Facultad de Ingeniería, UNMDP

Departamento de Electrónica

PROYECTO FINAL:

Sistema WIFI de monitoreo y alimentación de animales a distancia

Autor: Bernardo Álvarez

Director:
Ing. Carlos Bonadero
Co-director:
Ing. Esteban Gonzalez

Resumen

En la actualidad, las mascotas son un integrante muy importante de las familias. Sin embargo, muy frecuentemente la mascota debe quedar sola en el hogar por distintas situaciones, entonces aquí la necesidad de encontrar una solución que sea capaz de alimentar e incluso cuidar al animal.
En este trabajo se diseñó e implementó un sistema, llamado *Wifeed*, que es capaz de entregar una cantidad determinada de alimento balanceado en el momento que el usuario lo desee, ya sea activándolo desde su teléfono móvil, de una página web o bien de manera automática a determinadas horas pre-programadas. Este dispositivo es también capaz de recolectar y enviar información al usuario. De forma de conocer cuanto alimento queda, si esta en buen estado, como así también de a qué horas se alimentó la mascota. Además envía una alerta indicando que se esta terminando el alimento, con nivel pre-programable.
El dispositivo desarrollado es capaz de tener acceso a Internet mediante la conexión con una red local WiFi, dándole la posibilidad al usuario de tener un control local o remoto. El dispositivo cuenta además con actuadores periféricos para recargar bebederos de agua, encender luces, accionar sistemas de calefacción o ventilación, abrir puertas, entre otros. Para el diseño del dispositivo se utilizó la iniciativa de código abierto Source NodeMCU para el desarrollo de sistemas IoT (Internet of Things, Internet de las cosas). La unidad principal del dispositivo se implementó en un integrado ESP8266 que es un chip de bajo costo WiFi con una pila TCP/IP completa y un microcontrolador.
En este microcontrolador también se aloja un servidor, el cual da respuesta al usuario a través de Internet. Además de la unidad principal, el sistema completo consta de una aplicación Android que es una alternativa de acceso a través de un smartphone o tablet.

Índice general

Índice de figuras

Capítulo 1

Introducción y objetivos

1.1. Planteamiento de la problemática

En la sociedad actual existen millones de personas que conviven diariamente con algún tipo de animal. Las mascotas prácticamente forman parte de la familia, es por esto que los dueños de los animales no dudan en gastar grandes sumas de dinero en el bienestar del animal.

Un tema fundamental que se plantea, es la situación en la que el animal debe quedar sólo en la casa. En muchas ocasiones, las personas deben viajar por cuestiones de trabajo, vacaciones o temas familiares, sin la posibilidad de llevar con ellos a sus mascotas, lo que ocasiona reales complicaciones.

De esta situación surge el proyecto que tiene por objetivo encontrar una solución que permita asistir y monitorear a la distancia al animal. La idea es no solo alimentar y recargar bebedero de agua sino también se busca que el dispositivo realice otras acciones necesarias como abrir o cerrar puertas, prender estufas o ventilación. Además, mediante el análisis de datos tomados por sensores, como horario en que comió, cuánto comió, etc, el dispositivo debe ser capaz de proveer información sobre el estado del animal. Finalmente es necesario conocer cuanto alimento hay disponible y para cuántos días de abastecimiento alcanza, y de enviar alertas.

Todas estas características permitirán al dueño de la mascota estar tranquilo sabiendo que puede estar conectado con el animal por más lejos que este, pudiendo cuidarlo, controlarlo y alimentarlo desde cualquier parte del mundo de manera fácil y segura.

En el mercado existen dispositivos similares, sin embargo, ninguno integra todas las características desarrolladas en este proyecto.

1.2. Sistemas comerciales existentes en el mercado

Si bien existen en el mercado algunos sistemas para la automatización en el proceso de alimentación de animales, no reúnen todas las funcionalidades requeridas, sobre todo en lo que respecta al monitoreo y almacenamiento de datos de interés vía una comunicación inalámbrica de manera remota. No se encuentra en el mercado actual sistemas inteligentes de acceso remoto con comunicación inalámbrica que integren automatización de alimentación, monitoreo de estados y almacenamiento de datos y eventos. Los sistemas existentes tan solo permiten automatización en la entrega de alimento mediante sistemas mecánicos o programación de horarios prefijados, sin la posibilidad de acceder al sistema para verificar su estado.

Realizando un estudio de mercado se elaboró un análisis FODA para visualizar con mayor claridad la situación:

- **Fortalezas:**
  - Bajo costo en hardware y materiales a utilizar para su construcción.
  - Utilización de softwares de desarrollo gratuitos y de libre uso.
  - Sistema con conectividad a Internet a través de red inalámbrica, convirtiéndolo en un sistema inteligente, sin encontrarse en el mercado sistemas con esta característica.
  - Interfaz para el usuario multiplataforma, a través de una página web y aplicación móvil para cualquier dispositivo con sistema Android.
  - Bajo consumo de energía. Batería de emergencia ante cortes de energía de la red eléctrica.
- **Oportunidades:**
  - Poca competencia en el mercado nacional actual.
  - Amplia superioridad tecnológica frente a productos actuales similares del mercado.
  - Gran crecimiento del sector agrícola-ganadero debido a las condiciones políticas y económicas actuales del país.
- **Debilidades:**

  - Para poder acceder al sistema de forma remota (de manera no local) se requiere que la zona geográfica donde se encuentre instalado el sistema tenga cobertura y acceso a Internet, ya sea Internet por cableado convencional, satelital, por aire, o Internet móvil. De lo contrario el sistema solo puede ser utilizado dentro de la red local de donde esté instalado.

- **Amenazas:**

  - Dada la explosión de desarrollo en IoT (Internet of Things, Internet de las cosas) podrían surgir proyectos de productos similares.

1.3. Funciones y características del sistema

El objetivo del proyecto es el diseño y construcción de un sistema inalámbrico que permita alimentar y monitorear animales sin la necesidad de estar presente en el lugar, pudiéndose programar para que el sistema realice acciones en forma automatizada o bien realizar acciones de forma remota desde una PC o teléfono móvil. Esto es posible ya que el dispositivo principal del sistema es capaz de tener acceso a Internet mediante la conexión con una red local WiFi, dándole la posibilidad al usuario de tener un control local o remoto.

El proyecto consiste en el diseño de un dispositivo almacenador de alimento balanceado, granos, u otros alimentos secos granulados o en polvo, que a su vez sea capaz de expender a determinadas horas del día porciones de dicho alimento predefinidas por el usuario, o ser programado para que la entrega de alimento se realice cuando la bandeja de alimentación se quede sin alimento, o bien cuando el usuario de la orden desde una PC o teléfono móvil.

Lo más innovador de este producto, es que todas las funciones mencionadas pueden ser realizadas en forma remota, es decir a distancia. Por lo que una persona puede controlar y monitorear el proceso de alimentación de los animales sin la necesidad de estar en el lugar. Esto es posible gracias a que el sistema es capaz de conectarse a una red WiFi de manera inalámbrica y así tener acceso a Internet y poder vincularse con dispositivos que se encuentren en cualquier lugar del mundo.

El dispositivo también cuenta con actuadores periféricos para recargar bebederos de agua, encender luces, accionar sistemas de calefacción o ventilación, abrir puertas, entre otros.

El sistema consta de:

- Unidad principal: Contiene la tolva de almacenaje de alimento, sistema electromecánico de entrega del mismo, microprocesador y circuitería principal.
- Electroválvula para recargar bebedero de agua.
- Dos sensores de peso para controlar el peso del alimento almacenado y el peso del alimento en bandeja.
- Cuatro salidas de relés programables para poder controlar luces, sistemas de ventilación, sistemas de calefacción o electroimanes.
- Sistema de energía de emergencia por medio de batería.
- Interfaces para ejecutar el control, programación y visualización de eventos y sensores.

El diseño también consiste en una interfaz multiplataforma que permita al usuario la interacción con el dispositivo principal. Esta etapa consta de dos interfaces diferentes para que el usuario escoja la que mejor se adapte a sus necesidades:

- Interfaz web, donde el usuario accede a través de una página web.
- Aplicación móvil para teléfonos celulares y dispositivos Android.

Cualquiera de estas dos interfaces pueden usarse en simultaneo y cumplen la mismas funciones, permitirle al usuario configurar y programar el dispositivo principal, realizar acciones en tiempo real como alimentar o accionar los diferentes actuadores. También permite al usuario monitorear el estado del dispositivo, poder visualizar la cantidad de alimento disponible, consultar el estado de diferentes mediciones de sensores, almacenar en registros diferentes datos para un posterior análisis, recibir alertas y notificaciones ante determinados eventos. Si bien las dos opciones tienen las mismas funciones, brindan al usuario diferentes opciones de conectividad y portabilidad.

Por medio de la aplicación e interfaz web es posible:

- Definir y visualizar tamaño en peso de la porción de alimento a entregar.
- Programar los horarios de alimentación para los diferentes días de la semana, o que se ejecute ante un determinado evento o simplemente alimentar cuando se da la orden desde la aplicación en tiempo real.

- Monitorear cantidad de alimento restante en tolva, días de abastecimiento y recibir alarmas cuando se esté por agotar el alimento.
- Activar electroválvula para recargar bebedero.
- Activar diferentes dispositivos periféricos mediante la utilización de salidas a rele auxiliares.
- Visualizar las mediciones de los diferentes sensores.
- Visualizar eventos, como por ejemplo ultima hora en la que se recargo bebedero o se entregó alimento.
- Establecer la conexión a Internet y configuración.

Capítulo 2
Dispositivo Wifeed

La unidad principal del dispositivo es la que se encarga de comandar las siguientes operaciones:

- Ejecutar acciones programadas: recargar alimento, recargar bebedero, abrir/cerrar puertas, prender/apagar estufas, prender/apagar luces o ventilación.
- Ejecutar acciones por demanda: recargar alimento, recargar bebedero, abrir/cerrar puertas, prender/apagar estufas, prender/apagar luces o ventilación.
- Alojar un servidor web para interfaz con el usuario via un navegador web.
- Interfaz con el usuario via un dispositivo Andriod.
- Programación de acciones grabadas.
- Toma de datos por sensores.
- Análisis de datos.
- Avisos de alerta a usuario en pagina web y hacia dispositivo Android.

Para la implementación se utilizó un integrado ESP8266. Este es un chip de bajo costo que integra conexión por WiFi con una pila TCP/IP completa y un microcontrolador. En la siguiente sección se describen, a modo introductorio, las características principales, así como también se detalla cómo se realizó su programación, desde el entorno, hasta el lenguaje empleado. Al final del capítulo se describe la unidad central del dispositivo desarrollado *Wifeed*.

2.1. ESP8266

El ESP8266, Fig. 2.1, es un chip altamente integrado diseñado para las necesidades de un nuevo mundo conectado, el mismo permite diseñar sistemas que se conecten a una red WiFi. Fue creado por la empresa china Espressif ubicada en Shangai. Si bien fue creado hace bastantes años, su conocimiento masivo se dio en agosto del 2014 en su version básica ESP-01 fabricada por Al-Thinker.

Este integrado ofrece una solución completa y muy económica para conexión de sistemas a redes WiFi, permitiendo al diseñador delegar todas las funciones relacionadas con WiFi y TCP/IP del procesador que ejecuta la aplicación principal. Es capaz de funcionar como "adaptador de red" en sistemas basados en microcontroladores que se comunican con él a través de una interfaz UART (Universal Asynchronous Receiver-Transmitter).

Si bien el ESP8266 es mayormente utilizado como se menciona anteriormente como esclavo de un microcontrolador a través de comandos AT para poder acceder a Internet o una red WiFi, es en si mismo un microcontrolador. Por lo tanto tiene potentes capacidades de procesamiento y almacenamiento que le permiten integrarse con sensores y dispositivos específicos de aplicación a través de sus GPIOs (General Purpose Input Output, Entrada Salida de Propósito General) con un desarrollo mínimo y carga mínima durante el tiempo de ejecución. Ésto le permite ser la unidad lógica principal de un sistema. Su alto grado de integración en el chip permite una circuitería externa mínima, y la totalidad de la solución, ya que el módulo está diseñado para ocupar el área mínima en un PCB. Esta característica y su bajo consumo lo convierten en el producto ideal para aplicaciones wireless y dispositivos del IoT.

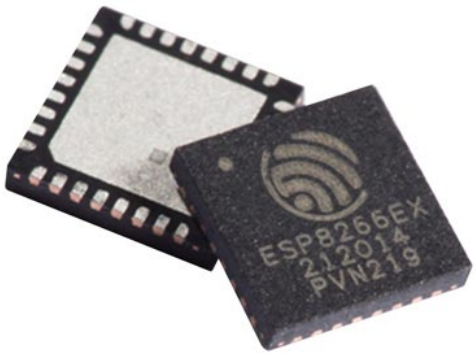


Figura 2.1 – *Integrado ESP8266.*

Este dispositivo consiste de un módulo WiFi serial muy económico el cual concentra las características mínimas necesarias para poder utilizarse sin problemas, debido a que la documentación en ingles y español de este módulo ya esta siendo desarrollada y se ha formado una verdadera comunidad realizando innumerables aportes día a día.

2.1.1. Características

Hardware: Las principales características del dispositivo son:

- Procesador integrado RISC Tensilica Xtensa LX106 de 32 bits con una velocidad de clock de 80MHz.
- RAM de instrucción de 64 KB, RAM de datos de 96 KB
- IEEE 802.11 b/g/n WiFi
  - Tiene integrados: TR switch, balun, LNA, amplificador de potencia de RF y una red de adaptación de impedancias
  - Soporte de autenticación WEP y WPA/WPA2
- Stack TCP/IP integrado.
- Posee GPIO, I2C, SPI, PWM,UART.
- ADC integrado de 10 bit de precisión.
- 80 KBytes de memoria RAM.
- Capacidad de memoria externa flash QSPI - 512 KB a 4 MB (puede soportar hasta 16 MB.)
- Potencia de salida: +19,5dBm en modo 802.11b
- Rango de temperatura de operación: $-40 \sim 125\ ^{\circ}C$
- Consumo en modo de baja energía: ¡10 uA.
- Tensión de alimentación de 3,3 V.

Consumo eléctrico

El consumo del dispositivo depende de diferentes factores como del modo en el que se este trabajando, de los protocolos que se estén utilizando, de la calidad de la señal WiFi y sobre todo de si envía o recibe información a través de la red. Oscila entre los 0,5uA (microamperios) cuando el dispositivo está apagado y los 170mA cuando transmite a máxima intensidad de señal.

Debido a los sectores a los que va enfocado, dispositivos del IoT y móviles, el ESP8266 requiere de una gestión de energía eficaz. Dispone de una arquitectura de bajo consumo que trabaja en 3 modos.

- Active mode o modo activo: a pleno rendimiento.
- Sleep mode o modo dormido: sólo el RTC (Real Time Clock) está activo para mantener la sincronización. Se queda en modo alerta de los posibles eventos que requieran atención. Mantiene en memoria los datos de conexión y así no hace falta volver a establecer la conexión con la WiFi. Consume entre 0,6 mA y 1 mA.
- Deep sleep o modo en sueño profundo: el RTC está encendido pero no operativo. Debe pasar por el modo dormido antes de despertar. Hay que llevar especial cuidado con los datos ya que en este estado es como si estuviera apagado y todos los datos que no estén almacenados se pierden. Consume alrededor de 20 uA.

2.1.2. Variantes y módulos del ESP8266

Existen en el mercado diferentes presentaciones de módulos WiFi ESP8266, pero es importante recalcar que todas las variantes se basan en el mismo integrado ESP8266, las diferencias consisten en la cantidad de memoria flash disponible y en el montaje del mismo que hacen los fabricantes para permitir acceso a mayor número de pines y el agregado de algún componente adicional como pueden ser antenas, convertidores de nivel de señal, leds, entre otros. A continuación se muestran algunas de las diferentes presentaciones de dichos módulos.

ESP-01:

Esta version inició siendo la más popular y con mayores ventas. Tiene disponible dos pines GPIO digitales para controlar sensores y actuadores. También se puede llegar a utilizar para este uso los pines Rx y Tx si no se utilizan para la comunicación a través del puerto serie. Se puede programar a través de un adaptador serie/USB.

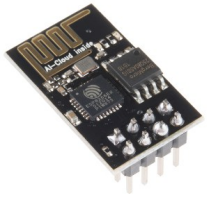


Figura 2.2 – *ESP-01*

- Dimensiones: 14,30 mm x 24,80 mm.
- Conexiones: 8 pines entre alimentación y GPIO.
- Antena impresa en la PCB.
- Sin apantallar.
- Alimentación: 3,3 V.

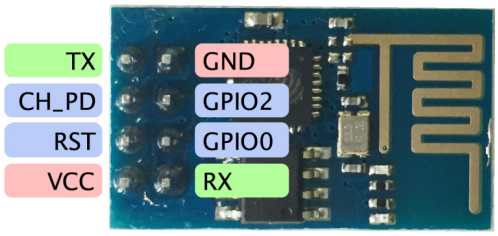


Figura 2.3 – *Pines ESP-01*

ESP-02:

Este modulo consta de 3 pines digitales al exterior GPIO0, GPIO2, y GPIO15 y además acepta una antena WIFI externa lo que le hace muy interesante, para montajes que requieran mayor alcance de la señal.

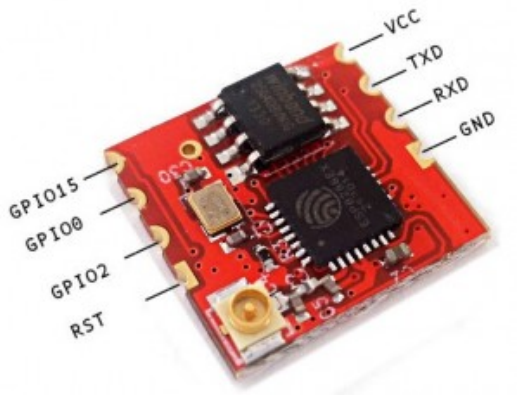


Figura 2.4 – *ESP-02*

- Dimensiones: 14,20 mm x 14,20 mm.
- Conexiones: 8 conexiones de superficie.
- Sin antena en la placa pero con un conector para antena externa.
- Sin apantallar.
- Alimentación: 3,3 V.

ESP-03:

Esta variante presenta un aumento significativo de las GPIO disponibles. Incluye una antena cerámica.

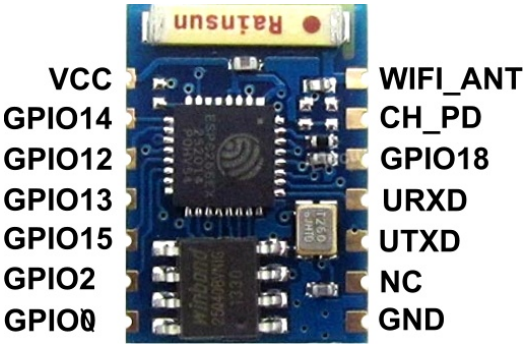


Figura 2.5 – *ESP-03*

- Dimensiones: 17,30 mm x 12,10 mm.
- Conexiones: 14 conexiones de superficie en los dos lados mayores.

- Antena de tipo cerámico.
- Sin apantallar.
- Alimentación: 3,3 V.

ESP-12E:

En este caso se tiene acceso a 11 puertos GPIO, de los cuales uno es una estrada analógica/digital con una resolución de 10-bit (1024 valores posibles). También tiene pines para comunicación SPI (Serial Peripheral Interface), lo que hace posible su interconexión con pantallas lcd, memorias y otros dispositivos.

Figura 2.6 – *ESP-12E*

- Dimensiones: 24,00 mm x 16,00 mm.
- Conexiones: 16 conexiones de superficie en los dos lados mayores.
- Antena de tipo cerámico.
- Con blindaje contra interferencias.
- Alimentación: 3,3 V.

NodeMCU ESP-12E Dev Kit:

Éste módulo fue el seleccionado para ser utilizado en el proyecto ya que por sus características resulta óptimo para realizar las tareas necesarias. El circuito NodeMCU ESP-12E Dev Kit fue desarrollado en el entorno del proyecto open source NodeMCU, descripto en la sección siguiente, un detalle de este módulo puede verse en la sección 2.2.1.

2.2. NodeMCU

NodeMcu es una iniciativa de código abierto (open source) [1, 2] para el desarrollo de modelos sencillos que ayuden a integrar la IoT en distintas áreas. Para ello, se desarrollaron modelos de hardware y software que facilitan la utilización de programas y aplicaciones basados en WiFi. El código del firmware y el diseño del hardware están disponibles en GitHub [1]. El nombre de NodeMCU representa por tanto la unión de la placa de desarrollo junto con el firmware.

2.2.1. NodeMCU ESP-12E Dev Kit:

El módulo NodeMCU ESP-12E Dev Kit es una variante para implementaciones IoT basado en el ESP8266 con una memoria flash de 4 MBytes, Fig. 2.7. El diagrama de pines del módulo puede verse en la Fig. 2.8.

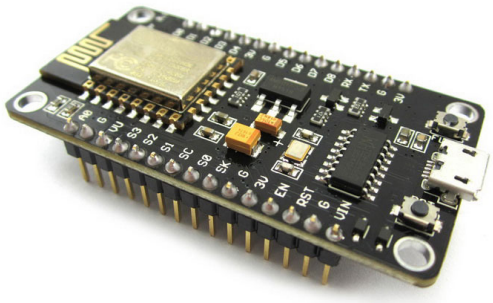


Figura 2.7 – *Módulo ESP-12E DEV Kit.*

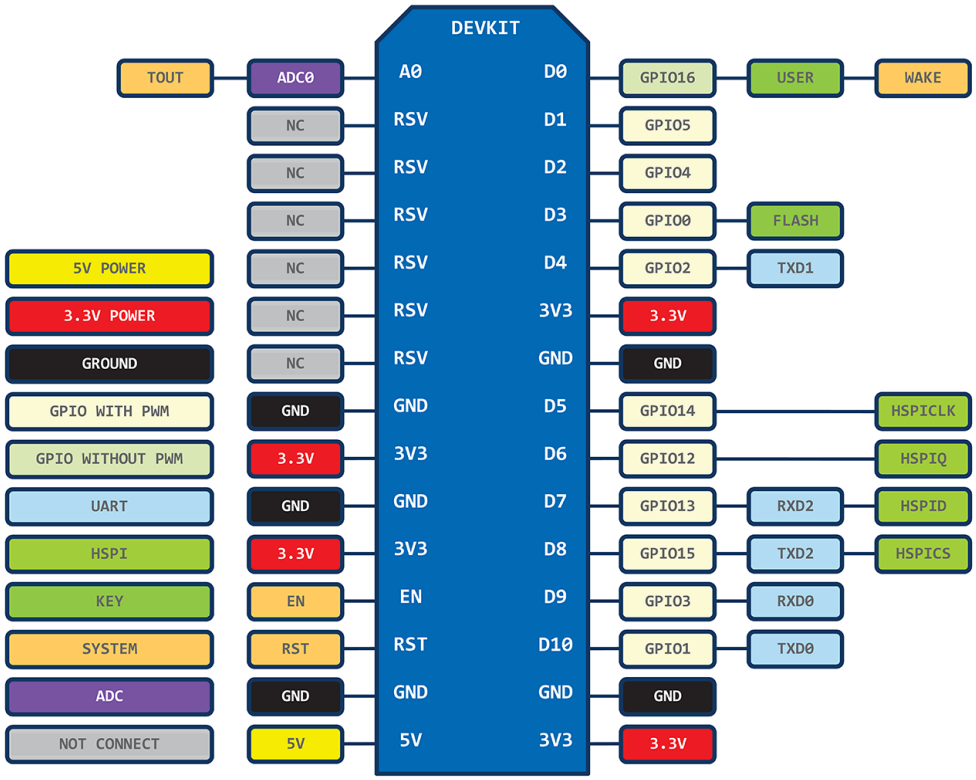


Figura 2.8 – *Pines ESP-12E DEV Kit.*

A diferencia de otros módulos viene con todo lo necesario para empezar a trabajar de forma autónoma. Incluye un adaptador serie/USB, lo que permite conectarlo a una PC y programarlo directamente. También posibilita alimentarlo a través del mismo conector microUSB ya que la placa integra la circuitería necesaria para la adaptación de tensión de 5V del puerto USB a 3,3 V que es la tensión de alimentación del integrado ESP8266. Está basado en el ESP-12E y la última versión oficial es la 2.

Lo más interesante de este módulo es que se puede descargar un firmware que permite programar en lenguajes como LUA, Python, Arduino, Basic o JavaScript, que son lenguajes muy conocidos y desarrollados en el mundo de la programación, permitiendo así la independencia de los comandos AT.

- Dimensiones: 30,85 mm x 47,35 mm.
- Conexiones: 30 pines separados una décima de pulgada (apto para conectar en protoboard) y USB.

- Conversor serie/USB.
- Antena impresa en la PCB.
- Con blindaje contra interferencias.
- Alimentación: 3,3 V y 5V.
- Dos pulsadores RESET y FLASH.

2.2.2. Firmware

Un firmware es un software que maneja físicamente a un hardware dado. Por ejemplo el programa BIOS de una computadora es un firmware cuyo propósito es activar una máquina desde su encendido y preparar el entorno para cargar un sistema operativo en la memoria RAM y disco duro.
En este caso, el módulo NodeMCU ESP-12E permite trabajar con un firmware. De esta forma es posible escribir código con el lenguaje de programación LUA.
Básicamente existen tres formas de construir un firmware NodeMCU: (i)servicio de compilación en la nube, (ii)imagen de Docker, (iii)entorno Linux dedicado.
La primer opción provee un firmware listo para usar. Existe un servicio de compilación en la web con una interfaz de usuario agradable y opciones de configuración [3] para que el usuario elija los módulos requeridos en su aplicación, los agregue y finalmente compile el firmware que luego cargará en el dispositivo.
La segunda opción, imagen de Docker, se emplea cuando no es necesario un control total sobre la cadena completa de herramientas. De esta forma no es necesario configurar una máquina virtual Linux con el entorno de compilación. Ésta solución se encuentra en la web [4].
Finalmente, la tercer opción, es construir un entorno propio de desarrollo completo con toda la cadena de herramientas. Un detalle sobre esto puede verse en [5].

En este proyecto se utilizó la primer opción que es el servicio de compilación en la nube.
Es importante deshabilitar los módulos que no se usarán para reducir el tamaño del firmware y liberar mamoria RAM ya que la RAM disponible es limitada y quedarse sin memoria puede causar un pánico en el sistema. La configuración predeterminada está diseñada para ejecutarse en todos los módulos ESP, incluidos los módulos de 512 KB, como ESP-01, y sólo incluye módulos de interfaz de propósito general que requieren como máximo dos

pines GPIO.

El compilador genera dos firmwares de salida con extensión *.bin*. Una de ellas es una compilación compatible con variables de coma flotante (float) y la otra con variables enteras (integer). En caso de no necesitar trabajar con variables en coma flotante es posible utilizar la compilación entera para reducir el tamaño de la memoria.

Lenguaje de programación Lua

El lenguaje de programación utilizado es el Lua (luna en portugués), éste es un lenguaje de programación interpretado de origen brasileño. Nació en 1993 al interior del Instituto TeCGraf de la Pontifícia Universidade Católica do Rio de Janeiro, con el objetivo puesto en los sistemas embebidos. El logo puede verse en la Fig. 2.9.
Un lenguaje interpretado es el lenguaje cuyo código no necesita ser preprocesado mediante un compilador, ya que consisten en scripts que son interpretados en tiempo real por un intérprete (el firmware en este caso), esto permite maximizar la eficiencia de los programas. El microcontrolador es capaz de ejecutar la sucesión de instrucciones dadas por el programador sin necesidad de leer y traducir exhaustivamente todo el código.
Lua es además un lenguaje de extensión, lo cual significa que no es un lenguaje para escribir programas ejecutables stand-alone. En lugar de esto, está pensado para integrarse dentro de un programa anfitrión (el firmware en este caso). Este programa anfitrión es el que se encarga de lanzar y ejecutar el programa Lua. Además, el programa anfitrión puede declarar funciones escritas en C que pueden ser invocadas desde el programa Lua.

Figura 2.9 – *Logo oficial Lua.*

Este lenguaje tuvo como principio fundamental la simplicidad, partiendo de un lenguaje pequeño y evolucionando hasta lo que es hoy en día: un lenguaje usado internacionalmente en áreas tan diversas como la creación de videojuegos, el desarrollo de aplicaciones de escritorio y la robótica. Actualmente, se encuentra en su versión 5.3.1, liberada en junio de 2015.

Es un lenguaje de código abierto debido a que se distribuye bajo licencia MIT (MIT, Massachusetts Institute of Technology) y está construido con ANSI C, facilitando estas dos características su portabilidad a una gran cantidad de sistemas operativos, entre ellos GNU/Linux, Windows y Mac OS X.

Características principales:

- Lua tiene manejo de memoria automático, esto significa que el programador no tiene que preocuparse por liberar la memoria RAM que su programa ya no este usando.
- Lua es un lenguaje de tipado dinámico, es decir, Lua asigna el tipo de dato correcto a una variable.
- Ofrece soporte para una programación orientada a objetos.
- Tiene un léxico basado en C: los punto y coma al final de sentencias son opcionales. Los identificadores constan de letras, números y underscore (no pueden empezar con números; tampoco se recomienda que empiecen con underscore, ya que esta notación se utiliza para identificadores internos de Lua).

Entorno de desarrollo ESPlorer

Existen varios entornos de desarrollo o IDE (Integrated Development Environment) para programar el modulo nodeNCU, en este caso se optó por el ESPlorer [6], Fig. 2.10. Una de las ventajas que tiene este IDE es que corre bajo Java y por lo tanto es multiplataforma, por lo cual se puede utilizar bajo Windows, Linux o MAC sin problemas. En cambio, LuaLoader, otro IDE alternativo, corre solo en Windows, ademas de tener muchas menos herramientas y un aspecto gráfico muy inferior.

El ESPlorer también puede ser utilizado con Python en vez de Lua ya que el ESP8266 también puede ser programado con Python, si es que previamente se le instaló el firmware de Micro Python en lugar de el NodeMCU. También da la posibilidad de comunicarse con el modulo directamente mediante comandos AT, sin la necesidad de escribir código.

A continuación se presentan algunas características del ESPlorer IDE.

- Realzado de sintaxis LUA y Python.
- Temas para personalizar el entorno de desarrollo.
- Comandos deshacer y rehacer.
- Repertorio de Snippets para acceso rápido a funciones muy usadas por el programador
- Autocompletado de código (Ctrl+Espacio).
- Log detallado.
- Repertorio de Snippets para acceso rápido a funciones muy usadas por el programador.

