguridad del programa al evitar hackeos. Blynk también ha creado un concepto de “pines virtuales” los cuales se utilizaron en la programación del código, ya que facilitan el uso del controlador sin necesidad de usar pines físicos, es decir, se pueden usar muchos pines virtuales para controlar acciones en el código sin necesidad de usar los pines físicos, que, a diferencia de los pines virtuales, son limitados en los controladores.

Dentro de la App también se tendrá la opción de notificaciones, lo que nos permite saber cuándo el alimentador automático ecológico se active automáticamente por el temporizador. Esta opción es muy útil ya que el usuario no debe estar pendiente de si el alimentador está funcionando o no, sino que puede realizar cualquier otra actividad y el alimentador le notificará automáticamente mostrando un mensaje de aviso por vía wifi cuando se active el motor en el intervalo de tiempo configurado. Esta notificación también podría ser de gran ayuda para llevar un mejor control del alimentador. La figura 3.28 muestra la interfaz de la App en Blynk.

Figura 3. 28: Interfaz de la app en Blynk.
Elaborado por: Autor.

La App creada tiene el nombre de “Alimentador Automático” y se puede dividir en 3 secciones según su diseño. La primera sección es la superior en la cual se encuentra el reloj con la hora actual y el símbolo de las notificaciones.

Para esta sección se utilizó una herramienta llamada “Display” y otra herramienta con el nombre “Notifications”. En el display se configuró un pin virtual “V2” el cual es utilizado en el código para que refleje el horario actual con Horas, minutos y segundos. También se definió en intervalo de refresco, el color y tamaño de fuente del texto y finalmente se lo nombró como “Hora Actual” para que el usuario pueda ver ese texto reflejado en la pantalla. Por otro lado, se colocó la herramienta “Notifications” a la derecha del display en la cual se definió que su prioridad es baja. En el código se definió que cada vez que el alimentador se active, se envié una notificación que diga

“Alimentador Automático Activado”, entonces esta herramienta responderá al llamado del código y reflejará una notificación de texto. La figura 3.29 muestra la primera sección de la interfaz de la App en Blynk. Se puede observar la configuración de las herramientas utilizadas en esta sección de la app en los anexos 4 y 5.

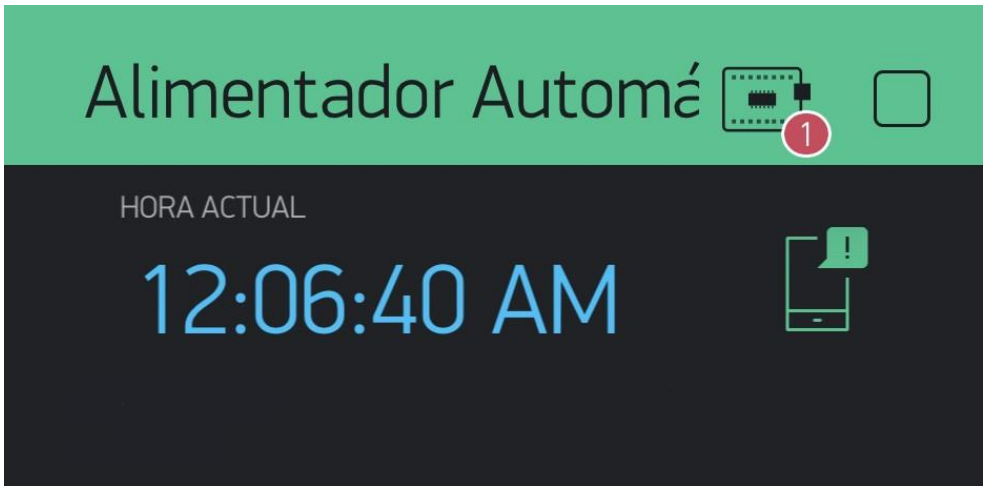


Figura 3. 29: Interfaz App sección 1.
Elaborado por: Autor.

La segunda sección es en la que se encuentra el estado del alimentador automático (encendido o apagado) y el botón para cambiar de estado manualmente cuando el usuario lo requiera. En esta sección se utilizaron las herramientas “Led” y “Button” las cuales van ubicadas debajo de la primera sección. La herramienta “Led” tiene asignado el nombre “Estado” y se la configuró con el pin virtual “V4”, esta misma cumple la función de mostrar al usuario si el alimentador automático se encuentra activado o desactivado, ya sea por activación automática o manual.

Por otro lado, la herramienta “Button” tiene asignado el nombre “Control Wifi” y cumple la función de activar y desactivar manualmente el alimentador automático según lo requiera el usuario. Esta herramienta se la configuró con el pin virtual “V1”, también se definió si es un botón modo push o switch, se nombró los estados on y off como “Alimentador ON” y “Alimentador OFF” respectivamente, y por último se definió el color del botón y el texto. La figura 3.30 muestra la segunda sección de la interfaz de la App en Blynk. Las configuraciones de las herramientas utilizadas en esta sección de la app se pueden observar en los anexos 6 y 7.

