Diseño e implementación de un sistema automático de identificación, monitoreo y registro de las diferentes actividades pecuarias en la producción lechera de la Hacienda Bellavista ubicada en la parroquia rural Lloa, provincia de Pichincha

Cajamarca Soliz, Nelson Alejandro y Sanipatín Espinosa, Isaac Fernando

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica, Automatización y Control

Ing. Vargas Vallejo, Vanessa Carolina, PhD.

31 de julio del 2022

COPYLEAKS

Trabajo Titulacion_Cajamarca_Nelson_Sanipatin_Isaac.pdf

Scanned on: 16:55 August 2, 2022 UTC

VANESSA CAROLINA VARGAS VALLEJO

2.1%	36	44183
Overall Similarity Score	Results Found	Total Words in Text

Identical Words	400
Words with Minor Changes	85
Paraphrased Words	434
Omitted Words	0

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica Automatización y Control

Certificación

Certifico que el trabajo de titulación: "**Diseño e implementación de un sistema automático de identificación, monitoreo y registro de las diferentes actividades pecuarias en la producción lechera de la Hacienda Bellavista ubicada en la parroquia rural Lloa, provincia de Pichincha**" fue realizado por los señores **Cajamarca Soliz, Nelson Alejandro y Sanipatín Espinosa, Isaac Fernando**; el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 31 de julio del 2022

Firma:

Firmado electrónicamente por:
VANESSA CAROLINA VARGAS VALLEJO

.....

Ing. Vargas Vallejo, Vanessa Carolina, PhD

C. C.: 1711309045

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Responsabilidad de Autoría

Nosotros, **Cajamarca Soliz, Nelson Alejandro** y **Sanipatín Espinosa, Isaac Fernando**, con cédulas de ciudadanía n° 0350140265 y n° 1004037139, declaramos que el contenido, ideas y criterios del trabajo de titulación: **Diseño e implementación de un sistema automático de identificación, monitoreo y registro de las diferentes actividades pecuarias en la producción lechera de la Hacienda Bellavista ubicada en la parroquia rural Lloa, provincia de Pichincha**, es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 31 de julio del 2022

Firma:

.....

Nelson Alejandro Cajamarca Soliz

C. C.: 0350140265

Firma:

.....

Isaac Fernando Sanipatín Espinosa

C. C.: 1004037139

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Autorización de Publicación

Nosotros **Cajamarca Soliz, Nelson Alejandro** y **Sanipatín Espinosa, Isaac Fernando,** con cédulas de ciudadanía n° 0350140265 y nª 1004037139, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Diseño e implementación de un sistema de identificación, monitoreo y registro de las diferentes actividades pecuarias en la producción lechera de la Hacienda Bellavista ubicada en la parroquia rural Lloa, provincia de Pichincha** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Sangolquí, 31 de julio del 2022

Firma:

.....

Nelson Alejandro Cajamarca Soliz

C. C.: 0350140265

Firma:

.....

Isaac Fernando Sanipatín Espinosa

C. C.: 1004037139

Dedicatoria

Este trabajo de titulación le dedico con todo cariño a mis padres Nelson Cajamarca y Carmen Soliz, las personas más importantes en mi vida, quienes, con su gran sabiduría, motivación, paciencia y amor, han sabido guiarme en todo momento y me han ayudado a tomar las decisiones más difíciles. A mis queridos hermanos, por entenderme y apoyarme a cumplir mi meta, este es el resultado.

Una dedicatoria dirigida al cielo, para mi abuelita María Orellana, quien pudo compartir hermosas experiencias, anécdotas y lecciones de vida, que me ayudaron a ser mejor un buen hijo, un buen alumno y un buen amigo.

Nelson Alejandro Cajamarca Soliz

Dedicatoria

Este proyecto de titulación va dedicado en primer lugar a Dios, quien me ha dado la fortaleza para poder culminar con esta etapa.

A las personas que más han influenciado en mi vida dándome los mejores consejos, apoyándome y guiándome para ser una persona de bien para la sociedad. Dedico con mucho cariño a mi padre Fausto Sanipatín y a mi madre Violeta Espinosa, que con gran dedicación me han inculcado los valores de responsabilidad, humildad y gratitud.

A mi hermano y hermanas, quienes han sido una parte fundamental en mi vida, son quienes han creído en mí siempre, dando ejemplos de superación, perseverancia, y motivándome a cumplir con mis sueños.

A mi novia Diana, que con su corazón bondadoso y su paciencia me ha acompañado en este camino.

A mis amigos, los cuales me han brindado su ayuda desinteresadamente cuando más lo necesitaba y también hemos compartido gratos momentos.

Isaac Fernando Sanipatín Espinosa

Agradecimiento

Culminado este trabajo quiero utilizar este espacio para agradecer a Dios por todas sus bendiciones, por ser mi guía y fortaleza en este camino lleno de experiencias.

Las metas alcanzadas, refleja el esfuerzo y amor incondicional de los padres a sus hijos. Gracias a mis padres he podido concluir mi principal objetivo y con mucho orgullo agradezco a Nelson Cajamarca y Carmen Soliz, mi mayor inspiración, por confiar y creer en mis expectativas, por los principios y valores que me han inculcado.

Un hermano es un obsequio que nos regala la naturaleza, y que generosa ha sido la naturaleza conmigo. Mi agradecimiento infinito a mis hermanos, mi motor de motivación, quienes me han apoyado desinteresadamente en el transcurso de esta carrera y estar siempre presentes en todo momento.

Dice un proverbio, <<Quien encuentra un amigo, encuentra un tesoro>>. Mis sinceros agradecimientos a las grandes y sinceras amistades que he conocido en el transcurso de esta carrera, les deseo éxitos en su vida profesional.

Agradezco a mi tutora de tesis Ing. Vanessa Vargas y a todo el personal que conforma el departamento de Electrónica, por compartir sus conocimientos y experiencias, por ser un ejemplo a seguir como futuro profesional.

También agradezco al dueño de la Hacienda Bellavista por abrir sus puertas y permitir realizar nuestro trabajo de titulación dentro de sus instalaciones.

Sin más me permito decir, ¡Lo hemos logrado!, este no fue solo mi sueño sino de ustedes también.

Nelson Alejandro Cajamarca Soliz

Agradecimiento

A Dios, por brindarme la salud, la sabiduría, la fortaleza para conseguir este nuevo logro y acompañarme siempre.

A mis padres, por ser un gran ejemplo para mí, por darme más de lo que tenía, por esto y muchas otras cosas más estoy y estaré eternamente agradecido. Siempre estará en mis pensamientos y sé que estará satisfecho por este tan esperado logro en mi vida.

A mi madre, por ir hasta donde mí preocupándose de que no me faltara nada, por su amor incondicional, por la devoción que tiene a sus hijos, por su apoyo y su confianza, no hay palabras en este mundo para agradecerle.

A mi hermano Pablo y mis hermanas Ximenita, Jacquita, Tanya, Miriam quienes han sido parte de mi vida, quienes me han apoyado y acompañado en momentos importantes, y sé que siempre puedo contar con ellos.

A mi novia, por entenderme en todo, por la ayuda que me ha brindado en los momentos y situaciones tempestuosas, por sus palabras de motivación para alcanzar este objetivo.

A mi tutora de tesis, Ing. Vanessa Vargas, PhD. por su paciencia y su tiempo, además por guiarnos en el transcurso de este proyecto con su experiencia y conocimientos.

A la Hacienda Bellavista y a su personal, por su financiamiento, por abrirnos las puertas de su hacienda y darnos la oportunidad de realizar este proyecto.

A mi compañero de tesis por su amistad, esfuerzo y dedicación durante la realización de este trabajo.

A mis amigos y compañeros quienes han sido parte de esta etapa, quienes han compartido su tiempo, alegrías, conocimientos.

A los docentes, quienes nos han impartido su conocimiento, sus experiencias y consejos en esta etapa universitaria.

Isaac Fernando Sanipatín Espinosa

Índice de Contenido

Índice de Tablas

Índice de Figuras

Resumen

Hoy en día la utilización de nuevas tecnologías en el sector pecuario es una necesidad, puesto que permite automatizar los procesos de producción y gestión de una hacienda. Debido al alto costo de inversión que representan los sistemas de automatización existentes en el mercado, surge la necesidad de diseñar e implementar un sistema de bajo costo que permita el registro automático de la producción lechera por cabeza de ganado. Para ello se requiere integrar tres sistemas: a) identificación automática del ganado bovino en sala de ordeño, b) la medición de la producción de leche por cabeza de ganado y c) el registro de la producción en una base de datos. El sistema de identificación se desarrolló empleando tecnología RFID, usando para ello dos lectores que se comunican con un controlador mediante protocolo propietario. Para la medición de la producción de leche se implementó un sistema de visión artificial que permite el reconocimiento de los caracteres de la pantalla de los medidores de volumen de leche. Por otra parte, para el registro de la información se diseñó e implementó una base de datos, que permitirá gestionar el registro de otras actividades pecuarias. Finalmente, se desarrolló una aplicación de escritorio que es la interfaz entre el usuario y la base de datos y permite realizar y mostrar información de la actividad pecuaria en tiempo real. Para ello, se empleó *QtDesigner* para el diseño del *Front-End*, Python para la implementación del *Back-End*, y SQL para la base de datos. El integrador del sistema es una SBC Odroid XU4 y la gestión de las comunicaciones se realiza mediante protocolo MODBUS en RS-485 y protocolo propietario de lectores/RS-485. Las pruebas de funcionamiento realizadas al sistema demuestran que existe una buena fiabilidad en cuanto, a conexión, transmisión de datos y reconocimiento de caracteres.

Palabras Clave: identificación automática, visión artificial, base de datos.

Abstract

Nowadays, the use of new technologies in the livestock sector is an increasing necessity, since it allows automating the production and management processes of a farm. Due to the high investment cost that existing automation systems represent; it is important to design and implement a low-cost system that allows automatic registration of milk production per head of cattle. For this, it is necessary to integrate three systems: a) automatic identification of cattle in the milking parlor, b) measurement individual mill production per cow, and c) database containing milk production. The identification system was developed using RFID technology, using two readers that communicate with a controller through a proprietary protocol. For the measurement of milk production, an artificial vision system was implemented that allows the recognition of the characters on the screen milk volume meters. On the other hand, a database was designed and implemented for the registration of information, which will make it possible to manage the registration of other livestock activities. Finally, a desktop application was developed. This application serves as interface between the user and the database and allows real-time production and display of livestock activity information for this purpose, QtDesigner was used for designing the Front-End, Python for implementing the Back-End, and SQL for the database. The system integrator is an Odroid XU4 SBC and the communications management is done through the MODBUS/RS-485 protocol and the readers proprietary protocol over RS-485. Performance tests carried out on the system show that there is good reliability in terms of connection, data transmission and character recognition.

Keywords: automatic identification, artificial vision, database.

Capítulo 1: Introducción

Antecedentes

A nivel mundial, la ganadería ha tenido un enorme crecimiento debido principalmente al incremento de la demanda de productos de origen animal. Según un estudio de las perspectivas agrícolas 2019-2028, de la Organización para la Cooperación y el Desarrollo Económicos/Organización de las Naciones Unidas para la Agricultura y la Ganadería (OCDE/FAO) se menciona que la leche tuvo un incremento de la producción mundial de 1.6% y se prevé que en América Latina y el Caribe para el 2028 se incremente el 18% de la producción ganadera (OCDE/FAO, 2019).

Según la publicación de la FAO (2019), la ganadería en el Ecuador representa el 18.07% del PIB del sector agropecuario, lo que le convierte en una de las actividades de mayor importancia en el país. En el estudio “Categorías de Población de Ganado Bovino de Ecuador” del Ministerio de Agricultura y Ganadería MAG (2018) se evidenció que el total de ganado vacuno fue de 4’330.224 animales, distribuido entre 280.709 productores. Esto conlleva a un gran impacto socio-económico por el número de fuentes de trabajo directo e indirecto que dependen de este sector. Según el Análisis 2014-2019 del Sector Ganadero, realizado por Sánchez et al. (2019), el 52% del ganado vacuno se ubica en la sierra, el 40% en la costa y 8% corresponden a la Amazonía. En el mismo análisis, se determinó que el 57% del ganado vacuno pertenece a la ganadería lechera ubicada principalmente en la región sierra mientras que el 43% a la industria cárnica.

En un reportaje del diario El Telégrafo del mes de junio de 2019 se indica que la producción e industrialización de la leche y sus derivados generan aproximadamente 1.400 millones de dólares al año (Guevara, 2019). En el informe de la encuesta “Superficie y Producción Agropecuaria Continua 2020” del INEC (2021) se señala que la producción total de leche en el Ecuador es de aproximadamente 6.15 millones de litros diarios, siendo Pichincha la

provincia de mayor producción con el 13.49% del total nacional y con un rendimiento de 10.48 litros diarios por vaca.

Por los antecedentes expuestos se puede evidenciar que la producción lechera en el Ecuador constituye una actividad económica fundamental. En consecuencia, el Gobierno Nacional ha establecido el Plan Nacional Agropecuario 2020-2030 para impulsar el crecimiento la actividad ganadera lechera, el cual considera como unos de sus objetivos estratégicos el "Fortalecer la producción agropecuaria incrementando su rentabilidad, por medio de la productividad y la reducción de costos productivos a través de la tecnificación, investigación y mano de obra cualificada" (MAG, 2021).

Con la finalidad de alcanzar este objetivo estratégico el Ministerio de Agricultura y Ganadería ha emitido políticas reguladoras o normativas para el beneficio de este sector, entre las más relevantes se puede señalar los siguientes:

- Acuerdo ministerial 001-2013 de marzo del 2013 dado por el Ministerio de Agricultura, Ganadería, Acuacultura y Pesca (MAGAP), tiene como fin controlar y regular la línea de producción tanto de la leche como de sus derivados con el objetivo de proteger la salud, la seguridad alimentaria de la población y con el hecho de que no se den prácticas inadecuadas en el proceso (MAGAP, 2013a).

- Acuerdo Ministerial 394 de septiembre de 2013 que tiene como objetivo la regulación y control del precio del litro de leche cruda. Así mismo, el artículo 7 de este acuerdo contiene las bonificaciones por calidad sanitaria y buenas prácticas ganaderas aplicadas en la obtención por litro de leche cruda (MAGAP, 2013b).

- En el Acuerdo Interinstitucional Nro. 036, MAG (2018a), en la sección de la responsabilidad de los entes reguladores se sugiere la implementación de las tecnologías tanto para el control en los procesos de ordeño como también para la

verificación de la identificación y registro de cada animal. En el mismo acuerdo se establece la norma INEN 9:2012 que tiene como objetivo la regulación del tratamiento de la leche cruda destinada al procesamiento.

Como se puede evidenciar, la tecnificación de la producción es un eje fundamental para mejorar la situación económica de los productores de leche. Sin embargo, es importante señalar que siendo la leche un producto de origen animal es indispensable optimizar la tenencia y las características del ganado para incrementar la calidad de la leche. El estudio de Martínez (2007) resume los factores que afectan la productividad del ganado vacuno agrupándolos en las siguientes categorías: factores genéticos, el ciclo productivo o etapa de lactancia, alimentación, medio ambiente y número de partos.

Con la finalidad de contrarrestar los factores antes mencionados, se debe llevar un registro adecuado del animal, en donde se sugiere a los productores contar con al menos un registro de producción, registro genealógico, registro reproductivo, registro de identificación, registro de pastoreo, registro de vacunación y desparasitación (PRONACA, 2020).

El registro de producción constituye una retroalimentación fundamental para el ganadero con la finalidad de plantear nuevas estrategias para mejorar la eficiencia en la producción a través de la combinación de insumos y planificaciones (Lascano et al., 2020), Adicionalmente al registro de producción, que consiste en conocer la cantidad de litros producidos por cada vaca ya sea de manera diaria, semanal, mensual o anual, es necesario relacionar la productividad de leche con otros factores como la genética, la alimentación, el ciclo productivo de las vacas, la tenencia del animal, etc., para cumplir con los objetivos de mejora.

En el estudio de (Hidalgo et al., 2021) se tiene que para aumentar la producción de leche de las vacas, es necesario tener una tendencia positiva de los valores genéticos estimados. De ahí que, el llevar un registro genealógico puede mejorar la producción ya que sirve para optimizar la selección de animales que se emplearán en los cruces y con esto

también se podrá registrar la descendencia del ganado con el fin de preservar o mejorar las características genéticas (Caiza, 2020). En este aspecto es importante mencionar que según Méndez et al. (2020) la raza Holstein es la de mayor producción de leche por lo que es la predilecta para este sector.

La alimentación es un factor primordial en la transferencia de nutrientes, para lo cual las vacas tienen dos fuentes de alimentación, mediante balanceado y a través de los potreros de las haciendas, (Marcillo & Toala, 2021); de aquí que es indispensable tener un registro de productividad, ya que permite tener una relación de la productividad de las vacas con su alimentación.

Hay que tomar en cuenta también al ciclo productivo de la vaca porque existen fases que interfieren con la producción de leche como la lactancia y la influencia del periodo seco con su efecto para la otra lactancia (Murguía & Castillo, 2012).

En cuanto al registro reproductivo, se debería almacenar el método de reproducción del ganado, sea mediante monta natural o a través de la inseminación artificial, así como también el número de identificación del toro o pajuela que se ha usado. Para ello, se debe conocer datos importantes como el estado reproductivo gestante o no de la vaca, celo, raza, entre otros datos (Arias et al., 2018). Esto con el objetivo de tener un seguimiento a los animales de la nueva generación como de sus progenitores, lo que permitirá establecer si el cruce fue productivo o no.

Para llevar los registros anteriores es necesario llevar un registro de identificación animal, ya que cada animal tiene su respectiva información. Por ello que ambiguamente, se han usado marcas de fuego, marcaje de cola, tatuajes, aretes de plástico etc. (Bonilla & Galárraga, 2009).

Además de lo mencionado anteriormente, es importante llevar calendarios de vacunación porque constituyen una estrategia de bioseguridad para el ganado, evitando enfermedades contagiosas que en algunos casos ocasionarían la muerte del (Frías, 2020);

también se debe considerar un calendario de desparasitación para tener un programa para las desparasitaciones del ganado bovino (Guagala, 2019).

De lo mencionado anteriormente, se debe tener en cuenta también el registro de pastoreo que consiste en la diversificación de los terrenos de acuerdo a las necesidades requeridas por el ganado. Además, es importante conocer las semillas de los pastos o los tipos de pasto que están presentes en los terrenos con la finalidad de conocer sus propiedades y beneficios ya que los pastos también son fuente de energía para el ganado (Batallas, 2020).

Cabe destacar que según Torres (2018), Guamán (2017), Chilpe y Chuma (2015), la mayoría de productores no llevan todos los registros y cuando lo hacen lo realizan de manera manual, muchas veces los registros se extravían o no se dispone de los datos de una manera eficiente, rápida o se encuentran desorganizados. Por lo tanto, es necesario automatizar estos procedimientos manuales mediante un sistema de registro en una base de datos. La alimentación de la información en la base de datos puede ser automática o manual. Para ingresar la información de producción de manera automática, se requiere de un sistema de identificación electrónico del ganado, así como de un sistema de medición.

Con los años, a nivel mundial la ganadería ha sufrido grandes cambios en el sector lechero con el objetivo de mejorar su productividad y facilitar el trabajo a los ganaderos. Según el artículo de (Bonifaz & Requelme, 2011) las buenas prácticas de ordeño, en especial la inclusión de la tecnología, influyen positivamente en la mejora de la productividad ganadera. Gracias a la automatización y robótica las empresas generadoras de tecnología han creado innovadores sistemas de monitoreo del ganado y producción de lechera. Según una publicación de National Geographic del 9 de noviembre del 2017 en muchos países se utilizan drones para el pastoreo del ganado (Brady, 2017).

En la última década la robótica ha influido en el sector ganadero y tal es el caso que, en la actualidad, tanto en países europeos como en Estados Unidos, se usan sistemas de ordeño automáticos y voluntarios, los cuales condicionan a las vacas para que ingresen a ser

ordeñadas de manera voluntaria sin que una persona se encuentre en el proceso. Este sistema recolecta los datos de cada animal, el cual sirve para realizar un seguimiento más adecuado, además, incorpora un software de análisis de datos que sirve como un medio de consulta de la información. En la actualidad el sistema de ordeño robótico más grande del mundo se encuentra en Chile y ordeña cuatro mil vacas al día con 64 robots, produciendo un promedio de 42 litros de leche por vaca por día (Mazziotti, 2019).

Por otro lado, en el Ecuador el monitoreo de la producción lechera ha sido un tema de interés en la última década. Tal es el caso del trabajo de Obando (2011) que permitió tener un sistema de monitoreo simple de la producción lechera, con un registro inalámbrico de cada animal y cuyos datos fueron mostrados en una HMI, las limitaciones de este proyecto se enfocan principalmente en la nula existencia de un repertorio de producción de las vacas, para que el usuario pueda verificar y tener un conocimiento de la cantidad producida en distintos momentos.

Otro proyecto ecuatoriano es el presentado por Gavilanes (2013), donde se logró llevar a cabo el monitoreo de la hacienda “El Yagual de Cananvalle”, ubicada en la provincia de Pichincha, utilizando un programa de gestión “Software Ganadero” e identificadores electrónicos ruminales. Si bien este software facilita la gestión y seguimiento del ganado, no permitía conocer la producción de leche individual.

La identificación animal también es fundamental en la tecnificación de una hacienda ganadera y siendo la identificación por radiofrecuencia o RFID una de las más utilizadas en el país debido al bajo costo (Ponce, 2015), (Carriel et al., 2016). Estos proyectos también manejan sistemas de base de datos y medición de la producción lechera con limitaciones en cuanto a costes de implementación y eficiencia.

En el Ecuador la mayoría de productores no aplican técnicas ni herramientas tecnológicas para llevar a cabo la producción, esto se evidencia en varios estudios realizados a nivel nacional, por ejemplo, Guamán (2017) en su tesis realizada en el sector Capricho, cantón

Carlos Julio Arosemena Tola, provincia de Napo determinó que, de doce productores con un total de 300 cabezas de ganado, el 10% de las haciendas están semi-tecnificados y un 90% no se encuentra tecnificados.

Así mismo en un estudio de Torres (2018) se identificó el bajo nivel de tecnificación que tienen los productores de leche del cantón Cayambe en la aplicación de sus procesos, puesto que el 50% de ellos realizan sus actividades de forma manual, conformado principalmente por los pequeños y medianos productores, el 30% utilizan técnicas de ordeño mecánico que generalmente están representados por los grandes productores y el 20 % emplean los dos métodos, tal estudio determinó que una de las acciones para mejorar la productividad es "Fomentar el desarrollo de tecnología para la agroindustria artesanal, ordeño, acopio y enfriamiento de leche".

Finalmente, con base en un estudio realizado por (Chilpe & Chuma, 2015), para determinar el nivel de tecnificación de las Unidades de Producción Agropecuaria en el Ecuador, se observó que el 3% cuentan con sistemas productivos tecnificados, un 10% semi-tecnificados y el 87% muy poco tecnificado. Por tal motivo se concluye que el nivel de tecnificación de los procesos de producción lechera en el Ecuador es escaso para los pequeños y medianos productores.

Si bien en el mercado existe sistemas que permiten realizar una gestión ganadera adecuada: Orvalex (Francia), DHI-Provo / EZfeed (USA), Polanes Ltd. (Polonia), ATL-Agricultural Technology Ltd (Ecuador), sin embargo, el costo de adquisición es elevado para la mayoría de ganaderos en el Ecuador que son medianos y pequeños productores. Por lo tanto, a través de proyectos innovadores locales se ha buscado generar propuestas que faciliten el trabajo de los ganaderos. Lamentablemente hasta el momento, no se ha generado una propuesta que realice un monitoreo completo de la producción que permita realizar la identificación automática del animal y llevar a cabo todos los registros en una base de datos de fácil acceso para el ganadero.

Adicionalmente, en el mercado existen sistemas de ordeño automático como son WestFalia, Lely, DeLaval, entre otros. Estos sistemas poseen su propio software de gestión que permite visualizar los datos de producción de cada vaca, pero al ser muy complejos, sus costos son elevados alcanzando un estimado de 175000 dólares un solo equipo (Zulovich et al., 2018).

Por otra parte, estos sistemas robotizados también deben tener un costo operativo de mantenimiento mensual que resulta elevado para medianos y pequeños productores.

Por las razones mencionadas, se evidencia que un sistema tecnificado del proceso de producción de leche que se adapte a las necesidades puntuales de este sector a un costo accesible mejoraría las condiciones de este sector. Cabe señalar que el sistema mínimo deseable para realizar esta tecnificación sería contar con el ordeño mecánico, la identificación automática del ganado, un sistema de registro del ganado y de monitoreo de la producción lechera por vaca.

Justificación e Importancia

Las herramientas tecnológicas brindan varias ventajas al ganadero, entre las cuales se puede mencionar que: evita que el ganado presente enfermedades debido al ordeño excesivo, protege a los animales de un sobreordeño y ahorra tiempo en las tareas diarias como la recopilación de información que se necesita para llevar a cabo el monitoreo de la producción. Estas ventajas hacen que sea prioridad el uso de la tecnología para mejorar las condiciones de trabajo de los ganaderos.

El sistema de monitoreo y registro de la producción lechera conjuntamente con la identificación automática del animal presenta las siguientes ventajas:

- Facilita la administración y la toma de decisiones en la hacienda dado que se tienen registros de las actividades pecuarias dentro de una base de datos.

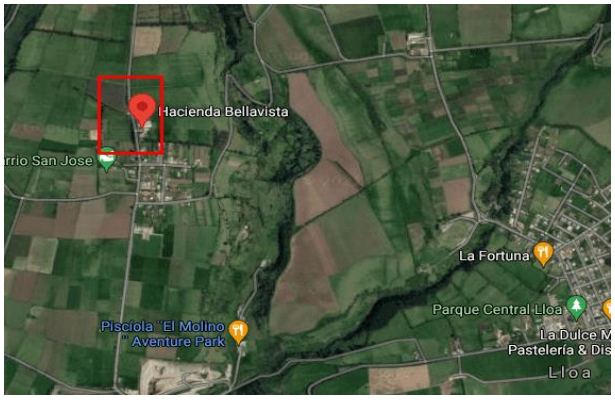
- Permite identificar a las vacas, conocer el historial productivo de cada animal y por ende determinar la cantidad en litros de leche que rinde cada vaca por cada turno de ordeño.
- Conociendo la cantidad de litros que produce cada vaca se podrá mediante un análisis estadístico periódico, caracterizar el ciclo productivo del animal.
- Así mismo, mediante ese análisis del ciclo productivo puede ser usado posteriormente para la correcta dosificación de alimentos para cada animal.

El propósito de este trabajo es crear un sistema de registro y monitoreo de la producción lechera basada en la identificación animal y el almacenamiento de datos, que sea eficiente y accesible en términos de costos para las pequeñas y medianas haciendas ganaderas. Lo cual implica la generación de conocimiento y el aporte de la academia a la sociedad, enmarcando una mayor gestión estratégica para el productor, aumento de la productividad, generación de innovación y adaptación a tecnologías en donde las personas serán partícipes de una mejor toma de decisiones.

Considerando las razones expuestas que sustentan la importancia de fortalecer este sector a través de la tecnificación de los procesos de ordeño, acopio y enfriamiento de leche, algunas haciendas han optado por empezar a tecnificarse e implementar este tipo de tecnología. Tal es el caso de la Hacienda Bellavista, ubicada en la provincia de Pichincha, parroquia rural Lloa, que actualmente cuenta con 152 cabezas de ganado de los cuales 52 están en etapa de producción.

Figura 1

Ubicación de la Hacienda Bellavista en la Parroquia Lloa



Nota. Figura Obtenido de Google Maps.

Si bien es cierto la hacienda ha comenzado a tecnificar sus procesos, pero aún es necesario realizar la identificación automática del animal, llevar un monitoreo de la producción de leche individual y crear un registro en una base de datos para el almacenamiento de información de cada vaca con el fin de mejorar estos procesos. Por lo tanto, esta hacienda se ha establecido como el lugar apropiado para desarrollar un prototipo de tecnificación a través del presente proyecto de titulación.

Alcance del proyecto

El proyecto consta de cuatro etapas identificables y complementarias. En primer lugar, se diseñará e implementará un sistema de identificación automática del ganado basado en tecnologías inalámbricas. Esta información será procesada mediante un controlador y posteriormente se enviará hacia una base de datos mediante un protocolo de comunicación adecuado.

La segunda etapa corresponde a la implementación de un sistema de medición de la cantidad de leche producida por vaca en cada turno de ordeño. Los datos serán enviados a un controlador para el procesamiento de la información y finalmente, se enviarán los datos para su almacenamiento hacia el servidor que tiene la base de datos.

En la tercera etapa se diseñará un sistema de registro y base de datos donde se almacenará la información de cada animal como el registro reproductivo, registro genealógico, registro de pastoreo, registro de producción, registro de vacunación y desparasitación. Cabe señalar que el sistema de identificación del animal y de medición de leche estarán conectados directamente con la aplicación que interactúa con la base de datos.

Finalmente, en la cuarta etapa, se realizará un aplicativo entrelazado con el sistema de registro que permitirá generar el historial de la producción lechera periódicamente tanto por animal como por hato, basado en una interfaz gráfica amigable para el usuario. En la Figura **22** se observa el diagrama de bloques del sistema a implementar.

En conjunto, el sistema permitirá realizar un monitoreo exhaustivo de la producción lechera en toda la hacienda de manera eficiente, ayudando a mejorar la productividad al permitir administrar de mejor manera los recursos tanto de alimentación como de tiempo de ordeño de cada animal, con un coste de inversión accesible para los pequeños y medianos productores.

Objetivos del proyecto

Objetivo general

Diseñar e implementar un sistema para la identificación automática, monitoreo y registro del ganado que permita documentar con precisión los eventos que se produzcan en la hacienda ganadera Bellavista destinada al sector lácteo, ubicada en la parroquia rural de Lloa.

Objetivos específicos

- Diseñar e implementar un sistema de identificación del ganado basado en la tecnología inalámbrica para diferenciar de manera automática a cada animal.
- Diseñar e implementar un sistema de medición de leche mediante sensores de flujo, permitiendo conocer la cantidad de litros que produce cada vaca y por cada

turno de ordeño, para que posteriormente dicha información sea almacenada en una base de datos.

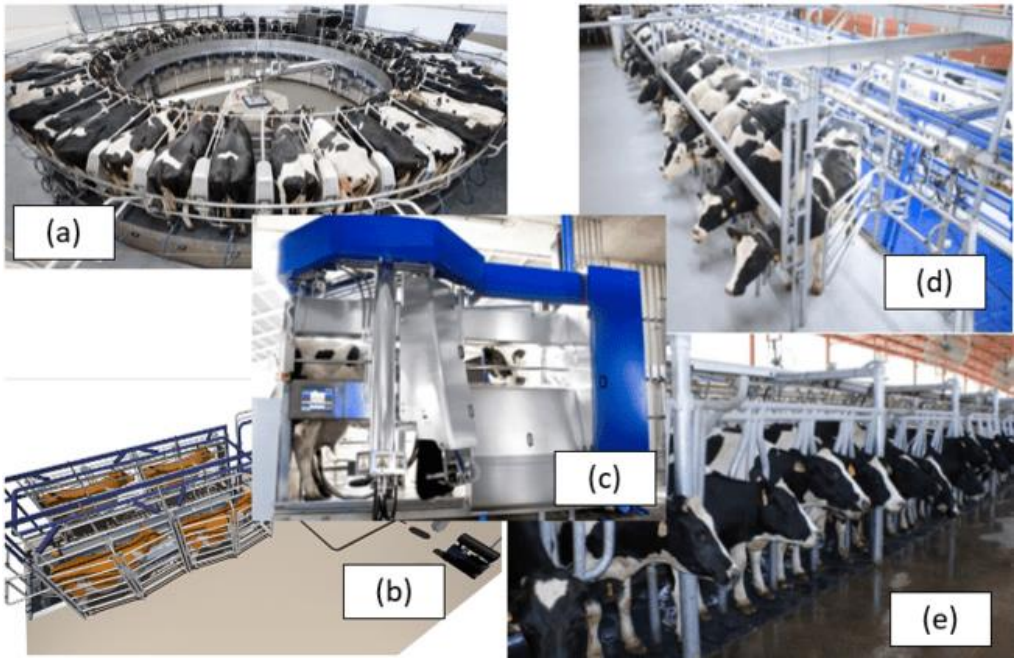
- Desarrollar un sistema de registro de todas las actividades pecuarias de la hacienda mediante una base de datos para su posterior procesamiento.
- Diseñar y realizar una base de datos de fácil acceso que permita la conexión y almacenamiento de la información de los sistemas de identificación, monitoreo y registro.
- Diseñar una interfaz amigable para que el usuario pueda visualizar la información de los registros y reportes estadísticos de cada vaca.

Capítulo 2: Marco Conceptual

En la actualidad, gracias a los avances tecnológicos, el proceso de ordeño del ganado bobino es cada vez más eficiente y eficaz. Y para dicha actividad existen varios tipos de salas de ordeño, que deben ser seleccionados de acuerdo a las necesidades de la hacienda y del propietario, estas pueden ser de tipo: espina de pescado, tándem, paralelo, rotativo, robotizado (Hernández, 2020). Todos estos casos se pueden observar mediante imágenes en la Figura 2.

Figura 2

Tipos de salas de ordeño



Nota. Tipos de salas de ordeño: (a) rotativo, (b) tándem, (c) robotizado, (d) espina de pescado, (e) paralelo. Fuente (PeruLactea, 2018).

La Hacienda en cuestión cuenta con una sala de ordeño tipo espina de pescado, al ser de este tipo quiere decir que, por cada punto de ordeño, existen dos puestos de ordeño uno a cada lado, en donde las vacas se sitúan para ser ordeñadas, de tal forma que puede haber varias variantes las cuales van desde los 12 puestos de ordeño hasta los 48. Además, este tipo

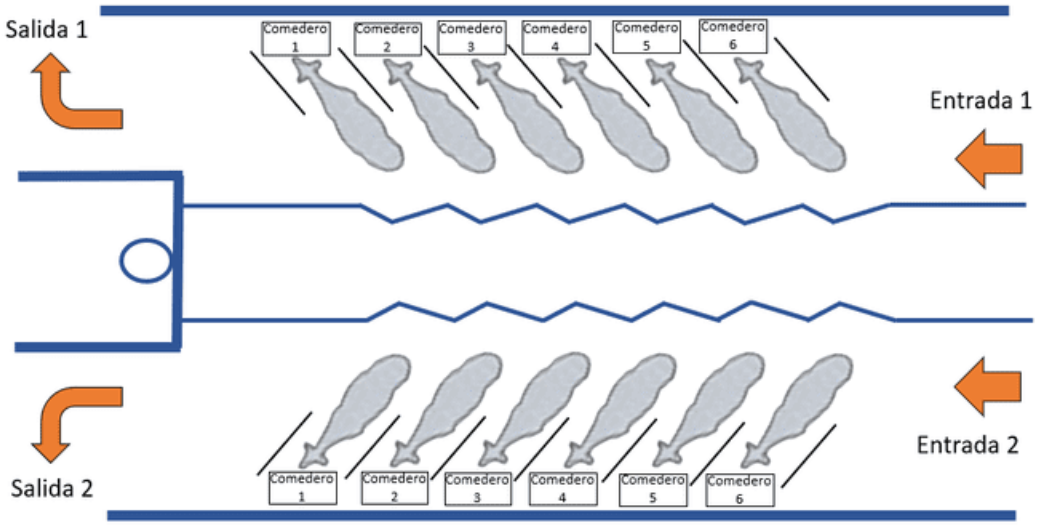
de salas de ordeño son muy versátiles, por su espacio y comodidad para el ordeño, por lo que son muy utilizadas en haciendas lecheras de mediana y grande producción (Sánchez, 2002).

Es muy importante mencionar el procedimiento de preordeño, que consiste en una serie de pasos previos al ordeño de las vacas. Primero se debe arrear a las vacas desde su ubicación en los potreros a la sala de ordeño, en donde cada vaca ingresará al puesto de ordeño. Una vez ubicada la vaca se realiza la etapa de limpiado y desinfectado a las ubres de las vacas, también se realiza el pre sellado que consiste en la inmersión del pezón en una solución de yodo o cloro. Además, se realiza el despunte, que es la expulsión de los primeros chorros de leche de cada ubre sobre un recipiente de color oscuro, con el objetivo de identificar si ha existido algún cambio de color en la leche. Todo esto se realiza para detectar posibles lesiones en las ubres y pezones de las vacas (Alvarado et al., 2019).

Ya que la infraestructura espina de pescado cuenta con un foso, la persona encargada del ordeño coloca las pezoneras en las ubres de las vacas. En la Figura 3 se observa con más detalle a este tipo de sala de ordeño.

Figura 3

Sala de ordeño de la Hacienda Bellavista

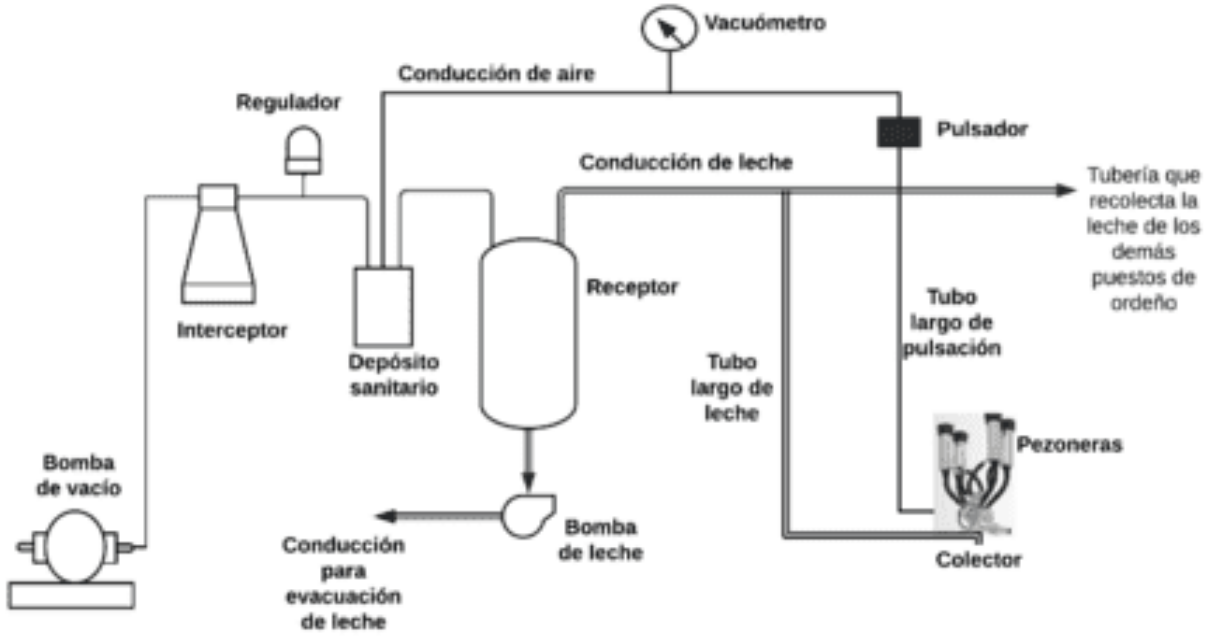


Con la sala de ordeño tipo espina de pescado y con el sistema de ordeño mecánico se tiene un proceso mucho más rápido, pero también es importante conocer cómo los pequeños productores debido a la falta de presupuesto realizan el ordeño de forma manual. Para este caso, ellos se encargan de pesar la cantidad de leche producida por las vacas, por lo que usan básculas, dicho instrumento les sirve para que mediante cálculos puedan obtener el valor equivalente en litros. Para esto se considera que la densidad de la leche es de 1.032 g/ml, por lo que, un litro pesa alrededor de 1,03 Kg que generalmente se aproxima a 1Kg (Contexto ganadero, 2017).

Dentro de las instalaciones de la Hacienda, se encuentra instalado un sistema de ordeño mecánico, cuyo funcionamiento según K. González (2018) se basa en imitar la acción de la cría de la vaca que es el de succionar la leche. Este sistema es el encargado del ordeño de las vacas, su esquema se puede observar en la Figura 4.

Figura 4

Esquema del sistema de ordeño mecánico general de la Hacienda Bellavista



Nota. Esquema y componentes del sistema mecánico de ordeño de la Hacienda Bellavista.

El funcionamiento de este sistema de ordeño inicia con el encendido de la bomba de vacío lo que hará que se extraiga el aire del sistema de ordeño, creado un vacío en el conducto principal de aire, este aire es controlado por el regulador, que se encarga de mantener el vacío constante durante el turno de ordeño. Los pulsadores, funcionan como válvulas, hacen que la entrada de aire y vacío sea alternando entre la pezonera y la portapezonera lo que emula la succión de la cría de la vaca.

Las pezoneras son las que tienen contacto directo con las ubres de las vacas, y la leche succionada se recolecta en el colector que se encuentran en la parte baja de las pezoneras. A su vez la leche es canalizada al tubo largo de leche para después ir al receptor que es el encargado de recoger la leche bajo vacío de la conducción de leche. Finalmente, la bomba de leche recoge la leche bajo vacío del receptor y la pasa a presión atmosférica para ser enviada al tanque de enfriamiento que almacena toda la producción de leche.

El encargado de indicar el nivel de vacío con el que está trabajando el sistema es el vacuómetro el cual marca el valor de 50Kpa, este debe ubicarse antes del primer punto de ordeño. El interceptor es el encargado de evitar que líquidos o sólidos externos entren en la bomba de vacío, el elemento que limita el paso de líquidos y otros contaminantes entre el sistema de vacío y la leche es el depósito sanitario, el cual se halla entre estos dos (K. González, 2018).

Por otro lado, el registro de la cantidad de producción de leche puede ser manual o automático, de ahí que, con la finalidad de realizar un registro adecuado se debe realizar la identificación de las vacas. Siendo así, en la siguiente sección se realiza un breve resumen de la importancia que conlleva tener la identificación de cada animal en una hacienda.

Identificación animal

Un factor importante en la automatización de los sistemas de ordeño y la gestión de las actividades pecuarias es la identificación animal, la cual consiste en determinar la identidad del animal. Eso se realiza mediante un código o nombre para posteriormente enlazar con un

conjunto de registros que determinan el estado del animal y la producción del mismo (Felmer et al., 2006).

Tecnologías para la identificación animal

Existen varios métodos que permiten identificar al animal, comúnmente se usan etiquetas implantadas en las orejas, los cuales no ofrecen garantía suficiente debido a que se pueden borrar las letras o números que contienen éstas, esto provocaría que la información del ganado se pierda y por ende no habría un control adecuado en las actividades ganaderas. La solución ante estos posibles problemas de identificación de los animales se ve reflejada con el uso y aplicación de las tecnologías disponibles en el mercado, las cuales contribuyen con la modernización de las haciendas. Una de estas tecnologías es el denominado sistema de identificación automática a través de la identificación por radiofrecuencia (RFID), el cual es un método inalámbrico, el cual constituye una herramienta ampliamente utilizada y que contribuye con la administración del hato ganadero (Gavilanes, 2013).

Tecnologías inalámbricas para la identificación animal

Para Negrete y Hernández (2018), las tecnologías basadas en la identificación por radiofrecuencia constituyen una de las tecnologías referentes para un sistema de identificación automática, su funcionamiento consiste en el uso de ondas electromagnéticas para la captura y/o grabación de datos entre el lector y la etiqueta. Dentro de los dispositivos que se usan en los animales para su identificación mediante este tipo de tecnologías son las etiquetas o tags, collares, bolos ruminales y cápsulas subcutáneas.

Etiquetas RFID usados para la identificación animal

Como se mencionó anteriormente, existen distintos dispositivos usados para la identificación electrónica usando tecnología RFID que pueden ser usados en el ganado, y se describirán a continuación.

Aretes RFID. También denominaos etiquetas o tags, estos son colocados en las orejas del animal. Internamente cuentan con un chip el cual une a la antena, la cual hace posible la comunicación el lector RFID (p. 21).

Collares. Son similares a los aretes RFID, con la única diferencia es que en lugar de ir en la oreja va en el cuello de animal (p. 19).

Bolos ruminales. Este método de identificación es el más costoso, pero es considerado el más confiable ya que el dispositivo es colocado dentro de un bolo cerámico, que a su vez es alojado en el retículo del animal lo que implica que no existe la posibilidad de pérdida del dispositivo (p. 22).

Dispositivos subcutáneos. Este dispositivo es inyectado en la piel del animal, cerca de la zona muscular del animal, generalmente es colocado bajo la piel de la oreja. Así mismo este método para identificación animal tiene un alto costo, pero presenta una ligera desventaja que es la migración del dispositivo a otras partes del cuerpo (p. 21).

Hasta aquí se ha enumerado los principales dispositivos que más se usan para la identificación del ganado, en donde se pudo notar que cada uno tiene una característica que le destaca del otro. Pero nos centraremos únicamente en los aretes RFID.

Tag RFID y sus componentes

Debido a la variedad de ganados como lo son porcinos, bovino, ovino e incluso animales de casa en los cuales se usan las etiquetas o tags RFID, se han convertido en los dispositivos más populares en cuanto a la identificación animal se trata. Estas etiquetas están compuestas internamente por un chip el cual se une a una antena, y en el mercado existen de distintas configuraciones, tamaños, estándares por lo que se han clasificado de acuerdo a la capacidad de memoria y su forma cómo son energizadas. Para entender más sobre los tags a continuación se detallan sus partes y en la Figura 5 se muestra los componentes.

Chip. Es el que guarda la información del animal u objeto que se quiere identificar y cualquier programación que es indicada por el lector.

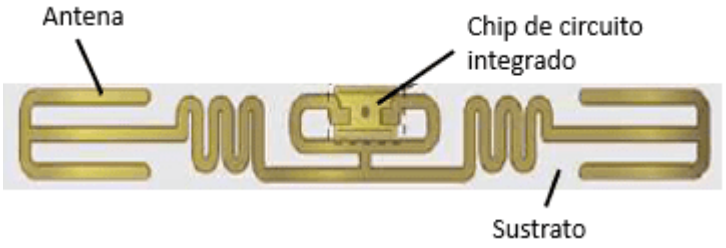
Antena. Habilita al circuito integrado para comenzar la comunicación con el lector, su sensibilidad está determinada por el tamaño y cantidad del material conductivo

Sustrato. Contiene al chip, antena y hace de ensamblaje de estos.

Encubrimiento. Es la parte externa que cubre los componentes anteriores, generalmente son de plástico y depende del fabricante su forma (Bonilla & Galárraga, 2009).

Figura 5

Componentes de una etiqueta RFID



Nota. Figura tomada de (Bonilla & Galárraga, 2009)

Clasificación de las etiquetas RFID

Las etiquetas se clasifican según la configuración de memoria y según la manera de cómo se energizan.

Según la configuración de memoria. Existen de tres tipos los cuales son:

- Sólo lectura: Su identificación es única y únicamente es personalizado durante la elaboración de la etiqueta.
- De una sola escritura: Se puede escribir el código de identificación por una sola ocasión, y después usarse como una etiqueta de sólo lectura.
- De lectura y escritura: La identificación puede ser modificada a través del lector.

Según la manera de cómo es energizada. Existen de tres tipos los cuales son:

- Pasivas: No requiere alimentación, por lo que se activan únicamente cuando el lector induce corriente eléctrica generando un campo magnético lo que hace operar al chip y a su vez transmitir una respuesta.
- Semi-pasivas: Tienen alimentación propia por lo que mantendrá energizado al circuito integrado. Responden más rápido debido a que tiene mayor velocidad de lectura que las pasivas, así mismo tiene un rango de lectura superior.
- Activas: Poseen una fuente de alimentación propia brindando energía al chip y propagando su señal al lector. Éstas tienen un rango de lectura por encima las anteriores pero su tamaño es mayor (Bonilla & Galárraga, 2009).

Lector RFID y sus componentes.

El lector RFID es el elemento que forma parte del sistema, y estos pueden ser manuales, móviles o fijos; y está constituido por los siguientes componentes.

Antenas. Es el componente más sensible de todo el sistema RFID, debido a que captura las señales de baja potencia que irradian las antenas de las etiquetas.

Módulo de radiofrecuencia. Se encarga de brindar una señal portadora al modulador y una señal de referencia al demodulador.

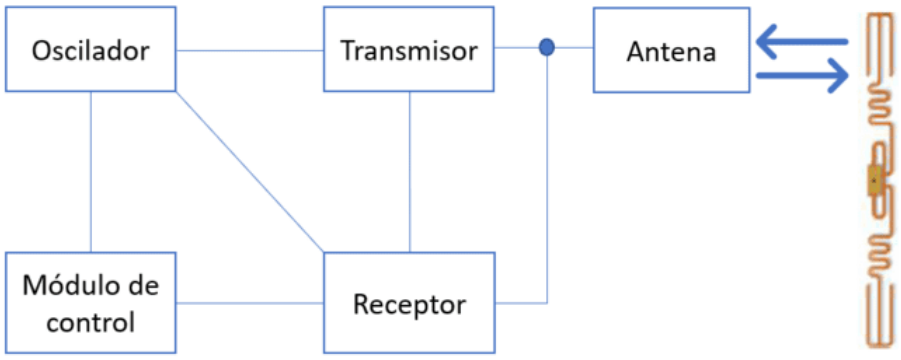
Transmisor. Crea una señal modulada que incluye los comandos y la información hacia las etiquetas RFID.

Receptor. Demodula la señal recibida por el lector y envía la información a la unidad de control.

Unidad de Control. Es el componente encargado de procesar tanto los datos transmitidos como los recibidos, recibe comandos y tiene la memoria para guardar los datos recibidos por los tags. Además, permite el control de los elementos del lector y puede interactuar con el usuario (Bonilla & Galárraga, 2009).

Figura 6

Diagrama de bloques de un lector RFID



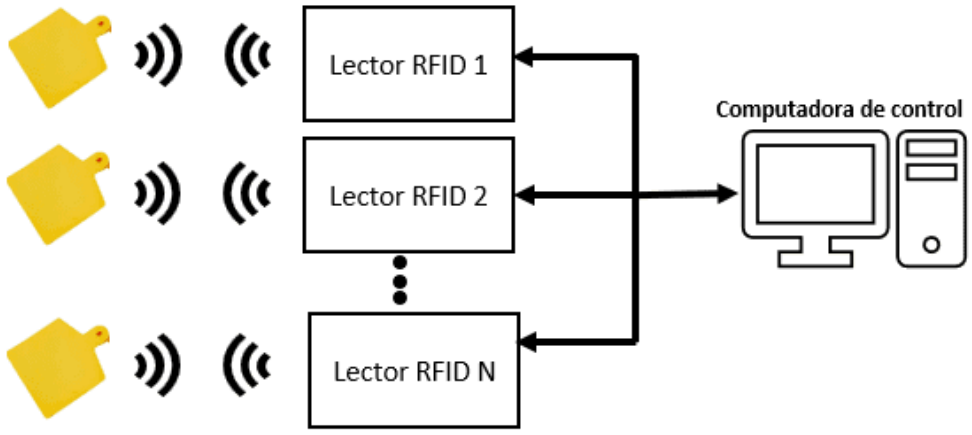
Nota. Adaptado de (Bonilla & Galárraga, 2009)

Arquitectura de la identificación animal usando tecnología RFID

Estos sistemas están conformados por la etiqueta o tag, un lector RFID y una computadora central. El lector RFID envía una señal mediante ondas electromagnéticas hacia la etiqueta, en el caso de que esta sea pasiva tomará esta señal como alimentación y responderá con el código de identificación, luego el lector RFID recibe la información y almacena en la memoria interna, finalmente la información almacenada en el lector se envía mediante un protocolo hacia la computadora como se ilustra en la Figura *7* (Bonilla & Galárraga, 2009).

Figura 7

Arquitectura y componentes RFID



Nota. En este gráfico se puede observar la arquitectura y elementos que conforman un sistema basado en RFID

Software de configuración de los lectores RFID

Es el encargado de administrar la interacción del lector con las etiquetas y del lector con la computadora, mediante este software se configura la potencia de la señal lector-etiqueta, el medio físico para la transmisión lector-computadora, la velocidad de transmisión, el modo de operación del lector y los códigos de identificación que estarán almacenados en la memoria de la etiqueta (Bonilla & Galárraga, 2009).

Medidores de flujo

Según Suhissa (2017) para la medición del volumen ya sea líquidos o gases, se debe determinar el caudal lineal o la masa, para ello se utilizan flujómetros o medidores de flujo, que son dispositivos que permiten entender el comportamiento de los fluidos. Existen muchas opciones de medidores de flujo, en donde cada uno tiene su propia funcionalidad, así como también su nivel de precisión y su precio. Por tal razón, la elección del dispositivo debe ser evaluado según los requisitos y la aplicación en específico en que se requiera usar.

Al tratarse de una aplicación cuyo fluido es leche, se debe conocer más sobre este tipo de medidor, por consiguiente, el siguiente subtema tratará sobre este dispositivo.

Medidores de flujo de leche

Existen dos tipos de medidores utilizados en los sistemas de medición de leche: el proporcional y el electrónico. Los medidores proporcionales van conectados entre el tubo de leche de cada puesto y la tubería principal, almacena en una cámara calibrada parte de la leche proporcional a la cantidad total ordeñada, un ejemplo de esto se muestra en la Figura *8*. Este tipo de instrumento son útiles en pequeñas haciendas con establos portátiles y motores a combustible, pues estos no consumen energía eléctrica.

