INSTITUTO TECNOLÓGICO SUPERIOR DE CAJEME

Diseño e implementación de sistema autónomo para control y dosificación de alimento.

TESIS

PARA OBTENER EL GRADO DE

MAESTRO EN INGENIERÍA MECATRÓNICA

PRESENTA

ING. ALAN ENRIQUE HERNÁNDEZ PABLOS

DIRECTOR DEL PROYECTO

DR. JOSÉ GUADALUPE CASTRO LUGO

CD. OBREGÓN, SON. DICIEMBRE DE 2018

DEDICATORIA

A MI HIJO: Por todo lo que haz aportado en mí vida. Por ser la motivación más grande que pueda tener, por ser el regalo que me dió la vida. Por inconscientemente, con el solo escuchar tu risa, lograr que mis energías sean renovadas. Esto es por tí y para tí Enriquito.

A MIS PADRES: Por siempre haber confiado en mí. Por haberme apoyado en todos los momentos de mi vida. Por siempre tener esas palabras de aliento y ese sabio consejo. Por ser más que un ejemplo a seguir. Los amo. Gracias Enrique Hernández L. y Karla L. Pablos R.

AGRADECIMIENTOS

A DIOS: Por haberme dado la vida, por darme fuerza para levantarme todos los días, y por permitirme lograr todas mis metas.

MI FAMILIA: Por siempre haber estado para mí, por la unión y el cariño que siempre me han demostrado.

MI ASESOR DE TESIS: El Dr. José G. Castro L., por su preocupada y atenta dirección en el desarrollo de este trabajo de Tésis, por compartir sus conocimientos, por su gran vocación por la orientación y la enseñanza y por haber depositado su confianza en mí.

MIS MAESTROS DE POSGRADO: Por los conocimientos y sabiduría compartida, por su pasión y compromiso con el desarrollo académico de sus estudiantes.

AL CONACYT: Por el apoyo concedido, gracial al cual fue posible desarrollar este trabajo.

AL ITESCA: Por tener siempre las puertas abiertas a todas aquellas personas con sed de conocimientos y superación. Siempre serás mi casa.

A MIS AMIGOS: Por su entrañable cariño y desinteresado apoyo, con quienes he compartido momentos muy importantes de mi vida.

RESUMEN

El presente trabajo muestra una propuesta para solucionar un problema que radica en una actividad económica muy importante de nuestra localidad, la cual es el cultivo de especies marinas en estanques artificiales. Entre las diversas especies que se cultivan, el camarón representa una gran parte de la producción local.

Entre los diversos factores que afectan y acrecentan los costos de estas prácticas, se encuentra la ineficiente dosificación de alimentos en los estanques, generadas ya sea por malas técnicas de dosificación o por falta de equipos que garanticen una dosificasión precisa.

Por dicho motivo fue necesario el plantear una solución a este problema, el cual se desarrolla en este trabajo de tésis. Se propone el diseño e implementación de un sistema de dosificación de alimentos, mismo que tendrá la ventaja de estar montado sobre el chasis de una camioneta para poder dosificar varios estanques. Este debe garantizar que se dispensen cantidades precisas de alimento.

Para poder llevar acabo este proyecto, fue necesario desarrollarlo desde su etapa inicial. Se realizaron distintas tareas y fue necesario integrar distintos Softwares para lograr que se cumpliera con los objetivos propuestos.

Se plantearon dos soluciones funcionales para este sistema, uno de ellos consta de la captura manual de los datos, mientras que otro, un poco más complejo, basa la toma de desiciones, correspondiente a la cantidad de alimento a dosificar, en la inteligencia artificial, haciendo uso del método de las redes neuronales artificiales.

Se utilizó el Software Solidworks para realizar el diseño de distintos elementos mecánicos, incluyendo estructuras, equipos de almacenamiento, transmisiones mecánicas y equipos de dosificación, entre otros. Se utilizó el Software de Arduino para realizar la programación estructurada, la cual fue creada para controlar los actuadores, las tarjetas de apoyo y el

funcionamiento general del sistema. Se utilizó el Software Ionic Framework para crear una aplicación móvil, el cual será el encargado de enviar datos, por medio de un dispositivo móvil, hasta el microcontrolador. Se utilizó el software Rstudio para realizar un programa que simulara el entrenamiento de una red neuronal y a su vez, se utilizó MatLab 2015 para realizar simulaciones de la red creada.

El propósito de este proyecto es la integración final de distintas tarjetas de apoyo diseñadas para Arduino, para un mejor desarrollo y control del alimento a dosificarse.

ABSTRACT

The present work shows a proposal to fix a problem which lies in a economic activity very important in our locality, which is the marine species in outdoor culturing ponds. Between these species, the shrimp represent the biggest part of the local production.

Between the divers factors which affect and increase the costs of these practics, the inneficient dosage of feed in the artificial pond is the principal. This inneficient is generated by the inappropiate tecniques of dosage or because of the lock of equipment which can to guarantee a precise dosage.

For that reason it was necessary to raise a solution to this problems, which is development in this tesis work. Its propose the design and implementation of a feeder system, which will be mounted on the chassis of a truck, this so it can be uses to feed several ponds. This system must to guarantee a precise dosage of shrimp's feed.

It was necesary to develop this proyect since initial stage. It was necesary to integrate several softwares to curry the proposal objetives. It was posed to funtional solucions for this system, the first consisted of capturing data manually, while the other, based his operation on the artificial inteligence method, using the artificial neuronal networks to control the decision making about the amount of food to dosage.

It was necesary to use software Solidworks to design all the mechanical elements, including the structures, storage elements, mechanical transmitions and dosage equipments. It was used software Arduino to perform the structurated programation, which controled the actuators, support plataforms and the general performance of the system. It was used software Ionic Framework to develop a mobile aplication, which will to send data from a mobile device to the microcontroller. It was used software Rstudio to develop a support program which could simulete a neuronal training and it was used MatLab 2015 software to permorm several simulations of

created network.

The purpose of this project is the integration of Arduino's software to development and design of a shrimp feeder system.

ÍNDICE

1.. Introducción 13
1.1. Antecedentes 13
1.2. Definición del Problema 19
1.3. Justificación 19
1.4. Objetivos 20
1.4.1. Objetivo General 20
1.4.2. Objetivos Específicos 20
1.5. Alcances y Limitaciones 21
1.5.1. Alcances 21
1.5.2. Limitaciones 21

2.. Marco Teórico 22
2.1. Dispositivos específicos utilizados en el Sistema 22
2.1.1. Microcontrolador 22
2.1.2. Sensores 25
2.1.3. Actuadores 26
2.2. Elementos Mecánicos 29
2.2.1. Catarina 29
2.2.2. Cadena Industrial 30
2.3. Herramientas de Programación 32
2.3.1. SolidWorks 32
2.3.2. Ionic Framework V2.0 32
2.3.3. RStudio 33

2.3.4. MatLab R2015 34
2.4. Inteligencia Artificial 35
2.4.1. Redes Neuronales 37

3.. Desarrollo 44
3.1. Diseño Mecánico 45
3.1.1. Selección de Elementos Mecánicos y de Almacenamiento 45
3.1.2. Acoplamiento Mecánico 51
3.2. Programación del Microcontrolador 52
3.2.1. Captura de Datos Manual 52
3.2.2. Control Automático 55

4.. Conclusiones y Recomendaciones 66

Referencias bibliográficas 69
Referencias 69

ÍNDICE DE FIGURAS

1.1. Gráfica de Producción Global derivada de la pesca y la acuicultora. 14
1.2. Producción de pesca en granjas acuícolas en México por Estado en año 2014. . . 16
1.3. Diseño de ASAF1 (Auto Switch Aqua Feeder 1). 17
1.4. Aparato soplador para alimentación. 18
1.5. Método de alimentación utilizado en campos acuícolas de la región. 19

2.1. Placa Arduino Mega 23
2.2. Módulo HC-05 Bluetooth Para Arduino. 23
2.3. Sensor de Proximidad Inductivo 25
2.4. Motor a Pasos 26
2.5. Especificaciones de los motores a Pasos 27
2.6. Motor 12 VCD 28
2.7. Motor Blower a Gasolina 28
2.8. Catarina 30
2.9. Cadena 30
2.10. Transmisión de Cadena 31
2.11. Solidworks. 32
2.12. IONIC FrameWork V2.0 33
2.13. Software RStudio. 34
2.14. MatLab R2015. 35
2.15. Técnicas básicas utilizadas en Inteligencia Artificial 37
2.16. Modelo de Neurona Artificial descrito por Rumelhart y McClelland 38
2.17. Modelo de Neurona Artificial descrito por Rumelhart y McClelland 40

2.18. Función sigmoidal 41
2.19. Salida de Neurona 42

3.1. Funcionamiento General del Sistema. 44
3.2. Matriz Morfológica 46
3.3. Diseño de Tolva Contenedora 47
3.4. Diseño de VDR 48
3.5. Diseño de Motor 12 VCD 49
3.6. Diseño de Blower con Motor a Gasolina 50
3.7. Selector de posición 51
3.8. Diseño Final de Sistema alimentador 52
3.9. Diagrama Funcional en Sistema de Captura Manual 53
3.10. Diagrama funcional del sistema automático 56
3.11. Estructura de Red Neuronal 57
3.12. Resultados de entrenamiento Neuronal 59
3.13. Pesos sinápticos obtenidos 60
3.14. Conexión Bluetooth mediante Aplicación Móvil 64
3.15. Dosificación mediante Aplicación Móvil 65

ÍNDICE DE TABLAS

3.1. Jerarquización de Requerimientos de Sistema alimentador 45
3.2. Especificaciones de Motor Wiper 49
3.3. Especificaciones del Blower 50
3.4. Datos De red Neuronal 58
3.5. Comprobación de Datos 63

1. INTRODUCCIÓN

En este trabajo de tésis se presenta el desarrollo de un sistema de dosificación de alimento para camarón, utilizado en cultivos acuícolas de estanques al exterior. Se muestra el diseño, programación, pruebas e implementación del sistema mencionado.

El carácter profesional de este proyecto de tesis pertenece a la especialidad del diseño mecatrónico, ya que se aplican e integran las distintas áreas pertenecientes, las cuales son: mecánica, electrónica y sistemas computacionales. Durante el desarrollo del proyecto, se realizó el diseño completo tanto de la máquina, como de la tolva contenedora y el acoplamiento entre ellos, es aquí donde se utilizan los conocimientos de diseño mecánico. Es también utilizada la electrónica debido a que es necesario controlar distintos actuadores, entre ellos un motor de corriente directa y, realizar la adquisición e interpretación de los datos obtenidos por los sensores.

También se utilizan los conocimientos de programación y sistemas computacionales para el diseño de distintos programas de control gracias a los cuales es posible realizar las funciones previamente definidas, además, estos conocimientos fueron utilizados para el desarrollo de una aplicación móvil para el control general del sistema. Y por último, se utilizan los conocimientos adquiridos para el diseño, uso e interpretación correcta de datos partiendo del entrenamiento de redes neuronales, método en el cual se basa el control inteligente creado para la resolución de este problema.

1.1 Antecedentes

La acuacultura abarca sobre todo el control del crecimiento y producción de las especies susceptibles al cultivo o crianza en el medio acuático. Es una actividad orientada a la selección y manejo de organismos reproductores, producción de huevos, de larvas, de crías y de engorda.

Pasando por el transporte, procesamiento y comercialización del producto hasta su consumo; siendo por tanto una actividad interdisciplinaria, orientada a la creación de unidades de producción (Manzo, 2000).

La acuacultura ha sido el sector de producción alimentaria de más rápido crecimiento del mundo, con una tasa de crecimiento anual del 8 %, abasteciendo el 49 % de la demanda global de alimentos marinos. En la figura 1.1 se muestra una gráfica de comparación entre el suministro de alimentos marinos provenientes del mar y los resultantes de la acuacultura. El suministro total brindado por la acuacultura ha incrementado dramáticamente desde 0,7 mMt que existía en 1950 hasta 90 mMt en el 2012. El crecimiento del suministro global de alimentos marinos ha hecho posible el implementar nuevas tecnologías para poder lograr un mayor control sobre los procesos de producción acuícolas (Kumar y Engle, 2016).

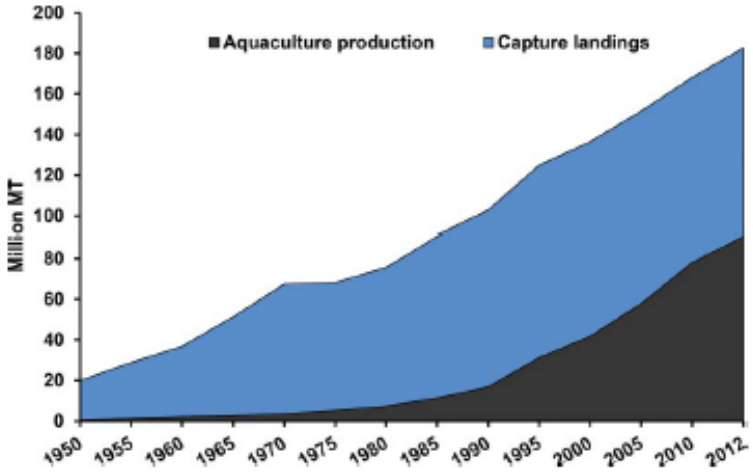


Fig. 1.1: Gráfica de Producción Global derivada de la pesca y la acuicultora. (FAO, 2014).

En los sistemas acuícolas, los gastos de alimentación representan entre el 40 % y el 60 % de los gastos de producción (Hasan, Hecht, De Silva, Tacon, y cols., 2007). Para poder asegurar la rentabilidad de los sistemas, es indispensable que los acuacultores tengan acceso a alimentos de buena calidad y que estos puedan ser conseguidos a precios razonables, además, es importante que ellos optimicen este proceso instituyento los métodos y prácticas de alimentación apropiadas para granjas de cultivo acuícolas (Hasan y New, 2013).

El camarón es uno de los alimentos de origen marino más buscado por las personas en todo

el mundo, y su cultivo ha comenzado ha tener mucha importancia en el comercio internacional (Mathiesen, 2015). El cultivo de camarón es una de las actividades acuícolas mas importantes y la alimentación manual es el método que ha sido practicado desde hace mucho tiempo. En la mayoría de los cultivos de camarón, los costos de alimentación representan más de la mitad total de los gastos y, si no se cuenta con un sistema eficiente de alimentación, el alimento no es consumido en su totalidad por el camarón, por lo tanto, esto desemboca en el deterioro del agua y de la calidad del suelo (Newport y Jawahar, 1995).

En México, la acuacultura nace como una actividad complementaria de apoyo social a las comunidades rurales, con lo cual se pretendía incrementar el consumo de proteína animal y mejorar así los niveles nutricionales de la población (Juárez-Palacios, 1987). Aunque esta actividad se ha diversificado más hacia la crianza de peces de agua dulce en los campos acuícolas, también se lleva a cabo en especies marinas, ya que la piscicultura marina en México como en otros países, es una alternativa tecnológicamente viable ante la creciente demanda de alimentos de origen proteico para el consumo generalizado de la población humana. En México la piscicultura Marina se inicia a finales de la década de los 80, cuando se realizan los estudios para la engorda del pámpano (Trachinotus paitiensis) en jaulas flotantes en Baja California Sur. A partir de 1989 la piscicultura marina empezó a desarrollarse de manera experimental, iniciándose básicamente con los estudios biológico-reproductivos de especies de alto valor comercial como la cabrilla, pargo, róbalo, huachinango, corvina, pámpano, totoaba y lenguado (Avilés, 2000).

Sonora ha tenido uno de los crecimientos más rápidos y espectaculares del cultivo de camarón en el mundo al pasar de una producción de $6,690$ toneladas en el año 2000 a más de $68,000$ toneladas en el año 2007, lo que representa un incremento del $900\,\%$, con un crecimiento promedio de $7,000$ toneladas anuales, superando a Sinaloa. Durante este periodo de expansión, solamente en el año 2001 la producción no creció significativamente al pasar de $23,280$ toneladas a $23,750$ en el 2002, debido a los primeros brotes de "mancha blanca" en el estado. Sin embargo, ya para el 2003, la producción había aumentado nuevamente $13,250$ toneladas dando un total de $37,000$ toneladas, marcando una nueva etapa de crecimiento sostenido. Se espera que a finales del 2008, la producción camaronícola supere las 70 mil toneladas, la cifra más alta de un estado

en la historia de México (SAGARPA, 2013).

Desde el 2003 la producción de camarón en Sonora ha crecido a un ritmo promedio de 8,7 % anual, hasta llegar a 68, 510 toneladas en el 2007. Si bien el consumo nacional ha crecido, es difícil suponer que el ritmo de este crecimiento se sostendrá en los próximos años, pues en el contexto mundial, la producción de camarón también se ha incrementado en este orden porcentual anual desde hace varios años, y los canales tradicionales de comercialización han resentido estos aumentos, lo cual ha repercutido en los precios finales al productor. Como se muestra en la figura 1.2, el estado de Sonora es el estado con mayor producción de pesca y extracción de grangas acuícolas.

Producción pesquera Nacional
Año 2014
Toneladas

Estado	Peso vivo (toneladas)	Peso desembarcado (toneladas)	Precio (pesos por kilogramo)	Valor de la producción (miles de pesos)
Total	**1,751,952.56**	**1,634,252.06**		**24,109,810**
Sonora	507,736.31	457,990.32	8.47	3,879,946
Sinaloa	339,227.43	318,240.57	18.3	5,824,508
Baja California Sur	188,835.63	172,415.03	10.28	1,772,226
Baja California	159,670.40	141,599.39	11.39	1,612,452
Veracruz	78,904.38	78,206.46	14.32	1,119,897
Chiapas	58,945.18	59,267.77	14.87	881,582
Campeche	51,665.83	47,369.61	22.82	1,080,775
Jalisco	47,324.80	47,287.62	17.25	815,511
Tabasco	46,169.25	46,051.14	12.68	583,954
Yucatán	42,143.82	38,707.62	31.81	1,231,410
Nayarit	38,581.94	37,561.91	27.38	1,028,356
Tamaulipas	37,120.59	36,337.85	27.25	990,310
Colima	37,081.33	36,877.13	13.59	501,184
Michoacán	36,811.27	36,484.18	23.78	867,711
Oaxaca	17,175.26	16,435.01	22.14	363,805
México	15,728.04	15,727.95	24.98	392,850
Guerrero	14,855.31	14,581.36	23.72	345,911
Hidalgo	8,576.56	8,540.79	16.25	138,795
San Luis Potosí	5,339.56	5,261.88	18.31	96,333
Quintana Roo	4,419.33	3,791.02	42.35	160,537
Puebla	4,014.34	4,013.60	42.17	169,273
Guanajuato	3,258.89	3,255.44	10.01	32,591
Zacatecas	2,267.83	2,266.12	21.54	48,815
Coahuila	1,867.49	1,815.21	9.78	17,748
Morelos	1,185.49	1,185.49	68.88	81,654
Chihuahua	1,008.26	957.65	29.89	28,620
Durango	734.31	733.48	24.05	17,639
Querétaro	583.50	582.76	21.76	12,682
Tlaxcala	405.92	405.92	17.14	6,959
Aguascalientes	198.41	190.64	20.31	3,872
Nuevo León	115.92	111.12	17.13	1,904

Fig. 1.2: Producción de pesca en granjas acuícolas en México por Estado en año 2014.

Por lo mencionado anteriormente, es que la construcción de un sistema dispensador automatizado cobra factibilidad. Para poder realizarlo, es necesario tener distintos conocimientos de mecatrónica, los cuales conjugan distintas partes de la ingeniería para poder realizar distintas funciones.

La mecatrónica es una etapa natural en el proceso evolutivo del diseño en la Ingeniería moderna. El desarrollo de las computadoras, microcomputadoras, sistemas embebidos, y la información asociada a la tecnología y los softwares avanzados, hicieron a la mecatrónica indispensable en la última parte del siglo XX (Bishop y cols., 2002). Un sistema mecatrónico

no es solo una combinación de un sistema eléctrico y un sistema mecánico y es más que solo un sistema de control. Un sistema mecatrónico es una completa integración de todos ellos (William, 2001).

Actualmente, existen distintos prototipos de alimentadores los cuales han sido diseñados para cumplir con funciones específicas dentro de los distintos campos acuícolas que lo han solicitado. Actualmente, uno de los modelos que ha tenido más auge, es el ASAF1(Auto Switch Aqua Feeder 1) (Appana, Alam, y Basnet, 2016). Dicho modelo es un sistema que cuenta con un controlador de tiempos, el cual tiene dos estados (On-Off), mismo que controla los ciclos de encendido y apagado de los motores que dispensan el alimento dentro del estanque.

Este prototipo se fija dentro del estanque con ayuda de 4 soportes metálicos. Dependiendo del periodo en el que se encuentre el camarón, el usuario puede realizar cambios con ayuda de un teclado, mismo que es utilizado para mandar la señal al PIC utilizado (microcontrolador). El PIC utilizado es el dsPIC5011, microcontrolador de 16 bits. Este microncontrolador cuenta con una memoria flash y con una memoria interna de $4\,KB$, el cual opera entre los 2,5 Volts y 5,5 Volts. Este PIC brinda la opción de estarse reprogramando para poder ser utilizado a lo largo de todo el ciclo de crianza del camarón. En la figura 1.3 se puede observar el diseño que presenta el sistema mencionado, así como el diseño que presentan los motores encargados de dispensar el alimento.

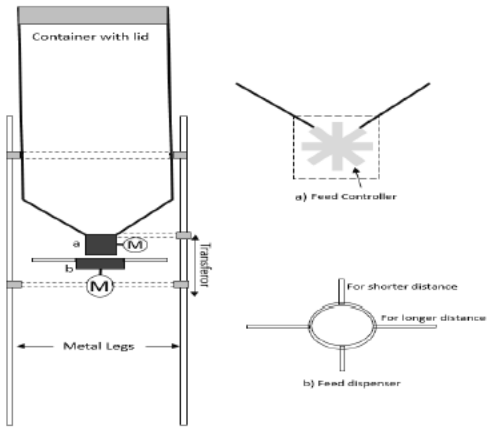


Fig. 1.3: Diseño de ASAF1 (Auto Switch Aqua Feeder 1).

Otro sistema que es utilizado en la actualidad es el aparato soplador de alimento (Aparato Soplador para alimentación) (Hemeda, MM, y Abdel-Mottaleb, s.f.), el cual se muestra en la

figura 1.4. Este dispositivo es un sistema de aire, el cual, por medio de un compartimiento se le agrega el alimento que se debe dispensar y, al momento de encender el rotor del soplador, este arroja el alimento dentro de los estanques donde se encuentran los especímenes a alimentar.