Figura 3. 30: Interfaz App sección 2.
Elaborado por: Autor.

La tercera sección es la más extensa usando el 50% de espacio de la pantalla y 5 herramientas las cuales son “Image”, 2 herramientas “Display” y 2 herramientas “Numeric Input”. La herramienta “Image” se utiliza simplemente para insertar una imagen con fines estéticos. Luego le siguen los dos “Display” de los cuales uno tiene configurado el pin virtual “V8” y el otro el pin virtual “V6”, para los dos se definió un intervalo de refresco de 2 segundos, se puso color al texto y se les asigno los nombres “Tiempo de Encendido” y “Temporizador” respectivamente. También se definió en el código que los números que aparezcan el en primer display de esta sección tengan seguido el sufijo “Sec/s” y los que aparezcan en el segundo display tengan el sufijo “Hora/s”. Finalmente se utilizaron los dos “Numeric Input”, los cuales son botones que tienen la función de permitir al usuario ingresar valores numéricos enteros para personalizar los parámetros del alimentador automático.

El primer “Numeric Input” tiene asignado el nombre “Ingresar duración” lo cual hace referencia al tiempo que dura encendido el alimentador, a este mismo se lo configuró con el pin virtual “V7” y se definió un rango de 0 a 9999999 como límite numérico, luego se fijó como sufijo el texto “Sec” para los números que se ingresen, se configuró para que la entrada numérica se incremente o decremente por una unidad con los botones “+” y “-”, y por último

se definió el color del botón y tamaño de fuente. En el segundo “Numeric Input” se utilizó una configuración parecida al primero, este botón tiene asignado el nombre “Ingresar tiempo” lo cual hace referencia a cada cuanto tiempo en horas se activa el alimentador automático, se lo configuró con el pin virtual “V5”, se definió el mismo rango numérico que el anterior, se fijó el texto “Hr” como sufijo para los números que ingresen, y finalmente se escogió el mismo color y tamaño de fuente. La figura 3.31 muestra la tercera sección de la interfaz de la App en Blynk. Se pueden observar las configuraciones de cada herramienta utilizada en esta sección de la app en los anexos 8, 9, 10, 11 y 12.

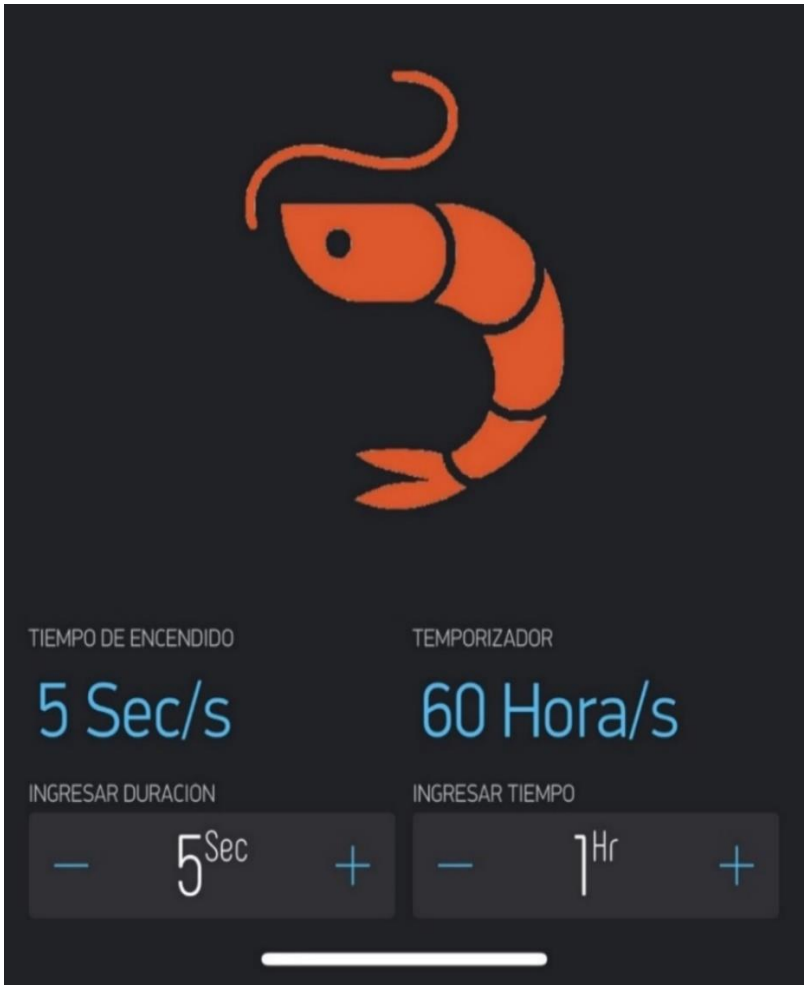


Figura 3. 31: Interfaz App sección 3.
Fuente: (Autor, 2021).

