ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN PROTOTIPO DE DISPENSADOR DE COMIDA PARA MASCOTAS CONTROLADO A TRAVÉS DE INTERNET

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES

GABRIEL DAVID MANOSALVAS SEIBA

gabriel.manosalvas01@epn.edu.ec

DIRECTOR: ING. LEANDRO ANTONIO PAZMIÑO ORTIZ, MSC.

leandro.pazmiño@epn.edu.ec

DMQ, agosto 2023

CERTIFICACIONES

Yo, GABRIEL DAVID MANOSALVAS SEIBA declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

GABRIEL DAVID MANOSALVAS SEIBA

gabriel.manosalvas01@epn.edu.ec

davidmanosalvas724@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por GABRIEL DAVID MANOSALVAS SEIBA, bajo mi supervisión.

LEANDRO ANTONIO PAZMIÑO ORTIZ
DIRECTOR

leandro.pazmiño@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

GABRIEL DAVID MANOSALVAS SEIBA

DEDICATORIA

El presente trabajo va dedicado a la Escuela Politécnica Nacional por permitirme estudiar dentro de sus instalaciones para adquirir los conocimientos necesarios para mi desarrollo profesional.

Dedicado también a mi mamá por trabajar duro para brindarme a mí y a mis hermanos el alimento y la educación.

Y en general a mi familia por apoyarme para seguir adelante.

AGRADECIMIENTO

Quiero expresar mi gratitud a todas las personas que me han apoyado en el transcurso de mi vida universitaria.

Principalmente, quiero agradecer a mi mami por su ayuda y comprensión, así como su esfuerzo, haciendo todo lo posible para que no falte comida en el hogar. Asimismo, a mis hermanos Axel, Zafiro y Abigail por su compañía y por haber creído en mí.

Además, agradezco a mi amigo Walter, mi tía Blanca y a mi abuelito Luis por ayudarme cuando lo he necesitado.

Agradezco a mis perritos Benjï y Bruno por siempre acompañarme en las noches de desvelo y alegrarme la vida cada que llego a casa.

Finalmente, quiero agradecer a mi tutor de tesis, el Ing. Leandro Pazmiño por ayudarme en el desarrollo del presente documento, además de impartir sus conocimientos como docente dentro de las aulas de la Escuela de Formación de Tecnólogos. Asimismo, a los ingenieros Javier Armas y Fernando Becerra por su apoyo durante el desarrollo de mi carrera universitaria.

Y en general, agradezco a todos los que me han ayudado de alguna manera a no rendirme y seguir siempre adelante para cumplir mis metas.

ÍNDICE DE CONTENIDOS

RESUMEN

Como objetivo principal del presente proyecto, se tiene el desarrollo de un dispensador de alimento para mascotas, cuyo control debe darse de manera remota con el uso de plataformas dedicadas al Internet de las cosas (IoT) y la aplicación de mensajería Telegram. Este dispositivo permite la obtención de datos para la emisión de alertas según la detección de escasez de alimento dentro de sus compartimentos, tanto para agua como para comida, señales producidas mediante el uso de sensores dedicados a cada aplicación y procesadas por un microcontrolador.

Además decide la dispensación del alimento dado cierto horario establecido por el usuario, y de acuerdo a la detección de la presencia de la mascota cumplido este criterio.

Para el desarrollo del prototipo, se puso en marcha diferentes fases que permitieron una elaboración óptima del dispositivo requerido, el cual fue definido para el cumplimiento de lo propuesto como plan del proyecto.

Dentro de la metodología, se presenta una síntesis de cada uno de los pasos necesarios para el establecimiento y satisfacción de cada objetivo planteado. Seguido de esto, se presenta la sección de resultados, donde se expone de manera detallada el cumplimiento de cada procedimiento establecido inicialmente en el desarrollo de la metodología.

Así, se da la elección de los elementos físicos y lógicos necesarios para la realización del proyecto, el diseño del prototipo físico, código de programación y placa de circuito impreso (PCB) con las conexiones necesarias, la implementación del dispensador dentro de una maqueta, finalmente, la documentación de pruebas sobre el funcionamiento del dispensador.

PALABRAS CLAVE: control remoto , IoT, microcontrolador, PCB , Telegram.

ABSTRACT

The main objective of this project is the development of a pet feeder whose control must be given remotely with the use of platforms dedicated to the Internet of Things (IoT) and the messaging application Telegram.

This device allows the collection of data for the issuance of alerts according to the detection of food shortage inside its compartments, both for water and food, signals produced through the use of sensors dedicated to each application and processed by a microcontroller that allows to connect to the network.

In addition to deciding the dispensing of food given a certain schedule set by the user and according to the detection of the presence of the pet fulfilling this criterion.

For the development of the prototype, different phases were implemented to allow an optimal elaboration of the device required for the fulfillment of the proposed project plan.

Within the methodology, a synthesis of each of the necessary steps for the establishment and satisfaction of each proposed objective is presented. Following this, the results section is presented, where the fulfillment of each procedure initially established in the development of the methodology is presented in detail.

Thus, the choice of the physical and logical elements necessary for the realization of the project, the design of the physical prototype, programming code and printed circuit board (PCB) with the necessary connections, the implementation of the dispenser within a mock-up, and finally, the documentation of tests on the operation of the dispenser.

KEYWORDS: *IoT, Telegram, remote control, PCB, microcontroller.*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

A través del presente proyecto se realizará el diseño e implementación de un prototipo que permita la dispensación de agua y comida para mascotas, ya sean gatos o perros, tomando en cuenta el censado de la presencia de la mascota y los horarios de comida establecidos, controlado a través de Internet por medio de una aplicación en Android y de Telegram.

1.1 Objetivo general

Implementar un prototipo de dispensador de comida para mascotas controlado a través de Internet por medio de una aplicación en Android y de Telegram.

1.2 Objetivos específicos

- Identificar los requerimientos para el diseño del prototipo.
- Seleccionar el *hardware* y *software* acorde a los requerimientos establecidos.
- Diseñar el prototipo del dispensador.
- Implementar el prototipo en una maqueta.
- Realizar pruebas de funcionamiento del prototipo.

1.3 Alcance

Con el desarrollo del proyecto presentado en este documento, se busca el cumplimiento de elaboración, diseño e implementación de un prototipo de dispensador de comida y agua para mascotas que sea controlado por medio de Internet y de Telegram. Por consiguiente, dicho prototipo tendrá la capacidad de realizar las siguientes acciones:

- Censar la presencia de la mascota para permitir la dispensación de la comida y agua.
- Controlar la dispensación de comida en función de los horarios establecidos.
- Permitir el control del dispensador a través de Internet y de Telegram.
- Alertar al dueño en caso de que la comida/agua se haya terminado.

1.4 Marco Teórico

IoT

Con la evolución del campo de desarrollo de tecnologías como herramientas interactivas con el ser humano, se ve necesario el desarrollo de una capacidad de cómputo que permita la administración de una variedad de dispositivos, sensores, objetos, etc.

Para esto, se plantea la tecnología IoT, la cual es una herramienta que permite la conexión de distintos elementos y dispositivos electrónicos con el Internet para su manejo remoto y automatización de actividades; estos dispositivos utilizados en diferentes ambientes pueden ser componentes de uso doméstico como refrigeradores, cocinas, entre otros; así también, accesorios personales como relojes, celulares, etc.

Según la función que cumple cada elemento integrado a la red, dichos componentes, son clasificados de la siguiente manera: sensores, interruptores y actuadores, los cuales al conectarse a un controlador se pueden coordinar para su funcionamiento [1].

En adición, para el control remoto por parte del usuario, existen plataformas que permiten la coordinación de objetos o 'cosas' conectadas a Internet, dentro de las más conocidas para el desarrollo de interfaces de usuario se encuentran las siguientes:

Blynk

Consiste en una plataforma de IoT utilizada para la conexión de dispositivos electrónicos mediante el uso de su propia infraestructura denominada "Blynk cloud", o mediante la implementación de un servidor local basado en dicha plataforma, permitiendo realizar las configuraciones necesarias para el diseño del *dashboard* o plantilla que se presentará al usuario para permitir el control y supervisión remota del objeto IoT.

Esta plataforma permite la interconectividad entre dispositivos y plataformas de distintos fabricantes, tanto plataformas de usuarios finales como Android o *iOS*; así como dispositivos controladores como Esp, Arduino o Raspberry [2].

Arduino IoT Cloud

Es una plataforma que permite la configuración de un cuadro de mando para el manejo remoto mediante conexión a la nube por parte de dispositivos microcontroladores destinado al desarrollo de proyectos de IoT permitiendo al igual que Blynk la configuración de automatizaciones, alertas dados ciertos eventos, entre otras funcionalidades.

Además, Arduino IoT Cloud integra su entorno de desarrollo dentro de las opciones de configuración en la plataforma, facilitando la compilación del código sin el uso de *software* extra [3].

Microcontrolador

También denominado *'MCU'* o circuito integrado programable, es un dispositivo electrónico capaz de emitir y recibir señales eléctricas para la activación programada de diferentes módulos o actuadores conectados a este.

Se encuentra compuesto de una o varias memorias que alojan el programa y las variables establecidas para su funcionamiento y un procesador el cual coordina las tareas y maneja la información definida mediante su programación. Además, para la conexión de sensores o actuadores, contiene periféricos para la interacción con las salidas y entradas digitales o analógicas [4].

Dentro del listado de microcontroladores comerciales, se tienen los siguientes modelos más utilizados:

Esp32

Es un microcontrolador, utilizado en la implementación de dispositivos orientados al IoT. Permite conectividad mediante tecnologías como *Wi-Fi* y *Bluetooth* con el uso de módulos integrados, asimismo, maneja tecnologías de comunicación SPI, I2C, I2S, entre otras.

Al encontrarse orientado a aplicaciones de IoT, Esp32 se encuentra diseñado para un consumo bajo de energía, permitiendo el uso de baterías como alternativas de alimentación. En adición, Esp32 permite la configuración de un modo de *Wi-Fi* híbrido, el cual permite su funcionamiento como estación o punto de acceso, garantizando la conexión a Internet o permitiendo la conexión de dispositivos dentro de una red creada, respectivamente [5].

Arduino

Es una plataforma destinada a la creación de circuitos electrónicos como placas de desarrollo que integran microcontroladores para la automatización de tareas, pero esencialmente centrado en un entorno educacional, el cual, mediante el uso de su entorno de desarrollo integrado o *IDE*, permite la programación de diferentes placas para su funcionamiento junto a sensores y actuadores eléctricos [6].

Sensores

Como su nombre lo indica, son dispositivos electrónicos capaces de sentir determinadas magnitudes, tanto físicas como químicas, presentes en el medio convirtiéndolas en señales eléctricas facilitando su medida y permitiendo la recolección de datos del entorno o automatización de acciones dado cierto valor medido.

Los sensores pueden estar conectados a microcontroladores o circuitos integrados programables con el fin de permitir el accionamiento o la visualización de un evento por parte del usuario [7].

Entre la amplia gama de sensores que permiten la comunicación mediante señales eléctricas con un microcontrolador, se tienen los siguientes:

- Sensor infrarrojo reflectivo: Este sensor permite la detección de cercanía de algún objeto mediante la activación de un led infrarrojo que produce una luz no visible, la cual, al re*bot*ar con un objeto, es percibida por un diodo receptor de luz infrarroja integrado junto al led emisor y envía una señal de acuerdo al tiempo de retraso con el que llegó la luz emitida [8].

- Sensor de radiación infrarroja pasiva (PIR): Este módulo es capaz de enviar una señal digital mediante uno de sus pines cada que se detecte movimiento dentro del rango óptimo de medición. Su funcionamiento consiste en el censado de la magnitud de calor que puede emitir un individuo cercano a dicho sensor o la medición de la cantidad de luz infrarroja radiada por ciertos objetos para la transmisión de una señal de salida lógica alta hacia el microcontrolador [9].

- Sensor de nivel de agua: Es un dispositivo eléctrico que permite la medición de la altura de un líquido dentro de un recipiente, este puede ser discreto o continuo según las medidas que transmite de forma eléctrica; siendo que, para el primer caso, el sensor emite un rango de valores según la altura del líquido, mientras que el continuo únicamente emite una señal cuando el líquido ha sobrepasado o a disminuido su altura respecto al lugar de referencia [10].

Actuadores

Son elementos electrónicos que transforman la energía eléctrica en movimiento o variaciones de las diferentes magnitudes eléctricas en el ambiente. Mediante su conexión con un microcontrolador o el uso de sensores, estos dispositivos facilitan la interacción con el mundo real; siendo dirigido por un sistema de control, los actuadores permiten realizar acciones o poner en marca otros equipos conectados [7].

Telegram

Telegram es una aplicación de mensajería que permite establecer comunicaciones rápidas y seguras mediante mensajes de texto sincronizados en la nube conectada mediante Internet.

Esta aplicación se encuentra disponible tanto para plataformas móviles como para ordenadores, para permitir el envío de mensajes entre individuos y grupos o canales; además de tener funciones extras como la creación de *bots* que permiten la automatización de mensajes o integración con otras plataformas para la recepción de alertas o envío de datos hacia y desde el usuario [11].

2 METODOLOGÍA

Como etapa inicial se analizó el alcance y objetivos definidos en el plan para el proyecto como método para establecer todas las características principales e indispensables que cumplan con los requerimientos para el desarrollo del prototipo.

Una vez establecidas y comprendidas estas características, se procedió a definir los componentes físicos y lógicos, como lo son *hardware* y *software*, respectivamente, precisos para la implementación de las interfaces del usuario, comunicaciones y configuraciones en el dispositivo IoT para permitir su control remoto desde Internet.

Asimismo se definieron elementos electrónicos como los sensores, actuadores, microcontroladores y en general el *hardware* necesario para la implementación del prototipo que permita dispensar el alimento y censar la presencia de la mascota, además de brindar la comunicación y capacidad de procesamiento requerido para garantizar el funcionamiento óptimo del dispensador.

Dentro de la selección del *hardware* se definió inicialmente la estructura del prototipo, así como el mecanismo de funcionamiento para el dispositivo mediante la búsqueda de dispensadores comerciales, así, se facilitó la elección de sensores y actuadores útiles para la realización de las actividades planteadas en el alcance del plan.

Además, se establecieron los detalles para la elección adecuada del microcontrolador mediante la comparación entre diferentes tarjetas de procesamiento de datos, ya sea entre distintos fabricantes como Arduino y Espressif, así como entre placas de la misma marca como el Esp32 y Esp8266, para esta elección se tomó en cuenta la tecnología de comunicación disponible para cada microcontrolador, siendo que en el presente caso es fundamental la conexión a Internet para el envío de información.

En adición, una vez definido el modelo de microcontrolador como núcleo de procesamiento de la información y comunicación para el prototipo y los elementos que componen los mecanismos de detección y accionamiento; se comparó y analizó las contras y beneficios de las diferentes plataformas IoT para el control remoto del dispensador, eligiendo según convenga, entre plataformas como Arduino IoT Cloud, Blynk Cloud y Adafruit IO para una interfaz de usuario limpia y sencilla que permita establecer horarios de dispensación de comida, así como las notificaciones de alerta al detectar la falta de comida y agua dentro del dispensador.

Por otro lado, para la comunicación con la plataforma Telegram, se definió los diferentes *bots* que permitirán la creación de un *bot* propio para la solicitud de comandos que permitan el control del dispensador; asimismo, la limitación de comunicación con un solo usuario mediante la definición del id del desarrollador del dispensador.

En últimas estancias, ya teniendo establecida la plataforma a utilizar, se procedió al diseño de la maqueta, así como la realización del esquema de conexiones de los elementos que lo conforman; para, seguido de esto, configurar cada uno de los pines GPIO del microcontrolador y sus funciones según: los comandos (en el caso de Telegram) o los parámetros ingresados por el cliente en la aplicación de Android.

Con el *software* y *hardware* configurados, se unió las partes del prototipo en la maqueta para obtener una estructura que cumpla con el alcance del proyecto y permita evidenciar el correcto funcionamiento.

Finalmente, se realizaron las debidas pruebas para validar el desempeño óptimo del dispensador, así como la toma de errores ocurridos en su ejecución para permitir la corrección de estos mismos y garantizar el cumplimiento del proyecto definido dentro del plan de integración curricular, además del desarrollo del documento que evidencie todo el proceso y resultados obtenidos

3 RESULTADOS

En la presente sección se brindan las características sobre cada uno de los objetivos resueltos como metodología para la implementación del proyecto estando compuesto de identificación de requerimientos, elección de tanto *hardware* como *software* adecuados según las exigencias planteadas, diseño del prototipo e implementación de este, para finalmente realizar las pruebas correspondientes con el objetivo de constatar el adecuado funcionamiento del dispensador de comida planteado en el plan donde se describe el proyecto.

3.1 Identificación de los requerimientos para la implementación del prototipo

De acuerdo con el alcance definido para el presente Trabajo de Integración Curricular y tras un análisis previo realizado, se identificaron los siguientes requerimientos necesarios para la realización del prototipo de dispensador.

Conectividad al Internet

El prototipo debe contar con un módulo de comunicación inalámbrica preferiblemente integrado para evitar el establecimiento de comunicación serial con un módulo externo que pueda decrementar la velocidad de procesamiento de los datos, dicho módulo debe permitir la conectividad a Internet por medio de tecnología *Wi-Fi* o *Ethernet* para permitir la conexión remota con el usuario por medio de las plataformas indicadas como lo son la aplicación para Android o Telegram.

Mecanismos de alimentación eléctrica

El prototipo necesita una conexión a la alimentación eléctrica que permita la energización continua del dispositivo; sin embargo, no es imprescindible el uso de una alimentación sin ninguna interrupción, evitando el uso de dispositivos de alimentación redundantes que aumenten el costo para la implementación del prototipo.

Adicional a la fuente principal de alimentación para los sensores y actuadores junto con el microcontrolador, es necesario el uso de una conexión independiente para la alimentación de motores que consuman una cantidad alta de corriente para evitar daños en el microcontrolador.

Censado de presencia

Es necesario el uso de un sensor de presencia para determinar que la mascota se encuentre dentro de un área establecida, esto para ejecutar la activación del dispensador según una porción de alimento diaria establecida dependiendo de la cantidad promedio de consumo diario ideal.

Establecimiento de horarios de dispensación programables

Se debe programar dentro del *software* del microcontrolador la capacidad de permitir la activación del dispensador dados ciertos tiempos establecidos por el usuario como método de asegurar la alimentación de la mascota mediante rutinas programables.

Alerta de escasez de alimento

El prototipo debe enviar una notificación de alerta al usuario al determinar un nivel de comida y agua por debajo de lo establecido dentro de los compartimentos del dispensador para evitar la escasez de alimento.

Integración con aplicación para Android

El dispensador debe permitir su control mediante una interfaz de usuario desarrollada para la plataforma Android. Esta interfaz debe ser sencilla de utilizar y que permita el control de horarios y cantidades de alimento a dispensar. Además, debe ser desarrollada con el uso de plataformas IoT.

Integración con Telegram

El control del dispensador debe darse también por medio de un chat de Telegram, donde además del control de horarios y porciones de comida se establezca la capacidad de recibir alertas como mensajes de texto cuando se termine el alimento disponible dentro de los compartimentos; asimismo, brindar un menú que contenga las instrucciones, órdenes y sintaxis de comandos necesarias para la configuración correcta del dispositivo por parte del usuario.

Dispensador de comida y agua

El prototipo debe permitir la dispensación de una cantidad controlada de comida y agua según lo desee el usuario, además de permitir el almacenamiento de estos alimentos en diferentes compartimentos protegidos de forma que la mascota no pueda acceder a estos sin permiso del dueño; adicional a esto, debe presentar un plato y bebedero integrados donde se dispense el alimento de la mascota y evitar el uso de recipientes pequeños los cuales no puedan contener el alimento recomendado por horario.

Finalmente, la maqueta del dispensador, debe tener un peso y tamaño manejable para que el usuario pueda posicionarlo en cualquier sitio dentro del hogar; además de contener el microcontrolador y sus conexiones protegidas para evitar daños por factores externos.

Desarrollo de placa de circuito impreso (PCB)

Para la conexión de los diferentes actuadores y sensores con el microcontrolador, se debe desarrollar un PCB que contenga *headers* o encabezados que permitan la conexión de cables jumper tipo macho, tanto para los pines de cada señal, así como para la alimentación positiva y conexión a tierra de todos los elementos que componen el circuito.

3.2 Determinación del *hardware* y *software* requeridos

De acuerdo con los requerimientos definidos para el desarrollo del dispensador, se optó por el uso de los siguientes elementos para satisfacer las necesidades del proyecto tanto en *hardware* como *software:*

Elección del *hardware*

Para la elección del microcontrolador apto para las tareas de dispensación y comunicación con las plataformas para el usuario e Internet, se comparó entre dos desarrolladores de placas electrónicas conocidos, como son: Arduino y Espressif, y sus modelos destinados a proyectos IoT, donde se opta por los modelos de este último ya que cuentan con un módulo de conexión *Wi-Fi* integrado.

Definido el fabricante de placas de desarrollo para el proyecto, se compara entre los dos modelos o placas disponibles distribuidos por Espressif.

Como se observa en la Tabla 3.1, la placa de desarrollo Esp32 tiene capacidades de procesamiento y memoria mejores que el Esp8266 con un precio obviamente mayor, siendo que, al necesitar la configuración de horarios, comunicación con plataformas y lectura de sensores, se opta por el Esp32 al tener una mayor eficiencia.