Además de los sistemas anteriores, existen distintas patentes y prototipos que se han diseñado durante los últimos años. Entre los principales prototipos, se puede encontrar los siguientes: Food Dispensing System, perteneciente a la patente $4,359,014$ (Molaug, Kluge, y Havrevold, 1982), Offshore Aquaculture System, perteneciente a la patente $US\ 7,992,522\ B2$ (Harrison, Dobson, y Hoar, 2011) y System and Method for controlling feeding of farmed fish, perteneciente a la patente $US\ 2013/0206078\ A1$ (Melberg y Torgersen, 2011).

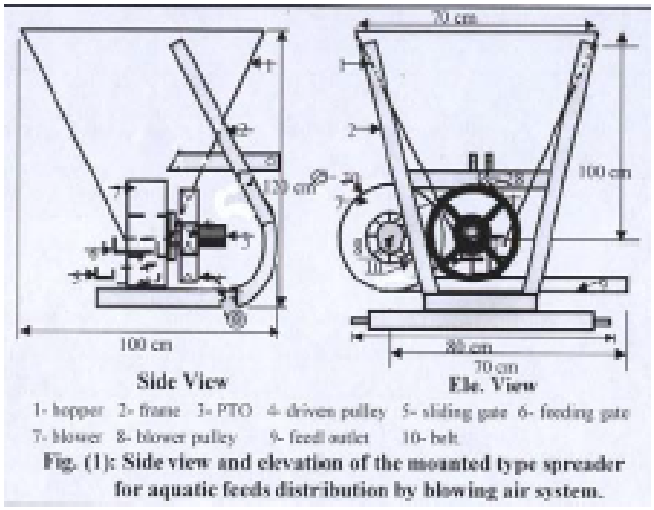


Fig. 1.4: Aparato soplador para alimentación.

Los sistemas presentados anteriormente muestran distintos inconvenientes, por lo cual la factibilidad del proyecto a desarrollar cobra importancia. En el caso del ASAF1, el principal inconveniente es que, si el estanque que se desea alimentar es de una dimensión grande, es necesario colocar más de un sistema dentro del estanque, y si se tienen muchos estanques, significa tener más de un dispositivo, por lo cual el costo se aumenta. Mientras que, el sistema del FBA (Feed Blowing Apparatus) tiene el inconveniente que no es un sistema completamente automático, debido a que es necesario que un operador esté introduciendo el alimento a dispensarse cada vez que se encienda el blower, por lo cual, se puede decir que es solamente un apoyo para lograr esparcir completamente el alimento en el estanque, pero no es un dispensador automático.

Por lo anterior, y considerando todos los factores anteriores, es que cobra relevancia el proyecto del dispensador de alimento montado en la estructura de un camión, el cuál será controlado de manera automática y puede alimentar un sinfin de estanques en un mismo recorrido. Con lo anteriormente mencionado, es posible corregir los problemas descritos anteriormente.

1.2 Definición del Problema

El problema que se tiene dentro de los campos acuícolas de la localidad es que no cuentan con métodos ni técnicas correctas de alimentación. En la figura 1.5 se muestra el método más utilizado para dispensar alimentos en los distintos campos acuícolas de la región.



Fig. 1.5: Método de alimentación utilizado en campos acuícolas de la región.

Como se puede observar en la imagen, el método actual de alimentación es una técnica que no es confiable, debido a que no es posible cuantificar la cantidad de pellet que se dispensa a cada estanque, además, el hecho de no conocer la cantidad exacta de alimento que se brinda produce un alto costo para las distintas empresas, tanto en el costo generado por la sobrealimentación como en el mantenimiento que se debe brindar al estanque.

1.3 Justificación

La maestría en Mecatrónica es de carácter profesionalizante, es por eso que el proyecto a realizar cobra su importancia, debido a que puede ser aplicado en infinidad de empresas locales,

las cuales podrán utilizar el proyecto a desarrollar para cumplir con todas sus funciones dentro de sus procesos y de esta manera reducir los problemas presentes actualmentes relacionados con la mala alimentación en los estanques.

El proyecto beneficiará a la Universidad debido a que, al momento de realizar este tipo de proyectos tan necesarios dentro de nuestra región, tanto la universidad como los egresados de esta misma serán solicitados para poder realizar y colaborar con las distintas empresas de este mismo giro, lo anterior para poder realizar proyectos parecidos y con funciones muy similares.

Si esta problemática no se resuelve, el proceso seguirá presentando errores dentro del proceso de alimentación de los estanques, lo que seguirá generando pérdidas económicas importantes para dichas empresas.

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar un sistema para la dosificación de alimentos, el cual pueda funcionar de manera manual y automática.

1.4.2 Objetivos Específicos

a) Diseñar una Tolva de 1500 Kilogramos, basados en la medida del chasís de la camioneta, para poder montarse sobre esta misma.

b) Diseñar una Válvula Dosificadora Rotatita (VDR) para poder cuantificar la cantidad de alimento que debe dispensarse.

c) Seleccionar el motor encargado de hacer girar la VDR y la transmisión mecánica necesaria para aumentar la potencia de este mismo.

d) Instalar un motor-soplador, el cual será el encargado de arrojar el alimento hacia los estanques.

e) Crear una base de datos la cual se descargará tanto al celular como al microcontrolador para el control general del sistema.

f) Diseñar y desarrollar un sistema inteligente para la dosificación de alimento utilizando redes neuronales.

1.5 Alcances y Limitaciones

1.5.1 Alcances

- El proyecto debe realizarse entre el mes de Agosto del 2017 a Diciembre del 2018.
- El proyecto podrá ser controlado de manera manual y de manera automática, apoyándose en dispositivos móviles.

1.5.2 Limitaciones

- El proyecto solo será un prototipo simulado, debido a los altos costos del equipo.

2. MARCO TEÓRICO

En este capítulo se muestran los principales conceptos, definiciones y herramientas utilizadas para el logro de este trabajo. Entre ellos se incluyen, softwares, hardwares, diseños, elementos, etc. Por último, se presentan todos aquellos recursos que fueron necesarios para poder lograr que este proyecto sea funcional.

2.1 Dispositivos específicos utilizados en el Sistema

2.1.1 Microcontrolador

Arduino es una plataforma física computacional open-hardware basada en una sencilla placa con entradas y salidas (E/S), analógicas y digitales, y en un entorno de desarrollo que implementa el lenguaje Processing/Wiring. Arduino puede utilizarse en el desarrollo de objetos interactivos autónomos o puede conectarse a un PC a través del puerto serie utilizando lenguajes como Flash, Processing, MaxMSP, etc. Las posibilidades de realizar desarrollos basados en Arduino tienen como límite la imaginación.

El Arduino Mega está basado en el microcontrolador $ATMega2560$. Tiene 54 pines de entradas/salidas digitales (14 de las cuales pueden ser utilizadas como salidas PWM), 16 entradas analogas, 4 UARTs (puertos serial por hardware), cristal oscilador de 16 Mhz, conexión USB, jack de alimentación, conector ICSP y botón de reset. El Arduino Mega ahora utiliza un microcontrolador ATMega8U2 en vez l chip FTDI. Esto permite mayores velocidades de transmisión por su puerto USB y no requiere drivers para Linux o MAC (archivo inf es necesario para Windows) además ahora cuenta con la capacidad de ser reconocido por el PC como un teclado, mouse, joystick, entre otros (Mega, 2011).

En la figura 2.1 se muestra una imagen de la placa de Arduino Mega.

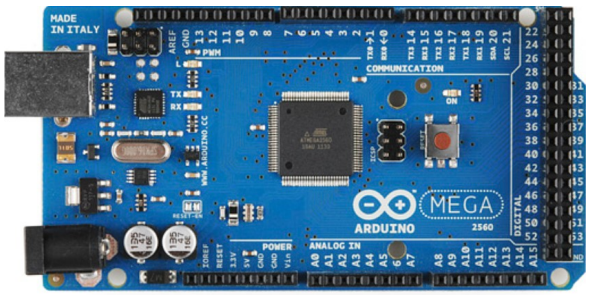


Fig. 2.1: Placa Arduino Mega

Módulo Bluetooth HC-05

El módulo puede ser configurado como maestro o como esclavo. Cuando es configurado como esclavo se comporta similar a un $HC-06$, espera que un dispositivo bluetooth maestro se conecte a este, generalmente se utiliza cuando se necesita comunicarse con una PC o Celular, pues estos se comportan como dispositivos maestros. La figura 2.2 muestra el Módulo HC-05 Bluetooth de Arduino.

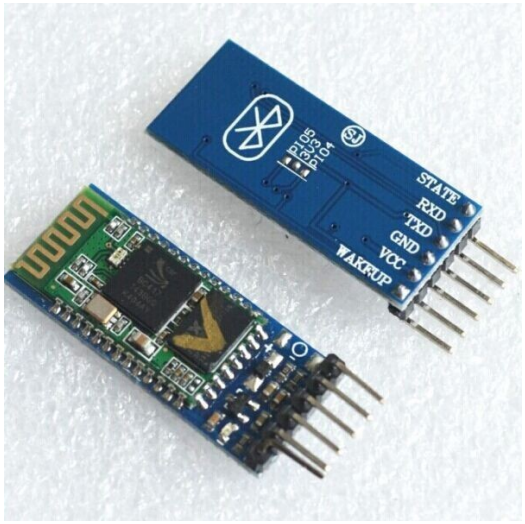


Fig. 2.2: Módulo HC-05 Bluetooth Para Arduino.

Cuando se configura como maestro, el $HC-05$ es el que inicia la conexión. Un dispositivo maestro solo se puede conectar con un dispositivo esclavo. Generalmente se utiliza este modo para comunicarse entre módulos bluetooth, pero es necesario antes especificar con que dispositivo se tiene que comunicar.

EL módulo $HC-05$ tiene 4 estados los cuales es importante conocer:

- Estado Desconectado:

- Entra a este estado tan pronto alimentas el modulo, y cuando no se ha establecido una conexión bluetooth con ningún otro dispositivo.

- El LED del módulo en este estado parpadea rápidamente.

- En este estado a diferencia del $HC-06$, el $HC-05$ no puede interpretar los comandos AT.

- Estado Conectado o de comunicación:

- Entra a este estado cuando se establece una conexión con otro dispositivo bluetooth.

- El LED hace un doble parpadeo.

- Todos los datos que se ingresen al $HC-05$ por el Pin RX se trasmiten por bluetooth al dispositivo conectado, y los datos recibidos se devuelven por el pin TX. La comunicación es transparente.

- Modo AT 1:

- Para entrar a este estado después de conectar y alimentar el módulo es necesario presionar el botón del $HC-05$.

- En este estado, podemos enviar comandos AT, pero a la misma velocidad con el que está configurado.

- El LED del módulo en este estado parpadea rápidamente igual que en el estado desconectado.

- Modo AT 2:

- Para entrar a este estado es necesario tener presionado el botón al momento de alimentar el módulo, es decir el módulo debe encender con el botón presionado, después de haber encendido se puede soltar y permanecerá en este estado.

- En este estado, para enviar comandos AT es necesario hacerlo a la velocidad de 38400 baudios, esto es muy útil cuando nos olvidamos de la velocidad con la que hemos dejado configurado nuestro módulo.

- EL LED del módulo en este estado parpadea lentamente.

2.1.2 Sensores

Sensor de Proximidad Inductivo

Los sensores inductivos son una clase especial de sensores que sirve para detectar materiales ferrosos. Son de gran utilización en la industria, tanto para aplicaciones de posicionamiento como para detectar la presencia o ausencia de objetos metálicos en un determinado contexto: detección de paso, de atasco, de codificación y de conteo. En la figura 2.3 se observa una de las presentaciones más comúnes de los sensores inductivos.



Fig. 2.3: Sensor de Proximidad Inductivo

Un sensor de proximidad inductivo puede detectar objetos metálicos que se acercan al sensor, sin tener contacto físico con los mismos. Los sensores de proximidad inductivos se clasifican más o menos en los siguientes tres tipos, de acuerdo con su principio de funcionamiento: el tipo de oscilación de alta frecuencia que utiliza la inducción electromagnética; el tipo magnético que emplea un imán; y el tipo de capacitancia que aprovecha los cambios en la capacidad eléctrica.

Un campo magnético de alta frecuencia es generado por la bobina L en el circuito de oscilación. Cuando un objeto se acerca al campo magnético, fluye una corriente de inducción (corriente de Foucault) en el objeto, debido a la inducción electromagnética. Conforme el objeto se acerca al sensor, aumenta el flujo de corriente de inducción, lo cual provoca que la carga en el circuito de oscilación crezca. Entonces, la oscilación se atenúa o decrece. El sensor detecta

este cambio en el estado de oscilación mediante el circuito de detección de amplitud, y emite una señal de detección.

2.1.3 Actuadores

Motor a Pasos

El motor paso a paso conocido también como motor de pasos es un dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, lo que significa que es capaz de girar una cantidad de grados (paso o medio paso) dependiendo de sus entradas de control. El motor paso a paso se comporta de la misma manera que un conversor digital-analógico (D/A) y puede ser gobernado por impulsos procedentes de sistemas digitales. La figura 2.4 muestra la estructura de un motor a pasos.



Fig. 2.4: Motor a Pasos

Este motor presenta las ventajas de tener precisión y repetitividad en cuanto al posicionamiento. Entre sus principales aplicaciones destacan los robots, drones, radiocontrol, impresoras digitales, automatización, fotocomponedoras, preprensa, etc.

Existen distintos motores a pasos, en los cuales varían desde su torque hasta las dimensiones de cada motor. Las especificaciones se muestran en la siguiente figura 2.5:

Especificaciones del motor paso a paso *SureStep*™				
Números de artículo	**STP-MTR-17048**	**STP-MTR-23055**	**STP-MTR-23079**	**STP-MTR-34066**
Flanje del Motor	NEMA 17	NEMA 23	NEMA 23	NEMA 34
Torque de eje detenido máximo	0,59 N•m	1,29 N•m	2,08 N•m	3,06 N•m
	83 onzas•pulgada	166 onzas•pulgada	276 onzas•pulgada	434 onzas•pulgada
	5,2 libras•pulgada	11,4 libras•pulgada	18,4 libras•pulgada	27,1 libras•pulgada
Inertia del rotor	0,0000068 kg•m²	0,000027 kg•m²	0,000047 kg•m²	0,00014 kg•m²
	0,45 oz•in²	1,483 oz•in²	2,596 oz•in²	7,66 oz•in²
	0,00006 lb•in•s²	0,00024 lb•in•s²	0,00042 lb•in•s²	0,0012 lb•in•s²
Corriente nominal	2,0 A/fase	2,8 A/fase	2,8 A/fase	2,8 A/fase
Angulo del paso básico	1,8° (motores bifásicos con cable con conector)			
Peso	210 gramos 0,7 lbs	457 gramos 1,50 lbs	1 Kg 2,2 lbs	1,17 Kg 3,85 lbs
Tolerancia del eje	0,0508 mm o 0,002 pulgadas			
Juego radial del eje @ fuerza de 1 libra	0,0254 mm o 0,001 pulgadas máximas			
Perpendicularidad	0,0762 mm o 0,003 pulgadas			
Concentricidad	0,0508 mm o 0,002 pulgadas			
Temperatura de operación	-20 °C a 50 °C (La temperature externa del motor debe mantenerse abajo de 100 °C (212 °F))			
Fuerza máxima radial	1,82 Kg o 6 lbs	4,57 Kg o 15 lbs	4,57 Kg o 15 lbs	11,89 Kg o 39 lbs
Fuerza máxima axial	1,82 Kg o 6 lbs	3,96 Kg o 13 lbs	3,96 Kg o 13 lbs	7,62 Kg o 25 lbs
Clase de aislación	Clase B 130 °C			
Aprobación de Agencia	CE (cumple con EN55014-1 (1993) y EN60034-1.5.11)			
Cable de extensión (motor al accionamiento)	6 metros o 20 pies - Número de artículo **STP-EXT-020**			

Fig. 2.5: Especificaciones de los motores a Pasos

Motor de Corriente Directa $12\,V$

El motor de corriente continua (denominado también motor de corriente directa, motor CC o motor DC por las iniciales en inglés direct current) es una máquina que convierte energía eléctrica en mecánica, provocando un movimiento rotatorio, gracias a la acción de un campo magnético.

Un motor de corriente continua se compone principalmente de dos partes. El estátor da soporte mecánico al aparato y contiene los polos de la máquina, que pueden ser o bien devanado de hilo de cobre sobre un núcleo de hierro, o imanes permanentes. El rotor es generalmente de forma cilíndrica, también devanado y con núcleo, alimentado con corriente directa a través de delgas, que están en contacto alternante con escobillas fijas (también llamadas carbones).

En la figura 2.6 se puede observar una de las múltiples configuraciones en las que se puede presentar un motor de 12 VCD. El principal inconveniente de estas máquinas es el mantenimiento, muy costoso y laborioso, debido principalmente al desgaste que sufren las escobillas o carbones al entrar en contacto con las delgas. Es posible controlar la velocidad

y el par de estos motores utilizando técnicas de control de motores de corriente continua.



Fig. 2.6: Motor 12 VCD

Motor Soplador a Gasolina

Un soplador es cualquier máquina capaz de desplazar un gas o vapor desde una zona hacia otra, forzándolo a circular a través de un venturi (o conducto de menor dimensión). Las diferentes definiciones de la palabra soplador suelen incluir características como: aspas móviles, compresiones entre 1,1 y 3 con flujos desde 500 hasta 10000 m^3/h. Estos sistemas pueden funcionar de manera eléctrica, ya sea con 120 VAC, o 12 VDC, o mediante acoplamientos con motores a combustión interna. En la figura 2.7 se muestra un soplador a gasolina.



Fig. 2.7: Motor Blower a Gasolina

Los sopladores tienen aplicaciones tanto industriales como comerciales y residenciales, suelen utilizarse en sistemas de refrigeración, enfriamiento de motores eléctricos, en máquinas de soplado de plásticos, circulación de gases hacia calderas y máquinas de vapor, tienen aplicaciones en sistemas de transporte neumático, en procesos industriales como la siderurgia, la criogenia, entre otros. En el hogar se utilizan como parte de las herramientas de soplado de hojas.

Los tipos más comunes de sopladores son:

- Soplador centrífugo: Está formado por una cubierta metálica, la rueda de un soplador y las aletas metálicas. Las aspas o aletas posicionadas sobre un único eje que gira, succiona el aire/gas por una entrada amplia y lo aceleran mediante fuerza centrífuga para ser descargado del lado contrario.

- Soplador de flujo axial: Contiene una serie de aspas rotativas en forma de abanico que aceleran el gas de un lado al otro, comprimiéndolo. Su función y acción es parecida a la de las turbinas.

2.2 Elementos Mecánicos

2.2.1 Catarina

Se refiere a un engrane de dientes en V es utilizado en maquinaria como máquinas de tortillerías, cortadoras, troqueladoras, etcétera. También es usado en mecánica de motocicletas para la transferencia de poder, en los vehículos antiguos se utilizaba para la distribución, este término es poco técnico, ya que el correcto es engrane de dientes cónicos. Están fabricadas con acero 1045, acero inoxidable, acero comercial y acetal. Así mismo hay medidas que van de paso 25 hasta 240 en modelo sencillo, doble, triple, y con mamelón tipo A, B, C y D. Sirven para diferentes aplicaciones de una transmisión. En la figura 2.8 se muestran distintas catarinas, con paso y cantidad de dientes distintos.

El número nominal de una Catarina es el mismo número nominal que el de la cadena correspondiente. Los rollos de cadena se colocan deslizando la misma sobre la Catarina, por lo tanto deben estar suficientemente apretadas para su resistencia en un uso constante. Las



Fig. 2.8: Catarina

catarinas de paso 40 al paso 120 de un mamelón para cadena sencilla y doble, son prácticamente endurecidas inductivamente, también el número de dientes es pequeño. El endurecimiento por inducción o por llama será utilizado como mejor convenga a cada aplicación individual. En realidad el diámetro y paso del sprocket determinan:

- Piñón o motriz donde la reducción sea de 4 : 1 o mayor.
- Transmisiones de velocidad lenta (100 FPM o menos).
- Donde el factor de seguridad sea menor que el estándar.
- Condiciones abrasivas poco usuales.

2.2.2 Cadena Industrial

Una cadena es un componente confiable de una máquina, que transmite energía por medio de fuerzas extensibles, y se utiliza sobre todo para la transmisión y transporte de energía en los sistemas mecánicos. La función y las aplicaciones de la cadena son similares a la de una correa. Un tipo de cadena se muestra en la figura 2.9.



Fig. 2.9: Cadena

La cadena de rodillo de acero está formada por una serie de piezas de revolución que actúan como cojinetes, estando situadas cada conjunto a una distancia precisa del otro mediante otras piezas planas llamadas placas. El conjunto cojinete está formado por un pasador y un casquillo

sobre el que gira el rodillo de la cadena. El pasador y el casquillo son cementados para permitir una articulación bajo presiones elevadas, y para soportar las presiones generadas por la carga y la acción de engrane impartida a través de los rodillos de cadenas, generalmente las placas exteriores e interiores se someten a un proceso de templado para obtener una mayor tenacidad.

Las principales características de las transmisiones de cadena, como la mostrada en la figura 2.10, son las siguientes:

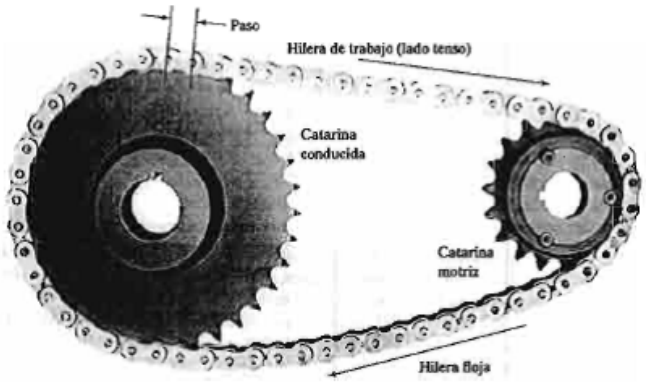


Fig. 2.10: Transmisión de Cadena

- La reducción y aumento de la velocidad puede ser acomodada fácilmente.
- La cadena se puede acomodar a grandes distancias de eje-centro (menos de 4 m), y son más versátiles.
- Es posible utilizar la cadena con los ejes múltiples o puede ser operada a ambos lados de la cadena.
- La estandarización de cadenas bajo la American National Standards Institute (ANSI), la organización internacional de la estandarización (ISO), y estándares industriales japoneses (JIS) permite la facilidad de la selección de las cadenas.
- Es fácil cortar y ensamblar las cadenas.
- El diámetro del piñón para un sistema de cadena puede ser más pequeño que una polea de una correa, mientras que en ambas se transmite el mismo esfuerzo de torsión.

- Los piñones están conforme a menor desgaste que los engranajes porque los piñones distribuyen la carga de esfuerzo a través de sus dientes.