Figura 8

Medidor de leche proporcional

Nota. En la figura se muestra un ejemplo de un medidor de flujo proporcional de la marca DeLaval.

Por otro lado, los medidores electrónicos generalmente son utilizados en salas de ordeño más automatizadas, son más precisos en la medición y la mayoría permite la comunicación con dispositivos electrónicos para una gestión automática de la producción (Contexto ganadero, 2017).

Una vez que el valor de la cantidad de leche sea obtenido por el medidor, lo ideal es enviar a un sistema de gestión que permite a los ganaderos evaluar la producción individual de leche de cada vaca, lo que contribuiría con la corrección de toma de datos y ayudaría a mejorar la productividad de las haciendas.

Sistemas de gestión de los procesos en la producción de leche

Un sistema de monitoreo de la producción de leche permite al ganadero mejorar la eficiencia de los procesos, disminuyendo el tiempo del operador en el proceso de pesaje y registro, evitando posibles errores humanos tanto en la determinación de la producción como en el registro de la producción. Estos sistemas son diseñados e implementados gracias las tecnologías presentes en el mercado y el ingenio de la humanidad que han hecho que existan sistemas que permitan medir la producción de leche, identificar el ganado y tener un registro individual y colectivo de las actividades pecuarias.

En consecuencia, un sistema automático de monitoreo de la producción de leche permite a los ganaderos evaluar la producción individual de leche de cada vaca, lo que contribuiría con la corrección de toma de datos y ayudaría a mejorar la productividad de las haciendas. Por lo que existen distintos softwares que son usados para gestionar la producción de las haciendas ganaderas. Sin embargo, debido al alto costo de inversión estos son comprados únicamente por los grandes productores.

Uno de estos sistemas de gestión de las haciendas productoras de leche es el “Software Ganadero” de la empresa Usati Ltda. Este software tiene varias funciones, como por ejemplo permite visualizar la cantidad total de animales, la producción de cada vaca y muestra gráficas para conocer la curva de producción de una vaca en un ciclo productivo. El contar con información clave permite una mejor toma de decisiones al momento de buscar alternativas para incrementar la productividad y la rentabilidad del negocio (Software Ganadero, 2021).

El sistema de gestión de los procesos en la producción permite visualizar la información a través de una aplicación de escritorio, por lo que esta es un medio muy importante para las

haciendas ya que es una interfaz en donde los usuarios podrán realizar consultas e ingresar datos, de ahí que debe ser lo más sencillo posible su interacción.

Aplicación de escritorio

Una aplicación de escritorio es un programa que es instalada en el ordenador del usuario directamente en el sistema operativo para realizar operaciones o funciones específicas en ambientes de trabajo internos a una organización ya que solo se ejecutan en el lado del cliente. El rendimiento depende de las características del hardware como la memoria RAM, memoria de video, almacenamiento interno, etc., esto es una ventaja en comparación a las aplicaciones web ya que les permite aprovechar al máximo el hardware en cuestión, además puede abordar un amplio número de aplicaciones desde aplicaciones de gestión de procesos industriales hasta acceso y manejo de base de datos, entre otras ventajas se tiene lo siguiente:

- Dispone de herramienta de desarrollo más evolucionadas y depuradas en lo que respecta a la creación de interfases o manejo de base de datos.
- Tiene mejor tiempo de respuesta ya que tiene acceso total al software y hardware del equipo.
- No depende de una conexión a internet ya que trabajan, en su mayoría, en una red local (Junquera, 2015).

Lenguajes de programación

Con la finalidad de seleccionar el lenguaje de programación más adecuado para el desarrollo de la aplicación se debe analizar varias opciones, además, tener en cuenta la popularidad de cada uno, ya que de eso depende si aún existen desarrolladores que contribuyan a dichos lenguajes, lo que consecuentemente habrá mayor cantidad de guías o tutoriales que faciliten el progreso de la aplicación de escritorio.

En la actualidad, existe una gran variedad de lenguajes de programación, y de acuerdo a la popularidad del lenguaje de programación (PYPL) que se crea analizando la frecuencia con

la que se buscan tutoriales de los lenguajes de programación en Google (Carbonnelle, 2022a), se presenta los 10 primeros del ranking en la Tabla 1.

Tabla 1

Índice de popularidad de lenguajes de programación

Rango	Lenguaje	Cuota	Tendencia
1	Python	27.61 %	-2.8 %
2	Java	17.34 %	-0.7 %
3	JavaScript	9.21 %	+0.4 %
4	C#	7.79 %	+0.8 %
5	C/C++	7.01 %	+0.4 %
6	PHP	5.27 %	-1.0 %
7	R	4.26 %	+0.5 %
8	TypeScript	2.43 %	+0.7 %
9	Objective – C	2.21 %	+0.1 %
10	Swift	2.17 %	+0.4 %

Nota. Fuente (Carbonnelle, 2022a)

Como se observa en la Tabla 1, Python es el lenguaje de programación más en auge, y su popularidad se debe a que presenta numerosas librerías que facilitan a los programadores el desarrollo de sus proyectos.

Python. Es un lenguaje de programación libre, Después, en donde Python 3.0 presenta indiscutibles mejoras y una amplia variedad de funcionalidades en lo que es programación, esto es lo que atraería la atención de programadores para la creación de sus proyectos, además de fomentar una cultura denominada Cultura de Python (Challenger et al., 2014).

Python al ser un lenguaje de programación de alto nivel y multiplataforma es uno de los más usados en el mundo de la informática, pero esto se debe a que presenta varias

características que lo hacen más popular que otros, a continuación, se enuncian algunas de ellas:

- Es Multi-paradigma (imperativo, POO y funcional).
- La sintaxis de Python es sencilla y consistente.
- Puede correr en distintas plataformas, como Windows, Linux y Mac Os.
- Apropiado para la programación de scripts y de aplicaciones con mayor tamaño.
- Alta legibilidad, dado que presenta sangrías de manera obligatoria.
- Python incluye una extensa y poderosa biblioteca, se puede decir que cuenta con una de las librerías más completas en el mundo de la programación.
- Cuenta con una comunidad muy extensa que se preocupa por su desarrollo y en promover su adopción (J. García, 2017).

Vemos que las características de este lenguaje de programación presentan son muy significativas para el desarrollo de un proyecto, para lo cual sus líneas de código se pueden realizar en un editor de texto, ya que facilita al programador la escritura y edición de código.

Editores de texto: Visual Studio Code

Los editores de texto, técnicamente son programas que permiten la escritura de los diferentes tipos de lenguaje de programación, sirven para el desarrollo y diseño de aplicaciones ya que consiente editar o crear texto sin formato, teniendo en cuenta el código fuente de los programas son textos planos (Martínez, 2019).

Existe una gran variedad de editores de texto, y de acuerdo al índice de editores de texto que se crea analizando la frecuencia con la que se busca en Google la página de descarga del entorno de desarrollo (Carbonnelle, 2022b), se presenta en los 10 primeros del ranking en la Tabla 2.

Tabla 2

Índice de editores de texto

Rango	Lenguaje	Cuota	Tendencia
1	Visual Studio	29.71 %	+1.4 %
2	Eclipse	14.42 %	-0.4%
3	Visual Studio Code	12.41 %	+1.7 %
4	PyCharm	8.51 %	+0.1 %
5	Android Studio	8.44 %	-1.2 %
6	IntelliJ	6.74 %	+0.5 %
7	NetBeans	4.96 %	-0.6 %
8	Sublime Text	3.67 %	-0.0 %
9	Xcode	3.21 %	+0.1 %
10	Atom	2.69 %	-0.6 %

Nota. Fuente (Carbonnelle, 2022b)

Vemos que Visual Studio Code es el tercer editor de texto más popular en la actualidad, que sigue atrayendo a desarrolladores de software, y con tendencia a seguir creciendo. En consideración a esto y más que nada por su versión gratuita es el que se va a usar para el desarrollo del proyecto.

Visual Studio Code

Visual Studio Code es un editor de texto totalmente gratuito y de código abierto, disponible para Windows, macOS y Linux. Permite trabajar con varios lenguajes de programación y proporciona la utilidad de descargar, además de gestionar extensiones para que el desarrollador pueda personalizar e incrementar las funcionalidades de esta herramienta. Por otra parte, Visual Studio Code brinda documentos y vídeos tutoriales para las personas que se inicien en lo que es la programación (VisualStudioCode, 2022).

Visual Studio Code es un producto enteramente distinto si lo comparamos con Visual Studio, ya que este último cuenta con versiones de pago para acceder a ciertas funcionalidades; otro aspecto que se debe considerar, es que a diferencia de Eclipse que se centra en desarrolladoras de Java, Visual Studio Code soporta varios lenguajes de programación. A continuación, se presenta algunas características relevantes:

- Cuenta con un soporte integrado para varios lenguajes de programación en donde es posible detectar con facilidad la existencia de fallas o referencias en otros lenguajes.
- Detecta si el código está incompleto.
- Presenta un buen ambiente de extensiones para otros lenguajes.
- Tiene un gran repositorio dado a que la demanda es cada vez mayor, además está conectado con Git Hub o tiene la posibilidad de conectarse cualquier repositorio.
- Su estructura es jerárquica, ya que los archivos de código se hallan en archivos y carpetas. Otra característica está el de poder abrir a la misma vez varios proyectos que contienen archivos y carpetas (RecluIT, 2021).

Visual Studio Code presenta significativas características, y con Python incluido contribuyen en el desarrollo del proyecto, pero también se debe considerar el uso de un IDE para crear la interfaz de la aplicación de escritorio. Para esto se ha seleccionado QtDesigner que es un IDE que trabaja conjuntamente con el editor de texto y lenguaje de programación ya mencionados.

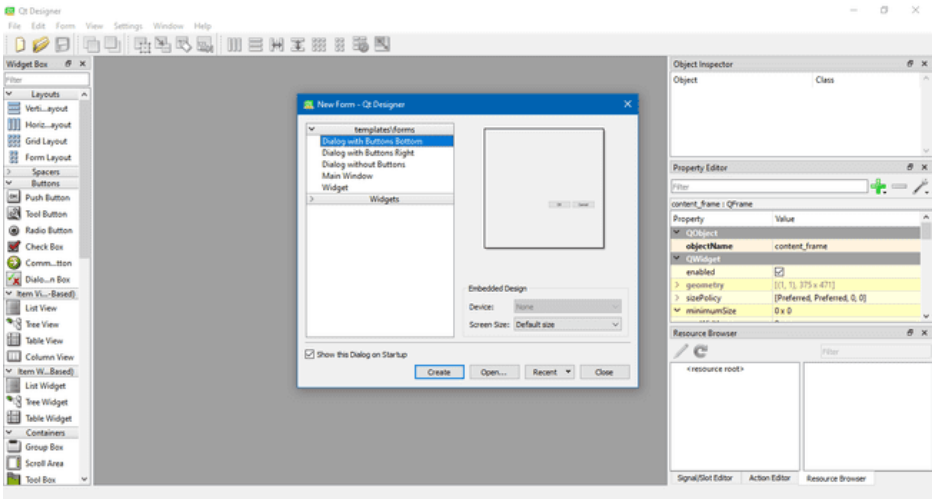
IDE QtDesigner para el diseño de la interfaz de la aplicación de escritorio

Qt Designer es un instrumento que permite realizar interfaces gráficas de usuario de manera rápida y sencilla, mediante widgets del marco Qt GUI. Su interfaz es dinámica en

donde se puede arrastrar y soltar para poder dibujar componentes como botones, cuadros combinados, campos de texto, entre otros. Qt Designer generar archivos *.ui*, que es un formato que se basa en XML el cual permite almacenar los widgets de forma jerárquica. Estos archivos pueden ser cargados o traducidos en lenguajes de programación como C++ o Python. Generalmente, Qt Designer es usado más con Python debido a que es considerado más productivo, pero para combinar los dos es necesario PySide6 o PyQt. Si se quiere instalar PyQt se debe ejecutar el comando *pip install pyqt6,* en cambio para instalar PySide6 el comando es *pip install pyside6.* En la Figura 9, se puede visualizar la interfaz de Qt Designer en Windows (Herrmann, 2020).

Figura 9

Ventana principal de Qt Designer instalado en Windows



Nota. Captura de pantalla de QtDesigner

Los elementos o widgets que permiten realizar el diseño de cualquier aplicación se describen a continuación:

- **Label:** Se usa solamente para manifestar algún texto o una imagen. Este elemento no permite la interacción con el usuario.
- **Frame:** Es utilizado para encerrar y agrupar más widgets, lo que permite una mejor visualización para el usuario.

- **GroupBox:** Se utiliza para agrupar elementos que pertenecen a una misma categoría.
- **LineEdit:** Permite la interacción con el usuario, ya que admite ingresar y editar una línea de texto sin formato.
- **TableWidget:** Proporciona la vista de una tabla la cual se basa en componentes de un modelo predeterminado.
- **PushButton:** Proporciona un botón de comando, en donde el usuario podrá dar clic en el botón para ordenar a la computadora que realice alguna acción.
- **HBoxLayout:** Aseguran que los widgets que se ubican dentro de los frames queden alineados horizontalmente.
- **VBoxLayout:** Aseguran que los widgets que se ubican dentro de los frames queden alineados verticalmente
- **ComboBox:** Es un botón el cual al presionar se despliega una lista con las opciones para que el usuario seleccione.
- **DateEdit:** Permite editar la fecha en el mismo widget (QTCompany, 2022).

Una vez ya desarrollado la interfaz, se debe añadir una base de datos que permita almacenar los registros de producción de leche, así como de otros aspectos en la Hacienda para su posible revisión u observación de información, con la finalidad de evitar pérdidas de datos.

Base de datos

Una base de datos es un conjunto de información o de datos organizados de manera estructurada, generalmente son almacenados de manera electrónica a través de un sistema informático. La base de datos es controlada por un sistema de administración de bases de datos (DBMS), que, junto a los datos y a las aplicaciones asociadas con ellos, se conocen como un sistema de base de datos, que comúnmente se le conoce como base de datos. Todos estos datos se modelan normalmente en filas y columnas en una serie de tablas para que su

procesamiento y su consulta de datos sean eficientes. De esta manera es fácil acceder a la información, además de organizar y modificar todos los datos que se hayan registrado. Las bases de datos en su mayoría usan un lenguaje de consulta estructurado (SQL) para poder escribir y consultar los datos. SQL es un lenguaje de programación desarrollado por IBM en los años 70 en donde Oracle también participó como uno de los principales creadores, y que en la actualidad es usado por numerosas bases de datos (Oracle, 2022).

Siempre se ha hablado que existe un parecido entre la base de datos y la hoja de cálculo (Microsoft Excel) ya que ambos constituyen formas de almacenar información, pero existen diferencias en el cómo se guardan y manejan los datos, en quién puede entrar a ver a la información, y también en la cantidad de información o datos que se puedan almacenar.

Existen algunos tipos distintos de bases de datos, en donde es importante el uso que se le da a los datos, a continuación, se describen los tipos de bases de datos más usados:

- Base de datos relacionales, esta base de datos brinda la manera más flexible y eficiente de permitir el acceso a la información estructurada.
- Base de datos orientadas a objetos, cuya información se representa en forma de objetos, tal como en la programación orientada a objetos.
- Base de datos distribuidas, la información o datos se pueden almacenar en varias computadoras que pueden ubicarse en el mismo lugar físicamente o dispersas en distintas redes.
- Bases de datos de código abierto, tal como indica su nombre su código fuente es de código libre.
- Bases de datos en la nube, es uno de los tipos de base de datos más recientes y la información se almacena en la nube, en este caso se tiene dos modelos el uno que es tradicional y el otro se denomina base de datos como servicio

(DBaaS), con este último las tareas de mantenimiento y tareas administrativas son realizadas por un proveedor de servicios (Oracle, 2022).

MySQL. MySQL inicialmente fue desarrollado por la compañía MySQL AB en 1994, después pasó a manos de Sun Microsystems en 2008, finalmente Oracle adquiere a Sun Microsystems en 2010 y desde esa fecha ha sido propiedad de Oracle. Se define como un sistema de gestión de bases de datos relacionales (RDBMS) de código libre que se basa en un modelo cliente-servidor, en otros términos, es un software usado para la creación y administración de bases de datos del tipo relacional.

En el modelo cliente-servidor, los clientes son las computadoras, las cuales tienen instalado y ejecutan los RDBMS. Cuando se requiere acceder a los datos, los clientes se conectan al servidor RDBMS. Basado en este modelo, existen algunas aplicaciones web muy conocidas como por ejemplo Facebook, YouTube y Google, todas esas usan MySQL para el almacenamiento de datos e información (Bustos, 2022).

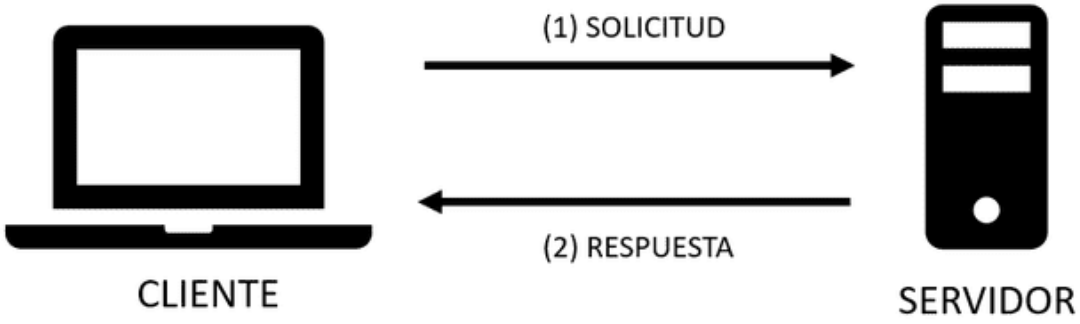
Como se mencionó anteriormente, el cliente o los clientes son quienes pueden usar una interfaz gráfica de usuario para realizar una solicitud y conectarse con el servidor, en cambio el servidor es el que producirá la salida deseada, siempre y cuando ambas partes comprendan la instrucción. En la Figura 10 se ve representado una estructura básica del modelo cliente-servidor, el cual es usado por MySQL. Los procesos más importantes que realiza un entorno MySQL son:

- MySQL crea la base de datos almacenar y manipular datos a través de tablas que a su vez están definidas por relaciones entre sí.
- Los clientes realizan las solicitudes a través de instrucciones SQL específicas en MySQL.
- La aplicación del servidor va a responder a la solicitud con la información y aparecerá en los clientes

MySQL es el más popular para los desarrolladores después de Oracle Database que ocupa el primer lugar, esto debido a que es seguro, flexible y fácil de usar, es multiplataforma, y es de código abierto (Bustos, 2022).

Figura 10

Estructura básica cliente-servidor



Nota. Adaptado de (Bustos, 2022)

Reconocimiento de imágenes con Python

Para llevar a cabo el desarrollo del sistema de medición de la producción lechera de la hacienda se optó por realizar un reconocimiento de caracteres de los displays de siete segmentos del medidor de flujo, esto se detalla en el capítulo 3, por ello es importante mencionar algunas metodologías y herramientas de la visión artificial y el aprendizaje autónomo. Para el reconocimiento de imágenes u objetos existen varios métodos y librerías que pueden ser usados por los lenguajes de programación, a continuación, se indicarán los más conocidos y más utilizados en la detección de imágenes o reconocimiento de caracteres.

Procesamiento de imágenes con OpenCV

OpenCV (Open Source Computer Vision) es una librería de código abierto, el cual contiene implementaciones de más de 2500 algoritmos disponibles de manera gratuita ya sea para fines académicos o comerciales. Es parte de la inteligencia artificial, específicamente se basa en la visión artificial, la cual permite a las máquinas y sistemas ver imágenes de manera parecida a como es la visión humana, también permite el identificar y procesar las imágenes

para ser analizadas y medir datos que posteriormente servirán para supervisar procesos y por ende tomar decisiones. Gracias a la extensa variedad de algoritmos, es capaz de realizar algunas tareas como las que se enumera a continuación:

- Encontrar imágenes parecidas.
- Reconocer escenarios.
- Identificar objetos.
- Reconocimiento facial.
- Clasificar acciones humanas que se encuentren en videos.
- Extraer modelos 3D.
- También es útil para aplicaciones en la robótica y la realidad aumentada (Rodríguez, 2021).

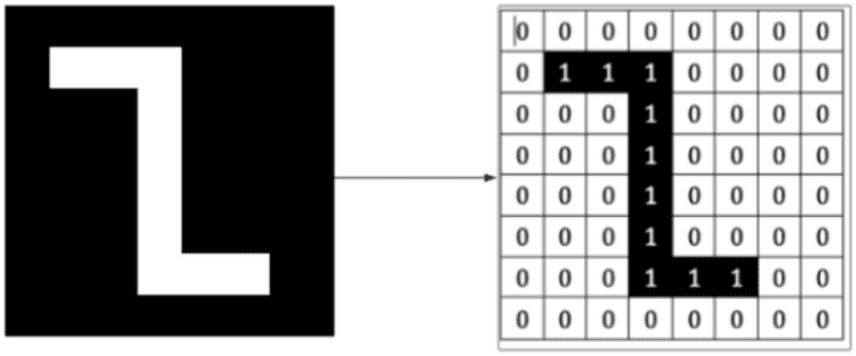
Esta librería tiene interfaces para algunos lenguajes de programación siendo Python uno de los más importantes. OpenCV-Python es la API para OpenCV en Python, y es multiplataforma. Para poder usar esta librería hay que considerar la librería Numpy, para su instalación se debe ejecutar en el editor de texto de Python el comando *pip install opencv-python* y si se requiere módulos adicionales se debe ejecutar *pip install opencv-contrib-python.*

El por qué se debe considerar la librería Numpy, es debido a que para Numpy una imagen es una matriz que contiene píxeles como puntos de datos, que, para ser procesado por una computadora, dicha imagen se debe convertir en una forma binaria que consta de 1 bit/píxel lo que conlleva a que solo puede tener dos colores posibles que son el blanco o el negro, como se puede observar en la Figura 11 de la cual se observa la representación de la imagen en una matriz donde el valor 1 representa el color blanco y el 1 representa el negro. También hay que tomar en cuenta que una imagen a color es representada como una combinación de los colores rojo, verde y azul (RGB) por lo que los demás colores se obtienen

al mezclar estos colores, además, aquella imagen a color consta de 8 bits/píxel lo que podría representar 256 tonos de distintos colores, lo mismo que una imagen en escala de grises pero que puede representar 256 sombras diferentes que son útiles para evaluar la neutralidad de los tonos reproducidos (Marín, 2020).

Figura 11

Imagen a blanco y negro que consta de 1 bit/píxel y su representación en una matriz



Nota. Fuente (Marín, 2020)

TensorFlow

TensorFlow es una de las principales librerías de código abierto usada para el aprendizaje automático a través del entrenamiento de modelos, que fue lanzado en 2015 por el equipo de Google Brain. Engloba una extensa variedad de algoritmos y modelos de aprendizaje profundo y aprendizaje automático.

Su funcionamiento se basa en crear gráficos de flujo de datos, estructuras que indican la manera con la que los datos se transportan a través de un gráfico, para esto las márgenes de la gráfica es la matriz de datos multidimensional (tensores) que se comunican entre sí y donde cada nodo del gráfico es una operación matemática. Para esto TensorFlow proporciona sus funcionalidades al desarrollador a través de Python, el cual que es el lenguaje más usado para esta librería (Yegulalp, 2022).

Gracias a sus características como el poder detectar predicciones, distinguir patrones y clasificar objetos, TensorFlow ha sido considerado en distintas áreas de aplicación como lo son la medicina, en la fotografía, procesamiento de imágenes, entre otras(L. González, 2021).

Tesseract

Antes de hablar de Tesseract, se debe a que hace referencia el término OCR (Optical Character Recognition) que en español significa Reconocimiento Óptico de Caracteres, OCR es un método que consiste en la transformación de una imagen en 2D que contiene texto en texto descifrable para la máquina o computadora. Para lo cual es necesario pasar por algunos subprocesos como el preprocesamiento de la imagen, la localización del texto, la segmentación, el reconocimiento y por último el posprocesamiento (Zelic & Sable, 2022).

Tesseract es un instrumento de OCR que fue creada por Hewlett-Packard Labs en los 90 en donde inicialmente era muy poco usado y conocido, pero a partir de 2005 se da el libre acceso para que los usuarios puedan usar de manera gratuita y a la vez se da apertura a la compatibilidad con algunos lenguajes de programación. Lo último de Tesseract es la incorporación de un enfoque basado en la Inteligencia Artificial con el objetivo mejorar la eficiencia para obtener el reconocimiento de diferentes textos y de distintos tamaños de manera eficiente (Solis, 2022).

El funcionamiento de Tesseract OCR en Python tiene como primer paso, el de convertir en binaria a la imagen la cual es analizada para posteriormente almacenar los contornos de las palabras o texto, éstos se unen como blobs que constituyen líneas de texto y se separan según los caracteres que han sido detectados del texto inicial; el siguiente subproceso es el reconocimiento en donde se pasa por un filtro adaptativo las palabras que han sido examinadas, luego se hace la identificación de las palabras no reconocidas anteriormente. Por último, se realiza la corrección de espacios difusos con el objetivo de mostrar en la computadora o máquina el texto de la imagen procesada. Una vez conocido a detalle todo el proceso para la extracción de la imagen en texto, es conveniente mencionar que se usa el

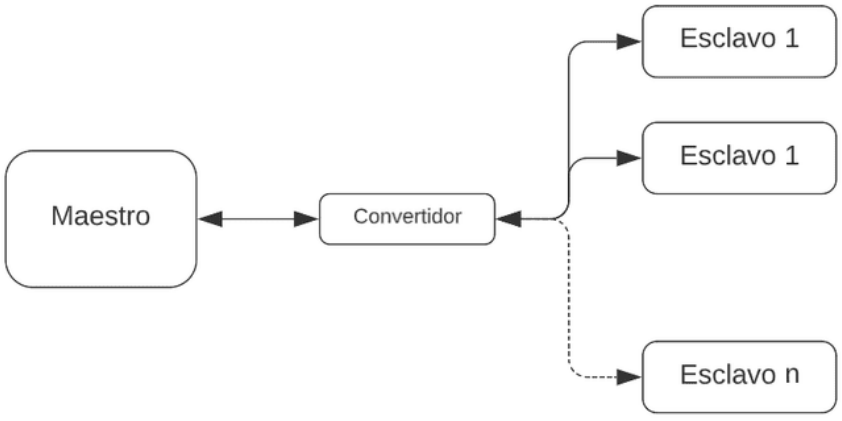
comando *pip install pytesseract* para hacer uso de esta herramienta dentro del entorno Python (Solis, 2022).

Integración de los sistemas

En la integración de sistemas hacemos referencia al conjunto de hardwares y protocolos que permiten la conexión e interacción de varios sistemas entre sí, en donde cada sistema es considerado como parte modular que compone al sistema total o integral, y además, cada uno de ellos debe tener algún dispositivo que los controle y a la vez permita la conexión entre todos ellos de manera física y también mediante algún protocolo de comunicación. En la Figura 12 se observa el muy conocido modelo maestro-esclavo, el cual es un esquema simple de comunicación entre los sistemas que se requiere implementar, en donde para iniciar dicha comunicación se requiere que el maestro envíe cierta información para comenzar con la conexión con el resto de dispositivos, además es el que tiene las funciones de administrar la red y realizar peticiones como la escritura o lectura de registros principalmente, en cambio el esclavo es quien responde a dichas peticiones.

Figura 12

Modelo de comunicación maestro-esclavo



Existe una gran variedad de controladores disponibles en el mercado siendo los más populares los PLCs que comúnmente son usados para aplicaciones industriales, otras opciones menos robustas son los microcontroladores. Además de los ya mencionados, existen las

minicomputadoras, las cuales pueden hacer el papel de controlador, dicho esto, se ha establecido que de este tipo serán los que se utilizará en este proyecto. Por tal razón en el siguiente punto se menciona las minicomputadoras más populares y las características principales que presentan cada una.

Minicomputadoras o computadora de placa simple SBC

Las minicomputadoras cuyo término es el más conocido, son las computadoras de placa simple cuyo acrónimo en inglés Single Board Computer (SBC). Estas consisten en una placa pequeña en donde se encuentra toda la circuitería principal para su funcionamiento, aquí se encuentran la CPU, la memoria RAM, los distintos periféricos de entrada/salida y más. En la actualidad existen numerosas computadoras de placa simple en el mercado, haciendo que desarrolladores o programadores así como también investigadores las usen debido a las prestaciones que presentan y la variedad de usos, además de las aplicaciones que se pueden realizar con estas (Roca, 2021).

Con los años, la tecnología ha ido evolucionado paulatinamente lo que ha permitido desarrollar numerosas SBC's de distintos fabricantes, lo que consecuentemente ha hecho que muchos desarrolladores e investigadores como (Johnston et al., 2018), (Baiza, 2020) y (Márquez, 2020) escojan entre todas ellas la que más se apegue a la aplicación que quieran desarrollar.

Raspberry PI 3

La Raspberry Pi 3 de la Figura 13 es básicamente una minicomputadora que pertenece a la familia de Raspberry, la cual constituye una herramienta para desarrollo de proyectos de software. Sus características son:

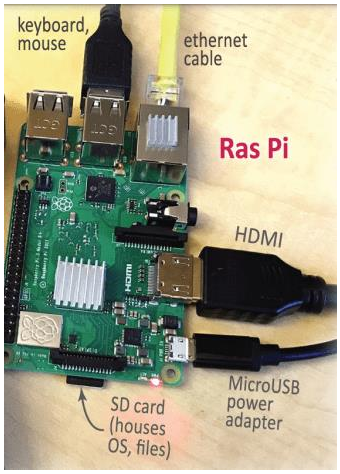
- Presenta 4 puertos USB y un puerto HDMI.
- Tiene 40 pines de uso general denominados GPIO.
- Su sistema operativo de libre acceso Raspbian.

- Su procesador ARM de 64 bits de 1,2 GHz.
- Tiene una memoria RAM de 1GB
- Posee una ranura para una tarjeta microSD
- Tiene una interfaz Ethernet (Fletcher & Mura, 2019).

Interfaz Ethernet (Fletcher & Mura, 2019).

Figura 13

Raspberry PI 3 modelo B



Nota. Fuente (Fletcher & Mura, 2019)

Libre Computer AML-S905X-CC

La placa AML-S905X-CC, también denominada *la patata*, es una SBC que puede ser usada para numerosos proyectos que requieran altas exigencia. Para tener una idea de esta placa, se tiene la Figura 14 en donde se observa la estructura y el color de esta. Sus características principales son:

- Cuenta con una CPU de clase ARM Cortex-A de alto rendimiento.
- Consume la mitad de potencia que una Raspberry Pi3
- Tiene núcleos cuádruples de bajo consumo de 64 bits,

- Presenta una GPU penta core 3 y un motor de vídeo Amlogic AVE10.
- Su superficie cuenta con 40 pines compatibles, los cuales son compatibles con UART, SPI, PWM, I2C y GPIO.
- Funciona con los sistemas operativos Debian, Armbian, Raspbian, Ubuntu Mate (LibreComputer, 2021).

Figura 14

AML-S905X-CC (La patata)

Nota. Fuente (LibreComputer, 2021).

Odroid XU4

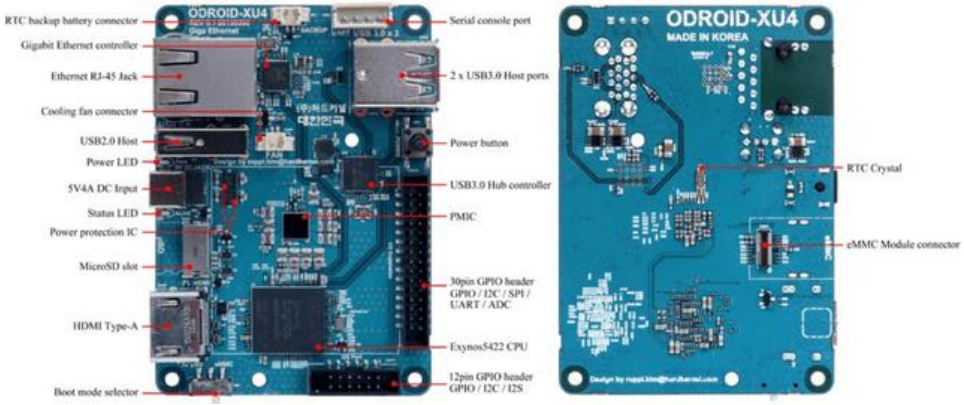
La Odroid-XU4 es una minicomputadora con altas prestaciones, se trata de una plataforma de desarrollo de software, así como también de hardware. Para conocer más sobre esta SBC se presenta la Figura 15, en donde se puede observar las partes que la integran y su estructura. Sus características principales están:

- Cuenca con un procesador octa-core a 2 GHz.
- Tiene 2GB de RAM con su almacenamiento flash integrado.
- Cuenta con 3 puertos USB, de los cuales dos puertos son USB 3.0 y un puerto USB es 2.0

- Posee una salida HDMI y un puerto Ethernet, sin olvidar que también presenta GPIOs.
- Los sistemas operativos que son compatibles para esta minicomputadora son Android, Ubuntu y Debian (Rojo, 2016).

Figura 15

Odroid XU4



Nota. Fuente (Rojo, 2016)

Interfaz y protocolos de comunicación

Para realizar la comunicación entre los controladores se consideran dos partes fundamentales; la interfaz física y el protocolo de comunicación. La interfaz es el medio físico por donde fluyen los datos y el protocolo de comunicación es el conjunto o aglomeración de reglas con las cuales se puede transmitir los datos e información entre varios dispositivos que conforman una red (Estrada, 2018).

Interfaz RS232

Esta interfaz permite la transferencia confiable de corto alcance de manera desbalanceada y punto a punto para cada señal en una comunicación, además de que su velocidad de transmisión máxima es hasta los 230,4 kbps, lo que representa menor velocidad en comparación con el RS485 (Weis, 2020a). En la Tabla 3 se puede observar los parámetros más importantes de la interfaz RS 232.

Tabla 3

Parámetros de la interfaz RS232

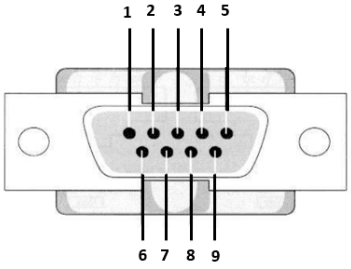
Parámetro	RS232
Configuración de la línea	De un solo extremo
Modo de funcionamiento	Simplex o Full Dúplex
Longitud máxima de cable	50 ft
Velocidad máxima de datos	230,4 kpbs
Niveles lógicos típicos	± 5 a 15 V
Impedancia de entrada mínima del receptor	3 a 7 Ω
Sensibilidad del receptor	± 3 V

Nota. Adaptado de (Weis, 2020a)

La norma TIA/EIA Rs-232 especifica las características del conector 9 pines, ver Figura 16. Para control industrial generalmente el puerto RS232 utiliza únicamente los pines 2, 3 y 5.

Figura 16

Interfaz RS232 para la comunicación serial



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Nota. Este gráfico presenta la interfaz reducida a 9 pines de la norma TIA/EIA Rs-232. Fuente (Shenzhen, 2018).

Interfaz RS485.

Es una interfaz de comunicación parecido al RS232 para la implementación de comunicaciones de datos seriales, es muy usado para entornos industriales porque permite servir a varios dispositivos que se encuentran conectados al mismo bus, además presenta inmunidad a ruidos eléctricos (Weis, 2020b). Es parte del estándar de comunicaciones de la capa física del Modelo OSI (Cachumba, 2019). El estándar RS485 permite tener un enlace de hasta 32 dispositivos, además, contiene un tercer estado que hace que un dispositivo pueda quedarse en alta impedancia (Caballero & Zambrano, 2018). En la Tabla 4 se puede observar los parámetros más importantes de la interfaz RS 485.

Tabla 4

Parámetros clave de la interfaz RS485

Parámetro	**RS485**
Modo de operación	Diferencial
Modo de funcionamiento	Simplex o Half Dúplex
Longitud máxima de cable	1200 m
Velocidad máxima de datos (Baudios)	460,8 kbps
Niveles lógicos típicos	± 1.5 a ± 6 V
Impedancia de entrada mínima del receptor	12 Ω
Sensibilidad del receptor	± 200 mV

Nota. Adaptado de (Weis, 2020b).

El RS-485 no tiene un conector específico, sin embargo, puede ser conectado en los conectores de 9 pines antes mencionado. Al hablar de la interfaz RS485 es necesario indicar que hoy en día existen convertidores de USB a RSd485 que utilizan el circuito integrado SN75176 para convertir los niveles de tensión, lo cual permite enlazar dispositivos con bus

RS485 a un computador de puertos USB, este dispositivo es un adaptador como el que se puede observar en la Figura 17.

Figura 17

Adaptador de USB a RS485

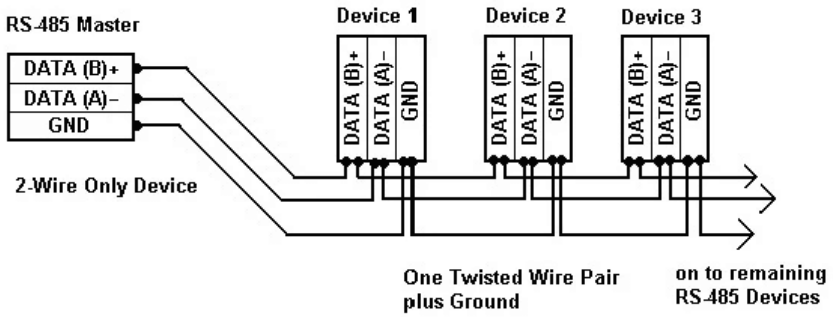
Nota. Fuente ((Weis, 2021)

En la conexión de una red RS485, se utilizan un cable trenzado con dos o cuatro hilos, el primero permite una comunicación half-duplex, en cambio el de cuatro hilos permite full-duplex. Para el primer caso los cables representan uno para datos y el otro para datos invertidos; cuyas funciones son de transmitir la señal original y el otro envía su copia invertida, de este modo existe una mayor resistencia ante interferencias. El de 4 hilos usa dos pares trenzados lo que permite una conexión punto a punto (Weis, 2021). La conexión serie RS485 se puede observar en la

Figura 18.

Figura 18

Conexión serie RS485



Nota. La figura representa la conexión serie RS485, en donde se puede observar al maestro conectado con tres dispositivos esclavos. Figura tomada de (Weis, 2021).

Modbus

Es un protocolo desarrollado por Modicon a finales de los años 70, diseñado para la comunicación de controladores con unidades de adquisición de datos, lo cual le llevó a ser el protocolo usado en la industria por su facilidad de uso y confiabilidad. Modbus usa una topología Maestro-Esclavo, donde el Maestro es el que maneja la comunicación mediante el envío y la recepción de mensajes (Mejía, 2016).

Los registros y peticiones que se basan en la arquitectura Modbus están conformados por un inicio del mensaje, dirección del registro que se requiere, función de lectura o escritura que se requiera, cantidad de datos que se debe leer, apartado para la verificación de errores y su final (B. García, 2021).

La relación maestro-esclavo es propia del protocolo Modbus que es un protocolo de solicitud-respuesta, dicha comunicación se produce en pares, donde un dispositivo debe iniciar una solicitud para después esperar una respuesta, de aquí que el dispositivo de inicio es el maestro el cual generalmente es una interfaz humano-máquina y el esclavo es un sensor o un controlador (Huayta, 2018).

Existen dos modos de transmisión que pueden ser aplicados al standard Modbus; el modo RTU (Remote terminal Unit) y el modo ASCII (American Standar Code for Information Interchange), estos especifican los bits contenidos en una trama, además determina el empaquetado y la decodificación de los datos a ser enviados (Torres, 2021).

Modo de transmisión RTU

Si un controlador está configurado para trabajar en modo RTU los mensajes son transmitidos en flujo continuo en grupos de 8 bits, dos dígitos de 4 bits, debido a la densidad de carácter tiene mayor rendimiento que el modo de ASCII trabajando a la misma velocidad.

El sistema de codificación es binario de 8 bits, números hexadecimales 0-F. Por cada byte de mensaje se tiene; 1 bit de arranque, 8 bits de datos, 1 bit de paridad par o impar, 1 bit de paro si se utiliza paridad o dos bits de paro si no se usa y una Comprobación Cíclica Redundante.

En cuanto a la trama, la transmisión inicia con un intervalo silencioso de aproximadamente 4 ciclos de carácter (T1, T2, T3, T4), luego se transmite el campo correspondiente a la dirección del dispositivo, los dispositivos conectados decodifican para verificar si es su dirección, luego se transmite continuamente la función, los datos y el CRC y luego de 4 ciclos de carácter se finaliza la transmisión, esto se representa en la Tabla 5.

Tabla 5

Trama del mensaje RTU

INICIO	DIRECCIÓN	FUNCIÓN	DATOS	CRC	FINAL
T1-T2-T3-T4	8 bits	8 bits	N° 8 bits	16 bits	T1-T2-T3-T4

En este capítulo se trató el marco conceptual cuya información sirve de cierto modo como herramienta que brinda soporte para el desarrollo del proyecto, consecuentemente en el

siguiente capítulo se realizará el diseño de cada etapa como tal, y que finalizará con la integración de estos.

Capítulo 3: Diseño

Debido a que el diseño e implementación de este proyecto se realizará como prototipo en la Hacienda Bellavista ubicada en la parroquia rural de Lloa. En este capítulo se iniciará con el levantamiento de la información de la planta para luego proponer el diseño del proyecto en 4 etapas:

- Identificación automática del ganado, en el cual se definirá la tecnología de identificación que se va a emplear y se dimensionará los equipos.
- Medición automática de leche, el que se encarga de medir y transmitir la información medición de la cantidad de leche que produce cada vaca.
- Selección y diseño de gestor de base de datos, sistema que permitirá almacenar los diferentes registros de las actividades pecuarias de la Hacienda.
- Integración de los componentes anteriores. En esta fase, adicionalmente se desarrollará una interfaz gráfica para que los usuarios puedan observar a través de esta, los datos almacenados.

Levantamiento de la planta y requisitos del diseño del sistema

La Hacienda Bellavista está ubicada a 2 km del pueblo de Lloa, cuenta con 152 cabezas de ganado. Actualmente, cada animal está identificado mediante un arete de plástico, donde consta el número que representa el id del animal y su nombre. Esta identificación se realiza al momento de la compra del animal o se su nacimiento. De las 152 cabezas de ganado, al momento 52 vacas están en etapa de producción. Este número fluctúa debido a que muchas de ellas salen para prepararse para un nuevo parto y otras ingresan al rejo una vez que han parido. Las vacas que se encuentran en rejo son ordeñadas mediante un sistema de ordeño mecánico instalado en la sala de ordeño de tipo espina de pescado, la cual puede observarse en la Figura 19.

Figura 19

Sala de ordeño de la Hacienda Bellavista

Con la finalidad de tener un registro por cabeza de ganado, se ha instalado en cada unidad de ordeño un medidor electrónico de flujo de leche que cuenta con un display que muestra la cantidad de flujo total de leche que pasa por el medidor, ver Figura 20. Cuando la leche ingresa en el canal del medidor, esta es detectada por porciones a través de unos electrodos que envían las señales captadas a la tarjeta electrónica interna del equipo quien mediante un algoritmo calcula la cantidad de flujo y muestra la cantidad de litros medida a través de los displays de siete segmentos de la pantalla del medidor. Este sensor tiene una precisión de ±3% tras una calibración adecuada, la cuál es bastante alta en medidores de flujo. Dentro de las características técnicas dadas por el proveedor se tiene que su tensión de funcionamiento es de 10 a 27 VCC, tiene un consumo eléctrico máximo de 5,5 W y cuenta con cuatro cables, el cable de color marrón sirve para el suministro eléctrico, el blanco para la tierra, el morado y el azul representa a la entrada y salida respectivamente del bucle de corriente. Su funcionamiento está dado de la siguiente manera, al encender el equipo la pantalla muestra los parámetros de fábrica, después, en la pantalla se muestra P-UP que indica que el medidor de

flujo está realizando la autocomprobación, una vez terminado esta acción está listo para la medición de la leche que circula en su canal (DeLaval, 2021).

Figura *20*

Medidor de flujo de leche Fi7 de la marca DeLaval

Nota. Fuente (DeLaval, 2021)

Previo el proceso del ordeño de las vacas, es necesaria realizar una serie de actividades que se detallan a continuación:

1. Se enciende la bomba de vacío a través de un tablero de control.
2. Se procede a limpiar las mangueras de leche a través de una bomba de agua.
3. Se enciende los medidores de leche a través del tablero de control.

Una vez realizada estas actividades se procede a realizar el ordeño, este proceso se encuentra detallado en la siguiente sección.

Proceso durante el ordeño

La sala de ordeño al ser espina de pescado presenta 12 puestos de ordeño, con 6 puestos de ordeño en cada lado. La sala cuenta con 6 sistemas de succión y medición, es decir un par de puestos comparten el mismo sistema y por tanto solo 6 animales pueden ser ordeñados de manera simultánea. De manera que el proceso se realiza de la siguiente manera.

Seis vacas ingresan a los puestos de ordeño del primer lado, una vez ubicadas una tras otra en cada sitio la persona encargada del ordeño realiza el procedimiento de preordeño.

En esta etapa de preordeño, una de sus actividades es la de verificar enfermedades de las ubres y pezones, seguido a esto, se procede limpiar los pezones, para luego proceder a colocar las pezoneras. Estas últimas hacen el trabajo de succionar la leche, que a través del colector y el tubo de leche llega al canal del medidor, lo que permite medir los litros producidos y son mostrados en la pantalla. Cuando las pezoneras ya no succionen más leche, estas son retiradas por la persona encargada del ordeño, quién procede posteriormente a sellar los pezones de las vacas para evitar mastitis.

Acto seguido, otra persona es la encargada de llevar el registro manual del valor indicado en la pantalla del medidor. Posteriormente esta medida es manualmente ingresada a un archivo en Excel que lleva el administrador de la hacienda. Finalmente se presiona el pulsador del punto de ordeño para reiniciar el medidor electrónico a cero y poder efectuar una nueva medida.

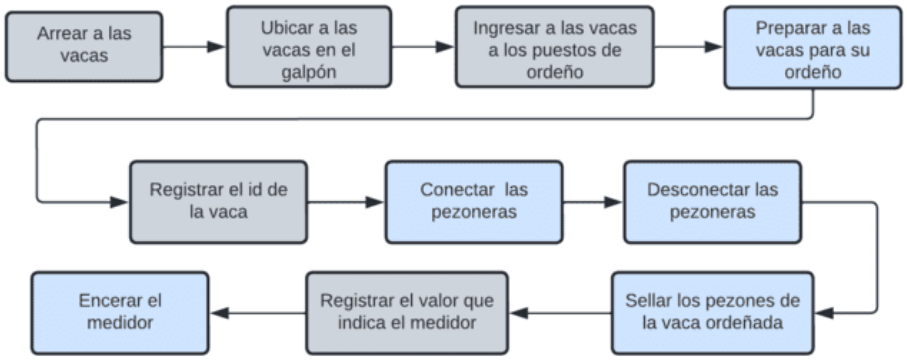
Mientras las vacas del primer lado se encuentran en proceso de ordeño, se hace pasar un segundo grupo de 6 vacas al segundo lado para realizar el proceso de preordeño. Una vez que las pezoneras se encuentran libres del puesto compartido del otro lado, se procederá a colocar las pezoneras a la vaca del segundo lado para iniciar el proceso de ordeño, de registro manual de datos y de sellamiento de pezones.

De manera similar mientras las vacas del segundo lado están en proceso de ordeño se ingresan un nuevo grupo de vacas al primero lado para realizar el proceso de preordeño y los pasos subsiguientes. Este procedimiento se repite y continua hasta que todas las vacas destinadas a la producción sean ordeñadas. Este procedimiento se resume en la Figura 21. En el caso que una vaca presente mastitis u otra enfermedad, esta es separada del resto de vacas para ser ordeñada al final y evitar contaminar el resto de leche.

Para realizar el registro de producción manual, el empleado de la hacienda lo realiza mediante el número de identificación del arete que lleva cada vaca con el valor de la cantidad de litros producidos que se muestra en el medidor. La leche que produce el hato es almacenada en el tanque reservorio, de esta manera se almacena en el tanque toda la producción de leche del ganado por turno de ordeño. Una vez finalizado el ordeño se procede a realizar una medición manual del tanque reservorio.

Figura 21

Proceso para el registro de producción



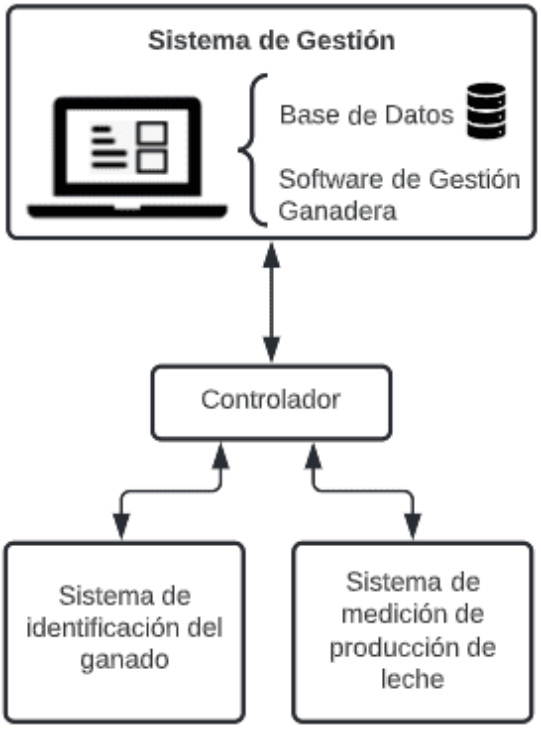
Nota. Se muestra el proceso de las actividades que se realizan durante el ordeño de las vacas, en donde el color gris representa a que es ejecutado por una persona, en cambio el color azul es realizado por otra.

Como se puede observar en la Figura 21 se requiere de dos personas para este procedimiento. Siendo una tarea sensible el registro de producción manual, considerando que se realiza esta tarea mientras se ejecutan otras tareas. Esto puede causar registros erróneos. Por lo tanto, para cumplir con el objetivo del presente proyecto se propone diseñar e implementar un sistema de identificación automática, que relacione la identificación con la medición del sensor de flujo de manera automática. Es decir, se diseñara e implementará un sistema que capte los datos de los sensores, relaciona la identificación y envié la información codificada a un gestor de base de datos. Adicionalmente, este sistema de gestión permitirá

almacenar otros registros importantes del ganado facilitando la administración de la hacienda ganadera. Para ello, se ha planificado el diseño e implementación de una interfaz de usuario que permita buscar, filtrar y desglosar la información particular que requiera el administrador. La Figura 22 muestra el esquema general del sistema planteado siendo el controlador el elemento integrador del sistema de identificación y medición que va a interactuar con el gestor de datos.

Figura 22

Diagrama de bloques del sistema



Diseño del sistema de identificación del ganado

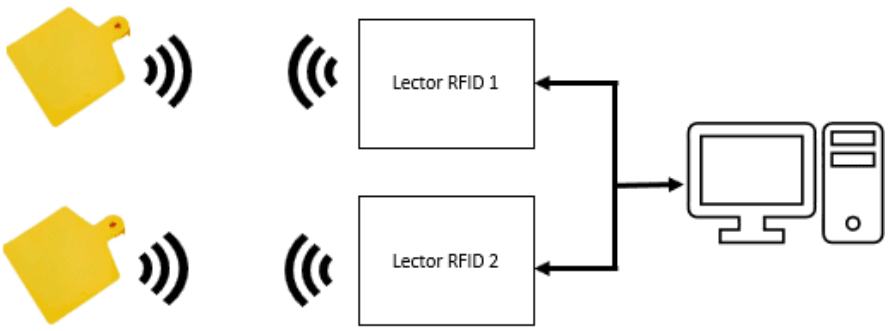
El sistema de identificación del ganado constituye una parte esencial del proyecto, puesto que permite hacer un control individual del ganado mediante la elaboración de registros tanto de producción como de las actividades pecuarias de la hacienda. Por tal motivo es muy importante seleccionar adecuadamente la tecnología y metodología para el diseño.

Selección de la tecnología empleada

Para el sistema de identificación automática se ha elegido la tecnología RFID, debido a las características mencionadas en el capítulo anterior. Por consiguiente, se hará uso de la arquitectura de la identificación animal mediante tecnología RFID que se indica en la Figura 23, con el objetivo de definir el lector y el tag RFID.

Figura 23

Arquitectura y componentes RFID.



Nota. En este gráfico se puede observar la arquitectura y elementos que conforman un sistema basado en RFID

En el mercado actual existen numerosas marcas y proveedores de estos equipos, pero su elección depende de la aplicación y lugar de instalación. En este caso, teniendo en cuenta que el lector va a realizar la lectura únicamente de las vacas que ingresan al ordeño y que deben ser dos los dispositivos que se deban adquirir debido a la infraestructura de la sala de ordeño que cuenta con dos entradas, estos lectores RFID deben ser fijos. Teniendo en consideración la compatibilidad con los tags, el alcance, la versatilidad y el software de configuración que proporciona Yanzeo, además de que su costo en comparación con los denominados bastones RFID es considerablemente mejor, se ha seleccionado al lector RFID SR681.