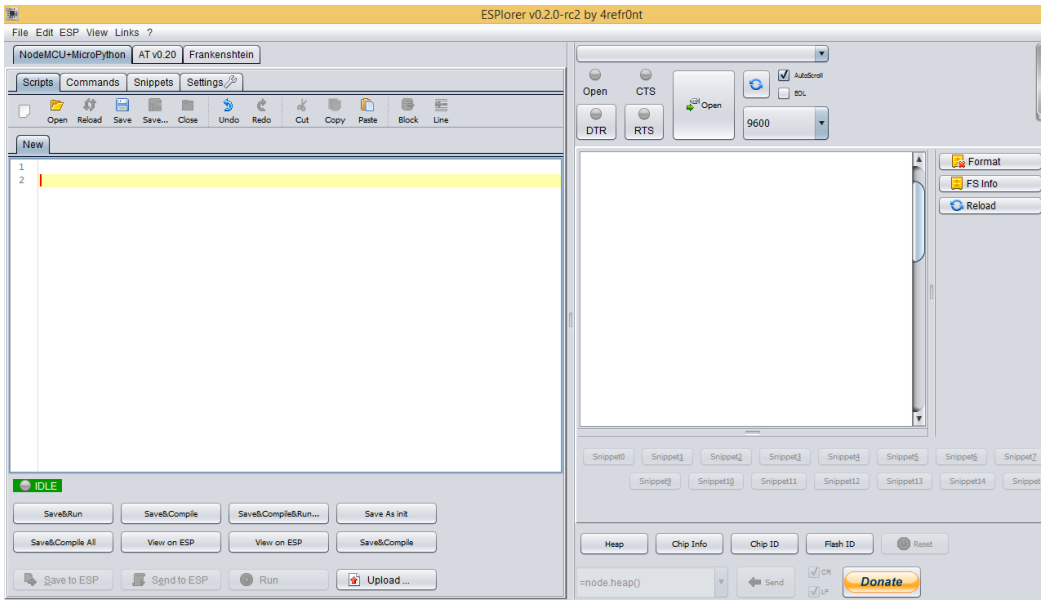


Figura 2.10 – *Entorno de desarrollo ESPlorer.*

Además incluye funcionalidades gráficas útiles como:

- Envío de comandos.
- Botón Heap para liberación de memoria.
- Botón Restart para resetear el dispositivo.
- Panel de configuración de Baudios.
- Panel de configuración WiFi. Se puede crear un punto de acceso en el dispositivo, o que el dispositivo se conecte en modo cliente a una red WiFi.
- Panel para subir ficheros LUA al dispositivo, mostrar los ficheros que tiene el dispositivo, o subir el contenido que estamos escribiendo en consola.
- Panel GPIO para la configuración de pines en el dispositivo, tanto lectura, escritura y repetición de eventos.

Todo esto de manera visual, sin la necesidad de introducir comandos. Lo que resulta muy práctico a la hora de desarrollar proyectos ahorrando mucho tiempo de programación.

Ventana principal

A continuación se presenta una breve descripción de los diferentes componentes que forman el IDE.

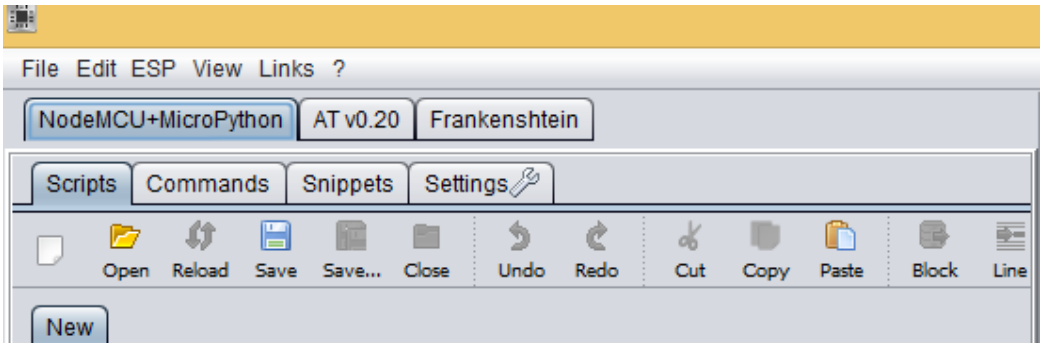


Figura 2.11 – *Detalle comandos de ESPlorer.*

En la esquina superior izquierda, Fig. 2.11, se pueden ver todas las opciones habituales que se encuentran en cualquier software. Permitiendo Crear un nuevo archivo, Abrir un nuevo archivo, Guardar archivo, Guardar archivo como, Deshacer, Rehacer, etc.

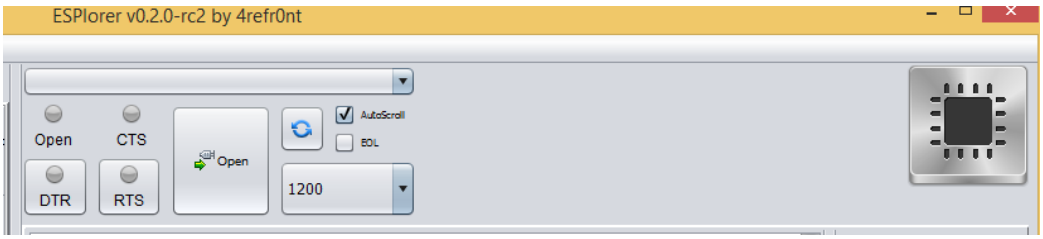


Figura 2.12 – *Detalle parámetros de comunicación con módulo.*

En la esquina superior derecha, Fig. 2.12, tiene todas las opciones necesarias para establecer la comunicación serie con el modulo.

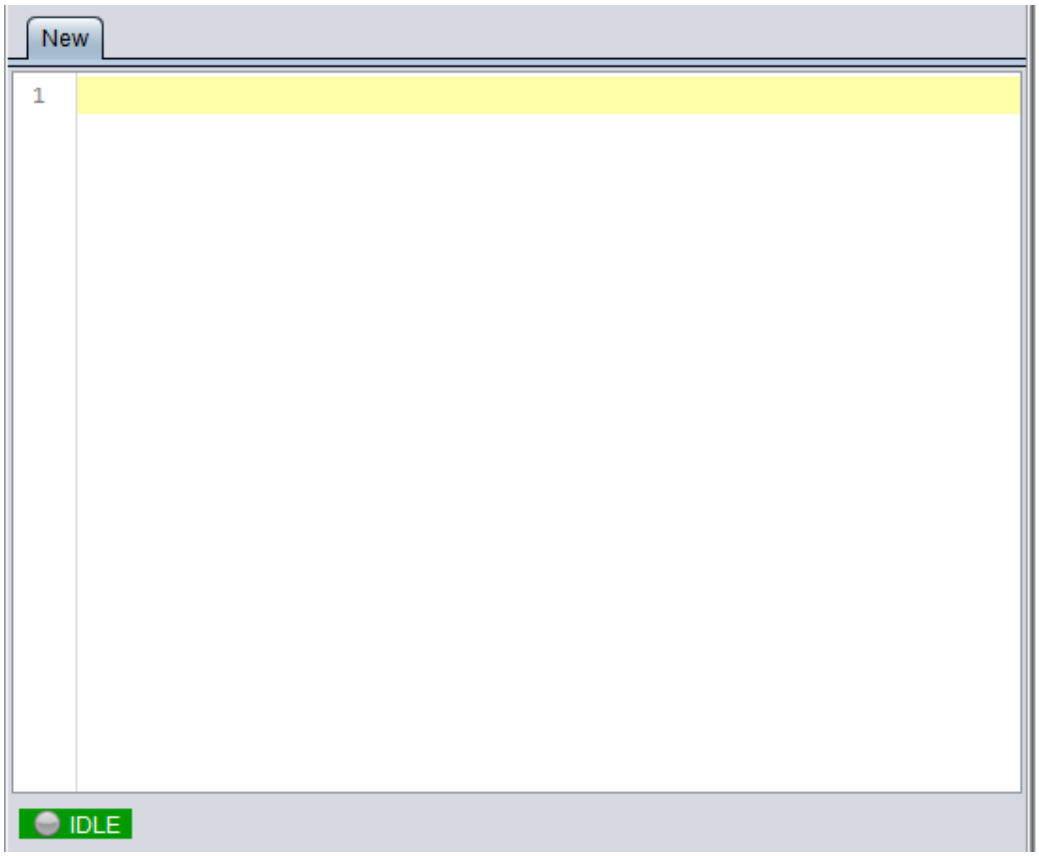


Figura 2.13 – *Detalle ventana de código.*

La ventana de código, Fig. 2.13, en la cual se puede o bien abrir un archivo que contenga el código o escribirlo. Los diferentes scripts y elementos del lenguajes se destacan en colores dependiendo de su función y tipo.

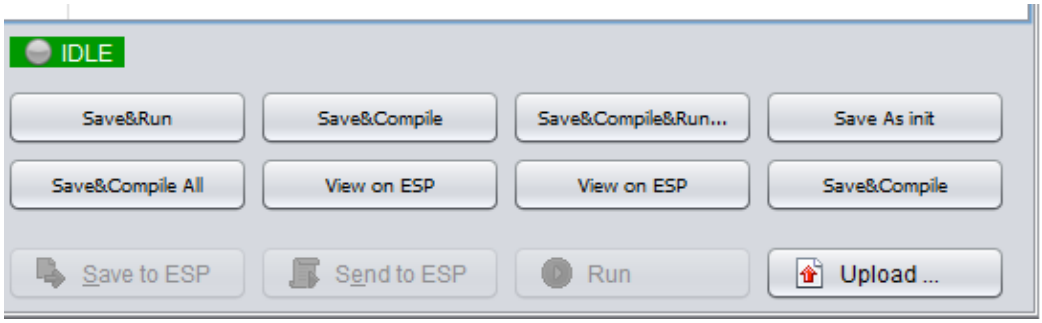


Figura 2.14 – *Detalle comandos para interctuar con el módulo.*

Debajo de la ventana de código, Fig. 2.14, se encuentran 12 botones

que ofrecen todas las funciones necesarias para interactuar con el ESP8266. “Guardar en ESPż “Enviar a ESP”son las que mas se usan.

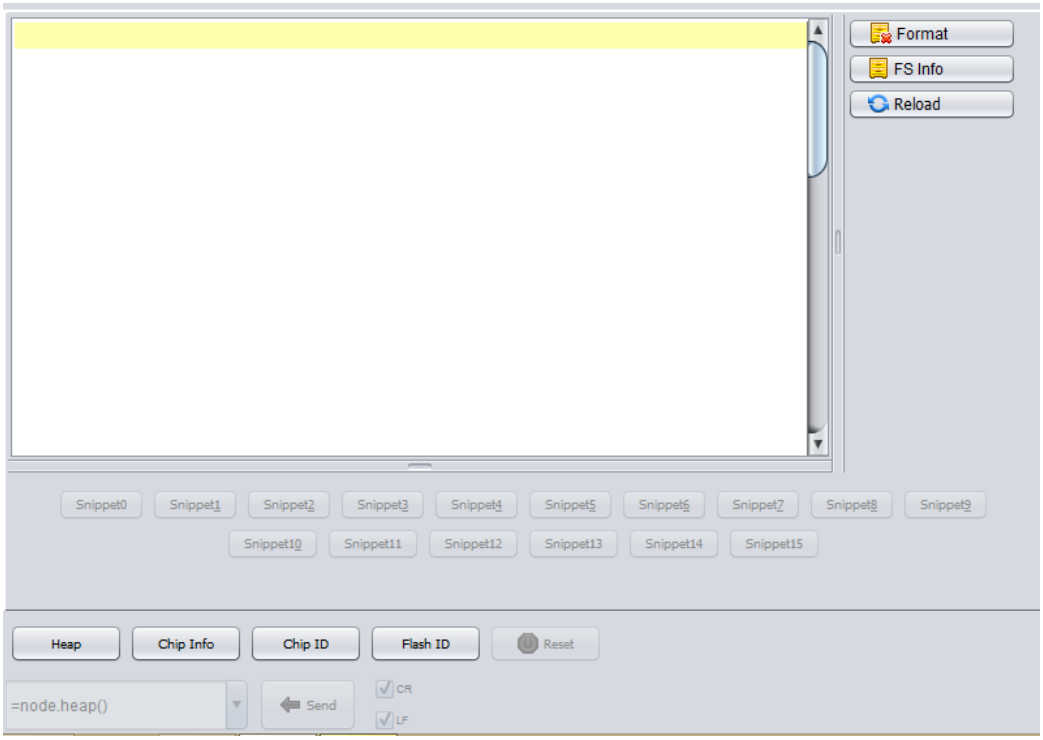


Figura 2.15 – *Detalle ventana de salida.*

En la ventana de salida, Fig. 2.15, se indica exactamente lo que esta pasando en el ESP8266. Permite visualizar errores y utilizar impresiones en el código para depurar los proyectos.

2.3. Funcionamiento y hardware

Como ya se dijo anteriormente, el sistema *Wifeed* esta diseñado para poder asistir y monitorear a animales de forma remota. Asistir se refiere a recargar comida y/o agua, abrir o cerrar puertas, prender o apagar luces, manejar sistemas de calefacción o ventilación, etc. Por otro lado, monitorear se refiere a que el usuario tiene la posibilidad de llevar control en tiempo real de el estado del animal, mediante el sensado y análisis de distintas variables como los horarios en los que el animal come, la cantidad de alimento comido, etc, y de esta forma saber si la mascota se esta alimentando correctamente y si tiene comida a disposición.

El dispositivo *Wifeed* es el componente más importante del sistema, ya que es donde se encuentra la parte lógica que se encarga de la comunicación, procesamiento de ordenes, lectura de sensores y control de actuadores. Siendo además la unidad donde se encuentra el dispenser de almacenamiento del alimento y el sistema mecánico encargado de entregar el mismo.

Alimentación

La función principal del dispositivo es la de alimentar al animal en forma remota. Las opciones son acceder en forma remota o simplemente automatizar la tarea. El control y programación de la misma puede realizarse tanto por interfaz web como por la interfaz móvil.

El primer paso es definir desde la configuración el tamaño de la porción de alimento que se entregará en cada suministro de comida. Este valor debe ser ingresado en gramos y dependerá de cada mascota y persona. Dicha porción no puede ser menor a 10 gramos ni superior a 999 gramos.

Una vez determina el peso de la porción por el usuario, el control se realiza mediante un sensor de peso del tipo celda de carga colocado debajo del recipiente de alimentación. Al momento de expender el alimento, se activa el sistema mecánico que hace que el alimento comience a caer, a medida que el alimento cae sobre el recipiente, el sistema lee cada pequeños intervalos de tiempo el valor emitido por el sensor de peso colocado debajo de dicho recipiente, una vez que la cantidad de alimento en este es igual al tamaño de la porción configurada la entrega de alimento se detiene dejando de girar.

Otro sensor de peso se encarga de medir la cantidad de alimento disponible en el dispenser de almacenamiento. El valor medido por este sensor es consultado luego de cada entrega de alimento y mostrado en la pantalla de Estado, así, el usuario puede visualizar las reservas de comida y recibir alertas en caso de que la misma se este por acabar.

La alimentación se puede llevar a cabo de dos maneras, a demanda, es decir cuando el usuario da una orden en en el momento que desee, presionando el botón de “Alimentar ahora”de cualquiera de los medios de control, o bien, programar el dispositivo para que entregue el alimento a determinadas horas del día de manera autónoma. Esta última opción permite configurar de una a tres veces de entregas al día, debiéndose indicar los horarios en los que se realizarán las entregas del alimento.

En el bucle principal del programa, cada 60 segundos, el timer realiza una interrupción para incrementar en un minuto la hora, luego del incremento el dispositivo compara la hora actual con las horas programadas, si coinciden ejecuta la acción de alimentar. Además se realiza un ajuste cada hora mediante Internet, lo que se explica en la sección 2.3.

En el diagrama siguiente, Fig. 2.16,se muestra este proceso.

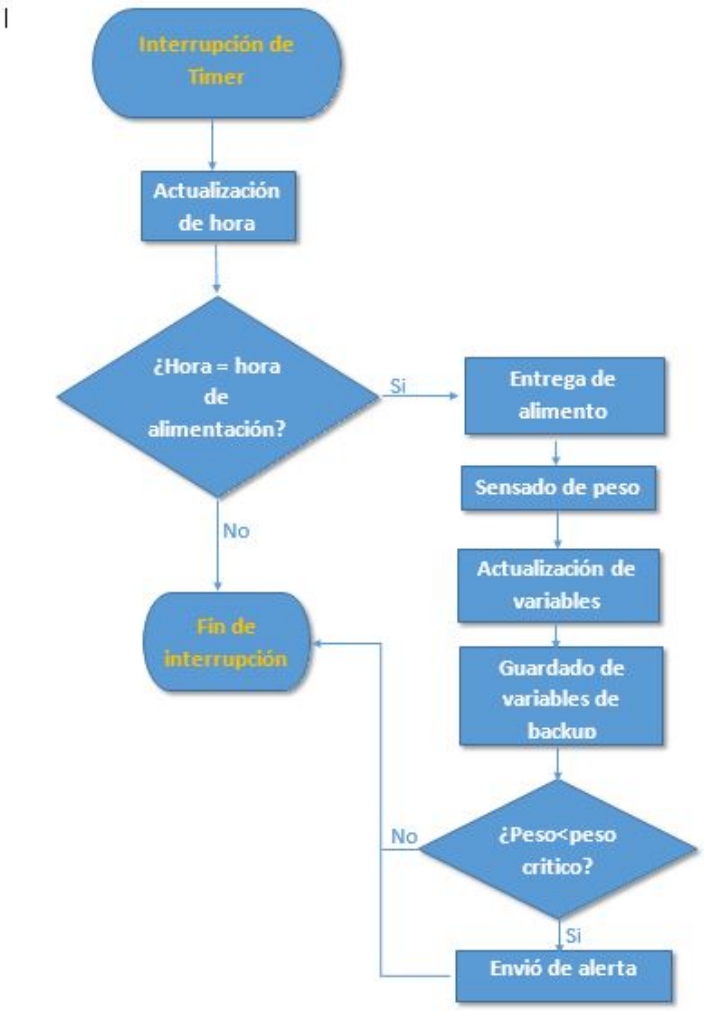


Figura 2.16 – *Diagrama de flujo interrupción del timer.*

Cada vez que el dispositivo hace una entrega de alimento, se registra el día y hora de dicho evento. Este valor se almacena y muestra en el visor de eventos de las interfaces de control, permitiendo así que el usuario sepa cuando fue la última vez que se alimentó a la mascota y verificar la correcta entrega del

alimento. Ademas, también realiza un pitido característico haciendo sonar un buzzer para que el animal asocie ese sonido con el momento de comer.

Periféricos

Como se mencionó anteriormente, el módulo ESP8266 tiene 16 entradas/salidas disponibles. Esto da la posibilidad de poder controlar muchos dispositivos secundarios necesarios para realizar las funciones requeridas por el sistema.

Una de las salidas más importantes es la que va a comandar el sistema mecánico encargado de hacer caer el alimento. Éste consiste en un tirabuzón o sin fin agarrado sobre el eje de un motor de corriente continua de 12 volts. El mismo deberá tener el torque suficiente para evitar atascamientos del mecanismo.

Los dos sensores de peso destinados al sensado del nivel de alimento tanto en la tolva principal y el recipiente de alimentación son del tipo celda de carga de 2 mV/V, el primero de 20 kg de carga máxima y el segundo de 1 kg de carga máxima. La lectura de estos es realizada por el conversor analógico digital de 10 bits interno del modulo ESP8266. Como este conversor sólo tiene una entrada, y los sensores a leer son dos, hay una etapa de selección previa a la entrada del conversor, la misma consiste en un multiplexor de dos entradas y una salida, con la entrada de selección comandada por una de las salidas del microcontrolador.

Para la acción de recargar bebedero de agua se asigna una de las salidas que controla la apertura de una electrovalvula que permite el paso de agua y así llena el recipiente. La cantidad de agua que se entregue en cada orden dependerá de la presión de agua, es por esto que el usuario podrá configurar el tiempo en segundos en el que la válvula deberá permanecer abierta. Así regulara él el tiempo necesario para recargar a un nivel deseado su recipiente.

Otra salida es destinada al control de la apertura y cierre de una puerta para que el animal pueda entrar o salir de un determinado ambiente. Dicha salida comanda una cerradura electrónica que es habilitada o deshabilitada por el usuario cuando lo desee conveniente o a horas pre-definidas.

El sistema cuenta también con cuatro salidas de reles programables, las cuales son activadas y desactivadas desde las diferentes interfaces de control

y permiten visualizar su estado. éstas pueden ser usadas para encender o apagar luces, controlar la calefacción, habilitar un sistema de ventilación, subir persianas o cualquier cosa que el usuario desee controlar, permitiéndose un consumo de hasta 16 amperes por cada salida.

Seguridad

Un requerimiento importante para este sistema es la confiabilidad y seguridad, ya que el bienestar de la mascota dependerá de que nuestras ordenes y configuraciones se mantengan y cumplan como lo deseamos. Es por esto que el sistema cuenta con un sistema de abastecimiento de energía de emergencia en caso de cortes de la red eléctrica. Este se basa en un circuito con una batería de 12 volts, que en caso de detección de corte de alimentación, automáticamente comienza a proveer de la corriente necesaria para abastecer al sistema y que todo siga funcionando normalmente. Esta batería se recarga automáticamente cuando vuelve el suministro eléctrico.

Otra medida de seguridad del dispositivo es realizar constantemente una copia de seguridad en forma de archivo de texto del valor de las variables y configuraciones importantes. Este se guarda en la memoria no permanente del microcontrolador. En caso de reiniciarse o colgarse el programa esta copia de respaldo siempre se mantiene y se lee al iniciar el sistema.
Como se puede notar, es de vital importancia que el sistema tenga acceso a la hora, es decir que tenga noción del tiempo de manea exacta. Durante la configuración inicial, se pide al usuario que ingrese por única vez la hora actual indicando horas y minutos, a partir de ahí el dispositivo es interrumpido cada un minuto por el timer interno, para realizar las tareas requeridas, e incrementar la hora. Sin embargo, este forma de actualizar la hora no es exacta y siempre tiene corrimientos generando errores a largo plazo.
Por esto, el dispositivo, cada una hora, se conecta a través de Internet con un servidor de tipo NTP (Network Time Protocol), *1.ar.pool.ntp.org* que entrega la cantidad de segundos transcurridos desde el 1ro de enero de 1970 a las cero horas. El dispositivo lo lleva al formato convencional y ajusta su propio reloj. En caso de que *Wifeed* se este empleando sin conexión a Internet, el ajuste de la hora se realiza cada vez que un dispositivo móvil con la aplicación este accesible.

2.3.1. Conectividad

Una de las característica a destacar y la más importante de este desarrollo es la de poder acceder al sistema desde cualquier parte del mundo. Esto es posible ya que el dispositivo se conecta a Internet a través de una red WiFi. O bien, si la red no tiene acceso a Internet, se lo puede programar accediendo localmente para que luego realice tareas automáticamente. Con esto se quiere evidenciar que si bien la característica más innovadora es el control a distancia, también se puede usar como dispositivo de alimentación programada en lugares donde no se cuente con Internet.

Ya sea de manera remota o local, la comunicación con el sistema se realiza accediendo al servidor embebido que se encuentra alojado en el microcontrolador, las diferentes interfaces de control se conectan a él mediante una dirección IP a traves de un puerto, ya sea la IP interna de la red generada por *Wifeed*, o bien, en el caso de uso remoto, la IP publica dada por el proveedor del servicio de Internet.

Para lograr las configuraciones y comunicaciones el dispositivo debe operar en dos modos diferentes, uno llamado "Modo APz el otro "Modo Host".

El primero llamado así por el concepto de Punto de Acceso (AP, Access Point). En este modo, el dispositivo genera una red WiFi local. Dicha red aparecerá bajo el nombre de *Wifeed* y estará asegurada con una contraseña de tipo WPA, esta red podrá ser detectada por todos los dispositivos que estén a su alrededor. Este primer modo resulta vital para el proceso de configuración inicial ya que es la primera puerta de comunicación que tiene el usuario para poder interactuar con el dispositivo y poder indicarle a que red WiFi se tiene que conectar en caso de querer hacer uso del modo remoto. Si se quiere usar el sistema de manera local ya que no se cuenta con una red con acceso a Internet, esta conexión a través de la red generada por *Wifeed* será el vínculo que permita el uso y programación del dispositivo, siendo esto último posible sólo cuando se este dentro del área de cobertura de dicha red.

En el modo Host, *Wifeed* es capaz de conectarse a la red WiFi del hogar, y a través de esta puede interactuar con los dispositivos que estén dentro de ella, donde cada uno es identificado por la dirección IP asignada por el router. Para poder acceder a la red es necesario que tenga la información del SSID de la misma y su password.

Una vez conectado a la red local, *Wifeed* pasa a ser accesible desde cualquier parte del mundo si la red tiene acceso a Internet, a través de la IP pública otorgada por el proveedor del servicio y habilitando un puerto determinado se puede acceder al dispositivo y controlarlo.

Como ya se dijo, para poder lograr que *Wifeed* se conecte a la red es necesario indicarle a qué red se tiene que conectar y de ser necesario introducir la contraseña. Para lograr esto, el dispositivo principal, en el proceso de configuración WiFi, hace un escaneo de las redes que están disponibles en el área, obtiene sus SSID que las identifican y almacena sus nombres. Estos son mostrados en pantalla en forma de lista. El usuario debe seleccionar una de esta y se desplegara un cuadro de texto para ingresar la contraseña. El sistema toma estos dos datos ingresados y comienza el proceso de autenticación con dicha red. Si el SSID y la contraseña ingresada son correctos el dispositivo guarda estos datos en un archivo de texto y cambia a modo Host dando por finalizado el proceso de configuración WiFi.

Una vez que el dispositivo tiene acceso a Internet, consulta al servidor *http://httpbin.org/ip*, este servidor tiene la particularidad que devuelve el valor de la dirección IP pública desde donde se le hace la consulta, este valor es mostrado en pantalla para que el usuario sepa con que dirección deberá acceder al dispositivo para el uso remoto.

Podemos representar el proceso de inicio del microcontrolador con el siguiente diagrama en bloques:

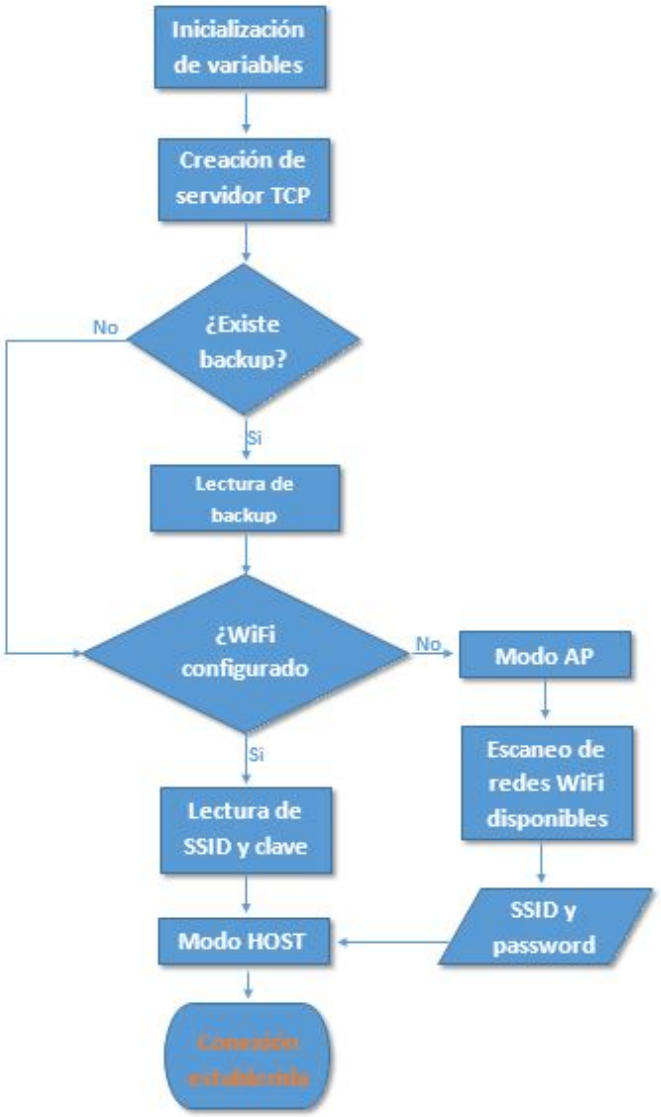


Figura 2.17 – *Diagrama de flujo inicio microcontrolador.*

2.3.2. Servidor

Como se dijo anteriormente, el dispositivo ESP8266 utilizado en este proyecto, tiene la capacidad de alojar un servidor TCP/IP embebido. Para crear este servidor el primer paso es el de indicar el puerto que se dejará abierto para recibir datos. El servidor quedará escuchando y al recibir un dato se generará una interrupción en el microcontrolador para atenderla.

En este sistema, el servidor puede recibir dos tipos de datos, solicitudes web desde un navegador si es que el usuario esta controlando el dispositivo desde la interfaz web, o datos desde una aplicación Android en caso de que se este usando la interfaz móvil.
Al enviar datos desde las diferentes interfaces, no solo se estarán enviando órdenes, también se envían configuraciones, es decir, se envían valores de variables que el dispositivo principal necesita leer y almacenar. Por ejemplo, al momento de definir el tamaño de la porción de alimento que debe entregarse, el servidor al recibir el dato, no solo tiene que saber leer esa magnitud sino que también tiene que saber a qué corresponde ese dato. Es decir, deberá "entender"que lo tiene que almacenar como tamaño de porción y cuál es el valor a guardar.
Para solucionar este problema se desarrolló un protocolo. Se estructuraron los mensajes de datos en palabras de longitud variable según el tipo de dato enviado. Los dos primeros dígitos indican qué tipo de dato se esta recibiendo, y el resto el valor de la variable. Por ejemplo, en el caso recibir el tamaño de porción de alimento, la variable tendrá en el encabezado el numero *01*, y si el tamaño de porción se definió en 150 gramos, entonces el servidor recibirá el dato *01150*. De esta forma el microcontrolador leerá los dos primero dígitos y sabrá que los dígitos siguientes corresponden al tamaño de porción, en este caso esperará recibir tres datos que son el peso. En caso de que la cantidad de gramos fuera un número de menos de tres dígitos se completará con ceros. Así, el microcontrolador almacenará su valor en la variable correspondiente. En la tabla 2.1 se puede ver los distintos encabezados utilizados, sus correspondientes significados y la cantidad de datos.

Variable	Encabezado	Número de datos
Peso de porción de alimento	01	3
Cantidad de entregas de alimento/ día	02	1
Hora de entrega del primer alimento diario	03	2
Minuto de entrega del primer alimento diario	04	2
Hora de entrega del segundo alimento diario	05	2
Minuto de entrega del segundo alimento diario	06	2
Hora de entrega del tercer alimento diario	07	2
Minuto de entrega del tercer alimento diario	08	2
Alimentar	09	0
Dar agua	10	0
Abrir/cerrar puerta	11	0
Resetear dispositivo principal	12	0
Restablecer a fábrica dispositivo principal	13	0
Escanear redes WiFi	14	0
Nombre de red a conectar (SSID)	15	largo variable
Contraseña de red a conectar	16	largo variable
Estado de conexión	17	4
Hora/Minutos/Día/Mes	18	8

Tabla 2.1 – *Protocolo empleado en la comunicación con el usuario.*

2.3.3. Sistema expendedor de alimento

Motor paso a paso

Los motores paso a paso son motores eléctricos ideales para la construcción de mecanismos en donde se requieren movimientos muy precisos. La característica principal de estos motores es el hecho de poder moverlos un paso a la vez por cada pulso que se le aplique. Este paso puede variar desde 90 grados hasta pequeños movimientos de tan solo 1.8 grados, es decir, que se necesitarán 4 pasos en el primer caso (90 grados) y 200 para el segundo caso (1.8 grados), para completar un giro completo. Los motores paso a paso están constituidos normalmente por un rotor sobre el que van aplicados distintos imanes permanentes y por un cierto número de bobinas excitadoras bobinadas en su estator. Las bobinas son parte del estator y el rotor es un imán permanente. La excitación de las bobinas deber ser manejada externamente por un controlador.

Existen dos tipos de motores paso a paso de imán permanente, los unipolares y los bipolares:

Bipolar: Estos generalmente están compuestos por dos bobinas y se los logra identificar ya que tienen cuatro cables. Su manejo no es tan sencillo, debido a que requieren del cambio de dirección del flujo de corriente a través de las bobinas en la secuencia apropiada para realizar un movimiento. Para controlar un motor Paso a Paso bipolar se necesita dos puentes H.