Capítulo 4: Conclusión y recomendaciones

Conclusiones

Los resultados del proyecto fueron los esperados, se cumplió a totalidad los objetivos propuestos y, lo más importante, el alimentador automático funciona a la perfección. Gracias al buen detallado de cada proceso para el diseño y ensamble de la máquina fue posible ejecutar todo a la perfección en el menor tiempo posible. Gracias a este proyecto se podrá mejorar la eficiencia en las camaroneras del país, lo cual no solo beneficiará directamente a cada camaronera, sino que también beneficiará indirectamente al país, mejorando sus índices de exportación y generando más ingresos.

Se diseñó correctamente el programa del microcontrolador y la App. Con este conjunto se logró controlar el temporizador y tiempo de duración de encendido del AAE, también el encendido manual por wifi y se creó una interfaz amigable con el usuario.

Uno de los objetivos cumplidos más importante fue usar una fuente renovable de energía, ya que no solo resuelve el problema de alimentación, sino que también se pone un grano de arena para el cuidado del medio ambiente.

Se cumplió con el diseño de una estructura resistente para soportar cualquier tipo de clima. Esta estructura se ensambló con materiales rígidos y de alta durabilidad para asegurar su resistencia ante cualquier cambio climático.

Recomendaciones

Llevar un buen orden de cada paso que se realiza, y si es posible realizar todo digitalizado primero, ya que esto ahora una gran cantidad de tiempo y dinero. Ahora se tiene la ventaja de que existen muchos programas de simulación, los cuales permiten saber cómo va a funcionar el proyecto sin siquiera haber comprado los materiales. Hay que aprovechar estas herramientas tan importantes que se pueden tener al alcance fácilmente hoy en día.

Buscar buenas plataformas que cumplan con nuestras necesidades. Por ejemplo, para este proyecto la plataforma de Particle y el controlador Photon fueron claves para que el alimentador pueda funcionar de manera remota a través de wifi. Siempre hay que buscar la plataforma que más convenga, como en este caso también fue muy útil Blynk, ya que se pudo hacer una interfaz controlada con la misma plataforma en la que se programó el código.

Bibliografía

(s.f.).

Argos. (2015). *Convertidores Buck*. Obtenido de nomadaselectronicos.wordpress.com: https://nomadaselectronicos.wordpress.com/2015/04/12/convertidores-dcdc-buck/

Augusthalem. (2018). *Camarones blancos*. Obtenido de dreamstime.com: https://es.dreamstime.com/cierre-de-camarones-blancos-en-bruto-camar%C3%B3n-blanco-y-pac%C3%ADfico-litopenaeus-vannamei-antes-penaeus-image157959272

Blynk. (2021). *Examples Blynk*. Obtenido de examples.blynk.cc: https://examples.blynk.cc/?board=Particle%20Photon&shield=Particle%20WiFi&example=GettingStarted%2FBlynkBlink

Crespi, V., & New, M. (2009). *Ténicas de Engorde*. Obtenido de fao.org: http://www.fao.org/tempref/FI/DOCUMENT/aquaculture/CulturedSpecies/file/es/es_whitelegshrimp.htm

DIYmechanics. (2017). *DIY Relay Module*. Obtenido de Instructables.com: https://www.instructables.com/DIY-RELAY-MODULE/

Gracia, M. (2018). *Internet of Things*. Obtenido de deloitte.com : https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html

Gutiérrez, L. (2019). *Alimentación Automática*. Obtenido de docplayer.es: https://docplayer.es/86130968-Alimentacion-automatica.html

Hons. (2020). *Buck Converters*. Obtenido de learnabout-electronics.org: https://learnabout-electronics.org/PSU/psu31.php

Lineaverdehuelva. (2018). *Energías Renovables*. Obtenido de Lineaverdehuelva.com: http://www.lineaverdehuelva.com/lv/consejos-ambientales/energias-renovables/Que-son-las-energias-renovables.asp#:~:text=Las%20energ%C3%ADas%20renovables%20son%20aquellas,son%20fuentes%20renovables%20de%20energ%C3%ADa.

