ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"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."

INFORME DE PROYECTO INTEGRADOR

Previa a la obtención del Título de:

INGENIERO EN TELEMÁTICA

PUBLIO EDUARDO PONCE PALMA

GUAYAQUIL – ECUADOR

Año: 2015

AGRADECIMIENTO

A Dios por bendecirme con la familia que tengo.

A mis padres por la educación y por inculcarme buenos valores, además de todo el cariño, apoyo incondicional y confianza brindada durante toda mi carrera.

A mi esposa por demostrarme su amor y contagiarme de sus ganas de salir adelante y crecer profesionalmente.

Al ingeniero Marcos Millán por compartir sus conocimientos y ayuda para culminar esta materia con éxito.

Publio Eduardo Ponce Palma

DEDICATORIA

A Dios y a mi familia por todo el apoyo y confianza, que me han dado fuerzas para culminar con éxito mi objetivo.

Publio Eduardo Ponce Palma

TRIBUNAL DE EVALUACIÓN

Ing. Marcos Millán	**Ing. José Menéndez**
PROFESOR EVALUADOR	PROFESOR EVALUADOR

DEDICATORIA EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, me corresponde exclusivamente; y doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual".

_____

Publio Eduardo Ponce Palma

RESUMEN

En el presente trabajo, se muestra el diseño e implementación de un sistema de telecontrol y telemetría de lo que involucra la extracción, conservación de la leche y alimentación del ganado vacuno, que nos va a permitir controlar y administrar la ganadería de una granja. Para la visualización del sistema, se ha desarrollado una interfaz gráfica a través de un servidor web basado en software libre que permite mostrar el status de la ganadería en tiempo real.

El proyecto fue desarrollado utilizando un dispositivo *SBC (Single Board Computer)* Beaglebone de la Fundación BeagleBoard.org. Para esto, nos basamos en herramientas proporcionadas por la comunidad de software libre como *Debían, Php (PHP Hypertext Preprocessor), Apache Tomcat, Python, Mysql, Java* usados para el desarrollo del Hardware y Software del sistema.

De la amplia familia de las Beaglebone el proyecto fue desarrollado usando una Beaglebone Black. El sistema se ha divido en etapas, en la cuales se encuentra la adquisición, procesamiento, control, actuador y visualización de la señales recibidas. El proyecto se lo ha estructurado en 4 capítulos que se detallan a continuación:

En el capítulo 1, se describe los objetivos generales y específicos del proyecto, así como también los alcances y limitaciones.

En el capítulo 2, se presenta el fundamento teórico del sistema ROAC (Rfid, ordeño, alimentación y conservación) y la forma de funcionamiento, además de detallar cada elemento de la tecnología utilizada para el desarrollo del proyecto.

En el capítulo 3, se muestra el diseño e implementación del proyecto, donde se describe con detalle cada etapa del sistema, además de la programación que se realizó para la adquisición y procesamiento de los sensores.

En el capítulo 4, se presenta los resultados obtenidos de las diferentes pruebas realizadas al sistema.

Finalmente, las conclusiones y recomendaciones del proyecto.

ÍNDICE GENERAL

CAPÍTULO 1

1. GENERALIDADES

1.1 OBJETIVOS

1.1.1 OBJETIVOS GENERALES

El objetivo principal del proyecto es diseñar e implementar un sistema de ROAC para poder depurar, mejorar y optimizar la administración del ganado.

1.1.2 OBJETIVOS ESPECÍFICOS

- Implementar un sistema ROAC capaz de analizar y resolver el manejo del ganado.
- Desarrollar a través de Php una aplicación web para el usuario.
- Crear un código en Python para la adquisición y procesamiento de datos.
- Diseñar la arquitectura de la mini-computadora basado en Debían.

1.2 ALCANCES

Entre los alcances de proyecto se tiene:

- Adquirir los datos del ordeño a través de uno de los móviles.
- Analizar varias variables simultáneamente ya que posee múltiples funciones.
- Los sensores podrán ser visualizadas como estados de actividad.
- La aplicación fue desarrollada en varios lenguajes de programación de. Que permiten visualizar el sistema a través de una página web.

1.3 LIMITACIONES

Entre las limitaciones del proyecto tenemos:

- El sistema ROAC está diseñado solo para el uso de una ordeñadora con dos vacas.
- El sistema no se puede integrar con otro mismo dispositivo.

Las pruebas se realizaron dentro del ambiente de laboratorio.

CAPÍTULO 2

2. MARCO TEÓRICO

2.1 PROBLEMÁTICA

Actualmente el sector ganadero carece de un plan estratégico y de desarrollo tecnológico para la recolección de leche de su ganado vacuno, lo que ocasiona que no se optimice la cantidad de leche que podrían producir con tecnología de punta y con la implementación de técnicas actualizadas para su recolección.

A esta fecha han cambiado las políticas del país en el uso de tierras, y ahora se pretende exigir la utilización de la mayor parte de los terrenos sin producir[9], además de que los propietarios de estos terrenos tienen la dificultad de sostener una cantidad necesaria de mano de obra motivo por el cual en la actualidad se está subarrendando las fincas para evitar estos inconvenientes, de allí nace la necesidad de implementar un sistema rentable tanto para los trabajadores, el dueño del terreno y la economía del país.

2.2 CONCEPTOS BASICOS

A continuación describiremos las fases del proyecto:

Rfid.- Es la fase que permite identificar cada vaca dentro del establo siempre y cuando tenga asignada un Tag Rf, con ello podemos ingresar nuevas vacas al sistema o eliminarlas.

Los Tags son reciclables, si una vaca deja de servir para el propósito, entonces se le extrae y asigna a otra vaca ya sean nuevas o que ya ha pasado el proceso de crecimiento. Cada vaca tiene su id en la base de datos que no depende de este valor.

Ordeño.- En esta fase se procede a ejecutar el ordeño mecánico, donde el vaquero debe amarrar las patas de la vaca para poder colocar las pezoneras y continuar con la extracción de la leche para lo cual tenemos varias opciones: manual, por tiempo, por peso o a su vez una combinación de ellas.

La cantidad de leche que da la vaca depende del tipo de alimentación, clima y cuidados que se tengan pero en promedio el ganado lechero en ecuador va de 4

a 5 litros promedio, dura entre 3 a 6 min el lapso que puede ser ordeñada, más tiempo de eso podría atrofiar las mamas de la vaca. [1]

Conservación.- En esta fase vamos a almacenar la leche para su posterior venta, se exige tener un tanque refrigerado para llegar a los parámetros ideales, estos están entre 4 y 6 °C si la temperatura cae bajo los 4 °C puede sufrir congelación y si sobrepasa los 6 °C se expone a proliferación de bacterias.[3]

Alimentación.- En esta fase se da de alimentar a la vaca y es recompensada por su producción. La porción es proporcional al número de litros que da en ese día de ordeño.

ROAC

Llamamos a esto a la integración de Rfid, ordeño, alimentación y conservación para el telecontrol de la leche del ganado vacuno, esto sirve nos para tabular y llevar una correcta administración.

2.3 ROAC Y PLATAFORMA TECNOLÓGICA

SBC Y BEAGLEBONE BLACK

Un *SBC* es un ordenador de placa reducida que puede tener un amplio número de aplicaciones, cuyo objetivo es el desarrollo tecnológico a bajo costo, en el cual enfocamos nuestro proyecto es el Beaglebone Black cuyas características que destacan son, excelente precio/rendimiento, gran cantidad de cabeceras de expansión, software y hardware libre.

Características y arquitectura

Es una tarjeta de desarrollo ideal para el desarrollo y emprendimientos de proyectos de pequeña escala.

Una de las ventajas es su peso y tamaño que permite desarrollar una amplia variedad de proyectos de diseños portables y móviles. Así como también la capacidad de ser reconfigurado por el usuario de acuerdo a sus requerimientos y necesidades.