La mayoría de las cadenas se fabrican de metal (generalmente de acero) o de plástico dirigido, que pueden ser afectadas por las condiciones ambientales donde ellas se utilizan. Por ejemplo, la temperatura o la cantidad de polvo en el aire puede afectar a la cadena. Cuando seleccionamos una cadena para su uso, debemos considerar el ambiente en el cual estas van a operar.

2.3 Herramientas de Programación

2.3.1 SolidWorks

SolidWorks es un software CAD (diseño asistido por computadora) para modelado mecánico en $3D$, desarrollado en la actualidad por SolidWorks Corp., una filial de Dassault Systèmes, S.A. (Suresnes, Francia), para el sistema operativo Microsoft Windows. Su primera versión fue lanzada al mercado en 1995 con el propósito de hacer la tecnología CAD más accesible. Su logotipo se muestra en la figura 2.11.



Fig. 2.11: Solidworks.

El programa permite modelar piezas y conjuntos y extraer de ellos tanto planos técnicos como otro tipo de información necesaria para la producción. Es un programa que funciona con base en las nuevas técnicas de modelado con sistemas CAD. El proceso consiste en traspasar la idea mental del diseñador al sistema CAD, çonstruyendo virtualmente"la pieza o conjunto. Posteriormente todas las extracciones (planos y ficheros de intercambio) se realizan de manera bastante automatizada (Cantor Forero y cols., 2015).

2.3.2 Ionic Framework V2.0

Ionic es un completo SDK de código abierto para el desarrollo de aplicaciones móviles híbridas. Construido sobre AngularJS y Apache Cordova, Ionic proporciona herramientas y

servicios para desarrollar aplicaciones móviles híbridas usando tecnologías Web como CSS, HTML5 y Sass. Las aplicaciones se pueden construir con estas tecnologías Web y luego se distribuyen a través de tiendas de aplicaciones nativas para que se instalen en dispositivos aprovechando Cordova. Ionic fue creado por Max Lynch, Ben Sperry, y Adam Bradley de Drifty Co. en 2013. Su logotipo se puede observar en la figura 2.12.



Fig. 2.12: IONIC FrameWork V2.0

Ionic Creator es una herramienta de construcción de interfaz de arrastrar y soltar. App Builder para Ionic también es proporcionado por Appery.io. Ionic proporciona toda la funcionalidad que se puede encontrar en SDKs de desarrollo móvil nativo. Los usuarios pueden crear sus aplicaciones, personalizarlas para Android o iOS e implementarlas a través de Córdoba. Ionic incluye componentes móviles, tipografía, paradigmas interactivos y un tema de base extensible.

Usando Angular, Ionic proporciona componentes personalizados y métodos para interactuar con ellos. Uno de estos componentes, la repetición de la colección, permite a los usuarios desplazarse por una lista de miles de elementos sin resultados de rendimiento. Otro componente, scroll-view, crea un contenedor desplazable con el que los usuarios pueden interactuar utilizando un sistema delegado con influencia nativa.

2.3.3 RStudio

R es un proyecto de software libre de GNU y se podría definir desde dos puntos de vista, por una parte es un lenguaje de programación y por otra un entorno de trabajo, estando ambos orientados al cálculo estadístico y a la generación de gráficas.

Como lenguaje de programación proporciona una amplia variedad de técnicas y recursos para el trabajo con gráficas y análisis estadístico y, a su vez, es altamente ampliable. Cuenta con

una comunidad extensa de desarrolladores, investigadores y usuarios. Se distribuye con licencia GNU GPL v2 y está disponible para distintos sistemas operativos de tipo Unix y similares. Como entorno de trabajo se entiende como un sistema totalmente planificado y coherente y no una acumulación incremental de herramientas muy específicas y poco flexibles, como es frecuentemente el caso con otro software de análisis de datos. En este caso el entorno de trabajo R nos proporciona una serie de utilidades para manipulación de datos, cálculo y representación gráfica.

RStudio (ver figura 2.13) es un entorno de desarrollo integrado (IDE) para el lenguaje de programación R, dedicado a la computación estadística y gráficos. Incluye una consola, editor de sintaxis que apoya la ejecución de código, así como herramientas para el trazado, la depuración y la gestión del espacio de trabajo. RStudio está disponible para Windows, Mac y Linux o para navegadores conectados a RStudio Server o RStudio Server Pro (Debian / Ubuntu, RedHat / CentOS, y SUSE Linux), además, RStudio tiene la misión de proporcionar el entorno informático estadístico R. Permite un análisis y desarrollo para que cualquiera pueda analizar los datos con R.



Fig. 2.13: Software RStudio.

2.3.4 MatLab R2015

MatLab (abreviatura de MATrix LABoratory, "laboratorio de matrices") es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows, Mac OS X y GNU/Linux .

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI)

y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MatLab dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MatLab con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets).

Es un software muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL. Su logotipo se observa en la figura 2.14.



Fig. 2.14: MatLab R2015.

2.4 Inteligencia Artificial

En la actualidad uno de los proyectos más ambiciosos de la informática es la inteligencia artificial, por tal motivo es difícil definir exactamente qué es y los alcances que tiene. Es de fundamental importancia conocer los orígenes de su nombre, es decir el significado de la palabra inteligencia y así mismo el de la palabra artificial, mismos que según (Auruz, 1998) son:

- Inteligencia: Es la potencia intelectiva, la facultad de conocer o de entender. El grado en que un individuo puede resolver satisfactoriamente una nueva situación o un problema. La inteligencia está basada en el nivel de conocimientos individuales y en la habilidad de manipular y reformular apropiadamente los conocimientos en base a los datos que se proporcionan como requerimientos para resolver algún problema o situación.

- Artificial: Es lo hecho por mano y arte del hombre, falso, no natural.

Ahora bien, según (Gutiérrez, 2006) la inteligencia artificial es una de las áreas más fascinantes y con más retos de las ciencias de la computación ya que ha tomado a la inteligencia como la característica universalmente aceptada para diferenciar a los humanos de otras criaturas ya sean vivas o inanimadas, para construir programas o computadoras inteligentes.

Por otro lado (Bourcier, 2003) dice que la inteligencia artificial es una rama de la informática que intenta reproducir las funciones cognitivas humanas como el razonamiento, la memoria, el juicio o la decisión y, después, confiar una parte de esas facultades, que se consideran signos de inteligencia, a los ordenadores.

En general se puede decir que la inteligencia artificial es aquella disciplina que tiene como objetivo el estudio de la conducta humana, mediante el análisis del comportamiento inteligente del ser humano, mismo que se puede denominar análisis de los procesos cognoscitivos, debido a que estos se enfocan en el estudio de los procesos internos que conducen al aprendizaje (Haces y Hernández, 2011).

Se puede decir que en la actualidad la inteligencia artificial es una disciplina fundamental en la ciencia y la tecnología, misma que tras el paso del tiempo ha creado una serie de conocimientos básicos que le permiten emular diversas capacidades del ser humano para exhibir comportamientos inteligentes. Debido a lo anterior se han desarrollado diversos sistemas que tienen por objetivo perfeccionar las distintas capacidades del ser humano con el fin de la reproducción de las mismas.

Según (Auruz, 1998) una técnica de la Inteligencia Artificial es un método que utiliza conocimiento representado de tal forma que:

- Representa generalizaciones.
- Es comprendido por las personas que lo proporcionan.
- Se puede modificar fácilmente.
- Puede usarse en gran cantidad de situaciones.

Desde el punto de vista de la Ingeniería la Inteligencia Artificial utiliza diversas herramientas en la solución de problemas, estas herramientas se presentan en distintas técnicas, mismas que

proveen elementos fundamentales en las áreas de la Inteligencia Artificial, entre las técnicas básicas se pueden citar aquellas que aparecen en la figura 2.15.



Fig. 2.15: Técnicas básicas utilizadas en Inteligencia Artificial

2.4.1 Redes Neuronales

Según (Vázquez, 2009) son sistemas compuestos por estructuras de red con un gran número de conexiones entre diferentes capas de procesadores, los cuales a su vez tienen asignadas diferentes funciones, dentro de dichos procesadores se efectúa una labor de aprendizaje por la reproducción de las salidas de un conjunto de señales de entrenamiento.

Se puede decir que las redes neuronales constituyen una tecnología, la cual trata con éxito algunos problemas clásicos de la IA, haciendo énfasis en el reconocimiento de formas y de la palabra hablada.

Por otro lado, según (Zampayo, 2004) las redes neuronales son sistemas de cómputo distribuidos y paralelos inspirados en la estructura del cerebro humano. El cerebro humano consta de miles de millones de neuronas; cada una conectada a miles de otras neuronas en una estructura distribuida, con paralelismo masivo. Este tipo de estructura otorga al cerebro una gran ventaja en la mayoría de las capacidades perceptivas, motrices y creativas.

Las redes neuronales almacenan la información de manera distinta que las computadoras tradicionales. Los conceptos se representan como patrones de actividad entre varias neuronas, de modo que son menos susceptibles a averías de la máquina. Una de las ventajas de utilizar las redes neuronales es que pueden seguir funcionando aunque se destruyan algunas de sus neuronas, esto es atribuido a su estructura de red.

Sin embargo, (Huerta, 2009) dice que ante las dificultades de imitar algunas funciones del cerebro humano, las nuevas arquitecturas para redes neuronales se han inspirado en hechos que conciernen al tratamiento de grandes cantidades de información procedente de los sentidos, la memoria, la capacidad de aprendizaje y la capacidad de procesamiento de información a gran velocidad. En términos prácticos, las arquitecturas de las redes neuronales pueden usarse para tareas como la visión; mecanismos de aprendizaje para el reconocimiento de voz, de forma que sus resultados alimenten programas simbólicos de la IA.

Modelo de una Neurona Artificial

El modelo presenta (Rumelhart y McClelland, 1986), define un elemento de proceso (EP), o neurona artificial, como un dispositivo que a partir de un conjunto de entradas, x_i $(i = 1...n)$ o vector x, genera una única salida y_k.

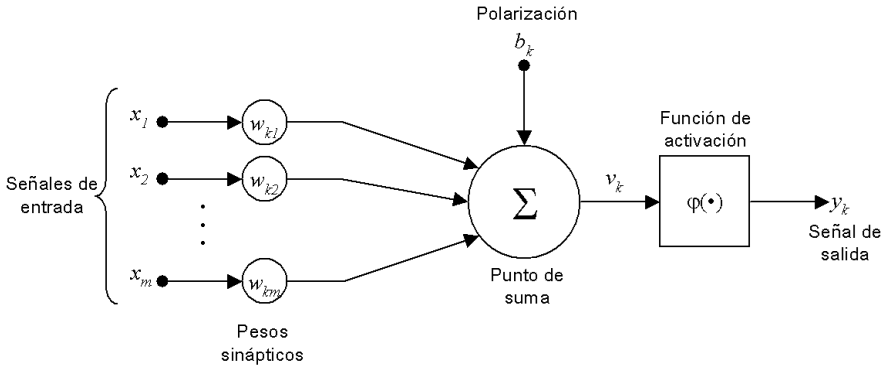


Fig. 2.16: Modelo de Neurona Artificial descrito por Rumelhart y McClelland

En el modelo de una neurona presentado en la figura 2.16 se pueden identificar cuatro elementos básicos del modelo neuronal.

- Enlaces de conexión.- Parametrizados por los pesos sinápticos w_{kj}. Es importante notar

que el primer subíndice corresponde a la neurona receptora, mientras que el segundo subíndice corresponde a la neurona emisora. Si $w_{kj} > 0$ entonces la conexión es excitadora así mismo si $w_{kj} < 0$ la conexión es inhibidora.

- Sumador ($\sum$).- Suma los componentes de las señales de entrada multiplicadas por w_{kj} .
- Función de activación (φ).- Función que describe el valor de la entreda del sistema menos el umbral.
- Umbral .- Desplaza la entrada a $\varphi(x_j)$.

Es posible describir la neurona k en las siguientes ecuaciones:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.1}$$

$$y_k = \varphi(u_k + b_k) \tag{2.2}$$

donde $x_1, x_2, ...x_m$ son las señales de entrada; $w_{k1}, w_{k2}, ...w_{km}$ son los pesos sinápticos de la neurona k; u_k es la combinación lineal de las entradas ponderadas; b_k es la polarización; φ es la función de activación; finalmente y_k es la señal de salida de la neurona.

La polarización es un parámetro externo de la neurona k, pero es posible considerarla como parte de las señales de entrada, de tal forma que si se combinan (2.1) y (2.2) se tiene:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \tag{2.3}$$

$$y_k = \varphi(v_k) \tag{2.4}$$

En la ecuación (2.3) se ha agregado una nueva sinapsis. Su entrada es:

$$x_0 = +1$$

y el peso correspondiente es:

$$w_{k0} = b_k \tag{2.5}$$

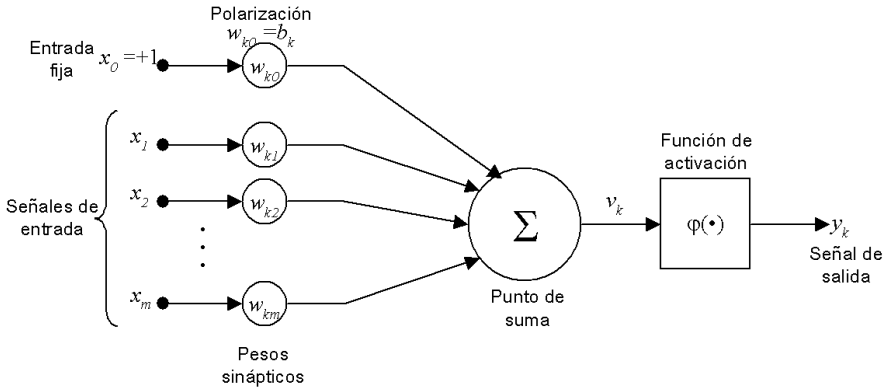


Fig. 2.17: Modelo de Neurona Artificial descrito por Rumelhart y McClelland

De tal forma que el modelo neuronal presentado anteriormente se puede reformular como ver figura 2.17:

Función de Activación

Una neurona biológica puede estar activa (excitada) o inactiva (no excitada); es decir, que tiene un "estado de activación". Las neuronas artificiales también tienen diferentes estados de activación; algunas de ellas solamente dos, al igual que las biológicas, pero otras pueden tomar cualquier valor dentro de un conjunto determinado (Matich, 2001).

La función activación calcula el estado de actividad de una neurona; transformando la entrada global (menos el umbral) en un valor (estado) de activación, cuyo rango normalmente va de ($0a1$) o de ($-1a1$). Esto es así, porque una neurona puede estar totalmente inactiva ($0o-1$) o activa (1).

En este caso, fue utilizada la función de activación sigmoidal, la cual se describe a continuación:

Los valores de salida que proporciona esta función están comprendidos dentro de un rango que va de $0a1$. Al modificar el valor de g se ve afectada la pendiente de la función de activación. La ecuación característica de una función sigmoidal es la siguiente:

$$\varphi(v) \frac{1}{1 + \exp^{-av}} \tag{2.6}$$

y la gráfica que describe esta función se observa en la figura 2.18.

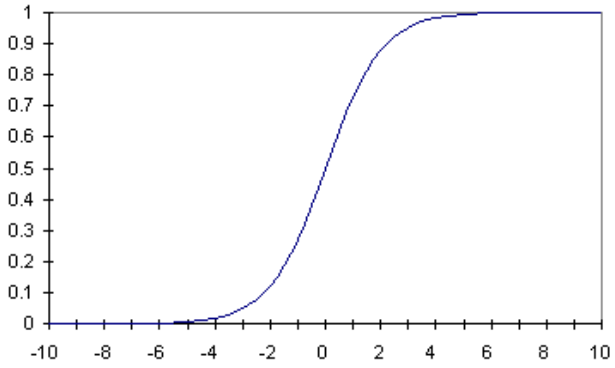


Fig. 2.18: Función sigmoidal

Aprendizaje Supervizado

- Aprendizaje por corrección de error: Consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos a la salida de la red, es decir, en función del error cometido en la salida. Un ejemplo de este tipo de algoritmos lo constituye la regla de aprendizaje del Perceptron, utilizada en el entrenamiento de la red del mismo nombre que desarrolló Rosenblatt en 1958 (Matich, 2001). Esta es una regla muy simple, para cada neurona en la capa de salida se le calcula la desviación a la salida objetivo como el error. El cual luego se utiliza para cambiar los pesos sobre la conexión de la neurona precedente. El cambio de los pesos por medio de la regla de aprendizaje del Perceptron se realiza según la siguiente regla:

$$\Delta wij = \sigma * y_j * (a_{qi} - y_i) \tag{2.7}$$

Donde:

- a_{qi} es la salida deseada/objetivo de la neurona de salida N_i.
- $(a_{qi}–y_i)$ la desviación objetivo de la neurona N_i .
- σ es aprendizaje.

La salida de la neurona N_j (y_j) se utiliza, porque este valor influye en la entrada global y, por ende, en la activación y luego en la salida de la neurona N_i. Esto es semejante a un "efecto en cadena", ver figura 2.19.



Fig. 2.19: Salida de Neurona

Otro algoritmo muy conocido y que pertenece a esta clasificación es la regla de aprendizaje Delta o regla del mínimo error cuadrado (LMS Error: Least Mean Squared Error), que también utiliza la desviación a la salida objetivo, pero toma en consideración a todas las neuronas predecesoras que tiene la neurona de salida. Esto permite cuantificar el error global cometido en cualquier momento durante el proceso de entrenamiento de la red, lo cual es importante, ya que cuanto más información se tenga sobre el error cometido, más rápido se puede aprender. Luego el error calculado (δ) es igualmente repartido entre las conexiones de las neuronas predecesoras.

Por último se debe mencionar la regla de aprendizaje de propagación hacia atrás o de backpropagation, también conocido como regla LMS multicapa, la cual es una generalización de la regla de aprendizaje Delta. Esta es la primer regla de aprendizaje que permitió realizar cambios sobre los pesos en las conexiones de la capa oculta.

El nombre de backpropagation resulta de la forma en que el error es propagado hacia atrás a través de la red neuronal, en otras palabras el error se propaga hacia atrás desde la capa de salida. Esto permite que los pesos sobre las conexiones de las neuronas ubicadas en las capas ocultas cambien durante el entrenamiento.

El cambio de los pesos en las conexiones de las neuronas además de influir sobre la entrada global, influye en la activación y por consiguiente en la salida de una neurona. Por lo tanto, es

de gran utilidad considerar las variaciones de la función activación al modificarse el valor de los pesos. Esto se llama sensibilidad de la función activación, de acuerdo al cambio en los pesos.

3. DESARROLLO

El objetivo principal de este proyecto fue el diseñar e implementar un sistema para dosificar y dispensar el alimento en granjas acuícolas. Para poder lograr esto, se diseñó un plan de trabajo, basando el desarrollo en tres distintas fases, entre las cuales se pueden destacar las siguientes:

- Diseño Mecánico.
- Diseño Electrónico y de Control.
- Diseño de Aplicación Móvil.

Antes de comenzar con el desarrollo del proyecto, y de cada una de las fases de diseño mencionadas anteriormente, se definió la estructura básica que debe tener el sistema, la cual, se puede resumir como se muestra en la siguiente figura 3.1.

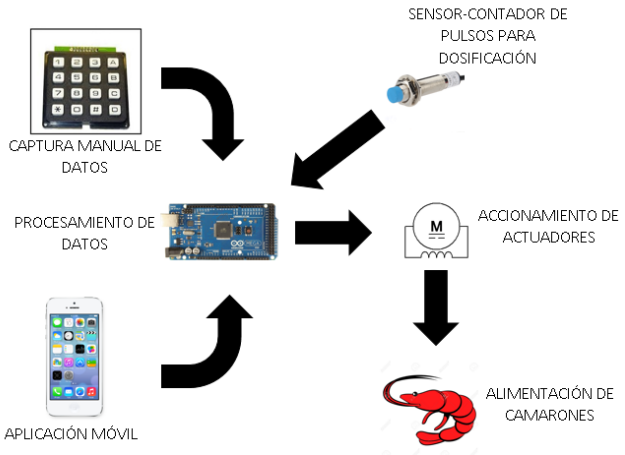


Fig. 3.1: Funcionamiento General del Sistema.

3.1 Diseño Mecánico

En esta sección, se justifica la selección de los equipos utilizados, se muestran los diseños y sistemas realizados. Como se explicó anteriormente, el sistema se puede describir en dos grandes rasgos, aquel que basa su funcionamiento en la selección manual de la cantidad de alimentos a dosificar, ya sea por el dispositivo móvil o por un teclado matricial empotrado en la camioneta, y el sistema automático, el cual requiere una mínima intervención del operador.

Ambos sisteman deberían de cumplir con la condición de funcionar indistintamente sobre el mismo sistema electromecánico, por este motivo, se decidió comenzar con la elaboración del diseño de estos sistemas.

3.1.1 Selección de Elementos Mecánicos y de Almacenamiento

Lo primero que se realizó fue un análisis de distintos factores, para lo cual, se realizó una jerarquización de los requerimientos y limitaciones que podía presentar este diseño.

A continuación se anexa una tabla 3.1 que contiene los datos utilizados para la jerarquización:

Requerimiento	Ponderación
Debe funcionar con 12V	10
Tolva con capacidad de $1.5Tn.$	5
Soplador con alimentación de 12V	6
Diseño de VDR con compartimientos de volúmen definido	10
Ancho de estructura propircional a Chasís	8
Altura total no mayor a 6 m.	7
Estructura de material inoxidable	9
Estructura recubierta con pintura epóxica	6
VDR puede ser cambiada con facilidad	3
Distintos tamaños de VDR para distintos ciclos	4

Tab. 3.1: Jerarquización de Requerimientos de Sistema alimentador

Basándose en los requerimientos necesarios para el funcionamiento adecuado del sistema, se procedió a realizar la selección de elementos adecuados para el sistema. Para poder realizar una elección precisa de dichos componentes, se diseñó una matriz morfológica para poder realizar la comparación de los elementos mecánicos disponibles y, en base a los resultados obtenidos, poder seleccionar los dispositivos óptimos para garantizar un funcionamiento adecuado del sistema.

La matriz morfológica que se desarrolló se muestra en la siguiente figura 3.2 :

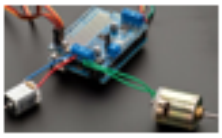
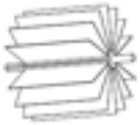
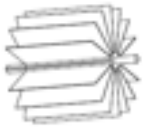
SISTEMA	SOLUCION 1 Arduino Mega + Motor a Pasos + VDR 16 Secciones + Blower DC	SOLUCION 2 Arduino Mega + Motor DC + VDR 4 secciones + Blower con motor a gasolina	SOLUCION 3 Adafruit Motor Shield V2 p/Arduino Uno + 1.4. + Motor a pasos + VDR 16 Secciones + Blower DC
Microcontrolador			
Actuadores			
Tipo de Rotor para VDR			
Mecanismo			

Fig. 3.2: Matriz Morfológica

En base a las observaciones realizadas, y en relación a los requerimientos del sistema, se llegó a la conclusión de que el sistema más viable es el segundo (ver figura 3.2). En base a lo anterior, se comenzó con el diseño mecánico de los elementos del sistema.