Logicbus. (2019). *Artículos la automatización*. Obtenido de logicbus.com: https://www.logicbus.com.mx/automatizacion.php

Maquilón, J. (2017). *Método tradicional de alimentación*. Obtenido de repositorio.ug.edu.ec: http://repositorio.ug.edu.ec/bitstream/redug/22900/1/Proyecto%20de%20Investigacion%20Alimentadores%20Automaticos%20Aquamar%20S.A..pdf

Pancoast, L. (2016). *Particle Photon*. Obtenido de allaboutcircuits.com: https://www.allaboutcircuits.com/projects/working-with-the-particle-photon/

Particle. (2021). *docs.particle.io*. Obtenido de docs.particle.io: https://docs.particle.io/reference/device-os/firmware/photon/#

Particle. (2021). *Software Timers*. Obtenido de docs.particle.io: https://docs.particle.io/reference/device-os/firmware/photon/#software-timers

Piedrahita, Y. (2018). *Cultivo de Camarón*. Obtenido de aquaculturealliance.org: https://www.aquaculturealliance.org/advocate/la-industria-de-cultivo-de-camaron-en-ecuador-parte-2/

Robots-argentina. (2020). *Módulos de relé y Arduino*. Obtenido de robots-argentina.com.ar: http://robots-argentina.com.ar/didactica/modulos-de-rele-y-arduino-domotica-1/

Saúl. (2019). *Tasa de conversión alimenticia*. Obtenido de molinoschampion.com: https://www.molinoschampion.com/tasa-de-conversion-alimenticia/

sc.ehu. (2001). *Sistema automatizado*. Obtenido de sc.ehu.es : http://www.sc.ehu.es/sbweb/webcentro/automatica/WebCQMH1/PAGINA%20PRINCIPAL/Automatizacion/Automatizacion.htm

Schwarz, L. (2005). *Sector acuícola nacional*. Obtenido de fao.org: http://www.fao.org/fishery/countrysector/naso_ecuador/es

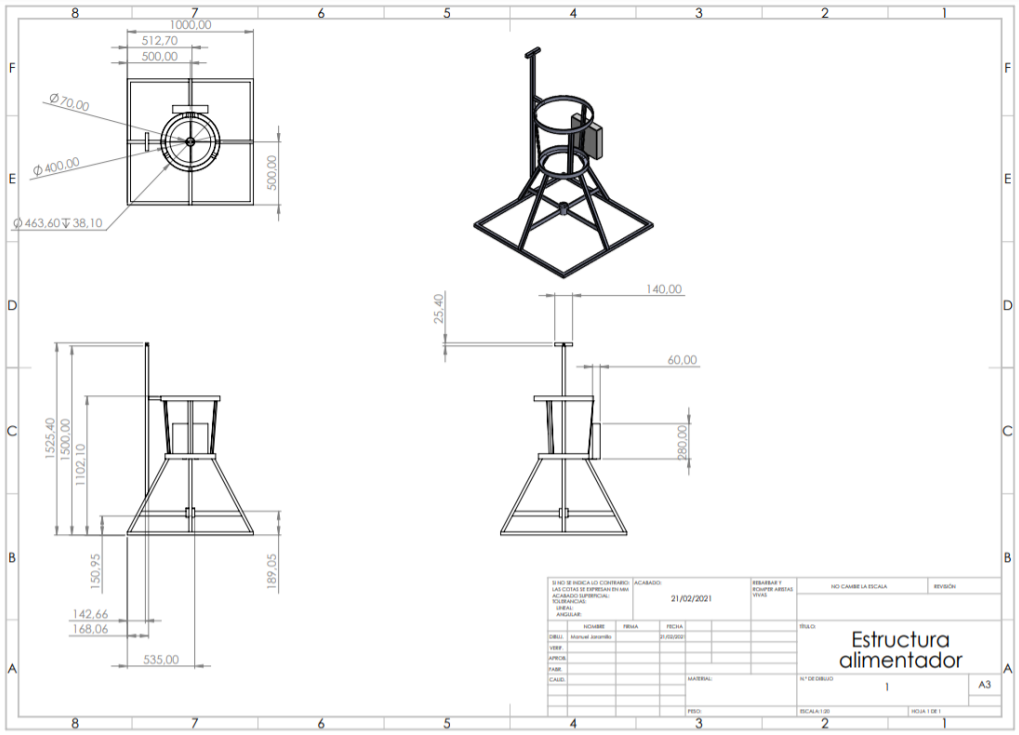
SunEnergise. (2021). *Solar Panel*. Obtenido de sunenergise.com: https://sunenergise.com/collections/solar-panel-kits/products/20w-12v-solar-panel-kits-20-watt-mono-crystalline-solar-panel-intelligent-10a-charge-controller-with-dusk-to-dawn-control-cable-adapters-for-off-grid-boat-marine-trailer-cabin

Tatoma. (2019). *Sol como energía renovable*. Obtenido de Grupotatoma.com: http://www.grupotatoma.com/noticia.php/es/El-sol-como-energia-renovable/194