En la figura 2.1 podemos observar la tarjeta Beaglebone Black

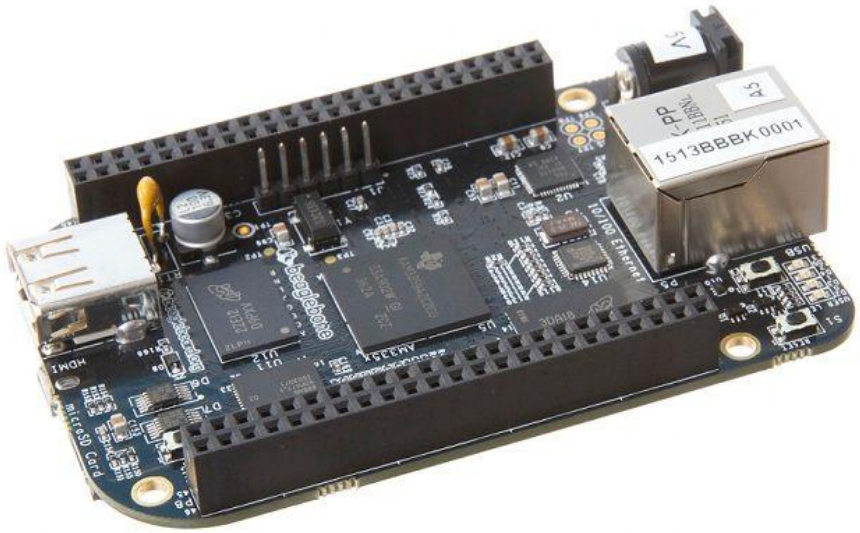

Figura 2.1: Beaglebone Black

La tarjeta mencionada posee las características descritas en la Tabla 1:

Características	
Procesador	1Ghz Sitara AM3359AZCZ100
Motor Gráfico	SGX530 3D, 20M Polygons/S
Memoria SDRAM	512 MB DDR3L 400Mhz
Flash	4GB,8bit Embedded MMC
PCB y peso	3.4” x 2.1”, 6 capas / 39.68 gramos
Indicadores	1 Poder, 2 Ethernet, 4 Leds para control del usuario
Ethernet	10/100, RJ45
SD/Conector MMC	Micro SD, 3.3V
Salida de audio y video	Hdmi
Conectores de Expansión	65 gpio, 8 pwm, 7 AI, 4.5 serial uart, 2 I2C, 2SPI
Consumo	210-460mA @ 5V Dependiendo de la actividad y velocidad del procesador.

Tabla 1: Características de la tarjeta Beaglebone Black [5]

Analizando las características principales tenemos, un procesador que es usado en las mejores marcas de Smartphone como Samsung, Apple, Nokia, etc. Viene de una familia de procesadores que en la actualidad es ampliamente

implementado en todo tipo de aparatos electrónicos inteligentes por su gran capacidad, arquitectura y precio.

Consta de un puerto de Ethernet de 10/100 Mb que en la actualidad es muy útil puesto que ahora hay mucho acceso a tecnología móvil, cada vez es más común el uso de ella y se sigue creando día a día nuevas aplicaciones.

Otra de las características principales es el número de conectores de expansión, tiene 65 gpio (*general proposite input output*), 8 pwm (*pulse-width modulation*), 7 ai (*analog input*), 4.5 serial uart (*Universal Asynchronous Receiver-Transmitter*), 2 I2C (*Inter-Integrated Circuit*), 2 SPI (*Serial Peripheral Interface*) con lo que podríamos comunicarnos con una gran cantidad de dispositivos electrónicos con este tipo de conexiones.

Adicional a todo esto, podemos instalar varias distribuciones Linux fuera de la distribución original (angstrom) y ejecutarlas con gran comodidad, tales como: Ubuntu, Android, Fedora, Debían, ArchLinux, Gentoo, Backtrack, etc.

LAMP

LAMP (Linux Apache Mysql Php) es un grupo de software de código abierto que se utiliza para poner los servidores web en marcha y funcionando El acrónimo de Linux, Apache, MySQL y Php. [4]

La distribución Linux escogida fue Debian por su compatibilidad con el Beaglebone Black y por tener buen soporte, estabilidad y confiabilidad, este es usado extensamente en servidores y estaciones de trabajo lo que lo hace uno de los sistemas operativos más potentes para el uso de servicios de red.

Apache Tomcat, entre sus ventajas son: altamente configurable, código abierto, buen soporte y multiplataforma, con todo esto lo hacen de lo más popular y recomendable.

Mysql tenemos la facilidad de elaboración de bases de datos su velocidad, seguridad y conectividad lo hacen y fácil de configurar.

Y finamente tenemos a Php, un lenguaje orientado a la programación web, es sencillo de crear una página segura y confiable.

Con todo lo dicho anteriormente tenemos un sistema seguro, confiable, rápido, fácil administración y muy soportado por la comunidad libre capaz de responder de la mejor manera a todas las necesidades.

2.4 LENGUAJES Y PLATAFORM AS DE PROGRAMACIÓN

2.3.1 PYTHON – CLOUD 9

Se ha usado este lenguaje que viene por defecto instalado en el sistema y puede usar todas las herramientas del *SBC* de una manera fácil, con lo cual podemos interactuar con todos los sensores desde que están en contacto con el hardware.

Puede ser editado desde la web con el IDE (*Integrated Development Environment*) Cloud 9 lo que nos ahorra el tiempo de cargar los datos al sistema y poder programar más rápido y eficientemente.

2.3.2 PHP

Se escogió este lenguaje por que lo hace más práctico de programar y se puede trabajar para la transición de datos con Python desde los sensores hasta la Apk de Android para el ordeño.

2.3.3 JAVA – ANDROID STUDIO

Basado en java se trabajó en el sistema de adquisición de datos RF y envió de órdenes de ordeño he instalado en un móvil con sistema operativo Android lo que lo hace portable.

2.5 MÓDULOS Y SENSORES

SENSOR DE TEMPERATURA

El LM35 es un sensor de temperatura. Está calibrado de fábrica con una precisión de 0.1ºC y es capaz de tomar lecturas entre -55ºC y 150ºC. En este caso no necesitamos más que controlar la temperatura del tanque de conservación que debe estar entre 4 a 6ºC y que no vamos a superar el límite de voltaje que puede recibir el módulo ADC (*analog-to-digital converter*) que es de 1.8V, se muestra la imagen de un integrado LM35 en la Figura 2.2.

Figura 2.2: Sensor de temperatura LM35

SENSORES DE PROXIMIDAD

Aquí se usó hc-sr04 que es un sensor de proximidad y podemos medir la cantidad de leche en las cantaras de leche a la hora del ordeño, tiene una precisión de 0.1L y el tanque de conservación tiene una precisión de 5L aproximadamente ya que su volumen a calcular es mucho mayor, en la Figura 2.3 tenemos la imagen frontal del sensor.

Figura 2.3: Sensor de ultrasonido HC-SR04

LECTOR RFID

Se ha usado le lector NFC embebido en el dispositivo móvil ya que nos da la movilidad de registrar la vaca a ordeñar, transmitiendo los datos vía wi-fi al servidor para identificar la vaca en cuestión. El alcance del lector es de 5cm máximo con lo que nos da oportunidad de activarlo teniendo en cuenta que la vaca está en movimiento generalmente, en la Figura 2.4 mostramos un modelo de teléfono móvil que tiene la característica mencionada anteriormente.

Figura 2.4: Dispositivo Android con NFC

ELEVADOR

Basándonos en un producto ya probado y para efectos de práctica para demostrar la auto-alimentación del sistema, se ha ubicado una compuerta de salida de comida de una tolva usando un elevador de seguros equipado en vehículos motorizados, en la Figura 2.4 tenemos el modelo de elevador usado en el proyecto.

Figura 2.5: Dispositivo Actuador

CAPÍTULO 3

3. SOLUCIÓN TECNOLÓGICA

3.1 DISEÑO DE SOFTWARE Y HARDWARE

3.1.1 SOFTWARE

En la Figura 3.1 se muestra el algoritmo del sistema.