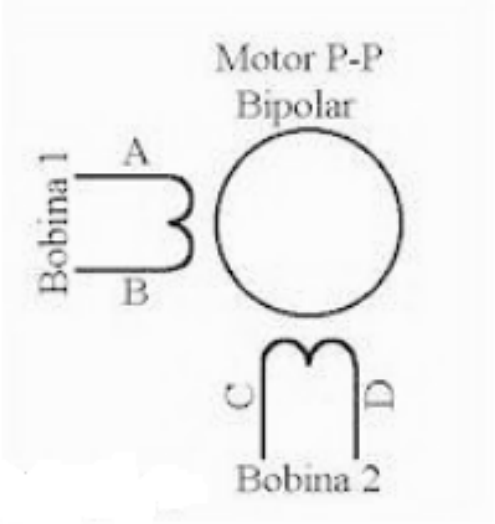


Figura 2.18 – *Bobinas de motor bipolar*

Como se dijo anteriormente, estos motores necesitan la inversión de la corriente que circula en sus bobinas en una secuencia determinada. Cada inversión de la polaridad provoca el movimiento del eje en un paso, cuyo sentido de giro está determinado por la secuencia seguida. A continuación se puede ver la tabla con la secuencia necesaria para controlar motores paso a paso del tipo Bipolares:

PASO	TERMINALES			
	A	B	C	D
1	+V	-V	+V	-V
2	+V	-V	-V	+V
3	-V	+V	-V	+V
4	-V	+V	+V	-V

Figura 2.19 – *Secuencia de pulsos para controlar motor bipolar.*

Unipolar: Los motores paso a paso unipolares, básicamente, se componen de dos bobinas, cada una con una derivación en el centro. Las derivaciones del centro son llevadas fuera del motor como dos cables separados o conectados entre sí internamente y llevados fuera del motor como uno de los cables. Como resultado, los motores unipolares tienen 5 o 6 cables. Independientemente del número de cables, los motores unipolares son manejados de la misma manera.

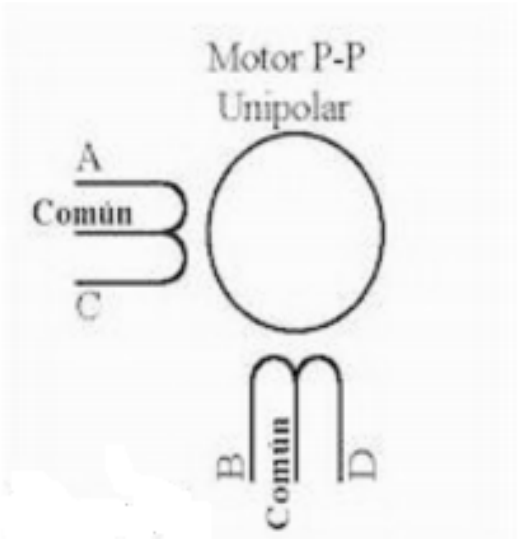


Figura 2.20 – *Bobinas de motor unipolar*

Existen tres secuencias posibles para este tipo de motores: secuencia normal, secuencia del tipo wave drive, secuencia medio paso. Todas las secuencias comienzan nuevamente por el paso 1 una vez alcanzado el paso final (4 u 8). Para revertir el sentido de giro, simplemente se deben ejecutar las secuencias en modo inverso.

- Normal: Esta es la secuencia más usada y la que generalmente recomienda el fabricante. Con esta secuencia el motor avanza un paso por vez y debido a que siempre hay al menos dos bobinas activadas, se obtiene un alto torque de paso y de retención.
- Wave drive: En esta secuencia se activa solo una bobina a la vez. En algunos motores esto brinda un funcionamiento más suave. Pero al estar solo una bobina activada, el torque de paso y retención es menor.
- Medio paso: En esta secuencia se activan las bobinas de tal forma de brindar un movimiento igual a la mitad del paso real. Para ello se activan primero dos bobinas y luego solo una y así sucesivamente.

Cabe destacar que debido a que los motores paso a paso son dispositivos mecánicos y como tal deben vencer ciertas inercias, el tiempo de duración y la frecuencia de los pulsos aplicados es un punto muy importante a tener en cuenta. En tal sentido el motor debe alcanzar el paso antes que la próxima secuencia de pulsos comience. Si la frecuencia de pulsos es muy elevada, el motor puede reaccionar en alguna de las siguientes formas:

- No realizar ningún movimiento.
- Solo vibrar.
- Girar erraticamente.
- Girar en sentido opuesto.

El motor utilizado para construir el prototipo fue un motor paso a paso bipolar de 42 V extraído de una impresora tradicional de hogar. Este da el torque necesario para hacer girar el sin fin, y ademas tiene un tamaño apropiado para instalarlo en el sistema previsto.

Controlador L297A

Como se menciono en el apartado anterior, para controlar un motor paso a paso bipolar es necesario utilizar dos drivers del tipo puente H, para esto se necesita manejar cuatro señales pulsadas diferentes que serán las que exciten las bases de los transistores de dichos puentes, ademas se debe generar la secuencia correcta para lograr que el motor gire adecuadamente. Generar estas cuatro señales implicaría usar cuatro salidas del microcontrolador ademas del software necesario. Es por esto que se decidió utilizar un circuito integrado diseñado especialmente para el manejo de motores paso a paso, el controlador L297A. Este integra toda la circuiteria de control necesaria para controlar motores paso a paso bipolares y unipolares. Usado con un driver de puente H doble como el L298N forma una completa interfaz entre un microprocesador y un motor paso a paso bipolar. Está principalmente destinado para ser utilizado con un puente L298N o L293N.

El L297A es capaz de generar 4 u 8 códigos, según el modo de funcionamiento que se preselecciones (paso entero, con 1 o 2 fases activas, o medio paso). Además dispone de un circuito de chopeado que realiza el control de la corriente que fluye por el motor al funcionar, cuya frecuencia de corte podrá ser definida por el usuario. Internamente y se pueden identificar los siguientes bloques:

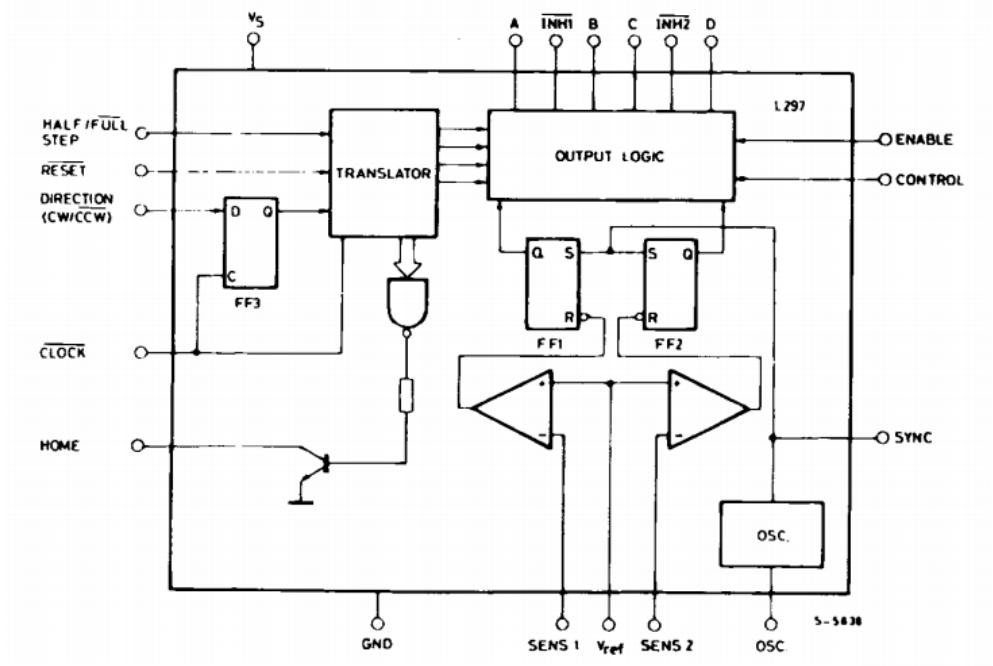


Figura 2.21 – *Diagrama en bloques internos L297A*

Este integrado sólo necesita que le proporcionemos, además de la alimentación, una señal de reloj con la que enviará los códigos al puente en H , el sentido de giro y las distintas señales de control (inicialización, habilitación, etc).

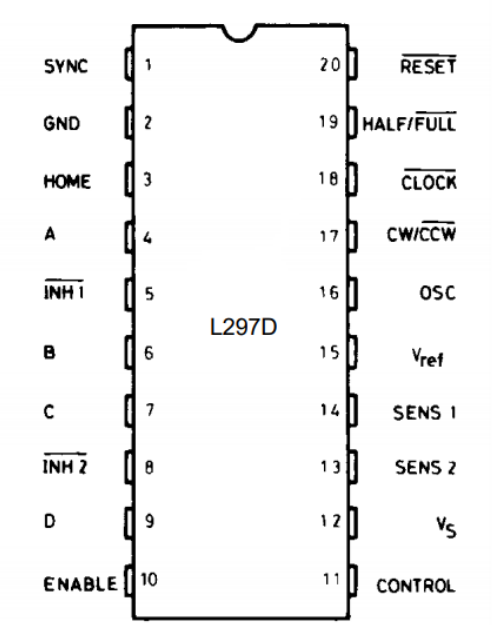


Figura 2.22 – *Integrado L297*

Puente H

Para la etapa de potencia del driver de control del motor se usó el integrado L298, se trata de un doble puente completo para controlar alto voltaje y alta corriente (48 V y hasta 2Amp) diseñado para aceptar estándares niveles lógicos TTL y manejar cargas inductivas tales como relés, solenoides, motores paso a paso y de corriente continua. Cuenta con dos entradas de habilitación para activar o desactivar el dispositivo de forma independiente de las señales de entrada.

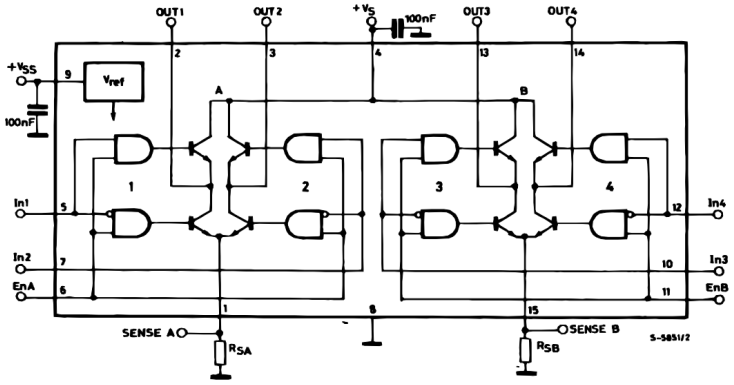


Figura 2.23 – *Componentes internos*

Es este integrado el verdadero controlador del motor, ya que recibe del L297 todas las secuencias lógicas para manejar los 8 transistores de potencia situados en su interior. Estos transistores tienen como misión alimentar a las bobinas del motor.

Los emisores de los transistores inferiores de cada puente están conectados entre sí y el terminal externo correspondiente se puede utilizar para la conexión de una resistencia externa de sensado de corriente para realizar el control de corriente que pasa por las bobinas del motor. Si la corriente absorbida por las bobinas sobrepasa el valor fijado el L297 inmediatamente la corriente de salida para evitar daños tanto en el puente como en el motor.

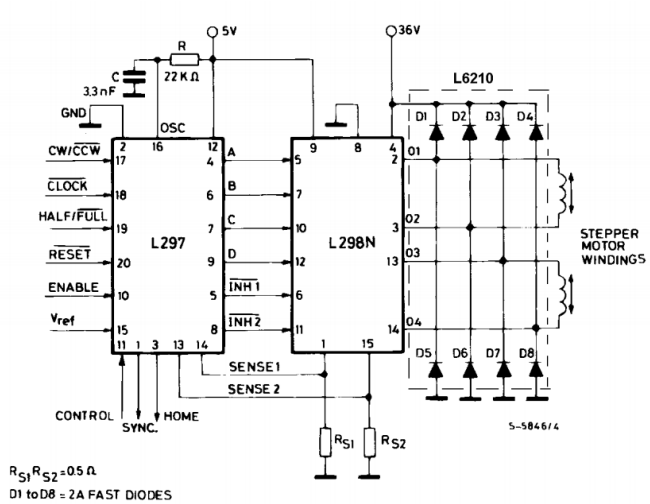


Figura 2.24 – *Conexionado con puente H*

Los diodos conectados a las salidas de los puentes, tanto sobre el positivo como sobre la masa, sirven para proteger al circuito integrado de tensiones peligrosas que se generan en las bobinas propias de la conmutación. Es por esto que es importante que estos diodos sean rápidos.

Clock de velocidad de pasos del motor

Como se dijo en la sección que habla del controlador L297A, este necesita de una señal de clock cuya frecuencia establece la velocidad con la que se conmutan las secuencias de excitación de las bobinas del motor, es decir que determina la velocidad de giro del eje. Para no tener que utilizar una salida ni recursos del microcontrolador para conformar esta señal, se la obtuvo mediante la utilización del popular integrado 555N en su configuración como monoestable de ancho de pulso fijo.

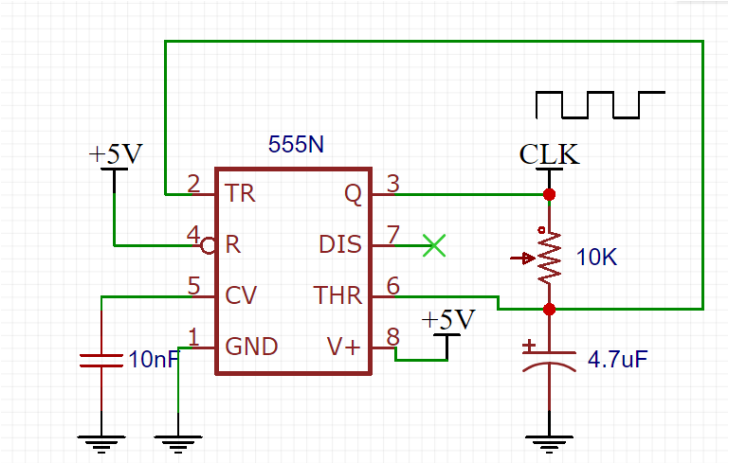


Figura 2.25 – *555N en configuracion monoestable*

Se puede observar en el diagrama de conexionado anterior, se coloco un preset para poder variar la frecuencia de la señal de salida, y así poder ajustarla a la frecuencia que de la velocidad y torque al motor más convenientes para hacer girar el tornillo sinfin. Luego de varias pruebas con el sistema completo se determinó que el funcionamiento es óptimo ajustando la señal de clock a una frecuencia de 300 Hz.

Sistema anti atascamiento

2.3.4. Sensado de cantidades de alimento

Celdas de carga

Una celda de carga es un transductor utilizado para convertir una fuerza en una señal eléctrica. Esta conversión empieza a partir de un dispositivo mecánico, es decir, la fuerza que se desea medir, deforma la galga extensiométrica. Y por medio de medidores de deformación (galgas) obtenemos una señal eléctrica con la cual podemos obtener el valor de la fuerza.

La medición se realiza con pequeños patrones de resistencias que son usados como indicadores de tensión con eficiencia, a los cuales llamamos medidores. Cuando se deforma el medidor de deformación la resistencia eléctrica cambia en proporción a la carga. Esto se logra por medio de un puente Wheastone, el cual se utiliza para medir resistencias desconocidas mediante el equilibrio de "brazos" del puente. Estos están construidos por cuatro resistencias que forman un circuito cerrado. En el caso de las celdas de carga las resistencias son los medidores de deformación.

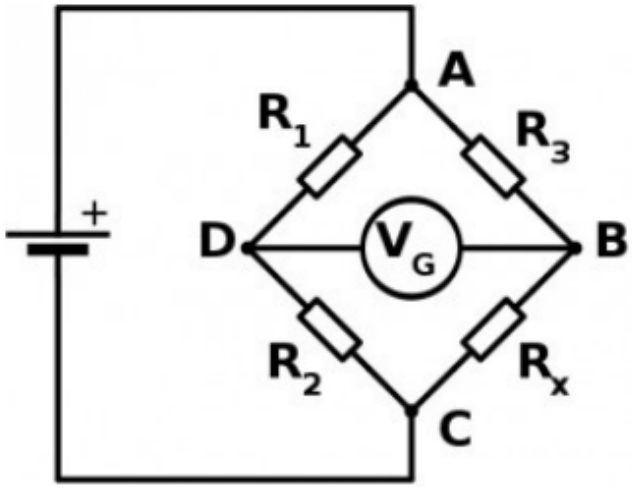


Figura 2.26 – *Puente de Wheastone*

Existen muchos tipos de celdas de cargas, en este proyecto se utilizaron dos celdas del tipo de un solo punto. Estas celdas de un solo punto se utilizan en pequeñas escalas, como joyas, o balanzas de cocina, existen celdas de máximo 100g hasta celdas de máximo de 50kg. Esta se monta por medio de

pernos hacia abajo en cada extremo de la celda de carga, donde los cables se unen, y la aplicación de la fuerza debe ser en el sentido de la flecha lateral. Donde se aplica la fuerza, no es una zona crítica, ya que esta celda de carga mide un efecto de elasticidad sobre la viga, no la flexión de la viga. De tal manera que si se monta una pequeña plataforma en la celda de carga, como se haría en una pequeña escala, esta celda proporcionaría lecturas precisas, independientemente de la posición de la carga en la plataforma.

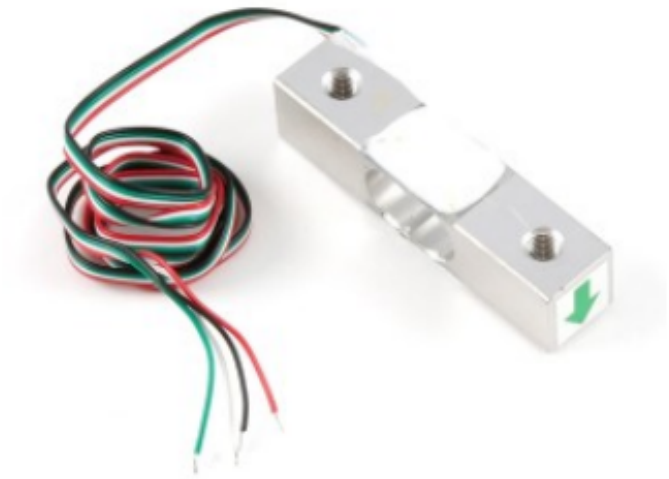


Figura 2.27 – *Tipica celda de carga de un solo punto*

Como se dijo, para la elaboración del prototipo del este proyecto se usaron des celdas de carga de un solo punto, una para medir la porción de alimento a entregar, y otra para medir la cantidad de alimento almacenado en tolva. Para la primera se eligió una de carga máxima de 5 kilo gramos, ya que el dispositivo admite una porción máxima de un kg y se deja un margen para las posibles compresiones del animal. Esta celda entrega 1mV/V, lo que significa que si se la alimenta por ejemplo con 10 V, a carga máxima tendremos en su salida 10 mV.

La otra celda de carga utilizada es de 30 kg de peso máximo, se utilizo esta ya que es un valor tipico de las utilizadas en balanzas comerciales, por lo que su costo es bajo, ademas pensando en que la tolva pueda tener una capacidad de 10 kg de alimento, y tenga que soportar una compresión constante e ininterrumpido es mejor sobredimensionarla para evitar deformaciones permanentes. La misma es de 2 mV/V, lo que a máxima carga se tendrá una tension máxima de 20 mV.

Amplificación de señales

Dado que las señales que entregan las celdas de cargas son del orden de los pocos mV es necesario amplificarlas. Para amplificar este tipo de señales es conveniente usar un amplificador de instrumentación. Estos amplificadores permiten elaborar circuitos de acondicionamiento de señal, para minimizar los efectos del ruido que pueden provocar errores de medición. Algunas de sus aplicaciones son:

- Para acondicionar la salida de un puente de Wheatstone.
- Para amplificar señales eléctricas biológicas (por ejemplo en electrocardiogramas).
- Como parte de circuitos para proporcionar alimentación a corriente constante.
- En fuentes de alimentación.
- En aplicaciones en las que se requiere gran precisión y estabilidad a corto y largo plazo.

El amplificador de instrumentación es un amplificador operacional en modo diferencial con ganancia controlada. La ventaja de este arreglo se resume en un alta impedancia de entrada y un control de ganancia simplificado. La desventaja de otras configuraciones, como la del amplificador diferencial, es que la impedancia depende de una de las resistencias de la entrada. La impedancia es la del amplificador operacional, que en un caso ideal tiende a ser infinita. En la etapa de ganancia, en el modo diferencial se tiene que mantener la relación entre las diferentes resistencias. En el de instrumentación solo depende de una resistencia.

Los amplificadores de instrumentación deben cumplir con los siguientes requisitos:

- Ganancia: seleccionable, estable, lineal.
- Entrada diferencial: con CMRR (Razón de rechazo en modo común) alto.
- Error despreciable debido a las corrientes y tensiones de offset.
- Impedancia de entrada alta.
- Impedancia de salida baja.

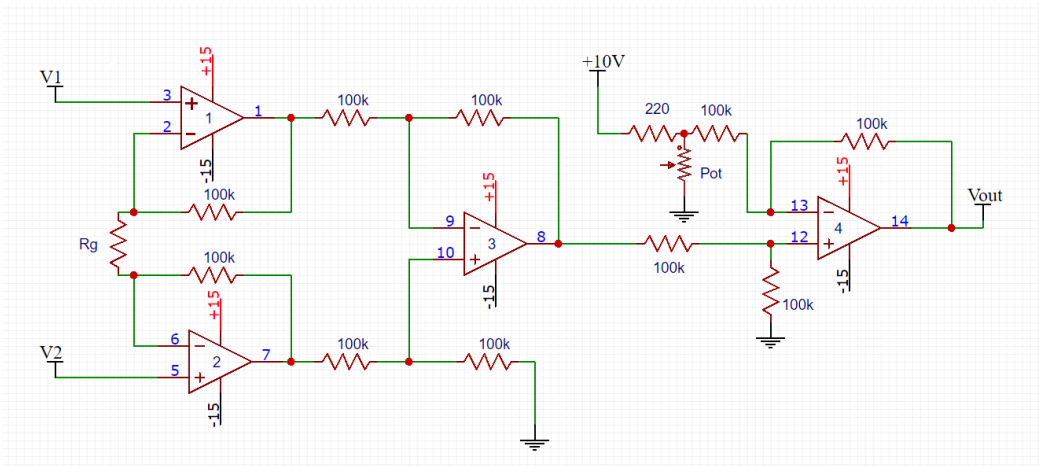


Figura 2.28 – *Amplificador de instrumentacion*

Donde:

$$V_{out} = (V_2 - V_1)(1 + \frac{2R}{R_g})$$

Dando una ganancia:

$$G = (1 + \frac{2R}{R_g})$$

En este proyecto se armó un amplificador de instrumentación utilizando amplificadores operacionales y resistencias de 100K. Para esto se utilizó el integrado TL074, el cual tiene en su interior 4 amplificadores operacionales.

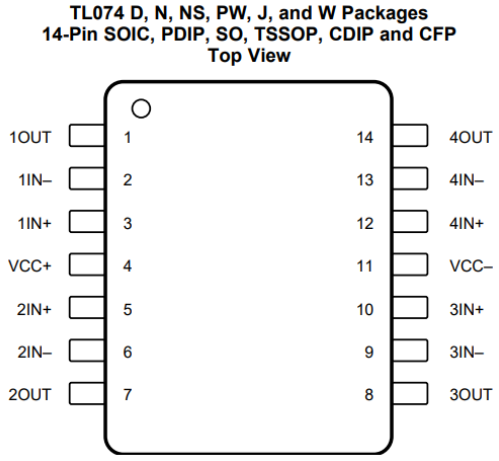


Figura 2.29 – *Integrado TL074*

Sabiendo que la entrada del conversor analógico-digital solo puede admitir señales de hasta 3 volts se realizan los siguientes cálculos:

Pesaje de alimento almacenado:

Estableciendo una carga máxima de 5 Kg de alimento en la tolva se desea una ganancia tal que a plena carga tengamos una tension de salida de 3 volts. Sabiendo que la celda de carga es de 2mV/V, que es de una capacidad máxima y que se la alimenta con 10 V podemos deducir que para 5Kg la salida será de 2,5 mV.

$40Kg - - - - - -20mV$
$5Kg - - - -2,5mV$

Por lo tanto se quiere una ganancia tal que:

$2,5mV.G = 3V$
$G = \frac{3}{2,5}, 1000$

Entonces:

G=1200

CalculandoR_g para obtener esta ganancia

$1200 = \frac{200K}{R_g}$
$R_g = \frac{200K}{1200}$
$R_g = 166,6ohm$

Se adopta una resistencia de 180 ohm.

Pesaje de porción:

Estableciendo una carga máxima de 500 g de alimento en el comedero se desea una ganancia tal que a plena carga tengamos una tension de salida de 3 volts. Sabiendo que la celda de carga es de 1 mV/V, que es de una capacidad máxima y que se la alimenta con 10 volts podemos deducir que para 5Kg la salida será de 1 mV.

$5Kg - - - - - - 10mV$

$500g - - - - 1mV$

Por lo tanto se quiere una ganancia tal que:

$1mV.G = 3V$

$G = \frac{3}{1}, 1000$

Entonces:

G=3000

CalculandoR_g para obtener esta ganancia

$$3000 = \frac{200K}{R_g}$$

$$R_g = \frac{200K}{3000}$$

$$R_g = 66,6ohm$$

Se adopta una resistencia de 68 ohm.

Multiplexado de señales analógicas

El módulo ESP8266, como ya se dijo, tiene solo un conversor analógico digital, y aquí es necesario poder leer dos señales, la de cada celda de carga. Para solucionar esto se utilizo un multiplexor analógico que permita elegir cual de las dos señales deseamos leer y así introducirla al conversor. El multiplexor utilizado es un HEF4052B, de tres canales que pueden ser multiplexados en cuatro diferentes salidas cada uno. Ademas es fácil de conseguir en el mercado y a muy bajo costo.

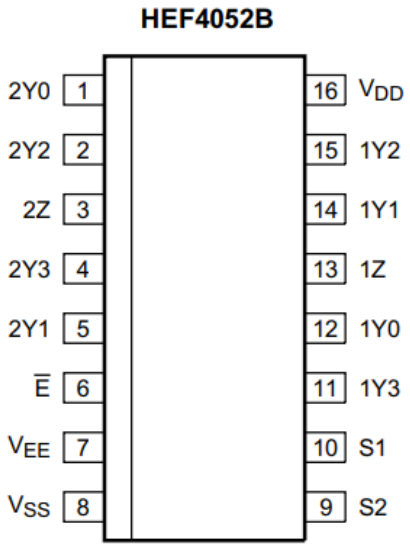


Figura 2.30 – *Multiplexor analogico HEF4052B*

El bit más significativo de los bits de control se conectó directamente a masa y el menos significativo es controlado por una de las salidas del micro, pudiendo así, decidir mediante software cual de las dos señales leer en cada determinado momento. Cuando este bit de selección se pone en estado bajo se toma la señal del sensor de la tolva de almacenaje y cuando toma el valor de 1 toma la señal del pesaje del alimento en el comedero.

2.3.5. Hardware adicional

Fuente de alimentación principal

El dispositivo cuenta con una entrada de alimentación de 220 Vac. Dicha tension es utilizada para las salidas auxiliares de alta tension y para alimentar los diferentes circuitos. Para esto ultimo se bajo la tension con un transformador 12+12, configurando una clásica fuente partida con salidas +15 y -15, siendo estas las tensiones necesarias para alimentar los amplificadores operacionales de la etapa de amplificación de las señales de las celdas de carga. Luego, a partir de lo 15 V se obtuvieron los 5 V que alimentan la lógica.

Cabe destacar que se pusieron componentes de protección como fusible para proteger por una sobre corriente y un varistor como protección contra sobretensiones.

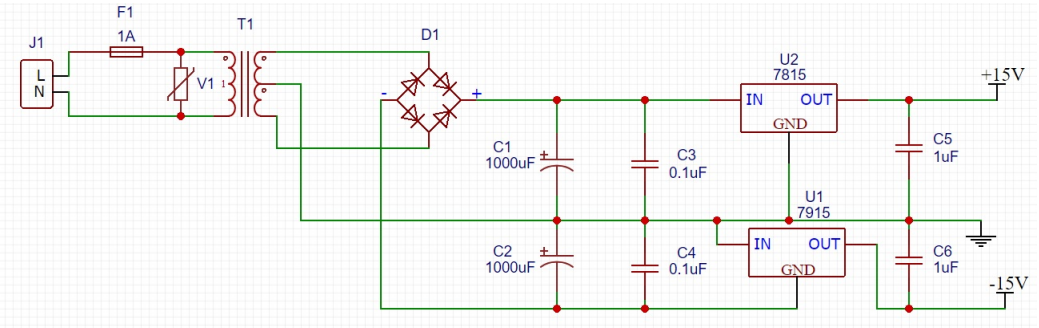


Figura 2.31 – *Circuito fuente de alimentación*

Salidas auxiliares y periféricos

Otra función importante del dispositivo Wifeed es poder mantener el bebedero del animal con agua, para lograr esto se coloco una electroválvula, del tipo que se utilizan en los lavarropas. Se eligió este tipo por su simplicidad y muy bajo costo. Al aplicar una tension de 220 V de alterna entre sus bornes permite el paso del flujo del agua, al extinguir dicha tension de sus bornes el paso del agua se interrumpe. Esta electroválvula da la posibilidad de con una manguera con rosca se la conecte directamente a una canilla o a algún dispenser de agua o botellon. El control de la válvula la tiene el usuario que como ya se dijo la puede abrir desde cualquiera de las interfases de control. La tension a la válvula se la da un rele de 12 V que es controlado con una salida del micro. Dado que la salida del microcontrolador no tiene la capacidad de suministra ni la tension ni la corriente para excitar el rele, se utilizo un buffer ULN2004 el cual suministra los 12 V y corriente necesaria.

Con este integrado también se controla los otros tres reles que dan alta tension a las tres salidas auxiliares.

Energía de emergencia

Es importante en este dispositivo que no se pierda nunca la fecha y hora. Ante un corte de luz, dependiendo del modo en el que se esté usando equipo, esta información puede perderse, es por eso que es importante que la el microcontrolador siempre este alimentado. Para esto se utilizo un circuito que si se corta el suministro de alta tension el microcontrolador pasa a ser alimentado por una batería.

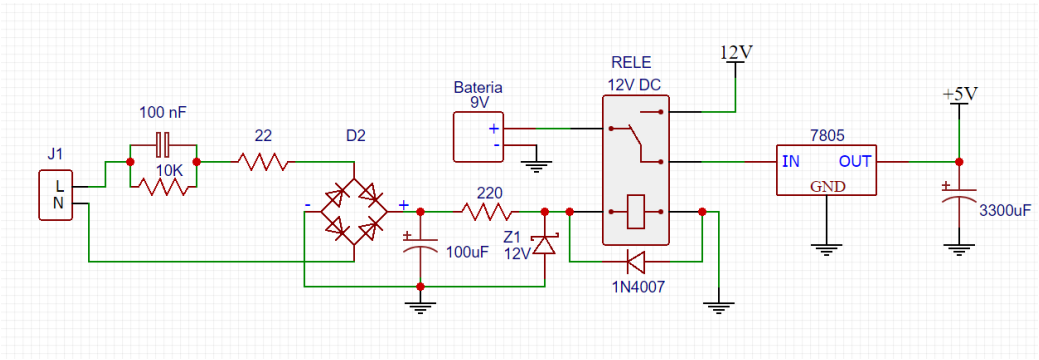


Figura 2.32 – *Circuito fuente de alimentación*

Consiste en un sistema conformado por un relé con un contacto normal abierto y otro normal cerrado. El contacto denominado como común de este relé es el que alimentara un regulador de 5 V. Cuando hay alta tension, es decir suministro de energía eléctrica este relé es excitado, conectando el común con el contacto normal abierto, el cual está conectado a la fuente de 12V generados por la fuente principal del sistema, quedando la batería de emergencia en un circuito abierto, sin ninguna perdida ni consumo. Cuando hay un corte de luz, el relé deja de ser excitado y toma el estado normal cerrado, siendo la batería de 9 V la que comienza a alimentar la lógica de control. Dado a que la transición de un estado al otro puede generar una discontinuidad en la tension de alimentación y provoque que el microcontrolador se reinicie se colocó un capacitor electrolítico, este se encarga mantener constante la tension en dicha transición.