El lector RFID SR681 que se ve en la Figura 24 es un lector UHF de alto rendimiento para operaciones de largo alcance de hasta 6 m. Este lector tiene un rango de frecuencia de

865 MHz a 928 MHz y fue diseñado para la lectura de etiquetas de identificación con el protocolo ISO18000-6C y también ISO18000-6B. El lector puede ser aplicado para muchos sistemas de aplicaciones RFID, uno de estos es que puede ser usado para un sistema de identificación automática para animales de granja (Yanzeo, 2022). En la Tabla 6 se puede observar las características que presenta el lector de esta marca.

Figura 24

Lector RFID SR681 de la marca Yanzeo

Nota. Fuente (Yanzeo, 2022).

Tabla 6

Características principales de lector SR681

Descripción	Detalles técnicos
Protocolo de interfaz área	ISO 18000-6B, ISO 18000-5C
Indicador de activación	Parpadeo de luz LED o pitido
Rango de lectura	> 6 m
Frecuencias de operación	865 - 928 MHz
Tasa de transmisión	9,2 a 57,6 kb/s
Antena	Antena de polarización circular
Modo de trabajo	Modo activo, modo pasivo, modo comando
Interfaces de comunicación	RS232, RS485, Wiegand26, Wiegand34, RJ45, WIFI

Dimensiones (alto, ancho, largo)	227, 60, 227 mm
Tensión de trabajo	+ 12 V
Potencia de salida	0-30 dBm (ajustable)
Temperatura de funcionamiento	- 20°C - + 70°C
Humedad de trabajo	20% - 95%
Software	Kits de desarrollo en software

Nota. Fuente (Yanzeo, 2022)

En el caso particular del proyecto, con la finalidad de identificar a los animales en la sala de ordeño y dada la distribución física de la misma, es necesario colocar dos lectoras una a cada entrada de la sala. Cabe recalcar que al configurar el sistema es importante considerar el alcance y el lugar de instalación de los lectores, con el fin de que no haya interferencias en la lectura de los tags. Para la instalación se debe considerar la funcionalidad de cada uno de los pines y cables de la lectora. La Tabla 7 resume la funcionalidad de cada cable del lector RFID.

Tabla 7

Funcionalidad de los cables del lector

Color	**Descripción**
Rojo	+12V DC
Negro	GND
Marrón	TXD (Pin 2)
Amarillo	RXD (Pin 3)
Azul	GND (Pin 5)
Gris	Trigger
Verde	DATA0
Blanco	DATA1
Morado	485+

Naranja	485-

Nota. Fuente (Yanzeo, 2022)

Por otra parte, para configurar el sistema es necesario seleccionar las etiquetas o tags RFID que van a ser colocadas en el animal. De las varias opciones que hay en el mercado, se ha seleccionado un tag de la misma marca que del lector, ya que se complementan perfectamente porque trabaja en el mismo rango de frecuencias, en la Figura *25* se puede observar la imagen del tag.

Figura 25

Tag RFID de la marca Yanzeo

Nota. Fuente (Yanzeo, 2022).

La etiqueta RFID de Yanzeo es una etiqueta UHF de alto rendimiento para operaciones de largo alcance de hasta 6 m, dependiendo de su lector. Fue diseñada con el protocolo ISO18000-6C para su lectura, y esta debe ser ubicada en la oreja del animal para su identificación, en la Tabla 8 se muestran las principales características.

Tabla 8

Características del tag de Yanzeo

Descripción	**Detalles técnicos**
Protocolo de interfaz área	ISO 18000-6B, ISO 18000-6C
Rango de lectura	<= 6 m
Frecuencias de operación	860 - 960 MHz

Dimensiones	5 x 5 cm
Material	PET y material exterior TPU
Tipo	Pasiva
Tamaño de memoria	512 bits
Chip	Alien H3
Modo de funcionamiento	Lectura/Escritura
Temperatura de funcionamiento	-20 - +50 °C

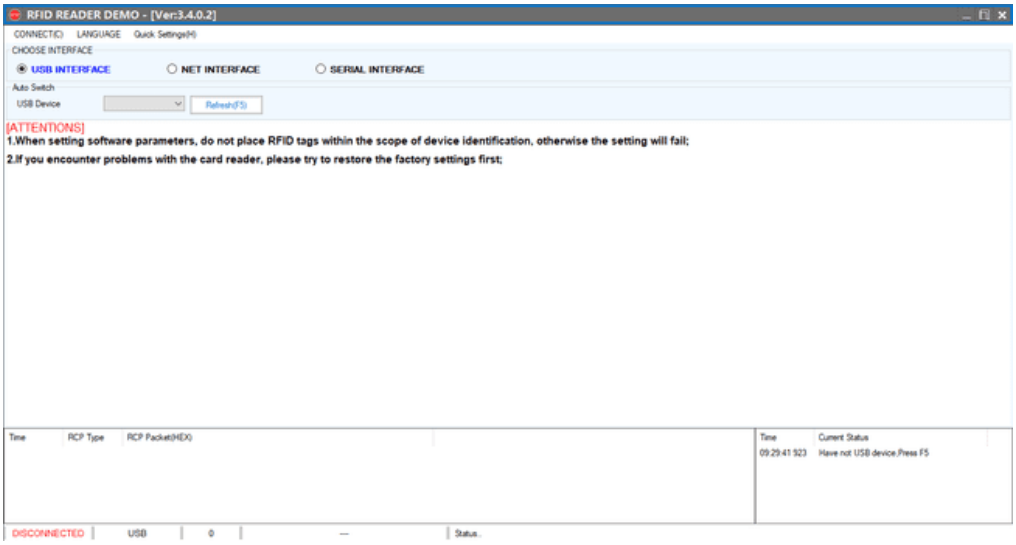
Nota. Fuente (Yanzeo, 2022)

Software para la configuración del lector SR681

El software que proporciona Yanzeo es el RFID Reader Demo, el cual se puede descargar gratuitamente de la página oficial de la empresa, además en la misma página web se tiene a disposición el manual para la configuración del lector RFID. Este software permite seleccionar la interfaz física con la que se conecta el lector, además al ser configurable se puede escoger la velocidad de transmisión de datos. En la Figura 26 se muestra la pantalla de inicio.

Figura 26

Pantalla de inicio del software RFID Reader Demo



Configuración del sistema

Modos de operación del lector. El lector RFID SR681 de Yanzeo dispone de tres modos de funcionamiento, los cuales se detallan a continuación:

- **Activo:** El lector trabaja de manera automática, es decir, al momento de detectar una etiqueta RFID envía el dato directamente al host.
- **Pasivo:** Permite realizar la lectura de los tags únicamente cuando la computadora envía el comando al lector.
- **Comando:** En este modo de trabajo el lector funciona una sola vez cuando recibe un comando del host. Este modo es muy útil en este proyecto ya que evita colisiones en la red, de esta forma el host puede comunicarse simultáneamente con dos lectores evitando que estas transmitan la información de las etiquetas al mismo tiempo (Yanzeo, 2022).

En este proyecto se va a utilizar el modo comando. El lector puede recibir comandos de solicitud de datos de la memoria o comandos de configuración como: encender o apagar el buzzer o el led, cambiar la potencia de la antena, escribir en la memoria del tag, entre otras. Independientemente del comando que se emplee, la trama de datos que se transfieren debe seguir un protocolo que se explica a continuación.

Protocolo de comunicación del lector RFID SR681. En este proyecto se va a utilizar el modo comando. El lector puede recibir comandos de solicitud de datos de la memoria o comandos de configuración como; encender o apagar el buzzer o el led, cambiar la potencia de la antena, escribir en la memoria del tag, entre otras. Independientemente del comando que se emplee, la trama de datos que se transfieren debe seguir un protocolo, este protocolo es propietario y tiene el siguiente formato.

Tabla 9

Formato básico de la trama

No.	1	2	3	4	5	6	7
Byte	1	2	1	1	1	LONGITUD	1
Registro	SOI	ADR	CID1	CID2	LONGITUD	INFO	CHECKSUM

Nota. Fuente (Yanzeo, 2022)

Los registros que se presentan en la Tabla *9* se describen a continuación:

- **SOI:** Inicio de la información. En donde 7C representa al comando y CC es usado para respuesta.
- **ADR:** Dirección del dispositivo (identificación de lector), hasta 65535, 0 reserva.
- **CID1:** Define la función del comando (descripción del tipo de datos).
- **CID2:** Comando: Código de identificación de la función (descripción del tipo de acción). Para la respuesta del lector: RTN (código de retorno, donde 00H significa *éxito*, 01H representa *falla*, 02H es el mensaje de respuesta para el mando y 05H es el envío automático a la *unidad de supervisión*.
- **LONGITUD**: Longitud de los datos.

- **INFO:** Corresponde al *dato* que se envía o recibe. En el caso de una solicitud se enviará la información de la contraseña y dirección del registro de la memoria de almacenamiento, y en el de una respuesta corresponde a la información del tag, Está formado por 4 registros, *AP* (Accsess Password) que está conformado por 32 bits, *MB* (Memory Bank) que se refiere al banco de memoria 0, *SA* (Starting Address) es el puntero de palabra de dirección inicial y *DL* (Data Length) es la dirección de la memoria *EPC* (Electronic Code of Products).
- **CHKSUM:** Es el código de suma de comprobación, permite al lector saber si los datos enviados por el host fueron leídos correctamente. Se obtiene primero, realizando la suma de los bytes de los demás registros; después a este resultado se le convierte a decimal y se realiza el módulo con 256, luego se le convierte a binario para realizar el complemento a dos, finalmente se transforma a hexadecimal (Yanzeo, 2022)..

A continuación, se presentan los comandos empleados en este proyecto para realizar la lectura de una etiqueta RFID.

Tabla 10

Comandos de lectura de la etiqueta RFID

Función del comando	Valor del CD1	Valor del CD2
Leer tipo C UII	20H	00H
Leer datos de la etiqueta tipo C	21H	00H
Establecer estado de coincidencia	2DH	00H

Nota. Fuente (Yanzeo, 2022)

Conformación de la trama para la solicitud de lectura de un tag. Los valores que conforman la trama de datos para realizar una solicitud de lectura de tags RFID desde la PC hacia el lector se describe en la Tabla 11.

Tabla 11

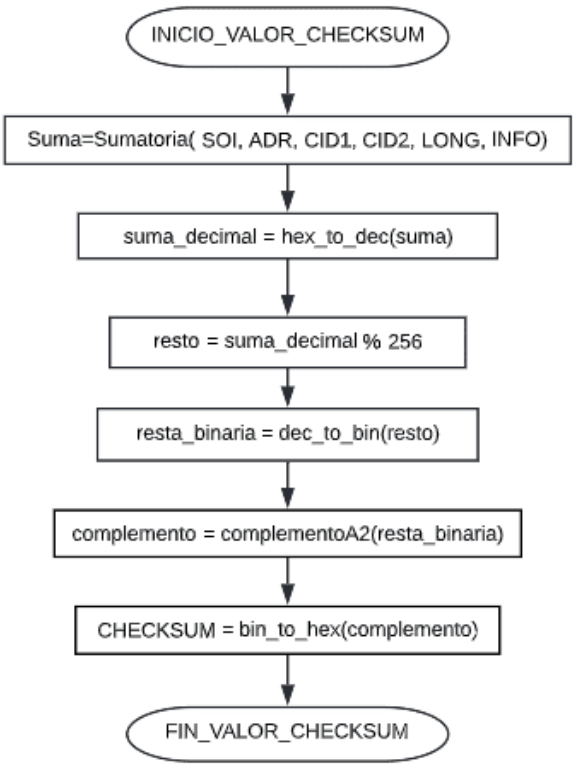
Conformación de la trama para el lector 1

Registro	**Lector 1**	
SOI	7CH: Debido a que es una trama de solicitud.	
ADR	0001H: Es el número de identificación del lector 1	
CID1	21H: Valor del CID1 que permite leer datos de la etiqueta.	
CID2	00H: Valor del CID2 que permite leer datos de la etiqueta.	
LONGITUD	07H: Para tener un mayor rango de almacenamiento de datos.	
INFO	AP	00000000H*:* Debido a que los lectores no poseen contraseña
	MB	01H: Se va a leer la memoria EPC, que es donde se almacena la información del id del animal.
	SA	02H: Puntero de inicio de datos.
	DL	02H: Dirección de la memoria EPC.

Para el cálculo del valor del registro de verificación (*CHKSUM)*, se sigue los pasos que se describe en el diagramade flujo de la Figura 27.

Figura 27

Diagrama de flujo del proceso para la obtener el valor de CHECKSUM



El cálculo de *CHECKSUM* para el lector, ver Figura 27, se muestra a continuación:

- Suma de los demás registros

  (7C+01+00+21+00+07+00+00+00+00+01+02+02)H = AAH

- Que en decimal corresponde al valor de 170

- Al realizar el módulo 170 % 256 = 170

- Dicho valor convertido a binario es: 10101010

- Realizando el complemento a dos: 1010110

- Después dicho valor se convierte en hexadecimal, lo que es igual a 56H. Este sirve para verificar si no ha existido errores en la transferencia de datos de este lector.

Finalmente, en la Tabla 12 se puede observar todos los valores de los registros para el caso del lector 1.

Tabla 12

Conformación de la trama de solicitud de lectura de tags para el lector 1

SOI	**ADDR (LSB)**	**ADDR (MSB)**	**CID1**	**CID2**	**LONG**	**AP (MSB)**
7C	01	00	21	00	07	00
--	**--**	**AP (LSB)**	**MB**	**SA**	**DL**	**CHECKSUM**
00	00	00	01	02	02	56

Para el lector 2 la trama es igual al mostrado en la Tabla 12, con la única diferencia del registro ADR, que en este caso será 0002H. El valor del *CHKSUM* también cambia y se calcula a continuación:

- Suma de los demás registros

  (7C+02+00+21+00+07+00+00+00+00+01+02+02)H = ABH

- Que en decimal corresponde al valor de 171
- Al realizar el módulo 171 % 256 = 171
- Dicho valor convertido a binario es: 10101011
- Realizando el complemento a dos: 01010101

Finalmente, el valor obtenido se transforma a hexadecimal, consiguiendo el número 55H. Este valor se usa para verificar el éxito o error en la transferencia de datos por parte de este lector.

Una vez definidos los registros de la trama, el resultado se observa en la Tabla 13.

Tabla 13

Conformación de la trama de solicitud de lectura de tags para el lector 2

SOI	ADDR (LSB)	ADDR (MSB)	CID1	CID2	LONG	AP (MSB)
7C	02	00	21	00	07	00
--	**--**	**AP (LSB)**	**MB**	**SA**	**DL**	**CHECKSUM**
00	00	00	01	02	02	55

Interfaz de comunicaciones. El lector permite la comunicación a través de la interfaz RS-232 y RS-485. La interfaz que se utilizó para la comunicación es el RS-485 debido principalmente a la inmunidad al ruido que este posee. Esto debido a que en la sala de ordeño se encuentran componentes que podrían alterar la señal enviada por el lector hacia la CPU central y a la opción que permite tener varios dispositivos distribuidos en una red. Además, el RS485 puede ser utilizado para distancias de hasta 1200m con una mayor tasa de transmisión que el RS-232.

Diseño de la arquitectura del sistema de identificación

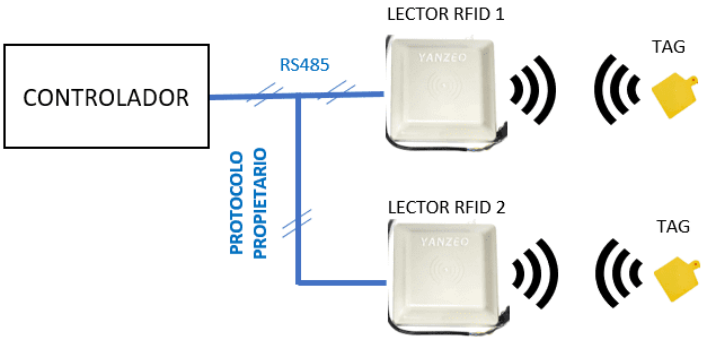
Considerando que la sala de ordeño es de tipo espina de pescado y se tiene dos entradas por donde ingresan las vacas para ser ordeñadas, se requiere de un lector RFID por entrada, estos se denominarán lector RFID 1 y lector RFID 2 respectivamente. Los lectores se instalarán de manera fija, al inicio del pasillo de entrada, ubicados sobre el pasillo para poder sensar tanto si el tag está ubicado en la oreja izquierda o derecha del ganado.

Para la comunicación se trabajará con el modelo maestro-esclavo, como se indica en la Figura ***28***, el maestro es el controlador, que es el que controla el tráfico de datos en el bus y el que solicita información de las etiquetas a los lectores RFID que funcionan como esclavos. Y

esto se puede observar en la Figura 28 donde se ha diseñado un diagrama para el sistema de identificación animal.

Figura 28

Diagrama del sistema de identificación automática del ganado

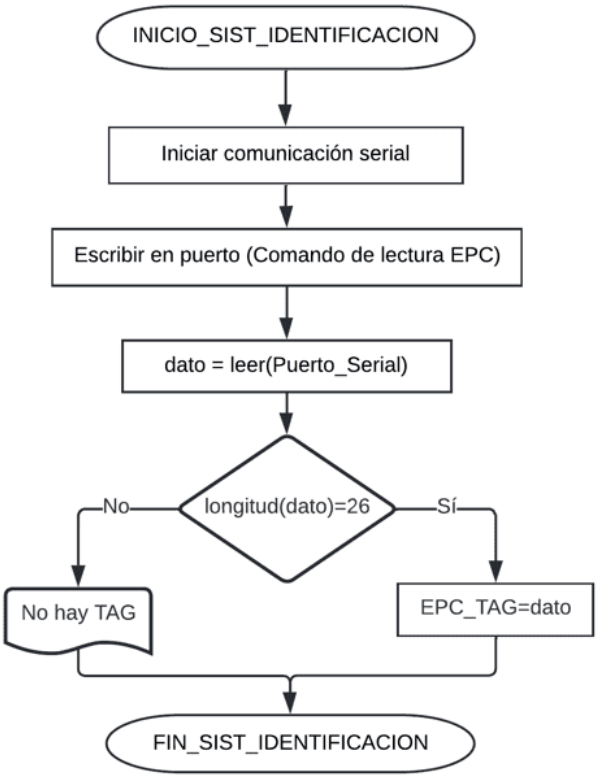


Diseño del software del controlador

El primer paso para lograr un funcionamiento adecuado del sistema de identificación automática es iniciar la comunicación serial entre la CPU central (controlador) y los lectores RFID 1 y 2. Una vez inicializada la comunicación, el controlador envía el comando de lectura EPC a través del puerto serial utilizando interfaz RS-485 hacia los lectores, ambos reciben el mensaje y verifican la dirección del dispositivo. En el caso de que corresponda a su dirección el lector RFID, este leerá todos los bytes del comando y responderá con un valor de 26 caracteres dentro de los cuales consta el dato de la etiqueta RFID detectada. En el caso de que el lector RFID 1 o el lector RFID 2 no ha detectado ningún tag envía una respuesta de error. En la Figura 29 se puede observar el diagrama de flujo del procedimiento para la detección de los tags que se va a implementar en el controlador del sistema. Esta rutina de detección es similar para cualquiera de los dos lectores RFIDS y ha sido implementada en el controlador para identificar cada cabeza de ganado que ingresa a la sala de ordeño.

Figura 29

Diagrama de flujo para la identificación automática del ganado



Diseño del sistema de medición automático de la producción de leche

Uno de los objetivos del proyecto es conocer la producción individual de la vaca en cada turno de ordeño, para lo cual se requiere identificar el animal que va a ser ordeñado y medir la producción de leche del animal en cuestión. El sistema de identificación ha sido abordado en la sección anterior, en tanto que en esta sección se hará referencia a la medición automática de leche. Para ello se utilizará la información entregada por los medidores de flujo de leche instalados en la sala de ordeño de la hacienda. Como se indicó anteriormente un medidor es compartido por dos puestos de ordeño, uno de cada ala de la configuración en espina. Originalmente, el proyecto pretendía tomar la información del medidor a través del transmisor de cada medidor. De acuerdo a la información otorgada por el fabricante, cada medidor entrega la información en in protocolo de lazo de corriente. Los pines del medidor se describen en la Tabla 14.

Tabla 14

Descripción de pines del medidor de flujo

Color del cable	Descripción
Marrón	Suministro eléctrico (+)
Blanco	Tierra
Púrpura	Entrada del bucle de corriente
Azul	Salida del bucle de corriente

Luego de varios intentos fallidos de lograr determinar y descifrar el protocolo de comunicaciones y de una larga búsqueda en la literatura, se pudo determinar que este dispositivo emplea el protocolo propietario ALCOM. Sin embargo, el fabricante no entrega ninguna información al respecto ya que vende soluciones completas con equipos controladores de su un sistema cerrado. A continuación, se ejemplifica una de las pruebas realizadas.

Para esta prueba se consideró que el estándar de bucle de corriente era de 4 a 20 mA. Por tal motivo se realizó un conversor de corriente a tensión considerando los siguientes parámetros:

$$I_{min} = 4\ [mA]\ , I_{max}\ = 20\ [mA] \quad \text{y} \quad V_{max} = 5\ [V]$$

$$R = \frac{V_{max}}{I_{max}}. \quad \text{reemplazando los valores conocidos} \quad R = \frac{5\ [V]}{0.02\ [A]} = 250\ [\Omega]$$

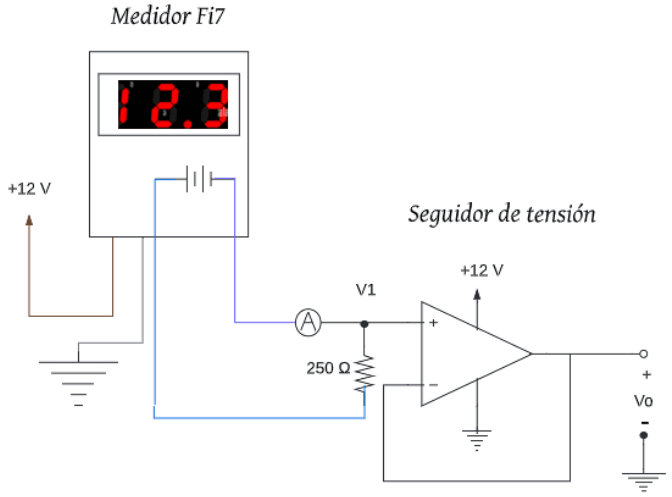
$$\text{Entonces el valor de } V_{min} = R \cdot I_{min}$$

$$V_{min} = 250\ [\Omega] \cdot 0.004\ [A] = 1[V]$$

Así la tensión obtenida tendría un rango de 1 a 5 V. Adicionalmente, se implementó un seguidor de tensión para realizar el acoplamiento de impedancia. Luego se implementó un sistema de pruebas para emular el ordeño, haciendo circular leche por el medidor. El sistema de pruebas utilizaba una bomba de líquido de tres velocidades (V1, V2, V3) que se muestra en la Figura 30.

Figura 30

Circuito de prueba para la obtención de la señal de salida del medidor de flujo

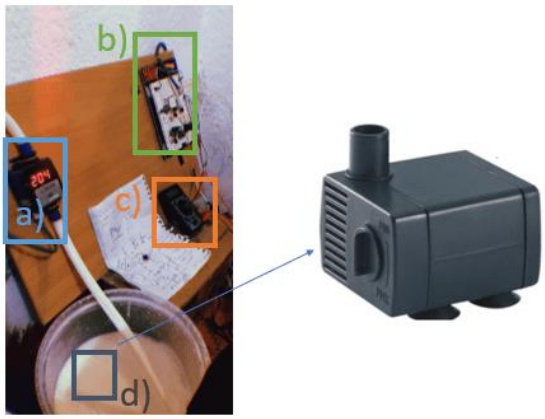


En la Figura 30 se presenta el circuito implementado. Después de haber realizado numerosas mediciones se obtuvo lo siguiente:

- El amperímetro A muestra un valor aproximado de 0 mA cuando no existe flujo de leche, por lo tanto, en la salida del amplificador operacional se muestra un valor de 0 V.
- El amperímetro A muestra un valor aproximado de 14 mA cuando la bomba está funcionando a la velocidad mínima (V1) y la tensión en la salida del amplificador es 3.5 V.
- Para las velocidades V2 y V3 de la bomba se tuvieron los mismos valores de corriente y tensión que para la velocidad V1.

Figura 31

Prueba para la obtención de la señal de salida del medidor de flujo de leche



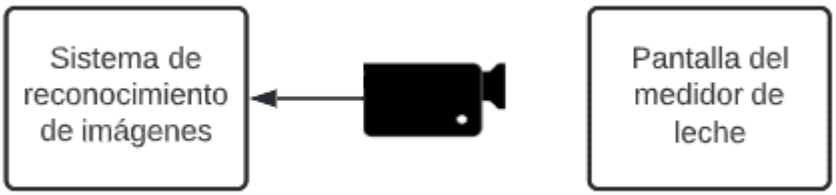
Nota. La imagen representa a) medidor, b) circuito seguidor c) amperímetro d) bomba de agua sumergida en el balde de leche

Adicionalmente se procedió a observar las señales en un osciloscopio, pero tampoco hubo variación entre las diversas cantidades de leche medidas. También se buscó otras opciones en el mercado para verificar otros tipos de protocolos, pero no se tuvo éxito. Dados estos inconvenientes y con la finalidad de obtener la medición de cada sensor de flujo en tiempo real se optó por realizar el diseño e implementación de un sistema de reconocimiento de imágenes de los caracteres mostrados en el displays de cada medidor.

El sistema de reconocimiento requiere de una cámara por medidor para captar la imagen del display. Esta imagen será procesada por un dispositivo electrónico que ejecute la rutina de reconocimiento. La Figura 32 ilustra el diagrama de bloques por medidor para el reconocimiento de los caracteres del display del medidor.

Figura 32

Diagrama de bloques para el diseño del sistema de medición de la producción de leche



Para el sistema de reconocimiento de imágenes se debe seleccionar adecuadamente el lenguaje de programación, el controlador y la manera de visualizar el dato recopilado. Esto con el objetivo de obtener el valor que indica el medidor sin ninguna interferencia o tipo de error ya que implicaría un mal monitoreo de la producción.

Obtención de medida por reconocimiento de imágenes

Selección del lenguaje de programación. Para esta aplicación el sistema de reconocimiento de imágenes debe ser en tiempo real, confiable y de bajo costo lo que exige una selección adecuada del lenguaje de programación y el controlador. Dadas las ventajas que ofrece el lenguaje de programación Python y que fueron descritas en el capítulo anterior y a las librerías que incluye para el reconocimiento de imágenes, se ha optado por elegir este lenguaje para la programación.

Para el reconocimiento y detección de caracteres, en Phyton existen diversos métodos y librerías como:

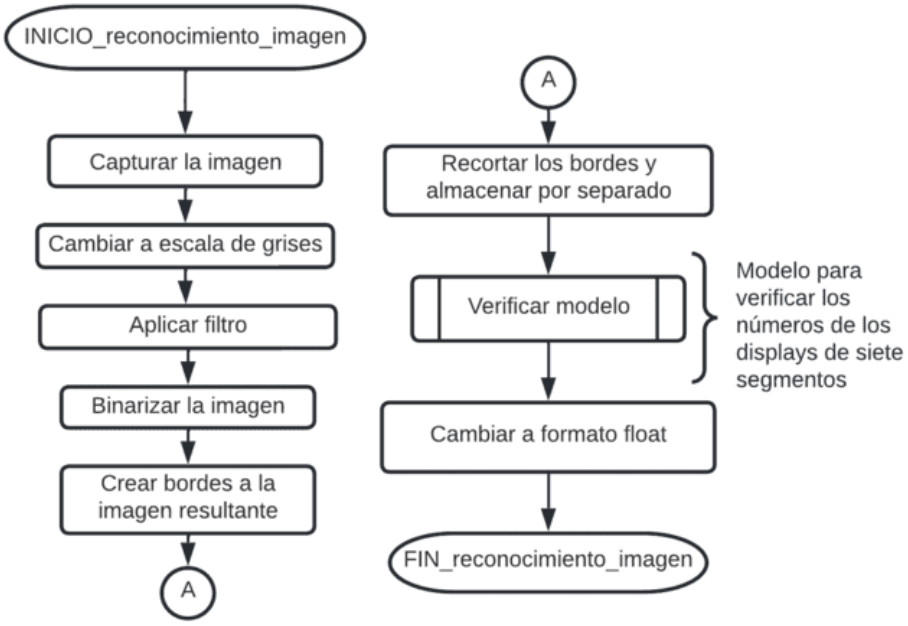
- **Tesseract** que requiere de pre-entrenamiento para su funcionamiento adecuado.
- **TensorFlow** que se centra en el uso de redes neuronales con algoritmos de aprendizaje profundo que encarece el sistema

- **OpenCV** que son librerías con menor consumo de recursos, simples, y bajo tiempo de ejecución. Cuya eficiencia es suficiente para el reconocimiento de caracteres de un display de siete segmentos.

Metodología y procedimiento de reconocimiento de caracteres. El reconocimiento de caracteres del display de siete segmentos tiene dos partes principales; el procesamiento digital de imágenes y el modelo del display de siete segmentos. En la Figura 33 se muestra el diagrama de flujo generalizado, el cual representa el diseño del sistema de reconocimiento de caracteres de un display de siete segmentos y con esto se determinaría la producción individual del ganado.

Figura 33

Diagrama de flujo para el reconocimiento y detección de los displays de siete segmentos del medidor de flujo



Las funciones de OpenCV que se usan y que tienen relación secuencialmente con el diagrama de flujo expuesto en la Figura 33, se describen a continuación.

- **VideoCapture():** Esta función permitirá capturar una imagen mediante la cámara WEB, para este proyecto se deberá especificar el parámetro que define el puerto en la que se encuentra conectada la cámara.

- **CV2.CVTCOLOR(IMG, CV2.COLOR_RGB2GRAY):** Esta función se utiliza para convertir una imagen de un espacio de color a otro, de ahí que entre paréntesis se tiene "*cv2.color_rgb2gray"* la cual sirve para pasar de los colores RGB a una escala de grises.

- **CV2.MEDIANBLUR():** Debido a que las imágenes a veces presentan un ruido denominado de sal y pimienta o también conocido como ruido impulsivo, se debe de aplicar algún método para reducir aquel ruido y la forma más eficaz es aplicando un filtro mediano (Júnez, 2011). Por tal razón la función "*medianBlur()"* permite la operación de desenfoque de la mediana, en donde el elemento central de la imagen es reemplazado por la mediana de todos los píxeles en el área del núcleo, lo cual permite procesar los bordes mientras es eliminado el ruido. El parámetro que se va a usar es "*cv2.mediaBlur(gray,1)"* en donde *gray* representa la imagen en escala de grises como entrada y el *1* es el tamaño del kernel.

- **CV2.THRESHOLD():** La moralización es la mejor manera de fraccionar imágenes en donde, el objetivo sea diferenciar un objeto de interés respecto al fondo de la imagen. Para esto, es necesario cambiar a escala de grises la imagen seleccionada, y tener umbral para la clasificación del color. El parámetro que se usa es "*cv2.threshold(gray,120,255,THRESH_BINARY)",* donde, *gray* representa la imagen en escala de grises, *120* es el umbral mínimo que se va a comparar con cada valor de los pixeles de la imagen, *255* es el valor máximo, que representa al color blanco, y "*THRESH_BINARY"* es la técnica de

umbralización, de tal forma que si la intensidad de píxeles es mayor que el umbral establecido, el valor se establece en 255 (blanco), de lo contrario, se establece en 0 (negro).

- **CV2.FINDCOUNTOURS():** Los contornos son la curva que une a todos los puntos continuos que tienen el mismo color o la misma intensidad, lo cual es muy útil para la detección y reconocimiento de objetos. Dicho esto, la función que permite encontrar los contornos de la imagen es "*cv.findContours(thresh1, cv2.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)"*, en donde *thresh1* es la imagen binarizada resultante, "*cv2.RETR_EXTERNAL"* es el modo de recuperación de contornos, "*APPROX_SIMPLE"* es el método de aproximación de contornos. Así se genera los contornos y la jerarquía, en donde cada contorno individual es una matriz *numpy* de coordenadas (x,y) de puntos límite.

- **CV2.DRAWCONTOURS():** Una vez encontrados los contornos se deben dibujarlos, siendo "*cv2.drawContours(img,[contour],0,(0,255,0),3)"* el método usado, su primer argumento es la imagen de origen, el segundo son los contornos que deben pasarse como una lista de Python, el tercer argumento es el índice de contornos.

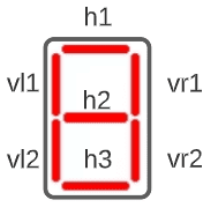
- Una vez dibujado los contornos en la imagen, estos se recortan y son almacenados para poder ser verificados con el modelo que identifica los números de los displays de siete segmentos.

Una vez dibujado los contornos en la imagen, estos se recortan y son almacenados para poder ser verificados con el modelo que identifica los números de los displays de siete segmentos. Para diseñar el modelo de display de siete segmentos primero se asigna nombres para cada uno de los segmentos. En la Figura 34 los segmentos horizontales se llamarán h1,

h2 y h3, los segmentos verticales del lado izquierdo vl1 y vl2, y los segmentos verticales de la derecha vr1 y vr2.

Figura 34

Definición de las variables para cada segmento del display



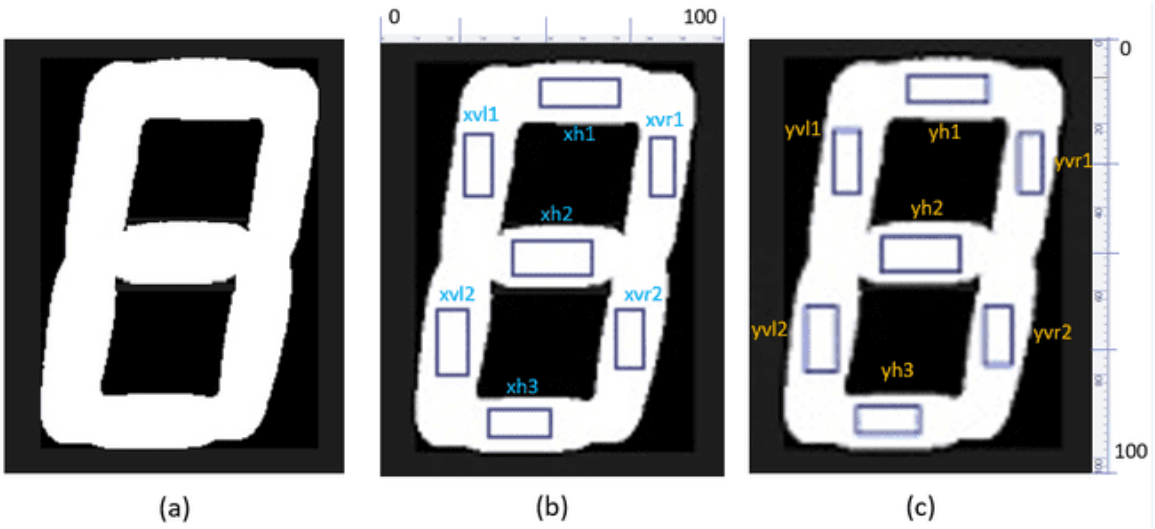
Nota. Las variables h1, h2, h3, vl1, vl2, vr1, vr2 se utilizan para desarrollar el algoritmo para la detección de los números en pantalla del medidor

Para el siguiente paso se utiliza la imagen procesada que ha sido binarizada y recortada, ver Figura 35 apartado (a). Posteriormente se dibuja rectángulos dentro de cada segmento, de esta forma si el rectángulo tiene fondo blanco entonces dicho segmento está activo y si tiene fondo negro esta inactivo. Fue necesario realizar varias capturas con varios números en el display para definir una posición general de los rectángulos.

Luego se determina el valor de la abscisa (eje x) de los extremos de cada rectángulo, para ello se escala el ancho del rectángulo externo donde 100 unidades en la figura es 1 para el modelo. De la misma forma para el eje y, ver Figura 35 apartado (b) y (c).

Figura 35

Modelo del display de siete segmentos



Nota. a) Imagen procesada, b) Imagen con escala horizontal de 100:1, c) Imagen con escala vertical de 100:1

A partir de la Figura 35 se tienen las abscisas y ordenadas de todos los rectángulos, por ejemplo, xh1 = [0.46, 0.70], 0.46 representa el valor de la abscisa del extremo superior izquierdo del rectángulo inscrito en el segmento xh1, mientras que 0.70 representa la abscisa del extremo superior derecho. El origen se consideró el extremo superior izquierdo del rectángulo externo. El resto de valores se muestra a continuación.

$$xh1 = [0.46, 0.70], \quad xh2 = [0.38, 0.61], \quad xh3 = [0.31, 0.44],$$

$$xvl1 = [0.24, 0.32], \quad xvl2 = [0.15, 0.25], \quad xvr1 = [0.78, 0.86]$$

$$xvr2 = [0.68, 0.76], \quad yh1 = [0.8, 0.16], \quad yh2 = [0.44, 0.53]$$

$$yh3 = [0.84, 0.89], \quad yvl1 = [0.20, 0.36], \quad yvl2 = [0.61, 0.76]$$

$$yvr1 = [0.21, 0.36] \quad yvr2 = [0.61, 0.74]$$

El punto flotante se determina encontrando el área del rectángulo externo, si el área es inferior a un umbral (menor a la cuarta parte del rectángulo que encierra el display), entonces es un punto flotante. Otro caso a considerar es cuando se detecta el número uno en el display,

el rectángulo que lo encierra tendrá una relación ancho-alto mucho menor al resto de números debido a que solo estarán activos los segmentos $yvr1$ y $yvr2$, para identificar se comprar dicha relación con un valor de ajuste (entre 0.2 y 0.5).

Para el resto de números se determina el área pintada de blanco dentro de cada rectángulo interno a cada segmento, esta área se divide para el área de total de los rectángulos definidos en el modelo, si la relación es mayor a una constante implica que el segmento está encendido. Finalmente se asigna valores para cada combinación de segmentos de acuerdo a la Tabla 15.

Tabla 15

Codificación del modelo de display de siete segmentos

Segmentos encendidos	Caracter asignado
h1, h3, vl1, vl2, vr1, vr2	‘0’
vr1, vr2	‘1’
h1, h2, h3, vl2, vr1	‘2’
h1, h2, h3, vr1, vr2	‘3’
h2, vl1, vr1, vr2	‘4’
h1, h2, h3, vl1, vr2	‘5’
h1, h2, h3, vl1, vl2, vr2	‘6’
h1, vr1, vr2	‘7’
h1, h2, h3, vl1, vl2, vr1, vr2	‘8’
h1, h2, h3, vl1, vr1, vr2	‘9’
h1, h2, vl1, vl2, vr1	‘1’ (P)
h3, vl1, vl2, vr1, vr2	‘11’ (U)
Otro caso	‘-1’

En la Figura 36 se muestra el diagrama de flujo del proceso para verificar modelo de display de siete segmentos. La función *shape* pertenece a la librería numpy de Python y permite calcular la dimensión de una lista, en este caso la dimensión coincidirá con el ancho y el alto del rectángulo que contiene la imagen del display previamente procesada. La función *count_nonzero()* también es parte de la librería numpy y devuelve el número de elementos diferentes de cero dentro de una lista, de esta forma se determina el área de color blanco en el modelo de display.

Figura 36

Diagrama de flujo del proceso verificar modelo

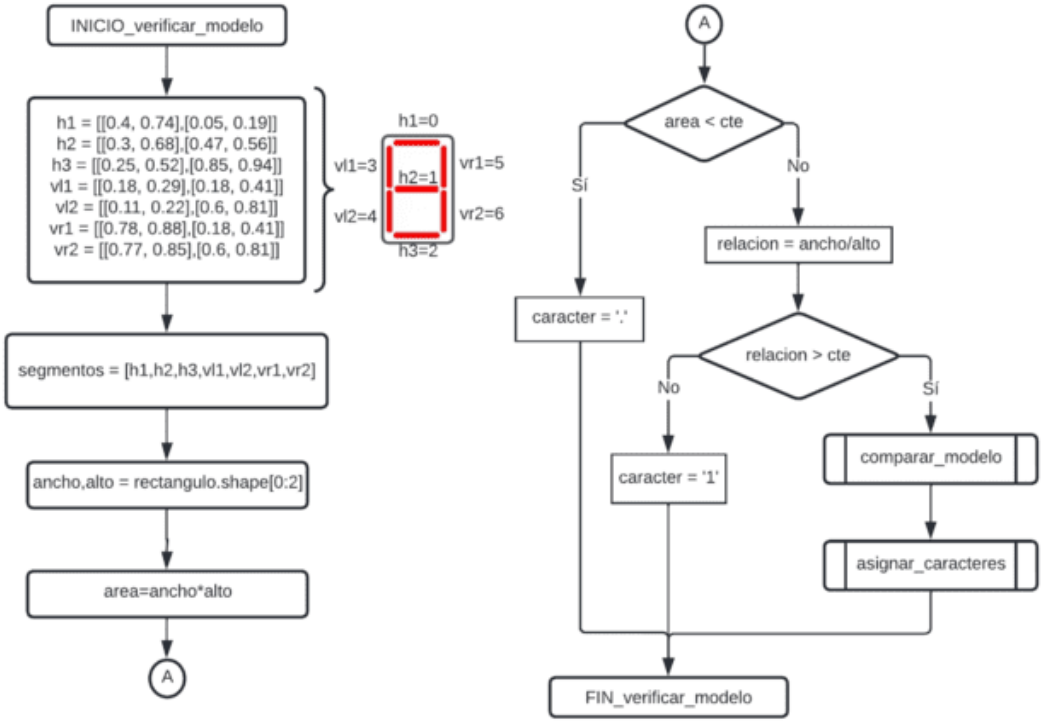


Figura 37

Diagrama de flujo del proceso comparar modelo

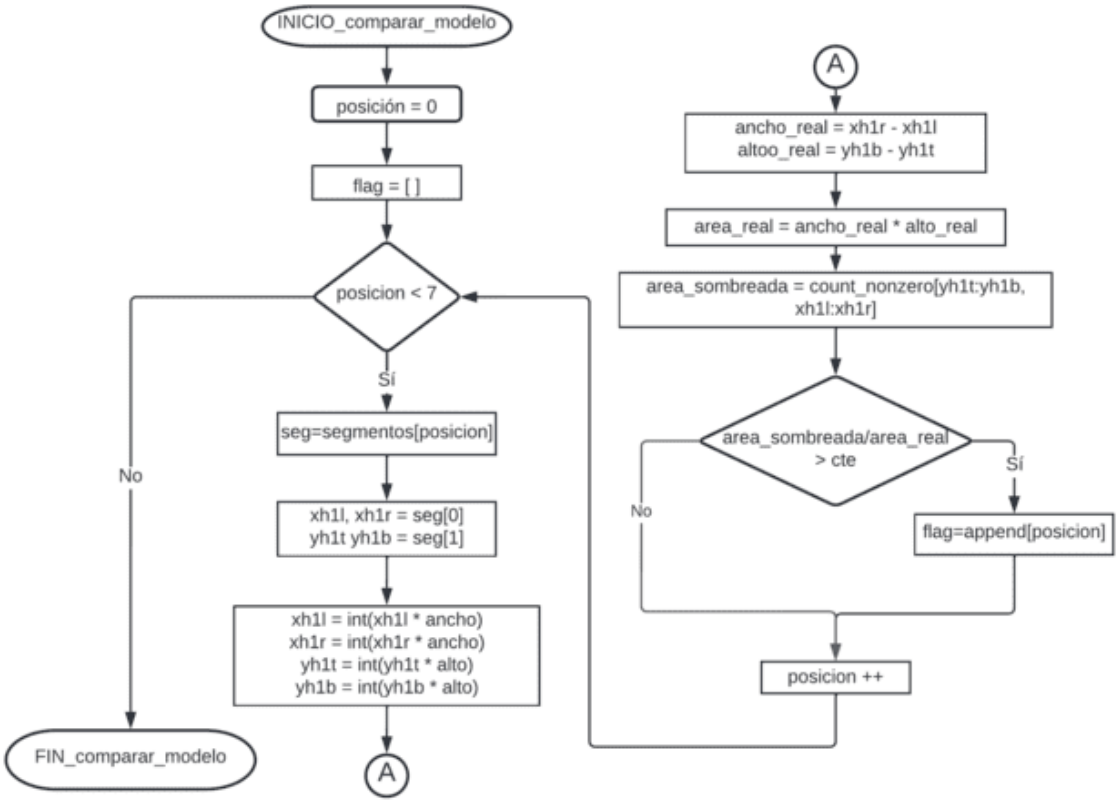
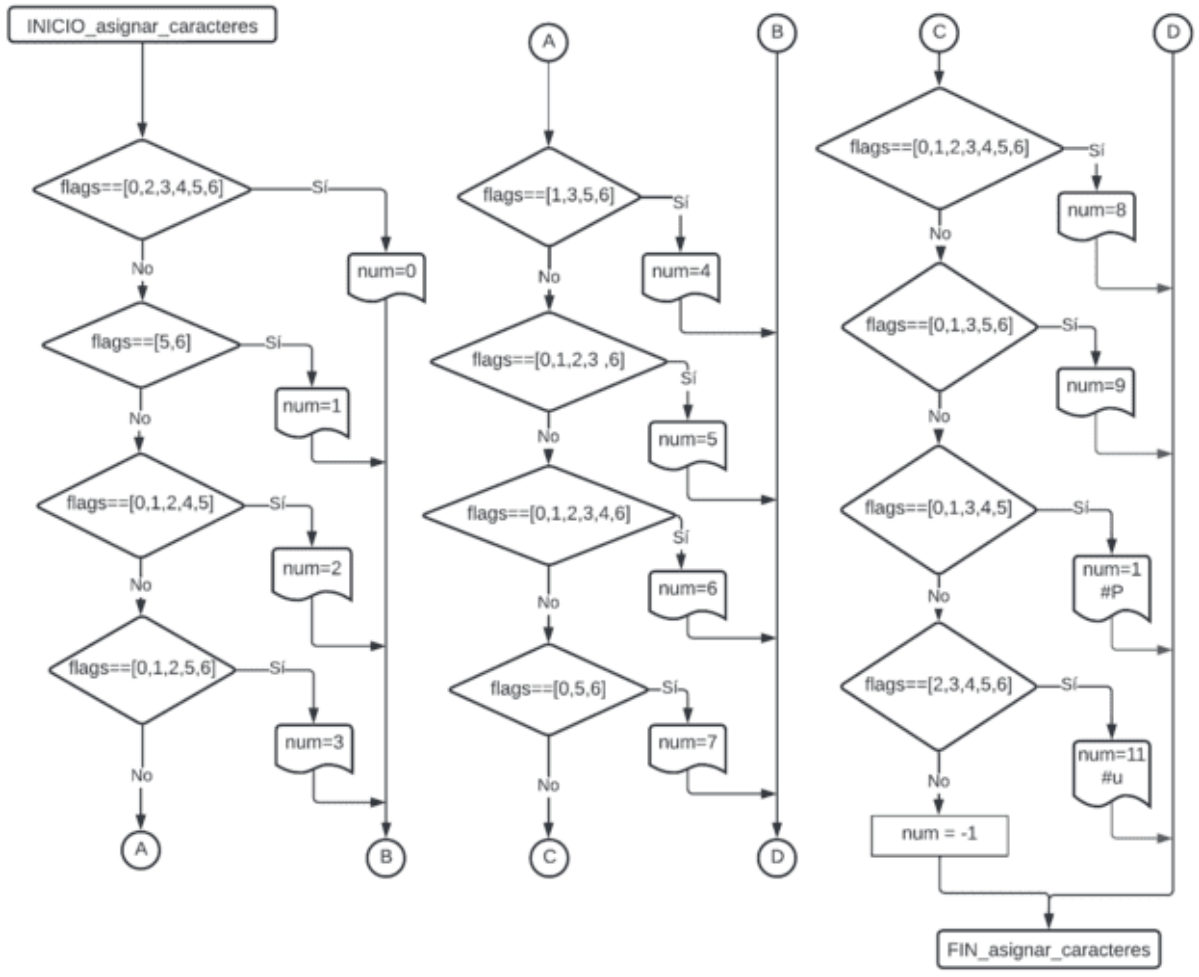


Figura 38

Diagrama de flujo del proceso asignar caracteres



Selección de los componentes del sistema

Para implementar el algoritmo del procesamiento digital de imágenes empleando Python y OpenCV es necesario disponer de una computadora de bajo costo. Por lo que se enfocó la selección a computadores de placa única. Inicialmente se utilizó una Raspberry Pi3, por su popularidad, y la disponibilidad de información dado que cuenta con una gran comunidad, soporte y foros de ayuda.

Dado que la sala de ordeño tiene seis medidores es necesario usar seis cámaras que se instalarán frente a cada medidor. El utilizar una sola Raspberry no abastecería para procesar toda la información, ya que cuenta con cuatro puertos USB. Uno de los cuáles se

utilizaría para la conexión a la red con los demás dispositivos. Por esta razón se diseñó en función de utilizar una computadora de placa única (SBC de sus siglas en inglés Single Board Computer) para procesar la información de dos cámaras. Dado el costo elevado de la raspberry y su poca disponibilidad en tiendas de equipos electrónicos, se prefirió seleccionar otra SBC con características similares. En el mercado internacional se pudo evidenciar que existen numerosas SB. De entre ellas se seleccionó la Computadora Libre AML-S905X-CC debido a la disponibilidad y precio. Las características de esta tabla se presentan en la Tabla 16.

Tabla 16

Características de la SBC AML-S905X-CC

Descripción	Detalles técnicos
Procesador	ARM Cortex-A de alto rendimiento GPU penta core 3
Núcleos	Núcleos cuádruples de bajo consumo de 64 bits
Puertos	Cuenta con dos puertos USB 3.0 y dos puertos USB 2.0
Salidas	Cuenta con 40 pines compatibles, los cuales son compatibles con UART, SPI, PWM, I2C y GPIO
Sistemas operativos compatibles	Debian, Armbian, Raspbian, Ubuntu Mate

En cuanto a la selección de la cámara que pueda capturar la pantalla del medidor se ha elegido una Web Cam de 1080 píxeles, conexión USB 2.0, con enfoque automático y que tiene compatible con varios sistemas operativos, ver Figura 39. Para este proyecto se requerirá de seis de estas cámaras, las cuales se instalarán enfocando a la pantalla de cada medidor.

Figura 39

Web Cam Full HD 1080P

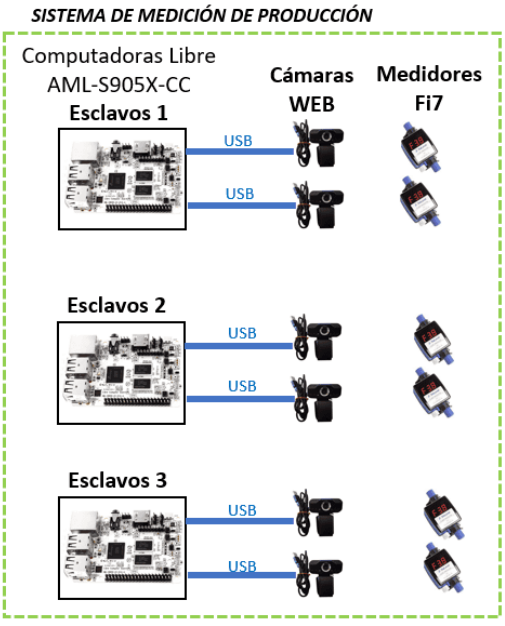
Nota. Cámara que se va a usar para capturar la imagen de los displays de siete segmentos de las pantallas de los medidores para su posterior detección y reconocimiento de caracteres de los números mostrados en los medidores.

Arquitectura del sistema

Para la arquitectura del sistema de medición se utiliza tres SBC's AML-S905X-CC que realizará el proceso de reconocimiento de caracteres de la pantalla del medidor, cada SBC tiene conectado dos cámaras en sus puertos USB como se indica en la Figura 40. Las cámaras deben estar ubicadas frente al medidor de leche Fi7.

Figura 40

Esquema de conexión del sistema de medición de la producción de leche



Diseño del software de gestión ganadera

El software de gestión ganadera cumple un papel importante en el proyecto, pues dentro de él se realizará todo el registro de la hacienda. Con la finalidad de alimentar automáticamente la información proveniente del sistema de medición e identificación automática del animal durante el ordeño, es necesario establecer una interfaz intermedia de comunicaciones. Adicionalmente, previo el diseño de la base de datos es necesario recabar la información que requiere el administrador de la hacienda para determinar los requisitos y funcionalidades que debe tener el software a implementarse. A continuación se exponen los requisitos funcionales que tendrá el software además de los no funcionales que determinan las características de la aplicación.

Requisitos funcionales

Son los requisitos que definen las instrucciones y funcionalidades que la aplicación de escritorio debe cumplir en su diseño e implementación. Dicho esto, y de acuerdo a las necesidades de la hacienda, el software debe permitir:

- RF1. Control de acceso
- RF2. Registrar o eliminar usuarios
- RF3. Ingresar o editar información de la hacienda
- RF4. Registrar o eliminar ganado
- RF5. Registrar o eliminar partos
- RF6. Registrar o eliminar muertes
- RF7. Registrar o eliminar montas
- RF8. Registrar o eliminar inseminación
- RF9. Registrar o eliminar enfermedades y medicamentos

- RF10. Registrar o eliminar potreros
- RF11. Registrar o eliminar datos de ordeño
- RF12. Comunicar con el sistema de medición de la producción de leche
- RF13. Realizar consultas de la información del ganado

Requisitos no funcionales

Estos requisitos tienen que ver con las propiedades o cualidades que el software debe tener, es decir, las características que hacen que le aplicación sea atractiva a la vista, usable, eficiente y confiable. De igual manera pensando en el bienestar de la hacienda se definen los siguientes requerimientos:

Requerimiento de apariencia. La aplicación de escritorio debe proporcionar al usuario una interfaz gráfica interactiva y fácil de utilizar en todas las funciones que presente, además debe ser sencilla y coherente.

