ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ECUELA DE INGENIERÍA ELECTRÓNICA CONTROL Y REDES INDUSTRIALES

IMPLEMENTACIÓN DE UN ALIMENTADOR AUTOMÁTICO AUTÓNOMO PARA PECES CON COMUNICACIÓN GSM EN LA EMPRESA ALFA CONSTRUCCIONES EN ACEROS

"Tesis de Grado, previa obtención del título de Ingeniero en Electrónica Control y Redes industriales"

Johnny Ricardo Villavicencio Velastegui

Riobamba Chimborazo Ecuador 2016

NOMBRE	FIRMA	FECHA
Ing. Washingtong Luna E. **DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**	…….	…..…..…….
Ing. Freddy Chávez V. **DIRECTOR DE LA ESCUELA DE INGENIERÍA ELECTRÓNICA EN CONTROL Y REDES INDUSTRIALES**	…….	……..…….
Ing. Edwin Altamirano **DIRECTOR DE TESIS**	…….	……..…….
Ing. José Guerra S. **MIEMBRO DEL TRIBUNAL**	…….	……..…….
NOTA DE LA TESIS	…….	……..…….

DECLARACIÓN DE AUTENCIDAD

"**YO JOHNNY RICARDO VILLAVICENCIO VELASTEGUI**, soy el responsable de las ideas, doctrinas y resultados expuestos en este: Trabajo de titulación, y el patrimonio intelectual de la misma pertenecen a la empresa **ALFA CONSTRUCCIONES EN ACEROS**".

Riobamba, 9 de Agosto del 2016

Johnny Ricardo Villavicencio Velastegui
C.I. 16004141716

CONTENIDO.

LISTA DE FIGURAS

Número Página

LISTA DE TABLAS

Número Página

LISTA DE DIAGRAMAS

Número Página

RESUMEN

En la piscicultura nacional se ejecuta todo el proceso de producción de forma manual, la parte que menos control y conocimiento de su ejecución, tiempo y costo tienen los piscicultores es la alimentación. Por lo que la investigación desarrolla un alimentador automático autónomo electrónico que permita el control y proporcionar el costo de producción.

Para suministrar el alimento en cantidad y en horario necesario, se diseñó el alimentador automático autónomo con la tarjeta de desarrollo Arduino MEGA 2560. El sistema utiliza una cantidad de entradas y salidas digitales que permiten adquirir y entregar datos desde los *shields* a los actuadores. Siendo la programación del sistema la que toma la decisión de la entrega del alimento basándose en datos previamente ingresados por el usuario; las decisiones del sistema se basan en la propuesta de alimentación presente en la tabla de nutrición del productor de balanceados EMPAGRAN S.A. y en la experiencia de los piscicultores. El *software* de manejo se desarrolló en el IDE de Arduino, la programación de lectura de datos, tiempos y decisiones de encendido apagado de *shields* y actuadores se encuentra en su memoria *flash*, los datos que necesita el *software* se almacenan en la memoria *EEPROM*.

El alimentador automático autónomo cumple con cada uno de los requisitos necesarios para que el proceso de alimentación sea ejecutado de forma correcta, mostrando su efectividad en cada una de las pruebas realizadas para comprobar con los parámetros de referencia. La entrega de alimento fue realizada en la cantidad necesaria, el número de dosis requeridas, en las horas previstas y sin producir desperdicio al esparcir los *pellets*, dejando claro el beneficio de la alimentación por medio del alimentador automático autónomo desarrollado comparado con la alimentación manual.

SUMMARY

In national aquaculture entire production process is executed manually, the party less control and knowledge of their execution, time and cost have is feeding fish farmers. As research develops an electronic automatic feeder which allows autonomous control and provide the cost of production. To provide food in quantity and time required, the autonomous automatic feeder with MEGA 2560 Arduino development board. The system uses a number of digital inputs and outputs that can acquire and deliver data from the shields to the actuators was designed. Begin system programming which makes the decision to food delivery based on data previously entered by the user, system decisions are based on the proposal feed present in the nutrition table producer balanced EPAGRAN S.A. and the experience of the farmers. Management software developed in the IDE Arduino, reading programming data, times and decisions on off of shields and actuators is in it flash memory, and data needed by the software are stored in the EEPROM memory.

The autonomous ADF meets each of the requirements for the feeding process is executed correctly, showing its effectiveness in each of the tests performed to check with the required in the hours planned and without producing waste by spreading the pellets, making clear the benefits of power through automatic self-feeder developed compared to manual feed.

GLOSARIO

- **GSM**. Sistema Global para las comunicaciones Móviles
- **RTC**. Reloj en tiempo real
- **LCD.** Pantallas de cristal líquido
- **TFT.** Transistor de películas finas
- **CELDA SOLAR.** Transforman la luz solar en energía eléctrica.
- **PANEL SOLAR.** Conjunto de celdas solares.
- **MICROCONTROLADOR.** Circuito integrado programable.
- **GND.** Tierra del sistema.
- **VCC.** Voltaje colector común
- **DC.** Corriente directa.
- **Hz.** Hercios
- **ION.** Átomo o un grupo de átomos que tiene una carga neta positiva o negativa
- **I.** Intensidad de corriente
- **A.** Amperio.
- **mA.** Mili amperio.
- **s.** segundo.
- **ms.** Mili segundo
- **W.** Vatio.
- **h.** Hora.
- **°.** Grado.
- **I2C.** Es un bus con múltiples maestros, lo que significa que se pueden conectar varios chips al mismo bus y que todos ellos pueden actuar como maestro, sólo con iniciar la transferencia de datos.
- **SPI.** (*Serial Peripheral Interface*) es un protocolo de datos en serie síncrono utilizado por los microcontroladores para comunicarse con uno o más dispositivos periféricos

rápidamente en distancias cortas. También se puede utilizar para la comunicación entre dos microcontroladores.

- **MISO.** (*Master in Slave Out*) - La línea de esclavo para el envío de datos al maestro,
- **MOSI.** (Maestro Fuera Esclavo dentro) - La línea principal para el envío de datos a los periféricos,
- **SCK.** (Reloj serie) - Los impulsos de reloj que sincronizan la transmisión de datos generado por el maestro y una línea específica para cada dispositivo.
- **UART.** Puertos serial por *hardware*

INTRODUCCIÓN

El presente trabajo fue desarrollado tomando como base las necesidades y problemáticas de los pequeños y medianos productores de peces, que se encuentran sin ayuda tecnológica para mejorar la producción piscícola. El alimentador automático autónomo es un sistema capaz de ejecutar la alimentación de forma fiable, permanente, precisa y flexible siendo de pequeño volumen, de estructura liviana, amigable con el medio ambiente, interactiva, fácil de configurar y manejar; cualidades que permite ayudar a los productores con este trabajo.

En la primera parte del documento se da un preámbulo de la situación actual de la piscicultura nacional, la determinación de la problemática que se encuentra debido a la falta de equipos o tecnologías que ayuden al productor en las diferentes etapas de la producción de peces y se determina las cualidades que debe tener la alimentación automática.

En la segunda parte se establece las tecnologías a utilizar y cómo conjugarlas, como la estructura física para obtener el alimentador automático autónomo necesario que responda a la problemática.

PLANTEAMIENTO DEL PROBLEMA

En el Ecuador se desarrolló la piscicultura hace cinco lustros, dándose un desarrollo significativo para la economía de quienes se dedican a ella y del país, especialmente en los últimos seis años; esto propició que la piscicultura dé grandes avances en mejoras genéticas, alimentación y métodos de cultivo. A pesar de ser uno de los primeros exportadores de peces de la especie tilapia roja y tilapia negra del Nilo a nivel de Latinoamérica, la piscicultura ecuatoriana no ha visto mejoras tecnológicas para la producción y manejo de peces, siendo la crianza de forma manual en su totalidad.

En la crianza de peces para consumo humano se deben tener en cuenta múltiples factores para una producción eficiente, uno de estos factores es la alimentación, misma que es un pilar indispensable para la obtención de tamaño, consistencia de la carne y sabor, así también para el desarrollo de los animales en lapsos necesarios son indispensables para generar ganancia a los piscicultores.

La dosificación de alimento depende de la pericia y experiencia de quien la realiza, produciéndose deficiencia o sobredosis de alimento en relación a las necesidades de los animales, haciendo que los

resultados físicos, temporales, gastos e ingresos económicos se vean afectados; provocando una perdida en el negocio piscicultor.

Para la obtención de la entrega de alimento de forma automática el alimentador automático autónomo ejecutará la alimentación por medio de un actuador que extrae el alimento desde el contenedor mismo que entregará la cantidad necesaria, se esparcirá el alimento a través de un impulsor colocado al extremo del tobogán que lleva el alimento extraído del contenedor, para los horarios el alimentador automático autónomo posee un reloj de tiempo real junto con el controlador para que este pueda dosificar el alimento en la hora precisa, horarios que se establecerán automáticamente a partir de la edad de los peces ingresada por el usuario.

Durante el desarrollo del alimentador automático autónomo se hizo evidente según entrevistas que los piscicultores no llevan registro de la cantidad de alimento que se entrega durante la producción, desconociendo además el costo que tuvo el alimento entregado a lo largo de la misma.

Como resultado de las consultas no se encontraron trabajos o equipos en el país referentes al tema; se ubicó un trabajo realizado sobre el tema el Ingeniero Jhonatan Gallo de Colombia, en su paper publicado en el año 2013 titulado: "DISEÑO DE UN SISTEMA AVANZADO DE DOSIFICACIÓN DE CONCENTRADO PARA PECES EN CAUTIVERIO

Concluyen que:

El alimentador es asequible para los piscicultores al ser de bajo costo, facilitando la explotación de los cultivos acuícolas entregando a tiempo y regulando la cantidad de alimento establecido por la influencia de las variables externas, con una alimentación adecuada se garantiza un mejor crecimiento de estos y por ende una mayor competitividad y rentabilidad al reducir los desperdicios.

Pero no se considera que sea el alimentador el que realice el cálculo de cuanto alimento entregar, a qué hora y cuantas dosis al día entregar, ya que en su alimentador se debe ingresar manualmente cada una de estas cantidades; a más de que no se lleva un registro del costo del alimento entregado, no se comunica con el usuario para informarle de la realización de la alimentación, de la necesidad de cambio de tipo de alimento ni el momento de cosecha; así como tampoco muestra el peso del pece ni el tiempo de producción ejecutado, así mismo su alimentador no utiliza energías alternativas, ni evita el consumo innecesario estando conectado a la red eléctrica y comandado por una PC

además de su microcontrolador, y no tiene la precisión en la dosificación del alimento que se manifiesta en el alimentador automático autónomo ejecutado en este trabajo de titulación.

FORMULACIÓN DEL PROBLEMA

¿Se conseguirá reducir el desperdicio de alimento y entregarlo en los tiempos previstos de alimentación en la producción mediante la implementación del sistema de alimentación automático?

SISTEMATIZACIÓN DEL PROBLEMA

¿Cómo se está realizando actualmente el proceso de alimentación de peces?

¿Cuáles son los diseños eléctricos, electrónicos y mecánicos que satisfagan la automatización de la alimentación de peces?

¿Ayudará a controlar de manera óptima el proceso de alimentación de los peces?

¿La automatización permitirá la entrega oportuna en horario y volumen de alimento a los peces?

¿Se tiene conocimiento exacto del costo del alimento entregado durante la producción?

JUSTIFICACIÓN TEÓRICA DE LA INVESTIGACIÓN

Un proceso automatizado es más eficiente, fiable y rentable que un proceso artesanal o manual; para cada edad y tamaño de paces es necesario un determinado número de comidas al día de un alimento específico, para el cumplimiento de esto se crea el *software* de la placa de desarrollo que maneja el *hardware* del alimentador automático autónomo, permitiendo controlar los horarios de las dosis diarias y volumen a entregar.

El alimentador automático autónomo posee una interfaz clara que facilita su manipulación por parte del operador a través del teclado en su pantalla LCD TFT táctil, entregando avisos de errores, alarma de alimento próximo a terminarse, necesidad de cambio del tipo de alimento, nivel bajo de batería y tiempo de pesca próximo, por medio de mensajes de texto hacia el teléfono móvil del productor a través de su módulo GSM integrado; además nos entrega un resumen de coste, tiempo y volumen total de alimento entregado a lo largo de la producción.

El sensor de detección nos permite conocer la presencia de alimento y su nivel, datos previamente necesarios para ejecutar la dosificación hacia el impulsor para la dispersión. El control de la cantidad de alimento se calibra previamente con una báscula electrónica misma que permite determinar la posición angular del tornillo para un peso determinado.

Para la extracción del alimento desde el contenedor se utiliza un motor de pasos que controla la posición angular del tornillo según la cantidad de alimento necesario.

La dispersión del alimento se realiza por un sistema centrífugo necesario para realizar el trabajo; una vez iniciada la extracción del alimento desde el contenedor, el alimento se desliza a través de la guía para ser dispersado en el estanque.

La automatización en la entrega de alimento se realiza por medio de una placa de desarrollo la cual almacena una tabla de alimentación dada por los fabricantes de alimento, usando esta como base para el cálculo de la cantidad y horarios en los que debe ser distribuido. Para la autonomía está provista de una fuente de energía recargable (batería) por medio de paneles solares que son amigables con el ambiente y proveen energía limpia, haciéndolo portátil.

La configuración del dosificador automático autónomo se lo realiza a través de una interfaz amigable y clara, por medio de sus botones y teclado en la pantalla LCD TFT táctil, con los cuales podemos introducir la cantidad y edad de los peces, hora actual, costo de cada libra según el tamaño del alimento.

JUSTIFICACIÓN APLICATIVA DE LA INVESTIGACIÓN

Uno de los problemas directos en la producción es el alimento desperdiciado en el proceso de alimentación; por exceso, esto hace que el alimento se descomponga en el cuerpo de agua produciendo baja en el nivel de oxígeno y aumento en los niveles de amoniaco, a más de regarse una pequeña parte en el suelo al trasladarlo hacia el estanque y al arrojarlo al mismo; por deficiencia, la falta de alimento influye directamente en el desarrollo de los peces. La entrega insuficiente o excesiva de alimento así como en horarios distintos repercute en un menor tamaño y peso de los peces, y mayor tiempo de levantamiento.

A más de lo mencionado anteriormente el pequeño y mediano piscicultor desconoce el monto total en alimento invertido en un estanque en el transcurso de su producción, así como el coste de trasladarse varias veces al día hacia el estanque aparte del abandono de sus otras actividades.

El alimentador automático autónomo permite entregar el alimento en los horarios precisos de ingesta, dispensando la cantidad óptima que necesitan los peces para desarrollarse adecuadamente, evitando así el desperdicio o la ingesta insuficiente.

Al eliminar el desperdicio de alimento se reducirá la inversión en este, y al garantizar la entrega necesaria para la ingesta en horarios precisos que permiten alcanzar un desarrollo mejor de los animales, ayudando a tener mayor rendimiento de la inversión para el piscicultor.

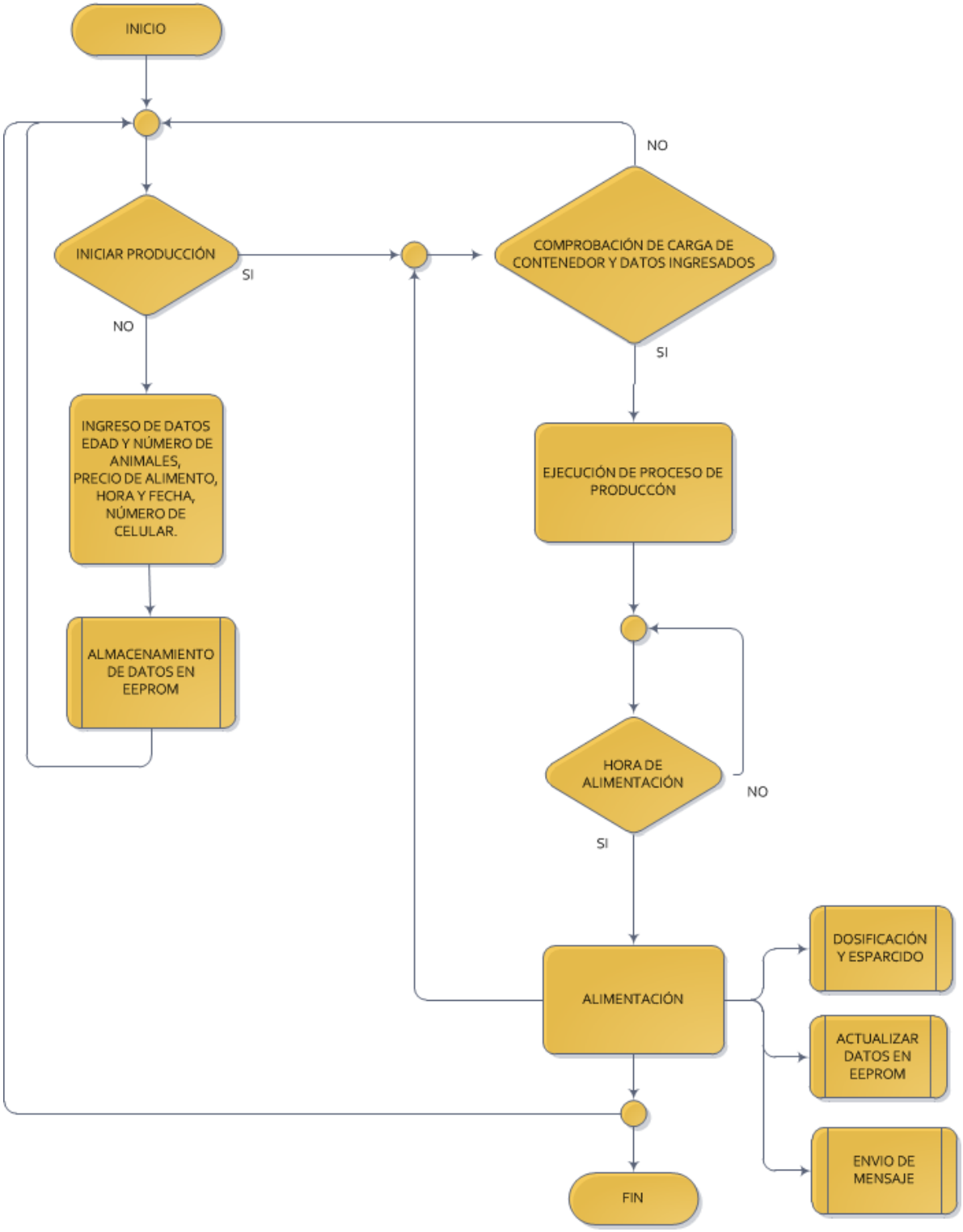


Diagrama 1 Flujo de ejecución del *software*

Fuente: Villavicencio 2016

OBJETIVOS

OBJETIVO GENERAL:

Implementar un alimentador automático autónomo para peces con placa de desarrollo y módulo GSM en la empresa ALFA Construcciones en Aceros

OBJETIVOS ESPECÍFICOS:

Investigar sobre la dosificación y control automático para alimentar peces.

