UNIVERSIDAD DE PIURA

REPOSITORIO INSTITUCIONAL PIRHUA

IMPLEMENTACIÓN DE UNA RED DE SUPERVISIÓN PARA RIEGO MECANIZADO CON PROTOCOLO MODBUS, PLCS Y MATLAB

Luighi de Francesch-Saavedra

Piura, setiembre de 2015

FACULTAD DE INGENIERÍA

Departamento de Ingeniería Mecánico-Eléctrica

De Francesch, L. (2015). *Implementación de una red de supervisión para riego mecanizado con protocolo Modbus, PLCs y Matlab*. Tesis de pregrado no publicado en Ingeniería Mecánico Eléctrica. Universidad de Piura. Facultad de Ingeniería. Programa Académico de Ingeniería Mecánico Eléctrica. Piura, Perú.

Esta obra está bajo una licencia
Creative Commons Atribución-
NoComercial-SinDerivadas 2.5 Perú

Repositorio institucional PIRHUA – Universidad de Piura

U N I V E R S I D A D D E P I U R A
FACULTAD DE INGENIERÍA

“Implementación de una red de supervisión para riego mecanizado con protocolo Modbus, PLCs y Matlab”

Tesis para optar el Título de
Ingeniero Mecánico Eléctrico

Luighi Giuseppe De Francesch Saavedra

Piura, Setiembre 2015

U N I V E R S I D A D D E P I U R A
FACULTAD DE INGENIERÍA

“Implementación de una red de supervisión para riego mecanizado con protocolo Modbus, PLCs y Matlab”

Tesis para optar el Título de
Ingeniero Mecánico Eléctrico

Luighi Giuseppe De Francesch Saavedra

Asesor: Dr. Ing. Justo Oquelis Cabredo

Piura, Setiembre 2015

A Dios.

A mis padres Giuseppe De Francesch y Luz Saavedra, por todo su amor, apoyo y ahínco para cumplir esta meta.

Prólogo

La implementación de una red de supervisión para pivotes de riego central tiene como mayor antecedente la instalación de 69 pivotes en la región de San Luis, Argentina. Este proyecto comprende la irrigación de más de 6550 ha de cultivo. Durante esta experiencia participaron profesores de la facultad de Ingeniería. Fueron parte de la planificación y puesta en marcha de este gran proyecto el cual logró aprovechar los recursos hídricos y habilitar más áreas de cultivo.

Ahora este mismo tipo de riego se está ejecutando en nuestro país en la región de Olmos (Lambayeque) y contará con un total de 115 pivotes en su etapa final. La cantidad de terrenos habilitados para la agricultura con este sistema de riego mecanizado asciende a más de 10 000 ha y son solo una parte del área habilitada gracias al proyecto especial de Irrigación e Hidroenergético de Olmos. También como parte de la introducción de tecnologías al sector agrícola incluye la programación de riegos los cuales vienen de la mano con las opciones que ofrecen para este caso Valley Irrigation, empresa dedicada a esta actividad.

El tema del estudio nace como iniciativa de buscar algún medio de compatibilizar los paneles de riego Valley con los PLCs y SCADA comunes para obtener una programación y monitoreo del riego más flexible. Para lograr este tipo de requerimiento, se estudia primero el protocolo de intercambio de datos que realiza el equipo en mención. Una vez estudiado el protocolo de comunicación, se da parte a las simulaciones, respetando la sintaxis en que se construye las tramas de comunicación, en este caso del panel de riego Valley. Como parte del proyecto se utilizaron los softwares: Click Programming, Base Station 2, Docklight, Free Serial Port Monitor y Matlab con estos programas se ha podido llevar a cabo el algoritmo insertado en el PLC, el análisis de las tramas de control y la interfaz gráfica para el monitoreo de los pivotes de riego. Todo esto es importante porque ha permitido la integración de varios sistemas para implementar el actual sistema de control del proyecto Agrolmos asegurando una buena producción y rentabilidad de estos terrenos.

Como corresponde también, mi agradecimiento a mis profesores y compañeros de investigación por la motivación brindada durante el desarrollo de la presente tesis. De manera especial agradezco al Dr. Ing. Justo Oquelis Cabredo, por sus valiosas enseñanzas, certero asesoramiento académico y confianza en la materia de investigación. Asimismo a los ingenieros Edilberto Vásquez Díaz, César Chingel Arrese y Dantty Ramos por su

colaboración y atención que me ofrecieron ante cualquier duda o problema que se presentaba durante el desarrollo del proyecto.

Resumen

El proyecto de irrigación Agrolmos tiene en su planeamiento la instalación de 115 pivotes de riego central, los cuales habilitaran más de 10 000 ha de cultivo, por ello la cantidad y disponibilidad del recurso hídrico es importantísima. Para utilizarla eficientemente se ha optado por los sistemas de aspersión y consecuencia de ello la tecnología utilizada debe ser capaz de cumplir con el requerimiento de usar eficiente el agua. Para lograr esto, se deben integrar varios equipos, el más importante de ellos es el panel de control de cada pivote. Este equipo contiene un protocolo de comunicación independiente y es justamente el motivo de investigación de la presente tesis.

Al cabo de la investigación se ha cumplido con los objetivos trazados:

- Desarrollar un sistema de supervisión y control de pivotes centrales de riego integrando herramientas y tecnologías acorde a la necesidad del proyecto.
- Hacer compatible el panel de riego Valley con los sistemas de comunicación industriales, permitiéndole aumentar su capacidad de enlazarse con más procesos de automatización.
- Crear un programa ejecutable en un PLC que migre las tramas del protocolo Modbus a tramas de comunicación con el panel de riego.
- Crear una interfaz gráfica, en la cual el usuario puedo controlar y monitorear las áreas de cultivo que tenga a cargo.

Los resultados obtenidos comprenden también una metodología para la extracción de protocolos de comunicación y su beneficio en la integración de sistemas automáticos de control. La implicancia económica también es un factor a favor pues implementar un tipo de sistema donde la instrumentación puede ser muy variada y con protocolos de comunicación independiente el lograr enlazarlos genera un gran ahorro en la parte productiva.

Finalmente se desarrolló una aplicación para iniciar los riegos en una hora programada por el usuario, de esta forma se puede irrigar durante la noche aprovechando el bajo perfil de velocidad del viento y de temperatura para dar mejor uniformidad en la distribución de agua y disminuir la cantidad de agua que se evapora antes de infiltrar respectivamente.

Índice

Introducción

El desarrollo de la ingeniería, la generación de tecnologías, los nuevos estándares de calidad y los exigentes cambios del entorno, implica una rápida adaptación y ello se logra con competitividad, investigación, desarrollo, innovación. A este tipo de situaciones no es ajena el aprovechamiento de los recursos naturales, uno de ellos es el uso eficiente del agua. Para ello nuestro país viene invirtiendo en una serie de obras hidráulicas. Forma parte de este plan, el proyecto especial de Irrigación e Hidroenergético de Olmos, el cual ha habilitado nuevas áreas de cultivos y por ende ha dado pie a la instalación de sistemas de riego acorde a la magnitud de este gran proyecto. Aquí nace el tema de tesis que tiene por objetivo integrar los equipos que participan en el riego. Para ello la tesis se ha dividido en cinco capítulos, los cuales se resumen a continuación.

En el primer capítulo, se da una visión global de lo que comprenden las obras de irrigación, los sistemas de riego por aspersión en específico el riego por pivotes y su impacto en la agronomía. También se da una explicación panorámica de la situación hídrica del Perú y los proyectos relacionados al aprovechamiento del agua.

En el segundo capítulo se hace un análisis económico de la implementación de un sistema integrado para el monitoreo de pivotes de riego. Estos dos primeros capítulos son importantes para situar al lector dentro del contexto de la tesis, debido a que es el proyecto Agrolmos en el cual se basa el estudio para la integración de los paneles de riego Valley.

El tercer capítulo abarca el aspecto teórico y explica a detalle el protocolo Modbus, la forma en que se construye una trama de comunicación y la sintaxis del mismo. Se dan ejemplos de las funciones de escritura y lectura para las diferentes opciones de intercambio de datos.

Para el desarrollo del proyecto se usan equipos de automatización, ejemplo de ellos son los PLCs, por ello el capítulo cuatro detalla el algoritmo que se ha desarrollado. Se muestran los diagramas de flujo y la metodología llevada a cabo para migrar el protocolo de comunicación.

El quinto capítulo al igual que el anterior son la parte práctica del proyecto. A lo largo de
o capítulo se desarrolla una interfaz gráfica en Matlab para que el usuario pueda
r los cultivos. Finalmente queda indicar que los tres últimos capítulos son
ntarios en el sentido teórico-práctico y sus aportes son muy importantes, pues
nsolidar los objetivos de la tesis.

Capítulo 1

Sistemas de riego mecanizado y su impacto en la agronomía

Debemos aprovechar el agua de las diferentes cuencas y regiones. Si bien es cierto que el régimen en la costa es irregular, la sensación de escasez de agua es mucho mayor a la real y esto se evidencia por la falta de infraestructura. No se puede permitir durante el tiempo de lluvias no almacenar el agua y durante sequías continuar arrojando cantidad de agua al mar.

Por ello nuestro país necesita contar con más estructuras de captación que permitan almacenar y derivar el agua para los diversos usos benéficos, más aún para el sector agrícola, donde el 80% del uso del agua del país se destina para riego.

Para nuestras regiones el agua es un recurso vital de desarrollo, producción y calidad de vida, por ende no se puede privarlas de este. (Barreto Escobedo, 2015)

El riego es una labor de gran importancia para el desarrollo de los cultivos y requiere siempre ser optimizado, con el fin de ahorrar agua y dar una seguridad alimentaria a las personas. Para esta problemática se necesita implementar sistemas de riego que no malgasten el agua y contribuyan al aprovechamiento de este recurso.

El uso de sistemas de riego tradicionales como los de inundación y surcos si bien son accesibles por el bajo costo de implementación tienen inconvenientes debido a la gran cantidad de agua que usan. La eficiencia y uniformidad de estos tipos de riego son relativamente bajas en comparación con los sistemas de riego presurizado como lo son aspersión, micro aspersión y goteo (Macías Macías, Vergara Sabando, Macías Solórzano, & Bazurto Zambrano, 2011).

1.1 Obras de irrigación

Irrigación es la aplicación artificial del agua en el suelo con el propósito de suplir a ésta de la humedad esencial para el crecimiento de las plantas.[1] También podemos definir la irrigación como el aporte a los terrenos de cultivo de un volumen controlado y oportuno de agua para lograr el desarrollo de los cultivos hasta la maduración de sus frutos.

Así mismo se define como Irrigación: toda acción, efecto o técnica para regar y desarrollar la agricultura. Las obras de irrigación en sí mismas son un conjunto de canales, estructuras

[1] Irrigation Principles and Practices, Orson W. Israelsen

y trabajos de acondicionamiento realizados por el hombre con la finalidad de poder suministrar a la tierra a cultivar del agua en la cantidad y en el tiempo que lo requieran.

Considerando esto, una obra de riego en general consta de cuatro partes: Captación, Conducción, Distribución y Zona de riego.

1.1.1 Captación de agua

La captación es la parte inicial de cualquier obra de irrigación, pues para empezar a utilizar el agua, se necesita primero disponer de ella. La captación consiste en recolectar y almacenar agua proveniente de diversas fuentes para su uso benéfico. Pueden ser de tres tipos

- Superficial: Hace referencia al agua que deriva de los deshielos de los glaciares, lluvia y que eventualmente llega a los ríos y lagos. Antes de que regrese a los océanos o se evapore, el agua puede ser represada y derivada para su posterior uso.

- Subterránea: Esta agua se origina por el almacenamiento que se produce en el subsuelo, como consecuencia de la permeabilidad del terreno y la existencia de capas impermeables que permiten retenerla. El agua puede alcanzar capas porosas saturadas con agua conocidas como acuíferos, en términos generales se puede decir que un acuífero es aquél estrato o formación geológica que permite la circulación del agua por sus porosas grietas, permitiendo su aprovechamiento.

- Subálvea: Esta fuente de recurso hídrico hace mención a su término en sí, subálveo que corresponde al agua subterránea bajo la corriente de un río, riachuelo, quebradas o arroyo que se va acumulando con el largo paso del tiempo.

1.1.2 Conducción

Una vez captada el agua mediante la ejecución de alguna obra, la cual garantiza la disponibilidad de este recurso viene ahora el paso de la conducción. La conducción tiene como principal función transportar el agua desde la zona de captación hasta la zona de distribución. Por lo general, la obra fundamental es un canal de conducción principal, donde debe procurarse en lo posible que la longitud sea la mínima.

Los sistemas de conducción en los proyectos de riego pueden estar conformados por conductos abiertos que fluyen bajo la acción de la gravedad, a los que se les denominan canales, o por conductos cerrados que fluyen parcialmente llenos como los túneles. Es también de vital importancia las obras inherentes a la conducción como lo es la estructura hidráulica del desarenador.

Cuando se capta agua de un río, inevitablemente estaremos captando también sedimentos en suspensión y de arrastre. Un desarenador es la estructura que permite eliminar estas partículas que se encuentran en suspensión en el agua y posteriormente deben ser arrojadas mediante una adecuada operación.

Los desarenadores, se suelen construir aguas arriba de las centrales hidroeléctricas. Con el fin de quitar ciertas partículas en suspensión y obtener una menor erosión en las turbinas y los equipos de generación, brindando así un mayor tiempo de vida a la central.

1.1.3 Zona de distribución

Después de haber captado con eficacia las aguas provenientes de una fuente, y luego de haberlas dirigido hasta la zona donde se quiere contar con este recurso. Entramos a la zona de distribución, que viene a ser el conjunto de estructuras, regaderas, canales secundarios y terciarios que llevan el agua hasta los puntos de donde se aplicará a la tierra. Esta zona de distribución viene a comprender toda esa red que alimenta a la zona de riego, se caracteriza por su ramificación en diversos sectores para cubrir de manera completa toda el área a irrigar.

1.1.4 Zona de riego

Es la superficie de tierras por regar, son las llamadas zonas de cultivo y que se miden en hectáreas (ha). Las áreas o zonas de riego es el destino final del agua, el objetivo de todo proyecto de irrigación. Cuando se desarrolla una obra de irrigación siempre se contempla la cantidad de hectáreas que van a ser regadas, éste es un parámetro importante de todo proyecto ya que también determina su rentabilidad.

Por otro lado, uno de los problemas que se debe tener en cuenta es la eficiencia en el uso del agua y en la forma de aplicarse a los cultivos. Pues, existen pérdidas desde el agua captada inicialmente hasta la cantidad de agua que utiliza cada planta para su desarrollo. En estas pérdidas no solo están incluidas las referidas a evaporación e infiltración, sino también las que tienen que ver con una mala operación y desperdicio de la misma. (Barreto Escobedo, 2015)

1.2 Consideraciones generales en sistemas de riego

Una buena práctica de regado es la que busca siempre optimizar la eficiencia en la aplicación de agua, esta debe ser entendida como la fracción del agua aplicada al cultivo que es utilizada para satisfacer las necesidades de la planta. En términos más estrictos es la cantidad de agua que queda almacenada en la zona radicular a disposición del cultivo. (Tarjuelo Martín-Benito, 1991)

Para conseguir ello es necesario minimizar las perdidas por evaporación, escorrentía[2], percolación[3] y otras perdidas menores, para lo cual se precisa que el sistema haya estado bien diseñado, se opere de forma correcta y reciba el mantenimiento adecuado.

Los términos más utilizados para describir el comportamiento del riego y la distribución de agua respecto del área del cultivo son eficiencia y uniformidad. Siendo estos los parámetros más importantes suelen ir acompañados de términos adicionales para caracterizar a una actividad de riego en específico.

Uniformidad en el riego indica el grado de igualdad de dosis de agua recibida en los diferentes puntos del terreno. Por su lado la eficiencia indica el porcentaje de agua aplicada o consumo bruto, respecto de la cantidad que es aprovechada para satisfacer las necesidades del cultivo.

---

[2] Corriente de agua que se vierte al rebasar su depósito o cauce naturales o artificiales. (Española, Real Academia)

[3] Ensayo para medir las características de infiltración de un terreno. Dicho de un líquido: Moverse a través de un medio poroso. (Española, Real Academia)

1.2.1 Objetivos generales del riego

- Brindarle a los cultivos, la humedad necesaria en su terreno, para que puedan desarrollarse.

- Disolver las sales, abono, fertilizantes y nutrientes en general y así puedan ser asimilables por las raíces de las plantas.

- Asegurar las cosechas contra periodos de sequias.

- Refrescar la temperatura de la atmosfera cercana y la del suelo con el fin de mejorar las condiciones ambientales para el desarrollo vegetal.

- Asegurar un recurso alimentario que a su vez genere un comercio rentable.

El objetivo general que persigue o busca con ahínco el riego es aplicar a los cultivos, de forma eficiente y sin alterar la fertilidad del suelo, el agua en el momento adecuado y en la cantidad necesaria para lograr un crecimiento óptimo. (González P., 2007)

1.2.2 Tipos de riego

Es en la zona de riego donde se desarrollan los diversos sistemas o tipos de riego, siendo entre los más empleados los siguientes. (Barreto Escobedo, 2015)

- Riego por gravedad: cuando el agua se distribuye a los cultivos mediante acequias, canales y el agua fluye por acción del desnivel o pendiente.

- Riego por anegamiento: cuando los terrenos de cultivo son sumergidos en una columna de agua, caso del riego del arroz en los valles de la costa.

- Subirrigación: se denomina cuando el agua no fluye superficialmente sino que se distribuye debajo del nivel del terreno, para alimentar directamente la napa freática y el agua llega a las raíces por capilaridad.

- Riego por aspersión: cuando el riego se efectúa por sistemas similares a la lluvia.

- Irrigación por infiltración: cuando una regadera, o canal puede humedecer en toda su longitud una profundidad de terreno para permitir el humedecimiento de las raíces.

- Riego por goteo: mediante esta técnica se lleva el agua hasta la proximidad de las raíces de cada una de las plantas y se provee de la humedad suficiente al suelo, de acuerdo con los requerimientos de los cultivos.

1.2.3 Elección del tipo de riego

Son varios los factores que se deben abordar para optar por un tipo de riego, se detallará los más influyentes al momento de analizar. (González P., 2007)

- La topografía: Cuando el terreno tiene una geografía abrupta, la nivelación es necesaria y puede ocasionar daños al suelo original. Si las características topográficas son difíciles se debe estudiar el costo que demande lograr una superficie trabajable y el costo que demandaría hacer un riego por aspersión u otro diferente.

- Las características físicas del suelo: Es muy importante hacer estudios de las propiedades del suelo a irrigar, los valores de infiltración que es la velocidad de captación de agua no debe ser superada con el flujo de agua que se pretende regar pues habrá problemas de encharcamiento. En el caso de aspersión y goteo, si el suelo tarda en infiltrar el agua, se pueden dar problemas de escorrentía y erosión.

- El tipo de cultivo: Para seleccionar el modo en que se va regar, se debe tener en cuenta que plantaciones son las que se tienen en crecimiento. Por ejemplo el tamaño nominal de la planta, sus necesidades hídricas, su velocidad de crecimiento y cosecha son aspectos de análisis cara al riego que se instalará. Hay cultivos que se desempeñan de forma óptima en algunos sistemas de riego. Así, los árboles frutales se desarrollan muy adecuadamente en riego por goteo y los cultivos hortícolas les va mucho mejor con el riego por surcos.

- La disponibilidad de agua: Saber con qué frecuencia y cantidad se dispondrá del recurso hídrico o tener un buen pronóstico de este, ayudará en el planeamiento de irrigación del cultivo, por ende ayudará a optar por el mejor método de irrigación. Los sistemas que puedan resultar más eficientes en una determinada situación serán los más adecuados cuando la disponibilidad de agua sea baja.

- La calidad del agua: Las condiciones en que es captada el agua y llevada a la zona de riego influye en la elección del riego pues dependiendo de las características, niveles de salinidad, sedimentos y elementos en suspensión será más o menos dañina para el cultivo. El riego por aspersión no se recomienda en la mayoría de los casos en que el agua tiene un alto nivel de salinidad porque acelera la corrosión en la instalación.

- La disponibilidad de mano de obra: El contar con técnicos calificados para la instalación de los equipos que se elijan también es un tema a tomar en cuenta. Por ejemplo el riego por pivotes precisa de personal entrenado y capacitado. Saber que se dispone del talento humano especializado para el riego que se quiere elegir alivia los temas de la ingeniería de detalle, la puesta en marcha y mantenimiento.

- El costo de la instalación: El tema económico y un balance del retorno de la inversión son factores claves para decidir entre qué tipo de riego implantar. El rendimiento de la cosecha y todo lo relacionado al tema monetario son también claves para optar por uno u otro tipo de riego.

- El efecto sobre el medio ambiente: Este factor abarca con mayor fuerza a las obras de irrigación como son los embalses pues modifican fuertemente el cauce de un rio y su retención forma micro ecosistemas diferentes al original. Es tarea de los especialistas en temas medioambientales determinar su impacto y si es beneficioso

o contraproducente para la conservación de la flora y fauna cercana. Actualmente es un tema que va tomando mayor fuerza en todos los proyectos industriales.

1.3 Riego por aspersión

En este apartado se expondrá a mayor detalle sobre este tipo de riego dado que la implementación realizada para el monitoreo y control esta aplicada sobre pivotes de riego central, los cuales son un ejemplar del riego por aspersión.

El riego por aspersión es un tipo de riego en que el agua llega a las plantas en forma de lluvia. Se busca que la lámina de riego que se está aplicando infiltre en el mismo punto donde cae. Para lograr este objetivo es necesario contar con una red de distribución y una presión suficiente para que el agua llegue hasta los aspersores.

En general los sistemas de riego por aspersión se adaptan bastante bien a topografías ligeramente accidentadas, el consumo de agua es moderado y la eficiencia muy aceptable. Sin embargo esta forma de aplicar el agua es extremadamente sensible a las condiciones meteorológicas en especial al viento y a la aridez de clima.

Por un lado si las gotas son demasiado pequeñas, es decir muy pulverizadas podrían evaporarse antes de llegar al suelo y en el otro extremo de la situación es que se genere un chorro demasiado denso que al momento de impactar al suelo lo erosione y hasta dañe a la planta especialmente en la fase de germinación.

El riego por aspersión tiene un potencial adicional en cuanto a la aplicación de fertilizantes y uso de productos fitosanitarios ya que se pueden inyectar disueltos en el agua en el sistema de bombeo. (Macías Macías, Vergara Sabando, Macías Solórzano, & Bazurto Zambrano, 2011)

1.3.1 Componentes de un sistema por aspersión

Las unidades básicas que componen un sistema de riego por aspersión son: el grupo de bombeo, las tuberías principales con sus hidrantes, las tuberías portaemisores (alas de riego) y los emisores propiamente dichos.

Los emisores pueden ser: tuberías perforadas, difusores fijos y aspersores. Del grupo de los emisores los más utilizados son los aspersores que en su mayoría llevan una o dos boquillas cuyos chorros de agua forman ángulos de 25º a 28º con la horizontal.

La clasificación de los aspersores puede darse tomando distintos aspectos (Tarjuelo Martín-Benito, 1991):

a) Según la velocidad de giro:

   -De giro rápido, cuando es mayor a 6 vueltas/min. Su uso es en jardinería, viveros y en algunos casos en horticultura.
   -De giro lento, cuando va desde 1/4 a 3 vueltas/min. Su uso es general en agricultura. Dependiendo de la presión de trabajo se les deberá separar o juntar más a cada aspersor.

b) Según el mecanismo de giro:

-De reacción, la inclinación del orificio de salida origina el giro.
-De turbina, el chorro incide sobre la turbina que origina el giro.
-De choque, el chorro ataca sobre un brazo con un muelle y este hace girar al aspersor de forma intermitente. Este tipo de aspersor puede llevar un mecanismo especial que solamente lo haga girar un sector circular en lugar de completar el círculo completo.

c) Según la presión de trabajo:

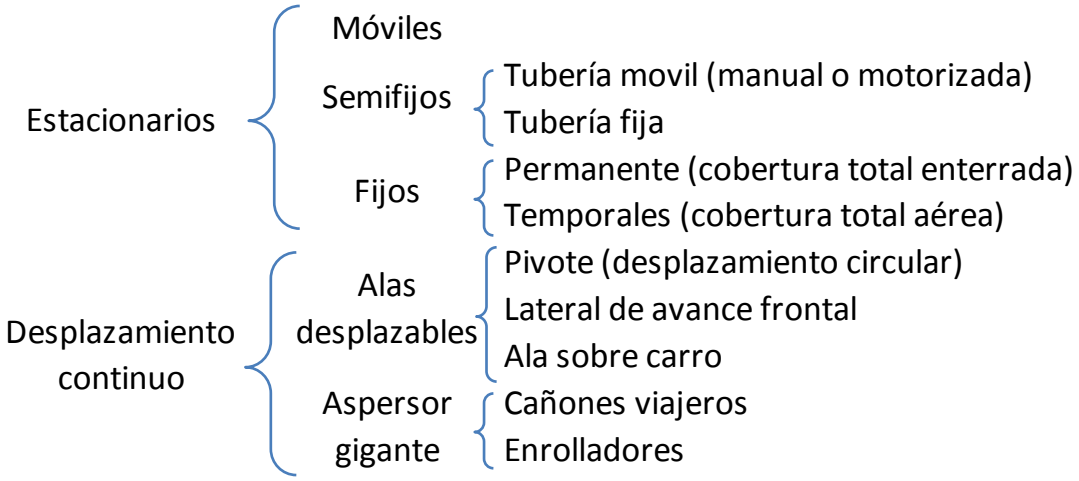
-De baja presión, cuando es menor a los 2kg/cm^2. Suelen tener mecanizados menores a los 4mm de diámetro en las boquillas y descargar flujos inferiores a 1m^3/h. Son adecuados para trabajar en parcelas rectangulares con separación entre aspersores del orden de los 12m o en disposición triangular con separaciones menores a los 15m. Su uso mayoritario es en el riego de frutales así el chorro de agua va por debajo de la cota de la copa de los árboles.

-De media presión, cuando la presión de trabajo se da en el rango de 2 a 4kg/cm^2. Los diámetros de las boquillas de estos aspersores están comprendidas entre 4 a 7mm y los caudales de descarga bordean el 1m^3/h a 6m^3/h. La separación adecuada entre aspersores de este tipo va desde los 12m hasta los 24m.

-De alta presión, es considerada para presiones mayores a los 4kg/cm^2. Suelen ser aspersores de gran tamaño llamados también cañones con 2 a 3 boquillas y caudales de descarga que llegan hasta los 40m^3/h. El mecanismo de giro por lo general es de choque o de turbina y tiene alcances de 25 a 70m.

1.3.2 Clasificación de los sistemas de riego por aspersión

El modo más cómodo de clasificar los sistemas de aspersión es en función de la movilidad de los elementos del sistema. Estos pueden agruparse en dos grandes bloques: los estacionarios, que permanecen fijos mientras riegan y los de desplazamiento continúo.



Dentro del primer bloque de los estacionarios están los sistemas móviles y es que todos los elementos de la instalación son móviles, incluyo el equipo de bombeo.

Luego están los sistemas semifijos que por lo general tienen fijas las siguientes partes: la estación de bombeo, la red de tuberías principales y los hidrantes donde se conectan las tuberías de alimentación. La parte móvil son los ramales de riego que llevan conectado como parte final los aspersores mediante el uso de mangueras lo cual permite colocarlos en diferentes posiciones y distancias.

Los sistemas estacionario fijos permanentes mantienen todos sus componentes fijos durante la vida útil, en cambio los sistemas fijos temporales se colocan al inicio de la campaña de riego y son retirados al finalizar la misma.

En cuanto a los de desplazamiento continuo, los pivotes son tendencia actual debido a que se prefieren los sistemas de baja presión que permitan el riego nocturno ya que hay menor evaporación, velocidad de viento y menor coste energético. Además son de fácil manejo y automatización.

Los laterales de avance frontal son muy adecuados para parcelas en forma rectangular y que abarcan una gran longitud, consiguen también una alta uniformidad en la distribución de la lámina de riego a baja presión pero requieren una mayor inversión y tienen un manejo más complicado que los pivotes.

Las alas sobre carro son sistemas con gran movilidad y adecuación a diferentes topografías de terrenos y manejan un amplio abanico de cultivos en los que pueden ser usados. Estos sistemas sustituyen en buena medida a los aspersores gigantes, por sus problemas de elevada presión de trabajo, tamaño de gota, distorsión por efecto del viento que los hacen únicamente adecuados para riego de socorro, riego de praderas, etc. (Tarjuelo Martín-Benito, 1991)

1.3.3 Ventajas e inconvenientes del riego por aspersión

De forma general se puede citar que el riego por aspersión tiene las siguientes ventajas (Macías Macías, Vergara Sabando, Macías Solórzano, & Bazurto Zambrano, 2011):

- ✓ El agua al caer en forma de lluvia permite y estimula la germinación.
- ✓ La distribución de agua es bastante uniforme.
- ✓ Puede ser utilizado en suelos que tengan una gran velocidad de infiltración.
- ✓ Su uso es adecuado en terrenos con pendientes moderadas.
- ✓ Se puede aplicar los fertilizantes junto con el agua de bombeo.
- ✓ Alta eficiencia en la aplicación de la lámina de riego lo cual lo hace muy beneficioso en regiones con limitaciones de agua.

- ✓ Se puede automatizar y por ende operar en condiciones óptimas la mecanización del sistema.

- ✓ Se puede trabajar durante horas de la noche donde el viento es menor y la evaporación también.

- ✓ Los sistemas pueden ser instalados con mayor rapidez por ser modulares.

- ✓ Permite aplicar normas de riego pequeñas y con mayor frecuencia.

- ✓ No interfiere en las labores de cosecha agrícola.

- ✓ Permite regar casi todo tipo de cultivo.

- ✓ El manejo de las gotas al momento de regar no erosiona ni destruye el terreno.

Tomando también en cuenta las desventajas o inconveniente tenemos (Macías Macías, Vergara Sabando, Macías Solórzano, & Bazurto Zambrano, 2011):

- ✓ Muy sensibles a las condiciones meteorológicas como el viento ya que distorsiona la uniformidad en la aplicación del agua de riego.

- ✓ El equipo de bombeo demandara un gasto energético cada vez que entre en operación, por tanto eleva los consumos y costos de riego.

- ✓ El costo de instalación y la inversión inicial es mayor que otro tipo de riego.

1.4 Riego por pivotes

El pivote de riego central forma parte de los sistemas de riego por aspersión mecanizados. Es un ramal de riego con un extremo fijo por donde recibe el agua y la energía eléctrica y otro extremo móvil que describe un círculo girando con respecto al primero. Los pivotes al momento de regar dejan una huella satelital en forma de círculo.

El equipo de riego se basa en el movimiento de una tubería portaemisores que se apoya en sus torres automotrices. Estas torres están dotadas de un motor y dos ruedas. La tubería portaemisores suele ser de acero galvanizado que junto con los cables y las barras de conexión entre torres sirve para darle una mayor resistencia al equipo.

En las figuras 1 y 2 se muestran las estructuras pivotantes, conformadas por torres las cuales como se observa se apoyan en dos ruedas y van conectadas entre ellas para alcanzar el radio de giro de diseño.

La separación entre torres está comprendida entre 35 a 75m, y la longitud acumulada puede llegar hasta los 800m. Por tanto un solo pivote puede irrigar áreas circulares de más de 200ha.

Normalmente los pivotes riegan círculos completos sin embargo hay situaciones en que, por la disposición y limitación catastrales de la zona solamente se riega un sector circular. (Macías Macías, Vergara Sabando, Macías Solórzano, & Bazurto Zambrano, 2011)

Cuando se instalan pivotes tangentes, a veces se opta por añadir al equipo un dispositivo llamado *corner* o de esquina. Este dispositivo es un alero articulado en la tubería portaemisores que se despliega y se pone en funcionamiento al pasar por las zonas de las esquinas y lograr regar áreas que en un principio se hubieran quedado sin regar.

Figura 1- Pivote operando en zona de riego

Fuente: Fotografía del proyecto Agrolmos

Figura 2- Pivote operando en cultivo

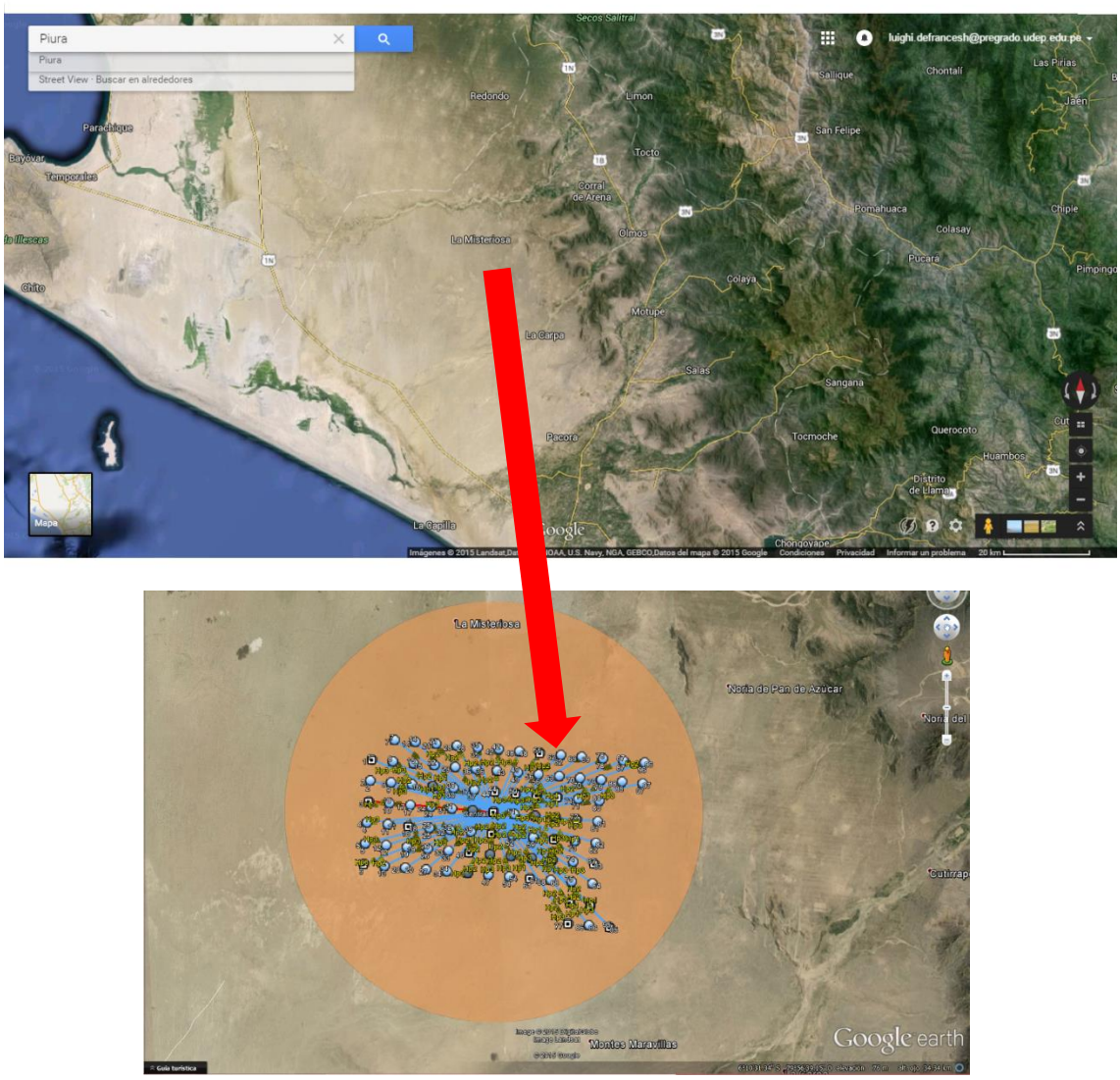
Fuente: Fotografía del proyecto Agrolmos

En las siguientes imágenes se muestra la huella satelital que deja este tipo de riego.

1.4.1 Proyecto Agrolmos - Lambayeque

Este proyecto se ha desarrollado en el transcurso de los años 2014 y 2015. En la figura 3 se puede apreciar la ubicación de estos equipos en la zona de Lambayeque junto con la disposición planificada para cada pivote.