Requerimiento de usabilidad. Los usuarios finales deben recibir una capacitación para utilizar la aplicación de forma apropiada.

Requerimiento de hardware y software. Se debe contar con una computadora de escritorio o laptop con el sistema operativo Windows instalado, además debe contar con una base de datos local.

Modelo de caso de uso del sistema

Una vez definido las necesidades o requisitos de la aplicación iniciamos el diseño, para ello se divide el procedimiento en tres partes; definición de los actores del sistema, determinación de los casos de uso y representación mediante un diagrama de casos de uso. Siendo los actores, las entidades o usuarios que van a interactuar con la aplicación de escritorio. Los tipos de usuario que interactuarán con esta aplicación se describen en la Tabla 17.

Tabla 17

Tipos de usuario que interactuaran con la aplicación de escritorio.

Actores	Descripción
Administrador	Este usuario tiene acceso a todas las funcionalidades de la aplicación ganadera, incluso la de registrar nuevos usuarios (en este caso un empleado o un nuevo administrador)
Empleado	El empleado tendrá acceso a todas las funciones excepto; editar información de la hacienda, registrar nuevos usuarios, eliminar nuevos usuarios y eliminar algún tipo de registro.

La Figura 41 y la Figura 42 describen el diagrama de caso del administrador y del empleado respectivamente. El diagrama de caso define la relación entre el actor y el aplicativo del sistema.

Figura 41

Diagrama de caso de uso del actor administrador

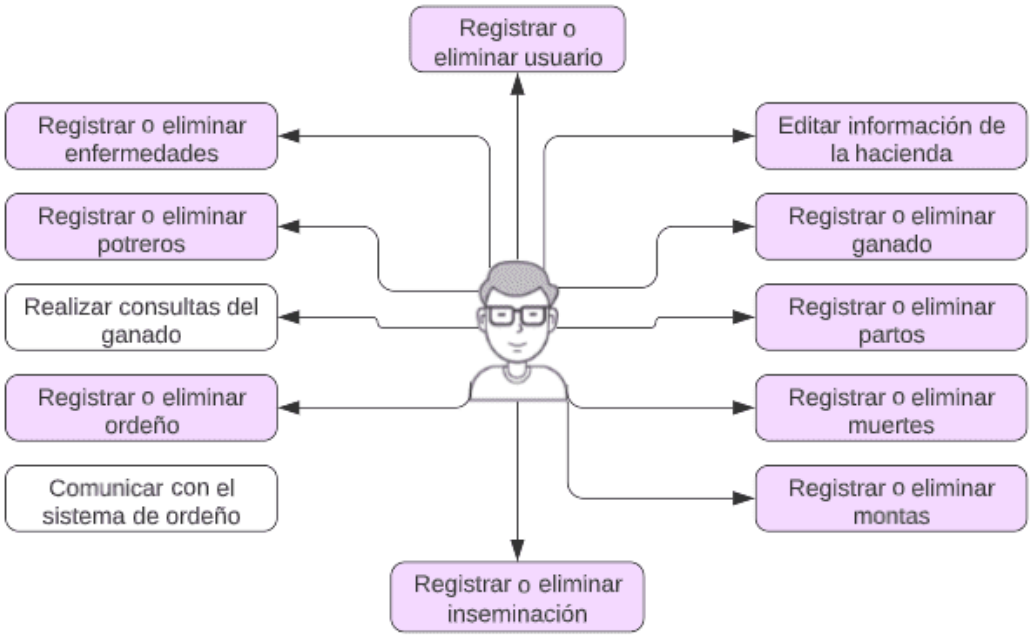
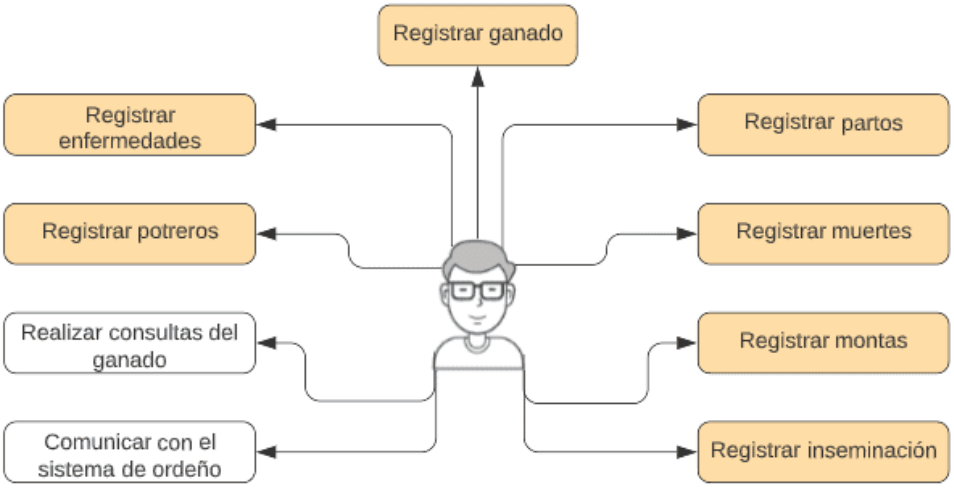


Figura 42

Diagrama de caso de uso del actor empleado



Descripción de los casos de uso del sistema

Una vez definido los casos de uso para cada actor, es necesario detallar cada una de las funcionalidades de la aplicación, de esta manera se puede definir los parámetros y elementos con les que interactuará el usuario por cada ventana de la aplicación. De la Tabla 18 a la Tabla 38 se detallan las funcionalidades para cada actor y los elementos de interacción.

Tabla 18

Descripción del caso de uso registrar usuario

Caso de uso	
Nombre	Registrar usuario
Actores	Administrador
Descripción	El actor tiene la opción de crear un nuevo usuario con información básica del mismo.
Precondiciones	Se debe verificar que los campos obligatorios no estén en blanco y que los datos ingresados tengan coherencia
Curso normal de los eventos	

Acción del actor	**Respuesta del sistema**
Ingresa datos	
Accionar el botón de agregar foto	Se despliega el explorador de archivos
Seleccionar una foto del usuario	Se copia la imagen en un directorio de la aplicación y se guarda el nombre en una variable temporal.
Accionar el botón de guardar	Guarda datos del usuario y ubicación de la imagen copiada en la base de datos.
Postcondiciones	Actualizar los datos de la tabla de usuarios

Tabla 19

Descripción del caso de uso eliminar usuario

Caso de uso	
Nombre	Eliminar usuario
Actores	Administrador
Descripción	El actor tiene la opción de eliminar un usuario existente
Precondiciones	No se debe eliminar el usuario principal (primer usuario).
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Accionar el botón de eliminar	Eliminar usuario de la base de datos
Postcondiciones	Actualizar los datos de la tabla usuarios.

Tabla 20

Descripción del caso de uso editar información de la hacienda

Caso de uso

Nombre	Editar información de la hacienda
Actores	Administrador
Descripción	El actor tiene la opción editar los datos de la hacienda.
Precondiciones	Validar los datos registrados, por ejemplo, el número de teléfono no debe contener letras.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Accionar el botón de editar	Se habilitan los contenedores (labels) para modificar la información.
Editar los campos	Ninguna
Accionar el botón de guardar	Se modifica la información de la base de datos y se bloquea los contenedores.
Postcondiciones	Actualizar los datos de la tabla de usuarios

Tabla 21

Descripción del caso de uso registrar ganado

Caso de uso	
Nombre	Registrar ganado
Actores	Administrador, empleado
Descripción	El actor puede realizar el registro de un nuevo ganado, siempre y cuando cuente con un número de etiqueta.
Precondiciones	Se debe verificar que los campos obligatorios no estén en blanco y que los datos ingresados tengan coherencia.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**

Ingresar la información del ganado.	Ninguna.
Accionar el botón de agregar foto.	Se despliega el explorador de archivos
Seleccionar una foto del usuario.	Se copia la imagen en un directorio de la aplicación y se guarda el nombre en una variable temporal.
Accionar el botón de guardar	Guarda datos del ganado y ubicación de la imagen copiada, en la base de datos.
Postcondiciones	Actualizar los datos de la tabla de registro de ganado.

Tabla 22

Descripción del caso de uso eliminar ganado

Caso de uso	
Nombre	Eliminar ganado
Actores	Administrador
Descripción	El actor puede eliminar un registro de ganado.
Precondiciones	Debe existir al menos un animal registrado.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Accionar el botón de eliminar.	Busca al animal en la base de datos y lo elimina.
Postcondiciones	Actualizar los datos de la tabla de registro de ganado.

Tabla 23

Descripción del caso de uso registrar parto

Caso de uso	
Nombre	Registrar parto
Actores	Administrador, empleado
Descripción	El actor tiene la opción de registrar un nuevo parto.
Precondiciones	El tag de la vaca debe estar registrada en la base de datos.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Ingresar la información del parto.	Ninguna.
Accionar el botón de agregar parto.	Guarda la información del parto en la base de datos. Luego verifica si la cría está viva o muerta, si está viva habilita el botón de registrar cría y limpia los contenedores, si la vaca está muerta solo limpia los contenedores.
Postcondiciones	Actualizar los datos de la tabla de partos.

Tabla 24

Descripción del caso de uso eliminar parto

Caso de uso	
Nombre	Eliminar parto
Actores	Administrador

Descripción	El actor puede eliminar un registro de parto.
Precondiciones	Debe existir al menos un registro de parto.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Accionar el botón de eliminar.	Busca el ID del parto registrado y lo elimina de la base de datos.
Postcondiciones	Actualizar los datos de la tabla de partos.

Tabla 25

Descripción del caso de uso registrar muerte

Caso de uso	
Nombre	Registrar muerte
Actores	Administrador, empleado
Descripción	El actor tiene la opción de registrar la muerte del animal.
Precondiciones	El tag de la vaca debe estar registrada en la base de datos.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Ingresar la información de la muerte.	Ninguna.
Accionar el botón de agregar muerte.	Guarda la información de la muerte en la base de datos.
Postcondiciones	Actualizar los datos de la tabla de muertes y la tabla de registro de ganado.

Tabla 26

Descripción del caso de uso eliminar muerte

Caso de uso	
Nombre	Eliminar parto
Actores	Administrador
Descripción	El actor puede eliminar un registro de muerte.
Precondiciones	Debe existir al menos un registro de muerte.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Accionar el botón de eliminar muerte.	Busca el ID de la muerte registrada y lo elimina de la base de datos.
Postcondiciones	Actualizar los datos de la tabla de muertes.

Tabla 27

Descripción del caso de uso registrar muerte

Caso de uso	
Nombre	Registrar monta.
Actores	Administrador, empleado
Descripción	El actor tiene la opción de registrar una monta de una vaca en específico.
Precondiciones	El tag de la vaca y del toro deben estar registradas en la base de datos.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**

Ingresar la información de la monta.	Ninguna.
Accionar el botón de agregar monta.	Guarda la información de la monta en la base de datos.
Postcondiciones	Actualizar los datos de la tabla de montas.

Tabla 28

Descripción del caso de uso eliminar muerte

Caso de uso	
Nombre	Eliminar monta
Actores	Administrador
Descripción	El actor puede eliminar un registro de monta.
Precondiciones	Debe existir al menos un registro de monta.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Accionar el botón de eliminar muerte.	Busca el ID de la monta registrada y lo elimina de la base de datos.
Postcondiciones	Actualizar los datos de la tabla de montas.

Tabla 29

Descripción del caso de uso registrar inseminación

Caso de uso	
Nombre	Registrar inseminación
Actores	Administrador, empleado

Descripción	El actor tiene la opción de registrar una inseminación que se haya realizado en la hacienda.
Precondiciones	El tag de la vaca debe estar registrada en la base de datos.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Ingresar la información de la inseminación.	Ninguna.
Accionar el botón de agregar inseminación.	Guarda la información de los datos de inseminación en la base de datos.
Postcondiciones	Actualizar los datos de la tabla de inseminaciones.

Tabla 30

Descripción del caso de uso eliminar inseminación

Caso de uso	
Nombre	Eliminar inseminación.
Actores	Administrador
Descripción	El actor puede eliminar un registro de inseminación de la base datos y de la tabla visual de la aplicación.
Precondiciones	Debe existir al menos un registro de inseminación.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Accionar el botón de eliminar inseminación.	Busca el ID de la inseminación registrada y lo elimina de la base de datos.
Postcondiciones	Actualizar los datos de la tabla de inseminación.

Tabla 31

Descripción del caso de uso registrar enfermedad

Caso de uso	
Nombre	Registrar enfermedad
Actores	Administrador, empleado
Descripción	El actor tiene la opción de registrar una enfermedad. Debe incluir el tag del animal, fecha de registro, tipo de enfermedad como
Precondiciones	El tag de la vaca debe estar registrada en la base de datos.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Ingresar la información de la muerte.	Ninguna.
Accionar el botón de agregar muerte.	Guarda la información del registro de enfermedad en la base de datos.
Postcondiciones	Actualizar los datos de la tabla de enfermedades.

Tabla 32

Descripción del caso de uso eliminar enfermedad

Caso de uso	
Nombre	Eliminar enfermedad
Actores	Administrador

Descripción	El actor puede eliminar un registro de enfermedad del ganado.
Precondiciones	Debe existir al menos un registro de enfermedad.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Accionar el botón de eliminar muerte.	Busca el ID de la enfermedad registrada y lo elimina de la base de datos.
Postcondiciones	Actualizar los datos de la tabla de enfermedades.

Tabla 33

Descripción del caso de uso registrar potreros

Caso de uso	
Nombre	Registrar potrero
Actores	Administrador, empleado
Descripción	El actor tiene la opción de registrar un potrero. Además, tendrá la opción asignar el número de potrero donde se encuentra un animal.
Precondiciones	El tag de la vaca debe estar registrada en la base de datos y para asignar un potrero a un animal, el potrero ya debe estar registrado.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Ingresar la información del potrero.	Ninguna

Accionar el botón de agregar potrero.	Guarda la información del registro de potrero en la base de datos.
Ingresar información de la ubicación del animal	Ninguna
Accionar el botón de agregar ubicación	Verifica si el número del tag y el id del potrero existe. Si existe guarda los datos en la base de datos, sino se muestra un mensaje de error.
Postcondiciones	Actualizar los datos de la tabla de potreros y pastoreo dependiendo de que botón se accione.

Tabla 34

Descripción del caso de uso eliminar potrero

Caso de uso	
Nombre	Eliminar potrero
Actores	Administrador
Descripción	El actor puede eliminar un registro de potrero o un registro de ubicación del ganado.
Precondiciones	Debe existir al menos un registro de potrero o pastoreo.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Accionar el botón de eliminar potrero.	Busca el ID del potrero registrado y lo elimina de la base de datos.
Accionar el botón de eliminar potrero	Busca el ID de la ubicación del animal registrado y lo elimina de la base de datos.

Postcondiciones	Actualizar los datos de la tabla de potreros y pastoreo dependiendo de que botón se accione.

Tabla 35

Descripción del caso de uso registrar ordeño

Caso de uso	
Nombre	Registrar ordeño
Actores	Administrador, empleado
Descripción	El actor tiene la opción de registrar la producción individual del ganado de forma manual.
Precondiciones	El tag del animal debe estar registrado.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Ingresar la información de la producción del animal.	Ninguna
Accionar el botón de agregar potrero.	Guarda la información del registro de ordeño en la base de datos.
Postcondiciones	Actualizar los datos de la tabla de potreros y pastoreo dependiendo de que botón se accione.

Tabla 36

Descripción del caso de uso eliminar ordeño

Caso de uso	
Nombre	Eliminar ordeño
Actores	Administrador, empleado

Descripción	El actor puede eliminar un registro de ordeño de una vaca en específico.
Precondiciones	Debe existir al menos un registro de producción
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Accionar el botón de agregar producción.	Busca el ID del registro de producción seleccionado y lo elimina de la base de datos.
Seleccionar filtro por fecha	Habilita los contenedores para ingresar el rango de fechas y bloquea la búsqueda por parto.
Seleccionar filtro por parto	Habilita los contenedores para ingresar el número de parto y bloquea la búsqueda por fecha.
Accionar el botón filtrar	Si se filtra por fecha, busca las producciones del tag seleccionado en la base de datos y selecciona las que se encuentra en el rango de fechas establecido. Si se filtra por parto, primero busca los registros de parto de la base de datos y obtiene las fechas del parto seleccionado y el próximo parto y posteriormente busca la producción dentro de ese rango. Luego actualiza la tabla de producción.
Postcondiciones	Actualizar los datos de la tabla de producción.

Tabla 37

Descripción del caso de uso realizar consulta.

Caso de uso	
Nombre	Realizar consulta

Actores	Administrador, empleado
Descripción	En este caso de uso el actor puede visualizar toda la información de un animal. Además, se mostrarán gráficas de producción que es indispensable para lograr los objetivos del proyecto.
Precondiciones	El tag del animal seleccionado debe estar registrado.

Curso normal de los eventos

Acción del actor	**Respuesta del sistema**
Ingresar el Id del animal	Ninguna
Presionar la tecla ENTER	Se muestra toda la información del animal, tabla de enfermedades, vacunas y tabla de producción.
Seleccionar filtro por fecha	Habilita los contenedores para ingresar el rango de fechas y bloquea la búsqueda por parto.
Seleccionar filtro por parto	Habilita los contenedores para ingresar el número de parto y bloquea la búsqueda por fecha.
Accionar el botón filtrar	Si se filtra por fecha, busca las producciones del tag seleccionado en la base de datos y selecciona las que se encuentra en el rango de fechas establecido. Si se filtra por parto, primero busca los registros de parto de la base de datos y obtiene las fechas del parto seleccionado y el próximo parto y posteriormente busca la producción dentro de ese rango y actualiza la tabla.
Accionar el botón de ver gráficas de producción	Muestra la gráfica de la producción dentro del rango de fechas filtrado. También calcula el valor máximo, mínimo y el promedio de la producción de ese animal.

Postcondiciones	Actualizar los datos de la tabla de producción.

Tabla 38

Descripción del caso de uso comunicar con el sistema de ordeño

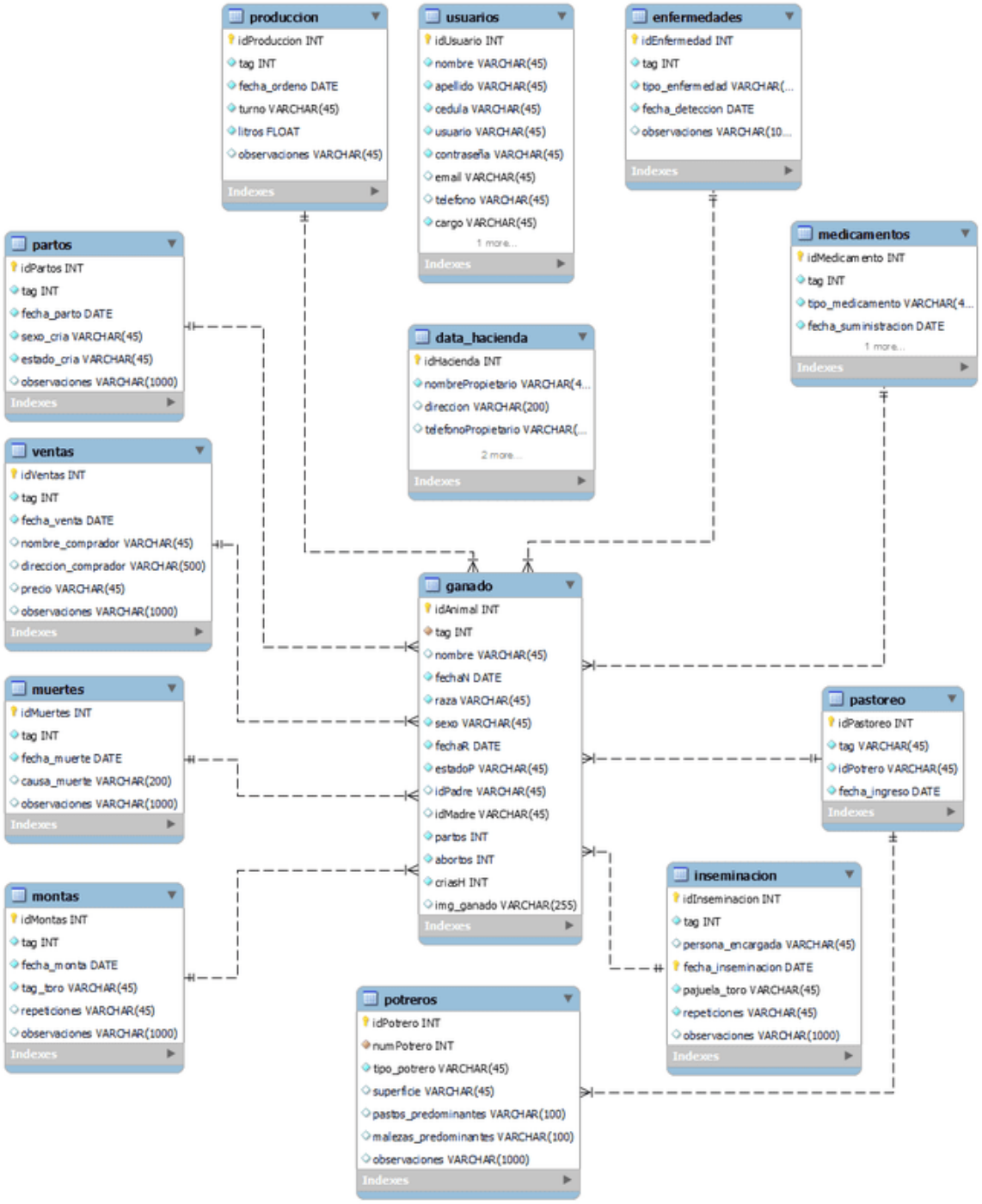
Caso de uso	
Nombre	Comunicar con el sistema de ordeño
Actores	Administrador, empleado
Descripción	La comunicación permite importar los datos desde el sistema de medición de la producción hacia la base de datos o desde un archivo plano hacia la base de datos.
Precondiciones	Establecer el protocolo de comunicación.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Seleccionar el puerto de comunicación.	Ninguna.
Accionar el botón conectar	Se establece la comunicación, recibe los datos desde el sistema de medición de la producción y carga los datos a la base de datos.
Accionar el botón seleccionar archivo	Se despliega la ventana del explorador de archivo.
Seleccionar el archivo .csv o .txt	Obtiene los datos del archivo.
Accionar el botón cargar datos.	Carga los datos a la base de datos.
Postcondiciones	Actualizar tabla de producción.

Esquema de la base de datos

La plataforma utilizada para el desarrollo de la base de datos es MySQL debido a sus características y ventajas, entre las más importantes se tiene que MySQL permite crear la base de datos a través de tablas, además es seguro, flexible y fácil de usar, es multiplataforma, y es de código abierto. La Figura 43 ilustra un esquema de la base de datos y las 13 tablas que la conforman. Cada tabla representa la entidad del modelo entidad-relación. Las interrelaciones se asignan con líneas como se muestra en la misma figura donde la relación es “Tiene”, por ejemplo, un animal o ganado tiene enfermedades o un animal tiene un registro de producción, etc.

Figura 43

Modelo entidad relación de la base de datos



Diccionario de datos

En este apartado se definirán los tipos de datos para los atributos de cada entidad, las claves primarias, y el tipo de atributo, es decir, si es opcional u obligatoria. De la Tabla 39 a la **Tabla 51** se presenta el diccionario de datos para cada entidad.

Tabla 39

Tabla de la entidad Usuario

Nombre de la tabla	usuarios			
Descripción	Esta tabla contendrá la información básica y de contacto de los actores del sistema.			
Columnas de la tabla				
No.	**Nombre**	**Tipo de dato**	**Nulo**	**Descripción**
1	idUsuario	INT	No	Número de registro
2	nombre	VARCHAR(45)	No	Nombre del actor
3	apellido	VARCHAR(45)	Si	Apellido del actor
4	cedula	VARCHAR(45)	Si	Cédula del actor
5	usuario	VARCHAR(45)	No	Nombre de acceso
6	contraseña	VARCHAR(45)	No	Contraseña de acceso
7	email	VARCHAR(45)	Si	Correo electrónico personal
8	telefono	VARCHAR(45)	Si	Teléfono o celular
9	cargo	VARCHAR(45)	No	Administrador o empleado
10	img_usuario	VARCHAR(255)	Si	Directorio de la imagen del usuario

Tabla 40

Tabla de la entidad hacienda

Nombre de la tabla	data_hacienda			
Descripción	Esta tabla contendrá la información básica y de contacto de la hacienda.			
Columnas de la tabla				
No.	**Nombre**	**Tipo de dato**	**Nulo**	**Descripción**
1	idHacienda	INT	No	Número de registro
2	nombrePropietario	VARCHAR(45)	No	Nombre del propietario
3	dirección	VARCHAR(200)	Si	Dirección de la hacienda
4	telefono	VARCHAR(45)	Si	Teléfono del propietario
5	email	VARCHAR(45)	Si	Email del propietario
6	fechaInicio	DATE	No	Fecha de inicio del registro

Tabla 41

Tabla de la entidad ganado

Nombre de la tabla	ganado			
Descripción	Esta tabla contiene la información del ganado			
Columnas de la tabla				
No.	**Nombre**	**Tipo de dato**	**Nulo**	**Descripción**
1	idAnimal	INT	No	Id de registro
2	tag	INT	No	Número de etiqueta
3	nombre	VARCHAR(45)	Si	Nombre del animal
4	fechaN	DATE	No	Fecha de nacimiento
5	raza	VARCHAR(45)	No	Raza del animal

6	sexo	VARCHAR(45)	No	Sexo del animal
7	fechaR	DATE	No	Fecha de registro
8	estadoP	VARCHAR(45)	No	Tipo de ganado
9	idPadre	VARCHAR(45)	Si	Tag del padre
10	idMadre	VARCHAR(45)	Si	Tag de la madre
11	partos	INT	No	Número de partos
12	abortos	INT	No	Número de abortos
13	criasH	INT	No	Número de crías hembras
14	Img_ganado	VARCHAR(255)	Si	Directorio de la imagen.

Tabla 42

Tabla de la entidad producción

Nombre de la tabla	producción			
Descripción	Esta tabla contendrá la información del registro de producción de una vaca.			
Columnas de la tabla				
No.	**Nombre**	**Tipo de dato**	**Nulo**	**Descripción**
1	idProduccion	INT	No	Id de registro
2	tag	INT	No	Número de la etiqueta
3	fecha_ordeno	DATE	No	Fecha de registro
4	turno	VARCHAR(45)	No	Primer o segundo turno
5	litros	FLOAT	No	Producción en litros
6	observaciones	VARCHAR(200)	Si	Observaciones

Tabla 43

Tabla de la entidad partos

Nombre de la tabla	partos			
Descripción	Esta tabla contendrá la información del registro de partos en la hacienda.			
Columnas de la tabla				
No.	**Nombre**	**Tipo de dato**	**Nulo**	**Descripción**
1	idPartos	INT	No	Id de registro
2	tag	INT	No	Número de la etiqueta
3	fecha_parto	DATE	No	Fecha de registro
4	sexo_cria	VARCHAR(45)	No	Sexo de la cría
5	estado_cria	VARCHAR(45)	No	Si nació viva o muerta
6	observaciones	VARCHAR(200)	Si	Observaciones

Tabla 44

Tabla de la entidad ventas

Nombre de la tabla	ventas			
Descripción	Esta tabla contendrá la información del registro de venta de ganado en la hacienda.			
Columnas de la tabla				
No.	**Nombre**	**Tipo de dato**	**Nulo**	**Descripción**
1	idVentas	INT	No	Id de registro
2	tag	INT	No	Número de la etiqueta
3	fecha_venta	DATE	No	Fecha de registro
4	nombre_comprador	VARCHAR(45)	No	Nombre del comprador

5	direccion_comprador	VARCHAR(200)	No	Dirección
6	precio	FLOAT	Si	Precio de la venta
7	observaciones	VARCHAR(200)	Si	Observaciones

Tabla 45

Tabla de la entidad muertes

Nombre de la tabla	muertes			
Descripción	Esta tabla contendrá la información del registro de muertes de ganado en la hacienda.			
Columnas de la tabla				
No.	**Nombre**	**Tipo de dato**	**Nulo**	**Descripción**
1	idMuertes	INT	No	Id de registro
2	tag	INT	No	Número de la etiqueta
3	fecha_muerte	DATE	No	Fecha de registro
4	causa_muerte	VARCHAR(200)	Si	Causa de la muerte
6	observaciones	VARCHAR(200)	Si	Observaciones

Tabla 46

Tabla de la entidad montas

Nombre de la tabla	montas			
Descripción	Esta tabla contendrá la información del registro de montas en la hacienda.			
Columnas de la tabla				
No.	**Nombre**	**Tipo de dato**	**Nulo**	**Descripción**
1	idMontas	INT	No	Id de registro

2	tag	INT	No	Número de la etiqueta
3	fecha_monta	DATE	No	Fecha de registro
4	tag_toro	VARCHAR(45)	Si	Etiqueta del toro
5	repeticiones	INT	No	Número de intentos
6	observaciones	VARCHAR(200)	Si	Observaciones

Tabla 47

Tabla de la entidad inseminación

Nombre de la tabla	Inseminación			
Descripción	Esta tabla contendrá la información del registro de inseminación en la hacienda.			
Columnas de la tabla				
No.	**Nombre**	**Tipo de dato**	**Nulo**	**Descripción**
1	idInseminacion	INT	No	Id de registro
2	tag	INT	No	Número de la etiqueta
3	persona_encargada	VARCHAR(45)	Si	Nombre de la persona encargada.
4	fecha_inseminacion	DATE	No	Fecha de registro
5	tag_toro	INT	Si	Etiqueta de la pajuela
6	repeticiones	INT	No	Número de intentos
7	observaciones	VARCHAR(200)	Si	Observaciones

Tabla 48

Tabla de la entidad enfermedades

Nombre de la tabla	enfermedades			
Descripción	Esta tabla contendrá la información del registro de enfermedades de las vacas en la hacienda.			
Columnas de la tabla				
No.	**Nombre**	**Tipo de dato**	**Nulo**	**Descripción**
1	idEnfermedad	INT	No	Id de registro
2	tag	INT	No	Número de la etiqueta
3	Tipo_enfermedad	VARCHAR(45)	No	Nombre de la enfermedad
4	fecha_deteccion	DATE	No	Fecha de registro
7	observaciones	VARCHAR(200)	Si	Observaciones

Tabla 49

Tabla de la entidad medicamentos

Nombre de la tabla	medicamentos			
Descripción	Esta tabla contendrá la información del registro de vacunas y medicamentos del ganado en la hacienda.			
Columnas de la tabla				
No.	**Nombre**	**Tipo de dato**	**Nulo**	**Descripción**
1	idMedicamento	INT	No	Id de registro
2	tag	INT	No	Número de la etiqueta
3	tipo_medicamento	VARCHAR(45)	No	Nombre del medicamento

4	fecha_suministracion	DATE	No	Fecha de registro
7	observaciones	VARCHAR(200)	Si	Observaciones

Tabla 50

Tabla de la entidad potrero

Nombre de la tabla	potreros
Descripción	Esta tabla contendrá la información del registro de potreros en la hacienda.

Columnas de la tabla

No.	**Nombre**	**Tipo de dato**	**Nulo**	**Descripción**
1	idPotreros	INT	No	Id de registro
2	num_potrero	INT	No	Número del potrero
3	tipo_potrero	VARCHAR(45)	No	Definir que tipo de potrero es.
4	superficie	FLOAT	Si	Área del potrero
5	pastos_predominantes	VARCHAR(100)	Si	Pastos predominantes
6	malezas_predominantes	VARCHAR(100)	Si	Malezas predominantes
7	observaciones	VARCHAR(200)	Si	Observaciones

Tabla 51

Tabla de la entidad pastoreo

Nombre de la tabla	pastoreo
Descripción	Esta tabla contendrá la información del registro de pastoreo en la hacienda, es decir la ubicación del animal.

Columnas de la tabla				
No.	**Nombre**	**Tipo de dato**	**Nulo**	**Descripción**
1	idPastoreo	INT	No	Id de registro
2	Tag	INT	No	Etiqueta del ganado
3	fecha_ingreso	INT	No	Id del potrero registrado
4	observaciones	VARCHAR(200)	Si	Observaciones

Una vez definidas las funcionalidades del software, las entidades la base de datos y sus relaciones se procede a realizar el diseño del frontend y backend. El frontend corresponde a los elementos que el usuario puede ver y con los que puede interactuar, por ejemplo, los colores, botones, tablas, imágenes, etc., por otro lado, el backend corresponde a los eventos que le dan funciones a los botones. La parte del frontend denominamos interfaz de usuario y se detalla a continuación.

Interfaz de usuario

Debido a la popularidad y al gran soporte que brinda la herramienta de desarrollo Qt Designer se seleccionó como herramienta para el desarrollo de la interfaz de usuario y para cumplir con los requerimientos funcionales del software se diseñó una ventana para el control de acceso denominada “acceso” y una ventana con 14 páginas denominada "inicio" la cual consta de; una página para visualizar y editar la información de la hacienda, 11 páginas de registro, una página de consultas y una página para cargar los datos desde el sistema de medición de la producción de leche hacia la base de datos. La interfaz de usuario con sus respectivas páginas se presenta en el Capítulo 4.

Backend

El backend como se mencionó anteriormente corresponde a los eventos que le dan funciones a los botones. Para el desarrollo de las funcionalidades del software se hace uso del

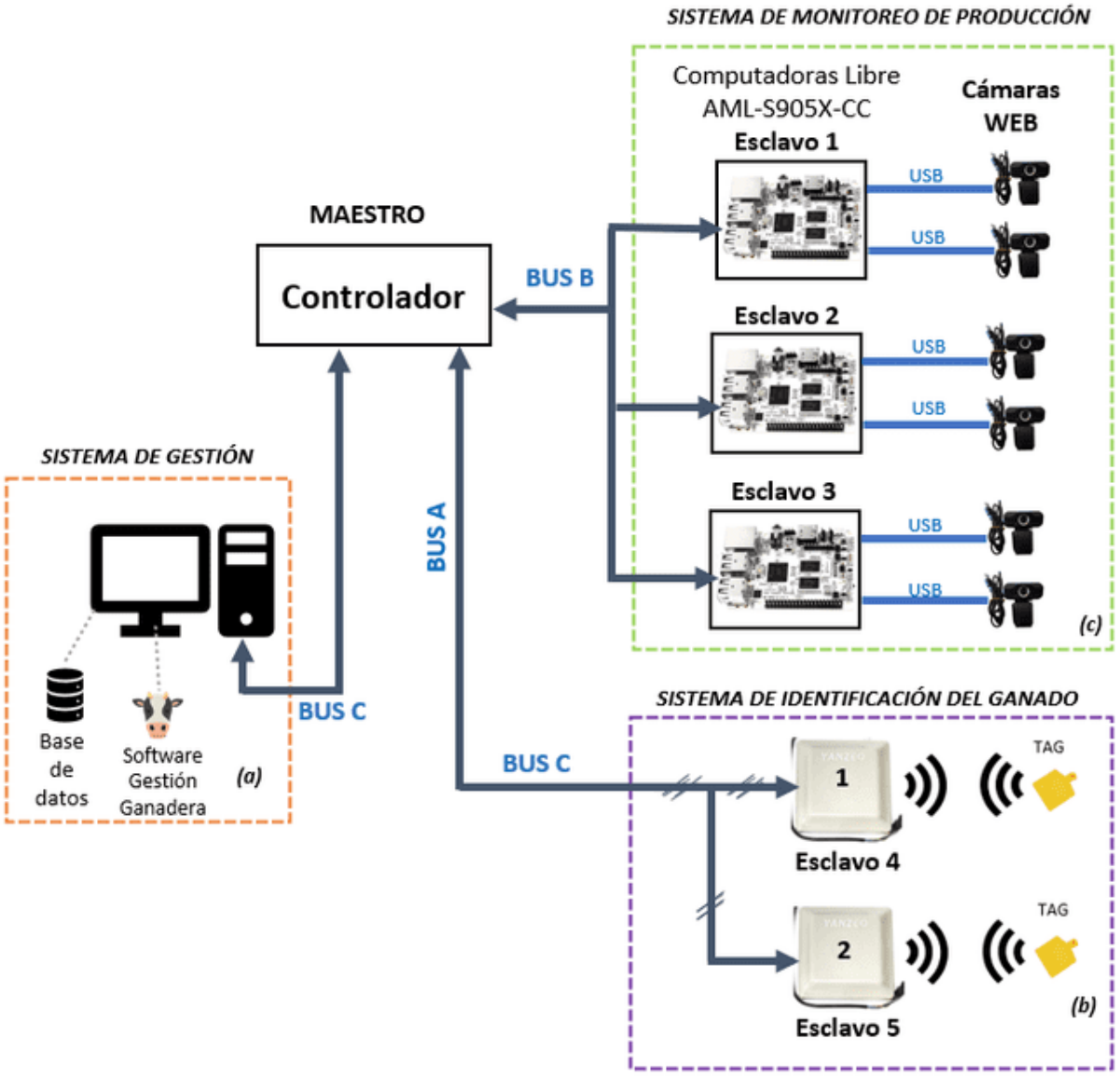
lenguaje de programación Python y la biblioteca de *pyside6.* Esto se detallará en el siguiente capítulo.

Diseño de la arquitectura integral del sistema

En este apartado se diseñará la integración de los tres sistemas que comprende el proyecto y que fueron analizados en la sección anterior: a) Identificación automática del ganado, b) Medición de producción lechera durante el ordeño y c) gestor de base de datos. La integración se realizará empleando un solo dispositivo controlador. Adicionalmente en esta sección se definirán los protocolos e interfaces de comunicaciones. En la Figura 44 se presenta un diagrama de bloques de la integración de los tres sistemas. En el diagrama se puede observar tres *BUSES,* el *BUS A que* realiza la comunicación entre el sistema de Identificación y el controlador, el *BUS B* permite la comunicación entre el sistema de medición de leche y el controlador, y el *BUS C* que comunica entre el sistema de Gestión y el controlador. Los buses B y C se comunicarán mediante protocolo MODBUS, en tanto que el BUS A emplea el protocolo propietario de la lectora, mismo que fue explicado en la sección *diseño del sistema de identificación del ganado.* El resultado final del proceso de integración es tener un sistema completo que permite que el sistema de gestión tenga acceso a la identificación del ganado y a la medición de la producción individual de manera automática.

Figura 44

Diagrama de bloques del sistema integral



Nota. (a) Sistema de gestión, el cual está formado por la computadora que consta con la base de datos y el software de gestión ganadera (b) Sistema de identificación del ganado conformado por la Odroid XU4 y los lectores RFID (c) Sistema de monitoreo de producción de leche constituido por los esclavos conectados a las cámaras web

Determinación de protocolos e interfaces de comunicación

Comunicación entre el sistema de identificación y el controlador. En la sección *diseño del sistema de identificación del ganado* se definieron para la comunicación, la interfaz RS-485 y el protocolo propietario de las lectoras. Para facilitar la implementación del protocolo de comunicación en el controlador se hará uso del lenguaje de programación Python, pues cuenta con librerías que permite la lectura y escritura en el puerto serial de una manera muy simple y transparente.

Comunicación entre el sistema de medición de la producción y el controlador. La comunicación entre el dispositivo de control y el sistema de medición de la producción deben emplear como medio físico una interfaz RS-485 debido al ruido eléctrico que genera la bomba de vació del sistema de ordeño mecánico y así evitar posibles errores en la transferencia de datos. En cuanto al protocolo de comunicación, se emplea el protocolo MODBUS que trabaja en la capa de aplicación del modelo OSI cuyo objetivo es realizar la transmisión de información entre varios dispositivos conectados en una misma red, además de ser el protocolo más estandarizado en el sector industrial.

Comunicación entre el sistema de gestión y el controlador. El sistema de gestión, cumple la función de un esclavo dentro del sistema integrado, a diferencia de los otros sistemas este recibe datos del controlador correspondientes a las mediciones de producción y tags de identificación, por tal motivo para ahorrar recursos en cuanto a hardware (puertos de comunicación), el sistema de gestión se integra al mismo bus del sistema de medición de la producción.

Selección del controlador

El controlador es el elemento principal del sistema, es el que controla el tráfico de datos en los buses, además en él se almacenará de manera temporal los datos de medición de la leche y las etiquetas de identificación de las vacas ordeñadas para ser enviadas al sistema de

gestión cuando este lo solicite. Por tal motivo es importante seleccionar adecuadamente el controlador. A continuación se mencionará algunos requisitos que debe cumplir el controlador y sobre los cuales se basa la selección del mismo.

a) Permita instalar Python, puesto que para la implementación del protocolo de comunicación del sistema de identificación se empleará la librería *pyserial*.

b) Disponga de al menos tres puertos USB, uno que se usará para la comunicación con el sistema de identificación empleando la interfaz RS-485, otro para la comunicación con el sistema de gestión y otro para la comunicación con el sistema de medición de la producción.

c) Permita trabajar con el sistema operativo Linux o sistemas operativos basados en Linux ya que se puede tener un mayor acceso a las configuraciones de hardware desde la terminal.

d) Permita trabajar de manera ilimitada en el tiempo sin interrupciones.

e) Permita el manejo de archivos planos.

f) Debe ser económico, ya que se busca tener un sistema accesible para medianos y pequeños ganaderos.

Con base en los requisitos mencionados anteriormente se seleccionó la SBC Odroid XU4, cuyas características principales se detallan a continuación.

- Posee un procesador octa-core a 2 GHz
- 2GB de RAM
- Cuenta con almacenamiento flash integrado
- Posee 2 puertos USB 3.0, 1 puerto USB 2.0
- Puerto Ethernet

- Salida HMDI
- También cuenta con GPIOs
- Compatible con sistemas operativos como Ubuntu, Android.

La Odroid-XU4 es una minicomputadora con altas prestaciones, se trata de una plataforma de desarrollo de software, así como también de hardware, dentro de sus características está su procesador octa-core a 2 GHz, 2GB de RAM, su almacenamiento flash integrado, cuenta con 2 puertos USB 3.0, 1 puerto USB 2.0, un puerto Ethernet y una salida HDMI, además cuenta con GPIOs. Compatible con sistemas operativos como Ubuntu, Android, Debian, entre otros (Rojo, 2016). Se muestra en la Figura 15 los componentes por los que está compuesta esta SBC.

Adicionalmente, debido a que se están trabajando con puertos USB tanto en el controlador como en los sistemas de medición y gestión es necesario convertir las señales a nivel da capa física, para esto se emplean convertidores de USB a RS-485. En la Figura 45 se representa

Figura 45

Esquema del diseño del sistema integral para la identificación del ganado, monitoreo de producción de leche y gestión ganadera

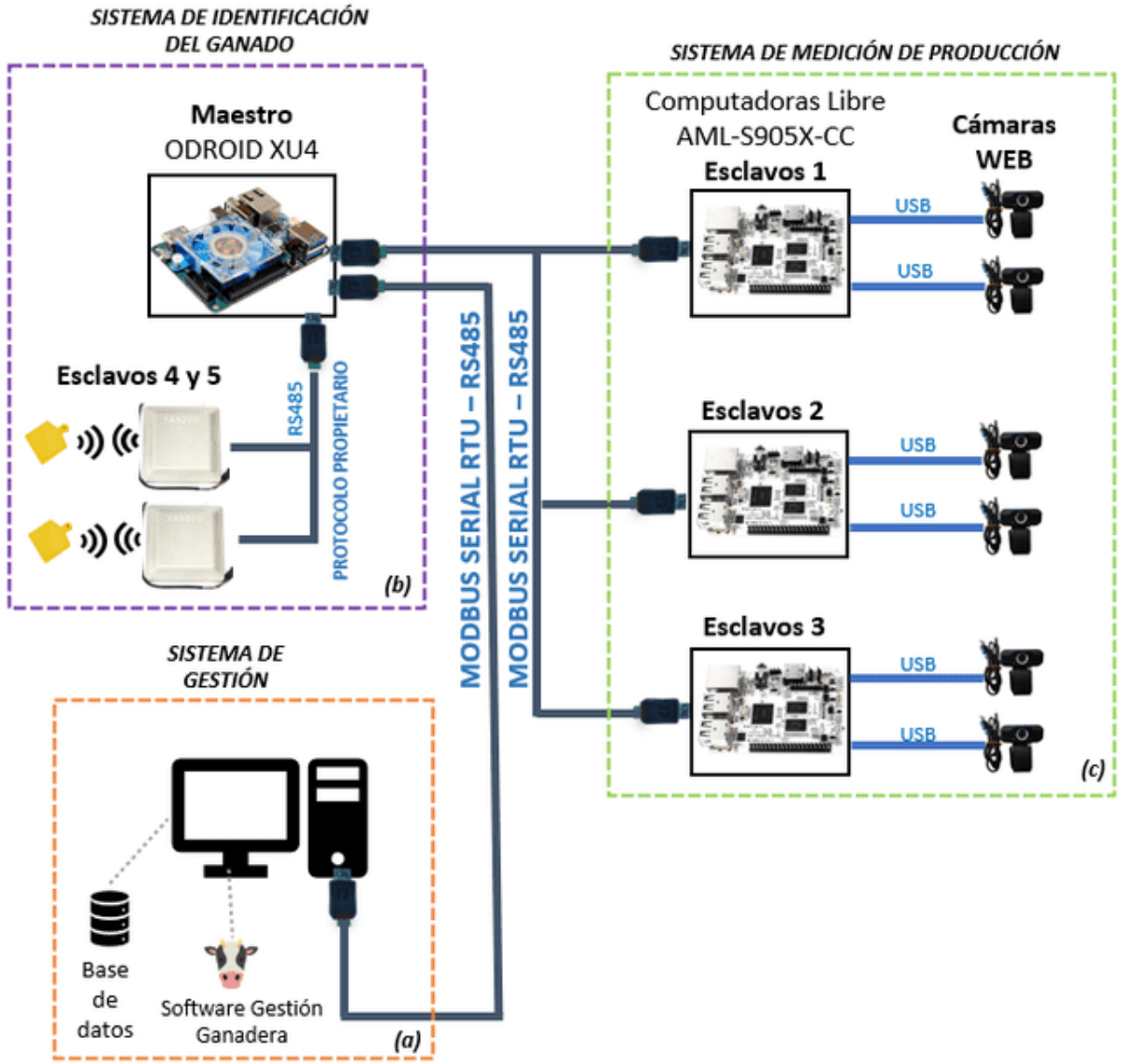
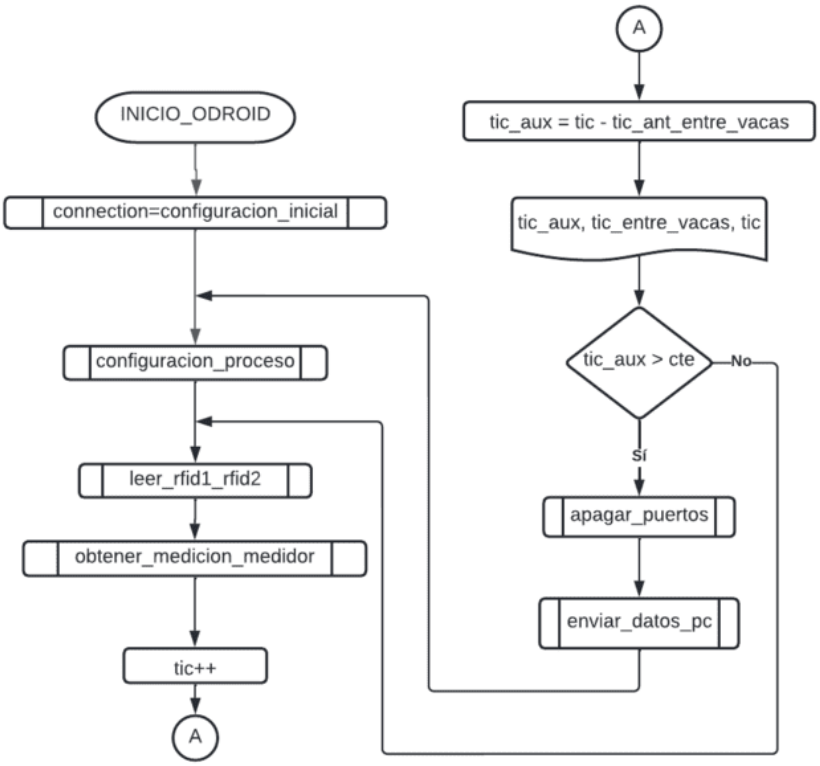


Diagrama de flujo del sistema integral: Maestro

Una vez definido el protocolo de comunicación, se establece la lógica de programación, es decir, la función que realiza el controlador. En la Figura 46 se muestra el diagrama de flujo que describe su funcionamiento.

Figura 46

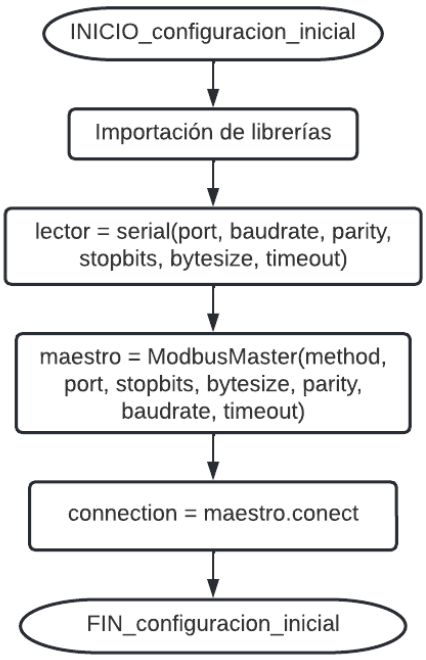
Diagrama de flujo principal para el funcionamiento de la Odroid



Primero el controlador realiza el proceso de configuración inicial que se muestra en el diagrama de flujo de la Figura 47, dentro del cual se importan las librerías, se establece la comunicación MODBUS, y la comunicación con los lectores (protocolo propietario), si la comunicación se realiza correctamente la función devuelve un True caso contrario devuelve un False.

Figura 47

Diagrama de flujo de la función configuracion_inicial de la Odroid



Luego el controlador inicia la configuración de proceso, que consiste en declarar e inicializar variables, ver Figura 48. En la Tabla 52 se presenta una descripción de cada variable declarada y el valor de inicialización.