Seleccionar los sensores, actuadores y dispositivos que permitan el trabado del alimentador automático autónomo.

Diseñar los sistemas eléctricos y electrónicos que permitan la operación de los sensores, actuadores y dispositivos.

Implementar los sistemas eléctricos y electrónicos diseñados.

Diseñar la estructura y mecanismos de manera que alojen a los diferentes elementos, sensores, actuadores y dispositivos.

Integrar los elementos, sensores, actuadores, dispositivos y sistemas con la estructura del alimentador automático autónomo.

Implementación del *software* de control del alimentador automático autónomo.

Ejecutar las pruebas de funcionamiento del alimentador automático autónomo.

Comprobar la reducción de la perdida de alimento y tiempos de alimentación en la producción

CAPÍTULO I

1. MARCO TEÓRICO REFERENCIAL

En este capítulo se hace una revisión de los sistemas necesarios en el desarrollo del presente trabajo de titulación como la dosificación y sus tipos, la automatización, la alimentación de peces, los tipos de dispersión y un resumen de la situación actual de la piscicultura en el país.

1.1. DOSIFICACIÓN

Es la acción de determinar una cantidad proporcional de alguna solución, el tipo y forma de dosificación es determinado por las características propias del material a dosificar, como también por las condiciones necesarias e impuestas para cada tarea determinada.

Los sistemas de dosificación deben preparar cantidades determinadas de material en un lapso concreto y en la relación definida. Para mantener de forma fiel la receta, lo más importante no es el volumen sino la masa del producto a dosificar. Por otro lado, el resultado del proceso de dosificación depende de las "características del producto a granel", de las condiciones del medio y del proceso de dosificación con relación al sistema seleccionado para ello. Con respecto al procedimiento, hay diferenciar la dosificación volumétrica de la gravimétrica. (Tecnología del Plástico, 2016)

1.1.1. TIPOS DE DOSIFICACIÓN

Los sistemas de dosificación volumétricos proporcionan el material en función de las cantidades, los sistemas de dosificación gravimétricos pesan el material y proporcionan este en función de la masa, como muestra la figura 2.

1.1.1.1. DOSIFICACIÓN VOLUMÉTRICA

La expulsión del material se realiza únicamente en función del volumen, y por ello, de las cantidades. Como en esta dosificación no se mide la masa, los elementos de dosificación se tienen que calibrar según del material antes de cada uso: es necesario determinar la cantidad de masa que tiene que dosificar el elemento en un periodo definido. Estos sistemas de dosificación no pueden compensar automáticamente los cambios en las propiedades del material. (Tecnología del Plástico, 2016)

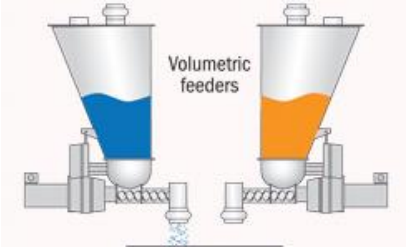


Figura 1, Dosificación volumétrica.
Fuente: http://www.ktron.com/industries_served/Pharmaceutical/Batch_Dispensing_via_Gain-in-Weight.cfm

1.1.1.2. DOSIFICACIÓN GRAVIMÉTRICA

Una o varias células de pesaje pesan el material a dosificar. Por lo que se dosifica en unidades de peso. La comparación entre lo teórico y lo real regula la dosificación, por esta razón los sistemas gravimétricos pueden compensar automáticamente los cambios en las propiedades del material. Una ventaja adicional es llevar el registro de las cantidades que se transportan. (Tecnología del Plástico, 2016)

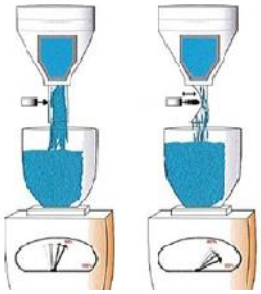


Figura 2, Dosificación gravimétrica.
Fuente: http://www.interempresas.net/Plastico/Articulos/22787-Wittmann-presenta-sus-nuevos-dosificadores-gravimetricos-y-sus-molinos-y-trituradores.html

1.2. AUTOMATIZACIÓN

Sistema tecnológico basado en la ingeniería y la informática, que proporciona una optimización de los procesos productivos mediante la regulación automática (auto regulador). (The Free Dictionary, 2016).

La automatización es un proceso donde se trasfieren tareas de producción, realizadas habitualmente por operadores humanos a un conjunto de elementos tecnológicos el cual consta de dos partes:

La *Parte Operativa* es la que actúa directamente sobre la máquina. Son los elementos que hacen que la máquina se mueva y realice la operación deseada, forman la parte operativa como los accionadores de máquinas como motores, cilindros, compresores y los captadores como fotodiodos, finales de carrera.

La *Parte de Mando* suele ser un autómata programable que se encuentra en el centro del sistema, es capaz de comunicarse con todos los constituyentes del sistema automatizado. Hasta hace poco se utilizaban relés electromagnéticos, tarjetas electrónicas o módulos lógicos neumáticos (tecnología cableada), ahora se trabaja con súper relés, circuitos lógicos programables, microcontroladores y computadores. (AUTÓMATAS PROGRAMABLES Curso Básico de Autómatas Programables, 2001)

Los objetivos de la automatización son:

- Mejorar la productividad de la empresa, reduciendo los costes de la producción y mejorando la calidad de la misma.
- Mejorar las condiciones de trabajo del personal, suprimiendo los trabajos penosos e incrementando la seguridad.
- Realizar las operaciones imposibles de controlar intelectual o manualmente.
- Mejorar la disponibilidad de los productos, pudiendo proveer las cantidades necesarias en el momento preciso.
- Simplificar el mantenimiento de forma que el operario no requiera grandes conocimientos para la manipulación del proceso productivo.
- Integrar la gestión y producción.

(AUTÓMATAS PROGRAMABLES Curso Básico de Autómatas Programables, 2001)

1.3. ALIMENTACIÓN

La alimentación llega a representar del 50 al 60% de los costos de producción. El fertilizante más caro es un alimento mal manejado. Si se posee un inadecuado programa de alimentación la rentabilidad del negocio disminuye. La producción intensiva y semi-intensiva depende directamente del alimento. Las cantidades y los tipos de alimento suministrados deben controlarse y evaluarse

periódicamente para evitar excesivos costos. El sabor del pez depende de la alimentación suministrada. La subalimentación provoca que el los peces se alimenten del fondo del estanque y adquiera un desagradable sabor.(Ortega, 2016)

De la eficiencia en el cultivo será el éxito de la actividad piscícola, según la calidad y cantidad del alimento suministrado. La tilapia es omnívora por lo que el tipo y su requerimiento de alimento varían con la edad del pez. Los alevines y juveniles se alimentan de fitoplancton, zooplancton y de pequeños crustáceos.(Ortega, 2016)

Los peces de estanque de producción para consumo humano se alimentan de dos tipos de alimento principalmente, el alimento primario que lo constituye el alimento balanceado, y el secundario que lo constituye el fitoplancton. (Ortega, 2016)

1.3.1. FITOPLANCTON

Se llama fitoplancton al conjunto de organismos acuáticos autótrofos del plancton, que poseen la capacidad fotosintética y que viven en el cuerpo de agua, es el alimento natural que se encuentra en el medio. (Koprucu y Ozdemir, 2005).

1.3.2. BALANCEADO

Generalmente los alimentos para tilapia se comercializan en tres o cuatro presentaciones con varias especificaciones nutricionales según la edad del pez, como pre-inicial, inicial, juvenil y engorde. Para la formulación de alimentos balanceados se usa un equilibrio de coste mínimo en función de los requisitos nutricionales del pez y los ingredientes disponibles. Los alimentos más comercializados en el país son ABA, PISIS Y PRONACA, en *pellets* de tres y cuatro tamaños y polvo para la etapa pre-inicial. (Koprucu y Ozdemir, 2005).(FAO, 2016)

La figura 3 se muestra un ejemplo de alimento balanceado tipo *pellets*.

Figura 3, Balanceado de peces tipo *pellets*.

Fuente: http://pisciculturaglobal.com/2012/09/alimentar-peces-utilizando-tablas-de.html

1.3.3. CANTIDAD DE ALIMENTO Y DOSIS

Es recomendable seguir muy bien el programa de alimentación sugerido, para obtener los resultados esperados, de acuerdo a las necesidades económicas,

En la tabla 1 podemos apreciar la ración de alimento balanceado por cada 1000 peces.

Semana	Peso	Dosis/Día	Ración/	Tito de
1	0.3-1	4-6	200	50% Polvo
2	1-2	4-6	360	50% Polvo
3	8	4	700	38% 1/16
4	15	4	900	38% 1/16
5	22	4	1000	32% 3/32
6	30	4	1300	32% 3/32
7	40	4	1650	32% 3/32
8	52	3	2100	32% 3/32
9	62	3	2400	32% 1/8
10	80	3	2800	32% 1/8
11	100	3	3150	32% 1/8
12	125	3	3600	32% 1/8
13	140	3	3800	32% 1/8
14	160	3	4300	32% 1/8
15	188	3	4700	32% 1/8
16	210	3	5000	32% 1/8
17	240	3	5500	32% 1/8
18	270	3	5700	32% 1/8
19	300	3	5900	32% 1/8
20	330	3	6100	32% 1/8

Semana	Peso	Dosis/Día	Ración/	Tito de
21	360	3	6400	32% 1/4
22	390	3	6800	32% 1/4
23	410	3	7000	32% 1/4
24	440	3	7500	32% 1/4

Tabla 1, Alimentación según edad del pez, en condiciones de agua normales 28°C.
Fuente: folleto EPAGRAM.S.A.

1.4. DISPERSIÓN

Separación, esparcimiento o extensión de un conjunto o de una cosa que está unida.(The Free Dictionary, 2016)

Dentro de la alimentación es un factor importante la entrega del alimento en el estanque de forma adecuada, esta se logra ejecutando una amplia dispersión del balanceado, dando lugar a que los animales no se aglomeren ocasionándose lesiones y asfixia.

Los *pellets* se dispersan usualmente de tres formas:

- Manual.
- Por aire.
- Centrifuga.

1.4.1. MANUAL

Donde una persona ejecuta la acción de dispersar el alimento de forma manual con la ayuda de una herramienta (balde, pala, pocillo, etc.), esta forma de dispersión es utilizada en el país; la figura 4 muestra la alimentación manual.

Figura 4, Alimentación manual.
Fuente: http://mundopangasius.mex.tl/386456_Alimentacion.html

1.4.2. POR AIRE

Los *pellets* son impulsados hacia el estanque por medio de aire forzado producido por un medio mecánico, este método es muy utilizado en países como Brasil, Méjico y Estados Unidos, la figura 5 muestra un impulsor por aire forzado a gasolina.

Figura 5, Soplador de balanceado
Fuente: http://santiago.all.biz/alimentador-portatil-g51798#.V4FQEPl97IU

1.4.3. CENTRÍFUGO

Los *pellets* son impulsados por la fuerza centrífuga del rotor, que los proyecta a gran velocidad hacia el estanque, es el menos utilizado no habiéndose encontrado datos sobre lugares que lo utilicen, la figura 6 muestra un esquema de trayectoria del alimento.

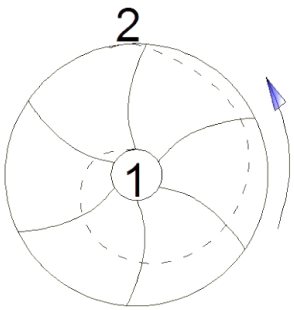


Figura 6, Esquema de la proyección centrífuga de los *pellets*.
Fuente: https://es.wikipedia.org/wiki/Bomba_centr%C3%ADfuga#/media/File:Bombcentr.jpg

1.5. SITUACIÓN ACTUAL DE LA PISICULTURA EN EL PAÍS

Esta se ejecuta de forma artesanal en su totalidad, donde los piscicultores buscan obtener ingresos tratando de dar un manejo necesario de los peces y mejorar la rentabilidad de esta actividad, objetivos que no se cumplen en este proceso pues no se lleva un registro claro y preciso de la cantidad de alimento suministrado y el costo que implica el proceso de producción. En este proceso el alimento se entrega sin considerar las cantidades, dosis y tipo de alimento requerido en los tiempos recomendados; por lo que la mayoría de los piscicultores desconocen cuanto invierten y si ganan o pierden en una determinada producción.

CAPÍTULO II

2. MARCO METODOLÓGICO

En este capítulo se hace una revisión referente a los requerimientos del alimentador automático autónomo, *hardware*, *software* y la estructura física necesaria para implementarlo.

2.1. DISEÑO DEL ALIMENTADOR AUTOMÁTICO AUTÓNOMO

El diseñó del alimentador automático autónomo se inspiró en la necesidad de alimentar los peces en cantidades y horarios precisos de los pequeños y medianos piscicultores a más de procurar reducir el desperdicio del alimento. Para el diseño se tomó en cuenta el tamaño, peso y capacidad necesarios acoplándose a los requerimientos de los emplazamientos; así mismo se incorporó una interfaz visual completamente táctil sencilla e intuitiva para la configuración. El *software* implementado ejecuta automáticamente el horario de alimentación, dosis y número de dosis diarias basado en los parámetros ingresados previamente por el usuario. El alimentador proporciona adicionalmente múltiples informaciones necesarias para la producción como el cálculo del costo de alimento, tiempo de producción, edad y peso de los peces, en su tabla de resumen, permite comunicar al piscicultor por medio de la red celular a través de mensajes de texto SMS sobre la ejecución de la alimentación y nivel de alimento en el contenedor.

Su diseño considera un fácil traslado en el campo, la falta de acceso al servicio eléctrico tanto en las zonas rurales como en las zonas agrícolas, por lo que el alimentador está provisto de un panel solar que provee de energía limpia. Para almacenar la energía posee un banco de baterías de ion de litio las cuales no utilizan agentes químicos nocivos; siendo estas tecnologías amigables con el medio ambiente y completamente reciclables, ayudando y promoviendo de esta manera el respeto y cuidado de la naturaleza.

Con la finalidad de que el alimentador tenga en su totalidad las características anteriormente citadas, se realizó esbozos de cómo debería ser el alimentador automático autónomo y posteriormente se realizó el diseño físico en *SOLID WORKS*, de esta forma se tuvo un modelo que facilitó su implementación.

2.2. REQUERIMIENTOS

Para que el alimentador automático autónomo funcione dentro de lo previsto es necesario que el usuario ingrese varios datos; como son la edad y la cantidad de animales, número de teléfono celular, precio de cada tamaño de alimento *pellet* por kilogramo, adicionalmente es necesario poner la hora actual en el reloj del *software*, estos datos usará el *software* para ejecutar el establecimiento de los horarios de alimentación, las dosis, destino al que se enviarán los mensajes de texto, el costo del alimento entregado y la ejecución de la entrega de alimento en las cantidades y horas ya determinadas previamente por el *software*.

Para estar seguro de que se ingresaron los datos correctamente, así como para corregirlos y cambiarlos es necesario establecer una opción en el menú que permita consultar los datos ingresados por el usuario previo a la ejecución de la alimentación, de ser necesario corregir o modificar algún dato o todos.

Es primordial que se establezca una interfaz sencilla e intuitiva para que la mayoría de los piscicultores puedan manipularla sin dificultad.

El alimentador debe ser capaz de operar sin conexión a la red eléctrica, consumir poca energía, usar energía limpia, reciclable en su mayor parte, liviano para su fácil transporte, resistente a las condiciones ambientales, ser de bajo costo, y tener una capacidad mínima y máxima de carga de alimento que se ajuste a las necesidades de los piscicultores.

2.3. *HARDWARE*

En esta sección se trata el hardware elegido según los requerimientos planteados para el alimentador automático autónomo, se explica su conjugación con un diagrama de bloques y se da una breve explicación de las tarjetas de desarrollo, Arduino, tarjeta Arduino mega 2560, sensor ultrasónico, pantalla TFT *touch*, motor a pasos, motor *brushless*, GSM, *shield* GSM, RTC, ESC y el *driver* L298n, zumbador, diagrama de conexiones, y se termina con el software para diseño de PCB.

2.3.1. DIAGRAMA DE BLOQUES

El **bloque de energía** está integrado por los paneles solares, el circuito de acondicionamiento de energía y la batería de ion de lito. Este bloque se encarga de entregar la energía necesaria y en los niveles requeridos por cada uno de los otros bloques,

El **bloque de control** está compuesto por la tarjeta Arduino MEGA 2560, se encarga del cálculo y procesamiento de datos, manejo y control de los bloques de potencia, comunicación SMS, tiempo y del bloque de medición de carga.

El **bloque de potencia** está a cargo de accionar los actuadores conforme lo mande el bloque de control, y está constituido por el L298N y el ESC.

El **bloque de comunicación SMS** es el encargado de enviar los mensajes de texto al número de celular ingresado por el usuario, aquí se encuentra el escudo GPRS/GSM.

El **bloque de actuadores** realiza la dosificación y el esparcido de los *pellets*, está integrado por el motor a pasos, el motor *brushless* y el zumbador que anuncia su inicio y culminación.

El **bloque de interfaz de entrada/salida** es el encargado de la interacción con el usuario, siendo la pantalla TFT *touch* la que lo conforma.

El **bloque de tiempo** está compuesto por el RTC, mismo que se encarga de llevar la hora y fecha, datos primordiales para ejecutar la alimentación y el cálculo de los datos de producción.

El **bloque de medición de carga** integrado por el sensor ultrasónico HC-SR04, el cual ejecuta la medición del nivel de alimento en el contenedor.

En el diagrama 2 podemos observar la composición y cómo se relaciona el *hardware*.