Figura 3- Ubicación y arreglo de pivotes del proyecto Agrolmos



Fuente: Google Earth 2015

El proyecto Agrolmos en su primera etapa dispone de 92 pivotes (cada uno riega aproximadamente 95 ha) y se planea ampliar a 115 en una segunda etapa. Toda el área aprovechada será para el cultivo de caña de azúcar y comprende cerca de 10900 ha.

Los pivotes instalados son de la marca *Valley* - *Valmont Irrigation* y fueron importados desde Nebraska (EEUU). El principal aporte de la tesis para este proyecto es en la identificación del protocolo de comunicación que maneja cada panel electrónico de pivote. Al manejar el lenguaje en que se dan los comandos a cada estación, se da paso a la integración de un sistema automático de control y monitoreo del riego.

Estos terrenos forman parte del proyecto especial de Irrigación e Hidroenergético de Olmos, desarrollado en la región de Lambayeque. (Ver figura 4) Consiste en el trasvase de las aguas del río Huancabamba de la vertiente del Atlántico a la vertiente del Pacífico a

través de un túnel trasandino de 20 km para su aprovechamiento en la irrigación y generación de energía eléctrica.

Parte de la obras del componente de Trasvase incluye la Presa Limón de 43 m de altura, cuyo objetivo es crear un embalse para regular los caudales estacionales del río Huancabamba y derivar luego las aguas a través del túnel trasandino, el túnel trasandino fue la obra más difícil de Olmos, al estar debajo de la cordillera de los Andes. (Ordinola Enríquez, 2009)

Figura 4- Fotografía aérea tomada sobre la zona del proyecto Agrolmos

Fuente: Fotografía Abril 2015.

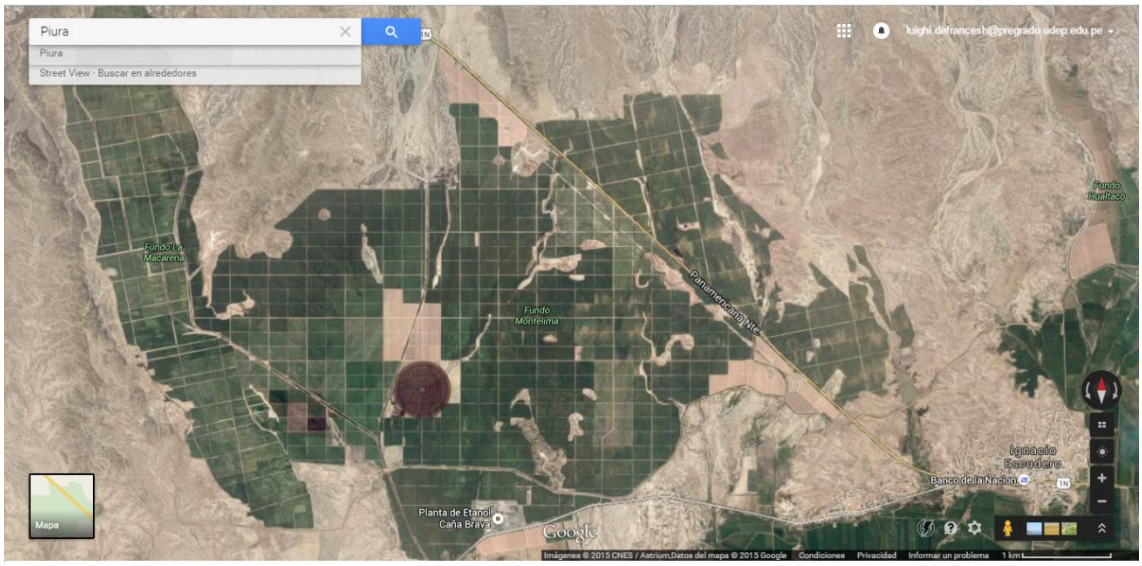
1.4.2 Pivote instalado en Caña Brava – Sullana

Es usado para el tratamiento y aprovechamiento de la vinaza, la cual es un efluente industrial del procesamiento de la caña de azúcar. La vinaza es un líquido de color marrón

oscuro, con alto grado de acidez. Por ello en la figura 5 se aprecia el círculo de ese color y también por la misma razón el pivote está revestido con polietileno. La vinaza proviene de las columnas de destilación con una temperatura próxima a los 100ºC. Para lograr un litro de etanol se obtienen también entre 10 a 15 litros de vinaza.

Es considerada como un desecho de composición variable dependiendo fuertemente del sistema de fermentación alcohólica, tipo de levadura, calidad de la caña de azúcar y del proceso de cristalización del azúcar. La posible eliminación de la vinaza es justamente el reciclaje de esta, hacia los campos de cultivo para sustituir total o parcialmente los fertilizantes. (Hernández, 2013)

Figura 5- Pivote instalado para el tratamiento de vinaza



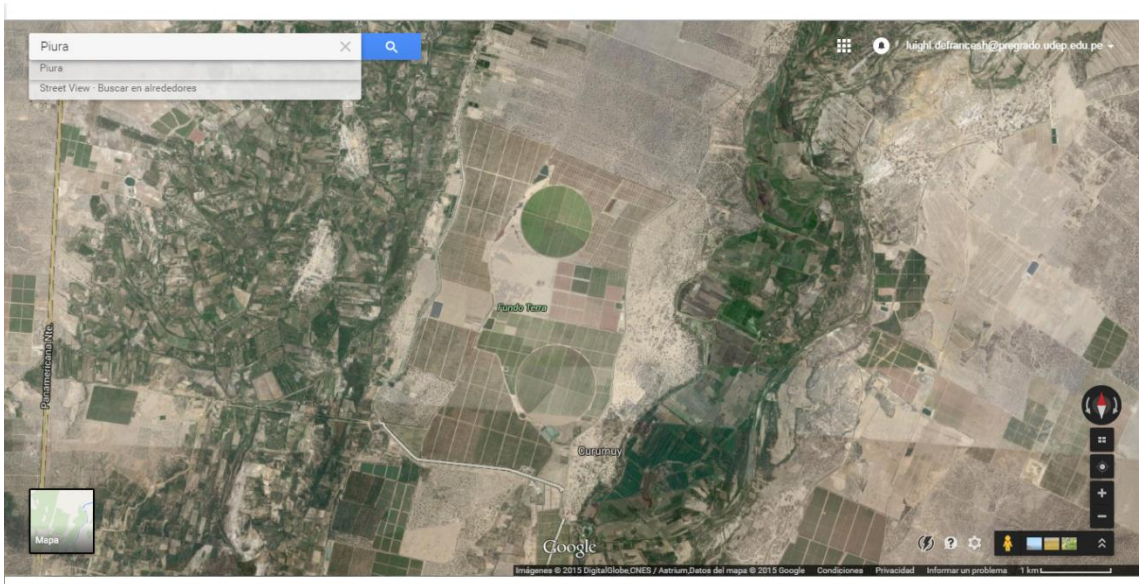
Fuente: Google Maps 2015

1.4.3 Pivotes instalados en la empresa Camposol – Piura

La empresa Camposol está involucrada en la cosecha, el proceso y la comercialización de productos agrícolas de alta calidad, tales como paltas, arándanos, espárragos, uvas, mangos, pimientos, alcachofas, mandarinas y granadas que son exportados frescos, en conserva y congelados a mercados clave en Europa, los Estados Unidos de América y Asia[4]. Tiene instalados dos pivotes que se pueden apreciar en la figura 6.

[4] http://www.camposol.com.pe/productos.html

Figura 6- Pivotes instalados en la empresa Camposol

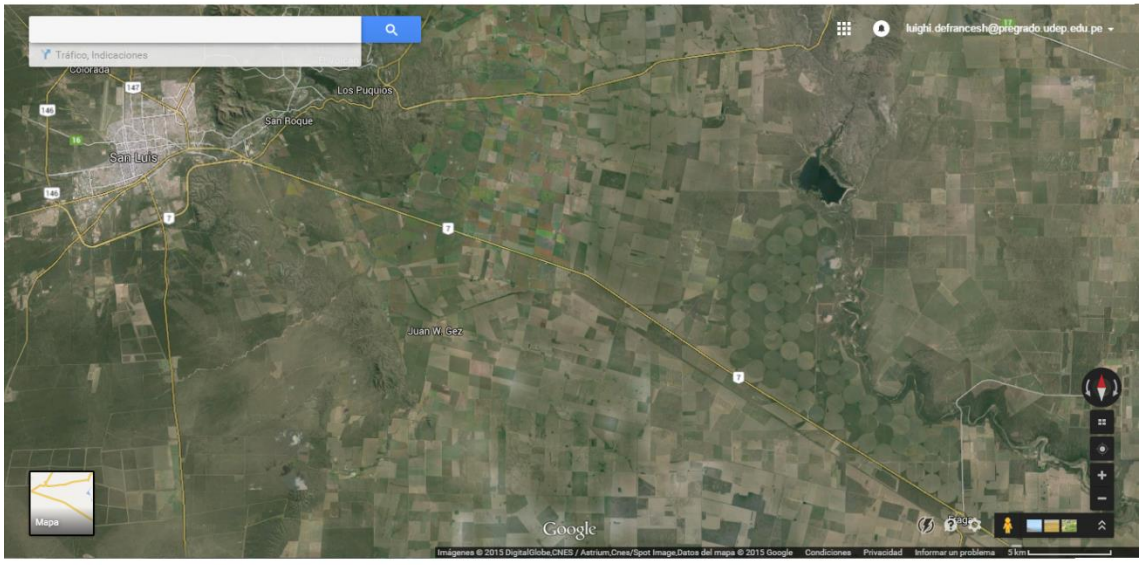


Fuente: Google Maps 2015

1.4.4 Pivotes instalados en la región de San Luis – Argentina

Este proyecto fue impulsado por la Sociedad Electrica Radice (SER) y contempla la instalación de 69 pivotes en la región de San Luis, Argentina. (Ver figura 7) El proyecto comprendió la irrigación de más de 6550 ha de cultivo y se desarrolló en el año 1998. Participaron de esta actividad profesores de la Universidad de Piura.

Figura 7- Pivotes de la región San Luis – Argentina

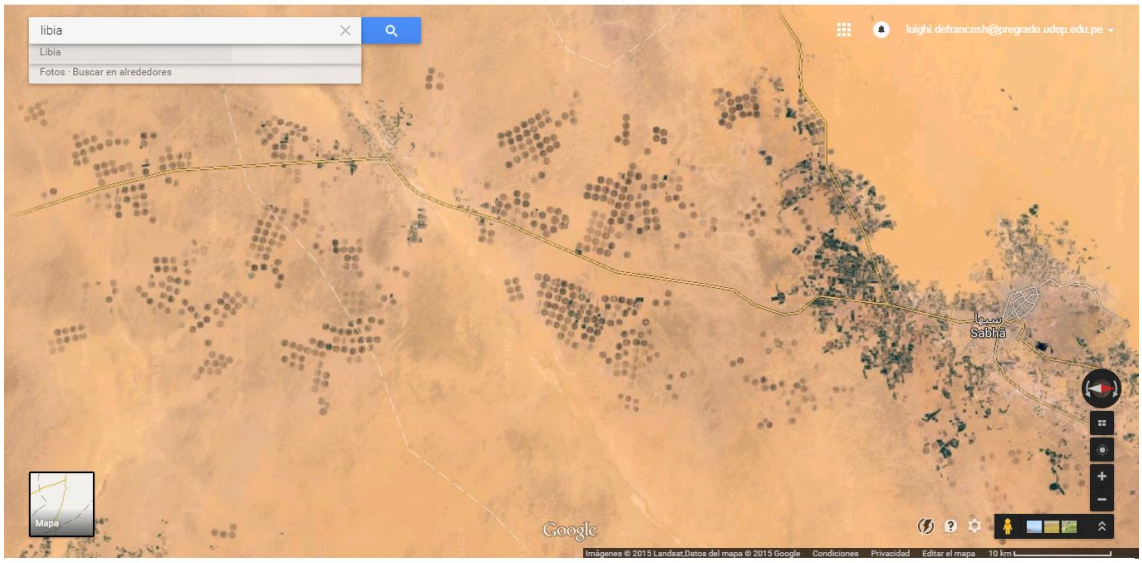


Fuente: Google Maps 2015

1.4.5 Pivotes instalados en los países de Libia y Jordania[5]

En un país como Libia, donde más del 95% del territorio lo compone el desierto del Sahara, este tipo de mecanización agrícola no es una inversión barata. Solo es posible aprovechando los depósitos de agua fósil de los acuíferos subterráneos. Cada parcela circular tiene más o menos un kilómetro de diámetro y en ellas se cultivan: grano, frutas y verduras, y forrajes para el ganado. (Ver figura 8)

Figura 8-Vista satelital de pivotes en el país de Libia



Fuente: Google Maps 2015

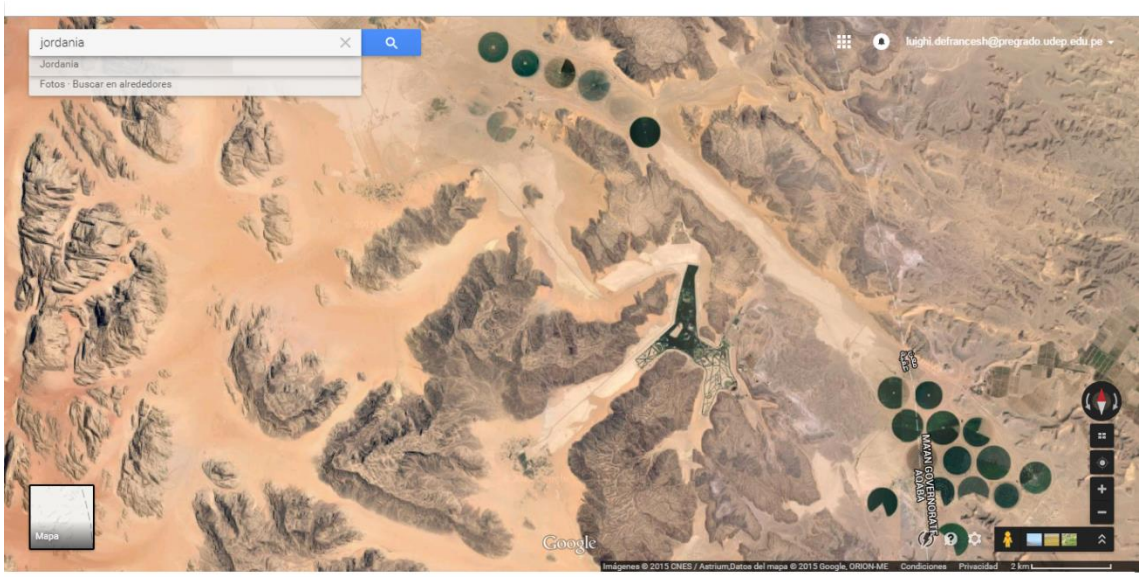
En cuanto a Jordania también ha invertido en pivotes de riego. (Ver figura 9) Este país dispone de una zona regada aproximada de 76.000 hectáreas. De ellas, 43.000 hectáreas son áreas de montaña y desérticas. Estos terrenos consumen el 65% del agua total de Jordania y el 53% del agua subterránea. Es uno de los países clasificados con recursos hídricos escasos; su demanda prácticamente excede el agua disponible. Sin embargo, la mayor parte de las tierras son cultivables si se hace llegar el agua a ellas.

Lo más importante a destacar es que el uso de pivotes de riego central va más por el alto coeficiente de uniformidad y distribución del agua, coeficientes que no se obtienen hasta el momento con otros métodos de riego. Se están cultivando con éxito jatrofa y jojoba para obtener biocombustibles. También se aprovecha las cosechas de colza, soja y girasol para la obtención de aceites.

Estos sistemas también se utilizan para aplicar aguas residuales, que gracias a su contenido en nitrógeno, micronutrientes y sustancias orgánicas enriquecen la tierra, permitiendo aprovechar el agua de desecho y convertirla en beneficio agrícola. Se satisfacen las necesidades de los habitantes y genera beneficios económicos para el país.

[5] http://www.traxco.es/blog/wp-content/uploads/2010/11/pivotes-de-riego-en-libia-y-jordania.pdf

Figura 9- Vista de pivotes en el país de Jordania



Fuente: Google Maps 2015

1.5 La irrigación en el Perú

Nuestro país es uno de los países con mayor diversidad de especies, de recursos genéticos, y de ecosistemas. Actualmente se encuentra entre los países más diversos del planeta. Toda esta riqueza de biodiversidad no fuera posible si no contáramos con los diferentes climas y altitudes, los diversos suelos y regiones; pero sobre todo, no fuera posible sin agua, el recurso más valioso sobre la tierra para que exista vida.

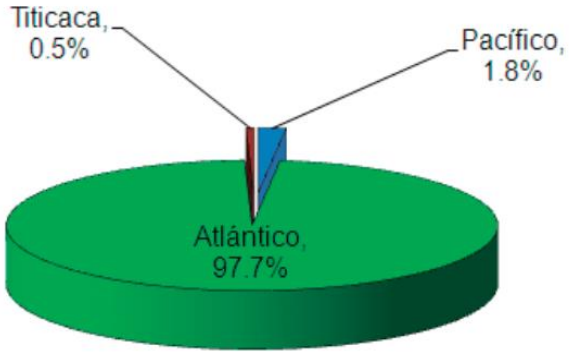
El Perú se ubica en zona tropical, su geografía se ve en gran parte influenciada por la cordillera de Los Andes y posee territorios amazónicos. En nuestro país se origina el río más caudaloso y grande del mundo. Estos son datos suficientes para dar una clara idea de la riqueza y las razones por las cuales poseemos gran cantidad de recursos hídricos.

Sin embargo, la ubicación de los principales focos de crecimiento poblacional no se ubican precisamente donde se dispone la mayor cantidad de agua. El Perú tiene una extensión de 1 285 215,60 Km^2, comprendida en tres regiones hidrográficas: Pacífico, Amazonas y Titicaca. Los recursos hídricos en el país, se encuentran distribuidos en 106 cuencas hidrográficas; 53 en el pacífico, 44 en el atlántico y 9 en el Titicaca[6].

En la figura 10 se muestra un gráfico circular donde se puede apreciar la proporción que tiene cada cuenca en nuestro país. Llama la atención el elevado valor que tiene la cuenca del atlántico en comparación con la del pacífico y Titicaca.

[6] Boletín Técnico: Recursos Hídricos del Perú en cifras. ANA (2010).

Figura 10- Disponibilidad hídrica por región hidrográfica



Fuente: Autoridad Nacional del Agua (ANA - OSNIRH)

Se cuenta con cerca de 12 200 lagunas en la Sierra y con más de 1007 ríos que tienen una disponibilidad que asciende a 2 046 287 Hm^3, de los cuales 1.82% provienen de la vertiente del pacífico, 97.68% del atlántico y apenas el 0.50% del Titicaca. (Ver Tabla 1)

Tabla 1- Distribución de la disponibilidad hídrica

REGION HIDROGRAFICA	SUPERFICIE (Km2)	POBLACION		DISPONIBILIDAD HIDRICA (Hm3)			
		HABITANTES	(%)	SUPERFICIAL	SUBTERRANEA	TOTAL	(%)
Pacífico	279 700	18 315 276	64.9	34 624	2739	37 363	1.8
Amazonas	958 500	8 579 112	30.4	1 998 752		1 998 752	97.7
Titicaca	47 000	1 326 376	4.7	10 172		10 172	0.5
TOTAL	**1 285 200**	**28 220 764**	**100**	**2 043 548**	**2739**	**2 046 287**	**100**

Fuente: Política y Estrategia de Recursos Hídricos del Perú - 2009

En el Perú, nuestra población no se encuentra proporcionalmente distribuida en relación a nuestros recursos hídricos; pues en la región hidrográfica del Pacífico, que constituye la franja de los valles de la costa, se encuentra el 65% de la población nacional.

Esta región dispone de muy poca cantidad de agua en comparación con la vertiente del Amazonas, donde se encuentran la mayor parte de nuestros ríos y vive tan solo el 30% de los habitantes. (Ver figura 11)

Para hacer frente a estas cifras, desde el año 1962, el Estado Peruano ha invertido grandes cantidades de dinero para incrementar la oferta de agua en la costa, con la finalidad de atender principalmente las necesidades del sector agrícola, que consume más del 80% de agua en el país.

Los resultados no han sido los esperados, pues la eficiencia de riego alcanza el 35% y los problemas de salinización de los suelos van en aumento. (Barreto Escobedo, 2015)

Según el último boletín técnico (2010) *Recursos Hídricos del Perú en Cifras* de la ANA, la demanda de uso de agua en el país obedece al siguiente orden. El sector agrícola utiliza más del 85% de los recursos hídricos, el uso poblacional cerca del 9%, el uso piscícola 1.90%, el uso industrial 0.72% y el uso minero 1.49%. (Ver figura 12)

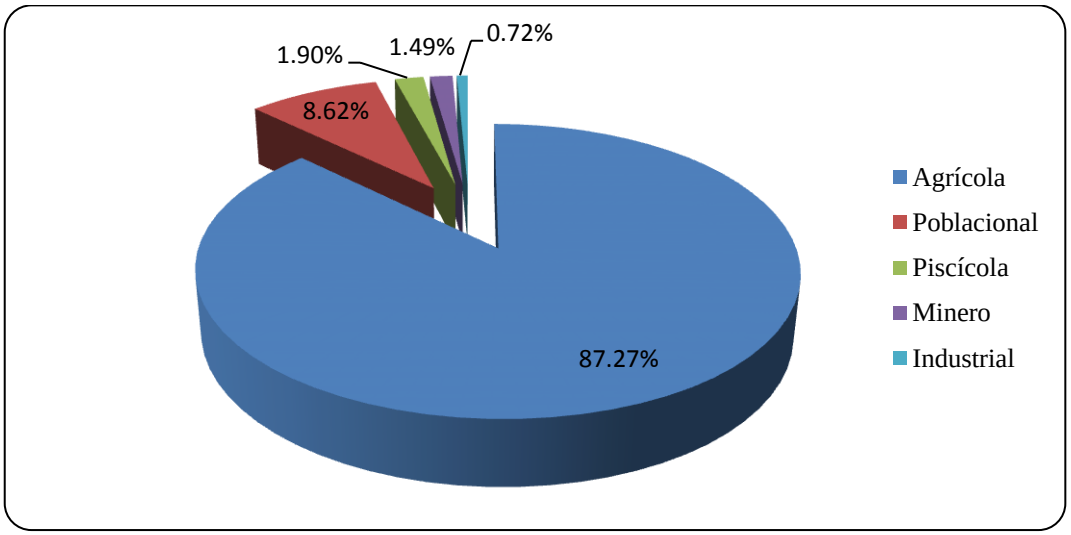
Figura 11- Disponibilidad de agua en el Perú

Fuente: Barreto Escobedo, D. (2015)

Teniendo presente estas cifras podemos darnos cuenta de la trascendencia que tiene el sector agrícola en el uso de agua del país. Por ello los Proyectos de Irrigación en Perú, no solo buscan ampliar la producción y terrenos agrícolas, sino sobre todo buscan aprovechar mejor y de manera muy eficiente el uso del agua.

Como podemos apreciar, el agua no se encuentra distribuida equitativamente en el país, tampoco se dispone de las mismas cantidades año tras año, y muchas veces no se dispone de la cantidad de agua en el momento oportuno (periodos de estiaje).

Figura 12- Demanda de uso de agua



Fuente: ANA

1.5.1 Necesidad de proyectos de irrigación en el Perú

En la costa peruana, la precipitación media anual varía de 20 a 50 mm, con estos valores no es posible el desarrollo de un cultivo. Entonces, podemos decir que la costa es un inmenso desierto que es surcada por algunos valles fértiles. La temperatura, que en promedio fluctúa entre 14 y 28°C, es la idónea para que crezcan los cultivos. Sin embargo, la escasez de los recursos hídricos hace que la costa requiera proyectos de Irrigación.

En la sierra, la precipitación aumenta relativamente desde 550 mm a 800 mm, pero es irregular y se presenta mayormente en los meses de diciembre a marzo, por lo cual también requiere de proyectos de riego, dado que la mayor parte de sus áreas son exclusivamente de secano[7].

La selva se caracteriza por una mayor precipitación, varía entre los 900 y 3300 mm anuales y las lluvias son mucho más intensas. En la selva existen partes altas donde se asientan los pueblos, mientras que en las depresiones y las zonas inundables no es posible cultivar.

En resumen, es una necesidad para el Perú contar con proyectos de Irrigación y sistemas de riego eficientes que permitan desarrollar aún más la agricultura nacional para un mayor aprovechamiento y control de nuestros recursos hídricos. A su vez estos proyectos pueden incorporar generación hidroeléctrica, piscicultura, y plantas de tratamiento de aguas para el abastecimiento de los poblados.

1.5.2 Principales proyectos por regiones

El Gobierno Peruano está llevando a cabo varios programas que tienen como objetivo hacer frente a los desafíos clave del sector riego, incluyendo: el deterioro de la calidad del agua, poca eficiencia de los sistemas de riego y drenaje, cambio climático, incluidas condiciones climáticas extremas y retroceso de los glaciares. (Tealdo Alberti, 1995)

[7]La agricultura de secano es aquella en la que el ser humano no contribuye con agua, sino que utiliza únicamente la que proviene de la lluvia.

Las inversiones en los principales proyectos de irrigación tienen básicamente tres componentes.

- Las inversiones relacionadas a la incorporación de tierras agrícolas.
- Las inversiones relacionadas al mejoramiento del riego.
- Las inversiones relacionadas a la generación de energía eléctrica.

Con estos proyectos de irrigación, se logran regar terrenos que son capaces de ser rentables y sostenibles en la actividad agraria, involucrando aspectos sociales, políticos y económicos. (Barreto Escobedo, 2015)

Los sistemas de riego en estos proyectos son la infraestructura que hace que grandes áreas peruanas puedan ser cultivadas con la aplicación del agua necesaria.

La mayoría de los grandes proyectos de irrigación se desarrollan por etapas, es decir, debido al gran alcance y costo que presentan estos proyectos, es conveniente ejecutarlos paso a paso, asegurándose así un avance y crecimiento para los próximos años. Muchos de los proyectos en la costa del Perú, ya se encuentran en su última etapa. (Ver figura 13)

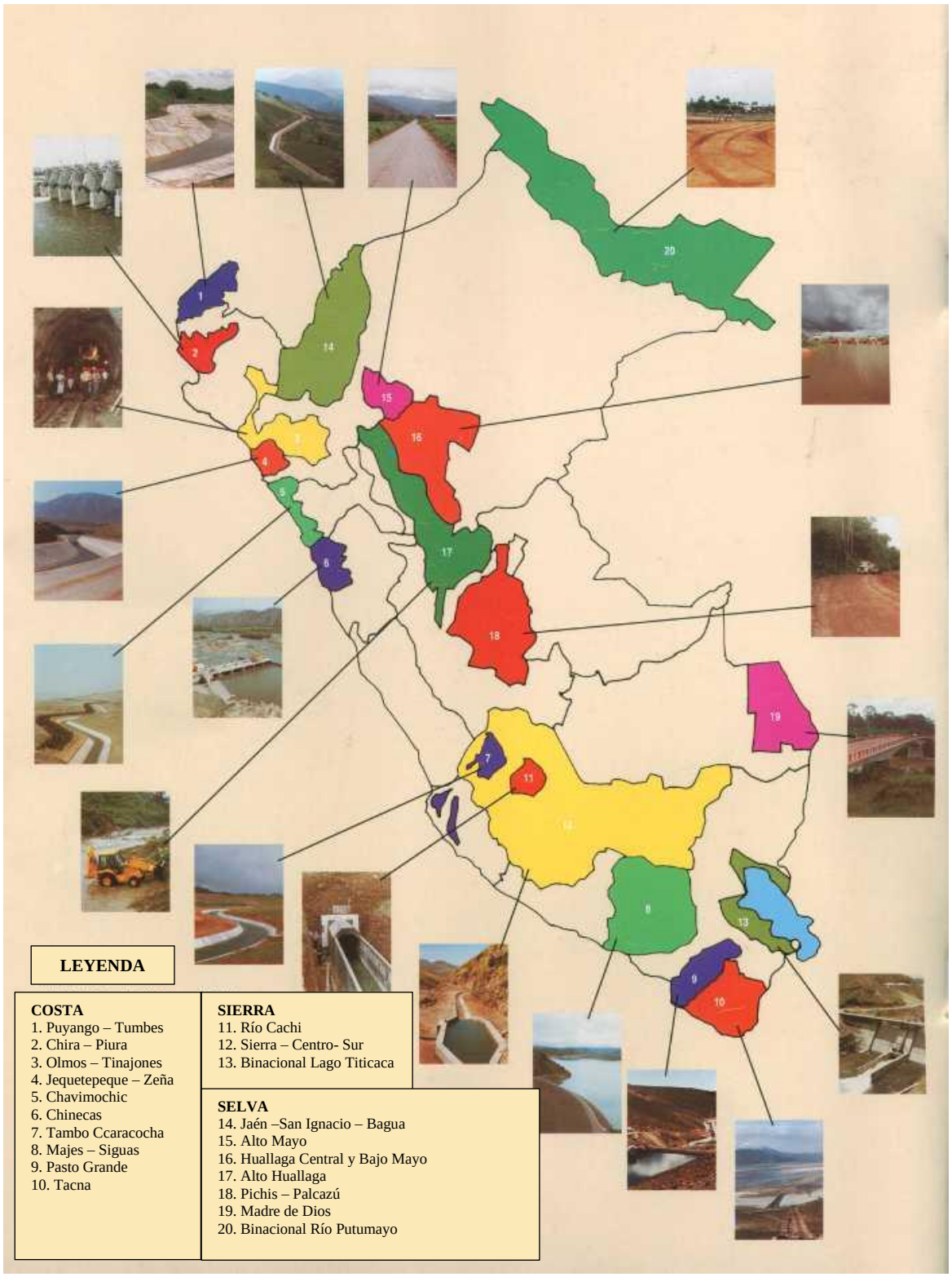
En la tabla 2 se muestra algunos de los proyectos junto con el área total que abarcan, las familias que serán beneficiadas y el costo final por hectárea habilitada.

Tabla 2- Parámetros globales de los principales proyectos hidráulicos de la Costa

Proyecto	Área total (Ha.)	Familias Beneficiadas	Costo por Ha.(US $)
1. Puyango - Tumbes	48 000	25 000	15 600
2. Chira – Piura	115 600	33 000	5 600
3. Alto Piura	42 000	18 000	7 900
4. Olmos	112 000	25 000	14 700
5. Tinajones	100 000	30 000	5 350
6. Jequetepeque - Zaña	66 000	20 000	4 800
7. Chavimochic	132 000	50 000	6 150
8. Chinecas	52 000	10 000	7 550
9. Majes	57 000	12 000	24 400
10.Tacna - Moquegua	42 200	10 000	14 350
	766 800	233 000	9 700
	(TOTAL)	(TOTAL)	(PROM.)

Fuente: INADE

Figura 13- Proyectos por regiones



Fuente: INADE, construyendo el futuro del Perú. Lima (1994)

1.5.3 Gestión de la oferta y la demanda: caso de los proyectos especiales en la Costa del Perú

El problema de la agricultura está ligado al problema del riego y éste a su vez, con el problema del mal manejo del agua. Se pensaba que la escasez del agua quedaba solucionada con el incremento de grandes volúmenes de agua a través de la construcción de proyectos de irrigación.

Sin embargo, ya se ha visto que los resultados han sido poco favorables. La disponibilidad adicional de agua obtenida con los proyectos de irrigación, no ha sido bien aprovechada en el sector agrícola, el que consume más del 80% de la oferta de agua con una eficiencia de 35%, en promedio. En algunos valles hay un uso desmedido del agua. Por tanto, de nada sirve mejorar el recurso hídrico, si se va a continuar utilizando módulos de riego elevados, sembrando cultivos exigentes en agua y con serias deficiencias en el manejo del recurso. Aunque con la construcción de los proyectos especiales se atienda con agua a muchas personas, no se puede continuar desarrollando proyectos para luego desperdiciar el agua por no aplicar la tecnología adecuada de riego. (Barreto Escobedo, 2015)

1.6 El agua y la seguridad alimentaria de un país

Un acceso fiable al agua asegura un rendimiento agrícola por ende asegura la producción de alimentos que a su vez reembolsa en un ingreso monetario a las personas volcadas a esta actividad. En las zonas rurales en que su actividad principal es la agricultura, la disponibilidad, variabilidad y uso de este recurso determina en gran medida su economía.

Una inseguridad alimentaria puede producirse por acceso irregular al agua, por inundaciones o por grandes diferencias estacionales entre sequias y lluvias las cuales puedes producir escasez de alimentos a corto plazo y una inseguridad a largo plazo. Aquellos países donde tienen mejor acceso al agua son también los países con menor índice de desnutrición y pobrezas en menor nivel.

La introducción de tecnología a los proyectos de irrigación se justifica plenamente por cuanto mejora eficientemente la aplicación del agua con el beneficio de mejorar la producción de los cultivos. Además con la ejecución de estos proyectos no solo se está aportando con la solución del problema del uso adecuado del agua. También sirve de ejemplo para que se implementen este tipo de sistema que no solo beneficia a los cultivos sino a la economía y a la seguridad alimentaria de las familias. (Macías Macías, Vergara Sabando, Macías Solórzano, & Bazurto Zambrano, 2011)

Capítulo 2

Análisis de un sistema de comunicación integrado aplicado al riego por pivotes

Los sistemas integrados amplían la automatización.
Revista ABB 2/2006

La electrónica le ha dado un gran avance a los dispositivos de campo, por ejemplo ahora tienen capacidad para comunicación y procesamiento local de señales. Estas características permiten integrarlos en un control de procesos, también hacen más fiable y rentable el sistema de producción.

A nivel digital la incorporación de equipos en el nivel más bajo, es decir, el de campo conlleva a varias situaciones. La más importante es el aumento de datos. El aumento de datos a su vez requiere un orden para su almacenamiento, un estándar de tamaño de memoria, un protocolo de trasferencia, un modo de visualización y en otras situaciones mucho más delicadas requieren también se procesadas dentro un *tiempo crítico*[8]. (Oquelis Cabredo, 1997)

La integración de equipos con protocolos de comunicación independientes es limitada y en muchos casos muy costosa. En el presente capítulo se evaluara la solución para este tipo de problemas y su análisis económico.

2.1 Requisitos de los sistemas de comunicación en la industria

En la industria existe niveles para el tratamiento de los datos y cada nivel exige prestaciones diferentes. Por ejemplo, al nivel de gestión (nivel más alto de un sistema automatizado) la cantidad de datos tratados a través del bus informático es enorme (del orden los *megabytes*), sin embargo no requieren o no dañan nada si el tiempo de procesamiento es del orden de los minutos u horas.

[8] El tiempo crítico es el intervalo de tiempo, dentro del cual uno o más acciones deben ser completadas con un nivel de certeza definido. Una demora superior al tiempo crítico en el sistema específico, conlleva el riesgo de fallo con probables accidentes para los equipos, la planta e incluso para las personas.

Si seguimos descendiendo en los niveles de comunicación llegamos al nivel de campo o planta (el más bajo), aquí los requisitos son diferentes. Por ejemplo, el tamaño de los mensajes es más reducido pero debe ser procesado a tiempos mucho más acelerados.

La sincronización de equipos como actuadores y sensores requiere respuestas del orden de los milisegundos. En el nivel de gestión se usa para el intercambio de información estándares como *Ethernet* con TCP/IP con toda una normativa ya bastante estudiada y que está completamente definida en todos los niveles OSI (*Open System Interconnection*).

Sin embargo no sucede lo mismo con el nivel de campo debido a que existe un gran número de estándares, impuestos por cada fabricante, en el cual cada uno trabaja con protocolos individuales. Lograr la integración de estos dos grandes grupos es una ardua tarea.

2.2 Modelo OSI para la infraestructura de comunicaciones

El modelo OSI es un referente para la implementación de un sistema de intercambios de datos. La infraestructura que se plantea puede ser usada en muchas aplicaciones en especial, a las que se necesita ingresar a una gran base de datos. El modelo OSI enfatiza mucho en la exactitud de la respuesta ante una solicitud de información. Son siete niveles o capas en los que se basa el modelo OSI (Ver Tabla 3).