Tabla 52

Descripción de variables

Variable	Tipo	Valor inicial	Descripción
enc_med	vector	[170,170,170]	Indica el estado de los puertos (Encendido/Apagado)
tag_ant	Int	0	Número de etiqueta anterior del lector 0
tag_ant1	Int	0	Número de etiqueta anterior del lector 1

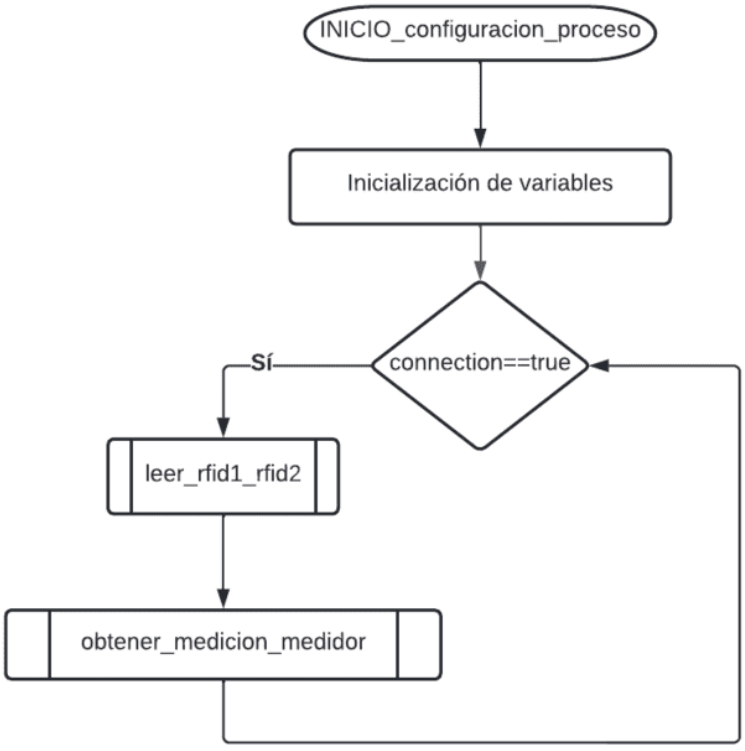
num_lector	Int	170	Almacena la identificación de los lectores. Si es el lector 0, asigna un 0 y si es el lector 1, asigna un 1.
primer_lado	Int	170	Especifica el lado por donde ingresa la vaca.
Prioridad	vector	[0,0,0,0,0,0]	Variable para especificar por qué lado entraron las vacas al inicio del ordeño.
cont_lectora0	Int	0	Cuenta el número de tags que detecta el lector 0.
cont_lectora1	Int	0	Cuenta el número de tags que detecta el lector 1.
encontró	Int	0	Bandera para indicar si una vaca extra ya fue registrada.
vaca_extra	Int	0	Cuenta las vacas extras ingresadas a la sala, es decir cuando entran mas de 6 en un mismo lado.
tag_puestos	vector	[[0,0,0,0,0,0], [0,0,0,0,0,0]]	Almacena el valor de los tags ingresados a la sala de ordeño.
num_vacas	vector	[0,0,0,0,0,0]	Cuenta las vacas ingresadas a la sala.
Tic	Int	0	Determina el tiempo transcurrido desde ingresa un animal a la sala hasta que finaliza el ordeño.
tic_ant0	Int	0	Cuenta el tiempo que tarda en ordeñarse 6 vacas del lado del lector 0.

tic_ant1	Int	0	Cuenta el tiempo que tarda en ordeñarse 6 vacas del lado del lector 1.
tic_aux	Int	0	Variable auxiliar del tic.
tic_entre_vacas 0	Int	1	Determina el tiempo entre vaca al entrar a la sala de ordeño por el lado del lector 0.
tic_entre_vacas 1	Int	0	Determina el tiempo entre vaca al entrar a la sala de ordeño por el lado del lector 1.
Medida	vector	[0,0,0,0,0,0,0,0 ,0,0,0,0]	Almacena los valores de las mediciones de cada grupo de ordeño.
med_ant	vector	[0,0,0,0,0,0]	Almacena los valores de las mediciones anteriores
med_act	vector	[0,0,0,0,0,0]	Almacena los valores de las mediciones actuales
tag_nuevo0	Int	0	Auxiliar para el valor del tag actual del lector 0
tag_nuevo1	Int	0	Auxiliar para el valor del tag actual del lector 1
Tag	Int	0	Valor del tag actual
num_med	Int	0	Especifica el número de medidor
Longitud	Int	0	Especifica el tamaño de bytes de datos recibidos de los lectores
medida_int	Int	0	Valor del medidor de leche

error_enviar_slave1	Int	1	Bandera para verificar el envio de datos a través del modbus.
aux_enc	Int	0	Variable que determina si el esclavo está encendido o apagado

Figura 48

Diagrama de flujo de la función configuracion_proceso de la Odroid

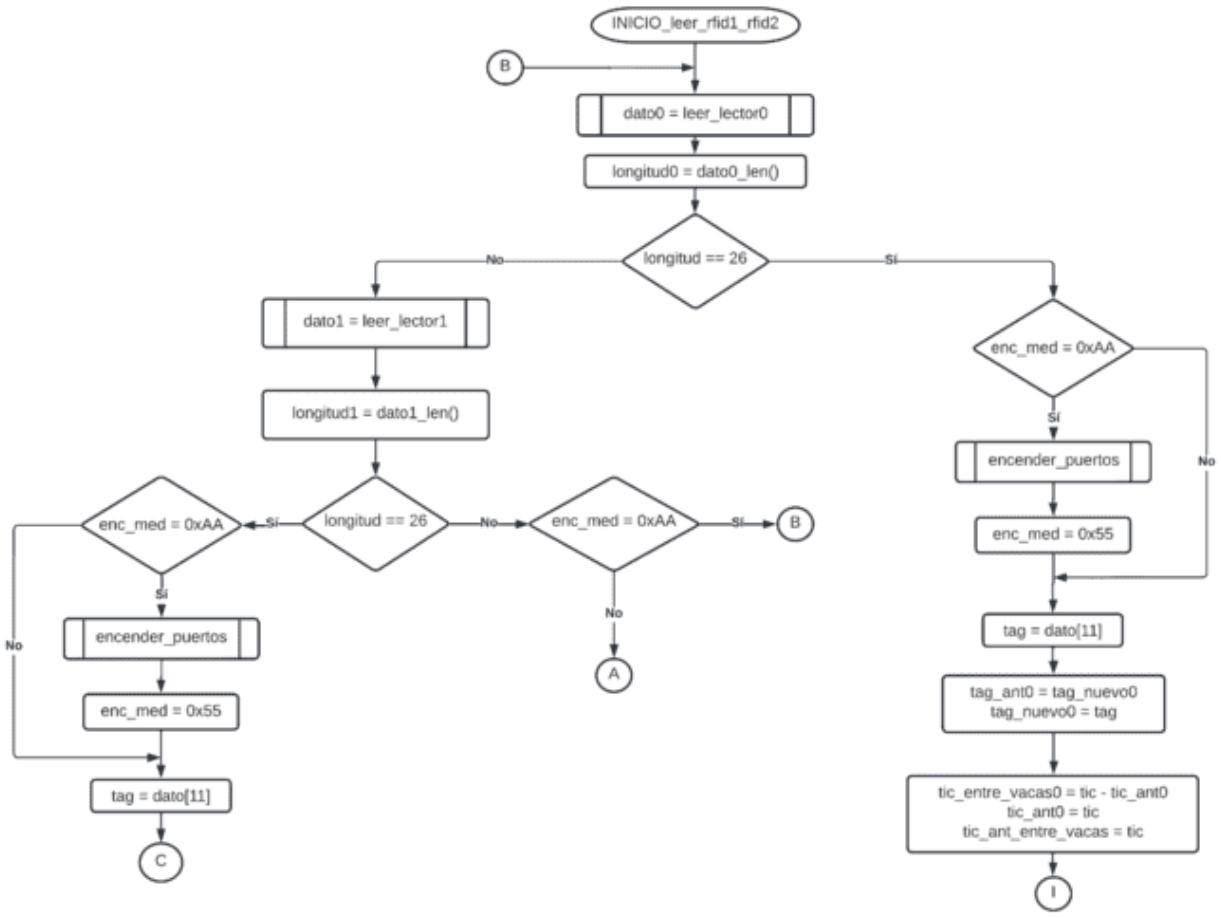


Una vez inicializada las variables el controlador realiza peticiones de lectura de etiquetas, ver diagrama de la Figura 50, a cada lector de manera alternada como se muestra en el diagrama de flujo de la Figura 49, el lector responde siempre con una trama de datos el cual se guarda en la variable dato0 o dato1 dependiendo del número de lectora solicitado. Si la longitud datos que recibe el controlador como respuesta de los lectores es igual a 26 significa que se ha detectado una vaca, por lo tanto, se debe inicializar el proceso de ordeño, es por ello

que se ejecuta la función encender puertos, cuyo diagrama de flujo se detalla en la Figura 49. El número de identificación de la vaca se almacena en la variable tag.

Figura 49

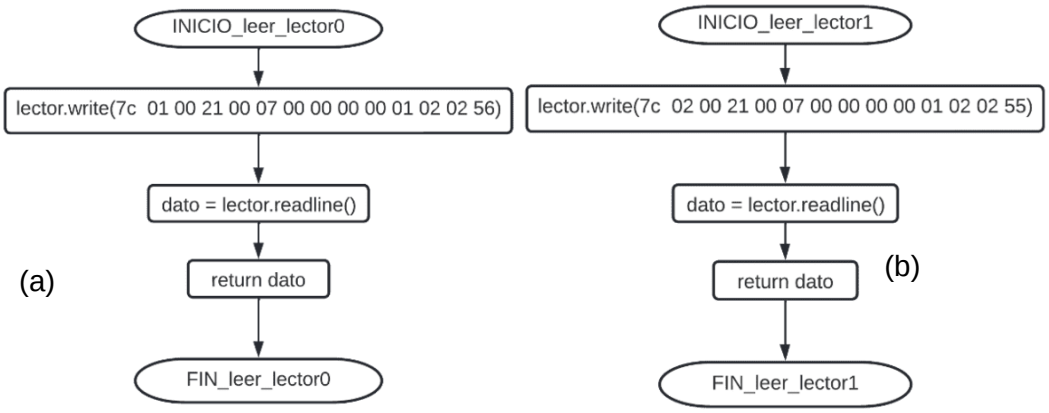
Diagrama de flujo de la función leer_rfid1_rfid2, primera parte



Para realizar una lectura de etiquetas RFID el controlador envía un comando de solicitud de EPC al lector a través del puerto serial, este comando se determinó en la sección *Diseño del sistema de identificación del ganado,* posteriormente el controlador lee la respuesta del lector y almacena el dato en la variable tag. Esto se realiza tanto para el lector 1 como para el lector 2, ver Figura 50.

Figura 50

Diagramas de flujo de las funciones leer_lector0 y leer_lector1

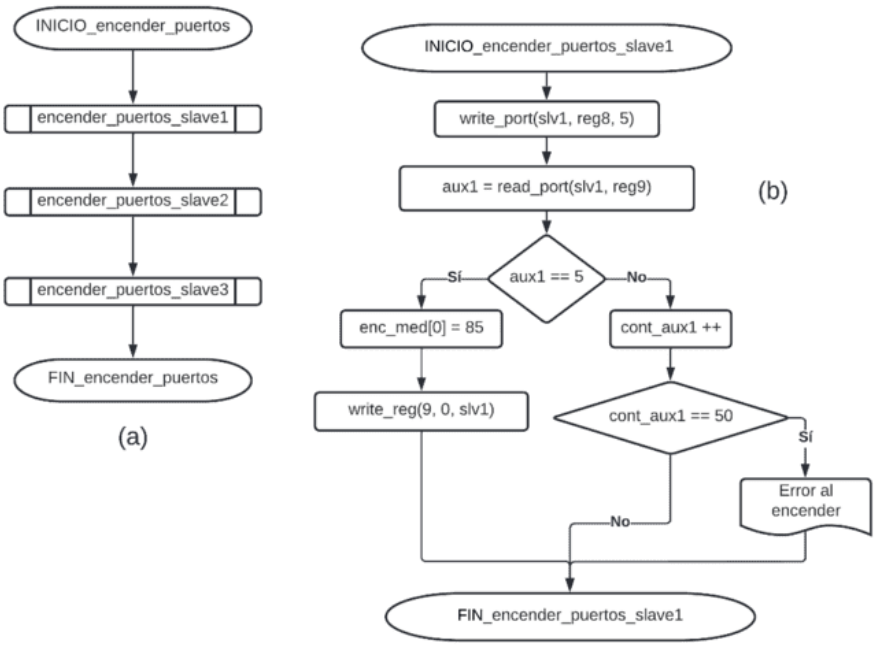


Nota. (a)leer_lector0 (b) leer_lector1

El proceso de encender puertos USB's de cada esclavo se lleva a cabo de la siguiente manera; El controlador envía una petición al esclavo 1 del sistema de medición mediante la escritura del dato "05H" en el registro 08H y el esclavo responde con un dato "06H" en el registro 09H en el caso de que se haya encendido los puertos de manera correcta y asigana a la variable enc_med[num_slv] un valor de "85", caso contrario si hubo un error en el encendido de puertos, devuelve un dato "00H" y el controlador vuelve a realizar la lectura del registro 09H, si continúa sin obtener una respuesta de encendido correcto el controlador vuelve a realizar una petición de encendido por un número n de veces, si no se encienden los puertos tras estos intentos, el controlador enviará un mensaje de error de encendido de puertos. Este proceso se realiza para los tres esclavos como se muestra en el apartado a de la Figura 51.

Figura 51

Diagrama de flujo de las funciones encender_puertos y encender_puertos_slave1

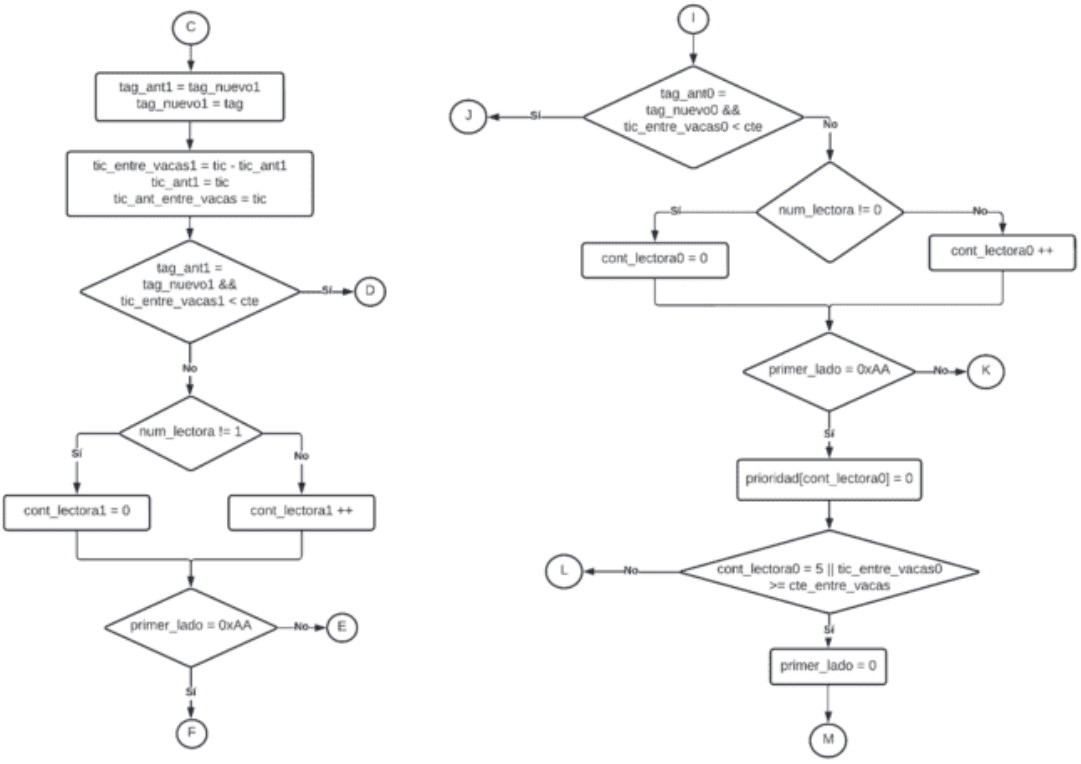


Nota. (a) Función encender_puertos (b) Función encender_puertos_slave1

Una vez encendido los puertos de cada esclavo del sistema de medición de la producción, el controlador compara si el tag leído es igual al anterior y que el tiempo entre vacas sea menor que una constante de tiempo, si no se cumple una de estas dos se incrementa el contador num_lectorax donde x representa el número de la lectora, además asigna la prioridad de la lectora x cuando se trata de la primera vaca ordeñada, en la Figura 52 se observa el diagrama de flujo de lo antes expuesto.

Figura 52

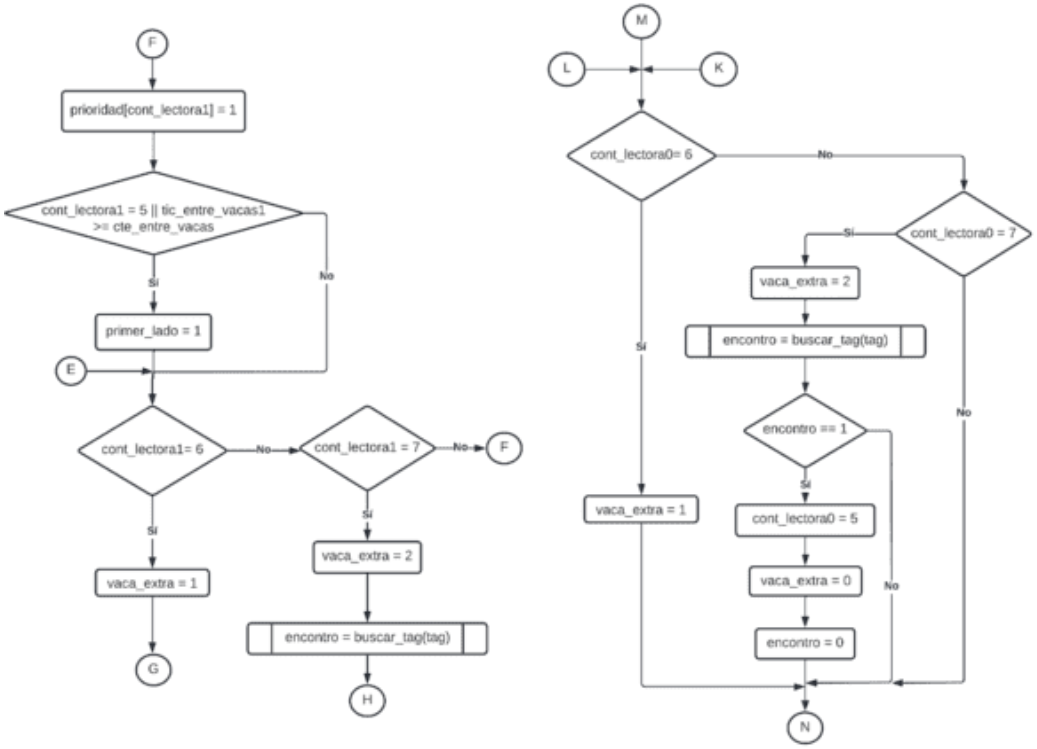
Diagrama de flujo de la función de flujo de la función leer_rfid1_rfid2, segunda parte



Antes de guardar el número de identificación de la vaca se verifica si el número de vacas ingresadas por lado no sea mayor a 6, esto se debe a que algunas veces entran más de 6 vacas a la sala, pero luego son extraídas de la sala de ordeño. Por lo tanto si existe más de seis vacas se incrementa la variable vaca_extra, en el caso de que se detecte 8 tags en el mismo lado, se verifica si es de una vaca antes ingresada o un tag diferente, si es un tag ya registrado significa que se extrajo la vaca, caso contrario se tendrán dos vacas extras, esto se puede ver en la Figura 53.

Figura 53

Diagrama de flujo de la función leer_rfid1_rfid2, tercera parte



Para el funcionamiento del diagrama de flujo de la Figura 54, se crea la variable *encontro* que sirve para indicar si es que el tag ha sido detectado y retornará el valor del tag, sea considerado esta acción para relacionar la lectura de los seis primeros tags leídos con los seis puestos de ordeño, a través de cada lector RFID. Finalmente, en la Figura 55 se indica la última parte del funcionamiento del Maestro, se observa que cada lector RFID es el encargado de devolver el tag de las vacas y el puesto de ordeño

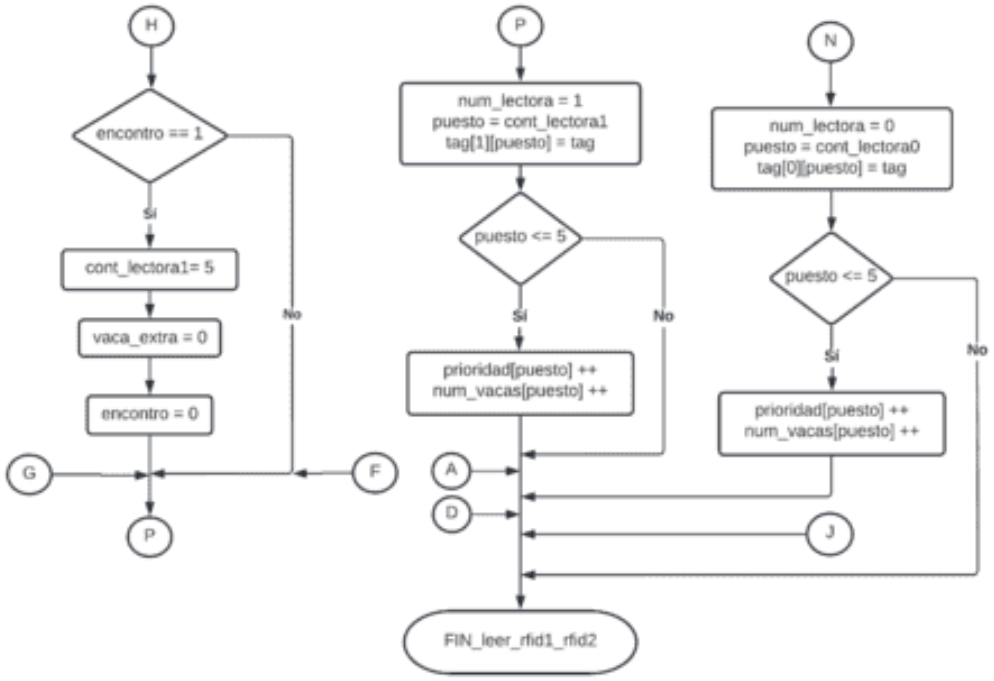
Figura 54

Diagrama de flujo de la función buscar_tag

Finalmente se guarda el número de identificación del animal registrado en la variable tag en el vector tag_puestos y se incrementan los contadores de prioridad y num_vacas. El mismo procedimiento se realiza para el lector 1 es decir cuando las vacas entran por el lado derecho.

Figura 55

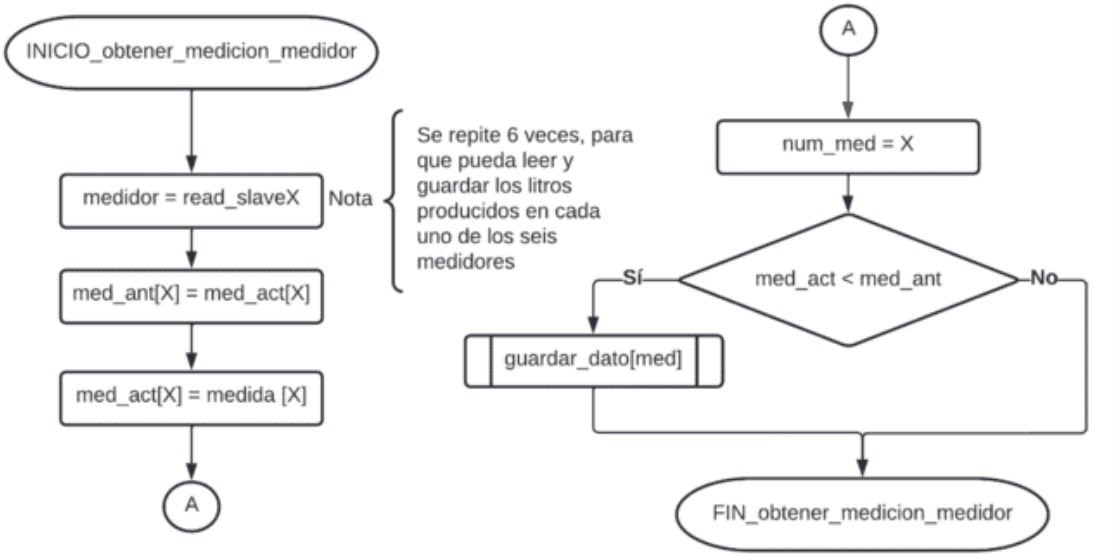
Diagrama de flujo de la función leer_rfid1_rfid2, cuarta parte



Una vez guardado los tags de las vacas ingresadas a la sala de ordeño, inicia el proceso de ordeño, por lo tanto, el controlador realizará el proceso de obtener medición de leche de cada esclavo. El controlador recibe los valores del medidor en todo momento durante el proceso de ordeño de una vaca y debe guardar el dato en el vector medida cuando el ordeño se termina. Para determinar el fin del ordeño de una vaca se compara los últimos valores de medición recibidos y si el valor actual es menos al valor anterior entonces significa que el ordeño culminó.

Figura 56

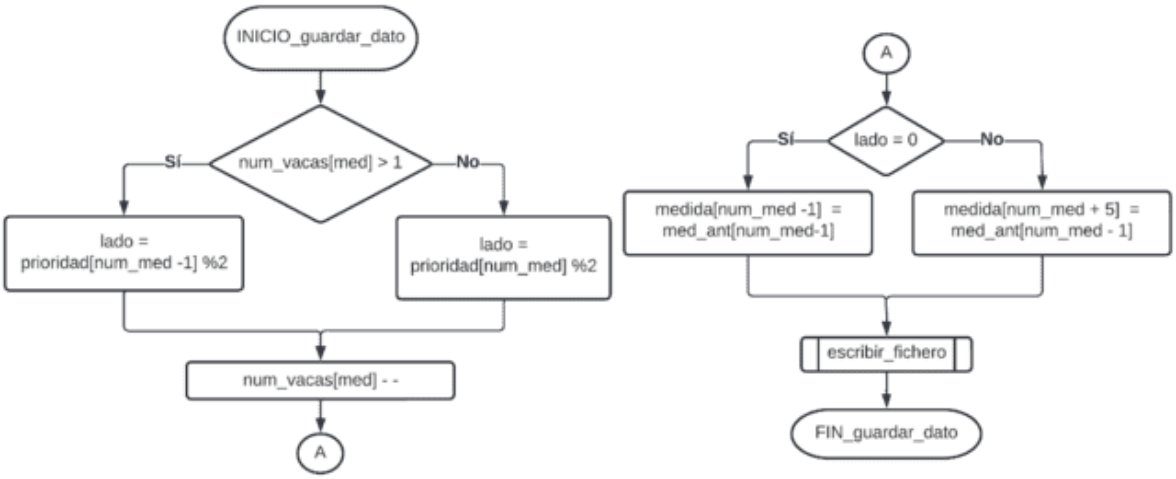
Diagrama de flujo de la función obtener_medicion_medidor



Luego de tener los datos de las etiquetas y las mediciones almacenadas en sus respectivos vectores se procede a guardar en un archivo plano, teniendo así una lista de la etiqueta con su respectiva medición de la producción. El proceso escribir fichero de la Figura 57hace la función de guardar los datos de etiquetas y mediciones en un archivo plano, y posteriormente se eliminan los valores almacenados en los vectores tag_puestos y medida.

Figura 57

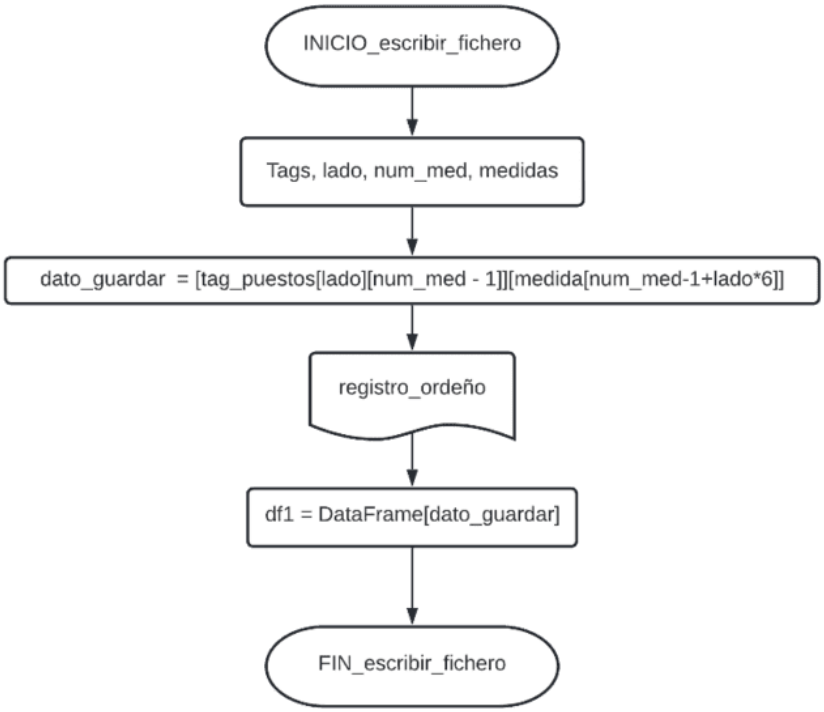
Diagrama de flujo de la función guardar_dato



En la Figura 58 se muestra el diagrama de flujo para la función de escribir datos en el fichero, que puede ser un archivo .txt, .csv, etc.

Figura 58

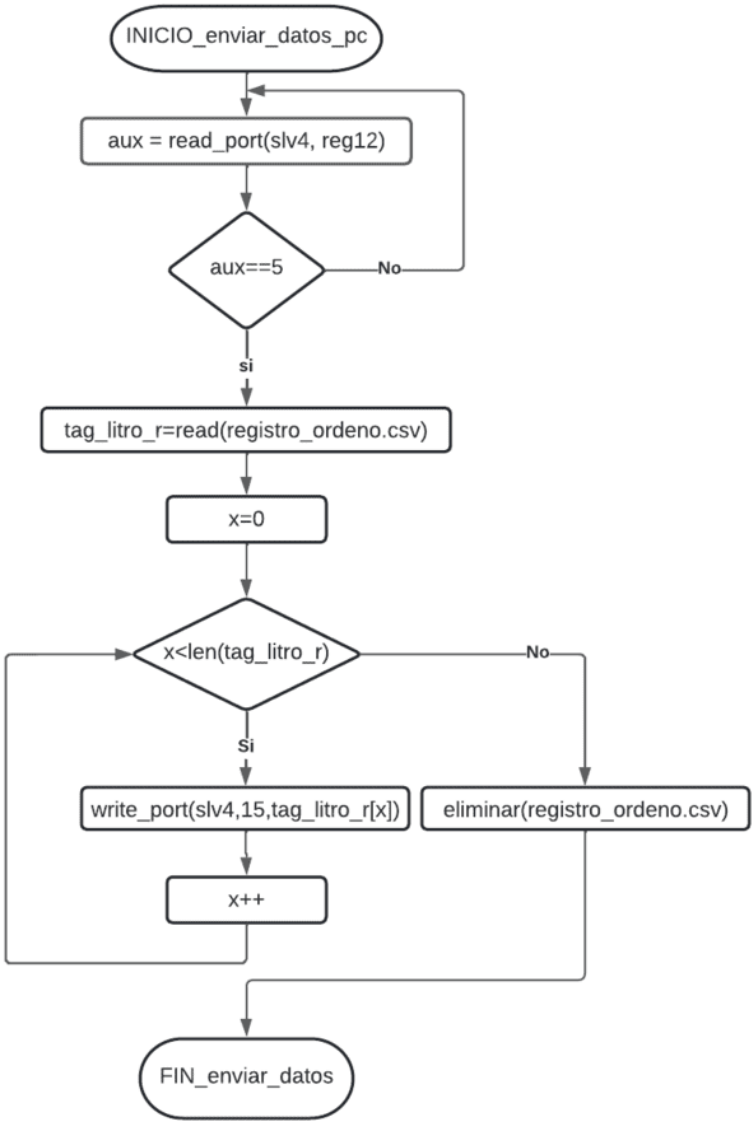
Diagrama de flujo de la función escribir_fichero



Con los datos almacenados en el archivo plano, el maestro los envía hacia el sistema de gestión siempre que este lo solicite, es importante mencionar que el maestro enviará los datos del archivo plano únicamente cuando se haya terminado el turno de ordeño y los puertos de los esclavos 1, 2 y 3 estén apagados. En la Figura 59 se muestra el diagrama de flujo del proceso de envío de datos del archivo plano desde el maestro hacia el esclavo, el cual funciona de la siguiente manera, el maestro en todo instante, luego de terminar el proceso de ordeño, verifica si el esclavo 6 solicita el envío de datos, esto lo hace a través del registro 12, si este registro es igual a 05H significa que el sistema de gestión está solicitando el envío de datos por lo tanto el maestro procede a enviar dichos datos. Terminado el envío de datos, el maestro elimina los el archivo plano.

Figura 59

Diagrama de flujo de envío de datos del maestro al sistema de gestión



El maestro realiza todo el proceso de manera indefinida.

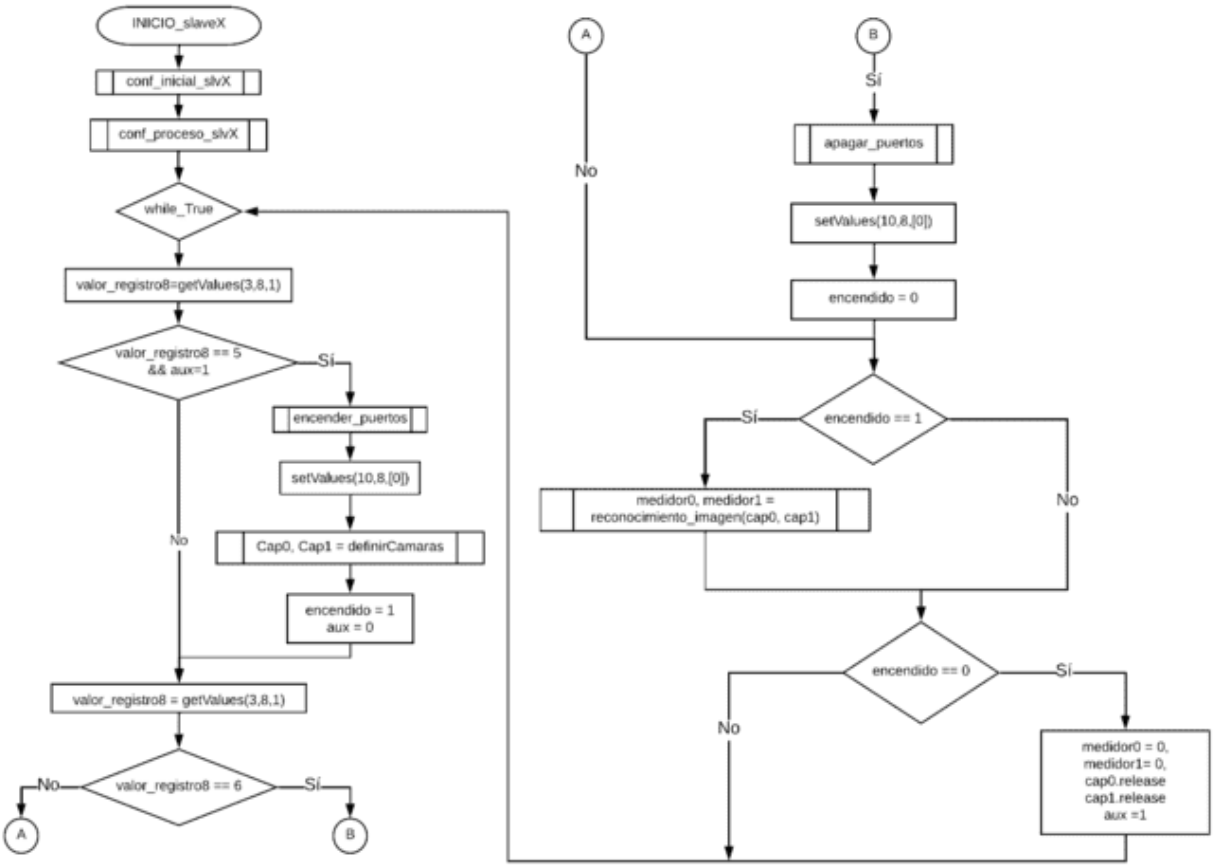
Diagrama de flujo del sistema integral: Esclavo 1, 2 y 3

Una vez realizado los diagramas de flujo de la parte del Maestro, los cuales contienen la lógica del funcionamiento del mismo, se debe realizar los diagramas de flujo para los esclavos. En esta sección se definirá la lógica de funcionamiento de los esclavos 1, 2 y 3 que

corresponden al sistema de medición de la producción de leche, los cuales al tener igual funcionamiento basta con indicar de solamente uno. En el diagrama de flujo de la Figura 60 se especifica el funcionamiento del esclavo dentro del sistema integral.

Figura 60

Diagrama de flujo principal del funcionamiento de los esclavos



Nota. Las funciones que se usan son getValues(registro, dirección, posición), setValues(registro, dirección, valor). Se destaca que el valor de registro si es 3 su función es de lectura y si es 10 en cambio es para la escritura.

El esclavo inicia su funcionamiento con la la configuracion inicial en la cual se importan las librerías, se definen los parámetros del protocolo de comunicación y se establece la comunicación , luego se realiza la configuracion del proceso que consiste en definir e inicializar

las variables. A continuacion, en la Tabla 53 se describen las variables del esclavo del sistema de medición de la producción.

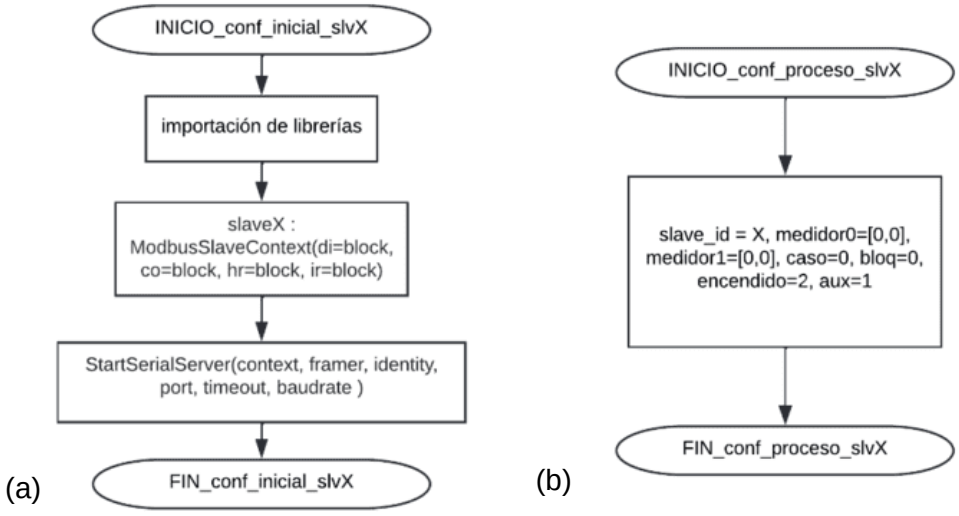
Tabla 53

Descripción de variables del esclavo del sistema de medición de la producción.

Variable	**Tipo**	**Valor inicial**	**Descripción**
slave_id	Int	1	Especifica el número de esclavo
medidor0	tupla	[0,0]	Almacena el valor del medidor 0
medidor1	tupla	[0,0]	Almacena el valor del medidor 1
Caso	Int	0	Variable auxiliar para el proceso reconocimiento_imagen.
Bloq	Int	0	Variable auxiliar para el proceso reconocimiento_imagen.
Encendido	Int	0	Variable para indicar si los puertos están encendidos o apagados.
aux1	Int	1	Variable auxiliar para indicar si se encendieron los puertos

Figura 61

Diagramas de flujo de las funciones conf_inicial_slvX y conf_proceso_slvX

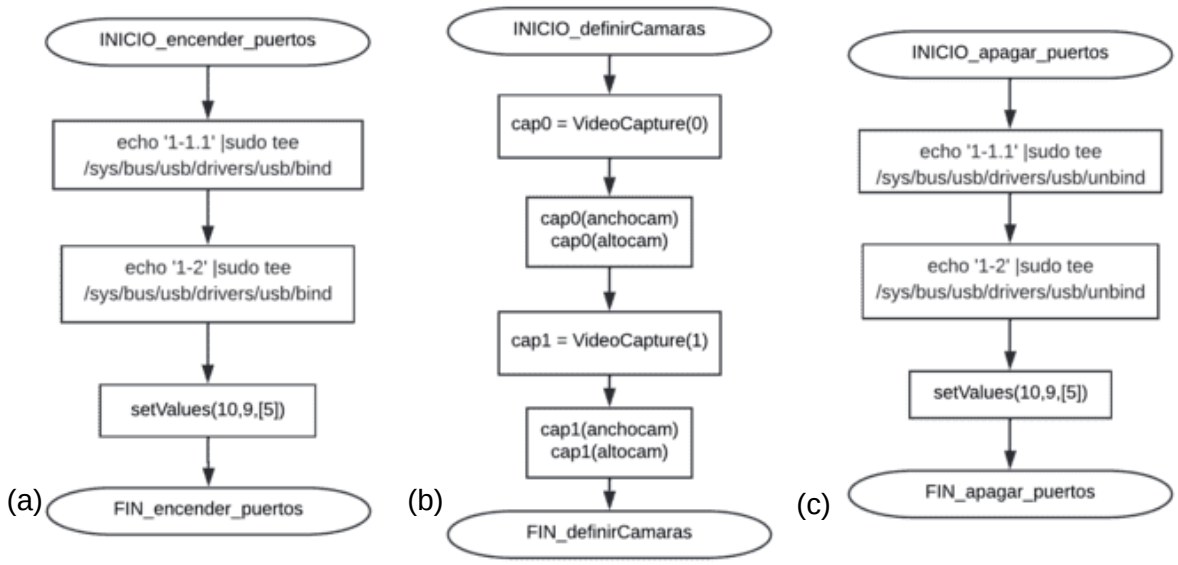


Nota. (a) conf_inicial_slvX, (b) conf_proceso_slvX

Realizada la configuración del proceso el esclavo entra en un bucle esperando la orden del maestro para encender los puertos, la orden se refleja en el registro 08H del esclavoX. Si el registro 08H es igual a 05H y los puertos aún no han sido encendidos entonces el esclavo realiza la función de encender puertos, esta función se describe en la Figura 62 inciso a, para no repetir el proceso de encender puertos se escribe el valor 00H en el registro 08H. Con los puertos encendidos, se tiene acceso a las cámaras de los puertos USB´s, por lo tanto, el esclavo realiza el proceso de definir cámaras. El proceso de definir cámaras es muy importante ya que aquí se define el ancho y el alto del cuadro a ser capturado, además se especifica el puerto donde está conectado la cámara y dependiendo de ello se asigna como medidor0 o medidor1, esto se indica en la Figura 62, inciso b.

Figura 62

Diagramas de flujo de las funciones encender_puertos, definirCamaras y apagar_puertos



Nota. (a) encender_puertos, (b) definirCamaras, (c) apagar_puertos

Una vez encendido los puertos se inica el proceso de reconocimiento de imágenes. El esclavo llama a la función reconocimiento_imagen() que acepta como parámetros cap0 y cap1, ver diagrama de flujo de la Figura 60, y devuelve el valor correspondiente a la medición de leche, este valor se escribe en un registro para que el maestro pueda acceder a él. El esclavo además está pendiente de la petición de apagado que hace le hace el maestro, esta petición se refleja con un valor 06H en el registro 08H. La función correspondiente al apagado de los puertos del esclavo se observa en la Figura 62 inciso c.

Diseño de la aplicación que se ejecuta en el servidor de la Base de datos

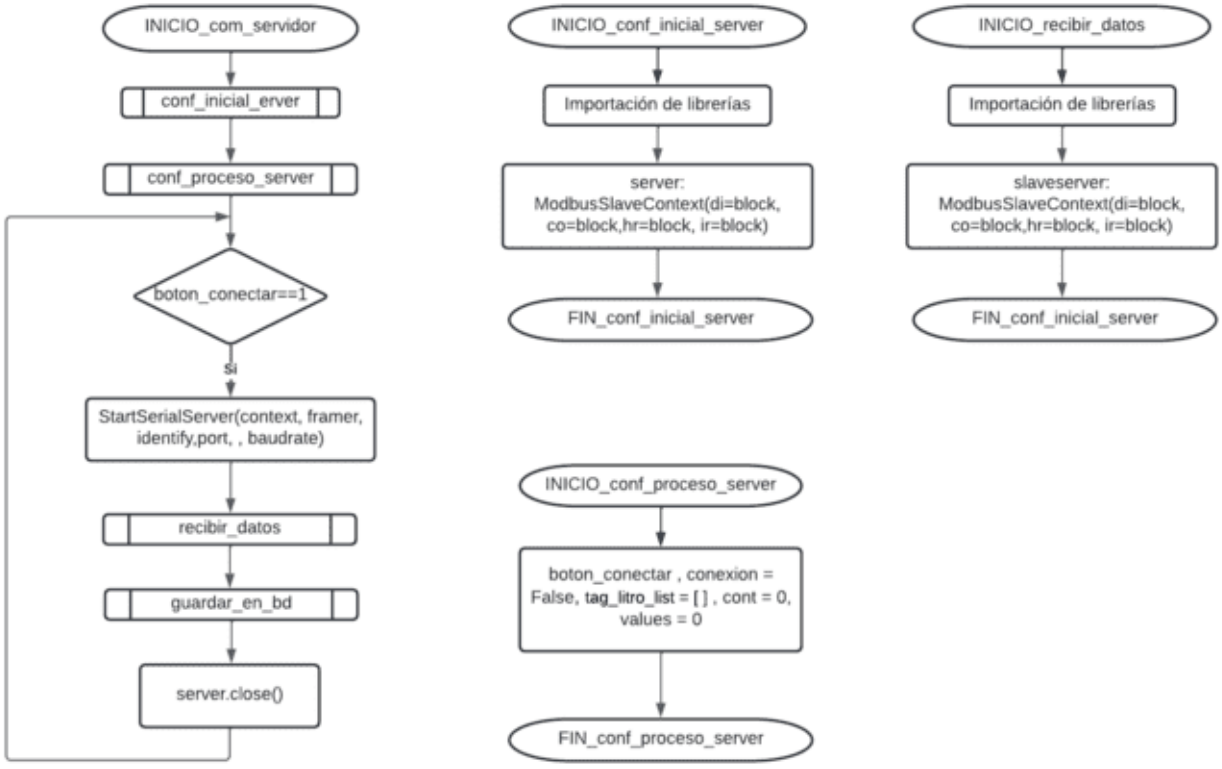
El servidor corresponde a la computadora que realiza el proceso de gestión, es el encargado de ejecutar la aplicación de escritorio que lleva el registro de las actividades pecuarias dentro de la hacienda. Por lo tanto, es necesario que este sistema esté conectado con el controlador maestro. Para llevar a cabo la comunicación del sistema de gestión con el maestro se crea un nuevo MODBUS y de esta forma actuará como maestro. En el diagrama de flujo de la Figura 63 se presenta el proceso que realiza el servidor para comunicarse con el

maestro. En primer lugar, el servidor realiza la configuración inicial y la configuración de proceso que consiste en establecer los parámetros de comunicación del MODBUS y asignar e inicializar las variables del proceso. Es importante mencionar que esta función del servidor es un módulo de la aplicación de escritorio que también forma parte de este esclavo.

El servidor se conecta al MODBUS solamente cuando se accione el botón conectar de la aplicación de escritorio, luego de la conexión se realiza el proceso de recibir datos y subir datos a la base de datos.

Figura 63

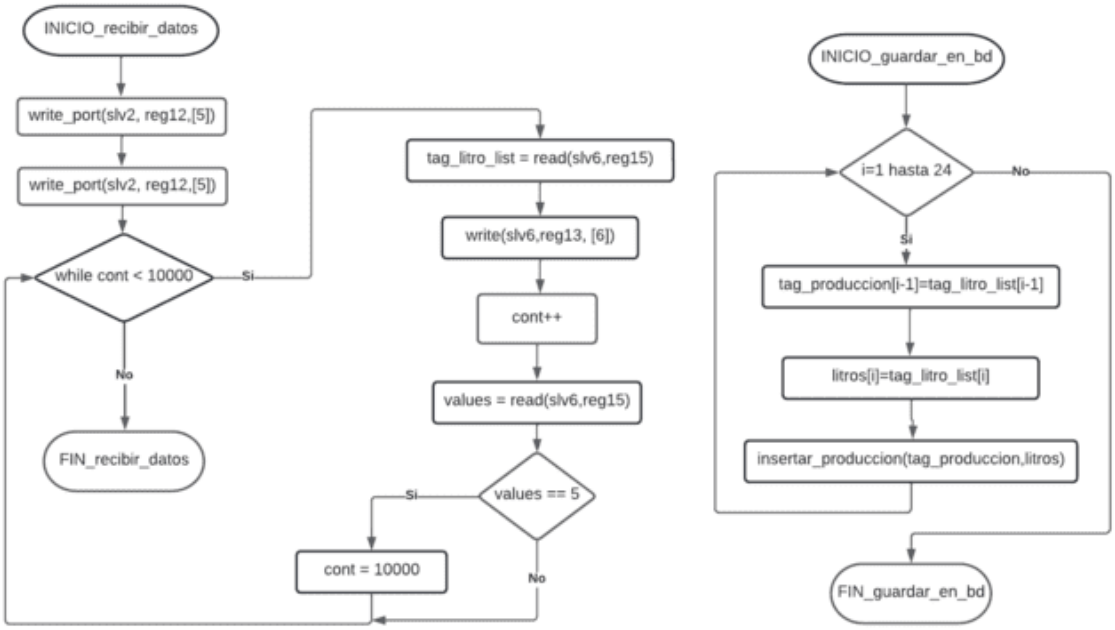
Diagramas de flujo servidor la comunicación con el maestro



En Figura **67** se muestran los diagramas de flujo de las funciones *recibir_datos* y *guardar_en_bd.*

Figura 64

Diagramas de flujo del servidor 6 para la comunicación con el maestro



Es muy importante tomar en cuenta que los protocolos de comunicación estarán funcionando todo el tiempo, tanto en el servidor como en el cliente, si el servidor necesita realizar otras tareas deberá hacerlo en paralelo con la implementación del MODBUS u otro protocolo. Para lograr esto es recomendable trabajar con hilos, que son muy usados en Python.

Capítulo 4: Implementación y pruebas de funcionamiento

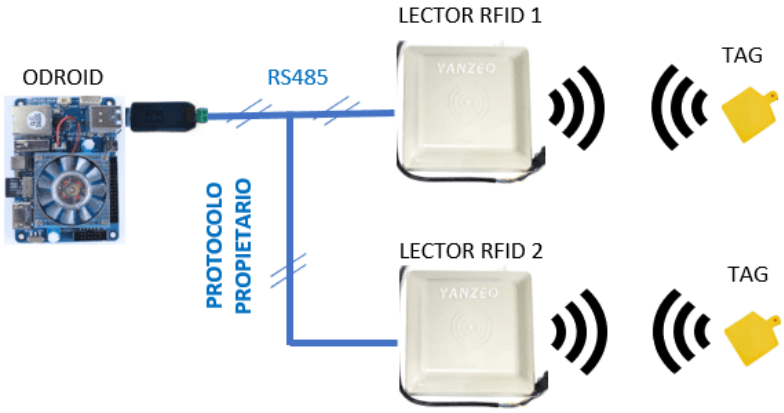
En el capítulo anterior se realizó el diseño de las tres etapas del proyecto y la forma de como integrarlas, además se seleccionaron los elementos de cada sistema, como controladores, lectores, tags RFID´s, protocolos de comunicación, etc., que son necesarios para la implementación del proyecto. Para la implementación se toma como punto de partida el esquema del diseño integral de la Figura 45.

Implementación del sistema de identificación automático del ganado

Para el sistema de identificación se tiene el esquema de la Figura 65, la SBC controla el bus y realiza las peticiones de lectura de etiquetas RFID, los lectores responden a dichas peticiones con un código de 26 bytes que incluye el código de identificación del ganado. Como ya se mencionó, el protocolo empleado es propietario de YANZEO y los comandos que permite la interacción maestro-esclavo se definieron en el capítulo anterior.

Figura 65

Esquema de conexión del sistema de identificación



Antes de iniciar con la implementación de código del sistema de la Figura 65 se debe preparar la SBC y realizar algunas configuraciones que son muy importantes para el correcto funcionamiento del sistema. Es importante mencionar que la SBC cumple con el papel de maestro tanto para el protocolo propietario de YANZEO (sistema de identificación) como para el MODBUS (sistema de medición de la producción de gestión).

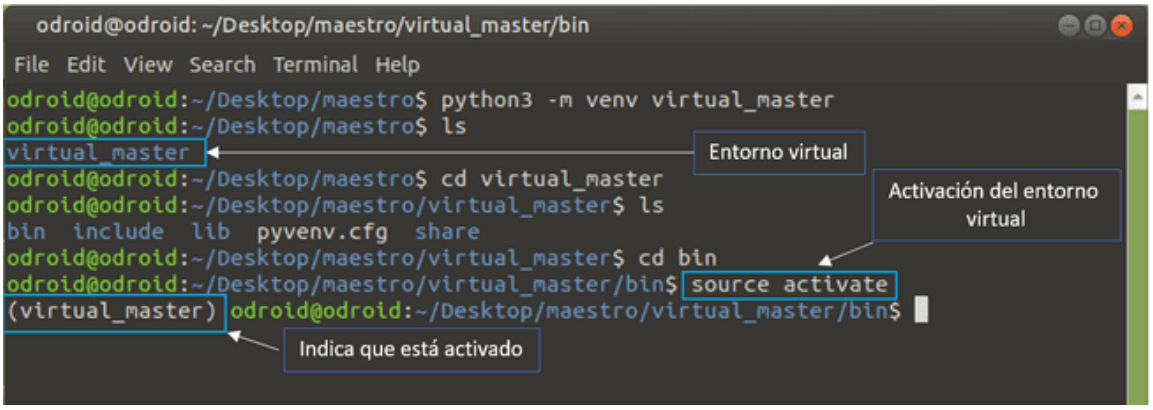
Configuración de la SBC Odroid utilizada como Maestro

En la SBC maestro se instaló el sistema operativo Ubuntu Mate 20.04 que permite la ejecución de los programas de manera rápida, dado esto se prosiguió a instalar el editor de texto Visual Studio Code. Ubuntu Mate 20.04 cuenta con la versión 2.7 de Python lo que implicó actualizar a la versión 3.7, que es la versión compatible con librerías que se instalarán más adelante.

Para evitar errores debido a futuras actualizaciones de librerías o paquetes de Python es recomendable crear un entorno virtual para la instalación de las librerías. Para instalar el entorno virtual de Python se hace uso de la herramienta virtualenv la cual se obtiene instalando con el comando *pip install virtualenv.* El entorno virtual se crea con el comando *python3 -m venv name* (donde *name* es el nombre del entorno virtual) y para activarlo se lo hace con comando *source activate,* como se indica en la Figura 66.