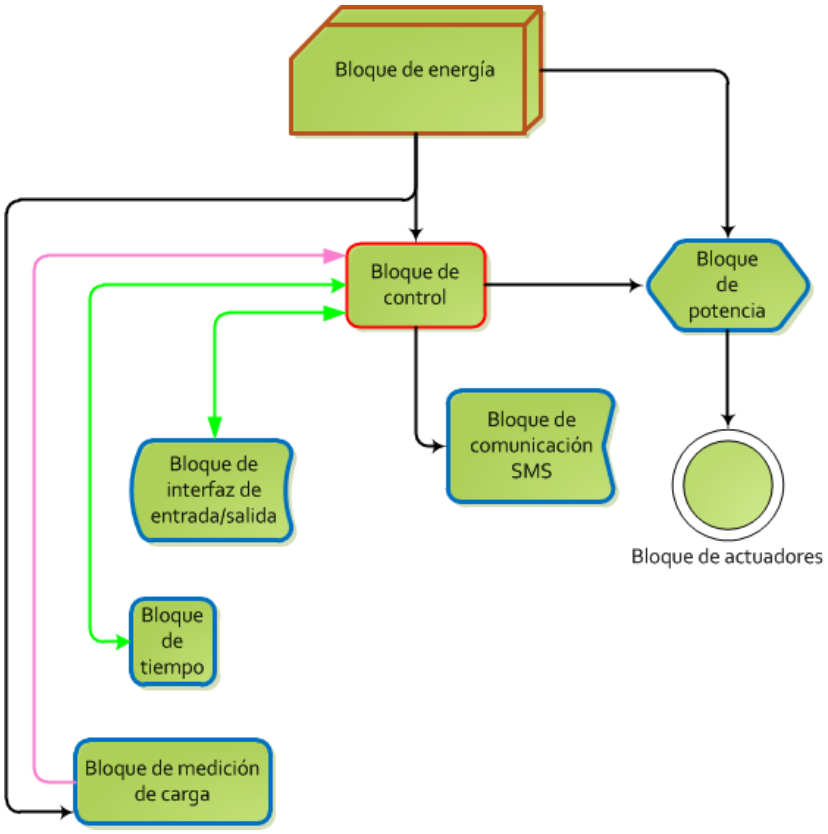


Diagrama 2, Esquema de bloques del *hardware*.

Fuente: El autor.

2.3.2. TARJETAS DE DESARROLLO

En la actualidad la electrónica se ha inclinado en varios frentes, uno de ellos es la mejora y la variedad disponible de placas de desarrollo y los *shields* conectables con estas, cada vez son más utilizadas para ejecutar y desarrollar prototipos con un nivel de complejidad y alcance mayor, gracias a que el mercado se ha visto inundado de una gran gama de estas, que a su vez poseen una variedad de cualidades y características que permiten a los desarrolladores, crear sistemas y equipos que realizan trabajos dedicados con bajo coste, relativamente fáciles de ensamblar y programar. Teniendo en estos una facilidad para configurar, cambiar, ampliar o reducir el *hardware*, junto con la capacidad de procesamiento, programabilidad rápida y reconfiguración en la construcción de prototipos, han permitido un avance en la aplicación de estas tecnologías en múltiples aspectos de la vida cotidiana, recreación y de producción.(Gonzales, 2010)

2.3.3. ARDUINO

Es una compañía de *hardware* libre, y con una comunidad tecnológica, que se dedica al diseño y manufactura de placas de desarrollo de *hardware* y *software* compuesta por circuitos impresos que integran un microcontrolador, y un entorno de desarrollo (IDE) en donde se programa cada placa. Arduino se enfoca en acercar y facilitar el uso de la electrónica y programación de sistemas embebidos en proyectos multidisciplinarios. La plataforma en su totalidad, tanto para los componentes de *hardware* y de *software* son liberados bajo licencia de código abierto, lo cual permite total libertad de acceso a estos. (Wikipedia La enciclopedia libre, 2016)

El *hardware* es una placa de circuito impreso con un microcontrolador, donde sus puertos digitales y analógicos pueden conectarse a placas de expansión (*shields*) que amplían el rango de acción de la placa Arduino. Su puerto USB el cual puede alimentar la placa y establecer comunicación serial con el computador. (Wikipedia La enciclopedia libre, 2016)

El *software* consta de un entorno de desarrollo (IDE) basado en *Processing* y lenguaje de programación basado en *Wiring*, y el cargador de arranque (*bootloader*) que se ejecuta en la placa. (Wikipedia La enciclopedia libre, 2016)

2.3.4. TARJETA ARDUINO MEGA 2560 R3

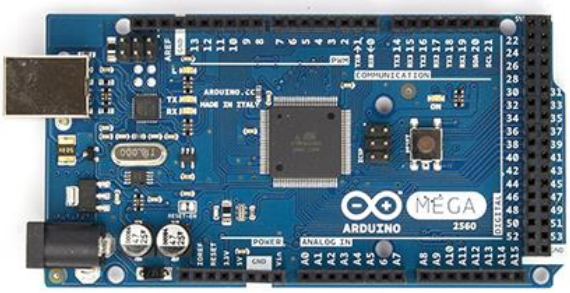


Figura 7, Arduino MEGA 2560

Fuente: https://forum.arduino.cc/index.php?topic=330302.0

En la figura 7 se aprecia la placa de desarrollo Arduino MEGA 2560.

Se eligió la palca Arduino MEGA 2560 por poseer una cantidad de puertos analógicos y digitales suficientemente amplia para la conexión de los múltiples *shields* utilizados, por sus buenas prestaciones de procesamiento y por su facilidad de adquisición en el mercado local.

2.3.5. CARACTERÍSTICAS

El Arduino Mega posee 54 terminales de entradas/salidas digitales (14 de estas tienen la capacidad de operar PWM), 16 entradas análogas, 4 UARTs (puertos serial por *hardware*), cristal oscilador de 16 MHz, conexión USB, *Jack* de alimentación, conector ICSP y botón de *reset*. Incorpora todo lo necesario para que el microcontrolador trabaje con simplemente conectarlo la PC por cable USB o con una fuente de alimentación externa. El Arduino Mega es compatible con la mayoría de los *shields* diseñados para Arduino Duemilanove, diecimila o UNO. (Ojeda, 2016)

2.3.6. ULTRASONIDO

La velocidad de propagación del sonido varía según el tipo y características del material por el que se desplaza. La velocidad del sonido depende de las características de la sustancia, como son la densidad y la compresibilidad, así en los materiales con mayor densidad y menor compresibilidad transmiten el sonido con mayor velocidad.

El ultrasonido es una onda mecánica longitudinal con frecuencia de vibración superior a 20 000 ciclos por segundo, por lo que el oído humano es incapaz de percibirlo. Se lo aplica en diversos campos, se lo puede dirigir como un haz, cumple las leyes de refracción y reflexión y los objetos pequeños pueden reflejarlo. Se propaga muy poco a través de los gases y la energía acústica reflejada depende de las desigualdades acústicas del medio en el que se propaga. (Peñarreta, s.f.)

La figura 8 muestra las frecuencias en las que se clasifica los sonidos.

Figura 8, Rangos de frecuencia de los sonidos.
Fuente: http://www.monografias.com/trabajos90/ultrasonido-frecuencia/ultrasonido-frecuencia.shtml

2.3.6.1. SENSOR ULTRASÓNICO HC-SR04

El sensor de ultrasonidos se utiliza normalmente para medir distancias o superar obstáculos sin entrar en contacto con estos, entre otras aplicaciones. Esto se consigue enviando un ultrasonido desde uno de sus cilindros que compone el sensor y el otro cilindro recoge el rebote del sonido sobre un objeto; este sensor tiene un rango de distancias de 3cm a 3m con una precisión de 3mm.(Barbus, 2014)

La figura 9 muestra el sensor ultrasónico HC-SR04 *shield* para Arduino.

Figura 9, Ultrasónico HC-SR04

Fuente: http://microcontrollerelectronics.com/distance-sensing/

2.3.6.2. CÓMO TRADUCIR EL TIEMPO DE REBOTE EN DISTANCIA

Aprovechando que la velocidad en el aire de este ultrasonido es de **340 m/s**, o **0,034 cm/u-seg**. Siendo la velocidad igual a la distancia recorrida en un determinado tiempo tenemos que **v=d/t**; donde obtenemos **d=v·t**, siendo **v** constante y **t** el tiempo devuelto por el sensor a la placa Arduino, es necesario dividir el resultado entre **dos** dado que el tiempo recibido es de ida y vuelta. (Barbus, 2014)

2.3.7. PANTALLA LCD TFT *TOUCH*

TFT-LCD (*Thin Film Transistor-Liquid Crystal Display*, Pantalla de cristal líquido de transistores de película fina) es una variante de pantalla de cristal líquido para mejorar su calidad de imagen. Las LCD de TFT son un tipo de LCD de matriz activa, están disponibles comúnmente en tamaños de 12 a 30 pulgadas. (Wikipedia, 2016)

La figura 10 muestra el esquema interno del LCD TFT.

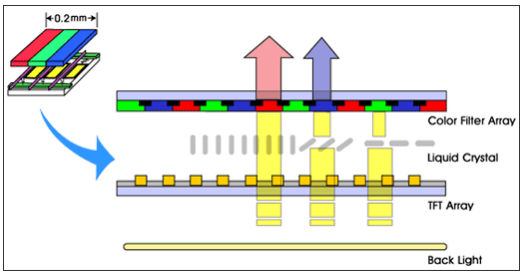


Figura 10, Constitución interna LCD TFT

Fuente: http://m.blog.daum.net/garnet-1/539980

2.3.7.1. SEGURIDAD

Los cristales líquidos contenidos en la pantalla son extremadamente tóxicos. No se deben ingerir o tocar. (Wikipedia, 2016)

2.3.8. HMI

Es el punto de acción en que un hombre entra en contacto con una máquina. El caso más simple es el de un interruptor: No se trata de un humano ni de una "alimentador automático autónomo" (la lámpara), más bien de una interfaz entre los dos. (COPADATA, s.f.)

La HMI está constituida por la LCD TFT táctil de 2.4’ a color, esta HMI consta de un menú y sub menús necesarios para la configuración y ejecución del *software*, así como para la muestra de la hora, botón de *START*, *STOP* y de datos necesarios para la comprobación de los parámetros ingresados y visualización de otros datos que se han calculado a partir de los ingresados por el usuario, todos estos pueden ser consultados por el usuario.

2.3.9. MOTOR A PASOS

Se utiliza especialmente cuando se precisa un sistema de posicionamiento seguro y fiable evitando recurrir a sistemas más complejos como los servomecanismos. Solucionando con relativa sencillez cuando hay exigencia de velocidad de accionamiento en varios movimientos ordinarios asociados a sistemas de frenado con gran seguridad y de problemática ejecución práctica. (Falak, 2010)

En la figura 11 se ve un motor de paso unipolar de 5 hilos.

Figura 11, Motor a pasos

Fuente: https://www.openhacks.com/page/productos/id/155/title/Stepper-Motor-con-Reductor-28BYJ48#.V36r7Pl97IU

2.3.9.1. PRINCIPIO DE FUNCIONAMIENTO

Está basado en un estator que contiene varios arrollamientos independientes, devanados alojados sobre un material ferromagnético y su rotor puede girar libremente en el estator. (Falak, 2010)

Los bobinados son alimentados uno a continuación del otro provocando un desplazamiento angular que se denomina "paso angular". El sentido de rotación es establecido por la secuencia en la que se excitan los arrollamientos. (Falak, 2010)

2.3.9.2. MODOS DE PASO A PASO

Las secuencias más comunes de control de los motores a pasos son las siguientes:

- Control de onda
- Control total del paso (*Full Step*)
- Control de la Mitad del Paso (*Half Step*)

2.3.9.3. CONTROL DE ONDA

En este modo se alimenta sólo una fase en un momento determinado. En los motores unipolares esto significa que se están utilizando el 25% de los bobinados existentes, para los motores bipolares se utiliza 50 %. (Falak, 2010)

Orden	Fase A	Fase B	Fase A	Fase B
1				
2				
3				
4				

Tabla 2, Control de onda.

Fuente: http://www.monografias.com/trabajos93/motor-paso-paso/motor-paso-paso.shtml

2.3.9.4. CONTROL TOTAL DEL PASO (*FULL STEP*)

En este modo se alimentan dos fases al unísono, Para los unipolares esto significa que se utiliza el 50% de los bobinados disponibles, para los motores bipolares se utiliza el 100%.(Falak, 2010)

Orden	Fase A	Fase B	Fase A	Fase B
1				
2				
3				
4				

Tabla 3, Control de paso completo.

Fuente: http://www.monografias.com/trabajos93/motor-paso-paso/motor-paso-paso.shtml

2.3.9.5. CONTROL DE MITAD DE PASO (*HALF STEP*)

En este modo se mezcla las secuencias de onda y control de paso completo, haciendo que el rotor esté alineado en la mitad de cada paso. Los motores unipolares en este modo están empleando el 37,5% de los bobinados disponibles, mientras que el uso alcanza el 75% en los bipolares. (Falak, 2010)

Orden	Fase A	Fase B	Fase A'	Fase B'
1				
2				
3				
4				
5				
6				
7				
8				

Tabla 4, Control medio paso.

Fuente: http://www.monografias.com/trabajos93/motor-paso-paso/motor-paso-paso.shtml

2.3.10. MOTOR SIN ESCOBILLAS

Los motores DC electrónicamente conmutados (*brushless* DC) se destacan por sus excelentes características de par, altas prestaciones, amplio rango de velocidades y por su alta duración en servicio. Carecen de conmutación mecánica al no tener escobillas. Sin embargo para realizar la conmutación requieren electrónica externa (o integrada). Vida útil prolongada, limitada por los rodamientos, a carga máxima mínimo 20.000 horas, giro suave, carece de par de retención, alcanza altas velocidades incluso con bajos voltajes; pueden alcanzar 50.000 rpm y 100.000 en algunos casos. (monografias.com, 2010)

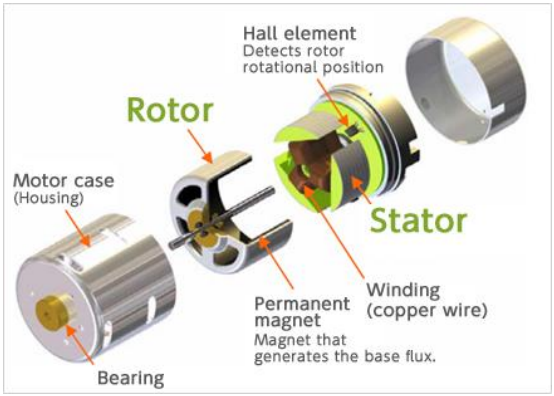


Figura 12, Despiece de un motor *brushless*.

Fuente: http://www.nidec.com/en-NA/technology/capability/*brushless*/

2.3.11. GSM

El Sistema Global para las comunicaciones Móviles es el más utilizado por la telefonía celular digital. La tecnología GSM utiliza técnicas por división de frecuencia y de tiempo (FDMA y TDMA) para optimizar la capacidad de carga de una red inalámbrica.

Los terminales GSM pueden operar en dos bandas de frecuencia por lo menos, lo que mejora la posibilidad de comunicación. Las frecuencias en uso son de 850MHz y 900 MHz (hasta 2W de potencia, frecuencias bajas) y 1,8GHz y 1,9GHz en altas frecuencias (hasta 1W de potencia), no existe un uso común de frecuencias a todos los países, por lo que la tendencia es hacia módulos cuadri-banda.

El servicio de mensajes cortos SMS permite enviar y recibir mensajes pequeños de texto, con tamaño máximo de 160 caracteres. Junto con el mensaje de texto se obtiene el remitente, la hora y la fecha de recepción. (Contreras, s.f.)

2.3.11.1. *SHIELD* GEEETECH GSM/GPRS

El *shield* GPRS/GSM incluye todas las piezas necesarias para conectar directamente a Arduino el módulo celular SIM900. Este trabaja en las bandas de 850/900/1800/1900MHz permitiendo utilizar fácilmente SMS, GSM /GPRS con tu Arduino. Todo necesario para agregar funcionalidad celular a un proyecto es una tarjeta SIM.

Figura 13, *Shield* GSM

Fuente: http://www.naylampmechatronics.com/arduino-*shield*s/146-*shield*-geeetech-gsm-gprs.html

La figura 13 muestra el *shield* GSM montable en la placa Arduino.

2.3.11.2. ESPECIFICACIONES TÉCNICAS

- Quad-Band 850/ 900/ 1800/ 1900 MHz
- GPRS multi-slot class 10/8
- GPRS mobile station class B
- Compliant to GSM phase 2/2+
- Trabaja solo con Tecnología 2G (en Perú movistar, Claro y Entel)
- Class 4 (2 W @850/ 900 MHz)
- Class 1 (1 W @ 1800/1900MHz)
- Control via comandos AT (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands)
- Voltaje de Operación: 5V
- Bajo consumo de corriente: 1.5mA(sleep mode)
- Temperatura de operación: -40°C to +85 °C
- Conector mini plug para audífono y micrófono
- (NAYLAMP mechatronics, 2016)

2.3.12. RTC

Un Reloj en tiempo real (real-*time clock*, **RTC**) es un reloj incluido en un circuito integrado que mantiene la hora actual. Los RTC están presentes en la gran mayoría de los aparatos electrónicos que necesitan conocer el tiempo exacto. (Wikipedia, 2015)

2.3.12.1. *SHIELD* RTC DS1307

El **DS1307 de Maxim/Dallas** es un circuito integrado capaz de almacenar y llevar la cuenta de la fecha y hora, además disponemos de unos cuantos bytes de datos de usuario en su memoria RAM no volátil (NVRAM). (GEEKFACTORY)

El *shield* RTC DS 1307 se muestra en la figura 14.

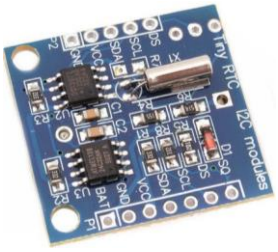


Figura 14, RTC DS1307

Fuente: http://www.ebay.com/itm/Tiny-RTC-DS1307-*Shield*-Module-Kit-V2-0-Arduino-Compatible-/270941621713

Bus I2C o TWI – Es el protocolo de comunicación físico mediante el cual se comunican el Arduino y el **módulo RTC DS1307**. El bus cuenta con dos líneas: de datos y de reloj. (GEEKFACTORY)

La figura 15 muestra una conexión básica del DS1307 con un microcontrolador a través del bus I2C.