Tabla 3.1. Comparativa Esp32 y Esp8266 *[12]*, *[13]*

	Esp32	**Esp8266**
Velocidad de reloj	240 (MHz)	80 (MHz)
Núcleos de procesamiento	2	1
Memoria RAM	520 (KB)	80 (KB)
Conectividad	*Wi-Fi* y *Bluetooth*	*Wi-Fi*
Puertos GPIO	34	17
Precio	~15 (USD)	~8 (USD)

Por otro lado, para la elección de sensores, interruptores y actuadores se realizó una búsqueda de dispositivos destinados a cada acción definida para cumplir con los objetivos del dispositivo dispensador, donde se estableció los siguientes elementos electrónicos:

El uso de un sensor PIR para la detección de presencia de la mascota, donde se compararon diferentes modelos de este, entre los cuales se tiene el HC-SR501, HC-SR505, y AM312 como se muestra en la Tabla 3.2:

Tabla 3.2. Comparación sensores PIR *[14]*, *[15]*, *[16]*

	HC-SR501	**HC-SR505**	**AM312**
Voltaje de operación	5 (V) – 20 (V)	4.5 (V) – 20 (V)	2.7 (V) – 12 (V)
Temperatura	-15 (°C) ~ + 70 (°C)	-20 (°C) ~ + 80 (°C)	-20 (°C) ~ + 60 (°C)
Voltaje de salida	3.3 (V)	3.3 (V)	3.3 (V)
Ángulo de detección	120°	100°	100°
Rango de sensibilidad	Ajustable 3 (m) – 7 (m)	3 (m)	3 (m) – 5 (m)
Disparo	Modo repetido o único	Único	Único
Retardo	Ajustable 5 (s) – 300 (s)	8 (s)	2 (s)
Consumo de energía	65 (mA)	65 (mA)	No especificado

Por consiguiente, se estableció el uso de específicamente el modelo HC-SR501, el cuál brinda al usuario la facilidad de ajustar el tiempo de retardo y rango de detección para limitar el área para el censado y la cantidad de señales enviadas al microcontrolador; además de permitir establecer un modo de disparo repetido que envía señales, mientras se detecta la presencia de la mascota de forma continua, a diferencia de los otros dos modelos que ejecutan un disparo único para la detección, esperando un periodo de tiempo para el reinicio de censado de presencia alrededor.

Por otro lado, para la detección de escasez de agua dentro de su respectivo compartimento, se estableció el uso de un sensor de tipo interruptor flotador para el nivel de agua que emita las señales lógicas necesarias que permitan la generación de su alerta correspondiente.

Para su elección, se comparó entre módulos que cumplan la misma función de detectar un determinado nivel de agua, entre los cuales se tienen los siguientes: sensor de nivel de combustible, descartado por su manejo de señales analógicas y su precio; sensor de distancia por sonido ultrasónico, descartado por la necesidad de dicho módulo expuesto a la humedad, asimismo como el módulo de detección de nivel de agua para Arduino, limitado a su ubicación en la parte superior del contenedor, lo cual impide la medición de un nivel de agua tan bajo o escaso. Por dichas razones, se escoge el sensor tipo interruptor mencionado anteriormente.

También, se elige un sensor de detección de obstáculos para generar la alerta de falta de comida; sensor elegido por sobre el sensor de peso al tener este un costo mayor, y descartando el uso de sensor de distancia para evitar sensores fijos en la tapa que dificulten el acceso a los compartimentos por parte del usuario.

Finalmente, para la dispensación de comida, se comparó entre dos mecanismos de dispensación; siendo el primero, el uso de un tornillo sinfín conectado a un motor, el cual desplace la comida a través de un conducto para terminar cayendo por un agujero hacia el plato, caso descartado por la complejidad de su implementación, al necesitar un armado 3D de la estructura.

Por otro lado, se tiene el mecanismo conformado por un servomotor que permita el movimiento de una placa para el paso de comida a través del compartimento hacia el plato. Por consiguiente, se define el uso del servomotor y su respectivo mecanismo debido a su facilidad de implementación.

Por último, para la dispensación de agua, se hace uso de una bomba sumergible que con el uso de un motor interno permite el paso de agua hacia el exterior mediante una manguera, conectado a un relé de 5 (V) para evitar dañar el microcontrolador por el consumo excesivo de corriente necesario para el funcionamiento de la bomba.

Este mecanismo es escogido por sobre el uso de una válvula solenoide al comparar su precio, siendo que además es la técnica más utilizada en proyectos con objetivos similares resaltando su uso con 5 (V) en lugar de válvulas solenoides cuya fuente de alimentación debe ser de 12 (V) [17], dificultando el proyecto al necesitar de una fuente de alimentación con tensión diferente a la usada por el microcontrolador.

Elección del *software*

Adicional a esto, para la elección de interfaces o plataformas IoT para el usuario, se comparó entre los beneficios que ofrecen las plataformas de Blynk y Arduino IoT Cloud para las funciones necesarias para el dispensador en sus planes más económicos.

Como se puede evidenciar en la Tabla 3.3, la mejor alternativa resulta ser la creación de un servidor local para el alojamiento de la aplicación para la conexión entre dispositivos, teniendo en cuenta que la única limitante presentada por dicha opción es la implementación del servidor y las configuraciones necesarias para su conexión con Internet, para lo cual, se definió el uso de una plataforma de alojamiento de servidores virtuales como lo es Google Cloud Platform *(GCP)*.

GCP, es una plataforma la cual ofrece a desarrolladores la prestación de su infraestructura para alojar servidores con un cobro relativamente bajo, siendo que para los tres primeros meses de uso permite la activación de un período de prueba sin costos adicionales [18].

Tabla 3.3. Comparativas plataformas IoT *[19]*, *[20]*

	Plan gratuito Blynk	**Plan gratuito Arduino IoT Cloud**	**Blynk con servidor local**
Dispositivos	2 dispositivos	2 dispositivos	Ilimitado
Usuarios permitidos	5 usuarios	Ilimitados	Ilimitado
Widgets	Limitado	Ilimitado	Ilimitado
Variables	10 por plantilla	5 por plantilla	Ilimitada

Finalmente, para la configuración del chat de Telegram para el control del dispensador, se definió el uso de *bots*, donde para la creación del *bot* dedicado a la comunicación con el microcontrolador, se optó por el uso de *BotFather*, que a diferencia de otras herramientas como Microsoft Bot Framework o *Botsify*, se encuentra integrado a la aplicación facilitando su uso y configuración de los *bots* creados dentro de la misma.

Además, para la obtención del número de identificación (ID) correspondiente al chat de usuario, se optó por *IDBots el cual asimismo es un bot integrado a la plataforma de mensajería*, este ID es utilizado para la limitación de comunicación a únicamente el dueño de la mascota, evitando la comunicación con cualquier otro usuario de Telegram, siendo que el *bot* es público pero únicamente responderá al usuario registrado [21].

Además, para la recopilación del código para el *microcontrolador*, se optó por el entorno de desarrollo integrado de Arduino al estar dedicado para desarrolladores y permitir la programación de microcontroladores de la marca Espressif.

3.3 Diseño del prototipo del dispensador

Arquitectura de comunicación

El desarrollo de la arquitectura de comunicación entre las plataformas y el prototipo dispensador, así como el esquema de recepción y emisión de señales, es esencial para comprender el funcionamiento básico que se define para la programación del microcontrolador. Para el proyecto actual, se plantea la conexión con las plataformas Blynk y Telegram mediante sus respectivas apps desarrolladas para el usuario.

Como se puede observar en la Figura 3.1, el dispensador requiere de las conexiones tanto con sensores que permitan realizar mediciones sobres escasez de alimento y detección de presencia, así como de actuadores para los mecanismos de dispensación y de llamado a la mascota; además, se debe tener en cuenta la conexión con la red privada del usuario para permitir su salida hacia Internet además de la coordinación con las plataformas Blynk y Telegram para la definición de horarios, recepción de alarmas y solicitud de recomendaciones acerca de la cantidad de alimentos a dispensar.

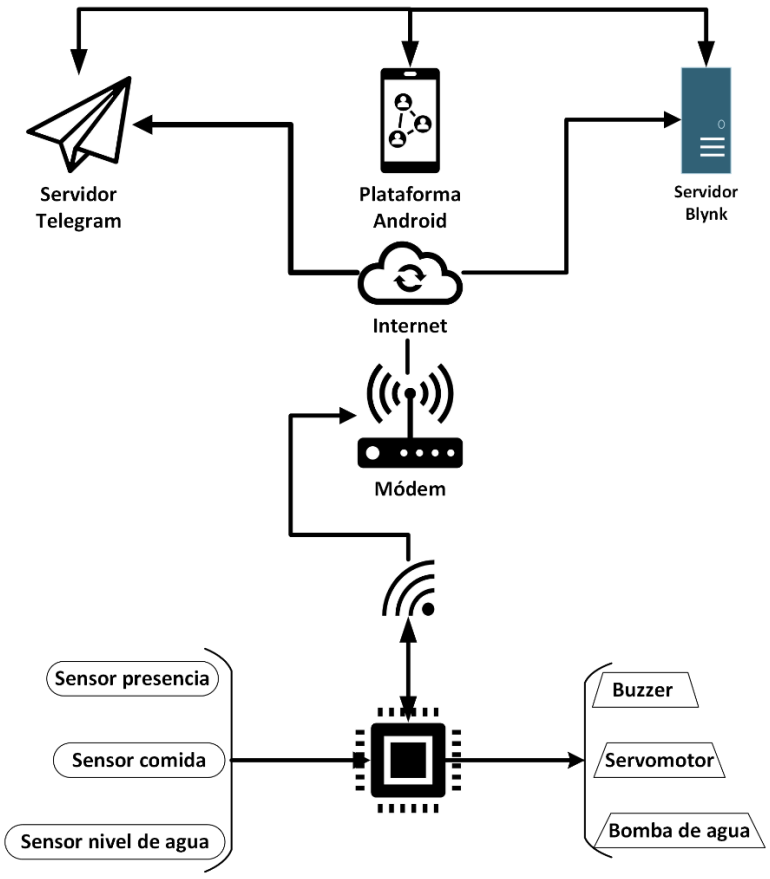

Figura 3.1. Diagrama de conexión

Presentado el diagrama de comunicación externa e interna con el microcontrolador, al igual que todos los dispositivos pertenecientes al IoT; se puede concluir que se trata sobre un sistema centralizado controlado por el Esp32 el cual recibe ordenes mediante las interfaces de usuario facilitadas por el uso de *software* para la plataforma Android.

Creación del cuadro de mando en aplicación para Android

Como se definió en la elección de *software* para cumplir con los requerimientos del proyecto, se utilizó la aplicación Blynk para la creación del cuadro de mando que permita el control del dispensador mediante cada uno de los *bot*ones, índices de selección y cuadros de ingreso de variables necesarios.

Cabe recalcar que al optar por la implementación de un servidor local de Blynk, es necesaria la utilización de la aplicación Blynk legacy (versión de Blynk app 2.27.14 y previas) para la compatibilidad con el servidor.

Además, para garantizar la conexión a Internet, se implementó el servidor Blynk *legacy* dentro de una instancia virtual en Google Cloud Platform para disponer de una dirección pública que permita la conexión remota desde Internet.

Para la implementación del servidor, se descargó el ejecutable desde el repositorio de Blynk y se creó la instancia dentro de la plataforma de Google Cloud con el uso de una imagen de Ubuntu, donde mediante conexión SSH con el servidor, se descargó e instaló dentro de la máquina los paquetes correspondientes a "openjdk-8-jdk", el cual es el *software* útil para la ejecución del archivo que contiene las características de la plataforma de Blynk *legacy.*

Para la ejecución del archivo, se ejecutó el comando "java -jar [nombre-de-archivo] -*dataFolder* /[ubicación-archivo-java]" con permisos de administrador desde el servidor.

Una vez ejecutado el archivo, el terminal del servidor imprime lo mostrado en la Figura 3.2 donde se puede evidenciar el *url* del servidor para la edición de características y cuadros de mando creados para cada proyecto; además brinda el usuario y contraseña que permiten el ingreso en la aplicación de Blynk y en el servidor

```
root@blynkserver:~# java -jar server-0.41.0-java8.jar -dataFolder /home/pi/Blynk

Blynk Server 0.41.1-SNAPSHOT successfully started.
All server output is stored in folder '/home/chuntezuka/logs' file.
Your Admin url is https://10.142.0.2:9443/admin
Your Admin login email is admin@blynk.cc
Your Admin password is admin
```

Figura 3.2. Ejecución servidor Blynk

Adicionalmente, se instaló la aplicación de Blynk en el móvil para la configuración de la interfaz para el usuario; siendo que, para el ingreso, se definió los parámetros de conexión como credenciales de acceso y la dirección IP pública del servidor local con el respectivo puerto utilizado para la comunicación como se muestra en la Figura 3.3.

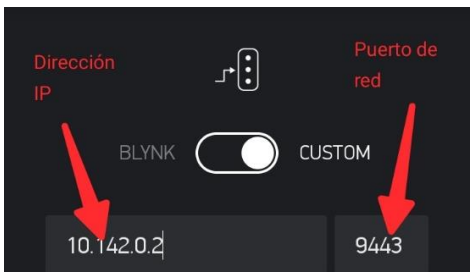

Figura 3.3. Configuración parámetros de conexión Blynk

Una vez configurados los parámetros del servidor, se crea una cuenta de usuario, como se presenta en la Figura 3.4; para permitir el guardado del cuadro de mando correspondiente al proyecto; esta cuenta permite al usuario contar con credenciales propias para el control del dispositivo, garantizando el uso autorizado del dispensador remotamente a través de la aplicación.

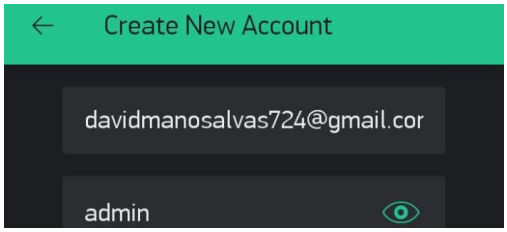

Figura 3.4. Creación de cuenta de usuario

Contando con una cuenta para el usuario, se creó el proyecto correspondiente, especificando el nombre del proyecto, el dispositivo o placa que se conectará a la aplicación, y el tipo de conexión con el que contará. Estas características se pueden observar en la Figura 3.5.

Figura 3.5. Creación del proyecto

Dentro del proyecto, se definió los bloques necesarios para la interacción entre usuario y dispositivo; como se muestra en la Figura 3.6, se estableció un total de 11 bloques necesarios para la creación del cuadro de mando interactivo.

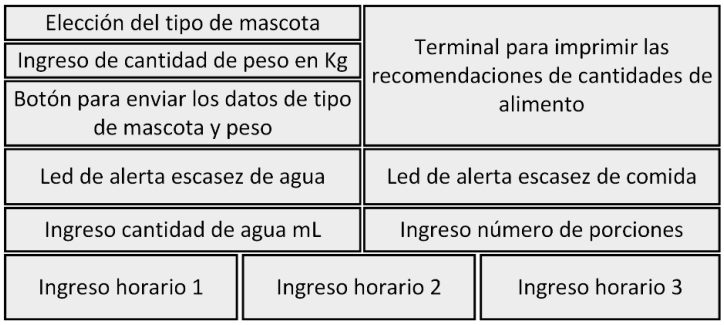

Figura 3.6. Diagrama de bloques para Dashboard

Estos bloques fueron plasmados dentro de la aplicación con el uso de los *widgets*, mostrados en la Figura 3.7, que permite agregar la aplicación en la plantilla del proyecto.

Gracias al uso del servidor local para la plataforma Blynk no se tiene limitación en el uso de *widgets*, esto permite al usuario utilizar una gran variedad de estos acomodándose al tipo de interacción que se necesita para el usuario.

Estos *widgets* se dividen en grupos de acuerdo a su funcionamiento; en este proyecto se utilizan *widgets* para controlar, mostrar e interfaces.

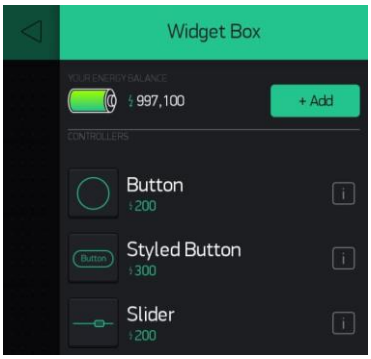

Figura 3.7. Panel de *widgets* para cuadro de mando

Para permitir la interacción del usuario de manera intuitiva, se utilizaron *widgets* que faciliten el ingreso y obtención de datos. Así, para la configuración de horarios se colocó tres *widgets* 'interfaces' de tiempo *(uno para cada horario)*, que permitan el ingreso del respectivo horario en formato 'HH:mm', los parámetros para el *widget* se observan en la Figura 3.8. Cabe recalcar que cada *widget* va enlazado a un pin virtual para su configuración lógica en la programación del microcontrolador.

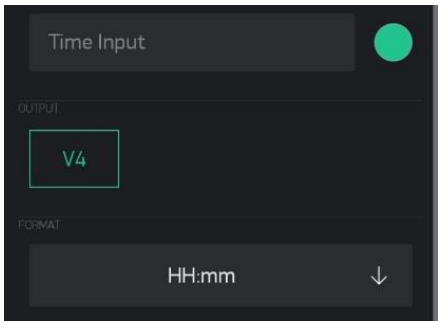

Figura 3.8. *Widget* de tiempo

Además, dentro del grupo de *widgets* 'interfaces', también se hizo uso de entradas numéricas como se muestra en la Figura 3.9. Dentro de los parámetros, se estableció un rango de valores de ingreso de acuerdo a la variable necesaria, donde se definió un rango de 0 a 10 para cantidad de porciones, de 0 a 40 para valor de peso, y de 0 a 800 para ingreso de cantidad de agua en mililitros.

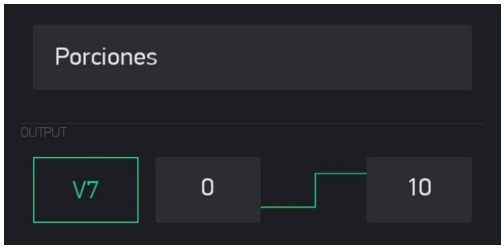

Figura 3.9. *Widget* de entrada numérica

En adición, se colocó un *widget* 'interfaz' tipo menú, este permite la elección del usuario sobre el tipo de mascota para la generación de la recomendación de cantidades para dispensar. Los parámetros configurados se muestran en la Figura 3.10.

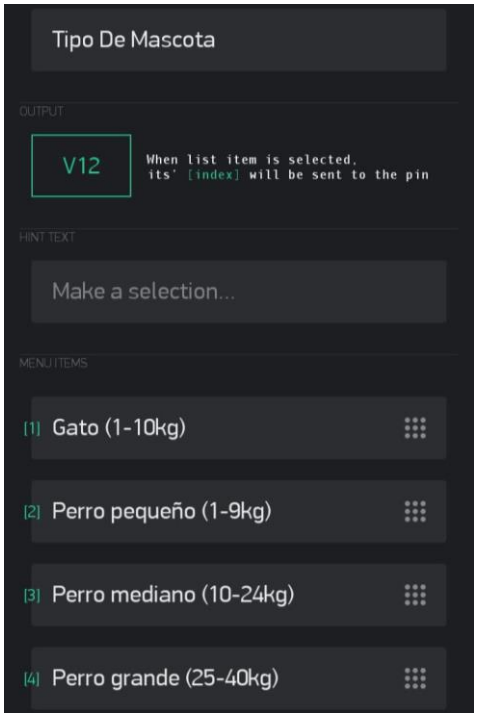

Figura 3.10. *Widget* tipo menú

Ahora, dentro del grupo de *widgets* 'para mostrar' se usó un par de leds que permitan evidenciar la alerta de escasez de agua y comida. La configuración de este *widget* se presenta en la **Figura *3.11***.

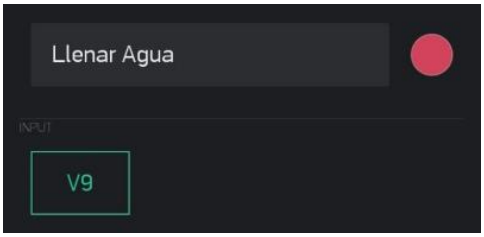

Figura 3.11. *Widget* led

Igualmente, dentro del mismo grupo se escogió el *widget* tipo terminal que permita la impresión de texto para mostrar al usuario la recomendación de las cantidades sugeridas para su tipo de mascota. Este *widget* mostrará la información requerida una vez pulsado el *widget* tipo botón encontrado en el grupo de '*controladores'.*

Los parámetros establecidos para estos *widgets* se observan en la Figura 3.12 y Figura 3.13, respectivamente.

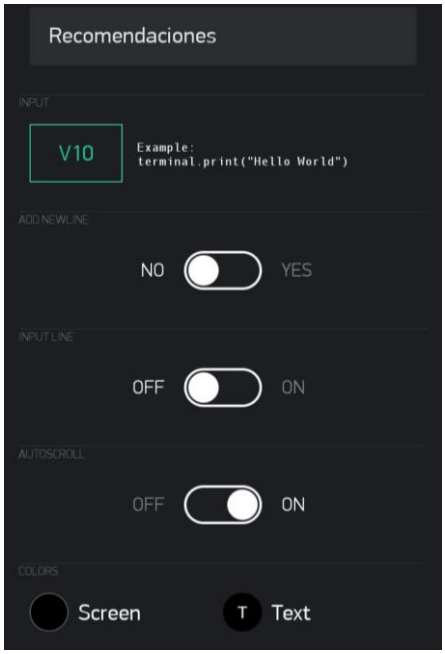

Figura 3.12. *Widget* tipo terminal

Figura 3.13. *Widget* tipo botón

Finalmente, dentro del apartado de configuración del proyecto se copió el *token* correspondiente al proyecto para definirlo dentro del código de programación permitiendo la autenticación del dispositivo con la aplicación. Para esto, se pulsó el botón "*copy all"* que se muestra en la Figura 3.14.