Figura 66

Creación del entorno virtual en la SBC Odroid



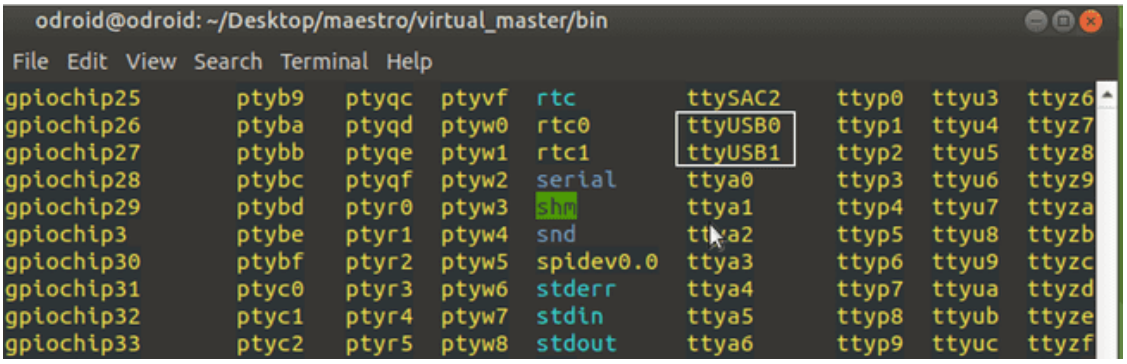
Dentro de este entorno virtual se instalan todas las librerías necesarias para la implementación del sistema de identificación y el sistema integral. En esta sección solo nos centraremos en el sistema de identificación animal, y para este se usan las librerías *pyserial* y *struct*, la primera se usa para la comunicación de los lectores con el Maestro, y la segunda se usa para la conversión de tipos de datos en Python, ya que pyserial trabaja con datos tipo byte,

por lo que estos datos deben ser transformados a entro para realizar operaciones dentro del sistema.

Además de la instalación de las librerías es indispensable realizar configuraciones de hardware de los módulos que se utilizarán en la SBC. La comunicación con los lectores se realizará a través de puertos USB del controlador. Dado que la comunicación se realizará mediante interfaz RS-485, es necesario el uso de un conversor USB/RS-485. Considerando que la misma SBC ODROID se conectará a otra red RS-485 para la comunicación con la red MODBUS. De manera general se utilizarán dos conversores USB/RS-485 conectados al controlador. En la Figura 67 se observan dos dispositivos RS-485, con los nombres asignados de forma automática por el sistema operativo de acuerdo al orden en que son conectados a los puertos USB, el dispositivo ttyUSB0 es el que primero se conectó al bus y el ttyUSB1 el segundo, por lo tanto, no es posible identificar el dispositivo conversor que está conectado a un puerto USB, si es el conversor perteneciente al MODBUS o si es el conversor que conecta los lectores.

Figura 67

Nombres de los dispositivos conversores RS-485

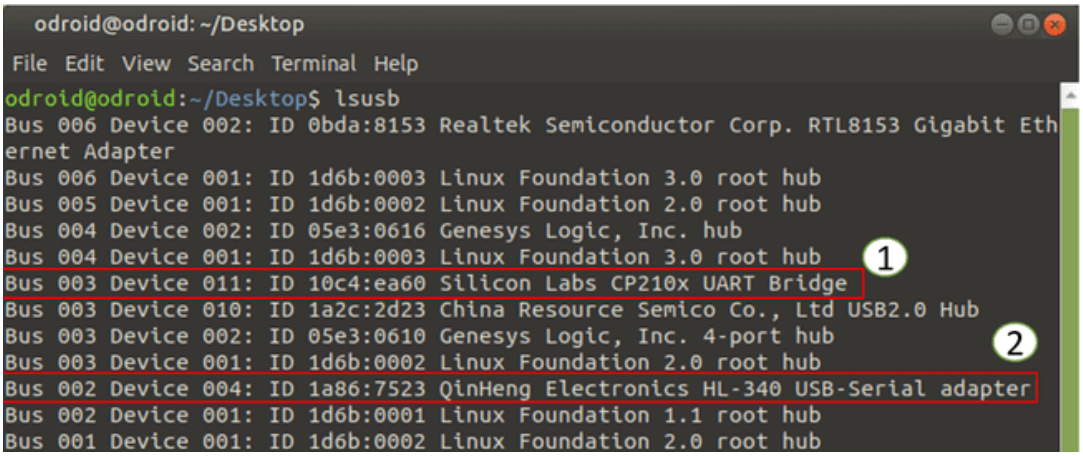


Para solucionar el problema antes mencionado es necesario crear una regla de Ubuntu que permita identificar un dispositivo conociendo información del fabricante y número de serie, esta regla se ejecuta cada vez que arranca el sistema operativo. Para obtener información sobre los dispositivos conectados se ejecuta el comando *lsusb,* ver Figura 68, la información “1”

corresponde al conversor que se conecta al bus de los lectores y la información “2” al conversor que se conecta al MODBUS. La información que nos interesa es “10c4:ea60”, el primer valor corresponde id del fabricante, mientras que el segundo corresponde al id del producto.

Figura 68

Información de los dispositivos conectados a los puertos USB de la Odroid



Con esta información creamos una regla de la siguiente manera; abrir el terminal en el directorio */etc/udev/rules.d,* luego ejecutar el comando sudo nano *99-usb-serial.rules* y se abre la GNU nano donde se escribirá la regla como se muestra en la Figura 69.

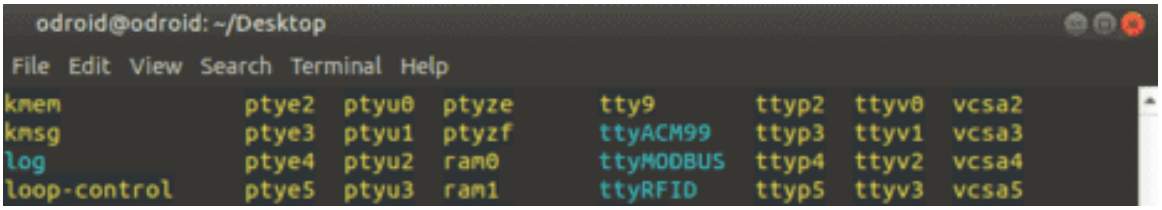
Figura 69

Reglas para asignación de dispositivos conversores RS-485

De esta forma se tiene identificado los conversores con nombres distintos, para verificar buscamos nuevamente los dispositivos activos en los puertos USB´s con el comando *ls /dev* y vemos en la Figura 70 los dispositivos ttyMODBUS y ttyRFID.

Figura 70

Dispositivos RS-485 identificados en los puertos USB de la Odroid a partir de la regla creada



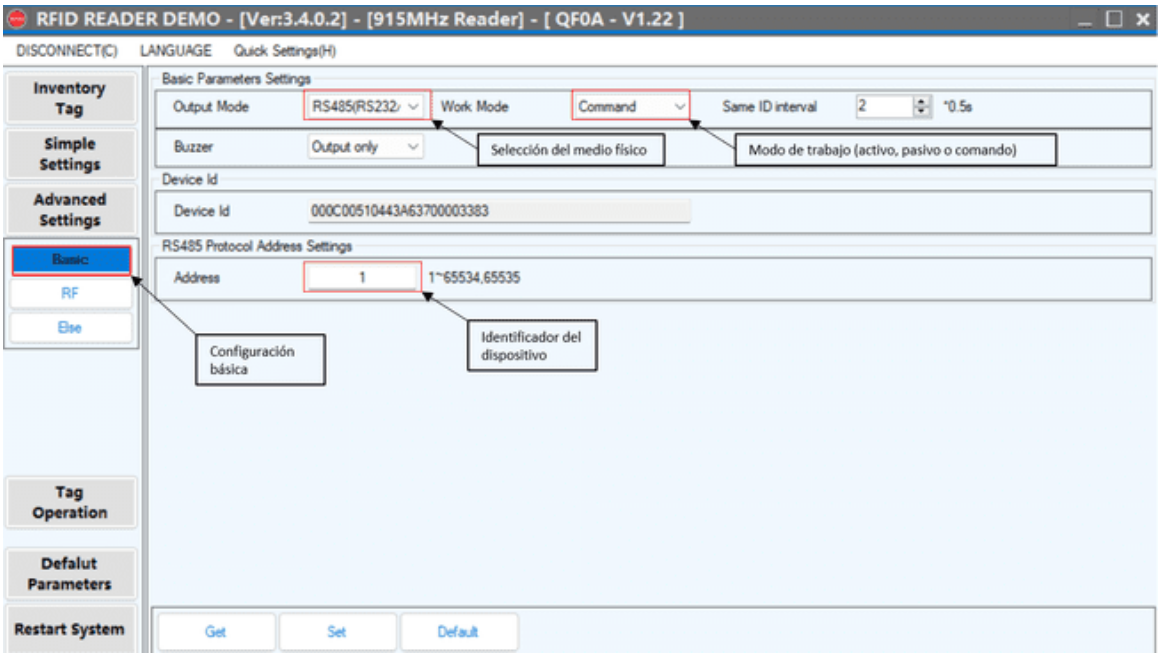
Nota. El dispositivo ttyMODBUS corresponde al conversor RS-485 conectado al MODBUS y el ttyRFID corresponde al conversor RS-485 conectado a los lectores.

Configuración de los lectores RFID mediante el software RFIDDemo

Para definir los parámetros de comunicación de los lectores se emplea el software propietario de YANZEO RFID Reader Demo que se muestra en la Figura 71. El medio físico se selecciona en *Output Mode* que para nuestro caso será el RS-485, el modo de trabajo, como se había definido anteriormente será el modo comando. El valor address corresponde al identificador del lector, que será “0” para el lector 1 y “1” para el lector 2.

Figura 71

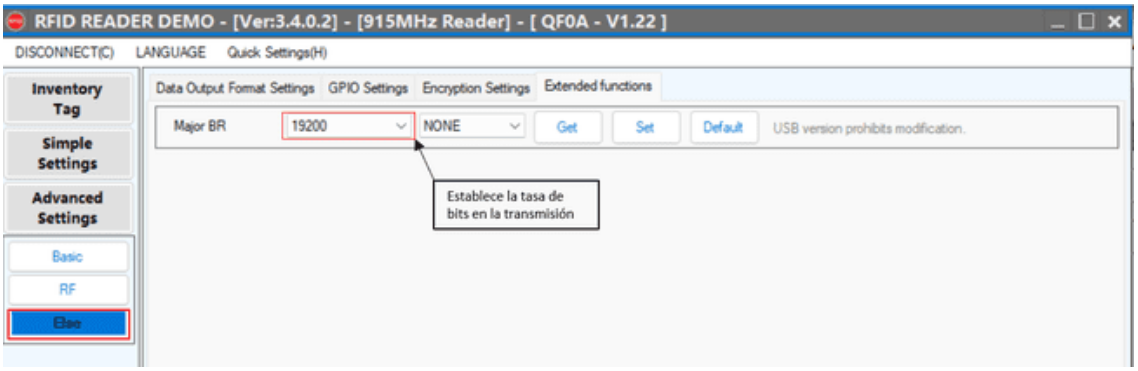
Software de configuración para los lectores RFID



Para establecer la tasa de bits de transmisión se configura en *Advanced Settings* en la opción *else.* Para la comunicación del sistema de identificación se trabajará con una tasa de bits de 19.200 como se muestra en la Figura 72.

Figura 72

Configuración de la velocidad de transmisión entre los lectores y el dispositivo maestro



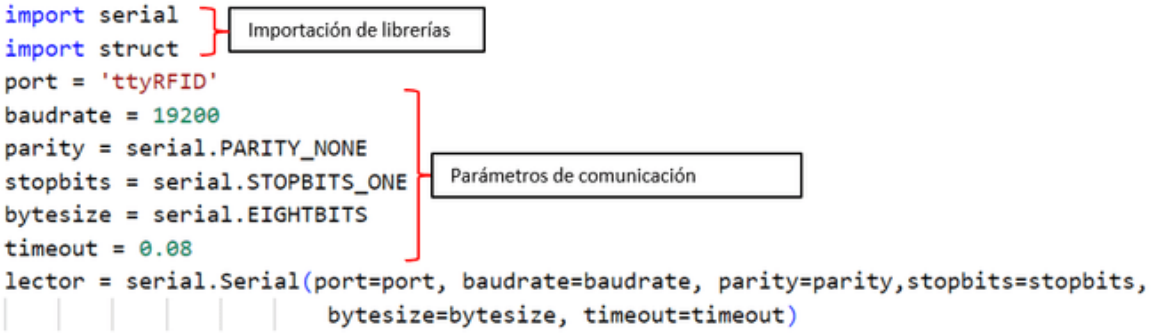
Implementación del código en Python

Una vez importadas las librerías que serán útiles en el transcurso de la programación para este sistema y realizado la configuración de los puertos y lectores, se procede a programar las líneas de código que contribuyen con la implementación de la primera etapa del sistema integral.

Para la implementación del código se tomará como referencia el diagrama de flujo de la Figura 29, por lo tanto, primero se inicia la comunicación serial, como se muestra en la Figura 73. Los parámetros de comunicación deben ser los mismos que los configurados en el lector, caso contrario no se realizará la conexión.

Figura 73

Código para establecer la comunicación




```
import serial
import struct
port = 'ttyRFID'
baudrate = 19200
parity = serial.PARITY_NONE
stopbits = serial.STOPBITS_ONE
bytesize = serial.EIGHTBITS
timeout = 0.08
lector = serial.Serial(port=port, baudrate=baudrate, parity=parity,stopbits=stopbits,
                       bytesize=bytesize, timeout=timeout)
```

Nota. El objeto lector permite realizar la conexión mediante la clase Serial.

Una vez definido los parámetros de la comunicación se procede a realizar la lectura de los lectores siguiendo la lógica de programación especificado en el diagrama de flujo del diseño. Primero se envía a través del bus el comando de lectura de etiquetas, como se indica en la Figura 74, este comando ya se calculó en el capítulo anterior y corresponde a la Tabla 12 para el lector 1 y la Tabla 13 para el lector 2.

Figura 74

Código para leer etiquetas RFID

```
while True:
    #LECTOR 1
    lector.write(bytes.fromhex('7c 01 00 21 00 07 00 00 00 00 01 02 02 56'))
    data0=lector.readline()
    longitud= data0.__len__()
    if longitud==26:
        print("RESPUESTA RFID1: ",data0.hex())
    #LECTOR 2
    lector.write(bytes.fromhex('7c 02 00 21 00 07 00 00 00 00 01 02 02 55'))
    data1=lector.readline()
    longitud= data1.__len__()
    if longitud==26:
        print("RESPUESTA RFID2: ",data1.hex())
```

Nota. El proceso está dentro de un bucle infinito para realizar pruebas de varias etiquetas.

De esta forma se tiene el valor de la etiqueta con un tamaño de trece bytes, como se muestra en la Figura 75. El id del ganado está ubicado en los bytes 13 y 14 expresados en hexadecimal, por lo tanto, se debe capturar estos bytes y convertirlos a enteros haciendo uso de la librería *struct* con la función *unpack().*

Figura 75

Respuesta de la lectora a la solicitud del maestro

Como ejemplo para la conversión de byte a hexadecimal se tomará la última respuesta del lector que se muestra en la Figura 75, donde el id del ganado en hexadecimal es *40 E5 H.* Primero se transforma los bytes por separado a entero, dando como resultado 64 229 y luego se emplea la siguiente fórmula.

$$numEntero = primerByteInt * 255 + segundoByteInt + primerByteInt$$

$$numEntero = 64 * 255 + 229 + 64$$

$$numEntero = 16613$$

Por lo tanto, el número *40 E5 H* corresponde en entero al valor 16613. El código en Python para mostrar solo el valor del id del ganado se muestra en la Figura 76.

Figura 76

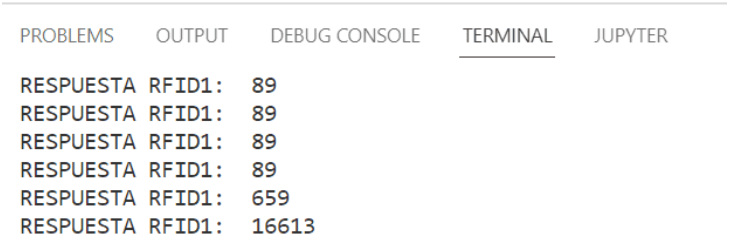
Obtención del id del ganado mediante lectores RFID

```
while True:
    #LECTOR 1
    lector.write(bytes.fromhex('7c 01 00 21 00 07 00 00 00 00 01 02 02 56'))
    data0=lector.readline()
    longitud= data0.__len__()
    if longitud==26:
        entero0=struct.unpack('<BBBBBBBBBBBBBBBBBBBBBBBBBB', data0)
        tag = entero0[13]*255+data0[14]+entero0[13]
        print("RESPUESTA RFID1: ",tag)
    #LECTOR 2
    lector.write(bytes.fromhex('7c 02 00 21 00 07 00 00 00 00 01 02 02 55'))
    data1=lector.readline()
    longitud= data1.__len__()
    if longitud==26:
        entero1=struct.unpack('<BBBBBBBBBBBBBBBBBBBBBBBBBB', data1)
        tag = entero1[13]*255+entero1[14]+entero1[13]
        print("RESPUESTA RFID2: ",tag)
```

La respuesta del programa de la Figura 76 para tres etiquetas RFID se muestra en la Figura 77.

Figura 77

Respuesta del programa de obtención del id del ganado mediante lectores RFID

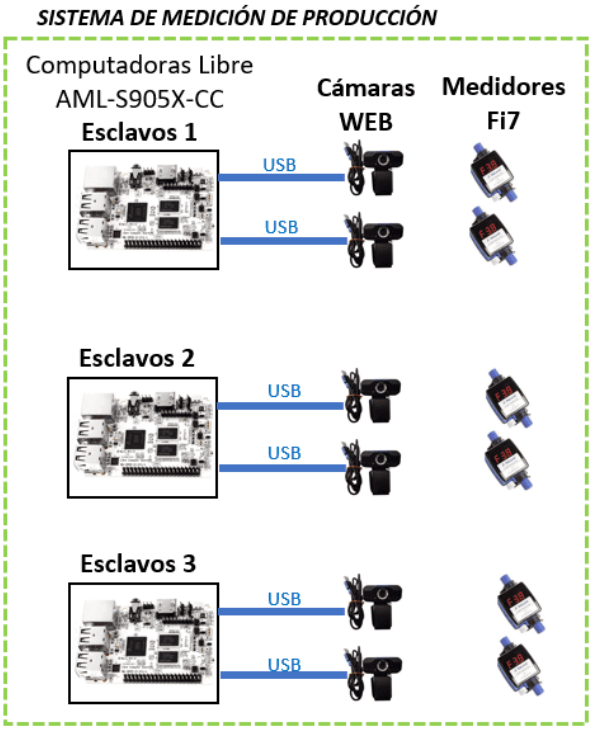


Implementación del sistema de medición de la producción de leche

Para el sistema de medición de la producción de leche, se tiene el esquema de la Figura 78. La SBC AML-S905X-CC realiza todo el procesamiento de imágenes para determinar el valor presente en los displays de los medidores de flujo de leche, estas imágenes son capturadas por las cámaras WEB que están conectadas a los puertos USB de la SBC como lo muestra la Figura 78.

Figura 78

Esquema de conexión del sistema de medición de la producción de leche



Al igual que el diseño anterior, para realizar la implementación de código del sistema de medición de la producción se debe realizar algunas configuraciones iniciales.

Configuración de la SBC AML-S905X-CC utilizada en el sistema de medición de la producción de leche

En la SBC SBC AML-S905X-CC se instaló el sistema operativo Raspbian que es una distribución del sistema operativo GNU/Linux basado en Debian, puesto que obtuvo mejor rendimiento en comparación con Ubuntu y Armbian que también son compatibles con esta SBC. Al igual que en la placa Odroid se creó un entorno virtual para la instalación de las librerías.

Dentro de este entorno virtual se instalan todas las librerías necesarias para la implementación del sistema medición de la producción de leche y las librerías que permiten la comunicación con el maestro en el sistema integral. En esta sección se considera solamente el

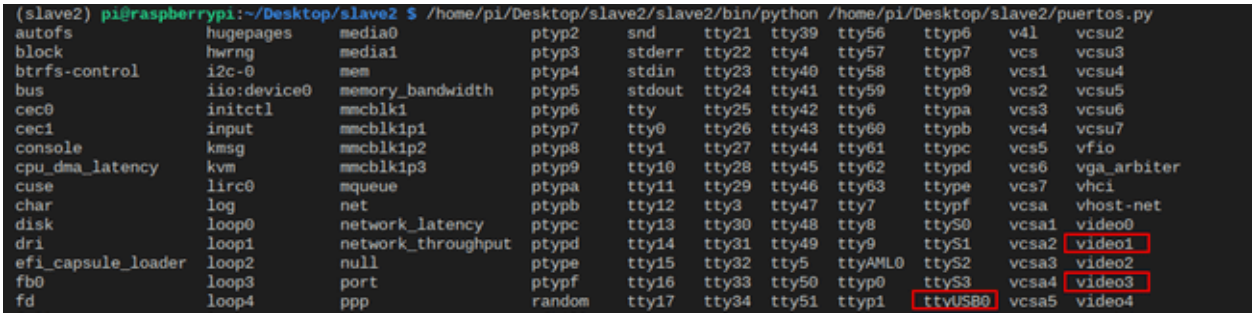
sistema de medición de la producción, en las cuales son necesarias las librerías *opencv* y *numpy.* Para instalar estas librerías se activa el entrono virtual y se emplean los siguientes comandos:

- *pip3 install numpy*
- *pip3 install opencv-python*

Además de la instalación de las librerías es necesario realizar configuraciones de hardware a la SBC. Cada esclavo tendrá conectados en sus puertos USB dos cámaras, que hace referencia a dos medidores de leche, y un conversor RS-485, por lo tanto, es necesario crear reglas para asignar a qué medidor corresponde cada cámara. Si revisamos los dispositivos conectados a la placa SBC mediante un comando *ls /dev* muestra el conversor RS-485 denominado *ttyUSB0*, *video 1* que corresponde a la cámara que se conectó primero al puerto y *video 3* que corresponde a la cámara que se conectó segundo al puerto USB, ver Figura 79.

Figura 79

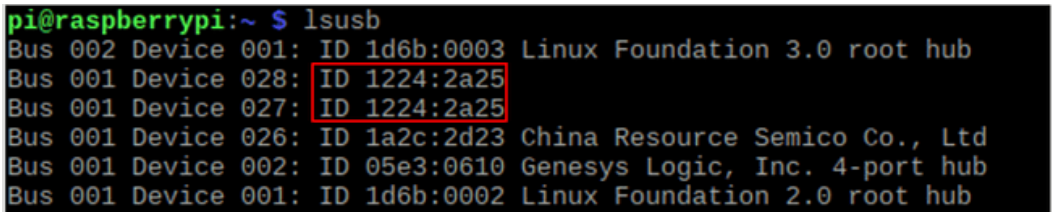
Dispositivos conectados a los puertos USB de la SBC AML-S905X-CC



A diferencia de los conversores, las cámaras tienen el mismo ID tanto del fabricante como del producto, como se muestra en la **Figura 80**, por tal motivo no se puede asignar el nombre del dispositivo solamente con esta información.

Figura 80

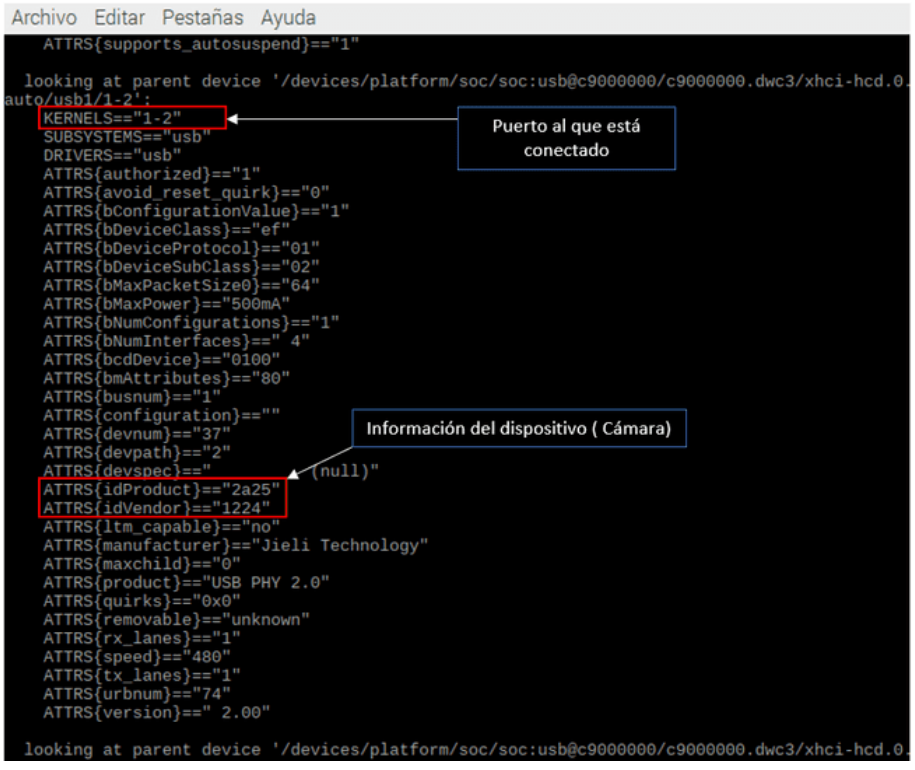
Información básica del dispositivo conectado al puerto USB.



Para solucionar este problema se debe obtener información no solo de las cámaras sino del puerto al que está conectado, esto se realiza con el comando *udevadm info -a /dev/video1* que da como respuesta la información de la cámara y del puerto al que está conectado, esto se pude ver en la Figura 81.

Figura 81

Información completa del dispositivo conectado al puerto USB.



Con esta información se crea una regla para que se ejecute cada vez que arranque el sistema operativo, es importante tomar en cuenta que el medidor 1 siempre debe estar

conectado al puerto 1-1.1, de acuerdo a la regla que se muestra en la Figura 82, mientras que el medidor 2 debe conectarse en el puerto 1-2.

Figura 82

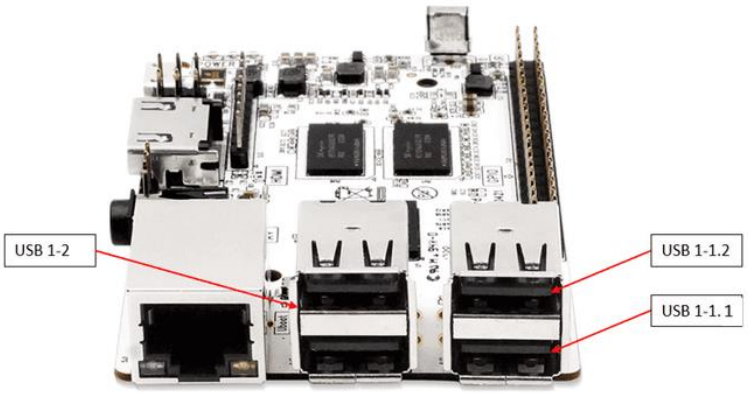
Regla para la asignación de dispositivos en los puertos USB´s.

```
Archivo  Editar  Pestañas  Ayuda
 GNU nano 2.7.4                  Fichero: 99-com.rules                    Modificado

SUBSYSTEM=="tty", ATTRS{idVendor}=="1a86", ATTRS{idProduct}=="7523", SYMLINK+="ttyMODBUS"
SUBSYSTEM=="video4linux", KERNELS=="1-1.1", ATTRS{idVendor}=="1224", ATTRS{idProduct}=="2a25", SYMLINK+="medidor1"
SUBSYSTEM=="video4linux", KERNELS=="1-2", ATTRS{idVendor}=="1224", ATTRS{idProduct}=="2a25", SYMLINK+="medidor0"
```

En la Figura 83 se muestra la enumeración de puertos USB´s de la SBC, dependiendo de ello se debe asignar las reglas al sitema operativo Raspbian.

Figura 83

Nombre de los puertos de la SBC AML-S905X-CC.



Implementación del código en Python

Una vez realizado todas las configuraciones se procede con la implementación del programa en Python tomando como referencia el diagrama de flujos de la Figura ***33***. El primer bloque que se implementa es el de capturar imagen, para ello primero se definen las cámaras como se muestra en la Figura 84. Los valores *anchocam,* y *altocam* definen las dimensiones de la imagen a capturar, depende de la resolución de la cámara y de la distancia del medidor a la cámara. Se crean dos objetos cap0 y cap1 de la clase *VideoCapture(),* la cual pertenece a la librería *Opencv* que fue importada como *cv2,* que reciben como parámetros el nombre de las

cámaras asignadas anteriormente. El objeto cap0 hace referencia al medidor 1 y el cap1 al medidor 2. La función *set()* es la que asigna las dimensiones de la imagen a capturar.

Figura 84

Función definirCamaras para capturar las imágenes de los medidores en un tamaño especifico

```
def definirCamaras():
    anchocam, altocam = 657, 360
    #-------DEFINICION DE CAMARAS-----------
    cap0 = cv2.VideoCapture("/dev/medidor1")
    cap0.set(3,anchocam)
    cap0.set(4,altocam)
    cap1 = cv2.VideoCapture("/dev/medidor2")
    cap1.set(3,anchocam)
    cap1.set(4,altocam)
    return cap0, cap1
```

Para capturar un imagen de la cámara se emplea la función *read(),* que se observa en la Figura 85 de la librería *Opencv,* la cual revuelve dos valores *ret* y *frame.* Si la captura de la imagen se realizó de forma correcta el valor de *ret* vale “True” y en *frame* el objeto imagen que representa la captura, esta imagen estará en formato BGR (Azul-Verde-Rojo). Para definir una área específica de la imagen donde se trabajará se dibuja un rectángulo dentro de la imagen capturada, esto se realiza empleando la función *rectangle()* de *Opencv* el área de recorte está dado por el valor de la variable *cuadro,* que en este caso se seleccionó un valor de “50”. Finalmente se guarda la imagen recortada en las variables *img0* e *img1*.

Figura 85

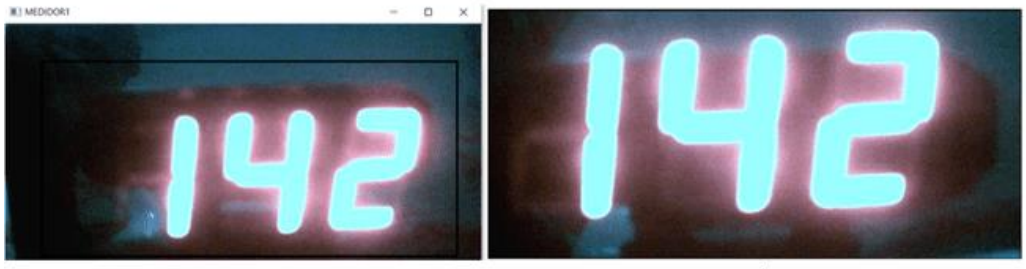
Código para capturar una imagen y realizar un recorte

```
cuadro = 50
ret0, frame0 = cap0.read()
assert ret0
ret1, frame1 = cap1.read()
assert ret1

cv2.rectangle(frame1,(cuadro, cuadro),(anchocam-cuadro, altocam-cuadro),(0, 0, 0), 2)
cv2.rectangle(frame0,(cuadro, cuadro),(anchocam-cuadro, altocam-cuadro),(0, 0, 0), 2)

x1, y1 = cuadro, cuadro
ancho, alto = (anchocam-cuadro)-x1, (altocam-cuadro)-y1
x2, y2 = x1 + ancho, y1 + alto
img0 = frame0[y1:y2, x1:x2]
img1 = frame1[y1:y2, x1:x2]

cv2.imshow("MEDIDOR1",frame0)
cv2.imwrite("test.jpg",img1)
cv2.imwrite("test.jpg",img0)
cv2.imshow("MEDIDOR2",frame1)
```

Para facilidad de explicación se mostrará resultados de una sola cámara. Como resultado del código de la Figura 85 se tiene dos imágenes, que se muestra en la Figura 86, la primera resulta de la captura de la cámara empleando la función *read(),* mientras que la segunda corresponde al recorte de la región de interés obtenida al dibujar el rectángulo mediante la función *rectangle().*

Figura 86

Captura de imagen de la pantalla del medidor mediante la cámara WEB.



Nota. a) Captura de pantalla de toda la cámara. b) Recorte del área de interés.

Con base en el diagrama de flujo del diseño, como siguiente paso a la imagen recortada se cambia a escala de grises empleando la función de Python *cv2.cvtColor(img0, cv2.COLOR_RGB2GRAY),* pero antes de esto se realiza un giro antihorario a la imagen debido a que los display's presentan cierta inclinación que afectaría al modelo que se creará más adelante, este giro se realiza con la función *getRotationMatrix2D()* y la función *warpAffine()* que permite rellenar los vacíos que se generan al rotar la imagen. Una vez cambiada la imagen a escala de grises se aplica un filtro para eliminar el ruido conocido como ruido impulsivo, esto se hace con la función *medianBlur()* de Python, y para binarizar la imagen se emplea la función *threshold(),* como se muestra en el código de la Figura 87.

Figura 87

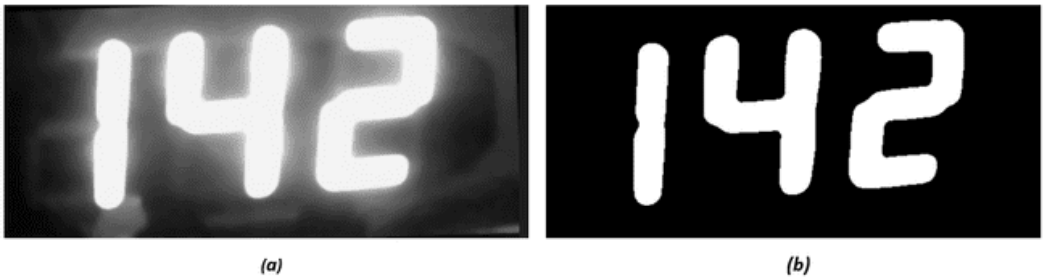
Código empleado para binarizar la imagen

```
# Rotación de la imagen alrededor del centro
h, w, _= img0.shape
centro = (w // 2, h // 2)
M = cv2.getRotationMatrix2D(centro, 2, 1.0)
img0 = cv2.warpAffine(img0, M, (w, h),flags=cv2.INTER_CUBIC, borderMode=cv2.BORDER_REPLICATE)

# Escala de grises
gray = cv2.cvtColor(img0, cv2.COLOR_RGB2GRAY);
# Filtro
gray = cv2.medianBlur(gray,9)
cv2.imwrite("Gay.jpg",gray)
# Binarización
ret,thresh1 = cv2.threshold(gray,238,255,cv2.THRESH_BINARY)
cv2.imwrite("thresh1.jpg",thresh1)
```

Como resultado de los procesos de cambiar a escalas de grises, aplicar filtro y binarizar la imagen se tienen las gráficas de la Figura 88.

Figura 88

Resultado del cambio a escala de grises y binarización de la imagen



Nota. a) Imagen en escala de grises, b) Imagen aplicada el filtro y binarizada.

Una vez binarizada la imagen se dibujan bordes alrededor de los números, esto es muy fácil debido a que se cuenta con dos colores únicamente, para encontrar los bordes se emplea la función *findContours()* tomando como parámetro con la imagen binarizada, con el parámetro *RETR_EXTERNAL* se obtiene solo los bordes externos y con *CHAIN_APPROX_SIMPLE* se considera solamente los puntos de los extremos de cada borde. Para dibujar los bordes se hace uso de la función *drawContours()* para posteriormente recortar cada número de forma individual. La implementación de este código se muestra en la Figura 89.

Figura 89

Código para el trazado de contornos de los números en la imagen

```
# Buscar bordes
contours, jerarq = cv2.findContours(thresh1, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE);
# Dibujar bordes
for contour in contours:
    cv2.drawContours(img0, [contour], 0, (0,255,0), 3);
cv2.imshow("CONTORNOS", img);
cv2.imwrite("CONTORNOS.jpg",img0)
# Recortar contornos
cuts = [];
number = 0;
for bound in bounds:
    tl, br = bound;
    cut_img = thresh1[tl[1]:br[1], tl[0]:br[0]];
    cuts.append(cut_img);
    number += 1;
    n=number
    cv2.imshow(str(number), cut_img);

# Fondo
font = cv2.FONT_HERSHEY_SIMPLEX;
```

Como salida del código anterior se tiene las imágenes de la Figura 90.

Figura 90

Trazado de contornos externos a los número y recortes individuales

Nota. a) Trazado de contornos externos. b) Recortes individuales de cada número.

Los pasos anteriores constituyen al procesamiento de la imagen, ahora con estos recortes se debe realizar un reconocimiento de caracteres. Como se mencionó en el diseño la forma más óptima de realizar el reconocimiento de caracteres es realizar un modelo de displays de siete segmentos. Tomando como referencia modelo de displays de siete segmentos de la

Figura 35 y los datos determinados en la Tabla ***15***, se implementa el código en Python para la creación del modelo, ver Figura 91.

Figura 91

Modelo para la detección de 7 segmentos

```
class Segments:
    def __init__(self):
        #Creación del modelo de 7 segmentos
        self.flags = [];
        self.segmentos = [];
        h1 = [[0.4, 0.74],[0.05, 0.19]];
        h2 = [[0.3, 0.68],[0.47, 0.56]];
        h3 = [[0.25, 0.52],[0.85, 0.94]];
        vl1 = [[0.18, 0.29],[0.18, 0.41]];
        vl2 = [[0.11, 0.22],[0.6, 0.81]];
        vr1 = [[0.78,0.88],[0.18, 0.41]];
        vr2 = [[0.77, 0.85], [0.6, 0.81]];
        self.segmentos.append(h1);
        self.segmentos.append(h2);
        self.segmentos.append(h3);
        self.segmentos.append(vl1);
        self.segmentos.append(vl2);
        self.segmentos.append(vr1);
        self.segmentos.append(vr2);
```

La clase Segments contiene dos métodos o procesos previamente diseñados en el capítulo anterior, ver Figura **36**, Figura 36 y Figura 37, y con base en ello se implementa el código. Para el proceso comparar modelo se tiene la función *comparar_modelo* de la Figura 92, acepta como parámetro los rectángulos recortados a partir de los contornos dibujados anteriormente. Para el número uno se hace una verificación directa de acuerdo a la relación del ancho y el alto del rectángulo. Para el resto de números se compara de acuerdo a las áreas, como se detalló en el diseño.

Figura 92

Implementación del código para el proceso comparar modelo

```
def comparar_modelo(self, number):
    self.flags = [];
    # Verifica si el número es un 1
    h, w = number.shape[:2];
    relacion  = w/h
    if relacion < 0.4:
        self.flags.append(5);
        self.flags.append(6);
        return;
    # Compara el número con el modelo
    for a in range(len(self.segments)):
        seg = self.segments[a];
        xl, xh = seg[0];
        yl, yh = seg[1];
        # convert to pix coords
        sw = xh - xl;
        sh = yh - yl;
        # Compara areas
        count = np.count_nonzero(number[yl:yh, xl:xh] == 255);
        rela = count / (sh * sw)
        if rela > 0.6: # 0.5 is a sensitivity measure
            self.flags.append(a);
```

La segunda función de la clase Segments, denominada *asignar_caracteres* corresponde al proceso de asignar caracteres que se especificó en el diagrama de flujo de la Figura 92, a continuación, en la Figura 93 se muestra el código en Python.

Figura 93

Implementación del código para el proceso asignar caracteres

```
def asignar_caracteres(self):
    if self.flags == [0,2,3,4,5,6]:
        return 0;
    if self.flags == [5,6]:
        return 1;
    if self.flags == [0,1,2,4,5]:
        return 2;
    if self.flags == [0,1,2,5,6]:
        return 3;
    if self.flags == [1,3,5,6]:
        return 4;
    if self.flags == [0,1,2,3,6]:
        return 5;
    if self.flags == [0,1,2,3,4,6]:
        return 6;
    if self.flags == [0,5,6]:
        return 7;
    if self.flags == [0,1,2,3,4,5,6]:
        return 8;
    if self.flags == [0,1,3,5,6]:
        return 9;
    if self.flags == [0,1,3,4,5]:#P
        return 1;
    if self.flags == [2,3,4,5,6]:#u
        return 11;
    # ERROR
    return -1;
```

Una vez creados el modelo y las funciones que permiten comparar las imágenes capturadas con el modelo y asignar caracteres se crea un objeto de esta clase, ver Figura 94. Como se mencionó en el diseño de este sistema, para determinar si una imagen recortada es un punto flotante se evalúa su área, si el área es inferior a 2000 entonces se le asigna una “,” al valor del carácter correspondiente a la imagen, si el área es superior se llama al proceso *comparar_modelo* de la clase *Segments,* el cual devuelve una lista con los segmentos activos, luego se asigna números dependiendo de los segmentos activos.

Figura 94

Código para la comparación de las imágenes recortadas con el modelo, mediante la clase Segments.

```
# Crear el objeto modelo de display
model = Segments();
index = 0;
numberstr= []
for cut in cuts:
    #Guarda la imagen
    cv2.imwrite(str(index) + "_" + str(number) + ".jpg", cut);
    if areas[index] <2000:
        numberstr.append('.')
    else:
        #Comparar modelo
        model.comparar_modelo(cut);
        #Asignar caracteres
        numberstr.append(str(model.asignar_caracteres()));
```

La variable *numberstr* contiene los valores de los caracteres en formato *string,* por lo tanto, es necesario cambiarlos a *float*. Antes de convertirlos a número real se establece que para los valores mostrados en el display del medidor desde 0.00 a 0.05 no se tomarán en cuenta y se asignará el valor de “1111” a su respectivo reconocimiento dado ya que permitiría saber cuándo la medida se reinició, se realiza la conversión a tipo flotante para obtener el valor que se muestra en el medidor en la Figura 95.

Figura 95

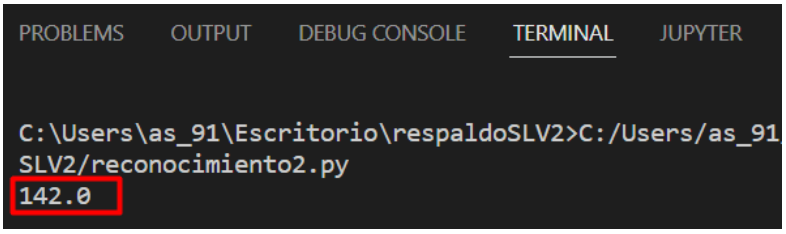
Líneas de código para el ordenamiento de los números detectados

```
medidorstr = ''
for i in range(0,dim):
    medidorstr= medidorstr + numberstr[i]

if medidorstr== '.00':
    medidorstr='1111'

if medidorstr== '000':
    medidorstr='1111'
        #*
        #*
        #*
        #*
if medidorstr== '.05':
    medidorstr='1111'

if medidorstr== '05':
    medidorstr='1111'

medidorstr = float(medidorstr)
return medidorstr
```

Con esto se tiene como salida del proceso de reconocimiento de caracteres de la pantalla del display lo que se muestra en la Figura 96.

Figura 96

Salida del proceso de reconocimiento de caracteres perteneciente al sistema de medición de la producción de leche



Implementación del software de gestión ganadera

A continuación se detallan las herramientas y librerías necesarias para el desarrollo de la aplicación de gestión ganadera. También se describirán la conexión e interacción con la base de datos. En cuanto al código al ser bastante extenso se lo colocará en el apéndice E.

Instalación de herramientas y librerías necesarias para la implementación

Como se mencionó en el diseño, para el desarrollo de las funcionalidades de la aplicación de escritorio se empleará Python como lenguaje de programación, Visual Studio Code como editor de texto, MySQL como gestor de base de datos y Workbench como herramienta de diseño de la base de datos. Por lo tanto, antes que nada, se deben instalar estas herramientas desde las páginas oficiales, a continuación se adjunta los links de descarga:

- Python: https://www.python.org/downloads/
- MySQL: https://www.mysql.com/downloads/
- Visual Studio Code: https://code.visualstudio.com/download

Una vez instaladas estas herramientas se crea un directorio de instalación aislado, este aislamiento permitirá localizar las instalaciones de las independencias del proyecto sin necesidad de instalar en todo el sistema, esto también evita que exista errores debido a actualizaciones de librerías o del mismo Python. Para crear un entorno virtual en Windows primero se instala la herramienta virtualenv desde la terminal mediante el comando *pip install virtualenv,* luego se crea el entorno virtual “entornov” desde el directorio del proyecto empleando el comando *python -m venv entornov,* finalmente se activa el entorno virtual con *source Scripts/activate.*

Dentro del entorno virtual creado se instalan las librerías necesarias para el desarrollo de la aplicación. La librería *pyside6* se instala empleando el *pip install pyside6* y la librería que permite interactuar con la base de datos, *mysql-connector* se instala con el comando *pip install mysql-connector-python.*

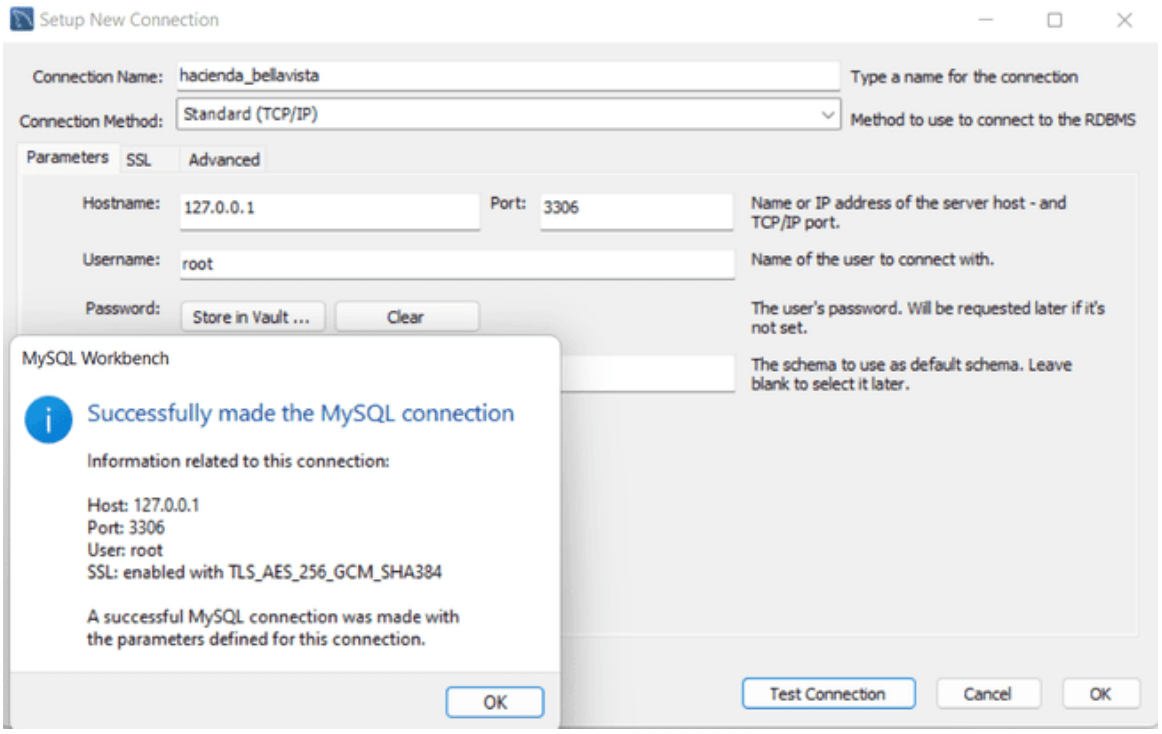
Creación de la base de datos

Para crear la base de datos se emplea la herramienta de diseño de base de datos Workbench. La base de datos tendrá como nombre “hacienda_bellavista”, el puerto de conexión por defecto es 3306, el nombre del host es “localhost”, por seguridad se debe

establecer una contraseña. En la Figura 97 se muestra la creación de la base de datos, además se realiza un test de conexión.

Figura 97

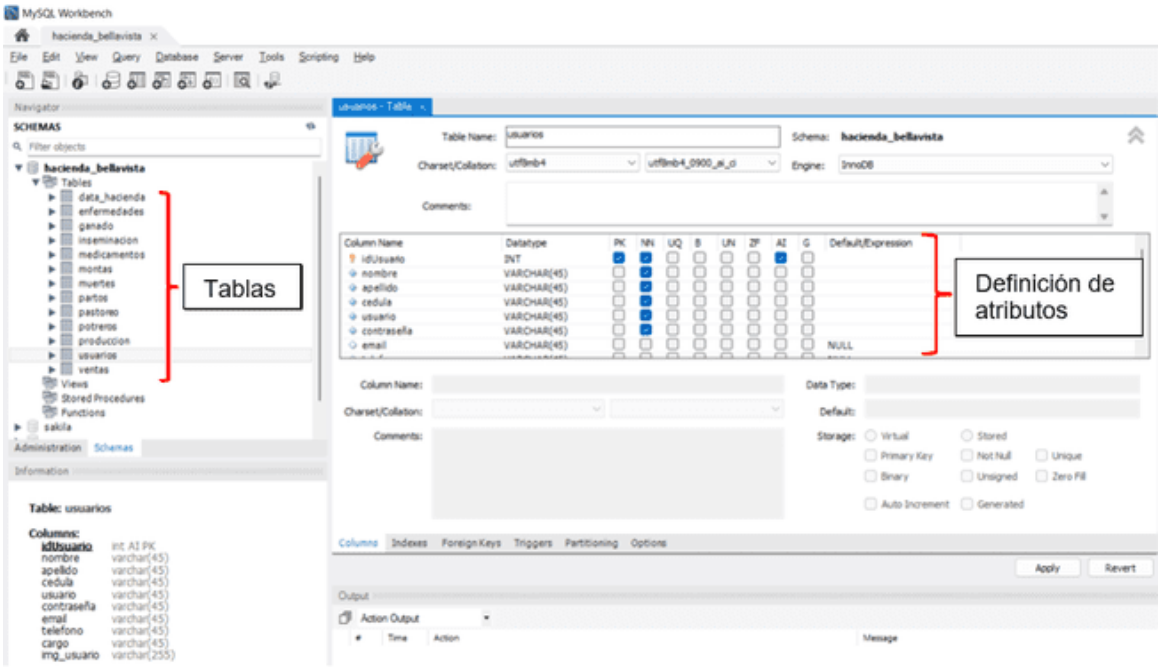
Creación de la base de datos



Una vez creada la base de datos se crean las 13 tablas que se definieron en el capítulo anterior, en la Figura 98 se muestra la base de datos diseñada en Workbench, en la parte izquierda de la gráfica se visualizan las trece tablas para cada entidad y en la parte derecha los atributos de la entidad, aquí se define el tipo de datos que tendrán los atributos, las claves principales, si es o no auto incrementable y el tipo de atributo, es decir si es opcional u obligatorio.

Figura 98

Base de datos de la hacienda bellavista.

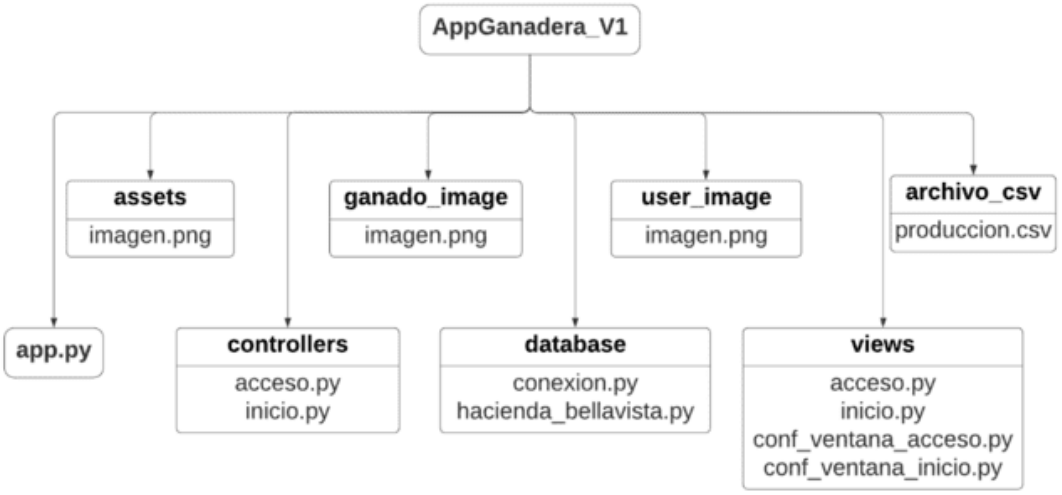


Ficheros de código

El código sobre el cual se ejecuta la aplicación de escritorio es bastante extenso, y es por tal motivo que se organiza siguiendo una lógica en directorios. Cada directorio contiene una serie de archivos y estos a su vez contienen distintos módulos que realizan las distintas funciones del programa. Además de los archivos creados se requieren otros módulos originarios del lenguaje Python, y para ser usados deben ser importados. Se puede importar todo un archivo “.py” o solamente un módulo del archivo. Los ficheros de Python creados siguen la lógica de la Figura 99.

Figura 99

Ficheros de código de la aplicación de escritorio



Directorio AppGanadera_V1. Es la carpeta principal dentro de las cuales se encuentran todos los directorios y el archivo principal.

Directorio assets. En este directorio se almacenan los íconos para las ventas de acceso e inicio.

Directorio ganado_image. En este directorio se almacenan las imágenes del ganado registrado desde la aplicación de escritorio que puede ser de tipo png o jpg.

Directorio user_image. En este directorio se almacena las imágenes de los usuarios registrados.

Directorio archivo_csv. En este directorio se almacena temporalmente el archivo plano que se selecciona para subir los datos de ordeño a la base de datos.

Directorio controllers. Dentro de este directorio se encuentra dos archivos ".py", acceso.py donde se encuentra todo el código para el control de acceso y el archivo inicio.py contiene todos los módulos que realizan las funcionalidades del software.