Como se puede observar en la imagen anterior, el relé es excitado por una fuente capacitiva de 12 V, este tipo de fuente se caracteriza por no necesitar un transformador que baje la alta tension, por lo que es una fuente de bajo costo y volumen. Como desventaja tiene que puede suministrar corrientes de no más de 150 mA. Se usó esta fuente para este propósito especifico de controlar este relé porque si se lo excitara directamente con la fuente principal, ante un corte de luz la alta capacidad de la etapa de rectificación y filtrado hace que la transición sea muy lenta y es inevitable el reinicio del micro. Otra solución hubiera sido utilizar directamente un relé de 220 V, pero no se opto por esta opción dado que son difíciles de conseguir y de alto costo.

2.3.6. Diseño 3D y prototipo

Previo a la construcción del prototipo del dispositivo se lo diseñó en el software SketchUp provisto por Google.

Figura 2.33 – *Diseño de dispoitivo Wifeed*

El diseño fue desarrollado a escala 1:1, tomándose como punto de partido el tubo que contiene el tornillo sin fin, ya que el caño utilizado fue un ramal 45 grados de PVC de 60 milímetros de diámetro comúnmente utilizado en obras sanitarias domiciliarias. Otra referencia que se tomó para definir las dimensiones finales fue la tolva de almacenaje, la cual en el prototipo fue realizada con bidón de agua de 10 litros, que da una capacidad de aproximadamente de 5 Kg de alimento balanceado.

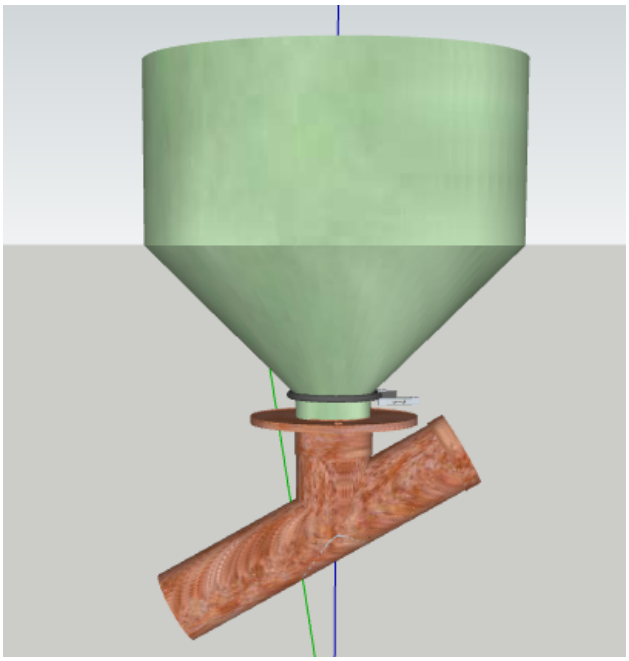


Figura 2.34 – *Vista de tolva y tubo expendedor*

El sin fin expendedor fue construido completamente en PVC. Existen cálculos precisos para la construcción de tornillos de Arquímedes, en los cuales se debe fijar el diámetro exterior deseado, la separación entre hélices y el largo total del sin fin, utilizando unas determinadas formulas matemáticas se pueden calcular las diferentes dimensiones de cada hélice para luego colocarlas en un eje y obtener las características deseadas. Como eje se utilizó un tubo de PVC de 20 mm de diámetro. Este último se agarra al eje del motor paso a paso que lo hara girar.

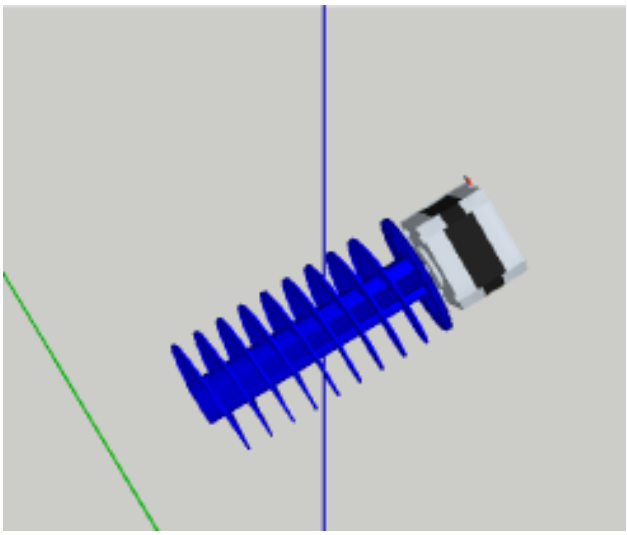


Figura 2.35 – *Vista de tolva y tubo expendedor*

El interior del artefacto esta compuesto por dos habitáculos separados por una división con un orificio de 7 centímetros de diámetro del cual se sujeta el caño del sin fin y donde se apoya la celda de carga que pesa la tolva de almacenaje. Quedando en la parte superior el espacio para la tolva, y en la inferior se encuentra el mecanismo expendedor de alimento y las placas electrónicas.

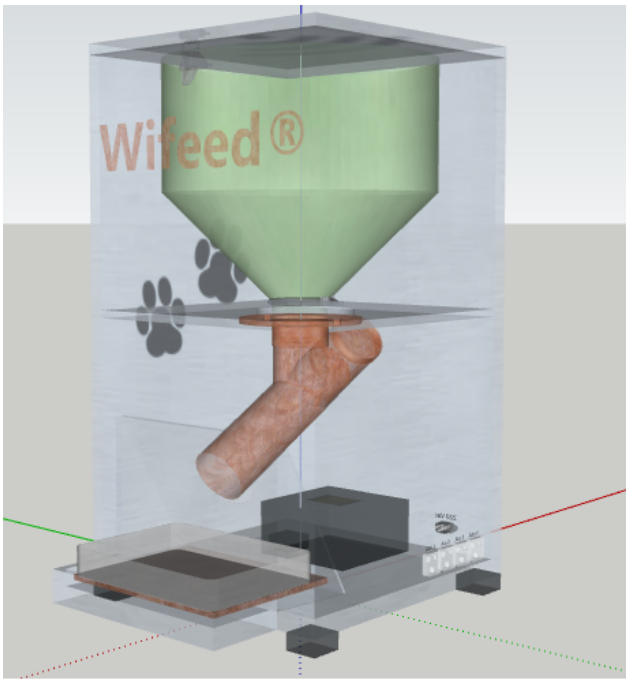


Figura 2.36 – *Vista de tolva y tubo expendedor*

En las Figuras 2.37 a 2.40 puede verse el prototipo final implementado. Puede verse el exterior (Figuras 2.37 y 2.38), imagen del interior y parte electrónica (Figuras 2.39 y 2.40).

Figura 2.37 – *Prototipo final.*

Figura 2.38 – *Prototipo final.*

Figura 2.39 – *Interior del dispositivo.*

Figura 2.40 – *Placa electrónica del dispositivo.*

Capítulo 3
Interfaz Movil

La interfaz móvil es una de las dos opciones que tiene el usuario para interactuar con el dispositivo, pudiendo monitorear el estado de la unidad central y los eventos más recientes, con la posibilidad de recibir notificaciones para diferentes alertas, realizar la configuración inicial, programar las funciones automáticas y ejecutar acciones en tiempo real desde cualquier parte del mundo.

Dicha interfaz se llevó a cabo mediante el desarrollo de una aplicación móvil para la plataforma Android, siendo éste un sistema operativo muy difundido, tanto en la telefonía móvil como en dispositivos tablet. La aplicación se programó bajo el entorno Basic4Android [7]. El cual es un entorno comercial para desarrollar aplicaciones para Android programando en un lenguaje muy similar a Visual Basic, sin embargo, está basado en Java, es decir, el compilador traduce el código al lenguaje Java.

La aplicación que se lleva a cabo en este trabajo recibe el nombre de *WifeedApp*.

3.1. Sistema operativo Android

El sistema Android está pensado especialmente para dispositivos móviles y que incluye tanto un sistema operativo (S.O.), como un middleware (software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos) y diversas aplicaciones de usuario. Este S.O. representa la primera incursión seria de Google en el mercado móvil y nace con la pretensión de extender su participación en dicho sector.

La gran mayoría de las aplicaciones para Android se programan en

lenguaje Java modificado y son ejecutadas por una máquina virtual especialmente diseñada para esta plataforma, que ha sido bautizada con el nombre de Dalvik [8]. El núcleo de Android está basado en Linux 2.6 y se distribuye bajo la licencia Apache 2.0, lo que lo convierte en software de libre distribución. A los desarrolladores se les proporciona de forma gratuita un set de desarrollo de software (Software Development Kit, SDK) y la opción de un complemento (plug-in) para el entorno de desarrollo Eclipse que incluye todas las interfaces de programación de aplicaciones necesarias (Aplication Programming Interface, API). Existe además disponible una amplia documentación de respaldo para este SDK.

El proyecto Android está liderado por Google y un conglomerado de otras empresas tecnológicas agrupadas bajo el nombre de Open Handset Alliance (OHA). El objetivo principal de esta alianza empresarial (que incluye a fabricantes de dispositivos y operadores telefónicos, con armas tan relevantes como Samsung, LG, Telefónica, Intel o Texas Instruments, entre otras muchas) es el desarrollo de estándares abiertos para la telefonía móvil como medida para incentivar su desarrollo y para mejorar la experiencia del usuario. La plataforma Android constituye su primera contribución en este sentido. Cuando en noviembre de 2007 Google anunció su irrupción en el mundo de la telefonía móvil a través de Android, muchos medios especializados catalogaron este novedoso producto como un nuevo sistema operativo, libre y especifico para teléfonos móviles. Sin embargo, los responsables del proyecto se han esforzado desde entonces en destacar que la motivación de Android es convertirse en algo más que un simple sistema operativo.

Con Android se busca reunir en una misma plataforma todos los elementos necesarios que permitan al desarrollador controlar y aprovechar al máximo cualquier funcionalidad ofrecida por un dispositivo móvil, así como poder crear aplicaciones que sean verdaderamente portables, reutilizables y de rápido desarrollo. En otras palabras, Android quiere mejorar y estandarizar el desarrollo de aplicaciones para cualquier dispositivo móvil y, por ende, acabar con la perjudicial fragmentación existente hoy día.

Mejorar el desarrollo y enriquecer la experiencia del usuario se convierte, por tanto, en la gran filosofía de Android y en su principal objetivo.

3.1.1. Arquitectura Android

Como ya se ha mencionado, Android es una plataforma para dispositivos móviles que contiene una pila de software donde se incluye un sistema

operativo, middleware y aplicaciones básicas para el usuario.

En las siguientes líneas se dará una visión global por capas de cuál es la arquitectura empleada en Android 3.1. Cada una de estas capas utiliza servicios ofrecidos por las anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores, tal como muestra la siguiente figura

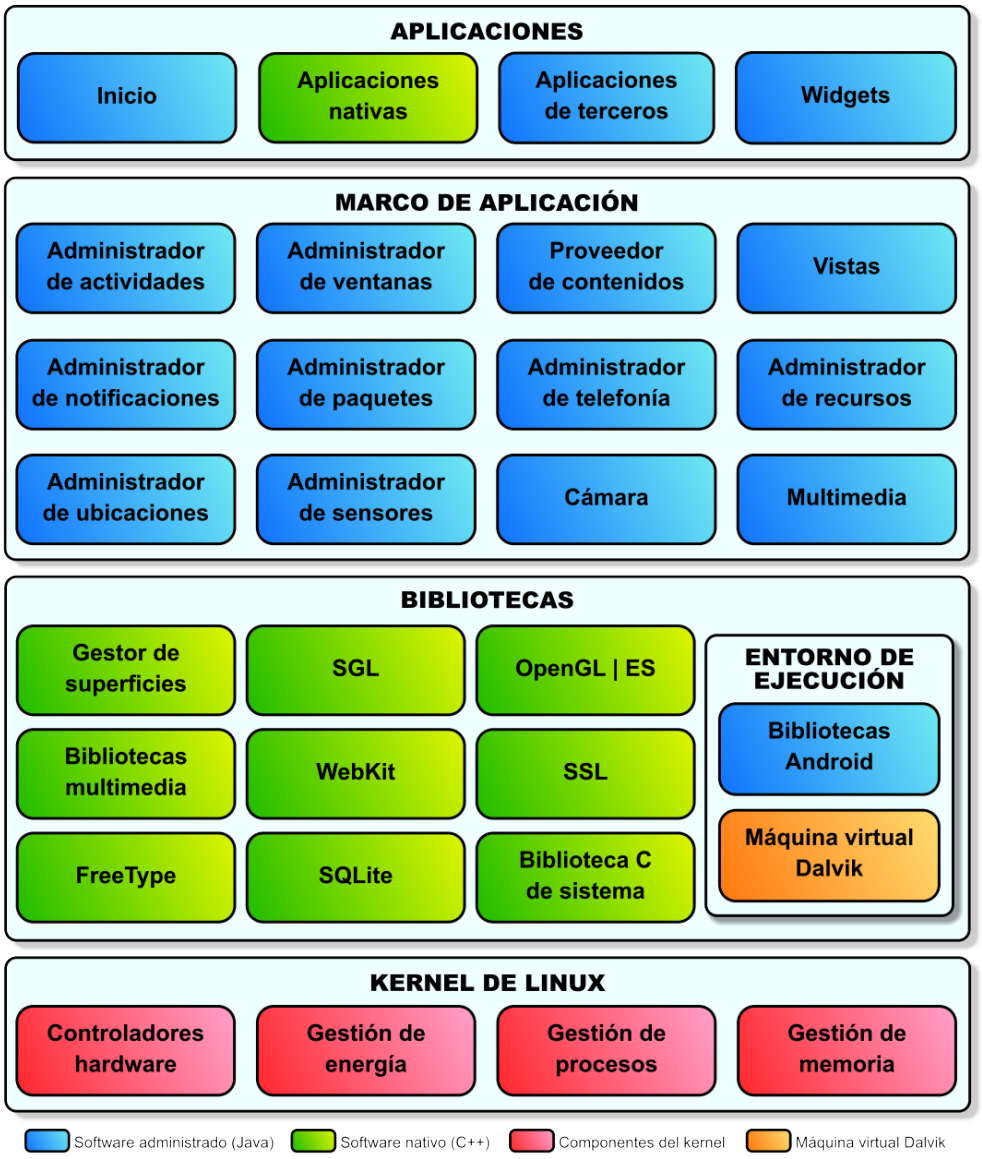


Figura 3.1 – *Arquitectura Android.*

Aplicaciones

Este nivel contiene, tanto las incluidas por defecto de Android como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo. Todas estas aplicaciones utilizan los servicios, las API y librerías de los niveles anteriores.

Framework de Aplicaciones

Representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación que se desarrolle para Android, ya sean las propias del dispositivo, las desarrolladas por Google o terceras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de API y el mismo "framework", representado por este nivel.

Entre las API más importantes ubicadas aquí, se pueden encontrar las siguientes:

- **Activity Manager:** Conjunto de API que gestiona el ciclo de vida de las aplicaciones en Android.
- **Window Manager:** Gestiona las ventanas de las aplicaciones y utiliza la librería Surface Manager.
- **Telephone Manager:** Incluye todas las API vinculadas a las funcionalidades propias del teléfono (llamadas, mensajes, etc.).
- **Content Provider:** Permite a cualquier aplicación compartir sus datos con las demás aplicaciones de Android. Por ejemplo, gracias a esta API la información de contactos, agenda, mensajes, etc. será accesible para otras aplicaciones, como puede ser el WhatsApp.
- **View System:** Proporciona un gran número de elementos para poder construir interfaces de usuario (GUI), como listas, mosaicos, botones, çheck-boxes", tamaño de ventanas, control de las interfaces mediante teclado, etc. Incluye también algunas vistas estándar para las funcionalidades más frecuentes.
- **Location Manager:** Posibilita a las aplicaciones la obtención de información de localización y posicionamiento.
- **Notification Manager:** Mediante el cual las aplicaciones, usando un mismo formato, comunican al usuario eventos que ocurran durante su ejecución: una llamada entrante, un mensaje recibido, conexión Wi-Fi disponible, ubicación en un punto determinado, etc. Si llevan asociada

alguna acción, en Android denominada Intent, (por ejemplo, atender una llamada recibida) ésta se activa mediante un simple clic.

- **XMPP Service:** XMPP Service: Colección de API para utilizar este protocolo de intercambio de mensajes basado en XML.

Librerías

La siguiente capa se corresponde con las librerías utilizadas por Android. Éstas han sido escritas utilizando C/C++ y proporcionan a Android la mayor parte de sus capacidades más características. Junto al núcleo basado en Linux, estas librerías constituyen el corazón de Android.

Entre las librerías más importantes ubicadas aquí, se pueden encontrar las siguientes:

- **Librería libc:** Incluye todas las cabeceras y funciones según el estándar del lenguaje C. Todas las demás librerías se definen en este lenguaje.
- **Librería Surface Manager:** Es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.
- **OpenGL/SL y SGL:**Representan las librerías gráficas y, por tanto, sustentan la capacidad gráfica de Android. OpenGL/SL maneja gráficos en 3D y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware encargado de proporcionar gráficos 3D. Por otro lado, SGL proporciona gráficos en 2D, por lo que será la librería más habitualmente utilizada por la mayoría de las aplicaciones. Una característica importante de la capacidad gráfica de Android es que es posible desarrollar aplicaciones que combinen gráficos en 3D y 2D.
- **Librería Media Libraries:**Proporciona todos los códecs necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc.)
- **FreeType:** Permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.
- **Librería SSL:** Posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.
- **Librería SQLite:** Creación y gestión de bases de datos relacionales.

- **Librería WebKit:** Proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incluido por defecto en la plataforma Android.

Núcleo Linux

Android utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. Siempre que un fabricante incluye un nuevo elemento de hardware, lo primero que se debe realizar para que pueda ser utilizado desde Android es crear las librerias de control o drivers necesarios dentro de este kernel de Linux embebido en el propio Android.

3.1.2. Componentes de una aplicación

Las aplicaciones en Android estan compuestas por cuatro componentes principales. Cada aplicación será una combinación de uno o más de estos componentes, que deberán ser declarados de forma explícita en un fichero con formato XML denominado AndroidManifest.xml, junto a otros datos asociados como valores globales, clases que implementa, datos que puede manejar, permisos, etc. Este fichero es básico en cualquier aplicación Android y permite al sistema desplegar y ejecutar correctamente la aplicación.

A continuación se exponen los cuatro tipos de componentes en los que puede dividirse una aplicación para Android.

Activity

Este es el componente más habitual de las aplicaciones para Android. Un componente Activity refleja una determinada actividad llevada a cabo por una aplicación, y que lleva asociada típicamente una ventana o interfaz de usuario; es importante señalar que no contempla únicamente el aspecto gráfico, sino que éste forma parte del componente Activity a través de vistas representadas por clases como View y sus derivadas. Este componente se implementa mediante la clase de mismo nombre Activity. La mayoría de las aplicaciones permiten la ejecución de varias acciones a través de la existencia de una o más pantallas.

Una actividad tiene un ciclo de vida muy definido, que será igual para todas las actividades. Este ciclo de vida es impuesto por el SDK de Android. Las actividades tienen cuatro posibles estados: Activa, pausada, parada y reiniciada.

Broadcast Intent Receiver

Un componente Broadcast Intent Receiver se utiliza para lanzar alguna ejecución dentro de la aplicación actual cuando un determinado evento se produzca (generalmente, abrir un componente Activity).

Por ejemplo, una llamada entrante o un SMS recibido. Este componente no tiene interfaz de usuario asociada, pero puede utilizar el API Notification Manager para avisar al usuario del evento producido a través de la barra de estado del dispositivo móvil. Este componente se implementa a través de una clase de nombre BroadcastReceiver. Para que Broadcast Intent Receiver funcione, no es necesario que la aplicación en cuestión sea la aplicación activa en el momento de producirse el evento. El sistema lanzará la aplicación si es necesario cuando el evento monitorizado tenga lugar.

Service

Un componente Service representa una aplicación ejecutada sin interfaz de usuario, y que generalmente tiene lugar en segundo plano mientras otras aplicaciones (éstas con interfaz) son las que están activas en la pantalla del dispositivo.

Un ejemplo típico de este componente es un reproductor de música. La interfaz del reproductor muestra al usuario las distintas canciones disponibles, así como los típicos botones de reproducción, pausa, volumen, etc. En el momento en el que el usuario reproduce una canción, ésta se escucha mientras se siguen visualizando todas las acciones anteriores, e incluso puede ejecutar una aplicación distinta sin que la música deje de sonar. La interfaz de usuario del reproductor sería un componente Activity, pero la música en reproducción sería un componente Service, porque se ejecuta en background. Este elemento está implementado por la clase de mismo nombre Service.

Content Provider

Con el componente Content Provider, cualquier aplicación en Android puede almacenar datos en un fichero, en una base de datos SQLite o en cualquier otro formato que considere. Además, estos datos pueden ser compartidos entre distintas aplicaciones. Una clase que implemente el componente Content Provider contendrá una serie de métodos que permiten almacenar, recuperar, actualizar y compartir los datos de una aplicación.

Existe una colección de clases para distintos tipos de gestión de datos en el paquete android.provider. Además, cualquier formato adicional que se quiera implementar deberá pertenecer a este paquete y seguir sus estándares de funcionamiento.

No todas las aplicaciones tienen que tener los cuatro componentes, pero cualquier aplicación será una combinación de estos.

3.1.3. Estado de los procesos

Como se ha mencionado anteriormente, cada aplicación de Android corre en su propio proceso, el cual es creado por la aplicación cuando se ejecuta y permanece hasta que la aplicación deja de trabajar o el sistema necesita memoria para otras aplicaciones. Android sitúa cada proceso en una jerarquía de ïmportancia"basada en estados, como se puede ver a continuación:

- **Procesos en primer plano (Active process):** Es un proceso que aloja una Activity en la pantalla y con la que el usuario está interactuando (su método onResume() ha sido llamado) o que un IntentReceiver está ejecutándose. Este tipo de procesos serán eliminados como último recurso si el sistema necesitase memoria.
- **Procesos visibles (Visible process):**Es un proceso que aloja una Activity pero no está en primer plano (su método onPause() ha sido llamado). Esto ocurre en situaciones dónde la aplicación muestra un cuadro de diálogo para interactuar con el usuario. Este tipo de procesos no será eliminado en caso que sea necesaria la memoria para mantener a todos los procesos del primer plano corriendo.
- **Procesos de servicio (Started service process):** Es un proceso que aloja un Service que ha sido iniciado con el método startService(). Este tipo de procesos no son visibles y suelen ser importantes para el usuario (conexión con servidores, reproducción de música).
- **Procesos en segundo plano (Background process):** Es un proceso que aloja una Activity que no es actualmente visible para el usuario (su método onStop() ha sido llamado). Normalmente la eliminación de estos procesos no suponen un gran impacto para la actividad del usuario. Es muy usual que existan numerosos procesos de este tipo en el sistema, por lo que el sistema mantiene una lista para asegurar que el último proceso visto por el usuario sea el último en eliminarse en caso de necesitar memoria.
- **Procesos vacíos (Empty process):** Es un proceso que no aloja ningún componente. La razón de existir de este proceso es tener una caché disponible de la aplicación para su próxima activación. Es común, que el sistema elimine este tipo de procesos con frecuencia para obtener memoria disponible.

Según esta jerarquía, Android prioriza los procesos existentes en el sistema y decide cuáles han de ser eliminados, con el fin de liberar recursos y poder lanzar la aplicación requerida.

Para los procesos en segundo plano, existe una lista llamada LRU (Least Recently Used). En función de esta lista se van eliminando los procesos; los primeros que se eliminan son aquellos que llevan más tiempo sin usarse. Así el sistema se asegura de mantener vivos los procesos que se han usado recientemente.

3.1.4. Ciclo de vida de una actividad

El hecho de que cada aplicación se ejecuta en su propio proceso aporta beneficios en cuestiones básicas como seguridad, gestión de memoria, o la ocupación de la CPU del dispositivo móvil. Android se ocupa de lanzar y parar todos estos procesos, gestionar su ejecución y decidir qué hacer en función de los recursos disponibles y de las órdenes dadas por el usuario.

El usuario desconoce este comportamiento de Android. Simplemente es consciente de que mediante un simple clic pasa de una a otra aplicación y puede volver a cualquiera de ellas en el momento que lo desee. No debe preocuparse sobre cuál es la aplicación que realmente está activa, cuánta memoria está consumiendo, ni si existen o no recursos suficientes para abrir una aplicación adicional. Todo eso son tareas propias del sistema operativo.

Android lanza tantos procesos como permitan los recursos del dispositivo. Cada proceso, correspondiente a una aplicación, estará formado por una o varias actividades independientes (componentes Activity) de esa aplicación. Cuando el usuario navega de una actividad a otra, o abre una nueva aplicación, el sistema duerme dicho proceso y realiza una copia de su estado para poder recuperarlo más tarde. El proceso y la actividad siguen existiendo en el sistema, pero están dormidos y su estado ha sido guardado. Es entonces cuando crea, o despierta si ya existe, el proceso para la aplicación que debe ser lanzada, asumiendo que existan recursos para ello. Cada uno de los componentes básicos de Android tiene un ciclo de vida bien definido; esto implica que el desarrollador puede controlar en cada momento en qué estado se encuentra dicho componente, pudiendo así programar las acciones que mejor convengan. El componente Activity, probablemente el más importante, tiene un ciclo de vida como el mostrado en la siguiente figura.

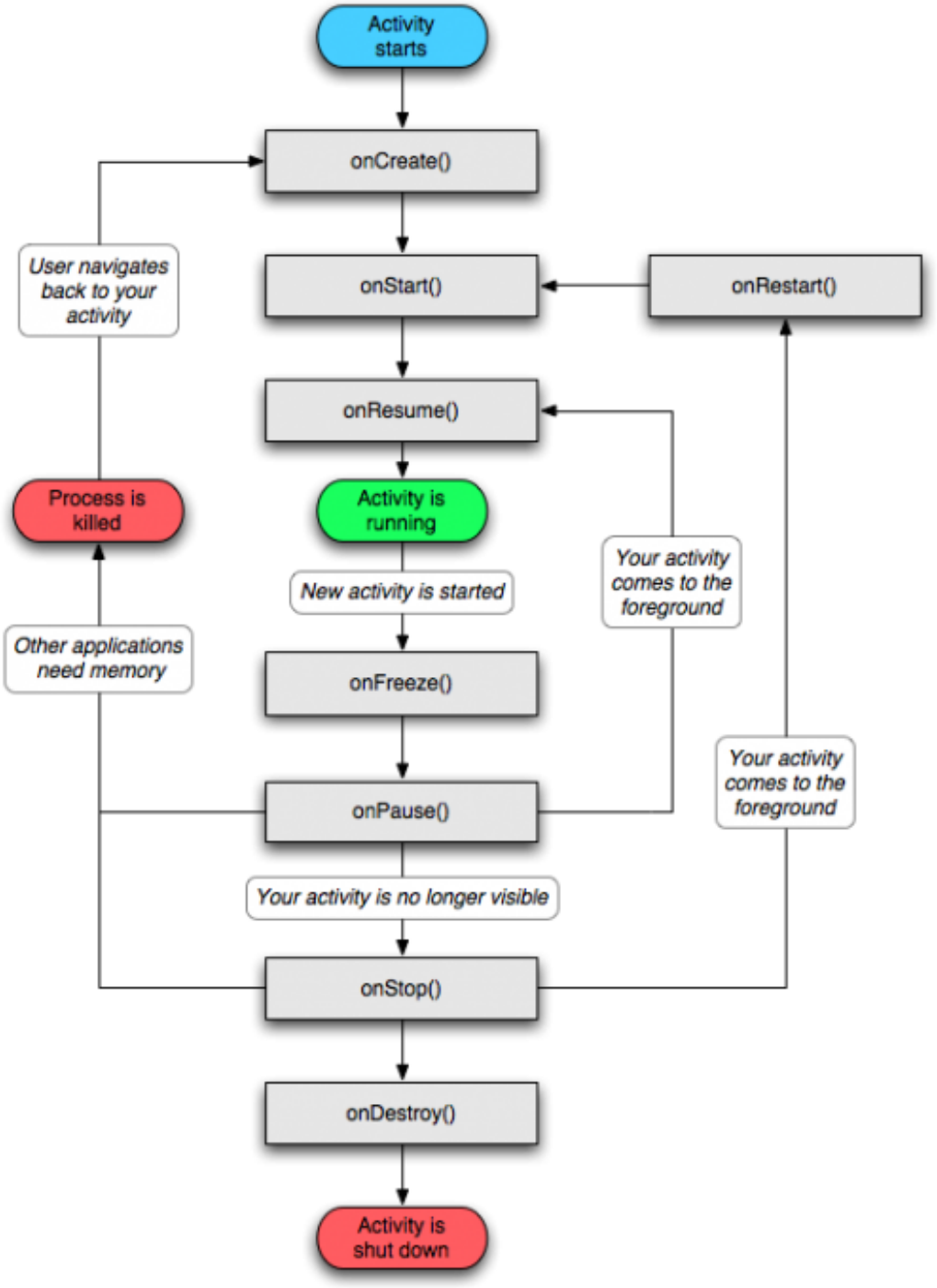


Figura 3.2 – *Ciclo de vida de una aplicación.*

De la figura anterior, pueden sacarse las siguientes conclusiones:

- **onCreate(), onDestroy():** Abarcan todo el ciclo de vida. Cada uno de estos métodos representan el principio y el fin de la actividad.
- **onStart(), onStop():** Representan la parte visible del ciclo de vida. Desde onStart() hasta onStop(), la actividad será visible para el usuario, aunque es posible que no tenga el foco de acción por existir otras actividades superpuestas con las que el usuario está interactuando. Pueden ser llamados múltiples veces.
- **onResume(), onPause():** Delimitan la parte útil del ciclo de vida. Desde onResume() hasta onPause(), la actividad no sólo es visible, sino que además tiene el foco de la acción y el usuario puede interactuar con ella.

Tal y como se ve en el diagrama de la figura anterior, el proceso que mantiene a esta Activity puede ser eliminado cuando se encuentra en onPause() o en onStop(), es decir, cuando no tiene el foco de la aplicación. Android nunca elimina procesos con los que el usuario está interactuando en ese momento. Una vez se elimina el proceso, el usuario desconoce dicha situación y puede incluso volver atrás y querer usarlo de nuevo. Entonces el proceso se restaura gracias a una copia y vuelve a estar activo como si no hubiera sido eliminado. Además, la Activity puede haber estado en segundo plano, invisible, y entonces es despertada pasando por el estado onRestart().

En el momento en el que Android detecta que no hay los recursos necesarios para poder lanzar una nueva aplicación, analiza los procesos existentes en ese momento y elimina los procesos que sean menos prioritarios para poder liberar sus recursos.

Cuando el usuario regresa a una actividad que está dormida, el sistema simplemente la despierta. En este caso, no es necesario recuperar el estado guardado porque el proceso todavía existe y mantiene el mismo estado. Sin embargo, cuando el usuario quiere regresar a una aplicación cuyo proceso ya no existe porque se necesitaba liberar sus recursos, Android lo crea de nuevo y utiliza el estado previamente guardado para poder restaurar una copia fresca del mismo. Como se ya ha explicado, el usuario no percibe esta situación ni conoce si el proceso ha sido eliminado o está dormido.