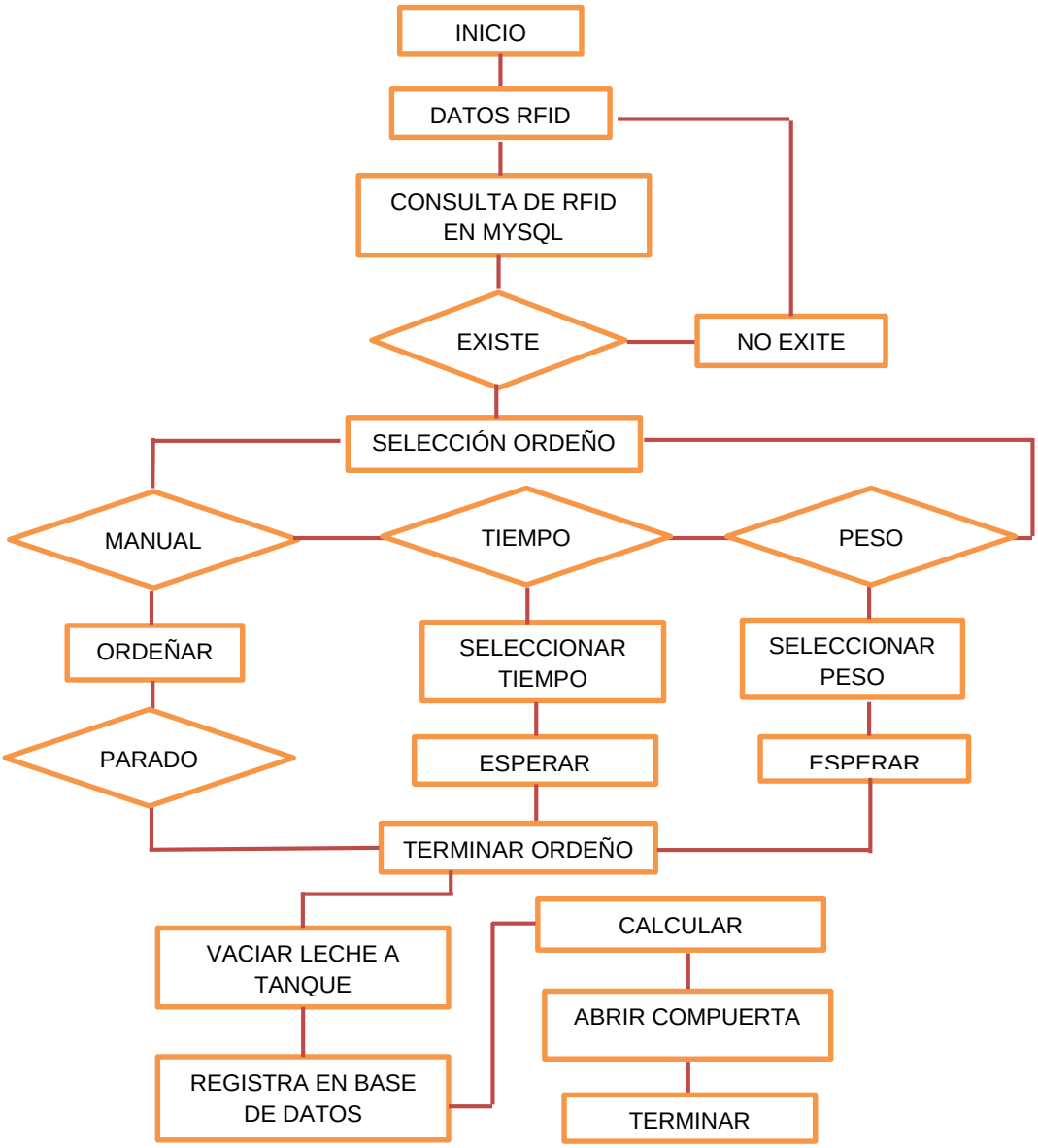

Figura 3.1: Algoritmo del sistema ROAC

Página de inicio del sistema. En la Figura 3.2 nos muestra el menú principal al lado derecho, la bienvenida del usuario en el centro y una barra de usuario en la parte superior.

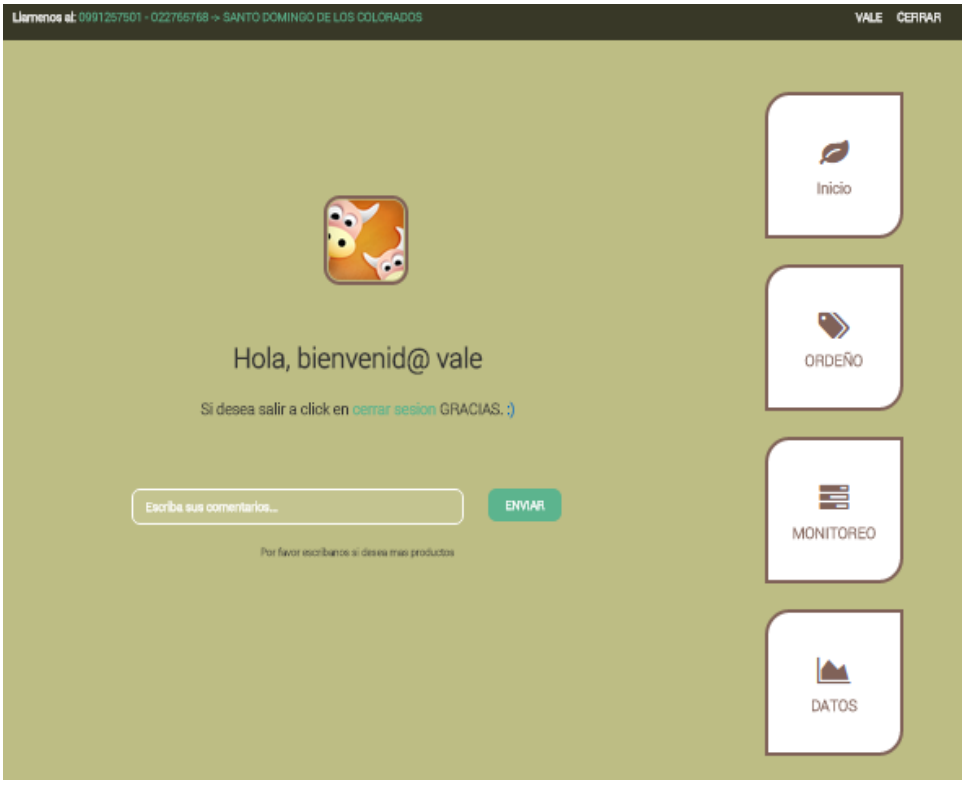

Figura 3.2: Página de inicio del sistema

En la barra superior se encuentra el menú de usuario con su nombre, opción de cerrar sesión y la información de contacto del propietario del sistema.

En la parte lateral derecha tenemos el menú principal con las opciones de: Inicio, Ordeño, Monitoreo, Datos que se describen más adelante.

En la parte central está la bienvenida del usuario, la opción de cerrar sesión y el formulario de envío de comentarios.

Página de ordeño del sistema. En la Figura 3.3 se muestra en 4 secciones para poder activar o desactivar el sistema y consulta básica de las vacas.

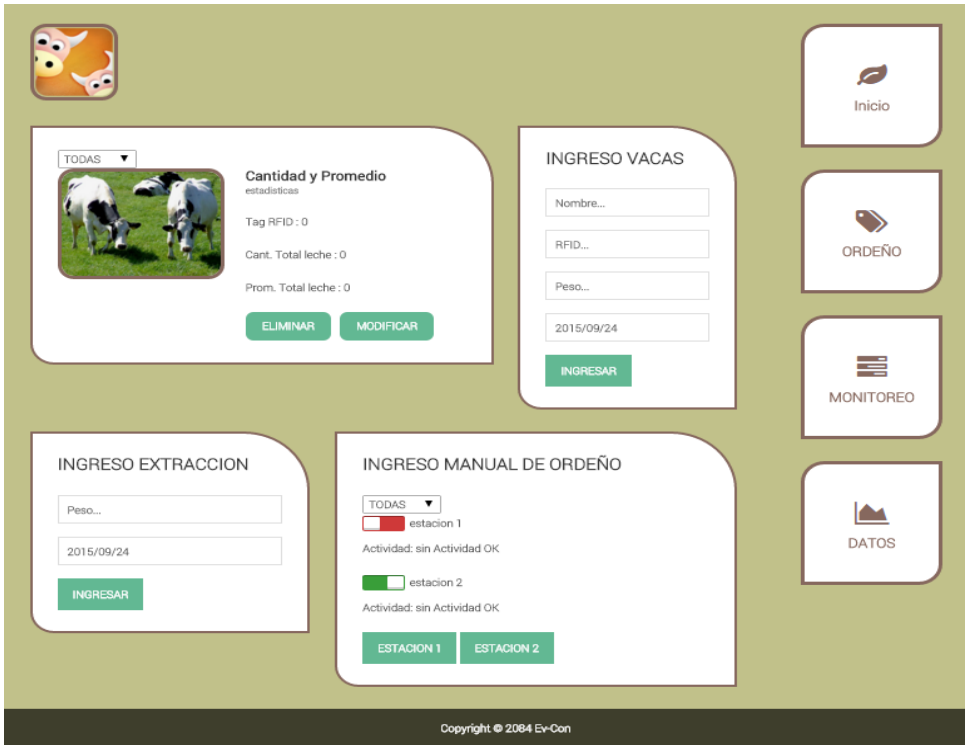

Figura 3.3: Página de ordeño del sistema

Sección 1: ubicado en la parte de superior izquierda, contiene el menú de consulta de cada vaca.

Sección 2: ubicado en la parte de superior derecha, contiene el menú de ingreso de las vacas al sistema.

Sección 3: ubicado en la parte de inferior izquierda, contiene el menú de ingreso de una extracción de leche del tanque.