Los niveles 1 y 2 proporcionan el transporte de datos básicos para una red simple. Los niveles 3 y 4 extienden estas funciones para una red que está compuesta por varias redes simples con diferentes propiedades. Los niveles 5 y 6 proporcionan un marco de trabajo para enlazar y negociar las comunicaciones orientadas al usuario. Por último el nivel 7 da los medios de decodificación de estas comunicaciones.

Tabla 3- Capas de referencia OSI

Capa 7	Capa de aplicación
Capa 6	Capa de presentación
Capa 5	Capa de sesión
Capa 4	Capa de transporte
Capa 3	Capa de red
Capa 2	Capa de enlace
Capa 1	Capa física

Este modelo sirve para identificar de manera genérica los grupos funcionales de un sistema de comunicación. No es necesario tampoco que todos los sistemas posean estas siete capas. Depende de la funcionalidad y alcance que ofrezca el sistema para que algunas capas intermedias puedan obviarse.

2.2.1 Nivel 1: Capa física

Este nivel define las características físicas de la parte de comunicación de un circuito. En este nivel se definen los niveles eléctricos, las corrientes, el tipo de conector, el sistema de codificación eléctrica u óptica de los *bits* durante la transmisión, etc.

2.2.2 Nivel 2: Capa de enlace

Define el formato de las tramas de comunicación, sintaxis y su codificación lógica, así como los métodos de comprobación de la integridad del mensaje. Por ejemplo el cálculo del CRC en una trama Modbus.

2.2.3 Nivel 3: Capa de red

En este nivel se describen las características topológicas de la red, su fin es garantizar que las tramas de información sean encaminadas de acuerdo con el recorrido elegido, desde la fuente al consumidor. Cuando un recorrido se encuentra activo y a disposición simultáneamente con otros posibles caminos, este nivel se torna indispensable para enrutar correctamente la información.

2.2.4 Nivel 4: Capa de transporte

Este nivel define el modo de proceder a la conexión y a la desconexión de dos o más usuarios, así como en el modo de establecer un canal de comunicación entre ellos a través de la red de comunicación.

2.2.5 Nivel 5: Capa de sesión

Una vez que se logra establecer la conexión lógica entre dos puntos en la red, recién pueden sincronizar sus aplicaciones y así abrir una sesión de trabajo o dar inicio a sus ciclos de petición- respuesta.

2.2.6 Nivel 6: Capa de presentación

Esta capa es muy importante cuando dos sistemas tienen modos diferentes de la recepción y envío de datos. Por ejemplo si se está usando un modo ASCII o un protocolo diferente con cantidad de *bits* diferentes también, esta capa tiene como fin establecer la conversión y correspondencia entre los datos.

2.2.7 Nivel 7: Capa de aplicación

Este nivel describe la interfaz hacia los programas de aplicación, como ejemplo tenemos la transferencia de ficheros, directorios de ficheros, etc. Aquí se incluye también:

- La forma de los inicios y finalizaciones de las tramas de comunicación.
- Los medios para proporcionar transacciones remotas que sean fiables.
- Lenguajes estándares para las aplicaciones específicas y cooperantes para la comunicación de información.

2.3 Limitaciones de las técnicas de integración

En una planta industrial, existen dispositivos de control de fabricantes diferentes. Esto origina que se formen ‘islas’ a nivel de comunicación con los demás procesos. Para los lazos de control independientes les es imposibles acceder o proporcionar datos a otros equipos que no comparten el mismo protocolo.

Para este tipo de casos existen un sinnúmero de ejemplos, por ello han surgido diversas soluciones para la integración, pero siempre con algunas limitaciones.

- El tener protocolos de comunicación diferentes en el mismo nivel implica el uso adicional de hardware.
- La integración se torna mucho más complicada cuando se quiere supervisar, gestionar o comandar sistemas que tengan gran cantidad de instrumentación instalada en campo.
- Las soluciones no llegan a ser tan versátiles y flexibles como se desean lograr.
- La decodificación de datos para envió de información de un equipo a otro agrega un tiempo considerable. Si la variable de trabajo tenía restricción de tiempo crítico va ser mucho más probable que no llegue a cumplir ese requisito.
- Si el sistema SCADA administra varios tipos de protocolos, aumenta el retardo de la presentación de datos y también se complica la estimación de este tipo de retardo.
- Si los sistemas a integrar distan mucho geográficamente (varios cientos de metros), la conexión física se vuelve inviable pues la integración suele darse en estándar RS-232 o RS-485 y a esas distancias ya no es factible.
- La configuración remota y adición de equipos implica también cambios en los direccionamientos de memoria y ajustes en los algoritmos de correspondencia, por ello no es tan simple la integración de elementos no planificados debidamente.

Las limitaciones antes mencionadas son a groso modo las más genéricas y concurrentes para la integración de sistemas a nivel de comunicación. (Oquelis Cabredo, 1997)

En la presente tesis se desarrolla el modo de migrar un protocolo propietario (*Base Station 2*) al protocolo *Modbus RTU* adicionando una interfaz gráfica para el monitoreo de riego con pivote central.

2.4 Protocolo de comunicación *Base Station 2*

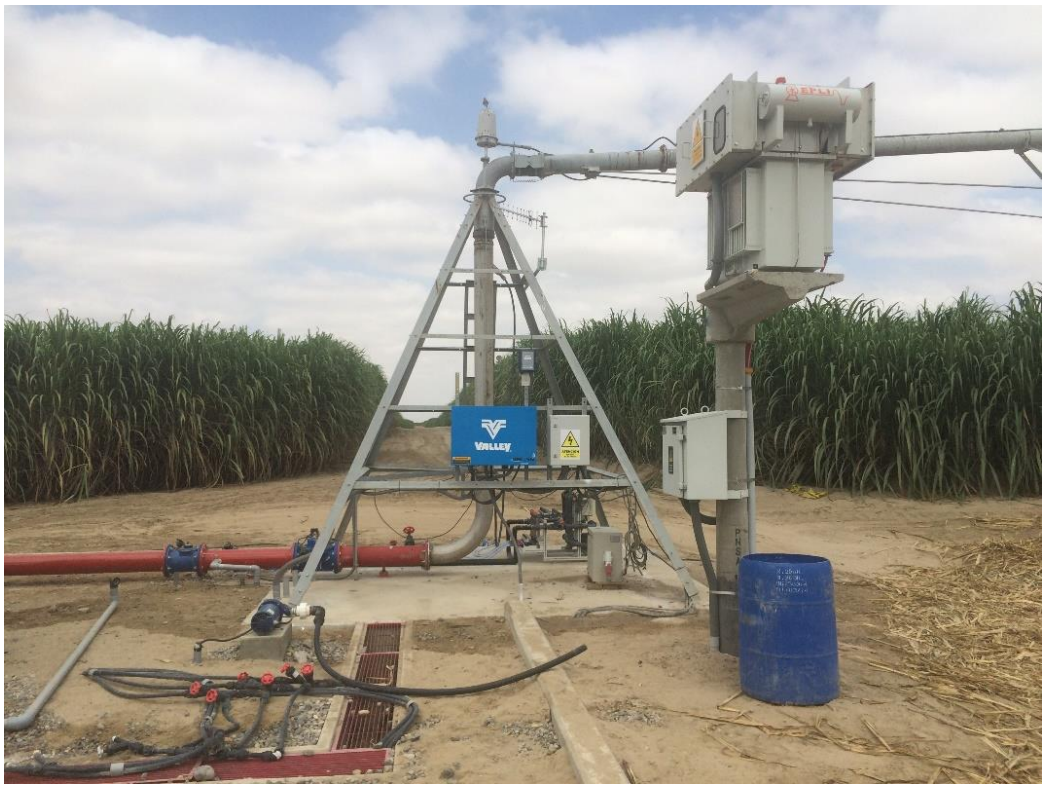
Dentro del marco de la tesis, hay un análisis de un protocolo individual para después ser insertado en otro genérico mediante el uso de PLCs. El protocolo del *software Base Station 2* es el formato que usan los paneles de riego Valley. Para este proyecto se plantea hacerlo compatible con los estándares de comunicación con el fin de integrarlos en una sola red.

En las figuras 14 y 15 se muestran unas fotografías de las instalaciones de pivotes en el proyecto Agrolmos. Se puede apreciar en ambas vistas el tablero de comandos Valley y los equipos electromecánicos asociados al funcionamiento del riego.

En la figura 14 el cultivo de caña de azúcar ya se encuentra regularmente avante, mientras que en la figura 15 aún está en una temprana etapa de crecimiento.

La señal que comunica el centro de control con cada pivote se transmite por radio, el mensaje lleva la trama de datos que debe de decodificar cada PLC. Se le indica al lector que el PLC instalado no es el más sofisticado ni el que más prestaciones tenga ya que la carga de comunicaciones se trabaja individualmente en un módulo especial agregado al slot de PLC y es el menor en su gama pues es suficiente para el tipo de tarea.

Figura 14- Vista del eje de unos de los pivotes instalados en Olmos



Fuente: Captura de cámara

Figura 15- Centro del radio de giro del pivote

Fuente: Captura de cámara

2.4.1 Descripción del protocolo *BS-2*

Cada mensaje en este protocolo tiene seis componentes. El primer componente es el carácter '('. Sirve para iniciar cualquier tipo de trama sea de petición o respuesta. Si un mensaje es iniciado con otro carácter simplemente es descartado del protocolo.

El segundo componente es quien lo envía, para indicar este dato se reservan tres caracteres.

El tercer componente es hacia quien va dirigido el mensaje, también se separan tres caracteres para esta información.

El cuarto componente es el cuerpo del mensaje, este componente lleva la información necesaria para que el panel realice una acción determinada. Si se desean llevar a cabo más de una acción, se separa por el carácter ';'. El detalle de las funciones que se colocan en este cuarto componente se explicará con un ejemplo más adelante.

El quinto componente es el *checksum,* este componente está formado por dos caracteres que son el resultado de un algoritmo de comprobación de la trama.

El sexto y último componente es la finalización del mensaje con el carácter *carriage return,* que en formato ASCII corresponde a '0D'. Este componente se agrega al mensaje original después del *checksum.*

Esta es la manera de construir un mensaje, ahora se explicará el modo de decodificación del mensaje, pues el receptor deberá también revisar el contenido. Cuando se recibe el mensaje, primero se extraen los caracteres hasta el cuarto componente. Se calcula nuevamente el *checksum* y se verifica que coincida el valor recibido, con el calculado. Si es así se procede con la acción indicada.

Para responder a la solicitud, el receptor debe enviar un mensaje al maestro, indicándole que ha llevado con éxito la acción solicitada. Este mensaje se envía por segunda vez si es que no hubo respuesta de confirmación en la primera.

La confirmación usa los caracteres 'AK', a continuación se agrega el valor del *checksum* y se crea por tanto una trama que tiene la misma sintaxis que un mensaje normal. En el siguiente ejemplo, el lector podrá observar la acción *activar agua* enviada desde la estación maestro a un pivote con numero de nodo 999.

Petición:

(000999POW59<*CR*

Primer componente: (

Segundo componente: 000

Tercer componente: 999

Cuarto componente: POW

Quinto componente: 59

Sexto componente: <*CR*

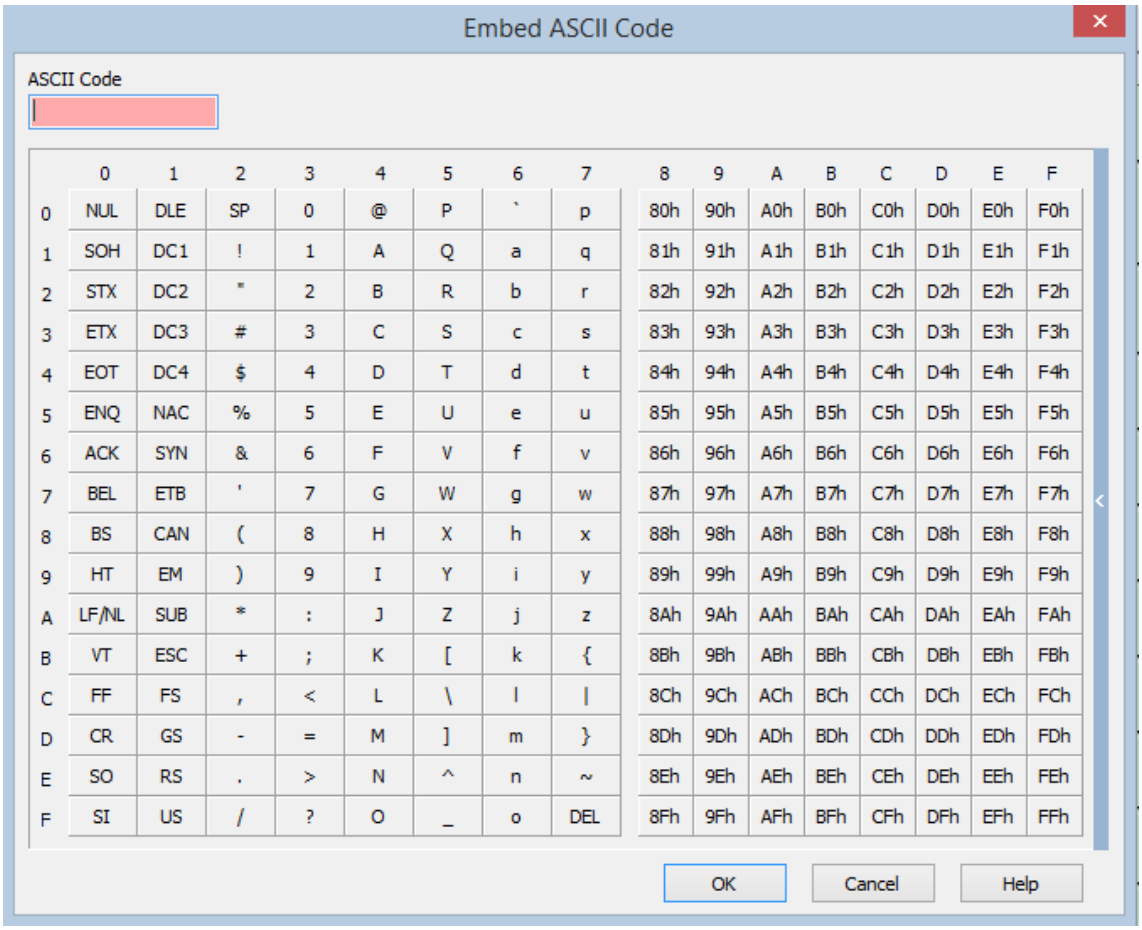
La correspondencia de valores ASCII (Ver Tabla 4) es muy utilizada para los algoritmos.

Respuesta:

(999000AK595D<*CR*

El reconocimiento de la función se ha realizado con éxito dado que los caracteres *AK* están acompañados del valor del *checksum* de la trama anterior.

Tabla 4- Valores hexadecimales del código ASCII



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DLE	SP	0	@	P	`	p	80h	90h	A0h	B0h	C0h	D0h	E0h	F0h
1	SOH	DC1	!	1	A	Q	a	q	81h	91h	A1h	B1h	C1h	D1h	E1h	F1h
2	STX	DC2	"	2	B	R	b	r	82h	92h	A2h	B2h	C2h	D2h	E2h	F2h
3	ETX	DC3	#	3	C	S	c	s	83h	93h	A3h	B3h	C3h	D3h	E3h	F3h
4	EOT	DC4	$	4	D	T	d	t	84h	94h	A4h	B4h	C4h	D4h	E4h	F4h
5	ENQ	NAC	%	5	E	U	e	u	85h	95h	A5h	B5h	C5h	D5h	E5h	F5h
6	ACK	SYN	&	6	F	V	f	v	86h	96h	A6h	B6h	C6h	D6h	E6h	F6h
7	BEL	ETB	'	7	G	W	g	w	87h	97h	A7h	B7h	C7h	D7h	E7h	F7h
8	BS	CAN	(	8	H	X	h	x	88h	98h	A8h	B8h	C8h	D8h	E8h	F8h
9	HT	EM	)	9	I	Y	i	y	89h	99h	A9h	B9h	C9h	D9h	E9h	F9h
A	LF/NL	SUB	*	:	J	Z	j	z	8Ah	9Ah	AAh	BAh	CAh	DAh	EAh	FAh
B	VT	ESC	+	;	K	[	k	{	8Bh	9Bh	ABh	BBh	CBh	DBh	EBh	FBh
C	FF	FS	,	<	L	\	l	|	8Ch	9Ch	ACh	BCh	CCh	DCh	ECh	FCh
D	CR	GS	-	=	M	]	m	}	8Dh	9Dh	ADh	BDh	CDh	DDh	EDh	FDh
E	SO	RS	.	>	N	^	n	~	8Eh	9Eh	AEh	BEh	CEh	DEh	EEh	FEh
F	SI	US	/	?	O	_	o	DEL	8Fh	9Fh	AFh	BFh	CFh	DFh	EFh	FFh

Fuente: Software Click Programming – Embed ASCII Code.

2.4.2 Funciones asociadas al protocolo *BS-2*

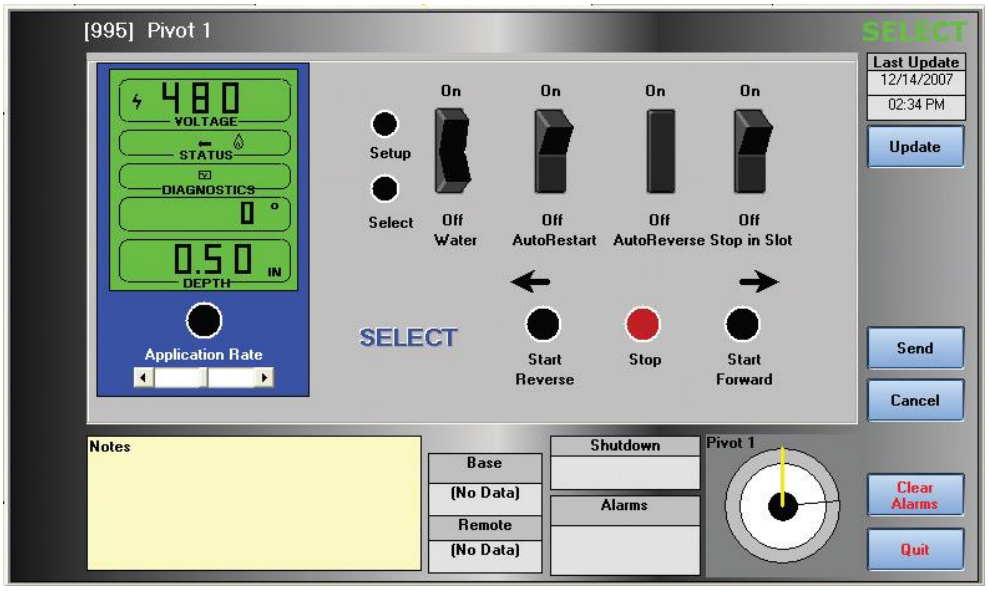
Las funciones asociadas al panel de riego y que se han estudiado en la presente tesis pueden visualizarse en la figura 16. La principal herramienta para actualización de datos es la función reporte del sistema, también están las funciones activar o desactivar la inyección de agua, iniciar riego en sentido horario u anti horario, variar la lámina de riego, etc.

Todos los ensayos se realizaron en el laboratorio de electrónica de la Universidad de Piura con el panel de riego Select Valley. (Ver figura 16)

Después se utilizó el simulador de pivote Valley PRO2, el cual se usó para la verificación de comandos en los tableros de los pivotes de Agrolmos. (Ver figura 17)

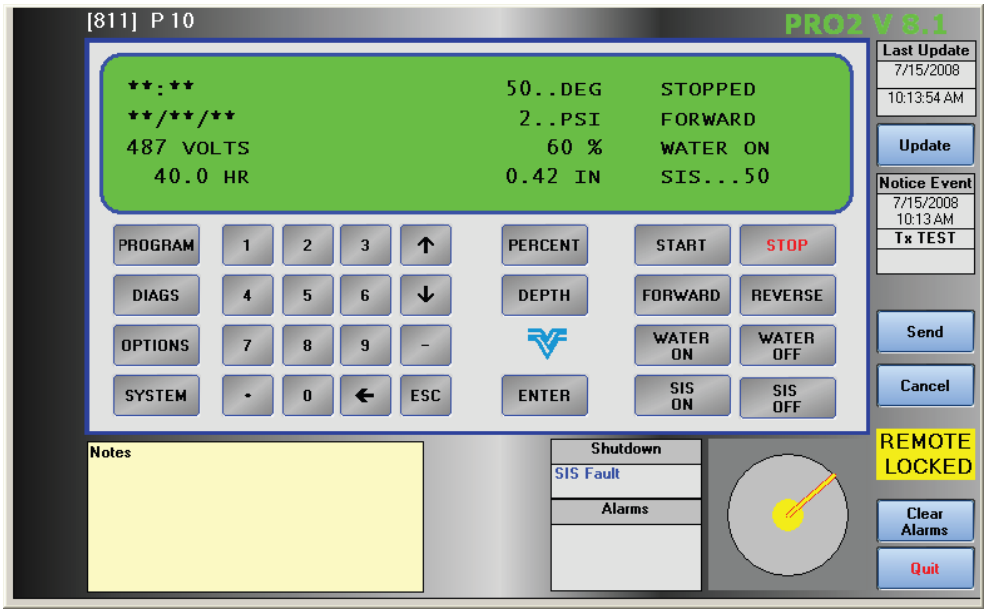
Los comandos que se han ejecutado vía digital y que han sido implementados en el algoritmo del PLC son: El reporte del estado del sistema, dirección actual, variar lámina de riego, reiniciar sistema, activar y desactivar flujo de agua, detener sistema. A continuación se explica la trama de comunicación utilizando el comando asociado a cada función.

Figura 16- Panel Select Valley



Fuente: Base Station 2-Guía del usuario.

Figura 17- Panel Valley PRO 2



Fuente: Base Station 2-Guía del usuario.

1. **(000999SD;SA;RE9B<*CR***
   Este comando solicita tres funciones en la misma oración. Primero SD (*System Direction*) el cual pide la dirección actual, esta puede ser *Forward* o *Reverse*. En el cuerpo del mensaje de la respuesta retorna una F o una R dependiendo del caso. Segundo SA (*Set Application Rate*), con esta función se pide el valor actual de la lámina de riego. Tercero RE (*Report System Status*) con este comando se puede recoger los datos de voltaje, ángulo del pivote, porcentaje de la lámina y presión.

2. **(000999SDF;RS20<*CR***
   Bajo esta trama, el panel de riego prepara al pivote para que en el siguiente inicio de riego se de en sentido *Forward.* El segundo, RS (*Restart System*) es el comando que ya ejecuta la acción.

3. **(000999SDR;RS2C<*CR***
   Bajo esta trama, el panel de riego prepara al pivote para que en el siguiente inicio de riego se de en sentido *Reverse.* El segundo, RS (*Restart System*) es el comando que ya ejecuta la acción.

4. **(000999SO05<*CR***
   La función SO (*System Off*), detiene al sistema. En la interfaz gráfica desarrollada en Matlab este comando está asociado al *stop* del pivote.

5. **(000999POW59<*CR***
   La función PO (*Pressure Override*) puede ser forzada con la activación o desactivación del flujo de agua en el cañón del pivote. Este comando representa el *water on.*

6. **(000999POD46<*CR***
   Bajo esta trama se realiza la función *water off*, o la desactivación del flujo de agua en el cañón del pivote.

7. **(000999SP2068<*CR***
   Las tramas con la función SP (*Set Percent*) modifican el valor de la lámina de riego, el ejemplo 7 y 8 son tramas que fuerzan a cambiar el valor a 20 y 70% respectivamente.

8. **(000999SP706D<*CR***
   Esta trama cambia el valor del porcentaje de la lámina de riego al 70%. Como se explicará más adelante existe una relación entre los milímetros de agua a regar con el valor del porcentaje. El archivo que ejecuta el PLC reserva 100 espacios de memoria para variaciones de uno en uno del valor del porcentaje.

2.5 Evaluación económica de la integración de protocolos

Un aspecto fundamental para que un proyecto sea viable es que sea económicamente atractivo. Cuando existen varias tecnologías que se pueden usar en un proyecto y si todas cumplen con la ingeniería de requisitos, la disponibilidad de repuestos, la facilidad de mantenimiento entonces es lógico optar por la de menor costo. Situaciones como estas también se incluyen en la integración de sistemas a nivel de *software.* Lo que se plantea en esta tesis es justo la integración de protocolos para los sistemas de riego que en general aún son aislados o no compatibles con los demás sistemas de una empresa.

2.5.1 Costos asociados de los equipos para la integración de sistemas

Los equipos que realizan las tareas de integración son por excelencia los PAC (*Programmable Automation Controller*). Un PAC es una tecnología industrial orientada al control automatizado, al diseño de prototipos y a la medición. El PAC se refiere al conjunto formado por un controlador (una CPU típicamente), módulos de entradas y salidas, y uno o múltiples buses de datos que lo interconectan todo. Este controlador combina eficientemente la fiabilidad de un PLC junto a la flexibilidad de monitorización y cálculo de un PC.

Los PAC pueden utilizarse en el ámbito investigador, pero es sobre todo a nivel industrial, para control de máquinas y procesos que es donde más se utiliza. También maneja múltiples lazos de control, adquisición de datos, análisis matemático, monitorización remota, visión artificial, control de movimiento, robótica, seguridad controlada, etc.[9]

Los PAC se comunican usando los protocolos de red abiertos como TCP/IP, OPC, puerto serie y es compatible con los privados (CAN, Profibus, etc) y pueden programarse además para protocolos independientes.
En cuanto a costos, las opciones más accesibles en el mercado y de las cuales se ha desarrollado un presupuesto son:

- National Instrument
- Red Lion
- Siemens
- Automation Direct

El manejo de precios, cantidad, licencias y accesorios comunes se ha basado en la instalación de 90 pivotes de riego del proyecto Agrolmos. Estos 90 pivotes recibirán las señales de control por radiofrecuencia hasta un receptor individual por panel. La decodificación por radio es transparente, y la demanda de equipos aguas arriba no es parte del presupuesto ni tampoco la instalación de tableros e instrumentación aguas abajo de los equipos de control. Solamente se compararán los precios que demande usar un PAC o PLC de las cuatros opciones anteriores.

El primer aspecto a comparar es la licencia para usar el *software* de programación. (Ver Tabla 5) El segundo aspecto a comparar es el costo del equipo requerido para cumplir la función de migración de protocolos. (Ver Tabla 6)

[9] http://es.wikipedia.org/wiki/Controlador_de_automatizaci%C3%B3n_programable

La Tabla 7, es el resumen general del gasto que implica cada una de las opciones tomando como base los precios unitarios, la cantidad requerida y un aproximado del desaduanaje y el IGV.

Tabla 5- Comparación de precios de las licencias

Opciones	Licencia de Software ($)
National Instrument	1100 (Lab View[10])
Red Lion	Gratis (Crimson[11])
Siemens	3500 (Simatic Step 7[12])
Automation Direct	Gratis (DirectSOFT[13])

Fuente: Elaboración propia.

Tabla 6- Comparación de precios de equipos

Opciones	Coste de equipo ($)
National Instrument	2080 (NI cRIO-9033[14])
Red Lion	2059 (DSPZR[15])
Siemens	3670 (CPU 315-2 DP +CP343-1[16])
Automation Direct	338 (DL05-D0-05AA[17]+ F0-CP-128[18])

Fuente: Elaboración propia.

Tabla 7- Ahorro en la compra de equipos para la migración de protocolos

Opciones	Ahorro - precio unitario ($)	Ahorro por los 90 pivotes ($)	Ahorro general ($)[19]
National Instrument	1742	156780	217924.2
Red Lion	1721	154890	215297.1
Siemens	3332	299880	416833.2
Automation Direct	✓	✓	✓

Fuente: Elaboración propia.

[10] http://www.ni.com/labview/buy/esa/
[11] http://www.redlion.net/crimson-30
[12] Cfr. Anexo A
[13] http://support.automationdirect.com/products/directsoft.html
[14] Cfr. Anexo B
[15] Cfr. Anexo C
[16] Cfr. Anexo A
[17] Cfr. Anexo D
[18] Cfr. Anexo E
[19] Este valor ha sido calculado considerando el valor de 18% de IGV y un valor de 21% para el valor del desaduanaje. (http://elcomercio.pe/economia/peru/como-desaduanar-compras-internet-noticia-764467) Estas dos consideraciones, genera un factor de 1.39 para la columna del ahorro general.

Cada opción maneja un *software* para la programación de su respectivo equipo, las notas a pie a de página son el enlace que brinda la misma empresa para su descarga y/o su compra según se indica.

National Instrument usa en su programación el *software* LabView, por ello brinda directamente el enlace para su compra con el perfil básico, existe opciones con más galerías, pero es suficiente con la indicada. Red Lion pone a disposición su *software* libre Crimson 3.0, con este programa se elaboran los algoritmos que precisa el PAC para su utilización. Siemens por su lado, ofrece el *software* Simatic Step7 para configuración, programación y diagnóstico de controladores Simatic de las gamas C7, S7-300 y S7-400.

Automation Direct pone a libre disposición el *software* DirectSOFT, este programa se puede descargar directamente de la misma página. Se le indica al lector que para los ensayos de comunicación con el panel de riego se usó el *software* Click Programming también de Automation Direct.

Tanto la opción de Siemens como la de Automation Direct necesitan un módulo de comunicaciones adicional al CPU. La opción de National Instrument y Red Lion no lo necesitan pues son PACs, los cuales ya están diseñados para soportar diferentes protocolos de comunicación en su *hardware*. Continuando con la evaluación económica se muestran los cálculos de ahorros respecto de la opción más económica considerando solamente la compra de equipos de la columna 4 de la tabla 7.

Es importante explicar que las opciones más caras son debido a que los equipos a adquirir son más sofisticados y además no se aprovecharía todo el potencial para el que han sido creados. La opción de Automation Direct demanda US$ 42283.8 y la más cercana es la que presupuesta Red Lion, con 215297.1 dólares adicionales. Ahorros de esta magnitud se pueden lograr en empresas como:

- ✓ Caña Brava
- ✓ Camposol
- ✓ Ecoacuicola
- ✓ Casa Grande

Donde el riego mecanizado puede ser integrado al sistema SCADA que manejan actualmente.

El beneficio económico es atractivo para el proyecto, pero esto no es simplemente eligiendo otros equipos. El usar PLCs para la migración de protocolos es a costa de implementar un algoritmo más exigente tanto para la fase de programación como para la cantidad de tareas que debe hacer el PLC en cada escaneo.

Por ello el siguiente punto a tratar es la metodología que se debe seguir para hacer la migración de un protocolo propietario a un protocolo estándar.

2.6 Metodología para llevar acabo la integración de protocolos

La integración de protocolos mediante la metodología que se ha usado, necesita analizar las tramas de comunicación entre los equipos y su software original por medio del puerto serial.

Una vez estudiada la forma de comunicación, se puede hacer compatibles con otros sistemas siempre y cuando se disponga de la forma en que se construye los mensajes y de un equipo que realice esta tarea. Para hacer los ensayos de comunicación con el panel de riego, primero se usó el *Software Base Station 2*, este es el programa original de los pivotes y del cual se logró extraer los comandos que utiliza para controlar las estructuras de riego.

Las tramas digitales fueron verificadas con el uso del *software Serial Port Monitor* hasta agotar todos los comandos que la tarjeta electrónica pudiera ejecutar. Después de la etapa de recolección de comandos se realizaron las pruebas en vacío con el *software* Docklight. Estos experimentos consistían en enviar una oración con la misma sintaxis del programa original y esperar la respuesta del panel.

Los ensayos fueron satisfactorios dado que, el panel Valley respondía a las peticiones de información según el comando. El siguiente paso fue implementar estas mismas peticiones pero desde un equipo independiente, ya no desde una PC.

Tomando como sustento la comparación económica y ahorro de implementación al usar el PLC de Automation Direct, se trabajó fuertemente en el algoritmo de comunicación para así lograr el enlace entre PLC y panel.

Otro requisito muy importante para la migración del protocolo es que el sistema debe actualizarse constantemente, por ende el PLC debe ser capaz de hacer esta tarea. Como parte de la metodología, se crearon subrutinas para mantener actualizados los parámetros de los pivotes, sin perder la sincronización de la información que se iba recogiendo y acciones que se iban ejecutando.

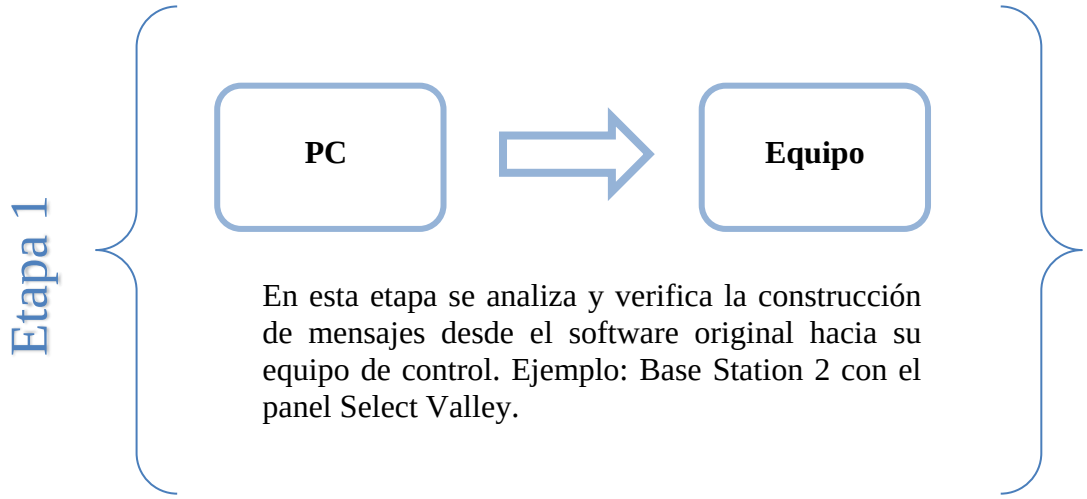
Siguiendo el orden en que se practicaron los ensayos de programación, se redujo cada subrutina que ejecutaba el panel a un solo bit de iniciación por cada tarea, al tener cada bit una única ubicación de memoria, se introdujo la construcción de tramas en protocolo Modbus desde la PC hacia el PLC. De manera análoga a la extracción del protocolo del panel de riego Valley, primero se usó del software Docklight para experimentar tramas sencillas de comunicación.

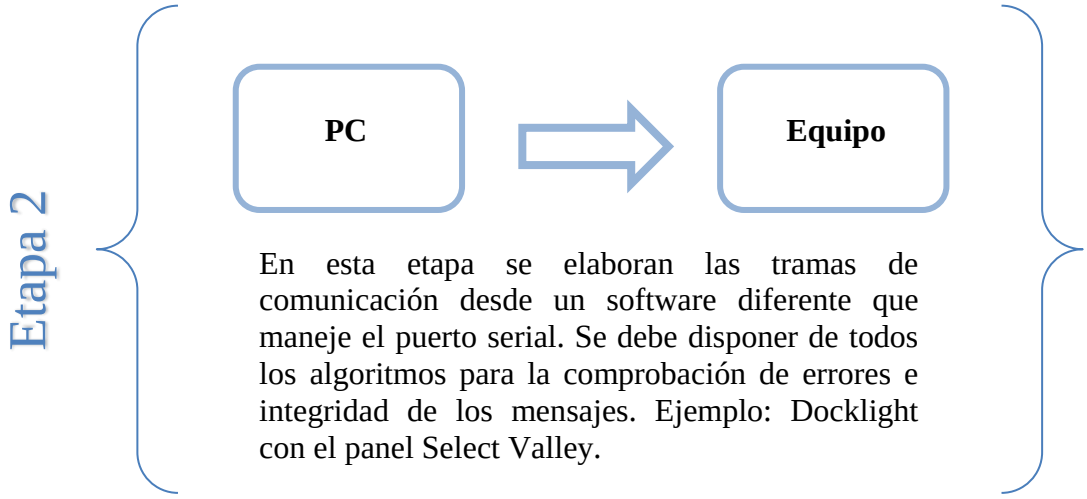
Luego se introdujeron a modo de vectores fila las tramas Modbus que energizaban el bit de inicio de cada subrutina que a su vez ejecutaba una acción específica en el panel. Como planteamiento final del proyecto, se trabajó con este conjunto de vectores pero ahora siendo llamados desde una interfaz gráfica. En la GUI de Matlab cada botón está asociado al llamado de un vector, este vector guarda una sintaxis Modbus que es exteriorizada mediante la manipulación del puerto serial de la PC.