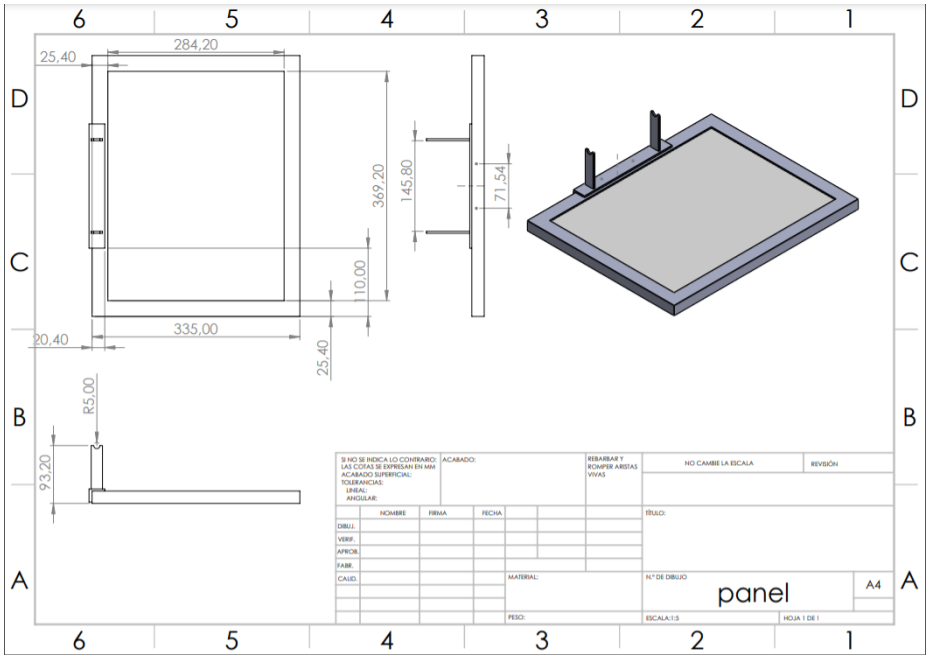
Zambritisa. (2018). *Litopenaeus vannamei*. Obtenido de Zambritisa.com: http://www.zambritisa.com/preguntas.html

Anexos

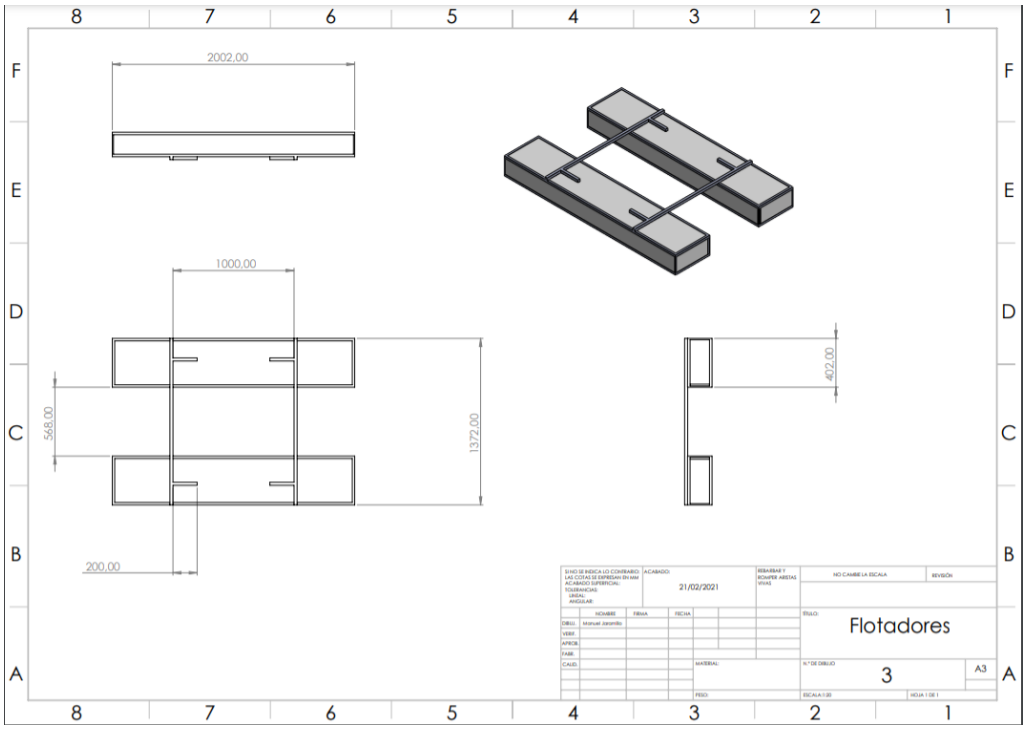
Anexo 1: Diagrama de mediciones de la estructura del alimentador automático.