Sección 4: ubicado en la parte de inferior derecha, contiene el menú de ordeño para poder enviar la orden de ordeño para cada estación.

Página de monitoreo del sistema. En la Figura 3.4 se muestra en 4 secciones para poder monitorear y visualizar el status del sistema.

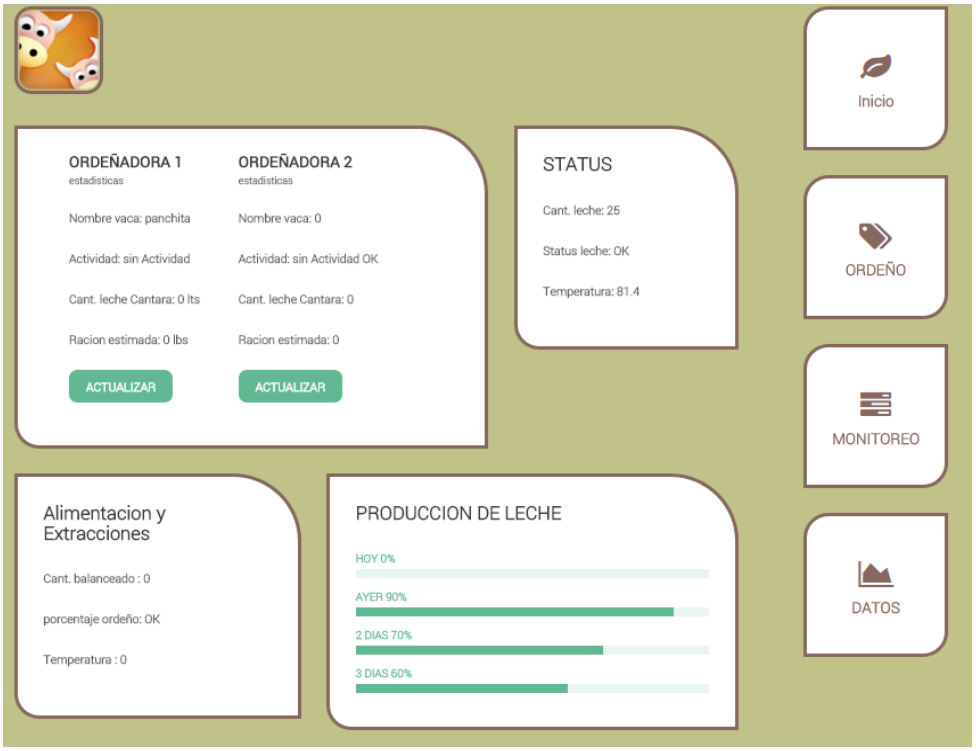

Figura 3.4: Página de monitoreo del sistema

Sección 1: ubicado en la parte de superior izquierda, contiene el menú de ordeño para poder consultar estado de cada estación, la vaca en proceso de ordeño, la actividad que se está realizando, la cantidad leche en la cantara y la cantidad de comida asignada.

Sección 2: ubicado en la parte de superior derecha, contiene el menú de estado del tanque de conservación, muestra la temperatura interna, el status y la cantidad de leche.

Sección 3: ubicado en la parte de inferior izquierda, contiene el menú de alimentación y extracciones, muestra la cantidad de balanceado asignado, porcentaje de vacas ordeñadas el día actual y la temperatura ambiente.

Sección 4: ubicado en la parte de inferior derecha, contiene el menú de la producción total diaria de los 4 últimos días.

Página de estadística del sistema. En la Figura 3.5 nos muestra en diagrama de barras la producción diaria de cada vaca.

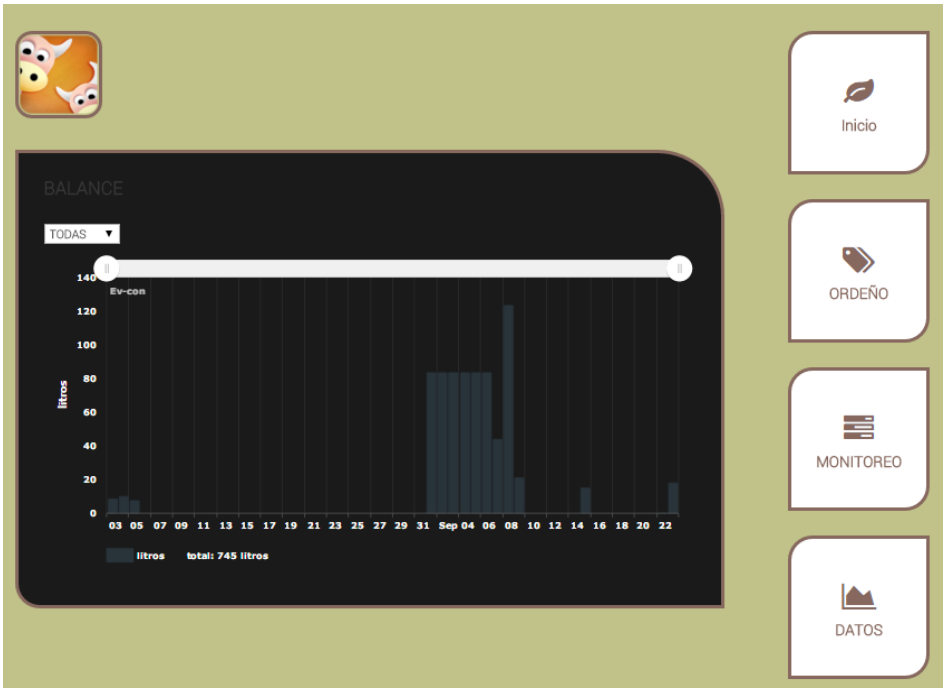

Figura 3.5: Página de estadísticas del sistema

Aquí podemos consultar las estadísticas de cada vaca o de todas, mostrándonos un día por barra y así observar la producción de leche por un periodo de tiempo dando como resultado una mejor perspectiva del rendimiento de la vaca.

Podemos escoger el tiempo de la consulta que va desde 1 día a 1 año, también nos da el total de la cantidad leche del periodo consultado.

En la figura 3.6 se muestra la interfaz de la aplicación para el manejo de la ordeñadora.

Figura 3.6: Aplicación de la ordeñadora

Los programas se realizaron en varios lenguajes. El diseño web se adapta a todo tipo de resolución y se puede visualizar desde cualquier dispositivo.

3.1.2 HARDWARE

El sistema fue introducido dentro de una caja reciclada y la tarjeta de control de sensores diseñado en un circuito impreso sin patrón y recortada para que quepa dentro de la caja junto con los elementos electrónicos ya soldados en ella. Además de una pantalla Lcd (*Display de cristal líquido*) pequeña para poder ver la dirección IP y la temperatura del tanque de conservación. Aquí se muestra en la Figura 3.7 el producto ya elaborado:

Figura 3.7: Prototipo del sistema

También cuenta con 4 botoneras para el manejo manual del ordeño ubicadas en la parte superior del producto, en su interior cuenta con relés, transistores y resistencias además de una entrada de 5v para iniciar los sensores ya que por sí misma la *SBC* no es capaz de entregar la potencia requerida para iniciar el circuito de sensores.

PASOS DE INSTALACIÓN

Preparación del servidor en Beaglebone Black

1. Se descarga la imagen .iso del sistema operativo Debían para Beaglebone desde el repositorio.
2. Se carga el sistema en un micro Sd de al menos 8GB con el programa Win32DiskImager.
3. Instalación de LAMP [4].
4. Instalación de *Phpmyadmin* con el comando: aptitude install phpmyadmin, esto sirve para una rápida configuración de la base de datos del sistema.
5. Configuración de puertos http y https, 80 y 443 respectivamente en el directorio /etc/apache2/ el archivo ports.conf para poder acceder a nuestra aplicación vía web.
6. Instalación de ftp (*file transfer protocol*) [10].
7. Creación de usuario ftp y asignación del directorio de /var/www/ para poder transferir la página web o modificaciones remotamente ya que aquí se aloja la página web.

8. Dar permisos de lectura y escritura al directorio /var/www/ con valor de 777, para que cualquier usuario pueda modificar el contenido de las páginas web.

Instalación de módulos y sensores

1. Asignación de pines a relés desde P9_11 a P9_16 los 2 primeros son para el llenado de las cantaras, los 2 siguientes son para el vaciado de las cántaras al tanque y los 2 últimos son para el sistema de conservación y la tolva respectivamente.
2. Sensor de temperatura a pin P9_39 (entrada analógica).
3. Sensores de ultrasonido en pines P8_14 y P8_16 como trigger de la primera y segunda estación respectivamente y P8_13 y P8_15 como echo de igual manera.
4. Lcd en pines P8_7 al P8_12, pines dispuestos según la configuración realizada en la programación.
5. Las botoneras en pines P8_17 al P8_19 y P9_23.