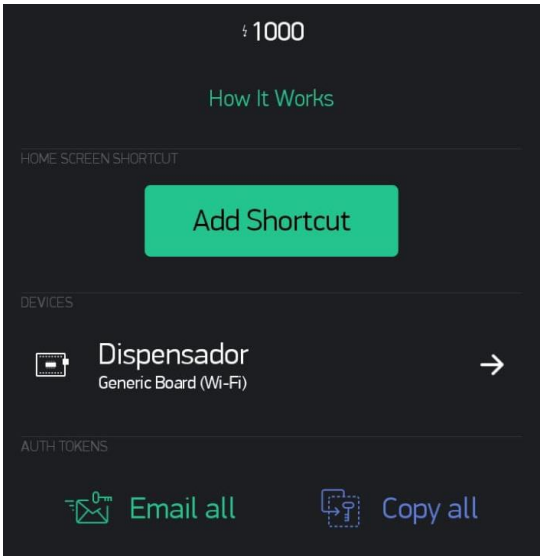

Figura 3.14. Configuración proyecto

Así, dentro del cuadro de mando, se estableció el tamaño de cada *widget* y se los colocó de manera organizada obteniendo la interfaz de usuario final configurada, la cual se muestra en la Figura 3.15.

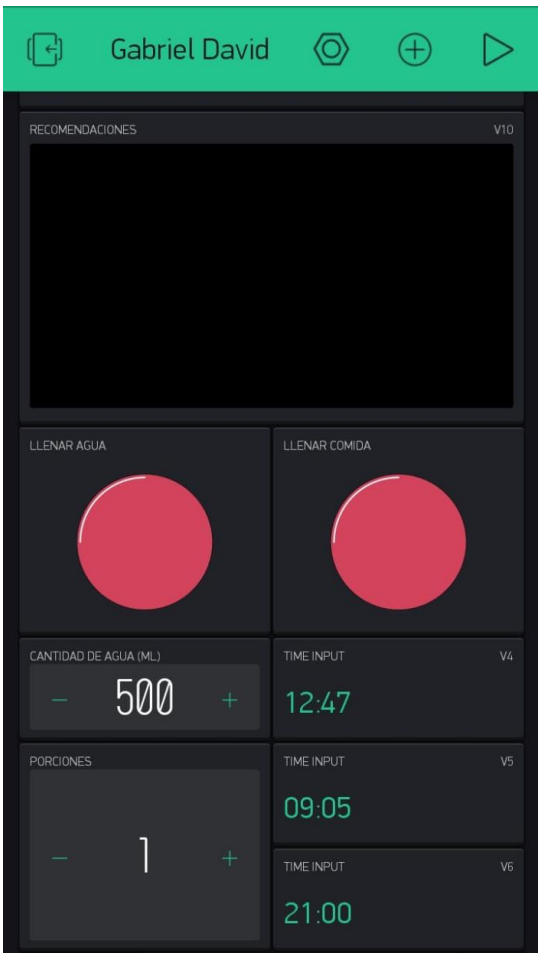

Figura 3.15. Interfaz de control Blynk

Creación del esquema de opciones para control Telegram

Para el control remoto desde la aplicación de Telegram, se definió el índice de opciones a mostrar para la definición de cada parámetro e ingreso a subíndices; para ello, se tomó de referencia la interfaz configurada para Blynk de manera que, en ambas plataformas, tanto Telegram como Blynk, se tengan las mismas opciones de configuración para el usuario. De esta forma, se realizó un diagrama con las opciones que se deben presentar en el chat de Telegram como se muestra en la Figura 3.16.

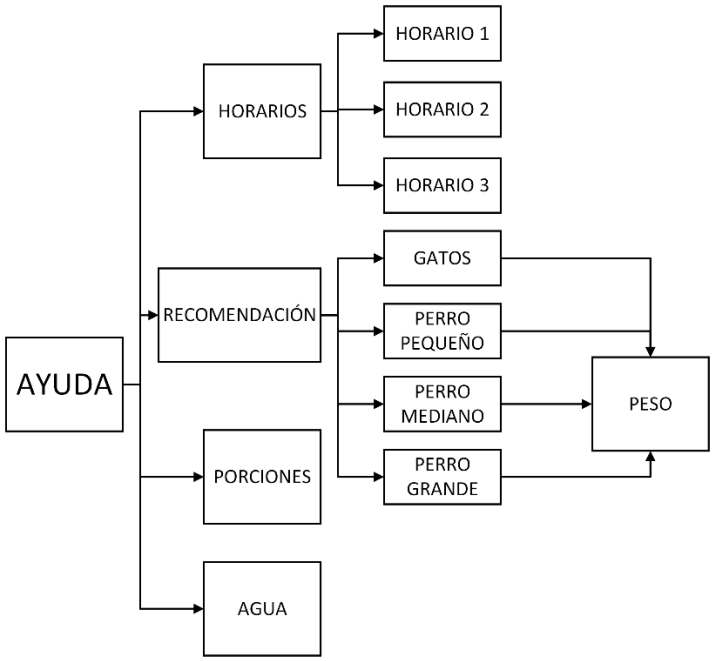

Figura 3.16. Diagrama de opciones Telegram

Creación del *Bot* de Telegram

Para permitir tanto la recepción como transmisión de mensajes entre el dispositivo controlador y el usuario mediante la plataforma de mensajería Telegram, es necesaria la configuración de un *bot* el cual permita el ingreso a un chat donde se presentaran los mensajes referentes a instrucciones, índices, alertas y recomendaciones, permitiendo una interfaz interactiva y que garantice una comunicación bidireccional, donde no únicamente el usuario ingresa comandos, sino que puede solicitar información o recibir mensajes de alertas sobre algún acontecimiento medido por los sensores conectados al microcontrolador.

Como herramienta para la creación del *bot*, se hizo uso de *BotFather*, una herramienta que permite la administración de *bots* dentro de la plataforma de Telegram. Como se muestra en la Figura 3.17, únicamente fue necesaria la asignación de un nombre al *bot* para la obtención del token que permite la configuración para la placa de desarrollo.

Figura 3.17. Creación del *bot* de Telegram

Por otra parte, para la configuración de la comunicación limitada a únicamente el usuario dueño de la mascota a través de la aplicación de Telegram, se hizo uso del *bot* denominado 'ID*Bot*' para la obtención del id del chat del usuario para su definición en el código de programación. Cabe recalcar que el uso de un *bot* dentro de Telegram permite la recepción de mensajes desde cualquier usuario, siendo que el microcontrolador lee todos los mensajes y envía sus respuestas al dueño, donde la función limitada a otros usuarios es la recepción del índice de comandos necesarios para el dispensador. La obtención del ID correspondiente al chat se muestra en la Figura 3.18.

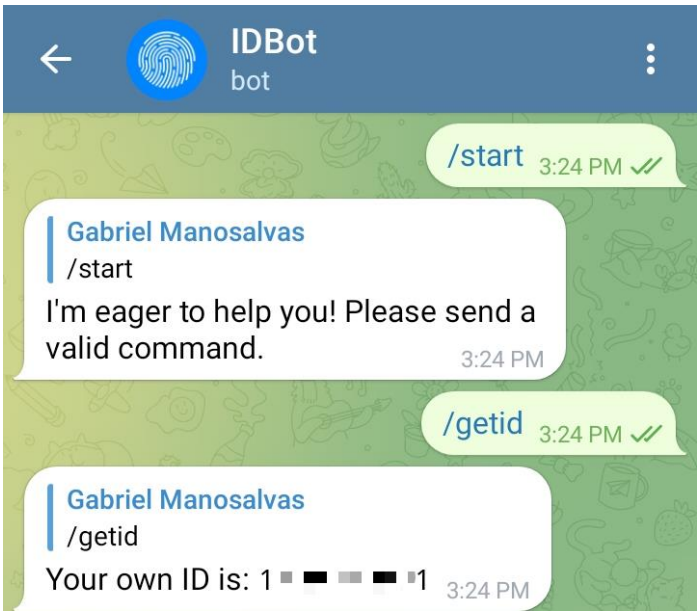

Figura 3.18. Obtención de ID

Creación del diagrama de flujo para cada función del dispensador

Finalmente, se desarrolló el diagrama de flujo para definir cada función, variable e instrucción necesaria para el funcionamiento e interpretación en código de programación para la configuración de la placa de desarrollo.

En la Figura 3.19 hasta la Figura 3.34, se presenta los diferentes bloques del diagrama de flujo, que representan cada función necesaria para programar el microcontrolador.

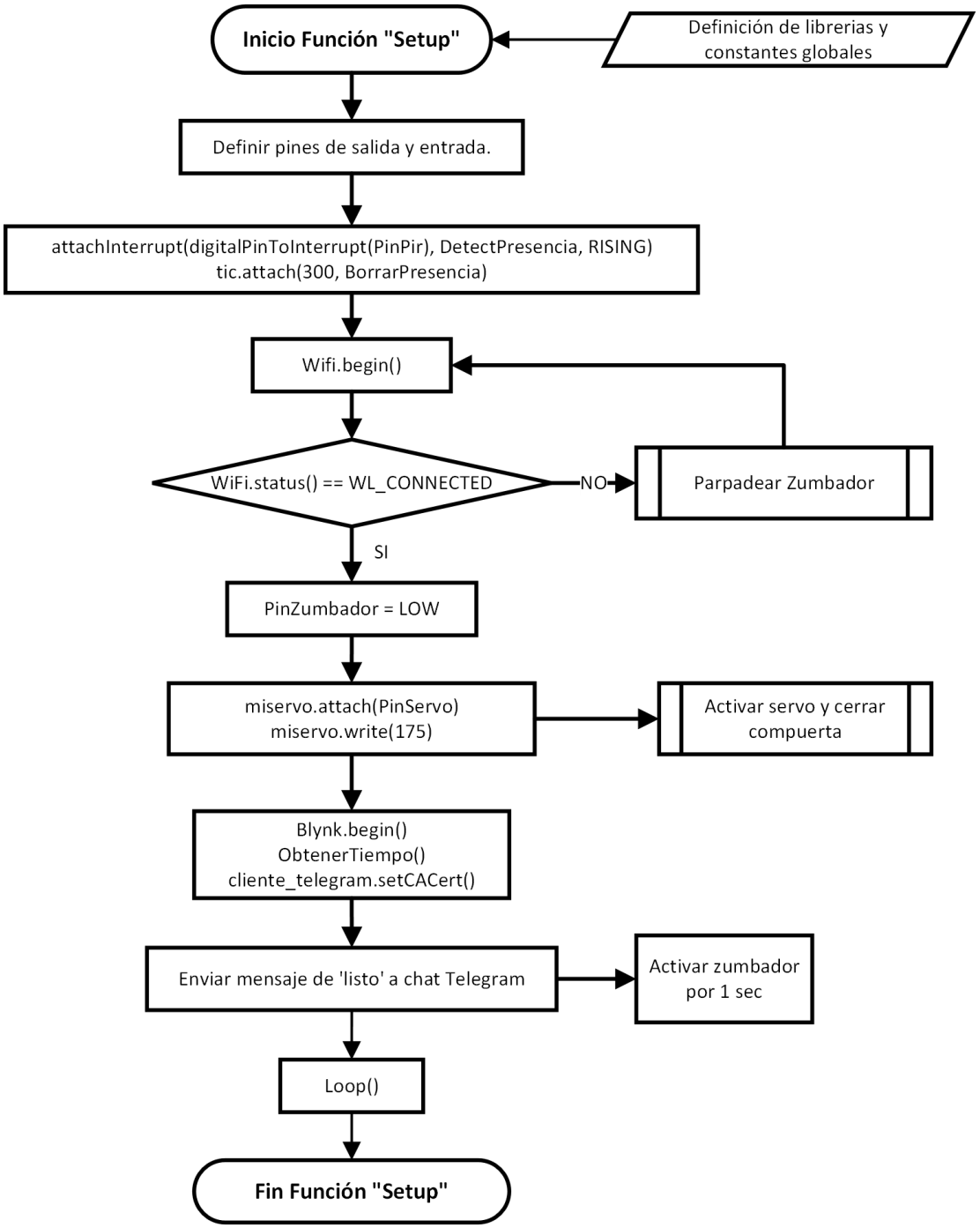

Figura 3.19. Diagrama de flujo función principal

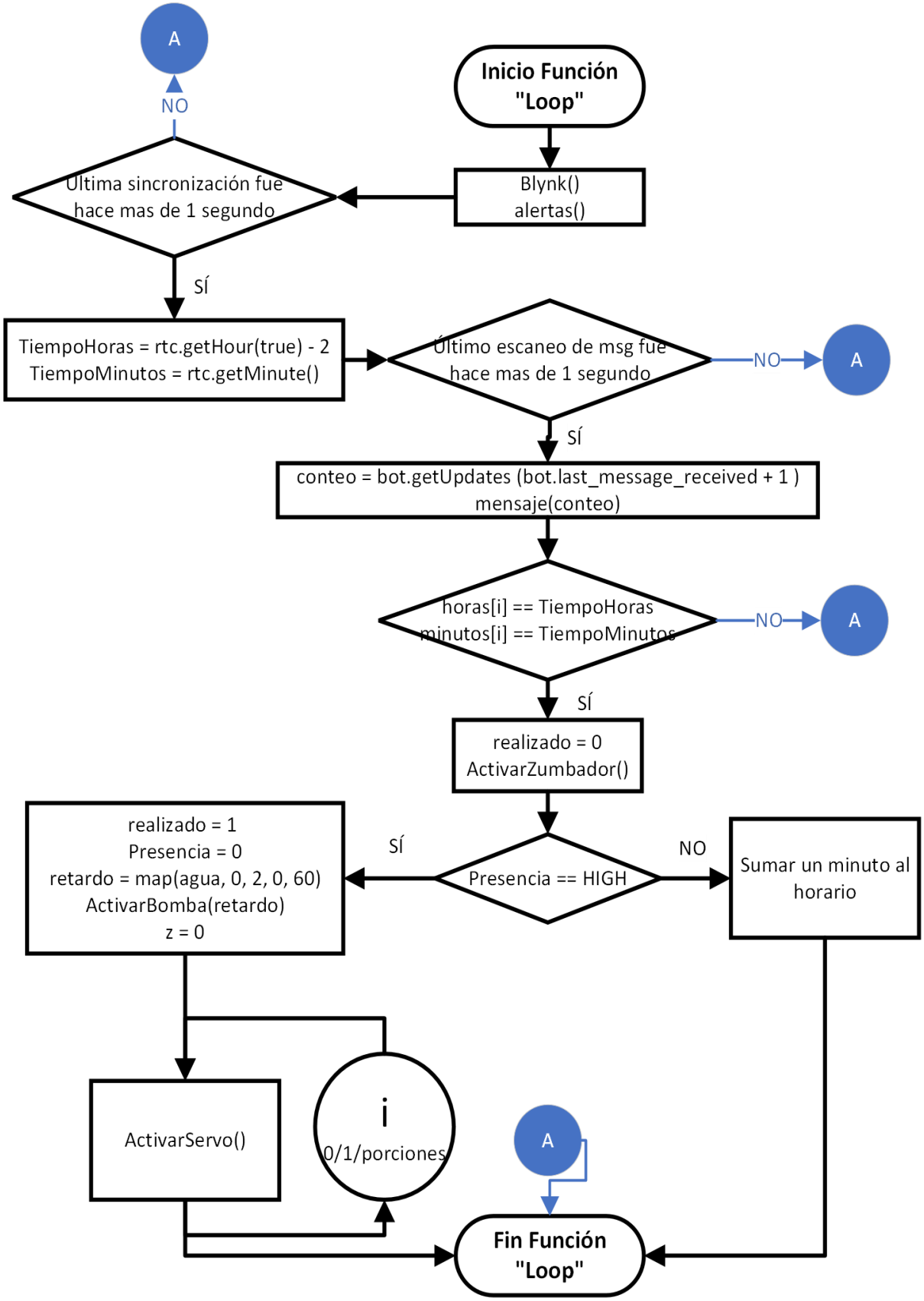

Figura 3.20. Diagrama de flujo función “Loop”

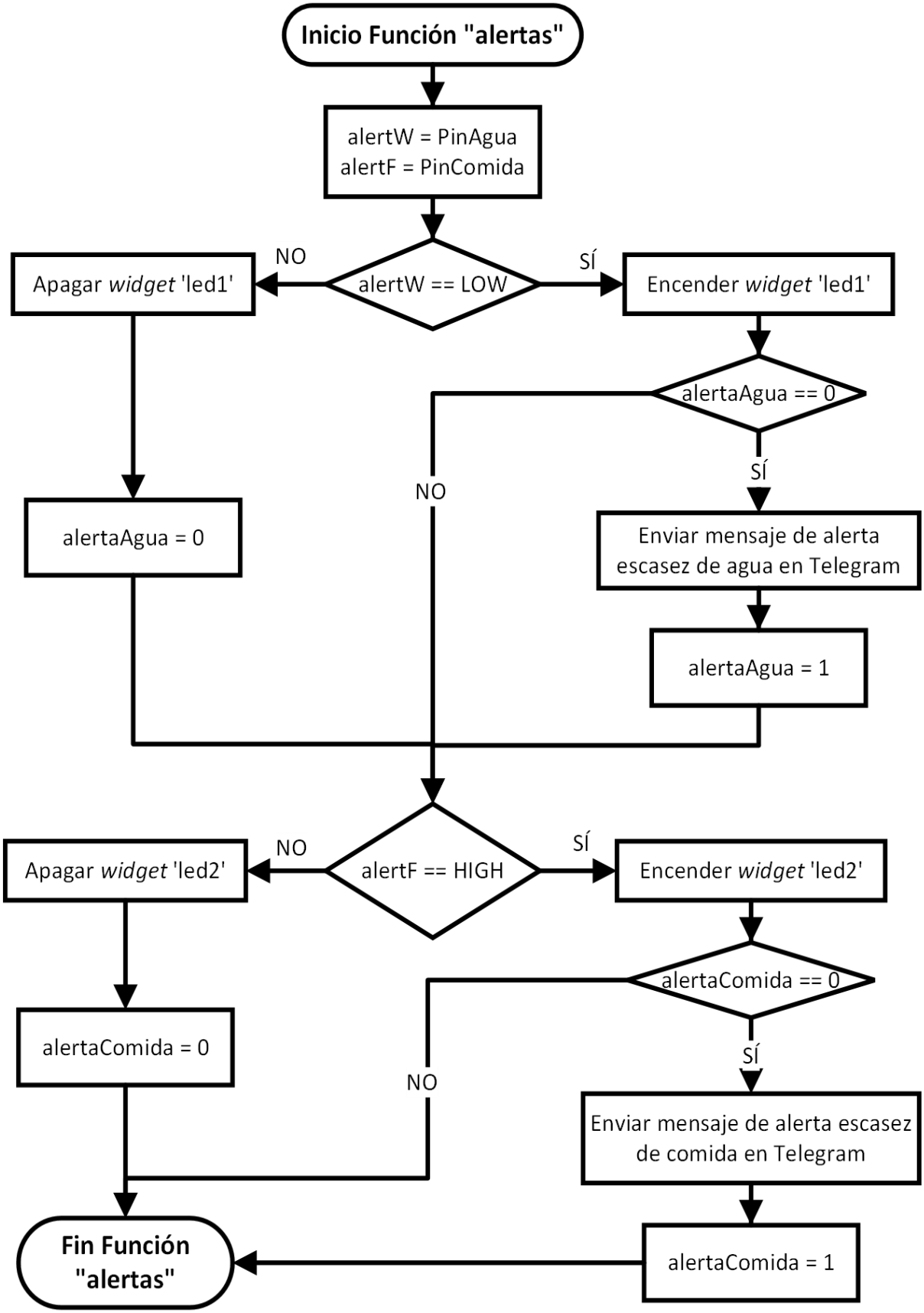

Figura 3.21. Diagrama de flujo función “Alertas”