3.2. Basic4Android

3.2.1. Introducción

Como se menciono anteriormente, Basic4Android es un entorno comercial que nos permite desarrollar aplicaciones para Android programando en un

lenguaje muy similar a Visual Basic, sin embargo al compilar, en el fondo aún seguirá siendo Java. Permite trabajar cómodamente con ciertas librerías que facilitan las tareas. Algunas de estas librerías posibilitan trabajar con el GPS del móvil, el bluetooth, interacción con sitios web usando HTTP, tratamiento multimedia con archivos locales y streaming, controlar la cámara del móvil, o incluso con SQLite o MySQL, además de trabajar con reconocimiento de voz también trabaja de forma especial con Admob (publicidad para móviles), entre otros.

Figura 3.3 – *Basic4Andorid.*

Basic4Android cuenta con un diseñador de interfaces, de esta manera podemos conectar el entorno con un emulador (ADV Manager) y diseñar en tiempo real nuestra aplicación, de igual forma se puede conectar el móvil como alternativa.

Este diseñador nos permite arrastrar controles ya sean botones, cajas de texto, labels, entre otros componentes y crear un diseño profesional en segundos.

Este entorno ha dado lugar a la creación de una comunidad de usuarios activos en línea respondiendo preguntas de usuarios y aportando tutoriales completos, proporcionando un mayor apoyo para el producto.

En la página oficial se encuentra toda la documentación, www.basic4ppc.com aquí se puede acceder a la sección de descargas, documentación, foros, características que lo hacen sobresalir sobre otros ambientes. También desde allí se accede la sección de pago. El precio regular de este entorno es de 49 dólares para la licencia de un desarrollador con actualizaciones de 2 meses, mientras que la licencia enterprise nos proporciona acceso actualizaciones por 2 años y cuesta 99 dólares. También esta la posibilidad de descarga de una version de prueba por 30 días.

3.2.2. Entorno de desarrollo integrado

Basic4Android tiene un IDE (Entorno de desarrollo integrado) muy completo y fácil de usar, el cual da todas las herramientas necesarias al diseñador para poder trabajar de manera cómoda y prolija.

Al ejecutar el IDE se obtiene el cuadro 3.4:

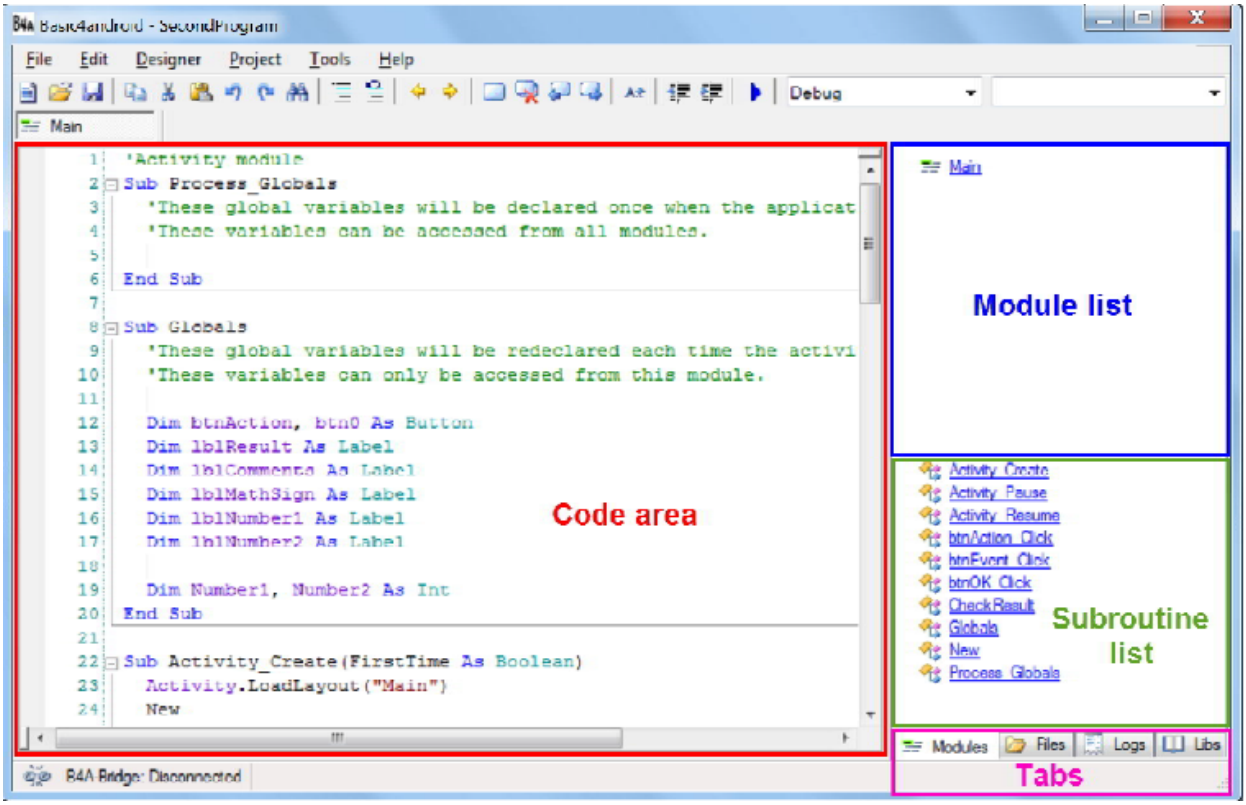


Figura 3.4 – *Entorno de desarrollo Basic4Andorid.*

Donde se pueden identificar los elementos con sus respectivas funciones según se ve en la Fig. 3.5:

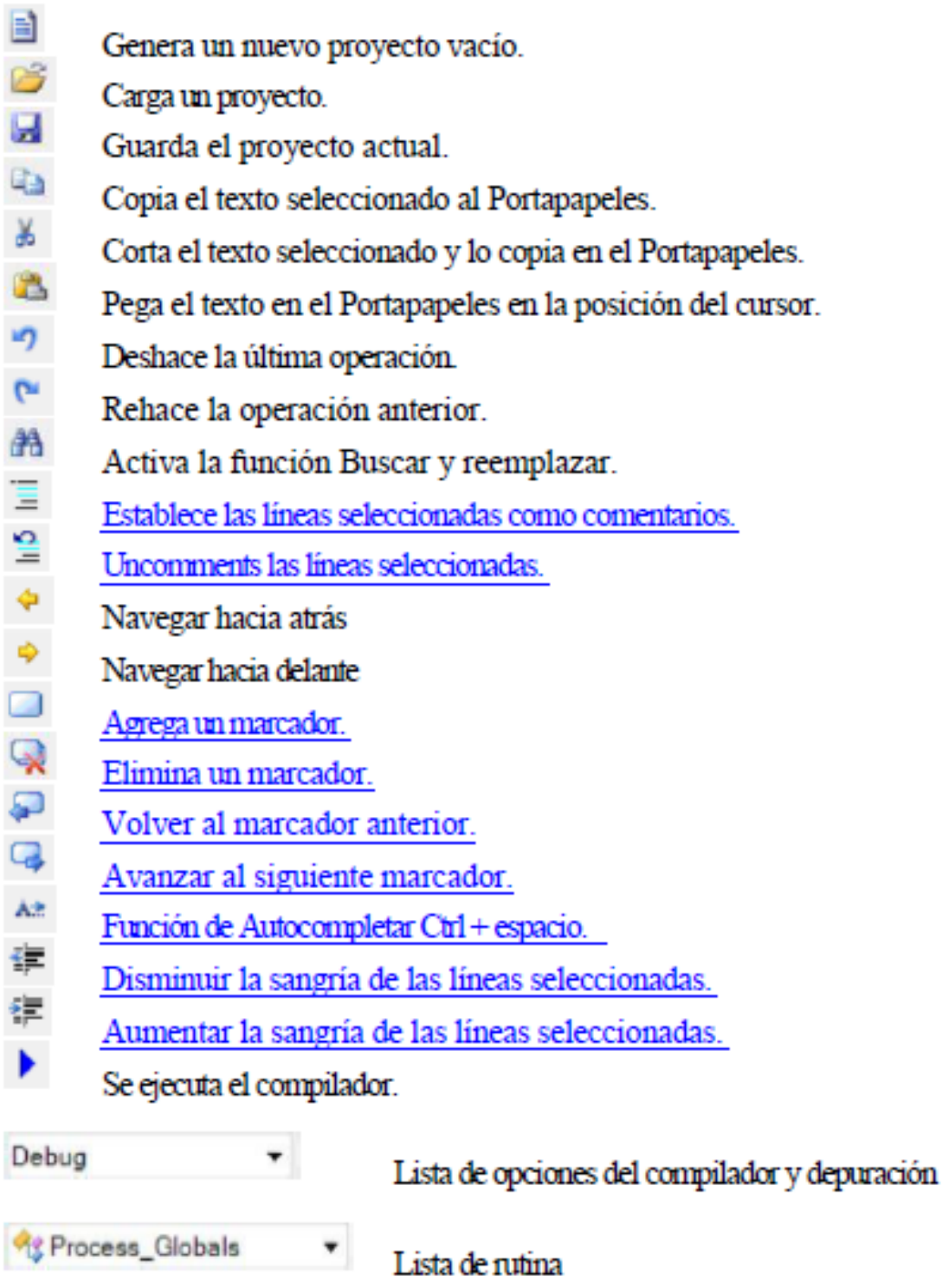


Figura 3.5 – *Elementos de Basic4Android.*

Si desplegamos el menú Project encontramos las diferentes opciones de acciones y funciones, como agregar módulos de actividad, servicios, cambiar nombre de módulos, elegir icono para el programa, cambiar nombre de la aplicación, entre otros, Fig. 3.6.

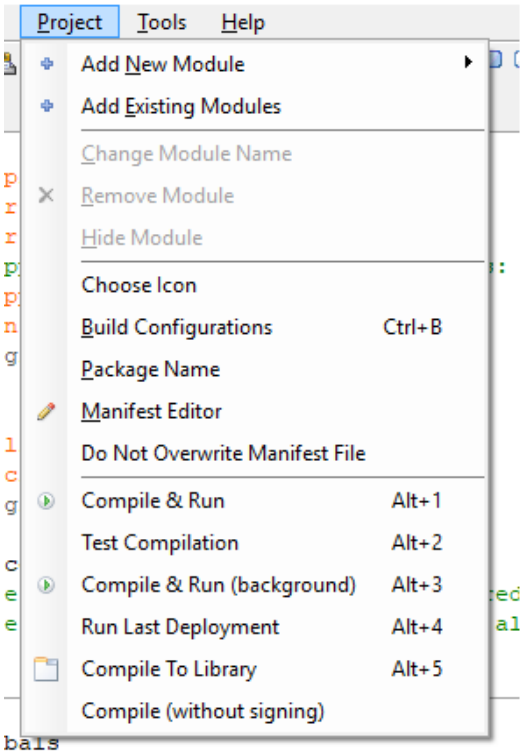


Figura 3.6 – *Menú Project*

Agregar nuevo Módulo

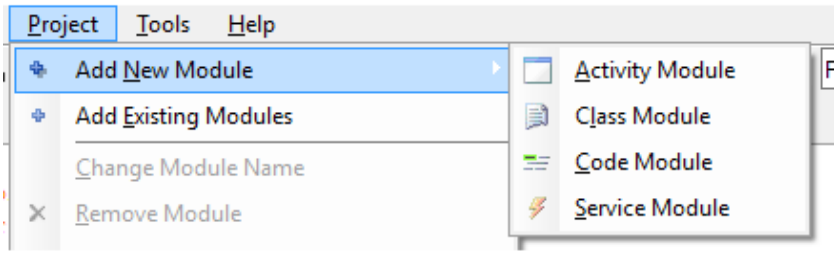
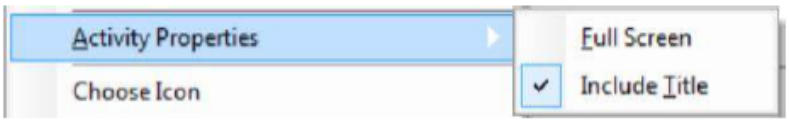


Figura 3.7 – *Menú Project.*

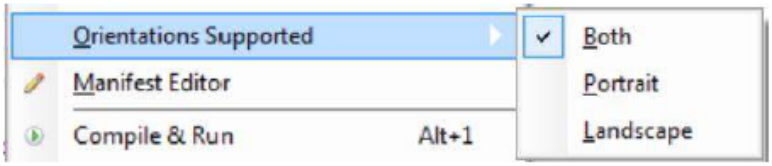
Propiedades de actividad

La aplicación puede utilizar la pantalla completa o no, o incluir la barra de titulo o no.

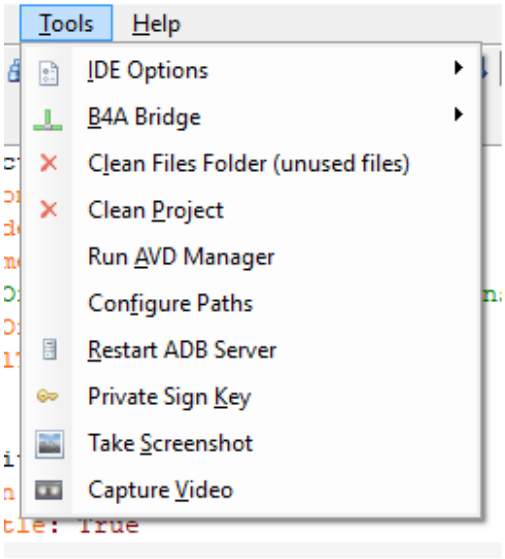


Orientaciones compatibles

Se puede definir si la aplicación se ejecute de manera vertical, horizontal o que rote según la posición del teléfono móvil.

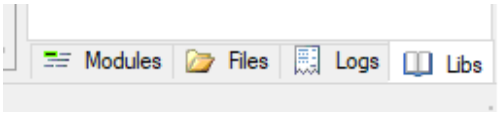


Menú Tools



3.2.3. Fichas

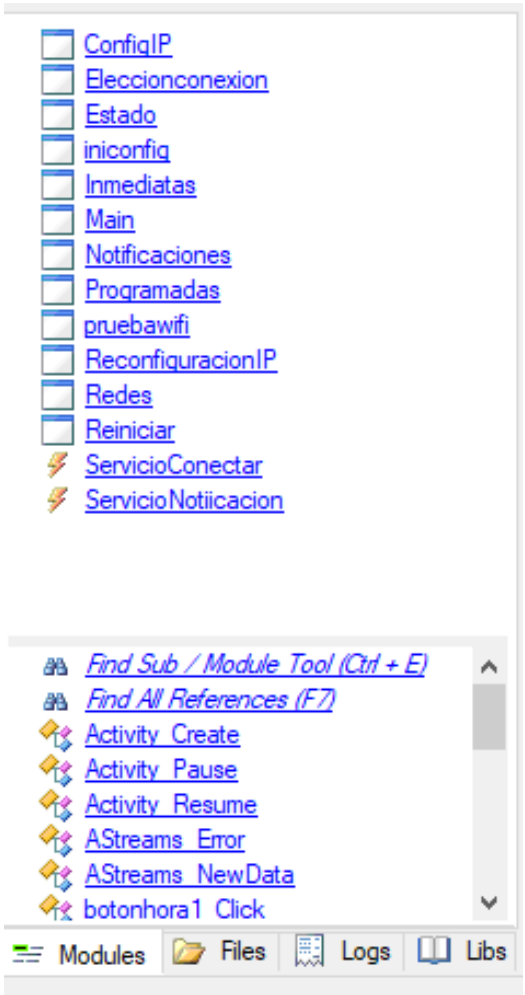
En la esquina inferior derecha del cuadro principal del IDE se pueden observar cuatro diferentes fichas de verdadera importancia a la hora de programar la aplicación:



Modules

Todos los módulos del proyecto y subrutinas del modulo seleccionado se pueden observar en esta ficha.

Existe siempre al menos un módulo, este módulo principal es denominado Main. Este es el que se ejecuta al iniciar la aplicación.



Hay tres tipos de módulos:

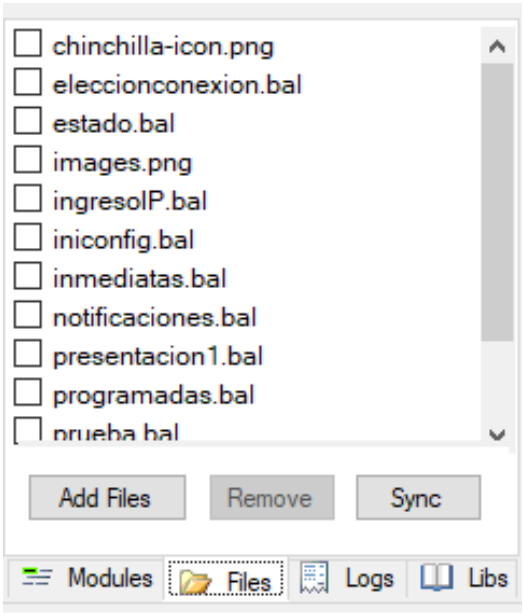
- **Módulos de actividad:** Cada actividad tiene su propio módulo. Cuando una actividad esta activa, las otras están en pausa o detenidas. Un módulo puede acceder a variables y procesos de otros módulos. Esto es posible si dichos objetos fueron declarados de manera global.

- **Módulos de código:** Los módulos de código solo contienen código. No es posible ninguna actividad en este tipo de módulo. Su propósito y ventaja es compartir un mismo código en diferentes programas, principalmente para cálculos u otra gestión general.

- **Módulos de servicio:** Los módulos de servicio pueden ejecutarse por mas que la aplicación no este corriendo en primer plano.

Files

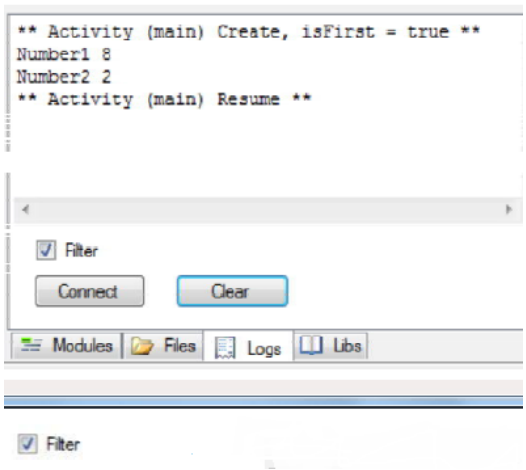
Esta ventana muestra todos los archivos que se han agregado al proyecto. Estos pueden ser cualquier tipo de archivos: diseños, imágenes, textos, iconos, etc. Estos archivos se guardan en la carpeta Files.DirAssets.



Logs

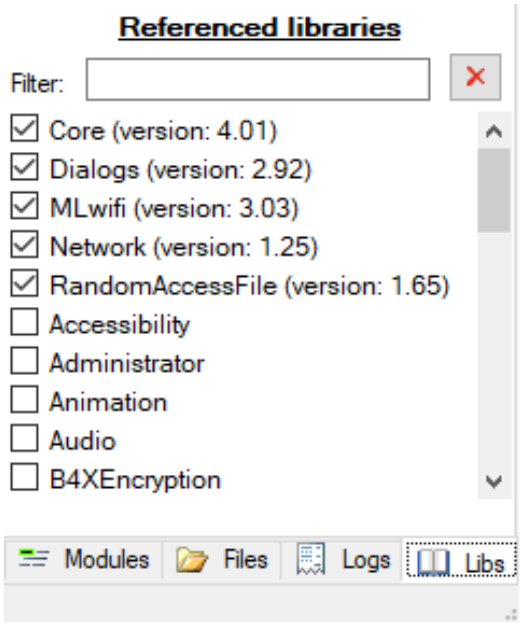
Aquí se pueden visualizar el flujo del programa, imprimiéndose todos los comentarios de registro generado por el programa cuando se esta ejecutando. Ademas de imprimir valores de variables y también se muestran errores. Esto resulta imprescindible para la búsqueda y depuración de errores cuando se esta programando aplicaciones.

Cuando se activa el filtro solo se verán mensajes relacionados con el propio programa, pero cuando no se activa se pueden observar todos los mensajes que se ejecutan en el sistema.



Libs

En esta sección se listan las bibliotecas disponibles que se pueden utilizar en el proyecto y las que ya están cargadas a nuestro proyecto. Las librerías permiten añadir mas objetos y funciones a B4A. Algunas ya vienen con B4A y son parte del sistema de desarrollo estándar. Otras, muchas veces desarrolladas por usuarios, pueden ser descargadas.



3.2.4. Compilación

Basic4Android tiene tres tipos de compilaciones posibles. Debug, Rlease y Releace (obfuscated). La primera permite la depuración del código, ya que se puede correr la aplicación e ir haciendo modificaciones, dar pausa, ejecutar determinados fragmentos del código, poner puntos de interrupción (el programa se detendrá cuando llegue al punto marcado y permite inspeccionar el estado actual). Al utilizar esta opción se abre en la parte inferior una ventana de depuración, Fig. 3.8.

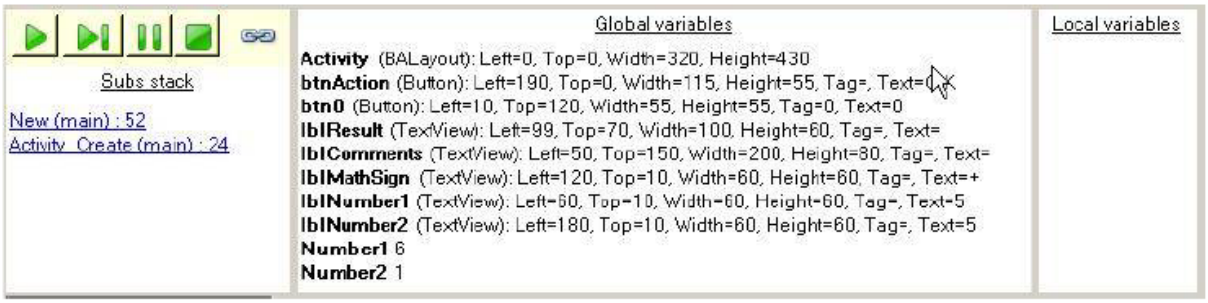


Figura 3.8 – *Cuadro de depuración.*

Para poder obtener el archivo *.apk* y poder distribuir la aplicación se deben usar las otras dos opciones de compilación, las Release. Durante la compilación Basic4Android traduce el código generando un código Java que luego es interpretado por un compilador de Java y lo convierte a formato de bytes de Android (Delvik). La diferencia con el Release ofuscado es que este tiene el propósito de hacer el código descompilado menos legible, más difícil de entender y hace proteger el código original.

Hasta ahora existen dos métodos para probar la aplicación durante el desarrollo: Un simulador de Android o el puente B4A, este último es conectando con un dispositivo físico por Wifi o Bluetooth.

Simulador Android

Durante la instalación de Basic4Android es necesario previamente instalar Android SDK. En Android SDK se incluyen las herramientas necesarias para dar los primeros pasos programando para esta plataforma: distintas APIs facilitadas por Google tanto para el control de las funciones del dispositivo como para la integración de servicios, un depurador, un emulador para testear las aplicaciones y toda la documentación necesaria para dar tus primeros pasos programando en Android.

Android Emulator, Fig. 3.9, simula un dispositivo y lo muestra en la computadora de desarrollo. Permite crear un prototipo de una app de Android, y también desarrollarla y probarla sin usar un dispositivo de hardware. El emulador es compatible con teléfonos y tablets Android, y con dispositivos Android Wear y Android TV. Viene con tipos de dispositivos predefinidos para comenzar rápidamente, y pueder crear definiciones propias de dispositivos y máscaras de emulador.

Android Emulator es rápido y potente, y presenta muchas funciones. Puede transferir información más rápido que un dispositivo de hardware conectado, lo que acelera el proceso de desarrollo. La función de núcleo múltiple permite que el emulador aproveche procesadores de núcleo múltiple en tu computadora de desarrollo para mejorar el rendimiento aún más.

Figura 3.9 – *Emulador Android.*

El emulador usa una configuración de Android Virtual Device (AVD) para determinar la apariencia, la funcionalidad y la imagen del sistema del

dispositivo simulado. Los AVD te permiten definir determinados aspectos de hardware de los dispositivos emulados y crear varias configuraciones para probar diferentes permutaciones de hardware y plataformas Android.

Cada AVD funciona como un dispositivo independiente, con su propio almacenamiento privado para datos de usuario, tarjetas SD, etc. Cuando inicias el emulador con una configuración de AVD, automáticamente se cargan los datos del usuario y de la tarjeta SD desde el directorio del AVD. En la configuración predeterminada, el emulador almacena los datos de usuario, los datos de la tarjeta SD y el caché en el directorio del AVD.

Bridge B4A

Bridge B4A, Fig. 3.10, es una aplicación disponible para descargar en el PlayStore de Google. Gracias a esta utilidad, la cuál esta programada en Basic4Android, podemos conectar mediante WiFi o BLUETOOH, nuestro dispositivo móvil al entorno de desarrollo, de forma que el testeo de las aplicaciones es mucho más rápido que en el dispositivo virtual de Android.

Figura 3.10 – *Bridge B4A.*

Diseñador

El diseñador permite generar diseños con el emulador o un dispositivo real, Fig. 3.11. Este diseñador muestra en tiempo real en el dispositivo o emulador los elementos gráficos que vamos agregando al diseño, ya sean botones, cuadros de texto, etiquetas, determinar sus ubicaciones y tamaños, pudiendo así tener una vista previa de como va quedando la interfaz gráfica. El diseñador cuenta también con una herramienta que se llama Diseñador Abstracto, que se muestra en simultaneo al diseño en el dispositivo, pero se muestra en la ventana del IDE, donde se ven los diferentes elementos

agregados pero de manera ilustrativa en forma de bloques, mostrando el lugar que ocupan y su nombre.

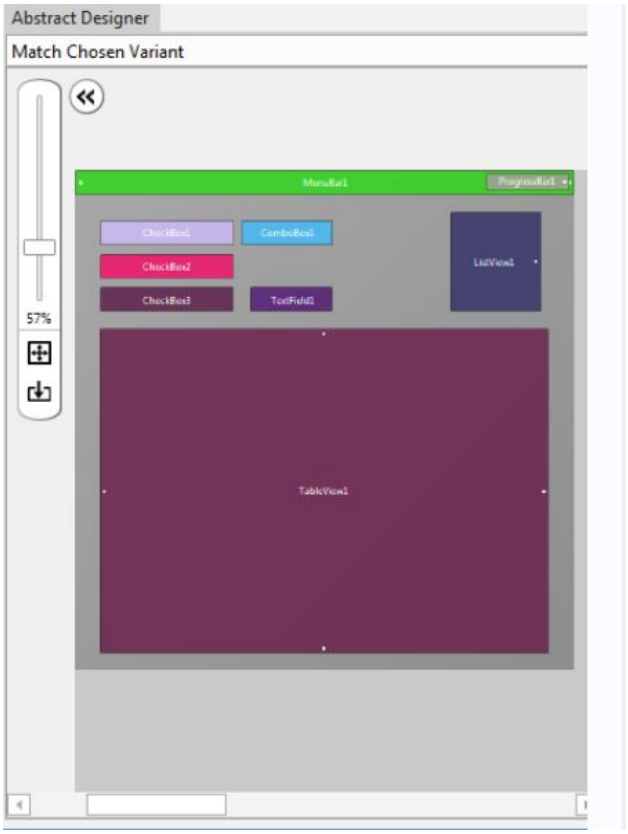


Figura 3.11 – *Diseñador abstracto.*

3.3. Aplicación *WifeedApp*

3.3.1. Introducción

En esta sección se hablará de la aplicación *WifeedApp*. Esta aplicación fue diseñada especialmente para poder controlar y monitorear el dispositivo *Wifeed* desde cualquier parte del mundo, de manera sencilla y portable. En la actualidad la gran mayoría de las personas cuentan con un teléfono celular smartphone, es por esto que los dispositivos Andorid resultan la mejor opción para una interfaz que permita que la persona pueda estar en constante contacto con el estado de sus mascota sin la necesidad de tener que tener acceso a una computadora. El objetivo de esta aplicación es poder configurar la unidad principal para que se pueda conectar a la

red WiFi local, configurar las acciones pre-programadas, como la entrega automática de alimento/agua, comandar los diferentes actuadores, como las electrovalvulas, abrir o cerrar una puerta, prender luces, o sistemas de calefacción y ventilación entre otros, ademas de permitir monitorear el dispositivo y las variables de interés y así conocer el estado de la mascota. El diseño y programación de ésta fue realizado con el entorno de desarrollo Basic4Android mencionado anteriormente, utilizando la version gratuita y con las bibliotecas que la misma trae incluida.

Figura 3.12 – *Pantalla de presentación.*

3.3.2. Interfaz gráfica

La aplicación *Wifeed*, Fig. 3.12, consiste en una serie de pantallas presentadas en forma de solapas, cada una de ellas tiene una finalidad diferente, más específicamente seis pantallas agrupas en dos secciones, la

primera orientada a la visualización de las variables de estado, acciones on line, y de acciones programadas. Por otra parte, una segunda sección orientada a la configuración de la aplicación, permitiendo definir las características de las alarmas y notificaciones, la reconfiguración del enlace con la unidad principal, y por último una solapa que permite reiniciar la aplicación y el dispositivo principal.

A continuación, se muestra un diagrama de estados para poder visualizar las diferentes pantallas y las transiciones de unas a otras al presionar los diferentes botones.

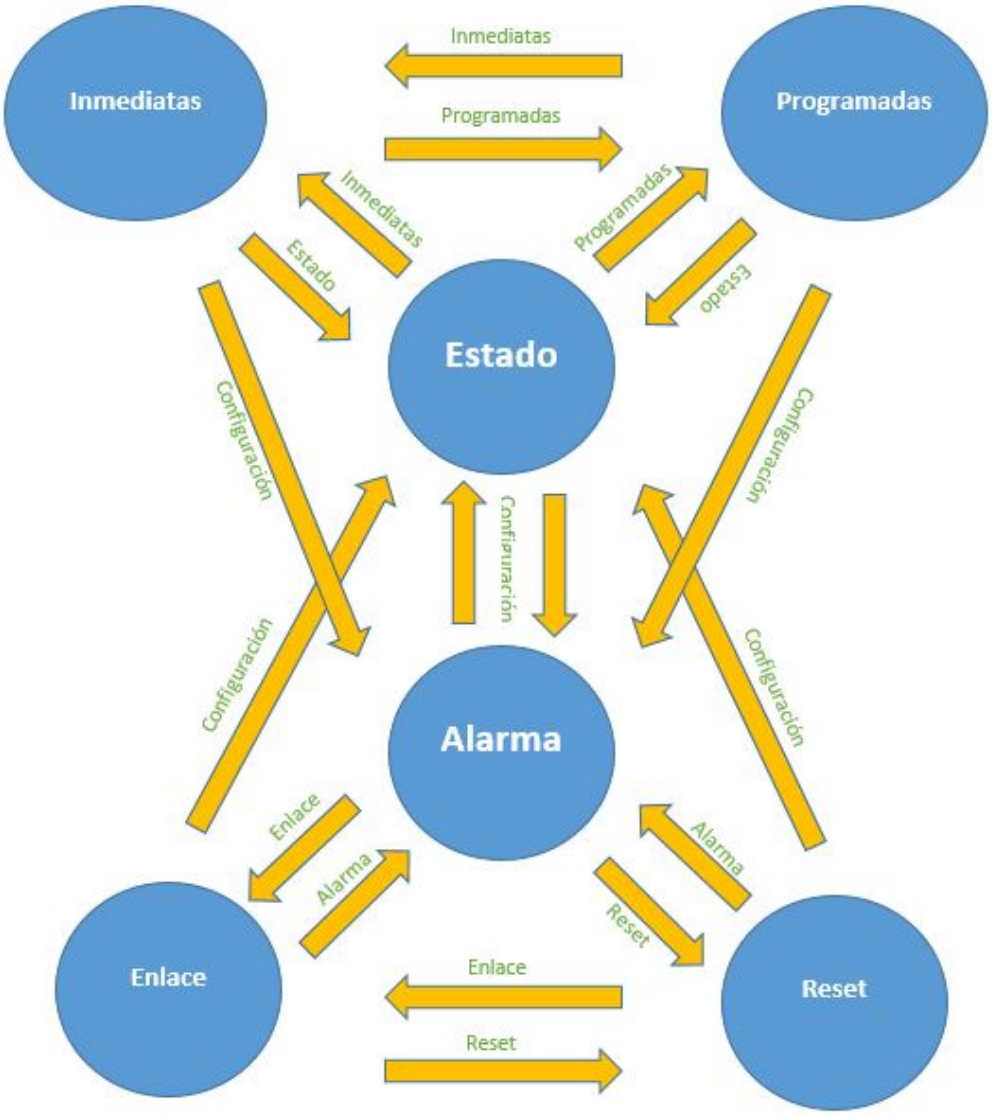


Figura 3.13 – *Diagrama de estados.*

En los siguientes párrafos se presentan y desarrollan los diferentes estados para explicar sus funcionalidades y características del programa. En la Fig. 3.14 puede verse los estados posibles de la aplicación, que se corresponden a las pantallas de la aplicación, y las transiciones entre ellas.