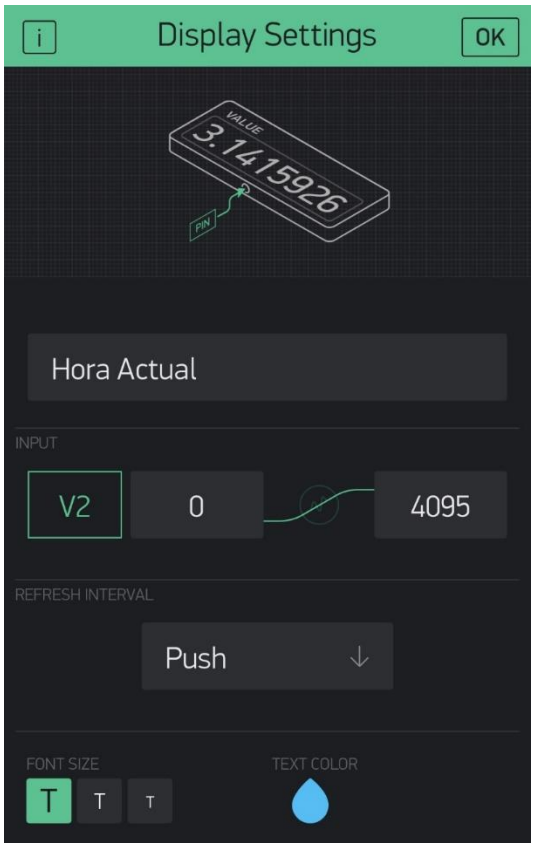
Anexo 2: Diagrama de mediciones del soporte para el panel solar.



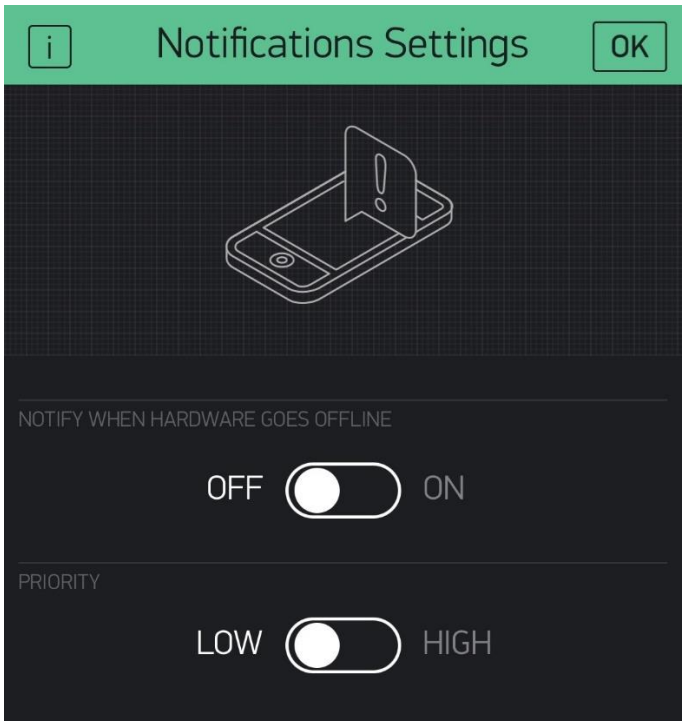
Anexo 3: Diagrama de mediciones de los flotadores.



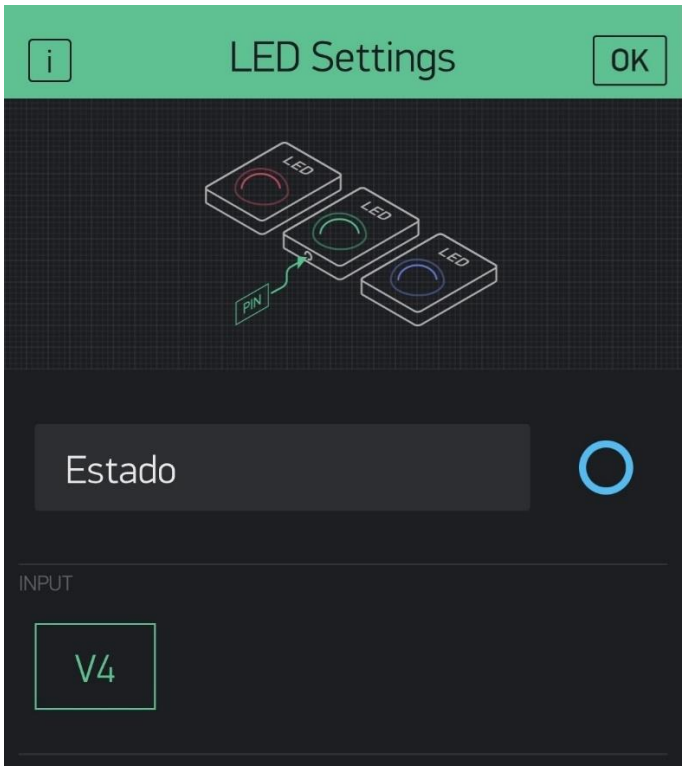
Anexo 4: Configuración del display “Hora Actual” en la App.