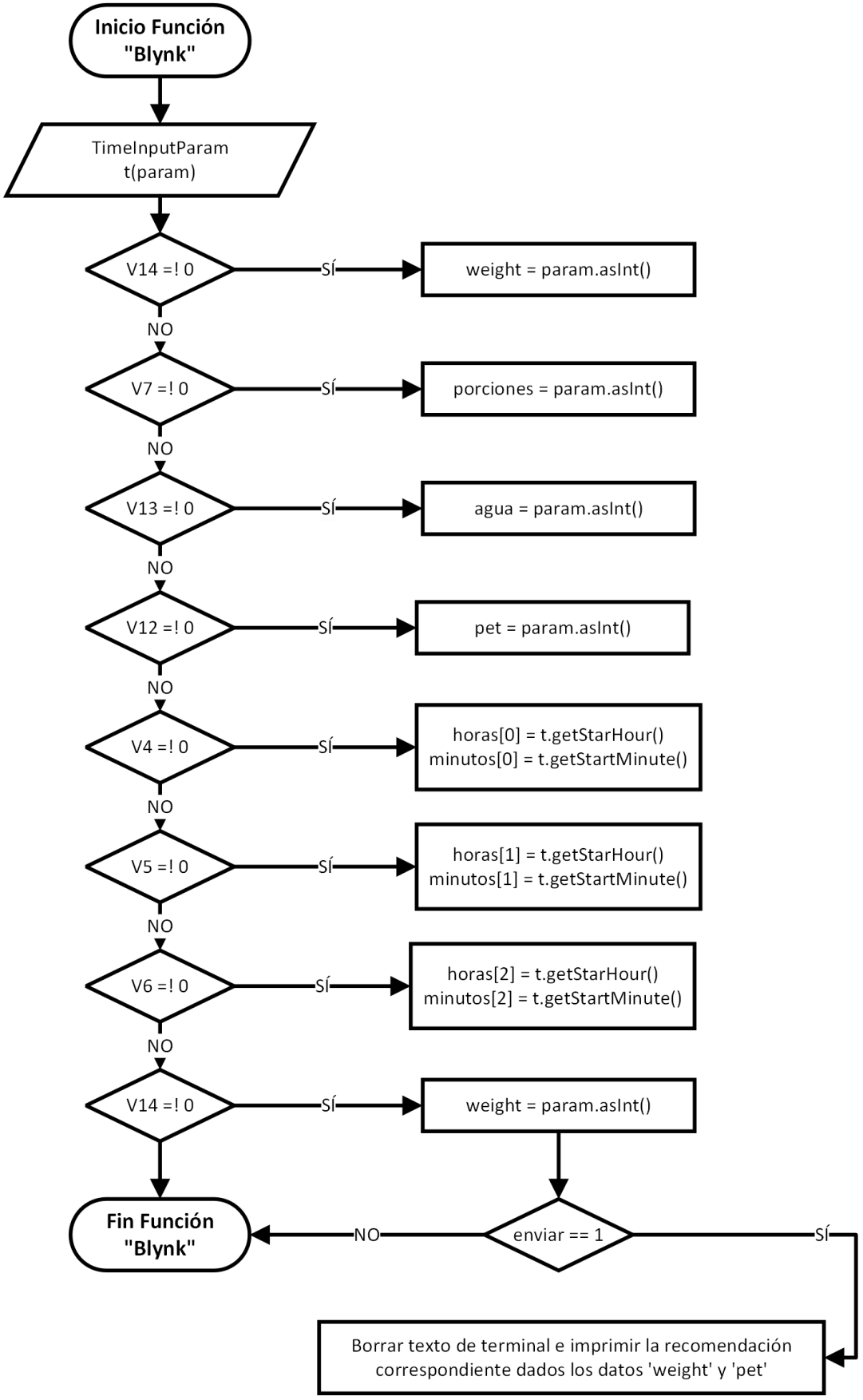

Figura 3.22. Diagrama de flujo función “Blynk”

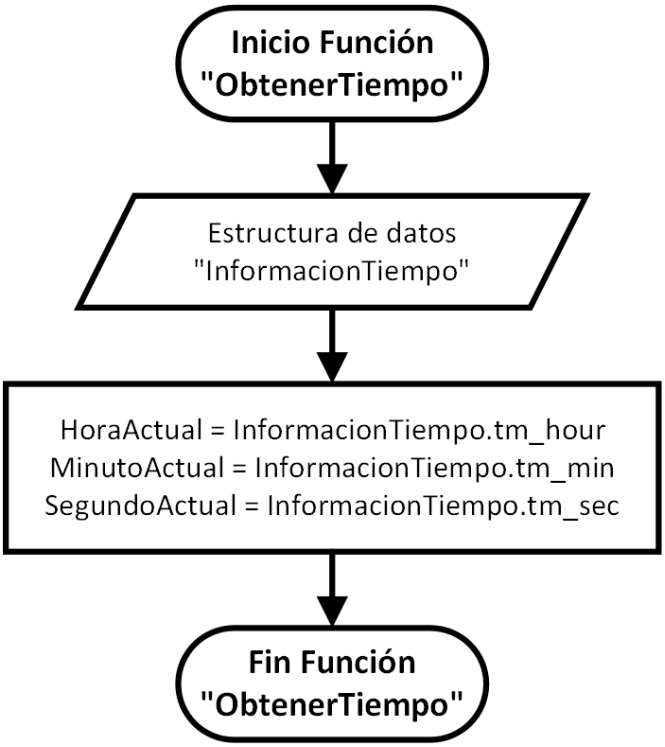

Figura 3.23. Diagrama de flujo función “ObtenerTiempo”

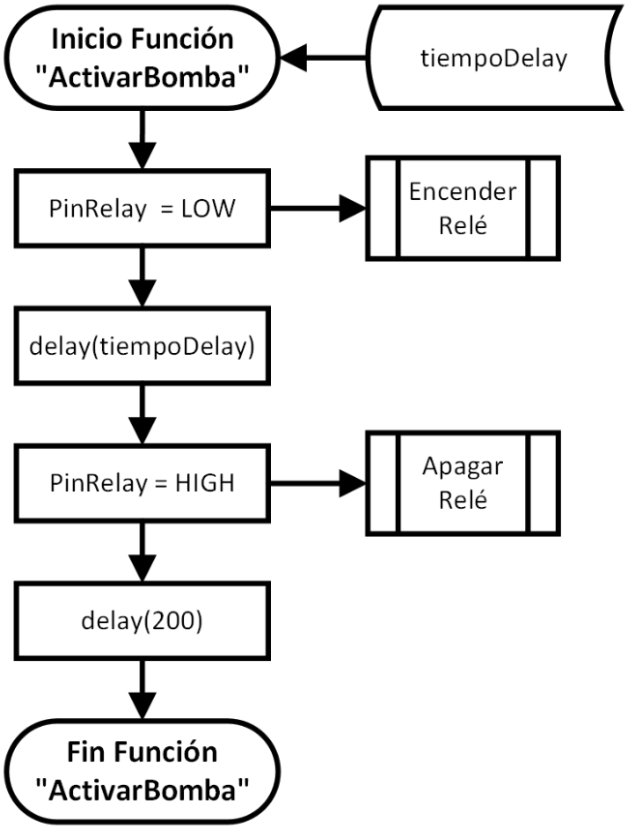

Figura 3.24. Diagrama de flujo función "ActivarBomba"

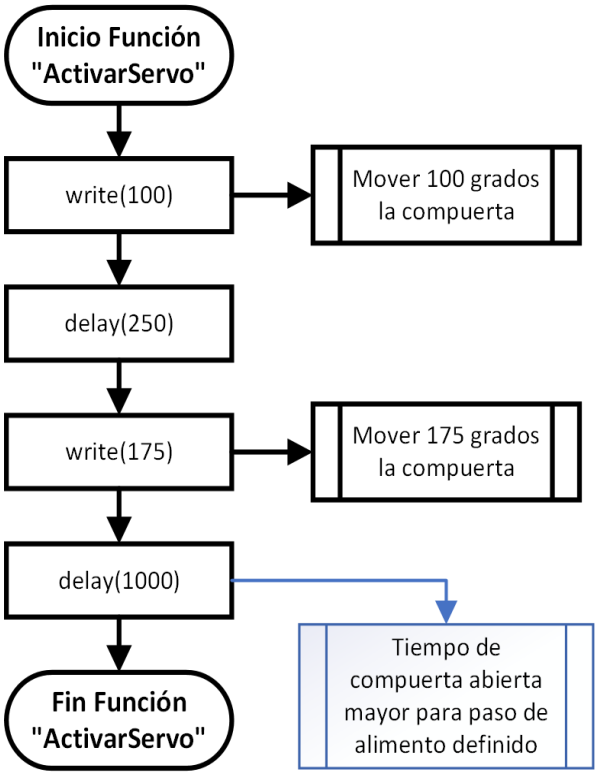

Figura 3.25. Diagrama de flujo función "ActivarServo"

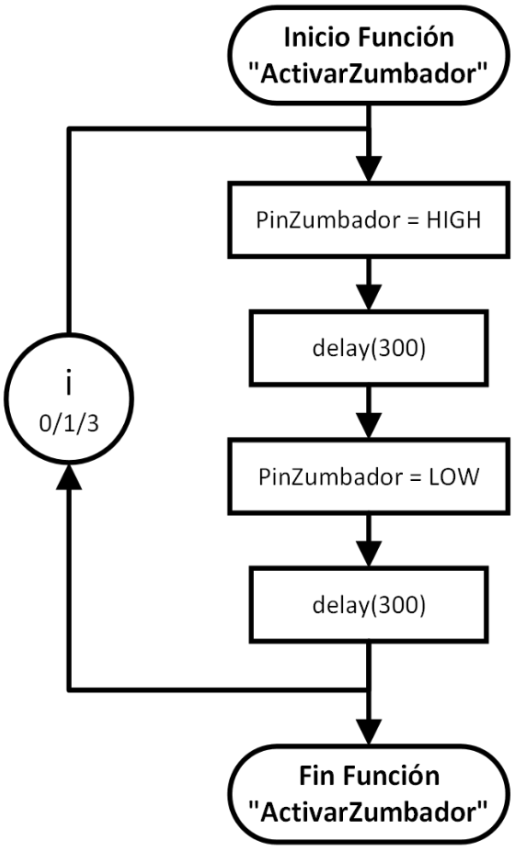

Figura 3.26. Diagrama de flujo función "ActivarZumbador"

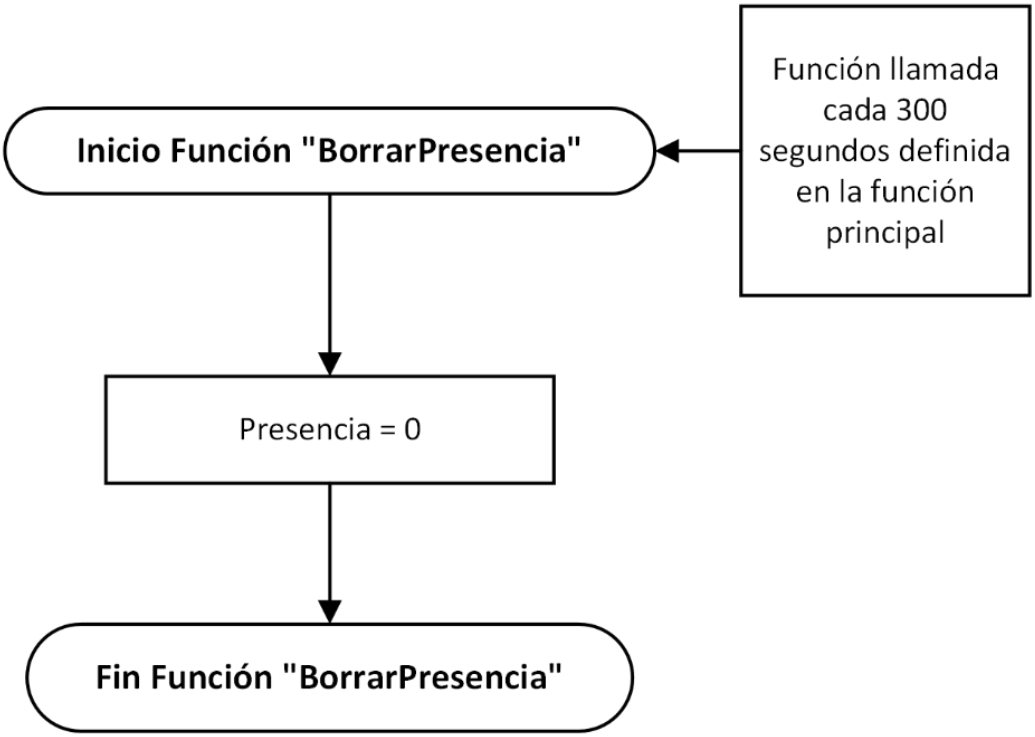

Figura 3.27. Diagrama de flujo función "BorrarPresencia"

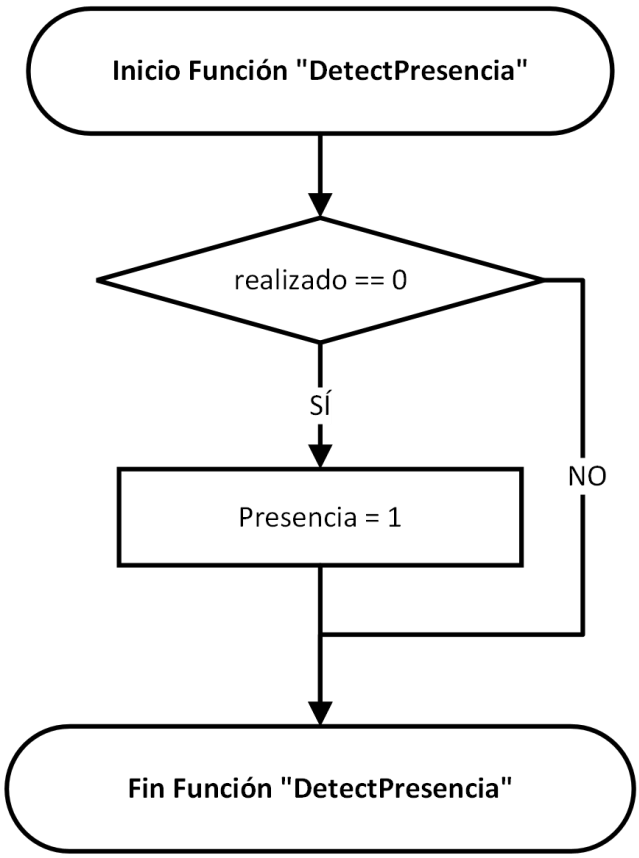

Figura 3.28. Diagrama de flujo función “DetectPresencia”