Estado

Esta pantalla es una de las más importantes de la aplicación, ya que la misma muestra el estado del dispositivo mediante la visualización de

diferentes variables que permiten por ejemplo conocer el estado de actuadores y sensores, la hora de ocurrencia de determinados eventos, y la programación de alimentación definida.

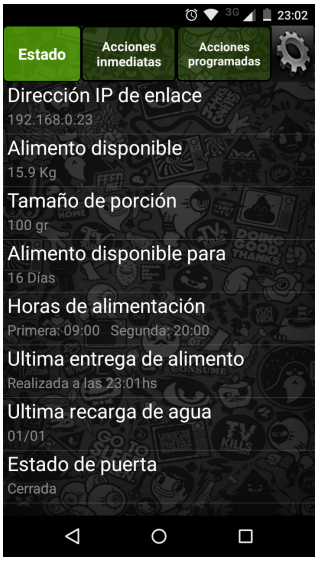


Figura 3.14 – *Pantalla Estado.*

- **Dirección IP de enlace:** Aquí se muestra la dirección IP con la cual se vincula la aplicación con el dispositivo principal, esta puede ser del tipo local si sólo se desea controlar el dispositivo estando dentro del hogar, o del tipo publica si se quiere acceder de forma remota.
- **Alimento disponible:** Se observa la cantidad de alimento disponible en la tolva de almacenaje. Este valor se va actualizando cada una hora y cada vez que se hace una entrega de alimento mediante la medición con un sensor de peso.
- **Tamaño de porción:** Tamaño de la porción que se programo en la sección de programación de alimentación.
- **Alimento disponible para:** Basándose en la cantidad de alimento disponible de alimento, el tamaño de porción y las veces al día que se debe entregar alimento, el dispositivo hace el calculo de cuantos días de disponibilidad de alimento tiene el animal y la aplicación muestra este valor.
- **Horas de alimentación:** Se visualizan los horarios programados para que se entregue el alimento de forma automática.

- **Última entrega de alimento:**Aquí queda registrado el día y la hora de la ultima entrega de alimento, ya sea automática o porque el usuario dio la orden, esto ayuda a tener un control y verificar que la alimentación se haya realizado.

- **Última recarga de agua:** Registro del día de la ultima entrega de agua.

- **Estado de puerta:** Indica si la puerta se encuentra abierta o cerrada.

Acciones Inmediatas

En esta solapa, ver Fig. 3.15, el usuario podrá realizar tres acciones de manera espontanea cuando se desea: realizar una entrega de alimento en el momento, recargar el bebedero de agua, y abrir o cerrar la puerta electrónica. La cantidad de alimento que se entregara sera la previamente programa por el usuario. De no haberse definido esta cantidad, un mensaje emergente notificara de la necesidad de definir dicho valor.

Figura 3.15 – *Pantalla Acciones Inmediatas.*

Para evitar enviar ordenes por pulsaciones accidentales de los botones, se diseñó a estos últimos de cierto modo de que requieren de la pulsación sea de al menos un segundo para que sea considerada válida.

Acciones Programadas

La aplicación permite en esta sección programar todo lo que respecta a la alimentación automática. Como puede observarse en la Fig. 3.16, permite introducir el tamaño en gramos de la porción de alimento que se expende tanto en la entrega automática como en la espontánea desde la solapa "Acciones Inmediatas". A su vez permite habilitar o deshabilitar la opción de entregas automáticas, por si el usuario solo desea alimentar cuando el lo indique.

Figura 3.16 – *Pantalla Acciones Programadas.*

Al tildar el casillero de "Habilitar"se despliega la opción de indicar cuantas veces al días desea que se hagan las entregas automáticas, las mismas pueden ser de una a tres veces, Fig. 3.17.

Figura 3.17 – *Pantalla Acciones Programadas.*

Dependiendo del número de veces que se seleccione la aplicación la aplicación dará la posibilidad de modificar las respectivas horas de entrega. Al presionar en el botón de "modificar hora.emerge un cuadro de selección de horas y minutos para setear el horario deseado, Fig. 3.18.

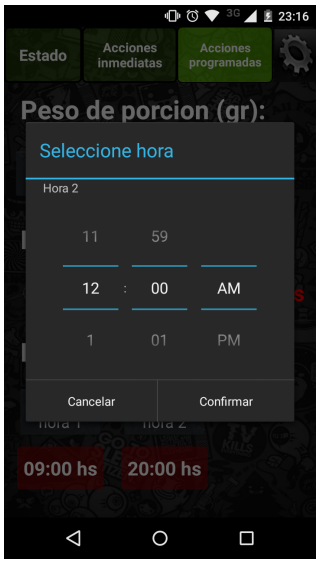


Figura 3.18 – *Modificación de hora de entrega.*

Alarma

En esta sección es posible configurar la forma en que la aplicación notifica de las diferentes alertas que el sistema emite, permitiendo habilitar o deshabilitar las notificaciones, Fig. 3.19. En caso de que las mismas se habiliten por definir las características de las mismas, es decir, que emitan sonido o que solo aparezca un aviso en la barra de notificaciones, que se produzca una vibración y que si el usuario desea que el led de notificaciones se encienda.

Figura 3.19 – *Configuración de notificaciones.*

Enlace

En caso de que por algún motivo la dirección IP que permite vincularse con el dispositivo principal haya sido modificada es posible reconfigurar la aplicación en esta solapa, dando la posibilidad de ingresar la nueva dirección sin tener que reiniciar la aplicación, evitando así la perdida de las configuraciones de alimentación y demás, Fig. 3.20.

Figura 3.20 – *Reconfiguración de enlace.*

Reset

En esta pestaña se da la opción para realizar reinicios a diferentes niveles, Fig. 3.21:

- **Reiniciar la aplicación:** Esta opción permite reiniciar la aplicación para volver al estado inicial de la misma cuando fue descargada, es decir que se perderá la configuración de enlace con el dispositivo principal y la configuración de las notificaciones. No se perderá la programación de la alimentación ya que esta es almacenada en la unidad central.

- **Resetear Wifeed:** Desde aquí se permite reiniciar el microcontrolador de la unidad principal por software, no se perderá ningún tipo de programación. Esto sirve por si se detecta que el dispositivo esta haciendo algún comportamiento erróneo y requiere de un reinicio.

- **Reestablecer a valores de fabrica:** Permite reestablecer todo el sistema a la condición original de fabrica, todas las variables y configuraciones se borran y la unidad principal hace el inicio para ser configurado y poder conectarse a la red WiFi.

Figura 3.21 – *Pestaña de Reset.*

Por cuestiones de seguridad cada vez que se pulse uno de estos tres botones la aplicación pedirá confirmación de la acción con una cartel emergente.

3.3.3. Conectividad y comunicación

Al iniciar por primera vez la aplicación se debe generar el enlace con el la unidad central, para esto previamente debe configurarse a esta última para que pueda tener acceso a la red local y así obtener la dirección IP de enlace. Esta configuración puede realizarse a través de *WifeedApp*, es decir, desde el teléfono móvil podemos comandar el dispositivo para indicarle a que red se debe conectar, Fig. 3.22.

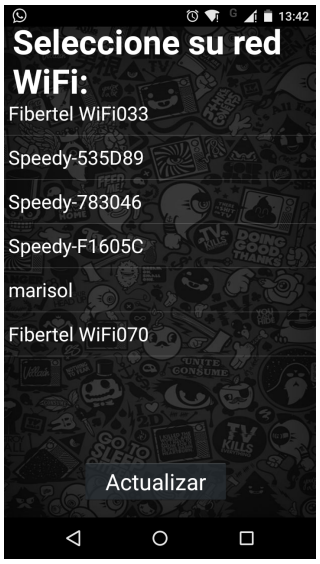


Figura 3.22 – *Redes disponibles.*

Para esto, al iniciar la tarea de configuración WiFi, la aplicación se conecta automáticamente a la red WiFi generada por el dispositivo principal, logrando así una primera vinculación necesaria para continuar con el proceso de configuración. Una vez esto el móvil hace un escaneo de las redes WiFi disponibles en el área y se muestran en pantalla. El usuario deberá escoger la red de su hogar y al seleccionarla un cartel emergente solicitara el ingreso de la contraseña. La aplicación le envía los datos al dispositivo, este los recibe e intenta conectarse a dicha red, de ser exitosa la conexión este responde con un mensaje de éxito y el proceso continua, Fig. 3.23. Si la clave resulta no ser la correcta se indica que la misma es incorrecta y se debe seleccionar nuevamente la red para reintentar el ingreso del password.

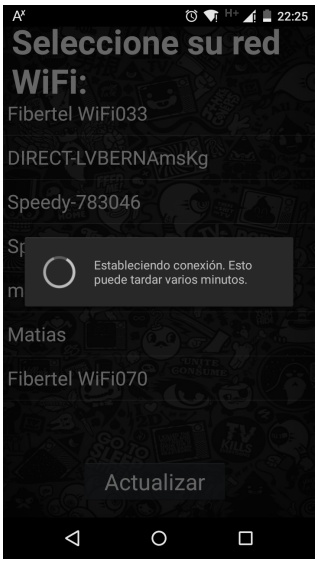


Figura 3.23 – *Conectando a red WiFi.*

Una vez acepada la contraseña y establecida la conexión a la red, la aplicación desconecta el teléfono de la red *Wifeed* y lo vuelve a conectar a la red del hogar. Luego la unidad principal consulta la IP local que el router le asignó y a través de Internet, si es que la red tiene acceso, consulta la IP pública. Así presenta en pantalla estas dos direcciones y da a elegir si se quiere establecer un enlace local o externo para darle un uso remoto, Fig. 3.24. Esto es por si la red local no tiene acceso a Internet pero igual se desea tener un control del dispositivo cuando se esta en el hogar. Luego de esto ya se puede ejercer control sobre el dispositivo y no se requerirá de estos pasos previos a menos que el sistema sea restablecido y eliminada la configuración.

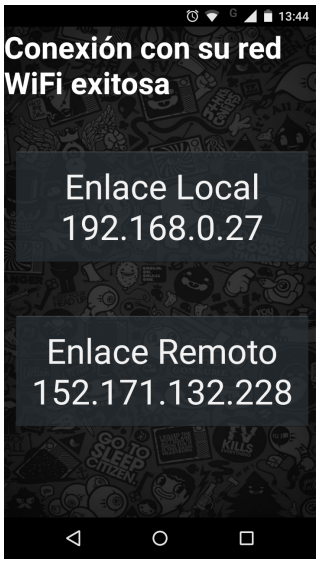


Figura 3.24 – *Tipo de enlace.*

En el caso en el que todo el proceso de conexión a la red local haya sido realizado previamente desde la interfaz web, no seria necesario realizar todos los pasos mencionados anteriormente, sólo se le debe indicar a la aplicación cuál es la dirección IP de enlace para que logre vincularse. Es por esto que al iniciar por primera vez la aplicación ésta da la opción de iniciar la configuración WiFi o introducir la dirección IP.

Figura 3.25 – *Configuración inicial.*

Al elegir esta opción, se muestra una pantalla para directamente introducir la dirección de enlace, Fig. 3.25, al ingresarla la aplicación intenta comunicarse con el servidor a través de ella, si esta comunicación es exitosa significa que la dirección es correcta, de lo contrario emerge un mensaje indicando que la dirección no es correcta y que se la ingrese nuevamente. Se repite el proceso hasta obtener respuesta del servidor y así establecer el vínculo, Fig. 3.26.

Figura 3.26 – *Ingreso de dirección de enlace.*

Una vez que el enlace fue generado, la dirección queda guardada en la aplicación y ya no sera necesario volver a ingresarla, en posteriores ejecuciones de la aplicación la comunicación se establecerá automáticamente en el proceso de inicialización de la misma.

En la Fig. 3.27 se muestra el diagrama de flujo de todo lo anteriormente mencionado para poder visualizar mejor el proceso completo.

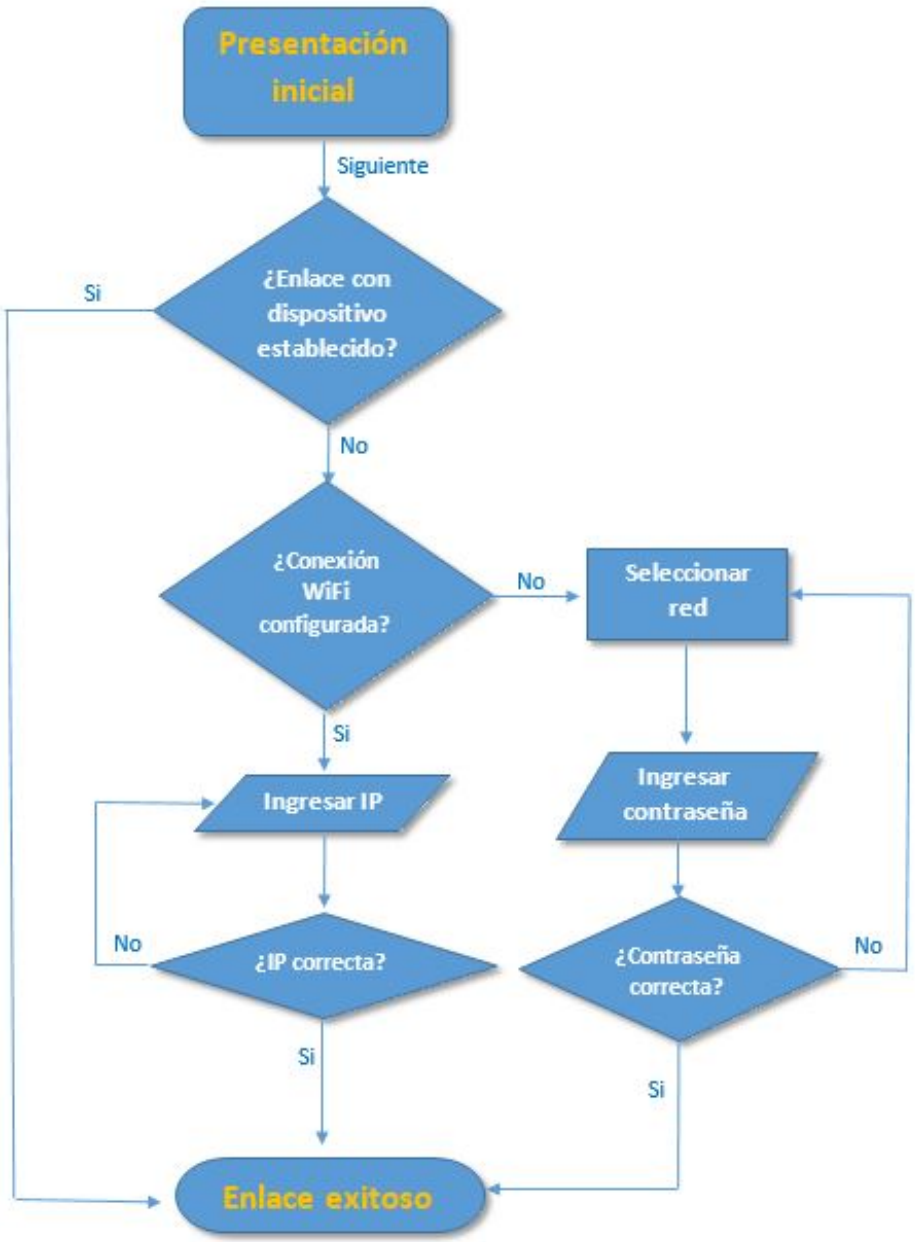


Figura 3.27 – *Diagrama de flujo del proceso de enlace.*

Como se observa en la Fig. 3.25, existe una opción más para seleccionar, la que se indica bajo el nombre "Usar sin internet". Ésta, como su nombre lo indica, es para comandar el sistema de manera local sin la necesidad de contar con una red local ni Internet. Esto es posible ya que el teléfono móvil se comunicará con el dispositivo principal mediante a la red WiFi generada

por el propio dispositivo, esta red como ya se mencionó, aparecerá bajo el nombre de *Wifeed*.

Al seleccionar esta opción, el celular se conectara automáticamente a esta red para poder vincularse con el dispositivo. Una vez conectado, el sistema obtendrá la información del día y la hora del propio teléfono, ya que estos datos son fundamentales para poder realizar las tareas de automatización. Una vez obtenida esta información, el sistema será el encargado de continuar con el conteo de hora y fecha, y cada vez que el teléfono esté dentro del área de cobertura de la red y logre comunicarse, consultará estos datos para que no ocurran posibles desfasajes y así tener siempre la hora exacta.

Es evidente que mediante esta opción de comunicación sólo se podrá ejercer un control local estando en el lugar donde se encuentre el dispositivo. Pensado principalmente para poder programar las acciones automáticas o simplemente dar ordenes en el momento sin tener que moverse.

3.3.4. Servicio y notificaciones

Si bien la aplicación podría mantenerse en ejecución solo cuando el usuario la tiene en primer plano, mientras realiza configuraciones, visualiza variables o ejecuta acciones y al salir la misma se cerrara, esto tendría un inconveniente, no seria posible que la aplicación notifique cuando se está por quedar sin alimento o cuando se pierda comunicación con el dispositivo *Wifeed* a menos que el usuario la tenga abierta. Este problema se resuelve con que la aplicación tenga corriendo un servicio en segundo plano, y que éste sea capaz de volver a ejecutarse automáticamente en caso que el sistema operativo, por cuestiones de falta de memoria decida cerrarlo y así liberar espacio en caché.
Es por lo mencionado en el párrafo anterior que *WifeedApp* cuenta con un servicio que está en constante ejecución, y si Android intenta matarlo, este es capaz de reiniciarse pasado tres segundos.
Éste servicio comienza a ejecutarse una vez concluido el proceso inicial de vinculación entre el móvil y el dispositivo central. Si el teléfono celular es apagado, al prenderlo nuevamente e iniciado el sistema operativo el servicio comienza a ejecutarse automáticamente.
La función principal de este servicio es mantener el enlace de comunicación entre el móvil y *Wifeed*. Con una determinada frecuencia ajustable, cuyo ajusto dependerá del consumo de datos deseado por el usuario, el servicio intenta comunicarse con la unidad principal, si no recibe respuesta significa que se ha perdido la comunicación, ya sea porque algunos de los dos

dispositivos se reiniciaron, perdieron acceso a Internet, o porque simplemente se cayo la comunicación, entonces automáticamente el servicio reanuda el enlace *Wifeed*. De no ser posible esta reanudación, deja pasar un minuto e intenta nuevamente, siguiendo esta secuencia tres veces, si en ninguna de ellas pudo comunicarse, se muestra un alerta en la barra de notificaciones, que indica que se ha perdido la comunicación con el dispositivo. Esta notificación no podrá ser removida hasta que se la desactive entrando en la aplicación y desactivando las alertas. Esto se hace para que el usuario sepa que no podrá controlar su *Wifeed* porque no tiene Internet en su móvil, o perdió conexión en su hogar, o lo que seria más grave, su *Wifeed* esta en falla y es posible que su mascota no pueda ser alimentada. No obstante, el servicio continúa intentando restablecer el enlace cada cinco minutos, en caso de éxito, la notificación desaparece de la barra de notificaciones evidenciando que todo el sistema esta en correcto funcionamiento operativo, Fig. 3.28.

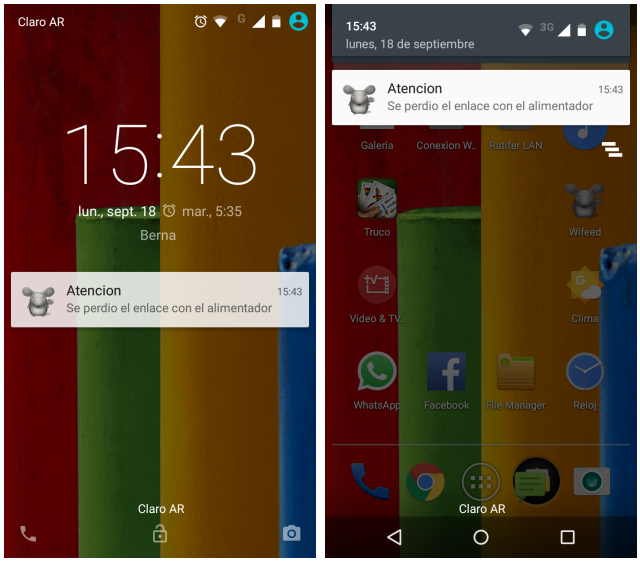


Figura 3.28 – *Alerta de perdida de enlace.*

Otra función importante de este servicio es el hacer consultas periódicas del estado del dispositivo para ir actualizando las variables a mostrar. El servidor al recibir la consulta responde con un paquete que incluye todos los valores actuales de las variables. La aplicación móvil recibe este paquete y actualiza cada una de las variables. Al leer el valor de la variable que indica cuanto alimento queda disponible en el dispenser de almacenaje se fija si es suficiente para al menos un día de alimentación, de no ser suficiente dará un alerta para indicar que el alimento se esta por acabar y que el animal se quedara sin comida. Al igual que en alerta por perdida de comunicación, esta

notificación aparecerá en la barra de notificaciones y no se desactivara hasta que se recargue el alimento o se desactive desde la sección de configuración, Fig. 3.29.

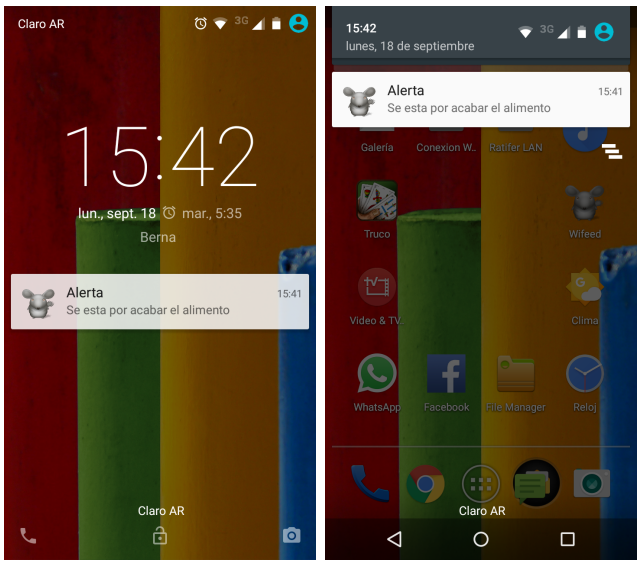


Figura 3.29 – *Alerta de bajo nivel de alimento.*

Capítulo 4
Interfaz Web

4.1. Protocolo TCP/IP

Una red es una configuración de computadora que intercambia información. Pueden proceder de una variedad de fabricantes y es probable que tenga diferencias tanto en hardware como en software, para posibilitar la comunicación entre estas es necesario un conjunto de reglas formales para su interacción. A estas reglas se les denominan protocolos. Un protocolo es un conjunto de reglas establecidas entre dos dispositivos para permitir la comunicación entre ambos.

Se han desarrollado diferentes familias de protocolos para comunicación por red de datos para los sistemas UNIX. El más ampliamente utilizado es el Internet Protocol Suite, comúnmente conocido como TCP / IP. Es un protocolo DARPA que proporciona transmisión fiable de paquetes de datos sobre redes. El nombre TCP / IP Proviene de dos protocolos importantes de la familia, el Transmission Contorl Protocol (TCP) y el Internet Protocol (IP). Todos juntos llegan a ser más de 100 protocolos diferentes definidos en este conjunto.

El TCP / IP es la base del Internet que sirve para enlazar computadoras que utilizan diferentes sistemas operativos, sobre redes de área local y área extensa. TCP / IP fue desarrollado y demostrado por primera vez en 1972 por el departamento de defensa de los Estados Unidos, ejecutándolo en el ARPANET una red de área extensa del departamento de defensa.

4.1.1. Capas conceptuales de protocolos

Pensemos los módulos del software de protocolos en una pila vertical constituida por capas. Cada capa tiene la responsabilidad de manejar una parte del problema. Conceptualmente, enviar un mensaje desde un programa de aplicación en una maquina hacia un programa de aplicaciones en otra,

significa transferir el mensaje hacia abajo, por las capas sucesivas del software de protocolo en la maquina emisora, transferir un mensaje a través de la red y luego, transferir el mensaje hacia arriba, a través de las capas sucesivas del software de protocolo en la maquina receptora. En la practica, el software es mucho más complejo de lo que se muestra en el modelo. Cada capa toma decisiones acerca de lo correcto del mensaje y selecciona una acción apropiada con base en el tipo de mensaje o la dirección de destino. Por ejemplo, una capa en la maquina de recepción debe decidir cuándo tomar un mensaje o enviarlo a otra maquina. Otra capa debe decidir que programa de aplicación deberá recibir el mensaje.

Independientemente del esquema de estratificación por capas que se utilice o de las funciones de las capas, la operación de los protocolos estratificados por capas se basa en una idea fundamental. La idea, conocida como principio de estratificación por capas puede resumirse de la siguiente forma:

Los protocolos estratificados por capas están diseñados de modo que una capa n en el receptor de destino reciba exactamente el mismo objeto enviado por la correspondiente capa n de la fuente. El principio de estratificación por capas explica por que la estratificación por capas es una idea poderosa. Esta permite que el diseñador de protocolos enfoque su atención hacia una capa a la vez, sin preocuparse acerca del desempeño de las capas inferiores. Por ejemplo, cuando se construye una aplicación para transferencia de archivos, el diseñador piensa solo en dos copias del programa de aplicación que se correrá en dos máquinas y se concentrará en los mensajes que se necesitan intercambiar para la transferencia de archivos. El diseñador asume que la aplicación en el anfitrión receptor es exactamente la misma que en el anfitrión emisor.

En términos generales, el software TCP/IP está organizado en cuatro capas conceptuales que se construyen sobre una quinta capa de hardware. La Fig. 4.1 muestra las capas conceptuales así como la forma en que los datos pasan entre ellas.

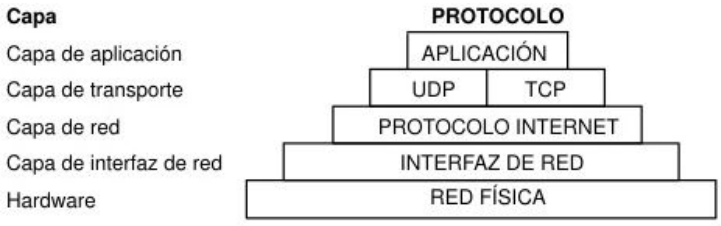


Figura 4.1 – *Capas del modelo TCP/IP.*

Capa de aplicación

La Capa de Aplicación es el nivel mas alto, los usuarios llaman a una aplicación que acceda servicios disponibles a través de la red de redes TCP/IP. Una aplicación interactúa con uno de los protocolos de nivel de transporte para enviar o recibir datos. Cada programa de aplicación selecciona el tipo de transporte necesario, el cual puede ser una secuencia de mensajes individuales o un flujo continuo de octetos. El programa de aplicación pasa los datos en la forma requerida hacia el nivel de transporte para su entrega.

Existen varios protocolos de capa de aplicación. En la lista siguiente se incluyen ejemplos de protocolos de capa de aplicación:

- Servicios TCP/IP estándar como los comandos ftp, tftp y telnet.
- Comandos UNIX "r", como rlogin o rsh..
- Servicios de nombres, como NIS o el sistema de nombre de dominio (DNS).
- Servicios de directorio (LDAP).
- Servicios de archivos, como el servicio NFS.
- Protocolo simple de administración de red (SNMP), que permite administrar la red.
- Protocolo RDISC (Router Discovery Server) y protocolos RIP (Routing Information Protocol).

Capa de transporte

La principal tarea de la capa de transporte es proporcionar la comunicación entre un programa de aplicación y otro. Este tipo de comunicación se conoce frecuentemente como comunicación punto a punto. La capa de transporte regula el flujo de información.

Puede también proporcionar un transporte confiable, asegurando que los datos lleguen sin errores y en secuencia. Para hacer esto, el software de protocolo de transporte tiene el lado de recepción enviando acuses de recibo de retorno y la parte de envío retransmitiendo los paquetes perdidos. El software de transporte divide el flujo de datos que se está enviando en pequeños fragmentos (por lo general conocidos como paquetes) y pasa cada paquete, con una dirección de destino, hacia la siguiente capa de transmisión. Aun cuando en el esquema anterior se utiliza un solo bloque para representar

la capa de aplicación, una computadora de propósito general puede tener varios programas de aplicación accesando la red de redes al mismo tiempo.

La capa de transporte debe aceptar datos desde varios programas de usuario y enviarlos a la capa del siguiente nivel. Para hacer esto, se añade información adicional a cada paquete, incluyendo códigos que identifican qué programa de aplicación envía y qué programa debe recibir, así como una suma de verificación para verificar que el paquete ha llegado intacto y utiliza el código de destino para identificar el programa de aplicación en el que se debe entregar.

Los protocolos de capa de transporte de este nivel son el Protocolo de control de transmisión (TCP), el Protocolo de datagramas de usuario (UDP) y el Protocolo de transmisión para el control de flujo (SCTP). Los protocolos TCP y SCTP proporcionan un servicio completo y fiable. UDP proporciona un servicio de datagrama poco fiable.

- **Protocolo TCP:** TCP permite a las aplicaciones comunicarse entre sí como si estuvieran conectadas físicamente. TCP envía los datos en un formato que se transmite carácter por carácter, en lugar de transmitirse por paquetes discretos. Esta transmisión consiste en lo siguiente: Punto de partida, que abre la conexión, transmisión completa en orden de bytes, y punto de fin que cierra la conexión. TCP conecta un encabezado a los datos transmitidos. Este encabezado contiene múltiples parámetros que ayudan a los procesos del sistema transmisor a conectarse a sus procesos correspondientes en el sistema receptor. TCP confirma que un paquete ha alcanzado su destino estableciendo una conexión de punto a punto entre los hosts de envío y recepción. Por tanto, el protocolo TCP se considera un protocolo fiable orientado a la conexión.

- **Protocolo SCTP:**SCTP es un protocolo de capa de transporte fiable orientado a la conexión que ofrece los mismos servicios a las aplicaciones que TCP. Además, SCTP admite conexiones entre sistema que tienen más de una dirección, o de host múltiple. La conexión SCTP entre el sistema transmisor y receptor se denomina asociación. Los datos de la asociación se organizan en bloques. Dado que el protocolo SCTP admite varios hosts, determinadas aplicaciones, en especial las que se utilizan en el sector de las telecomunicaciones, necesitan ejecutar SCTP en lugar de TCP.

- **Protocolo UDP:** UDP proporciona un servicio de entrega de datagramas. UDP no verifica las conexiones entre los hosts transmisores y receptores. Dado que el protocolo UDP elimina los procesos de

establecimiento y verificación de las conexiones, resulta ideal para las aplicaciones que envían pequeñas cantidades de datos.

