AUTOMATIZACIÓN Y MONITOREO DE UN SISTEMA DE BOMBEO, ALMACENAMIENTO Y RIEGO DE AGUA PLUVIAL

DANIEL FABIÁN LINARES CORTÉS

CÓD. 20112103743

MAYERLY TATIANA NARVÁEZ MARÍN

CÓD. 20112104763

UNIVERSIDAD SURCOLOMBIANA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA ELECTRÓNICA

NEIVA

2018

AUTOMATIZACIÓN Y MONITOREO DE UN SISTEMA DE BOMBEO, ALMACENAMIENTO Y RIEGO DE AGUA PLUVIAL

DANIEL FABIÁN LINARES CORTÉS

CÓD. 20112103743

MAYERLY TATIANA NARVÁEZ MARÍN

CÓD. 20112104763

Proyecto de grado para optar al título de:

Ingeniero Electrónico

Director:

AGUSTÍN SOLO OTÁLORA

Ingeniero Electrónico

UNIVERSIDAD SURCOLOMBIANA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA ELECTRÓNICA

NEIVA

2018

Nota de aceptación:

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

Vladimir Hoyos C.

_____

Firma del jurado

Neiva, 20 de Febrero de 2018

Este trabajo, símbolo de la culminación de una de las etapas y metas más importantes de mi vida está dedicado a mi madre, cuyo sacrificio y apoyo incansables lo han hecho posible. A mis familiares y amigos por brindarme siempre el cariño y la motivación para creer en mí.

—Mayerly T. Narváez M.

A mi hermosa madre, a quien de manera póstuma le dedico la culminación de la que quizás es la meta más sufrida pero importante hasta el momento en mi vida, a mi padre, pues sin su apoyo, sacrificio y determinación esto jamás hubiera sido posible, a mis hermanas que también fueron un gran soporte en este camino y a mis amigos, compañeros y profesores que aportaron en mi proceso de formación profesional. Mil gracias a todos ustedes.

—Daniel F. Linares C.

AGRADECIMIENTOS

Agradecemos especialmente al señor Alfredo Hernández por ser parte fundamental en el desarrollo de este proyecto mediante su apoyo, no solo económico sino en experiencia y consejo los cuales son mucho más valiosos en el camino del aprendizaje. Al profesor Agustín Soto Otálora y demás ingenieros docentes en el programa de ingeniería electrónica que impartieron el conocimiento necesario para llevar a cabo este trabajo de grado. A la Universidad Surcolombiana que durante poco más de 5 años se convirtió para nosotros en la fuente de saber científico, humano y cultural. Por último y no por ello menos importante a nuestros padres, familiares y amigos que mediante su apoyo incondicional lograron crear en nosotros la motivación para culminar esta etapa de la vida.

CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

LISTA DE ECUACIONES

Pág.

INTRODUCCIÓN

La automatización y monitoreo de un sistema de bombeo, almacenamiento y riego de agua pluvial consiste en la utilización de las herramientas tecnológicas pertinentes para lograr que los procesos involucrados en el sistema se realicen de manera automática, además de permitir la supervisión y vigilancia por parte del usuario a través del monitoreo de las variables del sistema. El sistema de control implementado se encarga de medir en tiempo real el nivel de los depósitos de agua ubicados en diferentes puntos remotos de la finca y allí se determinan según sus estados, la acción de control adecuada. Luego los datos son enviados para ser procesados y visualizados en la estación central. Dichas acciones de control permiten que el uso de agua sea racionado y eficiente pues no se producirá un gasto mayor al estrictamente necesario y el agua proveniente de la lluvia depositada en los puntos de recolección será la fuente principal de abastecimiento de los puntos de riego.

El sistema de riego automático y temporizado implementado utiliza el método de riego superficial por goteo lo que permite que las dosis de agua aplicadas al cultivo de tomate sean exactas logrando eficiencia en el uso del agua, ahorrando mano de obra para dicha labor y aumentando la producción del cultivo debido a la precisión al administrar el agua a la planta. Este proceso consiste en la aplicación de la cantidad de agua requerida por la planta de tomate 3 veces al día y así cumplir con sus procesos biológicos. Para esta tarea la estación de riego activa de manera autónoma las electroválvulas que permiten el paso del agua a los goteros según sea la hora del día indicada para esta tarea.

El diseño de la interfaz que grafica en tiempo real el comportamiento de las variables y dispositivos está desarrollada con el fin de presentar la información con facilidad de comprensión y uso, también que los objetos de interés sean de fácil identificación. Se ha construido a través de la comunicación serial con los dispositivos de control, los cuales proveen de toda la información.

La adquisición, control y supervisión de datos se logró mediante la selección adecuada de dispositivos hardware que permiten la implementación de unidades remotas en cada uno de los puntos de interés y una unidad central con su

correspondiente interfaz humano-máquina que recoge y presenta la información al operario, además de la comunicación adecuada y confiable entre ellas.

Debido a la reciente preocupación concerniente a la situación medioambiental mundial se ha aumentado el interés en la utilización no solo de sistemas tecnológicos que permitan un uso racional del agua, objetivo principal del proyecto, sino también el desarrollo de nuevas alternativas para el suministro de energía eléctrica por lo cual se ha optado por el uso de energía solar para alimentar los dispositivos de control (bajo consumo) y así incrementar el ahorro de recursos naturales.

Uno de los principales conceptos introductorios es el de cosecha de agua[1] que se define como la recolección o captación de la precipitación pluvial para usarla con fines productivos. El uso de agua lluvia con fines agrícolas se ha llevado a cabo desde hace miles de años, pero la tecnificación para el aprovechamiento del agua es lo que marca la diferencia y se ha perfeccionado a lo largo del tiempo. Uno de los métodos más utilizados es la captación y almacenamiento desde los tejados de vivienda y otras estructuras impermeables. México es uno de los países pioneros en esta técnica e invierte a nivel estatal para incentivarlo y aplicarlo. La utilización de dicha técnica conlleva a múltiples ventajas como ser funcional en lugares apartados y de difícil acceso a las tuberías públicas de acueducto, reducción de la demanda de agua, disminución del impacto ambiental negativo del uso indiscriminado de agua y finalmente a pesar de la no potabilidad, en medios agrícolas puede tener un uso adecuado. Existen varias dificultades que deben ser superadas invirtiendo en tecnología e investigación, entre ellas se tiene que el fenómeno pluvial no es constante sino por temporadas, su uso es limitado al no ser agua potable, para lograr sostenibilidad se necesita de grandes depósitos de agua y en países en vías de desarrollo no hay apoyo estatal para este tipo de alternativas que suelen ser costosas.

La implementación de la solución tecnológica propuesta para este caso presenta un alto costo económico para el agricultor lo cual se ve recompensado a largo plazo debido a que todas las técnicas utilizadas permiten no solo aumentar la producción de alimentos sino también el desarrollo de la región y lo más importante, se utilizaran los recursos naturales para beneficio de la agricultura disminuyendo el impacto ambiental nocivo.

[1]Secretaría de Agricultura, Ganadería, Desarrollo Rural, Pesca y Alimentación. México http://www.sagarpa.gob.mx/Delegaciones/coahuila/boletines/Paginas/2016B133.aspx

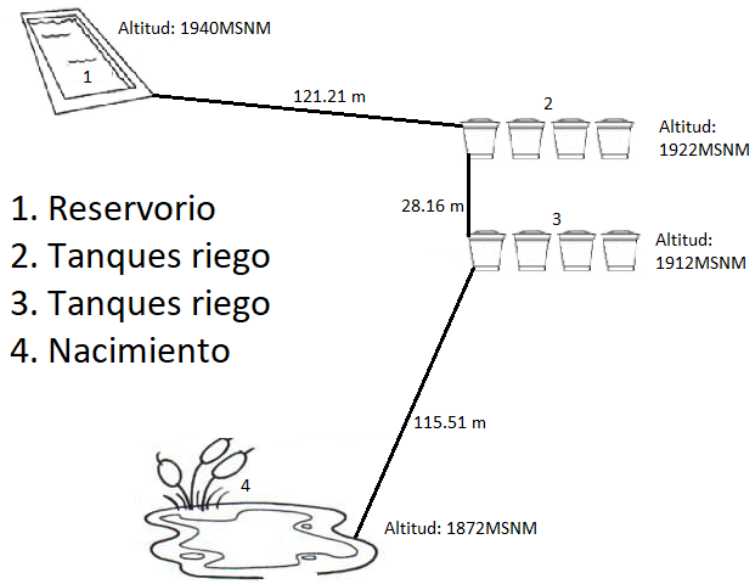
1. ESTUDIO PREVIO DEL SISTEMA

El sistema cuenta con particularidades que se tuvieron en cuenta a la hora de implementar la solución del problema. Partiendo de las consideraciones iniciales planteadas en el anteproyecto y teniendo las características físicas se procede a hacer un estudio que arroje los requerimientos que describen las necesidades del lugar.

1.1 CONSIDERACIONES INICIALES

Se tiene que el sistema hídrico consta básicamente de 4 puntos que se aprecian en la Figura 1 con sus respectivas altitudes en metros sobre el nivel del mar (MSNM) y distancias geográficas en metros.

Figura 1. Puntos iniciales de importancia hídrica en la granja



- Reservorio: Sitio de mayor almacenamiento hídrico, su principal función es proporcionar agua al punto Tanques riego N° 2 en temporadas de sequía, su capacidad aproximada es de treinta mil (30000) litros.
- Tanques riego N° 2: Lugar que almacena agua lluvia para regar un invernadero, tiene conexión directa con reservorio a través de una válvula manual, consta de cuatro tanques conectados por vasos comunicantes y su capacidad aproximada es de ocho mil (8000) litros.
- Tanques riego N° 3: De igual manera almacenan agua para riego de invernadero y tienen conexión con Tanques riego N° 2, los cuales por efecto de la gravedad entregan el excedente de agua pluvial, su capacidad aproximada es de seis mil (6000) litros.
- Nacimiento: Fuente de agua natural que por medio de electrobomba encendida manualmente entrega fluido a Tanques riego N° 3 en época de sequía.

Es importante resaltar que la numeración o nomenclatura utilizada en este apartado es la misma que se usó en el anteproyecto, por ende no es tenida en cuenta en otros apartados de este documento. Por otra parte, esta es la condición en que se encontró la granja y a partir de ahí se inició el proyecto.

1.2 REQUERIMIENTOS

A partir de los requerimientos del sistema, se plantearon los objetivos del proyecto, por tanto están directamente relacionados y se describen a continuación, con el fin de que el lector comprenda a plenitud lo que se busca o se desea.

1.2.1 RECOLECCIÓN AUTOMÁTICA

En búsqueda de la eficiencia hídrica se desea que haya un punto de recolección y bombeo de agua pluvial hacia el reservorio, de manera que la electrobomba se encienda automáticamente cuando se detecte el nivel máximo programado en los tanques y se apague cuando caiga a un nivel mínimo de protección de la electrobomba, ya sea porque cesa la lluvia o porque el caudal de entrada es menor al de salida, por lo cual naturalmente aunque haya lluvia el nivel estará descendiendo.

1.2.2 RETORNO AUTOMÁTICO

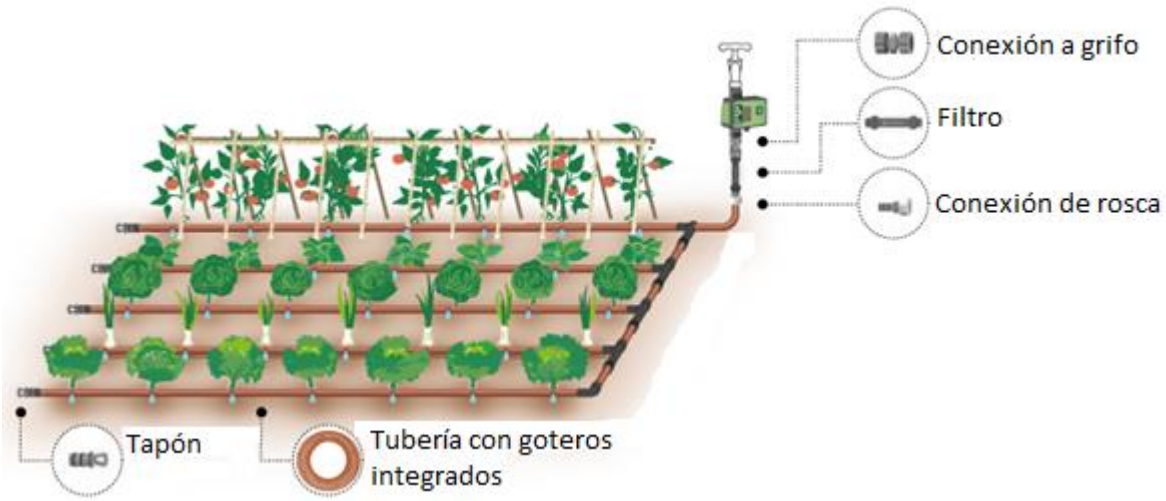
Se desea que los arreglos de tanques destinados a riego de invernaderos sean constantemente medidos para determinar el momento en que caen a un nivel demasiado bajo y se necesite abastecer del reservorio y el momento en que vuelvan a estar al nivel deseado, realizando la apertura y cierre automático de las respectivas electroválvulas.

1.2.3 RIEGO AUTOMÁTICO Y TEMPORIZADO

Se busca regar de la mejor manera posible, logrando la máxima eficiencia hídrica (menor cantidad de agua), sin olvidar que se debe mantener la humedad óptima del suelo para el cultivo, en este caso de tomate.

El propietario de la granja, ingeniero agrónomo, manifiesta que la cantidad ideal de agua diaria para cada planta de tomate es de 1 litro distribuido en 3 momentos del día, por ende considera que el mejor método es utilizar riego localizado, específicamente riego por goteo, que se basa en la instalación de un emisor (gotero) por planta como se aprecia en la Figura 2.

Figura 2. Componentes de un sistema de riego por goteo



Fuente: http://www.leroymerlin.es/ideas-y-consejos/comoHacerlo/riego-de-las-plantas-por-goteo.html

Debido a que la presión generada no será por bombeo sino por gravedad, se realizó la compra de goteros autocompensantes similares al de la Figura 3, que trabajan a un amplio rango de presiones, asegurando un caudal de salida constante e independiente de la presión.

Figura 3. Gotero autocompensante

Fuente: https://www.hunterindustries.com/es/product/riego-localizado/goteros-autocompensantes

El gotero utilizado, asegura un caudal de salida de 2 litros por hora, por lo cual se requiere la apertura temporizada de válvulas en 3 horas distintas del día por 10 minutos, logrando así el litro necesario por planta. Como se muestra en Figura 4, el arreglo de cuatro tanques consta de dos salidas, por lo tanto se deben controlar dos electroválvulas de riego temporizado.

Figura 4. Arreglo de tanques de riego y salidas

Finalmente, otro requerimiento importante es la disposición de un interruptor que permita la desactivación total del riego, pues no en toda la época del año hay cultivos dispuestos para regar.

1.2.4 ALIMENTACIÓN POR ENERGÍA SOLAR

Una de las principales preocupaciones es que el sistema de control se quede sin energía debido a una posible caída de la red eléctrica domiciliaria. De esta manera se hace necesario el uso de energías alternativas que además de aportar ambientalmente, contribuyen a la estabilidad del sistema, proporcionando energía constante y así evitando comportamientos indeseados del mismo. La energía solar, por razones de accesibilidad es la ideal para dar solución a este tipo de aplicaciones agrícolas.