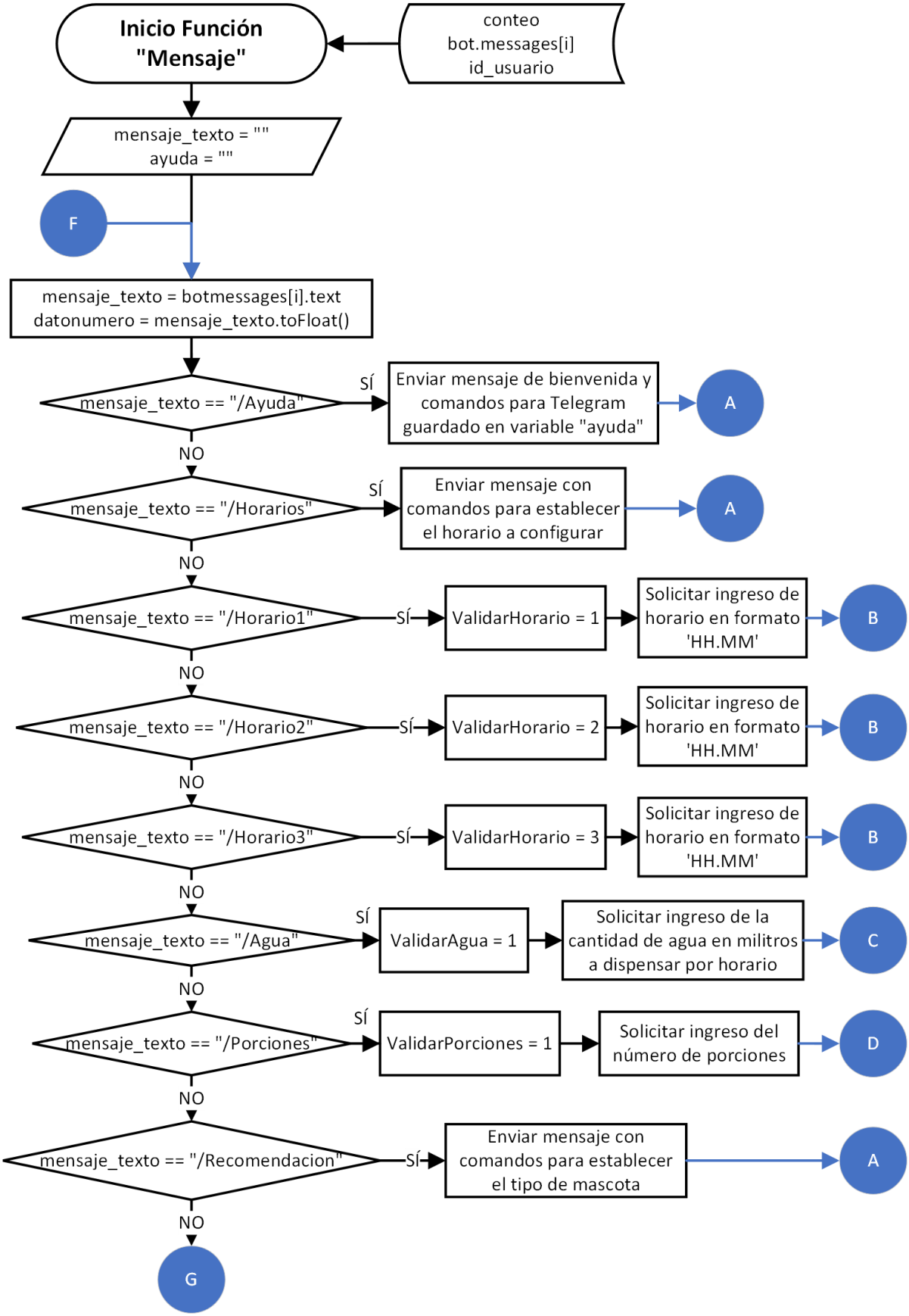

Figura 3.29. Diagrama de flujo función “Mensaje” parte A

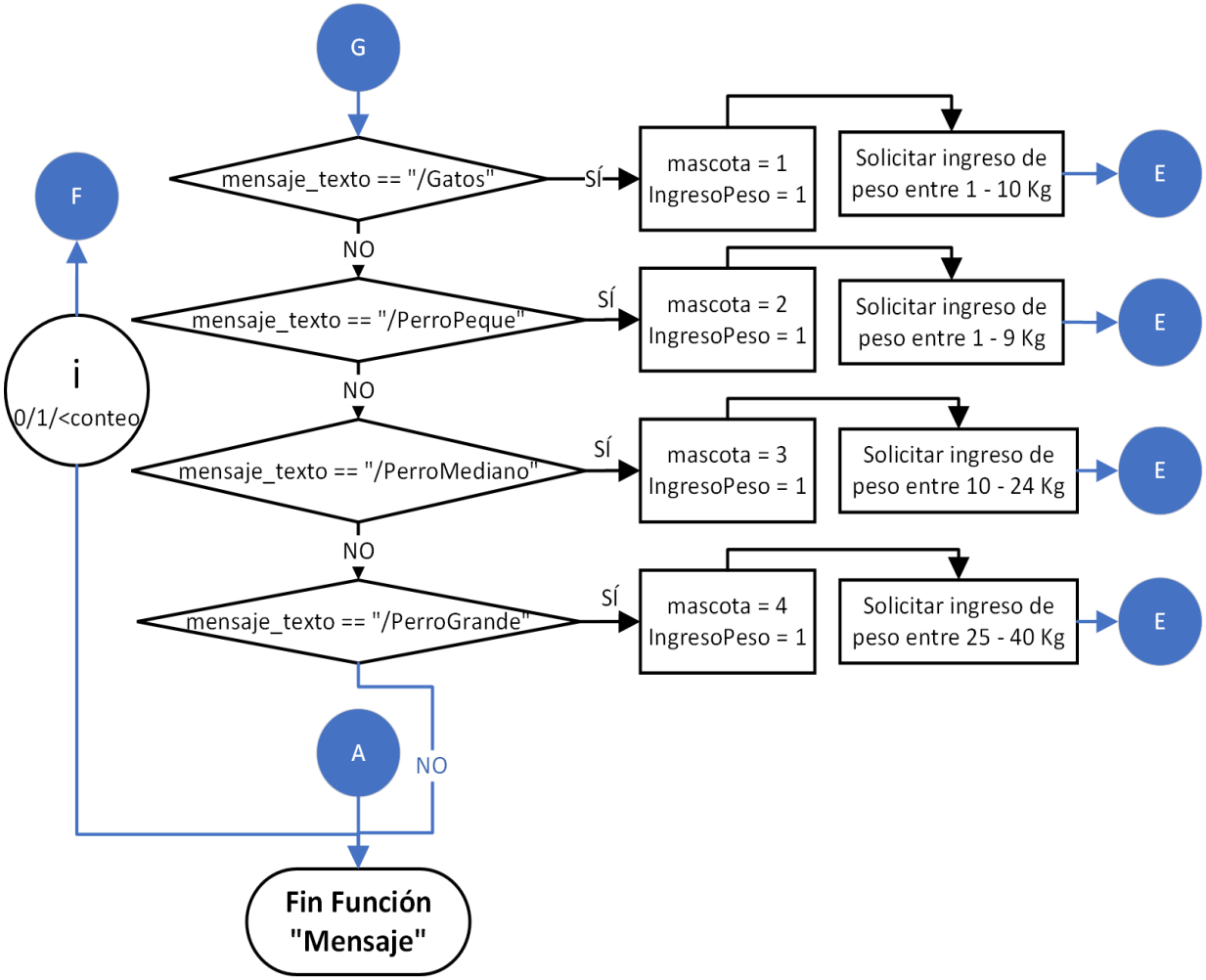

Figura 3.30. Diagrama de flujo función “Mensaje” parte B

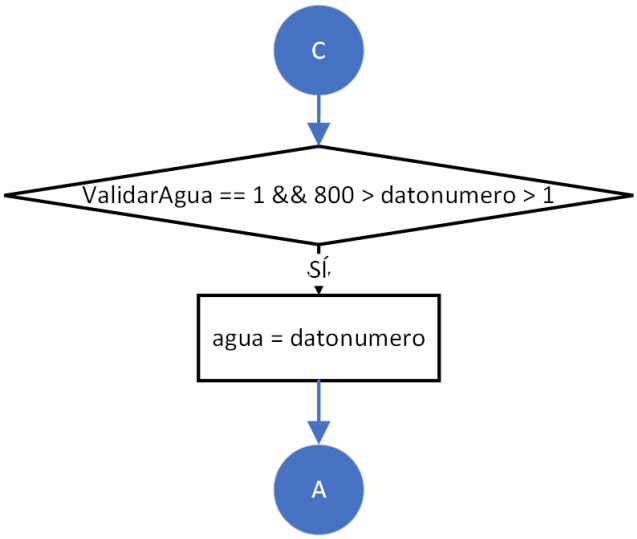

Figura 3.31. Diagrama de flujo función “Mensaje” parte C

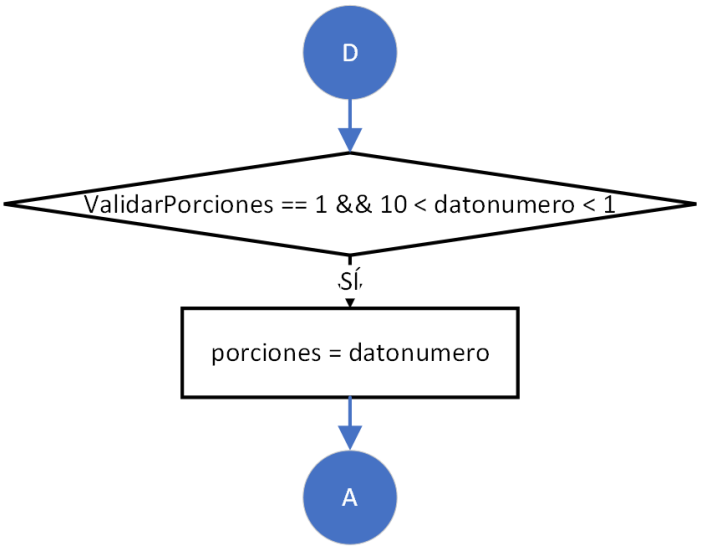

Figura 3.32. Diagrama de flujo función “Mensaje” parte D

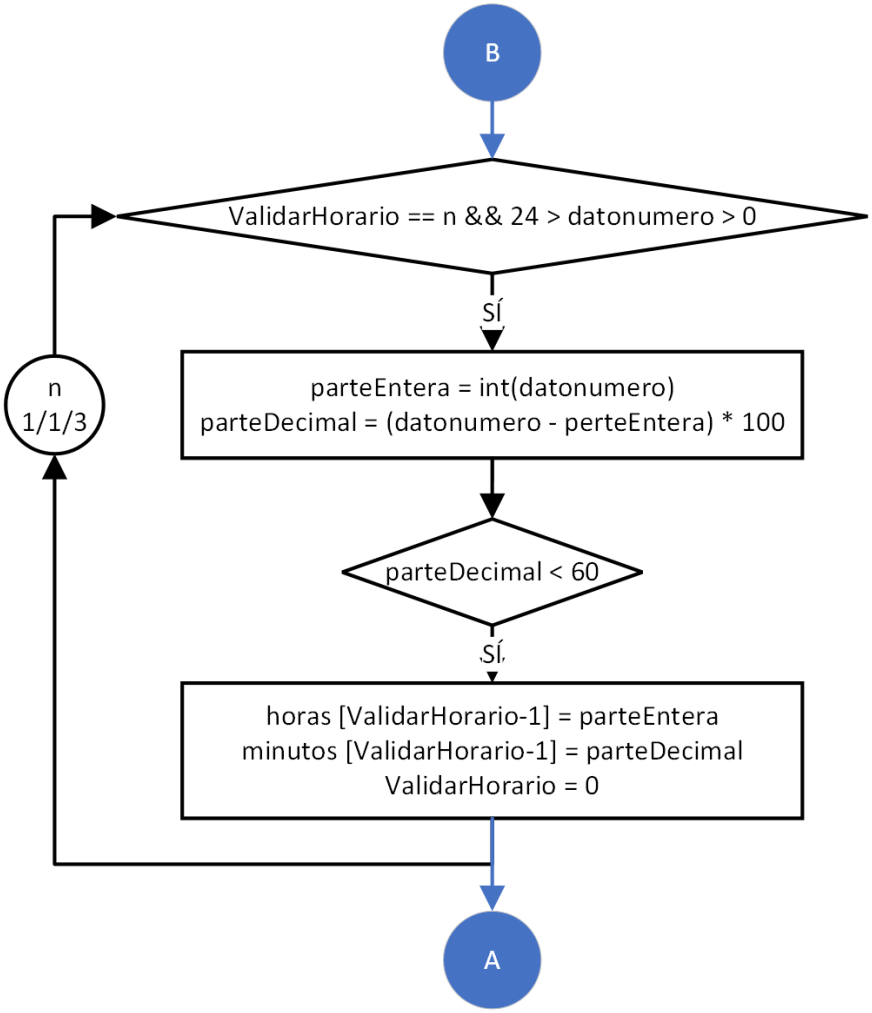

Figura 3.33. Diagrama de flujo función “Mensaje” parte E

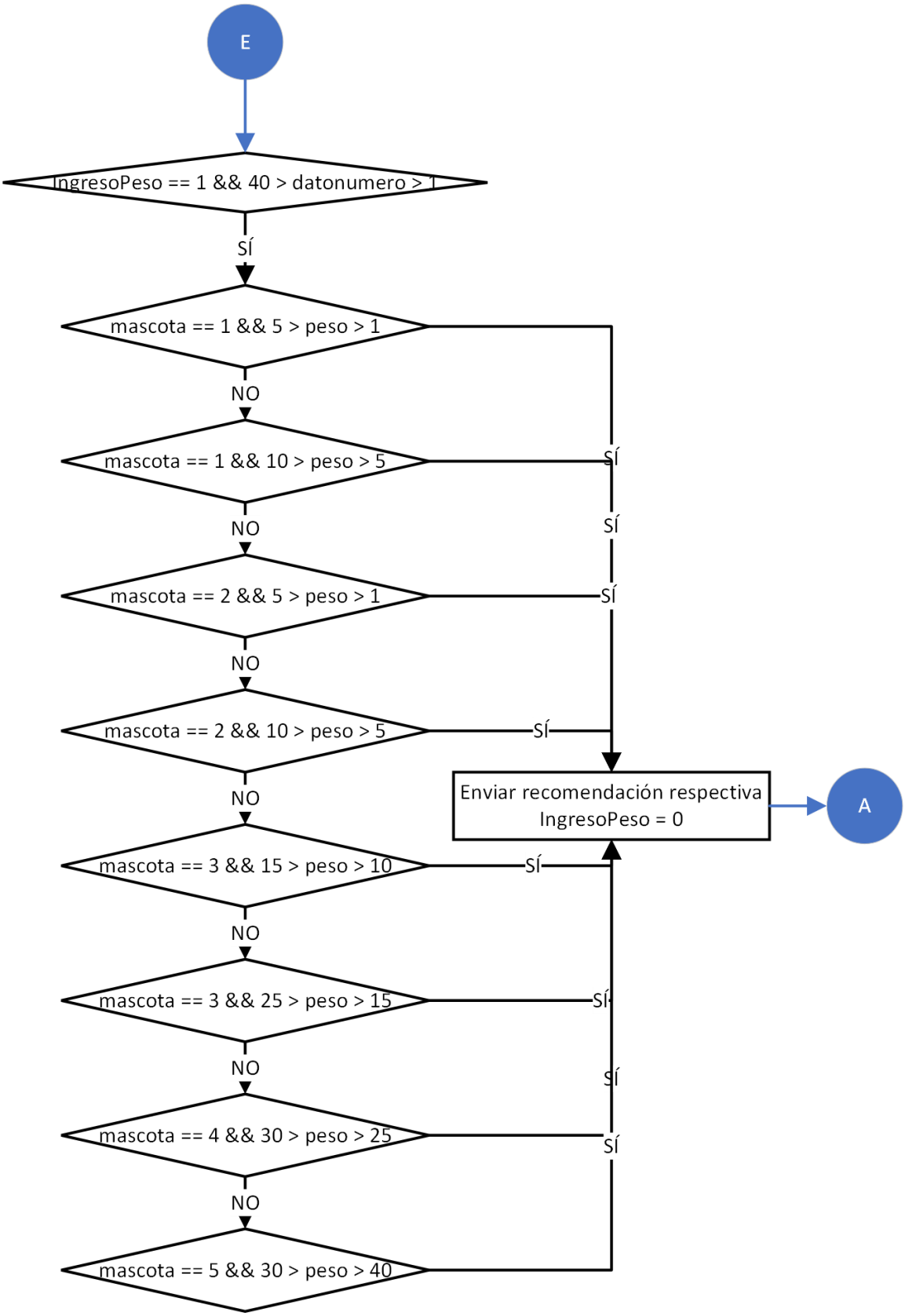

Figura 3.34. Diagrama de flujo función “Mensaje” parte F

Diseño del circuito electrónico

Previamente al desarrollo del PCB, se realizó el esquema de conexiones para el circuito electrónico del prototipo mediante el uso del *software* de simulación Proteus; donde, como se puede observar en la Figura 3.35, se conecta cada uno de los pines del microcontrolador a hileras de pines de comunicación que representan los *headers* que ayudarán con la conexión de los cables jumper para cada sensor y actuador necesario para el dispositivo, además del uso de tres hileras adicionales para alimentación de 5V, 3.3V y tierra o GND.

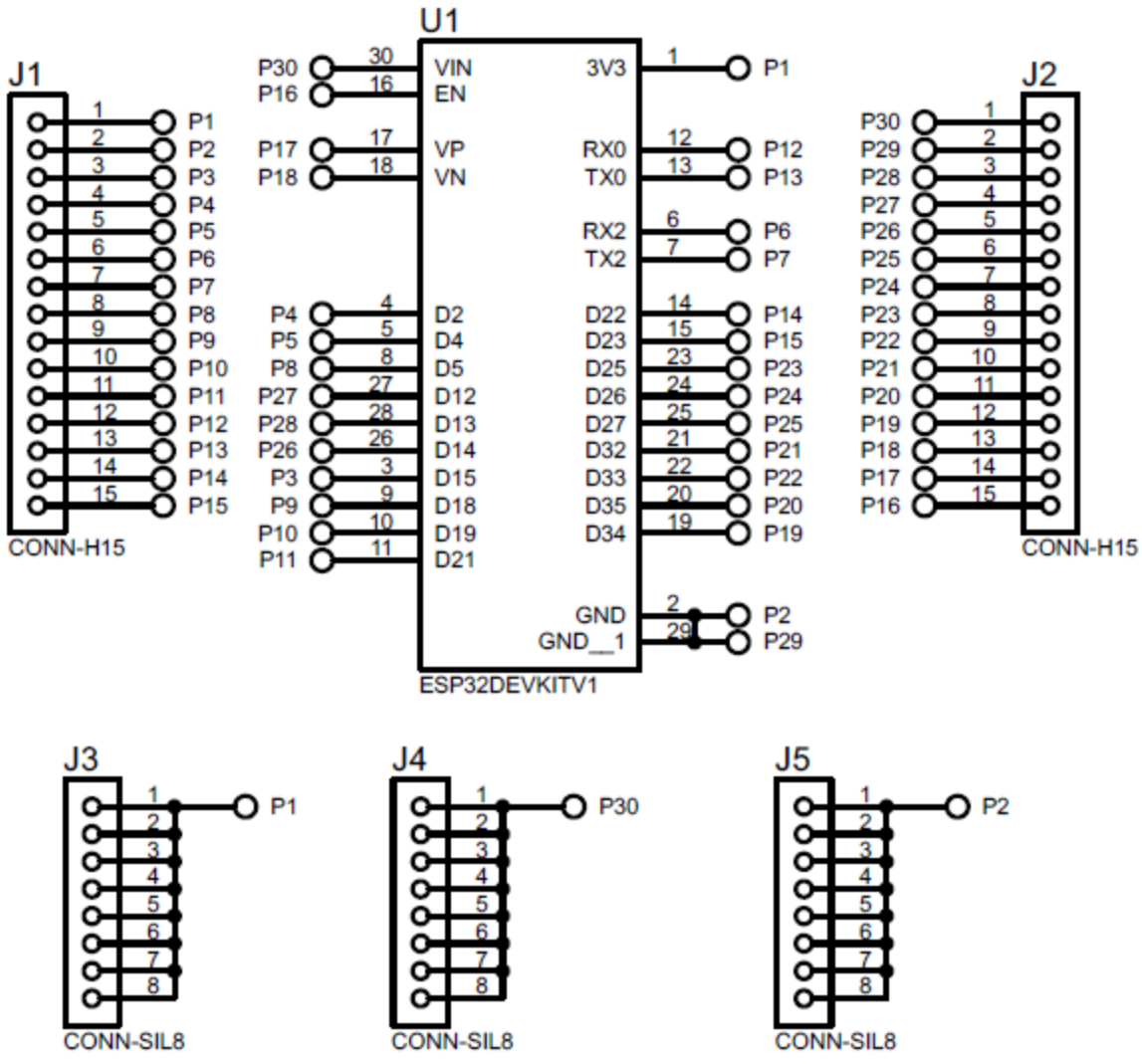

Figura 3.35. Esquema de conexiones circuito electrónico

Elaboración de las recomendaciones de alimento

Finalmente, previo a la implementación del dispensador con el armado de la maqueta, se realizó la construcción de un prototipo del mecanismo de dispensación de comida

para la mascota donde se lo puso a prueba y se recolectó las cantidades dispensadas para su medición de peso para la elaboración de recomendaciones de acuerdo a la cantidad de comida sugerida y la cantidad dispensada por porción, es decir, cada giro del servomotor. Así, se obtuvo la cantidad promedio aproximada de 50 (g) de alimento seco por porción como se muestra en la Figura 3.36 y Figura 3.37.

Figura 3.36. Primera muestra de comida

Figura 3.37. Segunda muestra de comida

Para la elaboración de las recomendaciones, se tomó un estudio sobre la cantidad de alimento seco recomendado por día tanto para perros como para gatos, donde se obtiene la recomendación de 2.5% del peso del perro en kilogramos de alimento seco diario [22], mientras que para los gatos se establece la ingesta de 40 (g) de alimento seco por kilogramo de peso de la mascota [23].

Así, se obtiene la Tabla 3.4 sobre el consumo recomendado de alimento para una mascota que mantiene actividad promedio y que no presenta ninguna enfermedad que pueda afectar a las cantidades recomendadas y, asimismo, se encuentra en un peso óptimo para su edad.

Por otra parte, para la elaboración de recomendaciones de ingesta de agua diaria, se obtuvo la siguiente información: en el caso de gatos, estos deben ingerir un máximo de 100 (ml) de agua por kilogramo de peso [24]; en cambio, los perros deben tomar un aproximado de 50 (ml) de agua por kilogramo de peso [25].

De esta forma, se elabora la tabla de cantidades recomendadas de agua.

Cabe recalcar, que las cantidades mostradas en la Tabla 3.4 representa las cantidades máximas sugeridas para el consumo de la mascota de forma diaria, siendo que estas cantidades se ven afectadas por el nivel de actividad, edad, temperatura del ambiente, tipo de raza, entre otras características adicionales de la mascota y su entorno.

Tabla 3.4. Cantidad recomendada de alimento

Peso (Kg)	Cantidad de comida (g)	Cantidad de agua (ml)
GATOS		
1 a 5	200	500
5 a 10	400	1000
PERROS		
1 a 5	125	125
5 a 10	250	250
10 a 15	375	375
15 a 25	625	625
25 a 30	750	750
30 a 40	1000	1000

3.4 Implementación del prototipo de dispensador

Desarrollo del PCB

Una vez desarrollados los diagramas para el desarrollo del código de programación adjuntado en los anexos, se desarrolló el PCB para lo cual, se adquirió una baquelita virgen de dimensiones 5 (cm) x 5 (cm) donde se realizó el circuito de manera manual, siendo que se utilizó un marcador para la realización de cada una de las pistas que conecten los pines GPIO del microcontrolador con los *headers* del circuito siendo fieles a las rutas obtenidas mediante el *software* Proteus evitando gastos en 'papel transfer' e impresiones del circuito en láser como lo requiere el método del planchado para PCB. El diagrama de conexiones a ser dibujado en la baquelita se muestra en la Figura 3.38.

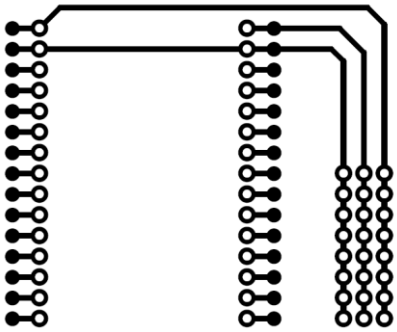

Figura 3.38. Diagrama de conexiones para PCB

Cabe recalcar que, para la realización de las pistas en la placa, se aplicó el efecto espejo al diagrama obtenido para la generación del circuito definido en Proteus al aplicarle el ácido férrico que elimina las partes de cobre que no se encuentren cubiertas por la tinta del marcador, formando así las pistas requeridas para el circuito. El diagrama plasmado en la baquelita se muestra en la Figura 3.39.

Figura 3.39. Circuito plasmado en placa virgen

Plasmado el esquema de conexiones obtenido en Proteus con su herramienta de auto enrutamiento de pistas en la placa virgen, además de un ícono y nombre como adicional para el PCB, se procede a la preparación del ácido dentro de un envase plástico para sumergir la placa como se muestra en la Figura 3.40, para obtener la PCB final.

Figura 3.40. Quemado del PCB

Seguido del quemado de la placa, se procedió a limpiar la tinta de las pistas conservadas con el uso de acetona para la obtención de la placa final mostrada en la Figura 3.41, donde se harán los agujeros para la ubicación de cada elemento del PCB. Adicional a esto, se comprobó la continuidad en cada conexión para evitar cortocircuitos.

Figura 3.41. PCB final obtenido

Medida la continuidad de las pistas y ubicadas las hileras de *headers* para cada pin, se procedió a soldar cada uno de los elementos como se muestra en la Figura 3.42, teniendo en cuenta que se usó *headers* en los pines correspondientes a la ubicación del microcontrolador para evitar dañar la placa de desarrollo y permitiendo el cambio del microcontrolador en caso de daños internos producidos.

Figura 3.42. Soldado de pines en PCB

Seguidamente, se pintó la placa en su parte frontal para mejor estética del PCB, además de la agregación de una imagen de la placa de desarrollo que indique la orientación de conexión del mismo y etiquetas para cada hilera de alimentación como se puede observar en la Figura 3.43.

Figura 3.43. Parte frontal PCB

Módulo del circuito

Ya fabricada la placa, se fijó el PCB en una superficie junto al módulo relé necesario para el uso de la mini bomba de agua como se es posible observar en la Figura 3.44; este módulo conformado, se ubicará protegido dentro de un compartimento en la maqueta. Cabe recalcar que la fabricación de la placa se define con la conexión de cada uno de los pines pertenecientes a la placa de desarrollo Esp32 con un pin hembra para cada uno, permitiendo el uso de módulos adicionales, asimismo, el uso de un relé de 2 canales garantiza el uso de una conexión extra en caso de necesitarla a futuro, obteniendo un proyecto escalable.

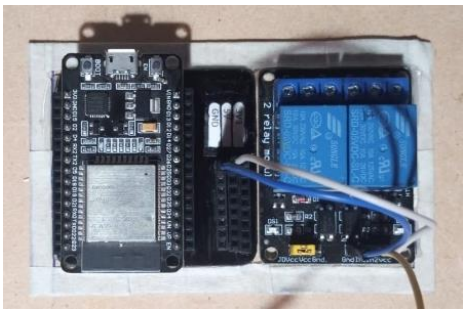

Figura 3.44. Módulo para conexiones

Elaboración de la maqueta

Para la realización de pruebas de funcionamiento, se construyó una maqueta con los mecanismos de dispensación necesarios para cumplir con los requerimientos planteados inicialmente, donde, basándose en modelos de dispensadores comerciales, se elaboró un esquema que muestre de manera básica la estructura definida para posteriormente plasmarla de forma física.

Como se observa en la Figura 3.45, se establece la construcción de dos compartimentos, uno para comida y otro para agua. En el compartimento de comida, se determina la incorporación de un sensor de obstáculos cerca de la compuerta manejada por el servomotor para el paso de comida; de manera que, si se termina la comida no se detectará ningún objeto cerca del sensor, por lo que, se enviará la señal para el envío de alerta de escasez de comida.

Por otro lado, se plantea la incorporación de un módulo sensor flotador para detección del nivel de agua para el compartimento del líquido, donde al encontrarse el nivel del agua por debajo del sensor, se envíe la señal para en este caso emitir al usuario la alerta de escasez de agua dentro del dispensador.

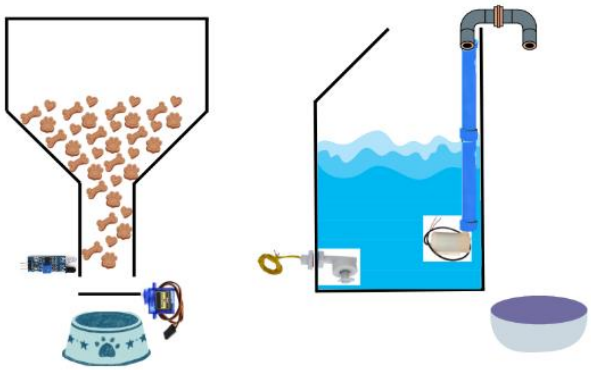

Figura 3.45. Estructura de la maqueta

Como puntos adicionales, se plantea la ubicación del sensor de movimiento en la parte frontal de la maqueta, con el fin de permitir la detección de la mascota cerca del dispositivo. Además, se define la elaboración de un espacio a parte que permita ubicar el módulo de conexiones con un agujero para el paso de los cables hacia su interior.