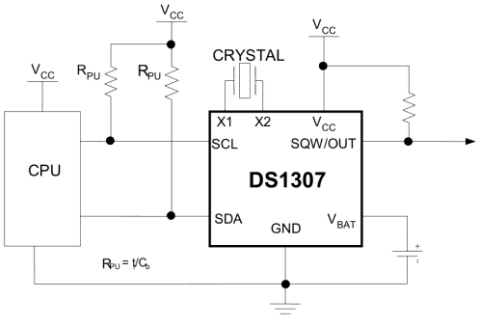


Figura 15, Conexión DS1307.

Fuente: http://www.geekfactory.mx/tutoriales/tutoriales-arduino/ds1307-en-tinyrtc-con-arduino/

2.3.12.2. ZUMBADOR

Es un transductor electro acústico que puede producir un sonido continuo o intermitente de generalmente agudo. Se utiliza como mecanismo de señalización acústica. La figura 16 muestra la forma física de un zumbador piezoeléctrico.

Figura 16, zumbador piezoeléctrico de 5v.
Fuente: Villavicencio 2016

2.3.13. ESC

Un **control electrónico de velocidad** o **ESC** es un circuito electrónico encargado de controlar la velocidad de un motor eléctrico, su dirección y también puede actuar como freno dinámico. Utilizados a menudo en la propulsión de modelos de radio control. (Wikipedia, 2016)

La estructura del circuito que forma un ESC es mostrado en la figura 17.

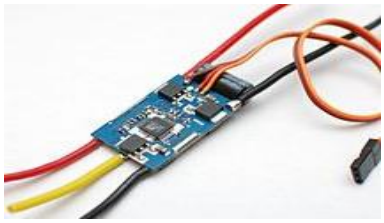


Figura 17, ESC
Fuente: https://en.wikipedia.org/wiki/File:ESC_35A.jpg

2.3.13.1. FUNCIÓN

Un ESC interpreta la información de manera que varía la velocidad de conmutación de una red de transistores. La rápido conmutación de los transistores es lo que hace que el motor emita un sonido agudo característico.

Se maneja con una señal de entrada PWM servo 50 Hz nominales donde su ancho de pulso varía entre 1 ms y 2 ms. Al suministrar un ancho de pulso de 1 ms a 50 Hz el ESC responde apagando el

motor. Un ancho de pulso de 1,5 ms acciona el motor cerca a la mitad de su velocidad. Cuando el pulso tiene un ancho de 2,0 ms, el motor funciona a máxima velocidad. (Wikipedia, 2016)

2.3.14. L298N

El L298N es un circuito integrado que está formando un puente H, este es un circuito electrónico que permite cambiar de dirección a un voltaje que se aplica a una carga. Estos circuitos se utilizan a menudo en la robótica y otras aplicaciones que permiten a los motores de corriente continua correr hacia adelante y hacia atrás. En particular un motor de paso a paso está casi invariablemente accionado por un controlador de motor que contiene dos puentes de H. (Wikipedia, 2016)

La figura 18 muestra la estructura del *shield* L298N.

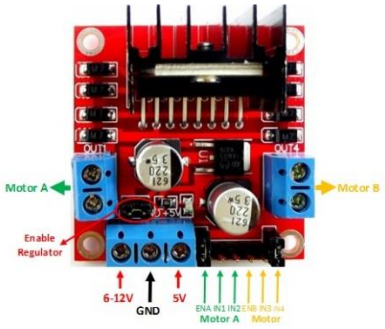


Figura 18, L298N

Fuente: http://articulo.mercadolibre.com.mx/MLM-550367373-driver-de-motores-doble-puente-h-l298n-arduinopicrobotica-_JM

2.3.14.1. CARACTERÍSTICAS PRINCIPALES:

- Control puente H dual L298N
- Voltaje Lógico: 5 V.
- Voltaje de potencia: 5-35 V.
- Corriente Lógica: 0-36 mA.
- Corriente de potencia: 2,5A (máxima de un solos canal).
- Temperatura de almacenamiento -20 ºC a 135 ºC.
- Potencia Máxima: 25 W.

2.3.15. BATERÍA DE ION DE LITIO

Energía específica	100–265 W·h/kg[1] [2] (0.36–0.95 MJ/kg)
Densidad energética	250–730 W·h/L[2] (0.90–2.23 MJ/L)
Potencia específica	~250-~340 W/kg[1]
Eficiencia carga/descarga	80–90%[3]
Energía / precio consumidor	2.5 W·h/US$
Velocidad de auto descarga (%/mes)	8% a 21 °C 15% a 40 °C 31% a 60 °C (por mes)
Durabilidad (ciclos)	400–1200 ciclos
Voltaje de célula nominal	NMC 3.6 / 3.7 V,LiFePO4 3.2 V

Tabla 5, Características de batería ion de Litio

Fuente: https://es.wikipedia.org/wiki/Bater%C3%ADa_de_ion_de_litio

La batería de iones de litio o **Li-Ion**, es un dispositivo creado para el almacenamiento de energía eléctrica que usa como electrolito una sal de litio que adquiere los iones necesarios para la reacción electroquímica reversible que se ejecuta entre el cátodo y el ánodo.

Sus propiedades, su bajo peso, la elevada capacidad energética, resistencia a la descarga, el poco efecto memoria, el elevado número de ciclos de regeneración, permiten diseñar acumuladores ligeros, pequeños y de variadas formas, con alto rendimiento, especialmente adaptados a las aplicaciones de la industria electrónica de gran consumo.(Wikipedia, 2016)

2.3.16. PANEL SOLAR

Un panel fotovoltaico está formado por numerosas celdas que convierten la luz en electricidad. Las celdas dependen del efecto fotovoltaico el cual produce cargas positiva y negativa a partir de la

energía lumínica que incide en dos semiconductores próximos de diferente tipo, generando un campo eléctrico capaz de producir una corriente. (Wikipedia , 2016)

Las celdas solares suelen estar hechas de silicio cristalino o arseniuro de galio. El silicio poli cristalino posee menor eficacia de conversión y menor coste. (Wikipedia , 2016)

Hay tres subcategorías en las células de silicio empleadas en paneles fotovoltaicos:

Las de silicio mono cristalino están constituidas por un único cristal de silicio, estas células presenta un uniforme color azul oscuro. (Wikipedia , 2016)

Las células de silicio poli cristalino están constituidas por un conjunto de cristales de silicio, lo que explica que su rendimiento sea algo inferior al de las células mono cristalinas. Se caracterizan por un color azul más intenso. (Wikipedia , 2016)

Las células de silicio amorfo son menos eficientes que las de silicios cristalinos y más baratos también. Este tipo de células se emplea en relojes o calculadoras. (Wikipedia , 2016)

Figura 19, Panel solar

Fuente: http://www.batanga.com/curiosidades/2011/03/03/como-funciona-un-panel-solar

2.3.17. SISTEMA ELÉCTRICO

El alimentador automático autónomo necesita de dos voltajes para funcionar, 5v dc y 10.8v dc; los 5v son utilizados para alimentar la tarjeta de desarrollo, el HC-SR04, el RTC, la LCD TFT y el *shield* GSM, mientras que los 10.8v son suministrados a los motores.

La batería de ion de litio es cargada por la energía proporcionada por el panel solar a través del circuito de acondicionamiento de voltaje, que establece el voltaje de 18 voltios a 660 mA del panel a los 10.8v que necesita la batería para cargarse completamente, adicionalmente el circuito posee un na salida de 5v dc a 1000 mA a plena carga, utilizada para energizar los elementos arriba citados.

El circuito de acondicionamiento de la energía eléctrica suministrada por los paneles solares fue diseñado, concebido, construido y ensamblado acorde a las necesidades de los sistemas electrónicos y eléctricos del alimentador automático autónomo.

En la figura 20 se muestra el circuito pcb diseñado en 3D, el diagrama esquemático y el circuito implementado.

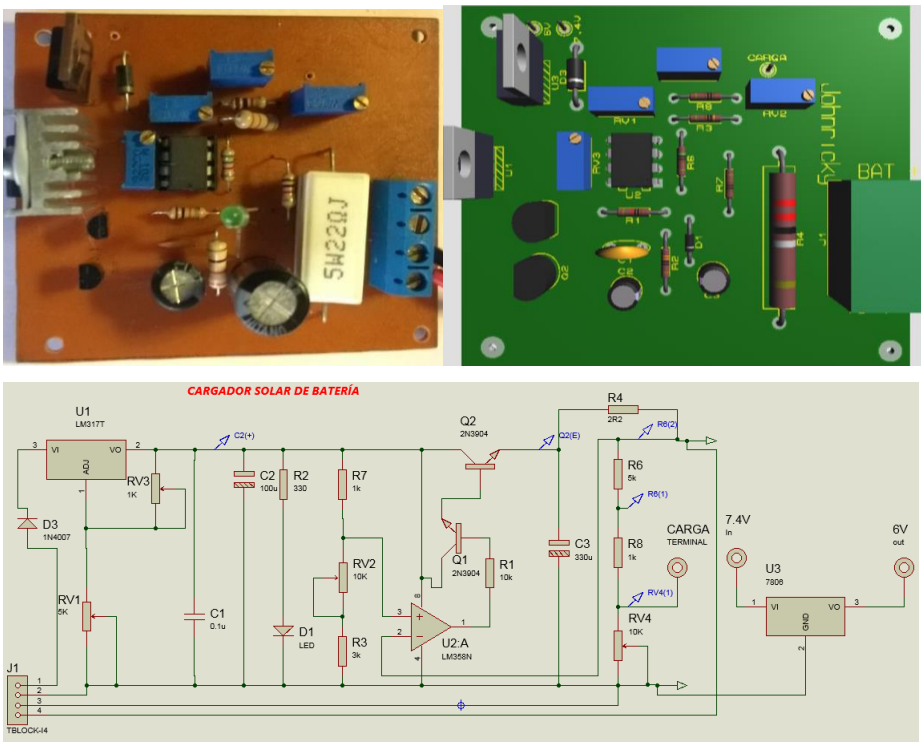


Figura 20, Circuito de acondicionamiento de energía, implementado izquierda, diseo 3d derecha, equemático abajo.

Fuente: Villavicencio 2016

2.3.18. DIAGRAMA DE CONEXIONES

En este se detalla cada una de los cables que se conectan desde la Arduino hacia los módulos que integran el alimentador automático autónomo y su conexión de VCC y GND, como se aprecia en el diagrama 3.

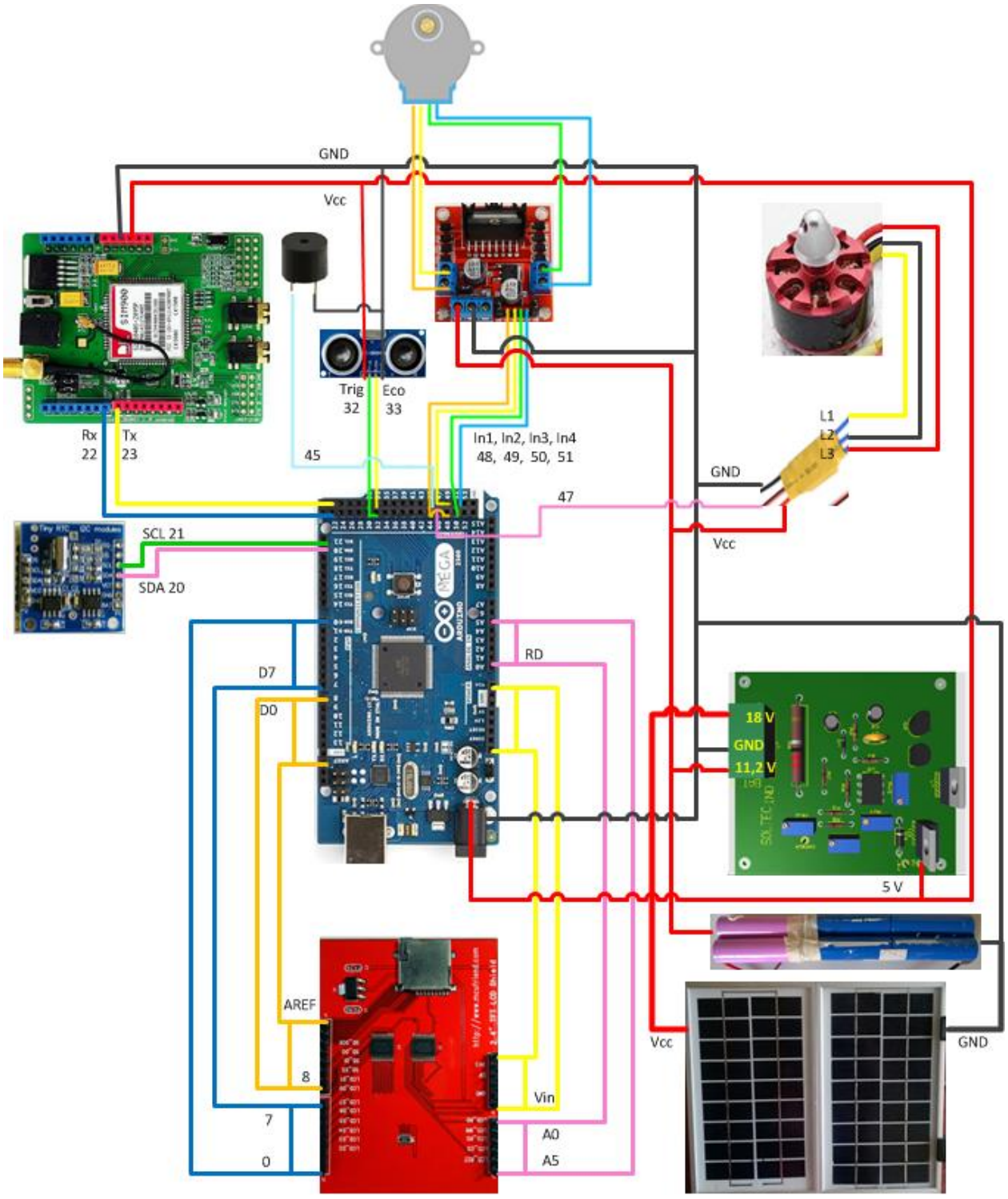


Diagrama 3, Conexión de los módulos con la Arduino.

Fuente: Villavicencio 2016.

2.3.19. SOFWARE DE DISEÑO DE LA PCB

Proteus es un entorno integrado diseñado para la realización completa de proyectos electrónicos en todas sus etapas, diseño, simulación, depuración y construcción. El depurado de errores puede convertirse en una labor larga en tiempo y recursos, lo que conlleva un alto coste económico. (Hubor-Proteus, 2016)

ISIS *Intelligent Schematic Input System* (Sistema de Enrutado de Esquemas Inteligente) permite diseñar el plano eléctrico del circuito que se desea realizar con componentes muy variados. Los diseños realizados en Isis pueden ser simulados en tiempo real, mediante el módulo VSM, asociado directamente con ISIS. (Hubor-Proteus, 2016)

2.3.19.1. El MÓDULO VSM

Virtual System Modeling (Sistema Virtual de Modelado), con la cual se puede simular, en tiempo real, todas las características de varias familias de microcontroladores. Se pueden simular circuitos con microcontroladores conectados a distintos dispositivos, como motores eléctricos, pantallas de cristal líquido (LCD), teclados en matriz, etc. Incluye, las familias de microcontrolador PIC. Combina un entorno de diseño de una potencia excepcional con una enorme capacidad de controlar la apariencia final de los dibujos. (Hubor-Proteus, 2016)

2.3.19.2. ARES

Advanced Routing and Editing Software (*Software* de Edición y Ruteo Avanzado); es la herramienta de enrutado, ubicación y edición de componentes, se utiliza para la fabricación de placas de circuito impreso. (Wikipedia, 2016)

2.4. *SOFTWARE*

En esta sección se trata el *IDE* elegido para la creación del *software* que rige al Arduino MEGA2560 y maneja a los *shields* según los requerimientos planteados para el alimentador automático autónomo, se da una breve explicación de entornos de desarrollo, lenguaje de programación, comandos AT, se explica su conjugación con un diagrama del *software* y se termina con una breve explicación de las funciones citadas en el diagrama del *software*.

2.4.1. ENTORNOS DE DESARROLLO

Al existir una multitud de tarjetas de desarrollo los IDE's en los que se puede programar son muy variados, existiendo IDE's orientados a la programación en bajo nivel, en alto nivel, en bloques, interactivas, patch y de introducción a modo de juego; a pesar de que las placas pueden soportar y manejar varios lenguajes para su programación la comunidad como los fabricantes se han inclinado por lenguajes tanto compactos, sencillos y dominantes en el mundo de la programación, siendo los más utilizados, C, C++, assembler, python, visual basic, IDE Arduino, C#, bascoon, entre otros; para el desarrollo del software de manejo del *hardware* escogido se eligió el IDE de Arduino por su gran cantidad de ejemplos, apoyo desde su comunidad, la familiarización y conocimiento previo de este.

2.4.2. LENGUAJE DE PROGRAMACIÓN

La plataforma Arduino se programa mediante el uso de un lenguaje propio basado en el lenguaje de programación de alto nivel *Processing* que es similar a C++. Arduino está basado en C y soporta todas las funciones del estándar C y algunas de C++. (Wikipedia La enciclopedia libre, 2016)

2.4.3. COMANDOS AT

Los comandos AT son instrucciones codificadas que conforman un lenguaje de comunicación entre el hombre y un terminal MODEM. Tienen como finalidad la comunicación con módems. Todos los teléfonos móviles GSM poseen un juego específico de comandos AT que sirven para configurar y proporcionar instrucciones a los terminales, permiten realizar llamadas de datos o de voz, escribir y leer en la agenda de contactos, enviar SMS, además de opciones de configuración del terminal. (Contreras, s.f.)

El envío de comandos AT requiere la siguiente estructura mostrada en la figura 21:

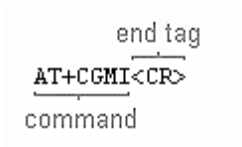


Figura 21, Comandos AT de ENTER

Fuente: http://www.monografias.com/trabajos93/tecnologia-gsm-aplicada-automatizacion-traves-micro-controladores/tecnologia-gsm-aplicada-automatizacion-traves-micro-controladores.shtml

Los comandos utilizados fueron:

- AT+CMGF Comando para enviar un MSM.
- AT + CMGS Recipiente de número móvil en formato internacional.