1.2.5 MONITOREO DEL SISTEMA

Inicialmente el monitoreo no hacía parte de los requerimientos del sistema, pero al ser una capa importante de los niveles de automatización se hace evidente la necesidad de que el operario pueda supervisar el estado de los componentes y así asegurarse de que todo está en orden o de lo contrario tomar las decisiones pertinentes. Se desea que la interfaz sea capaz de lanzar algunas alertas en caso de funcionamiento inapropiado del sistema, tales como fallas en los sensores o carencia de agua en los puntos de abastecimiento.

2. DISEÑO DEL ALGORITMO

2.1 DEFINICIÓN GENERAL DE COMPONENTES

Una vez estudiado el sistema, sabiendo lo que se tiene y lo que se requiere, se procede a plantear la solución y el primer paso fue diseñar el algoritmo, para lo cual fue necesario la previa definición general de los componentes, es decir sin especificaciones exactas y su ubicación espacial dentro del sistema, tal como se observa en la Figura 5.

En el punto A solo se requiere sensar el nivel y transmitir el dato a los demás puntos que lo precisen, por ende se necesita un sensor de nivel, una unidad de procesamiento o control remota y un sistema de alimentación eléctrica.

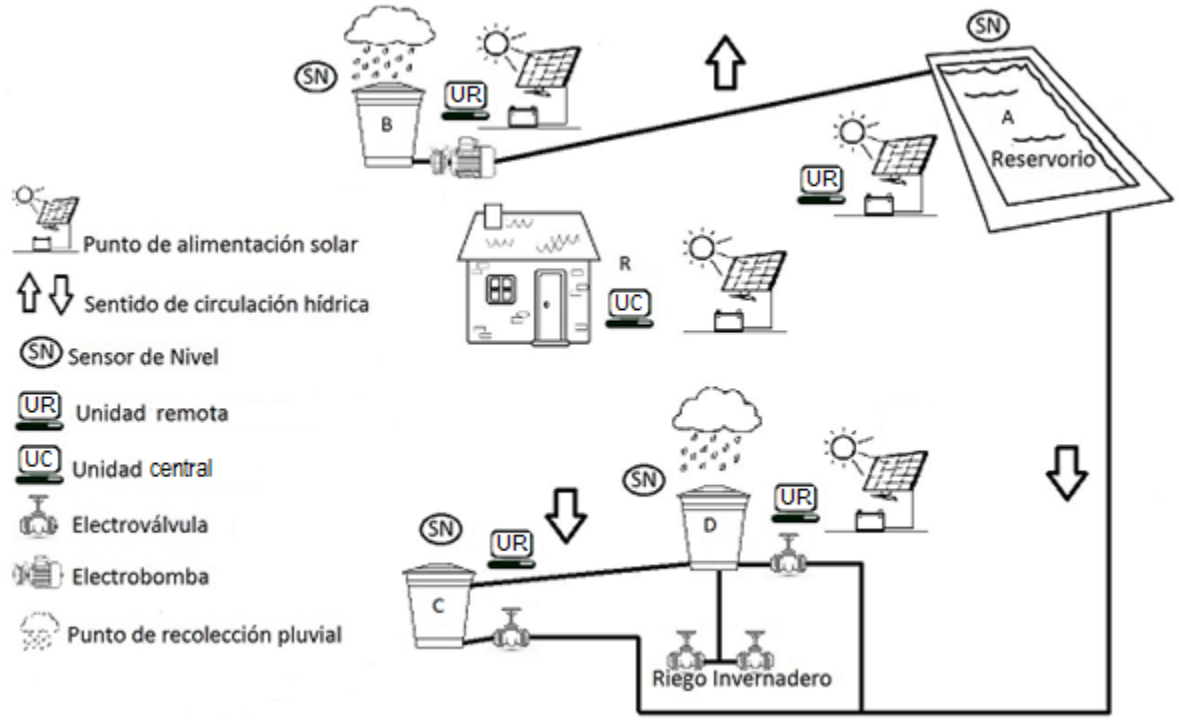
Para cumplir el requerimiento de recolección automática, se determinó la adición del punto B, encargado del acopio y bombeo, siendo así se necesita un sensor de nivel, una unidad de procesamiento o control remota, un sistema de alimentación eléctrica y una electrobomba de potencia adecuada para cumplir su tarea.

En el punto C se debe cumplir el requerimiento de retorno automático por lo tanto, también necesita un sensor de nivel, una unidad de procesamiento o control remota, un sistema de alimentación eléctrica y una electroválvula que permita realizar la acción de control.

El punto D es homólogo al punto C pero además debe cumplir el requerimiento de riego automático y temporizado, por ende lleva los mismos componentes más la adición de dos electroválvulas de riego.

Finalmente se determinó la adición el punto central en la vivienda principal del predio; el punto R debe recibir todos los datos del sistema para procesarlos y permitir la visualización gráfica de los mismos, logrando de esta manera que se cumpla el requerimiento de monitoreo. Por tanto necesita una unidad de procesamiento y visualización central y su respectivo sistema de alimentación eléctrica.

Figura 5. Componentes del sistema



2.2 DIAGRAMA DE FLUJO

Para la elaboración de cualquier código de programación estructurada es vital el conocimiento y comprensión total del flujo de datos que permiten el correcto funcionamiento de un sistema de información, en términos de diseño de algoritmos es menester la representación gráfica que sirva de base en la escritura de código en un lenguaje de programación.

El diagrama de flujo para este sistema particular se define como un solo gráfico en el que la información se va desplazando y las instrucciones de control de nivel y riego se van ejecutando en cada uno de los dispositivos involucrados en el proceso, siguiendo la misma nomenclatura de la Figura 5.

A partir de la Figura 6 se encuentra el diagrama dividido en partes y unido por conectores, con el fin de facilitar la visualización del mismo.

Figura 6. Parte 1 del diagrama de flujo

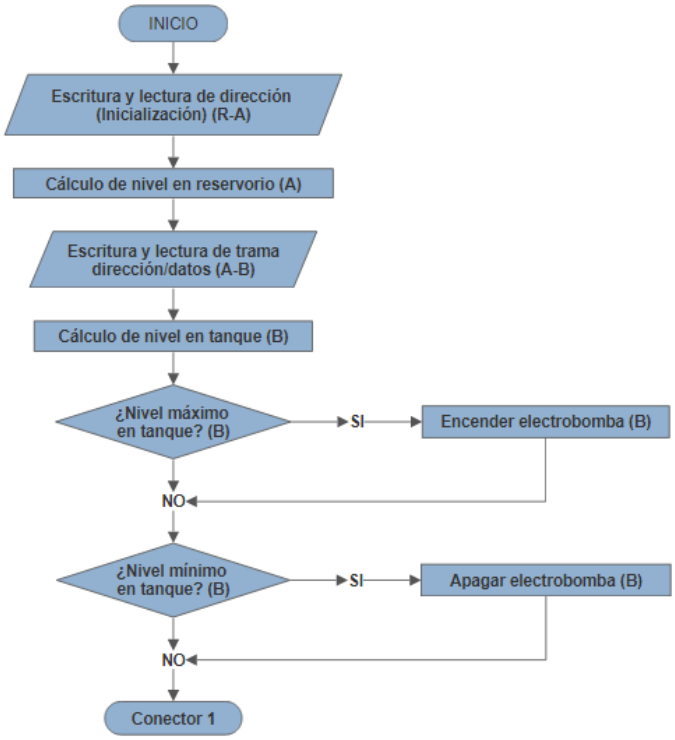
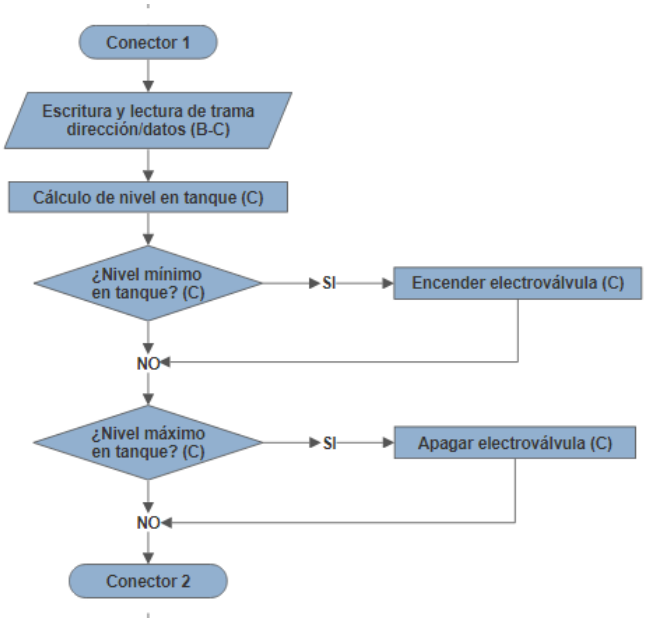


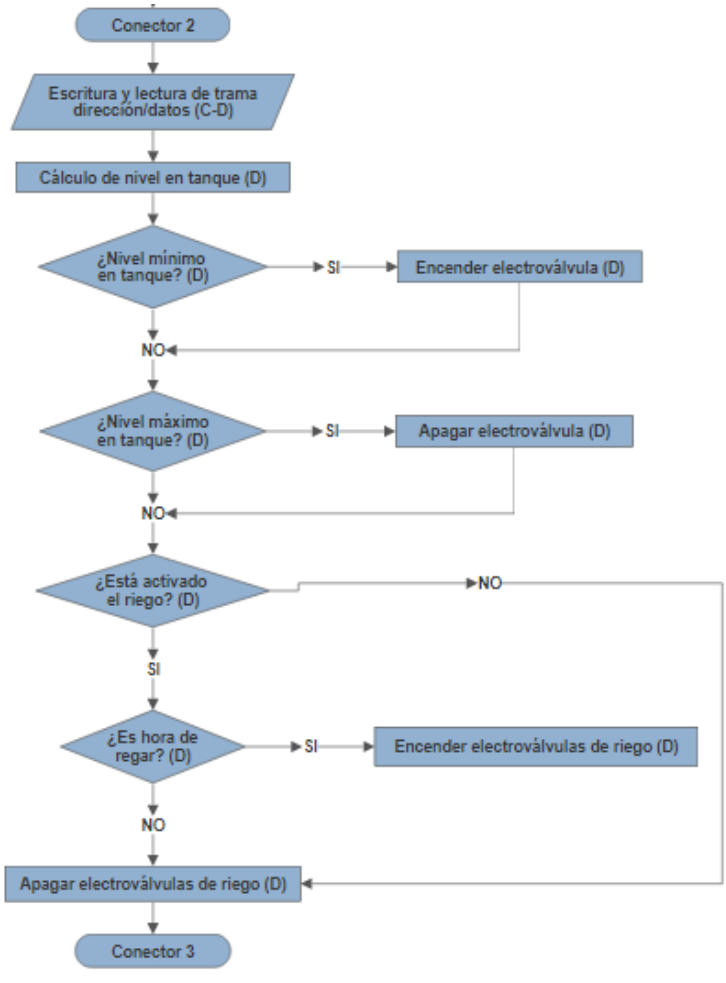
Figura 7. Parte 2 del diagrama de flujo



En la Figura 6 se puede ver, cómo se genera la dirección de inicialización y se transmite al primer punto o estación A, el cual fue elegido estratégicamente como primero para que el dato enviado no se distorsione en envíos innecesarios, debido al control de nivel en el tanque de bombeo (Estación B) que depende del nivel de reservorio. Por su parte en la Figura 7 se aprecia la escritura de trama conformada por la dirección, que es la letra que identifica la estación y los datos, que son los variables del punto, tales como nivel de tanque y estado de los actuadores. Es importante resaltar que todo el diagrama contiene en paréntesis las direcciones de las estaciones que están implicadas en cada paso del proceso.

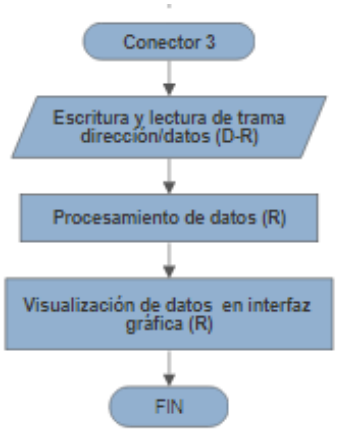
La Figura 8 contiene condiciones de control de nivel homólogas a las de las estaciones presentadas previamente pero además incluye las instrucciones de automatización de riego con la activación o apagado de las electroválvulas respectivas en la estación D.

Figura 8. Parte 3 del diagrama de flujo



Finalmente en la Figura 9 se puede apreciar la escritura final de trama en la estación central R, la cual se encarga de procesar todos los datos para permitir la visualización gráfica de los mismos.

Figura 9. Parte 4 del diagrama de flujo



3. SENSORES

Los sensores utilizados para detectar las magnitudes físicas involucradas en el sistema y transformarlas en variables eléctricas para el desarrollo de este proyecto, corresponden a los denominados sensores de nivel de líquidos. Estos tienen la función de detectar los cambios en la cantidad de líquido contenida en los tanques de almacenamiento hídrico de tal manera que pueda controlarse su llenado y vaciado. Para ésta tarea existen diferentes tipos: ultrasónicos, resistivos, ópticos, capacitivos, inductivos, interruptor de flotador, etc.

3.1 SENSOR ULTRASÓNICO

3.1.1 VENTAJAS

La selección del sensor ultrasónico como componente principal de medición, se da debido a que tiene múltiples ventajas que lo hacen ideal para la función a desempeñar en el proyecto:

- No es invasivo.
- Funciona para cualquier material, independiente del color o densidad.
- Fácil calibración.
- Fácil mantenimiento.
- Medida continua.
- Costo accesible a una calidad media.

Se debe evitar usar este tipo de sensores en líquidos que produzcan espuma en su superficie, estén sometidos a altas temperaturas o presiones o en condiciones de vacío, sin embargo en este caso no hay inconvenientes puesto que el líquido a tratar es agua a presiones y temperaturas ambiente.

3.1.2 SENSOR JSN-SR04T

Es un sensor ultrasónico, el principio de funcionamiento de éste, consiste en el uso de un material piezoeléctrico que permite generar una onda de ultrasonido, es decir por encima de los 20 KHz, a través de la excitación eléctrica del mismo. Esta onda emitida de manera cíclica y al propagarse en el espacio con obstáculos se verá reflejada, esta reflexión o eco es captada por el mismo sensor por medio de un receptor. Mediante un acondicionamiento de la señal se calcula el tiempo transcurrido entre la señal emitida y recibida, a través de su relación con la velocidad del sonido, es posible hallar la distancia a la que el objeto se encuentra, a este modo de operación se conoce como modo medidor de distancia. A continuación en la Tabla 1, las especificaciones técnicas del sensor:

Tabla 1. Especificaciones técnicas sensor ultrasónico JSN-SR04T

CARACTERÍSTICA	VALOR-RANGO
Voltaje de operación	5V
Consumo de corriente en operación	30 mA
Consumo de corriente en reposo	5 mA
Frecuencia de emisión acústica	40 KHz
Distancia mínima de detección	20 cm
Distancia máxima	450 cm
Resolución	5 mm
Temperatura de operación	-10°C ~ 70°C
Temperatura de almacenamiento	-20°C ~ 80°C
Precisión	3 mm

Fuente: **http://www.didacticaselectronicas.com/index.php/sensores/sensor-de-distancia-por-ultrasonido-jsn-sr04t-jsnsr04t-detail**

Como se puede apreciar el sensor dispone de muy buenas características que dan soporte a su uso en trabajos de ingeniería en la vida real, pues consta de excelente precisión, bajo consumo, buen rango de detección y robustez teniendo un buen rango de temperatura y resistencia a la humedad. Adicionalmente tiene un costo asequible pero muy superior al de sensores usados para prácticas de laboratorio y fines didácticos. La Figura 10 enseña la estructura física del sensor, compuesto de dos elementos, la cabeza con el emisor y transmisor y el circuito de acondicionamiento de las señales.