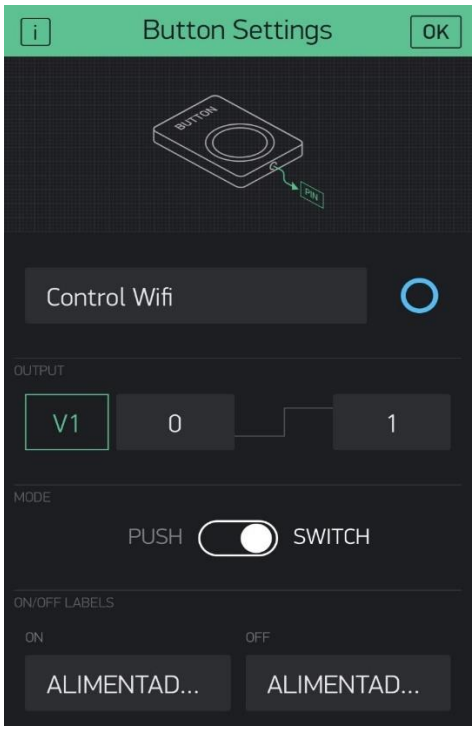
Anexo 5: Configuración de las notificaciones en la App.



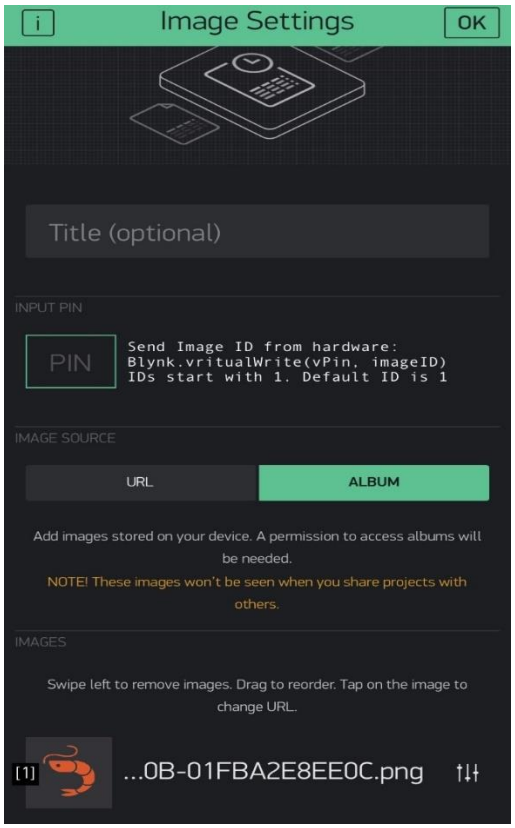
Anexo 6: Configuración del led en la App.



Anexo 7: Configuración del botón “Control Wifi” en la App.

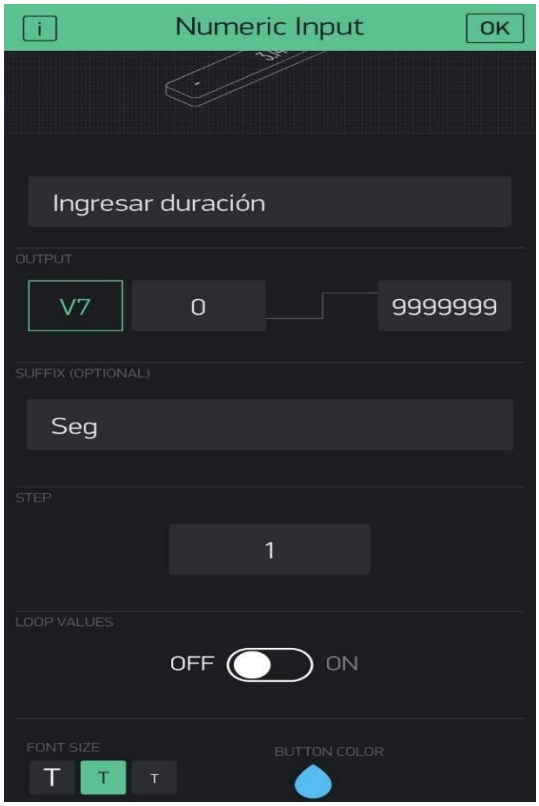


Anexo 8: Configuración de la imagen en la App.