Implementación del sistema ROAC

1. Creación del script de inicialización de sensores y módulos, además de dar permisos de ejecución de los pines involucrados, así no tener inconvenientes en el arranque del sistema.
2. Ubicar el script en el directorio /etc/init.d/ para su inicio automático, aquí se encuentran todos los archivos de arranque del sistema operativo.
3. Creación de los scripts para un ordeño por medio del lector RFID y otro para el ordeño manual por botoneras a través de Cloud 9 y ubicadas en el directorio /var/lib/cloud9/.
4. Implementación de la página web basada en Php para el control y administración del sistema y cargada mediante Ftp. Además de archivos de transferencia de información entre la aplicación Android y el servidor mediante este mismo lenguaje.
5. Creación de archivos Php para la interacción con la base de datos, pagina web, aplicación Android y scripts Python.

6. Creación de Apk Android para obtener datos del RF usando el programa Android Studio, este programa está enfocado a la creación de aplicaciones Android.
7. Edición de tablas en la base de datos que son: usuarios, vacas, ordeño y extracción. Vía web usando *phpmyadmin* previamente instalado o usando comandos a través de *ssh* (Secure SHell) al servidor.

3.2 IMPLEMENTACIÓN DEL ROAC

3.2.1 ETAPA DE ADQUISICIÓN

Para captar las múltiples señales se ha usado varios sensores. A continuación se muestran las principales características por la cual se ha decidido usar estos sensores.

- ✓ Costo muy bajo
- ✓ Fácil programación
- ✓ Hardware libre

Los sensores usados básicamente son: de proximidad y temperatura de forma análoga, y un lector Rfid desde un móvil con sistema operativo Android. A continuación se describe la función de cada uno de los ya mencionados sensores.

Sensor de temperatura.- Es el que está testeando continuamente que los valores de temperatura del tanque de leche para verificar que estén en los parámetros ideales, si la temperatura es menor a los 4°C se envía una alerta al sistema informando que la temperatura es muy baja, en cambio si la temperatura sobrepasa los 6°C se envía una alerta que hay un problema de enfriamiento.

Sensores de proximidad.- hay 4 en el proyecto, 2 ubicados en el tanque de enfriamiento y 2 en los tanques de recolección, uno por cada cantata de estación de ordeño. Estos a su vez no son exactos pero son mucho más económicos que implementar una solución basado en celdas de sobrecarga.

Sensor Rfid.- este se encarga de verificar la identidad de cada vaca, está ubicado dentro del móvil Android y nos permite poder realizar el ordeño de cada vaca teniendo en cuenta que la vaca está ya registrada en la base de datos.

3.2.2 ETAPA DE PROCESAMIENTO

Las señales adquiridas en la unidad de entrada deben ser analizadas en tiempo real. Para esto se ha creado un programa en lenguaje Python en el entorno Cloud 9 anteriormente detallado.

Las señales adquiridas son:

- Sensor de temperatura del tanque
- Sensores de proximidad del tanque
- Sensores de proximidad de la cantara
- Sensor de RFID

El sensor de temperatura del tanque va de 10mV/C° entonces tendríamos que tener un valor aproximado de entre 40mv a 60mv que son los valores de 4 a 6 °C, temperatura recomendable para el sustento adecuado de la leche puesto que un aumento de la temperatura provoca la proliferación de las bacterias en la leche y una disminución puede dar lugar a fenómenos de congelación.

Los sensores de proximidad están ubicados dentro del tanque para medir la cantidad de leche a través de una fórmula propuesta según sea el tipo de tanque, se usa también el mismo modelo de sensor del tanque para pesar los envases de recolección de leche.

Una vez que se han obtenido los datos, la unidad de control toma su parte que se detalla a continuación.

3.2.3 ETAPA DE CONTROL

En esta etapa se analiza los datos obtenidos para llevar a cabo el control del sistema, se divide en control continuo y eventual.

Control continuo.- aquí verificamos cada minuto el valor de temperatura del tanque que se encuentre en el rango especificado.

Control eventual.- son los demás sensores que una vez se está realizando el ordeño de las vacas, se controla el vaciado de la leche, el alimentador de vacas, identificación de las vacas.

3.2.4 ETAPA DE ACTUADOR

En esta etapa podemos interactuar con los distintos sensores. Entre las funciones tenemos

1. En la fase de ordeño que puede ser manual o automático aquí podemos mandar a activar las electroválvulas que vienen con la ordeñadora ya sea para activar o desactivar el ordeño, además que hay la posibilidad de un pare si se lo envía de forma automática, censando la actividad del ordeño y no habiendo cambios en la cantidad de leche.
2. En la fase de conservación una vez finalizado el ordeño de la vaca se activa la electroválvula para dar paso a la descarga del contenido hacia el tanque de conservación y poder iniciar otro ordeño una diferente vaca en la estación.
3. En la fase de alimentación cada que termine las 2 fases anteriormente mencionados, se la recompensa a la vaca con su ración proporcional de comida, se eleva la compuerta de salida de comida de la tolva para que caiga su porción.

Este método nos sirve para alimentar a las vacas, que den más leche y seleccionar las vacas que hacen más rentable este negocio.

3.2.5 ETAPAS DE VISUALIZACIÓN

Para presentar gráficamente la información procesada en las etapas anteriores se ha desarrollado una interfaz web amigable en plataforma libre que permita al usuario observar simultáneamente el funcionamiento de los sensores y fases, facilitando el análisis del ordeño sistematizado y estados de la leche.

Aquí podemos monitorear todo lo que sucede en el momento del ordeño, visualizando que actividad está realizando cada estación.

3.3 COSTO DE IMPLEMENTACIÓN

El costo de la implementación no es muy elevado respecto a las ganancias que pueden generarse a partir del control y monitores del ganado.

A continuación en la Tabla 2 se examina los costos del producto:

***SBC* y sensores**			
Equipo	**Precio**	**Cantidad**	**Total**
Beaglebone Black*	52,95$	1	52,95
Hc-sr04	4$	4	16
Elevador	8$	1	8
Móvil Nfc	130$	1	130
Sensor de temperatura	3$	1	3
Electroválvula	85$	1	85
Lcd 16x2	6,5$	1	6,5
Relés	0,75$	8	6
Otros elementos	5$	1	5
Circuito impreso	3$	1	3
Caja proyecto	10$	1	10
Cables	0,45$	15	6,75
Costos importación	61,5$	1	61,5
		total	393,7$

Tabla 2: Costos de inversión del producto

*este precio es del exterior, no existe este producto de venta en ecuador

Con un costo aproximado de 400 podemos reorganizar el ganado y dar mayores ganancias al dueño, este valor se lo puede recuperar en cuestión de meses, los dispositivos en el sistema son de bajo costo con respecto a los de uso industrial.

El método de ahorro se define en 4 partes.

- Selección de vacas con el método de alimentación
- Extracción de leche máxima de cada vaca
- Control de perdida de leche
- Conservación del producto para mayor pureza y valor

CAPÍTULO 4

4. PRUEBAS Y RESULTADOS

4.1 PRUEBAS DE HARDWARE

Se realizo varias pruebas de hardware, así como el cálculo de la cantidad de leche.

Con la finalidad de verificar la fiabilidad del producto se han realizado varias pruebas a los equipos instalados, para poder asegurar una buena respuesta en situaciones de estrés del sistema, se muestra los resultados en las Tabla 3 y 4.

	Beaglebone en función	**Beaglebone suspendido**
Uso de Cpu promedio	13.2%	9.6%
Uso de memoria Ram	432.15mb	413.86mb
Temperatura Cpu	38.1 °C	37.6 °C
Temperatura hardware	34.8 °C	32.9 °C

Tabla 3: Respuestas de hardware

Sensores de Proximidad (prueba en cántara)

	Sensor sin filtro de datos	**Sensor con filtros de datos**
En vacío	15 %de error	8% de error
En nivel medio(5)	12% de error	6% de error
Lleno(9)	10 %de error	5 %de error

Tabla 4: Probabilidades de error del sensor de proximidad

Pruebas de tiempo de respuesta del sistema

El tiempo de respuesta de cada orden se procesa en un tiempo promedio de 3 segundos aproximadamente, este tiempo se debe a los cambios de estado de las peticiones al sistema.

La recopilación de datos de los sensores de ultrasonido es de 35 muestras filtradas, cada aproximadamente 5 segundos se envía el dato del sensor con el valor de la cantidad de líquido.