2.4.4. DIAGRAMA DEL SOFTWARE

En la **sección portada de presentación** se encarga de ejecutar la rutina que extrae las imágenes desde la memoria mico SD y las muestra en la pantalla según el orden establecido.

En la **sección ingreso y visualización de datos** para la interacción entre el usuario y el *software*, se emplea la LCD TFT de 2.4', misma que permite desplazarse por el menú, ingresar datos requeridos, comprobarlos y modificarlos por medio de botones. La rutina de manejo de la LCD TFT es ejecutada cuando el alimentador automático autónomo está en estado *STOP* y cuando ejecuta la alimentación en estado *START*, cada vez que el usuario toca sobre uno de sus botones la rutina identifica y ejecuta la acción asociada a este.

En la **sección costo del alimento** se ejecuta la rutina que muestra el teclado numérico con su respectivo título, permitiendo ingresar el dato dentro de un rango y lo guarda en la EEPROM al presionar *OK*.

En la **sección visualizar** se ejecuta la rutina que muestra la pantalla con cada uno de los datos ingresados por el usuario que están almacenados en la EEPROM.

En la **sección número y edad de peces** se ejecuta la rutina que muestra el teclado numérico con su título respectivo, permitiendo ingresar cada dato dentro de un rango y lo guarda en la EEPROM al presionar *OK*.

En la **sección ingreso hora/fecha** se ejecuta la rutina que muestra el teclado de cambio de Hora (hora y minuto), después de igualar la hora se muestra la fecha para su actualización (día/mes/año), permitiendo ingresar cada dato dentro de un rango y lo guarda en el RTC al presionar *OK*.

En la **sección ingreso número celular** se ejecuta la rutina que muestra el teclado numérico con su título respectivo, permitiendo ingresar el dato dentro de un rango y lo guarda en la EEPROM al presionar *OK*.

En la **sección datos del equipo y propietario** se ejecuta la función que muestra la pantalla con el menú para la elección de datos del equipo y propietario.

En la **sección visualización datos del equipo** se ejecuta la rutina que muestra la pantalla con datos referentes a la creación del equipo.

En la **sección visualización datos del propietario** se ejecuta la rutina que muestra la pantalla con datos referentes al dueño del equipo.

En la **sección lectura de la carga de alimento** se ejecuta la función cuando necesita conocer si el contenedor posee alimento dentro del contenedor, el *software* ejecuta la rutina de manejo del sensor HC-SR04 y toma acción según el resultado.

En la **sección espera por hora de alimentación** se ejecuta la función cuando no es la hora establecida en el horario, la rutina pone a dormir a la Arduino hasta que la hora actual sea la hora de alimentación.

En la **sección dosificación y esparcido del alimento** se ejecuta la función cuando es la hora establecida en el horario, la rutina comanda los *drivers* de los motores paso a paso y *brushless*.

En la **sección guardar en EEPROM datos actualizados** la función actualiza los datos que han cambiado con la alimentación realizada.

En la **sección envía un MSM** la función actúa al culminar la actualización de los datos en la EEPROM, enviando un mensaje de texto al celular del usuario ejecuta la rutina que maneja el *SHIELD GEEETECH* GSM/GPRS, carga el mensaje correspondiente y lo envía al número establecido.

En la **sección visualización del horario de entrega del alimento** la rutina ejecutada muestra el número de dosis al día y las horas a las que se entrega.

En la **sección visualización del resumen de producción** se ejecuta la función que muestra la pantalla con el tamaño de alimento que está entregando, numero de dosis, semana de producción, costo del alimento entregado y el peso estimado del pez.

El diagrama 4 muestra la composición del *software* residente en el alimentador automático autónomo.

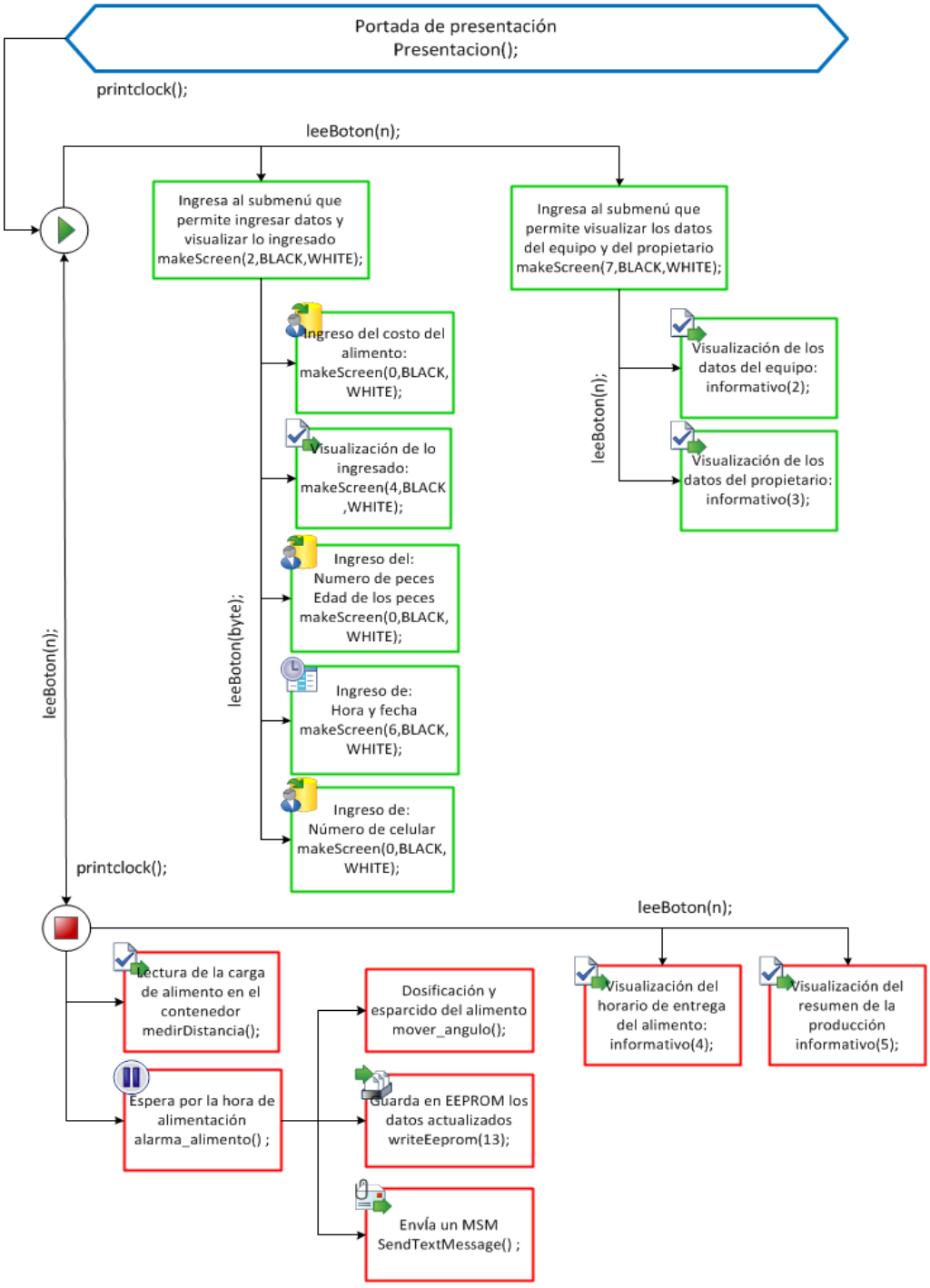


Diagrama 4, Composición del *software* que comanda el alimentador y sus funciones principales

Fuente: Villavicencio 2016.

2.4.5. DESCRIPCION DE LAS FUNCIONES DEL DIAGRAMA DEL *SOFTWARE*

Para la comunicación y manejo de los diferentes módulos con la Arduino, además de la manipulación y procesamiento de los tipos de datos se utilizaron las siguientes bibliotecas:

```
#include <String.h>
#include <Adafruit_GFX.h>    // Libreria de graficos
#include <Adafruit_TFTLCD.h> // Libreria de LCD
#include <SD.h>              // Libreria de tarjeta SD
#include <SPI.h>             // Libreria bus SPI
 #include <avr/pgmspace.h>
#include <TouchScreen.h>     // Libreria del panel tactil
#include <EEPROM.h>
#include <Time.h>            // librerias para manejar hora y fecha
#include <Wire.h>
#include <DS1307RTC.h>  // a basic DS1307 library that returns time as a time_t
#include <avr/wdt.h> // Incluir la librería que contiene el watchdog (wdt.h)
#include <avr/power.h>
#include <avr/sleep.h>
#include <SoftwareSerial.h>     // LIBRERIA PARA EL GSM
#include <Servo.h>         // LIBRERIA PARA EL BRUSHLESS
```

La función Presentacion(); no tiene parámetros y es la encargada de mostrar la presentación en pantalla.

La función makeScreen(2,BLACK,WHITE); tiene tres parámetros que hacen referencia a los gráficos que se deben mostrar en pantalla, color de la letra y color del botón respectivamente.

La función informativo(3); tiene un parámetro que hace referencia a los datos que se deben mostrar en pantalla.

La función medirDistancia(); no tiene parámetros, esta verifica cuan lleno está el contenedor y llama a la función que sea necesaria según el nivel de alimento.

La función alarma_alimento(); no tiene parámetros, esta duerme a la Arduino mientras no sea hora de alimentar y lo despierta cuando verifica que es hora de ejecutar la alimentación.

La función SendTextMessage(); no tiene parámetros, esta función activa el *shield* GSM envía el mensaje y lo vuelve a desactivar; los mensajes que envía son: "ALIMENTADOR v 1.2, MOTIVO: ALIMENTACION DE PECES EJECUTADA", "VACIO EN 24 HORAS", "USAR ALIMENTO DE 1/16", "USAR ALIMENTO DE 3/32", "USAR ALIMENTO DE 1/8", "USAR ALIMENTO DE 1/4", "TIEMPO DE COSECHAR LOS PECES".

La función leeBoton(n); tiene un parámetro, este hace referencia a los botones que se están mostrando en pantalla y determina si se ha presionado sobre uno de ellos.

La función printclock(); no tiene parámetros, esta función es la encargada de mostrar la hora y fecha actualizadas en las pantallas de *START* y de *STOP*.

La pantalla principal que se ve en la figura 22, muestra el resultado que se obtuvo en el software de aplicación implementado.

Figura 22, Pantalla de inicio del HMI, menú principal de configuración.
Fuente: Villavicencio 2016

2.5. ESTRUCTURA FÍSICA

En esta sección se trata el cada una de las partes físicas que se adaptan a los requerimientos planteados para el alimentador automático autónomo, se da una breve explicación del mueble, contenedor, soporte para motor de pasos, soporte para motor *brushless*, tobogán, tornillo sin fin, arreglo de los paneles solares, arreglo de la baterías de ion de litio, y se culmina con una explicación ligera de la calibración, tamaño de pellets que dosifica y resultado final

2.5.1. MUEBLE

La figura 23 muestra la estructura del alimentador, este está constituido por el mueble que sostiene el contenedor, los motores, el tobogán y el gabinete de control. Este se encuentra construido de tubo de acero de 1/2’ y 1.2 mm de espesor, mismo es capaz de sostener una carga superior a los 100kg.

Figura 23, Mueble del alimentador.
Fuente: Villavicencio 2016

2.5.2. CONTENEDOR

En la figura 24 se ve el contenedor, mismo que está construido de metacrilato blanco de 3 mm, teniendo la capacidad de almacenar alimento tipo pellet con un peso de 60kg. Aloja en su interior un tornillo sin fin que se encuentra conectado al motor de pasos.

Figura 24, Contenedor del alimentador.
Fuente: Villavicencio 2016

2.5.3. SOPORTE PARA MOTOR PASO A PASO

En la figura 25 se muestra el soporte plástico que es atornillado al motor de pasos y a su vez al contenedor.

Figura 25, Soporte para motor a pasos.
Fuente: Villavicencio 2016

2.5.4. SOPORTE PARA MOTOR *BRUSHLESS*

Construido en chapa metálica galvanizada, atornillado a este y el soporte a su vez está sujeto al tobogán.

Figura 26, Dispersador centrífugo.
Fuente: Villavicencio 2016

2.5.5. TOBOGÁN

El tobogán permite deslizar los *pellets* extraídos desde el contenedor hacia el esparcidor de forma segura y evitando el contacto con la intemperie, está construido en metacrilato blanco de 3mm como muestra la figura 27.

Figura 27, Tobogán.
Fuente: Villavicencio 2016

2.5.6. TORNILLO SIN FIN

Elaborado en teflón, el tornillo tiene la función de extraer el balanceado de *pellets* desde el contenedor hacia el tobogán que deposita en el esparcidor, en la figura 28 lo podemos apreciar.

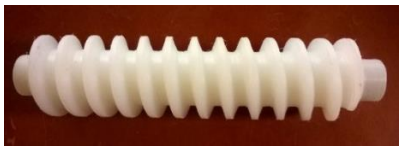


Figura 28, Tornillo sin fin.
Fuente: Villavicencio 2016

2.5.7. ARREGLO DE LOS PANELES SOLARES

Con el fin de suministrar una potencia suficiente para la recarga de la batería se realizó un arreglo de paneles solares, con 9 v cada uno logramos obtener una potencia de 6 W a 18 v, misma que es entregada a la batería a través del circuito de acondicionamiento.

La figura 29 muestra el ensamble de los 2 paneles solares utilizados.

Figura 29, Arreglo de paneles solares.
Fuente: Villavicencio 2016

2.5.8. ARREGLO DE BATERÍAS DE ION DE LITIO

La batería está constituida por un arreglo de seis baterías de ion de litio con 3,7 v a 2600 mA., dando como resultado una batería de 11,1 v a 5200 mA., suficientes para alimentar toda la electrónica y motores del alimentador, y asegurar su funcionamiento a carga máxima.

La figura 30 muestra el arreglo de baterías realizado.

Figura 30, Arreglo de 6 baterías de ion de litio de 3,7v a 2600mA.
Fuente: Villavicencio 2016

2.6. CALIBRACIÓN

Con el objetivo de lograr entregar la cantidad de alimento más precisa según sea necesario se realizó la calibración de la dosificación del alimento, para lo cual fue necesario determinar la cantidad de alimento que se podía extraer con el tornillo en un giro cerrado, para cada tamaño de alimento se tiene una cantidad determinada de alimento. Esta calibración se la ejecuta previo al ensamblaje del alimentador automático autónomo, en el trabajo normal usa los datos de la calibración cargados previamente en el *software*.

La figura 31 muestra la báscula y un recipiente con alimento balanceado durante el proceso de calibración.

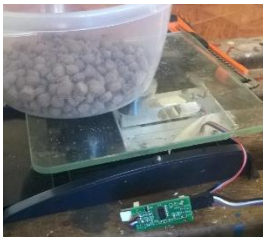


Figura 31, Calibración con báscula.
Fuente: Villavicencio 2016

2.7. TAMAÑOS DE *PELLETS* QUE DOSIFICA

El alimentador es capaz de entregar cuatro tamaños de balanceado en *pellets*, de las medidas: 1/16, 3/32, 1/8, y 1/4, en la figura 32 se muestra los cuatro tamaños de alimento balanceado.

Figura 32, Balanceado en *pellets*.
Fuente: Villavicencio 2016

2.8. RESULTADO FINAL

Después de montar, conectar y ensamblar cada parte se obtuvo el resultado final que se deseaba, con un peso total de **23,5 libras** siendo posible movilizarlo por una persona pero por su volumen es necesario dos personas. En la figura 33 podemos observar el alimentador automático autónomo terminado.

Figura 33, Estructura física terminada con todo el *hardware* montado sobre él, y con el *software* cargado.

CAPITULO III

3. MARCO DE RESULTADOS

En este capítulo se trata los resultados obtenidos en las pruebas realizadas a lo largo de la implementación como la ejecutadas en el campo con el alimentador automático autónomo terminado, se da una breve explicación de la tabulación de datos, determinación de pesos, resultados de funcionamiento, inversión del alimentador, y se culmina con una reseña de la evaluación final del alimentador automático autónomo.

Entre las pruebas realizadas los lapsos fueron diferentes, así pues las pruebas de medición del nivel de alimento en el contenedor y las pruebas de funcionamiento se realizaron por tres días, las pruebas de dosificación, esparcido y hora de alimentación se realizaron por una semana, las pruebas de envío de mensajes se realizaron por un mes, las pruebas de menús, manejo, control, almacenamiento, recuperación y actualización de datos se realizaron por tres meses, y finalmente las pruebas de carga y descarga de la batería se ejecutaron por una semana.

3.1. TABULACIÓN DE DATOS

La tabulación nos permite visualizar la valides del funcionamiento del alimentador automático autónomo en los parámetro medibles, dichos valores al ser procesados estadísticamente muestra el error siendo este el que cualificará el intervalo de confianza.

Para la realización de la prueba t *student* se tomó diez medidas de cada tamaño de alimento, haciendo referencia a cuatro valores establecidos en la tabla1, se obtuvo el promedio de estos valores para contrastarlos con los establecidos en la tabla1.

La figura 34 muestra los valores promedio de los pesos medidos y deseados.

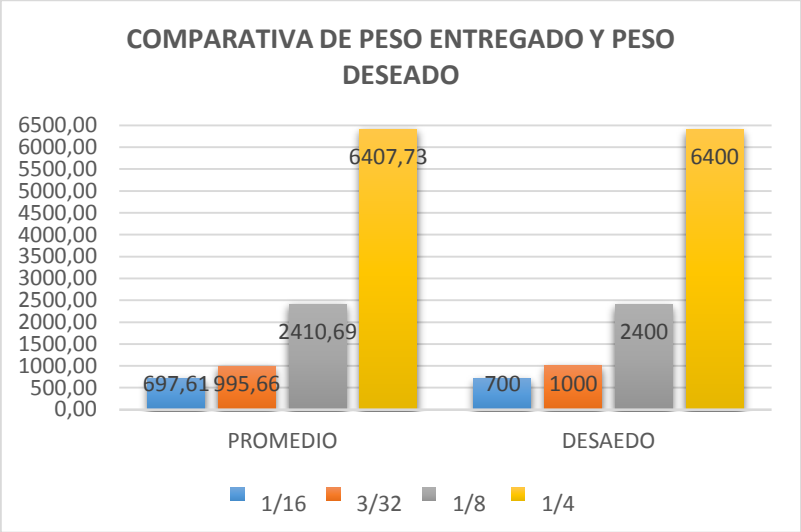


Figura 34, Comparativa de los promedios de los pesos obtenidos vs los deseado.