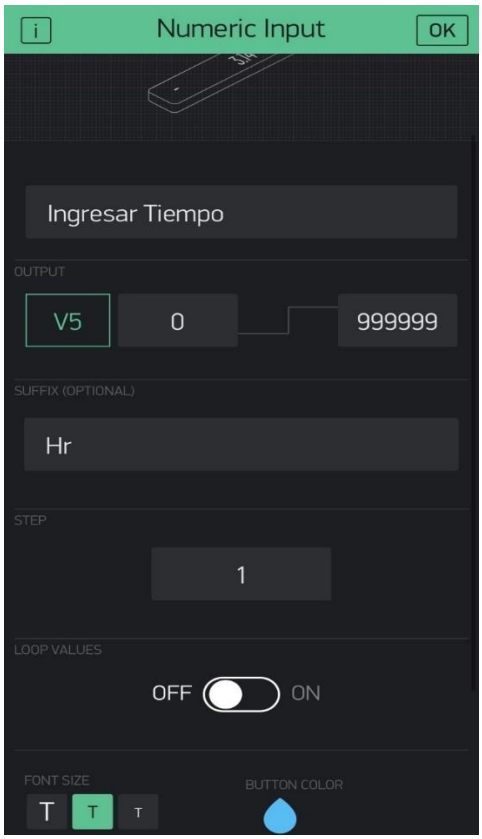
Anexo 9: Configuración del display “Tiempo de Encendido” en la App.

Anexo 9: Configuración del Numeric Input “Ingresar duración” en la App.



Anexo 11: Configuración del display “Temporizador” en la App.

Anexo 12: Configuración del Numeric Input “Ingresar Tiempo” en la App.



DECLARACIÓN Y AUTORIZACIÓN

Yo, **Jaramillo Abril, Manuel Alejandro** con C.C: # 092070493-9 autor del Trabajo de Titulación **Diseño y Ensamble de un Prototipo de Alimentador Automático Ecológico para Piscinas de Cría de Camarón** previo a la obtención del título de **INGENIERO ELECTRÓNICO EN CONTROL Y AUTOMATISMO** en la Universidad Católica de Santiago de Guayaquil.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, 8 de marzo del 2021

f. _____

Nombre: Jaramillo Abril, Manuel Alejandro

C.C: 092070493-9

REPOSITORIO NACIONAL EN CIENCIA Y TECNOLOGÍA

FICHA DE REGISTRO DE TESIS/TRABAJO DE TITULACIÓN

TÍTULO Y SUBTÍTULO:	Diseño y Ensamble de un Prototipo de Alimentador Automático Ecológico para Piscinas de Cría de Camarón		
AUTOR(ES)	JARAMILLO ABRIL, MANUEL ALEJANDRO		
REVISOR(ES)/TUTOR(ES)	M. Sc. SUAREZ MURILLO, EFRAIN OSWALDO		
INSTITUCIÓN:	Universidad Católica de Santiago de Guayaquil		
FACULTAD:	Facultad de Educación Técnica para el Desarrollo		
CARRERA:	Electrónica en Control y Automatismo		
TITULO OBTENIDO:	Ingeniero Electrónico en Control y Automatismo		
FECHA DE PUBLICACIÓN:	8 de marzo del 2021	**No. DE PÁGINAS:**	69
ÁREAS TEMÁTICAS:	Sistemas Microcontroladores, Diseño Electrónico		
PALABRAS CLAVES/ KEYWORDS:	Programación, automatización, camaronicultura, ecológico, esquemático, Proteus, Blynk y Particle IDE.		

RESUMEN/ABSTRACT (150-250 palabras): En el presente trabajo se realiza el diseño y ensamble de un alimentador automático ecológico para piscinas de camarón. Se empieza explicando antecedentes importantes de la cría de camarón en Ecuador y se definen los objetivos que se desea cumplir en este trabajo, esto se encuentra dentro del primer capítulo. Para entender mejor el contexto de que trata un alimentador automático ecológico se da una fundamentación teórica en el capítulo 2, en el cual se explican conceptos importantes como la automatización, tipo de cultivo, la camaronicultura sustentable, etc. Luego de repasar la fundamentación teórica, se explica todo el diseño y ensamble de la máquina en el capítulo 3, en el cual también se indican mediciones, circuitos esquemáticos en Proteus, programación en Particle IDE, diseño de la app en Blynk, etc. En cada una de las secciones del capítulo 3 se explica a detalle cada paso, conexión y diseño para poder ensamblar el alimentador automático ecológico en su totalidad. Para el mayor entendimiento posible de cada proceso se elaboraron diferentes diagramas, ilustraciones y modelos 3d, los cuales cubren cada detalle del diseño de la estructura y circuitos del alimentador automático.

ADJUNTO PDF:	☒ SI	☐ NO
CONTACTO CON AUTOR/ES:	**Teléfono:** 0939223739	E-mail: manuelja_10@hotmail.com
CONTACTO CON LA INSTITUCIÓN: COORDINADOR DEL PROCESO DE UTE	**Nombre:** Palacios Meléndez, Edwin Fernando	
	Teléfono: +593-9-67608298	
	E-mail: edwin.palacios@cu.ucsg.edu.ec	

SECCIÓN PARA USO DE BIBLIOTECA	
Nº. DE REGISTRO (en base a datos):	
Nº. DE CLASIFICACIÓN:	
DIRECCIÓN URL (tesis en la web):	