Capa de Internet

La capa Internet maneja la comunicación de una máquina a otra. Ésta acepta una solicitud para enviar un paquete desde la capa de transporte, junto con una identificación de la máquina, hacia la que se debe enviar el paquete. La capa Internet también maneja la entrada de datagramas, verifica su validez y utiliza un algoritmo de ruteo para decidir si el datagrama debe procesarse de manera local o debe ser transmitido. Para el caso de los datagramas direccionados hacia la máquina local, el software de la capa de red de redes borra el encabezado del datagrama y selecciona, de entre varios protocolos de transporte, un protocolo con el que manejará el paquete. Por último, la capa Internet envía los mensajes ICMP de error y control necesarios y maneja todos los mensajes ICMP entrantes.

Esta capa incluye el potente Protocolo de Internet (IP), el protocolo de resolución de direcciones (ARP) y el protocolo de mensajes de control de Internet (ICMP).

- **Protocolo IP:** El protocolo IP y sus protocolos de enrutamiento asociados son posiblemente la parte más significativa del conjunto TCP/IP. El protocolo IP se encarga de:
  - **direccion IP:** Las convenciones de direcciones IP forman parte del protocolo IP. Cómo diseñar un esquema de direcciones IPv4 introduce las direcciones IPv4 y Descripción general de las direcciones IPv6 las direcciones IPv6.
  - **Comunicaciones de host a host :**El protocolo IP determina la ruta que debe utilizar un paquete, basándose en la dirección IP del sistema receptor.
  - **Formato de paquetes:**el protocolo IP agrupa paquetes en unidades conocidas como datagramas. Puede ver una descripción completa de los datagramas en Capa de Internet: preparación de los paquetes para la entrega.
  - **Fragmentación:**Si un paquete es demasiado grande para su transmisión a través del medio de red, el protocolo IP del sistema de envío divide el paquete en fragmentos de menor tamaño. A continuación, el protocolo IP del sistema receptor reconstruye los fragmentos y crea el paquete original.

- **Protocolo ARP :**El protocolo de resolución de direcciones (ARP) se encuentra conceptualmente entre el vínculo de datos y las capas de Internet. ARP ayuda al protocolo IP a dirigir los datagramas al sistema receptor adecuado asignando direcciones Ethernet (de 48 bits de longitud) a direcciones IP conocidas (de 32 bits de longitud).
- **Protocolo ICMP:**El protocolo de mensajes de control de Internet (ICMP) detecta y registra las condiciones de error de la red. ICMP registra:
  - **Paquetes soltados:** Paquetes que llegan demasiado rápido para poder procesarse.
  - **Fallo de conectividad:**No se puede alcanzar un sistema de destino.
  - **Redirección:**Redirige un sistema de envío para utilizar otro enrutador.

Capa de interfaz de red

El software TCP/IP de nivel inferior consta de una capa de interfaz de red responsable de aceptar los datagramas IP y transmitirlos hacia una red específica. Una interfaz de red puede consistir en un dispositivo controlador (por ejemplo, cuando la red es una red de área local a la que las máquinas están conectadas directamente) o un complejo subsistema que utiliza un protocolo de enlace de datos propios (por ejemplo, cuando la red consiste de conmutadores de paquetes que se comunican con anfitriones utilizando HDLC).

Capa de red física

La capa de red física especifica las características del hardware que se utilizará para la red. Por ejemplo, la capa de red física especifica las características físicas del medio de comunicaciones. La capa física de TCP/IP describe los estándares de hardware como IEEE 802.3, la especificación del medio de red Ethernet, y RS-232, la especificación para los conectores estándar.

Desventajas de la estratificacion por capas

La estratificación por capas es una idea fundamental que proporciona las bases para el diseño de protocolos. Permite al diseñador dividir un problema complicado en subproblemas y resolver cada parte de manera

independiente. Por desgracia, el software resultante de una estratificación por capas estrictas puede ser muy ineficaz. Si se considera el trabajo de la capa de transporte, debe aceptar un flujo de octetos desde un programa de aplicación, dividir el flujo en paquetes y enviar cada paquete a través de la red de redes. Para optimizar la transferencia, la capa de transporte debe seleccionar el tamaño de paquete más grande posible que le permita a un paquete viajar en una trama de red. En particular, si la máquina de destino está conectada a una máquina de la misma red de la fuente, solo la red física se verá involucrada en la transferencia, así, el emisor puede optimizar el tamaño del paquete para esta red. Si el software preserva una estricta estratificación por capas, sin embargo, la capa de transporte no podrá saber como ruteará él modulo de Internet él trafico o que redes están conectadas directamente. Mas aun, la capa de transporte no comprenderá el datagrama o el formato de trama ni será capaz de determinar como deben ser añadidos muchos octetos de encabezado a un paquete. Así, una estratificación por capas estricta impedirá que la capa de transporte optimice la transferencia.

Por lo general, las implantaciones atenúan el esquema estricto de la estratificación por capas cuando construyen software de protocolo. Permiten que información como la selección de ruta y la MTU de red se propaguen hacia arriba. Cuando los buffers realizan el proceso de asignación, generalmente dejan espacio para encabezados que serán añadidos por los protocolos de las capas de bajo nivel y pueden retener encabezados de las tramas entrantes cuando pasan hacia protocolos de capas superiores. Tal optimización puede producir mejoras notables en la eficiencia siempre y cuando conserve la estructura básica en capas.

4.1.2. Movimiento de la información

TCP/IP define cuidadosamente cómo se mueve la información desde el remitente hasta el destinatario. En primer lugar, los programas de aplicación envían mensajes o corrientes de datos a uno de los protocolos de la capa de transporte de Internet, UDP (User Datagram Protocol) o TCP (Transmission Control Protocolo). Estos protocolos reciben los datos de la aplicación, los dividen en partes más pequeñas llamadas paquetes, añaden una dirección de destino y, a continuación, pasan los paquetes a la siguiente capa de protocolo, la capa de red de Internet. La capa de red de Internet pone el paquete en un datagrama de IP (Internet Protocol), pone la cabecera y la cola de datagrama, decide dónde enviar el datagrama (directamente a un destino o a una pasarela) y pasa el datagrama a la capa de interfaz de red. La capa de interfaz de red acepta los datagramas IP y los transmite como tramas a través de un hardware de red específico, por ejemplo redes Ethernet o de Red

en anillo.

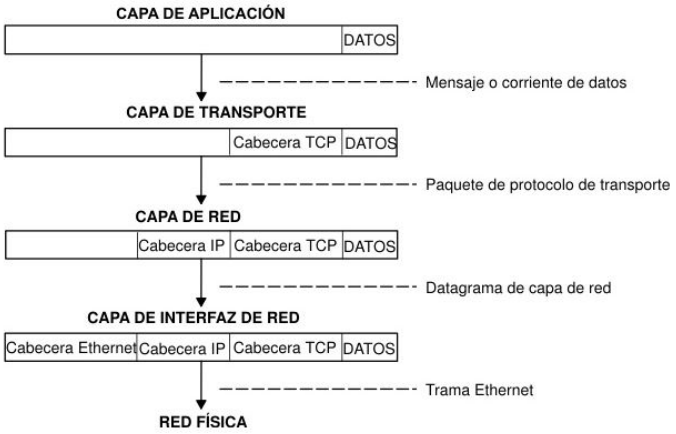


Figura 4.2 – *Movimiento de la información desde la aplicación remitente hasta el sistema principal destinatario.*

La Fig. 4.2 muestra el flujo de información de las capas de protocolo TCP/IP del remitente al host.

Las tramas recibidas por un sistema principal pasan a través de las capas de protocolo en sentido inverso. Cada capa quita la información de cabecera correspondiente, hasta que los datos regresan a la capa de aplicación.

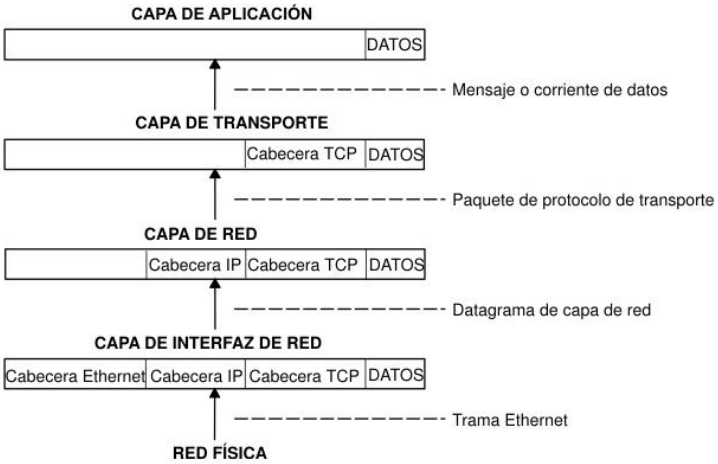


Figura 4.3 – *Movimiento de la información desde el sistema principal hasta la aplicación.*

La Fig. 4.3 muestra el flujo de información de las capas de protocolo TCP/IP desde el sistema principal al remitente. La capa de interfaz de red (en este caso, un adaptador Ethernet) recibe las tramas. La capa de interfaz

de red quita la cabecera Ethernet y envía el datagrama hacia arriba hasta la capa de red. En la capa de red, Protocolo Internet quita la cabecera IP y envía el paquete hacia arriba hasta la capa de transporte. En la capa de transporte, TCP (en este caso) quita la cabecera TCP y envía los datos hacia arriba hasta la capa de aplicación.

4.2. Protocolo HTTP

4.2.1. Generalidades

El Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol), [9], es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP, Fig. 4.4.

Diseñado a principios de la década de 1990, HTTP es un protocolo ampliable, que ha ido evolucionando con el tiempo. Es lo que se conoce como un protocolo de la capa de aplicación, y se transmite sobre el protocolo TCP, o el protocolo encriptado TLS, aunque teóricamente podría usarse cualquier otro protocolo fiable. Gracias a que es un protocolo capaz de ampliarse, se usa no solo para transmitir documentos de hipertexto (HTML), si no que además, se usa para transmitir imágenes o vídeos, o enviar datos o contenido a los servidores, como en el caso de los formularios de datos. HTTP puede incluso ser utilizado para transmitir partes de documentos, y actualizar páginas Web en el acto.

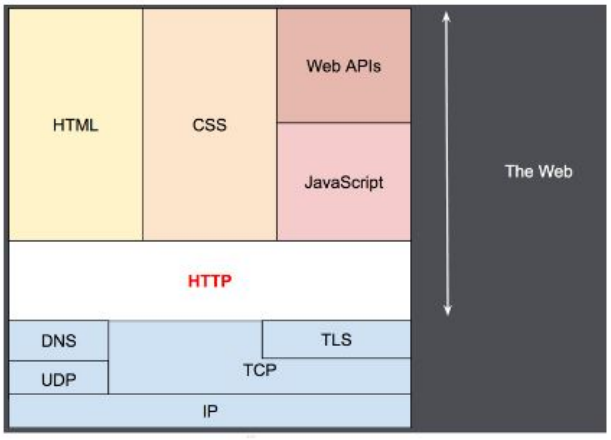


Figura 4.4 – *Protocolo HTTP.*

Clientes y servidores se comunican intercambiando mensajes individuales (en contraposición a las comunicaciones que utilizan flujos continuos de

datos). Los mensajes que envía el cliente, normalmente un navegador Web, se llaman peticiones, y los mensajes enviados por el servidor, se llaman respuestas.

HTTP se basa en sencillas operaciones de solicitud/respuesta, ver Fig. 4.5. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL.

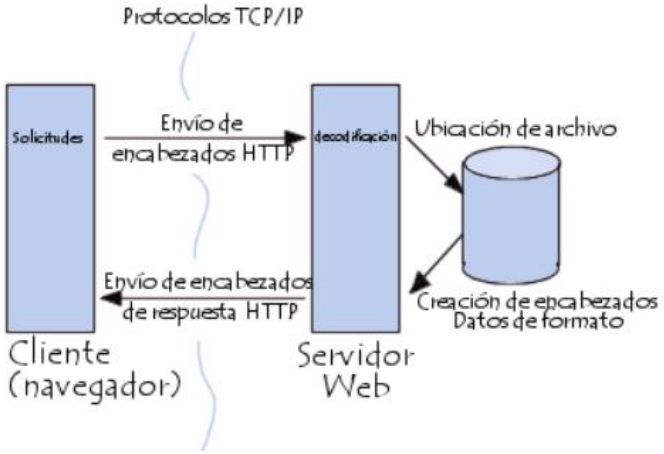


Figura 4.5 – *Comunicación cliente-servidor.*

4.2.2. Arquitectura de los sistemas basados en HTTP

HTTP es un protocolo basado en el principio de cliente-servidor, las peticiones son enviadas por una entidad, el agente del usuario (o un proxy a petición de uno, Fig. 4.6). La mayoría de las veces el agente del usuario (cliente) es un navegador Web, pero podría ser cualquier otro programa, como por ejemplo un programa-robot, que explore la Web, para adquirir datos de su estructura y contenido para uso de un buscador de Internet.

Cada petición individual se envía a un servidor, el cuál la gestiona y responde. Entre cada petición y respuesta, hay varios intermediarios, normalmente denominados proxies, los cuales realizan distintas funciones, como: gateways o caches.

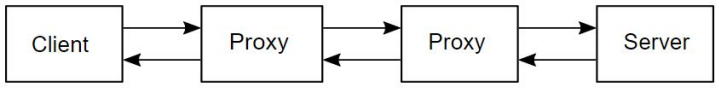


Figura 4.6 – *Agentes en comunicación HTTP.*

En realidad, hay más elementos intermedios, entre un navegador y el servidor que gestiona su petición: hay otros tipos de dispositivos: como routers, modems ... Es gracias a la arquitectura en capas de la Web, que estos intermediarios, son transparentes al navegador y al servidor, ya que HTTP se apoya en los protocolos de red y transporte. HTTP es un protocolo de aplicación, y por tanto se apoya sobre los anteriores. Aunque para diagnosticar problemas en redes de comunicación, las capas inferiores son irrelevantes para la definición del protocolo HTTP .

Cliente

El agente del usuario, es cualquier herramienta que actué en representación del usuario. Esta función es realizada en la mayor parte de los casos por un navegador Web. Hay excepciones, como el caso de programas específicamente usados por desarrolladores para desarrollar y depurar sus aplicaciones.

El navegador es siempre el que inicia una comunicación (petición), y el servidor nunca la comienza (hay algunos mecanismos que permiten esto, pero no son muy habituales).

Para poder mostrar una página Web, el navegador envía una petición de documento HTML al servidor. Entonces procesa este documento, y envía más peticiones para solicitar scripts, hojas de estilo (CSS), y otros datos que necesite (normalmente vídeos y/o imágenes). En navegador, une todos estos documentos y datos, y compone el resultado final: la página Web. Los scripts, los ejecuta también el navegador, y también pueden generar más peticiones de datos en el tiempo, y el navegador, gestionará y actualizará la página Web en consecuencia.

Una página Web, es un documento de hipertexto (HTTP), luego habrá partes del texto en la página que puedan ser enlaces (links) que pueden ser activados (normalmente al hacer click sobre ellos) para hacer una petición de una nueva página Web, permitiendo así dirigir su agente de usuario y navegar por la Web. El navegador, traduce esas direcciones en peticiones de HTTP, e interpretara y procesará las respuestas HTTP, para presentar al usuario la página Web que desea.

Servidor Web

Al otro lado del canal de comunicación, está el servidor, el cual "sirve"los datos que ha pedido el cliente. Un servidor conceptualmente es una unica entidad, aunque puede estar formado por varios elementos, que se reparten la carga de peticiones, (load balancing), u otros programas, que gestionan otros computadores (como cache, bases de datos, servidores de correo electrónico, ...), y que generan parte o todo el documento que ha sido pedido.

Un servidor no tiene que ser necesariamente un único equipo físico, aunque si que varios servidores pueden estar funcionando en un único computador. En el estándar HTTP/1.1 y Host , pueden incluso compartir la misma dirección de IP.

Proxies

Entre el cliente y el servidor, además existen distintos dispositivos que gestionan los mensajes HTTP. Dada la arquitectura en capas de la Web, la mayoria de estos dispositivos solamente gestionan estos mensajes en los niveles de protocolo inferiores: capa de transporte, capa de red o capa física, siendo así transparentes para la capa de comunicaciones de aplicación del HTTP, además esto aumenta el rendimiento de la comunicación. Aquellos dispositivos, que sí operan procesando la capa de aplicación son conocidos como proxies. Estos pueden ser transparentes, o no (modificando las peticiones que pasan por ellos), y realizan varias funciones:

- caching (la caché puede ser pública o privada, como la caché de un navegador).
- filtrado (como un anti-virus, control parental, etc).
- balanceo de carga de peticiones (para permitir a varios servidores responder a la carga total de peticiones que reciben).
- autentificación (para el control al acceso de recursos y datos).
- registro de eventos (para tener un histórico de los eventos que se producen).

4.2.3. Características clave del protocolo HTTP

Sencillo

Incluso con el incremento de complejidad, que se produjo en el desarrollo de la versión del protocolo HTTP/2, en la que se encapsularon los mensajes,

HTTP esta pensado y desarrollado para ser leído y fácilmente interpretado por las personas, haciendo de esta manera más facil la depuración de errores, y reduciendo la curva de aprendizaje para las personan que empieza a trabajar con él.

Extensible

Presentadas en la versión HTTP/1.0, las cabeceras de HTTP, han hecho que este protocolo sea fácil de ampliar y de experimentar con él. Funcionalidades nuevas pueden desarrollarse, sin más que un cliente y su servidor, comprendan la misma semántica sobre las cabeceras de HTTP.

Protocolo sin estados

HTTP es un protocolo sin estado, es decir: no guarda ningún dato entre dos peticiones en la mísma sesión. Esto plante la problemática, en caso de que los usuarios requieran interactuar con determinadas páginas Web de forma ordenada y coherente, por ejemplo, para el uso de "cestas de la compra.[en] páginas que utilizan en comercio electrónico. Pero, mientras HTTP ciertamente es un protocolo sin estado, el uso de HTTP cookies, si permite guardar datos con respecto a la sesión de comunicación. Usando la capacidad de ampliación del protocolo HTTP, las cookies permiten crear un contexto común para cada sesión de comunicación.

Conexiones

Una conexión se gestiona al nivel de la capa de trasporte, y por tanto queda fuera del alcance del protocolo HTTP. Aún con este factor, HTTP no necesita que el protocolo que lo sustenta mantenga una conexión continua entre los participantes en la comunicación, solamente necesita que sea un protocolo fiable o que no pierda mensajes (como mínimo, en todo caso, un protocolo que sea capaz de detectar que se ha pedido un mensaje y reporte un error). De los dos protocolos más comunes en Internet, TCP es fiable, mientras que UDP, no lo es. Por lo tanto HTTP, se apoya en el uso del protocolo TPC, que está orientado a conexión, aunque una conexión continua no es necesaria siempre.

En la versión del protocolo HTTP/1.0, habría una conexión TPC por cada petición/respuesta intercambiada, presentando esto dos grandes inconvenientes: abrir y crear una conexión requiere varias rondas de mensajes y por lo tanto resultaba lento. Esto sería más eficiente si se mandaran varios

mensajes.

Para atenuar estos inconvenientes, la versión del protocolo HTTP/1.1 presentó el 'pipelining' y las conexiones persistentes: el protocolo TPC que lo transmitía en la capa inferior se podía controlar parcialmente, mediante la cabecera 'Connection'. La versión del protocolo HTTP/2 fue más allá y usa multiplexación de mensajes sobre un única conexión, siendo así una comunicación más eficiente.

Todavía hoy se sigue investigando y desarrollando para conseguir un protocolo de transporte más conveniente para el HTTP. Por ejemplo, Google está experimentado con QUIC, que se apoya en el protocolo UDP y presenta mejoras en la fiabilidad y eficiencia de la comunicación.

4.2.4. ¿Qué se puede controlar con HTTP?

La característica del protocolo HTTP de ser ampliable, ha permitido que durante su desarrollo se hayan implementado más funciones de control y funcionalidad sobre la Web: caché o métodos de identificación o autentificación fueron temas que se abordaron pronto en su historia. Al contrario la relajación de la restricción de origen solo se ha abordado en los años de la década de 2010.

Se presenta a continuación una lista con los elementos que se pueden controlar con el protocolo HTTP:

- **Cache:** El como se almacenan los documentos en la caché, puede ser especificado por HTTP. El servidor puede indicar a los proxies y clientes, que quiere almacenar y durante cuanto tiempo. Aunque el cliente, también puede indicar a los proxies de caché intermedios que ignoren el documento almacenado.

- **Flexibilidad del requisito de origen:**Para prevenir invasiones de la privacidad de los usuarios, los navegadores Web, solamente permiten a páginas del mismo origen, compartir la información o datos. Esto es una complicación para el servidor, asi que mediante cabeceras HTTP, se puede flexibilizar o relajar esta división entre cliente y servidor.

- **Autentificación:**Hay páginas Web, que pueden estar protegidas, de manera que solo los usuarios autorizados puedan acceder. HTTP provee de servicios básicos de autentificación, por ejemplo mediante el uso

de cabeceras como: WWW-Authenticate, o estableciendo una sesión especifica mediante el uso de HTTP cookies.

- **Proxies y tunneling:**Servidores y/o clientes pueden estar en intranets y esconder así su verdadera dirección IP a otros. Las peticiones HTTP utilizan los proxies para acceder a ellos. Pero no todos los proxies son HTTP proxies. El protocolo SOCKS, por ejemplo, opera a un nivel más bajo. Otros protocolos, como el FTP, pueden ser servidos mediante estos proxies.

- **Sesiones:**El uso de HTTP cookies permite relacionar peticiones con el estado del servidor. Esto define las sesiones, a pesar de que por definición el protocolo HTTP es un protocolo sin estado. Esto es muy útil no sólo para aplicaciones de comercio electrónico, sino también para cualquier sitio que permita configuración al usuario.

4.2.5. Flujo de HTTP

Cuando el cliente quiere comunicarse con el servidor, tanto si es directamente con él, o a través de un proxy intermedio, realiza los siguientes pasos:

1. Abre una conexión TPC: la conexión TPC se usará para hacer una petición, o varias, y recibir la respuesta. El cliente pude abrir una conexión nueva, reusar una existente, o abrir varias a la vez hacia el servidor.

2. Hacer una petición HTTP: Los mensajes HTTP (previos a HTTP/2) son legibles en texto plano. A partir de la versión del protocolo HTTP/2, los mensajes se encapsulan en franjas, haciendo que no sean directamente interpretables, aunque el principio de operación es el mismo.

```
1 GET / HTTP/1.1
2 Host: developer.mozilla.org
3 Accept-Language: fr
```

Figura 4.7 – *Petición HTTP.*

3. Leer la respuesta enviada por el servidor:

```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html

<!DOCTYPE html... (here comes the 29769 bytes of the requested web page)
```

Figura 4.8 – *Respuesta HTTP.*

4. Cierre o reuso de la conexión para futuras peticiones.

Si está activado el HTTP pipelining, varias peticiones pueden enviarse sin tener que esperar que la primera respuesta haya sido satisfecha. Este procedimiento es difícil de implementar en las redes de computadores actuales, donde se mezclan software antiguos y modernos. Así que el HTTP pipelining ha sido substituido en HTTP/2 por el multiplexado de varias peticiones en una sola trama.

4.2.6. Mensajes HTTP

En las versiones del protocolo HTTP/1.1 y anteriores los mensajes eran de formato texto y eran totalmente comprensibles directamente por una persona. En HTTP/2, los mensajes estan estructurados en un nuevo formato binario y las tramas permiten la compresión de las cabeceras y su multiplexación. Así pues, incluso si solamente parte del mensaje original en HTTP se envía en este formato, la sematica de cada mensaje es la misma y el cliente puede formar el mensaje original en HTTP/1.1. Luego, es posible interpretar los mensajes HTTP/2 en el formato de HTTP/1.1.

Existen dos tipos de mensajes HTTP: peticiones y respuestas, cada uno sigue su propio formato.

Peticiones

Un ejemplo de petición HTTP puede verse en la Fig. 4.9:

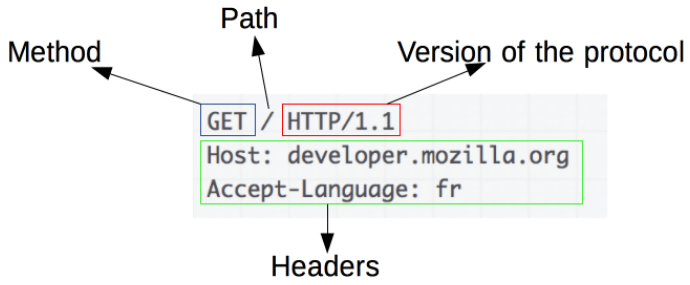


Figura 4.9 – *Ejemplo petición web.*

Una petición de HTTP, está formado por los siguientes campos:

- Un método HTTP, normalmente pueden ser un verbo, como: GET, POST o un nombre como: OPTIONS o HEAD, que defina la operación que el cliente quiera realizar. El objetivo de un cliente, suele ser una petición de recursos, usando GET, o presentar un valor de un formulario HTML, usando POST, aunque en otras ocasiones puede hacer otros tipos de peticiones.
- La dirección del recurso pedido; la URL del recurso, sin los elementos obvios por el contexto, como pueden ser: sin el protocolo (http://), el dominio (aquí developer.mozilla.org), o el puerto TCP (aquí el 80).
- La versión del protocolo HTTP.
- Cabeceras HTTP opcionales, que pueden aportar información adicional a los servidores.
- O un cuerpo de mensaje, en algún método, como puede ser POST, en el cual envía la información para el servidor.

Lista de comandos HTTP puede verse en la Fig. 4.10:

Comando	Descripción
GET	Solicita el recurso ubicado en la URL especificada
HEAD	Solicita el encabezado del recurso ubicado en la URL especificada
POST	Envía datos al programa ubicado en la URL especificada
PUT	Envía datos a la URL especificada
DELETE	Borra el recurso ubicado en la URL especificada

Figura 4.10 – *Comandos.*

Respuestas

Un ejemplo de repuesta puede verse en la Fig. 4.11:

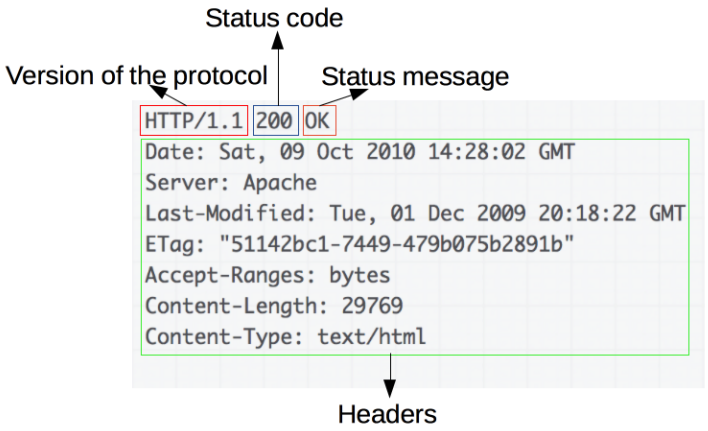


Figura 4.11 – *Ejemplo respuesta web.*

Las respuestas están formadas por los siguientes campos:

- La versión del protocolo HTTP que están usando.
- Un código de estado, indicando si la petición ha sido exitosa, o no, y debido a qué.
- Un mensaje de estado, una breve descripción del código de estado.
- Cabeceras HTTP, como las de las peticiones.
- Opcionalmente, el recurso que se ha pedido.

4.2.7. Conclusión

El protocolo HTTP es un protocolo ampliable y fácil de usar. Su estructura cliente-servidor, junto con la capacidad para usar cabeceras, permite a este protocolo evolucionar con las nuevas y futuras aplicaciones en Internet.

Aunque la versión del protocolo HTTP/2 añade algo de complejidad, al utilizar un formato en binario, esto aumenta su rendimiento, y la estructura y semántica de los mensajes es la misma desde la versión HTTP/1.0. El flujo de comunicaciones en una sesión es sencillo y puede ser fácilmente estudiado e investigado con un simple monitor de mensajes HTTP.

4.3. Lenguaje de programación html

4.3.1. Introducción

El lenguaje de marcas de hipertexto, HTML o (HyperText Markup Language) se basa en el metalenguaje SGML (Standard Generalized Markup Language) y es el formato de los documentos de la World Wide Web, Fig. 4.12. El World Wide Web Consortium (W3C) es la organización que desarrolla los estándares para normalizar el desarrollo y la expansión de la Web y la que publica las especificaciones relativas al lenguaje HTML.

Figura 4.12 – *Lenguaje HTML.*

HTML fue concebido como un lenguaje para el intercambio de documentos científicos y técnicos adaptado para su uso por no especialistas en tratamiento de documentos. HTML resolvió el problema de la complejidad de SGML sirviéndose de un reducido conjunto de etiquetas estructurales y semánticas apropiadas para la realización de documentos relativamente simples. Pero, además de simplificar la estructura de los documentos, HTMLhtml soportaba el hipertexto.

En un corto período de tiempo, HTML se hizo muy popular y rápidamente superó los propósitos para los que había sido creado. Desde sus inicios, ha habido una constante invención de nuevos elementos para usarse dentro de HTML como estándar y para adaptar HTML a las nuevas posibilidades de la Web, como la posibilidad de usar elementos multimedia o la utilización de elementos dinámicos (animaciones Java, uso de Flash, controles ActiveX,

etc. que hacen las páginas web mucho más llamativas e interactivas para el usuario. Sin embargo, esta ampliación de nuevos elementos también ha traído problemas de compatibilidad de los documentos entre las distintas plataformas y programas.

El lenguaje HTML nace en 1991 de manos de Tim Bernes-Lee del CERN como un sistema hipertexto con el único objetivo de servir como medio de transmisión de información entre los científicos que se ocupaban de la Física de alta energía ,como parte de la iniciativa World Wide Web. Así pues, HTML tuvo lugar a la par que el origen de la Web, ya que se trata del lenguaje que sirve para crear páginas web. En 1993 Dan Connelly escribe la primera DTD (Document Type Definition) de SGML describiendo el lenguaje y, desde entonces, el lenguaje HTML ha estado sometido a incesantes cambios. De hecho, han existido distintas versiones: 1.0 (en 1993), 2.0 (en 1995), 3.0 (en 1995), 3.2 (en 1997), 4.0 (en 1997, revisada en 1998).

Ya en 1994 el sistema había tenido tal aceptación que la especificación se había quedado obsoleta. Por aquel entonces WWW y Mosaic eran casi sinónimos debido a que el navegador Mosaic del NCSA (National Center for Supercomputing Applications) era el más extendido gracias a las mejoras que incorporaba. Es entonces cuando nace el HTML 2.0 en una especificación también realizada por Dan Connelly. El crecimiento exponencial que comienza a sufrir el sistema lleva a organizar la "First International WWW Conference.^{en} mayo de 1994. El principal avance de 2.0 de HTML es la incorporación de los llamados "forms.^{o} formularios que permiten que el usuario cliente envíe información al servidor y ésta sea recogida y procesada allí. Precisamente con este fin, NCSA presenta la especificación del CGI, Common Gateway Interface, versión 1.0 que define una interfaz entre programas ejecutables y el sistema WWW. Desde entonces, el lenguaje HTML ha seguido creciendo como algo dinámico hasta llegar no sólo a la especificación HTML 4.01 en septiembre de 2001 (http://www.w3.org/TR/html401/) -la última versión está disponible en el sitio web del W3 Consortium en la dirección (http://www.w3.org/TR/1998/REC-html40-19980424)-, sino también a otros lenguajes de transición como XHTML que es una reformulación de HTML como aplicación XML.

La norma ISO/IEC 15445:2000 recoge el lenguaje HTML estandarizado (http://purl.org/NET/ISO+IEC.15445/15445.html). Roger Price y David Abrahamson han escrito una guía al respecto: la User's

guide to ISO/IEC 15445:2000 HyperText Markup Language (HTML) (http://purl.org/NET/ISO+IEC.15445/Users-Guide.html) donde establecen de forma pormenorizada la descripción de documentos y el procesamiento del lenguaje HTML.