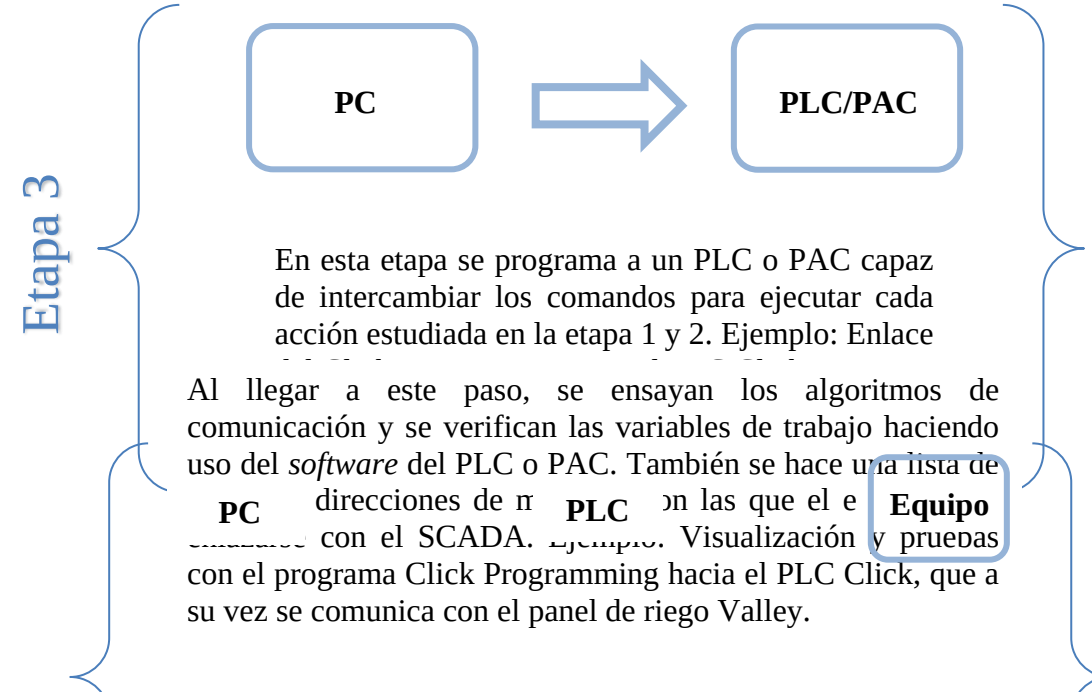
Bajo este enfoque pueden volverse compatibles los datos de los lazos de control de un proceso y dejar de trabajar aisladamente.

Finalmente la metodología para integrar los protocolos se logró con éxito y en los siguientes capítulos se explicará a detalle todos los aspectos teóricos y prácticos involucrados para la implementación.

2.6.1 Esquema de etapas para la integración de protocolos







Etapa 4

Etapa 5

La etapa 5 es análoga a la 2. Se realizan pruebas de comunicación pero desde un *software* diferente al del PLC o PAC. En este proyecto se ha trabajado con el protocolo de comunicación Modbus y con el *software* Docklight para los ensayos de esta fase.

Etapa 6

En esta etapa se configura al SCADA o GUI que muestre al sistema que se ha planteado en la ingeniería de requisitos. Al tener las tramas de comunicación desarrollada en la etapa 5 y las direcciones de memoria resumida de la etapa 4, se torna más fácil la programación de la interfaz de usuario. Ejemplo: GUI de Matlab usa los vectores Modbus para comunicarse con el PLC que a su vez se comunica con el panel de riego.

Fuente: Elaboración propia.

Capítulo 3

Transmisión de datos en protocolo Modbus

El conocimiento es la más importante materia prima.
El conocimiento es la fuente de valor agregado más importante.
El conocimiento es el más valioso rendimiento.
Si no se gestiona el conocimiento no se está prestando atención a la organización.
Thomas A. Stewart (The Wealth of Knowledge).

En la industria se usan varios tipos de comunicación para los equipos instalados en campo, en este capítulo se explicará uno en especial, el Protocolo *Modbus*. Este protocolo tiene una estructura de comunicación predefinida y estandarizada, que los elementos de control reconocen y pueden usarla para ejecutar las órdenes que reciben por parte del equipo maestro. La estructura de mensajería fue desarrollada por la compañía MODICON (*Modular Digital Controller*) y nació con el fin de conectar PLCs con sus herramientas de programación[20].

Es importante aclarar que una cosa es el protocolo de comunicación y otra es la capa física de la red. *Modbus,* por ejemplo puede hacerlo bajo una capa física en estándar RS-232, RS-485 o *Ethernet*. Este protocolo reúne una secuencia de pasos para que la trama digital a intercambiar, se valide correctamente entre ambas partes y se convierta a fin de cuentas en una instrucción para algún equipo. (Tejada Calderón, 2009)

Dependiendo del protocolo de comunicación a utilizar cada controlador reconocerá a los demás nodos de su propia red, a los códigos de función mediante los cuales se van a intercambiar datos y a la forma de cómo validar la trama de la instrucción. En las transacciones sobre redes *Modbus*, los controladores se enlazan con una técnica Maestro-Esclavo, el maestro puede iniciar las transacciones de peticiones y los esclavos o nodos responden a las peticiones ejecutando la acción descrita en la trama digital.

Es posible que existan mensajes de error en caso la trama haya sufrido alguna distorsión o la instrucción sea incoherente. Para que el dispositivo electrónico logre entablar comunicación con otro, el ciclo de petición-respuesta debe darse con ciertas condiciones, las cuales se explicarán a lo largo de todo el capítulo.

[20] http://www.simplymodbus.ca/

3.1 El ciclo petición-respuesta en protocolo *Modbus*

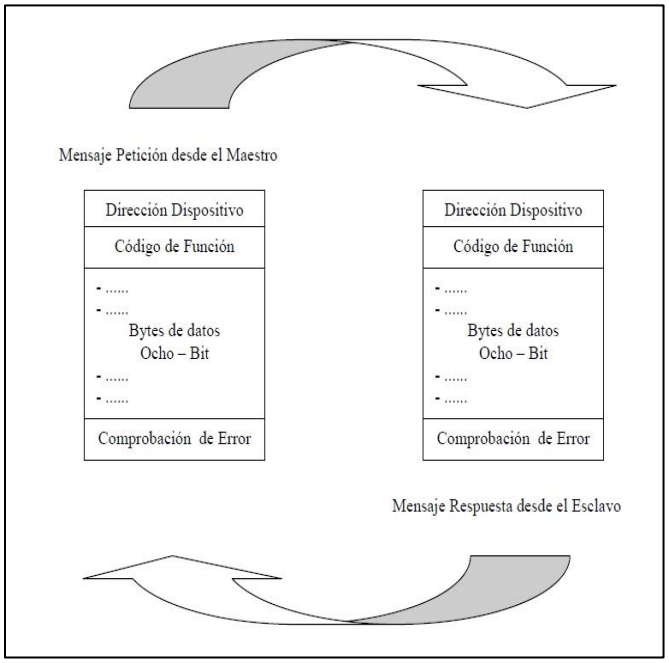
El dispositivo maestro puede enviar un mensaje a un esclavo específico o a todos los nodos. Si es un mensaje de difusión, los esclavos no emiten respuesta alguna, se suele usar este modo cuando se inicia un proceso con el fin de reiniciar todas las variables asociadas. Si el mensaje va direccionado a un nodo, él responderá de acuerdo al código de función solicitado.

El mensaje de respuesta del esclavo también cumple con la sintaxis de la trama *Modbus*, así se puede reconocer en que campo se encuentra el dato pedido. La sintaxis de la trama *Modbus* está conformada por los siguientes campos:

- Campo dedicado al número de esclavo
- Campo de código de función dictado por el maestro
- Campo de la dirección de memoria a trabajar
- Campo de datos solicitados
- Campo de comprobación de error[21].

En la figura 18, podemos apreciar esquemáticamente como se desarrolla el ciclo de petición-respuesta.

Figura 18- Ciclo de petición-respuesta en protocolo Modbus



Fuente: Internet

3.1.1 La petición o solicitud en protocolo Modbus

[21] La comprobación del error es un algoritmo para expresar la conformidad de los *bytes* que conforman la trama. En caso de recibir un mensaje defectuoso, el esclavo construirá un mensaje de error y lo enviará como respuesta.

La petición *Modbus* es la trama enviada desde el maestro hacia los esclavos. Dentro de esta trama existe un campo para el código de función, en el cual se le indica a un esclavo en específico que acción debe realizar. Los *bytes* de datos contienen información adicional que el esclavo necesita para llevar a cabo la función pedida.

Por ejemplo, un *byte* indica la dirección de memoria en qué debe comenzar a leer; otro *byte*, la cantidad de ubicaciones sobre las que debe actuar y otro, el campo de comprobación de error. Este orden proporciona un método para que el esclavo valide la integridad del contenido del mensaje recibido.

3.1.2 La respuesta en protocolo *Modbus*

La respuesta *Modbus* es el modo en que se dan los datos solicitados por el maestro. Si el esclavo elabora una respuesta normal el código de función de la respuesta, es igual al código de función enviado en la petición. Los *bytes* de datos contienen los valores y estados de los registros de los espacios de memoria solicitados.

Si ocurre un error, el código de función contenido en la respuesta será diferente al código de función enviado en la petición y por ende los siguientes *bytes* serán para indicar y describir el tipo de error.

3.2 Modos de transmisión Modbus: ASCII o RTU

En los PLCs configurados para comunicación en protocolo *Modbus* se les debe especificar la forma de transmisión a utilizar: ASCII o RTU. La selección del modo establece cómo deben ser empaquetados y decodificados los datos contenidos en los campos del mensaje.

Cabe resaltar que independientemente del modo de transmisión, los usuarios deben elegir los parámetros de comunicación del puerto serial como: velocidad de transmisión, *bit* de paridad, *bit* de *stop*, control de flujo, etc. Además estos valores deben ser los mismos para todos los dispositivos de la red y se configuran antes de hacer las pruebas de comunicación.

3.2.1 Modo RTU

Si los controladores son configurados para comunicarse en una red *Modbus* usando el modo RTU (*Remote Terminal Unit*) el mensaje deberá contener dos caracteres hexadecimales y cada caracter necesita de 4 *bits* para su representación. (Ver Tabla 8).

La ventaja más importante del modo RTU es que envía menos cantidad de caracteres para una misma acción, esto es sinónimo de mayor flujo de información y así se logra un mejor rendimiento sobre la velocidad de trasmisión.

Tabla 8- Correspondencia entre sistema Binario y Hexadecimal

0000→0	0100→4	1000→8	1100→C
0001→1	0101→5	1001→9	1101→D
0010→2	0110→6	1010→A	1110→E
0011→3	0111→7	1011→B	1111→F

El formato para cada *byte* en modo RTU es:

Sistema de codificación	Binario 8-bits, hexadecimal 0-9, A-F. Dos caracteres hexadecimales contenidos en cada campo de 8 bits del mensaje.
Bits por *byte*	1 bit de arranque. 8 bits de datos, el menos significativo se envía primero. 1 bit para paridad Par o Impar; ningún bit para No paridad. 1 bit de paro si se usa paridad; 2 bits si no se usa paridad.
Comprobación de error	Comprobación Cíclica Redundante (CRC).

3.2.2 Modo ASCII

Cuando los controladores se configuran para comunicarse en una red *Modbus* ASCII (*American Standar Code for Information Exchange)*, cada *byte* en un mensaje se envía como dos caracteres ASCII. La principal ventaja de este modo es que permite intervalos de tiempo de hasta un segundo entre caracteres disminuyendo posibles errores.

El formato para cada *byte* en modo ASCII es:

Sistema de codificación	Hexadecimal, caracteres ASCII 0-9, A-F. Un carácter hexadecimal contenido en cada carácter ASCII del mensaje.
Bits por *byte*	1 bit de arranque. 7 bits de datos, el menos significativo se envía primero. 1 bit para paridad Par o Impar; ningún bit para No paridad. 1 bit de paro si se usa paridad; 2 bits si no se usa paridad.
Comprobación de error	Comprobación Longitudinal Redundante (LRC).

3.3 Trama del mensaje Modbus

En cualquiera de los modos de transmisión (ASCII o RTU), el mensaje *Modbus* es emitido por el dispositivo maestro, en el cual la trama digital enviada tiene un comienzo y un final predeterminados para que el nodo lo pueda decodificar. Esto permite a los dispositivos receptores dar inicio a la lectura del mensaje, cuya primera parte de la trama es el campo del número de nodo y es el primer condicional para empezar a actuar.

Si la dirección leída es igual a su número de esclavo designado en la red, procederán a elaborar el mensaje según el código de función solicitado. En caso la dirección leída no corresponda con la que se designó al esclavo simplemente este no realizará acción alguna.

Si el campo de dirección viene con el valor de cero significa que es un mensaje para todos los nodos. En este caso el protocolo establece que ningún equipo debe responder, dado que si respondieran todos el bus de red colapsaría ya que el equipo maestro no sabrá a quien pertenece la información ni en qué orden respondieron los demás esclavos.

3.3.1 Trama RTU

En modo RTU los mensajes comienzan con un lapso de silencio de al menos 3.5 veces el tiempo de caracter. Pasado este tiempo, la comunicación se inicia con el campo de dirección. Los caracteres permitidos para todos los campos son del 0 a la F hexadecimal.

Al igual que en el modo ASCII los dispositivos monitorean constantemente la red, incluso en los espacios silenciosos. Cuando se recibe el primer campo todos los dispositivos lo decodifican a fin de enterarse si son o no los dispositivos direccionados.

Si durante el envío de un mensaje sucede un tiempo silencioso de mayor a 3.5 veces el tiempo de caracter, el dispositivo receptor entenderá que el mensaje terminó, por lo que procederá a comprobar la integridad del mensaje y dependiendo de eso actuará y estará habilitado a recibir un nuevo mensaje.

De igual manera, si antes de que pase 3.5 veces el tiempo de caracter, se recibe otra trama, el dispositivo entenderá que es una continuación del mensaje anterior y esto dará lugar a un error pues el CRC será solo del segundo paquete de datos.

3.3.2 Trama ASCII

En modo ASCII los mensajes comienzan con el caracter (:) el cual corresponde al '3A' hexadecimal y terminan con dos caracteres CRLF[22].

Los caracteres permitidos para todos los campos son 0-9 y A-F hexadecimal. En este caso los dispositivos de la red también están monitoreando constantemente para detectar los *dos puntos*. Cuando se reciben los *dos puntos*, decodifican el próximo campo para saber a qué dispositivo está direccionado el mensaje. Pueden darse intervalos hasta de un segundo entre caracteres; en caso de pasar más tiempo, el dispositivo receptor interpretará que ocurrió un error.

La tabla 9, muestra una trama de un mensaje en modo ASCII.

Tabla 9- Trama de un mensaje Modbus en el modo ASCII

Inicio	Dirección	Función	Datos	LRC	Final
Caracter (:)	2 caracteres	3 caracteres	N caracteres	2 caracteres	2 caracteres CRFL

1 Carriage Return-Line Feed .

3.4 Los campos internos en una trama Modbus

En un mensaje *Modbus* se forman 4 campos, el primero es la dirección, el segundo la función, el tercer campo es para los datos en general y el último campo es el de comprobación del error. (Tejada Calderón, 2009)

3.4.1 El campo de dirección

El campo de dirección de un mensaje contiene dos caracteres (ASCII) u ocho bits (RTU) y las direcciones de esclavo validas están en el rango de 0 a 247 (decimal). A cada dispositivo esclavo se le asigna un valor numérico, de este modo los dispositivos de la red tienen direcciones asignadas en el rango de 1 a 247.

El valor cero esta designado para mensajes de difusión. Un maestro direcciona un esclavo colocando su dirección numérica en el campo de dirección del mensaje. Cuando el esclavo envía su respuesta, reescribe su propia dirección y así da a conocer al maestro que él fue quien respondió.

3.4.2 El campo de función

El campo de código de función contiene dos caracteres (ASCII) u ocho bits (RTU). Cuando el mensaje es enviado desde un maestro hacia un dispositivo esclavo, el campo de código de función le indica que tipo de acción debe ejecutar. Los códigos de función más comunes se listan en la tabla 10.

Tabla 10- . Códigos de Función para protocolo Modbus

FUNCIONES DE LECTURA		
01	READ COIL STATUS	LEER ESTADO DE BOBINA
02	READ INPUT STATUS	LEER ESTADO DE ENTRADA
03	READ HOLDING REGISTER	LEER REGISTRO DE RETENCION
04	READ INPUT REGISTER	LEER REGISTROS DE ENTRADA
FUNCIONES DE ESCRITURA		
05	WRITE SINGLE COIL	ESCRIBIR SOBRE UNA BOBINA
06	WRITE SINGLE REGISTER	ESCRIBIR SOBRE UN REGISTRO
15	WRITE MULTIPLE COIL	ESCRIBIR EN MULTIPLES BOBINAS
16	WRITE MULTIPLE REGISTERS	ESCRIBIR EN MULTIPLES REGISTROS

Fuente: Elaboración propia

3.4.3 El campo de datos

El campo de datos se construye utilizando dos dígitos hexadecimales, en el rango de 00-FF hexadecimal. (Ver Tabla 8). Pueden formarse a partir de un par de caracteres ASCII o desde un carácter RTU, de acuerdo al modo de transmisión. El campo datos va acompañado de datos adicionales que el esclavo debe usar para tomar la acción definida por el código de función. Por ejemplo, ubicación de direcciones discretas, de registros, la cantidad de partes que han de ser manipuladas y la cantidad de *bytes* de datos a trabajar.

Si el maestro necesita leer un grupo de registros contenidos en un esclavo, el campo de datos especificará el registro de comienzo y cuántos registros han de ser leídos. Si el maestro ordena escribir sobre un grupo de registros en el esclavo, el campo datos especifica el registro de comienzo, cuántos registros debe escribir, la cantidad de datos a manipular y los valores de los datos que se deben forzar en los registros.

Si todo ocurre sin problemas, el campo de datos de la respuesta del esclavo contiene los datos solicitados o las confirmaciones de escritura. Si ocurre un error, el campo contiene un código de excepción que la aplicación del maestro puede decodificar para determinar la acción inmediata a ejecutar.

3.4.4 El campo de comprobación de error

El método para validar el campo de comprobación del error depende del modo de comunicación.

3.4.4.1 En modo RTU

En el modo RTU, el campo comprobación de error contiene un valor de 16 *bits* implementado como 2 *bytes* de 8 *bits* cada uno. El valor de comprobación de error es el resultado de un cálculo de Comprobación Cíclica Redundante o CRC (*Cyclical Redundancy Check*) realizado sobre el vector digital del mensaje.

Este algoritmo de comprobación, ha sido utilizado en la GUIDE de Matlab, en el capítulo V correspondiente a la explicación de la interfaz gráfica del monitoreo de pivotes. Los *bytes* del campo CRC son añadidos al mensaje como último campo de la trama. La forma de hacerlo es añadir primero el *byte* de orden bajo del campo, seguido del *byte* de orden alto, el cual es el último *byte* a enviar.

3.4.4.2 En modo ASCII

Aquí el campo de comprobación de error contiene dos caracteres ASCII. Dichos caracteres son el resultado del cálculo de Comprobación Longitudinal Redundante o LRC (*Longitudinal Redundancy Check*) que es realizado sobre el contenido del mensaje excluyendo los *dos puntos* del comienzo y los caracteres CRLF de finalización.

Los caracteres LRC son añadidos al mensaje como el último campo que precede a los caracteres CRLF.

3.5 Métodos de comprobación de error

Existen dos tipos de comprobación de error. El primero lo constituye la comprobación de paridad (par o impar) que puede ser aplicada opcionalmente a cada caracter del mensaje y forma parte de una seguridad más que todo eléctrica. El segundo, consiste en la comprobación de la trama (LRC o CRC) y es aplicada al mensaje completo.

Ambas comprobaciones por paquetes unitarios y por tramas completas son generadas en los dispositivos de la red y aplicadas a los contenidos del mensaje durante la transmisión y recepción. El PLC maestro es configurado para aguardar un tiempo predeterminado antes de abortar la transacción. Este intervalo es establecido para ser lo suficientemente largo como para que cualquier esclavo de la red responda normalmente, este valor se configura en el *time out*.

3.5.1 Control de paridad

Los usuarios pueden configurar los controladores para hacer control de paridad: Par, impar, o sin control de paridad; esto determinará cómo será iniciado el *bit* de paridad en cada carácter. Si se especifica cualquier control de paridad (Par o Impar), se contabilizará la cantidad de *bits* que tienen valor 1 en la porción de datos de cada caracter (siete *bits* de datos para modo ASCII y ocho para RTU).

Al bit de paridad se le dará el valor 0 o 1, para que se obtenga finalmente un número par o impar de bits con valor 1 respectivamente. Por ejemplo, estos 8 bits (1100 0101) forman parte de una trama RTU.

La cantidad de *bits* que tienen el valor 1 es cuatro. Si se utiliza control de paridad par, el *bit* de paridad de la trama debe establecerse a valor 0, haciendo que la cantidad de bits de valor 1 siga siendo un número par (cuatro en este caso). Si se utiliza control de paridad impar, el *bit* de paridad deberá tener valor 1, resultando una cantidad de *bits* de valor 1, impar (cinco).

Cuando el mensaje es transmitido, el bit de paridad es calculado y aplicado a la trama de cada carácter. El dispositivo receptor cuenta la cantidad de bits de valor 1 y establece un error si no coincide la paridad con la existente en el mensaje. Cabe resaltar que todos los dispositivos en la red *Modbus* deben ser configurados para usar el mismo método de control de paridad.

Obsérvese que la comprobación de paridad sólo detecta si un número impar de bits se han alterado en una trama de caracter durante la transmisión. Si se utiliza control de paridad Impar y dos *bits* de valor 1 de un caracter que tiene en origen 3 *bits* con valor 1, han quedado falseados (pasan a valor 0) durante la transmisión, el resultado es todavía un cómputo impar de *bits* de valor 1 y por lo tanto el error no es detectado por este método. En conclusión el control de paridad es vulnerable cuando se altera más de un *bit* durante la transmisión, para amortiguar este problema se usa la comprobación global del mensaje.

Si se especifica que no se realice el control de paridad, no se transmite *bit* de paridad y no se realiza dicha comprobación pero, se transmite un bit de paro adicional para completar la trama.

3.5.2 Generación del LRC

El LRC se calcula sumando entre sí los sucesivos *bytes* del mensaje, descartando cualquier *bit de acarreo* y luego se hace el complemento a 2 del valor resultante. El LRC es un campo de 8 *bits,* por lo tanto cada nueva suma de un caracter que sea mayor a 255 simplemente coloca el valor del campo a cero.
Esto sucede porque en la representación hexadecimal los números mayores a 255 necesitan un *byte* más para ser transmitidos, como no se puede agregar un noveno *bit* el acarreo se descarta automáticamente.
El procedimiento para generar un LRC es:

1. Sumar todos los *bytes* del mensaje, excluyendo los *dos puntos* de comienzo y el par CRLF del final. Esa sumatoria se debe realizar en un campo de 8 *bits*, así serán descartados los acarreos.

2. Restar el resultado del paso anterior del valor FF hexadecimal (todos los bit a 1), para producir el complemento a uno.

3. Sumar 1 al resultado del paso anterior para producir el complemento a 2.

Cuando los 8 bits del LRC (2 caracteres ASCII) son transmitidos en el mensaje, el caracter de orden alto será transmitido en primer lugar, seguido por el caracter de orden bajo.

3.5.2.1 Comprobación LRC

La Comprobación de Redundancia Longitudinal (LRC) es un *byte*. En modo ASCII, los mensajes incluyen un campo de comprobación de error que está basado en un método de LRC que controla el contenido del mensaje, a excepción del caracter *dos puntos* del comienzo y el par CRLF que va al final. Esto es aplicado independientemente de cualquier método de control de paridad.

El campo LRC es un *byte* calculado por el dispositivo emisor que se añade al mensaje original. El dispositivo receptor compara el LRC con el valor que él ha calculado de forma independiente. Si los dos valores no son iguales, el mensaje ha sufrido algún daño durante el envió.

3.5.3 Generación del CRC

Para calcular el valor CRC se carga un registro de 2 *bytes*, con los 16 *bits* puestos en el valor de 1 (FF-FF hexadecimal). Luego comienza un proceso en el que se toman los *bytes* sucesivos del mensaje. (Tejada Calderón, 2009). Se les opera con el contenido del registro y este se actualiza con el resultado obtenido. Sólo los 8 bits de dato de cada caracter son utilizados para generar el CRC.

Durante la generación del CRC, se efectúa una operación booleana OR exclusivo (XOR) a cada carácter de 8 *bits* con el contenido del registro. Entonces al resultado se le aplica un desplazamiento de *bit* en la dirección del *bit* menos significativo (LSB), rellenando la posición del *bit* más significativo (MSB) con un cero.

El LSB es extraído si fuese un 1, se realiza un XOR entre el registro y un polinomio de valor fijo. Si el LSB fuese un 0, no se efectúa el XOR. Este proceso se repite hasta haber cumplido 8 desplazamientos. Después del último desplazamiento, el siguiente *byte* y el valor actual del registro son operados con la función XOR. Este proceso es iterativo con

todos los *bytes* del mensaje. El contenido final del registro, después que todos los *bytes* del mensaje han sido procesados, es el valor del CRC.

Procedimiento para generar el CRC es:

1. Cargar un registro de 16 *bits* que denominaremos registro CRC inicial, con FF-FF (todos con el valor de 1).

2. Se toma el primer byte del mensaje y se realiza la función XOR con el *byte* de orden bajo del registro CRC de 16 *bits*, este resultado se sobrescribe en el registro CRC.

3. Desplazar el registro CRC un *bit* a la derecha (hacia el LSB), rellenando con un cero el MSB. Extraer y examinar el LSB.

4. Si el LSB era 0: Repetir paso 3 (otro desplazamiento).

5. Si el LSB era 1: Hacer XOR entre el registro CRC y el valor A001 hexadecimal (1010 0000 0000 0001- Polinomio característico de un CRC de 16 *bits*). Repetir los pasos 3 y 4 hasta que se hayan efectuado 8 desplazamientos. Una vez hecho esto, se habrá procesado un *byte* completo.

6. Repetir los pasos del 2 al 5 para el próximo *byte* del mensaje. Continuar este procedimiento hasta que todos los *bytes* del mensaje original hayan sido procesados.

7. El contenido final del registro CRC es el valor CRC.

8. Cuando el CRC es situado en el mensaje, sus *bytes* de orden alto y bajo han de ser permutados

9. Cuando el CRC de 16 *bits* (2 *bytes*) es transmitido en el mensaje, el *byte* de orden bajo se transmitirá primero, seguido por el *byte* de orden alto.

3.5.3.1 Comprobación CRC

En el modo RTU, los mensajes incluyen un campo de comprobación de error que está basado en un método de Comprobación de Redundancia Cíclica (CRC), el cual valida el contenido del mensaje completo. Se aplica con independencia de cualquier método de control de paridad utilizado para los caracteres individuales del mensaje.

El campo CRC es de dos *bytes*, que contiene un valor binario de 16 *bits*. El valor CRC es calculado por el dispositivo emisor, que añade el resultado del cálculo al mensaje. El dispositivo receptor calcula el CRC durante la recepción del mensaje y compara el valor calculado con el valor recibido en el campo CRC.

Los *bits* de arranque, *stop* y el *bit* de paridad, no se tienen en cuenta para el cálculo del CRC y cuando éste es añadido al mensaje, primero se añade el *byte* de orden bajo seguido del *byte* de orden alto; es decir, a la inversa del modo ASCII.

3.6 Comparación entre modo RTU y ASCII

La comparación de ambos modos se explicará con las Tablas 11, de solicitud *Modbus* y la Tabla 12, de respuesta *Modbus.* En las tablas se muestra un ejemplo de solicitud y respuesta normal, en cada una de ellas hay una columna para el modo RTU y otra, para el modo ASCII. Se detalla también cómo están distribuidos los valores de cada campo.

La Tabla 11, es una solicitud de Lectura de Registros Mantenidos (función 03) hacia el dispositivo esclavo con dirección 06. El mensaje pide los datos numéricos de tres registros. Observe que el mensaje especifica la dirección inicial con 00-6B hexadecimal que en el sistema decimal es la memoria 107.

Tabla 11- Petición Modbus comparando tramas ASCII/RTU

PETICIÓN	**Datos del ejemplo**	**Caracteres**	**Campo 8-Bits**
Nombre del campo	[Hex]	ASCII	RTU
Cabecera		:	Ninguno
Dirección del esclavo	06	0 6	0000 0110
Función	03	0 3	0000 0011
Dirección de inicio (MSB)	00	0 0	0000 0000
Dirección de inicio (LSB)	6B	6 B	0110 1011
Cantidad de registros (MSB)	00	0 0	0000 0000
Cantidad de registros (LSB)	03	0 3	0000 0011
Comprobación de error		LRC CR	CRC (16 *bits)*
Terminación		LF	Ninguno
	Total *Bytes*	17	8

Fuente: Elaboración propia

La respuesta (Ver Tabla 12) copia el código de función, indicando que se trata de una respuesta normal. El campo *computo de bytes* específica cuantos *bytes* se devuelven, que en este caso son 6 ya que se solicitaron 3 registros. Es decir, muestra la cantidad de *bytes* de datos que vienen a continuación, bien ASCII o RTU.

El objetivo de esta comparación es que se observe la variación en la cantidad de *bytes* totales que demanda cada uno. En la petición se diferencian en 9 y en la respuesta en 12.

Tabla 12- Respuesta Modbus comparando tramas ASCII/RTU

RESPUESTA	**Datos del ejemplo**	**Caracteres**	**Campo 8-Bits**
Nombre del campo	[Hex]	ASCII	RTU
Cabecera		:	Ninguno
Dirección del esclavo	06	0 6	0000 0110
Función	03	0 3	0000 0011
computo de bytes	06	0 6	0000 0110
Primer registro (MSB)	02	0 2	0000 0010
Primer registro (LSB)	2B	2 B	0010 1011
Segundo registro (MSB)	00	0 0	0000 0000
Segundo registro (LSB)	00	0 0	0000 0000
Tercer registro (MSB)	00	0 0	0000 0000
Tercer registro (LSB)	63	6 3	0110 0011
Comprobación de error		LRC	CRC (16 bits)
Terminación		CR	Ninguno
		LF	
	Total *Bytes*	23	11

Fuente: Elaboración propia

3.7 Ejemplos de tramas Modbus

Los siguientes ejemplos están enfocados para lograr una mejor comprensión de las transacciones en protocolo *Modbus* RTU. El motivo principal es que, el PLC utilizado en el desarrollo de la tesis es el Click Koyo, el cual posee protocolo de comunicación *Modbus* RTU.

La Tabla 13 resume las variables con las que trabaja el PLC. Se detalla los códigos de función que soporta cada tipo de variable, la dirección *Modbus* y la cantidad de espacios de memoria reservados para cada tipo de variable.

Tabla 13- Variables de trabajo en el entorno Click Programming Software

	Tipo	Códigos de función admisibles	Dirección CLICK	Dirección Modbus Hexadecimal	Cantidad de espacios de memoria
X	*Inputs*	02	X001-X816	00 00 - 01 0F	144
Y	*Outputs*	01,05,15	Y001-Y816	20 00 - 21 0F	144
C	*Control relays*	01,05,15	C1-C2000	40 00 - 47 CF	2000
T	*Timers*	02	T1-T500	B0 00 - B1 F3	500
CT	*Counters*	02	CT1-CT250	C0 00 - C0 F9	250
SC	*System control relays*	02	SC1-SC1000	F0 00 - F3 E7	1000
DS	*Data register - single word*	03,06,16	DS1-DS4500	00 00 - 11 93	4500
DD	*Data register - double word*	03,06,16	DD1-DD1000	40 00 - 47 CE	1000
DH	*Data register - hex format*	03,06,16	DH1-DH500	60 00 - 61 F3	500
DF	*Data register - floating point*	03,06,16	DF1-DF500	70 00 - 73 E6	500
XD	*Inputs - word*	04	XD0-XD8	E0 00 - E0 10	9
YD	*Outputs - word*	03,06,16	YD0-YD8	E2 00 - E2 10	9
TD	*Timers current values*	04	TD1-TD500	B0 00 - B1 F3	500
CTD	*Counters current values*	04	CTD1-CTD250	C0 00 - C1 F2	250
SD	*System data register*	04	SD1-SD1000	F0 00 - F3 E7	1000
TXT	*Text data buffer*	03,06,16	TXT1-TXT1000	90 00 - 91 F3	1000

Fuente: Elaboración propia

3.7.1 *Read Coil Status* (Leer estado de bobina) FC=01

Petición

11 01 00 13 00 25 0E 84

Este comando está pidiendo el estado *ON / OFF* de bobinas discretas # 20 a la 56 desde el dispositivo esclavo con dirección 17.

11: La dirección del esclavo (17 = 11 hex)
01: El código de función (Leer estado de bobina)
00 13: La dirección de memoria de la primera bobina a leer. (Bobina 20 - 1 = 19 = 13 hex)
00 25: El número de bobinas a leer. (Bobinas 20 a 56 = 37 = 25 hex)
0E 84: El CRC para la comprobación de errores.

Respuesta

11 01 05 CD 6B B2 0E 1B 45 E6

11: La dirección del esclavo (17 = 11 hex)
01: El código de función (Leer estado de bobina)
05: El número de *bytes* que vienen a continuación (37 bobinas / 8 *bits* por *byte* = 5 bytes)
CD: Recoge el estado de las bobinas 27-20 (1100 1101)
6B: Recoge el estado de las bobinas 35-28 (0110 1011)
B2: Recoge el estado de las bobinas 43-36 (1011 0010)
0E: Recoge el estado de las bobinas 51-44 (0000 1110)
1B: Recoge el estado de las bobinas 56-52, pero el estado de la bobina 57, 58 y 59 como no fueron solicitadas en la petición inicial, el protocolo establece usar ceros para estos estados y así lograr que la trasmisión complete los 8 bits (*000*1 1011)
45 E6: El CRC para la comprobación de errores.

Los bits más significativos contienen el estado de la bobina más alta, es decir que la bobina 35 se encuentra desenergizada y que la bobina 28 esta energizada.

3.7.2 *Read Input Status* (Leer estado de entrada) FC=02

Petición

11 02 00 C4 00 16 BA A9

Este comando está pidiendo el estado *ON / OFF* de las entradas discretas #197 a la 218 desde el dispositivo esclavo con dirección 17.

11: La dirección del esclavo (17 = 11 hex)
02: El código de función (Leer estado de entrada)
00 C4: La dirección de memoria de la primera entrada a leer. (197-1 = 196 = C4 hex)
00 16: La cantidad de entradas solicitadas. (197 a 218 = 22 = 16 hex)
BA A9: El CRC para la comprobación de errores.

Respuesta

11 02 03 AC DB 35 20 18

11: La dirección del esclavo (17 = 11 hex)
02: El código de función (Leer estado de entrada)
03: El número de *bytes* que vienen a continuación (22 entradas / 8 *bits* por *byte* = 3 *bytes*)
AC: Entradas discretas 204 -197 (1010 1100)
DB: Entradas discretas 212 hasta 205 (1101 1011)
35: Entradas discretas 220-213 (*00*11 0101)
20 18: El CRC para la comprobación de errores.

Los *bits* más significativos contienen las entradas discretas más altas. Esto demuestra que la entrada 197 está apagada (0) y la 204 se encuentra energizada (1).
De manera análoga al caso anterior, la necesidad de completar los 8 *bits* hace que el mensaje devuelva con ceros las entradas 220 y 219 a pesar de estar energizadas o no.

3.7.3 *Read Holding Registers* (Leer registros de retención) FC = 03

Petición

11 03 00 6B 00 03 76 87

Esta trama está pidiendo el contenido de los registros #108 al 110 del dispositivo esclavo con la dirección 17.