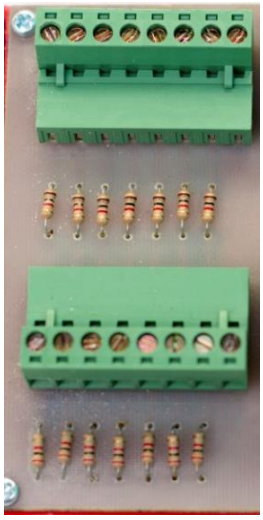
Figura 10. Sensor de ultrasonido JSN-SR04T

Fuente: https://www.amazon.com/Ultrasonic-Distance-Measuring-Transducer Waterproof/dp/B00XRRN3G6

3.1.3 ACONDICIONAMIENTO DE LA SEÑAL DE ULTRASONIDO

El acondicionamiento de la señal viene dado de dos maneras, en hardware y software. Con respecto a la parte física se tiene en cuenta que la salida proporcionada por el circuito del sensor es de 5V, el cual debe ser adaptado al rango del dispositivo que recibe y procesa la señal, por ejemplo los ARM poseen entradas de 3.3V por lo cual se debe proveer de una interfaz compuesta por un divisor de voltaje y bornera de conexionado, en la Figura 11 se puede observar.

Figura 11. Placa divisores de voltaje



A nivel de software se debe realizar el cálculo del tiempo transcurrido entre la emisión y recepción de las ondas ultrasonido, seguido de esto mediante la relación entre la velocidad del sonido y el tiempo se puede calcular la distancia entre el sensor y la superficie del agua de acuerdo a la Ecuación 1.

Ecuación 1. Distancia en función de velocidad del sonido y tiempo

$$d = \frac{1}{2} V_s t$$

Donde d es distancia en metros, V_s es la velocidad del sonido en metros/segundo y t el tiempo en segundos, transcurrido entre emisión de la señal y recepción del eco.

Debido a que el parámetro de la velocidad del sonido depende de las condiciones ambientales tales como temperatura se debe hacer la consideración de la Ecuación 2, que permite adaptar la velocidad del sonido:

Ecuación 2. Velocidad del sonido de acuerdo a temperatura ambiental

$$Vs = V_s 0° \sqrt{1 + \frac{T}{273}}$$

En donde $V_s 0°$ es la velocidad del sonido a 0°C y T la temperatura.

Para una temperatura de 20°C (temperatura promedio en el municipio de Oporapa) se tiene que V_s corresponde a 343.2 m/s por lo que en la Ecuación 3 se obtiene la distancia para dicha velocidad, en función de un tiempo variable:

Ecuación 3. Distancia en función del tiempo a 20°C

$$d(m) = \frac{1}{2} * 343.2 \left(\frac{m}{s}\right) * t(s) = 171.6 \left(\frac{m}{s}\right) * t(s)$$

Haciendo la medición del tiempo en microsegundos y esperando el resultado en centímetros, se obtiene la Ecuación 4 con su respectiva conversión:

Ecuación 4. Conversión de tiempo en microsegundos y distancia en centímetros

$$d(cm) = 171.6 * 100 \left(\frac{cm}{s}\right) t * 10^{-6}(s) = 0.01716t \approx \frac{t(\mu s)}{58}$$

La hoja de datos del sensor, solamente entrega la fórmula *d(cm)= t(µs)/58*, pero los cálculos realizados, permitieron demostrarla matemáticamente.

3.2 SENSOR INTERRUPTOR FLOTADOR

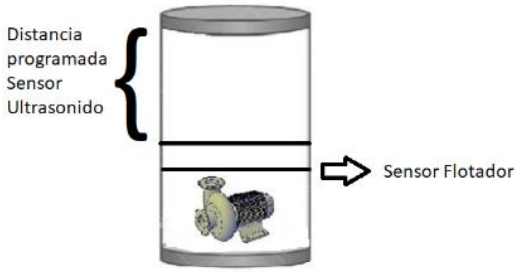
Este sensor es uno de los más simples mecanismos de detección de nivel en líquidos y por ende uno de los más económicos y de bajo consumo energético. Se basa en un contacto tipo *Reed Switch* y un flotador magnético que produce movimiento y cierra o abre el contacto; el sensor utilizado puede ser apreciado en la Figura 12. Sus desventajas radican en que es un sistema invasivo y requiere de un montaje dentro del tanque.

Figura 12. Sensor de nivel interruptor flotador

Fuente: https://www.amazon.es/Interruptor-Flotador-Sensor-Indicador-%C3%81ngulo/dp/B00Q6FNUXC

Se incluye este tipo de sensores como mecanismo de seguridad en el tanque que posee electrobomba, debido a que es un componente relativamente costoso y se le debe garantizar un nivel mínimo de agua para que no trabaje en seco y se produzca algún tipo de daño. La utilización de este sensor provee redundancia y seguridad al sistema, pues en caso de falla en el sensor principal de nivel ultrasónico, pues como se observa en la Figura 13 existe la segunda opción unos centímetros más abajo.

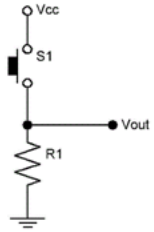
Figura 13. Sensor de seguridad para protección de electrobomba



3.2.1 ACONDICIONAMIENTO DEL SENSOR INTERRUPTOR

Debido al simple funcionamiento del sensor flotador en el que únicamente se comporta como un interruptor de acuerdo al nivel del agua, el acondicionamiento se basa en una resistencia de puesta en bajo, véase la Figura 14, con el fin de mantener el cero lógico y así evitar que ingrese al microcontrolador alguna señal indeseada que produzca lecturas inadecuadas al mantener la entrada flotante.

Figura 14. Resistencia de puesta en bajo o Pull Down



Fuente: https://tuelectronica.es/resistencia-pull-up-y-pull-down/

4. ACTUADORES

Los actuadores son dispositivos que convierten la energía eléctrica en energía mecánica, también denominados elementos finales de control, permitiendo ejercer la acción de control a las variables involucradas en el sistema. En este caso el control es ejercido sobre el nivel de los depósitos de agua por lo cual la acción consiste en abrir o cerrar el paso de suministro de agua y activar o desactivar las electrobombas y electroválvulas que permitan el vaciado o llenado de los mismos.

4.1 ELECTROVÁLVULAS

4.1.1 PRINCIPIO DE FUNCIONAMIENTO Y CARACTERISTICAS

En este caso es adecuada la utilización de una electroválvula solenoide doble vía, de acción directa, normalmente cerrada y sin presión para ser accionada en su totalidad. Por acción directa se refiere a que la excitación eléctrica de la bobina opera directamente sobre la apertura o cierre de la válvula a través de un embolo y al ser normalmente cerrada indica que su posición normal o inicial será cerrada por lo tanto el embolo estará dispuesto cerrando el orificio para no permitir el flujo del líquido sino hasta que sea energizada y cambie su estado a abierta. Las válvulas que en su funcionamiento no necesitan de altas presiones para poder dar paso al líquido son ideales para aplicaciones donde la presión se da por gravedad como está dispuesto en el sistema a tratar. La electroválvula seleccionada presenta las siguientes características técnicas de la Tabla 2 y se puede apreciar en la Figura 15.

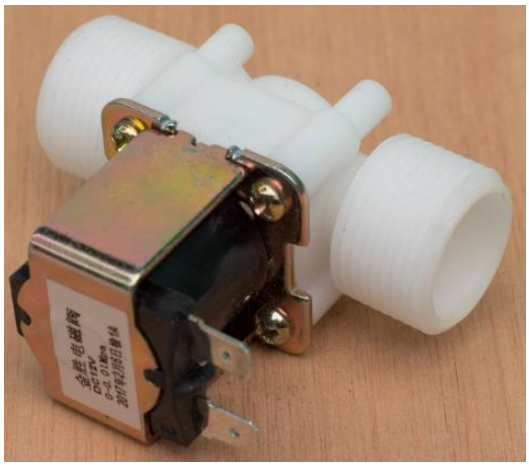
Tabla 2. Especificaciones técnicas de electroválvula seleccionada

CARÁCTERISTICA	VALOR
Medio de funcionamiento	Agua y aire

Voltaje de operación	12 VDC
Modo de operación	Acción directa
Tipo de accionamiento	Normalmente cerrada
Diámetro nominal	8 mm
Diámetro del puerto	3/4”
Presión de salida del agua	0.0Mpa~0.2Mpa
Temperatura de operación	-5°C~+60°C

Fuente: https://www.vistronica.com/valvulas/electrovalvula-de-plastico-12v-12-sin-presion-detail.html

Figura 15. Electroválvula solenoide seleccionada



4.2 ELECTROBOMBA

Las electrobombas son actuadores que permiten transformar la energía mecánica que provee un motor accionado eléctricamente en energía hidráulica al alterar la presión del flujo que pasa a través de sus conductos. El sistema posee una electrobomba cuya tarea consiste en bombear el agua lluvia almacenada en el punto de recolección hasta el punto de almacenamiento mayor (reservorio). Debido a que la distancia que debe recorrer el líquido desde el punto de

recolección hasta el punto de almacenamiento es grande, el rendimiento de la electrobomba debe garantizar esta tarea.

4.2.1 CÁLCULOS DE POTENCIA DE ELECTROBOMBA

La elección de una electrobomba adecuada depende de muchos factores y es menester tenerlos en cuenta para definir la especificación más importante, la potencia que garantiza que es competente para su función. Se realizó una serie de cálculos para definir la potencia del motor en la electrobomba del sistema.

Primero se debe calcular la carga dinámica (cd), que indica las pérdidas por fricción, para ello se utiliza la Ecuación 5 de Hazen-Williams (HW).

Ecuación 5. Ecuación de Hazen-Williams de carga dinámica

$$cd(m) = \frac{10.674 * L * Q^{1,852}}{C^{1,852} * D^{4.871}}$$

Dónde:

- L=Longitud de tubería (m)
- Q=Caudal (m^3/s)
- C=Coeficiente de HW-rugosidad (adimensional)
- D=Diámetro interno de la tubería (m)

El coeficiente de HW depende del material de la tubería, como se aprecia en la Tabla 3.

Tabla 3. Materiales y su rugosidad

Material	**C**	**Material**	**C**
Asbesto cemento	140	Hierro galvanizado	120
Latón	130-140	Vidrio	140
Ladrillo de	100	Plomo	130-140

saneamiento			
Hierro fundido, nuevo	130	Plástico (PE, PVC)	140-150
Hierro fundido, 10 años	107-113	Tubería lisa nueva	140
Hierro fundido, 20 años	89-100	Acero nuevo	140-150
Hierro fundido, 30 años	75-90	Acero	130
Hierro fundido, 40 años	64-83	Acero rolado	110
Concreto	120-140	Lata	130
Cobre	130-140	Madera	120
Hierro dúctil	120	Hormigón	120-140

Fuente: https://es.slideshare.net/karinagimenezabreu/presentacion-perdida-de-cargas-de-tuberias

La tubería en la granja está conformada por mangueras de alta presión, por tanto la constante seleccionada corresponde a plástico.

Los requerimientos del sistema son:

- L=80 m
- Q=75 l/min=0.00125 m^3/s
- C=140
- D=1”= 0.0254 m

Por lo tanto y de acuerdo a la Ecuación 5 la carga dinámica es de:

$$cd = \frac{10.674 * 80 * 0.00125^{1.852}}{140^{1.852} * 0.0254^{4.871}} = 22.4\, m$$

La Ecuación 6 corresponde a la altura dinámica (ad), que es la suma de carga dinámica y la altura estática diferencial que es de 10 metros (altitudes medidas por GPS):

Ecuación 6. Altura dinámica

$$ad = 10 + 22.4 = 32.4\, m$$

Teniendo este valor es posible por medio de la hallar la potencia teórica de la electrobomba:

Ecuación 7. Potencia teórica de electrobomba

$$Pb(Watts) = \rho * g * Q * ad$$

Dónde:

- ρ=Densidad del fluido(agua)=1000 kg/m^3
- g=Aceleración de gravedad=9.81 m/s^2
- Q=Caudal
- ad=Altura dinámica

$$Pb = 1000 * 9.81 * 0.00125 * 32.4 = 397.3\, W$$

La potencia real requerida para una electrobomba es la que tiene en cuenta que la eficiencia no es ideal, por lo tanto para un valor de 70% se tiene la Ecuación 8.

Ecuación 8. Potencia real de la electrobomba

$$Pbreal = \frac{Pb}{Ef} = \frac{397.3}{0.7} = 567.6\, W$$

Debido a que las bombas hidráulicas se venden en caballos de fuerza (Horse Power) se hace la conversión a esta unidad de potencia de acuerdo a Ecuación 9, teniendo en cuenta que 1 HP equivale a 745.7 W.

Ecuación 9. Potencia en caballos de fuerza o HP

$$Pbreal = \frac{567.6}{745.7} = 0.76\, HP$$

El valor de potencia comercial por encima y que proporciona un rango amplio de seguridad es de 1 HP.

4.2.2 ELECTROBOMBA SELECCIONADA: EVANS 3HME100

Es una electrobomba con una potencia de 1 HP destinada para el bombeo de agua limpia y con pequeñas cantidades de impurezas. Presenta beneficios como el ahorro de energía, protección térmica y durabilidad debido a que está construida en hierro fundido, se puede observar en la Figura 16. Sus características técnicas se exponen en la Tabla 4.

Figura 16. Electrobomba EVANS 3HME100

Fuente: http://www.ieh.mx/?product=3hme100

Tabla 4. Especificaciones técnicas de electrobomba Evans 3HME100

Características	Valor
Voltaje	127 / 220 V Monofásico
Corriente	6 / 12,5 A
Caudal Máximo	115LPM
Potencia del Motor	1 HP
Altura Optima	24 m
Frecuencia de rotación	3450 RPM

Fuente: http://bomhsa.com.mx/bombas/centrifuga-evans-3hme100/

5. DISPOSITIVOS DE CONTROL

5.1 SELECCIÓN DE DISPOSITIVOS

Existen múltiples opciones en el mercado y cada una de ellas presenta ventajas y desventajas para las condiciones del sistema particular a automatizar.

La selección de los dispositivos de control es un paso sumamente vital e importante ya que van a ser la parte principal de las unidades de procesamiento y control remotas. Deben ser dispositivos de características apropiadas y que se adapten a su función dentro del sistema.

5.1.1 TIPOS DE CONTROLADOR

Para la correcta elección del dispositivo controlador se debe hacer un comparativo de ciertos parámetros en cada una de las opciones disponibles que son: computadores de placa simple (SBC), tarjetas de desarrollo, microcontroladores (MCU), y controladores lógicos programables (PLC).