Directorio database. Este directorio también contiene dos archivos Python, conexion.py que realiza la conexión entre Python y la base de datos MySQL previamente

creada y el archivo hacienda_bellavista.py que contiene los módulos que le permiten la interacción con la base de datos.

Directorio views. Este directorio cuenta con cuatro archivos Python; *inicio.py* y *acceso.py* contiene el código de la interfaz de usuario generado automáticamente utilizando la librería *pyside6*, el archivo *conf_ventana_acceso.py* contine los módulos de configuración visual de la ventana de acceso, *conf_ventana_inicio.py* contine los módulos de configuración visual de la ventana de inicio y el archivo *componentes* donde se implementa el código para crear los botones de editar y eliminar filas de las tablas de registros.

app.py. Este es el archivo principal del proyecto, por lo tanto, el más importante. Hace la función del archivo raíz dentro del código, desde aquí se accede a los módulos y archivos Python de otros directorios.

Conexión con la base de datos desde Python

Una vez creada la base de datos, se procede a realizar la conexión entre la base de datos y Python, para ello se hace uso de la librería *connector* que se encuentra dentro del paquete de *mysql* ya instalado en el entorno virtual. El código que permite realizar dicha conexión se muestra en la Figura 100.

Figura 100

Código para la creación de la base de datos

```
from mysql import connector
config = {
    'user' : 'root',
    'password' : 'bellavista123',
    'host' : 'localhost',
    'database' : 'hacienda_bellavista'
}
def crear_conexion():
    conn = None
    try:
        conn = connector.connect(**config)
    except connector.Error as err:
        print(f"Error en la funcion crear_conexion: {err.msg}")
    return conn
```

El constructor *connect()* permite crear una conexión con el servidor MySQL y devuelve un objeto *MySQLConnection.* Para manejar los errores de conexión con la base de datos se emplea la declaración *try* y utilizando la excepción *errors.Error* se captura los posibles errores.

Interacción con la base de datos

Otro aspecto importante para el desarrollo de la aplicación es la interacción de Python con la base de datos, es decir, como realizar una consulta, obtener datos de una tabla con base en una búsqueda, insertar datos en las tablas, actualizar o eliminar datos de las tablas. A continuación, se detalla cada una de estas funciones y cómo se implementa en Python.

Insertar datos en una tabla. Consiste en agregar una fila de datos en una tabla, para ello primero se crea la conexión haciendo uso de la función *crear_conexion* antes mencionada, luego se emplea la instrucción *INSERT INTO,* ver en la cual se debe especificar el nombre de la tabla y las columnas donde se quiere insertar los datos, con *VALUES* especificamos el número de datos que se va a insertar. Para cargar los datos se define un objeto *cur* de tipo cursor que permite agregar los datos, siempre que realiza algún cambio en la base de datos se hace un *comit()* para que se realicen los cambios. Para manejar los errores de conexión con la base de datos se emplea la declaración *try* y utilizando la excepción *errors.Error* se identifica los posibles errores. Es importante mencionar que los resultados de una consulta se reciben en una tupla, cuyos elementos son otras tuplas, ver Figura 101.

Figura 101

Código para insertar datos en una tabla de la base de datos

```
def insertar_usuario(data):
    conn = crear_conexion()
    sql = """INSERT INTO usuarios (nombre, apellido, cedula, usuario, contraseña,
                                   email, telefono, cargo, img_usuario)
             VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s)
             """
    try:
        cur = conn.cursor()
        cur.execute(sql,data)
        conn.commit()
        return True
    except connector.Error as err:
        print(f"Error en la funcion insertar_usuario: {err.msg}")
        return False

    finally:
        cur.close()
        conn.close()
```

Seleccionar todos los datos de una tabla. Consiste en obtener todas las filas de una tabla, de igual manera se crea la conexión, luego se emplea la instrucción *SELECT-FROM* donde se define las columnas y el nombre de la tabla, tal como se muestra en la Figura 102.

Figura 102

Código para seleccionar todos los datos de una tabla

```
def seleccionar_todo_usuarios():
    conn = crear_conexion()
    sql = f"""SELECT idUsuario, nombre, apellido, cedula, usuario, contraseña,
                     email, telefono, cargo FROM usuarios"""
    try:
        cur = conn.cursor()
        cur.execute(sql)
        dataPerson = cur.fetchall()
        return dataPerson
    except connector.Error as err:
        print(f"Error en la función seleccionar_todo_usuarios: {err.msg}")
        return False
    finally:
        cur.close()
        conn.close()
```

Seleccionar solo datos por coincidencia. Consiste en seleccionar los registros o filas que cumplan con un requisito o parámetro de búsqueda. Por ejemplo, en el caso de que se quiera obtener determinadas columnas de una tabla solo de las filas q tenga un tag en específico que se encuentre entre un rango de fechas. En la Figura 103 se muestra la implementación de este ejemplo, las columnas se a extraer se seleccionan con la instrucción *SELECT*, la tabla con la instrucción *FROM* y las condiciones o fechas con la instrucción *WHERE*.

Figura 103

Código para seleccionar datos por parámetros

```
def selec_por_tag_fecha_produccion(tag, param1, param2):
    conn = crear_conexion()
    sql = """SELECT idProduccion, tag,  fecha_ordeno, turno, litros, observaciones
            FROM produccion
            WHERE tag LIKE %s AND (%s <= fecha_ordeno AND fecha_ordeno <= %s)
            """
    try:
        cur = conn.cursor()
        cur.execute(sql, (tag, param1, param2))
        dataPerson = cur.fetchall()
        return dataPerson
    except connector.Error as err:
        print(f"Error at select_by_parameter_consulta_fechas_produccion: {err.msg}")
        return False

    finally:
        cur.close()
        conn.close()
```

Actualizar tabla. Esta funcionalidad es usada cuando se desea editar los datos de una determinada fila, se emplea la instrucción *UPDATE* seguido del nombre de la tabla, luego con la instrucción *SET* se asigna los valores de la tupla *data* que contiene los datos nuevos a las columnas de la tabla y finalmente con la instrucción *WHERE* definimos el tipo de búsqueda de la fila a editar. La implementación se aprecia en la Figura 104.

Figura 104

Código para actualizar datos de una tabla

```
def actualizar_produccion(_id, data):
    conn = crear_conexion()
    sql = f"""UPDATE produccion SET
                                    tag = %s,
                                    fecha_ordeno = %s,
                                    turno = %s,
                                    litros = %s,
                                    observaciones = %s
            WHERE idProduccion = {_id}
            """
    try:
        cur = conn.cursor()
        cur.execute(sql,data)
        conn.commit()
        return True
    except connector.Error as err:
        print(f"Error at actualizar_pastoreo function: {err.msg}")
        return False
    finally:
        cur.close()
        conn.close()
```

Eliminar datos de una tabla. Esta funcionalidad como su nombre lo indica, permite eliminar una fila de una tabla. Se emplea la instrucción *DELETE FROM* seguido del nombre de la tabla, con WHERE se asigna el parámetro que permite seleccionar la fila de la tabla que se va a eliminar. La implementación se muestra en la Figura 105.

Figura 105

Código para eliminar datos de una tabla

```
def eliminar_produccion(_id):
    conn = crear_conexion()
    sql = f"""DELETE FROM produccion
            WHERE idProduccion = {_id}
            """
    try:
        cur = conn.cursor()
        cur.execute(sql)
        conn.commit()
        return True
    except connector.Error as err:
        print(f"Error en la funcion eliminar_producción : {err.msg}")
        return False

    finally:
        cur.close()
        conn.close()
```

Estos módulos en conjunto conforman la parte importante del software, pues mediante ellas se puede acceder y manipular todos los datos. La programación de las funcionalidades del software se logra empleando metodologías de programación con diagramas de flujo que no tiene caso describirlas, pero se presenta en el Apéndice E. Una función importante es la conexión con el sistema de medición de la producción y esto se comentará en la siguiente sección.

Una vez realizado la implementación se tienen los siguientes resultados en la aplicación de escritorio.

Módulo de control de acceso. Este módulo tiene acceso directo a la entidad usuarios de la base de datos, consta de un *lineEdit* para ingresar el usuario y otro para ingresar la contraseña, dependiendo del tipo de actor al accionar el botón “iniciar” se abrirá la ventana con sus funcionalidades habilitadas. El botón “cerrar” cierra la aplicación, esto se ilustra en la Figura 106.

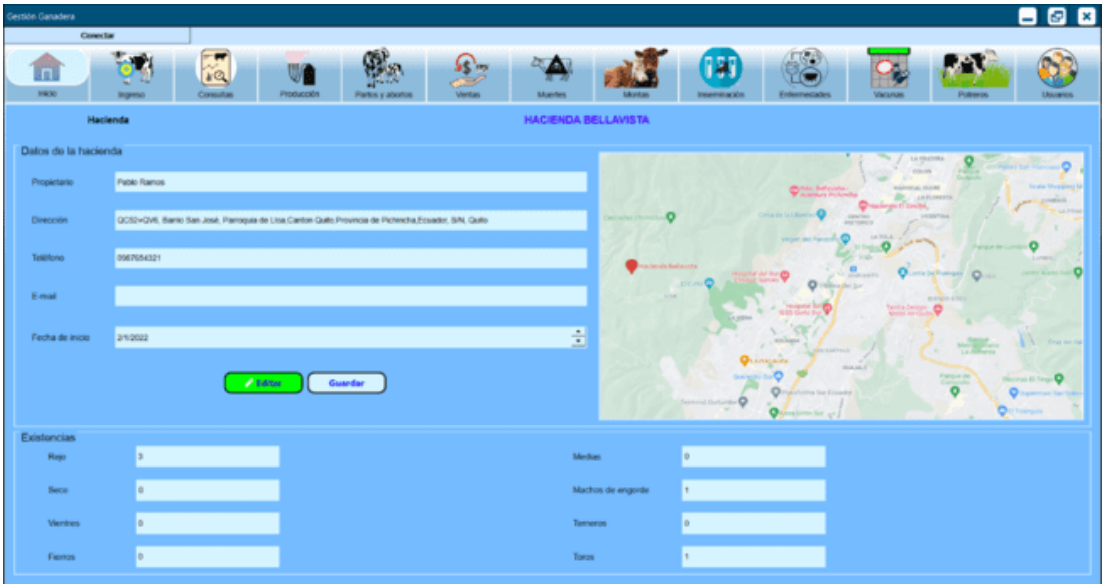
Figura 106

Ventana de control de acceso

Módulo de visualización y edición de la información de la hacienda. La ventana de inicio mustra por defecto la primera página denominada también “Inicio”, ver Figura 106 el cual contiene dos “groupBox”, en el primero se detalla la infomación básica y de contacto de la hacienda y en el segundo se detalla las existencias de ganado en la hacienda. Contiene dos botones, uno para editar la información (botón “editar”) y otro para guardar (botón “guardar”), ver Figura 107.

Figura 107

Página de inicio



Módulo registrar ganado. La página registrar ganado permite al usuario ingresar los dátos del animal, dispone de tres botones; el botón “agregar foto” permite seleccionar una foto del animal desde el explorador de archivos, el botón “agregar animal” carga los datos y la foto a la base de datos y el botón “Ver tabla de registro” muestra todos los animales registrados en la hacienda, ver Figura 108.

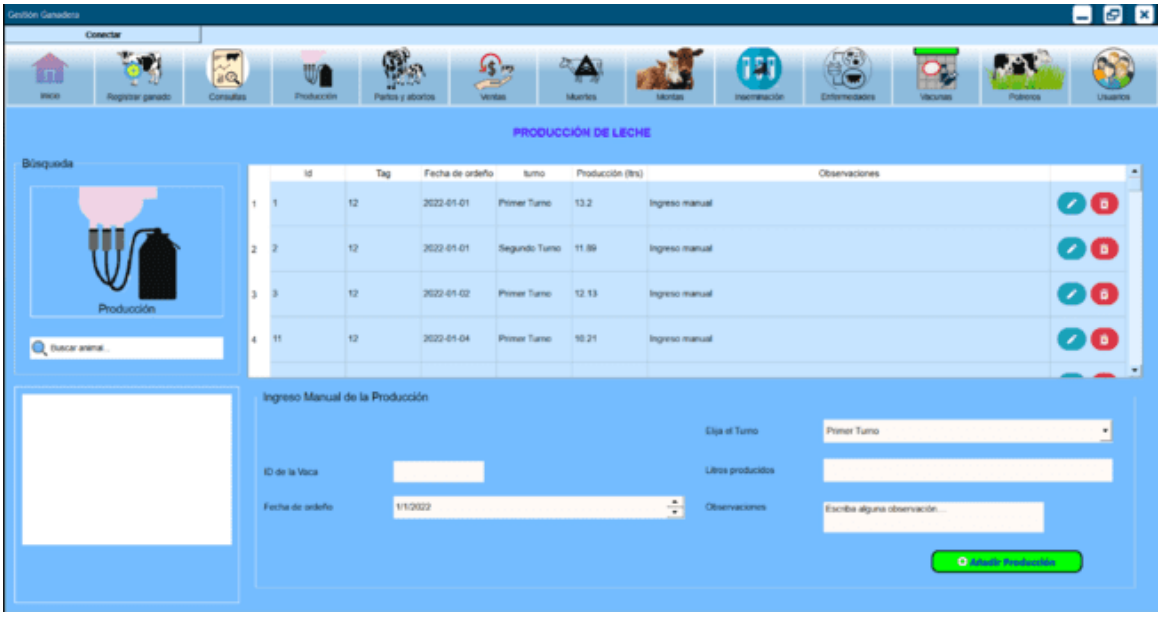
Figura 108

Página registrar ganado

Módulo registrar producción. En esta página se muestra la tabla con el registro de producción de toda la hacienda, también se pude registrar un nuevo ganado haciendo uso del botón “añadir producción”. Además, se posee dos botones por cada fila de la tabla, un botón de editar registro y otro de eliminar registro, ver Figura 109.

Figura 109

Página registrar producción



Módulo registrar partos. Esta funcionalidad permite registrar los partos existentes en la hacienda mediante el botón “agregar parto”. Si la cría nace viva entonces se debe registrar dicho animal haciendo uso del botón “agregar cría”, ver Figura 110.

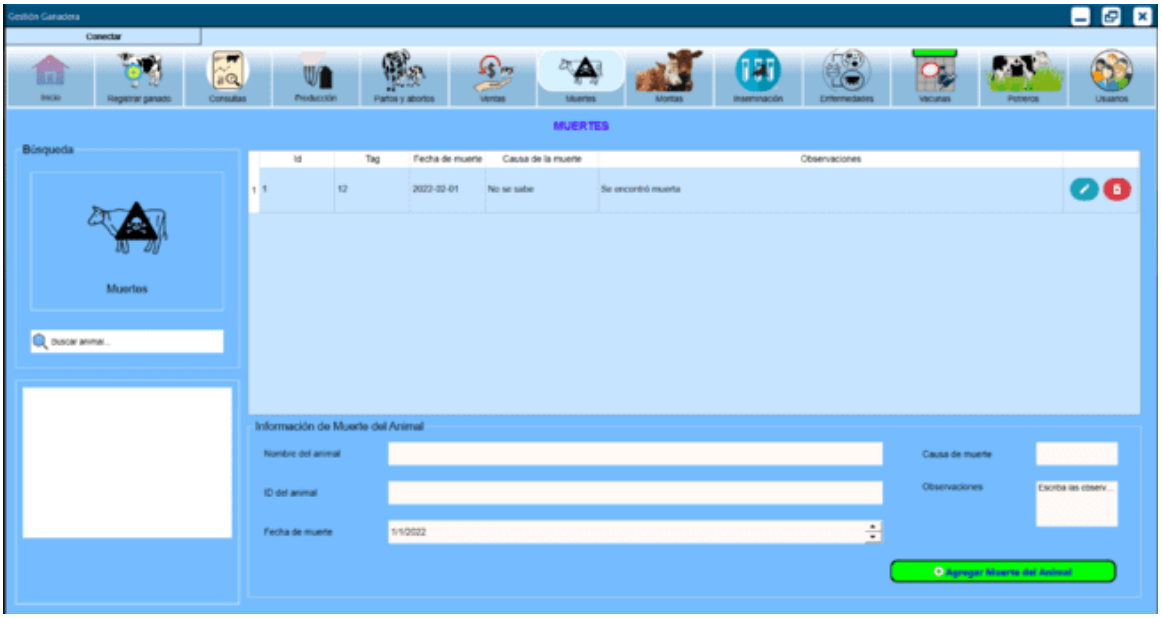
Figura 110

Página registrar partos

Módulo registrar bajas. Permite registrar una muerte y con ello se actualiza las existencias de ganado en la hacienda. Tambié se puede eliminar o editar un registro, ver Figura 111.

Figura 111

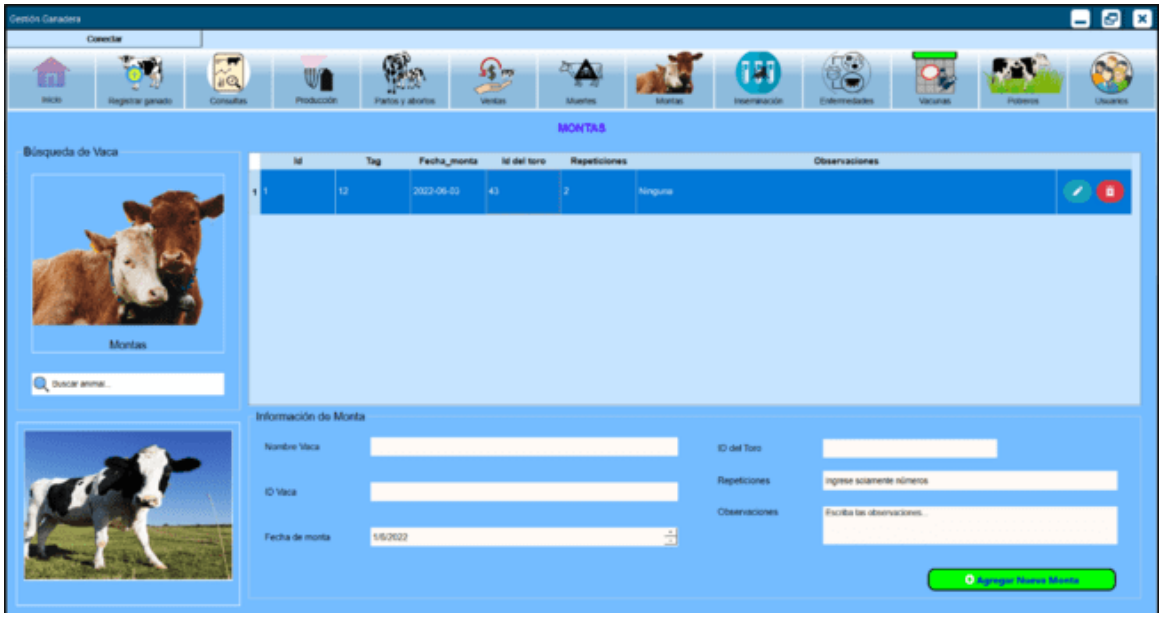
Página registrar bajas



Módulo registrar montas. Permite registrar, editar o eliminar una monta. Esto se muestra en la Figura 112.

Figura 112

Página registrar montas



Módulo registrar inseminación. Permite registrar, editar o eliminar la inseminacion de una vaca la interfaz se muestra en la Figura 113.

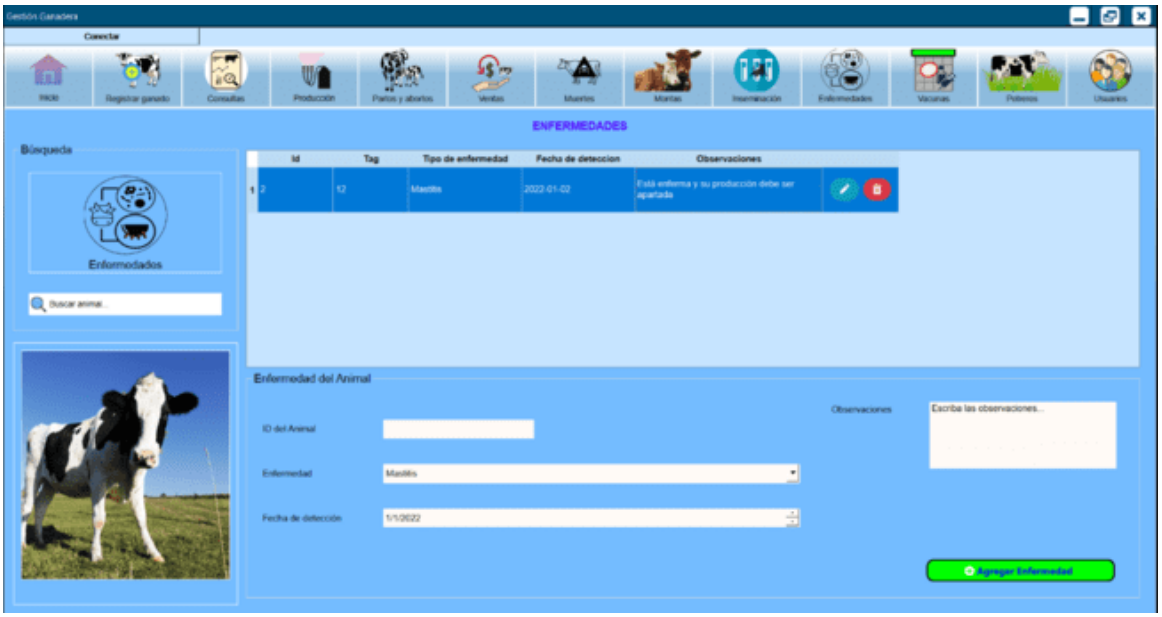
Figura 113

Página registrar inseminación

Módulo registrar enfermedad. Permite registrar, editar o eliminar enfermedades del ganado, la interfaz se visualiza en la Figura 114.

Figura 114

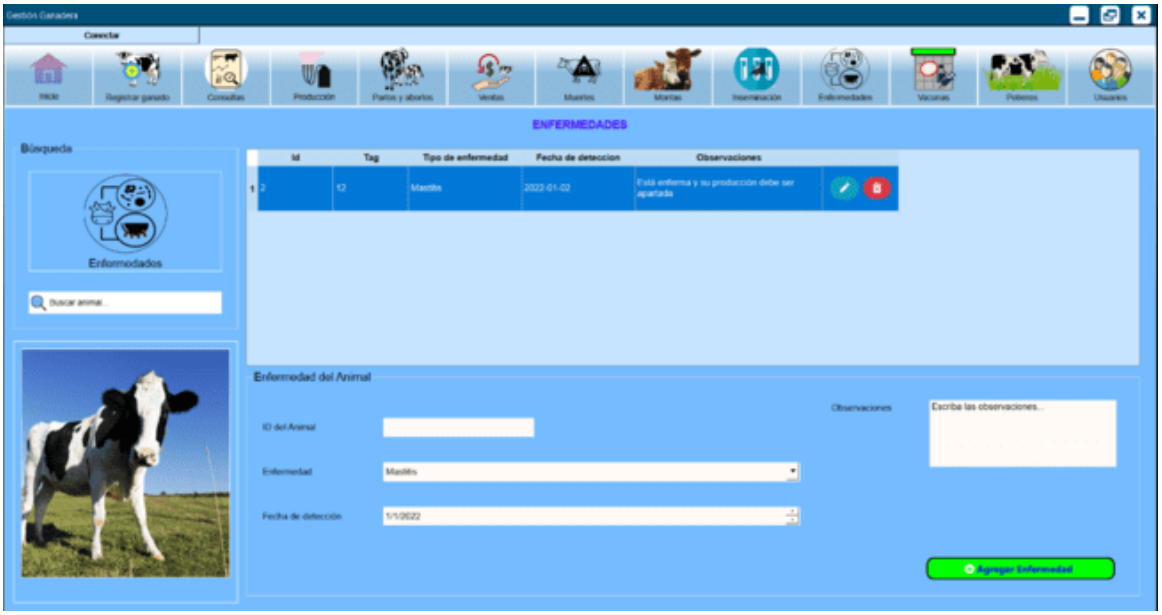
Página registrar enfermedad



Módulo registrar medicamentos. Permite registrar, editar o eliminar tratamientos del ganado los elementos de la interfaz se muestra en la Figura 115.

Figura 115

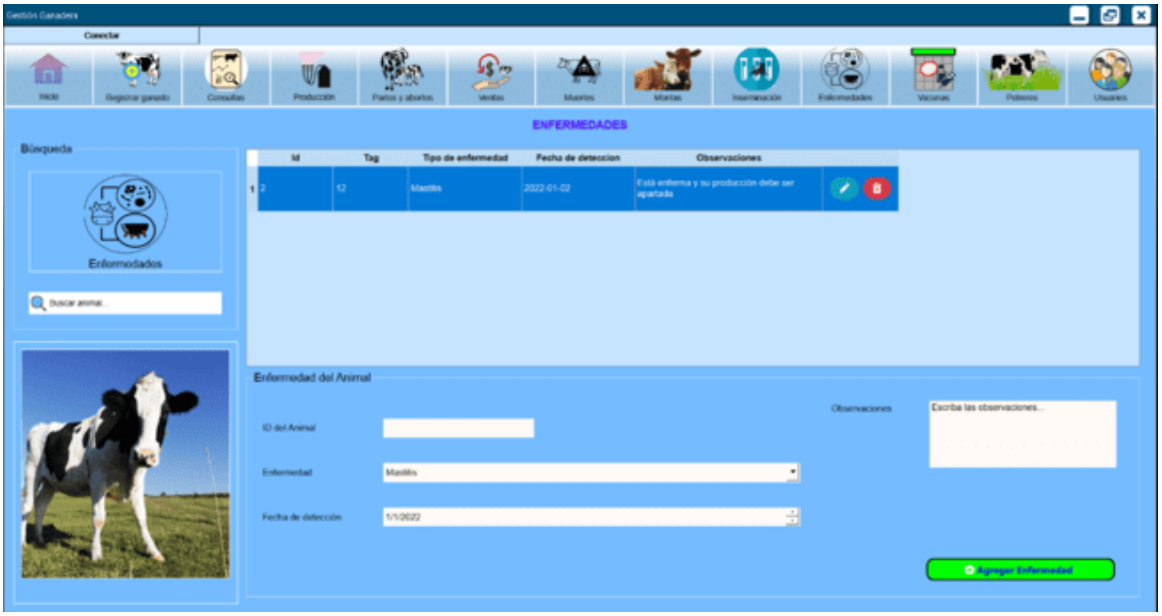
Página registrar medicamento



Módulo registrar medicamentos. Permite registrar, editar o eliminar tratamientos del ganado como se indica en la Figura 116.

Figura 116

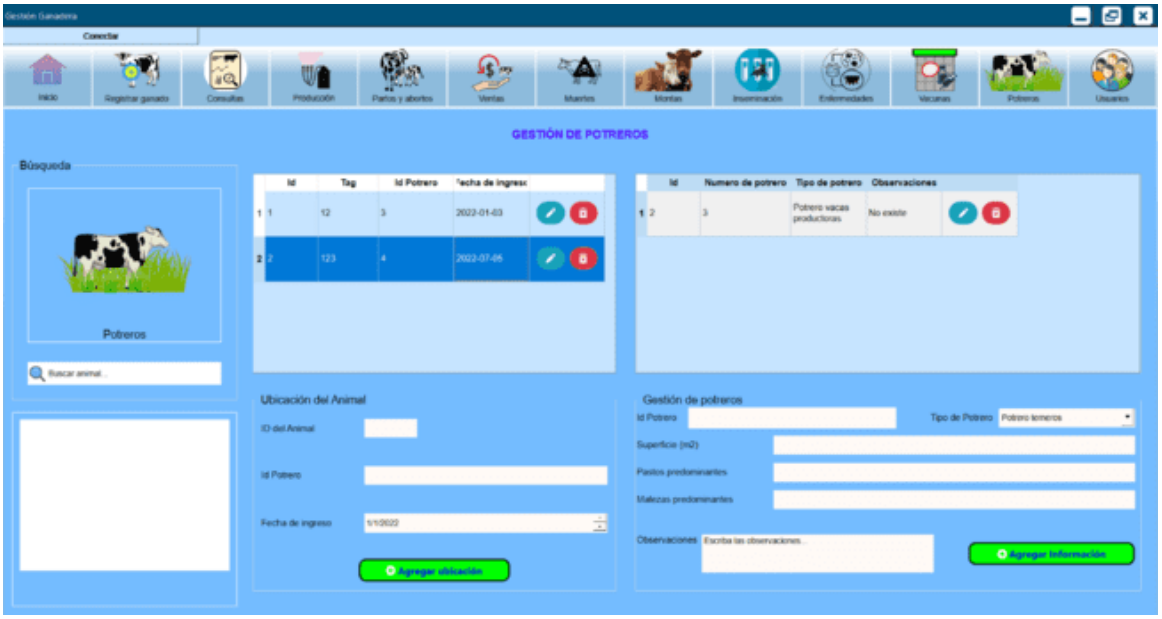
Página registrar medicamento



Módulo potreros. Permite registrar, editar o eliminar potreros, además, se puede agregar la ubicación de un animal en el potrero como se indica en la Figura 117.

Figura 117

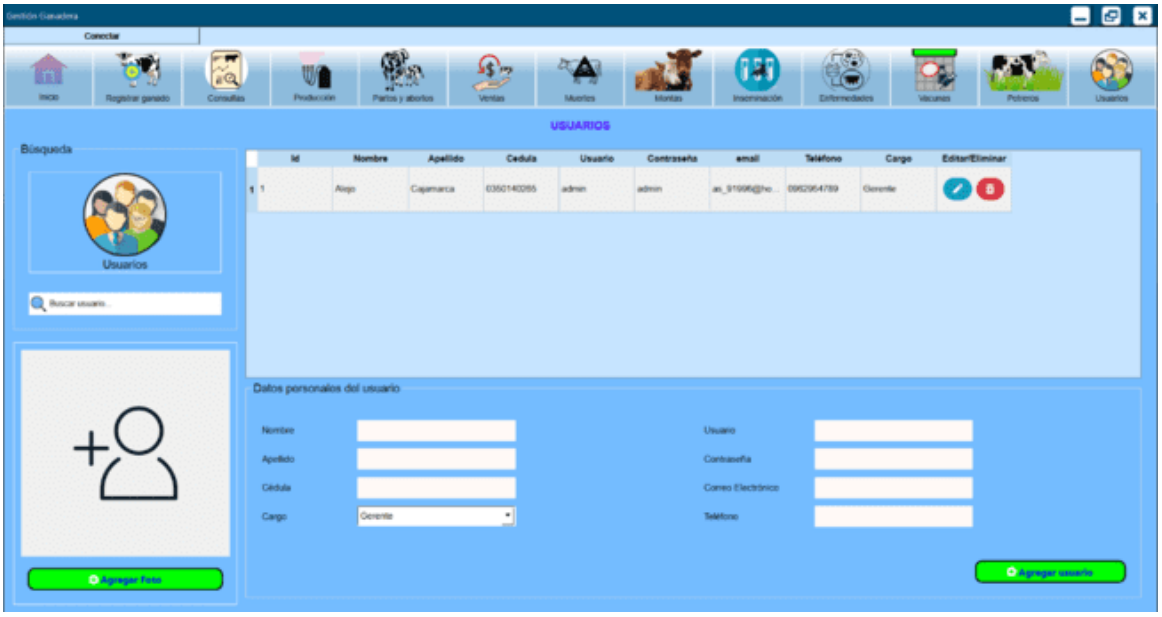
Página potreros o ubicación



Módulo registrar usuario. Permite registrar, editar o eliminar usuarios para el control de acceso, ver Figura 118.

Figura 118

Página registrar usuario



Módulo consulta. Permite visualizar la información del ganado. Posee dos botones, el botón filtrar que permite filtrar la producción por fecha o por parto y un botón de visualizar gráficas de la producción, ver Figura 119.

Figura 119

Página consulta

En la Figura 120 presenta la interfaz gráfica del historial de la producción de manera gráfica, donde se puede filtrar por fecha o por parto.

Figura 120

Página grafica de producción

Integración de los sistemas

Para la implementación del sistema integral, se tomará como referencia los diagramas de flujo expuestos en el diseño. En resumen, se cuenta con tres buses, el bus que comunica los lectores con el controlador, el cual maneja un protocolo propietario y como medio físico el protocolo RS-485, el MODBUS que comunica el sistema de medición de la producción mediante protocolo RS-485 para el medio físico, y el MODBUS que comunica el sistema de gestión mediante protocolo RS-485 para el medio físico. La comunicación del sistema de identificación es parte del sistema como tal, por lo tanto, ya está implementado.

Para la comunicación mediante protocolo MODBUS, Python dispone de la librería pymodbus, esta librería pose varios modos de transmisión y la que se eligió es el MODBUS RTU. Para hacer uso de esta librería se debe instalar dentro del entorno virtual creado creado anteriormente en la SBC Odroid, haciendo uso del comando *pip3 install -U pymodbus,* de igual

manera se realiza la instalación en los esclavos empleando el mismo comando tanto para Linux como para Windows.

Implementación del controlador principal

Con base en el diseño de la arquitectura integral del sistema desarrollado en el capítulo anterior se realiza la implementación del código en Python para el controlador principal que es la SBC ODROID. La aplicación inicia con la configuración inicial que se muestra en la Figura 121, en el que se crea dos objetos de comunicación, uno para la comunicación con los lectores (objeto lector) y otro para la comunicación a través del MODBUS (objeto client). La función *configuracion_inicial* devuelve la variable connection y el objeto lector, si se inicia la comunicación correctamente la variable connecction tendrá un valor de True, caso contrario False. La SBC Odroid es el maestro en la comunicación del sistema.

Figura 121

Código empleado para inicializar el MODBUS en el lado del maestro

```
def configuracion_inicial():
    port = 'ttyRFID'
    baudrate = 19200
    parity = serial.PARITY_NONE
    stopbits = serial.STOPBITS_ONE
    bytesize = serial.EIGHTBITS
    timeout = 0.1
    lector = serial.Serial(port=port, baudrate=baudrate, parity=parity,stopbits=stopbits,
                           bytesize=bytesize, timeout=timeout)
    client = ModbusClient(method='rtu', port='ttyMODBUS1', stopbits=1, bytesize=8, parity='N',
                          baudrate=19200,timeout=0.01)
    connection = client.connect()
    client2 = ModbusClient(method='rtu', port='ttyMODBUS2', stopbits=1, bytesize=8, parity='N',
                           baudrate=19200,timeout=0.01)
    connection2 = client2.connect()
    return connection, connection2, lector
```

El proceso siguiente consiste en la inicialización de las variables que se emplearán en la implementación del código, ver Figura 122.

Figura 122

Código empleado para inicializar las variables del maestro

```
def configuracion_proceso():
    tag_ant0 = 0; tag_ant1 = 0; num_lectora = 170 ; primer_lado = 170
    prioridad = np.zeros(6); cont_lectora0 = 0; cont_lectora1 = 0
    encontro = 0; vaca_extra = 0; tag_puestos = np.zeros((2,8))
    num_vacas = np.zeros(8); tic = 0; tic_ant0 = 0; tic_ant1 = 0
    tic_aux = 0; tic_entre_vacas0 = 1; tic_entre_vacas1 = 0
    tic_ant_entre_vacas = 0; medida = np.zeros(12); med_ant = np.zeros(6)
    med_act = np.zeros(6); tag_nuevo0 = 0; tag_nuevo1 = 0; cont_aux1 = 0
    aux_enc = 0; tag = 0; num_med = 0; longitud = 0; medida_int = 0
    lado = 170; error_enviar_slave1 = 1; error = 0; aux_enc1 = 0
```

Una vez inicializado las variables se verifica si la conexión con el MODBUS fue exitosa, si lo fue entonces inicia con el sistema de identificación, es decir el maestro envía peticiones de lectura de etiquetas RFID, de acuerdo al diagrama de flujo de la Figura **49**. En la Figura 123.

Figura 123

Código empleado para la implementación del proceso leer_rfid1_rfid2

```
def leer_rfid1_rfid2(self):
    if self.enc_med == [85,85,85]: self.aux_enc1 = 0
    if self.aux_enc == 1:
        objeto.control1()
    lector.write(bytes.fromhex('7c 01 00 21 00 07 00 00 00 00 01 02 02 56'))
    dato0=lector.readline()
    longitud0= dato0.__len__()
    if longitud0==26:
        objeto.control()
        entero0=struct.unpack('<BBBBBBBBBBBBBBBBBBBBBBBBBB', dato0)
        self.tag = entero0[13]*255+dato0[14]+entero0[13]
        self.tag_ant0 = self.tag_nuevo0
        self.tag_nuevo0 = self.tag

        self.tic_entre_vacas0 = self.tic - self.tic_ant0
        self.tic_ant0 = self.tic
        self.tic_ant_entre_vacas = self.tic
```

Sistema de identificación

Cálculo del tiempo entre vacas

Al iniciar el proceso por primera vez, los puertos USB de los esclavos están apagados, para encenderlos se debe enviar una orden desde el maestro cada vez que se detecte la primera vaca en cada turno de ordeño, la función *control1* de la Figura 123, sirve para

direccionar hacia la función de encendido de puertos, pero antes de hacerlo verifica si aún no se han encendido. En la Figura 124 se implementa el código, el valor 170 hace referencia al estado apagado de los puertos, es por ello que si el puerto está apagado entonces llama a la función *encender_puertos_slaveX.*

Figura 124

Código para llamar a la función encender puertos

```
def control1(self):
    self.aux_enc1=1
    if (self.enc_med[0] == 170): objeto.encender_puertos_slave1()
    if (self.enc_med[1] == 170): objeto.encender_puertos_slave2()
    if (self.enc_med[2] == 170): objeto.encender_puertos_slave3()
    return
```

El encendido de puertos se lleva a cabo mediante el código de la Figura 125. Debido a que algunas veces no se realiza correctamente la lectura o escritura en el MODBUS, se emplea la instrucción *try-except* para saltar los posibles errores presente en el programa, y dado a que los esclavos cumplirán con el mismo funcionamiento, el proceso de encendido de los otros esclavos es el mismo. Este código esta implementado de acuerdo el diagrama de flujo de la Figura 49.

Figura 125

Código empleado para apagar los puertos USB de los esclavos del sistema de medición de la producción

```
def encender_puertos_slave1(self):
    try: client.write_register(8, 5, unit=1) #(addres, value, slave)
    except: pass
    try:
        aux1 = client.read_holding_registers(address=9, count=1, unit=1)
        aux1 = aux1.registers[0]
    except:
        try:
            aux1 = client.read_holding_registers(address=9, count=1, unit=1)
            aux1 = aux1.registers[0]
        except:
            try:
                aux1 = client.read_holding_registers(address=9, count=1, unit=1)
                aux1 = aux1.registers[0]
            except: pass
    if aux1 == 5:
        self.enc_med[0] = 85
        try:
            client.write_register(9, 0, unit=1); return
        except: pass
    else:
        self.cont_aux1 = self.cont_aux1 + 1
        if self.cont_aux1 == 50:
            self.cont_aux1=0; return
    return
```

Continuando con el código para la implementación de la función *leer_rfid1_rfid2(),* cada que se detecta y lee una etiqueta se verifica si no es igual a la anterior y además se verifica que el tiempo entre vacas no sea muy grando, porque significaría que es un nuevo grupo de seis vacas las que están ingresando. Antes de guardar el valor del id del ganado se verifica si se han ingresado seis o menos vacas, en el caso de que hayan ingresado mas de seis, entonces se realiza el contéo de las vacas extras y si es el caso la verificacion de que se hayan sacado dichas vacas. El código para realizar este proceso se muestra en la Figura 126, en el cual al final se tendrá registrado los ids de las seis vacas ingresadas a la sala de ordeño, este mismo procedimiento se hace para el lector 2.

Figura 126

Segunda parte del código que realiza la función leer_rfid1_rfid2

```
if (self.tag_ant0 == self.tag_nuevo0 and self.tic_entre_vacas0<100): return
else:
    if self.num_lectora != 0: self.cont_lectora0 = 0
    else: self.cont_lectora0 = self.cont_lectora0 + 1
    if self.primer_lado == 170:
        self.prioridad[self.cont_lectora0] = 0
        if (self.cont_lectora0 ==5 or self.tic_entre_vacas0 >5000):
            self.primer_lado = 0
    if (self.cont_lectora0 == 6): self.vaca_extra = 1
    else:
        if self.cont_lectora0 ==7:
            self.vaca_extra = 2
            self.encontro = objeto.buscar_tag()
            if self.encontro == 1:
                self.cont_lectora0 = 5
                self.vaca_extra = 0
                self.encontro = 0
    self.num_lectora = 0
    self.puesto = self.cont_lectora0
    self.tag_puestos[0][self.puesto] = self.tag

    if self.puesto<=5:
        self.prioridad[self.puesto]= self.prioridad[self.puesto] + 1
        self.num_vacas[self.puesto]= self.num_vacas[self.puesto] + 1
```

Cuenta el número de vacas ingresadas

Asigna el número de lado

Si se detectan mas de 6 etiquetas verifica si es una vaca extra

Guarda los identificadores del ganado en la matriz de tags

Al encenderse los puertos también se realiza la lectura de la medición de la producción en todo momento, y de todos los esclavos. En la Figura 127 se presenta el código que permite leer los datos de la medión de la producción en “tiempo real”, además se verifica cuando hay un reseteo de la media, lo que implica que se terminó el ordeño de una vaca, para guardar el dato en un fichero.

Figura 127

Código para la obtención de los valores de medición de la producción de los esclavos 1, 2 y 3

```
def obtener_medicion_medidor(self):
    if self.enc_med != [85,85,85]: return
    slaveId = 0
    for self.num_med in range(1,7):
        slaveId = slaveId + self.num_med%2; address = self.num_med - 1
        try:
            result = client.read_holding_registers(address=address, count=1, unit=slaveId)
            self.medida_int = result.registers[0]/100
            self.med_ant[self.num_med - 1] = self.med_act[self.num_med - 1]
            self.med_act[self.num_med - 1] = self.medida_int
            if ((self.med_act[self.num_med - 1])+0.01) < self.med_ant[self.num_med - 1]:
                objeto.guardar_dato()
        except:
            try:
                result = client.read_holding_registers(address=address, count=1, unit=slaveId)
                self.medida_int = result.registers[0]/100
                self.med_ant[self.num_med - 1] = self.med_act[self.num_med - 1]
                self.med_act[self.num_med - 1] = self.medida_int
                if ((self.med_act[self.num_med - 1])+0.01) < self.med_ant[self.num_med - 1]:
                    objeto.guardar_dato()
            except:
                try:
                    result = client.read_holding_registers(address=address, count=1, unit=slaveId)
                    self.medida_int = result.registers[0]/100
                    self.med_ant[self.num_med - 1] = self.med_act[self.num_med - 1]
                    self.med_act[self.num_med - 1] = self.medida_int
                    if ((self.med_act[self.num_med - 1])+0.01) < self.med_ant[self.num_med - 1]:
                        objeto.guardar_dato()
                except: pass
    return
```

El proceso de guardar los datos de los tags y mediciones de leche se realiza mediante las funciones *guardar_dato()* y *escribir_fichero()* que se muestra en la Figura 128. En la función guardar el dato se almacena en el vector *medida* de doce posiciones que corresponde a los doce puestos de ordeño y la función escribir fichero crea un archivo plano, es este caso un archivo .csv, y almacena los datos de identificacion del ganado con sus respectivas mediciones.

Figura 128

Código para la guardar un dato en un fichero (archivo .csv)

```
def guardar_dato(self):
    if self.num_vacas[self.num_med-1] > 1:
        self.lado = self.prioridad[self.num_med-1]%2
    else:
        self.lado = (self.prioridad[self.num_med-1]+1)%2
    self.num_vacas[self.num_med-1] = self.num_vacas[self.num_med-1] - 1

    if self.lado == 0:
        self.medida[self.num_med-1] = self.med_ant[self.num_med-1]
    else:
        self.medida[self.num_med+5] = self.med_ant[self.num_med-1]

    objeto.escribir_fichero()

def escribir_fichero(self):
    dato_guardar = [int(self.tag_puestos[int(self.lado)][self.num_med-1]),
                    self.medida[int(self.num_med-1+self.lado*6)]]
    archivo = "registro_ordeño.csv"
    columnas = ['Tag', 'Medicion']
    df1 = pd.DataFrame([dato_guardar],columns=columnas)
    df1.to_csv(archivo, index=None, mode="a", header=not os.path.isfile(archivo))
```

Algo importante de mencionar es que el código anterior crea un archivo siempre y cuando no exista, si ya existe solo guarda los datos. Para manejar archivos planos es importante instalar las librerías *pandas* con el comando *pip3 install pandas* dentro del entorno virtual.

La última función de la Odroid que se diseñó en el capítulo anterior es el envío de datos desde el controlador hacia el sistema de gestión, esta función se ejecutará de forma indefinida solamente cuando no se esté ordeñando, es decir cuando los puertos de las SBC AML-S905X-CC que corresponden a las cámaras estén apagados, En la Figura 129 se muestra el código que permite realizar el envío de los datos del archivo plano hacia el sistema de gestión.

Figura 129

Código para el envío de datos desde la Odroid al sistema de gestión

```
client2.write_register(14, 0, unit=4)
try:
    aux0 = client2.read_holding_registers(address=12, count=1, unit=4)
    aux0 = aux0.registers[0]
except:  pass
while connection2 == True and aux==1 and aux0 == 5:
    nombre_archivo = "ord_evidencia.csv"
    tag_litro_r=[]
    with open(nombre_archivo,"r") as archivo:
        next(archivo, None)
        for linea in archivo:
            linea = linea.rstrip()
            separador = ","
            lista = linea.split(",")
            tag_litro_r.append(int(lista[0]))
            tag_litro_r.append(float(lista[1]))
    aux2=0
    while aux2 < len(tag_litro_r):
        try:
            aux2 = client2.read_holding_registers(address=13, count=1, unit=4)
            aux2 = aux2.registers[0]
        except: pass
        if aux2 == 6:
            if np.mod(aux2,2) == 0:
                client2.write_register(15, int(round(tag_litro_r[aux2],2)), unit=4)
            else:
                client2.write_register(15, int(round(tag_litro_r[aux2],2)*100), unit=4)
            client2.write_register(13, 0, unit=4)
            aux2 = aux2+1
    aux=0
    client2.write_register(14, 5, unit=4)
```

Implementación del lado de los esclavos del sistema de medición de la producción.

Anteriormente se implementó el código para el reconocimiento de imágenes del display del medidor, es decir se implementó un sistema para obtener el valor de la medida de la producción de cada vaca, este valor está presente en la SBC AML-S905X-CC de manera temporal, para no perder estos datos se debe enviar hacia el controlador. Por tal motivo es necesario realizar una comunicación por protocolo MODDBUS entre el controlador Odroid y la

SBC esclavo. En el maestro ya se implementó el código para solicitar al esclavo los datos de producción, y en esta sección se implementará la respuesta del esclavo.

Para lograr la comunicación se instala en el esclavo la librería *pymodbus* en el entorno virtual antes creado y siguiendo el diagrama de flujo de la Figura 130 se implanta el código. Antes de seguir con la lógica de programación se instala las librerías necesarias, en la Figura 130 se importa todas las clases de la librería pymodbus que utilizarán el cual forma parte de la configuración inicial, la librería threading permite trabajar con hilos los cuales se emplearán para trabajar en paralelo tanto el reconocimiento de imágenes y la comunicación. La librería *sys* permite ejecutar comandos de consola desde Python.

Figura 130

Librerías necesarias para la comunicación mediante protocolo MODBUS

```
from pymodbus.version import version
from pymodbus.server.sync import StartSerialServer
from pymodbus.device import ModbusDeviceIdentification
from pymodbus.datastore import ModbusSparseDataBlock
from pymodbus.datastore import ModbusSlaveContext, ModbusServerContext
from pymodbus.transaction import ModbusRtuFramer

import threading
import sys
```

El proceso del esclavo inicia con el establecimiento de la comunicación y el proceso obtener medida, ambos procesos funcionan en paralelo de forma indefinida, por tal motivo se llama a la función *obtenerMedida* mediante un hilo como se ilustra en la Figura 131.

Figura 131

Ejecución del proceso obtenerMedida y la comunicación MODBUS

```
# ------------------------------HILO-------------------------
hilo = threading.Thread(target=obtenerMedida, args=(context,))
hilo.daemon = True
hilo.start()
# MODBUS RTU:
try:
    print("Start server...")
    updating(a=(context,))
    StartSerialServer(context, framer=ModbusRtuFramer, identity=identity,
                      port='//dev/ttyMODBUS', timeout=.005, baudrate=19200)
except:
    print("")
    updating(a=(context,))
    print("Shutdown server ...")
    sys.exit()
```

Con base en el diseño, como siguiente paso se inicializan las variables del proceso, como se observa en la Figura 132.

Figura 132

Código correspondiente a la configuración del proceso del esclavo 1

```
#---------INICIALIZAMOS VARIABLES-------------
slave_id = 0x01
medidor0= [0,0]
medidor1= [0,0]
caso =0
caso1=0
bloq1=0
bloq=0
encendido = 2
aux=1
```

El proceso principal ejecutado como un hilo cumple la función de responder al Odroid las peticiones de encender puertos o apagar puertos, al encender los puertos inmediatamente inica el proceso de reconocimiento de imágenes y al apagarse los puertos se reinician las variables, esto se describe en la Figura 133.

Figura 133

Función principal del esclavo

```
while True:
    register = 3    #Función de lectura de registros
    address = 8     #Dirección de registro
    values = context[slave_id].getValues(register, address, count=1)

    if values[0] == 5 and aux==1:
        calibrarCamara2.encender_puertos(context,slave_id)
        register = 0x10     #Función de escritura de registros
        address = 8 ;       #Dirección de registro
        context[slave_id].setValues(register, address, [0])
        cap0, cap1 = definirCamaras()
        encendido = 1; aux=0
    values = context[slave_id].getValues(3, 8, count=1)

    if values[0] == 6:
        calibrarCamara2.apagar_puertos(context,slave_id)
        context[slave_id].setValues(0x10, 8, [0])
        encendido = 0

    if encendido == 1:
        medidor0,medidor1,caso,caso1,bloq,bloq1 = calibrarCamara2.iniciarProcesamiento(context,cap0,cap1,medidor0,
                                                                                        medidor1,caso,caso1,bloq,bloq1)
        t = cv2.waitKey(1)
        if t == 27:
            encendido = 0

    if encendido == 0:
        medidor0[0]=0
        for ele in range(1,len(medidor0)):
            del medidor0[1]
        medidor1[0]=0
        for ele in range(1,len(medidor1)):
            del medidor1[1]
```

Si recibe un valor de 5 en el registro 8 por parte del esclavo, entonces enciende los puertos USB.

Si recibe un valor de 6 en el registro 8 por parte del esclavo, entonces apaga los puertos USB.

Una vez encendido los puertos, se inicia el proceso de reconocimiento de imágenes.

Si se apagan los puertos, se reinician las variables que almacenan el valor de las mediciones

Del mismo modo se realizan para los otros dos esclavos, teniendo finalmente la comunicación entre el controlador y el sistema de medición.

Implementación del lado del esclavo del sistema de gestión

Para realizar las funciones de respuesta del sistema de gestión al controlador se emplea la misma lógica que los esclavos 1, 2 y 3, con las mismas librerías, pero el inicio de la comunicación se establece cuando el usuario quiere realizar una lectura de datos del sistema de ordeño desde la aplicación de escritorio. La petición de lectura se realiza con el botón “Conectar” que se encuentra en la parte superior izquierda de la aplicación de escritorio.

El código para ejecutar la comunicación se presenta en Figura 134.