11: La dirección del esclavo (17 = 11 hex)
03: El código de función (Leer registros de memoria)
00 6B: La dirección de memoria del primer registro solicitado. (108-1 = 107 = 6B hex)
00 03: El número total de registros solicitados (Leer 3 registros del 108 hasta el 110)
76 87: El CRC (comprobación de redundancia cíclica) para la comprobación de errores.

Respuesta

11 03 06 56 52 43 40 AE 41 49 AD

11: La dirección del esclavo (17 = 11 hex)
03: El código de función, se le recuerda al lector que, al no haber problema de comunicación, el código de función en la petición, es igual al código de función en la respuesta.
06: El número de bytes que vienen a continuación (3 registros por 2 *bytes* cada uno = 6 *bytes*).

Dado que la representación utilizada es la hexadecimal y el máximo dato que puede guardar un registro es FF-FF, significa que se está limitado al valor de 65535.

$$FF\ FF\ hex = 16^3 * 15 + 16^2 * 15 + 16^1 * 15 + 16^0 * 15 = 65535$$

AE 41: El contenido del registro 108, es decir el valor de 44609.
43 40: El contenido del registro 109, es decir el valor de 17216.
56 52: El contenido del registro 110, es decir el valor de 22098.
49 AD: El CRC (comprobación de redundancia cíclica).

3.7.4 *Read Input Registers* (Leer registros de entrada) FC = 04

Petición

11 04 00 08 00 01 B2 98

Este comando está pidiendo el contenido del registro # 9 desde el dispositivo esclavo con dirección 17.

11: La Dirección del esclavo (17 = 11 hex)
04: El código de función (Leer registros de entrada)
00 08: La dirección de dato del primer registro solicitado. (9-1 = 08 hex)
00 01: El número total de registros solicitados. (Leer 1 registro)
B2 98: El CRC (comprobación de redundancia cíclica) para la comprobación de errores.

Respuesta

11 04 02 00 0A F8 F4

11: La dirección del esclavo (17 = 11 hex)
04: El código de función (Leer registros de entrada)
02: La cantidad de *bytes* que vienen a continuación (1 registro por 2 *bytes* cada uno = 2 *bytes*)
00 0A: El contenido del registro 9, es decir el valor de 10.
F8 F4: El CRC (comprobación de redundancia cíclica).

3.7.5 *Write Single Coil* (Escribir o forzar una bobina) FC=05

Petición

11 05 00 AC FF 00 4E 8B

Este comando está forzando el contenido de bobina discreta # 173 en *ON* en el dispositivo esclavo con la dirección 17.

11: La dirección del esclavo (17 = 11 hex)
05: El código de función (Forzar una bobina)
00 AC: La dirección de memoria de la bobina. (173 - 1 = 172 = AC hex)
FF 00: El estado a escribir (FF 00 = *ON*, 00 00 = *OFF*)
4E 8B: El CRC (comprobación de redundancia cíclica) para la comprobación de errores.

Respuesta

11 05 00 AC FF 00 4E 8B

La respuesta normal ante una función 05 es una réplica de la petición devuelta después de que, el dispositivo ha llevado a la bobina al estado requerido.

11: La dirección del esclavo (17 = 11 hex)
05: El código de función.
00 AC: La dirección de memoria de la bobina.
FF 00: El estado escrito sobre la bobina (FF 00 = *ON*, 00 00 = *OFF*)
4E 8B: El CRC (comprobación de redundancia cíclica) para la comprobación de errores.

3.7.6 *Write Single Register* (Escribir o forzar un registro) FC=06

Petición

11 06 00 01 00 03 9A 9B

Este comando está escribiendo el valor de 3 sobre el registro # 2 sobre el dispositivo esclavo con dirección 17.

11: La dirección del esclavo (17 = 11 hex)
06: El código de función (Escribir sobre un registro)
00 01: La dirección de memoria del registro. (# 2 -1 = 01 hex)
00 03: El dato a escribir, es decir el valor de 3.
9A 9B: El CRC (comprobación de redundancia cíclica) para la comprobación de errores.

Respuesta

11 06 00 01 00 03 9A 9B

La respuesta normal es una réplica de la petición, devuelto después, de que el dispositivo ha escrito el valor deseado sobre la dirección de memoria indicada.

11: La dirección del esclavo (17 = 11 hex)
06: El código de función (Escribir sobre un registro)
00 01: La dirección del registro. (# 2 - 1= 1)
00 03: El valor que se ha forzado sobre la dirección de memoria, enviada en la petición.
9A 9B: El CRC (comprobación de redundancia cíclica) para la comprobación de errores.

3.7.7 *Write Multiple Coil* (Forzar múltiples bobinas) FC = 15

Petición

11 0F 00 13 00 0A 02 CD 01 BF 0B

Este comando está forzando el estado de 10 bobinas, desde las direcciones # 20 a la 29 sobre el dispositivo esclavo con dirección 17.

11: La dirección del esclavo (17 = 11 hex)
0F: El código de función (Escribir sobre varias bobinas)
00 13: La dirección de memoria de la primera bobina (# 20 - 1 = 19 = 13 hex)
00 0A: El cantidad de bobinas que se forzarán (10 = 0A hex)
02: La cantidad de *bytes* que demanda el paquete de bobinas a forzar (10 bobinas / 8 *bits* por *byte* = 2 *bytes*)
CD: Estado que se forzará sobre las bobinas 27-20 (1100 1101)
01: Estado que se forzará sobre las bobinas 35-28 (*0000 00*01).

Cabe indicar que al igual que en los casos anteriores, el protocolo decreta completar con ceros los espacios que no han sido solicitados, sin alterar el verdadero estado en que se encontraba la variable. Esto significa que a la bobina 29 se le forzará el estado de apagado, pero es falso decir que a la bobina 35 se le des-energizará.

BF 0B: El CRC (verificación de redundancia cíclica) para la comprobación de errores.

Los *bits* más significativos contienen las variables de bobina más altos. Esto demuestra que la bobina 20 está en (1) y la 21 está en (0). Los *bits* no utilizados en el último *byte* de datos se completan con ceros.

Respuesta

11 0F 00 13 00 0A 26 99

11: La dirección del esclavo (17 = 11 hex)
0F: El código de función (Escribir en múltiples bobinas, 15 = 0F hex)
00 13: La dirección de datos de la primera bobina (bobina # 20 - 1 = 19 = 13 hex)
00 0A: El número de bobinas en las que se ha trabajado (10 = 0A hex)
26 99: El CRC (comprobación de redundancia cíclica) para la comprobación de errores.

3.7.8 *Write Multiple Registers* (Escribir en múltiples registros) FC = 16

Petición

11 10 00 01 00 02 04 01 02 00 0A C6 F0

Este comando está escribiendo en los registros #2 y 3 los valores de 10 y 258 respectivamente sobre el dispositivo esclavo con la dirección 17.

11: La dirección del esclavo (17 = 11 hex)
10: El código de función (Escribir múltiples registros, 16 = 10 hex)
00 01: La dirección de memoria del primer registro. (# 2-1 = 1)
00 02: La cantidad de registros sobre los cuales se va a escribir.
04: La cantidad de *bytes* que demanda esta operación (2 registros por 2 *bytes* cada uno = 4 *bytes*)
01 02: El dato a escribir en el registro 3, es decir el valor 258.
00 0A: El dato a escribir en el registro 2, es decir el valor 10.
C6 F0: El CRC (comprobación de redundancia cíclica) para la comprobación de errores.

Respuesta

11 10 00 01 00 02 12 98

11: La dirección del esclavo (17 = 11 hex)
10: El código de función.
00 01: La dirección del primer registro sobre el cual se ha trabajado (# 2-1 = 1)
00 02: La cantidad de registros sobre los cuales se trabajó.
12 98: El CRC (comprobación de redundancia cíclica) para la comprobación de errores.

Después de analizar los ejemplos anteriores, el lector está en la capacidad de interpretar las tramas *Modbus* que se generan entre equipos, para ello se debe contar con programas como el *Docklight*, el *Visual Serial Port Monitor*, etc.

Para construir o elaborar un vector *Modbus* desde cero se debe saber calcular y agregar el valor del CRC. Por ello, en el presente proyecto, este conjunto de instrucciones que se necesitan iterar para hallar el CRC es cargado directamente al Matlab con el fin de aprovechar su gran motor matemático en este tipo de tareas.

3.8 Respuestas de excepción en Modbus

El protocolo también está preparado para dar respuestas de excepción, ante estos casos la trama de respuesta sufre algunos cambios, por ende es evidente que se debe saber

interpretar una situación de error, las siguientes líneas explican cómo identificar el tipo de error y la situación que lo ha generado.

La única situación en que, el protocolo decreta no generar respuesta es ante un mensaje tipo difusión, en cualquier otro caso si el esclavo no responde es debido a un error.

Cuando el maestro envía una petición a un esclavo pueden ocurrir cuatro situaciones.

- El dispositivo esclavo recibe la petición sin error de comunicación y puede manejar la orden sin problemas y por lo tanto, devuelve una respuesta normal.

- El dispositivo esclavo no recibe la petición debido a un error de comunicación, es decir que la solicitud no llego a destino y por tanto no hay devolución de respuesta. El programa del equipo maestro eventualmente procesará una condición de tiempo excedido (*timeout*).

- El dispositivo esclavo recibe la petición pero detecta un error en la sintaxis de la trama *Modbus*, o un valor de CRC incoherente o una no conformidad en el control de paridad, ante esta situación no habrá devolución de respuesta dado que el esclavo hace caso omiso a la solicitud. El programa del maestro eventualmente procesará una condición de tiempo excedido (*timeout*).

- El dispositivo esclavo recibe la petición sin error, pero la solicitud no puede ser procesada o no es manejable para el esclavo por otros motivos. Por ejemplo, si la solicitud es leer una bobina o forzar un registro inexistente, el esclavo devolverá una respuesta de excepción informando la naturaleza del error.

El mensaje de respuesta de excepción, tiene dos campos que lo diferencian de una respuesta normal; la modificación del código de función y el campo de código de excepción. Se hace ahínco que, en una respuesta normal, el esclavo devuelve el código de función idéntico al de la solicitud y el campo de datos acorde a lo requerido.

3.8.1 Cambio en el código de función

El primer aviso de una respuesta de excepción es que el código de función que retorna posee el *bit* más significativo energizado.

Por tanto es señal que el dispositivo esclavo no puede procesar la solicitud. En la tabla 14, muestra el cambio de estado del MSB.

Tabla 14- Representación decimal, hexadecimal y binaria del cambio que conlleva una respuesta de excepción

Código de función normal	Código de función de una respuesta de excepción
01 (01 hex) **0**000 0001	129 (81 hex) **1**000 0001
02 (02 hex) **0**000 0010	130 (82 hex) **1**000 0010
03 (03 hex) **0**000 0011	131 (83 hex) **1**000 0011

04 (04 hex) **0**000 0100	132 (84 hex) **1**000 0100
05 (05 hex) **0**000 0101	133 (85 hex) **1**000 0101
06 (06 hex) **0**000 0110	134 (86 hex) **1**000 0110
15 (0F hex) **0**000 1111	143 (8F hex) **1**000 1111
16 (10 hex) **0**001 0000	144 (90 hex) **1**001 0000

Fuente: Elaboración propia

3.8.2 Cambio en el campo de datos

La primera modificación que sufre el campo de datos, es que pasa a llamarse campo de código de excepción, aquí el dispositivo esclavo devolverá la naturaleza del error que ha llevado a esta situación. .

A continuación el lector podrá interpretar los códigos de excepción que maneja el protocolo *Modbus.*

01 (01 hex)
Función ilegal, el código de función recibido en la petición no es una acción permitida en el esclavo.

02 (02 hex)
Dato de dirección ilegal, el dato de dirección recibido en la petición no es una dirección permitida para el esclavo.

03 (03 hex)
Dato ilegal del valor, un valor contenido en el campo de datos, de la petición, no es un valor admisible para el esclavo.

04 (04 hex)
Falla en el dispositivo esclavo, significa que ha ocurrido un error no recuperable mientras el esclavo estaba intentando ejecutar la acción solicitada.

05 (05 hex)
El esclavo ha aceptado la petición y está procesándola, pero requerirá de un tiempo prolongado para terminar la tarea, esta respuesta se da para advertirle al maestro que el silencio del bus no es consecuencia de una falla, sino de un aviso de la demora del mensaje de respuesta.

06 (06 hex)
El dispositivo esclavo se encuentra ocupado procesando una tarea de larga duración. Le corresponde al maestro retransmitir el mensaje más tarde, cuando el esclavo esté libre.

07 (07 hex)
Reconocimiento negativo, el esclavo no puede efectuar la función recibida en la petición. El maestro debería pedir al esclavo un diagnóstico o información de error.

08 (08 hex)

Error en la paridad en memoria, es decir, que el esclavo ha intentado leer la memoria extendida pero ha detectado un error de paridad.

La siguiente trama es un ejemplo de una solicitud con una respuesta de excepción.

Petición

0A 01 04 A1 00 01 AC 63

Este comando está pidiendo el estado *ON / OFF* de la bobina discreta # 1186 desde el dispositivo esclavo con dirección 10.

0A: La dirección del esclavo (10 = 0A hex)
01: El código de función (*Read Coil Status*)
04 A1: La dirección de datos de la primera bobina a leer. (1186-1 = 1185 = 04 A1 hexadecimal)
00 01: El número total de bobinas solicitadas.
AC 63: El CRC (comprobación de redundancia cíclica) para la comprobación de errores.

Respuesta

0A 81 02 B0 53

0A: La dirección del esclavo (10 = 0A hex)
81: El código de función de una respuesta de excepción (FC=01, con el bit más alto energizado)
02: El código de excepción, el cual explica que la dirección de memoria en la que se tiene que trabajar, no es una dirección permitida para el esclavo.
B0 53: El CRC (comprobación de redundancia cíclica).

Capítulo 4

Software Click Programming

El verdadero progreso es el que pone la tecnología al alcance de todos.

Henry Ford

El uso de las nuevas tecnologías cambia el panorama industrial, la automatización de procesos ayuda en gran medida a las empresas a mejorar su control, calidad y productividad. Ante este nuevo escenario, surge un mercado el cual ofrece el servicio de instalación, venta de equipos y análisis de propuestas. Son muchísimas las empresas que brindan este tipo de soluciones. Es por ello que, las personas encargadas de iniciar un nuevo proyecto de automatización también deben evaluar el tipo de tecnología a usar, la ingeniería de requisitos se verá muy favorecida si personas con amplia experiencia en este rubro participan de esta etapa.

Cuando se ha decidido por un fabricante, ya sea por los beneficios, precios, tiempos de entrega y/o ejecución, accesibilidad, repuestos, historial y ejemplos de trabajos realizados con anterioridad. Además, es importante evaluar el soporte técnico que brindan, ya que muchas de las fallas precisan ser corregidas en el menor tiempo posible por los efectos negativos que podrían acarrear para la empresa. Dicho esto, se le explica al lector que los motivos de uso de los PLCs Click para la realización de la presente tesis son; la experiencia de su implementación en diferentes proyectos de la región por parte del grupo asesor, su facilidad de compra y el constante soporte técnico.

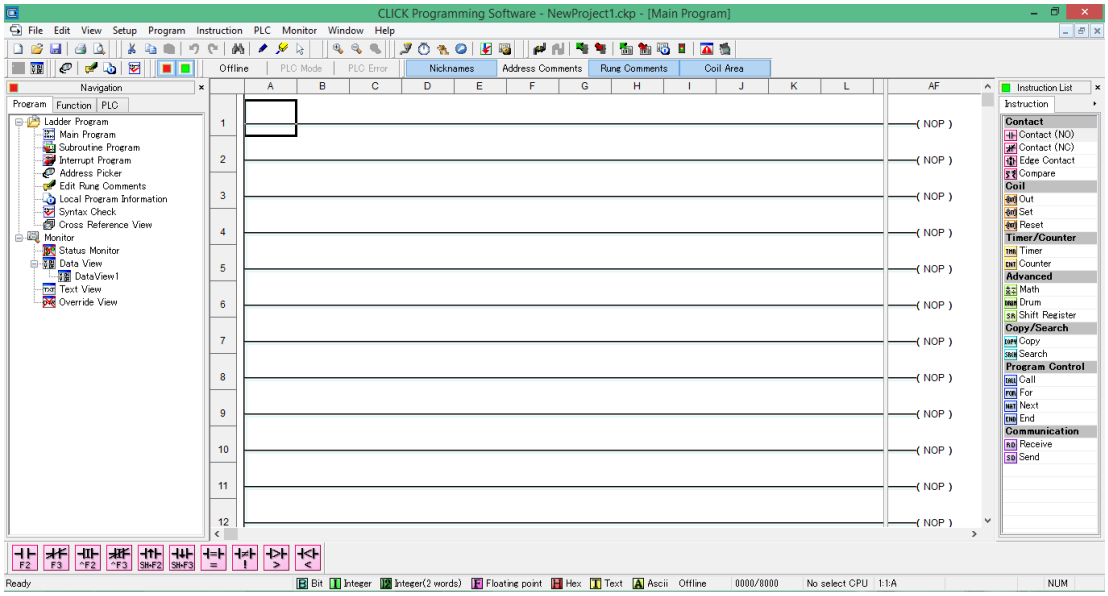
Automation Direct ofrece un abanico completo de productos que pueden ser visualizados en su página web para proyectos de este tipo, también pone a disposición su software Click Programming, que es de libre acceso, está en constante actualización y ha sido empleado para esta tesis.

En este capítulo se explicará la parte más importante del proyecto, la cual es el modo de comunicación entre el panel de riego Valley con el PLC. También se abordaran las características del equipo utilizado y el algoritmo realizado para el funcionamiento de la red de pivotes de riego. Posteriormente, se detallara la forma del intercambio de datos en modo ASCII.

4.1 Entorno de programación Click programming

El lenguaje de programación del PLC Click Koyo es un entorno de desarrollo en escalera, es decir las rutinas que se creen tendrán un esquema de secuencias lógicas conforme se vayan cumpliendo los condicionales de un proceso y estos se podrán observar en línea mientras van sucediendo. En la figura 19se visualiza la ventana de inicio del software.

Figura 19-Interfaz de inicio del Software Click Programming



Fuente: Software Click Programming

Esta clase de PLC, tiene como ventaja las diferentes opciones de comunicación industrial que puede procesar. Dispone también de módulos de señales y de una fácil instalación y manejo del software. La figura 20 es la portada del software, se aprecia en el cómo son estos equipos.

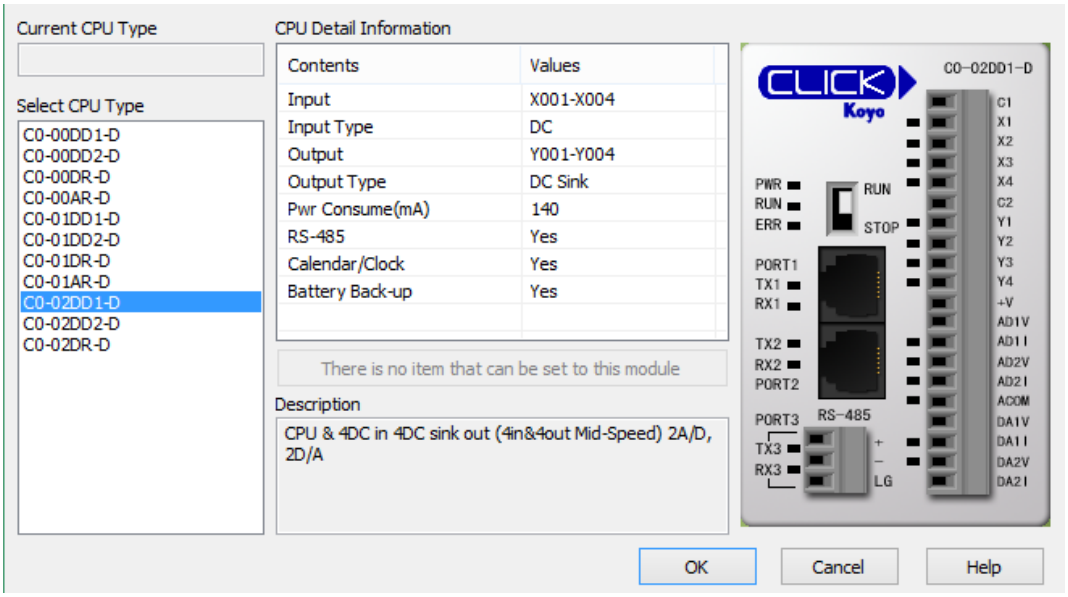
Figura 20- Inicialización del Software Click Programming

Fuente: Software Click Programming

La figura 21 nos detalla las características de la CPU utilizada para los ensayos de programación. Retomando la figura 19, es la misma que se visualiza al momento de elegir un nuevo proyecto. La ubicación de las herramientas es bastante sencilla y organizada, allí se encuentran las instrucciones lógicas que maneja el equipo y se pueden implementar en el algoritmo de control que se desee, por ejemplo:

- El uso de contactores cerrados y abiertos.
- Utilizar flancos positivos o negativos de los contactos.
- Energizar y desenergizar bobinas internas, ingresar valores y estados retentivos.
- Usar *timers* de diferentes intervalos de tiempo y contadores para administrar la cantidad de eventos.
- Operaciones matemáticas.
- Comparaciones de datos almacenados.
- Introducir y extraer valores de bits en paquetes de variables hexadecimales.
- Crear subrutinas internas e interrupciones locales.
- Habilitar el intercambio de datos con los comandos de envío y recepción.

Figura 21- Muestra las características del CPU C0-02DD1-D utilizado para los ensayos



Fuente: Software Click Programming

4.2 Puertos de configuración de la CPU del PLC Click

Estas son las herramientas básicas para el control de procesos. En cuanto a la capa física de instalación, los PLCs Click poseen entradas y salidas digitales, también tienen entradas y salidas analógicas tanto en voltaje como en corriente.

Los equipos Click son expandibles y modulares; es decir, son capaces de aumentar la cantidad de señales que pueden procesar al aumentar la cantidad de módulos que acompañan a la CPU.
También poseen la característica de comunicación en modo bidireccional, es decir envío y recepción de datos a través de los equipos que pertenezcan a la red industrial. Los siguientes apartados se detallarán la configuración de los puertos dado que es muy importante manejar estos datos para lograr la integración de equipos.

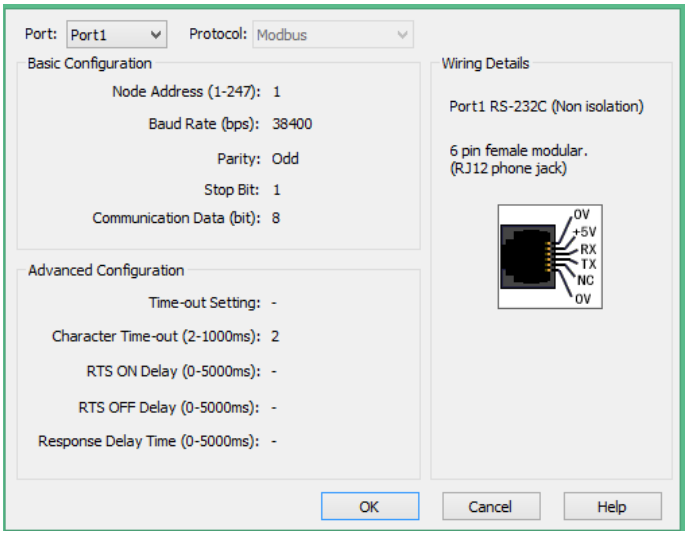
4.2.1 Detalle del puerto número uno de la CPU.

La figura 22, detalla los parámetros del puerto número uno del PLC, estos valores no son modificables y además, es el puerto por donde se realiza la programación interna de la CPU. Mediante este puerto también se pueden realizar transacciones *Modbus* que no estén relacionadas a la programación interna del equipo.

La conexión física a este puerto es con un RJ12 con estándar RS-232, por ende está limitado a una comunicación punto a punto. Los parámetros para el intercambio de información entre PLC y PC son:

Dirección de nodo: 1 Significa que las tramas *Modbus* inician declarando al esclavo uno de la red, el cual no se puede cambiar.
Velocidad de transmisión: 38400 bps
Paridad: Impar
Bit de parada: 1
Datos de comunicación: 8 bits

Figura 22- Pantalla de configuración del puerto uno



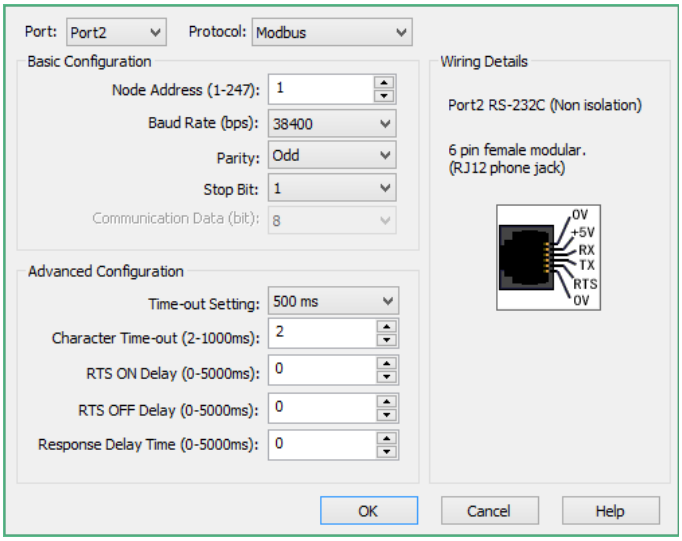
Fuente: Software Click Programming

4.2.2 Detalle del puerto número dos de la CPU

El siguiente puerto de comunicación es el número dos. También se accede con un RJ12 y tiene el estándar RS-232. Está preparado para el intercambio de datos punto a punto. La principal diferencia respecto al puerto son sus distintas opciones de configuración, a continuación se detallará cada una de ellas. (Ver figura 23)

Se puede utilizar el modo de comunicación ASCII y el protocolo *Modbus*. También es posible cambiar el número de nodo si se utiliza *Modbus*, esto es una condición necesaria para lograr interconectar distintos equipos en una red.

Figura 23- Pantalla de configuración del puerto dos



Fuente: Software Click Programming

La velocidad de transmisión, el bit de stop y la paridad son seleccionables. La vía de comunicación del PLC hacia el panel Valley se realiza por este puerto con los siguientes parámetros:

Protocol: ASCII
Baud Rate (bps): 9600
Parity: None
Stop Bit: 1
Communication Data (bit): 8

Todas las transacciones de petición y respuesta son bajo el protocolo de Valley. Si se analiza los modos de envío ASCII estos están considerados en la opción de mensaje de *static text*. Lleva este nombre porque siempre se envía el mismo paquete de datos a diferencia del modo *dynamic text*.

En el modo dinámico, el paquete de datos que se envía es variable y está asociado al valor de una dirección de memoria específica. Por lo general, esta dirección de memoria participa de una rutina dentro del algoritmo del PLC y actualiza de manera constante un valor que debe ser monitoreando. Cuando el panel de riego Valley responde a las peticiones del PLC entrega una trama de datos que corresponde al código solicitado.

Una vez recibido el paquete de datos se decodifica del formato texto a valores enteros con el fin de hacer más fácil las operaciones matemáticas y posteriormente los datos que van a ser enviados a Matlab.

En este proyecto, la capa física de comunicación es entre el puerto dos del PLC Click y el puerto de comunicación DB9 del panel de riego Valley.

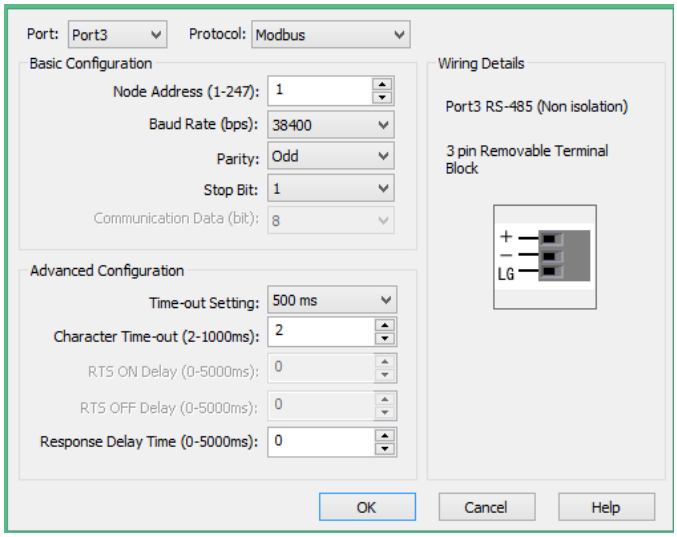
Para finalizar la descripción, se le indica al lector que entre el conector que posee el panel de riego y el PLC se ha hecho una adaptación entre los pines para que haya correspondencia entre ambos equipos.

4.2.3 Detalle del puerto número tres de la CPU

El último puerto de comunicación es el puerto tres, su característica principal es el modo de transmisión en el estándar RS-485.

En la figura 24, se muestra las diferentes configuraciones que se le pueden atribuir al puerto RS-485. Es importante indicar que en este estándar, el intercambio de datos es multipunto. La dirección de nodo se utiliza para establecer entre que equipos se dará la comunicación. Al igual que el puerto anterior se puede modificar la velocidad de transmisión, la paridad, el bit de parada, el modo ASCII y *Modbus.*

Figura 24- Pantalla de configuración del puerto tres



Fuente: Software Click Programming

4.3 Detalle de envío y recepción de datos en Click Programming

Ahora que ya se han explicado las características de los puertos de comunicación que posee el PLC, pasaremos a detallar los comandos asociados al envío y recepción de datos. Se le recuerda al lector que el puerto uno esta predefinido con protocolo *Modbus*, es por ello que no es necesario advertir que habrá una transacción en ese estándar, en los demás puertos sí.

A continuación vienen 4 apartados esenciales para llevar acabo la migración de protocolos. Se expondrá la forma de enviar datos en Modbus y en ASCII y también se explicará la recepción de datos en ambos modos.

4.3.1 Descripción de envió de datos en protocolo Modbus

Si se desea realizar un envío de información, primero se establece a quien va dirigido el mensaje definiendo el número de esclavo. (Ver figura 25, *Sending Data Setup*)

Posteriormente se especifica cual es el código de función, este puede ser, forzar una o varias bobinas o escribir en uno o varios registros. Para estudiar a detalle el tema de códigos de función, el lector puede revisar el tercer capítulo de esta tesis.

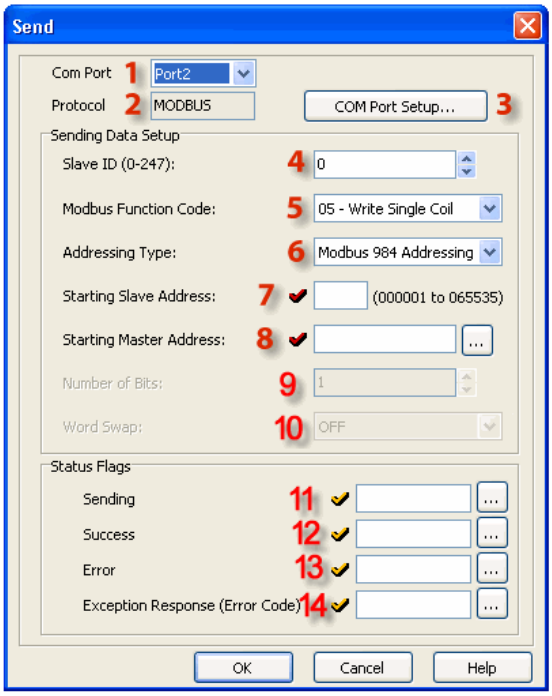
Después se estipula el tipo de direccionamiento, que puede darse en tres opciones Modbus hexadecimal, Modbus 984 y *Click Addressing*. Los dos primeros sirven para comunicarse con otros equipos y el tercero únicamente para comunicaciones entre PLCs Click. (Ver *Addressing type* en figura 25).

Luego viene un detalle muy importante, en el casillero de *Starting Slave Address* se coloca la dirección de memoria en la cual, el esclavo que reciba el mensaje deberá empezar a copiar los datos o forzar los estados de acuerdo a la información que esté recibiendo y en el casillero de *Starting Master Address* se coloca la primera dirección de memoria desde la cual se tomarán los datos de referencia para que posteriormente sean enviados. En el

siguiente casillero de *number of bits* se indica la cantidad de estados que se desean enviar desde el maestro hacia el esclavo seleccionado.

Por último existe la posibilidad de utilizar los *Status flag*, estos indicadores son muy usados al momento de coordinar que equipo de la red utilizará el bus de comunicación. Los estados *sending* y *success* alertan si el mensaje aún está enviándose o si ya ha finalizado exitosamente, se utilizan con frecuencia pues un equipo no entrará en el bus mientras otro aún este transmitiendo.

Figura 25- Opciones de configuración para envío de datos en protocolo Modbus



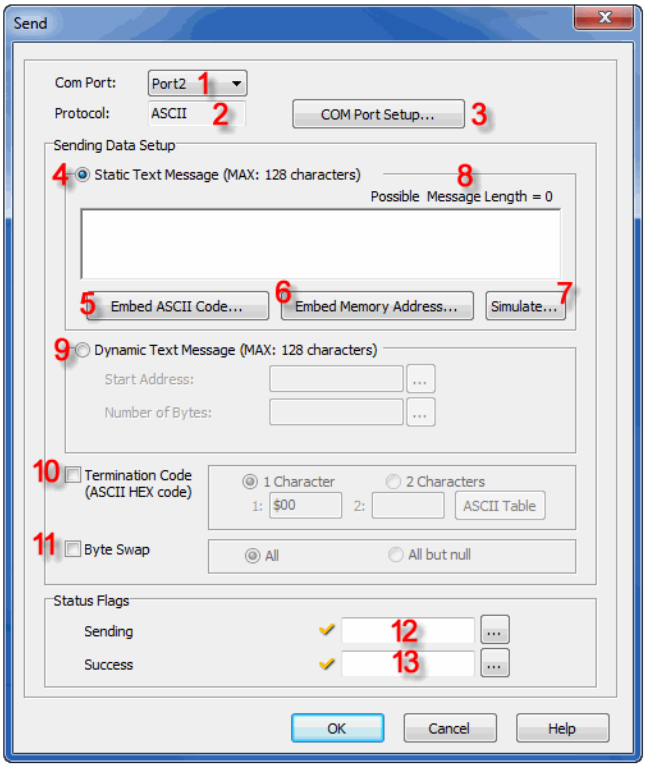
Fuente: Tutorial Software Click Programming

4.3.2 Descripción de envió de datos en modo ASCII

Ahora bien, para definir un envío en modo ASCII, primero se escoge si se trata de un mensaje estático o dinámico. (Ver *Sending data Setup* en figura 26) El primer caso siempre envía la misma trama de caracteres. En la segunda opción se anexa un dato de la memoria del PLC, este dato puede ser de contenido variable o se modifica ante un evento o condición programada.

Al igual que el caso anterior existe la posibilidad de utilizar los *Status flag* y sirven para coordinar el uso del bus de comunicación. En la presente tesis, el algoritmo que se ejecuta la comunicación con el panel de riego Valley utiliza en gran cantidad los *Status flag* con el fin de evitar conflictos en el bus.

Figura 26- Opciones de configuración para envío de datos en modo ASCII



Fuente: Tutorial Software Click Programming

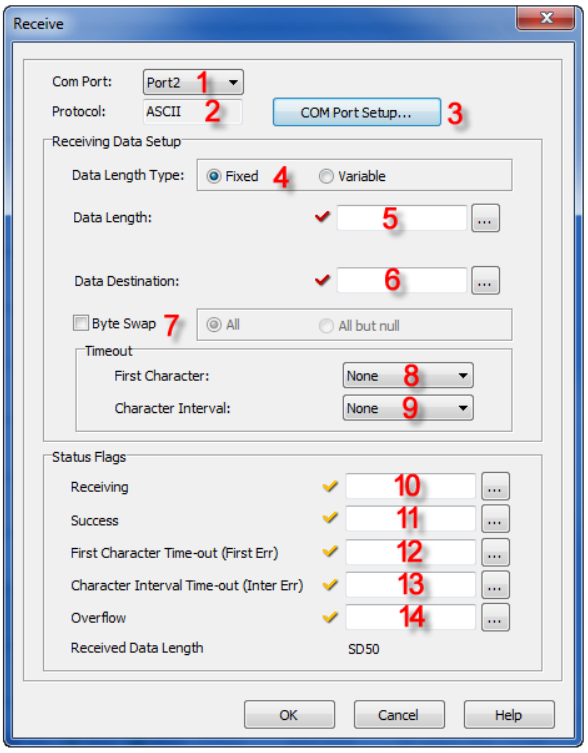
4.3.3 Descripción de recepción de datos en modo ASCII

En la figura 27, se muestra la ventana de recepción de datos en modo ASCII. Los puntos más importantes cuando se llama a esta opción son; elección del protocolo, Modbus o ASCII, para el segundo caso se debe definir la longitud del mensaje y el lugar de destino.