- SBC: Son computadoras completas y funcionales diseñadas y ensambladas en un solo circuito impreso. Cuentan con un procesador, puertos para conexión de periféricos y puerto GPIO (entradas y salidas de propósito general) para establecer comunicación con el mundo real a través de componentes externos como sensores. La más conocida y utilizada actualmente es la placa de origen británico Raspbberry Pi; la versión más reciente es la Raspberry Pi 3 Modelo B.
- MCU: Son circuitos integrados programables capaces de ejecutar las órdenes almacenadas en la memoria del dispositivo. Cuenta con las unidades básicas de una computadora: CPU, memoria y periféricos de entrada y salida. Dos fabricantes de mucha trayectoria y confiabilidad son Microchip con sus microcontroladores más avanzados de la familia PIC32 y NXP que actualmente es la empresa legado de grandes compañías como Motorola y Phillips.

- Tarjetas de desarrollo: Son circuitos impresos que integran la funcionalidad de un microcontrolador con puertos de entradas y salidas digitales y analógicas. Cuentan con un entorno de desarrollo integrado que permite programar la placa en determinado lenguaje de programación. La tarjeta de desarrollo más común y expandida es la placa Arduino en sus múltiples versiones.
- PLC: Son computadoras utilizadas de manera casi unánime en la ingeniería de control industrial, se usan para la automatización de procesos en los que interfiere maquinaria electromecánica. Están diseñados para trabajos de alta exigencia como temperaturas extremas, alto ruido eléctrico, vibración elevada y respuesta en tiempo real. Sus costos son elevados aunque existen PLC de tamaño y precio reducido para automatizar procesos donde la demanda de entradas y salidas no es tan alta. Panasonic es una compañía de calidad y trayectoria que ofrece este tipo de PLC como los de la línea AFPX.

5.1.2 TRANSICIÓN DE MICROCONTROLADOR A TARJETA DE DESARROLLO

Las tarjetas de desarrollo incorporan múltiples MCU de excelente calidad y bajo costo, mientras que el proceso de montaje y fabricación de una placa de circuito impreso que utilice dichos circuitos integrados es mucho más costoso si se hace con los estándares necesarios de impresión de pistas y soldadura, o de igual costo y menor calidad si se omiten y se hace de forma casera o artesanal, por lo tanto se decide omitir los MCU en su forma básica y comparar tarjetas de desarrollo que los utilicen, pues ofrecen mayor economía y calidad. De esta manera para la comparación y selección se utilizaron la Chipkit Max32 con MCU PIC32 y la FRDM-K64F con MCU Kinetis K64.

5.1.3 PARÁMETROS DE SELECCIÓN

El número de entradas y salidas (E/S) es un factor importante a tener en cuenta, en un inicio se optó por la implementación de un solo controlador conectado a todos los sensores y actuadores del sistema, pero debido a inconvenientes que se detallarán posteriormente, se opta por utilizar un controlador en cada punto de interés, siendo ésta la solución más adecuada al problema por múltiples razones.

Siendo así, el número de componentes en cada punto es reducido y de esta manera también el número de E/S. La escalabilidad de un sistema es la capacidad de adaptarse al crecimiento o a la expansión sin perder calidad, por ende se debe dejar un margen de entradas y salidas libres disponibles para fines de expansión. En la Tabla 5 se realiza la comparación de E/S entre las opciones disponibles.

Tabla 5. Comparación de número de entradas y salidas en distintos dispositivos

DISPOSITIVO	NÚMERO DE ENTRADAS Y SALIDAS DIGITALES	NÚMERO DE ENTRADAS ANALÓGICAS
Arduino Nano	22	8
Raspberry Pi 3	26	0
Chipkit Max32	82	16
FRDM-K64F	40	2
Panasonic AFPX-C14R	14	0

En la Tabla 6 se hace una comparación de las características hardware de cada dispositivo, que son las que permiten determinar la capacidad y velocidad de procesamiento de datos, siendo el rendimiento junto al costo reducido los parámetros más deseables del sistema.

Tabla 6. Especificaciones hardware en distintos dispositivos

DISPOSITIVO	RAM	PROCESADOR-MICROCONTROLADOR	VELOCIDAD DE RELOJ	TIEMPO REAL	RTC	PRECIO USD
Arduino Nano	2 KB	Atmega 328 8 bits	16 MHz	SI	NO	5
Raspberry Pi 3	1 GB	ARMv8 quad-core 64 bits	1.2 GHz	NO	NO	35

Chipkit Max32	128 KB	PIC32MX795F512L MIPS-32bits	80 MHz	SI	SI	50
FRDM-K64F	256 KB	Kinetis ARM-32bits MK64FN1M0VLL12	120 MHz	SI	SI	35
Panasonic AFPX-C14R	16 Ksteps	RISC 32 bits	0.32 us/step	SI	NO	140

Un parámetro importante es la seguridad y confiabilidad que proporciona el dispositivo de control y que permite un funcionamiento adecuado del mismo, este parámetro puede llegar a ser un poco subjetivo pues depende de las condiciones ambientales particulares de operación y del nivel de seguridad requerido. En este aspecto simplemente se hablará de forma global de otros dispositivos y PLC, se comparan en la Tabla 7.

Tabla 7. Parámetros de seguridad en PLC y otros dispositivos de control

DISPOSITIVO	AISLAMIENTO DE LAS SALIDAS	PROTECCIÓN A CONDICIONES AMBIENTALES (ROBUSTEZ)	DETECCIÓN DE FALLAS
Otros dispositivos	NO	NO	NO
PLC	SI (Transistores-relés)	SI	SI (En PLC's de alta gama)

La particularidad del sistema exige que los dispositivos a utilizar sean de consumo y tamaño reducido (debido a que serán instalados y alimentados en las estaciones solares), parámetros que se comparan en la Tabla 8.

Tabla 8. Tamaño del PCB y consumo eléctrico en distintos dispositivos

DISPOSITIVO	TAMAÑO (cm)	CONSUMO (mA)

Arduino Nano	1.8x4.5	19
Raspberry Pi 3	5.6x8.5	800
Chipkit Max32	5.3x10.2	90
FRDM-K64F	5.3x8.1	115
Panasonic FPX-C14R	9x6x10.1	185

5.1.4 ANÁLISIS Y SELECCIÓN FINAL DE DISPOSITIVOS

Al hacer el análisis de los parámetros de comparación se llega a que la mejor solución es usar Arduino Nano, pues cuenta con más que suficientes entradas y salidas; en cuanto a rendimiento no es necesario que posea características extraordinarias pues debido a que hay un dispositivo terminal remoto en cada punto las líneas de código se reducen considerablemente, como se puede apreciar en el diseño del algoritmo; lo importante es entender que es un dispositivo con respuesta en tiempo real dedicado en su totalidad a la tarea que se le programa, a diferencia de lo que puede hacer una placa con sistema operativo encargada de múltiples tareas y con diferente prioridad. Por otro lado y sin menos importancia presenta las mejores características en cuanto a costo, tamaño y consumo de corriente reducido.

Al estudiar la estación donde además de control de nivel se hace la automatización de riego temporizado se evidencia la necesidad de un dispositivo más complejo debido a que se debe tener la base de tiempo que permite regar a horas exactas y la cantidad de instrucciones en el algoritmo se aumenta, por ende se decidió en este único punto utilizar un dispositivo de mayor capacidad o rendimiento y que además incorpora un reloj en tiempo real (RTC). Se seleccionó la FRDM K64F de NXP.

Aunque los PLC proporcionan un mayor nivel de seguridad, se trabajó para suplir algunas falencias y ofrecer mayor seguridad; se utilizaron relés y optoacopladores con alimentación propia para aislar el controlador de salidas que utilizan corriente alterna o tensiones diferentes a los 3.3 V o 5 V TTL. En cuanto a condiciones ambientales es un parámetro que no aplica pues no será instalado en lugares con extremos de temperatura, alto ruido eléctrico o vibraciones excesivas. Finalmente

se utilizaron los componentes hardware y software necesarios para detectar las principales fallas del sistema y dar aviso por medio de la interfaz de monitoreo, dichas alertas se detallan en el apartado 9.

5.2 FRDM K64F

La FRDM K64F es una tarjeta de desarrollo de ultra bajo costo fabricada por NXP usando MCU Kinetis. Esta ha sido diseñada en colaboración con la plataforma de desarrollo para arquitecturas ARM Mbed, por lo cual cuenta con una gran cantidad de periféricos que le permiten un prototipado rápido. Gracias al uso del microcontrolador Kinetis K64F como núcleo ARM Cortex-M4 posee una gran potencia de procesamiento con velocidad de hasta 120MHz, 1024KB de memoria Flash y 256KB de RAM. Se puede observar en la Figura 17.

Figura 17. Tarjeta NXP FRDM K64F

Fuente: https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/kinetis-cortex-m-mcus/k-seriesperformancem4/k2x-usb/freedom-development-platform-for-kinetis-k64-k63-and-k24-mcus:FRDM-K64F

5.3 ARDUINO NANO

La Arduino Nano es una pequeña y compacta pero completa placa de desarrollo con microcontrolador ATmega 328. Cuenta con puertos de comunicación I2C, SPI y UART, salidas PWM y 8 entradas analógicas.

En cuanto a capacidad de procesamiento presenta un rendimiento menor al de la K64F, con un microcontrolador de 8 bits y velocidad de reloj de 16 MHz, pero es importante recalcar que este dispositivo cumple la función de automatizar con básicamente una única instrucción de control y junto al chip MAX485 (véase el apartado 6.1) cumple su segunda función de ser un puente de comunicación dentro de la topología de red establecida para la comunicación de datos. Se puede apreciar en la Figura 18.

Figura 18. Tarjeta Arduino Nano

Fuente: http://rees52.com/227-arduino-nano.html

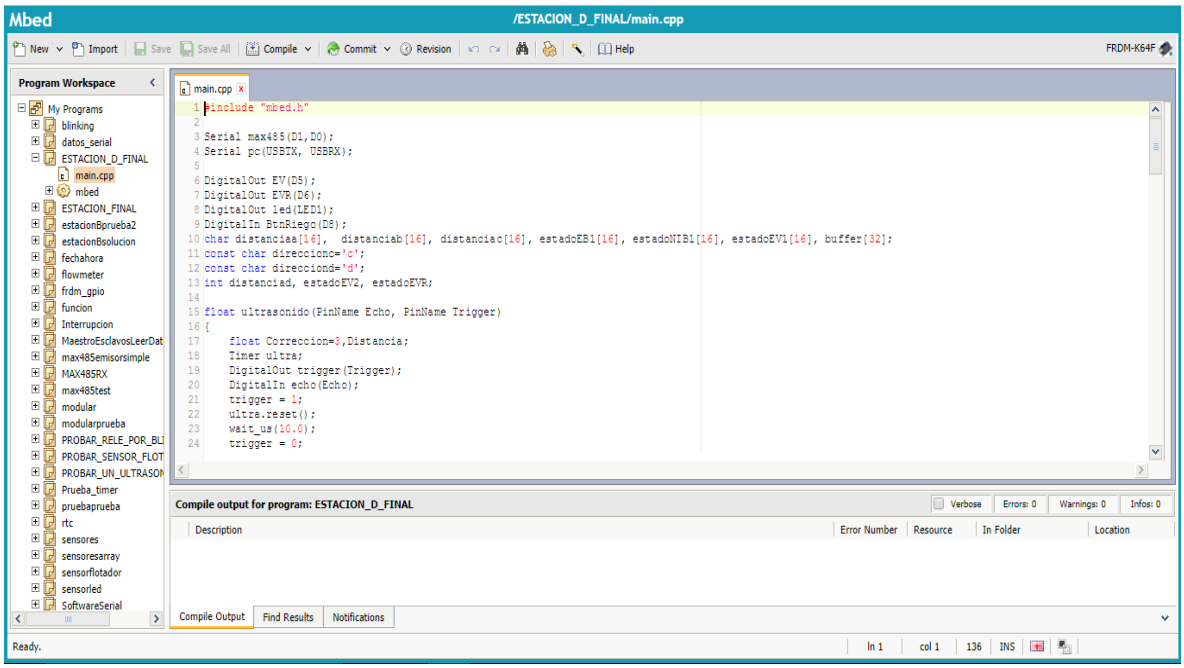
5.4 SOFTWARE DE DESARROLLO

La tarjeta FDRM K64F posee diferentes opciones en cuanto al software de desarrollo, desde el kit proporcionado por Kinetis, el fabricante del microcontrolador, hasta diferentes opciones de compiladores online. Los más usados son Mbed el cual tiene HDK y SDK habilitados y el SDK de MCUXpresso. En cuanto a Arduino se utilizó el popular IDE proporcionado por la compañía.

5.4.1 MBED

Para la programación de la K64F se utilizó mbed, una plataforma y sistema operativo en tiempo real (RTOS) desarrollado para los microcontroladores ARM Córtex para el internet de las cosas (IoT). Es una herramienta de código abierto que permite la realización de pruebas y prototipos con los microcontroladores ARM Cortex-M3 y ARM Cortex-M0 de 32 bits. Posee un entorno de desarrollo integrado (IDE) en la nube de acceso inmediato al desarrollo de microcontroladores en C/C++. Tiene a disposición librerías, almacenamiento de proyectos y un compilador online de manera que es posible iniciar sesión desde cualquier lugar y continuar en el mismo punto donde se había dejado el proyecto, además es compatible libremente con Windows, Mac, iOS, Android y Linux. El compilador produce un código eficiente ya que utiliza el motor ARMCC profesional. En la Figura 19 se aprecia la interfaz de trabajo del compilador para la cuenta creada y a la izquierda la lista códigos realizados.

Figura 19. Interfaz de trabajo del compilador MBED



6. COMUNICACIÓN REMOTA

Uno de los mayores inconvenientes al transmitir datos es la pérdida de las señales en líneas de gran longitud; en primer lugar el cable empieza a comportarse como una antena que recibe ruido e interferencia electromagnética, por supuesto indeseables, como segundo se tiene el problema de atenuación, directamente proporcional a la longitud del tendido. Un error común es ignorar estos hechos y creer que se puede transmitir por medios metálicos a cualquier distancia, sin usar ningún tipo de estándar físico o protocolo de comunicaciones. Teniendo en cuenta que el proyecto cuenta con tramos de cable de entre 40 y 150 metros aproximadamente, se hizo necesario buscar una estrategia que permita enviar la información de manera óptima. Existen soluciones inalámbricas tales como ZigBee o transceptores de radiofrecuencia, pero los principales problema de la transmisión inalámbrica suelen ser el alto costo de los módulos y la necesidad de una gran potencia de transmisión para lograr la distancia deseada. Finalmente consultando, se llega a un estándar que se acopla perfectamente a las necesidades del proyecto, el RS-485. Los problemas de comunicación y la posterior elección del estándar RS-485 son la principal razón que lleva a la implementación de un dispositivo controlador por cada estación o punto clave.

6.1 RS-485

Es una de las especificaciones de la norma RS449, publicada como estándar en 1983. Basa su funcionamiento en una señal diferencial, por lo que es muy útil en redes industriales a dos hilos. Actualmente es conocida como la norma TIA-485 ya que es administrado por la Asociación de Telecomunicaciones Industriales (TIA).

En cuanto a características mecánicas, el estándar define que se debe realizar con un par trenzado lo cual aumenta la resistencia a interferencias y supera la velocidad de transmisión de la norma RS-232. Permite la conexión en bus de hasta 32 terminales de transmisión full dúplex para enlazar procesadores de comunicación principal (maestros) con procesadores subordinados (esclavos) y las mismas topologías que en redes de datos comunes.