Tolva Contenedora

El sistema diseñado, el cual se encuentra montado sobre una camioneta, consta de una tolva contenedora, dentro de la cual se encuentra el alimento a dispensar dentro de los estanques. Se

decidió que la tolva iba a ser cuadrada, debido a que, por las limitaciones de espacio, es posible almacenar una mayor cantidad de alimento.

En cuanto al material con el cual se pretende elaborar la Tolva, se seleccionó un acero inoxidable con un espesor de $\frac{3}{16}$", esto debido a la exposición que va a tener el sistema con distintos elementos oxidantes y corrosivos, los cuales predominan en el ambiente de trabajo.

Los requerimientos de almacenamiento para la tolva, son de 1500 Kg, los cuales garantizan un abasto de la totalidad de los estanques a alimentar. Para lograr esto, fue necesario conocer el peso específico que tiene el alimento, debido a que existen distintas clases y tipos de alimento. Por tanto, se seleccionó uno en específico para todos los diseños. El alimento seleccionado cuenta con un peso específico de $650\frac{Kg}{m^3}$. Basándome en esta propiedad, fue que se diseñó la tolva alimentadora, la cual debe cumplir con las siguientes propiedades:

$$650\ Kg \rightarrow 1\ m^3$$
$$1500\ Kg \rightarrow 2{,}3077\ m^3 \tag{3.1}$$

Una vez que se obtuvo el volmen que debía tener la tolva contenedor se realizó, apoyado en SolidWorks, el diseño mecánico de este. En la figura 3.3 se observa el diseño creado.

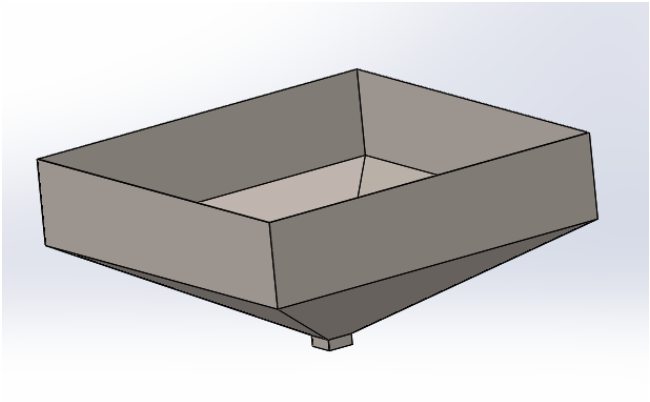
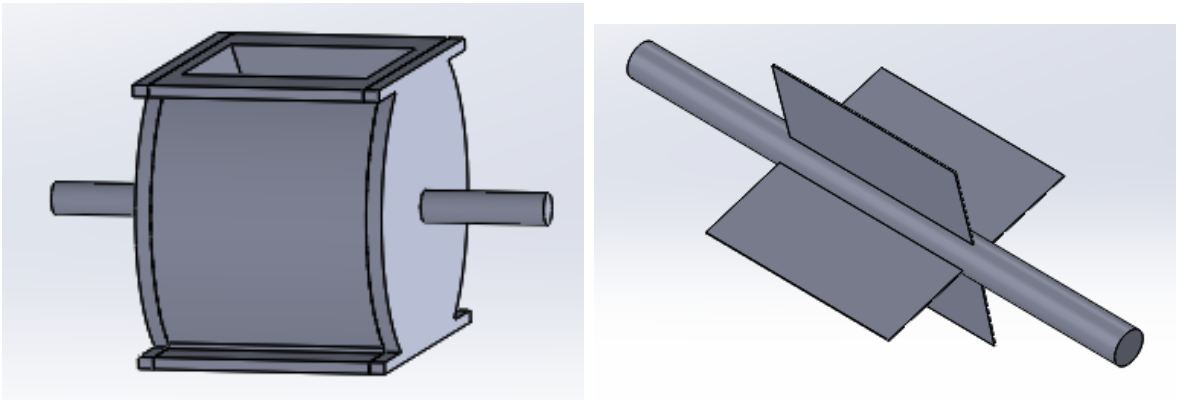


Fig. 3.3: Diseño de Tolva Contenedora

Válvula Dosificadora Rotativa

La Válvula Dosificadora Rotativa (VDR), es el elemento mecánico encargado de la dosificasión exacta del alimento. Este elemento se encuentra conectado a un elemento motriz que le permite rotar, además, se encuentra conectado en la parte inferior de la tolva contenedora. Esta válvula cuenta con 4 compartimientos, con similar capacidad volumétrica, las cuales se van llenando por medio de gravedad, dichos compartimientos anteriormente mencionados tienen una capacidad de almacenamiento de 250 gramos.

El funcionamiento de la VDR es muy básico, debido a que se van rellenando por gravedad, siempre se encuentran llenos los compartimientos de la parte superior. Cuando se les solicita que dispensen el alimento, comienzan a girar hasta que dispensen la cantidad de alimento deseado. Este funcionamiento se logra gracias al motor que hace girar esta válvula y a un sensor, el cual es utilizado como encoder (se describirá más adelante), gracias al cual es posible contabilizar la cantidad de giros dados por la VDR. El diseño de la VDR creado se muestran en las figuras 3.4a y 3.4b.



(a) Diseño de la Válvula Dosificadora Rotativa *(b)* Diseño de Rotor para VDR

Fig. 3.4: Diseño de VDR

Motor de 12VCD

El motor de 12 V de corriente directa seleccionado en este caso, es un motor comúnmente utilizando en los parabrisas automotrices. El motor seleccionado, pertenece al modelo 1379887. Las especificaciones de este motor se muestran en la tabla 3.2:

Voltaje de Operación	12 VCD	Prueba de voltaje: 13.5V
Corriente sin carga	Baja velocidad: ¡2.5A	Alta Velocidad: ¡3A
Velocidad sin carga.	Baja Velocidad: 45±5 rpm	Alta Velocidad: ¡65 ± rpm
Corriente de Operación	Baja Velocidad: ¡3.5A	Alta Velocidad: 5.5A
Velocidad con carga	Baja Velocidad: 40 ± 5	Alta Velocidad: 60 ± 5rpm
Ruido	Baja Velocidad: ¡55 Db	Alta Velocidad: ¡65 Db
Torque	5.5 N.m	

Tab. 3.2: Especificaciones de Motor Wiper

El motivo principal para seleccionar dicho este motor, es que su alimentación es de 12 V (ver fig. 3.5), motivo por el cual se facilita su control gracias al microcontrolador seleccionado. La velocidad y el torque del motor se asemejan mucho a las deseadas, pero para poder conseguir que funcionen de manera correcta, fue necesario implementar una transmisión mecánica con catarinas y cadenas.

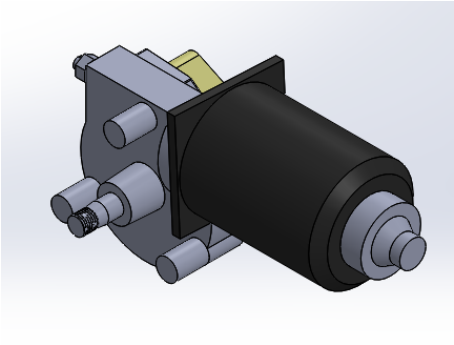


Fig. 3.5: Diseño de Motor 12 VCD

Transmisión de Potencia Mecánica

Como se menciona anteriormente, fue necesario el diseño de una transmisión mecánica. La finalidad del uso de este mecanismo fue el poder aumentar el torque de generado por el motor, además, de reducir un poco la velocidad del sistema. El sistema fue elaborado con catarinas y cadenas.

Los elementos seleccionados fueron los siguientes:

- Una cadena de Paso 40.
- Una Catarina Paso 40, 11 Dientes (motriz).

- Una Catarina Paso 40, 22 Dientes (inducido).

Motor Blower a Gasolina

La función principal del sistema es la de dispensar el alimento hacia los estanques. Para poder realizar dicha función, fue necesario encontrar un dispositivo capaz de realizar esta tarea. En esta ocasión, se seleccionó un blower industrial. Este dispositivo es capaz de generar un flujo de aire, el cual permite arrojar el agua hacia el estanque. El dispositivo seleccionado cumple con los siguientes requerimientos (ver tabla 3.3):

Potencia	1,5 HP
Motor	5 HP HONDA
Presion	22 KPA
Q/ M^3 / H	189 M^3 H
Max. Velocidad	3600 RPM
Velocidad Normal	$2500 - 3000$ RPM
Presión Normal	$12 - 16$ KPA
Capacidad Tanque	4 Lts
Consumo por hora	1,2 Lts
Máximo trabajo	3 Hrs

Tab. 3.3: Especificaciones del Blower

Como se puede observar, la alimentación de este dispositivo es a base de un motor a gasolina (ver figura 3.6). Durante el recorrido que dará el camión, el blower se encontrará encendido en todo momento.

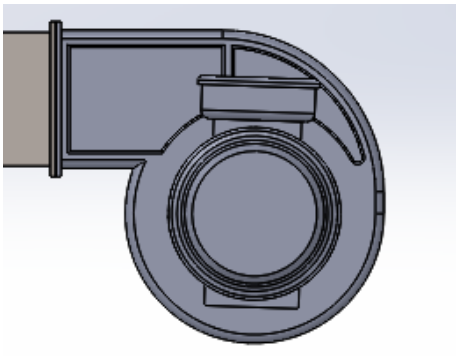


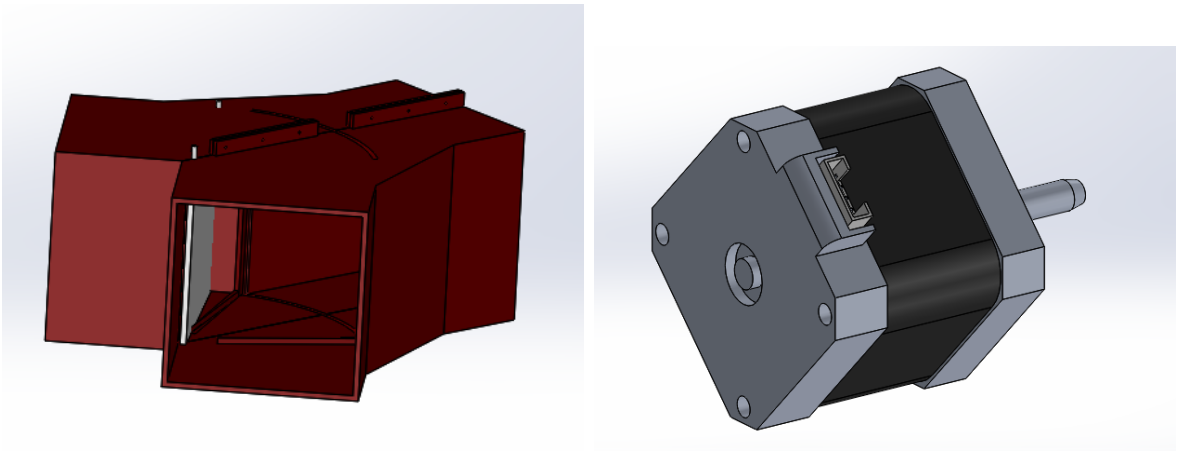
Fig. 3.6: Diseño de Blower con Motor a Gasolina

Tubo de Salida

En la parte final del sistema se encuentra un tubo de salida, el cual es el encargado de guiar y de direccionar el alimento hacia los estanques. El tubo de salida es un tubo de PVC de 4"de diámetro. Este tubo podrá arrojar el alimento a distintas alturas, dependiendo de las condiciones, tanto climáticas, como del camino dentro de las granjas acuícolas.

Selector de Posición (Motor a pasos)

Además, la salida contará con una válvula que le ayudará a seleccionar el lado por el cual saldrá el alimento (ver figura 3.7a). El funcionamiento de esta válvula se apoya en un motor a pasos (ver figura 3.7b), el cual abrirá y cerrará la válvula, dependiendo la selección realizada.



(a) Diseño de la válvula selectora *(b)* Diseño del motor de la válvula selectora

Fig. 3.7: Selector de posición

3.1.2 Acoplamiento Mecánico

El paso siguiente fue, inicialmente, el realizar el acoplamiento de los diseños realizados, esto para poder observar como iba a quedar el sistema antes de comenzar con el ensamble mecánico. El ensamble de los componentes diseñados se muestra en la figura 3.8.

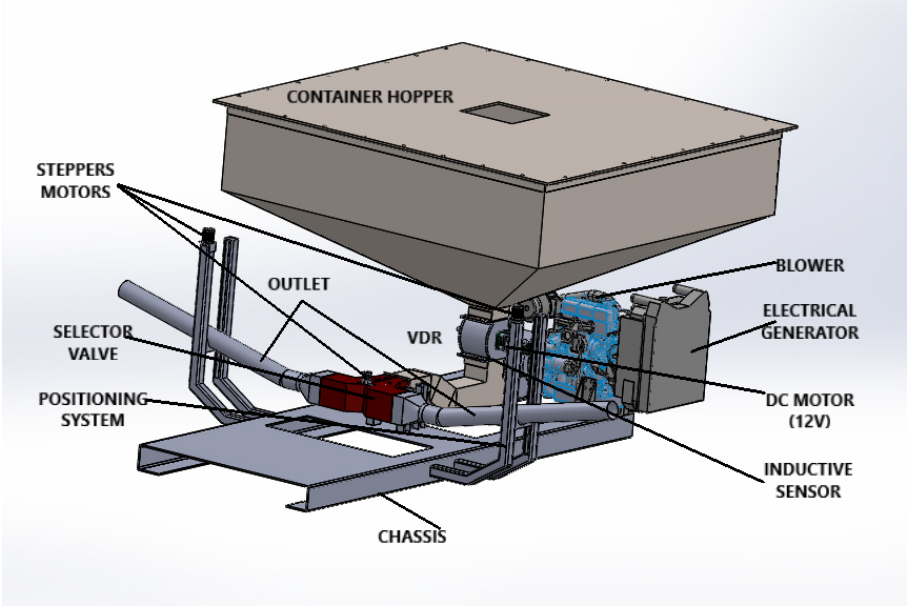


Fig. 3.8: Diseño Final de Sistema alimentador

3.2 Programación del Microcontrolador

Después de definir los elementos motrices electromecánicos del sistema, y siendo tomados como base los anteriores, se procedió a realizar el código de funcionamiento para el control del sistema. Inicialmente, se realizó la programación de un sistema al cual se le digitará, por medio de un teclado, el estanque en el que estaba, la cantidad de alimento a dosificar y una serie de datos de manera manual.

Cuando se observó la funcionalidad del sistema anterior, se procedió a realizar un sistema, el cual, basado en información de una base de datos, y con ayuda del método de redes neuronales, te indicará la cantidad de alimento a dosificar de manera automática.

Ambos sistemas requieron de programas de control distintos, así como el uso de técnicas y elementos variados, así como un funcionamiento distinto, por lo que se decidió separar cada uno de los métodos, los cuales se describen a continuación.

3.2.1 Captura de Datos Manual

El sistema de captura de datos manual se describe en este apartado. El sistema, básicamente, contaba con una serie de componentes los cuales iban montados dentro de la camioneta, entre

los cuales se rescatan un teclado matricial, mediante la cual se captaban los datos del sistema, una pantalla LCD donde se visualizaba la información del sistema, una memoria SD donde se registraba cada dosificación realizada, una tarjeta de reloj para conocer el tiempo real de cada acción realizada y una impresora térmica para lograr tener un documento tangible después de cada recorrido. El diagrama funcional del mecanismo creado se anexa en la figura 3.9.

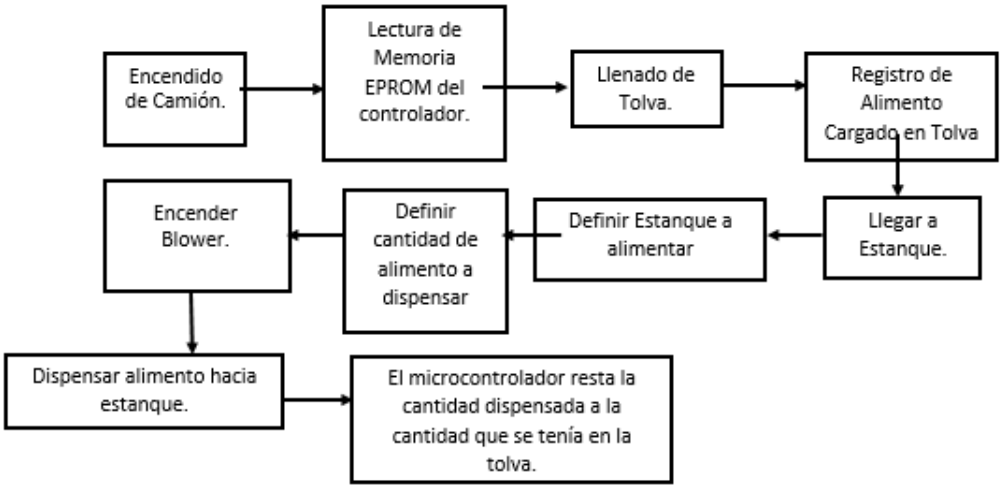


Fig. 3.9: Diagrama Funcional en Sistema de Captura Manual

En la imagen anterior, se muestra la secuencia de trabajo realizada por el sistema desarrollado. Para entender un poco mejor, se realizará la descripción general de sistema.

Lo primero que se supone es que al inicio de la jornada se encuentra el camión apagado. Al momento de encender, se muestra, por medio de una interfaz visual, en este caso, una Pantalla LCD la cantidad de alimento que se tiene en la tolva (datos extraídos de memoria interna de microcontrolador). Una vez que se conoce la cantidad de alimento que se tiene, el operador, basado en los datos que le son proporcionados, decide si es necesario recargar nuevamente la tolva. En nuestra interfaz con la pantalla LCD, nos da la opción de introducir la cantidad de alimento que fue ingresado por medio de un teclado numérico, todo esto para tener conocimiento de la cantidad de alimento que se tiene.

Suponiendo que ya se tiene la cantidad de alimento deseada cargada en la tolva, se comienza con el recorrido estanque por estanque. Al momento de llegar a un estanque, la pantalla LCD nos muestra una opción en la cual podemos ingresar, por medio del teclado, primeramente el

número de estanque al cual se va a alimentar, y después nos pregunta cuál es la cantidad de alimento que debe de dispensarse hacia el estanque. Después de esto, nuestra LCD muestra una opción, la cual nos pide confirmar la información deseada, todo esto se incluye como una opción de seguridad, es decir, si el operador llega a capturar algún valor erróneo, con esta opción se puede corregir.

Una vez que se tiene la cantidad de alimento a despachar, se procede a dispensar el alimento hacia los estanques. Primeramente, es necesario conocer el tiempo en el cual se debe realizar este proceso. Para eso, el operador debe indicar, con el teclado, la cantidad de tiempo (en segundos) que le llevará a él el recorrido del estanque. Esto es posible gracias a un Microcontrolador, el cual es el dispositivo electrónico responsable de todas las funciones necesarias de control para este proyecto. El microcontrolador, al momento de conocer la cantidad de alimento que se va a utilizar y el tiempo estimado en el cual se va a realizar el recorrido, realiza una operación matemática, la cual, nos permitirá, a través de las señales del microcontrolador, controlar los movimientos necesarios del motor de 12 VDC . Este motor está conectado, con ayuda de un acoplamiento mecánico al dispositivo encargado de dispensar el alimento, el cual es una Válvula Dosificadora Rotativa (VDR), la cual cuenta con compartimientos previamente definidos por un volumen determinado, gracias a esta válvula, el tamaño de los compartimientos, y un sensor inductivo que contabiliza los pulsos que se obtienen al hacer girar la VDR, es posible conocer y controlar exactamente la cantidad de alimento a dispensar.

Además, el sistema contaba con una impresora térmica que realizaba una impresión al terminar cada recorrido, y además, se guardaba toda la información en una memoria SD, incluyendo la fecha y la hora de la dosificación, el recorrido en el que se alimentó y la cantidad de alimento depositado en el estanque.

El sistema descrito anteriormente fue utilizado para realizar pruebas de funcionamiento. Se realizaron dosificaciones de alimento, donde se observabó un error de 0,5Kg., generado por el sistema mecánico. Se realizaron cerca de 100 dosificaciones, donde el promedio del error presentado fue el mismo. Esto sirvió para observar que el sistema, tanto mecánico, como el control realizado eran funcionales.

El código de programación del sistema manual se anexa en el apéndice 1.

Una vez que se observó un comportamiento óptimo del equipo, se decidió sustituir un par de elementos del sistema funcional desarrollado, esto debido a que se buscaba evitar que los datos de la dosificación fueran introducidos por el operador, por lo que se decidió, tomando como base el funcionamiento mecánico ya probado, agregar al sistema un método de control inteligente basado en redes neuronales artificiales para realizar la toma de decisiones con respecto a la cantidad de alimento.

El desarrollo de dicho sistema se muestra en el siguiente apartado.

3.2.2 Control Automático

Como se menciono anteriormente, se decidió desarrollar un sistema que evitara, en lo mayor posible, el contacto del operador con el sistema, por lo cual se decidió agregar al sistema un par de elementos, todos ellos referentes a programación, con la finalidad de tener una interfaz más sencilla para el operador y a su vez que cumpliera con la funcionalidad ya comprobada del sistema.

El hecho de tener que capturar los datos en un teclado, verlos en una pantalla, corroborrar la información impresa en el ticket, entre otras cuestiones, desembocaban en un sistema que resultaba complejo para quienes operaban el sistema, por lo que se decidió eliminar una serie de componenentes, los cuales serían sustituidos por una aplicación móvil, y datos que serían enviados, de manera inalámbrica al dispositivo, para que de esta manera el operador solo tuviera que indicar que ya estaba en posición (indicar estanque en el que se encontraba), y de esa manera se comenzaría la dosificación.

Para lograr el desarrollo de este sistema, se requirió de varias métodos y softwares los cuales se describiran a lo largo de este apartado. Además, se creó una aplicación móvil, de la cual se hablará en el próximo apartado. El diagrama funcional del sistema que se planea realizar se muestra en la figura 3.10.