Las electroválvulas usadas dan paso al líquido más rápido que los de una ordeñadora por lo que la estabilidad del líquido no es muy precisa para poder censar rápido y efectivo el volumen de líquido, por lo que hay q esperar a que se estabilice para una mayor precisión.

4.2 ESCENARIOS

Se ha planteado los 3 escenarios para mostrar el funcionamiento del proyecto que son web, aplicación Android y manual, se muestra el resultado en la Tabla 5.

	Tiempo promedio de envío	**Tiempo promedio de ejecución**
Apk Android	5 s	8 s
Vía web	7 s	11 s
Manual	8 s	9 s

Tabla 5: Tiempo de ejecución de órdenes de ordeño

Para el ordeño a través de la aplicación Android el tiempo que se demora en enviar la orden de ejecución de ordeño al sistema consiste en de leer el Rfid y envío de petición desde el móvil al sistema. Las demás desde el menú de elección de vaca, esto no toma en cuenta la amarrada de las patas de la vaca y la puesta de las pezoneras de la vaca, se muestra el resultado en la Tabla 6.

Él envió de órdenes de ordeño se demoran aproximadamente lo mismo.

	Retardo por fase	**Tiempo total ordeño**
Apk Android	4 s	17 s
Vía web	4 s	20 s
Manual	1s	13s

Tabla 6: Tiempo de ejecución de órdenes del sistema

Hemos comprobado que el sistema responde más rápido de modo manual y es más eficaz pero si se considera el tiempo que el ordeñador necesita para ir al sistema y regresar entonces esta opción se vuelve muy lenta respecto a las demás.

Tomando en cuenta lo anterior mencionado tenemos que por muy poco la mejor opción es la aplicación Android que le toma menos de 20s ejecutar todos los módulos del sistema. Esta fue una prueba práctica para verificar la rapidez del sistema sin contar el tiempo que emplea para llenar las cantaras y el tanque de conservación.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Con las pruebas realizadas podemos concluir que es posible administrar el ganado por medio del sistema ROAC y también es posible la rentabilidad del prototipo ya que el ahorro que se genera con la puesta en marcha del prototipo cubre su instalación.

1. Se observa que el diseño del circuito de acoplamiento de la señal funciona correctamente. Responde bien ante la sobrecarga de información.
2. Se pudo desarrollar una interfaz web capaz de controlar el prototipo y de una manera fácil y sencilla que puede ser manejada desde su misma red o a través de internet desde cualquier parte del mundo.
3. Se crearon varios scripts en Python para el control y funcionamiento del sistema.
4. Para contrarrestar el problema de ruido de la señales de entrada hay que colocar retardos de tiempo en el código de la adquisición de datos para evitar que el puerto externo tome como valor de entrada los niveles de voltajes provenientes del ruido, lo que nos da ayuda a evitar algún error del sistema.
5. Se implementó un servidor basado en debían, lo cual lo hizo robusto capaz de responder muy rápido a las peticiones del sistema.

RECOMENDACIONES

Las recomendaciones mencionadas ayudan a implementar un sistema estable.

1. Debido a la amplitud de las señales se recomienda usar divisores de voltaje a 3v para evitar un sobre voltaje y quemar el equipo.
2. Se recomienda realizar el PCB bajo las normas de diseño para disminuir los efectos del ruido electromagnético.
3. Se recomienda instalar el sistema operativo con debían 7.8 que contiene las últimas actualizaciones de los paquetes para programar la tarjeta, además de instalarlo sobre una micro SD para en caso de algún percance con el equipo poder migrar a otro con solo insertar la micro SD.
4. Debido a las limitaciones del equipo se recomienda usar doble fuente de alimentación una para la *SBC* y otra para el resto del sistema.

BIBLIOGRAFÍA

[1] M. Carvajal & B. Kerr, Factores genéticos que influencian la composición de la leche bovina, Instituto de Investigaciones Agropecuarias, INIA Remehue, Osorno B. Centro de Estudios Científicos (CECs), Valdivia, 2015.

[2] R. Vera, Perfiles por País del Recurso Pastura/Forraje, FAO, Viña del Mar, 2004.

[3] A. R. Novoa, aspectos nutricionales en la producción de leche, Vol. 1, Catie, San Jose, Costa Rica, 1983.

[4] E. Sverdlov, (2012, Octubre 4). Lamp on Debian [Online] Disponible en: http://www.digitalocean.com.

[5] G. Coley, (2014, Marzo 21). Books specifically on BeagleBone [Online]. Disponible en: http://www.elinux.org.

[6] J. Cooper, (2013, Junio 13). Setting up IO Python Library on Beaglebone Black [Online]. Disponible en: http://learn.adafruit.com.

[7] Cytron, (2013, Mayo). HC-SR04 User's Manual [Online]. Disponible en: http://www.cytron.com.my.

[8] Texas Instruments, (1999, Agosto). LM35 Precision Centigrade Temperature Sensors [Online]. Disponible en: http://www.ti.com.

[9] Constitución de la República del Ecuador, Ley Orgánica del Régimen de la Soberanía Alimentaria, Registro oficial No 446, febrero 2015.

[10] Redeszone, (2013, noviembre). Manual de instalación y configuración de este servidor FTP para Linux [Online]. Disponible en: http://www.redeszone.net/.

ANEXOS

ABREVIATURAS

GPIO	Entrada y Salida de propósito general.
I/O	Entrada y Salida.
PWM	Modulación por ancho de pulsos.
UART	Transmisor-Receptor Asíncrono Universal.
I2C	Circuitos Inter-Integrados.
SPI	Interfaz de periféricos serie.
LCD	Display de cristal líquido.
ROAC	Rfid, ordeño, alimentación y conservación.
PHP	PHP Hypertext Pre-processor.
SBC	SINGLE BOARD COMPUTER.
RFID	IDentificación por Radio Frecuencia.
Apache Tomcat	Servidor de páginas web.
ADC	convertidor analógico digital.
FTP	protocolo de transferencia de archivos.
SSH	intérprete de órdenes seguro.
IDE	ambiente de desarrollo integrado.