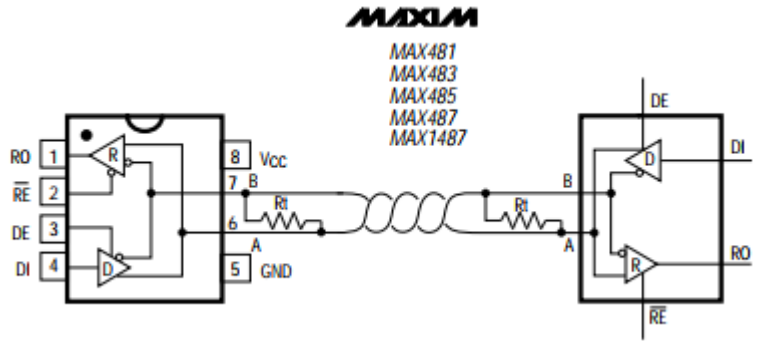
En las características eléctricas se tiene que el emisor opera el 1 lógico a un voltaje de -1.5 a -5 Voltios y el 0 lógico a la entrada del receptor en el rango de 0.2 a 12 Voltios y la máxima tensión aplicada a la línea de salida es de -7 a 12 Volitos.

El alcance de transmisión está dado según la cantidad de datos a transmitir en el tiempo, teniendo la longitud máxima de 1200 metros a 100 Kbps y mínima de 12 metros a 10 Mbps.

6.1.1 MAX485

Es un transceptor de baja potencia para la comunicación RS-485 entre dispositivos que manejan tensiones TTL. Su consumo de corriente está en el rango de 120 µA a 500 µA. Cuenta con pines de bus A y B, pines de conexión a receptor-transmisor y dos pines de control que mediante su estado lógico le indican al integrado que debe trabajar como emisor o receptor para comunicación half dúplex. Si se desea un mayor nivel de seguridad y evitar posibles colisiones de datos, es posible la comunicación full dúplex, utilizando dos CIs en cada punto. La Figura 20 indica la conexión típica con par trenzado y resistencias de final de línea que tienen como función evitar efectos indeseados debido a las altas frecuencias y distancias implicadas, idealmente deben ser del mismo valor de la impedancia característica del par trenzado es decir de 120Ω y así evitar reflexiones indeseadas que corrompan los datos.

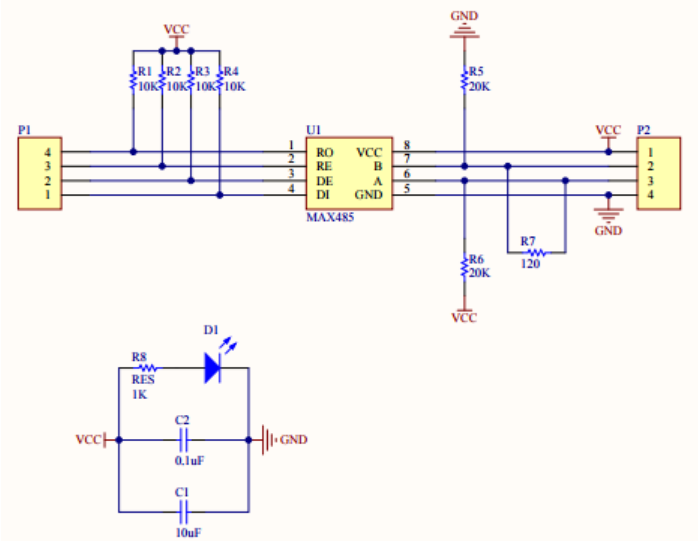
Figura 20. Conexionado típico del transceptor MAX485



Fuente: Hoja de datos del CI (Anexada)

Existe un módulo disponible en el mercado que utiliza el MAX485 con un conjunto de elementos que permiten un funcionamiento óptimo del circuito integrado (CI), tal como se ve en la Figura 21 el circuito cuenta con resistencias a prueba de fallos que garantizan la puesta en alto de los pines RO RE DE y DI. Al igual que resistencias a VCC y GND en el bus AB, resistencia Rt del estándar, capacitores de desacople para filtrar ruido de la fuente de alimentación y un diodo led indicador de alimentación del circuito.

Figura 21. Esquemático del módulo conversor RS485-TTL Serial

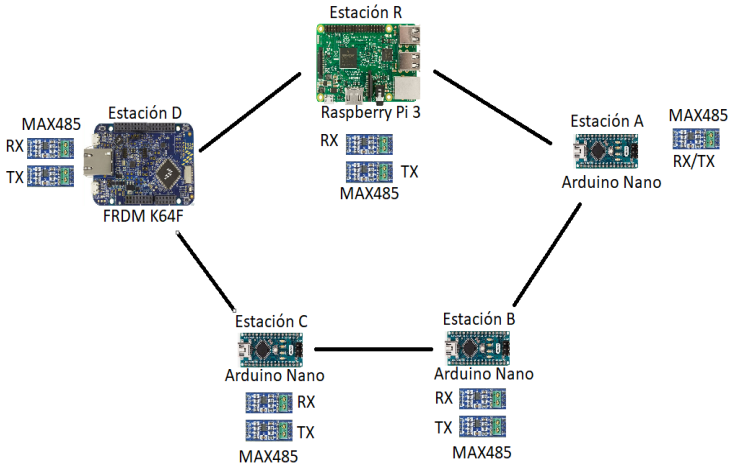


Fuente: http://www.vistronica.com/images/Documentos/Esquematico_MAX485.pdf

6.2 TOPOLOGÍA DE RED

La topología de red utilizada para la ejecución total del algoritmo es la topología de anillo, pues la dirección de inicialización se genera por parte del usuario al ejecutar la interfaz gráfica y la Raspberry Pi 3 lo transmite por medio del transceptor MAX485 configurado como transmisor, al receptor del punto A y tal como se evidencia en el diagrama de flujo, la trama se va transmitiendo a los demás puntos hasta que retorna al dispositivo central, como se aprecia en la Figura 22.

Figura 22. Topología de red de comunicaciones



Normalmente el estándar utiliza la topología de bus y cada dispositivo se conecta a la misma línea, pero en este caso más que un anillo, es una serie de tramos de conexión punto a punto por medio de un MAX485 configurado como transmisor y uno configurado como receptor (a excepción del punto A que funciona en modo half dúplex). De esta manera se logran varios aportes positivos a la red:

- Se evita colisión de datos ya que cada tramo es un bus independiente.
- Se cumple en su totalidad el estándar RS485 ya que cada módulo cuenta con una resistencia de 120 ohmios exigida por el estándar al inicio y final de línea. Si se utilizan en un único bus las resistencias se sumarán en paralelo y su valor se reduce, lo que puede acarrear problemas graves en la comunicación (En caso de hacerlo habría que modificar la PCB, retirando los resistores que sean necesarios).
- Aunque el estándar asegura 1.2 km a la tasa más baja, al hacerlo en líneas independientes se reduce a la distancia de cada tramo, por ende se puede transmitir a mayor tasa de bits.

7. SISTEMA DE ALIMENTACIÓN FOTOVOLTAICO

Las instalaciones solares fotovoltaicas implementaron para alimentar los dispositivos de bajo consumo del sistema para lograr un ahorro de energía eléctrica y la posibilidad de que el sistema permanezca energizado las 24 horas sin inconvenientes. Debido a que el sistema está distribuido en diferentes puntos con distintas tareas, se han dispuesto por cada punto una estación solar que provee de la energía necesaria para el funcionamiento de los diferentes dispositivos que permiten la realización de las tareas.

7.1 COMPONENTES DEL SISTEMA

7.1.1 MÓDULO FOTOVOLTAICO

El modulo fotovoltaico es elemento principal de los sistemas de energía fotovoltaica debido a que cumple con la tarea de convertir la radiación del sol en energía eléctrica gracias al efecto fotovoltaico. Dicho efecto consiste en la utilización de materiales (semiconductores) los cuales al ser excitados por la radiación del sol crean una fuerza electromotriz y generan un paso de corriente.

La célula solar es el generador de la instalación de energía solar. La célula se comporta como un diodo, de tal manera que la parte expuesta al sol es la región N y la parte que permanece en la oscuridad es la región p. Estas dos regiones crean un campo eléctrico permitiendo se cree una corriente eléctrica en dirección a este campo, a través de la circulación de electrones sobre el semiconductor cuando la radiación solar permite el rompimiento del enlace. La zona p se encuentra completamente metalizada y la región expuesta al sol, la región n posee un metalizado en forma de peine, estos contactos permiten obtener la tensión producida por la radiación solar recibida.

Un panel solar o módulo fotovoltaico es un arreglo formado por la unión eléctrica de varias células fotovoltaicas. La salida que dicho modulo proporciona se diseña para valores concretos de tensión continua que varían entre los 6V a 24V.

El fabricante proporciona las siguientes características eléctricas del panel, además de las características físicas que comprenden medidas, peso, materiales, etc. para permitir su correcta adecuación y soporte.

- Potencia máxima: Es el parámetro más importante debido a que representa la máxima potencia que puede entregar el panel en un determinado momento.
- Tensión máxima: Es la tensión que proporciona el panel cuando está produciendo su potencia máxima.
- Corriente máxima: Es la corriente máxima que proporciona el panel cuando está produciendo su potencia máxima.
- Tensión de corto circuito: Es la tensión que circula en el panel cuando su salida se encuentra en corto circuito.
- Corriente de corto circuito: Es la corriente que circula en el panel cuando la salida se encuentra cortocircuitada.

7.1.2 ACUMULADOR

Los acumuladores o baterías son dispositivos que mediante la transformación de energía potencial química a energía eléctrica permite alimentar la carga durante determinado tiempo cuando la producción del panel es baja o no hay presencia de radiación solar. Están formadas por dos electrodos sumergidos en un electrolito donde se produce la carga y descarga a través de reacciones químicas. Las baterías son recargadas por la electricidad proporcionada por los paneles, a través de un regulador de carga.

La batería cumple con tres funciones importantes, la primera es el almacenamiento de energía para un determinado número de tiempo, variando según sea la carga; la segunda corresponde a proporcionar una potencia instantánea elevada y por ultimo debe proporcionar una tensión de trabajo fija.

Los parámetros de fábrica importantes son:

- Capacidad: Es la cantidad de electricidad que puede obtenerse en la descarga completa del acumulador al estar en un estado de carga total.
- Eficiencia de carga: Corresponde a la relación entre la energía utilizada para cargar la batería y la energía que realmente es almacenada, idealmente debería ser del 100% así no se producirían perdidas de energía.

- Autodescarga: Se presenta cuando el acumulador se descarga autónomamente sin ser empleado en ningún uso.

7.1.3 REGULADOR O CONTROLADOR DE CARGA

La función del regulador de carga en el sistema es evitar que se causen daños al acumulador o batería debido a las sobrecargas de energía eléctrica que produce el panel limitando estos valores. El regulador trabaja en dos zonas para lograr su objetivo, en la zona de carga al garantizar una carga suficiente para el acumulador y en la zona de descarga para asegurar el suministro energético suficiente y evitar una descarga excesiva.

Los parámetros entregados por el fabricante comúnmente son:

- Tensión nominal: Es la tensión nominal de las baterías, por lo tanto la tensión nominal de la instalación.
- Corriente máxima en generación: La corriente eléctrica máxima recibida desde los paneles solares.
- Corriente máxima de consumo: Es el máximo de corriente que el sistema puede proporcionar.
- Perdida máxima generación/consumo: Es la relación existente entre la carga generada y consumida y relacionada con las caídas de tensión internas que producen perdidas.
- Autoconsumo: Es la energía necesaria para que el funcionamiento del propio regulador, como cualquier otro dispositivo eléctrico genera u gasto de energía.
- Sobrecarga: Es el porcentaje por encima al valor nominal y respecto al mismo que el regulador soporta sin producirse ningún daño.

7.2 CÁLCULOS DE DISEÑO FOTOVOLTAICO

De acuerdo al consumo de los componentes de cada estación se hizo el diseño para obtener los valores adecuados de cada dispositivo solar.

Inicialmente es importante saber la demanda de potencia de cada punto, para lo cual es necesario saber el consumo aproximado de corriente y la tensión de

trabajo de cada componente. A continuación y a partir de cada tabla de dispositivos y consumo, se detallan los cálculos realizados para cada punto o estación.

Se utilizó la Ecuación 10 para determinar la potencia total consumida en cada punto o estación.

Ecuación 10. Potencia en cada estación

$$P_n = I_n * V_n$$

La Ecuación 11 permitió determinar el consumo diario por estación.

Ecuación 11. Potencia de consumo diaria o demanda energética

$$C_{día} = DE = P_n * 24$$

La potencia pico mínima entregada por el panel solar o módulo fotovoltaico se calculó mediante la Ecuación 12.

Ecuación 12. Potencia pico de panel fotovoltaico

$$\boldsymbol{PP_n = \frac{FP * DE}{IS}}$$

La Ecuación 13 y la Ecuación 14 permitieron hallar la capacidad de almacenamiento de las baterías.

Ecuación 13. Capacidad máxima de las baterías en Wh

$$P_{Bn} = \frac{AUT * DE}{EF * DEM}$$

Ecuación 14. Capacidad máxima de las baterías en Ah

$$\boldsymbol{C_{Bn} = \frac{P_{Bn}}{V}}$$

El amperaje mínimo que debe soportar el regulador de carga fotovoltaica se halló mediante la Ecuación 15.

Ecuación 15. Corriente mínima del regulador de carga

$$\boldsymbol{I_{R1} = \frac{PP_1}{V}}$$

Tabla 9. Dispositivos y consumo de estación A

DISPOSITIVOS	CONSUMO (mA)	VOLTAJE(V)
Sensor Ultrasonido	30	5
Arduino Nano	20	5
TOTAL	50(I_1)	5(V_1)

Potencia consumida:

$$P_1 = I_1 * V_1 = 0.05\ A * 5\ V = 0.25\ W$$

En una hora de funcionamiento se tendrá el consumo de 0.25 Wh, trabajando durante todo el día se tiene que el consumo diario es de:

$$C_{día} = 0.25Wh * 24 = 6\ Wh/día$$

Potencia mínima de panel solar:

- Demanda energética (DE): 6 Wh/día
- Radiación solar en la zona (IS) [2]: 4 KWh/m^2/día
- Compensación de pérdidas (FP): 1.2 (20%)

$$\boldsymbol{PP_1 = \frac{FP * DE}{IS} = \frac{1.2 * 6}{4} = 1.8\ Wp}$$

Capacidad de las baterías:

- Autonomía (AUT): 2 días
- Demanda energética (DE): 6 Wh/día
- Eficiencia (EF): 80 % para ciclo profundo
- Descarga máxima (DEM): 50%
- Voltaje del sistema (V): 12 V

$$P_{B1} = \frac{AUT * DE}{EF * DEM} = \frac{2 * 6}{0.8 * 0.5} = 30\ Wh$$

$$\boldsymbol{C_{B1} = \frac{P_{B1}}{V} = \frac{30}{12} = 2.5\ Ah}$$

[2] Atlas de radiación solar. IDEAM
http://atlas.ideam.gov.co/visorAtlasRadiacion.html

Corriente mínima de regulador de carga:

- Potencia máxima de panel (PP_1): 1.8 Wp
- Voltaje del sistema (V): 12 V

$$\boldsymbol{I_{R1} = \frac{PP_1}{V} = \frac{1.8}{12} = 150\ mA}$$

Aunque los dispositivos funcionan a 5 V el controlador de carga entrega esta tensión por medio de puerto USB, no es la batería quien entrega dicha tensión directamente por ello se calcula todo con 12 V (tensión de batería). También se ignora el consumo de los MAX485 debido a que su consumo no supera los 500 µA, se considera despreciable para el sistema.