Finalmente, se estableció la integración de bebedero y plato para la comida integrados en el dispensador, evitando el uso de platos con un tamaño inadecuado.

Con el uso de *software* de diseño, se realizó un esquema de los contenedores de comida y agua, donde se presentan las medidas en centímetros de cada uno para integrarlos dentro de la maqueta.

Estos gráficos presentados en la Figura 3.46, representan al contenedor de comida en la parte izquierda el cual se encuentra armado con el uso de tubos PVC y uniones para darle la forma mostrada, mientras que el contenedor de agua consiste en un frasco plástico con tapa por el cual atraviesa una manquera para el paso de agua hacia el plato, líquido empujado hacia el exterior mediante el uso de una bomba de agua sumergible mostrada en la parte inferior del frasco.

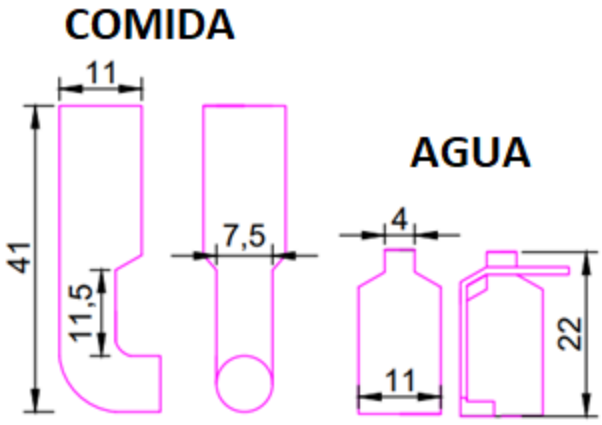

Figura 3.46. Esquema de compartimentos de alimento

Finalmente, se realizó el diseño de forma gráfica de la maqueta para implementarlo de forma física. Como se muestra en la Figura 3.47, se realizó el esquema de la parte frontal, lateral y superior de la maqueta con sus respectivas medidas en centímetros.

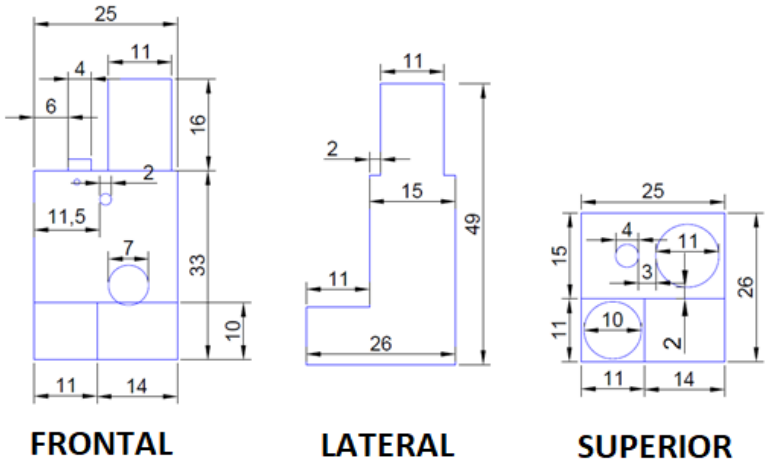

Figura 3.47. Esquema de maqueta

Realizados sus diseños tanto de la parte física (maqueta) y la parte lógica (código de programación), se procede a la implementación real del dispensador para su posterior puesta a prueba. La maqueta elaborada se muestra en la Figura 3.48.

Figura 3.48. Maqueta parte interna y final

3.5 Realización de pruebas de funcionamiento del prototipo

Terminada la etapa de implementación de la maqueta del dispensador, se llevó a cabo el proceso de pruebas del dispensador para comprobar su correcto funcionamiento que certifique o valide el producto establecido dentro del plan del proyecto.

Inicio de plataformas de comunicación

Como paso inicial, se verifica la instalación de las aplicaciones que permitirán la conexión con las plataformas de comunicación y el microcontrolador como lo son Blynk IoT (versión *Legacy)* y Telegram. Ya instaladas las apps, se procede a buscar el *bot* del dispensador como se muestra en la Figura 3.49; además se inicializa el *bot* para que aparezca dentro del índice de chats del usuario:

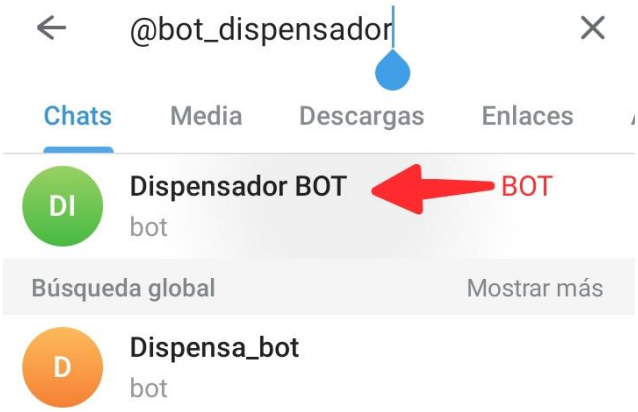

Figura 3.49. Búsqueda del *bot* Telegram

Por otro lado, dentro de la aplicación de Blynk, se procede a ingresar los parámetros de conexión al servidor propio, paso que se mostró en la Figura 3.3., seguido del ingreso de credenciales para acceso a la interfaz como se muestra en la Figura 3.50:

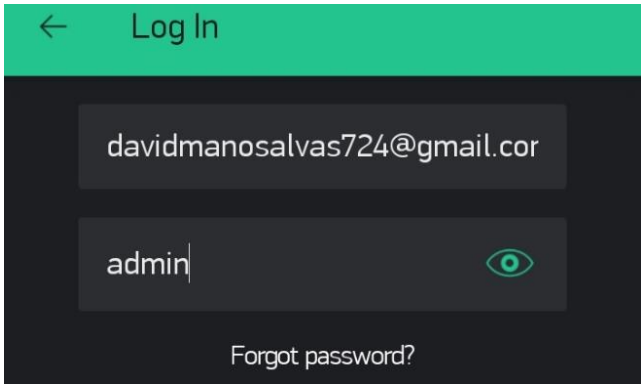

Figura 3.50. Credenciales de acceso Blynk

Conexión del dispensador

Una vez iniciadas las interfaces de comunicación, se procede a conectar el dispensador a la alimentación eléctrica con el uso de un transformador de 120 (V) a doble salida de 5 (V), con una salida conectada al cable de microcontrolador, mientras que la otra conectada al circuito del relé. Seguido de esto, se verifica la llegada del mensaje de inicio para la plataforma Telegram, así como la conexión a la plataforma Blynk.

Uso de plataforma Blynk

Así, se puede observar en la Figura 3.51 que el mensaje de inicio viene acompañado de dos mensajes de alerta de escasez tanto de agua como comida, estas mismas alertas mostradas mediante leds en la aplicación Blynk, como se muestra en la Figura 3.52, por lo que se procede a llenar los compartimentos y se verifica dentro de la aplicación Blynk el apagado de los leds de alerta.

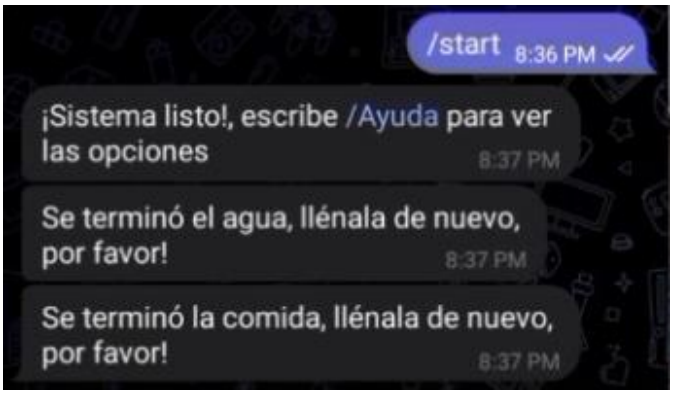

Figura 3.51. Mensaje de inicio Telegram

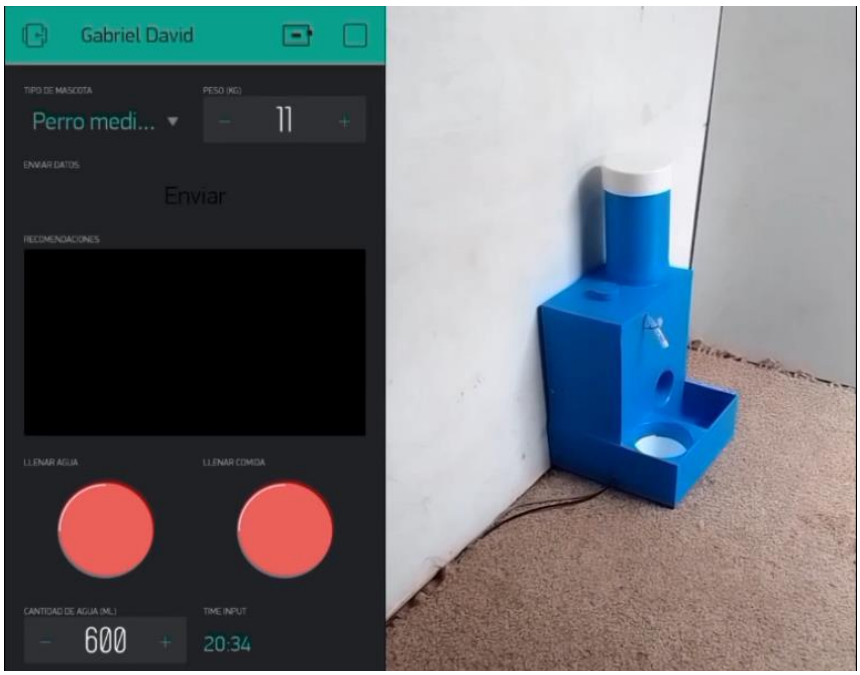

Figura 3.52. Alerta escasez de comida y agua

Como se observa en la Figura 3.53, los leds se muestran apagados al haber llenado los compartimentos de agua y comida.

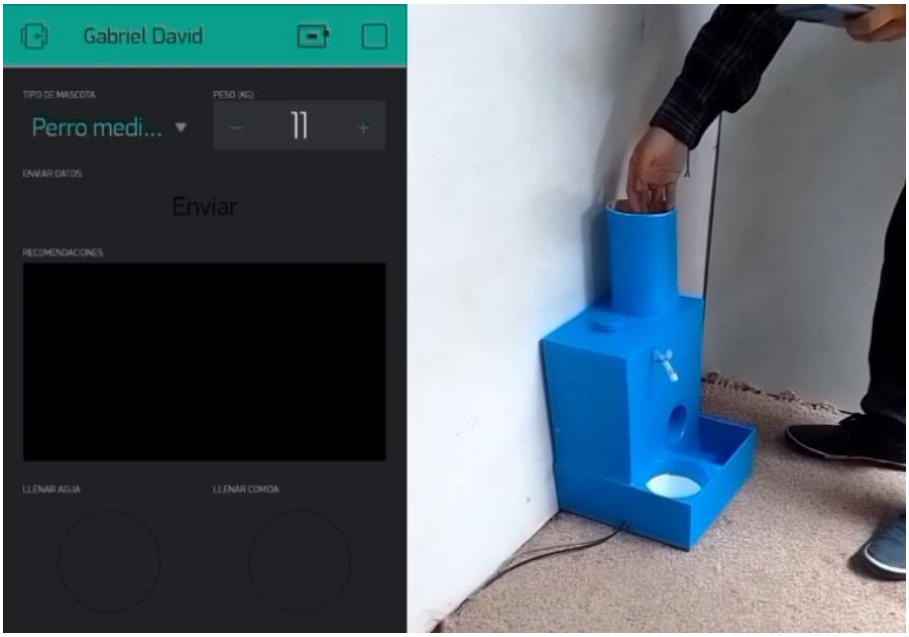

Figura 3.53. Llenado de compartimentos

Habiendo llenado los compartimentos de alimento del dispensador, se procede a establecer un horario para la dispensación de 250 (ml) de agua y 2 porciones de comida, y se permite el acercamiento de la mascota para la detección de su presencia de modo que permita dispensar el alimento. En la Figura 3.54 se puede observar la dispensación de agua dentro del bebedero integrado del dispensador.

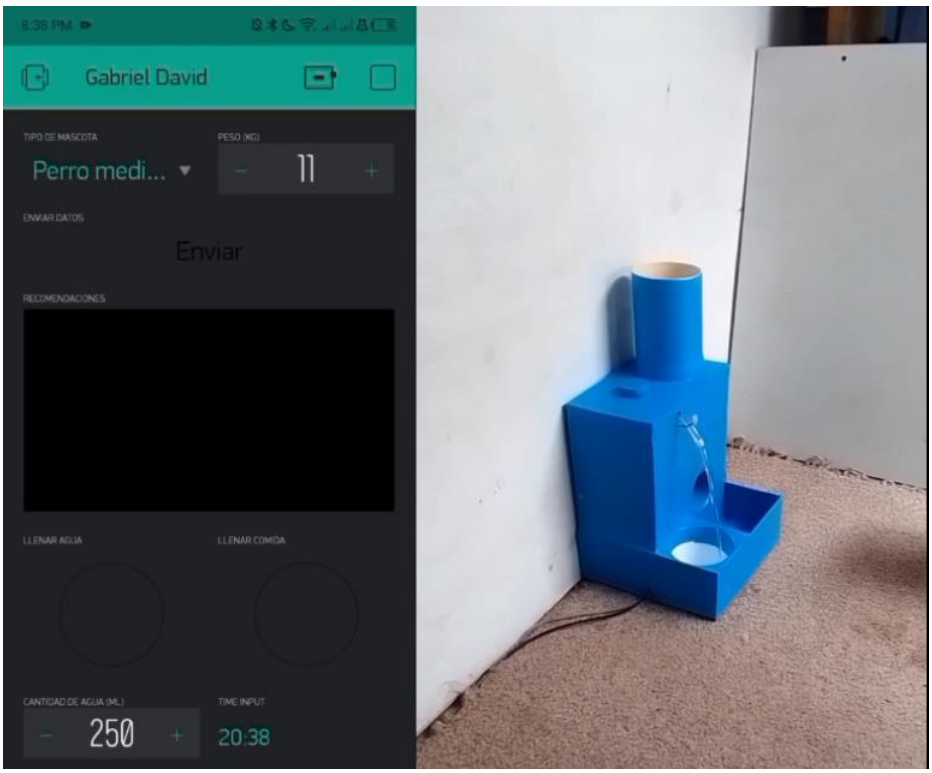

Figura 3.54. Dispensación en horario

Al haber llenado el dispensador con una cantidad mínima de comida, se puede observar el encendido de la alerta de escasez de comida mediante el led de la aplicación y la recepción de un mensaje en Telegram como se observa en la Figura 3.55.

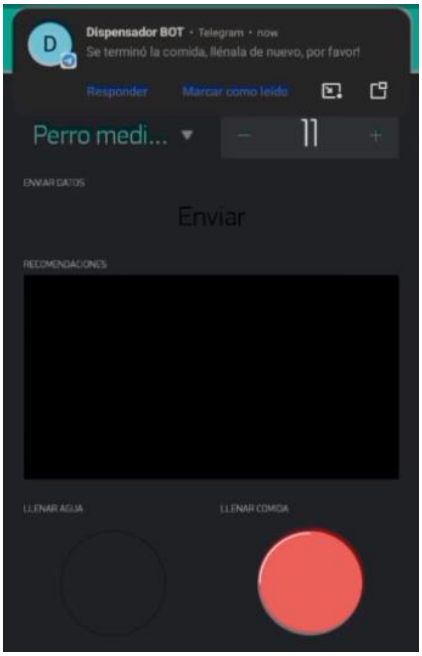

Figura 3.55. Alerta de escasez de comida

En adición, se procede a establecer la dispensación de una cantidad adicional de agua con 0 porciones de comida, para dispensar el agua sobrante en los compartimentos y permitir la observación del funcionamiento correcto de la alerta para escasez de agua mostrado en la Figura 3.56.

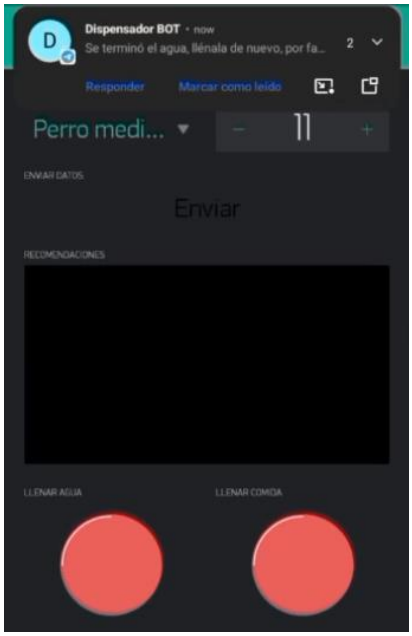

Figura 3.56. Alerta escasez de agua

Uso plataforma Telegram

Luego, se procede a realizar la prueba de dispensación con el uso de la aplicación Telegram, para lo cual se procede a llenar nuevamente los compartimentos. Así, se envía el comando “/Ayuda” para la recepción del índice de comandos útiles para las diferentes funciones del dispensador. Como se evidencia en la Figura 3.57.

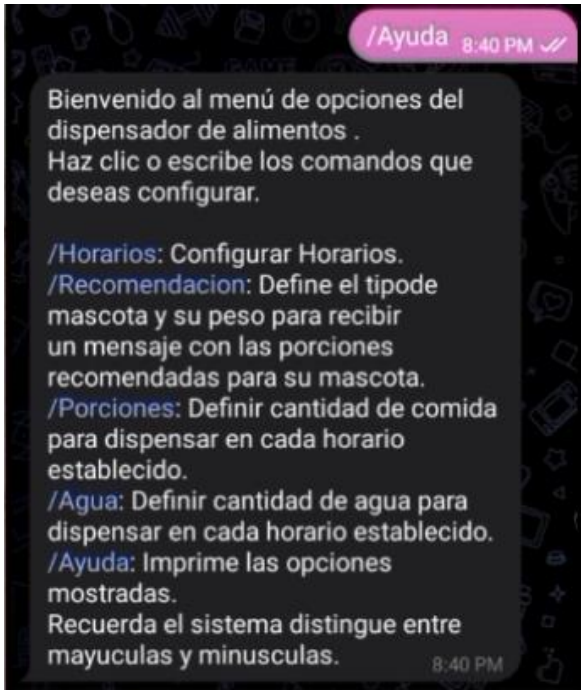

Figura 3.57. Solicitud de índice de comandos en Telegram

Obtenido el índice respectivo, se procede a establecer la cantidad de porciones a dispensar y cantidad de agua, como se observa en la Figura 3.58, para posteriormente establecer un horario para la observación del correcto funcionamiento.

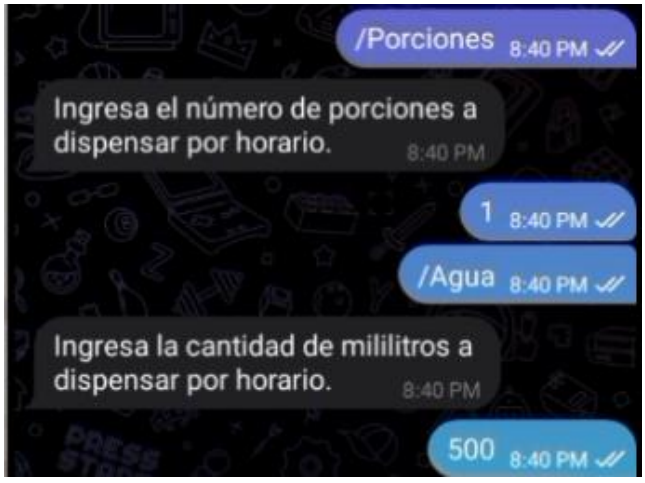

Figura 3.58. Definición de cantidades a dispensar

Además, para evidenciar la **no** dispensación de alimento al no detectar a la mascota, se evita el acercamiento de la misma durante el horario de dispensación establecido, siendo que se permite observar el llamado a la mascota pasando un minuto al no detectarla y dispensando la comida una vez percibida su presencia después del llamado.

Como se observa en la Figura 3.59, se permite el acercamiento de la mascota transcurridos 2 minutos del horario establecido, por lo cual se procede a dispensar el alimento en el horario de 20h44 siendo que se estableció a las 20h42 sin contar con la detección de la presencia de la mascota.

Figura 3.59. Dispensación después del tiempo establecido

Solicitud de recomendaciones de alimento

Una vez comprobado el funcionamiento de los mecanismos de detección y dispensación de alimento controlado mediante ambas plataformas, se procede a verificar el funcionamiento de la emisión de recomendaciones respecto a las cantidades sugeridas para dispensar de acuerdo con el peso de la mascota.

Como se observa en la Figura 3.60, se definen los parámetros necesarios y se pulsa el *bot*ón enviar para la recepción de la recomendación de cantidades.