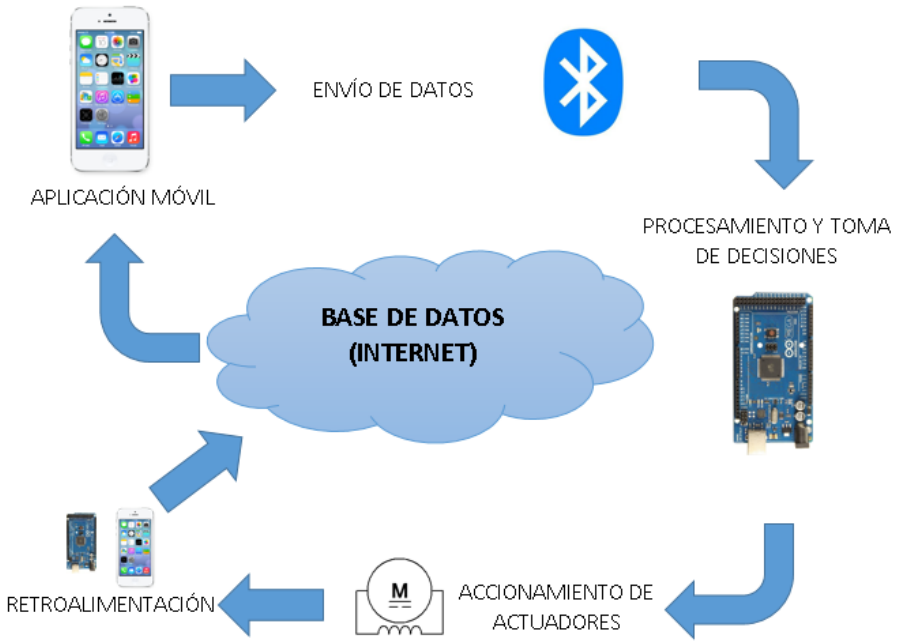


Fig. 3.10: Diagrama funcional del sistema automático

Inicialmente, se debió elegir el método para la toma de desiciones apropiado para el sistema. Debido a una serie de conocimientos adquiridos en la materia de control inteligente, y con las bases de haber realizado algunas prácticas con este método, se decidió utilizar el método de las redes neuronales artificiales. El procedimiento realizado se describe a continuación:

Método de Redes neuronales artificiales

Inicialmente, se decidió cual sería el método de entrenamiento para nuestra red. Se utilizó un método de aprendizaje supervisado, basado en la correción del error cuadrático medio, también conocido como aprendizaje de propagación hacia atrás.

Lo siguiente que se realizó fue definir las entradas y salidas del sistema, así como el número de capas ocultas que tendrá. Se decidió que el sistema contaría con lo siguiente:

- Entradas: Se tendrán 2 entradas, las cuales corresponden a dos datos utilizados para conocer la cantidad de alimento a dosificar, los cuales son, la cantidad de días que tiene el camarón sembrado en el estanque y la cantidad de camarónes sembrados por m^2.

- Salidas: El sistema tendrá 1 salida, la cual indicará la cantidad de alimento, en kilogramos,

que debe de dosificarse al estanque.

- Capas ocultas: El sistema contará con 1 capa oculta, compuesta de 5 neuronas.
- Función de Activación: Función Sigmoidal.

La estructura de la red se presenta en la figura 3.11.

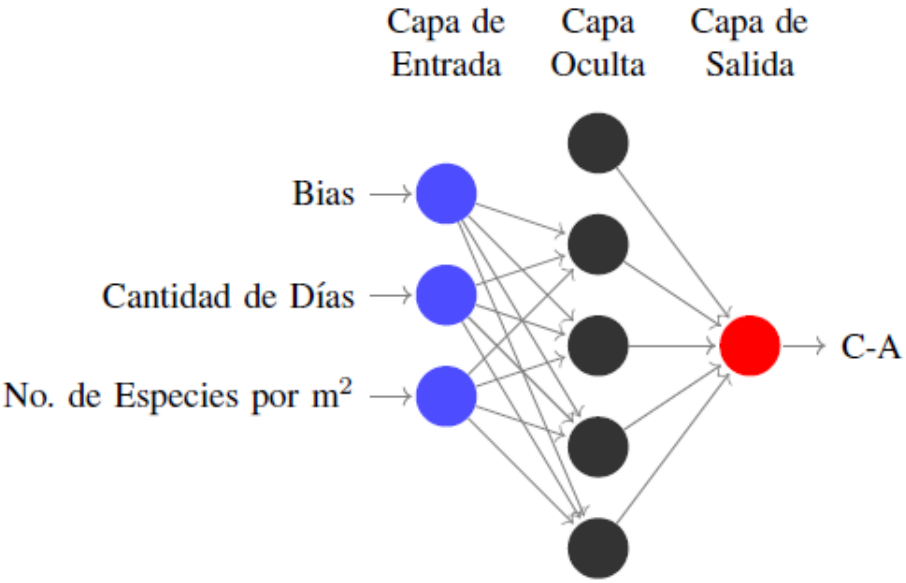


Fig. 3.11: Estructura de Red Neuronal

Una vez que se contó con lo anterior, se realizó la elección de aquellos datos con los cuales se entrenaría la red. Debido a que los métodos de alimentación son muy variados, debido a que cada biólogo tiene su técnica, se utilizaron los datos obtenidos en el periodo de prueba que se tuvo, los cuales se representan acontinuación:

- Días de cultivo : 1 a 120 días.
- Cantidad de especies por m^2: 6 a 15 especies.
- Cantidad de Alimento: 10 a 110 Kg por hectárea.

Teniendo como consideración que la red se entrenó tomando como base un estanque de 1 Ha, se comenzará con el entrenamiento de la red Neuronal. Para poder lograr lo anterior,

se tomararon una serie de datos para poder realizar una tabla para registrar los datos que se tienen.

Los datos se muestan en la siguiente tabla, donde se está considerando que, los valores, tanto de entradas como salidas, fueron mapeados a valores que van desde 0,1 a 0,9, esto debido a que para realizar el entrenamiento de la red se utilizó como función de activación la función sigmoidal.

La tabla que se realizó fue la siguiente:

DT	D	E	C	DT	D	E	C	DT	D	E	C	DT	D	E	C
1	0.1	0.1	0.1	22	0.3	0.4	0.34	43	0.5	0.7	0.58	64	0.8	0.1	0.63
2	0.1	0.2	0.13	23	0.3	0.5	0.37	44	0.5	0.8	0.61	65	0.8	0.2	0.66
3	0.1	0.3	0.16	24	0.3	0.6	0.4	45	0.5	0.9	0.64	66	0.8	0.3	0.69
4	0.1	0.4	0.19	25	0.3	0.7	0.43	46	0.6	0.1	0.48	67	0.8	0.4	0.72
5	0.1	0.5	0.22	26	0.3	0.8	0.46	47	0.6	0.2	0.51	68	0.8	0.5	0.75
6	0.1	0.6	0.25	27	0.3	0.9	0.49	48	0.6	0.3	0.54	69	0.8	0.6	0.78
7	0.1	0.7	0.28	28	0.4	0.1	0.33	49	0.6	0.4	0.57	70	0.8	0.7	0.81
8	0.1	0.8	0.31	29	0.4	0.2	0.36	50	0.6	0.5	0.6	71	0.8	0.8	0.84
9	0.1	0.9	0.34	30	0.4	0.3	0.39	51	0.6	0.6	0.63	72	0.8	0.9	0.87
10	0.2	0.1	0.18	31	0.4	0.4	0.42	52	0.6	0.7	0.66	73	0.9	0.1	0.67
11	0.2	0.2	0.21	32	0.4	0.5	0.45	53	0.6	0.8	0.69	74	0.9	0.2	0.7
12	0.2	0.3	0.24	33	0.4	0.6	0.48	54	0.6	0.9	0.72	75	0.9	0.3	0.73
13	0.2	0.4	0.27	34	0.4	0.7	0.51	55	0.7	0.1	0.5	76	0.9	0.4	0.765
14	0.2	0.5	0.3	35	0.4	0.8	0.54	56	0.7	0.2	0.58	77	0.9	0.5	0.79
15	0.2	0.6	0.33	36	0.4	0.9	0.57	57	0.7	0.3	0.61	78	0.9	0.6	0.82
16	0.2	0.7	0.36	37	0.5	0.1	0.40	58	0.7	0.4	0.64	79	0.9	0.7	0.85
17	0.2	0.8	0.39	38	0.5	0.2	0.43	59	0.7	0.5	0.67	80	0.9	0.8	0.87
18	0.2	0.9	0.42	39	0.5	0.3	0.46	60	0.7	0.6	0.7	81	0.9	0.9	0.9
19	0.3	0.1	0.25	40	0.5	0.4	0.49	61	0.7	0.7	0.73				
20	0.3	0.2	0.28	41	0.5	0.5	0.52	62	0.7	0.8	0.76				
21	0.3	0.3	0.31	42	0.5	0.6	0.55	63	0.7	0.9	0.79				

Tab. 3.4: Datos De red Neuronal

Se recopilaron, dentro de todas las opciones posibles, 81 datos para crear una tabla. Los datos utilizados en esta tabla son los siguientes:

- D.- La letra D representa los Días que tiene la especie plantada en el estanque. Para esta prueba, se tomarán en cuenta 120 días.
- E.- La letra E representa la canditad de Especies sembradas por metro Cuatrado. En este

caso se tomarán Valores desde 6 a 15 Especies.

- C.- La letra C representa la cantidad de alimento que debe dosificarse. La cantidad de alimento a dosificarse va desde 10 hasta 110 KG.

- DT.- Dato numérico.

Teniendo en cuenta los datos anteriores, se procedió a realizar el entrenamiento de la neurona. Para lograr esto se desarrolló, con apoyo del Ing. Arnulfo José Suárez Gaekel, un programa en el software Rstudio para diseñar un algoritmo de aprendizaje de progagación hacia atrás o retropropagación. Cabe resaltar que de la totalidad de datos iniciales (81), solamente se introdujeron a la red el 80 % de ellos, mientras que el porcentaje restante fue utilizado para las pruebas. El programa realizado se anexa en el Apéndice 2.

- Error cuadrático médio esperado: 0,0005.

- Épocas necesarias: $796,975$ (Se habían programado un máximo de $1,000,000$).

- N: 0,1.

En la figura 3.12 se muestra la cantidad de épocas realizadas, así como el error que se iba presentando en la época correspondiente.

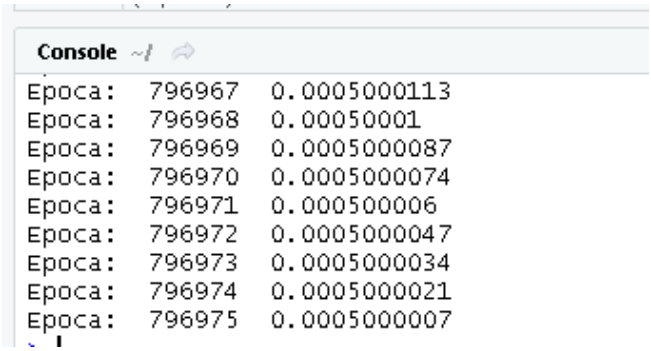


Fig. 3.12: Resultados de entrenamiento Neuronal

Como resultado de este código se obtuvieron el valor final de los pesos sinápticos de la neurona. Los pesos obtenidos se pueden ver en la figura 3.13.

```
> retorno[[1]][[1]]
           [,1]       [,2]       [,3]
[1,]  0.6561117 -1.832009  1.7512827
[2,]  1.5291060 -0.114930  0.5213453
[3,] -0.5862820 -6.280922 -1.6976867
[4,]  4.4599577 -2.022048 -2.0806504
[5,] -1.4629441  2.760987 -0.9299505
> retorno[[1]][[2]]
           [,1]     [,2]      [,3]      [,4]      [,5]     [,6]
[1,] -2.723413 5.166674 0.1700119 -5.257298 -3.428121 6.863272
>
```

Fig. 3.13: Pesos sinápticos obtenidos

Teniendo en cuenta los pesos obtenidos, el paso siguiente fue comprobar que los valores de los pesos sinápticos obtenidos fueran los correctos. Para poder lograr lo anterior, se utilizó el programa MatLab 2015R con el cual se realizó un programa en el cual se simula una propagación de los datos de entrada hacia la salida de la neurona, en el cual se introdujeron los datos de entrada, los pesos correspondientes y como resultado arrojaba el valor de la salida, en escala del 0,1 al 0,9.

El programa elaborado en MatLab se anexa en la apéndice 3. Este programa sirvió como base para comprobar que los valores de pesos sinápticos obtenidos en Rstudio son correctos y que la red quedó entrenada de manera funcional.

En la tabla 3.5 la cual se anexará más adelante, se muestra una relación de los resultados obtenidos por el MatLab, en comparación con los datos obtenidos de la tabla original.

Programación de Microcontrolador con Redes Neuronales

Una vez que se comprobó el funcionamiento de la red neuronal, el paso siguiente fue desarrollar un código en Arduino, capaz de simular dicho comportamiento (Red con propagación hacia adelante).

Tomando como base los valores obtenidos en el programa de Rstudio, y el programa realizado anteriormete, donde se observó el comportamiento manual de sistema, se realizó un programa nuevo para poder cumplir con dicha función.

A diferencia del código manual, este sistema funciona de la siguiente manera:

En su etapa inicial, el opedador debe conectarse con su dispositivo móvil, por medio de bluetooth, con el arduino. Una vez que logra esto, el solamente deberá de indicar el estanque en

el que se encuentra, para que el sistema, que toma lecturas de una base de datos, meta este dato a la red neuronal, para que el sistema decida cual es la cantidad de alimento, en kilogramos, que deben dosificarse.

Este programa se muestra desglosado en el apéndice 4, donde se puede observar que, con conocimientos adquiridos durante los estudios, se pudo desarrollar una programación más clara, estructurada y por funciones.

Los resultados obtenidos de esta prueba, se anexan en la tabla 3.5.

DATO	Dias	Especies	Esperado	MatLab	Arduino	Error MatLab	Error Arduino.
1	0.1	0.1	0.1	0.1072	0.1107376	0.0072	0.0107376
2	0.1	0.2	0.13	0.1327	0.1370791	0.0027	0.0070791
3	0.1	0.3	0.16	0.1607	0.1660031	0.0007	0.0060031
4	0.1	0.4	0.19	0.1903	0.1965799	0.0003	0.0065799
5	0.1	0.5	0.22	0.2208	0.2280864	0.0008	0.0080864
6	0.1	0.6	0.25	0.2516	0.2599028	0.0016	0.0099028
7	0.1	0.7	0.28	0.2822	0.2915126	0.0022	0.0115126
8	0.1	0.8	0.31	0.3124	0.3227092	0.0024	0.0127092
9	0.1	0.9	0.34	0.3423	0.3535959	0.0023	0.0135959
10	0.2	0.1	0.18	0.1757	0.1814981	-0.0043	0.0014981
11	0.2	0.2	0.21	0.2052	0.21197167	-0.0048	0.0019716
12	0.2	0.3	0.24	0.2359	0.2436847	-0.0041	0.0036847
13	0.2	0.4	0.27	0.267	0.275811	-0.003	0.005811
14	0.2	0.5	0.3	0.298	0.307834	-0.002	0.007834
15	0.2	0.6	0.33	0.3284	0.3392372	-0.0016	0.0092372
16	0.2	0.7	0.36	0.3583	0.3701239	-0.0017	0.0101239
17	0.2	0.8	0.39	0.3875	0.4002875	-0.0025	0.0102875
18	0.2	0.9	0.42	0.4164	0.4301412	-0.0036	0.0101412
19	0.3	0.1	0.25	0.2509	0.2591797	0.0009	0.0091797
20	0.3	0.2	0.28	0.2807	0.2899631	0.0007	0.0099631
21	0.3	0.3	0.31	0.3112	0.3214696	0.0012	0.0114696
22	0.3	0.4	0.34	0.3419	0.3531827	0.0019	0.0131827
23	0.3	0.5	0.37	0.3723	0.3845859	0.0023	0.0145859
24	0.3	0.6	0.4	0.4021	0.4153693	0.0021	0.0153693
25	0.3	0.7	0.43	0.4315	0.4457395	0.0015	0.0157395
26	0.3	0.8	0.46	0.4604	0.4755932	0.0004	0.0155932
27	0.3	0.9	0.49	0.4892	0.5053436	-0.0008	0.0155932
28	0.4	0.1	0.33	0.3278	0.3386174	-0.0022	0.0086174
29	0.4	0.2	0.36	0.3566	0.3683678	-0.0034	0.0083678
30	0.4	0.3	0.39	0.3863	0.3990479	-0.0037	0.0090479
31	0.4	0.4	0.42	0.4164	0.4301412	-0.0036	0.0101412
32	0.4	0.5	0.45	0.4465	0.4612345	-0.0035	0.0112345

DATO	Dias	Especies	Esperado	MatLab	Arduino	Error MatLab	Error Arduino.
33	0.4	0.6	0.48	0.4764	0.4921212	-0.0036	0.0121212
34	0.4	0.7	0.51	0.506	0.522698	-0.004	0.012698
35	0.4	0.8	0.54	0.5353	0.5529649	-0.0047	0.0129649
36	0.4	0.9	0.57	0.5645	0.5831285	-0.0055	0.0131285
37	0.5	0.1	0.4	0.4056	0.4189848	0.0056	0.0189848
38	0.5	0.2	0.43	0.4334	0.4477022	0.0034	0.0177022
39	0.5	0.3	0.46	0.4626	0.4778658	0.0026	0.0178658
40	0.5	0.4	0.49	0.4926	0.5088558	0.0026	0.0188558
41	0.5	0.5	0.52	0.5229	0.5401557	0.0029	0.0201557
42	0.5	0.6	0.55	0.5533	0.5715589	0.0033	0.0215589
43	0.5	0.7	0.58	0.5835	00.6027555	0.0035	0.0227555
44	0.5	0.8	0.61	0.6134	0.6336422	0.0034	0.0236422
45	0.5	0.9	0.64	0.6429	0.6641157	0.0029	0.0241157
46	0.6	0.1	0.48	0.4829	0.4988357	0.0029	0.0188357
47	0.6	0.2	0.51	0.5102	0.5270366	0.0002	0.0170366
48	0.6	0.3	0.54	0.5392	0.5569936	-0.0008	0.0169936
49	0.6	0.4	0.57	0.5694	0.5881902	-0.0006	0.0181902
50	0.6	0.5	0.6	0.6002	0.6200066	0.0002	0.0200066
51	0.6	0.6	0.63	0.6311	0.6519263	0.0011	0.0219263
52	0.6	0.7	0.66	0.6617	0.6835361	0.0017	0.0235361
53	0.6	0.8	0.69	0.6916	0.7144228	0.0016	0.0244228
54	0.6	0.9	0.72	0.7205	0.7442765	0.0005	0.0242765
55	0.7	0.1	0.55	0.5567	0.5750711	0.0067	0.0250711
56	0.7	0.2	0.58	0.5838	0.6030654	0.0038	0.0230654
57	0.7	0.3	0.61	0.6129	0.6331257	0.0029	0.0231257
58	0.7	0.4	0.64	0.6433	0.6645289	0.0033	0.0245289
59	0.7	0.5	0.67	0.6743	0.6965519	0.0043	0.0265519
60	0.7	0.6	0.7	0.7052	0.7284716	0.0052	0.0284716
61	0.7	0.7	0.73	0.7354	0.7596682	0.0054	0.0296682
62	0.7	0.8	0.76	0.7643	0.7895219	0.0043	0.0295219
63	0.7	0.9	0.79	0.7915	0.8176195	0.0015	0.0276195
64	0.8	0.1	0.63	0.6235	0.6440755	-0.0065	0.0140755
65	0.8	0.2	0.66	0.6506	0.6720698	-0.0094	0.0120698
66	0.8	0.3	0.69	0.6796	0.7020268	-0.0104	0.0120268
67	0.8	0.4	0.72	0.7099	0.7333267	-0.0101	0.0133267
68	0.8	0.5	0.75	0.7405	0.7649365	-0.0095	0.0149365
69	0.8	0.6	0.78	0.7706	0.7960298	-0.0094	0.0160298
70	0.8	0.7	0.81	0.7994	0.8257802	-0.0106	0.0157802
71	0.8	0.8	0.84	0.8261	0.8533613	-0.0139	0.0133613
72	0.8	0.9	0.87	0.8504	0.8784632	-0.0196	0.0084632

DATO	Dias	Especies	Esperado	MatLab	Arduino	Error MatLab	Error Arduino.
73	0.9	0.1	0.67	0.6808	0.7032664	0.0108	0.0332664
74	0.9	0.2	0.7	0.7077	0.7310541	0.0077	0.0310541
75	0.9	0.3	0.73	0.7365	0.7608045	0.0065	0.0308045
76	0.9	0.4	0.76	0.7661	0.7913813	0.0061	0.0313813
77	0.9	0.5	0.79	0.7956	0.8218548	0.0056	0.0318548
78	0.9	0.6	0.82	0.8241	0.8512953	0.0041	0.0312953
79	0.9	0.7	0.85	0.8505	0.8785665	0.0005	0.0285665
80	0.9	0.8	0.87	0.8744	0.0.9032552	0.0044	0.0332552
81	0.9	0.9	0.9	0.8952	0.0.9247416	-0.0048	0.0247416

Tab. 3.5: Comprobación de Datos

Como se puede observar en la tabla anterior, los resultados obtenidos con el procesamiento de datos del microcontrolador son muy cercanos a los valores esperados ideales presentados en la tabla. Se puede observar que el error que se presenta es mínimo. En la tabla, se encuentran resaltadas con color amarillo los datos que no fueron incluídos en la red al momento del entrenamiento, en los cuales se puede observar que la red se encuentra bien entrenada.

Por último, se requiere volver a convertir esos datos, representados en valores de 0,1 a 0,9 en su valor representativo de Kgs. La ecuación que describe el valor de kilogramos a entregar, la cual se incluye en la programación del microcontrolador, es la siguiente:

$$Kgs.\ de\ Salida = (x * 125) - 2{,}5$$

Donde la x es el valor de salida, ya sea de Arduino o de Matlab.

Al momento de realizar la conversión de estos datos a kilogramos, se observa que el error má grande que se presenta es de 1,8 Kg, lo cual es un error menor al 5 %, lo cual es un error bastante pequeño, y más aún si se compara con la cantidad de alimento que se desperdicia en cada dosificación actualmente.

El paso siguiente fue desarrollar una aplicación móvil, esto con el fin de enviar y recibir todos los datos mediante este sistema. El desarrollo de este sistema, se observa en el siguiente apartado.

Desarrollo de Aplicación Móvil

El desarrollo de la aplicación móvil se realizó con el uso del software Ionic Frameworks. El sistema que se desarrolló consta de dos pantallas, las cuales se describirán más adelante, mediante las cuales se realiza, primeramente la búsqueda y conexión del dispositivo bluetooth del microcontrolador con el dispositivo móvil. Esta pantalla se observa en la figura 3.14.