Figura 134

Código para establecer la comunicación MODBUS en el sistema de gestión

```
def comunicar(self):
    if self.aux_com==1:
        self.hilo = threading.Thread(target=self.conectar, args=(self.context,))
        self.hilo.daemon = True
        self.aux_com=0
        self.hilo.start()
    else: pass

    if self.aux_com == 0:
        self.hilo2 = threading.Thread(target=self.leer_datos, args=(self.context,))
        self.hilo2.daemon = True
        self.hilo2.start()
```

La función conectar tiene la misma lógica de funcionamiento que en los esclavos del sistema de medición con la diferencia que aquí el MODBUS se ejecuta como un hilo y la el código que ejecuta la aplicación de escritorio funciona como el *while* principal de este esclavo. Luego de establecer la comunicación se inicia el proceso de lectura de datos del controlador y subida a la base de datos, también como un hilo, este código se muestra de forma detallada en la Figura 135, los datos de ordeño están almacenados en la lista *tag_litros_list.*

Figura 135

Código para leer datos de ordeño enviados desde el controlador

```
def leer_datos_odroid(self):
    tag_litro_list = []
    self.context[4].setValues(0x10, 12, [5])
    self.context[4].setValues(0x10, 13, [6])
    cont =0
    tagr_nuevo = 55

    while cont < 50000 and self.aux_com==0:
        values1 = self.context[4].getValues(3, 13, count=1)
        if values1[0] == 0:
            tag_r = self.context[4].getValues(3, 15, count=1)
            tagr_ant = tagr_nuevo
            tagr_nuevo=tag_r
            if tag_r != 0 and tagr_ant==tagr_nuevo:
                tag_litro_list.append(tag_r)
                self.context[4].setValues(0x10, 13, [6])
                cont = 0
        values2 = self.context[4].getValues(3, 14, count=1)
        cont = cont+1
        print(cont)
        if values2[0]  == 5:
            cont = 100002
```

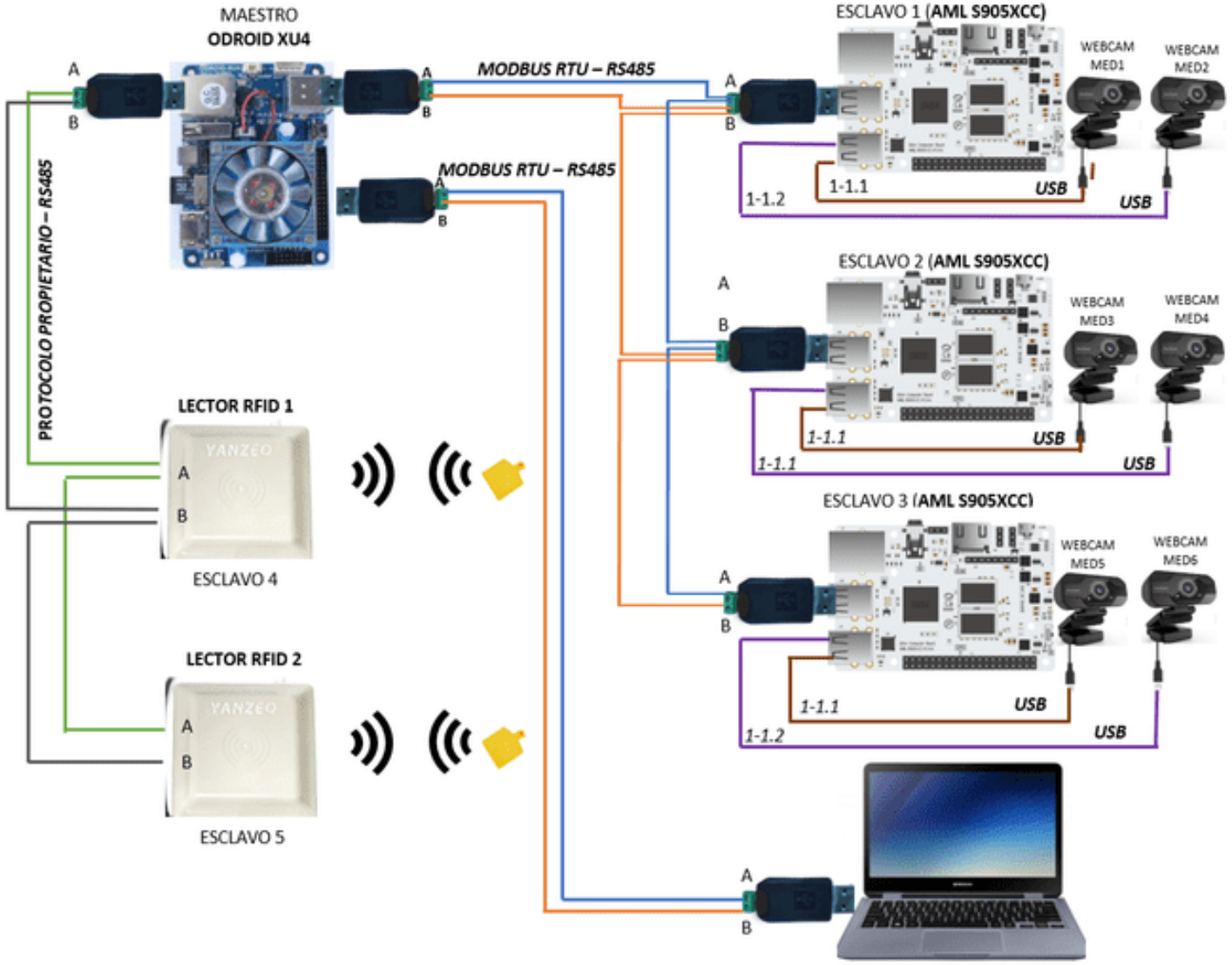
Adicionalmente se realiza el proceso de guardar los datos en la base de datos, empleando la lógica de “insertar datos” en la base de datos presente en la Figura 135. Todo este proceso se realiza al accionar el botón conectar de la aplicación de escritorio siempre y cuando el sistema de ordeño no se encuentre activo, es decir cuando no hay vacas en el establo.

Instalación y pruebas finales

En esta sección se realiza la aplicación del sistema propuesto, es decir la instalación en campo bajo condiciones de temperatura, humedad, ruido eléctrico producido por la bomba de vacío, distancia de transmisión de los datos, entre otros, que le caracterizan a un sistema de ordeño mecánico y que pueden producir errores en la transmisión de datos entre sistemas. Como base para la instalación se considera el esquema eléctrico de la Figura 136.

Figura 136

Diagrama esquemático de los sistemas de identificación, medición y de gestión para la Hacienda Bellavista



Instalación

Para realizar la instalación en primera instancia se estableció el lugar estratégico donde se ubicarán las SBC's, tomando como referencia la ubicación de las cámaras. En la Figura 137 se muestra el lugar elegido con base en lo siguiente:

- Debe ser un lugar no accesible por los usuarios que transitan al momento del ordeño.
- Dado a que se realiza frecuentemente lavados en toda la sala de ordeño, debe evitarse que el agua llegue al tablero.
- Al estar conectadas las cámaras a las SBC's debe elegirse un lugar central, y de este modo se evita el uso de adaptadores USB demasiados grandes, mismos que pueden generar errores en la captura de la imagen.

Figura 137

Ubicación del tablero donde se ubicarán las SBC's

Como siguiente paso se definen las dimensiones del tablero, esto depende del número de dispositivos que se ubicarán dentro del él, además debe ser posible la manipulación en el

caso de que se quiera realizar una revisión o incluso en la misma instalación. Dentro del tablero estarán ubicadas las 4 computadoras, 5 conversores USB a RS-485 y además debe disponer de espacio adicional para futuras conexiones, por lo tanto, las dimensiones del tablero serán de 40cm de alto, 30 de ancho y 20 cm de profundidad, en la Figura 138 se muestra dicho tablero.

Figura 138

Tablero general para el montaje de las SBC's

De igual manera se define el lugar donde serán ubicadas las cámaras, estas serán ubicadas frente del medidor con el lente apuntando a la pantalla, ver Figura 139.

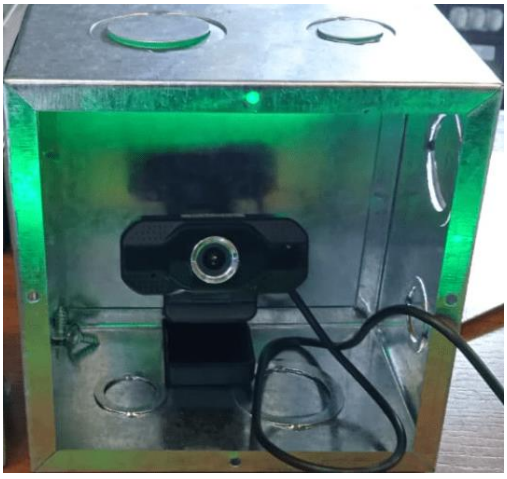
Figura 139

Ubicación de las cámaras

Para proteger las cámaras del agua y de la luz que se refleja en la pantalla se sobrepuso una caja como se puede apreciar en la Figura 140 cuyas dimensiones son; 15 cm de largo, 15 cm de alto y 10 cm de ancho.

Figura 140

Caja protectora para las cámaras



Para el sistema de identificación del ganado se define una ubicación estratégica de los lectores RFID considerando el alcance máximo que permite detectar las etiquetas. Por consiguiente, los lectores serán fijadas en las dos entradas de la sala de ordeño, en los lugares señalados en la Figura 141. Los lectores se fijarán con la antena apuntando hacia abajo.

Figura 141

Ubicación de los lectores RFID

Continuando con la instalación, se establece el lugar donde se situará la computadora con la aplicación de escritorio, que por comodidad se ubicará en el cuarto de control.

De acuerdo a lo antes expuesto se realiza el montaje de los elementos del sistema. En la Figura 142 se ilustra el montaje de los componentes del sistema.

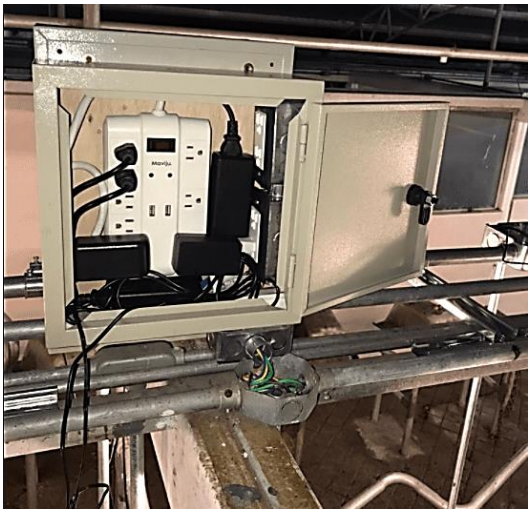
Figura 142

Montaje del tablero de control, cámaras y lectores

Para la alimentación de los dispositivos del sistema, como las SBC's, los lectores, etc., se instala un tablero adicional donde estarán ubicados los tomacorrientes, tal como se visualiza en la Figura 143.

Figura 143

Montaje del tablero de alimentación

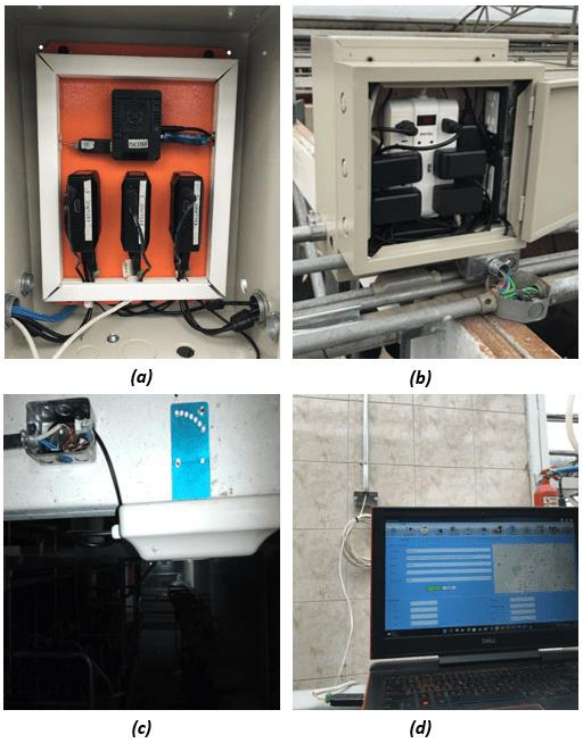


Como siguiente paso se selecciona las tuberías Conduit para el cableado del sistema, tomando en cuenta que los cables de datos deben estar aislados del cableado de alimentación.

Una vez realizado el montaje de todos los componentes se procede a el cableado y conexión física de todos los sistemas, es importante mencionar que para conectar las cámaras con las SBC’s se necesitó extensiones USB. La Figura 144 presenta las conexiones finales del sistema.

Figura 144

Conexiones de cada bloque del sistema



Nota. Instalación de: a) Tablero general, b) Tablero de alimentación, c) Lector RFID

Prueba de conectividad del sistema de identificación con el maestro

Parte de la implementación del sistema de identificación es el aretado, que consiste en colocar los tags RFID en la oreja del ganado, ver Figura 145. Previamente se debe almacenar el ID del ganado en la memoria interna del tag, esto se lo hace a través del software RFID Demo de Yanzeo, y para que sea visible se imprime el ID en la etiqueta.

Figura 145

Aretado del ganado

Para comprobar el correcto funcionamiento del sistema de identificación se conectó un monitor a la SBC utilizando su salida HDMI, posteriormente se seleccionaron 14 vacas y se ingresaron a la sala de ordeño, siete de cada lado, para ser detectadas por los lectores que anteriormente se colocaron en las dos entradas. La respuesta de los lectores se visualiza en la terminal de Visual Studio Code de la Figura 146. Este mismo procedimiento se realizó tres veces, dando un total de 42 vacas de prueba con cero errores, este resultado es muy importante, ya que el error en la identificación de una sola vaca implica mínimo seis errores en el sistema de registro de la producción.

Figura 146

Prueba de funcionamiento del sistema de identificación

Prueba de conectividad del sistema de medición con el maestro

Para verificar el funcionamiento del sistema de reconocimiento de medición se conectó una HMI a las SBC que funcionan como maestro y se verificó que el valor mostrado en el terminal como respuesta de del Script de Python muestre el mismo valor que se visualiza en la pantalla del medidor, de esta forma se verifica el funcionamiento del proceso de reconocimiento de imágenes y la comunicación mediante protocolo MODBUS. Para realizar las pruebas se ingresaron seis vacas al sistema de ordeño por la entrada del primer lector (lector 0), en la Figura 147 se muestra la respuesta del sistema. Al ingresar el primer animal el maestro envía ordenes de encendido de puertos a los tres esclavos (las tres SBC’s que realizan el procesamiento de imágenes), y una vez encendido todos los puertos inicia el procesamiento, inicialmente los medidores tienen en sus pantallas valores de “.00” por lo tanto en el maestro se reflejan esas medidas. En la Tabla 54 se muestra la comparación entre el valor real mostrado en la pantalla del medidor y el valor recibido en el dispositivo maestro.

Figura 147

Respuesta del maestro ante el ingreso del primer animal a la sala de ordeño

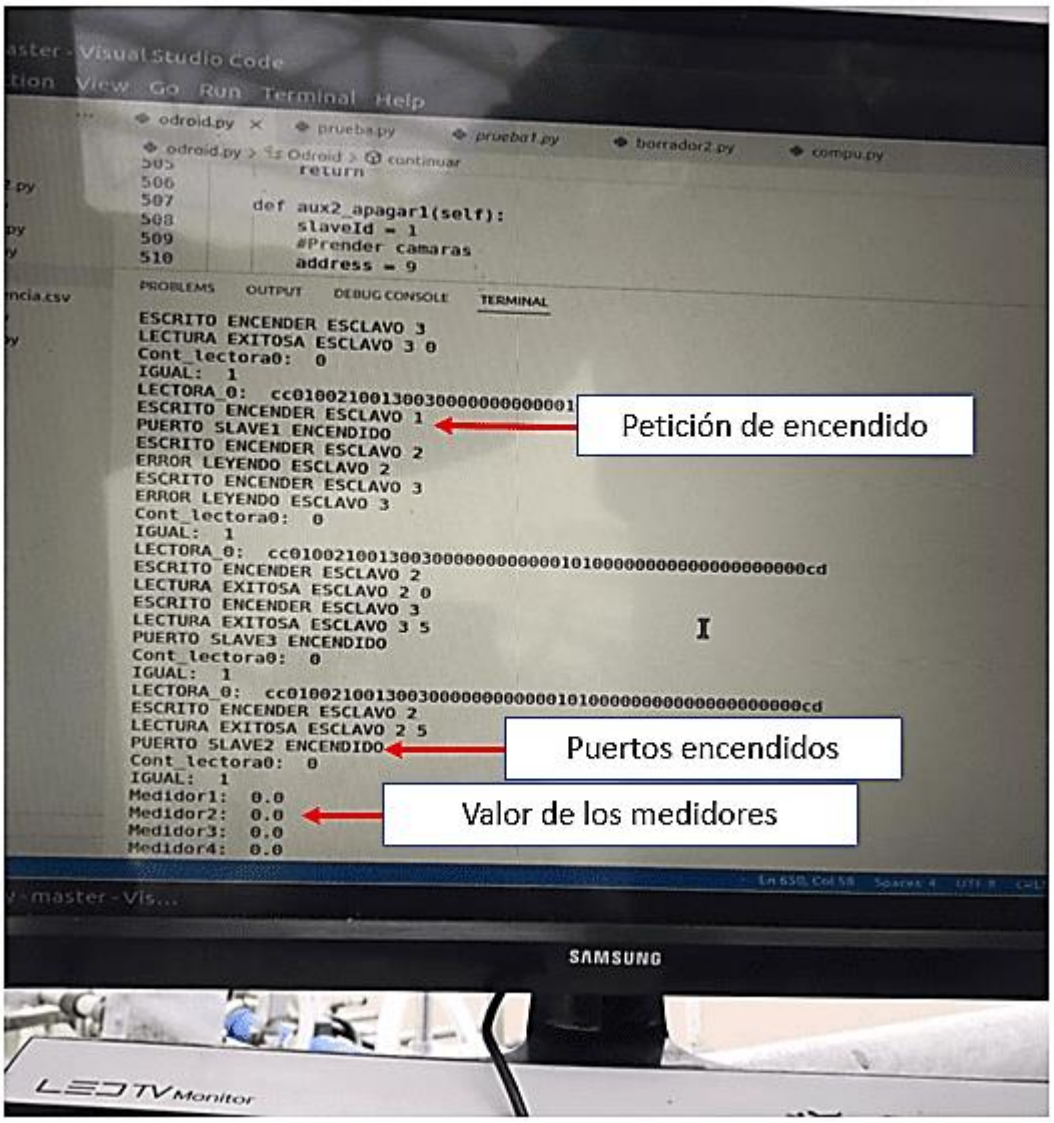


Tabla 54

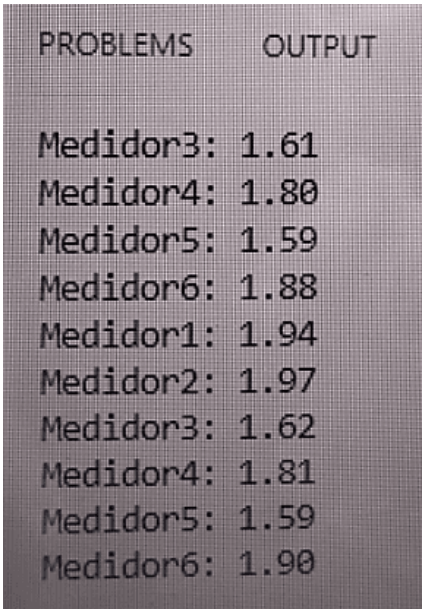
Comparación entre los valores reales del medidor y los valores obtenidos del reconocimiento de caracteres

Medidor	Pantalla	Dato reconocido
1		1.94
2		1.97
3		1.62
4		1.81
5		1.59
6		1.90

En la Figura 148 se muestra las mediciones recibidas por el maestro.

Figura 148

Respuesta del proceso de reconocimiento de caracteres



Prueba de conectividad del sistema de gestión con el maestro

Las pruebas finales se llevan a cabo en el sistema de gestión, ya que este se conecta con el sistema de medición e identificación mediante el maestro. Para comprobar que el sistema integral funciona correctamente se realiza una lectura de los datos de ordeño al finalizar este, la lectura se lleva a cabo desde la aplicación de escritorio accionando el botón “conectar”, como se muestra en la Figura 149.

Figura 149

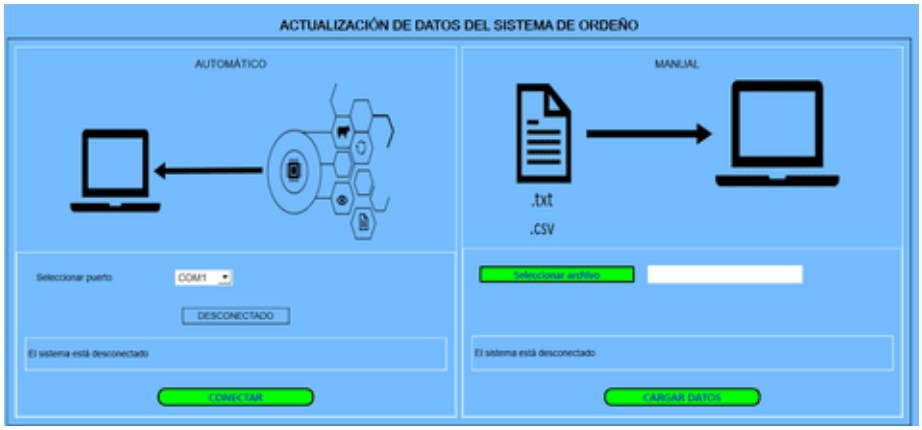
Botón que permite abrir la página de conexión con el sistema de ordeño

La ventana conectar presenta dos maneras de cargar los registros de producción a la base, la primera forma es mediante la comunicación por MODBUS y la segunda se realiza

mediante un archivo plano con los datos de laproducción. Esto se puede visualizar en la Figura 150.

Figura 150

Página de conexión con el controlador principal

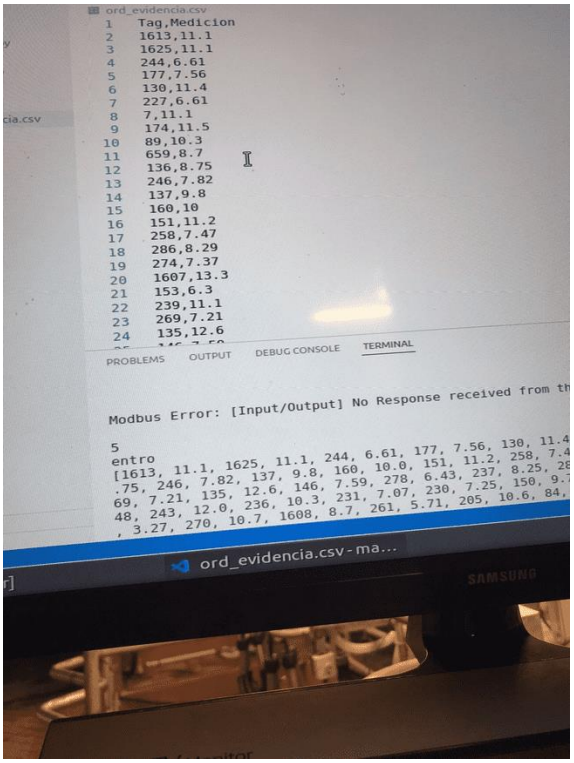


Para cargar los datos mediante la comunicación MODBUS se debe identificar el puerto, esto se lo realiza desde el administrador de dispositivos de la computadora. Una vez definido el puerto se establece la comunicación mediante el botón conectar, y los datos almacenados en el dispositivo maestro son reflejados en la tabla de producción de la aplicación de escritorio y en la base de datos.

Para verificar el funcionamiento de la comunicación se realizó una prueba con los datos de un turno de ordeño. Los datos almacenados en el maestro se muestran en la Figura 151, para visualizar los datos se conectó un monitor mediante HDMI al SBC maestro.

Figura 151

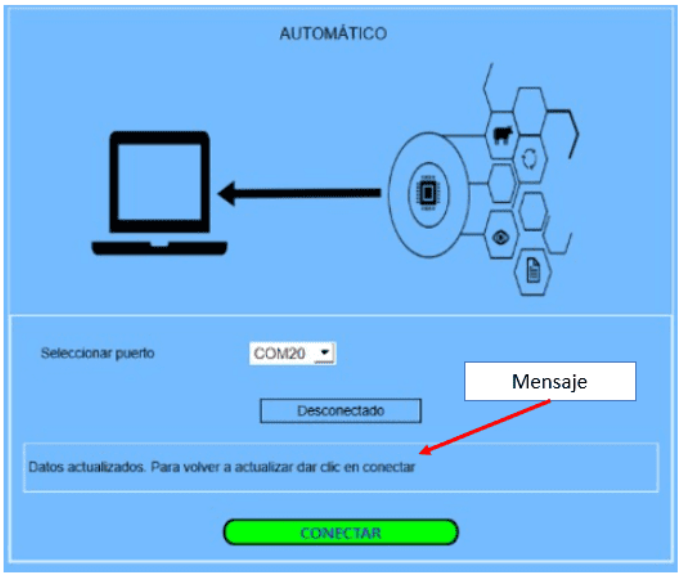
Datos de un turno de ordeño almacenado en la SBC Odroid



Al accionar el botón conectar se transmiten los datos al sistema de gestión. Cuando se finaliza la transmisión de datos, el software envía un mensaje que se muestra en la Figura 152, caso contrario se mostrará un error.

Figura 152

Mensaje de lectura exitosa de los datos recibidos del sistema del maestro



Posteriormente se verifica los datos recibidos en la tabla de la aplicación de escritorio y en la base de datos. En la Figura 153 se muestran algunos datos recibidos mediante la comunicación MODBUS, los datos fueron recibidos sin ningún error, por consiguiente se asegura el registro correcto de los datos de producción e identificacion del ganado.

Figura 153

Datos guardados en la tabla de la aplicación de escritorio y en la base de datos

Tag	Fecha de ordeño	turno	Producción (ltrs)	Observaciones
1613	2022-07-26	Segundo Turno	11.1	Obtenidos desde el sistema
1625	2022-07-26	Segundo Turno	11.1	Obtenidos desde el sistema
244	2022-07-26	Segundo Turno	6.61	Obtenidos desde el sistema
177	2022-07-26	Segundo Turno	7.56	Obtenidos desde el sistema

(a)

tag	fecha_ordeno	turno	litros	observaciones
1613	2022-07-26	Segundo Turno	11.1	Obtenidos desde el sistema
1625	2022-07-26	Segundo Turno	11.1	Obtenidos desde el sistema
244	2022-07-26	Segundo Turno	6.61	Obtenidos desde el sistema
177	2022-07-26	Segundo Turno	7.56	Obtenidos desde el sistema
130	2022-07-26	Segundo Turno	11.4	Obtenidos desde el sistema
227	2022-07-26	Segundo Turno	6.61	Obtenidos desde el sistema
7	2022-07-26	Segundo Turno	11.1	Obtenidos desde el sistema
174	2022-07-26	Segundo Turno	11.5	Obtenidos desde el sistema
89	2022-07-26	Segundo Turno	10.3	Obtenidos desde el sistema
659	2022-07-26	Segundo Turno	8.69	Obtenidos desde el sistema
136	2022-07-26	Segundo Turno	8.75	Obtenidos desde el sistema
246	2022-07-26	Segundo Turno	7.82	Obtenidos desde el sistema
137	2022-07-26	Segundo Turno	9.8	Obtenidos desde el sistema

(b)

Nota. a) Datos guardados en la tabla de la aplicación de escritorio, b) Datos guardados en la base de datos

Prueba de funcionalidad de la aplicación de escritorio

El sistema de gestión almacena todos los valores de producción a lo largo del tiempo. Estos registros se almacenan en la base de datos para crear una gráfica y estadística de producción individual y de la hacienda en general. En la Figura 154 se muestra un ejemplo de la información del animal, donde se puede visualizar información general del animal, información de enfermedades y medicamentos, y la información de producción. En cuanto a la producción se pude filtrar por fecha o por parto. En este ejemplo solo se muestran los datos de la producción desde el mes de abril hasta julio, por lo que se puede filtrar los datos en ese rango de fecha. Adicionalmente se puede visualizar la gráfica de la producción mediante el botón “Ver gráficas de producción” que se muestra en la Figura 154.

Figura 154

Ejemplo de consulta de una vaca en producción con ID 270

Como resultado de la gráfica para un filtro desde el 08/04/2022 hasta el 16/04/2022 se tiene la gráfica de la Figura 155. Donde el eje x representa el número de turnos de ordeño y el eje y la producción en litros por turno de ordeño.

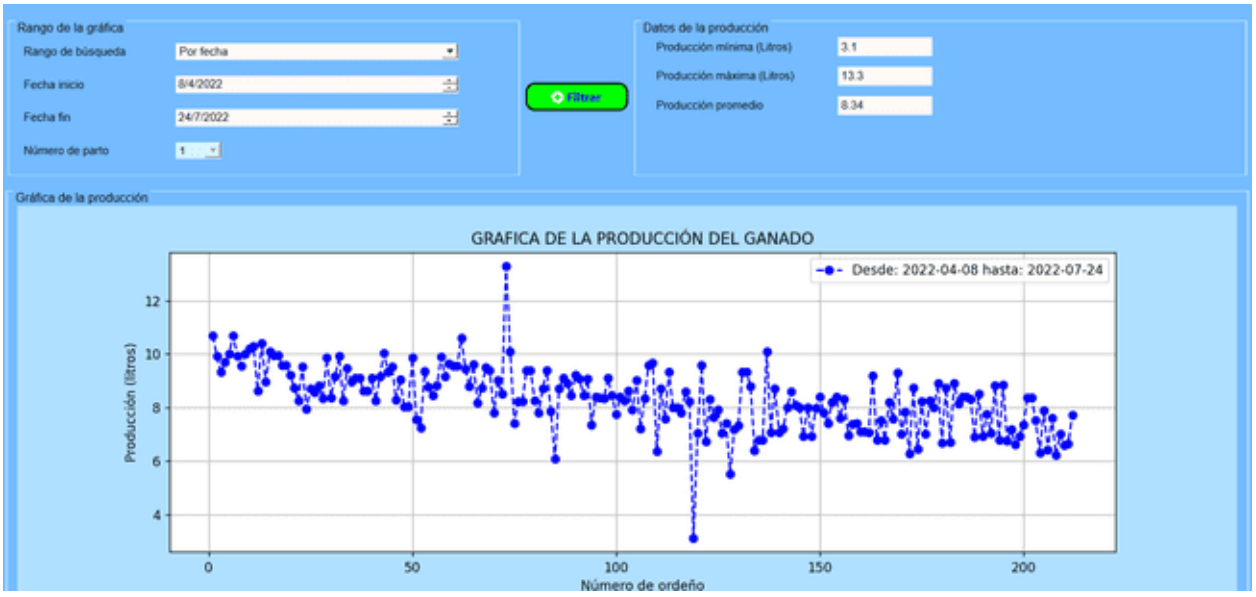
Figura 155

Curva de producción de la vaca con ID 270 desde el 08/04/2022 hasta el 16/04/2022

En la Figura 156 se muestra la producción de la vaca con ID 270 en los últimos 4 meses de ordeño, como se esperaba la producción decrece conforme pasa el tiempo hasta llegar a la etapa de “vaca seca”. La aplicación también indica los valores máximos y mínimos producidos de cada animal y el promedio de litros por turno de ordeño.

Figura 156

Curva de producción de la vaca con ID 270 desde el 08/04/2022 hasta el 24/07/2022



De esta forma verifica que el sistema funciona según los objetivos planteados al inicio del informe. Se tiene un sistema que permite realizar un registro de las actividades pecuarias

mediante una aplicación de escritorio, un sistema automático y eficiente que permite identificar al ganado y un sistema para determinar la producción de leche de cada vaca de la hacienda Bellavista. La integración de los tres sistemas permite realizar un registro automático de la producción del ganado por cada turno de ordeño y cargar en una base de datos local.

Capítulo 5: Conclusiones, recomendaciones y trabajos futuros

Conclusiones

Se diseñó e implementó un sistema para la identificación automática del ganado utilizando tecnología inalámbrica basada en la identificación por radiofrecuencia. La correlación entre la identificación del animal y el sistema de gestión de la información es posible siempre y cuando las etiquetas hayan sido registradas en la base de datos del sistema. Para evitar errores de interferencias entre los tags RFID debido a las aglomeraciones del ganado en la entrada de la sala de ordeño se construyó un embudo que obliga a las vacas entrar uno a la vez. La identificación se logró sin errores debido a que todo el tiempo se verifica si el tag(etiqueta) detectado es parte de las etiquetas existentes de la hacienda.

Se diseñó e implementó un sistema que permite obtener la medición de la producción de leche individual de la vaca utilizando los medidores de flujo de leche Fi7 del fabricante Delaval, que ya estaban instalados en la sala de ordeño de la hacienda. Considerando que el fabricante no libera el protocolo propietario de comunicación de los transmisores de los medidores, el presente proyecto resolvió la medición automática mediante la implementación de reconocimiento de imágenes de los indicadores de los medidores de flujo de leche utilizando para ello lenguaje de programación Python y librerías OpenCV. Los resultados fueron muy buenos, con errores de no más de 10 mililitros, que para el total de producción de una vaca no es representativo.

Se desarrolló una aplicación de escritorio con una interfaz de usuario amigable e interactiva que permite realizar el registro de las actividades pecuarias de la hacienda, incluido la interconexión con los sistemas de medición de la producción e identificación del ganado. Los registros que permite realizar la aplicación son:

- Registro de la información básica del ganado
- Registro de la producción

- Registro de partos
- Registro de muertes y ventas
- Registro de montas e inseminación
- Registro de enfermedades y medicamentos
- Registro de potreros y pastoreo

Con estos registros se realizó una interfaz que permite ver la información completa de un animal y mostrar el historial de la producción tabulada y mediante gráficas. Además del registro de actividades pecuarias se realizó un registro de usuarios con un control de acceso, la misma que permite habilitar las funcionalidades de acuerdo al tipo de cargo del usuario que ingresa a la aplicación.

En conjunto se desarrolló un sistema eficiente y con un costo de inversión accesible para pequeños y medianos productores. Los costos de inversión se detallan en el Apéndice A.

Recomendaciones

Para el sistema de identificación se recomienda evitar que ingresen más de seis vacas por lado, ya que puede producir errores en la asignación de los puestos y en consecuencia un mal registro de la producción.

Debido a que el inicio del proceso de registro de la producción inicia con la detección de un animal en la entrada de la sala de ordeño se debe evitar que tags RFID sueltos pasen por el área de detección de los lectores, debido a que estos están funcionando en todo momento.

Antes de iniciar con el proceso de ordeño los operarios deben realizar una limpieza de las pantallas de los medidores con una tela u otro material suave que no raye la pantalla, para evitar que la suciedad genere errores en el proceso de reconocimiento de imágenes, además se debe evitar mover las cámaras.

Para obtener los mejores beneficios del sistema se recomienda cargar los datos a la aplicación de escritorio por cada turno de ordeño, así se tendrá un registro con las fechas y horas exactas, además de esta forma se verifica si la comunicación está funcionando en todo momento.

Es aconsejable crear un protocolo para el proceso de ordeño tomando en cuenta las siguientes consideraciones:

- Ingresar no más ni menos de seis vacas por lado de ordeño, salvo el caso del último grupo.
- En el caso de que una vaca tenga mastitis u otra enfermedad que afecte a la calidad de la leche, se debe ordeñar, pero separar la leche del tanque reservorio. Esta situación es muy improbable que suceda, pero se debe considerar.
- En el caso de que una vaca extra haya pasado por el área de detección del lector RFID se debe extraer de la sala, de esta manera le sistema eliminará a dicho animal del registro.

Trabajos futuros

Este proyecto es la primera versión de un sistema electrónico de gestión ganadera realizado en la de la Hacienda Bellavista, por lo tanto, puede estar sometida a varios cambios, a continuación, se detallan algunas propuestas de mejoras en eficiencia y beneficios en cuanto a costos.

En esta primera versión del proyecto se empleó una SBC para controlar los buses de comunicación de datos entre los sistemas. La SBC puede ser reemplazada por un microcontrolador, que a su vez cuente con un sistema para el almacenamiento de datos de ordeño. Esto haría al sistema más robusto y con un mejor rendimiento a largo plazo, además

reduce el precio de manera considerable, pero el desarrollo de la lógica de programación tiene mayor grado de complejidad.

Para brindar mayores beneficios al administrador o dueño de la hacienda se podría crear una aplicación Web, de esta manera el usuario tendrá un acceso a los datos de registros de la hacienda de forma remota.

Para minimizar los costos de implementación en otra hacienda ganadera se recomienda realizar una exploración de medidores de flujo que permitan establecer una comunicación con protocolos estándar de comunicación para la obtención de la medida de leche ordeñada.

Referencias

Alvarado, T., Vargas, J., & Vargas, A. (2019). Prácticas de manejo de ordeño, acopio y su importancia en la calidad de la leche, Matahuasi, Concepción y Apata, Junín (Perú) . *Canales Científicos*, *80*(1), 225–239. https://dialnet.unirioja.es/servlet/articulo?codigo=7317401

Arias, M., Pesantez, J., Lammoglia, M., Rentería, I., & Marini, P. (2018). Impacto de la condición corporal sobre la fertilidad en vacas de la provincia de Pastaza- Ecuador. *Revista Biológico Agropecuaria Tuxpan*, *6*(1), 51–60. https://doi.org/10.47808/revistabioagro.v6i1.137

Baiza, C. (2020). *Sistema de detección y alerta del estado de somnolencia de conductores mediante visión artificial* [Universidad Israel]. http://repositorio.uisrael.edu.ec/handle/47000/2441

Batallas, C. (2020). El sistema de pastoreo intensivo en la alimentación de vacas lecheras. *Revista Ecuatoriana de Ciencia Animal*, *3*(3), 14–23. http://www.revistaecuatorianadecienciaanimal.com/index.php/RECA/article/view/174

Bonifaz, N., & Requelme, N. (2011). Buenas prácticas de ordeño y la calidad higiénica de la leche en el Ecuador. *La Granja*, *14*(2), 45–57. https://doi.org/10.17163/lgr.n14.2011.04

Bonilla, J., & Galárraga, E. (2009). *Diseño e implementación de un sistema prototipo para la identificación animal en la especie bovina con RFID (Radio Frecuency Identification)* [QUITO/ EPN/ 2009]. http://bibdigital.epn.edu.ec/handle/15000/1489

Brady, H. (2017, November 9). *Los drones, utilizados cada vez más en el mundo del pastoreo*. National Geographic. https://www.nationalgeographic.es/animales/2017/08/los-drones-utilizados-cada-vez-mas-en-el-mundo-del-pastoreo

Bustos, G. (2022, May 27). *¿Qué Es MySQL? Explicación Detallada Para Principiantes*. Hostinger Tutoriales. https://www.hostinger.es/tutoriales/que-es-mysql

Caballero, C., & Zambrano, R. (2018). Diseño e implementación de una red modbus/rtu entre

dos autómatas programables S7-1200 basados en el estándar RS485. *Universidad Politécnica Salesiana* . https://dspace.ups.edu.ec/handle/123456789/16357

Cachumba, G. (2019). Implementación de un prototipo para el control automático de nivel de agua para tanques de almacenamiento con interfaz HMI. *Universidad Israel*. http://repositorio.uisrael.edu.ec/handle/47000/1939

Caiza, J. (2020). *Evaluación genética de la eficiencia en la producción de leche de dos hatos en las parroquias de Guaytacama y San Buenaventura.* [Ecuador, Latacunga: Universidad Técnica de Cotopaxi (UTC)]. http://repositorio.utc.edu.ec/handle/27000/7009

Carbonnelle, P. (2022a). *PYPL PopularitY of Programming Language index*. https://pypl.github.io/PYPL.html

Carbonnelle, P. (2022b). *TOP IDE index*. https://pypl.github.io/IDE.html

Carriel, G., Coello, C., & Millán, M. (2016). *Telecontrol de pesaje y ordeño mecánico de ganado vacuno*. http://www.dspace.espol.edu.ec/xmlui/handle/123456789/31165

Challenger, I., Díaz, Y., & Becerra, R. (2014). El lenguaje de programación Python. *Ciencias Holguín*, *20*(2), 1–13. https://www.redalyc.org/articulo.oa?id=181531232001

Chilpe, M., & Chuma, J. (2015). *Parámetros productivos, reproductivos, manejo y sanidad en ganado lechero de las parroquias Tarqui, Cumbe y Victoria de Portete*. http://dspace.ucuenca.edu.ec/handle/123456789/21435

Contexto ganadero. (2017, May 11). *¿Sabe usted qué tipos de medidores de leche existen en el mercado? | CONtexto ganadero | Noticias principales sobre ganadería y agricultura en Colombia*. https://www.contextoganadero.com/ganaderia-sostenible/sabe-usted-que-tipos-de-medidores-de-leche-existen-en-el-mercado

DeLaval. (2021). *Indicadores de flujo y medidores de leche - DeLaval*. https://www.delaval.com/es-ec/our-solutions/milking/at-the-milking-point/milk-meters/

Estrada, J. (2018). Protocolos de comunicaciones industriales . *Logicbus*. https://www.logicbus.com.mx/pdf/articulos/Protocolos-de-Comunicación-Industrial

FAO. (2019). *Organización de las Naciones Unidas para la Alimentación y la Agricultura: Primer Curso Ecuatoriano Colombiano de Ganadería Climáticamente Inteligente | FAO en Ecuador | Organización de las Naciones Unidas para la Alimentación y la Agricultura: Primer Curso Ecuatoriano Colombiano de Ganadería Climáticamente Inteligente. FAO en Ecuador*. https://www.fao.org/ecuador/noticias/detail-events/es/c/1250954/

Felmer, R., Chávez, R., Catrileo, A., & Rojas, C. (2006). Tecnologías actuales y emergentes para la identificación animal y su aplicación en la trazabilidad animal. *Archivos de Medicina Veterinaria*, *38*(3), 197–206. https://doi.org/10.4067/S0301-732X2006000300002

Fletcher, A., & Mura, C. (2019). Ten quick tips for using a Raspberry Pi. *PLOS Computational Biology*, *15*(5), e1006959. https://doi.org/10.1371/JOURNAL.PCBI.1006959

Frías, M. (2020). *Gestión de hatos bovinos lecheros para la prevención de enfermedades zoonóticas en la parroquia Quimiag del cantón Riobamba.* [Universidad Católica de Santiago de Guayaquil]. http://repositorio.ucsg.edu.ec/handle/3317/14705

García, B. (2021). *Sistema de monitoreo de variables eléctricas en tiempo real* [Universidad de Antioquia]. http://hdl.handle.net/10495/18577

García, J. (2017). Python como primer lenguaje de programación textual en la Enseñanza Secundaria. *Education in the Knowledge Society*, *18*, 147–162. https://doi.org/10.14201/eks2017182147162

Gavilanes, A. (2013). *Generación de un sistema de trazabilidad de la leche del ganado bovino de la hacienda "El yagual de cananvalle", mediante identificación electrónica por bolo ruminal* [Universidad Politécnica Salesiana]. http://dspace.ups.edu.ec/handle/123456789/10851

González, K. (2018, January 31). *Ordeño Mecánico*. https://zoovetesmipasion.com/ganaderia/ordeno-mecanico/

González, L. (2021, June 1). *¿Qué es TensorFlow? ¿Cómo funciona?* AprendeIA. https://aprendeia.com/que-es-tensorflow-como-funciona/

Guagala, R. (2019). *Prevalencia de parásitos gastrointestinales en bovinos en producción de leche del cantón Urcuquí* [Pontificia Universidad Católica del Ecuador Sede Ibarra]. https://dspace.pucesi.edu.ec/handle/11010/420

Guamán, M. (2017). *Reactivación socioeconómica del sector ganadero en la localidad del capricho cantón Carlos Julio Arosemena Tola Provincia de Napo 2016- 2017*. https://dspace.uniandes.edu.ec/handle/123456789/7477

Guevara, J. (2019, June 29). Industria láctea mueve $ 1.400 millones en el año. *El Telégrafo*. https://www.eltelegrafo.com.ec/noticias/economia/4/industria-lactea-ingresos-ecuador

Hernández, M. (2020, September 10). *Sistemas de ordeño en rumiantes* . https://www.veterinariadigital.com/articulos/sistemas-de-ordeno-en-rumiantes/

Herrmann, M. (2020). *Descarga de Qt Designer para Windows y Mac*. https://build-system.fman.io/qt-designer-download

Hidalgo, Y., García, M., Gutiérrez, G., & Chagray, N. (2021). Tendencia genética y fenotípica de la producción de leche: caso de un establo comercial del valle de Huaura, Perú. *Ciencia y Tecnología Agropecuaria*, *22*(1). https://doi.org/10.21930/rcta.vol22_num1_art:1892

Huayta, A. (2018). *Diseño de un sistema de monitoreo de parámetros eléctricos utilizando tecnología GSM y Modbus para cajeros automáticos del BBVA Continental* [Universidad Nacional Tecnológica de Lima Sur]. http://repositorio.untels.edu.pe//handle/123456789/537

INEC. (2021). Encuesta de Superficie y Producción Agropecuaria Continua 2020. *Ecuadorencifras*. https://www.ecuadorencifras.gob.ec/documentos/webinec/Estadisticas_agropecuarias/espac/espac-2020/Presentacion ESPAC 2020.pdf

Johnston, J., Basford, J., Perkins, S., Herry, H., Tso, F., Pezaros, D., Mullins, D., Yoneki, E., Cox, S., & Singer, J. (2018). Commodity single board computer clusters and their applications. *Future Generation Computer Systems*, *89*, 201–212. https://doi.org/10.1016/J.FUTURE.2018.06.048

Júnez, C. (2011). *Algoritmos para reducción de ruido Gaussiano y sal y pimienta en imágenes digitales* [Universidad Michoacana de San Nicolás de Hidalgo]. http://bibliotecavirtual.dgb.umich.mx:8083/xmlui/handle/DGB_UMICH/3338

Junquera, P. (2015). *Diseño y desarrollo de una aplicación de escritorio dedicada a la composición fotográfica* [Universidad de Valladolid]. https://uvadoc.uva.es/handle/10324/15170

Lascano, P., Guevara, R., Guevara, G., Narváez, J., Arcos, C., Torres, C., Arcos, F., Garzón, R., & Beltrán, C. (2020). Ciclo de vida de granjas lecheras pequeñas familiares y diversificadas en la zona andina de Ecuador. *Revista Ecuatoriana de Ciencia Animal*, *4*(3), 8–17. http://www.revistaecuatorianadecienciaanimal.com/index.php/RECA/article/view/221

LibreComputer. (2021). *AML-S905X-CC (La patata)*. https://libre.computer/products/s905x/

MAG. (2018a). *Acuerdo Interinstitucional N^{o} 36. Ministerio de Agricultura y Ganadería*. https://www.agricultura.gob.ec/wp-content/uploads/2018/03/Acuerdo-Interinstitucional-No.-036.pdf

MAG. (2018b). *Categorías de Población de Ganado Bovino de Ecuador.* . Ministerio de Agricultura y Ganadería. https://www.agricultura.gob.ec/wp-content/uploads/2019/09/ANEXO-1.pdf

MAG. (2021). *Plan Nacional Agropecuario. Ministerio de Agricultura y Ganadería y GA*. https://www.rimisp.org/wp-content/uploads/2021/04/Plan-Nacional-Agropecuario-MAG-2021.pdf

MAGAP. (2013a). *Acuerdo Ministerial 2013 - 001. Ministerio de Agricultura, Ganadería, Acuacultura y Pesca*. https://vlex.ec/vid/expa-control-cadena-leche-derivados-434478898

MAGAP. (2013b). *Acuerdo Ministerial N^{o}394. Ministerio de Agricultura, Ganadería, Acuacultura y Pesca*. https://www.eltelegrafo.com.ec/images/eltelegrafo/banners/2013/17-09-13-Acuerdo_394_Leche.pdf

Marcillo, D., & Toala, J. (2021). *ANÁLISIS ECONÓMICO DE LOS SUPLEMENTOS EN LA*

ALIMENTACIÓN BOVINA CON FORRAJE VERDE EN EL SITIO LA ALEGRÍA, CANTÓN SANTA ANA [Jipijapa.UNESUM]. http://repositorio.unesum.edu.ec/handle/53000/2794

Marín, R. (2020, February 12). *OpenCV, Instalación en Python y ejemplos básicos*. Inesem Business School. https://www.inesem.es/revistadigital/informatica-y-tics/opencv/

Márquez, J. (2020). *Desarrollo e implementación de una red de sensores para monitorización basada en ZigBee* [Universidad Loyola]. https://repositorio.uloyola.es/handle/20.500.12412/2260

Martínez, A. (2007, April). Factores nutricionales que afectan a la composición de la leche - Engormix. *Lechería*. https://www.engormix.com/ganaderia-leche/articulos/factoresnutricionales-afectan-composicion-t27057.htm

Martínez, M. (2019). *Los mejores editores de texto para desarrolladores y diseñadores*. https://profile.es/blog/mejores-editores-texto/

Mazziotti, P. (2019, March). *La robótica en el campo: sistemas de ordeño*. https://www.editores-srl.com.ar/revistas/aa/11/mazziotti_sistemas_de_ordeno

Mejía, J. (2016). Desarrollo de una Red Modbus RTU con Raspberry. *Instituto Tecnológico Metropolitano*. http://hdl.handle.net/20.500.12622/4014

Méndez, S., Soria, M., Palacios, E., Andrade, O., Bustamante, J., Pesantez, J., Vásquez, J., Guevara, G., & Vallecillo, A. (2020). Parámetros de variabilidad genética de bovinos certificados de la raza Holstein del cantón Cuenca, Ecuador. *Chilean Journal of Agricultural & Animal Sciences*, *36*(1), 63–68. http://dx.doi.org/10.29393/chjaas36-3d40003

Murguía, A., & Castillo, G. (2012, September 4). *Vida productiva de vaca lechera*. https://www.engormix.com/ganaderia-leche/articulos/vida-productiva-vaca-lechera-t29690.htm

Negrete, E., & Hernández, L. (2018). *Diseñar e implementar un sistema de identificación y trazabilidad de ganado bovino para la administración de la finca notecebes del departamento de córdoba* [Facultad de Ingeniería].

https://repositorio.unicordoba.edu.co/handle/ucordoba/504

OCDE/FAO. (2019). *OCDE-FAO Perspectivas Agrícolas 2019-2028, OECD Publishing, París/Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO), Roma*. https://doi.org/10.1787/7B2E8BA3-ES

Oracle. (2022). *What Is a Database?* https://www.oracle.com/database/what-is-database/#relational

PeruLactea. (2018, February 22). *Diferentes Tipos de Salas de Ordeño en Bovinos*. http://www.perulactea.com/2018/02/22/diferentes-tipos-de-salas-de-ordeno-en-bovinos/

Ponce, P. (2015). *Diseño e implementación de un prototipo de telecontrol de sistema de identificación RFID, ordeño, alimentación y conservación de leche de ganado vacuno con interfaz web mediante uso de hardware y software libre*. https://www.dspace.espol.edu.ec/retrieve/94401/D-103073.pdf

PRONACA. (2020). *Importancia de manejo de registros ganaderos*. https://www.procampo.com.ec/index.php/blog/10-nutricion/101-importancia-de-manejo-de-registros-ganaderos

QTCompany. (2022). *Documentación Qt* . https://doc.qt.io/

RecluIT. (2021, April 12). *¿Qué es Visual Studio Code?* https://recluit.com/que-es-visual-studio-code/#.YqhJQXbMLIU

Roca, J. (2021, May 9). *Los SBC más potentes del mercado*. HardZone. https://hardzone.es/reportajes/comparativas/sbc-x86/

Rodríguez, H. (2021, April 27). *¿Qué es OpenCV y para qué sirve?* . Crehana. https://www.crehana.com/blog/desarrollo-web/que-es-opencv/

Rojo, A. (2016). *MANUAL DE USUARIO ODROID-XU4*. https://docplayer.es/18152996-Manual-de-usuario-odroid-xu4.html

Sánchez, A., Vayas, T., Mayorga, F., & Freire, C. (2019). Sector ganadero, un análisis de la producción ganadera en Ecuador. *CEDIA*. https://blogs.cedia.org.ec/obest/wp-

content/uploads/sites/7/2020/06/SECTOR-GANADERO-FINAL.pdf

Sánchez, M. (2002). *Producción Animal e Higiene Veterinaria (Grupo A)*. http://www.uco.es/zootecniaygestion/img/pictorex/16_20_02_tema_9chico2.pdf

Shenzhen, S. (2018). *Comparación de rendimiento entre RS-232 y RS-485*. https://www.serial-cable.com/info/performance-comparison-between-rs-232-and-30244518.html

Software Ganadero. (2021). *Software Ganadero*. https://www.softwareganadero.com/

Solis, J. (2022, May 12). *¿Cómo entrenar Tesseract OCR en Python?* ProjectPro. https://www.projectpro.io/article/how-to-train-tesseract-ocr-python/561

Suhissa. (2017). *Flujómetro: tipos de medidores de flujo y aplicaciones (parte 1) - .* https://suhissa.com.mx/flujometro-tipos-de-medidores-de-flujo-y-aplicaciones-parte-1/

Torres, S. (2021). *Implementación de un sistema HMI mediante aplicaciones de código abierto para el control y monitoreo de un sistema dinámico real*. http://repositorio.utn.edu.ec/handle/123456789/11651

Torres, X. (2018). *Estudio de la producción de la Industria Láctea del cantón Cayambe en el período 2009-2015*. https://repositorio.uasb.edu.ec/bitstream/10644/6052/1/T2544-MAE-Torres-

VisualStudioCode. (2022). *Documentación para el código de Visual Studio Code*. https://code.visualstudio.com/

Weis, O. (2020a). *Diferencia entre RS232 y RS485*. Electronic Team Publishing. https://www.virtual-serial-port.org/es/article/what-is-serial-port/rs232-vs-rs485.html

Weis, O. (2020b). *Modbus vs RS485* . Electronic Team Publishing. https://www.virtual-serial-port.org/es/articles/modbus-vs-rs485/

Weis, O. (2021). *Qué es RS485 - Guía de la Comunicación RS485* . https://www.eltima.com/es/article/rs485-communication-guide/

Yanzeo. (2022). *UHF RFID Reader, Yanzeo SR681*. Yanzeo y RFID. https://www.yanzeorfid.com/

Yegulalp, S. (2022, June 3). *¿Qué es TensorFlow? La biblioteca de aprendizaje automático explicada*. InfoWorld. https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html

Zelic, F., & Sable, A. (2022, May 10). *Tesseract OCR en Python con Pytesseract y OpenCV*. Nanonets. https://nanonets.com/blog/ocr-with-tesseract/#tesseract-ocr

Zulovich, J., Milhollin, R., Harner, J., & Horner, J. (2018). Automated Milking Systems for Dairy Operations. *American Society of Agricultural and Biological Engineers*. https://doi.org/10.13031/ILES.18-112

Apéndices