Tabla 10. Dispositivos y consumo de estación B

DISPOSITIVOS	CONSUMO (mA)	VOLTAJE(V)
Sensor Ultrasonido	30	5
Sensor Interruptor	5	5
Relé	200	5
Arduino Nano	20	5
TOTAL	255(I_2)	5(V_2)

Potencia consumida:

$$P_2 = I_2 * V_2 = 0.16\ A * 5\ V = 1.28\ W$$

En una hora de funcionamiento se tendrá el consumo de 1.28 Wh y trabajando durante todo el día se tiene que el consumo diario es de:

$$C_{día} = 1.28Wh * 24 = 30.72\ Wh/día$$

Potencia mínima de panel solar:

- Demanda energética (DE): 30.72 Wh/día
- Radiación solar en la zona (IS): 4 KWh/m^2/día
- Compensación de pérdidas (FP): 1.2 (20%)

$$\boldsymbol{PP_2 = \frac{FP * DE}{IS} = \frac{1.2 * 30.72}{4} = 9.22Wp}$$

Capacidad de las baterías:

- Autonomía (AUT): 1 día
- Demanda energética (DE): 30.72 Wh/día
- Eficiencia (EF): 80 % para ciclo profundo
- Descarga máxima (DEM): 50%

$$P_{B2} = \frac{AUT * DE}{EF * DEM} = \frac{2 * 30.72}{0.8 * 0.5} = 153.6\ Wh$$

$$\boldsymbol{C_{B2} = \frac{P_{B2}}{V} = \frac{153.6}{12} = 12.8\ Ah}$$

Corriente mínima de regulador de carga:

- Potencia máxima de panel (PP_2): 9.22 Wp
- Voltaje del sistema (V): 12 V

$$\boldsymbol{I_{R2} = \frac{PP_2}{V} = \frac{9.22}{12} = 0.76\ A}$$

Tabla 11. Dispositivos y consumo de estación C

DISPOSITIVOS	CONSUMO (mA)	VOLTAJE(V)
Sensor Ultrasonido	30	5
Relé	100	5
Arduino Nano	20	5
Electroválvula	200	12
TOTAL	150(I_{31}) – 200(I_{32})	5(V_{31}) – 12(V_{32})

Potencia consumida:

$$P_3 = I_{31} * V_{31} + I_{32} * V_{32} = 0.15\ A * 5\ V + 0.2\ A * 12\ V = 3.15\ W$$

En una hora de funcionamiento se tendrá el consumo de 3.15 Wh, trabajando durante todo el día se tiene que el consumo diario es de:

$$C_{día} = 3.15Wh * 24 = 75.6\, Wh/día$$

Potencia mínima de panel solar:

- Demanda energética (DE): 75.6 Wh/día
- Radiación solar en la zona (IS): 4 KWh/m^2/día
- Compensación de pérdidas (FP): 1.2 (20%)

$$\boldsymbol{PP_3 = \frac{FP * DE}{IS} = \frac{1.2 * 75.6}{4} = 22.68\, Wp}$$

Capacidad de las baterías:

- Autonomía (AUT): 2 días
- Demanda energética (DE): 75.6 Wh/día
- Eficiencia (EF): 80 % para ciclo profundo
- Descarga máxima (DEM): 50%
- Voltaje del sistema (V): 12 V

$$P_{B3} = \frac{AUT * DE}{EF * DEM} = \frac{2 * 75.6}{0.8 * 0.5} = 378\, Wh$$

$$\boldsymbol{C_{B3} = \frac{P_{B3}}{V} = \frac{378}{12} = 31.5\, Ah}$$

Corriente mínima de regulador de carga:

- Potencia máxima de panel (PP_3): 22.68 Wp
- Voltaje del sistema (V): 12 V

$$\boldsymbol{I_{R3} = \frac{PP_3}{V} = \frac{22.68}{12} = 1.89\, A}$$

Tabla 12. Dispositivos y consumo de estación D

DISPOSITIVOS	CONSUMO (mA)	VOLTAJE(V)
Sensor Ultrasonido	30	5
3 Relés	300 (100 c/u)	5
FRDM K64F	115	5
3 Electroválvulas	600 (200 c/u)	12
TOTAL	$445(I_{41}) – 200(I_{42}) – 400(I_{VRi})$	$5(V_{41}) – 12(V_{42})$

Potencia consumida:

$$P_4 = I_{41} * V_{41} + I_{42} * V_{42} = 0.45\ A * 5\ V + 0.2\ A * 12\ V = 4.65\ W$$

$$P_{VR} = I_{VRi} * V_{42} = 0.4\ A * 12\ V = 4.8\ W$$

Teniendo en cuenta que 2 de las 3 electroválvulas son para riego y solo se activan durante 30 minutos diarios es importante hacer una separación de potencia para no tener un valores muy elevados e innecesarios, la tercera electroválvula tiene un funcionamiento dependiente de las variables del sistema, aun así supondrá el funcionamiento de 24 horas, aunque de antemano se conoce que no es así pero dará un rango adicional de seguridad en la autonomía.

En una hora de funcionamiento se tendrá el consumo de 4.65 Wh y 4.8 Wh respectivamente, trabajando durante todo el día y media hora para electroválvulas de riego se tiene que el consumo diario es de:

$$C_{día} = 4.65Wh * 24 + 4.8Wh * 0.5 = 114\ Wh/día$$

Potencia de panel solar:

- Demanda energética (DE): 114 Wh/día
- Radiación solar en la zona (IS): 4 KWh/m^2/día
- Compensación de pérdidas (FP): 1.2 (20%)

$$\boldsymbol{PP_4 = \frac{FP * DE}{IS} = \frac{1.2 * 114}{4} = 34.2Wp}$$

Capacidad de las baterías:

- Autonomía (AUT): 1.5 días
- Demanda energética (DE): 114 Wh/día
- Eficiencia (EF): 80 % para ciclo profundo
- Descarga máxima (DEM): 50%
- Voltaje del sistema (V): 12 V

$$P_{B4} = \frac{AUT * DE}{EF * DEM} = \frac{1.5 * 114}{0.8 * 0.5} = 427.5\ Wh$$

$$\boldsymbol{C_{B4} = \frac{P_{B4}}{V} = \frac{427.5}{12} = 35.6\ Ah}$$

Corriente mínima de regulador de carga:

- Potencia máxima de panel (PP_4): 34.2 Wp
- Voltaje del sistema (V): 12 V

$$\boldsymbol{I_{R4} = \frac{PP_4}{V} = \frac{34.2}{12} = 2.85\ A}$$

Tabla 13. Dispositivos y consumo estación R

DISPOSITIVOS	CONSUMO (mA)	VOLTAJE(V)
Raspberry Pi	800	5
Ratón M-UAE96	100	5
TOTAL	900(I_R)	5(V_R)

Potencia consumida:

$$P_R = I_R * V_R = 0.9\ A * 5\ V = 4.5\ W$$

En una hora de funcionamiento se tendrá el consumo de 4.5 Wh, trabajando durante todo el día se tiene que el consumo diario es de:

$$C_{día} = 4.5Wh * 24 = 108\ Wh/día$$

Potencia de panel solar:

- Demanda energética (DE): 108 Wh/día
- Radiación solar en la zona (IS): 4 KWh/m^2/día
- Compensación de pérdidas (FP): 1.2 (20%)

$$\boldsymbol{PP_R = \frac{FP * DE}{IS} = \frac{1.2 * 108}{4} = 32.4\, Wp}$$

Capacidad de las baterías:

- Autonomía (AUT): 1 día
- Demanda energética (DE): 108 Wh/día
- Eficiencia (EF): 80 % para ciclo profundo
- Descarga máxima (DEM): 50%
- Voltaje del sistema (V): 12 V

$$P_{BR} = \frac{AUT * DE}{EF * DEM} = \frac{1 * 108}{0.8 * 0.5} = 270\, Wh$$

$$\boldsymbol{C_{BR} = \frac{P_{BR}}{V} = \frac{270}{12} = 22.5\, Ah}$$

Corriente mínima de regulador de carga:

- Potencia máxima de panel (PP_R): 32.4 Wp
- Voltaje del sistema (V): 12 V

$$\boldsymbol{I_{RR} = \frac{PP_R}{V} = \frac{32.4}{12} = 2.7\, A}$$

7.3 ESTRUCTURAS DE INSTALACIÓN

De acuerdo a los valores obtenidos con los cálculos de diseño, se seleccionaron los dispositivos o componentes fotovoltaicos. En búsqueda de homogeneizar y teniendo en cuenta que las potencias de panel solar, son valores mínimos que satisfacen la demanda energética dada, se seleccionaron reguladores de mayor capacidad que permiten el uso de paneles de hasta 120 Wp (vatios pico).

Como se puede ver en la Tabla 14 se guardan las proporciones de los cálculos y en el caso de los paneles se eligieron valores múltiplos de 20 Wp, en las baterías

múltiplos de 7.5 Ah o 35 Ah y los reguladores solares todos de 10 A, de ahí que se pueda afirmar que 120 Wp es la potencia máxima admitida en los paneles solares.

Tabla 14. Componentes solares seleccionados

PUNTO	Panel calc.	Panel selec.	Batería calc.	Batería selec.	Regulador calc.	Regulador selec.
A	1.8 Wp	20 Wp	2.5 Ah	7.5 Ah	0.15 A	10 A
B	9.72 Wp	40 Wp	12.8 Ah	15 Ah	0.48 A	10 A
C	22.68	60 Wp	31.5	35 Ah	1.89 A	10 A
D	34.2 Wp	60 Wp	35.6 Ah	35 Ah	2.85 A	10 A
R	32.4 Wp	60 Wp	22.5 Ah	35 Ah	2.7 A	10 A

A continuación en las ilustraciones de las páginas 61 y 62 se presentan respectivamente los componentes de una estación solar con panel fotovoltaico de 60 Wp, batería seca de ciclo profundo de 35 Ah y regulador o controlador solar de 10 A.

Figura 23. Panel fotovoltaico de 60 Wp

Figura 24. Batería de ciclo profundo de 35 Ah

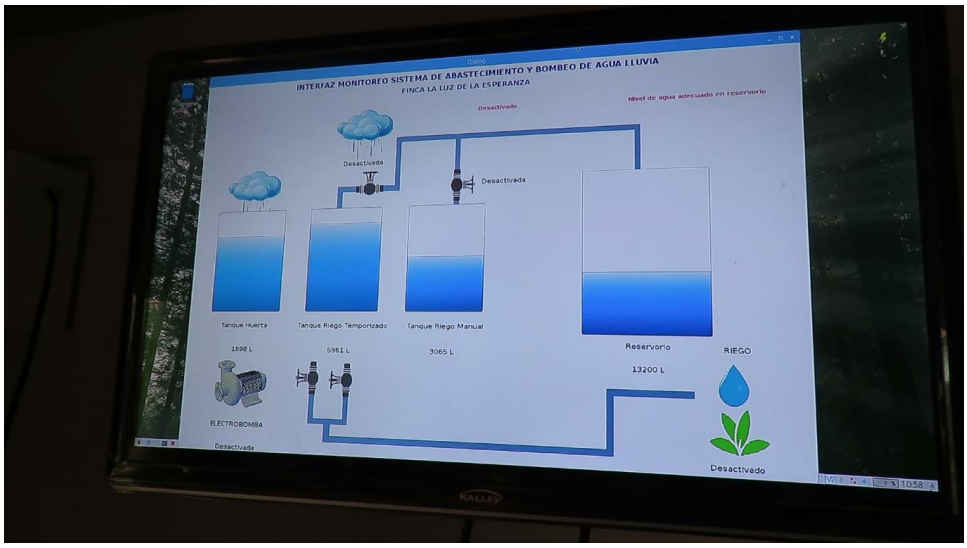
Figura 25. Controlador de carga solar de 10 A

Para la ubicación de los dispositivos se implementaron unas estructuras capaces de mantener paneles solares lo más alto posible y cajas que permiten mantener seguros los demás componentes, tanto regulador y batería, como los demás elementos electrónicos del sistema tales como sensores, microcontroladores, relés y optoacopladores. Un ejemplo de dichas estructuras se presenta en la Figura 26.

Figura 26. Estructura de instalación (estación C)

En la Figura 27 se presenta la HMI o estación central, que únicamente emplea una caja dentro de la casa que almacena los dispositivos de monitoreo y comunicaciones, además del respectivo monitor que permite la visualización.

Figura 27. Interfaz humano máquina HMI



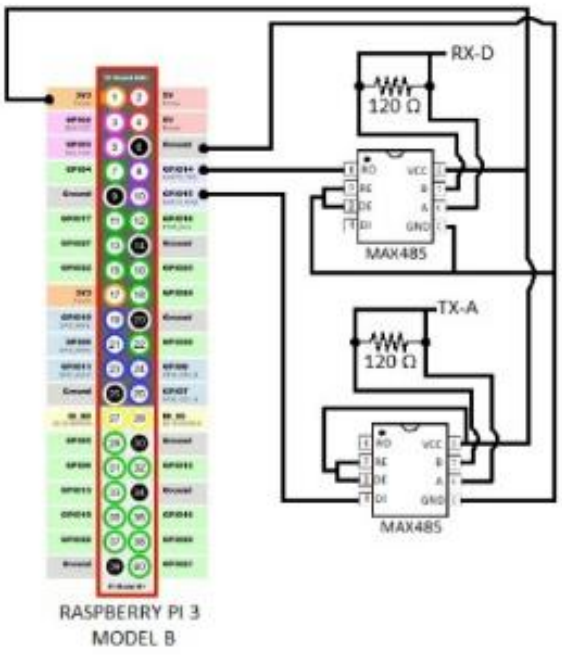
8. REPRESENTACIÓN GRÁFICA CIRCUITAL

Con el fin de que el lector comprenda en su totalidad el conexionado entre todos los componentes del sistema se realizaron representaciones gráficas de los circuitos, donde a manera esquemática se presentan los dispositivos y su cableado.

8.1 ESTACIÓN CENTRAL R

La dirección R, que se le dio a esta estación se da simplemente por el nombre del dispositivo principal que la compone, la SBC Raspberry Pi 3, en la Figura 28 además de la distribución de pines de la tarjeta se puede apreciar la conexión con los CIs transceptores MAX485 que permiten la comunicación.