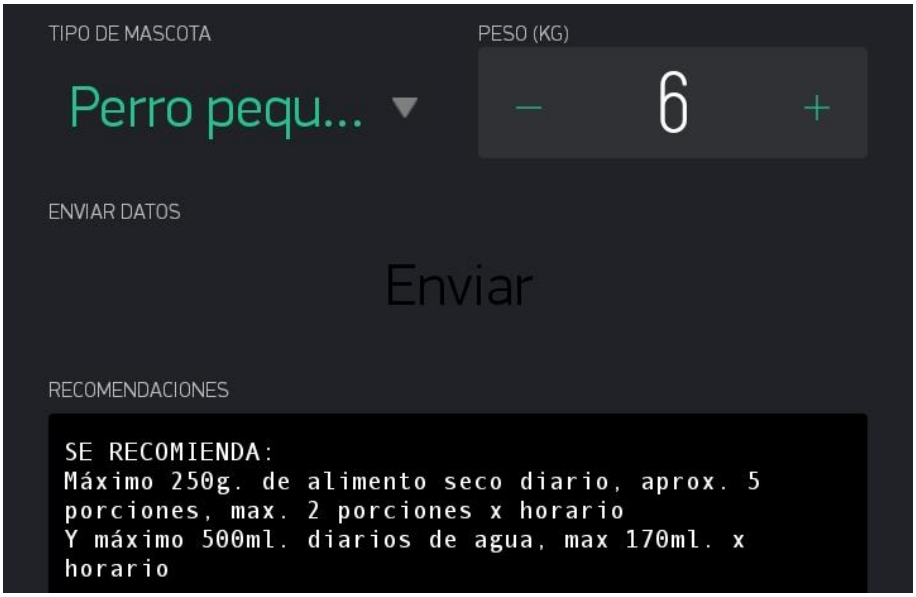

Figura 3.60. Recomendación de cantidades Blynk

Asimismo, se prueba dicha solicitud con el uso de la aplicación Telegram como se muestra en la Figura 3.61.

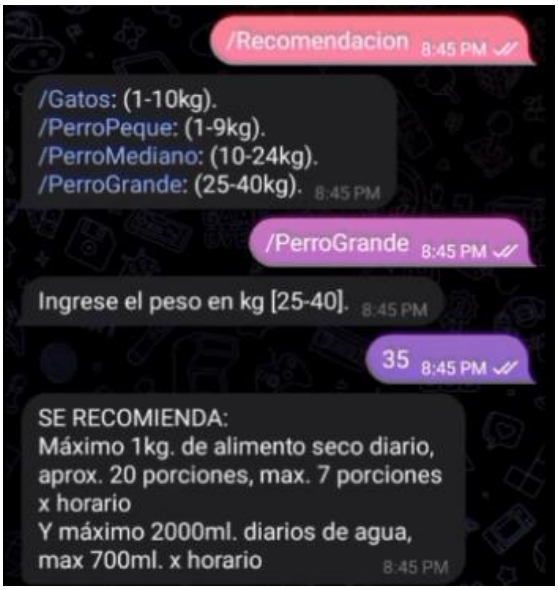

Figura 3.61. Recomendación de cantidades Telegram

En ultimas instancias, se realiza la Tabla 3.5 que resumen los resultados obtenidos mediante las pruebas expuestas anteriormente. En dicha tabla, se presenta un listado de actividades llevadas a cabo para comprobar el correcto funcionamiento del dispensador, enlistando acciones como detección de presencia de la mascota en frente del dispositivo, las alertas mostradas al dueño que indiquen la escasez de agua y comida dentro de los compartimentos elaborados para cada uno, alertas que deben mostrarse dentro de la aplicación Blynk así como en Telegram.

También se enlista la comprobación del funcionamiento de los mecanismos de dispensación que permitan obtener la cantidad definida en las plataformas de control y comunicación para un horario definido; asimismo, se tiene en cuenta el tiempo de respuesta del dispensador, evidenciado mediante la dispensación de las cantidades solicitadas exactamente cumplido el horario establecido.

Finalmente, se verifica el sonido emitido como llamado de la mascota cumplido el horario de dispensación configurado.

Tabla 3.5. Resumen de resultados

Prueba de funcionamiento	**Resultado Positivo**	**Resultado negativo**
Detección de mascota	X	
Mensaje de alerta de escasez de agua	X	
Mensaje de alerta de escasez de comida	X	
Led de alerta de agua	X	
Led de alerta de comida	X	
Dispensación de comida	X	
Dispensación de agua	X	
Dispensación en horario establecido	X	
Llamado de la mascota	X	
Sincronización entre plataformas y microcontrolador	X	

De esta manera, se permite constatar el funcionamiento requerido para el plan del proyecto. Ciertamente, las pruebas expuestas en el presente documento, permiten la verificación de los resultados obtenidos; sin embargo, para una mejor observación del funcionamiento del dispensador, se adjunta el link del video demostrativo en forma de URL y código QR encontrado en el apartado correspondiente al **Anexo II.**

Costos

La Tabla 3.6 contiene el detalle de los costos para cada material utilizado durante el desarrollo del proyecto. Los precios mostrados en la tabla representan un costo de referencia, siendo que los módulos y elementos presentados en la tabla pueden variar su costo según el lugar donde se los adquiera.

En adición, se toma en cuenta el tiempo de elaboración del proyecto como costo de mano de obra sumado al total de presupuesto del dispensador.

Tabla 3.6. Presupuesto del prototipo

Aspecto	Cantidad	Costo unitario	Costo Total
Microcontrolador Esp32-Devkit-V1	1	$ 17,00	$ 17,00
Sensor de nivel de agua tipo flotador	1	$ 5,80	$ 5,80
Sensor de presencia HC-SR501	1	$ 3,00	$ 3,00
Sensor de obstáculos FC-51	1	$ 1,65	$ 1,65
Servomotor Sg90	1	$ 3,40	$ 3,40
Relé de 2 canales 5V	1	$ 4,50	$ 4,50
Cable micro-USB a USB A	1	$ 3,00	$ 3,00
Baquelita	1	$ 0,60	$ 0,60
Bobina de estaño para soldar	1	$ 6,00	$ 6,00
Ácido Férrico	1	$ 0,35	$ 0,35
Cautín tipo Lápiz	1	$ 7,00	$ 7,00
Buzzer	1	$ 0,80	$ 0,80
Tubos y uniones	1	$ 18,00	$ 18,00
Envase de agua	1	$ 2,00	$ 2,00
Caja maqueta (tablas, tornillos, abrazaderas)	1	$ 5,00	$ 5,00
Cables jumper	5	$ 0,10	$ 0,50
Cargador doble USB	1	$ 8,00	$ 8,00
Cable UTP (m)	1	$ 0,60	$ 0,60
Mini bomba sumergible	1	$ 3,90	$ 3,90
Espadines 1x40	5	$ 0,45	$ 2,25
Mano de obra	24 (h)	$ 15,00	$ 360,00
		TOTAL	**$ 453,35**

4 CONCLUSIONES

- Para la realización del dispensador, se hizo uso del microcontrolador Esp32 integrado en una placa de desarrollo denominada “Doit Esp32 DevKit V1” cuyo módulo integrado de comunicación inalámbrica mediante la emisión de señales electromagnéticas de 2.4 [GHz], permitió su conexión a Internet garantizando la movilidad del dispensador dentro del área de cobertura de la red inalámbrica del usuario; siendo que al utilizar la frecuencia de emisión más baja utilizada en redes Wi-Fi (siendo 2.4 y 5 [GHz] las frecuencias utilizadas por dicha tecnología), permite la conexión del dispositivo desde lugares en un rango de cobertura más amplio. Este microcontrolador, realiza el procesamiento de los datos recibidos por los sensores que componen el proyecto para permitir la activación de actuadores que garantizan el paso de alimentos hacia los platos, los mismos que fueron integrados mediante la construcción de mecanismos adecuados para la dispensación.
- La estructura de la maqueta se encuentra compuesta por dos compartimentos, los cuales pueden almacenar 3 litros de agua y 1 kilogramo de alimento; además, la maqueta se encuentra sellada para evitar el acceso a los elementos electrónicos, circuitos y cables que lo conforman, previniendo posibles daños físicos que puedan comprometer el funcionamiento del dispensador. La maqueta se encuentra fabricada de madera al ser un material resistente a golpes o posibles caídas, donde a diferencia de materiales como el aluminio, este es lo suficientemente pesado para evitar que la mascota derrumbe el dispositivo, pero con un peso adecuado para no limitar su movilidad, permitiendo al usuario ubicarlo en cualquier lugar interno del hogar. Cabe recalcar que, a pesar de encontrarse sellado con una cubierta plástica, para evitar daños en la estructura de madera en caso de entrar en contacto con agua derramada al rellenar el compartimento para líquidos, no se define como un dispositivo de uso en exteriores.
- La electrónica del dispositivo se encuentra conformada por sensores, actuadores, controlador y comunicación, donde los sensores permiten la detección de la mascota en el entorno cercano, y la detección de escasez de alimento en los compartimentos; los actuadores permiten mediante los mecanismos elaborados, la dispensación del alimento. El controlador permite la administración de los datos enviados desde los sensores o desde las plataformas de comunicación para permitir el envío de decisiones que permitan

accionar los mecanismos de acuerdo a sentencias lógicas definidas dentro de su programación; finalmente, se utilizan las ya mencionadas plataformas de comunicación como herramienta que permitan el control remoto del dispensador por parte del usuario mediante una interfaz amigable e intuitiva.

- Para brindar al usuario facilidad de configuración y control del dispensador de forma remota desde Internet, se elaboró la interfaz de usuario, presentada tanto por la aplicación Blynk como en Telegram.

  Estas interfaces presentan un entorno amigable y sencillo para la configuración de parámetros de dispensación; además, que al ser plataformas con gran desarrollo, presentan una gran adaptabilidad para su uso en cualquier *smartphone* u otros dispositivos de usuario como *tablets* y ordenadores.
- El adiestramiento de la mascota frente a los sonidos y olores relacionados con su alimentación presentan un factor importante en la adaptación de la mascota frente a dispositivos mecánicos. Por ello, el dispensador incorpora un *buzzer* o zumbador que emite un sonido para llamar a la mascota permitiendo que esta lo relacione con la dispensación de comida.
- El prototipo implementado ayuda a personas que poseen mascotas en casa, ya que mediante el establecimiento de horarios, permite al usuario evitar preocupaciones sobre la alimentación de sus mascotas en caso de encontrarse ausente; sin embargo, se debe tener en cuenta la cantidad de alimento permitida para su almacenamiento, ya que esto representa un factor importante respecto al tiempo de autonomía presentado por el dispositivo, entendiéndose por autonomía el tiempo que el dispositivo puede funcionar sin la necesidad que se recargue alimento o agua.
- Para permitir el control remoto del dispensador desde Internet, se implementó servidor propio dentro de la plataforma Google Cloud que aloje las características necesarias para el desarrollo del mando de control que permita la interacción y sincronía con la aplicación Blynk. El uso de dicha plataforma, permite la obtención de una dirección pública que garantice la conexión al servidor desde cualquier lugar; sin embargo, el uso de la infraestructura facilitada por la plataforma, resulta en el pago mensual de acuerdo con el tiempo de uso del servidor. Para esto, se recomienda la adaptación del proyecto con otras plataformas IoT gratuitas, limitarlo a únicamente su control desde Telegram o adaptar el mando de control con los *widgets* disponibles en la versión gratuita de Blynk.

- Para evitar daños por sobre corrientes en el microcontrolador se emplean fuentes de alimentación separadas, uno para el microcontrolador y otro para el circuito del relé, para lo cual se hace uso de un transformador con 2 salidas independientes de 5 (V).
- Con la finalidad de evitar uso de contenedores de distintos tamaños y evitar el derrame de la comida y agua o una mala dispensación de estas se integraron platos y bebederos en el prototipo implementado.
- A través de la comparación entre dispensadores comerciales, se obtuvo una idea general para la elaboración del prototipo, resultando útil dicho método para la definición de fortalezas y debilidades de cada producto para la realización de un prototipo más completo frente a sus alternativas en el mercado.
- La realización de pruebas y toma de resultados verificó el funcionamiento óptimo del prototipo, presentando tiempos de respuesta cortos y dispensación de la cantidad definida de alimento.
- El uso de Proteus como *software* de diseño facilito la elaboración del circuito electrónico, así como la facilidad de creación del PCB mediante la herramienta Ares ya que permite realizar el auto enrutamiento de pistas.

  Además, la utilización del sitio web SnapEDA, permite obtener una amplia gama de módulos desarrollados por la comunidad para su simulación dentro del programa.
- Con el fin de garantizar el control del dispositivo únicamente por parte del dueño de la mascota, y teniendo en cuenta que la comunicación mediante la plataforma Telegram se da con el uso de un *bot* público, se realizó la limitación de la dicha comunicación estableciendo la emisión de respuestas por parte del microcontrolador solamente hacia el id del usuario definido en el código, evitando el control de usuarios no autorizados al no permitir una conversación bidireccional entre estos y el controlador.
- Para la dispensación de una cantidad controlada de comida, se realizó la implementación del respectivo mecanismo conformado por el servomotor y la placa de madera que actúa como compuerta para el paso de la comida; tras la elaboración de dicho mecanismo, se realizaron pruebas que permitieron determinar la cantidad promedio de alimento seco dispensado por cada apertura y cierre de la compuerta, dado un determinado ángulo de giro y tiempo de apertura establecido, en función de lo cual se tomó este valor como el peso de comida dispensado para porción definida.

- Como punto extra dentro de la implementación del proyecto, se elaboró un apartado de sugerencias de cantidad de alimento diario recomendado para la mascota; sin embargo, estas recomendaciones solo representan una sugerencia y el usuario debe tener en cuenta factores adicionales para establecer las cantidades que permitan una dieta saludable según el tipo de mascota.

5 RECOMENDACIONES

- Se recomienda como trabajo a futuro: añadir un sistema de respaldo de energía y diseñar compartimentos con mayor capacidad para incrementar el tiempo de autonomía del dispositivo, disminuyendo la preocupación del usuario frente a escasez de alimento o fallo en la energía del hogar.
- Se debe tener en cuenta que al momento de activación del servomotor para la dispensación de comida, se genera un ruido que puede presentar desconfianza para la mascota, por lo cual se recomienda el uso de un motor menos ruidoso para evitar asustar o estresar al animal.
- Incluir protección para los cables de alimentación para evitar daños por parte de la mascota o accidentes en general.
- Como mejora al proyecto se recomienda la incorporación de una cámara y altavoz para permitir al usuario interactuar con su mascota desde cualquier lugar con conexión a Internet; además, respecto a la comunicación mediante el uso de la plataforma Telegram, se recomienda la agregación de un mecanismo de registro dinámico de id para el control desde diferentes usuarios, teniendo en cuenta la definición de un usuario principal que administre los permisos de control de estos usuarios adicionales.
- Otra mejora que se puede realizar en trabajos futuros es establecer mecanismos para el ingreso dinámico de credenciales que permitan establecer la conexión a la red y al Internet; además, se recomienda establecer dichas credenciales de red dentro de la memoria no volátil del microcontrolador para evitar la solicitud de credenciales cada que se conecta el dispositivo a la alimentación o en caso de presentarse un corte de energía, de forma que no se eliminen estos datos; asimismo, tener en cuenta el número de ciclos de escritura permitidos dentro de esta memoria.
- A pesar de presentar una gran ayuda para el usuario frente a la alimentación programada de sus mascotas, se recomienda compartir tiempo con las mismas para evitar estrés o enfermedades provocadas por la falta de ejercicio.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] K. Rose, E. Scott y L. Chapin, The Internet of Things: An Overview, Geneva: Internet Society, 2015.

[2] Blynk, «About Blynk,» Blynk Inc, [En línea]. Available: https://blynk.io/about. [Último acceso: 7 Junio 2023].

[3] S. Karl, «Getting Started With the Arduino IoT Cloud,» Arduino, 16 Mayo 2023. [En línea]. Available: https://docs.arduino.cc/arduino-cloud/getting-started/iot-cloud-getting-started. [Último acceso: 7 Junio 2023].

[4] R. Keim, «All About Circuits,» EETech Media, LLC., 25 Marzo 2019. [En línea]. Available: https://www.allaboutcircuits.com/technical-articles/what-is-a-microcontroller-introduction-component-characteristics-component/. [Último acceso: 7 Junio 2023].

[5] Espressif Systems, «ESP32 Wi-Fi & Bluetooth MCU,» Espressif Systems, 2015. [En línea]. Available: https://www.espressif.com/en/products/socs/esp32. [Último acceso: 7 Junio 2023].

[6] Arduino, «About Arduino,» Arduino, 15 Septiembre 2021. [En línea]. Available: https://www.arduino.cc/en/about. [Último acceso: 7 Mayo 2023].

[7] L. G. Corona, G. S. Jimenez y J. Carreño, Sensores y actuadores, Azcapotzalco: Grupo Editorial Patrial S.A., 2014.

[8] Unit Electronics, «Módulo Sensor De Obstáculos Reflectivo Infrarrojo FC-51,» UNIT Electronics, [En línea]. Available: https://uelectronics.com/producto/fc-51-sensor-de-obstaculos-reflectivo-infrarojo/. [Último acceso: 07 Junio 2023].

[9] Electronica-basica.com, «¿Qué es un Sensor PIR?,» Electronica-basica, [En línea]. Available: https://electronica-basica.com/sensor-pir/. [Último acceso: 07 Junio 2023].

[10] Unit Electronics, «Sensor De Nivel Flotador Vertical,» UNIT Electronics, [En línea]. Available: https://uelectronics.com/producto/sensor-de-nivel-flotador-vertical/. [Último acceso: 07 Junio 2023].

[11] Telegram, «Telegram FAQ,» Telegram, [En línea]. Available: https://telegram.org/faq#q-what-is-telegram-what-do-i-do-here. [Último acceso: 7 Junio 2023].

[12] Espressif Systems, «Esp32 Datasheet,» 2023. [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. [Último acceso: 07 Junio 2023].

[13] Espressif Systems, «Esp8266 Datasheet,» 2023. [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. [Último acceso: 07 Junio 2023].

[14] Components101, «HC-SR501 PIR Sensor Working, Pinout & Datasheet,» 17 Julio 2021. [En línea]. Available: https://components101.com/sensors/hc-sr501-pir-sensor. [Último acceso: 07 Junio 2023].

[15] ETC2, «All Datasheet,» [En línea]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/1131987/ETC2/HC-SR501.html. [Último acceso: 07 Junio 2023].

[16] MANTECH, «Miniature PIR sensor,» [En línea]. Available: https://www.adrirobot.it/wp-content/uploads/2022/04/AM312-Miniature-_sensor.pdf. [Último acceso: 07 Junio 2023].

[17] Naylamp Mechatronics, «Valvula Solenoide 1/2" 12VDC,» Naylamp Mechatronic SAC, [En línea]. Available: https://naylampmechatronics.com/valvulas/314-valvula-solenoide-1p2-pulg-12vdc-nc.html. [Último acceso: 07 Junio 2023].

[18] Google, «Gogle Cloud infraestructure,» Google, [En línea]. Available: https://cloud.google.com/infrastructure/. [Último acceso: 07 Junio 2023].

[19] Blynk IoT Platform, «Pricing,» [En línea]. Available: https://blynk.io/pricing. [Último acceso: 07 Junio 2023].

[20] Arduino , «Arduino Cloud - Plans,» [En línea]. Available: https://cloud.arduino.cc/plans. [Último acceso: 07 Junio 2023].

[21] Telegram, «Bots: An introduction for developers,» [En línea]. Available: https://core.telegram.org/bots. [Último acceso: 07 Junio 2023].

[22] Pet Reader, «How Many Grams of Wet Food Does a Dog Need Per Day?,» Pet Reader, [En línea]. Available: https://petreader.net/how-many-grams-of-wet-food-does-a-dog-need-per-day/. [Último acceso: 07 Junio 2023].

[23] FERPLAST S.P.A., «HOW MUCH MUST THE CAT EAT?,» Love Ferplast, [En línea]. Available: https://blog.ferplast.com/en/how-much-must-the-cat-eat/. [Último acceso: 07 Junio 2023].

[24] Purina, «Cómo hidratar a un gato,» Nestlé, [En línea]. Available: https://www.purina.es/cuidados/gatos/comportamiento/viajar/como-hidratar-a-un-gato. [Último acceso: 07 Junio 2023].

[25] PURINA, «Keeping Your Dog Hydrated,» Nestlé, [En línea]. Available: https://www.purina.co.uk/articles/dogs/health/symptoms/dog-hydration. [Último acceso: 07 Junio 2023].

[26] S. Karl, «Getting Started With the Arduino IoT Cloud,» Arduino, 16 Mayo 2023. [En línea]. Available: https://docs.arduino.cc/arduino-cloud/getting-started. [Último acceso: 7 Junio 2023].

[27] Unit Electronics, «Sensor De Nivel Flotador Vertical,» [En línea]. Available: https://uelectronics.com/producto/sensor-de-nivel-flotador-vertical/. [Último acceso: 07 Junio 2023].

7 ANEXOS

La lista de los **Anexos** se muestra a continuación:

ANEXO I. Certificado de originalidad

ANEXO II. Enlaces

ANEXO III. Código fuente

ANEXO I: Certificado de Originalidad

Quito, D.M. 22 de agosto de 2023

De mi consideración:

Yo, **LEANDRO ANTONIO PAZMIÑO ORTIZ**, en calidad de Director del Trabajo de Integración Curricular titulado **IMPLEMENTACIÓN DE UN PROTOTIPO DE DISPENSADOR DE COMIDA PARA MASCOTAS CONTROLADO A TRAVÉS DE INTERNET**, componente **IMPLEMENTACIÓN DE UN PROTOTIPO DE DISPENSADOR DE COMIDA PARA MASCOTAS CONTROLADO A TRAVÉS DE INTERNET POR MEDIO DE UNA APLICACIÓN EN ANDROID Y DE TELEGRAM** elaborado por el estudiante **GABRIEL DAVID MANOSALVAS SEIBA** de la carrera en **TECNÓLOGÍA SUPERIOR EN REDES Y TELECOMUNICACIONES**, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 12 %.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el link del informe generado por la herramienta Turnitin.