El requisito *Data Length* prepara al PLC para recibir cierto número de bytes, si el mensaje en curso tiene una longitud mayor a la que se le preparó al PLC, solamente se tomarán en cuenta los datos que se definió en este casillero, el valor máximo para el equipo es de 128 caracteres.

El modo de recepción debe ser compatible con la trama del mensaje y el lugar de destino debe ser coherente con el formato de almacenamiento, por ejemplo no se puede recibir el dato de un número y tratar de guardarlo en una dirección de una bobina.
El mismo software en general está protegido contra este tipo de errores y se le advierte al usuario marcando en color rojo los *checks* adyacentes del casillero en cuestión.

Figura 27- Opciones de configuración para recepción de datos en modo ASCII



Fuente: Tutorial Software Click Programming

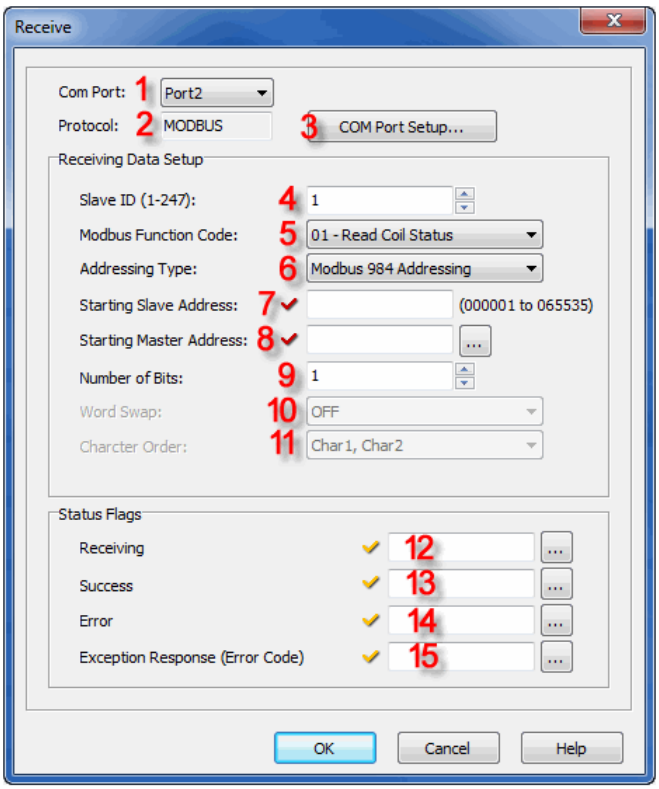
4.3.4 Descripción de recepción de datos en modo Modbus

En la figura 28, se muestra las opciones para configurar la recepción de datos en protocolo Modbus. Para ello es necesario indicarle al PLC que número de nodo es él, en la red. Después se escoge la función de lectura, esta puede ser leer un estado, leer un registro, leer varios estados o leer varios registros.

Posteriormente se establece el tipo de direccionamiento, el cual puede ser en formato hexadecimal o utilizando la misma nomenclatura de los PLCs Click.

Para retener los datos de la comunicación, se debe especificar cuáles son los datos de memoria que están siendo transmitidos y en donde serán guardados. Para complementar los parámetros de la recepción se indica la cantidad de bits si es que se envían estados o la cantidad de bytes si es que se usan las funciones de registros.

Figura 28- Opciones de configuración para recepción de datos en modo Modbus



Fuente: Tutorial Software Click Programming

4.4 Algoritmo de comunicación entre panel de riego Valley y PLC

La etapa de mayor exigencia en cuanto a migración de protocolo fue precisamente crear el algoritmo que se ejecutaría en el PLC. El algoritmo debía simular al programa original que comanda los pivotes de riego. La actualización y lectura de datos es información importante para mantener monitoreadas las estructuras de riego, por eso las subrutinas creadas están asociadas a las funciones de mayor uso.

Al empezar el proyecto, los conocimientos de lenguaje escalera eran mínimos, por ello hubo una gran inversión de tiempo en aprender a usar esta herramienta. Se empezó con programas sencillos, fáciles de comprobar y verificar con las borneras y leds a disposición. A medida que avanzaba la dificultad de los ejercicios de prueba, estos requerían ser implementados con opciones adicionales que ofrece el PLC progresivamente hasta usar los bloques de comunicación. En la etapa final de aprendizaje se llegó a usar hasta tres PLCs en una red para estudiar la forma de hacer el intercambio de datos.

Para hacer los ensayos de comunicación con el panel de riego primero se usó el *Software Base Station 2* este es el programa original de los pivotes y se logró extraer los comandos que utiliza para controlar las estructuras de riego. Las tramas digitales fueron verificadas con el uso del *software Serial Port Monitor* hasta agotar los recursos de la tarjeta electrónica. Todas las oraciones fueron archivadas para su posterior análisis.

Concluida la etapa de recolección de comandos del panel de riego se realizaron otro tipo de pruebas con el *software Docklight*. Estos experimentos consistían en enviar mediante el puerto de comunicación una oración en el mismo formato y esperar la respuesta del panel.

Los resultados fueron los que se esperaban ya que el panel Valley respondía según el comando. Estos experimentos fueron muy satisfactorios.

Ahora era turno de implementar estas mismas peticiones en el PLC, en primera instancia los envíos se hicieron en 'vacío' es decir, se creaba una rutina en el PLC tal que, envié solo una oración y después se volvía a leer adrede su mismo puerto para verificar el contenido.

Posterior a estas pruebas, se agregó el bloque de recepción de datos. La manera de verificar es direccionando la respuesta ASCII en espacios de memoria que puedan almacenar estos datos. El usuario lee estos datos desde el *text view* y se empieza la decodificación.

Ya en esta etapa, se tenía las tramas que solicitaban por ejemplo; el estado del sistema, en el cual indica presión, voltaje, ángulo de giro, el sentido de giro, también se tenía las tramas para forzar la inyección de agua y cambiar la profundidad de la lámina de riego.

Todas funcionaban y en todas se recogían las datos que se querían, pero lo hacían aún de forma independiente. Faltaba resolver la actualización constante de datos y en caso se desee forzar algún cambio no se pierda la sincronización de la rutina en curso.

4.4.1 Main Program "Panel de riego"

El archivo que se ejecuta en el PLC lleva el nombre de 'Panel de Riego' y tiene la extensión 'ckp' este algoritmo alberga dos grupos, el programa principal y las subrutinas. (Ver figura 29 parte superior izquierda del árbol de contenido)

El primero ha sido diseñado con el fin de coordinar las peticiones *Modbus* con las tareas propiamente dichas del PLC. En situaciones en que no se requiera forzar algún estado, el PLC se mantendrá preguntando y actualizando los datos del panel de riego. Si el usuario de la GUIDE de Matlab necesita cambiar un valor, como por ejemplo la lámina de riego, el algoritmo del PLC detiene la rutina y realiza la variación seleccionada, una vez que termina, retoma la actualización de datos.

El segundo grupo son las subrutinas, conformadas por la misma estructura de código, diferenciándose del mensaje que se transmite. Además de eso se ha considerado que la decodificación también sea una subrutina y así dejar las tareas de sincronización al *main program*.

4.4.2 Subrutina "*Report System"*

Pensando en un sistema SCADA, existen datos que deben ser siempre visualizados por los operadores de los pivotes. En este caso hay un ciclo repetitivo de pregunta y respuesta del PLC hacia el panel de riego en el que se está actualizando los datos de: presión, voltaje, ángulo de giro, salida de agua, lámina de riego y sentido de giro. Esta función la realiza la subrutina *report system.*

El panel responde por segunda vez si es que no recibió una confirmación al primer intento, el manejo de la construcción de la respuesta en dinámico es difícil pues el cálculo del *checksum* a pesar de no ser complicado aritméticamente se torna tedioso pues el tratamiento matricial lo hace una tarea complicada para el PLC. Por ello se consideró conveniente no trabajar en esta parte y dejar que el panel responda por segunda vez y

guardar la última oración en direcciones de memoria predefinidos para luego decodificarlos. (Ver figura 31, último paso si no existe una acción forzada)

La contraparte es el tiempo de espera del segundo envió ya que demora aproximadamente 3 segundos, por lo tanto la actualización aumenta su periodo a 6 segundos, con experimentos de prueba y error el *timer* de la quinta línea de la figura 30 se estableció a 6500 ms, con ello se da cierto rango para algún evento adicional.

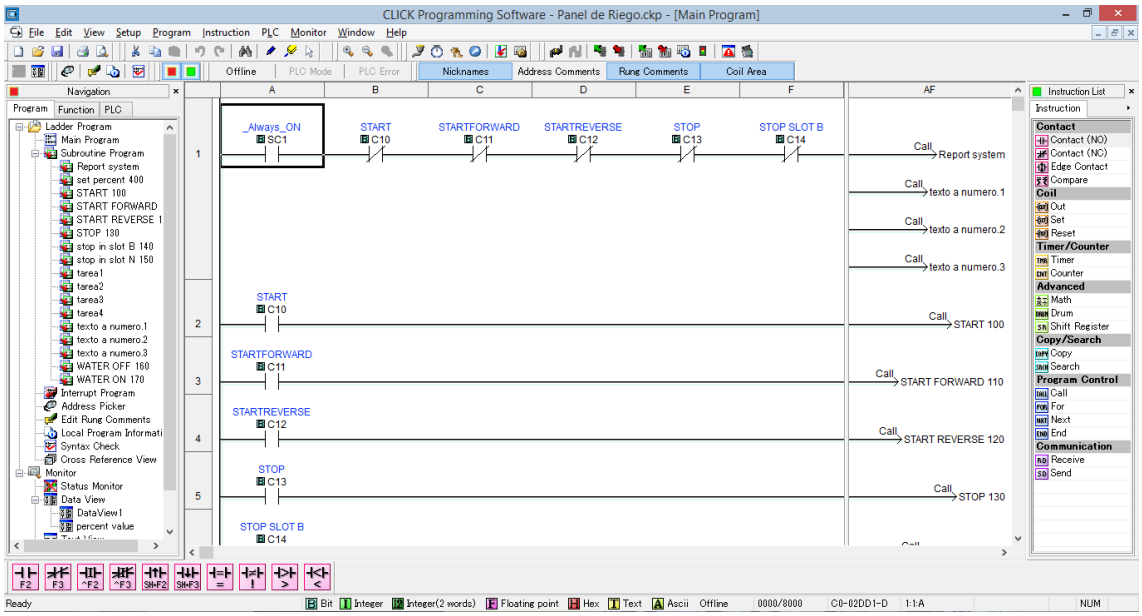
Terminado este tiempo se usa el flanco positivo del bit asociado a este evento, el cual se ha denominado 'reenganche' ya que este cambio de estado, reinicia el *timer* principal de la primera línea.

Dos características importantes que se deben mencionar es que el envío de la trama '(000999SD;SA;RE9B($0D)' correspondiente a la actualización de datos corre en paralelo con un *timer* de 50ms para que al terminar este tiempo se habilite el bus para la recepción del mensaje del panel. El valor de 50ms también fue ensayado a diferentes tiempos pero ese valor fue el que más se adecuo a la aplicación.

La segunda característica es que los espacios de memoria preparados para guardar los caracteres ASCII del panel empiezan en el TEXT600 y se reservan 115 espacios en adelante, no por ser de ese tamaño la respuesta sino porque el panel responde dos veces la misma trama.

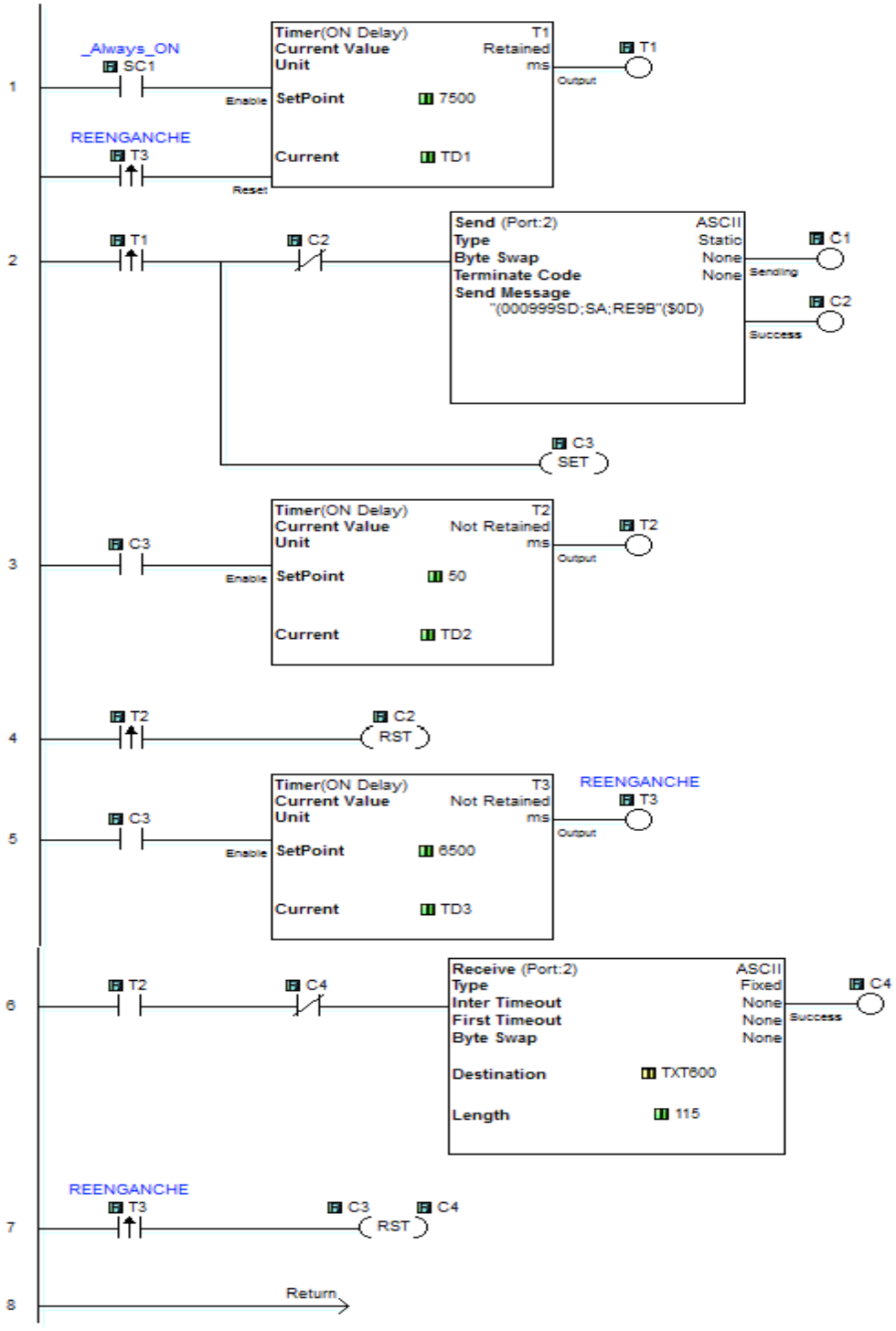
El tratamiento que se hace a partir del dato TEXT658 es la conversión de texto a valores enteros, para posteriormente ser almacenados en direcciones que van a ser solicitadas por peticiones *Modbus.*

Figura 29- Algoritmo instalado en el PLC Click, a la izquierda el desplegable de las subrutinas y en la pantalla principal las primeras 5 líneas de código



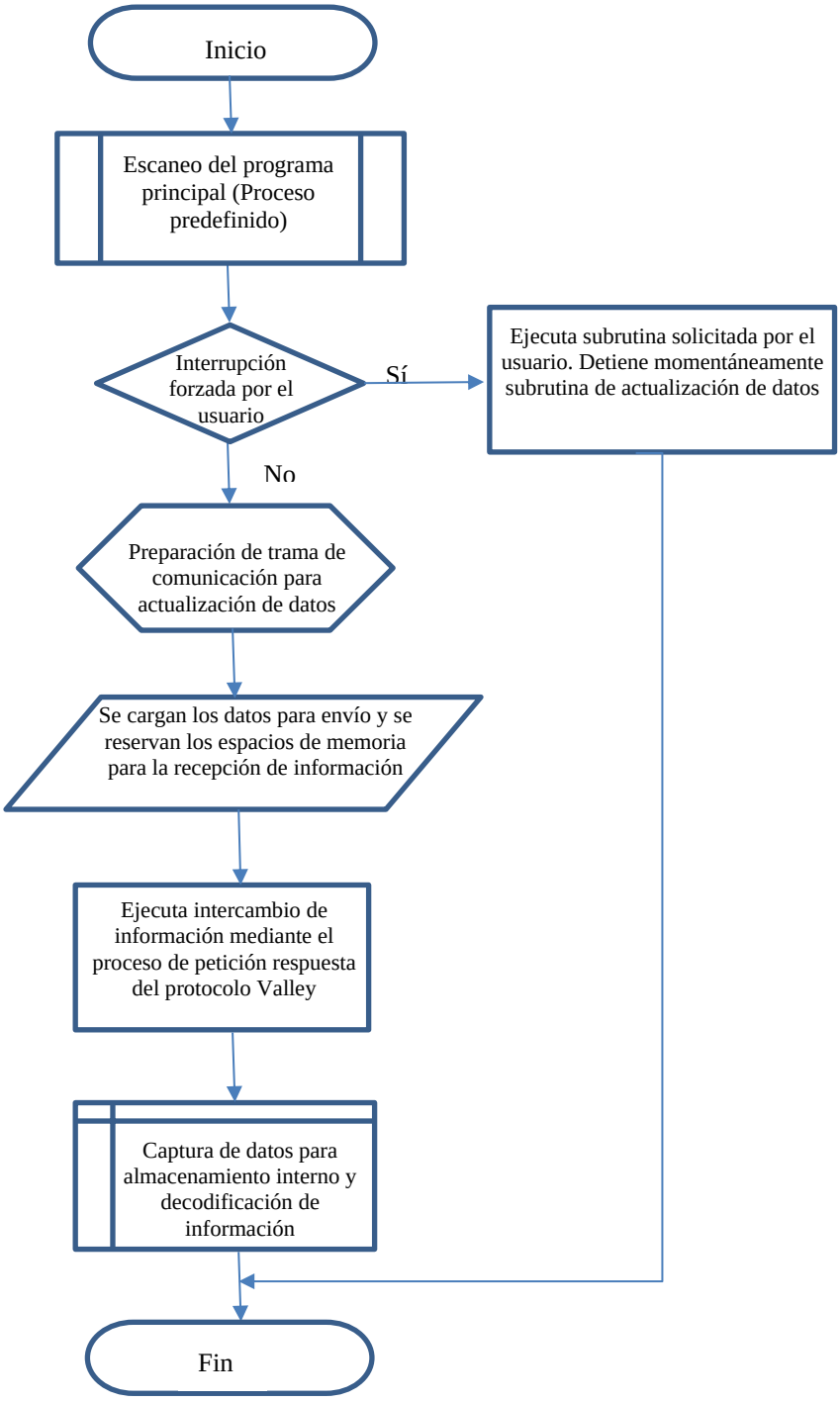
Fuente: Elaboración propia

Figura 30- Líneas de código correspondiente a la subrutina de actualización de datos



Fuente: Elaboración propia

Figura 31- Diagrama de flujo para la subrutina de actualización de datos



Fuente: Elaboración propia

4.4.3 Subrutina "*Water on*"

En condiciones normales, la única subrutina que se repite indefinidamente es la explicada en estado del sistema o *"report system"*. La condición para que una subrutina diferente pase a desarrollarse depende del usuario, tal y como muestra el condicional del diagrama de flujo de la figura 31. Si una tarea es llamada entonces corta el bucle principal y pasa a ejecutarse, termina y retoma la actualización de datos. Se usa la misma secuencia de envío y respuesta, por ello es la misma plantilla de código.

La diferencia a parte del mensaje '(000999POW59($0D)' son los tiempos de ejecución, ya que ante este cambio el panel responde afirmando que ha ejecutado la orden. Aproximadamente estos eventos tarda 1500 ms y determinan el *set point* del *timer* de la quinta línea de la subrutina de la figura 32.

Como método de verificación debería comprobarse la trama de respuesta del panel, pero como esto va ser visualizado por el usuario en el siguiente *report system* no se ha creado una subrutina de verificación. A modo de ejemplo se ha explicado la función activar agua, las demás subrutinas solamente se diferencian en la trama de envío por eso es suficiente explicar una.

4.4.4 Subrutina "*set percent"*

El riego de la planta varía con el crecimiento, ello significa que en diferentes etapas del desarrollo necesita también diferentes cantidades de agua. El panel de riego Valley puede modificar en una escala de cero a cien por ciento la profundidad de la lámina de riego. Los valores de cero y cien son ajustables a un cierto valor en milímetros, es decir que se puede establecer el inicio y fin de la escala acorde a la recomendación del cultivo.

Esta parametrización no es alcance de la tesis pues son datos que son manejados por los agrónomos o especialistas en el tema. Sin embargo, sí es objetivo del proyecto que el valor del *depth,* pueda ser modificado desde la sala de control. Este valor puede ser variado directamente desde la perilla del panel o por comandos.

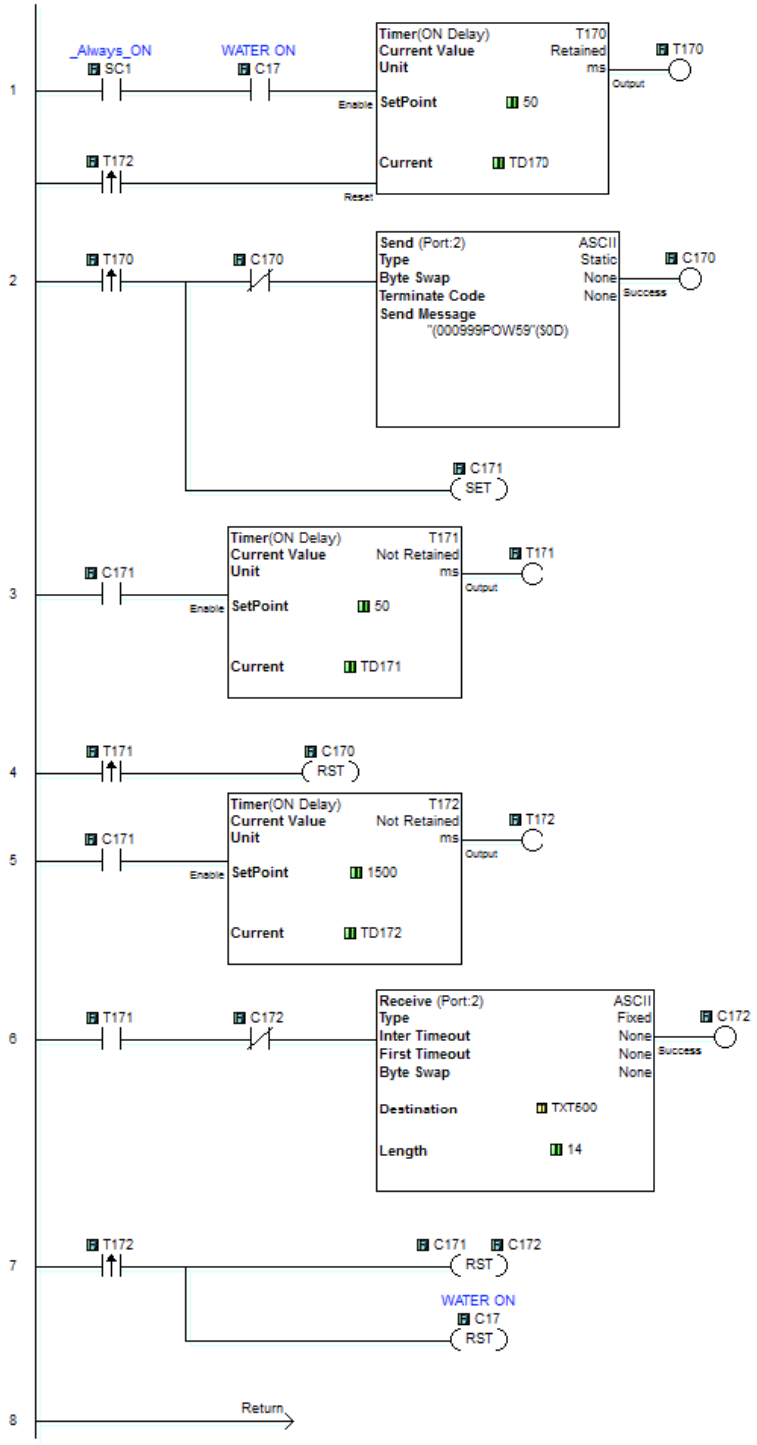
Para lograr que sea manejable desde la GUIDE se correlacionó una barra deslizable a la misma escala de cero a cien y la variación de uno en uno, de esta manera se reservan cien espacios de memoria y cada bit está asociado a un valor. (Ver figura diagrama de flujo de la figura 33)

Por ahora solo es necesario explicar el algoritmo del PLC, ya que en el siguiente capítulo se detalla la programación en Matlab para llevar de forma ordenada las oraciones *Modbus*. La figura 34, es un resumen de las sentencias que tiene esta subrutina. La primera línea de código reúne los condicionales para que se ejecute cualquiera de las tramas de envío. Los bits denominados 'modificar porcentaje n' están incluidos en el *main program* y detienen el *report system* al igual que las subrutinas anteriores.

Cuando un bit en específico es seleccionado, el PLC ejecuta el envío como cualquier otro mensaje ASCII. Se puede apreciar que desde la dirección C200 comienza el orden ascendente hasta llegar a C300, ultima dirección para el valor de 100, la cual energiza el bloque de envío de datos con la trama '(000999SP10097($0D)'.

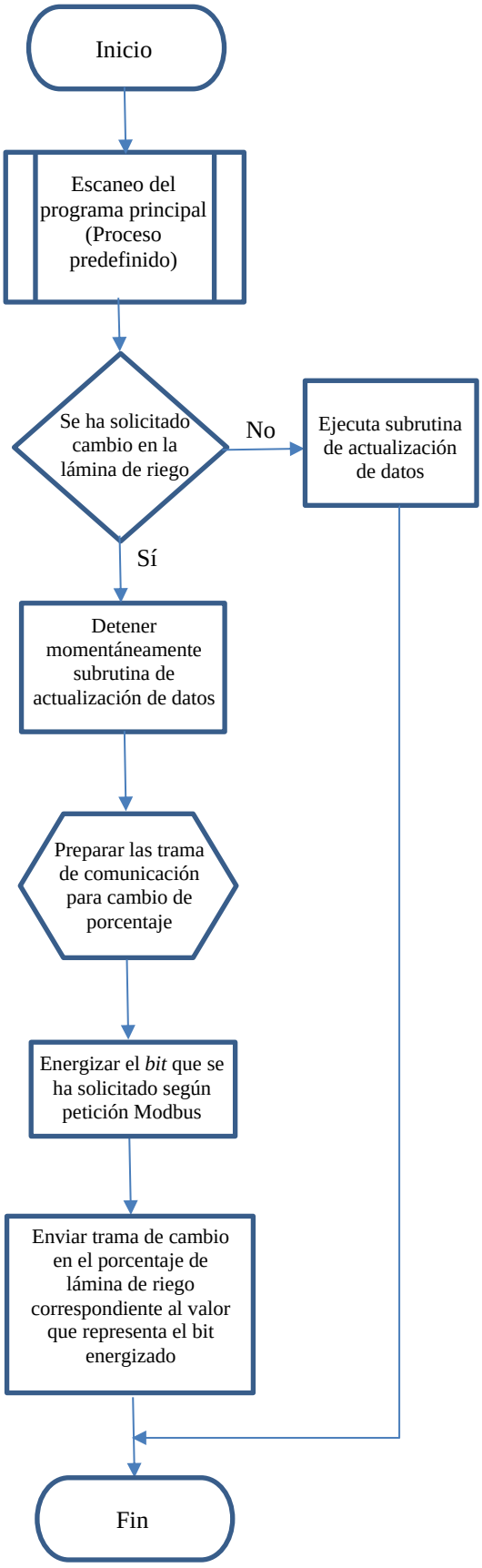
La diferencia con las demás subrutinas es que tiene 100 líneas de código adicionales, pero que al fin y al cabo solamente están modificando el valor del porcentaje. La estructura del *timer* de 50 y 1500 milisegundos es la misma. El reinicio una vez acabada la tarea también es igual.

Figura 32- Líneas de código correspondiente a la subrutina water on



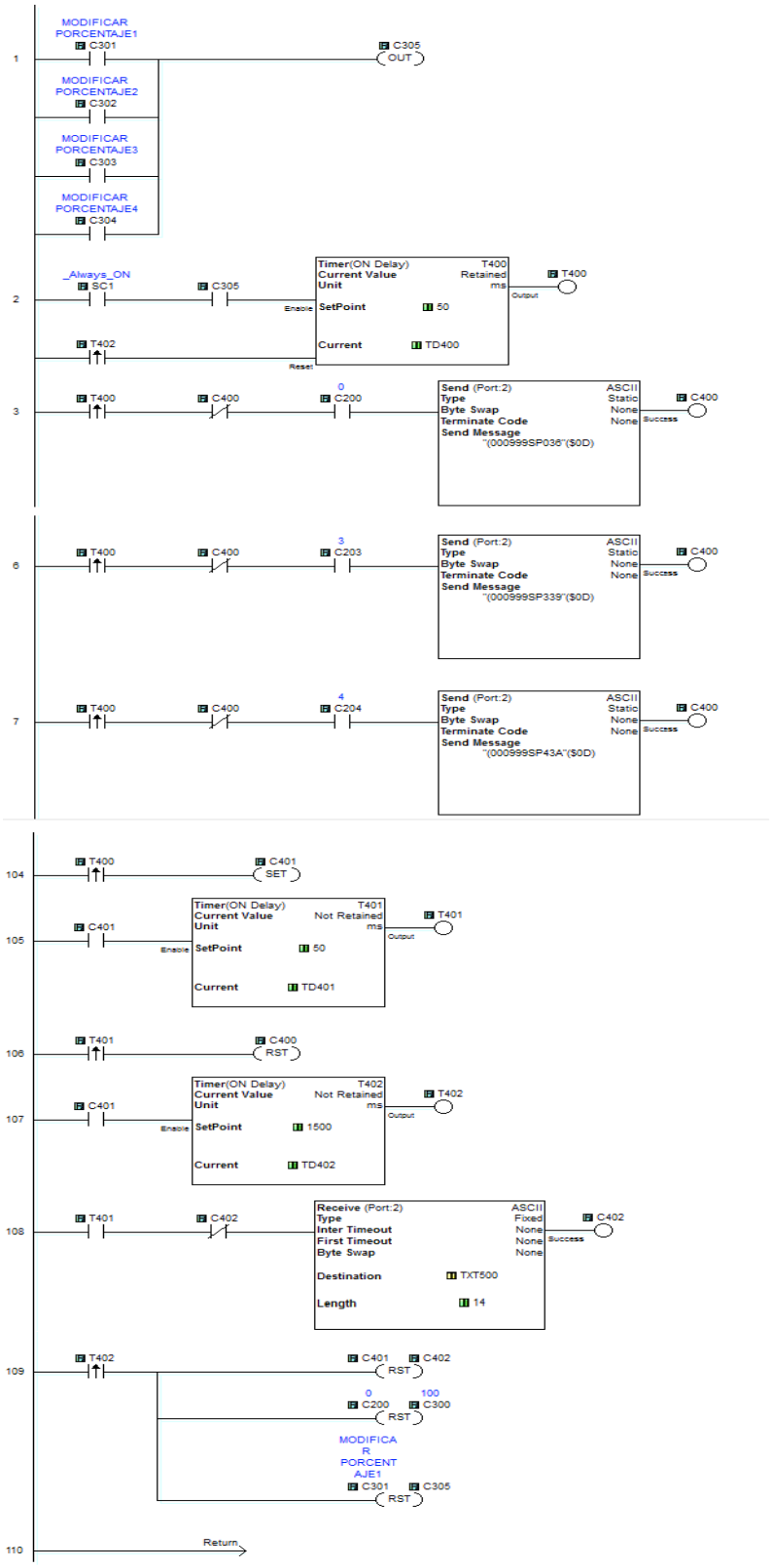
Fuente: Elaboración propia

Figura 33- Diagrama de flujo correspondiente a la subrutina 'set percent'



Fuente: Elaboración propia

Figura 34- Líneas de código correspondiente a la subrutina ‘set percent’



Fuente: Elaboración propia

4.4.5 Subrutina "*texto a número*"

Las tramas de recepción van dirigidas a dos bloques. El primer bloque del cual se extraen los datos de interés se almacenan a partir del TEXT600 así como se ha explicado anteriormente. También hay un grupo de respuestas que se asumen que siempre se han ejecutado y no es necesario verificarlas, estas se direccionan al TEXT500.

El panel de riego responde en el mismo formato siempre, por ello la ubicación de cada carácter se ha estudiado y se sabe a qué dato pertenece.

En la figura 35, podemos observar la sintaxis de comunicación, para el comando SR, esta oración viene definida por '(000999SR08($0D)', el *checksum* tiene el valor de 08 hexadecimal.

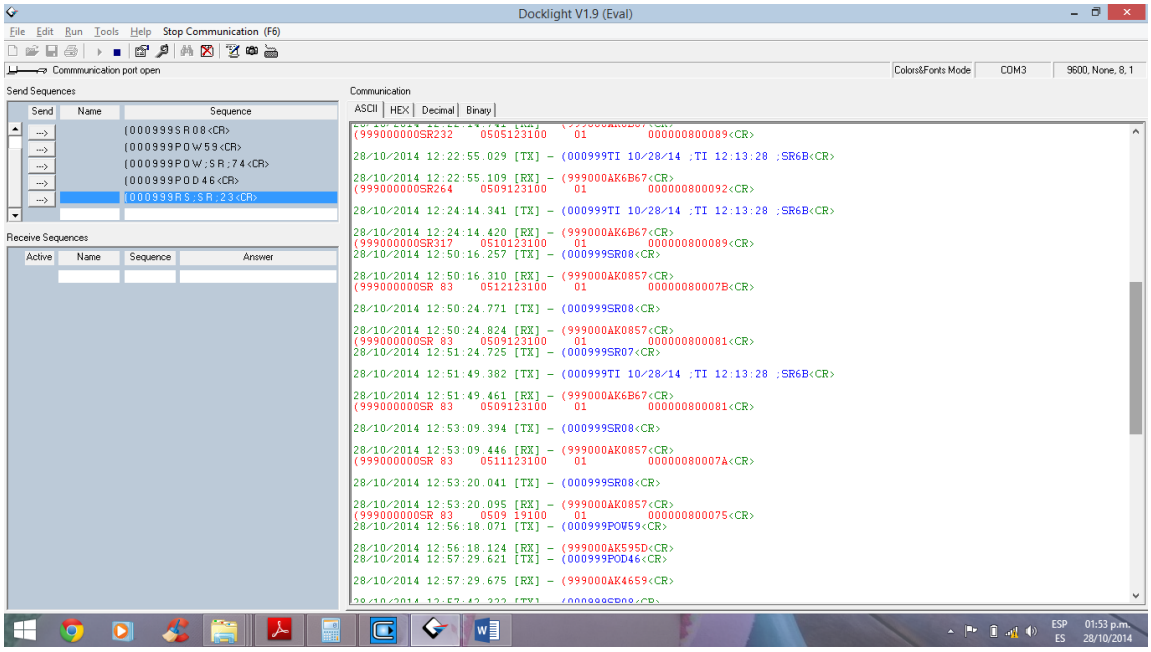
Ante esta petición el panel de riego Valley PRO2 da primero una respuesta de reconocimiento del comando con los caracteres 'AK', luego viene la trama con los datos esperados, esta es '(999000000SR 83 0509 19100 01 000000800075($0D)'

Petición: '(000999SR08($0D)'

Respuesta: '(999000000SR 83 0509 19100 01 000000800075($0D)'

Para decodificar, se compara cada carácter con el valor numérico y luego es tratado en el sistema decimal como unidad, decena o centena según corresponda. Por ejemplo para el valor de voltaje el cual es valor entre los caracteres 83 y 191 se tiene el texto 509, pero este valor lo transformaremos a 509 numérico.