Fuente: Villavicencio 2016

Los pesos mostrados en la figura 34 se utilizaron para la ejecución de la prueba t *student* con grado de confianza del 95%, con las hipótesis nula Ho y alterna Ha.

Ho = no hay diferencia entre medias de los pesos de alimento balanceado en las diferentes medidas.

Ha = si hay diferencia entre medias de los pesos de alimento balanceado en las diferentes medidas.

La prueba realizada en la aplicación Excel nos muestra los datos detallados en la tabla 6.

PRUEBA T PARA DOS MUESTRAS SUPONIENDO VARIANZAS IGUALES		
	DESAEDO	*PROMEDIO*
Media	2625	2627,92136
Diferencia hipotética de las medias	0	
Grados de libertad	6	
Estadístico t	-0,00157	
P(T<=t) dos colas	**0,99880**	
Valor crítico de t (dos colas)	2,44691	

Tabla 6, Valores de la t *student* para los pesos entregados por el alimentador automático autónomo.

Fuente: Villavicencio 2016

De los valores obtenidos en la tabla 4 y comprobados en la figura 35 se puede concluir que: al obtener que P (T<=t) es > que el alfa (0,05) inferimos que no hay diferencia estadística, por lo tanto

se rechaza la hipótesis nula y se acepta la hipótesis alterna, es decir, la media de los pesos de alimento balanceado en las diferentes medidas son iguales.

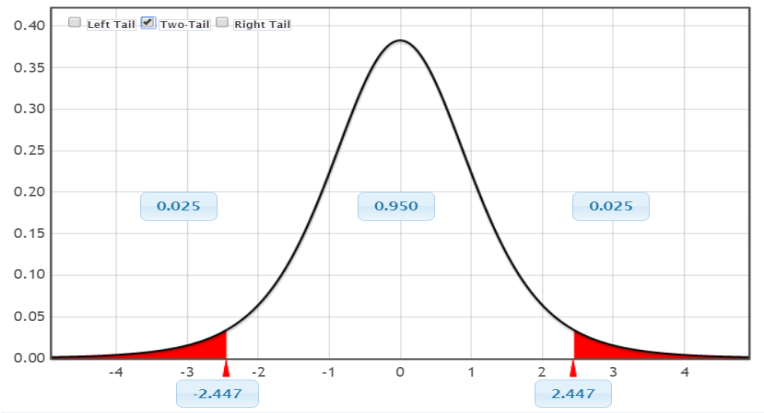


Figura 35, Curva de la t *student* aplicada a los valores obtenidos en la tabla 4.

Fuente: Villavicencio 2016

El alimentador automático autónomo tiene un error máximo de **0,45%** y un error mínimo de **0,12%** con relación a los valores establecidos en la tabla 1.

3.1.1. DETERMINACIÓN DE PESOS

En la implementación del dosificador se realizó la medición del alimento que se extrae mediante el tornillo al girar 360°, para poder tomar un valor que sea de mayor utilidad al momento de ejecutar la dosificación, se procedió a tomar **15 muestras** de cada tamaño de alimento *pellet* balanceado, para lo que se calculó el promedio de los valores obtenidos, siendo el promedio de cada tamaño de alimento el que se utiliza en el *software* según sea requerido.

En la tabla 7 se muestra los diferentes pesos medidos para el tamaño de alimento 1/16 con el dosificador al dar una vuelta el tornillo sin fin.

N°	ALIMENTO 1/16 EN g
1	40,66
2	40,92
3	39,55
4	40,13
5	43,13
6	40,73

N°	ALIMENTO 1/16 EN g
7	40,83
8	40,68
9	40,71
10	40,57
11	40,63
12	40,24
13	40,35
14	40,72
15	40,65
Promedio	**40,7**

Tabla 7, Pesos obtenidos en alimento de 1/16 en gramos.

Fuente: Villavicencio 2016

En la tabla 8 se muestra los diferentes pesos medidos para el tamaño de alimento 3/32 con el dosificador al dar una vuelta el tornillo sin fin.

N°	ALIMENTO 3/32 EN g
1	35,55
2	35,73
3	37,32
4	37,01
5	34,94
6	34,99
7	35,49
8	35,57
9	35,41
10	35,39
11	35,53
12	35,45
13	35,14
14	35,16
15	35,12
Promedio	**35,59**

Tabla 8, Pesos obtenidos de alimento de 3/32 en gramos.

Fuente: Villavicencio 2016

En la tabla 9 se muestra los diferentes pesos medidos para el tamaño de alimento 1/8 con el dosificador al dar una vuelta el tornillo sin fin.

N°	ALIMENTO 1/8 EN g
1	34,73
2	35,25
3	37,32
4	34
5	36,99
6	34,54
7	35,2
8	35,89
9	34,44
10	35,63
11	37,72
12	35,42
13	35,36
14	35,41
15	35,16
Promedio	**35,54**

Tabla 9, Pesos obtenidos de alimento de 1/8 en gramos.

Fuente: Villavicencio 2016

En la tabla 10 se muestra los diferentes pesos medidos para el tamaño de alimento 1/4 con el dosificador al dar una vuelta el tornillo sin fin.

N°	ALIMENTO 1/4 EN g
1	33,64
2	34,25
3	33,61
4	34,23
5	34,72
6	34,12
7	34,47
8	33,89
9	33,97

N°	ALIMENTO 1/4 EN g
10	34,63
11	34,51
12	34,43
13	34,37
14	33,67
15	34,16
Promedio	**34,18**

Tabla 10, Pesos obtenidos de alimento de 1/4 en gramos.
Fuente: Villavicencio 2016

En la figura 36 se observa una de las pruebas de dosificación de peso.

Figura 36, Pruebas de dosificación de peso.
Fuente: Villavicencio 2016.

3.2. RESULTADOS DE FUNCIONAMIENTO.

En esta sección se trata los resultados obtenidos en las pruebas ejecutadas en el campo, se da una breve explicación de la entrega de alimento, dispersión de los *pellets*, envío de mensajes de texto, alimentación en los horarios requeridos, recuperación de los datos ingresados por el usuario, consumo de potencia eléctrica, y se culmina con la potencia entregada por los paneles solares,

3.2.1. ENTREGA DE ALIMENTO.

Los resultados en la implementación del dosificador automático autónomo muestran que se logró entregar la cantidad de alimento balanceado de las diferentes medidas con un peso muy cercano a

los requeridos por la tabla 1, tabla de alimentación basada en el porcentaje de biomasa para 1000 peces.

Para la comprobación de la entrega del alimento lo más cercano a lo mostrado en la tabla 1, se midieron los pesos entregados por el alimentador para cuatro pesos deseados en la citada tabla.

La tabla 11 muestra los valores de los pesos obtenidos con el dosificador y la comparación del promedio de esos pesos con los pesos deseados de la tabla 1.

PESOS ENTREGADOS POR EL ALIMENTADOR AUTOMÁTICO AUTÓNOMO				
N°	TAMAÑOS DE *PELLETS*			
-	1/16	3/32	1/8	1/4
1	690,24	985,31	2380,43	6388,63
2	700,38	986,72	2520,22	6454,16
3	700,56	1000,82	2296,02	6345,56
4	702,28	1003,07	2497,93	6360,54
5	699,70	998,56	2332,49	6484,12
6	700,21	998,00	2377,06	6461,65
7	697,80	1001,95	2423,65	6446,67
8	698,84	999,69	2325,73	6435,44
9	692,13	990,95	2406,09	6304,37
10	694,02	991,51	2547,23	6396,12
PROMEDIO	697,61	995,66	2410,69	6407,73
DESAEDO	700	1000	2400	6400

Tabla 11, Comparación de pesos entregados y deseados.

Fuente: Villavicencio 2016

3.2.2. DISPERSIÓN DE LOS *PELLETS*.

Para lograr una mayor dispersión de los *pellets* se acelera el motor *brushless* al máximo para lograr alcanzar la mayor distancia posible desde la base del alimentador hasta el punto en que los *pellets* alcanzan el espejo de agua del estanque, dando como resultado una longitud de 6 metros de distancia y 8 metros de abanico en las pruebas como en el funcionamiento.

La figura 37 muestra el esparcido de los *pellets* en el estanque.

Figura 37, Esparcido del alimento en a la piscina.
Fuente: Villavicencio2016.

3.2.3. ENVÍO DE MENSAJES DE TEXTO.

El envío de mensajes se ejecutó de forma limpia y rápida, realizando el envío de cada mensaje en el momento requerido de igual manera como sucedió en las pruebas respectivas, teniendo un retardo desde su envío hasta su recibo de **90 segundos** el mayor y de **10 segundos** el menor.

La figura 38 muestra la llegada del mensaje al celular de destino.

Figura 38, Envío y recepción de mensajes de texto.
Fuente: Villavicencio 2016.

3.2.4. ALIMENTACIÓN EN LOS HORARIOS REQUERIDOS.

La ejecución de la alimentación se realizó a la hora establecida en el horario introducido en el *software*, gracias al RTC que maneja la hora y fecha de forma independiente, permitiendo a la rutina de alimentación consultar si la hora actual es la de hora de alimentación en el horario, la ejecución de la alimentación se inicia con un retardo de alrededor de **26 milisegundos**, debido a la ejecución de las rutinas de decisión y ejecución.

La figura 39 muestra la pantalla principal en el momento que se inicia la alimentación.

Figura 39, Hora de alimentación ejecutada y horario en el que se debe dar.
Fuente: Villavicencio 2016.

3.2.5. RECUPERCIÓN DE LOS DATOS INGRESADOS POR EL USUARIO

En la ejecución del trabajo del alimentador se pudo comprobar que los datos ingresados por el usuario permanecen inmutables, gracias a que estos son almacenados en la memoria EEPROM, dando como resultado la normal ejecución de los cálculos y por tanto del normal desarrollo de cada una de las rutinas que necesitan de estos datos, como se observa en la figura 40.

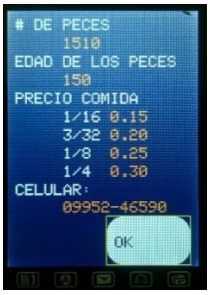


Figura 40, Recuperación de los datos almacenados en la EEPROM.
Fuente: Villavicencio 2016.

3.3. CONSUMO DE POTENCIA ELÉCTRICA

Al estar el alimentador automático autónomo alimentado por la energía provista por la batería de ion de Litio y esta a su vez cargada por los paneles solares, es indispensable evitar el consumo de energía de forma innecesaria como excesiva, siendo necesaria una batería con la capacidad de alimentar el conjunto de subsistemas del alimentador, para lo cual se calculó las potencias consumidas por cada subsistema y se comparó con la potencia que la batería almacena para asegurar el funcionamiento del alimentador, siendo el **motor *brushless*** el de mayor consumo con **2.63A**, y el **RTC** el de menor consumo con **0.0003A**, como se muestra en la tabla 12.

CONSUMO MINIMO Y MÁXIMO DE LOS COMPONENTES ELECTRÓNICOS									
	PASO	***BRUSHLESS***	**GSM**	**Arduino**	**RTC**	**ESC**	**L298N**	**HC-SR04**	**LCD**
REPOSO(R)	0	0,04	1,5	2,21	0,0003	3	37,5	1,8	15
ACTIVO(A)	2500	2630	400	121,42	0,0003	4	37,5	15	300
VOLTAJE	11,1	11,1	5	5	3	5	5	5	3,3
POTENCIA R	0	0,444	7,5	11,0325	0,0009	15	187,5	9	49,5
POTENCIA A	27750	29193	2000	607,1	0,0009	20	187,5	75	990
TOTAL R	0,28	W/h							
TOTAL A	60,82	W/h							
BATERÍA	58	W/h							

Tabla 12, Consumo de potencia de componentes vs carga de la batería.

Fuente: Villavicencio 2016

3.3.1. CONSUMO A MÁXIMA CARGA

El alimentador debe trabajar a cabalidad en cualquiera de las posibles configuraciones, siendo la más exigente de estas cuando el alimentador debe alimentar al número máximo de peces que permite su configuración, el alimentador tiene como límite de manejo la cantidad de 3000 peces.

De esta manera el consumo máximo de energía se suscita en esta configuración; consumiendo **7,94 vatios** al día en estado activo y **6,68 vatios** al día en estado de reposo, dando un total de **14,62 vatios** diarios lo que en perspectiva es menos que el consumo de una bombilla fluorescente compacta de **20 vatios** en una hora.

La tabla 13 muestra la potencia consumida por el alimentador cuando alimenta a 3000 peces, pudiéndose notar que no excede la potencia de la batería.

CONSUMO DIARIO MÁXIMO EN 3000 PECES			
	ACTIVO	**REPOSO**	**UNIDADES**
TIEMPO DIARIO	0,13	23,87	Horas
CONSUMO DIARIO EN ACTIVO	7,94		Watts
CONSUMO DIARIO EN REPOSO		6,68	Watts
CONSUMO DIARIO TOTAL		**14,62**	**W/día**

Tabla 13, Consumo diario con 3000 peces (máxima capacidad).

Fuente: Villavicencio 2016

3.3.2. POTENCIA MÍNIMA ENTREGADA POR LOS PANELES SOLARES.

Al estar el alimentador en el sitio de trabajo los paneles solares deben ser capaces de cargar a la batería a pesar de tener condiciones relativamente adversas, por lo que se calculó la potencia mínima que los paneles entregan a lo largo del día.

Potencia mínima cargada con el **40%** del tiempo útil del día para la carga de la batería, se obtuvo que es de **12 W/día**, mismos que son suficientes para mantener la carga de la batería en niveles necesarios hasta que las condiciones climáticas permitan un porcentaje mayor de tiempo útil para la carga.

Otro dato indispensable de conocer es el tiempo máximo de trabajo del alimentador sin que la batería sea cargada por la energía entregada por los paneles solares; dato que se calculó en **3,94 días** de trabajo sin cargar la batería y al límite de la capacidad del alimentador.

En la figura 41 se muestra los paneles cargando la batería.

Figura 41, Carga de la batería en un día común.
Fuente: Villavicencio 2016.

3.3.3. INVERSIÓN DEL ALIMENTADOR

El alimentador automático autónomo tuvo una inversión de **770,8 USD** con lo que se demuestra ser de bajo costo, siendo el elemento de mayor costo el metacrilato de 3mm con **125 USD**, y el de menor costo los tornillos con **2 USD**, como se observa en la tabla 14.

ANÁLISIS DE COSTOS	
ÍTEM	COSTO
METACRILATO 3mm	125
TUBOS DE 1/2'	14,8
PANELES SOLARES 9V, 3W	48
MOTOR A PSOS	25

ANÁLISIS DE COSTOS	
ÍTEM	COSTO
MOTOR BRUSHLESS	55
L298N	15
ESC	20
BATERÍAS ION DE LITIO 3.4V, 2,6A	21
HC-SR04	10
PANTALLA TFT 2.4'	20
GPRS/GSM	73
CHIP MOVIL	3
RTC	11
CAJAS PLÁSTICAS	29
CABLES	19
TARJETA PERFORADA	2,5
TORNILLOS	2
PEGAMENTOS	21
SUELDA	7
PINTURA	12
ACOPLES METALICOS	10
EACCESORIOS	19,5
CORTE Y DOBLADO	67
CIRCUITO DE ACONDICIONAMIENTO	10
ARDUINO MEGA 2560	73
TRANSPORTES Y ENCOMIENDAS	58
TOTAL	**770,8**

Tabla 14, Análisis de costos de la implementación del alimentador.

Fuente: Villavicencio 2016.

3.4. EVALUACIÓN FINAL DEL ALIMENTADOR AUTOMÁTICO AUTÓNOMO

El dosificador automático autónomo tiene la capacidad de dosificar cualquier alimento balanceado tipo *pellet* que esté dentro de los tamaños que maneja, siendo balanceados para especies acuáticas, como los de truchas, tilapias, camarones, langostas, para los cuales será necesario ejecutar la calibración previa para cada uno de estos alimentos.

3.5. CONCLUSIONES

- Se implementó un alimentador automático autónomo para peces aplicando el método de dosificación volumétrica para la entrega del alimento.
- La estructura del alimentador es resistente y liviana soportando sobre los 100 kg de peso.
- La estructura del alimentador es liviana pesando 23,5 libras.
- Integra energía renovable para su funcionamiento.
- El consumo de potencia a carga máxima es de 7,94 vatios.
- El consumo máximo diario es de 14,62 vatios.
- El alimentador automático autónomo durante las pruebas de alimentación entregó a la hora establecida con un retraso de 26 milisegundos aproximadamente.
- Es capaz de trabajar sin volver a cargar la batería por 3,94 días.
- Su error máximo en la entrega de alimento es de 0,45%.
- El costo de implementación es de 770,8 USD.

3.6. RECOMENDACIONES

- Sería posible incorporar un sensor de temperatura del agua para establecer un margen de incremento o decremento de la cantidad de alimento por la incidencia de esta en la ingesta.
- El ingreso de datos podría hacerse desde el dispositivo móvil.
- La integración de módulos de comunicación inalámbrica para que se comuniquen entre alimentadores.
- La integración de un módulo GPS para su ubicación en caso de robo.
- La medición del PH del agua por medio de un sensor para el trabajo.

4. ANEXOS

4.1. ANEXO 1: MANUAL DE USAURIO

Para el correcto funcionamiento del alimentador son necesarios varios datos que el usuario debe ingresar; como son la edad y la cantidad de animales, número de teléfono celular al cual enviarán los mensajes desde el alimentador, precio de cada tamaño de alimento *pellet* por kilogramo; adicionalmente es necesario poner la hora actual en el reloj del *software* para que se entregue el alimento en las horas necesarias ya determinadas en los horarios previamente seleccionados por el *software*.

Dentro del menú del alimentador se puede consultar los datos ingresados por el usuario para su comprobación previo a la ejecución de la alimentación, de ser necesario corregir o modificar algún dato o todos, se procede a ingresar nuevamente el dato a cambiar en el menú, el *software* guarda los datos ingresados al dar clic en *OK*, caso contrario mantiene el valor anterior de este dato.

Una vez establecido los horarios y la cantidad de alimento a entregar, es necesario presionar el botón *START* para que comience la ejecución de la rutina de alimentación; el alimentador suspende todos los sistemas innecesarios en el tiempo de espera, y los activa en el momento de ejecutar la alimentación, administrando de esta forma la energía eficientemente.