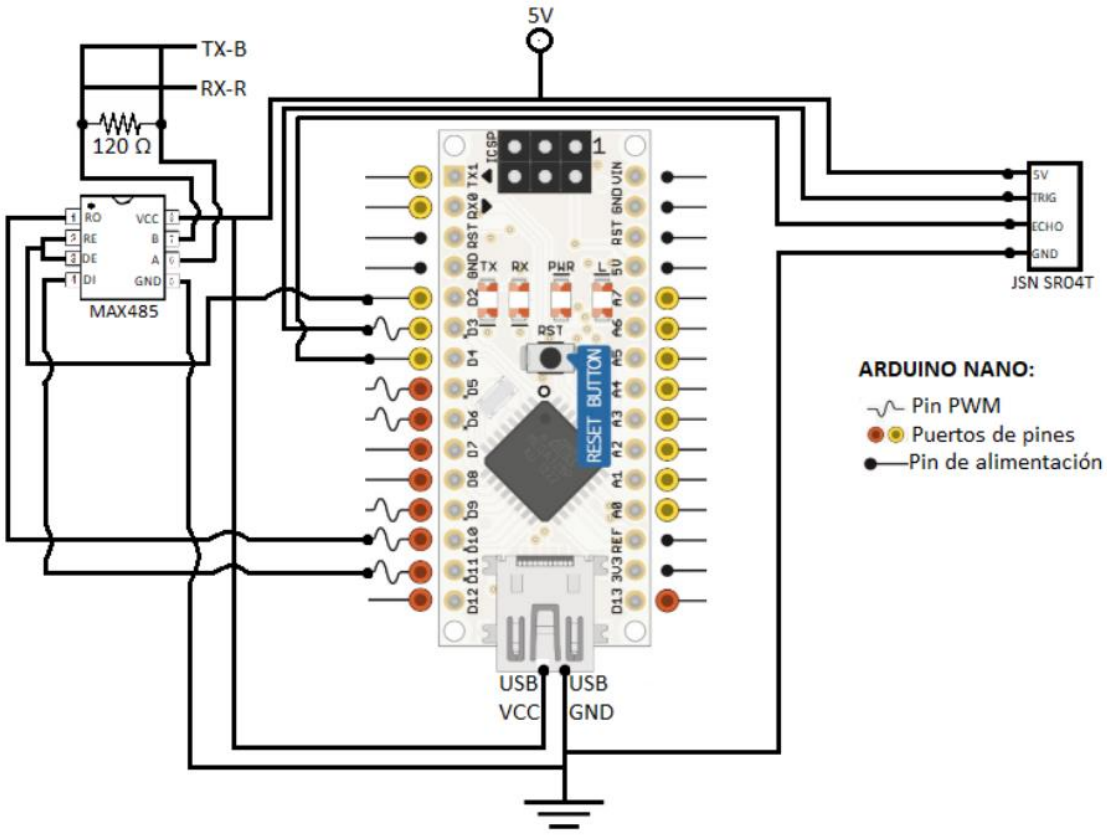
Figura 28. Representación gráfica del circuito en estación R



8.2 ESTACIÓN A

Como se aprecia en la Figura 29, la estación A está compuesta por un único integrado MAX485 trabajando en modo half dúplex, la tarjeta Arduino y el sensor ultrasónico. Recordar que esta estación se inicializa con la ejecución de la HMI y envía la lectura de nivel en el depósito hídrico llamado Reservorio.

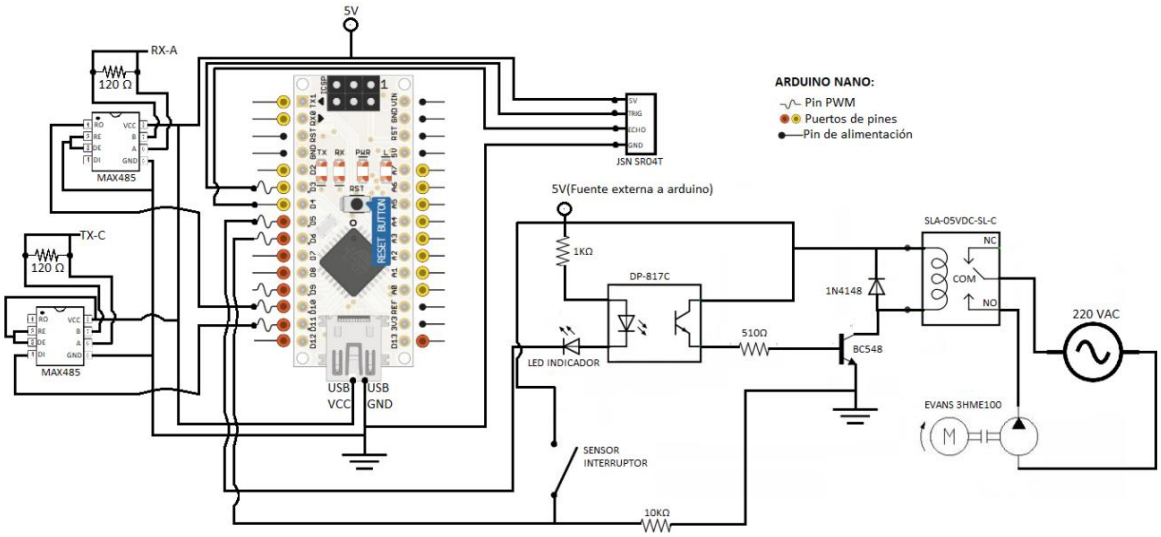
Figura 29. Representación gráfica del circuito en estación A



8.3 ESTACIÓN B

Estación encargada del bombeo automático, cuenta con los transceptores de comunicación, sensor de nivel ultrasónico, sensor de nivel por interruptor, y el circuito encargado del aislamiento de la tarjeta de control de los 220 VAC que hacen funcionar la electrobomba. Se enseña en la Figura 30.

Figura 30. Representación gráfica del circuito en estación B



El circuito aislante viene conformado por un módulo que contiene el opto acoplador, transistor y relé necesarios para cumplir con su función, puede ser apreciado en la base de la caja para proyectos electrónicos en la Figura 31.

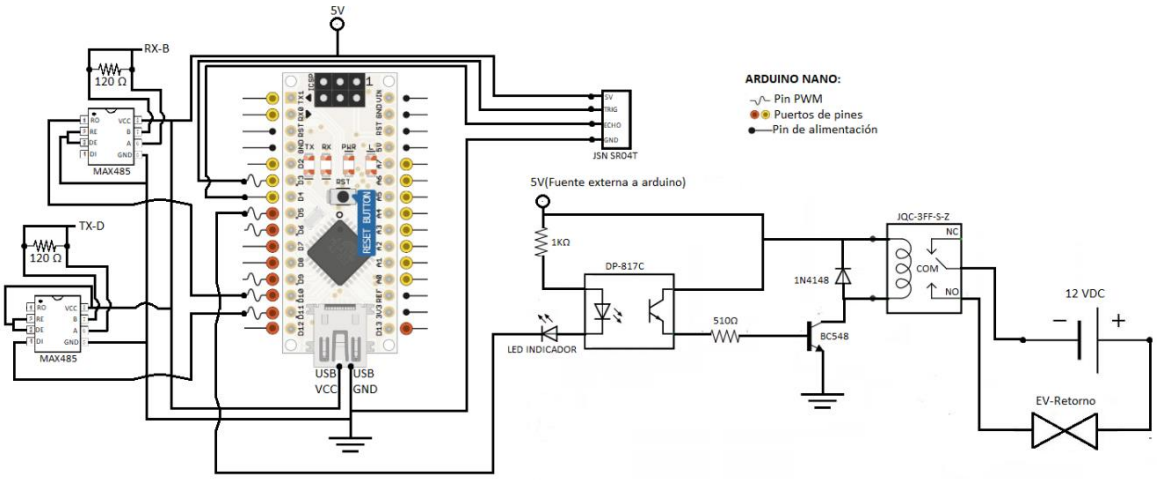
Figura 31. Módulo circuito aislante estación B

El relé utilizado en este circuito es de activación seleccionable en bajo o alto y cuenta con mayor capacidad de resistir altas tensiones y voltajes (250 VAC – 30 A).

8.4 ESTACIÓN C

Esta estación como se ve en la Figura 32 contiene básicamente los mismos componentes que la anterior, pero sin el uso del sensor de nivel interruptor y reemplazando el actuador por una electroválvula que trabaja a 12 VDC.

Figura 32. Representación gráfica del circuito en estación C



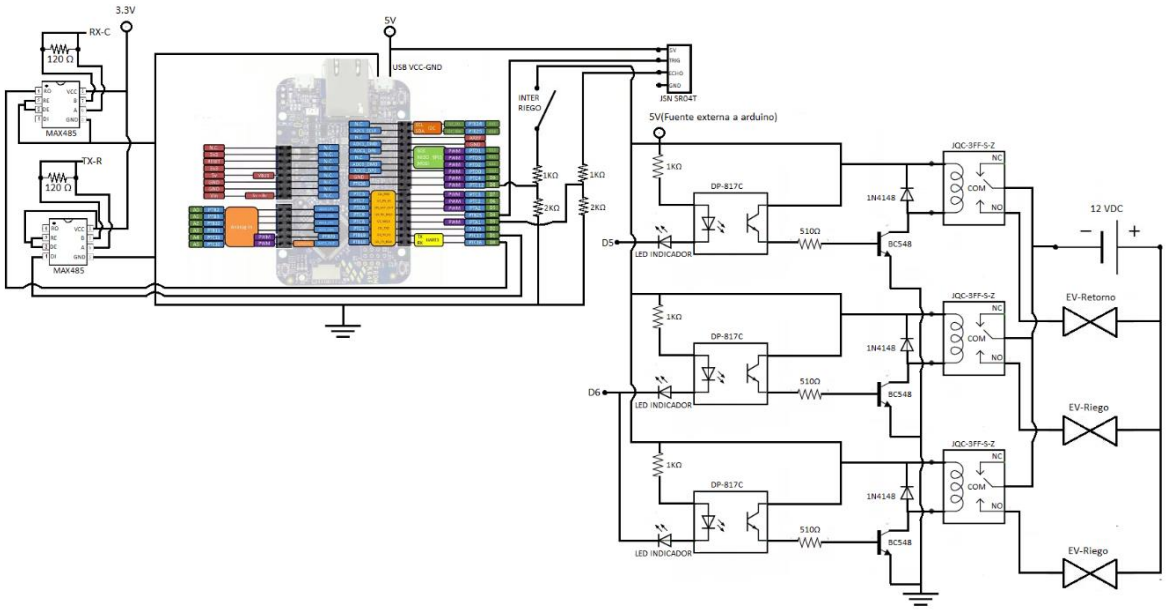
En la Figura 33, el relé utilizado en este circuito. Es de activación en bajo y de menor capacidad que la referencia del punto anterior, pero es competente para su función.

Figura 33. Módulo circuito aislante en estaciones C y D

8.5 ESTACIÓN D

Es la estación con mayor número de componentes, pues como se aprecia en la Figura 34 además de utilizar un dispositivo de control diferente, incorpora la placa de divisores de tensión para adaptar el sensor y el interruptor de desactivación de riego. Por otro lado el número de actuadores en este punto es mayor pues se dispone de dos electroválvulas de riego y una de retorno y llenado automático.

Figura 34. Representación gráfica del circuito en estación D

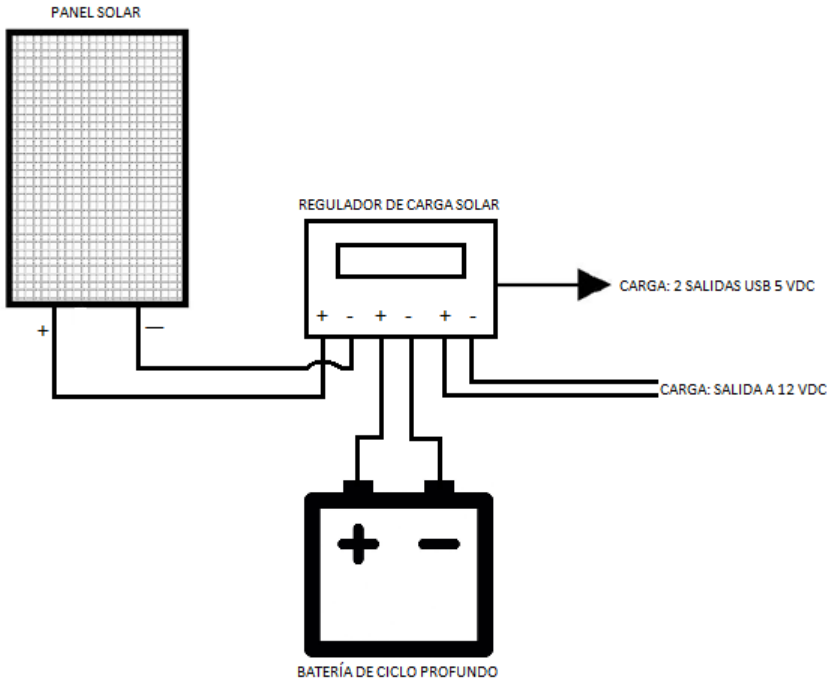


El módulo de aislamiento y protección del dispositivo de control es el mismo de la estación C (Figura 33).

8.6 INSTALACIONES SOLARES

La instalación de los componentes solares es un proceso homólogo en cada una de las estaciones y se puede apreciar de forma general en la Figura 35.

Figura 35. Representación gráfica del circuito en instalaciones solares



Las imágenes correspondientes a la representación gráfica del circuito en estaciones B, C y D se anexan con mejor calidad por si desean verse a mayor detalle.

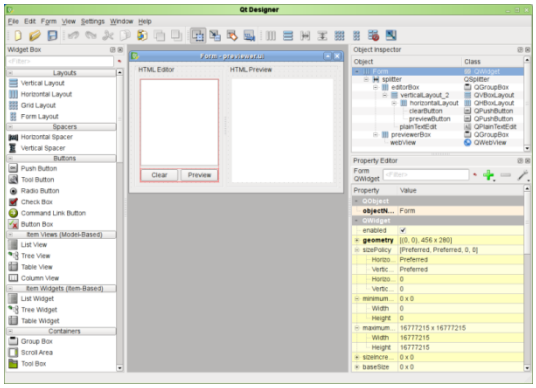
9. INTERFAZ GRÁFICA DE MONITOREO

En este apartado se documenta el software y hardware utilizado para poder realizar la interfaz gráfica de monitoreo o en términos SCADA la interfaz humano máquina HMI que facilita la supervisión del estado de las variables del proceso.

9.1 QT DESIGNER[3]

Corresponde a una de las herramientas dispuestas en QT Creator que es el IDE para diseñar y desarrollar aplicaciones con interfaz gráfica de usuario (GUI), todo dentro de la estructura QT, framework libre bajo licencia LGPL y escrito en C++. Para el desarrollo del proyecto solo se utilizó la herramienta QT Designer que permite el diseño netamente gráfico o visual de la aplicación, dispone una serie de Widgets que se integran al código por el método de señales y slots. En la Figura 36 se muestra el entorno de trabajo.

Figura 36. Entorno de trabajo QT Designer



Fuente: http://doc.qt.io/qt-5/designer-to-know.html

[3] QT Colaboradores, QT wiki. http://wiki.qt.io/Main

9.2 PYQT

Es uno de los bindings encargado de la transformación del lenguaje nativo de QT, C++, en uno compatible con el lenguaje de programación Python. Fue desarrollado por Riverbank Computing y está disponible para Windows, GNU/Linux y Mac OS X bajo diferentes licencias. Al igual que en QT, el desarrollo en PyQT se realiza mediante la programación orientada a objetos. Lo que se hace es correr un script que convierte código XML generado mediante el QT Designer a código Python donde cada uno de los widgets y sus características hacen parte de una clase y un método, creando un módulo que debe ser llamado en el código principal. Se debe entender como un simple binding, pues el desarrollo del código lo debe hacer enteramente el programador, creando las clases y métodos necesarios para obtener la aplicación final. El código realizado se adjunta en anexos.

9.3 COMPUTADOR DE PLACA SIMPLE: RASPBERRY PI 3

El dispositivo de bajo costo y consumo y que además cumple con todas las prestaciones necesarias para correr la aplicación diseñada es el SBC Raspberry Pi 3 Modelo B. Con un procesador ARM de 64 bits (4 núcleos), 1GB de RAM, puerto HDMI y unidad de procesamiento gráfico (GPU) encaja perfectamente como dispositivo hardware HMI. Es una placa que se ha popularizado mucho por su gran rendimiento y versatilidad, por ello no se entrará en detalles respecto a sus características o especificaciones técnicas. Sin ningún periférico tiene un consumo de 800 mA. Se puede observar en la Figura 37.