Código Python para ordeño por botoneras.

```
# -*- coding: iso-8859-15
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.ADC as ADC
import Adafruit_CharLCD as LCD
import threading
import MySQLdb
import time
import sys
import os

GPIO.setup("P9_12", GPIO.OUT)
GPIO.setup("P9_14", GPIO.OUT)
GPIO.setup("P9_15", GPIO.OUT)
GPIO.setup("P9_23", GPIO.IN)
GPIO.setup("P8_17", GPIO.IN)
GPIO.setup("P8_18", GPIO.IN)
GPIO.setup("P8_19", GPIO.IN)
lcd_rs       = 'P8_8'
lcd_en       = 'P8_7'
lcd_d4       = 'P8_10'
lcd_d5       = 'P8_9'
lcd_d6       = 'P8_12'
lcd_d7       = 'P8_11'
lcd_backlight = 'P8_5'
lcd_columns   = 16
lcd_rows     = 2

lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7,lcd_columns,
lcd_rows, lcd_backlight)
lcd.clear
def stop():
   fx = open( "/home/ev-con/cambioRele2", "wb" )
   fx.write("0")
   fx.close()
   time.sleep(1)
   fx = open( "/home/ev-con/cambioRele2", "wb" )
   fx.write("1")
   fx.close()
   time.sleep(0.2)
   fx = open( "/home/ev-con/cambioRele", "wb" )
   fx.write("0")
   fx.close()
   time.sleep(1)
```

```
    fx = open( "/home/ev-con/cambioRele", "wb" )
    fx.write("1")
    fx.close()
    time.sleep(0.2)
def alimentar(num):
    time.sleep(4)
    for i in range(num):
        GPIO.output("P9_15", GPIO.HIGH)
        time.sleep(0.2)
        GPIO.output("P9_15", GPIO.LOW)
        time.sleep(2)

def ord1():
    print int(os.popen("cat /home/ev-con/cambioRele1").read())
    while int(os.popen("cat /home/ev-con/cambioRele1").read()):
        auto = threading.Thread(target=AutoParado)
        auto.start()
        orda = "v"
        print("activar rfid")
        #name = raw_input("id vaca? ")
        db = MySQLdb.connect(host="localhost", # your host, usually localhost
            user="root", # your username
            passwd="eduedu2", # your password
            db="telecontrol") # name of the data base
        cur = db.cursor()
        cur.execute("SELECT * FROM vaca order by nombre")
        i=0
        nombres = []
        idVacas = []
        for row in cur.fetchall() :
            nombres.append(row[2])
            idVacas.append(row[0])
        rfid1 = nombres[i]
        print("rfid = " + rfid1)
        lcd.message("Vaca:" + rfid1)
        while not GPIO.input("P8_18"):#rf == 0:#activar ordeño
            time.sleep(0.2)
            if GPIO.input("P8_17"):#iniciar
                print("cambio rf")
                i += 1
                if i >= len(nombres):
                    i=0
                rfid1 = nombres[i]
                print("rfid =  " + rfid1)
                lcd.clear()
```

```
      lcd.message("Vaca: " + rfid1)
      time.sleep(0.2)
lcd.clear()
lcd.message("inicio ordeno")
stop()
time.sleep(0.5)
GPIO.output("P9_12", GPIO.HIGH)

auto = threading.Thread(target=AutoParado)
auto.start()
while not GPIO.input("P9_23"):#via == 0:#activar ordeño
   time.sleep(0.1)
   if GPIO.input("P8_18"):#iniciar
      lcd.clear()
      stop()
      time.sleep(0.5)
      if GPIO.input("P9_12"):
         print("pause ordeño")
         lcd.message("pause ordeno")
         GPIO.output("P9_12", GPIO.LOW)
         auto = threading.Thread(target=AutoParado)
         auto.start()
      else:
         print("reinicio ordeño")
         lcd.message("reinicio ordeno")
         GPIO.output("P9_12", GPIO.HIGH)#desactivar bomba
         auto = threading.Thread(target=AutoParado)
         auto.start()
      #time.sleep(0.5)

#xx = int(float(os.popen("cat /home/ev-con/cantara1.txt").read())*10)
stop()
time.sleep(0.5)
GPIO.output("P9_12", GPIO.LOW)
lcd.clear()
time.sleep(0.5)
lcd.message("inicio Vaciado")
GPIO.output("P9_14", GPIO.HIGH)
time.sleep(0.5)
auto = threading.Thread(target=AutoParado)
auto.start()
db = int(float(os.popen("cat /home/ev-con/cantara2.txt").read())*10)
alm2 = db/10
almm = threading.Thread(target=alimentar,args=(alm2,))
almm.start()
```

```
        while float(os.popen("cat /home/ev-con/cantara2.txt").read()) > 0:
            time.sleep(1)
            fx = open( "/home/ev-con/estado2.txt", "wb" )
            fx.write("Vaciando")
            fx.close()
            time.sleep(1)
        cur.execute("INSERT INTO ordeno (`idVaca`, `peso`, `fecha`) VALUES
(2,"+str(db)+",20150923)")
        fx = open( "/home/ev-con/estado2.txt", "wb" )
        fx.write("Sin Actividad")
        fx.close()
        stop()
        time.sleep(0.5)
        GPIO.output("P9_14", GPIO.LOW)
        lcd.clear()
        time.sleep(0.5)
        lcd.message("alimentando vaca")


        lcd.clear()
        time.sleep(0.5)
        lcd.message("fin del ordeno")



def ord2():
    print int(os.popen("cat /home/ev-con/cambioRele2").read())
    while int(os.popen("cat /home/ev-con/cambioRele2").read()):
        auto = threading.Thread(target=AutoParado)
        auto.start()
        orda = "v"
        print("activar rfid")
        #name = raw_input("id vaca? ")
        db = MySQLdb.connect(host="localhost", # your host, usually localhost
            user="root", # your username
            passwd="eduedu2", # your password
            db="telecontrol") # name of the data base
        cur = db.cursor()
        cur.execute("SELECT * FROM vaca order by nombre")
        i=0
        nombres = []
        idVacas = []
        for row in cur.fetchall() :
            nombres.append(row[2])
            idVacas.append(row[0])
```

```
rfid1 = nombres[i]
print("rfid = " + rfid1)
lcd.message("Vaca:" + rfid1)
while not GPIO.input("P8_18"):#rf == 0:#activar ordeño
   time.sleep(0.2)
   if GPIO.input("P8_17"):#iniciar
      print("cambio rf")
      i += 1
      if i >= len(nombres):
         i=0
      rfid1 = nombres[i]
      print("rfid =  " + rfid1)
      lcd.clear()
      lcd.message("Vaca: " + rfid1)
      time.sleep(0.2)
lcd.clear()
lcd.message("inicio ordeno")
stop()
time.sleep(0.5)
GPIO.output("P9_12", GPIO.HIGH)

auto = threading.Thread(target=AutoParado)
auto.start()
while not GPIO.input("P9_23"):#via == 0:#activar ordeño
   time.sleep(0.1)
   if GPIO.input("P8_18"):#iniciar
      lcd.clear()
      stop()
      time.sleep(0.5)
      if GPIO.input("P9_12"):
         print("pause ordeño")
         lcd.message("pause ordeno")
         GPIO.output("P9_12", GPIO.LOW)
         auto = threading.Thread(target=AutoParado)
         auto.start()
      else:
         print("reinicio ordeño")
         lcd.message("reinicio ordeno")
         GPIO.output("P9_12", GPIO.HIGH)#desactivar bomba
         auto = threading.Thread(target=AutoParado)
         auto.start()
      #time.sleep(0.5)

#xx = int(float(os.popen("cat /home/ev-con/cantara1.txt").read())*10)
stop()
```

```
        time.sleep(0.5)
        GPIO.output("P9_12", GPIO.LOW)
        lcd.clear()
        time.sleep(0.5)
        lcd.message("inicio Vaciado")
        GPIO.output("P9_14", GPIO.HIGH)
        time.sleep(0.5)
        auto = threading.Thread(target=AutoParado)
        auto.start()
        db = int(float(os.popen("cat /home/ev-con/cantara2.txt").read())*10)
        alm2 = db/10
        almm = threading.Thread(target=alimentar,args=(alm2,))
        almm.start()
        while float(os.popen("cat /home/ev-con/cantara2.txt").read()) > 0:
            time.sleep(1)
            fx = open( "/home/ev-con/estado2.txt", "wb" )
            fx.write("Vaciando")
            fx.close()
            time.sleep(1)
        cur.execute("INSERT INTO ordeno (`idVaca`, `peso`, `fecha`) VALUES
(2,"+str(db)+",20150923)")
        fx = open( "/home/ev-con/estado2.txt", "wb" )
        fx.write("Sin Actividad")
        fx.close()
        stop()
        time.sleep(0.5)
        GPIO.output("P9_14", GPIO.LOW)
        lcd.clear()
        time.sleep(0.5)
        lcd.message("alimentando vaca")


        lcd.clear()
        time.sleep(0.5)
        lcd.message("fin del ordeno")
def AutoParado():

    print "init 6"
    prom = 0
    var = 0
    promant = 0
    promT = 0

    while True:
        tx = 0
```

```
t0 = 0
#laz = time.time()
GPIO.output("P8_16", GPIO.HIGH)
time.sleep(0.015)
GPIO.output("P8_16", GPIO.LOW)
while not GPIO.input("P8_15"):
   t0 = time.time()
while GPIO.input("P8_15"):#not GPIO.event_detected("P8_13"):#GPIO.input("P8_13"):
   tx = time.time()
   if (tx-t0)>12:
      print "error laz"
muestra = (tx-t0)*10000

if muestra == 0:
   GPIO.output("P8_14", GPIO.HIGH)
   print("xx = %f" % muestra)
   time.sleep(0.5)
   GPIO.output("P8_14", GPIO.LOW)
   time.sleep(0.5)
   if GPIO.input("P8_13"):
      print "sleep laz2"

if muestra >= 2 and muestra <= 24:
   if promant == 0:
      if var < 35:
         promT = promT + muestra
      var += 1
   else:
      if muestra < promant + 3.2 and muestra > promant - 3.2:
         if var < 35:
            promT = promT + muestra
         var += 1
      #else:
         #print("conta = %f" % muestra)
#else:
#print("xx = %f" % muestra)


if var >= 35:
   promT = float(promT/35)
   promant=promT
   litros = 21.30-promT
   if litros < 0:
      litros = 0
   print("Timelit = %f -- litros = %f" % (promT,litros))
```

```
            fb = open( "/home/ev-con/cantara2.txt", "wb" )
            fb.write(str(round(litros,3)))
            fb.close()
            var = 0


        time.sleep(0.17)


Ordeno1 = threading.Thread(target=ord1)
Ordeno2 = threading.Thread(target=ord2)

GPIO.setup("P8_16", GPIO.OUT)
GPIO.setup("P8_15", GPIO.IN)
GPIO.output("P8_16", GPIO.LOW)
#GPIO.add_event_detect("P8_15", GPIO.RISING)
GPIO.add_event_detect("P8_15", GPIO.RISING)
Ordeno1.start()
Ordeno2.start()
```