Figura 35- Ensayos de comunicación entre Docklight y Modulo Valley PRO2

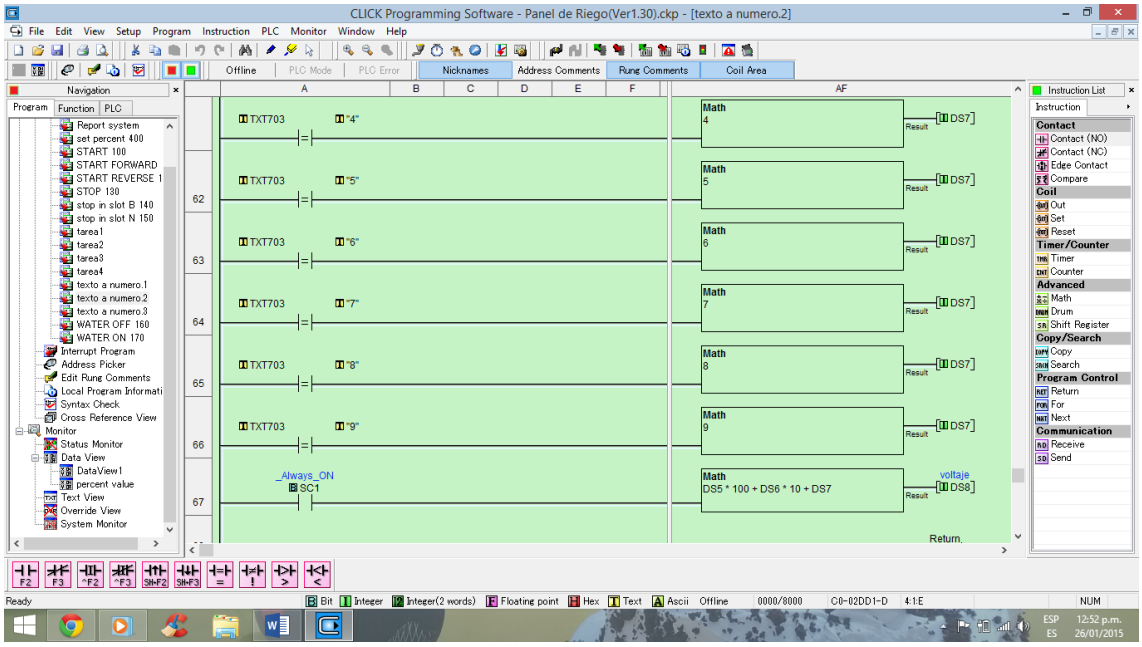


Fuente: Elaboración propia

El primer carácter (el de las unidades) lo comparamos con todos los valores de 0 a 9. Así también hacemos con las decenas y con las centenas. Para el condicional verdadero la siguiente acción es asignar ese valor numérico. Como paso final se reúne en una suma cada componente y se almacena en una dirección de memoria el valor entero. En la figura 36, observamos algunas líneas de código para esta tarea y en la figura 37, se esquematiza el diagrama de flujo correspondiente.

El dato del DS7 son las unidades, DS6 el de las decenas y DS5 el de las centenas, acorde a la línea de código 67 de la figura 36, el espacio de memoria DS8 almacena el número completo. Por otro lado la secuencia de trama TXT703 a la TXT705 son los datos recibidos en formato texto y que han sido transformados a un valor numérico.

Figura 36- Subrutina texto a número 2



Fuente: Elaboración propia

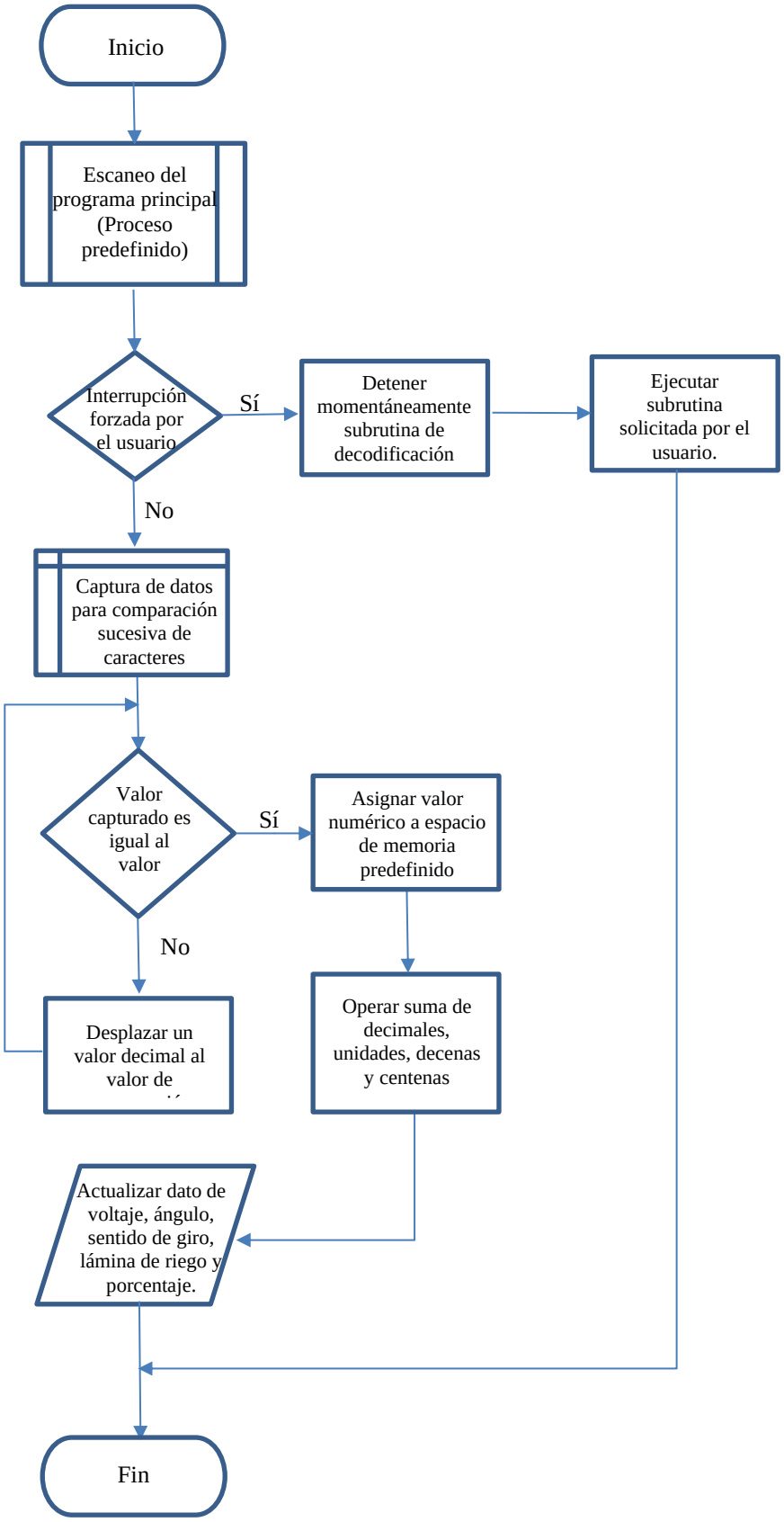
4.4.6 Subrutina “Tareas”

Unos de los propósitos del proyecto es hacer compatible un equipo (el panel de riego) con las comunicaciones industriales estándar, otro es hacer más versátil y aprovechar la capacidad de los PLCs para el control de procesos y su participación en la red.

Adicional a esto, el usar Matlab no fue tanto un propósito pues suelen usarse otros programas para los SCADAs sino más bien se vio como una oportunidad pues ya se tenían conocimientos previos de programación en este entorno.

Las subrutinas de tareas son calculadas por Matlab y los resultados son transferidos al PLC. El PLC simplemente inicia los contadores para ejecutar el inicio del riego a la hora y fecha que el operario de los pivotes ha programado. Matlab calcula cuanto tiempo falta para ese evento y envía los valores de los *set points* al contador de horas y minutos. En la figura 38, se observa el algoritmo desarrollado para esta opción.

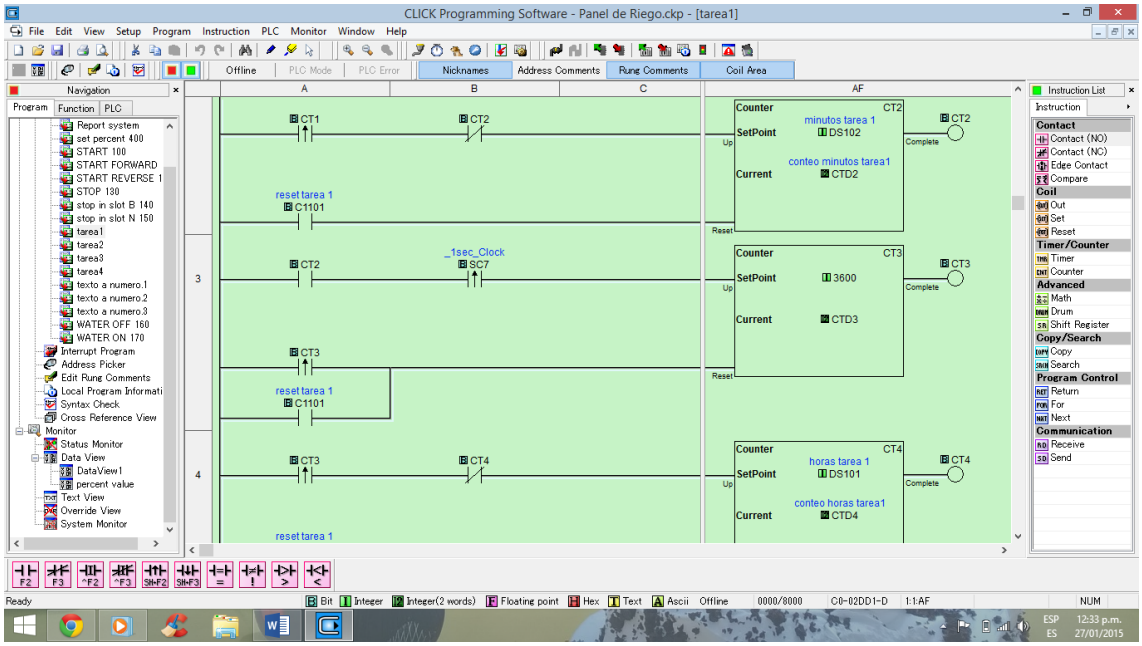
Figura 37- Diagrama de flujo de las subrutinas texto a números



Fuente: Elaboración propia

En la figura 38, la línea de código 2 y 4 contienen los contadores de minutos y horas respectivamente, cuando ambos llegan a su *set point* se da acción sobre las subrutinas *water on* y *start.* Las cuatro subrutinas de tareas hacen lo mismo, por tanto cada pivote puede tener en cola hasta cuatro riegos programados.

Figura 38- Subrutina Tarea 1



Fuente: Elaboración propia

Capítulo 5

Interfaz gráfica de pivotes de riego usando Matlab

Las matemáticas es el lenguaje común de gran parte de la ciencia y de la ingeniería. Matrices, ecuaciones diferenciales, arrays de datos y gráficas son los bloques de construcción básicos de las matemáticas aplicadas y de MATLAB. Es la base matemática fundamental que hace que MATLAB sea accesible y potente. Un profesor, que es un gran admirador de MATLAB, nos dijo:

¨La razón por la que MATLAB es tan útil para procesamiento de señales es que no fue diseñado específicamente para procesamiento de señales sino para matemáticas¨

Matlab ha sido utilizado en muchos campos diferentes:

- *Una estudiante graduada en física analiza y visualiza datos de sus experimentos con campos magnéticos de superconductores.*
- *Un restaurador de parques de atracción conocido internacionalmente lo utiliza para modelar los sistemas de control de sus fuentes de agua.*
- *Una compañía de televisión por cable investiga esquemas de codificación y compresión para la TV digital.*
- *Un fabricante de equipamiento deportivo modela los golpes de golf.*
- *Un estudiante de tercer grado aprende las tablas de multiplicar.*

En todos estos casos, y en miles más, el fundamento matemático de Matlab resultó útil en lugares y aplicaciones mucho más allá de los que contemplamos originalmente.

The Math Works Inc. MATLAB

Edición del estudiante – Versión 4 Página 3

En esta ocasión, Matlab será participe de la creación de un sistema de monitoreo y control de pivotes de riego, una aplicación más que se une a las listas donde se puede usar este potente *software*. La capacidad de cálculo de Matlab y la opción de crear presentaciones en pantalla en las que, intrínsecamente se puede designar algoritmos o tareas especiales a cada ítem que se muestre, habilita la opción de usarlo como un programa de supervisión de la instrumentación instalada en campo.

5.1 GUIDE con Matlab

GUIDE *(Graphical User Interface for a Development Environment)*, interfaz gráfica del usuario para un entorno de desarrollo, es una herramienta para creación de aplicaciones y tienen como fin controlar software gráficamente, lo que elimina la necesidad de aprender un lenguaje o los comandos de programación para el futuro usuario.

Los programas de Matlab que cuentan con una interfaz GUI automatizan una tarea o cálculo, por ello las pantallas de acceso suelen contener controles, tales como menús, barras de herramientas, botones, barras de desplazamiento. También se pueden anexar muchos productos de Matlab como: ajuste de curvas, procesamiento de imágenes, procesamiento de señales, etc.[23]

5.1.1 Funcionamiento de una GUI

Cuando el usuario selecciona un conjunto de datos en el menú emergente y luego hace clic en uno de los botones se está invocando a una función que usa los datos seleccionados, a su vez cada ítem tiene subrutinas que son ejecutables en Matlab, conocidas como devoluciones de llamada o *Callback*. Se llaman así por el hecho de que " vuelven a llamar" a Matlab para pedirle que use sus galerías en los comandos que ha seleccionado el usuario.

La ejecución de cada *callback* se activa por una acción particular del usuario, tales como: pulsar un botón de la pantalla, hacer clic en un botón del ratón, la selección de un elemento de menú, escribiendo una cadena o un valor numérico o pasando el cursor sobre un componente. Este tipo de programación se refiere a menudo como la programación orientada a eventos, una característica de este tipo de programación es que se activa por eventos externos al *software*. La secuencia de comandos o instrucciones deberán estar siempre dentro de un bloque asociado a un ítem.

Para lograr la coordinación de la GUI con los paneles de riego y los diferentes comandos que se pueden ejecutar, se hicieron diversas pruebas las cuales se detallan en el presente capítulo. A continuación se describe la interfaz gráfica desarrollada para el monitoreo de riego, estas contienen los comandos de mayor uso del panel de control. Primero se explicará la pantalla que observará el usuario cuando decide monitorear un pivote específico, luego a los pivotes en general y después se expondrán los algoritmos de programación así como las instrucciones o sentencias que abarcan cada archivo. Se detalla la forma de almacenamiento de información en direcciones de memoria del PLC y los vectores fila construidos para realizar intercambios en protocolo *Modbus* desde Matlab.

5.2 GUI de monitoreo de pivotes de riego

5.2.1 GUI de un pivote específico antes de la recepción de datos

Al iniciar la GUI todos los bloques de datos están vacíos, se debe hacer clic en ¨Conectar¨ para empezar a recopilar los datos del PLC que comanda dicho pivote, por lo tanto después de la recepción de datos ya se observará el estado de los bloques. (Ver figura 39)

- El bloque de estado actual, hace referencia si el pivote se encuentra con el flujo de agua activado o no.

[23] http://www.mathworks.com/help/matlab/creating_guis/what-is-a-gui.html

- El bloque de dirección actual, hace referencia si el giro del pivote es horario u antihorario.

- El bloque de estado de autorranque, hace referencia si el contacto del panel se encuentra energizado o no.

- El bloque de ángulo, hace referencia a la cantidad de grados sexagesimales que va girando el pivote respecto de la última posición que se ha dejado, el conteo de grados será ascendente si está girando antihorario y será descendente en el caso contrario.

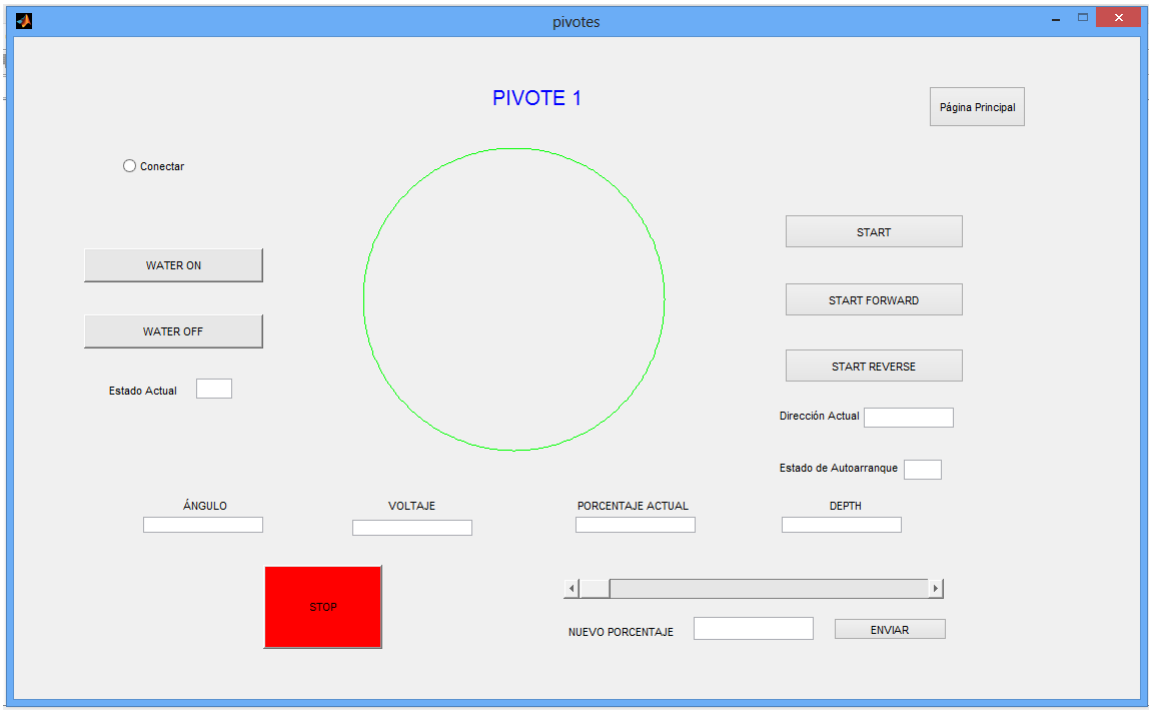
- El bloque de voltaje, hace referencia al potencial eléctrico que recibe el panel, este valor se ajusta a la escala que necesite el usuario.

- El bloque de porcentaje, hace referencia a la fracción porcentual de la lámina de agua que se está regando.

- El bloque de *depth* o profundidad, hace referencia a la lámina de riego, la cual tiene relación directa con la velocidad de la última torre del pivote. El bloque de porcentaje y *depth* son inversamente proporcionales dado que, mientras más rápido gire el pivote menos agua habrá recibido el cultivo.

- El bloque de nuevo porcentaje es capaz de variar la lámina de riego actual, este valor debe ser calculado por un especialista del cultivo, esta persona deberá ser capaz de determinar la lámina de riego que necesite la planta en base a los factores agrícolas en la que se encuentre.

Figura 39- Muestra la pantalla que se genera al hacer clic en un pivote determinado



Fuente: Elaboración propia

5.2.2 GUI pivote específico en actualización constante

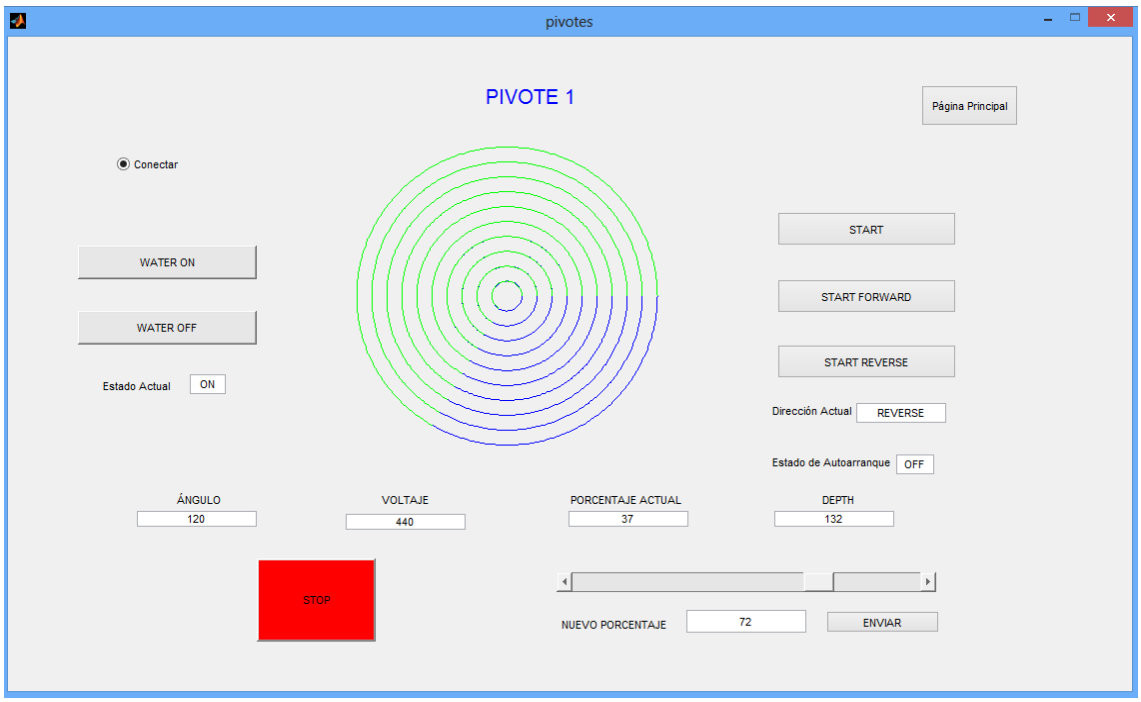
En la figura 40, se logra visualizar la respuesta ante la petición de datos generado por Matlab, ahora los bloques poseen un valor que nos informa la situación de esa variable. Las circunferencias concéntricas que se plasman son el resultado de dibujar el ángulo que va girando el pivote en color azul, haciendo referencia al área regada y el ángulo que le falta por girar el cual corresponde al área no regada, de color verde.

Para este caso, el ángulo que ya ha girado es de 120º en sentido horario, la referencia de cero grado sexagesimales para el dibujo es eje positivo de las abscisas del plano cartesiano, debe aclararse que esta referencia no corresponde necesariamente con la referencia del pivote, pues el panel solo informará de los grados desplazados desde la última posición de reposo.

Respecto al porcentaje 72 que se forzará (Ver figura 40, parte inferior derecha), al ser mayor que el actual (37%), significa que se utilizará más agua pues la lámina de riego durara más ya que se ha disminuido la velocidad de giro de la última torre del pivote.

Cabe resaltar que los demás *pussbothon* están habilitados sin necesidad de hacer clic en conectar, se le recomienda al usuario que si desea usar las funciones asociadas a los botones, no haga clic en conectar ya que este actividad demanda tiempo y memoria a Matlab y en consecuencia el tiempo de respuesta se verá extendido. Una buena práctica es que, antes de monitorear el riego, se definan los parámetros del pivote como lo es: el sentido de giro y la lámina de riego. Adicional a esta explicación se le indica al lector que la única diferencia con los demás pivotes es el título si es que solo se considera la presentación de pantalla, pero en cuanto a la programación hay variación en los nombres de variables, direcciones de memoria y vectores fila que se transmiten para la recopilación de datos.

Figura 40- Muestra la pantalla que se genera una vez conectado con el PLC



Fuente: Elaboración propia

5.2.3 GUI del cultivo general antes de la recepción de datos

Esta GUI es la página principal (Ver figura 41) aquí el usuario podrá observar de manera panorámica el estado de los pivotes instalados en campo, solamente para efectos didácticos se ha escogido esta disposición con 6 pivotes a los cuales se les puede designar tareas de riego con horas de inicio específicas.

Cada círculo hace referencia al área de riego que puede abarcar cada pivote. En el centro de cada pivote está el *pushbotton* hipervínculo a la página del pivote seleccionado.

El bloque para designar tareas de riego, es un conjunto de espacios predifinidos para recibir la fecha y hora en que se desee accionar al pivote, se le recomienda al lector que antes de iniciar la conexión con todos los demás pivotes, se introduzcan las tareas de riego, esto para no generar conflictos con el *bucle* de recopilación de datos que lleva a cabo Matlab.

La lista desplegable de la izquierda es para seleccionar al pivote, para este caso observaremos desde el pivote uno al seis. En la lista desplegable de la derecha se puede escoger la tarea de la uno a la cuatro, esto indica que a cada PLC se le pueden dejar en memoria hasta 4 tareas en cola. El algoritmo de control asume que el usuario será lo suficiente precavido para indicar estas fechas y horas, por ejemplo cuidar de no colocar fechas anteriores a la actual, o colocar inicios de riego cuando el pivote aún se encuentre efectuando alguno.

El botón de ‘conectar’ inicia el sondeo en la red. Cada trama de respuesta de la petición que se hace panel a panel traerá información del ángulo actual, dato necesario para dibujar en la GUI la actividad el pivote. Si no existe un riego en el momento solo presentará círculos verdes concéntricos. Cuando la trama de recepción sea un vector vacío, se mostrará un mensaje en el espacio de trabajo de Matlab indicando ‘pivote desconectado’ más un número, el número hace referencia al pivote que no ha respondido a la solicitud del maestro. Así el pivote se encuentre estático y sin regar debe responder su estado, por eso al emerger el mensaje ‘pivote desconectado’ en el *workspace* significará una alarma de error en la comunicación.

5.2.4 GUI del cultivo general actualizando datos de forma constante

El usuario de la GUI podrá observar de forma panorámica en qué estado se encuentran los pivotes de riego, los detalles de la presentación en pantalla (Ver figura 42) se describe a continuación.

- Pivote 1, ya ha regado 120º de su cultivo y lo está haciendo en sentido horario.
- Pivote 2, ya ha regado 180º de su cultivo y también lo está haciendo en sentido horario.
- Pivote 3, se encuentra estático, no está regando dado que las circunferencias en su totalidad están de color verde.

- Pivote 4, ya ha regado 270º de su cultivo y lo está haciendo en sentido horario.

- Pivote 5, ya ha regado 45º de su cultivo y lo está haciendo en sentido antihorario.

- Pivote 6, ya ha regado 120º de su cultivo, también en sentido antihorario.

El rol de tareas pendientes, muestra las ultimas 4 tareas pre asignadas, no quiere decir que no se pueden programar más, como se explicó anteriormente cada PLC puede guardar 4 tareas en cola.

En caso se desee reemplazar una tarea a un pivote, simplemente se ingresa la nueva hora y fecha con el mismo número de tarea y el PLC al recibir estos datos, reiniciará el conteo actual, en otras palabras, el algoritmo del PLC está preparado para cambiar sus valores así ya haya empezado.

En la imagen actual el rol de tareas es el siguiente:

- El pivote 2, deberá empezar a regar a las 3 am Del día 2 de octubre.

- El pivote 4, tiene dos tareas, la primera empieza 8:30 pm del 30 de setiembre y la otra tarea está programada para el 2 de octubre a las 4: 15 am.

- El pivote 6, deberá empezar a regar a las 6:38 am del 29 de setiembre.

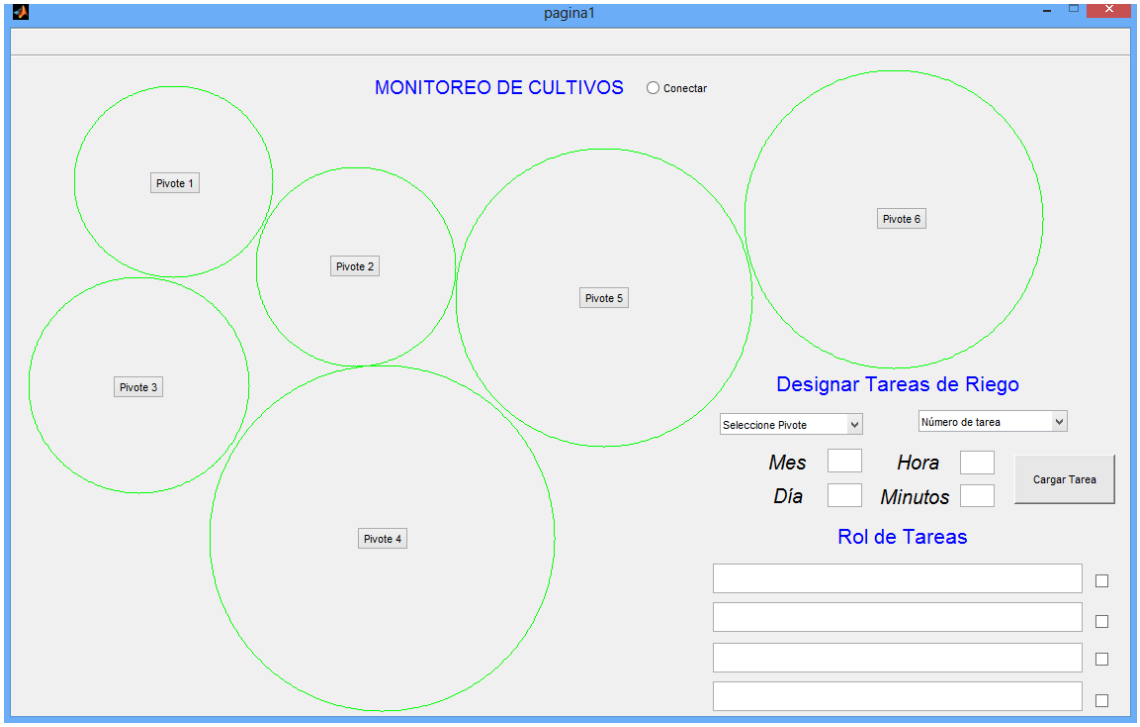
El lector podrá inferir que el formato para ingresar el dato de horas es en el formato de 24 horas día, como acción adicional se tiene que hacer un *check* en el recuadro adyacente a cada tarea, con esto damos pase al envió de la trama, que contiene el tiempo restante para iniciar el riego.

Si el usuario, tuvo una equivocación al ingresar los datos, no hay problema, simplemente se sobrescribe en el número de tarea en la cual ha ocurrido el error. Debemos recordar que Matlab no es un programa diseñado para control industrial sino para modelamiento matemático, es por ello que el conjunto de sentencias de código, no es más que la adecuación de vectores de datos a una dirección y función específica, donde la aritmética es la mayor carga computacional para Matlab. La versatilidad de usos que brinda el *software* tiene como condición, adecuarlo de la mejor manera a cada proyecto en el que se le use, en efecto para el protocolo *Modbus*, la primera adecuación es cálculo del CRC de un vector fila, pasando por el envío y recepción serial de una trama de datos así como el ploteo de anillos concéntricos simulando las trayectorias de las estructuras de riego.

Como contraparte de la comunicación serial es que, se debe esperar la recepción de datos para poder continuar con el sondeo de los siguientes pivotes. Si hubo algún cambio se verá reflejado hasta que la petición *Modbus* pase de nuevo por dicho esclavo. En líneas de código se traduce en esperar un nuevo bucle para actualizar las variables que están asociadas a cada nodo del cultivo. Matlab siempre va cumplir en orden estricto los comandos que se le den, por ello si se desea efectuar un cálculo durante la realización de una tarea o se precisa de hacer una subrutina, se debe programar cuidadosamente los eventos para retornar al algoritmo principal. Situaciones como las anteriores, existe una

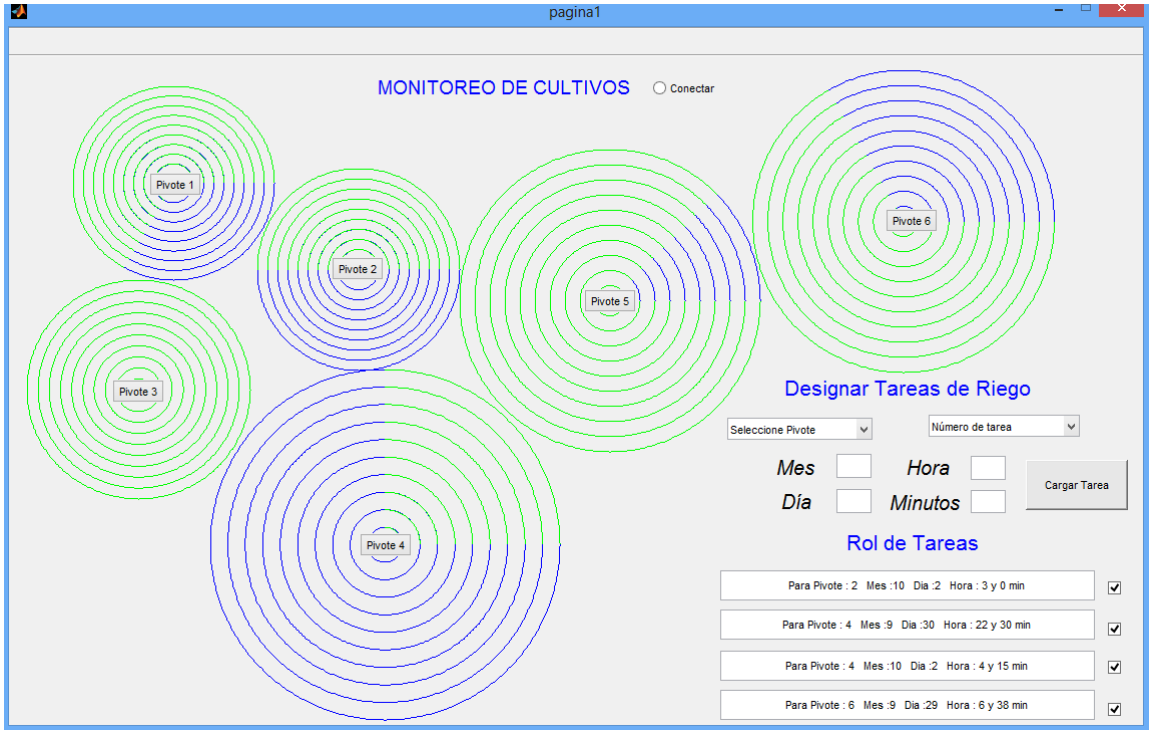
buena cantidad en la presente tesis. Los puntos que vienen explican la lógica a cada archivo desarrollado en la GUIDE de Matlab.

Figura 41- Muestra la pantalla que se genera al llamar al archivo 'pagina1'



Fuente: Elaboración propia

Figura 42- Pantalla de monitoreo general recopilando los datos de los PLCs de la red



Fuente: Elaboración propia

5.3 Calculo del CRC de una trama de datos

Cuando el modo RTU es usado en la comunicación, el campo comprobación de error contiene un valor de 16 *bits* implementado como dos *bytes* de 8 *bits*. El valor de comprobación de error es el resultado de un cálculo de Comprobación Cíclica Redundante o "CRC" (*Cyclical Redundancy Check*) realizado sobre el contenido del mensaje. Este algoritmo de comprobación, ya ha sido utilizado por el asesor por eso ha sido empleado directamente en el conjunto de archivos utilizados en la GUIDE de Matlab. Los *bytes* del campo CRC son añadidos al mensaje original como último campo de la trama. Se añade primero el *byte* de orden bajo del campo, seguido del *byte* de orden alto. En la figura 43, el lector podrá ver este conjunto de sentencias que generan el CRC de un vector de datos numéricos.