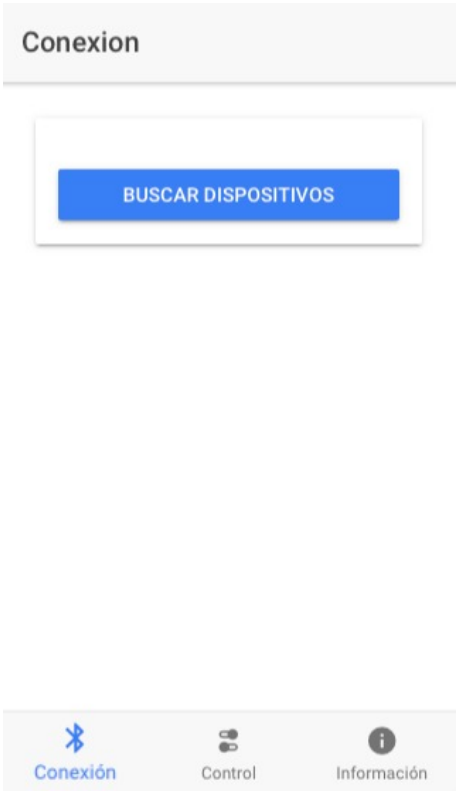


Fig. 3.14: Conexión Bluetooth mediante Aplicación Móvil

Una vez conectado, el dispositivo entra en una pantalla donde se puede realizar el envío de datos hacia el microcontrolador. En esta pantalla se tiene un par de barras, mediante las cuales se pueden seleccionar los parámetros de entrada a la red, los cuales son, la cantidad de días que tiene la especie cultivada y los camarones por m^2 que se tienen.

Además, la pantalla muestra en su parte inferior, la cantidad de alimento, expresada en Kgs. que se dosificarán. Esta pantalla se muestra en la figura 3.15.

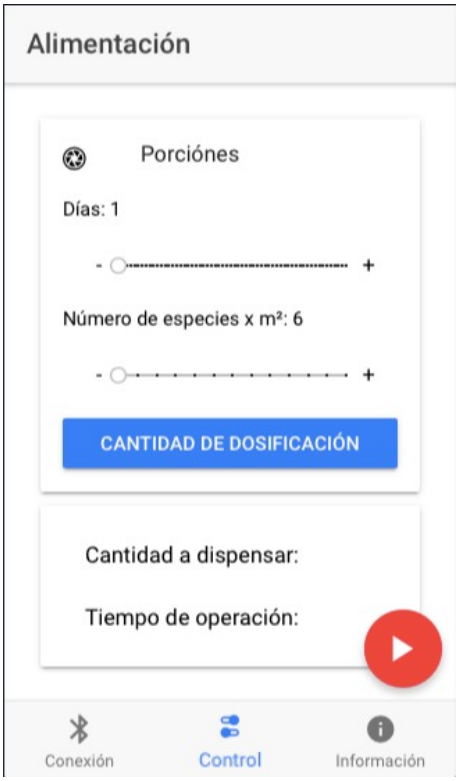


Fig. 3.15: Dosificación mediante Aplicación Móvil

Cabe resaltar que esta aplicación se generó como apoyo para ver el funcionamiento del sistema basado en redes, donde el microcontrolador recibia los datos desde esta plataforma, y a su vez regresaba el valor de la cantidad de alimento a dosificar, donde se comprobó el funcionamiento de la red.

La programación de la aplicación móvil se anexa en la sección de Apéndices. Con esta parte, se comprueba que el funcionamiento del sistema automático es funcional.

4. CONCLUSIONES Y RECOMENDACIONES

Con el desarrollo de este trabajo de tesis fue posible cumplir con los objetivos propuestos, los cuales se referían al diseño e implementación de un dispositivo para la dosificación de alimento para camarón, los cuales están cultivados en estanques artificiales exteriores. A lo largo de este trabajo se fueron desarrollando, de manera secuencial, los procedimientos necesarios para cumplir con los requerimientos principales del sistema. En este trabajo fue posible integrar distintos softwares tanto de diseño como de programación con la finalidad de cumplir con la función mencionada anteriormente.

También se comprobó la factibilidad del desarrollo de este tipo de dispositivos en un área que tiene mucho potencial a explotar en la región, como lo es la acuacultura. Como se mencionó con anterioridad, los costes de producción aumentan de manera significativa debido al gran desperdicio de alimento, por lo que, al realizar las pruebas con el sistema creado, pese a que tiene aún un error en la cantidad final generada por los métodos y recursos utilizados en este prototipo, se puede observar que la cantidad de alimento que se evitará desperdiciar es bastante significativa.

Se comprobó también que el uso de métodos basados en inteligencia artificial e incluso otro tipo de técnicas relacionadas con el control inteligente pueden ser de utilidad tanto en esta área como en otras, solamente es cuestión de encontrarles una aplicación concreta para que puedan ser tomadas como base para su integración en distintos proyectos de ingeniería y control.

El sistema mencionado cumple con los objetivos definidos inicialmente, y a lo largo de su desarrollo se fueron encontrando posibles mejoras que, en este caso, debido a que el sistema se fue desarrollando como prototipo y debido al gasto que estos cambios generarían no fueron aplicados en el primer sistema, pero de continuar con esta investigación, serían puntos importantes a retomar. Estas posibles mejoras son las siguientes:

- Se puede agregar un motor a pasos o un motor de $12\,VCD$ en el sistema, con el cual se podrá decidir hacia que lado del camión arrojar el alimento. En esta tésis, en el diseño mecánica ya viene incluida esa herramienta, pero en la programación no fue tomada en cuenta, por lo que esa válvula es activada de manera manual.

- Se puede agregar en la parte superior del sistema un sensor para conocer la velocidad y dirección del aire, esto con la finalidad de poder controlar el ángulo de disparo del cañon de salida de alimento, debido a que, de momento, la altura se controla de manera manual y se debe garantizar siempre que el alimento llegue hasta la parte central del estanque.

- Para poder lograr un sistema de control a lazo cerrado, se puede sustituir el método de cuantificación de alimento a dosificar. Se puede aplicar, en vez de la VDR un sistema parecido al que tienen las tolvas báscula, las cuales darán exactamente la cantidad de alimento que se está dosificando para de esta manera hacer el sistema un poco más preciso, logrando que el sistema funcione en razón del alimento dosificado y sin necesidad de utilizar un sensor que te indique el número de vueltas.

- Se puede agregar la opción de controlar la velocidad del motor de salida(VDR), o controlar el tiempo en el cual la dosificación podría llevarse a cabo, esto debido a que no todos los estanques tienen las mismas dimensiones, por lo que esta función podría retardar o acelerar un poco la dosificación del alimento, garantizando la dosificación ecuánime a todo el estanque.

- De ser necesario, podría elegirse un controlador un poco más potente que el Arduino para basar el funcionamiento del sistema. En este prototipo se observó que el arduino es una buena opción, pero para que el sistema se acerque un poco más, es necesario tomar en cuenta otros parámetros distintos (de menor importancia), lo cual podría saturar el procesador de este controlador.

- Se recomienda generar una base de datos, donde el Biólogo podrá cargar los datos de cada estanque, además, crear un sistema para entrar a esta plataforma, ya sea como operador o como Biólogo. De esta manera, el operador tendría que, solamente colocar en la aplicación,

el estanque en el cual se encuentra y los datos serían enviados al controlador de manera automática.

En conclusión se puede decir que el sistema es funcional y que soluciona en gran parte la problemática existente en nuestra localidad, y que puede servir como base para seguir desarrollando tecnología enfocada a esta actividad económica en crecimiento.

Referencias

Appana, D. K., Alam, M. W., y Basnet, B. (2016). A novel design of feeder system for aqua culture suitable for shrimp farming. *International Journal of Hybrid Information Technology*, *9*(4), 199–212.

Auruz, R. d. L. (1998). *Sistema experto de clasificación y apoyo al diseño arquitectónico de edificios inteligentes* (thesis). UNAM.

Avilés, A. (2000). Cultivo de peces marinos. cap. xv. *Álvarez-Torres, M. Ramírez-Flores, LM Torres-Rodríguez y A. Díaz de León-Corral (Eds.). Estado de Salud de la Acuacultura*.

Bishop, R. H., y cols. (2002). *The mechatronics handbook* (Vol. 1). CRC press Boca Raton.

Bourcier, D. (2003). *Inteligencia artificial y derecho*. Editorial UOC.

Cantor Forero, J. S., y cols. (2015). *Diseño y simulacion de un robot hexapodo de una sola juntura por pierna* (B.S. thesis). Universidad Militar Nueva Granada.

FAO, U. (2014). Faostat. *Retrieved Feb*, *2014*.

Gutiérrez, N. (2006). *Propuestas de aplicaciones de sistemas expertos en empresas mexicanas* (Tesis Doctoral no publicada). tesis de pregrado). Universidad Nacional Autónoma de México, México Distrito Federal, México.

Haces, J., y Hernández, B. (2011). *Sistema para generar micromundos para la asignatura de álgebra* (Tesis Doctoral no publicada). tesis de pregrado). Universidad Nacional Autónoma de México, México Distrito Federal, México.

Harrison, J., Dobson, P., y Hoar, D. (2011, agosto 9). *Offshore aquaculture system.* Google Patents. (US Patent 7,992,522)

Hasan, M. R., Hecht, T., De Silva, S. S., Tacon, A. G., y cols. (2007). *Study and analysis of feeds and fertilizers for sustainable aquaculture development*. Food and Agriculture Organization of the United Nations Rome.

Hasan, M. R., y New, M. B. (2013). *On-farm feeding and feed management in aquaculture*. Food and Agriculture Organization of the United Nations Rome.

Hemeda, B., MM, A.-R., y Abdel-Mottaleb, A. (s.f.). Development and evaluation of a mounted type machine for fish feeds distribution by blowing air system.

Huerta, J. (2009). *Diagramación de argumentos dialógicos y derrotantes en el sistema inteligente "expertius"* (Tesis Doctoral no publicada). tesis de pregrado). Universidad Nacional Autónoma de México, México Distrito Federal, México.

Juárez-Palacios, R. (1987). La acuicultura en méxico, importancia social y económica. *Desarrollo Pesquero Mexicano 1986-1987*, 219–232.

Kumar, G., y Engle, C. R. (2016). Technological advances that led to growth of shrimp, salmon, and tilapia farming. *Reviews in Fisheries Science & Aquaculture*, *24*(2), 136–152.

Manzo, D. (2000). Efecto de cuatro densidades de siembra sobre el crecimiento del camaron blanco litopenaeus vannamei (boone, 1931) cultivado en estanques rústicos. *Manzanillo, Colima. Tesis de Maestria en Ciencias. Facultad de Ciencias Marinas. Universidad de Colima. México*.

Mathiesen, Á. M. (2015). *The state of world fisheries and aquaculture 2012.* Food and Agriculture Organization of the United Nations.

Matich, D. J. (2001). Redes neuronales: Conceptos básicos y aplicaciones. *Universidad Tecnológica Nacional, México*.

Mega, A. (2011). 2560. *http://www. projetoarduino. com. br/arduinomega-2560-p19¿. Acesso em*, *20*, 05–14.

Melberg, R., y Torgersen, T. (2011, mayo 5). *System and method for controlling feeding of farmed fish.* Google Patents. (US Patent App. 13/698,627)

Molaug, O., Kluge, G., y Havrevold, S. (1982, noviembre 16). *Food dispensing system.* Google Patents. (US Patent 4,359,014)

Newport, J., y Jawahar, G. (1995). Brackishwater shrimp culture: Impact on eco-environment and socio-economic aspects of rural fisherfolk. *Fishing chimes*(5), 15–16.

Rumelhart, D. E., y McClelland, J. L. (1986). Parallel distributed processing: explorations in the microstructure of cognition. volume 1. foundations.

SAGARPA, S. (2013). Servicio de información agroalimentaria y pesquera. *Base de datos en línea. Consultado en septiembre del*.

Vázquez, A. (2009). *Sistema experto para la interpretación mamográfica.* UNAM.

William, B. (2001). *Mechatronics, electronic control systems in mechanical and electrical engineering.* Addison Wesley Longman.

Zampayo, C. (2004). *Técnicas y métodos para la representación y manipulación del conocimiento en ciencias computacionales* (thesis).

Apéndice 1. Código de Arduino para Captura manual de Datos

```
//********* Librerias *********************
#include <SPI.h>  // SD
#include <SD.h>   // SD
#include <SoftwareSerial.h>  // IMPRESORA
#include <Keypad.h> // TECLADO
#include <LiquidCrystal.h>  // PANTALLA
#include <Wire.h>
#define DS3231_I2C_ADDRESS 104
//******** INICIALIZACION **********
SoftwareSerial Thermal(2, 3);
int heatTime = 80;
int heatInterval = 255;
char printDensity = 15;
char printBreakTime = 15;
LiquidCrystal lcd(11, 10, 7, 6, 5, 4); // Pantalla LCD
const int chipSelect = 53;  // SD
File myFile;  // ARCHIVO SD
File diario;
File nombre;
File kgtolva;
const byte Filas = 4;         // TECLADO
const byte Cols = 4;    // TECLADO
char Teclas [ Filas ][ Cols ] =
{
{'1', '2', '3', 'A'},
{'4', '5', '6', 'B'},
```

```
{'7', '8', '9', 'C'},
{'*', '0', '#', 'D'}
};
byte Pins_Filas[] = {47, 49, 45, 43};     // PINS FILAS TECLADO
byte Pins_Cols[] = {35, 37  , 39, 41 };     // PINS COLUMNAS TECLADO
Keypad Teclado1 = Keypad(makeKeymap(Teclas), Pins_Filas, Pins_Cols, Filas, Cols);
// MAPEO DE FILAS Y COLUMNAS DE TECLADO
//**********************  Variables *********
int zero = 0;
int i = 0;
int j = 0;
//int k = 0;
int flag = 0;
int est = 0;
float kilo = 0;
int  seconds, minutes, hours, day, date, month, year;
String weekDay;
int enteros[2][10] = {{48, 49, 50, 51, 52, 53, 54, 55, 56, 57},
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}};  // CONVERSION DE ASKI A 0-9
int numero; // VALORES ALFANUMERICOS
int customKey;  // VARIABLE PARA RECIBIR DATOS DEL TECLADO
char estanque[2];             // ARREGLO NO. DE ESTANQUE
char kg[3];                   // ARREGLO KGS. DE ALIMENTO
char tiempo[3];
const int entrada = 29;
int estado;
int valido=0;
int lcda=26;
```

```
int bandera = 0;

int bandera2 = 0;

int recorrido = 0;

/*char tolva[4];

int tolv;

int mil;

int cien;

int diez;

int uno;

int res;

*/

String fechas;

int i2;

int sumatoria=0;

int rpm= 64; // Número de Vueltas por Minuto del Motor

float ttt =0; // tiempo de Recorrido

//__________ SENSOR ___

volatile int cont1 = 0;   // Variable entera que se almacena en la RAM del Micro

long T0 = 0 ;

//____________ MOTOR __________

int M1=27; // MOTOR

float kres=0;

float vueltas;

float camara= 0.250; // Kg. por camara

float tmotor; // tiempo de Encendido del motor

float t1; // Tiempo de RELAY ENCENDIDO POR RECORRIDO

float t2; // Tiempo de RELAY APAGADO POR RECORRIDO

float t3; // Tiempo motor apagado por ciclo
```

```
//_____________ CONDICIONES ___________________

float tfijo;

float tvuelta;

//*******************    SETUP *********************************

void setup() {

pinMode(M1, OUTPUT);

digitalWrite(M1, LOW);

pinMode(lcda, OUTPUT);

attachInterrupt(4,interrupcion0,FALLING);  // Interrupcion 4 (pin19)

// LOW, CHANGE, RISING, FALLING

digitalWrite(lcda, HIGH);

Wire.begin();

//setDate (2, 44, 10, 2, 28, 2, 17);

lcd.begin(20, 4) ;

Serial1.begin(19200); // SD

Serial2.begin(57600);

Thermal.begin(19200);

Serial.begin(9600); // SENSOR

getDate();

fechas += date;

fechas += month;

fechas += year;

fechas += ".txt";

initPrinter();

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

lcd.print("Iniciando SD");
```

```
if (!SD.begin(53)) {
lcd.println("Error SD");
delay(1000);
return;
}
delay(500);
lcd.clear();
lcd.home();
pinMode(entrada, INPUT);
lcd.clear();
myFile = SD.open(fechas, FILE_WRITE);
if (myFile)
{
lcd.clear();
lcd.print("iniciando memoriaa");
delay(1000);
lcd.clear();
}
else
{
lcd.clear();
lcd.print("no hay sd");
delay(1000);
}
/*EEPROM.write(0,0);
EEPROM.write(1,0);
EEPROM.write(2,0);
EEPROM.write(3,0);*/
```

```
// ***** aqui leemos los kilos en la tolva ***
digitalWrite(lcda, HIGH);
}
//***************** FUNCION LEER RELOJ
void getDate()
{
// Send request to DS3231 to receive the date
Wire.beginTransmission(DS3231_I2C_ADDRESS);
Wire.write(0x00);  // start at register 00h (seconds)
Wire.endTransmission();
Wire.requestFrom(DS3231_I2C_ADDRESS, 7);   // Request 7 bytes (00h - 006h)
while (Wire.available())
{
seconds = Wire.read();
minutes = Wire.read();
hours = Wire.read();
day = Wire.read();
date = Wire.read();
month = Wire.read();
year = Wire.read();
// convert BCD to decimal
seconds = (((seconds >> 4) * 10) + (seconds & B00001111));
minutes = (((minutes >> 4) * 10) + (minutes & B00001111));
hours = ((((hours & B00110000) >> 4) * 10) + (hours & B00001111));
day = (day & B00000111);
month = ((((month & B00010000) >> 4) * 10) + (month & B00001111));
date = ((((date & B00110000) >> 4) * 10) + (date & B00001111));
year = (((year >> 4) * 10) + (year & B00001111));
```

```
}
switch (day)  {
case 1:
weekDay = "Sun";
break;
case 2:
weekDay = "Mon";
break;
case 3:
weekDay = "Tue";
break;
case 4:
weekDay = "Wed";
break;
case 5:
weekDay = "Thu";
break;
case 6:
weekDay = "Fri";
break;
default: break;
}
}
// FUNCION CONFIGURAR RELOJ
void setDate(byte seconds, byte minutes, byte hours, byte day, byte date,
byte month, byte year)
{
// convert DEC to BCD
```

```
seconds = (seconds % 10) + ((seconds / 10) << 4);
minutes = (minutes % 10) + ((minutes / 10) << 4);
hours = (hours % 10) + ((hours / 10) << 4);
date = (date % 10) + ((date / 10) << 4);
month = (month % 10) + ((month / 10) << 4);
year = (year % 10) + ((year / 10) << 4);
Wire.beginTransmission(DS3231_I2C_ADDRESS);
Wire.write(0x00);
Wire.write(seconds);
Wire.write(minutes);
Wire.write(hours);
Wire.write(day);
Wire.write(date);
Wire.write(month);
Wire.write(year);
Wire.endTransmission();
}
void initPrinter()
{
Thermal.write(27);
Thermal.write(55);
Thermal.write(7); //Default 64 dots = 8*('7'+1)
Thermal.write(heatTime); //Default 80 or 800us
Thermal.write(heatInterval); //Default 2 or 20us
//Modify the print density and timeout
Thermal.write(18);
Thermal.write(35);
int printSetting = (printDensity << 4) | printBreakTime;
```

```
Thermal.write(printSetting); //Combination of printDensity and printBreakTime
Serial2.println();
Serial2.println("Printer ready");
}
int imprimir() {
do {
Thermal.write(27); //COMANDO NEGRITAS
Thermal.write(14); //COMANDO NEGRITAS
Thermal.write(27); // COMANDO JUSTIFICACION HACIA IZQUIERDA
Thermal.write(97); // COMANDO JUSTIFICACION HACIA IZQUIERDA
Thermal.write(1);// COMANDO JUSTIFICACION HACIA IZQUIERDA
Thermal.println("ARY");
Thermal.println("Agroindustrial");
Thermal.println("S.A. de C.V.");
Thermal.write(27); // QUITAR NEGRITAS
Thermal.write(20); // QUITAR NEGRITAS
Thermal.println("Granga El Quinto III");
Thermal.write(10); // ESPACIO
Thermal.write(10); // ESPACIO
j++;
} while (j < 1);
}
void intercambia() {
for (int j = 0; j <= 9; j++) {
if (enteros[0][j] == customKey) numero = enteros[1][j];
}
}
//******************** FUNCION DATOS
```

```
void datos() {
valido=0;
i = 0;
est = 0;
lcd.print("# de estanque:");
lcd.setCursor(7, 1);
while (i <= 1)  {
customKey = Teclado1.getKey();
if (customKey != 0)
{
if(customKey==66)
{
lcd.clear();
i=2;
valido=1;
continue;
}
else{
intercambia();
estanque[i] = numero;
lcd.print(numero);
delay(500);
i++;
}
}
}
est = ((estanque[0] * 10) + (estanque[1]));
kilo = 0;
```

```
i = 0;
lcd.clear();
lcd.home();
// limpia();
if(valido==0)
{
lcd.print("Kg a descargar:");
lcd.setCursor(6, 1);
while (i <= 2) {
customKey = Teclado1.getKey(); //se guarda en la variable customKey el
caracter de la tecla presionada
if (customKey != 0)         //se evalúa si se presionó una tecla
{
if(customKey==65)
{
lcd.clear();
i2=i;
for(i;i<=2;i++)
{
kg[i] = 0;
}
if(i2==1){
kg[2]=kg[0];
kg[0] = 0;
kg[1] = 0;

}
if(i2==2)
```

```
{
kg[2]=kg[1];
kg[1]=kg[0];
kg[0] = 0;
}
}
if(customKey==66)
{
lcd.clear();
valido=1;
i=4;
continue;
}
else{
intercambia();
lcd.print(numero);
/*   if (i == 3) {
kg[i - 1] = numero;
} */
kg[i] = numero;
i++;
}
}
}
kilo = ((kg[0] * 100) + (kg[1]*10) + kg[3]);
if(kilo==0){
lcd.clear();
lcd.home();
```

```
lcd.print("Numero invalido...");
lcd.setCursor(0,2);
lcd.print("Intentelo nuevamente");
delay(1000);
}
lcd.clear();
lcd.home();
i = 0;
lcd.print("t. alimentarcion:");
lcd.setCursor(6, 1);
while (i <= 2)  {
customKey = Teclado1.getKey();
if (customKey != 0)
{
if(customKey==65)
{
lcd.clear();
i2=i;
for(i;i<=2;i++)
{
tiempo[i] = 0;
}

if(i2==1){
tiempo[2]=tiempo[0];
tiempo[0] = 0;
tiempo[1] = 0;
}
```

```
if(i2==2)
{
tiempo[2]=tiempo[1];
tiempo[1]=tiempo[0];
tiempo[0] = 0;
}
}
if(customKey==66)
{
lcd.clear();
i=2;
valido=1;
continue;
}
else{
intercambia();
tiempo[i] = numero;
lcd.print(numero);
//Serial.println(numero);
i++;
}
}
}
ttt=(tiempo[0]*100)+(tiempo[1]*10) +tiempo[2];
lcd.clear();
lcd.home();
}
}
```

```
//******* FUNCION CONFIRMAR ******************
void confirmar() {
if(valido==0)
{
lcd.clear();
lcd.home();
lcd.setCursor(0, 0);
lcd.print("Est:  ");
lcd.print(est);
lcd.setCursor(10, 0);
lcd.print("KGs:");
lcd.print(kilo);
lcd.setCursor(3, 1);
lcd.print("Tiempo: ");
lcd.print(ttt);
lcd.setCursor(0, 2);
lcd.print(" Presiona 'A' para ");
lcd.setCursor(5, 3);
lcd.print("Continuar");
do {
customKey = Teclado1.getKey();
if (customKey != 0)
{
if (customKey == 65) flag = 1;
}
}
while (flag != 1);
flag = 0;
```

```
lcd.clear();
lcd.home();
}
else
{
lcd.clear();
lcd.home();
}
}
// ___________________ SENSOR____________________________
void interrupcion0()    // Funcion que se ejecuta durante cada interrupion{
if ( millis() > T0  + 200)
{
cont1++;     // Se incrementa en uno el contador
T0 = millis();
}
Serial.println(cont1);
}
//***** FUNCION DESPACHANDO *************
void despachar() {
if(valido==0)
{
tfijo= 60.0/rpm;
tvuelta= ttt/kilo;
vueltas=kilo/(camara*4);
kres=kilo;
t1=tfijo* kilo;
t2= ttt - t1;
```

```
t3= (t2 / kilo) * 1000;
tmotor= t3 + tfijo;
if (tvuelta >= tfijo) {
cont1=0;
// digitalWrite(M1,HIGH);
while(kres > 0){
digitalWrite(M1,HIGH);
lcd.setCursor(3, 2);
lcd.print("DESPACHANDO");
while (cont1 < 4){
}
delay(100);
interrupcion0();
digitalWrite(M1,LOW);
delay(t3);
if(cont1=4){
kres--;
cont1=0;
Serial.println(kres);
}
}
}
else{
lcd.clear();
lcd.home();
lcd.setCursor(2, 0);
lcd.print("NO ES POSIBLE");
lcd.setCursor(3, 1);
```

```
lcd.print("ALIMENTAR EN");
lcd.setCursor(2, 2);
lcd.print("TAN POCO TIEMPO");
}
delay(1000);
digitalWrite(M1, LOW);
lcd.clear();
lcd.setCursor(7, 2);
lcd.print("LISTO");
delay(3000);
}
else
{
lcd.clear();
lcd.home();
}
}
//** FUNCION GUARDAR EN SD *****************
void guardar() {
if(valido==0)
{
myFile = SD.open(fechas, FILE_WRITE);
if (myFile) {
lcd.clear();
delay(1000);
myFile.print(".:  ");
myFile.print("ESTANQUE[");
myFile.print(est);
```

```
myFile.print("]: ");
myFile.println(   kilo   );
myFile.close();
} else {
lcd.clear();
lcd.print("no hay sd");
delay(1000);
}
myFile = SD.open("primer.txt", FILE_WRITE);
if (myFile) {
myFile.print(".:  ");
myFile.print("ESTANQUE[");
myFile.print(est);
myFile.print("]: ");
myFile.println(   kilo   );
myFile.close();
}
}
else
{
lcd.clear();
lcd.home();
}
}
void loop()
{
estado = digitalRead(entrada);
delay(100);
```

```
//*****************************************************
while (estado == HIGH)
{
if (bandera == 0) {
recorrido++;
myFile = SD.open("primer.txt", FILE_WRITE);
if (myFile)
{
getDate();
myFile.print("Recorrido [");
myFile.print(recorrido);
myFile.print("]   ");
myFile.print(date);
myFile.print(" / ");
myFile.print(month);
myFile.print(" / ");
myFile.println(year);
myFile.println("   ");
myFile.print("hora inicial:   ");
myFile.print(hours);
myFile.print(":");
myFile.print(minutes);
myFile.print(":");
myFile.println(seconds);
myFile.close();
bandera++;
}
diario = SD.open(fechas, FILE_WRITE);
```

```
if (diario)
{
lcd.clear();
lcd.print("guardando...");
delay(500);
getDate();
diario.print("Recorrido [");
diario.print(recorrido);
diario.print("]   ");
diario.print(date);
diario.print(" / ");
diario.print(month);
diario.print(" / ");
diario.println(year);
diario.println("   ");
diario.print("hora inicial:   ");
diario.print(hours);
diario.print(":");
diario.print(minutes);
diario.print(":");
diario.println(seconds);
diario.close();
bandera++;
}
else {
// if the file didn't open, print an error:
Serial.println("error opening test.txt");
}
```

```
}
lcd.display();
digitalWrite(lcda, HIGH);
lcd.clear();
datos();
confirmar();
despachar();
guardar();
bandera2 = 0;
estado = digitalRead(entrada);
valido=0;
}
//*****************************************************************************
while (estado == LOW)
{
estado = digitalRead(entrada);
if (bandera2 == 0) {
if(valido==0){
myFile = SD.open("primer.txt", FILE_WRITE);
if (myFile)
{
getDate();
myFile.print("hora Final:   ");
myFile.print(hours);
myFile.print(":");
myFile.print(minutes);
myFile.print(":");
myFile.println(seconds);
```

```
myFile.close();
}
myFile = SD.open("primer.txt");
nombre = SD.open(fechas, FILE_WRITE);
if (nombre)
{
getDate();
nombre.print("hora Final:   ");
nombre.print(hours);
nombre.print(":");
nombre.print(minutes);
nombre.print(":");
nombre.println(seconds);
nombre.close();
}
else {
// if the file didn't open, print an error:
Serial1.println("error opening test.txt");
}
if (myFile) {
Serial1.println("primer.txt:");
Thermal.write(27); //COMANDO NEGRITAS
Thermal.write(14); //COMANDO NEGRITAS
Thermal.write(27); // COMANDO JUSTIFICACION HACIA IZQUIERDA
Thermal.write(97); // COMANDO JUSTIFICACION HACIA IZQUIERDA
Thermal.write(1);// COMANDO JUSTIFICACION HACIA IZQUIERDA
Thermal.println("ARY");
Thermal.println("Agroindustrial");
```

```
Thermal.println("S.A. de C.V.");
Thermal.write(27); // QUITAR NEGRITAS
Thermal.write(20); // QUITAR NEGRITAS
Thermal.println("Granja El Quinto III");
Thermal.write(10); // ESPACIO
Thermal.write(10); // ESPACIO
while (myFile.available()) {
Thermal.write(myFile.read());
}
myFile.close();
Thermal.write(10); // ESPACIO
Thermal.write(10); // ESPACIO
Thermal.write(10); // ESPACIO
Thermal.write(10); // ESPACIO
}
else {
// if the file didn't open, print an error:
Serial1.println("error opening test.txt");
}
bandera2++;
estado = digitalRead(entrada);
}
//*********
SD.remove("primer.txt");
}
digitalWrite(lcda, LOW);
lcd.noDisplay();
bandera = 0;
```

```
estado = digitalRead(entrada);
}
}
```