4.1.1. MENÚ DE CONFIGURCIÓN.

La pantalla de menú principal que se ve en la figura 23, muestra la hora y fecha actual, el botón para el inicio de la producción *START*, el botón MENU de ingreso de valores y datos, mismo en el que se puede ingresar los datos necesarios para el funcionamiento del alimentador automático autónomo, el botón DATOS permite consultar datos propios del alimentador automático autónomo y del propietario.

Figura 42, Pantalla de inicio del HMI, menú principal de configuración.
Fuente: Villavicencio 2016

4.1.2. SUBMENÚ 1.

En la figura 24 se observa el submenú 1 y el teclado de ingreso del precio del kilo de alimento de tamo 1/16, el submenú 1 contiene las opciones de ingreso de precio de alimento, consulta de los datos ingresados, y modificar (ingresar) los datos necesarios.

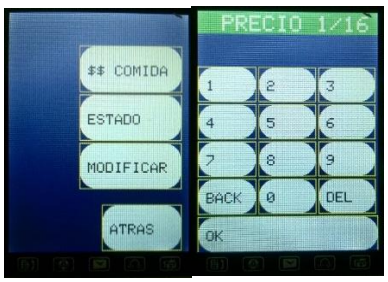


Figura 43, Submenú 1($$ COMIDA, ESTADO, MODIFICAR), y teclado de ingreso de costo por alimento
Fuente: Villavicencio 2016

En la figura 25 se muestra la pantalla con los datos que se han ingresado al alimentador automático autónomo desde las opciones $$ COMIDA y MODIFICAR, datos que se utilizan para el cálculo de horarios, costos y cantidad a entregar de alimento, así como el número de teléfono al que enviaran los mensajes.

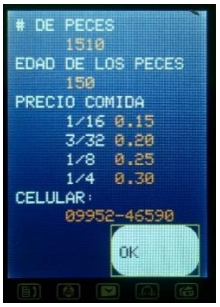


Figura 44, Pantalla del submenú ESTADO.
Fuente: Villavicencio 2016

4.1.3. SUBMENÚ 1.1

La figura 26 muestra el submenú 1.1, que consta de las opciones PRODUCCION, HORA/FECHA, # CELULAR, mismas que permiten el ingreso de los datos correspondientes a cada opción.

Figura 45, Submenú 1.1, dentro de MODIFICAR.
Fuente: Villavicencio 2016

La figura 27 muestra el submenú 1.1.1, que permite el ingreso de la cantidad de peces y la edad en la que se encuentran al iniciar la alimentación con el alimentador automático autónomo.

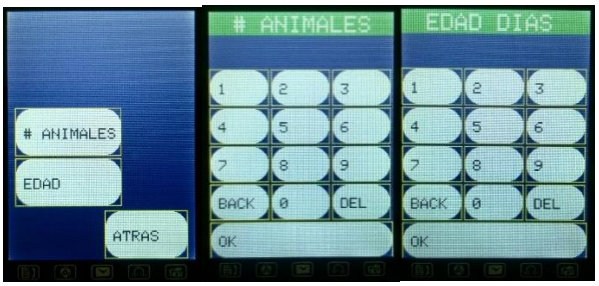


Figura 46, Submenú 1.1.1 dentro de PRODUCCIÓN, ingreso de edad y número de animales.
Fuente: Villavicencio 2016

En la figura 28 se muestra las pantallas de ingreso e igualación de la hora y fecha del reloj que se muestra en la pantalla de menú principal y en el de trabajo, los datos ingresados aquí se cargan en el RTC ds107, el cual mantiene actualizados estos datos.

Figura 47, Submenú 1.1.2 dentro HORA/FECHA, ingreso de hora y fecha.
Fuente: Villavicencio 2016

La figura 29 muestra la pantalla de ingreso del número de celular al que se enviaran los mensajes de texto, opción que se encuentra en el submenú 1.1.

Figura 48, Teclado de ingreso de número de celular, dentro de # CELULAR.
Fuente: Villavicencio 2016

4.1.4. SUBMENÚ 2.

La figura 30 muestra el submenú 2, que consta de las opciones EQUIPO Y PROPIETARIO, además las pantallas de cada opción, con los datos de fábrica y los propios de cada propietario.

Figura 49, Submenú 2 (EQUIPO, PROPIETARIO), y sus pantallas correspondientes.
Fuente: Villavicencio 2016

4.1.5. MENÚ DE FUNCIONAMIENTO (*START*)

La figura 31 muestra la pantalla de trabajo (*START*) la que posee los botones *STOP* para parar el proceso de alimentación y poder cambiar datos, HORARIO que muestra las horas de alimentación, y RESUMEN que muestra los datos calculados como: tamaño de alimento que se está entregando, número de comidas a día, tiempo que se está alimentando en semanas, costo del alimento entregado durante el tiempo de producción y el peso que debería tener el pez.

Figura 50, Menú principal en funcionamiento, y pantalla de resumen del proceso de alimentación.

Fuente: Villavicencio 2016

4.1.6. CONTROL Y TIPOS DE DATOS.

Al necesitar el *software* de múltiples datos y de diferentes clases se hace necesario que el *software* sea capaz de tratar y controlar el ingreso de cada uno de estos datos, de tal manera se muestra en las siguientes figuras las pantallas en las que se ingresa estos datas, siendo controlados en tamaño y rango, el teclado permite guardar el dato ingresado con el botón OK, borrar el ultimo digito ingresado con el botón DEL, y salir del teclado sin guardar con el botón BACK.

La figura 32 muestra el ingreso de los precios del kilogramo de balanceado desde el de 1/16 hasta el de 1/4, valor que es tratado como un número con decimales.

Figura 51, Manejo de valores decimales.

Fuente: Villavicencio 2016

La figura 33 muestra el ingreso de la cantidad de animales y su edad, valores que son tratados como números enteros.

Figura 52, Manejo de valores enteros.
Fuente: Villavicencio 2016

La figura 34 muestra el ingreso del número de teléfono, valor en el que es necesario el número cero precediendo a los demás y que debe poseer diez dígitos en su conjunto.

Figura 53, Manejo de números de teléfono celular.
Fuente: Villavicencio 2016

RECOMENDACIONES

- Para controlar un proceso, es indispensable investigar el proceso y los factores que lo afectan, siendo necesario el conocimiento de los equipos y sistemas que se emplearán para dicho control.
- Tener en consideración los rangos y valores de energía que el alimentador automático autónomo manejará en cada parte y/o subsistema para conseguir un diseño que cumpla con las necesidades energéticas del alimentador automático autónomo.
- Con el fin de evitar gastos adicionales de tiempo y recursos es beneficioso ejecutar las simulaciones de cada diseño realizado para comprobar su funcionamiento.
- No manipular los actuadores cuando estos se encuentran trabajando, evitando de esta forma daños al alimentador automático autónomo como a las personas.

- No poner el alimentador en modo *START*, cuando haya estado guardado o lejos de la incidencia de los rayos del sol, previamente dejar a la luz solar por uno o dos días para la recarga de la batería.
- Ingresar cada uno de los datos requeridos por el *software* para que el funcionamiento y resultados sean los esperados.
- Manipular la pantalla táctil con delicadeza y precaución para evitar la reducción de su vida útil y daños en ella.

4.2. ANEXO 2: CÓDIGO FUENTE DEL *SOFTWARE* DE APLICACIÓN

```
//#define DEBUG
#include <String.h>
#include <Adafruit_GFX.h>    // Libreria de graficos
#include <Adafruit_TFTLCD.h> // Libreria de LCD
#include <SD.h>              // Libreria de tarjeta SD
#include <SPI.h>             // Libreria bus SPI

#include <avr/pgmspace.h>
#include <TouchScreen.h>     // Libreria del panel tactil
#include <EEPROM.h>
#include <Time.h>            // librerias para manejar hora y fecha
#include <Wire.h>
#include <DS1307RTC.h>  // a basic DS1307 library that returns time as a time_t

#include <SoftwareSerial.h>     // LIBRERIA PARA EL GSM
SoftwareSerial SIM900(22,23);

#define LCD_CS A3   // Definimos los pines del LCD
#define LCD_CD A2   // para poder visualizar elementos graficos
#define LCD_WR A1
#define LCD_RD A0
#define LCD_RESET A4
```

-
- #if defined __AVR_ATmega2560__ // Para Arduino Uno/Duemilanove, conectamos la tarjeta SD en los pines del puerto SPI
- #define SD_SCK 13 // que se corresponden con los pines MOSI -> 11, MISO -> 12 y SCK -> 13
- #define SD_MISO 12
- #define SD_MOSI 11
- #endif
-
- // Chip Select del bus SPI correspondiente a la conexion con la tarjeta SD
- #define SD_CS 10
-
- // En la tarjeta SD debemos colocar imagenes en formato BMP de 24 Bits!
- // Otro tipo de formato de imagen no se puede visualizar por pantalla.
-
- Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, A4); // Instancia LCD
-
- // Pines necesarios para los 4 pines del panel tactil
- #define YP A1 // Pin analogico A1 para ADC
- #define XM A2 // Pin analogico A2 para ADC
- ///°°°!!!!! EL PIN "A2 y A1" SE REPITEN EN LAS DEFINICIONES UTILIZAR OTROS PIN,,, COMO???
- #define YM 7
- #define XP 6

```
•
• // Definimos la presion máxima y minima que podemos realizar sobre el panel
• #define MINPRESSURE 1
• #define MAXPRESSURE 1000
•
• // Para mejor precision de la presion realizada, es necesario
• // medir la resistencia entre los pines X+ y X-.
• // En Shield TFT 2.4" LCD se mide entre los pines A2 y 6
• // Instancia del panel tactil (Pin XP, YP, XM, YM, Resistencia del panel)
• TouchScreen ts = TouchScreen(XP, YP, XM, YM, 346);// ESTABA 364
•
• short TS_MINX = 206; // 150 Coordenadas del panel tactil para delimitar
• short TS_MINY = 89; // 120 el tamaño de la zona donde podemos presionar
• short TS_MAXX = 910; //850 y que coincida con el tamaño del LCD
• short TS_MAXY = 950; //891
•
• #define
                                                                    BL
  ACK   0x0000  // Definimos los colores
• #define
                                                                    BL
  UE    0x001F  // que utilizaremos para
• #define
                                                                    RE
  D     0xF800  // el texto y los elementos graficos
```

- #define GREEN 0x07E0
- #define CYAN 0x07FF
- #define CELESTE 0x075F
- #define MAGENTA 0xF81F
- #define YELLOW 0xFFE0
- #define NARANJA 0xFBC0
- #define ROSA 0xFBEA
- #define WHITE 0xFFFF
- #define FONDO 0x0005
- ////>>>>> DEFINICION DEL PIN PARA GSM <<<<///
- #define pwrkey 24
-
- ////>>>>> DEFINICION DE PINES PARA ULTRASÓNICO <<<<///
- #define trigPin 30
- #define echoPin 31
-
- /////(((((DEFINICION DE VARIABLES, CONSTANTES Y TIPOS DE DATOS))))))/////
- // Variables que almacenaran la coordenada
- //long lmax;
- int X,Y; // X, Y donde presionemos y la variable Z
- int Z; // almacenara la presion realizada

```
byte op,op_del,ban1=0,hm=0,hora=0,minuto=0,dia=1,mes=12,backlight=52, i_etiqueta;
word  lmax,lmin,anio=2016;
boolean a=1,bancell=0; // variable "flag" para control rebotes a
String cell,aux, aux1;
tmElements_t tm,t1;
word matbot[39][4]={{1, 75, 75, 45}, //BOTON 0
                    {81, 75, 75, 45},
                    {161, 75, 75, 45},
                    {1, 125, 75, 45},
                    {81, 125, 75, 45},
                    {161, 125, 75, 45},
                    {1, 175, 75, 45},
                    {81, 175, 75, 45},
                    {161, 175, 75, 45},
                    {1, 225, 75, 45},
                    {81, 225, 75, 45},//BOTON 10
                    {161, 225, 75, 45},
                    {1, 275, 238, 45},//BOTON OK
                    {10, 258, 105, 60},
                    {125, 258, 105, 60},
                    {40,140,160,90},   // START <<---//BOTON 15
                    {95,45, 135, 60},
```

- {95, 110, 135, 60},//192
- {95, 175, 135, 60},//258
- {125, 258, 105, 60},//258
- {10, 125, 135, 60}, // BOTON 20
- {10, 190, 135, 60},
- {125, 258, 105, 60},
- {125, 258, 105, 60},
- {10, 45, 135, 60},
- {10, 110, 135, 60}, // BOTON 25
- {10, 175, 135, 60},
- {125, 258, 105, 60},
- {10, 192, 60, 60},
- {80, 192, 60, 60},
- {165, 192, 60, 60}, //BOTON 30
- {10, 258, 105, 60},
- {125, 258, 105, 60},
- {10, 60, 135, 60},
- {10, 125, 135,60},
- {125, 258, 105, 60},//BOTON 35
- {10, 258, 105, 60}, // HORARIO
- {125, 258, 105, 60}, //BOTON RESULTADOS
- {40,140,160,90}}; // STOP <<---
-

```
char *nomb[]= {"1","2","3","4","5","6","7","8","9",
               "BACK","0","DEL","OK",
               "MENU","DATOS","START",
               "$$ COMIDA","ESTADO","MODIFICAR","ATRAS",
               "# ANIMALES","EDAD","ATRAS",
               "OK",
               "PRODUCCION","HORA/FECHA","# CELULAR","ATRAS",
               "+","-","-->","OK","ATRAS",
               "EQUIPO","PROPIETARIO","ATRAS",
               "HORARIO","RESUMEN","STOP"};


void setup()
{
  Serial.begin(9600);
  Serial.println("inicio");
  tft.reset();
  tft.begin(0x9341); // Iniciamos el LCD especificando el controlador ILI9341.
                     // (Otros controladores disponibles: HX8357D, HX8347G, ILI9328, ILI9325)


 //#endif //////////////////°°°°° FIN DEL SETUP DE SD°°°°°////////

```

- ////////////////++++++++ INICIO DEL SETUP DE TOUCH ++++++////////
- //tft.fillScreen(0xFBC0);/////--- PONE LA PANTALLA NARANJA
- //pinMode(13, OUTPUT);
- tft.setRotation(0); // Establecemos la posición de la pantalla Vertical u Horizontal
-
- ///<<<< SETUP PARA EL RELOJ >>>
- Wire.begin();// Inicia el puerto I2C
- //RTC.begin();// Inicia la comunicación con el RTC
- //RTC.adjust(DateTime(__DATE__, __TIME__)); // Establece la fecha y hora (Comentar una vez establecida la hora)
-
- setSyncProvider(RTC.get); // Vamos a usar el RTC
- //setTime(hora,minuto,00,dia,mes,2015); // Las 0:0:00 del dia 1 de Dic de 2015
- //RTC.write(hora,minuto,00,dia,mes,2015);
- if (timeStatus() != timeSet)
- tft.println("Unable to sync with the RTC");
- else
- {
- tft.println("RTC has set the system time");
- }
-
- ///<<<< SETUP PARA EL GSM >>>
- pinMode(24, OUTPUT);
- SIM900.begin(19200);//INICIAMOS EL GSM

```
 
        ///<<<< SETUP PARA EL ULTRASÓNICO  >>>
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
 
                                    //PARA LIMPIAR LA EEPROM ///
                                      //for(byte i=1;i<14;i++)
                                        // eepromClear(i);
  // PONE EL COLOR DEL FONDO
  tft.fillScreen(FONDO);
 /* pinMode(backlight,OUTPUT);
   digitalWrite(backlight,HIGH);
   delay(200);
    digitalWrite(backlight,LOW);*/
}////////////++++++++ FIN SETUP +++++////////
 
void loop()
{
  makeScreen(1,BLACK,WHITE); ///;;;; CREA LA PANTALLA DENTRO DE INICIO [MENU, DATOS]
   // digitalWrite(backlight,HIGH);
   do
     {
       printclock(0);
```

```
        leeBoton(1);
      } while(!(op >= 13 && op <= 15));

    switch (op)
     {
       case 13: do //;;; MANTIENE EN LA PANTALLA ACTUAL
              {
       ///;;;; CREA LA PANTALLA DENTRO DE MENU [NUEVO,ESTADO,MODIFICAR,ATRAS]
                clrprintclock(); //:: BORRA EL RELOJ
                clearBoton(op_del); //:: BORRA LOS BOTONES
                makeScreen(2,BLACK,WHITE);
                do
                 {
                  leeBoton(2);
                 } while(!(op >= 16 && op <= 19));
                switch(op)
                 {
                   case 16: clearBoton(op_del); //:: BORRA LOS BOTONES
                            makeScreen(0,BLACK,WHITE);///;;;; CREA LA PANTALLA DE TECLADO NUMERICO
                            i_etiqueta = 4;
                            //aux = "0.00";//String(precio);
                            do //;;; MANTIENE EN LA PANTALLA ACTUAL (TECLADO)
```

```
                {
                ///<<< INGRESA EL PRECIO DE LAS COMIDAS 1/16, 3/32, 1/8, 1/4 >>>///
                etiquetaTeclado(i_etiqueta);        /// HACE QUE SE IMPRIMAN LAS ETIQUETAS DE TAMAÑO COMIDA
                do
                {
                  leeTeclado(4);
                } while(!(op == 9 || op == 12)); //SALE EN CASO DE PRESIONAR 'BACK' U 'OK'
                if(op == 12)
                {
                  writeEeprom(i_etiqueta);   //>>> GUARDA EN LA EEPROM
                  if(i_etiqueta != 7)
                    op=16;
                }
                i_etiqueta += 1;
              }while (!(i_etiqueta > 7 || op == 9));
        ///;;; SALE DEL do while CUANDO SE HAYA PRESIONADO 'BACK' O YA SE HAYA INGRESADO LOPRECIOS DE TODAS LAS COMIDAS
                tft.fillScreen(FONDO);//::: LIMPIA LA PANTALLA
            break;
            case 17: ///;;;; CREA LA PANTALLA DENTRO DE ESTADO [OK]
                clearBoton(op_del); //:: BORRA LOS BOTONES
                makeScreen(4,BLACK,WHITE);
```

```
                    informativo(1);
                    do //;;; MANTIENE EN LA PANTALLA ACTUAL
                      {
                        leeBoton(4);
                      } while (op != 23);///;;; SALE DEL do while CUAN DO SE HAYA PRESIONADO EL BOTON 'OK'
                    tft.fillScreen(FONDO);
                break;
                case 18:  //  CREA LA PANTALLA DENTRO DE MODIFICAR [PRODUCCION, HORA/FECHA, #CELULAR, ATRAS]
                    do //;;; MANTIENE EN LA PANTALLA ACTUAL
                      {

            ///;;;;  CREA LA PANTALLA DENTRO DE MODIFICAR [PRODUCCION, HORA/FECHA,#CELULAR,ATRAS]
                      clearBoton(op_del); //:: BORRA LOS BOTONES
                      makeScreen(5,BLACK,WHITE);
                      do
                        {
                          leeBoton(5);
                        } while(!(op >= 24 && op <= 27));
                      switch (op)
                        {
                          case 24:  //LLAMA A PANTALLA DE PRODUCCION [# ANNIMALES, EDAD, ATRAZ]
```

```
•
•                                   do //;;; MANTIENE EN LA PANTALLA ACTUAL
•                                   {
•                                ///;;;; CREA LA PANTALLA DENTRO DE PRODUCCIOM [# DE ANIMALES, EDAD, ATRAS]
•                                   clearBoton(op_del); //:: BORRA LOS BOTONES
•                                   makeScreen(3,BLACK,WHITE);
•                                   do
•                                     {
•                                       leeBoton(3);
•                                     } while(!(op >= 20 && op <= 22));
•                                   //if(op!=21)  //CONTROLA  QUE  NO  HAYA  SIDO PRESIONADO ATRAS
•
•                                   switch (op)
•                                    {
•                                           ///<<< INGRESA EL # DE ANIMALES ///
•                                     case 20:  clearBoton(op_del); //:: BORRA LOS BOTONES
•                                           makeScreen(0,BLACK,WHITE);///;;;; CREA LA PANTALLA DE TECLADO NUMERICO
•
•                                           etiquetaTeclado(1);
•                                            do
```

```
                                                {
                                                  leeTeclado(1);
                                                } while(!(op == 9 || op == 12)); //SALE EN CASO DE ´RESIONAR 'BACK' U 'OK'
                                              if(op==12)
                                                 writeEeprom(1);
                                              tft.fillScreen(FONDO);//::: LIMPIA LA PANTALLA
                                          break;
                                                ///<<< INGRESA LA EDAD DE ANIMALES ///
                                          case 21:  clearBoton(op_del); //:: BORRA LOS BOTONES
                                                makeScreen(0,BLACK,WHITE);///;;;; CREA LA PANTALLA DE TECLADO NUMERICO

                                              etiquetaTeclado(2);
                                               do
                                                 {
                                                   leeTeclado(2);
                                                 } while(!(op == 9 || op == 12)); //SALE EN CASO DE ´RESIONAR 'BACK' U 'OK'
                                              if(op==12)
                                                 {
                                                    writeEeprom(2);
```