4.3.2. Generalidades

Los documentos HTML son archivos de texto plano (también conocidos como ASCII) que pueden ser creados mediante cualquier editor de texto, aunque también existen programas específicos para editar HTML (los editores más conocidos son Microsoft FrontPage, Netscape Composer, Macromedia Dreamweaver y Adobe PageMill), concebidos específicamente para editar páginas web en HTML.

HTML no permite definir de forma estricta la apariencia de una página, aunque en la práctica, se utiliza también como un lenguaje de presentación. Los archivos de HTML se leen en un navegador web tal como Chrome, Microsoft Explorer, Mozilla, etc. La presentación de la página es muy dependiente del navegador o browser utilizado ya que el mismo documento no produce el mismo resultado en la pantalla si se visualiza con uno u otro, o sea, HTML se limita a describir la estructura y el contenido de un documento, y no el formato de la página y su apariencia.

Una de las claves del éxito de la World Wide Web, aparte de lo atractivo de su presentación es, sin duda, su organización y coherencia. Todos los documentos WWW comparten un mismo aspecto y una única interfaz, lo que facilita enormemente su manejo por parte de cualquier persona. Esto es posible porque el lenguaje HTML no sólo permite establecer enlaces entre diferentes documentos, sino que es un lenguaje de descripción de página independiente de la plataforma en que se utilice. Es decir un documento HTML contiene toda la información necesaria sobre su aspecto y su interacción con el usuario, y es luego el navegador que utilicemos el responsable de asegurar que el documento tenga un aspecto coherente, independientemente del tipo de ordenador o de estación de trabajo desde donde estemos efectuando la consulta.

Los archivos HTML tienen la extensión .html ó htm y para ver la estructura de una página web en lenguaje HTML, los navegadores suelen disponer de un menú con la opción “Ver”desde la que se puede visualizar el código fuente de la página HTML. Dicho código fuente nos dará una idea clara de en qué consiste este lenguaje que, como hemos dicho anteriormente,

es un simple lenguaje de marcas entre cuyas funciones destaca la posibilidad de enlazar documentos y partes de documentos, esto es, la hipertextualidad.

Así pues, existen dos herramientas fundamentales e imprescindibles asociadas al lenguaje HTML, por un lado, los editores HTML (para crear documentos HTML) y, por otro, los navegadores (para visualizar dichos documentos). Aunque también existen otras herramientas automatizadas para generar páginas web, como son los conversores desde otros formatos y otro tipo de herramientas como los revisores y validadores que nos permiten analizar los documentos HTML ya creados para ver si se ajustan a los parámetros de este lenguaje.

4.3.3. Creación de documentos HTML

Para crear documentos HTML sólo es necesario :

- **Un procesador de textos o un editor de documentos HTML.**
- **Un navegador web.**

Editores

Los documentos HTML están en formato de texto plano (también conocido como ASCII), por eso se puede utilizar un simple procesador de textos para escribir un documento en lenguaje HTML. Este archivo será posteriormente interpretado por el programa navegador correspondiente, siempre que el documento esté guardado en formato: "sólo texto". Una vez creado el documento HTML, lo guardaremos con la extensión .html ó htm.

Se puede usar el Bloc de notas de Windows o cualquier otro editor de textos sencillo. Sin embargo, hay que tener cuidado con algunos editores más complejos puesto que formatean el texto y colocan su propio código especial al guardar las páginas y HTML debe ser únicamente texto plano, con lo que podríamos tener problemas. Si creamos un documento HTML en un editor de texto que formatee dicho texto, debemos guardar el documento como documento de texto sin formato.

El texto escrito tiene dos partes bien diferenciadas, el contenido de la información y el conjunto de etiquetas del lenguaje HTML, que permiten determinar el estilo y el tipo de letra que tendrá la presentación del documento final y que pueda ser leído por un programa cliente.

Para escribir un párrafo sin estilo específico (por defecto) no es necesario poner etiqueta alguna. Lo único que hay que tener en cuenta es que al presentar el documento se hace caso omiso de los espacios, tabulaciones y retornos de carro que se encuentren en el texto fuente. Por esta razón se utilizan una serie de etiquetas que sustituyen a estos elementos. El texto escrito no sufrirá ninguna modificación, exceptuando los acentos, la letra "ñz un conjunto de caracteres especiales.

En la actualidad no se precisa conocer ni el lenguaje SGML ni HTML en sí mismo a la hora de crear un hipertexto en la Web, pues los editores de HTML actuales simplifican enormemente la labor y ya no hay que trabajar con texto plano y etiquetas en código HTML, sino que los programas de edición ofrecen la posibilidad de disponer y ver la página mientras se crea y de insertar los comandos HTML y sus atributos de forma automática. Además, se ha generalizado la interfaz CGI o Common Gateway Interface, que posibilita que el navegador pueda trabajar conjuntamente con el servidor.

Los editores permiten también la inserción de elementos multimedia: imágenes, audio, vídeo, javascripts, etc; e incluso ofrecen plantillas de páginas con modelos de páginas personales o corporativas, diseño con marcos o bordes compartidos, índices, mapas del web, etc) y estilos predeterminados para dar homogeneidad a todo un sitio web. Los programas más recientes se orientan no sólo al formato físico de la WWW, sino también a su estructura y de esta forma es posible explorar y jerarquizar los nodos, mostrar las relaciones entre ellos, indicar la localización de los puntos de origen y destino de cada enlace, crear de forma automática tablas de contenido del web, índices, etc.

En la actualidad se han desarrollado un gran número de sistemas de gestión de hipertextos, ya sea para hipertextos independientes o para crear y gestionar hipertextos abiertos a la web y muchos de ellos son gratuitos. Sin embargo, los editores web que ofrecen mayor número de funcionalidades, suelen ser de pago. Entre los más conocidos destacan Hot Dog ([10]), Microsoft FrontPage ([11]) el editor de Microsoft y con el que se ha diseñado este hipertexto, DreamWeaver ([12]), Adobe GoLive ([13]), pero existen un gran número y una gran variedad de editores web que emplean el lenguaje HTML.

Sin embargo, el uso de un editor de páginas web presenta algunas desventajas en relación al uso de editores de texto sin formato. Por un lado,

se depende de un software propietario que suele estar protegido por derechos de autor y que no presenta una total compatibilidad entre navegadores y, por otro, al utilizar este tipo de editores se produce una gran número de las llamadas etiquetas meta o metaetiquetas que suelen ser redundantes. Estas etiquetas meta que aparecen en la parte superior de la página y que proporcionan información acerca de la página, pero que no afectan a su apariencia, son las que incluyen los metadatos que posteriormente facilitarán la búsqueda y recuperación de los documentos por parte de los robots y motores de búsqueda. Por ejemplo, la metaetiqueta generator indica el tipo de editor con el que se creó la página HTML.

Por otro lado, también es posible crear documentos con editores de texto sin formato y después modificarlos con un editor HTML, o bien realizar la operación contraria, con el fin combinar lo mejor de estos dos sistemas. Y además tenemos la posibilidad de crear un documento en un formato determinado y transformar dicho documento al formato HTML por medio de un conversor.

Los editores HTML se clasifican en 3 tipos:

- **Wysywyg:** (what you see is what you get -lo que ves es lo que obtienes-): son editores que muestran en pantalla de forma inmediata lo que se va creando. Son muy útiles para apreciar los colores y la disposición de los elementos.
- **No Wywywyg:**editores que necesitan una aplicación externa (el navegador o browser) para mostrar lo que se va creando. Son útiles para recordar los comandos HTML y sus atributos.
- **Mixtos:**presentan la pantalla dividida en dos ventanas, una con el código HTML y otra con el resultado final, o permiten la visualización de ambos en ventanas diferentes que se pueden visualizar de forma alternativa.

Conversores

Se trata de herramientas que permiten transformar un documento escrito en otro formato al formato de un documento HTML. La mayoría de los editores de HTML cuentan con herramientas de este tipo. Su utilización es muy eficaz cuando se ha realizado un gran trabajo previo en otro sistema diferente. Para usarlos, se debe simplificar al máximo el formato documento (eliminar las sangrías, uniformar los tamaños y tipos de letra, etc.) para luego

pasar el conversor y generar así un documento HTML. También es posible convertir muchos hipertextos creados con herramientas independientes, para abrirlos a la Web al ser transformados en documentos HTML.

Revisores y validadores

Uno de los problemas que ha acompañado al lenguaje HTML es la diversidad de navegadores, ya que no todos son capaces de interpretar un mismo código de una manera unificada. Esto obliga al diseñador de documentos HTML a que, una vez creada su página, compruebe que ésta puede ser leída satisfactoriamente por todos los navegadores, o al menos, por los más utilizados.

Existen una serie de herramientas llamadas revisores o validadores que permiten verificar si una página web se ajusta a las recomendaciones o especificaciones del W3C World Wide Web Consortium. Así, podemos comprobar si dicha página se ajusta al estándar HTML y a otros estándares complementarios, y si cumplen o no con las normas estandarizadas que hacen posible la accesibilidad.

4.3.4. Estructura de un documento HTML

Básicamente, HTML utiliza los mismos elementos que SGML, esto es, marcas o etiquetas compuestas de códigos enmarcados por los signos ¡¿. Cualquier documento HTML comienza con la etiqueta ¡html¿y termina con la etiqueta ¡/html¿. Dentro existen dos zonas bien identificadas: el encabezamiento, que se identifica con la etiqueta ¡HEAD¿y ¡/HEAD¿y sirve para definir una serie de valores válidos en todo el documento, y el cuerpo del documento, representado por etiqueta ¡BODY¿y ¡/BODY¿que muestra la información del documento.

Dentro del encabezamiento, la etiqueta principal es la del título, representado por la etiqueta ¡TITLE¿En el cuerpo del documento se pueden establecer diferentes categorías utilizando diversos tamaños de fuentes y estableciendo así la estructura básica del documento. Dentro del cuerpo se coloca el contenido de la página web: textos, imágenes, tablas, formularios, gráficos, mapas sensibles, etc. La información se puede estructurar en párrafos y/ o secciones con cabeceras de distinto nivel, enlaces a otras partes del mismo documento, a documentos distintos, o a documentos que estén en distinto servidor, etc.

También podemos determinar el estilo y tipo de letra mediante los atributos, como son los de final de línea ¡br¿y final de párrafo ¡p¿, los de presentación de texto preformateado, es decir, manteniendo los espacios y retornos de carro ¡pre¿o los de negrita ¡b¿, cursiva ¡i¿, centrado ¡center¿, etc. Además se pueden definir listas numeradas y no numeradas, glosarios, trazar líneas horizontales para separar textos, insertar comentarios, etc.

Pero, sin duda, los elementos más característicos del lenguaje HTML y que dotan a los documentos con él creados de la característica hipertextual, son las anclas o anclajes utilizados para establecer los enlaces, ya se trate de texto o gráficos. Para fijar un enlace, se debe enmarcar el texto que aparecerá subrayado para que el usuario pueda pulsarlo y le redirija a la dirección o URL marcada. La etiqueta ancla debe incluir el parámetro href=“URLz el texto para pulsar. Si se trata de imágenes, éstas se insertan mediante la etiqueta src=“URLz también podemos conseguir que el gráfico se convierta en un enlace mediante la etiqueta correspondiente.

Existen una gran cantidad de etiquetas, muchas de ellas pueden contener elementos y atributos que realizarán una función u otra según la voluntad del autor.

La estructura de un documento HTML es la siguiente:
¡HTML¿
¡HEAD¿
¡TITLE¿Título de la página¡/TITLE¿
¡/HEAD¿
¡BODY¿
[Aquí se sitúan otras etiquetas que hacen posible visualizar la página]
¡/BODY¿
¡/HTML¿

El texto puede tener unas cabeceras, comprendidas entre las etiquetas ¡H1¿y ¡/H1¿, ¡H2¿y ¡/H2¿, etc. (hasta el número 6), siendo el número indicativo del tamaño. El tamaño mayor es el correspondiente al número 1.

Hay otras etiquetas como ¡P¿, para separar los distintos párrafos, la etiqueta de centrado ¡CENTER¿y ¡/CENTER¿que sirve para centrar todo lo que esté dentro de ella, ya sea texto, imágenes, etc. la etiqueta ¡HR¿para obtener una raya horizontal tan ancha como la pantalla, y con la apariencia de estar embutida sobre el fondo, etc.

HTML no reconoce los finales de línea del editor de texto, pero la etiqueta ¡BR¿desplaza el texto a la línea siguiente, y la etiqueta ¡P¿también lo desplaza, dejando una línea de separación.

Títulos

Mediante los títulos, en sus diferentes niveles de importancia, se puede definir el esqueleto del documento o estructura básica. HTML tiene 6 niveles de cabeceras numeradas del uno al seis. En la Fig. 4.13 se muestra una tabla con ejemplos.

Código HTML	*Visualización*
Cabecera tipo 1	Cabecera tipo 1
Cabecera tipo 2	Cabecera tipo 2
Cabecera tipo 3	Cabecera tipo 3
Cabecera tipo 4	Cabecera tipo 4
Cabecera tipo 5	Cabecera tipo 5
Cabecera tipo 6	Cabecera tipo 6

Figura 4.13 – *Diferentes formatos de títulos.*

Tipos de letras

Los tipos básicos son negrita, cursiva y teletipo o máquina de escribir, que utilizan los códigos B, I, TT, respectivamente, como demuestra la siguiente Fig. 4.14:

Código HTML	Visualización
letra <B>negrita</B>	letra **negrita**
letra <I>cursiva</I>	letra *cursiva*
letra <TT>teletipo</TT>	letra `teletipo`

Figura 4.14 – *Tipos de letras.*

Para centrar texto (o, en general, para centrar cualquier otra cosa: un gráfico, por ejemplo) se usa la etiqueta ¡center¿.

Listas

Las listas se definen de forma muy sencilla: se dice dónde empieza la lista, dónde empieza cada punto y dónde acaba la lista. Las etiquetas que se utilicen en cada caso deben aparecer al principio de línea, o al menos sin texto por delante (sólo espacios o tabulaciones). Las listas se pueden anidar, es decir, en el lugar donde debería ir uno de los términos de la lista se pone una nueva lista, que no tiene porqué ser del mismo tipo, Fig. 4.15.

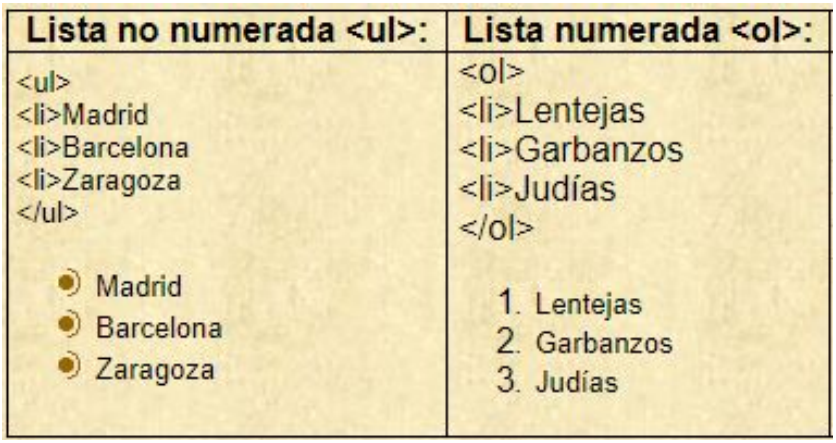


Figura 4.15 – *Tipos de listas.*

Formularios

Los formularios permiten que el usuario introduzca información (seleccionando ítems o directamente escribiendo en los lugares indicados) que será procesada por scripts asociados al servidor de acuerdo con la especificación CGI ('Common Gateway Interface'). Elementos como FORM, INPUT, SELECT, TEXTAREA y sus atributos permiten diseñar formularios que rellenará el usuario. Sin embargo, su funcionalidad no deviene del HTML sino de los scripts ejecutables que residen en el servidor y que reciben la información introducida por el usuario. El uso de formularios está muy extendido en las páginas web sobre todo por su utilización como interfaz de entrada para realizar las búsquedas.

Graficos

Para incluir una imagen o gráfico en un documento HTML se utiliza la etiqueta ¡IMG¿. En dicha etiqueta debe incluirse un parámetro SRC=“URL”, con el cual se indica dónde está el archivo con el gráfico concreto que se quiere para el documento. Esto permite una gran flexibilidad, ya que se puede complementar el contenido del documento tanto con gráficos que se encuentren disponibles en el propio servidor de WWW, como con una imagen situada en un servidor externo, sin que el lector final tenga por qué apreciar ninguna diferencia.

Existen algunas limitaciones respecto a los formatos gráficos que los navegadores o lectores de HTML puede interpretar sin problemas. Los formatos de archivo más utilizados son GIF y JPEG, pero existen otros muchos.

Hay un parámetro optativo de la directiva ¡IMG¿que sirve para proponer un texto alternativo a un gráfico. Este texto aparecerá cuando se esté usando un programa sin capacidades gráficas para leer el lenguaje HTML. Se trata de alt=“texto”. Conviene utilizarlo cuando los gráficos sirven como origen a hiperenlaces, porque si no los programas sin capacidades gráficas no podrán mostrar los enlaces establecidos.

4.4. Wifeed Web

4.4.1. Introducción

La interfaz web es la segunda opción para controlar el sistema y poder visualizar su estado. Esta interfaz básicamente es una página web a la cual se puede acceder desde cualquier navegador web. La misma cuenta con la posibilidad de ejercer el mismo control que la aplicación móvil, como son la de programar los horarios de alimentación, definir tamaño de la porción de alimento, abrir y cerrar puertas, recargar bebedero de agua, hacer una entrega de alimento en el momento deseado de forma espontanea, poder monitorear la cantidad de alimento que hay en la bandeja de alimentación y el restante en la tolva de almacenaje, como también tener presente cuantos días de abastecimiento de alimento queda para entregar comida, entre otras cosas. Con la diferencia de que no se podrán recibir alertas en forma espontanea ya que al cerrar el navegador se pierde la comunicación con el dispositivo principal. El desarrollo de la misma fue realizado por medio del lenguaje de programación HTML y se le agrego el diseño y estilo con código CSS.

Accediendo vía web también es posible realizar la configuración de la conexión WiFi. Que como en el caso de la aplicación Android, se presenta una lista con las redes disponibles en el área del dispositivo y el usuario selecciona su red e introduce la contraseña.
Si bien esta interfaz web cumple el mismo objetivo que la móvil, teniendo la ventaja de no tener que instalar ninguna aplicación, hay que aclarar que la aplicación Android es mucho más cómoda y especifica para poder controlar el sistema.

4.4.2. Configuración WiFi

Como se explicó anteriormente en esta tesis, para indicarle al sistema a qué red se debe conectar mediante la interfaz web, lo primero que el usuario

deberá hacer es una comunicación inicial con el dispositivo principal, esta se logra conectando su computadora a la red WiFi generada por el dispositivo *Wifeed*. Una vez conectado a dicha red, el usuario podrá, desde un navegador, efectuar la primera comunicación a través de la dirección IP propia del dispositivo que esta haciendo las veces de punto de acceso. Esta dirección sera siempre la misma, se prefijo durante el desarrollo y programación del microcontrolador, la cual es 192.168.4.1.
Al introducir como URL esta dirección, el navegador mostrará una primera pantalla para poder realizar la configuración WiFi, Fig. 4.16.

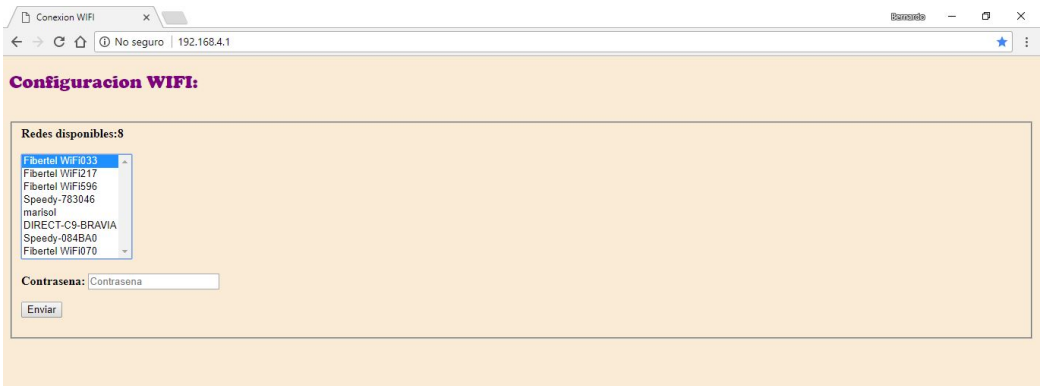


Figura 4.16 – *Redes disponibles.*

Este listado de redes se obtiene a través de un escaneo de las redes cercanas que se ejecuta en el modulo WiFi, estas redes se almacenan en una matriz junto a su nivel de intensidad de señal, MAC del dispositivo que la genera, y canal de comunicación. Luego, éstas se ordenan en la lista que el navegador muestra en orden de mayor intensidad de señal a menor intensidad.

Al seleccionar la red e ingresar la contraseña, el navegador le envía estos datos al servidor alojado en el micro, este los lee e intenta conectarse a a red, si la contraseña es incorrecta, el servidor responde nuevamente con el listado de redes e indica que la contraseña es incorrecta y que se vuelva a ingresar, Fig. 4.17.

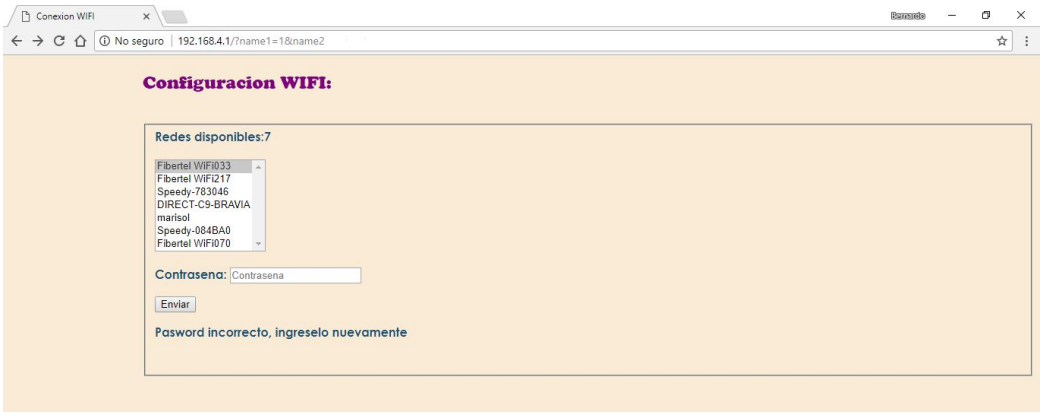


Figura 4.17 – *Contraseña incorrecta.*

En caso de lograr conexión exitosa con la red seleccionada, el micro ya teniendo acceso a Internet, consulta la dirección de IP pública y se la muestra en el navegador, Fig. 4.18.

Figura 4.18 – *Conexión exitosa.*

4.4.3. Diseño

Como ya se mencionó, las funcionalidades son las mismas que las de la aplicación móvil ya explicadas, es por esto que en esta sección sólo se mostrará la presentación gráfica de la interfaz web y no se detendrá con explicaciones sobre cada función.

Una vez la unidad central conectada a la red local, al ingresar a la dirección IP de control otorgada, se mostrará una página de bienvenida, esta hace las veces de presentación y de pantalla principal, Fig. 4.19.

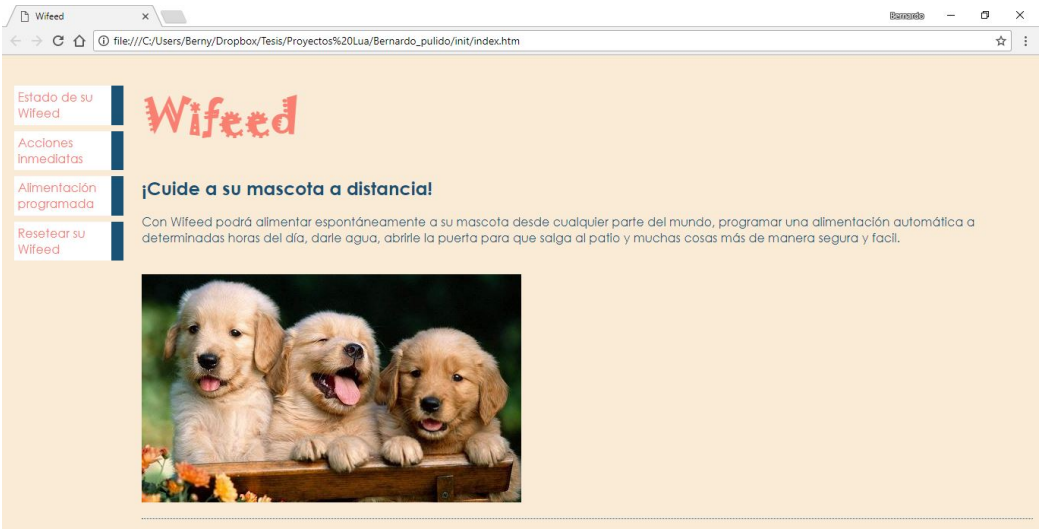


Figura 4.19 – *Pantalla principal.*

Desde aquí se puede acceder a las diferentes funciones. Cada una de ellas se presentan en forma de pestañas en el margen izquierdo.

La primera de ellas es la pestaña de estado. Esta muestra la pagina en la cual se puede visualizar las diferentes variables de monitoreo. Si se modifican las variables, para estos cambios verse reflejados en la pantalla deberá actualizarse la pagina web, 4.20.

Figura 4.20 – *Pantalla de estado.*

La segunda pestaña es la de “Acciones inmediatas”, con la cual se desplegara la página donde se podrán realizar acciones inmediatas, como

puede ser dar de comer, recargar el bebedero de agua y abrir o cerrar una puerta, Fig. 4.21.

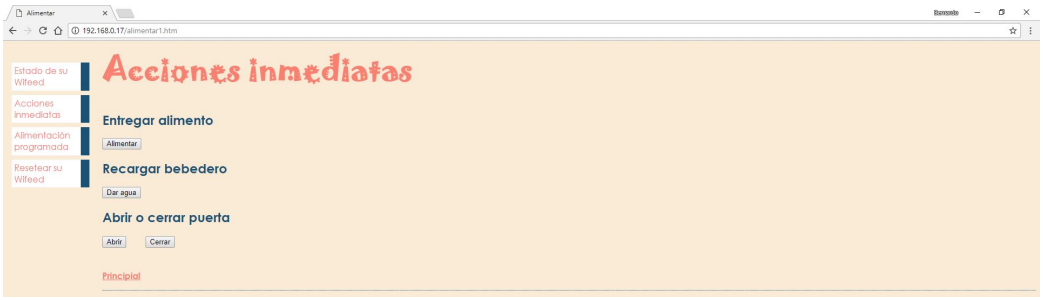


Figura 4.21 – *Pantalla de Acciones Inmediatas.*

En las siguiente pestaña se puede observar el nombre de "Acciones Programadas", aquí se podrán realizar las configuraciones relacionadas a la alimentación automática. Esta programación se ira haciendo de a pasos mediante la visualización de diferentes pantallas.

Al presionar en la pestaña primero se muestra una pantalla de presentación, Fig. 4.22:

Figura 4.22 – *Programación de alimentación.*

Dando click en comenzar se mostrara la pantalla que permitirá indicarle al sistema el tamaño de porción de alimento que se dejara caer en cada entrega, Fig. 4.23.

Figura 4.23 – *Programación de alimentación.*

La pantalla siguiente en el proceso de programación es la de elección de cuantas veces al día se desea que el dispositivo entregue comida de forma automática. Al igual que en la aplicación móvil, se puede escoger de una a tres veces por días, Fig. 4.24.

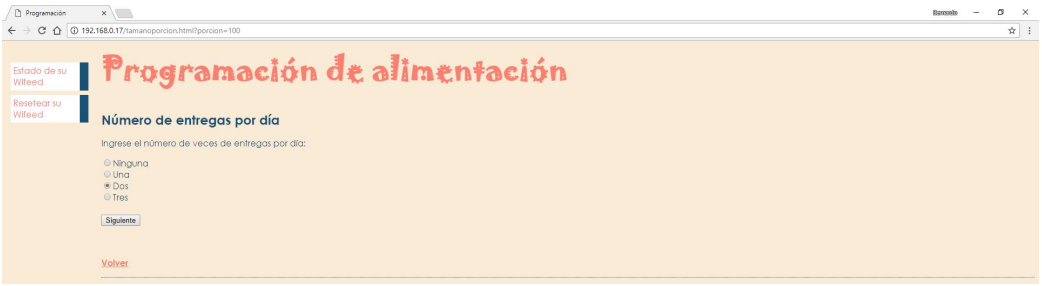


Figura 4.24 – *Programación de alimentación.*

Finalmente, dependiendo de la elección de número de veces, se muestra la pantalla para definir los diferentes horarios de alimentación automática, Fig. 4.25.

Figura 4.25 – *Programación de alimentación.*

Por último, la pestaña "Reiniciar Wifeed", aquí se da la posibilidad de reiniciar el dispositivo principal por si surge la necesidad, ya sea para borrar las programaciones de alimentación o por si se requiere de un reinicio en caso de observar problemas en el funcionamiento, Fig. 4.26.

Figura 4.26 – *Reseteo de dispositivo principal.*

Una vez reiniciado el dispositivo se muestra la pantalla de confirmación que muestra la Fig. 4.27:

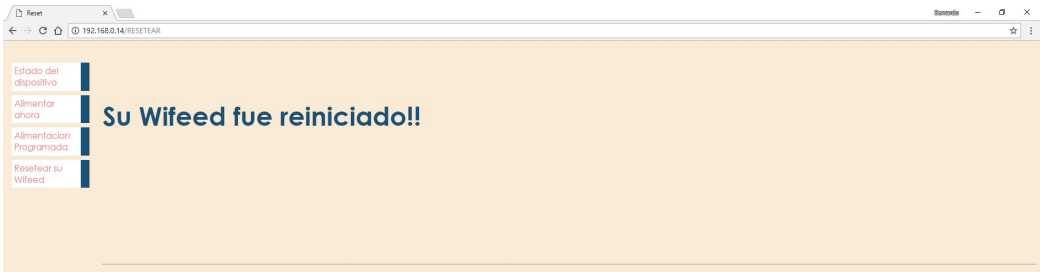


Figura 4.27 – *Reseteo de dispositivo principal.*

Capítulo 5
Conclusiones

Se diseñó e implementó un sistema, llamado *Wifeed*, que es capaz de entregar una cantidad determinada de alimento balanceado en el momento que el usuario lo desee, ya sea activándolo desde su teléfono móvil, de una página web o bien de manera automática a determinadas horas pre-programadas. Este dispositivo es también capaz de recolectar y enviar información al usuario. De forma de conocer cuanto alimento queda, si esta en buen estado, como así también de a qué horas se alimentó la mascota. Además envía una alerta indicando que se esta terminando el alimento, con nivel pre-programable. En este microcontrolador también se aloja un servidor, el cual da respuesta al usuario a través de Internet. Además de la unidad principal, el sistema completo consta de una aplicación Android que es una alternativa de acceso a través de una smartphone o tablet. Se concluye que los objetivos fijados fueron alcanzados, habiéndose adquirido una valiosa experiencia en el campo de IoT, en programación en LUA, http, Android, y en los conocimientos de domótica.

Bibliografía

[1] nodemcu-devkit.

[2] Nodemcu, connect things easy.

[3] Nodemcu custom builds.

[4] Docker NodeMCU build. Docker.

[5] ESP8266 community wiki. Toolchain.

[6] esp8266.ru. Esplorer, 2016.

[7] Reto Meier. *Professional Android 4 application development*. John Wiley & Sons, 2012.

[8] Android Developers. What is android, 2011.

[9] Tom White. *Hadoop: The definitive guide*. .º'Reilly Media, Inc.", 2012.

[10] Hot dog.

[11] Microsoft frontpage.

[12] Dreamweaver.

[13] Adobe golive.