Apéndice 2. Código de Rstudio para Entrenamiento Neuronal Artificial

```
sigmoide <- function(valor)
{
if(class(valor) == "matrix")
{
resultado <- matrix(0, nrow = nrow(valor), ncol = ncol(valor))
for(i in 1:nrow(valor))
{
for(j in 1:ncol(valor))
{
resultado[i,j] = (1 / (1 + exp(-1 * valor[i, j])))
}
}
return(resultado)
}
else
{
return(1 / (1 + exp(-1 * valor)))
}
}
d_sigmoide <- function(valor)
{
if(class(valor) == "matrix")
{
```

```
resultado <- matrix(0, nrow = nrow(valor), ncol = ncol(valor))
for(i in 1:nrow(valor))
{
for(j in 1:ncol(valor))
{
resultado[i,j] = valor[i, j] * (1 - valor[i, j])
}
}
return(resultado)
}
else
{
return(valor * (1 - valor))
}
}
entrenamiento <- function(entradas, salidasdes, estructurared, eta,
maxerror, maxepocas, bias)
{
# genera espacios de memoria para el manejo de los pesos y las neuronas
nocapas = nrow(estructurared)
capas <- list();
salidascapas <- list()
deltacapa <- list()
gradcapa <- matrix(0, nrow = maxepocas, ncol = 1)
errores <- matrix(0, nrow = maxepocas, ncol = 1)
xconbias <- matrix(bias, nrow = nrow(entradas), ncol = (ncol(entradas) + 1))
for(i in 1:nrow(entradas))
{
```

```
for(j in 1:ncol(entradas))
{
xconbias[i, j + 1] = entradas[i, j]
}
}
# la estructura de la red es una matrix donde los renglones son el numero
de capas y la columna
# indica el numero de entradas con el bias, para generar las matrices de pesos
colant = ncol(xconbias)
for(i in 1:nocapas)
{
capas[[i]] <- matrix(runif(estructurared[i, 1] * colant, 0.01, .99),
nrow = estructurared[i, 1], ncol = colant)
colant = estructurared[i, 1] + 1
salidascapas[[i]] <- matrix(0, nrow = colant, ncol = 1)
}
nocapas = nocapas + 1
capas[[nocapas]] <- matrix(runif(ncol(salidasdes) * colant, 0.01, .99),
nrow = ncol(salidasdes), ncol = colant)
salidascapas[[nocapas]] <- matrix(0, nrow = ncol(salidasdes), ncol = 1)
#comienza for de epocas
epoca = 1
lasterror = 99
while(epoca < maxepocas && abs(lasterror) > maxerror)
{
# if(epoca == 1 || (epoca %% 1000 ) == 0)
{
cat("Epoca: ")
```

```
cat("\t")
cat(epoca)
cat("\t")
cat(lasterror)
cat("\n")
}
#comienza for de numero de datos
sumerror = 0
for(dato in 1:nrow(xconbias))
{
#parte del feedforward
#calculo primera capa que son las entradas
xtrab = as.matrix(xconbias[dato, 1:ncol(xconbias)])
temp = sigmoide(capas[[1]] %*% xtrab)
salidascapas[[1]] = matrix(c(bias, temp), nrow = (nrow(temp) + 1), ncol = 1)
# Calcula las capas ocultas
if(nocapas > 2 )
{
for(capa in 2:(nocapas - 1))
{
temp = sigmoide(capas[[capa]] %*% salidascapas[[capa - 1]])
salidascapas[[capa]] = matrix(c(bias, temp), nrow = (nrow(temp) + 1), ncol = 1)
}
}
# Calcula la capa de salida
temp = sigmoide(capas[[nocapas]] %*% salidascapas[[nocapas - 1]])
salidascapas[[nocapas]] = matrix(c(temp), nrow = (nrow(temp)), ncol = 1)
# if(epoca == 1 || (epoca %% 1000 ) == 0)
```

```
# {
#   cat(salidascapas[[nocapas]])
#   cat("\n\t")
# }
# calcula el error de la salida vs lo deseado
err = as.matrix(salidasdes[dato, 1:ncol(salidasdes)] - salidascapas[[nocapas]])
# acumulaciÃ³n del error cuadratico para despues sacar el error cuadratico medio
sumerror = sumerror + (t(err) %*% matrix(1, nrow = ncol(salidasdes), ncol = 1))^2
gradcapa = err * d_sigmoide(salidascapas[[nocapas]])
deltacapa[[1]] = (gradcapa %*% t(matrix(1, nrow = nrow(salidascapas[[nocapas - 1]]),
 ncol = ncol(salidascapas[[nocapas - 1]]))) )
deltaact = deltacapa[[1]]
entradas = salidascapas[[nocapas - 1]]
capaactual = capas[[nocapas]]
for(col in 1:ncol(capaactual))
{
capaactual[1:nrow(capaactual), col] = capaactual[1:nrow(capaactual), col]
+ (deltaact[1:nrow(capaactual), col] * entradas[col, 1] * eta)
}
capas[[nocapas]] = capaactual;
if(nocapas > 2)
{
for(i in 1:(nocapas - 2))
{
capaactual = capas[[nocapas - i]]
capaant = capas[[nocapas - i + 1]]
entradas = salidascapas[[nocapas - i - 1]]
salidas = salidascapas[[nocapas - i]]
```

```
deltaant = deltacapa[[i]]
deltaact = matrix(0, nrow = nrow(capaactual), ncol = ncol(capaactual))
for(ren in 1:(nrow(salidas) - 1))
{
gradcapa = t(deltaant[1:nrow(deltaant), ren + 1]) %*%
capaant[1:nrow(deltaant), ren + 1]
deltaact[ren, 1:ncol(deltaact)] = matrix(1, nrow = 1, ncol = ncol(deltaact)) *
gradcapa[1, 1] * d_sigmoide(salidas[ren + 1, 1])
}
for(col in 1:ncol(capaactual))
{
capaactual[1:nrow(capaactual), col] = capaactual[1:nrow(capaactual), col]
+ (deltaact[1:nrow(capaactual), col] * entradas[col, 1] * eta)
}
deltacapa[[i + 1]] = deltaact
capas[[nocapas - i]] = capaactual
}
}
capaactual = capas[[1]]
capaant = capas[[2]]
entradas = xtrab
salidas = salidascapas[[1]]
deltaant = deltacapa[[nocapas - 1]]
deltaact = matrix(0, nrow = nrow(capaactual), ncol = ncol(capaactual))
for(ren in 1:(nrow(salidas) - 1))
{
gradcapa = t(deltaant[1:nrow(deltaant), ren + 1]) %*%
capaant[1:nrow(deltaant), ren + 1]
```

```
deltaact[ren, 1:ncol(deltaact)] = matrix(1, nrow = 1, ncol = ncol(deltaact)) *
gradcapa[1, 1] * d_sigmoide(salidas[ren + 1, 1])
}
for(col in 1:ncol(capaactual))
{
capaactual[1:nrow(capaactual), col] = capaactual[1:nrow(capaactual), col]
+ (deltaact[1:nrow(capaactual), col] * entradas[col, 1] * eta)
}
deltacapa[[nocapas]] = deltaact
capas[[1]] = capaactual
}
# if(epoca == 1 || (epoca %% 1000 ) == 0)
# {
#   cat("\n")
# }
# se guarda los errores de cada epoca para cuestion de grafica
errores[epoca] = sumerror / nrow(entradas)
lasterror = sumerror / nrow(entradas)
epoca = epoca + 1
}
# se genera una matriz de resultados con el doble de columnas,
# la primera es para los datos resultantates y la segunda parte es para
los errores calculados
salidasfinal <- matrix(0, nrow = nrow(xconbias), ncol = (ncol(salidasdes) * 2))
for(dato in 1:nrow(xconbias))
{
#parte del feedforward
#calculo primera capa que son las entradas
```

```
xtrab = as.matrix(xconbias[dato, 1:ncol(xconbias)])
temp = sigmoide(capas[[1]] %*% xtrab)
salidascapas[[1]] = matrix(c(bias, temp), nrow = (nrow(temp) + 1), ncol = 1)
# Calcula las capas ocultas
if(nocapas > 2 )
{
for(capa in 2:(nocapas - 1))
{
temp = sigmoide(capas[[capa]] %*% salidascapas[[capa - 1]])
salidascapas[[capa]] = matrix(c(bias, temp), nrow = (nrow(temp) + 1), ncol = 1)
}
}
# Calcula la capa de salida
temp = sigmoide(capas[[nocapas]] %*% salidascapas[[nocapas - 1]])
salidascapas[[nocapas]] = matrix(c(temp), nrow = (nrow(temp)), ncol = 1)
# calcula el error de la salida vs lo deseado
err = as.matrix(salidasdes[dato, 1:ncol(salidasdes)] - salidascapas[[nocapas]])
salidasfinal[dato, 1:ncol(salidasfinal)] = matrix(c(t(salidascapas[[nocapas]]),
t(err)), nrow = 1, ncol = ncol(salidasfinal))
}
return(list(capas, xconbias, salidascapas, salidasdes, deltacapa, err, gradcapa,
xtrab, errores, epoca, salidasfinal))
}
x <- matrix(c(
0.1,0.1,
0.1,0.2,
0.1,0.4,
0.1,0.5,
```

```
0.1,0.6,
0.1,0.7,
0.1,0.9,
0.2,0.1,
0.2,0.2,
0.2,0.3,
0.2,0.4,
0.2,0.5,
0.2,0.7,
0.2,0.8,
0.3,0.1,
0.3,0.2,
0.3,0.3,
0.3,0.4,
0.3,0.6,
0.3,0.7,
0.3,0.9,
0.4,0.1,
0.4,0.2,
0.4,0.3,
0.4,0.5,
0.4,0.6,
0.4,0.7,
0.4,0.8,
0.5,0.1,
0.5,0.3,
0.5,0.4,
0.5,0.5,
```

```
0.5,0.6,
0.5,0.7,
0.5,0.8,
0.6,0.1,
0.6,0.2,
0.6,0.3,
0.6,0.5,
0.6,0.6,
0.6,0.7,
0.6,0.8,
0.6,0.9,
0.7,0.1,
0.7,0.3,
0.7,0.4,
0.7,0.5,
0.7,0.6,
0.7,0.7,
0.7,0.8,
0.8,0.1,
0.8,0.2,
0.8,0.3,
0.8,0.4,
0.8,0.5,
0.8,0.7,
0.8,0.8,
0.9,0.2,
0.9,0.3,
0.9,0.4,
```

```
0.9,0.5,

0.9,0.6,

0.9,0.7,

0.9,0.8,

0.9,0.9

), nrow = 64, ncol = 2, byrow = TRUE)

d <- matrix(c(

0.1,

0.13,

0.19,

0.22,

0.25,

0.28,

0.34,

0.18,

0.21,

0.24,

0.27,

0.3,

0.36,

0.39,

0.25,

0.28,

0.31,

0.34,

0.4,

0.43,

0.49,
```

```
0.33,
0.36,
0.39,
0.45,
0.48,
0.51,
0.54,
0.4,
0.46,
0.49,
0.52,
0.55,
0.58,
0.61,
0.48,
0.51,
0.54,
0.6,
0.63,
0.66,
0.69,
0.72,
0.55,
0.61,
0.64,
0.67,
0.7,
0.73,
```

```
0.76,
0.63,
0.66,
0.69,
0.72,
0.75,
0.81,
0.84,
0.7,
0.73,
0.76,
0.79,
0.82,
0.85,
0.87,
0.9
), nrow = 64, ncol = 1, byrow = TRUE)
retorno <- entrenamiento(x, d, matrix(c(5), nrow = 1, ncol = 1),
 0.1, 0.0005,1000000, 1)
```

Apéndice 3. Código de MatLab para obtención de Resultados de Red Neuronal

```
x=[1 0.1 0.1];  % La matriz original es la siguiente, incluyendo el valor deseado [.1]

w11=[0.6561117 -1.832009  1.7512827];
w12= [1.5291060 -0.114930  0.5213453];
w13= [-0.5862820 -6.280922 -1.6976867];
w14= [4.4599577 -2.022048 -2.0806504];
```

```
w15= [-1.4629441  2.760987 -0.9299505];
w21= [-2.723413 5.166674 0.1700119 -5.257298 -3.428121 6.863272];
%Resultados de la primer capa
v11=x*(w11)';
v12=x*(w12)';
v13=x*(w13)';
v14=x*(w14)';
v15=x*(w15)';
y11=(1/(1+exp(-v11)));
y12=(1/(1+exp(-v12)));
y13=(1/(1+exp(-v13)));
y14=(1/(1+exp(-v14)));
y15=(1/(1+exp(-v15)));
y1=[1 y11 y12 y13 y14 y15];
%Resultados 2da. capa oculta
v21=y1*(w21)';
y21=(1/(1+exp(-v21)))
```

Apéndice 4. Código de Arduino Para funcionamiento Automático

En esta parte del código se declaran las variables globales del sistema.

```
#include <MatrixMath.h>
#include "Configuracion.h"
char cadena[30];
byte posicion =0;
uint8_t identificador;
double y21;
volatile int cont1 = 0;   // Variable entera que se almacena en la RAM del Micro
long T0 = 0 ;
```

```
float kres=0;
long lectura_sensor;
float salida1;
struct Digitalitation_Sensores_return{
double sensor1;
}Entrada_sensores;
/*****************************************************************
Inicializando entradas
*****************************************************************/
const int fil_input = 1;
const int col_input=3;                    // vias  D    E
double Input[fil_input][col_input] = {   // { 1,  0  , 0}
{1, 0.0, 0.0}
};
/*****************************************************************
Inicializando Valores Deseados
 double deseado[fil_deseado][col_deseado] = {{Motorderecha,Motorizquierda}}
*****************************************************************/
/*****************************************************************
inicializando pesos de las sinapsis de entrada a capa oculta
*****************************************************************/
const int fil_W11=1;
const int col_W11=3;
double Pesos_W11[fil_W11][col_W11] = {
{0.6561117, -1.832009,  1.7512827}
};
const int fil_W12=1;
const int col_W12=3;
```

```
double Pesos_W12[fil_W12][col_W12] = {
{1.5291060, -0.114930,  0.5213453}
};
const int fil_W13=1;
const int col_W13=3;
double Pesos_W13[fil_W13][col_W13] = {
{-0.5862820, -6.280922, -1.6976867}
};
const int fil_W14=1;
const int col_W14=3;
double Pesos_W14[fil_W14][col_W14] = {
{4.4599577, -2.022048, -2.0806504}
};
const int fil_W15=1;
const int col_W15=5;
double Pesos_W15[fil_W15][col_W15] = {
{-1.4629441, 2.760987, -0.9299505}
};
/*****************************************************************
inicializando pesos de las sinapsis de la capa oculta a la capa de salida
*****************************************************************/
const int fil_W21 = 1;
const int col_W21 = 6;
double Pesos_W21[fil_W21][col_W21] = {
{-2.723413, 5.166674, 0.1700119, -5.257298, -3.428121, 6.863272}
};
void setup() {
Serial.begin(9600);
```

```
Serial2.begin(9600);
Serial3.begin(9600);
pinMode(MOTOR,OUTPUT);
Matrix.Print((double*)Input, fil_input, col_input, "Entradas: ");
}
void loop() {
if (Serial3.available() > 0){
////////////////////////////////////////////////////////////////////////////////////
delay(5);
memset(cadena, 0,sizeof(cadena));//memset borra el contenido del array  "cadena" desde
la posición 0 hasta el final sizeof posicion=0;
while(Serial3.available()>0) //Mientras haya datos en el buffer ejecuta la función en
la cual se concatena la cadena recibida byte por byte
{
delay(5); //Delay para mejorar la recepción de datos
cadena[posicion]=Serial3.read();//Lee un carácter del string "cadena"
de la "posicion", luego lee el siguiente carácter con "posicion++"
posicion++;
}
if (cadena[0] == '1'){  //Busca el caracter 1 en la cadena recibida en
la primera posición
String dias = subStr(cadena , "/", 2);   //se ejecuta la función subStr para separar
LOS VALORES DIVIDIDOS POR '/' SubStr(cadena,separador,posición) lo que devuelve
se guarda en la variable
String camaronxm2 = subStr(cadena , "/", 3);
Serial.println();
cadena_TX(dias.toInt(),camaronxm2.toInt());
Serial.println();
```

```
}
}
Cadena_RX();
}
```