5. TIC_GManosalvas-Reporte_turnitin completo.pdf

Atentamente,

Leandro Pazmiño Ortiz

Docente

Escuela de Formación de Tecnólogos

ANEXO II: Enlaces

A continuación, se coloca el enlace al video demostrativo del funcionamiento del dispensador, así como el código QR para dicho video: GManosalvas

Anexo II.I Código QR: video de demostración del funcionamiento

ANEXO III: Código Fuente

```
#define BOT_TOKEN "6115207395:AAHEIi1R-PNbeyuWE-
3gEQNm4LZLFkPZT_g"//Declaración Token para conexión con Blynk
#define id_usuario "******"//Declaración identificador de Chat.
#define BLYNK_PRINT Serial //Declaración variable para comunicación Blynk
#define PinComida 5
#define PinAgua 15
#define PinPir 21
#define PinZumbador 4
#define PinServo 19
#define PinRelay 18
#include <UniversalTelegramBot.h>
#include <Ticker.h>
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <ESP32Servo.h>
#include <WiFiClientSecure.h>
#include <ESP32Time.h>
#include "time.h"
#include "sntp.h"
Servo miservo;
Ticker tic;
ESP32Time rtc(3600);
WiFiClientSecure cliente_telegram;
UniversalTelegramBot bot(BOT_TOKEN, cliente_telegram);
WidgetLED led1(V9);
WidgetLED led2(V8);
WidgetTerminal terminal(V10);
static unsigned long UltimaSincronizacion = 0;
const unsigned long intervalo_escaneo = 1000;
int agua = 0, IngresoPeso = 0, pet = 0, mascota = 0, peso, inicio = 1,
weight, TiempoHoras, TiempoMinutos;
uint8_t porciones = 0, horas[3], minutos[3], parteEntera, parteDecimal,
HoraActual, MinutoActual, SegundoActual, ValidarHorario = 0;
bool realizado = 1, ValidarPorciones = 0, ValidarAgua = 0, ValidarComida
= 0, alertaAgua = 0, alertaComida = 0, Presencia = false;
unsigned long ultimo;
float datonumero = 0;
const char token_autentic[] = "Y3UokQXGG9X_UP5mWtT5JQ6HO-ysB3ak";
char nombre_red[] = "RedGM", contrasenia[] = "M1753468964@-g";
const char* ntpServer1 = "sudamerica.pool.ntp.org";
const char* ntpServer2 = "time.nist.gov";
const int16_t gmtOffset_sec = -21600;
const uint16_t daylightOffset_sec = 3600;
//Función para obtener tiempo de NTP
void ObtenerTiempo() {
  struct tm InformacionTiempo;
  HoraActual = InformacionTiempo.tm_hour;
```

```
  MinutoActual = InformacionTiempo.tm_min;
  SegundoActual = InformacionTiempo.tm_sec;}
//Función para detección de presencia en interrupción
void DetectPresencia() {
  if (realizado == 0) {
    Presencia = 1;}}
//Función para borrar presencia periódicamente
void BorrarPresencia() {
  Presencia = 0;}
void setup() {
  pinMode(PinComida, INPUT);          //COMIDA
  pinMode(PinAgua, INPUT_PULLDOWN);   //AGUA
  pinMode(PinPir, INPUT);             //PIR
  pinMode(PinZumbador, OUTPUT);       //BUZZER
  pinMode(PinServo, OUTPUT);          //SERVO
  pinMode(PinRelay, OUTPUT);          //RELAY
  digitalWrite(PinRelay, HIGH);
  //Declaración interrupciones
  attachInterrupt(digitalPinToInterrupt(PinPir), DetectPresencia,
RISING);
  //Declaración función periódica
  tic.attach(300, BorrarPresencia);
  //Comunicación inalámbrica
  WiFi.begin(nombre_red, contrasenia);
  while (WiFi.status() != WL_CONNECTED) {
    //Indicar el proceso de conexión
    digitalWrite(PinZumbador, HIGH);
    delay(100);
    digitalWrite(PinZumbador, LOW);
    delay(100);}
  //Escritura de posición inicial de relé y servo
  digitalWrite(PinZumbador, LOW);
  miservo.attach(PinServo /*, 500, 3000*/);
  miservo.write(175);
  //Comunicación con Blynk
  Blynk.begin(token_autentic, nombre_red, contrasenia, IPAddress(10, 142,
0, 2), 8080);
  //Parámetro de NTP
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer1, ntpServer2);
  ObtenerTiempo();
  //Guardado de tiempo NTP en RTC
  rtc.setTime(SegundoActual + 3, MinutoActual, HoraActual, 5, 6, 2023);
  //Certificado de SSL de Telegram para comunicación
  cliente_telegram.setCACert(
    "-----BEGIN CERTIFICATE-----\n"
    "MIIDxTCCAq2gAwIBAgIBADANBgkqhkiG9w0BAQsFADCBgzELMAkGA1UEBhMCVVMx\n"
    "EDAOBgNVBAgTB0FyaXpvbmExEzARBgNVBAcTClNjb3R0c2RhbGUxGjAYBgNVBAoT\n"
    "EUdvRGFkZHkuY29tLCBJbmMuMTEwLwYDVQQDEyhHbyBEYWRkeSBSb290IENlcnRp\n"
    "ZmljYXRlIEF1dGhvcml0eSAtIEcyMB4XDTA5MDkwMTAwMDAwMFoXDTM3MTIzMTIz\n"
    "NTk1OVowgYMxCzAJBgNVBAYTAlVTMRAwDgYDVQQIEwdBcml6b25hMRMwEQYDVQQH\n"
```

```
    "EwpTY290dHNkYWxlMRowGAYDVQQKExFHb0RhZGR5LmNvbSwgSW5jLjExMC8GA1UE\n"
    "AxMoR28gRGFkZHkgUm9vdCBDZXJ0aWZpY2F0ZSBBdXRob3JpdHkgLSBHMjCCASIw\n"
    "DQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAL9xYgjx+lk09xvJGKP3gElY6SKD\n"
    "E6bFIEMBO4Tx5oVJnyfq9oQbTqC023CYxzIBsQU+B07u9PpPL1kwIuerGVZr4oAH\n"
    "/PMWdYA5UXvl+TW2dE6pjYIT5LY/qQOD+qK+ihVqf94Lw7YZFAXK6sOoBJQ7Rnwy\n"
    "DfMAZiLIjWltNowRGLfTshxgtDj6AozO091GB94KPutdfMh8+7ArU6SSYmlRJQVh\n"
    "GkSBjCypQ5Yj36w6gZoOKcUcqeldHraenjAKOc7xiID7S13MMuyFYkMlNAJWJwGR\n"
    "tDtwKj9useiciAF9n9T521NtYJ2/LOdYq7hfRvzOxBsDPAnrSTFcaUaz4EcCAwEA\n"
    "AaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMCAQYwHQYDVR0OBBYE\n"
    "FDqahQcQZyi27/a9BUFuIMGU2g/eMA0GCSqGSIb3DQEBCwUAA4IBAQCZ21151fmX\n"
    "WWcDYfF+OwYxdS2hII5PZYe096acvNjpL9DbWu7PdIxztDhC2gV7+AJ1uP2lsdeu\n"
    "9tfeE8tTEH6KRtGX+rcuKxGrkLAngPnon1rpN5+r5N9ss4UXnT3ZJE95kTXWXwTr\n"
    "gIOrmgIttRD02JDHBHNA7XIloKmf7J6raBKZV8aPEjoJpL1E/QYVN8Gb5DKj7Tjo\n"
    "2GTzLH4U/ALqn83/B2gX2yKQOC16jdFU8WnjXzPKej17CuPKf1855eJ1usV2GDPO\n"
    "LPAvTK33sefOT6jEm0pUBsV/fdUID+Ic/n4XuKxe9tQWskMJDE32p2u0mYRlynqI\n"
    "4uJEvlz36hz1\n"
    "-----END CERTIFICATE-----");
  //Envío de mensaje inicial en bot
  if (inicio == 1) {
    bot.sendMessage(id_usuario, "¡Sistema listo!, escribe /Ayuda para ver
las opciones");
    inicio = 0;}
  digitalWrite(PinZumbador, HIGH);
  delay(1000);
  digitalWrite(PinZumbador, LOW);}
void loop() {
  Blynk.run();
  alertas();
  //Sincronización reloj RTC
  if (millis() - UltimaSincronizacion >= 1000) {
    TiempoHoras = rtc.getHour(true) - 2;
    TiempoMinutos = rtc.getMinute();
    struct tm InformacionTiempo = rtc.getTimeStruct();
    UltimaSincronizacion = millis();}
  //Guarda mensajes en Telegram pasado 1 sec
  if (millis() - ultimo > intervalo_escaneo) {
    int conteo = bot.getUpdates(bot.last_message_received + 1);
    while (conteo) {
      mensaje(conteo);
      conteo = bot.getUpdates(bot.last_message_received + 1); }
    ultimo = millis();}
  //Comparación de horarios con hora RTC
  for (int i = 0; i < 3; i++) {
    if (horas[i] == TiempoHoras && minutos[i] == TiempoMinutos) {
      realizado = 0;
      ActivarZumbador();
      if (Presencia == HIGH) {
        realizado = 1;
        Presencia = 0;
        int retardo = map(agua, 0, 2000, 0, 60000);
```

```
        ActivarBomba(retardo);
        Presencia = 0;
        int z = 0;
        while (z != porciones) {
          ActivarServo();
          z++;}
        Presencia = 0;
        if (minutos[i] >= 0 && minutos[i] < 60) {
          minutos[i] = minutos[i] - 1;
        } else {
          horas[i] = horas[i] - 1;
          minutos[i] = 59;}
      } else {
        if (minutos[i] < 59) {
          minutos[i] = minutos[i] + 1;
        } else {
          minutos[i] = 0;
          horas[i] = horas[i] + 1;}}}}}
//Función de alertas de escasez
void alertas() {
  int alertW = digitalRead(PinAgua);
  int alertF = digitalRead(PinComida);
  if (alertW == LOW) {
    led1.on();
    if (alertaAgua == 0) {
      bot.sendMessage(id_usuario, "Se terminó el agua, llénala de nuevo,
por favor!");
      alertaAgua = 1;}
  } else {
    led1.off();
    alertaAgua = 0;}
  if (alertF == HIGH) {
    led2.on();
    if (alertaComida == 0) {
      bot.sendMessage(id_usuario, "Se terminó la comida, llénala de
nuevo, por favor!");
      alertaComida = 1;}
  } else {
    led2.off();
    alertaComida = 0;}}
//Funciones para cada pin virtual en Blynk
BLYNK_WRITE(V14) {
  weight = param.asInt();}
BLYNK_WRITE(V7) {
  porciones = param.asInt();}
BLYNK_WRITE(V13) {
  agua = param.asInt();}
BLYNK_WRITE(V11) {
  int enviar = param.asInt();
  if (enviar == 1) {
```

```
    terminal.clear();
    if (weight >= 1 && weight < 5 && pet == 1) {
      terminal.println("SE RECOMIENDA:");
      terminal.println("Máximo 200g. de alimento seco diario, aprox. 4
porciones, max. 2 porciones x horario");
      terminal.println("Y máximo 500ml. diarios de agua, max 170ml. x
horario");
      terminal.flush();}
    if (weight >= 5 && weight <= 10 && pet == 1) {
      terminal.println("SE RECOMIENDA:");
      terminal.println("Máximo 400g. de alimento seco diario, aprox. 8
porciones, max. 3 porciones x horario");
      terminal.println("Y máximo 1000ml. diarios de agua, max 340ml. x
horario");
      terminal.flush();}
    if (weight >= 1 && weight < 5 && pet == 2) {
      terminal.println("SE RECOMIENDA:");
      terminal.println("Máximo 125g. de alimento seco diario, aprox. 3
porciones, max. 1 porcion x horario");
      terminal.println("Y máximo 250ml. diarios de agua, max 80ml. x
horario");
      terminal.flush();}
    if (weight >= 5 && weight < 10 && pet == 2) {
      terminal.println("SE RECOMIENDA:");
      terminal.println("Máximo 250g. de alimento seco diario, aprox. 5
porciones, max. 2 porciones x horario");
      terminal.println("Y máximo 500ml. diarios de agua, max 170ml. x
horario");
      terminal.flush();}
    if (weight >= 10 && weight < 15 && pet == 3) {
      terminal.println("SE RECOMIENDA:");
      terminal.println("Máximo 375g. de alimento seco diario, aprox. 8
porciones, max. 3 porciones x horario");
      terminal.println("Y máximo 750ml. diarios de agua, max 250ml. x
horario");
      terminal.flush();}
    if (weight >= 15 && weight < 25 && pet == 3) {
      terminal.println("SE RECOMIENDA:");
      terminal.println("Máximo 625g. de alimento seco diario, aprox. 13
porciones, max. 4 porciones x horario");
      terminal.println("Y máximo 1250ml. diarios de agua, max 400ml. x
horario");
      terminal.flush();}
    if (weight >= 25 && weight < 30 && pet == 4) {
      terminal.println("SE RECOMIENDA:");
      terminal.println("Máximo 750g. de alimento seco diario, aprox. 15
porciones, max. 5 porciones x horario");
      terminal.println("Y máximo 1500ml. diarios de agua, max 500ml. x
horario");
      terminal.flush();}
```

```
    if (weight >= 30 && weight < 40 && pet == 4) {
      terminal.println("SE RECOMIENDA:");
      terminal.println("Máximo 1kg. de alimento seco diario, aprox. 20
porciones, max. 7 porciones x horario");
      terminal.println("Y máximo 2000ml. diarios de agua, max 700ml. x
horario");
      terminal.flush();}}}
BLYNK_WRITE(V12) {
  pet = param.asInt();}
BLYNK_WRITE(V4) {
  TimeInputParam t(param);
  horas[0] = t.getStartHour();
  minutos[0] = t.getStartMinute();}
BLYNK_WRITE(V5) {
  TimeInputParam t(param);
  horas[1] = t.getStartHour();
  minutos[1] = t.getStartMinute();}
BLYNK_WRITE(V6) {
  TimeInputParam t(param);
  horas[2] = t.getStartHour();
  minutos[2] = t.getStartMinute();}
//Comparación de cada comando para envío de respuestas y recepción
void mensaje(int conteo) {
  for (int i = 0; i < conteo; i++) {
    String mensaje_texto = bot.messages[i].text;
    datonumero = mensaje_texto.toFloat();
    //Imprimir opciones
    if (mensaje_texto == "/Ayuda") {
      String ayuda = " Bienvenido al menú de opciones del dispensador de
alimentos.\n";
      ayuda += "Haz clic o escribe los comandos que deseas
configurar.\n\n";
      ayuda += "/Horarios: Configurar Horarios. \n";
      ayuda += "/Recomendacion: Define el tipo de mascota y su peso para
recibir un mensaje con las porciones recomendadas para su mascota. \n";
      ayuda += "/Porciones: Definir cantidad de comida para dispensar en
cada horario establecido.\n";
      ayuda += "/Agua: Definir cantidad de agua para dispensar en cada
horario establecido.\n";
      ayuda += "/Ayuda: Imprime las opciones mostradas. \n";
      ayuda += "Recuerda que el sistema distingue entre mayúsculas y
minúsculas. \n";
      bot.sendMessage(id_usuario, ayuda, "");}
    if (mensaje_texto == "/Horarios") {
      bot.sendMessage(id_usuario, "Escoge el horario que deseas
configurar.\n /Horario1 \n /Horario2 \n /Horario3 \n");}
    if (mensaje_texto == "/Recomendacion") {
      bot.sendMessage(id_usuario, "/Gatos: (1-10kg).\n /PerroPeque: (1-
9kg).\n /PerroMediano: (10-24kg).\n /PerroGrande: (25-40kg).\n");}
    if (mensaje_texto == "/Gatos") {
```

```
      mascota = 1;
      IngresoPeso = 1;
      bot.sendMessage(id_usuario, "Ingrese el peso en kg [1-10].\n");}
    if (mensaje_texto == "/PerroPeque") {
      mascota = 2;
      IngresoPeso = 1;
      bot.sendMessage(id_usuario, "Ingrese el peso en kg [1-9].\n");}
    if (mensaje_texto == "/PerroMediano") {
      mascota = 3;
      IngresoPeso = 1;
      bot.sendMessage(id_usuario, "Ingrese el peso en kg [10-24].\n");}
    if (mensaje_texto == "/PerroGrande") {
      mascota = 4;
      IngresoPeso = 1;
      bot.sendMessage(id_usuario, "Ingrese el peso en kg [25-40].\n");}
    if (mensaje_texto == "/Agua") {
      bot.sendMessage(id_usuario, "Ingresa la cantidad de mililitros a
dispensar por horario.");
      ValidarAgua = 1;}
    if (mensaje_texto == "/Horario1") {
      bot.sendMessage(id_usuario, "Ingresa el horario con el siguiente
formato: 'HH.MM' \n");
      ValidarHorario = 1;}
    if (mensaje_texto == "/Horario2") {
      bot.sendMessage(id_usuario, "Ingresa el horario con el siguiente
formato: 'HH.MM' \n");
      ValidarHorario = 2;}
    if (mensaje_texto == "/Horario3") {
      bot.sendMessage(id_usuario, "Ingresa el horario con el siguiente
formato: 'HH.MM' \n");
      ValidarHorario = 3;}
    if (mensaje_texto == "/Porciones") {
      bot.sendMessage(id_usuario, "Ingresa el número de porciones a
dispensar por horario.");
      ValidarPorciones = 1;}
    if (ValidarAgua == 1 && datonumero < 800 && datonumero > 1) { agua =
datonumero; }
    if (IngresoPeso == 1 && datonumero > 1 && datonumero < 40) {
      peso = datonumero;
      if (mascota == 1 && peso >= 1 && peso < 5) {
        bot.sendMessage(id_usuario, "SE RECOMIENDA:\nMáximo 200g. de
alimento seco diario, aprox. 4 porciones, max. 2 porciones x horario\nY
máximo 115ml. diarios de agua, max 40ml. x horario\n");}
      if (mascota == 1 && peso >= 5 && peso <= 10) {
        bot.sendMessage(id_usuario, "SE RECOMIENDA:\nMáximo 400g. de
alimento seco diario, aprox. 8 porciones, max. 3 porciones x horario\nY
máximo 230ml. diarios de agua, max 80ml. x horario\n");}
      if (mascota == 2 && peso >= 1 && peso < 5) {
```

```
        bot.sendMessage(id_usuario, "SE RECOMIENDA:\nMáximo 125g. de
alimento seco diario, aprox. 3 porciones, max. 1 porciones x horario\nY
máximo 250ml. diarios de agua, max 80ml. x horario\n");}
      if (mascota == 2 && peso >= 5 && peso < 10) {
        bot.sendMessage(id_usuario, "SE RECOMIENDA:\nMáximo 250g. de
alimento seco diario, aprox. 5 porciones, max. 2 porciones x horario\nY
máximo 500ml. diarios de agua, max 170ml. x horario\n");}}
    if (mascota == 3 && peso >= 10 && peso < 15) {
      bot.sendMessage(id_usuario, "SE RECOMIENDA:\nMáximo 375g. de
alimento seco diario, aprox. 8 porciones, max. 3 porciones x horario\nY
máximo 750ml. diarios de agua, max 250ml. x horario\n");}
    if (mascota == 3 && peso >= 15 && peso < 25) {
      bot.sendMessage(id_usuario, "SE RECOMIENDA:\nMáximo 625g. de
alimento seco diario, aprox. 13 porciones, max. 4 porciones x horario\nY
máximo 1250ml. diarios de agua, max 400ml. x horario\n");}
    if (mascota == 4 && peso >= 25 && peso < 30) {
      bot.sendMessage(id_usuario, "SE RECOMIENDA:\nMáximo 750g. de
alimento seco diario, aprox. 15 porciones, max. 5 porciones x horario\nY
máximo 1500ml. diarios de agua, max 500ml. x horario\n");}
    if (mascota == 4 && peso >= 30 && peso <= 40) {
      bot.sendMessage(id_usuario, "SE RECOMIENDA:\nMáximo 1kg. de
alimento seco diario, aprox. 20 porciones, max. 7 porciones x horario\nY
máximo 2000ml. diarios de agua, max 700ml. x horario\n");}
    IngresoPeso = 0;}
  if (ValidarPorciones == 1 && datonumero < 10 && datonumero > 1) {
porciones = datonumero; }
  for (int n=1;n<4;n++){
    if (ValidarHorario == n && datonumero > 0 && datonumero < 24) {
    parteEntera = int(datonumero);
    parteDecimal = (datonumero - parteEntera) * 100;
    if (parteDecimal < 60) {
      horas[ValidarHorario - 1] = parteEntera;
      minutos[ValidarHorario - 1] = parteDecimal;
      ValidarHorario = 0;}}}}
void ActivarBomba(int tiempoDelay) {
  digitalWrite(PinRelay, LOW);
  delay(tiempoDelay);
  digitalWrite(PinRelay, HIGH);
  delay(2000);}
void ActivarServo() {
  miservo.write(100);
  delay(250);
  miservo.write(175);
  delay(1000);}
void ActivarZumbador() {
  for (int i = 0; i < 4; i++) {
    digitalWrite(PinZumbador, HIGH);
    delay(300);
    digitalWrite(PinZumbador, LOW);
    delay(300);}}
```