Código para ordeño principal

```
# -*- coding: iso-8859-15
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.ADC as ADC
import threading
import MySQLdb
import time
import sys
import os

GPIO.setup("P9_41", GPIO.IN)
GPIO.setup("P9_42", GPIO.IN)
GPIO.setup("P9_12", GPIO.OUT)
GPIO.setup("P9_14", GPIO.OUT)

time.sleep(0.5)

def orden():
   time.sleep(1)
   #ADC.setup()
   if sys.argv[2] == "1":
      if sys.argv[1] == "M":
         fx = open( "/sys/class/gpio/gpio30/value", "wb")
```

```
    fx.write("1")
    fx.close()
    time.sleep(1)
if sys.argv[1] == "T":
    time.sleep(float(sys.argv[3])/10)
    GPIO.output("P9_11", GPIO.LOW)
    print("ordeño terminado")
if sys.argv[1] == "P":
    var = int(float(os.popen("cat /home/ev-con/cantara1.txt").read()))
    peso = float(sys.argv[3])/10
    while var<peso:#alimentar
        GPIO.output("P9_11", GPIO.HIGH)
        var = float(os.popen("cat /home/ev-con/cantara1.txt").read())
    GPIO.output("P9_11", GPIO.LOW)
    print("ordeño terminado")
if sys.argv[1] == "V":
    xx = int(float(os.popen("cat /home/ev-con/cantara1.txt").read())*10)
    fx = open( "/home/ev-con/alim1.txt", "wb" )
    fx.write(str(xx/10))
    fx.close()
    while float(os.popen("cat /home/ev-con/cantara1.txt").read()) > 0:
        time.sleep(2)          #vaciar leche
        fx = open( "/home/ev-con/estado1.txt", "wb")
        fx.write("Vaciando")
        fx.close()
    fx = open( "/home/ev-con/estado1.txt", "wb" )
    fx.write("Sin Actividad")
    fx.close()
    fx = open( "/home/ev-con/cambioRele", "wb" )
    fx.write("0")
    fx.close()
    time.sleep(0.2)
    print("ok")
    print xx
    GPIO.output("P9_13", GPIO.LOW)
if sys.argv[1] == "A":
    alm1 = int(float(os.popen("cat /home/ev-con/alim1.txt").read()))
    print alm1
    for i in range(alm1):
        GPIO.output("P9_15", GPIO.HIGH)
        time.sleep(0.2)
        GPIO.output("P9_15", GPIO.LOW)
        time.sleep(2)
    fb = open( "/home/ev-con/alim1.txt", "wb" )
    fb.write(str(0))
```

```
            fb.close()
            time.sleep(1)
            print("ok6")
    if sys.argv[2] == "2":
        if sys.argv[1] == "M":
            while GPIO.input("P9_11"):
                #GPIO.output("P8_13", GPIO.LOW)#activar bomba
                time.sleep(1)
                #GPIO.output("P8_13", GPIO.HIGH)#DESactivar bomba
                time.sleep(1)
                print("asdw")
        if sys.argv[1] == "T":
            print("ok2")
        if sys.argv[1] == "P":
            print("ok3")
        if sys.argv[1] == "S":
            print("ok4")
        if sys.argv[1] == "V":
            print("ok5")
        if sys.argv[1] == "A":
            print("ok6")
def AutoParado():
    GPIO.setup("P8_14", GPIO.OUT)
    GPIO.setup("P8_13", GPIO.IN)
    GPIO.output("P8_14", GPIO.LOW)
    #GPIO.wait_for_edge("P8_13", GPIO.BOTH)
    GPIO.add_event_detect("P8_13", GPIO.RISING)
    print "init 0"
    prom = 0
    var = 0
    promant = 0
    promT = 0
    laz = 0

    print int(os.popen("cat /home/ev-con/cambioRele").read())
    while int(os.popen("cat /home/ev-con/cambioRele").read()):
        tx = 0
        t0 = 0
        #laz = time.time()
        GPIO.output("P8_14", GPIO.HIGH)
        time.sleep(0.011)
        GPIO.output("P8_14", GPIO.LOW)
        #time.sleep(0.005)
        while not GPIO.event_detected("P8_13"):
            t0 = time.time()
```

```
    while GPIO.input("P8_13") and laz == 0:#not
GPIO.event_detected("P8_13"):#GPIO.input("P8_13"):
        tx = time.time()
        if (tx-t0)>12:
            laz = 1
            print "error laz"
            time.sleep(0.5)
    muestra = (tx-t0)*10000
    laz = 0
    if muestra == 0:
        GPIO.output("P8_14", GPIO.HIGH)
        print("xx = %f" % muestra)
        time.sleep(0.5)
        GPIO.output("P8_14", GPIO.LOW)
        time.sleep(0.5)
        if GPIO.input("P8_13"):
            print "sleep laz2"


    if muestra >= 2 and muestra <= 24:
        if promant == 0:
            if var < 35:
                promT = promT + muestra
            var += 1
        else:
            if muestra < promant + 3.2 and muestra > promant - 3.2:
                if var < 35:
                    promT = promT + muestra
                var += 1


    if var >= 35:
        promT = float(promT/35)
        promant=promT
        litros = 21.35-promT
        if litros < 0:
            litros = 0
        print("Timelit = %f -- litros = %f" % (promT,litros))
        fb = open( "/home/ev-con/cantara1.txt", "wb" )
        fb.write(str(round(litros,3)))
        fb.close()
        var = 0

    time.sleep(0.17)
```

```
if len(sys.argv) >= 4:
    auto = threading.Thread(target=AutoParado)
    Ordeno = threading.Thread(target=orden)
    fx = open( "/home/ev-con/cambioRele", "wb" )
    fx.write("0")
    fx.close()
    time.sleep(1)
    fx = open( "/home/ev-con/cambioRele", "wb" )
    fx.write("1")
    fx.close()
    time.sleep(0.2)
    if sys.argv[2] == "1":
        if sys.argv[1] == "M":
            GPIO.setup("P9_11", GPIO.OUT)
            #GPIO.setup("P8_13", GPIO.OUT)
            GPIO.output("P9_11", GPIO.HIGH)#activar bomba
            #GPIO.output("P8_13", GPIO.HIGH)#activar bomba led
            print("ok activando bomba")
            Ordeno.start()
            time.sleep(1)
            auto.start()
            #print("manual 1 activando bomba")
            fx1 = open( "/sys/class/gpio/gpio30/value", "wb")
            fx1.write("1")
            fx1.close()
            fx = open( "/home/ev-con/estado1.txt", "wb" )
            fx.write("Ordeñando")
            fx.close()
        if sys.argv[1] == "T":
            GPIO.setup("P9_11", GPIO.OUT)
            GPIO.output("P9_11", GPIO.HIGH)
            print("x1 tiempo activando bomba")
            time.sleep(1)
            fx = open( "/home/ev-con/estado1.txt", "wb" )
            fx.write("Ordeñando")
            fx.close()
        if sys.argv[1] == "P":
            GPIO.setup("P9_11", GPIO.OUT)
            print("x peso")
            Ordeno.start()
        if sys.argv[1] == "S":
            #if os.popen("cat /sys/class/gpio/gpio30/value").read():
            GPIO.setup("P9_11", GPIO.OUT)#DESactivar bomba
            time.sleep(1)
```

```
            GPIO.setup("P9_11", GPIO.LOW)
            print("ok desactivando bomba")
            time.sleep(1)
            auto.start()
            fx2 = open( "/home/ev-con/estado1.txt", "wb" )
            fx2.write("sin Actividad")
            fx2.close()
        if sys.argv[1] == "V":
            print("vaciando leche")
            GPIO.setup("P9_13", GPIO.OUT)
            GPIO.output("P9_13", GPIO.HIGH)
            fx = open( "/home/ev-con/estado1.txt", "wb" )
            fx.write("Vaciando")
            fx.close()
            Ordeno.start()
            auto.start()
        if sys.argv[1] == "A":
            GPIO.setup("P9_15", GPIO.OUT)
            print("Alimentando Vaca")
            fx = open( "/home/ev-con/estado1.txt", "wb" )
            fx.write("Alimentando")
            fx.close()
            Ordeno.start()
    if sys.argv[2] == "2":
        if sys.argv[1] == "M":
            GPIO.setup("P9_16", GPIO.OUT)
            GPIO.output("P9_16", GPIO.HIGH)#DESactivar bomba
            print("ok1")
        if sys.argv[1] == "T":
            print("ok2")
        if sys.argv[1] == "P":
            print("ok3")
        if sys.argv[1] == "S":
            print("ok4")
        if sys.argv[1] == "V":
            print("ok5")
        if sys.argv[1] == "A":
            print("ok6")
else:
    print "Necesito parametros"
```