En esta parte del código se plantean las condiciones de las señales de salida del sistema.

```
void condiciones(){
float m1=y21;
Serial.println();
float kres= (m1*125)-2.5;
while(kres > 0){
digitalWrite(MOTOR,HIGH);
while (cont1 < 4){
}
delay(100);
interrupcion0();
digitalWrite(MOTOR,LOW);
delay(10);
if(cont1=4){
kres - 0.350;
cont1=0;
}
}
}
```

En esta parte se realiza la configuración de los pines del Arduino, donde se indican en que pin están conectados los elementos del sistema.

```
#ifndef Configuracion_h
#define Configuracion_h
```

```
#define MAX_STRING_LEN 30
#define SENSOR_VUELTAS 29
#define MOTOR 27
class Variables{
} ;
#endif
```

En esta parte se realiza la lectura y digitalización del sensor para el monitoreo de pulsos del sistema dosificador.

```
Digitalitation_Sensores_return Lectura_sensores() {
void interrupcion0()    // Funcion que se ejecuta durante cada interrupion
{
if ( millis() > T0  + 200)
{
cont1++;     // Se incrementa en uno el contador
T0 = millis();
}
}
return Entrada_sensores;
}
```

En esta parte se lee la cadena de recepción (RX). Esta cadena viene desde el dispositivo móvil hacia el microcontrolador.

```
void Cadena_RX(){
if (Serial2.available() > 0){
delay(5);
memset(cadena, 0,sizeof(cadena));//memset borra el contenido del array  "cadena" desde
 la posición 0 hasta el final sizeof posicion=0;
```

```
while(Serial2.available()>0) //Mientras haya datos en el buffer ejecuta la función en
 la cual se concatena la cadena recivida byte por byte
{
delay(5); //Delay para mejorar la recepción de datos
cadena[posicion]=Serial2.read();//Lee un carácter del string "cadena" de la "posicion",
luego lee el siguiente carácter con "posicion++"
posicion++;
}
Serial.print("testigo1");
if (cadena[0] == '2'){
Serial.print("testigo2");
String kilos = subStr(cadena , "/", 2);
Serial3.print('*');
Serial3.print('/');
Serial3.println(kilos.toInt());
Serial.print('*');
Serial.print('/');
Serial.println(kilos.toInt());
}
}
}
```

En esta parte del código se codifican los valores, en modo de cadena (TX), que serán enviados desde el microcontrolador hacia el dispositivo móvil.

```
void cadena_TX(uint8_t dias, uint8_t especies){
identificador = 1;
Serial2.print(identificador);
Serial2.print("/");
Serial2.print(dias);
```

```
Serial2.print("/");
Serial2.print(especies);
Serial2.println();
}
```

En esta parte del código se codifican y se leen las variables tipo cadena del sistema.

```
char* subStr (char* str, char *delim, int index) {
char *act, *sub, *ptr;
static char copy[MAX_STRING_LEN];
int i;
// ya que strtok consume el primer arg, se hace una copia
strcpy(copy, str);

for (i = 1, act = copy; i <= index; i++, act = NULL) {
//Serial.print(".");
sub = strtok_r(act, delim, &ptr);
if (sub == NULL) break;
}
return sub;
}
```

En esta parte del código se realiza la propagación hacia adelante de la red neuronal, donde se tienen declarados los pesos finales para dicho sistema (obtenidos con Rstudio).

```
void feed_foward (double dias,double especies){
// Serial.println();
// Serial.println("Sensor Delante Digitalizado: "+String(delante));
// datos que van en la entrada Input
// Serial.println("Sensor Izquierda Digitalizado: "+String(izquierda));
// Serial.println("Sensor Atrás Digitalizado: "+String(atras));
```

```
// Serial.println("Sensor Derecha Digitalizado: "+String(derecha));
// Serial.println();
/*****************************************************************
insertar los datos al arreglo para su multiplicación
*****************************************************************/
double Input[fil_input][col_input] = {
{1, dias, especies}
};
Matrix.Print((double*) Input, fil_input, col_input, "Entrada: ");
/*****************************************************************
Trasponer pesos W1 y W2 de la entrada a la capa oculta para multiplicar por la entrada
*****************************************************************/
int fil_W11_traspose=col_W11;
int col_W11_traspose=fil_W11;
double W11_Traspose[fil_W11_traspose][col_W11_traspose];
Matrix.Transpose((double*)Pesos_W11,fil_W11_traspose, col_W11_traspose,
 (double*) W11_Traspose);
// Matrix.Print((double*) W11_Traspose, fil_W11_traspose,
 col_W11_traspose, "w11_traspuesta:");
int fil_W12_traspose=col_W12;
int col_W12_traspose=fil_W12;
double W12_Traspose[fil_W12_traspose][col_W12_traspose];
Matrix.Transpose((double*)Pesos_W12,fil_W12_traspose, col_W12_traspose,
 (double*) W12_Traspose);
// Matrix.Print((double*) W12_Traspose, fil_W12_traspose,
 col_W12_traspose, "w12_traspuesta:");
int fil_W13_traspose=col_W13;
int col_W13_traspose=fil_W13;
```

```
double W13_Traspose[fil_W13_traspose][col_W13_traspose];
Matrix.Transpose((double*)Pesos_W13,fil_W13_traspose, col_W13_traspose,
 (double*) W13_Traspose);
// Matrix.Print((double*) W13_Traspose, fil_W13_traspose,
 col_W13_traspose, "w13_traspuesta:");
int fil_W14_traspose=col_W14;
int col_W14_traspose=fil_W14;
double W14_Traspose[fil_W14_traspose][col_W14_traspose];
Matrix.Transpose((double*)Pesos_W14,fil_W14_traspose, col_W14_traspose,
 (double*) W14_Traspose);
//Matrix.Print((double*) W14_Traspose, fil_W14_traspose,
 col_W14_traspose, "w14_traspuesta:");
int fil_W15_traspose=col_W15;
int col_W15_traspose=fil_W15;
double W15_Traspose[fil_W15_traspose][col_W15_traspose];
Matrix.Transpose((double*)Pesos_W15,fil_W15_traspose, col_W15_traspose,
 (double*) W15_Traspose);
// Matrix.Print((double*) W15_Traspose, fil_W15_traspose,
 col_W15_traspose, "w15_traspuesta:");
int fil_W21_traspose=col_W21;
int col_W21_traspose=fil_W21;
double W21_Traspose[fil_W21_traspose][col_W21_traspose];
Matrix.Transpose((double*)Pesos_W21,fil_W21_traspose, col_W21_traspose,
 (double*) W21_Traspose);
//Matrix.Print((double*) W21_Traspose, fil_W21_traspose,
 col_W21_traspose, "w21_traspuesta:");
/*************************************************************
Multiplicar la Entrada (Input) por cada peso (W11....W1x)
```

```
 que entra a cada neurona de la capa oculta
*****************************************************************/
double v11[fil_input][col_W11_traspose];
//matriz donde se va guardar el resultadoo (matriz 1x5 * matriz 5x1 = matriz 1x1)
Matrix.Multiply((double*)Input, (double*)W11_Traspose, fil_input, col_input,
 col_W11_traspose, (double*)v11);
//Matrix.Print((double*) v11, fil_input, col_W11_traspose, "v11:");
double v12[fil_input][col_W12_traspose];
 //matriz donde se va guardar el resultadoo (matriz 1x5 * matriz 5x1 = matriz 1x1)
Matrix.Multiply((double*)Input, (double*)W12_Traspose, fil_input, col_input,
col_W12_traspose, (double*)v12);
//Matrix.Print((double*) v12, fil_input, col_W12_traspose, "v12:");
double v13[fil_input][col_W13_traspose];
//matriz donde se va guardar el resultadoo (matriz 1x5 * matriz 5x1 = matriz 1x1)
Matrix.Multiply((double*)Input, (double*)W13_Traspose, fil_input, col_input,
 col_W13_traspose, (double*)v13);
//Matrix.Print((double*) v13, fil_input, col_W13_traspose, "v13:");
double v14[fil_input][col_W14_traspose];
 //matriz donde se va guardar el resultadoo (matriz 1x5 * matriz 5x1 = matriz 1x1)
Matrix.Multiply((double*)Input, (double*)W14_Traspose, fil_input, col_input,
col_W14_traspose, (double*)v14);
//Matrix.Print((double*) v14, fil_input, col_W14_traspose, "v14:");
double v15[fil_input][col_W15_traspose];
//matriz donde se va guardar el resultadoo (matriz 1x5 * matriz 5x1 = matriz 1x1)
Matrix.Multiply((double*)Input, (double*)W15_Traspose, fil_input, col_input,
col_W15_traspose, (double*)v15);
//atrix.Print((double*) v15, fil_input, col_W15_traspose, "v15:");
/*****************************************************************
```

```
Aplicar Función de activación a la salida v1 de cada neurona para convertirse en
la nueva entrada de la capa de salida
*****************************************************************/
double y11,y12,y13,y14,y15;
y11 = (1/(1+exp(-v11[0][0])));
y12 = (1/(1+exp(-v12[0][0])));
y13 = (1/(1+exp(-v13[0][0])));
y14 = (1/(1+exp(-v14[0][0])));
y15 = (1/(1+exp(-v15[0][0])));
// Serial.println();
// Serial.println("Las salidas de la capa oculta son: ");
// Serial.println("Salida y11: ");
// Serial.println(y11,4);
// Serial.println("Salida y12: ");
// Serial.println(y12,4);
// Serial.println("Salida y13: ");
// Serial.println(y13,4);
// Serial.println("Salida y14: ");
// Serial.println(y14,4);
// Serial.println("Salida y15: ");
// Serial.println(y15,4);
/const int fil_Y1=1;
const int col_Y1=11;
double Y1[fil_Y1][col_Y1] = {
{ 1, y11, y12, y13, y14, y15}
};
//Matrix.Print((double*) Y1, fil_Y1, col_Y1, "Las salidas de la capa oculta son: ");
/*****************************************************************
```

```
Multiplicar la nueva Entrada (Y1) por cada peso (W21....W2x) que entra a cada neurona
de la capa de Salida
*****************************************************************/
double v21[fil_Y1][col_W21_traspose];
//matriz donde se va guardar el resultadoo (matriz 1x10 * matriz 10x1 = matriz 1x1)
Matrix.Multiply((double*)Y1, (double*)W21_Traspose, fil_Y1, col_Y1, col_W21_traspose,
(double*)v21);
//Matrix.Print((double*) v21, fil_Y1, col_W21_traspose, "v21:");
/*****************************************************************
Aplicar Función de activación a la salida de la red neuronal
*****************************************************************/
y21 = (1/(1+exp(-v21[0][0])));    // SALIDA
Serial.println();
Serial.println("Las salidas de la red es: ");
Serial.println("Salida y21: ");
Serial.println(y21,4);
condiciones();
}
```

Apéndice 5. Código de Ionic Frameworks para la Aplicación Móvil de prueba

Código de Vista de Pantalla para búsqueda de dispositivos Bluetooth

```
<ion-header>
<ion-navbar>
<ion-title>Conexion</ion-title>
</ion-navbar>
</ion-header>
<ion-content padding>
<ion-card>
```

```
<ion-card-header>
</ion-card-header>
<ion-card-content>
<button ion-button block (click)="presentLoading()">Buscar dispositivos</button>
</ion-card-content>
</ion-card>
<ion-list>
<ion-item *ngFor="let unpired of listUnpired">
<ion-icon name="bluetooth" item-left></ion-icon>
<h2> {{unpired.name}}</h2>
<h3> Mac: {{unpired.address}}</h3>
<h3> Id: {{unpired.id}}</h3>
<button (click)="conectar(unpired.address)" block block ion-button icon-right>
Conectarse
<ion-icon name="arrow-dropright-circle"></ion-icon>
</button>
</ion-item>
</ion-list>
<ion-list>
<ion-item *ngFor="let pired of lists">
<ion-icon name="bluetooth" item-left></ion-icon>
<h2> {{pired.name}}</h2>
<h3> Mac: {{pired.address}}</h3>
<h3> Id: {{pired.id}}</h3>
<button (click)="conectar(pired.address)" block block ion-button icon-right>
Conectarse
<ion-icon name="arrow-dropright-circle"></ion-icon>
</button>
```

```
</ion-item>
</ion-list>
</ion-content>
```

Código de lógica para conexión Bluetooth

```
import { Component } from '@angular/core';
import { LoadingController } from 'ionic-angular';
import { NavController } from 'ionic-angular';
import { BluetoothSerial } from 'ionic-native';
import { AboutPage } from '../pages/about/about';
import { ContactPage } from '../contact/contact';
import { TabsPage } from '../tabs/tabs';
@Component({
selector: 'page-home',
templateUrl: 'home.html'
})
export class HomePage {
public lists = [];
public listUnpired =[];
public var2: string ;
public working:string;
constructor(public navCtrl: NavController,public loadingCtrl: LoadingController) {
// this.presentLoading();
}
presentLoading() {
// metodo
BluetoothSerial.isEnabled().then(res => {
console.log("Bluetooth encendido: "+res);
BluetoothSerial.discoverUnpaired().then(res=>{
```

```
console.log("dispositivos sin emparejar:");
console.log(res);
this.listUnpired = res;
},err=>{
console.log(err);
});
BluetoothSerial.list().then((allDevices) => {
console.log("dispositivos emparejados:");
console.log(allDevices);
this.lists = allDevices;
if(this.lists.length == 0){
this.var2 = "no se encontraron dispositivos";
}
});
this.loadingCtrl.create({
content: 'Buscando dispositivos...',
duration: 12000,
dismissOnPageChange: true
}).present();
// regresa los dispositivos disponibles emparejados
},err=>{
alert("Bluetooth desactivado");
// let alert = Alert.create({
//   title: 'Conexión',
//   subTitle: 'Bluetooth desactivado',
//   buttons: ['Aceptar']
//  });
});
```

```
}
conectar(macadress) {
console.log("msg");
console.log(macadress);
BluetoothSerial.isEnabled().then(res => {
BluetoothSerial.connect(macadress).subscribe(
(res)=>{
console.log(res);
alert("Conexión exitosa");
//  this.navCtrl.push(AboutPage);
// let alert = Alert.create({
//  title: 'Conexión',
//  subTitle: 'Conexión exitosa',
//  buttons: ['Aceptar']
// });
},
(err)=>{
console.log(res);
alert("Ocurrio un problema");
// let alert = Alert.create({
//  title: 'Error',
//  subTitle: 'Ocurrio un problema',
//  buttons: ['Aceptar']
// });
});
},err=>{
console.log(err);
});
```

```
}
}
```

Código de la Vista para la pantalla de Control del sistema

```
<ion-header>
<ion-navbar>
<ion-title>
Alimentación
</ion-title>
</ion-navbar>
</ion-header>
<ion-content padding>
<ion-card>
<ion-item>
<ion-avatar item-left>
<ion-icon name="md-aperture" item-left></ion-icon>
</ion-avatar>
<h2>Porciónes</h2>
<p></p>
</ion-item>
<!-- <img src="../img/caba.jpg"> -->
<ion-card-content>
<ion-list>
Días: {{this.dia}}
<ion-item>
<ion-range   step="1" min="1" max="60" snaps="true" [(ngModel)]="dia">
<ion-label range-left class="small-text">-</ion-label>
<ion-label range-right>+</ion-label>
</ion-range>
```

```
</ion-item>
Número de especies x m2: {{this.epm2}}
<ion-item>
<ion-range   step="1" min="6" max="18" snaps="true" [(ngModel)]="epm2">
<ion-label range-left class="small-text">-</ion-label>
<ion-label range-right>+</ion-label>
</ion-range>
</ion-item>
</ion-list>
<ion-list>
<button ion-button block (click)="Enviar_datos()"
 color="default">Cantidad de Dosificación</button>
</ion-list>
</ion-card-content>
</ion-card>
<ion-card *ngIf="tiempo !=''">
<ion-card-content >
<ion-item >Cantidad a dispensar: {{x1}}</ion-item>
<ion-item >Tiempo de operación: {{tiempo}}</ion-item>
</ion-card-content>
</ion-card>
<!-- Real floating action button, fixed. It will not scroll with the content -->
<ion-fab right bottom>
<button ion-fab color="danger"(click)=
"Enviar_datos()"><ion-icon name="md-play"></ion-icon></button>
</ion-fab>
</ion-content>
```

Código de Lógica de pantalla de Control

```
import { Component } from '@angular/core';
import { BluetoothSerial } from 'ionic-native';
import { NavController } from 'ionic-angular';
import { LoadingController } from 'ionic-angular';
import { AlertController } from 'ionic-angular';
@Component({
selector: 'page-about',
templateUrl: 'about.html'
})
export class AboutPage {
cadena: string;
dia = 1;
epm2 =6;
identificador =1;
public dato_rx: any = "";
x1;
tiempo;
constructor(public navCtrl: NavController,public loadingCtrl:
 LoadingController,public alertCtrl: AlertController) {
var Separacion:string[];
var kg;
var tp;
}
Enviar_datos(){
console.log(this.identificador);
console.log(this.dia);
console.log(this.epm2);
let Cadena: string = this.identificador+"/"+this.dia+"/"+this.epm2;
```

```
console.log(Cadena);
console.log("se ejecuto");
BluetoothSerial.isEnabled().then(res => {
BluetoothSerial.isConnected().then(res=>{
BluetoothSerial.write(Cadena).then(res=>{
console.log("dato Correctamente enviado");
let loader = this.loadingCtrl.create({
content: "Leyendo datos...",
duration: 4000
});
loader.present();
loader.dismiss();
}, err=>{
alert("error al enviar dato");
});
BluetoothSerial.subscribeRawData().subscribe(data => {
console.log("Subscription");
BluetoothSerial.readUntil("*").then(res=>{
console.log("datos recividos");
BluetoothSerial.read().then(tp =>{
let stringToSplit = tp;
let x = stringToSplit.split("/");
this.x1=x[1];
this.tiempo = x[1]/2;
console.log(this.x1);
console.log(this.tiempo);
//BluetoothSerial.clear();
},err=>{
```

```
alert("no hay datos en el buffer");
});
},err=>{
});
});
},
err=>{
alert("No hay dispositivos emparejados");
});
},err=>{
alert("Bluetooth deshabilitado");
});
}
garage(){
console.log("se ejecuto");
BluetoothSerial.isEnabled().then(res => {
BluetoothSerial.isConnected().then(res=>{
BluetoothSerial.write("h").then(res=>{
console.log("dato Correctamente enviado");
}, err=>{
alert("error al enviar dato");
});
},
err=>{
alert("No hay dispositivos emparejados");
});
},err=>{
alert("Bluetooth deshabilitado");
```

```
});
}
}
```

Archivo app.component.ts. En esta parte se carga la plataforma.

```
import { Component } from '@angular/core';
import { Platform } from 'ionic-angular';
import { StatusBar, Splashscreen } from 'ionic-native';
import { TabsPage } from '../pages/tabs/tabs';
@Component({
templateUrl: 'app.html'
})
export class MyApp {
rootPage = TabsPage;
constructor(platform: Platform) {
platform.ready().then(() => {
// Okay, so the platform is ready and our plugins are available.
// Here you can do any higher level native things you might need.
StatusBar.styleDefault();
Splashscreen.hide();
});
}
}
```

Archivo app.module.ts. Archivo de los imports, donde están los módulos y las páginas.

```
mport { NgModule, ErrorHandler } from '@angular/core';
import { IonicApp, IonicModule, IonicErrorHandler } from 'ionic-angular';
import { MyApp } from './app.component';
import { AboutPage } from '../pages/about/about';
```

```
import { ContactPage } from '../pages/contact/contact';
import { HomePage } from '../pages/home/home';
import { TabsPage } from '../pages/tabs/tabs';
@NgModule({
declarations: [
MyApp,
AboutPage,
ContactPage,
HomePage,
TabsPage
],
imports: [
IonicModule.forRoot(MyApp)
],
bootstrap: [IonicApp],
entryComponents: [
MyApp,
AboutPage,
ContactPage,
HomePage,
TabsPage
],
providers: [{provide: ErrorHandler, useClass: IonicErrorHandler}]
})
export class AppModule {}
```

En esta parte se configuran y se encuentran los archivos tipo: Archivo main.ts.

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app.module';
```

```
platformBrowserDynamic().bootstrapModule(AppModule);
```

En esta parte se encuentran los archivos tipo: Indes.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
<meta charset="UTF-8">
<title>Ionic App</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
 minimum-scale=1.0, maximum-scale=1.0, user-scalable=no">
<meta name="format-detection" content="telephone=no">
<meta name="msapplication-tap-highlight" content="no">
<link rel="icon" type="image/x-icon" href="assets/icon/favicon.ico">
<link rel="manifest" href="manifest.json">
<meta name="theme-color" content="#4e8ef7">
<!-- cordova.js required for cordova apps -->
<script src="cordova.js"></script>
<!-- un-comment this code to enable service worker
<script>
if ('serviceWorker' in navigator) {
navigator.serviceWorker.register('service-worker.js')
.then(() => console.log('service worker installed'))
.catch(err => console.log('Error', err));
}
</script>-->
<link href="build/main.css" rel="stylesheet">
</head>
<body>
<!-- Ionic's root component and where the app will load -->
```

```
<ion-app></ion-app>
<!-- The polyfills js is generated during the build process -->
<script src="build/polyfills.js"></script>
<script src="build/vendor.js"></script>
<!-- The bundle js is generated during the build process -->
<script src="build/main.js"></script>
</body>
</html>
```

En esta parte del código se encuentran los componentes de las tabs : tabs.html

```
<ion-tabs>
<ion-tab [root]="tab1Root" tabTitle="Conexión" tabIcon="bluetooth"></ion-tab>
<ion-tab [root]="tab2Root" tabTitle="Control" tabIcon="switch"></ion-tab>
<ion-tab [root]="tab3Root" tabTitle="Información"
 tabIcon="information-circle"></ion-tab>
</ion-tabs>
```

En esta parte del código se encuentra la lógica de tabs.ts

```
import { Component } from '@angular/core';
import { HomePage } from '../home/home';
import { AboutPage } from '../about/about';
import { ContactPage } from '../contact/contact';
@Component({
templateUrl: 'tabs.html'
})
export class TabsPage {
// this tells the tabs component which Pages
// should be each tab's root Page
tab1Root: any = HomePage;
```

```
tab2Root: any = AboutPage;

tab3Root: any = ContactPage;

constructor() {

}

}
```