- }
- tft.fillScreen(FONDO);//::: LIMPIA LA PANTALLA
- break;
- }
- } while (op != 22);///;;; SALE DEL do while CUAN DO SE HAYA PRESIONADO EL BOTON 'ATRAZ'
- clearBoton(op_del); //:: BORRA LOS BOTONES
- break;
- case 25: //LLAMA A PANTALLA DE HORA/FECHA
- ban1=0;
- hm=0;
- ///;;;; CREA LA PANTALLA DENTRO DE HORA FECHA [+, -, -->]
- clearBoton(op_del); //:: BORRA LOS BOTONES
- makeScreen(6,BLACK,WHITE);
-
- etiquetaClock(0,WHITE);// PONE EL TITULO 'HORA'
- updateclok(); // ACTUALIZA LOS VALORES DE LAS VARIABLES hora, minuto, dia, mes, anio, A LOS ACTUALES
- clock(1,0,WHITE,5);// HORA REAL/ HORA DE AJUSTE, HORA/FECHA, COLOR DE TEXTO, TAMAÑO DE TEXTO
- do
- {

```
leeClock();
}while(!(op == 32 || ban1 == 2));
///**** SI op = 32, ES 'ATRAS', SI ban = 2  HAY QUE GUARDAR EL VALOR
if(ban1 == 2)
 {
  clock(1,1,FONDO,3); //BORRA EL RELOJ
  etiquetaClock(1,FONDO);       //BORRA LA ETIQUETA HORA/FECHA
  settime(); //<<<   <<< PONE HORA Y FECHA INGRESADA EN EL RTC
 }
else
  if(ban1 == 1)   // SI PRESIONO OK EN HORA Y ATRAS EN FECHA
   {
    clock(1,1,FONDO,3);//BORRA EL RELOJ
    etiquetaClock(1,FONDO);//BORRA          LA ETIQUETA HORA/FECHA
   }
  else          // SI PRESIONO ATRAS EN HORA
   {
    clock(1,0,FONDO,5);//BORRA EL RELOJ
    etiquetaClock(0,FONDO);//BORRA          LA ETIQUETA HORA/FECHA
```

```
                }
                clearBoton(op_del); //:: BORRA LOS BOTONES
            break;
            case 26:  //LLAMA A PANTALLA DE #CELULAR
                clearBoton(op_del);
                makeScreen(0,BLACK,WHITE);///;;;;  CREA  LA PANTALLA DE TECLADO NUMERICO
                etiquetaTeclado(3);

                do
                 {
                  leeTeclado(3);
                 } while(!(op == 9 || op == 12)); //SALE EN CASO DE ´RESIONAR 'BACK' U 'OK'
                    if(op==12)
                     {
                       //eepromClear(3);
                       writeEeprom(3);
                     }
                     // si op = 12 guardar el valor ingresado en aux en la eeprom <<<----****
                tft.fillScreen(FONDO); //::: LIMPIA LA PANTALLA
            break;
          }
```

```
                    } while (op != 27);///;;; SALE DEL do while CUANDO SE HAYA PRESIONADO EL BOTON 'ATRAS'
                    clearBoton(op_del); //:: BORRA LOS BOTONES
                  break;// FIN case 18
                } ///  FIN 1er switch DEL case 13
              }while(op!= 19); // CONDICION DEL 1er do DEL case 13

        break;// FIN case 13

        case 14:  //LLAMA A PANTALLA DE DATOS
              do
              {
                clrprintclock();
                clearBoton(op_del); //:: BORRA LOS BOTONES
                clearBoton(8); //:: BORRA EL BOTON START/STOP
                makeScreen(7,BLACK,WHITE);
                do
                  {
                    leeBoton(7);
                  } while(!(op >= 33 && op <= 35));// SALE CUANDO op = 'OK'

                if(op != 35)
                  {
                    clearBoton(op_del);
```

```
                makeScreen(4,BLACK,WHITE);
                switch(op)
                   {
                     case 33:  informativo(2); ///<<< LLAMA AL LA PANTALLA DE INFORMACIÓN
                          do //;;; MANTIENE EN LA PANTALLA ACTUAL
                           {
                             leeBoton(4);
                           } while (op != 23);///;;; SALE DEL do while CUAN DO SE HAYA PRESIONADO EL BOTON 'OK'
                     break;
                     case 34:  informativo(3);
                          do //;;; MANTIENE EN LA PANTALLA ACTUAL
                           {
                             leeBoton(4);
                           } while (op != 23);///;;; SALE DEL do while CUAN DO SE HAYA PRESIONADO EL BOTON 'OK'
                     break;
                   }
                if(op == 23)
                 tft.fillScreen(FONDO);
               }
            }while(op != 35);
      break;
```

```
        case 15:  //LLAMA A LA PANTALLA STOP
                do
                 {
                   clearBoton(op_del);
                   makeScreen(8,BLACK,WHITE);

                   digitalWrite(backlight,LOW);
                   do
                    {
                     printclock(1);
                    // a_dormir();
                     leeBoton(8);
                    }while(!(op >= 36 && op <= 38));

                   if(op != 38)
                    {
                     clrprintclock();
                     clearBoton(op_del);
                     makeScreen(4,BLACK,WHITE);
                     switch(op)
                        {
                          case 36:  informativo(4); ///<<< LLAMA AL LA PANTALLA DE HORARIO
                                do //;;; MANTIENE EN LA PANTALLA ACTUAL
```

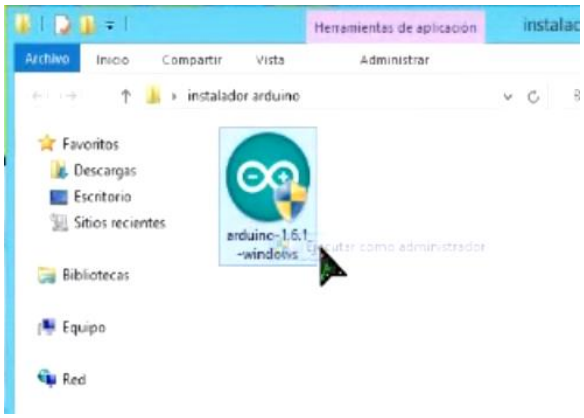
```
                                {
                                  leeBoton(4);
                                } while (op != 23);///;;; SALE DEL do while CUAN DO SE HAYA PRESIONADO EL BOTON 'OK'
                        break;
                        case 37:  informativo(5); ///<<< LLAMA AL LA PANTALLA DE RESUMEN
                                do //;;; MANTIENE EN LA PANTALLA ACTUAL
                                {
                                  leeBoton(4);
                                } while (op != 23);///;;; SALE DEL do while CUAN DO SE HAYA PRESIONADO EL BOTON 'OK'
                        break;
                      }
                    if(op == 23)
                     tft.fillScreen(FONDO);
                   }
                }while(op != 38);
          break;
          /* default:
                tft.fillScreen(0xFBC0);/////--- PONE LA PANTALLA NARANJA
                makeScreen(op);
                op=3;
                do
```

```
                    {
                      leeBoton(0);
                    } while(!(op >= 0 && op < 3));  */
            a=1;
          } //FIN DEL 1er switch
       clearBoton(op_del); //:: BORRA LOS BOTONES
    }
    //////////+++++ FIN DEL LOOP++++++/////
```

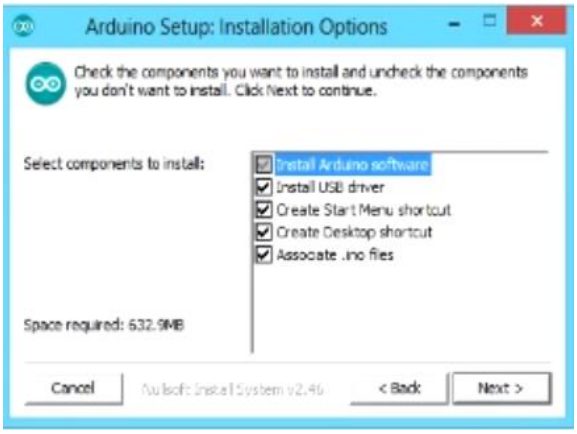
4.3. ANEXO 3: INSTALACIÓN IDE ARDUINO

- El IDE de Arduino se encuentra disponible en su página oficial o en CD ROM al adquirir los productos originales.
- Ingresar en la pagina oficial de arduino https://www.arduino.cc/en/Main/*Software#* dar clic en descargar, esperar que se descarge el archivo.

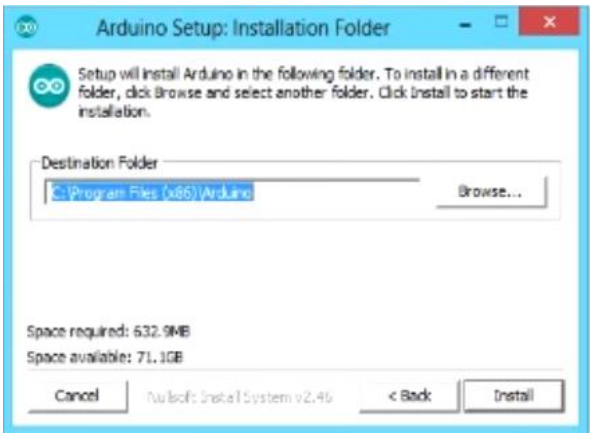
- Se abre automáticamente una ventana y dar doble clic en el icono de Arduino



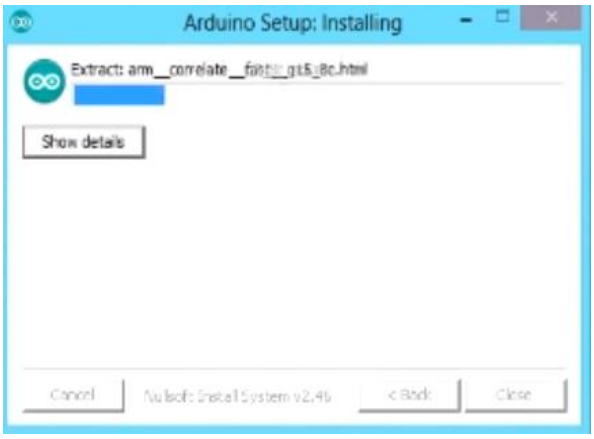
- Clic en next



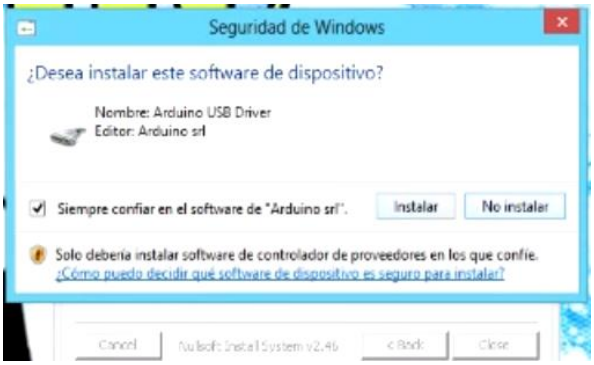
- Clic en *instal*



- Esperar mientras se instala



- Clic en Siempre confiar en el *software* de Arduino .srl
- Clic en instalar y listo



4.4. ANEXO 4: BIBLIOTECAS DE ARDUINO

- Las bibliotecas de mayor uso se encuentran dentro del IDE de Arduino que se haya instalado, cuando se necesita manejar componentes que no tienen una biblioteca estándar se descarga esta de la página oficial de Arduino o de las páginas de comunidades de Arduino presentes en la red.

4.5. ANEXO 5: DISEÑO EN SOLID WORKS

Bibliografía

AUTÓMATAS PROGRAMABLES Curso Básico de Autómatas Programables. (2001). Obtenido de http://www.sc.ehu.es/sbweb/webcentro/automatica/WebCQMH1/PAGINA%20PRINCIPAL/Automatizacion/Automatizacion.htm

Barbus, E. (Marzo de 2014). *ecda EL CAJÓN DE ARDU*. Obtenido de http://elcajondeardu.blogspot.com/2014/03/tutorial-sensor-ultrasonidos-hc-sr04.html

Contreras, J. C. (s.f.). *monografias.com*. Obtenido de http://www.monografias.com/trabajos93/tecnologia-gsm-aplicada-automatizacion-traves-micro-controladores/tecnologia-gsm-aplicada-automatizacion-traves-micro-controladores.shtml

COPADATA. (s.f.). Obtenido de https://www.copadata.com/es-es/soluciones-hmi-scada/interfaz-hombre-maquina-hmi/

Falak, O. (2010). *monografias.com*. Obtenido de http://www.monografias.com/trabajos93/motor-paso-paso/motor-paso-paso.shtml

FAO. (2016). Obtenido de http://www.fao.org/fishery/affris/perfiles-de-las-especies/nile-tilapia/formulacion-y-preparacion-produccion-de-alimentos/es/

Gardey, J. P. (2012.). *definición.DE*. Obtenido de http://definicion.de/mecanismo/

GEEKFACTORY. (s.f.). Obtenido de http://www.geekfactory.mx/tutoriales/tutoriales-arduino/ds1307-en-tinyrtc-con-arduino/

Gonzales, I. C. (2010). Obtenido de http://ri.uaq.mx/bitstream/123456789/2488/1/RI002308.pdf

Ing. Jhonatan Gallo, M. G. (2013). DISEÑO DE UN SISTEMA AVANZADO DE DOSIFICACIÓN DE CONCENTRADO PARA PECES EN CAUTIVERIO. *Revista Colombiana de Tecnologías de Avanzada*, 67-73.

monografias.com. (2010). Obtenido de http://www.monografias.com/trabajos-pdf4/tutorial-interactivo-motor-*brushless*/tutorial-interactivo-motor-*brushless*.pdf

NAYLAMP mechatronics. (2016). Obtenido de http://www.naylampmechatronics.com/arduino-*shield*s/146-s*hield*-geeetech-gsm-gprs.html

Ojeda, L. T. (2016). *ARDUINO.cl.* Obtenido de http://arduino.cl/arduino-mega-2560/

Ortega, L. (2016). Obtenido de http://cultivodetilapia.blogspot.com/

Peñarreta, J. P. (s.f.). *monnografias.com*. Obtenido de http://www.monografias.com/trabajos90/ultrasonido-frecuencia/ultrasonido-frecuencia.shtml

Publicado, J. P. (2013). *Definición.DE*. Obtenido de http://definicion.de/maquina/

Tecnología del Plástico. (2016). Obtenido de http://www.plastico.com/temas/En-que-se-diferencian-los-sistemas-de-dosificacion-volumetricos-de-los-gravimetricos+97156

The Free Dictionary. (2016). Obtenido de http://es.thefreedictionary.com/automatizaci%C3%B3n

Wikipedia . (2016). Obtenido de Panel solar: https://es.wikipedia.org/wiki/Panel_solar

Wikipedia. (2015). Obtenido de https://es.wikipedia.org/wiki/Reloj_en_tiempo_real

Wikipedia. (2016). Obtenido de Batería de ion de lito: https://es.wikipedia.org/wiki/Bater%C3%ADa_de_ion_de_litio

Wikipedia. (2016). Obtenido de LCD TFT: https://es.wikipedia.org/wiki/TFT_LCD

Wikipedia. (2016). Obtenido de https://en.wikipedia.org/wiki/Electronic_speed_control

Wikipedia. (2016). Obtenido de https://en.wikipedia.org/wiki/ULN2003A

wikipedia la enciclopedia libre. (2016). Obtenido de https://es.wikipedia.org/wiki/Fitoplancton
Wikipedia La enciclopedia libre. (julio de 2016). Obtenido de https://es.wikipedia.org/wiki/Arduino