Figura 43- Algoritmo CRC para 16 bits. En el proyecto se ha denominado 'calculacrc'

```
function [msj]=calculacrc(message)
N = length(message);
crc = hex2dec('ffff');
polynomial = hex2dec('a001');

for i = 1:N
    crc = bitxor(crc,message(i));
    for j = 1:8
        if bitand(crc,1)
            crc = bitshift(crc,-1);
            crc = bitxor(crc,polynomial);
        else
            crc = bitshift(crc,-1);
        end
    end
end

lowByte = bitand(crc,hex2dec('ff'));
highByte = bitshift(bitand(crc,hex2dec('ff00')),-8);

msj = message;
msj(N+1) = lowByte;
msj(N+2) = highByte;
return
```

Fuente: Archivo de trabajo de Mathworks Inc.

Esta función genera el CRC de un vector fila o de un vector columna, independientemente si cumple o no con la sintaxis de una trama *Modbus*. Se debe ingresar en la variable 'message' el vector al cual se le desea hallar el CRC, luego se llama a la función 'calculacrc' y la salida de esta función es entregada en la variable 'msj' la cual tiene dos elementos adicionales, el byte alto y el byte bajo que conforman el CRC de esa trama. El formato numérico de entrada y salida del vector es en base decimal.

5.4 Parámetros del puerto serial de comunicación

Ahora corresponde que la comunicación deba exteriorizarse de la PC, por ende se debe utilizar las funciones de Matlab que habiliten y se relacionen con el puerto serial. Los comandos que permiten esto son: *instrreset, serial, fopen, fwrite, fread, fclose, instrfind*.

También es de uso común los comandos *fprint* y *fscan*, sin embargo presentaron problemas al escribir y leer el 00 hexadecimal (NUL) ya que detenían la trama de trabajo. Se explica en el siguiente ejemplo la situación.

Petición o solicitud que debía salir por el puerto serial:

11 04 00 08 00 01 B2 98

Al usar el comando *fprint*, solo enviaba 11 04, al detectar un 00 HEX en la trama, se dejaba de enviar el resto de datos y por tanto se perdía la integridad del mensaje.

Respuesta que se debía recibir por el puerto:

11 04 02 00 0A F8 F4

Al usar el comando *fscan*, solamente se recepcionaba los datos 11 04 02, de igual manera que el comando anterior, al detectar el 00 HEX se detenía la recepción de datos y el mensaje no llegaba completo. Se hicieron muchas consultas al respecto y la solución fue cambiar el comando de lectura y escritura, por *fwrite* y *fread* respectivamente. En la figura 44, se muestra las líneas de código para habilitar el puerto COM, con las tareas de enviar y recibir en forma serial los datos de trabajo.

Existe una protección adicional en este archivo dado que se verifica que la recepción no sea un vector vacío, en caso se de esta situación, el pivote no ha logrado entregar el mensaje a tiempo o esta desconectado de la red, si la recepción no es un vector vacío se procede a verificar la integridad del mensaje con la comprobación del error.

Los valores que definen al puerto serial, deben estar acorde con el dispositivo de conexión, para nuestro caso son los valores de la tabla 15.

Al definir estos parámetros se logra hacer compatible el enlace físico entre PLC y el puerto de la PC. Para que ambos logren entenderse, deben usar el mismo protocolo, tanto dispositivo como computador, que en nuestro caso es *Modbus* RTU.

Tabla 15- Datos de comunicación del PLC

Velocidad de transferencia	38400 *bits*/segundo
Bits de datos	8 *bits*
Bit de *stop*	1
Tiempo de espera	1 segundo
Control de flujo	Sin control de flujo

5.5 Inicio de comandos de la GUIDE

Continuando con la descripción, se tomará como referencia al pivote número cuatro, la figura 45 es el inicio de la GUI del monitoreo del cultivo y las sentencias que se observan son ejecutadas previo a la presentación en pantalla.

Al elegir a un pivote se dibuja el círculo externo, el cual hace referencia al contorno del área circular. Para nuestro ejemplo se fija el valor de ‘handles.n’ igual a 4, ‘n’ es el número del esclavo y es hacia donde estarán direccionadas todas las peticiones *Modbus*.

Para los demás archivos de los pivotes lo que cambia es el valor de esta variable, la cual al estar declara desde el inicio lo torna más sencillo de programar. Por otro lado el nombre del eje ‘areacircular4’ sí requiere que se modifique en las líneas donde participe y sea cambiado por ‘areacircularn’, donde ‘n’ es el nuevo pivote instalado.

Figura 44- Algoritmo para habilitar la comunicación serial desde Matlab. En la presente tesis se le ha denominado ‘comunicar’

```
trama=calculacrc(message);
instrreset;
PS = serial('COM4');
set(PS,'BaudRate',38400);
set(PS,'DataBits',8);
set(PS,'StopBits',1);
set(PS,'Parity','odd');
set(PS,'TimeOut',1);
set(PS,'FlowControl','none');
respuesta=[];
message=[];
verificacion=[];
fopen(PS);
fwrite(PS,trama);
respuesta=fread(PS);
n=length(respuesta);
n=n-2;
if n<0
   disp('Pivote desconectado');
   respuesta=zeros(1,13);
   trama(1);
else
   for i=1:n
       message(i)=respuesta(i);
   end
   verificacion=calculacrc(message);
   respuesta=respuesta';
   k=respuesta-verificacion;
   if k==0
   else
      disp('falla en recepciòn de mensaje')
   end
end
fclose(PS);
delete(PS);
clear PS;
instrfind;
```

Fuente: Elaboración propia

5.6 *Pushbotton* para comandos predefinidos

En cada presentación de pivote individual, se muestran ocho *pushbotton*, seis estan asociados a las ordenes: *water on*, *water off*, *start*, *start forward*, *start reverse* y *stop*, uno está asociado a la variación de la lámina de riego y el último es una conexión para regresar a la página principal. Los siete primeros son tramas *Modbus* hacia el panel de riego y el último es una rutina interna para mostrar otra pantalla. En la tabla 16, se muestran las direcciones *Modbus* de las variables utilizadas en los *pushbotton*.

Figura 45- Algoritmo de inicio de cada GUIDE

```
% --- Executes just before pivotes is made visible.
function pivotes_OpeningFcn(hObject, eventdata, handles, varargin)
pintar_fondo=0:1:360;
xf=cosd(pintar_fondo);
yf=sind(pintar_fondo);
plot(handles.areacircular4,xf,yf,'g')
handles.n=4;
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pivotes (see VARARGIN)

% Choose default command line output for pivotes
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

Fuente: Elaboración propia

Tabla 16- Datos para construir los vectores fila de las tramas Modbus

FUNCIÓN EN PANEL	PROPERTY INSPECTOR	DIRECCIÓN MODBUS VECTOR DE SALIDA	DIRECCIÓN MODBUS HEXADECIMAL	CÓDIGO DE FUNCION	ESTADO FORZADO	ESTADO EN VECTOR DE SALIDA
WATER ON	PUSHBOTTON 1	64 16	40 10	05	1 LÓGICO	255 00
WATER OFF	PUSHBOTTON 2	64 15	40 0F	05	1 LÓGICO	255 00
START	PUSHBOTTON 3	64 09	40 09	05	1 LÓGICO	255 00
START FORWARD	PUSHBOTTON 4	64 10	40 0A	05	1 LÓGICO	255 00
START REVERSE	PUSHBOTTON 5	64 11	40 0B	05	1 LÓGICO	255 00
STOP	PUSHBOTTON 7	64 12	40 0C	05	1 LÓGICO	255 00

Fuente: Elaboración propia

El *pushbotton* seis de cada GUI tiene una construcción diferente y se ahondara en el apartado siguiente. Por lo pronto se le recuerda al lector que en el capítulo III se documenta la construcción de las tramas *Modbus* y el detalle de la sintaxis para el protocolo. Por ahora solamente corresponde ordenar el vector en la forma correcta, hallar el CRC y enviarlo por el puerto de comunicación. Se procederá con un ejemplo del comando *start* del pivote número cuatro. (Ver figura 46)

Figura 46- . Ejemplo de trama Modbus generada desde Matlab

```
Command Window
>> handles.n=4;
   message=[handles.n 05 64 09 255 0];
   comunicar

trama =

     4     5    64     9   255     0    73   173

Warning: Unsuccessful read: The specified amount of data was not returned within the Timeout
period.
Pivote desconectado
fx >>
```

Fuente: Elaboración propia

El vector de entrada ‘message’ aún no posee el CRC. El vector de salida ‘trama’ ya lo contiene y es esta variable es la que sale por el puerto serial. Matlab responde con el aviso

de ‘Pivote desconectado’ ya que la matriz de respuesta no ha recolectado datos durante el tiempo de espera máximo.

Esto sucede porque el nodo cuatro no fue configurado adrede para corroborar la protección. Los demás *pushbotton* seis de cada pivote tienen la misma estructura y la única variación, es el primer valor del vector.

5.7 *Pushbotton* para comandos especiales

El *pushbotton* seis (Ver figura 40, parte inferior derecha) es el botón que lleva el texto ‘enviar’ y está asociado a la variación del porcentaje de la lámina de riego. En el panel Valley es una perilla que nos permite llevar de 0 a 100% el valor del *depth*. Este *pushbotton* trabaja con el dato capturado del *slider*[24] que lo acompaña. Gracias a que se ha podido reducir la oración ASCII con la que se comunica el panel a un *bit* por valor de porcentaje, se torna más sencillo forzar el cambio en la lámina de riego.

Las tramas que se van a enviar primero recogen el dato que el usuario defina en la barra deslizable y se correlaciona con los cien espacios de memoria separados para esta función. Para nuestro caso, se hace una aproximación al entero inferior en un rango de cero a cien, después se ubica la dirección que ocupa tantas posiciones como el porcentaje deseado.

El *bit* que fuerza un valor de cero, tiene su dirección *Modbus* hexadecimal en 40 C7, por ello si deseamos tener un valor de 45% se deben contar 45 espacios de memoria desde la posición 40 C7. La figura 47, es el algoritmo que realiza la tarea de ubicación y designación del valor del espacio de memoria que se utilizará. Con esta lógica se aprovecha el orden ascendente en que se programó esta subrutina en el PLC.

Figura 47- Algoritmo que determina la dirección de memoria del bit correspondiente al valor obtenido en el slider

```
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
direccion_memoria=handles.slider1;
part1=[handles.n 05];
part3=[255 0];
s=dec2hex(hex2dec('40c7')+direccion_memoria);
bytehigh=[s(1) s(2)];
bytelow =[s(3) s(4)];
part2=[hex2dec(bytehigh) hex2dec(bytelow)];
message=[part1 part2 part3];
comunicar;
guidata(hObject,handles);
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Fuente: Elaboración propia

[24] El slider es una herramienta grafica que habilita un deslizador en una barra de longitud constante, cuando el deslizador está al inicio toma el valor mínimo y cuando llega a su posición más alejada toma el valor máximo.

5.8 *Pushbotton* para navegación

Los desarrolladores de SCADAs deben estar atentos a la distribución de gráficos, al contraste de color, a la manera de mostrar los mensajes emergentes, a la ubicación de botones y en gran medida a la navegación dentro del conjunto de pantallas. Por navegación se entiende el ir de una pantalla a otra para ver algún dato en específico u observar algunos gráficos con mayor claridad y tamaño.

En cuanto a la navegación dentro del sistema de monitoreo de pivotes se tiene una interfaz para todos los pivotes juntos y una interfaz para cada uno. En cada presentación se rescatan datos diferentes y el moverse de una a otra acarrea tareas diferentes, por este motivo el *pushbotton* ocho tiene la función de coordinar este requerimiento. En la figura 48, se describe las sentencias que engloba este botón.

En cada GUI de pivote individual se presenta el botón *Página Principal* (Ver figura 40, parte superior derecha) el cual enlaza a la pantalla general del cultivo. Este botón se usa para regresar a la pantalla principal, es decir a la GUI donde se observa de forma panorámica el estado de los pivotes, primero se llama al archivo que tiene el nombre de *pagina1*, luego se cancela el *bucle* actual, esto se hace para no crear conflictos con el *bucle* siguiente.

Durante la realización del programa, se hicieron gran cantidad de intentos para lograr la coordinación de las subrutinas. Sin lugar a duda fue una de las etapas más difíciles de este proyecto y que demandó bastante tiempo. El problema fue que al navegar por las demás páginas no se dejaba de hacer el algoritmo del archivo anterior y tampoco había un reconocimiento rápido del estado de algunos botones. Utilizando redundancias para preguntar y pausas para forzar el valor de algunas variables se logró solucionar el problema.

Cuando se hace clic en el botón *página principal*, se asigna el valor de cero a la variable *Conectar4*, tanto por sentencia directa como por búsqueda de objeto (Redundancia). La pausa colabora en dar un tiempo para actualizar (Forzar) el directorio de variables *handles.*

Figura 48- Bloque de sentencias que reúne el pushbutton8

```
% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
pagina1;
handles.Conectar4=0;
pause(0.1)
poll4=findobj(gcbf, 'tag', 'Conectar4');
set(poll4, 'value',0);
pause(0.1)
guidata(hObject,handles);
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Fuente: Elaboración propia

5.9 *Radio button* para la recepción de datos

Cuando está abierta la interfaz de un pivote individual, todas las tramas *Modbus* son direccionadas a dicho nodo, de manera continua y repetitiva se hace la pregunta del estado del agua, del sentido de giro, del voltaje y demás parámetros con el fin de acelerar la presentación en pantalla de dichos datos.

Esto no es más que un *bucle* de repetición indefinida que se termina cuando se regresa a la página principal o cuando se cancela adrede la actualización de datos al retirar la viñeta del *radiobutton*. En tabla 17, se detalla las direcciones de memoria del PLC con las que trabaja este *bucle*.

Tabla 17- Direcciones de memoria del PLC para la recopilación de información

Variable de Trabajo	Tipo de variable	DIRECCIÓN MODBUS VECTOR DE SALIDA	DIRECCIÓN MODBUS HEXADECIMAL	CÓDIGO DE FUNCION
Sentido de giro	Booleano	64 31	40 1F	01
Estado del agua	Booleano	64 29	40 1D	01
Auto Arranque	Booleano	64 30	40 1E	01
Angulo	Entero	00 16	00 10	03
Voltaje	Entero	00 17	00 11	03
Porcentaje	Entero	00 18	00 12	03
Depth	Entero	00 19	00 13	03
Decimal Depth	Entero	00 20	00 14	03

Fuente: Elaboración propia

La diferencia en cuanto a los *pushbotton* es que ya no se fuerzan estados, sino que se leen los valores de las variables. Otra diferencia es que el código de función cambia de valor. Se le recuerda al lector que en el protocolo *Modbus* el valor de uno y tres son para leer el estado de una bobina y de varios registros respectivamente.

Como acotación adicional, se aclara que el valor de *depth* se solicita en dos partes, la primera es el valor entero y el segundo es el valor en decimal, luego se operan para retener el valor correcto. Los registros se solicitan con una sola trama, es decir, se piden cinco datos empezando desde la dirección 00 10 hexadecimal.

La respuesta también regresa en un orden estable, por eso se tiene previsto que cada posición del vector respuesta se multiplique por un factor de conversión, con el fin de leer el valor en base decimal.

A los datos booleanos se les solicita uno por uno, guardando de esta manera el estado actual del *bit*. En la figura 49, se presenta un resumen de las principales líneas de código del *radiobutton Conectar4.*

Para que se inicie el *bucle* de la figura 49, se debe hacer clic en conectar, el cual aparece en la parte superior izquierda de cada interfaz de cada pivote individual (Ver figura 40) de esta manera el valor lógico del objeto es uno y se empieza a solicitar el paquete de datos.

Luego se realiza una gráfica con anillos concéntricos, simulando el avance del riego en el sentido de giro correspondiente, después se muestra en cada bloque de texto el valor capturado para ángulo, voltaje, porcentaje y *depth*. Este bloque de código, finaliza

preguntado si el *radiobutton* continua habilitado, si lo está vuelve hacer el algoritmo, en caso contrario se detiene.

Aquí Matlab presenta una desventaja ya que, si la desconexión se realizó durante la ejecución del bloque, el usuario tendrá que esperar que el computador llegue a este punto para tomar acción. Como atenuante a esta situación es la velocidad de procesamiento y cálculo que tiene Matlab, con ello el retardo a la respuesta es aceptable.

Sin embargo en el archivo de la página principal en que se envía el paquete de preguntas a cada uno de los pivotes en forma serial es desagradable el tiempo de espera. Durante las pruebas y ensayos se debía esperar hasta 23 segundos cuando se cortaba adrede la actualización. Para atenuar este problema se introdujo una pregunta del estado del *radiobutton* en cada paquete de solicitudes de cada pivote, es decir, que si se desea navegar por la demás páginas o se fuerza la desconexión ya no se tiene que esperar que el *bucle* entero termine, sino que debemos esperar a que termine el procesamiento de datos del último pivote, este tiempo es aproximadamente 3 segundos.

5.10 Designación de tareas para el inicio del riego

Es importante explicar que los inicios del riego están asociados a energizar los comandos de *water on* y *start*, básicamente el código de programación calcula cuanto tiempo falta para la hora indica por el usuario. Luego se construye la trama *Modbus* que registra los valores de horas y minutos para el evento y posteriormente son enviados al PLC.

Una vez recepcionados estos datos empezará el conteo y cuando se alcance los *set point* de tiempo se dará pase al riego. La figura 50, muestra el orden de recepción de los datos asignados por el usuario al seleccionar la tarea número uno, la única diferencia al seleccionar otra tarea es el valor de la variable *handles.tareas*.

El primer elemento guarda el número del pivote, es decir, hacia quien va dirigida la petición, luego en orden correlativo a la presentación de la GUI se almacena el mes, día, horas y minutos del riego. Como último elemento del vector se guarda el número de tarea. Designar la posición de la tarea es importante ya que depende de esto el uso de ciertos espacios de memoria. Cuando se dispone de estos datos y además son coherentes, el algoritmo que continúa se llevará a cabo sin problemas y sin mensajes de error.

Si el usuario asigna roles con fechas incoherentes, al procesar estos datos Matlab mostrará el error en el *workspace*, pero no quiere decir que el sistema colapse, la GUI continuará operativa y se podrán corregir los valores ingresados. Todo esto ocurre en la presentación panorámica del área de cultivo.

Los botones que aparecen en la GUI de la figura 42, por pertenecer a otra presentación pueden llevar los mismos nombres que ya existen en otras GUI tal es el caso del *pushbotton6* (Cargar Tarea) que toma el rol de capturar los bloques de textos, las listas desplegables las retorna a la posición inicial y muestra en los recuadros inferiores el formato de fecha en una sola oración.

Cuando se está en este punto solo falta hacer *clic* en el *checkbox* adyacente a cada oración, de esta manera el usuario llama a las funciones que calculan el tiempo restante y al código que construye la trama *Modbus.*

Figura 49- Bloque de sentencias para la actualización y visualización de datos

```
% --- Executes on button press in Conectar4.
function Conectar4_Callback(hObject, eventdata, handles)
handles.Conectar4=get(hObject,'value');
guidata(hObject,handles);
while   handles.Conectar4 == 1
        message=[handles.n 01 64 31 00 1];
        comunicar;
        sentido4=respuesta(4);
        message=[handles.n 01 64 29 00 1];
        comunicar;
        agua4=respuesta(4);
        message=[handles.n 01 64 30 00 1];
        comunicar;
        autoarranque4=respuesta(4);
        message=[handles.n 03 00 16 00 05];
        comunicar;
        matriz1=respuesta;
        angulo4=(matriz1(4)*256)+matriz1(5);
        voltaje4=(matriz1(6)*256)+matriz1(7);
        porcentaje4=matriz1(9);
        depth4=(matriz1(10)*256)+matriz1(11)+(matriz1(13)/10);
        clc
        .
        .
        .
         for i=0.1:0.1:1
             plot(handles.areacircular4,i*xf,i*yf,'b')
             plot(handles.areacircular4,i*xi,i*yi,'g')
         end
        .
        .
        .
        set(handles.edit1,'string', angulo4);
        set(handles.edit2,'string', voltaje4);
        set(handles.edit3,'string', porcentaje4);
        set(handles.edit4,'string', depth4);
        .
        .
        .
        if sentido4==1
           set(handles.edit7,'string', 'FORWARD');
        end
        if sentido4==0
           set(handles.edit7,'string', 'REVERSE');
        end
        pause(0.1);
        handles.Conectar4=get(hObject, 'Value');
end
guidata(hObject,handles);
```

Fuente: Elaboración propia

En la figura 51, se muestra el conjunto de sentencias que se ejecutan cuando ya se ha decidido enviar la tarea hacia el PLC haciendo clic en el *checkbox1*.

Figura 50- El pushbotton6 cargar tarea, reúne los datos en vectores filas

```
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)

pivotex=handles.lista;
mes=handles.edit2;
dia=handles.edit3;
hora=handles.edit4;
minuto=handles.edit5;
n=handles.tareas;

if n==1
    handles.tarea_recibida1=[pivotex mes dia hora minuto];

mostrar1=['Para Pivote : ',num2str(pivotex),'   ',
'Mes :',num2str(mes),'   ','Dia :',num2str(dia),'   ',
'Hora : ',num2str(hora),' y ',num2str(minuto),' min'];
set(handles.edit7,'string', mostrar1);
guidata(hObject,handles);

.
.
.

end
```

Fuente: Elaboración propia

Figura 51- Bloque de sentencias del checkbox1, este bloque ejecuta la subrutina de tiempo y construye el vector fila que será enviado al PLC

```
% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
handles.checkbox1=(get(hObject,'Value'));
fecha_tarea=[handles.tarea_recibida1(2) handles.tarea_recibida1(3)
handles.tarea_recibida1(4) handles.tarea_recibida1(5)];

if (fecha_tarea==0)
    disp 'Anulación de tarea'
    inicia_cronometro=[0 0];
    pre_trama_envio=[handles.tarea_recibida1(1) inicia_cronometro 1];
    construirtrama;
else
    inicia_cronometro=tiempo(fecha_tarea);
    pre_trama_envio=[handles.tarea_recibida1(1) inicia_cronometro 1];
    construirtrama;
end

guidata(hObject,handles);
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of checkbox1
```

Fuente: Elaboración propia

Se puede reemplazar las fechas de las tareas, haciendo uso del mismo número de tarea para la nueva orden de riego pero definitivamente habrá ocasiones en las que se tenga que anular alguna de las tareas. Para ello el usuario deberá escribir cero en todos los campos de texto, indicando a que pivote va direccionado y el número de tarea a anular.

Por ejemplo si se desea cancelar la tarea tres del pivote cinco, se escoge al pivote, la tarea correspondiente y se digita el número cero en mes, cero en día, cero en horas y por último cero en minutos.

El siguiente paso del algoritmo es preguntar si todos los valores del vector *fecha de tarea* son iguales a cero, en caso sea afirmativo se procede a mantener el *bit* del *reset* forzado, con lo cual se cancela cualquier conteo que esté realizando el PLC. Si fuese realmente una nueva tarea entonces se procede a calcular el tiempo y demás parámetros para luego sobrescribirlos en las direcciones de memoria que corresponda.

Para completar la comunicación, se debe disponer de un conjunto de espacios de memoria del PLC que estén asignados con anterioridad para recibir estos datos, con ello se vincula de forma directa la orden de Matlab con la subrutina de conteo del equipo. La tabla 18, muestra las direcciones de memoria reservadas para esta opción.

Tabla 18- Direcciones de memoria reservadas para las tareas de riego

DIRECCIÓN MODBUS (HEX)	TAREA Nº1	TAREA Nº2	TAREA Nº3	TAREA Nº4
… para almacenar el valor de horas	00 64	00 C8	01 2C	01 90
… para almacenar el valor de minutos	00 65	00 C9	01 2D	01 91
… que energiza el bit que inicia la subrutina	44 4B	44 AF	45 13	45 77
… que energiza el bit que reinicia la subrutina	44 4C	44 B0	45 14	45 78

Fuente: Elaboración propia

5.10.1 Tiempos de espera para inicio de tareas de riego

Para finalizar el último capítulo se dará una breve explicación del algoritmo realizado para determinar el tiempo que falta para una fecha aleatoria dentro del mismo año en que se declara la tarea.

Las consideraciones para este archivo son:

- El reloj de la PC de monitoreo debe estar a la hora.

- La comparación del año no se realiza dado que no hay un bloque de texto para introducir este dato, eso significa que hay una vulnerabilidad en la fecha de transacción del 31 de diciembre al 1 de enero donde varía el valor del año en curso.

- La distancia máxima entre la fecha actual y la fecha de la tarea no debe exceder más de 255 horas ya que en la trama de envío se ha considerado solo un *byte* para este número por ende no se programaran riegos a más de 10.6 días.

- La comparación de segundos tampoco se realiza, ello arrastra un retardo máximo de un minuto para la activación del riego.

- Si el año en curso en bisiesto, se deberá cambiar la cantidad de días del mes de febrero de 28 a 29, esto se hace entrando al archivo *tiempo* y modificando la variable *febrero=28* a *febrero=29.*

- Fechas con valores anteriores a la fecha actual sí entran al algoritmo de tiempo, sin embargo cuando se deba enviar la trama, el algoritmo que calcula el CRC dara una alerta de error debido al valor negativo que contiene el vector fila y no se enviará nada.

En la figura 52, se muestra el contenido del archivo denominado *tiempo* el cual necesita como dato de entrada el valor del mes, del día, la hora y minutos del evento y entrega como salida cuantas horas y minutos faltan para ese momento.

Primero se arma el calendario con los números correlativos de los días respetando la cantidad que tiene cada mes, luego se arma el vector de los meses repitiendo tantas veces el número del mes como días reunidos. Después se utiliza el comando *clock* para extraer desde Matlab la fecha actual, por eso la PC debe estar sincronizada con la zona horaria.

Del vector que entrega el comando *clock* se copian las posiciones de la dos a la cinco, no se toman en cuenta el año ni los segundos que son las posiciones uno y seis. Continuando con el algoritmo se compara el día extraído con la matriz *día*, dando lugar a 12 posiciones posibles. Para descartar se hace una comparación cruzada con la matriz de los meses, de esta forma se obtiene la posición ordinal de dicho día. Finalmente se restan las posiciones ordinales de estos dos días.

Aquí pueden darse tres situaciones, la primera y más sencilla es cuando la fecha dista más de un día, con lo cual las restas de las variables no tienen problemas. La segunda situación es cuando hay una tarea para el mismo día, el valor de la variable *días restantes* es cero y por tanto entra al primer condicional para determinar dicho conteo. A su vez aquí se puede presentar la tercera situación y es cuando la tarea de riego está dentro de la misma hora.

Ante todos los casos se ha preparado el algoritmo para poder determinar los valores que entregará al PLC y poder empezar con las subrutinas.

Figura 52- Bloque de sentencias de la función tiempo

```
function [inicia_cronometro]= tiempo(fecha_tarea)
m1=1:1:31; mes1=ones (1,31); febrero=28; m2=1:1: febrero;
mes2=2*ones (1, febrero); m3=m1; mes3=3*mes1; m4=1:1:30;
mes4=4*ones (1,30); m5=m1; mes5=5*mes1; m6=m4; mes6=6*ones (1,30);
m7=m1; mes7=7*mes1; m8=m1; mes8=8*mes1; m9=m4; mes9=9*ones (1,30);
m10=m1; mes10=10*mes1; m11=m4; mes11=11*ones (1,30); m12=m1;
mes12=12*mes1;

dias= [m1 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12];
meses= [mes1 mes2 mes3 mes4 mes5 mes6 mes7 mes8 mes9 mes10 mes11 mes12];
fecha_actual=clock;
mes_actual=       fecha_actual(2);
dia_actual=       fecha_actual(3);
hora_actual=      fecha_actual(4);
minuto_actual=    fecha_actual(5);
compara_mes =     (meses==mes_actual);
compara_dia =     (dias==dia_actual);
numero_actual=    (compara_mes==1) & (compara_dia==1);
[m,n]=            max(numero_actual(:));

fecha_actual=     fecha_tarea;
mes_actual=       fecha_actual(1);
dia_actual=       fecha_actual(2);
compara_mes =     (meses==mes_actual);
compara_dia =     (dias==dia_actual);
numero_actual=    (compara_mes==1) & (compara_dia==1);
[m,k]=            max(numero_actual(:));
dias_restantes=k-n;
if      dias_restantes ==0

        horas_restantes = fecha_tarea(3)-hora_actual-1;
        minutos_restantes=(60-minuto_actual)+fecha_tarea(4);
        inicia_cronometro=[horas_restantes minutos_restantes];
        if  horas_restantes==-1
            minutos_restantes=fecha_tarea(4)-minuto_actual;
            inicia_cronometro=[0 minutos_restantes];
        end
else
horas_restantes=(24-hora_actual-1)+fecha_tarea(3)+((dias_restantes-1)*24);
minutos_restantes=(60-minuto_actual)+fecha_tarea(4);
inicia_cronometro=[horas_restantes minutos_restantes];
end
return
```

Fuente: Elaboración propia

Conclusiones y Recomendaciones

En el Perú las vertientes hídricas no se encuentran distribuidas uniformemente ni por tiempo ni por requerimiento de las personas o actividades que demanden de este recurso, por ejemplo en la parte de la selva abunda este recurso durante casi todo el año en cambio en la región de la costa se dan etapas de sequías.

En este mismo sentido, la distribución de la población no es proporcional a la dotación hídrica de las cuencas, pues la vertiente del pacifico es la región con mayor crecimiento poblacional y demanda de agua, y es precisamente donde existe menor cantidad de este recurso.

No se puede permitir durante el tiempo de lluvias no almacenar el agua y durante sequías continuar arrojando esa cantidad de agua al mar. Por ello nuestro país necesita contar con más estructuras de captación que permitan almacenar y derivar el agua para los diversos usos benéficos, más aún para el sector agrícola, el cual usa gran porcentaje del agua nacional.

Para nuestras regiones el agua es un recurso vital de desarrollo, producción y calidad de vida, por ende no se puede privarlas de este recurso. El Estado es el principal responsable para un correcto planeamiento y gestión del agua en todas las cuencas.

Dicho esto, se explica al lector que en el trabajo realizado se ha desarrollado una aplicación capaz de integrar equipos de control aplicados al riego mecanizado, aunque la parte teórica de esta tesis es en base a la comunicaciones industriales, el fin último es ayudar en la gestión del riego y por ende optimizar el agua aplicada a los cultivos.

Las conclusiones puntuales al finalizar el proyecto de tesis son las siguientes:

- El objetivo general y los objetivos específicos de las tesis se lograron satisfactoriamente dado que la implementación de la red de supervisión es una realidad.

- El análisis del protocolo *Base Station 2* sirvió de forma primordial para la puesta en marcha la automatización del sistema de riego mecanizado.

- La integración de equipos como los paneles de riego de Valley Irrigation, variadores de potencia, sensores de humedad, flujómetros, válvulas y más instrumentación relacionada

con el proyecto Agrolmos en el cual se aplicó esta tesis ha dado muy buenos resultados.

- El tener una red de supervisión, con equipos de control bajo un mismo protocolo, como el propuesto en la tesis, facilita la conexión entre ellos y la flexibilidad de incorporarse a más procesos.

- El ahorro de agua es una característica del sistema de riego por aspersión, también lo es la uniformidad y eficiencia en la distribución de este recurso. Adicional a esto, se ha mejorado con la recopilación de datos ambientales, la confiabilidad con que se decide la profundidad de la lámina de riego que se aplicará al cultivo en un día concreto.

- Gracias a la metodología planteada en el segundo capítulo de la tesis, se realizó la migración del protocolo individual a uno estándar (Modbus RTU).

- Con el uso del PLC Click Koyo descrito en el capítulo 4, se hicieron las pruebas de comunicación de datos, también se analizaron los requerimientos y prestaciones que deberían tener los equipos que se iban a comprar, dentro de estos requerimiento estaba el tamaño de la información que debía ser capaz de intercambiar. Esto ayudó a prever la compra adecuada de los equipos de comunicación. De no haberse hechos los ensayos de comunicación con el protocolo Valley y demás sensores acoplados se hubiesen comprado los equipos inadecuados generando un gasto innecesario.

- El beneficio económico es atractivo para el proyecto, pero esto no es simplemente eligiendo otros equipos. El usar PLCs para la migración de protocolos es a costa de implementar un algoritmo más exigente tanto para la fase de programación como para la cantidad de tareas que debe ejecutar el PLC en cada escaneo. Si la cantidad de procesamiento aumenta se optará por un equipo de mayores prestaciones. El PLC instalado en el proyecto es el menor en su gama pero es suficiente y aborda todos los requerimientos.

- El quinto capítulo desarrolla en términos generales una conclusión muy importante la cual es que el manejo de una interfaz gráfica ayuda en la gestión de la automatización. Colabora en la toma de decisiones ya que aporta una buena cantidad de datos influyentes en la decisión final y eso mejora la parte productiva y operacional de las plantas.

Recomendaciones finales

Como recomendación general para todas aquellas personas interesadas en desarrollar una integración de protocolos de comunicación industrial, el primer paso del estudio es siempre evaluar las funcionalidades que tienen incorporadas los equipos, es decir si el instrumento que se quiere agregar es único en cuanto a mediciones o acciones.

Por ejemplo si brinda un parámetro muy especial, o reúne una fuerte cantidad de acciones de control ya implantadas en su electrónica, en ese caso sí valdrá la pena adherirlo a los procesos.

Otra recomendación, especialmente en la industria peruana es que nos encontraremos con equipos que ya llevan muchos años trabajando, antiguos pero que funcionan bien, sin embargo con las nuevas tecnologías nacen equipos que los suplantan a estos.

En estas situaciones el aspecto económico se vuelve muy importante ya que migrar a estos nuevos equipos conlleva a una serie de cambios en las instalaciones cuando evaluar una integración de protocolos podría ser más viable y atractiva en cuanto a inversión.

Como futuras líneas de investigación o mejoras del proyecto realizado es aumentarle la capacidad de realizar los análisis agronómico, las ecuaciones que contemplan las demandas de agua, datos meteorológicos, el perfil de viento, temperatura y necesidades de nutrientes tal que, formen parte de un algoritmo para el crecimiento adecuado de las plantas.

Referencias Bibliográficas

Autoridad Nacional del agua. (2010). *Boletín Técnico: Recursos Hídricos del Perú en cifras.* Lima: Ministerio de Agricultura.

Barreto Escobedo, C. D. (2015). *Investigación hidráulica utilizando un modelo numérico 3D de la presa Tablones Alto – Chinecas.* Piura: Universidad de Piura.

Base Station 2 SM. (s.f.). *Guía de instalación y tutorial del usuario (edición 7.2).*

Española, Real Academia. (s.f.). Diccionario de la lengua española. Versión electronica contenido de la 22.ª edición.

González P. (2007). *Introduccíon al riego y drenaje.* Cuba: Instituto de investigaciones del riego y drenaje.

Hernández, M. R. (2013). Comparación de diferentes métodos para el tratamiento de vinazas de la industria de etanol utilizando LCA. *Seminario Internacional sobre energías renovables*. Argentina.

Israelsen, O. (1981). *Irrigation Principles and Practices (4ª ed).* EE.UU.: Jhon Wiley & Sons Inc.

Click *help version 1.40 Tutorial del software Click Programming.*

Macías Macías, M. J., Vergara Sabando, M. M., Macías Solórzano, V. R., & Bazurto Zambrano, M. A. (2011). *Adapatacion e instalación de un sistema de riego por aspersión, para cultivos comerciales establecidos en la comunidad El Milagro del cantón Portoviejo.*

Oquelis Cabredo, J. E. (1997). *Metodología para la integración de sistemas heterogéneos de comunicaciones de campo en entornos industriales.* Las Palmas de Gran Canaria: Universidad de las Palmas de Gran Canaria E.T.S.I. Telecomunicaciones.

Ordinola Enríquez, J. (2009). *Simulación numérica tridimensional del comportamiento hidráulico del embalse limón - Proyecto Olmos.* Piura: Universidad de Piura.

Tarjuelo Martín-Benito, J. M. (1991). *El riego por aspersion: Diseño y Funcionamiento.* Albacete: Gráficas Colomer.

Tealdo Alberti. (1995). *Proyectos de irrigación en el Perú: Situación, análisis y políticas. Consejo Latinoamericano de ciencias sociales.*

Tejada Calderón, G. A. (2009). *Implementación de una red Modbus para aplicaciones de pesado dinámico.* Piura: Universidad de Piura.

The Math Work, I. (s.f.). *Matlab, edició de estudiante versión 4 Guía de usuario.*

Anexos