Figura 37. Computadora de placa simple Raspberry Pi 3

Fuente: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

9.4 CARACTERÍSTICAS DE LA INTERFAZ

Como se puede apreciar en la Figura 27 la interfaz cuenta con la representación gráfica de cada uno de los tanques de almacenamiento, el Reservorio se representa como un tanque de mayor tamaño; cuentan con una medida de nivel que es el volumen aproximado en litros. También tiene la representación gráfica de los actuadores es decir electroválvulas y electrobomba y sus respectivos estados lógicos (Activado/Desactivado), al igual que el estado del riego. Como último, se reservó un espacio en la parte superior para llamar las alertas en caso de ser necesario.

9.5 DESCRIPCIÓN DE LA PROGRAMACIÓN

Como ya se mencionó el lenguaje utilizado para desarrollar la aplicación software de monitoreo es Python en su versión 2.7. El programa se divide en tres partes principales:

- Primero se ejecuta la ventana principal (últimas 5 líneas de código) en la cual se llama la clase principal llamada *InterfazMonitoreo*.
- Hay una clase llamada *HiloClase* que utiliza el objeto QThread de PyQt para crear un hilo, es decir que ésta parte del código se ejecuta de forma paralela y simultánea al resto de la programación, es la única forma de hacerlo puesto que se utiliza un bucle infinito para la lectura de datos. Una vez se leen los datos se emiten como señal.
- Qt es un framework que se trabaja mediante el concepto de señales (signals) y ranuras (slots) por ello la tercera parte del código es la clase principal *InterfazMonitoreo* en la cual se conectan las señales emitidas a sus respectivas ranuras que en términos de programación son métodos de la clase. En cada uno de estos métodos se utilizan los datos recibidos en los respectivos widgets que permiten la visualización gráfica requerida en cada caso.

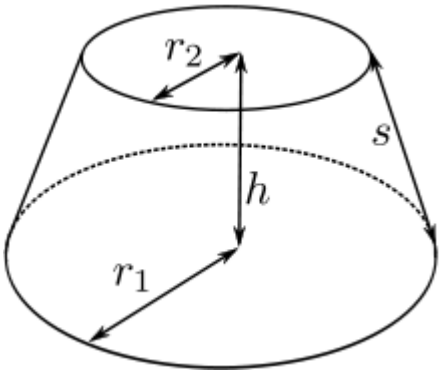
Esta es una descripción muy general, pero en el código anexado se utilizan comentarios que detallan toda la programación.

9.5.1 VOLUMEN EN LITROS

Los sensores de ultrasonido entregan la distancia a la que se encuentra el agua en los tanques, en términos de programación de control es adecuado, pero para el usuario la forma más amigable de ver la información es leyéndola como nivel en volumen, específicamente en litros y para conseguir este fin, fue necesario utilizar un método geométrico que permite convertir la distancia entregada por el sensor en litros, como se muestra a continuación:

Lo primero es saber la forma geométrica de los tanques, conocida como tronco de cono que es la porción de cono comprendido entre dos planos que lo cortan y son perpendiculares a su eje como se observa en la Figura 38.

Figura 38. Tronco de cono



Fuente: https://es.wikipedia.org/wiki/Tronco_de_cono#/media/File:CroppedCone.svg

Sabiendo esto, es posible calcular el volumen de la figura geométrica que corresponde a la Ecuación 16.

Ecuación 16. Volumen del tronco de cono

$$V = \frac{h\pi}{3}(r1^2 + r2^2 + r1r2)$$

Debido a que el tanque tiene la forma de tronco de cono invertido respecto a la Figura 38 se sabe que el radio inferior es constante pero el radio superior es

variable, de esta manera, realizando la medida de los radios superior e inferior es posible por medio de una función lineal, relacionar radio respecto a la altura.

Se procedió a realizar la medición de la altura y los perímetros superior e inferior como se muestra en la Figura 39 y así poder hallar los respectivos radios de acuerdo a la .

Ecuación 17. Radio respecto a perímetro

$$r = \frac{P}{2\pi}$$

Luego, es posible realizar la linealización, en el cual el radio depende de la altura, a continuación se procede a realizar el procedimiento para el tanque de dos mil litros de la estación B:

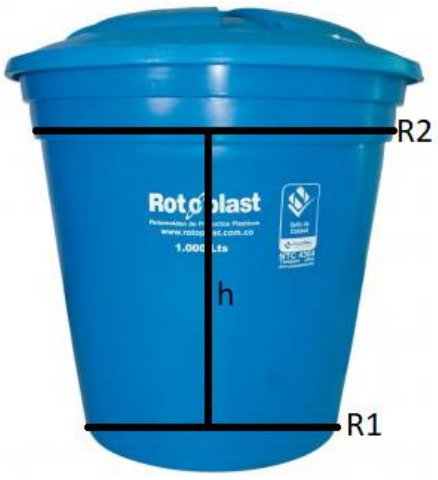
$$R_2 = \frac{436\ cm}{2\pi} = 69.39\ cm$$

$$R_1 = \frac{380}{2\pi} = 60.48\ cm$$

$$h = 120\ cm$$

$$P_2 = (120\ , 69.39) \qquad P_1 = (0\ , 60.48)$$

Figura 39. Puntos aproximados de medida de perímetros y altura



Fuente: http://www.rotoplast.com.co/tanques-rotoplast-y-acuaplast/

De acuerdo a los puntos dados y la Ecuación 18 se halló la función lineal según la pendiente (Ecuación 19) y el punto de corte con el eje Y (Ecuación 20).

Ecuación 18. Función lineal y relación con la geometría de los tanques

$$y = mx + b \quad \rightarrow \quad R_2 = mh + b$$

Ecuación 19. Pendiente

$$m = \frac{y_2 - y_2}{x_2 - x_1} = \frac{69.39 - 60.48}{120 - 0} = 0.07425$$

Ecuación 20. Punto de corte con el eje Y

$$60.48 = 0.07425 * 0 + b \quad \rightarrow \quad b = R_1 = 60.48$$

Finalmente la función lineal encontrada que relaciona altura y radio 2 es:

$$R_2 = 0.07425h + 60.48$$

De esta manera es posible hallar el volumen del tanque, únicamente en términos de altura como se aprecia a continuación:

$$V = n * \frac{\frac{h\pi}{3}[60.48^2 + (0.07425h + 60.48)^2 + 60.48(0.07425h + 60.48)]}{1000}$$

Donde n es el número de tanques y h es la altura proporcionada por el sensor ultrasónico (diferencia de la altura donde se encuentra puesto el sensor y el dato entregado por el mismo). La división por mil, se da debido a que la ecuación entrega el resultado en centímetros cúbicos y se espera en litros. Ejemplo, para un solo tanque y una altura de 80 cm el volumen es de 1013 litros.

10. RESULTADOS Y ANÁLISIS

10.1 EFICIENCIA HÍDRICA AL AUTOMATIZAR EL BOMBEO

La automatización del bombeo de agua pluvial logró disminuir el riesgo de error humano en la activación y desactivación de la electrobomba para el traslado de agua recolectada hacia el reservorio puesto que anteriormente para determinar el nivel del tanque adecuado para la desactivación de la bomba era necesario desplazarse hacia el punto de recolección y supervisar el vaciado para avisar a otro operario de su desactivación.

El bombeo, activado o desactivado según la lectura proporcionada por el sensor ultrasónico permite un aprovechamiento mucho más completo y eficiente del recurso pues no se desaprovecha a ninguna hora del día, muchas veces el operario se encuentra realizando otras tareas o si el evento pluvial se da en horas de la noche mientras se descansa, toda esta agua se desaprovecha con métodos manuales.

10.2 ABASTECIMIENTO HÍDRICO CONSTANTE

El almacenamiento de agua pluvial requería un constante desplazamiento hacia todos los puntos de recolección de agua de los techos, por lo que el operario prefería utilizar agua de nacimiento para uso en el riego de activación manual al invernadero ubicado en la parte inferior de la finca. A través de la automatización en el suministro de agua a los puntos de riego mediante el retorno de agua pluvial desde el reservorio, se consigue una disponibilidad de agua permanente en estos puntos y reduce la tarea a la simple activación manual de la válvula que proporciona el riego en el caso del invernadero que posee cultivos variados.

10.3 USO RAZONABLE DEL AGUA CON EL RIEGO TEMPORIZADO

La implementación de riego por goteo utilizando goteros autocompensantes permite que la cantidad de agua suministrada a la planta sea exacta pero a su vez requiere de exactitud en el suministro del agua debido a que la cantidad del líquido se determina con respecto al tiempo asegurando en este caso, 1 litro por 30 minutos en cada planta. Con la temporización del riego se asegura la exactitud en la distribución del agua, además de permitir un uso racional.

10.4 INTERFAZ HUMANO MÁQUINA QUE AUMENTA PRODUCTIVIDAD

Como se mencionó anteriormente la disponibilidad de agua y su uso en el riego de los cultivos de invernadero requerían de un constante desplazamiento a los puntos de interés y el empleo de uno o varios operarios de la finca. El desarrollo de la interfaz gráfica que permite la visualización de las variables del sistema y el estado de los actuadores le permite al operario la supervisión desde un solo punto y disminuye considerablemente el tiempo invertido en todas estas tareas lo que se traduce en un aumento en tiempo de disponibilidad para otras tareas de la finca.

10.5 ENERGÍA SOLAR LIMPIA Y SUSTENTABLE

La instalación de un sistema de alimentación fotovoltaico permitió la alimentación de los dispositivos de bajo consumo durante todo el día y toda la noche de manera satisfactoria lo que permitió que el consumo de energía eléctrica no aumentara y por ende no subiera en costos. También se logra una buena contribución ambiental pues este tipo de energía es amigable con la naturaleza, es ideal para medios agrícolas o urbanos.

10.6 LIMITACIONES EN LA COMUNICACIÓN SERIAL

La utilización de dispositivos de bajo consumo como las tarjetas Arduino Nano implican una velocidad de reloj[4] relativamente baja por lo que al momento de transmitir datos mediante el puerto serial se convierte en un factor determinante para el envío de información ya que al aumentar la velocidad en transmisión los datos presentan más errores tanto en el envío como recepción, por lo que se determinó trabajar con la tasa de 9600 baudios con la que se obtuvo la mejor relación velocidad errores.

10.7 INCONVENIENTES HARDWARE Y SOLUCIONES SOFTWARE

La utilización de sensores JSN-SR04T presentó varios inconvenientes debido a que su respuesta ante perturbaciones en la superficie del líquido ocasionadas por la corriente de agua que cae a los tanques en el almacenamiento genera en ocasiones, una medición inadecuada, es decir su medición es acertada únicamente en líquidos donde la perturbación es mínima, por lo que se determinó su instalación en tanques donde el nivel cambia sin perturbar la superficie, esto debido a que dos de los sistemas de almacenamiento consisten en varios tanques conectados mediante el método de vasos comunicantes lo que permite este hecho. Adicional a estas medidas físicas, a nivel de software se incluyeron códigos propios de detección y corrección de errores, en la lectura de sensor y en la recepción de los datos en la unidad central donde se visualiza el estado de las variables.

10.8 FLEXIBILIDAD EN LA TRANSMISIÓN DE DATOS

El uso del estándar físico RS485 permite la utilización de un protocolo propio, desde la topología de red, la trama de datos y los métodos de corrección y detección de errores. Aunque puede ser más complicado, el hecho de poder definir un protocolo propio en las comunicaciones permite una mayor flexibilidad para adaptarse a la necesidad particular en la transmisión de los datos. El uso del integrado MAX485 no supuso ningún problema y se comportó muy bien a pesar de tener que comunicar puntos con distancias de más de 150 metros.

[4] Comunicación serial y velocidad de transmisión
http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1

11. CONCLUSIONES

- Se logró la automatización de un sistema de bombeo y almacenamiento de agua pluvial, permitiendo su aprovechamiento en el riego de diferentes cultivos de la finca y disminuyendo la exigencia en mano de obra para esta labor. A través de la aplicación y mejora del concepto de cosecha de agua agregándole el componente tecnológico se permite un uso eficiente y racional del agua, disminuyendo el consumo domiciliario y de nacimientos naturales, siendo este un recurso limitado. La implementación de este tipo de sistemas automáticos incentiva la aplicación de conceptos sustentables con el medio ambiente generando un cambio positivo en el mismo.

- Con la implementación de un sistema de riego automático temporizado para un cultivo en invernadero, se puede dar un uso eficiente y exacto del agua lluvia recolectada, mejorando la producción y calidad medio ambiental de la finca La luz de la esperanza. Además se evidencia la disminución de las tareas a los empleados de tal manera que ese tiempo sea usado en otros asuntos productivos.

- La implementación de sistemas alimentados por energía solar facilita la instalación de diferentes puntos de control remoto de procesos donde se hace difícil llevar puntos de la red eléctrica domiciliaria y a su vez permite la aplicación del sistema en fincas con áreas más extensas siendo totalmente independientes y auto sostenibles tanto económica, ambiental y energéticamente.

- Se realizó una aplicación software que proporciona una interfaz de monitoreo que posee todos los elementos del sistema representados con iconos y figuras de fácil entendimiento para el usuario, quien no requiere de un conocimiento avanzado en tecnología para interpretarlo. Este sistema para la supervisión disminuyó considerablemente la necesidad de desplazamiento constante hacia los puntos de bombeo, almacenamiento y riego del agua lluvia captada.

- Las tarjetas de desarrollo seleccionadas como dispositivos de control fueron competentes y permitieron la realización del sistema de automatización

planteado. En general tuvieron un buen comportamiento durante el tiempo en que se hizo un seguimiento del sistema trabajando las 24 horas y cumpliendo con las funciones designadas.

- La transmisión de datos presentó algunos problemas en los dispositivos con baja velocidad de reloj a altas tasas de baudios, pero el estándar MAX485 en su flexibilidad para permitir la auto definición del protocolo de comunicaciones, sumado a líneas de código destinadas a la corrección de errores, permitió un control exitoso de la variable nivel y su visualización correcta en la interfaz humano máquina.

- La utilización de la computadora de placa simple Raspberry Pi 3 fue eficiente a la hora de procesar los datos y graficarlos, su gran capacidad hardware como lo es el tener cuatro núcleos en el procesador, permitió la utilización de hilos o procesos paralelos multitarea en el desarrollo de la interfaz gráfica.

- Para el desarrollo del software de control se utilizó básicamente el lenguaje C++ y programación estructurada, usando algunos apartes de programación orientada objetos, como el uso de constructores de objetos de clases para la utilización de interrupciones, puertos seriales, temporizadores u otras funciones propias de los microcontroladores usados en las tarjetas de desarrollo. En cuanto al desarrollo del software para la interfaz de monitoreo se utilizó lenguaje Python y principalmente programación orientada a objetos y algo de programación modular para el llamado de widgets o librerías.

- Se implementó un sistema que cumple con conceptos SCADA, que comúnmente se llevan a cabo solo en ámbitos industriales y con componentes de alto costo como PLC’s y sensores y actuadores industriales. Aunque a menor escala se logra la implementación de un sistema que cabe dentro de la categoría.

RECOMENDACIONES

- Una fuerte lluvia puede producir grandes volúmenes de agua que de no ser almacenados se desperdician, por esto es recomendable la extensión en la capacidad de almacenamiento y bombeo, ya que en el punto donde se ha dispuesto una electrobomba para el traslado del líquido solo dispone de la capacidad de un tanque de 2000 litros cuyo volumen bombeado es de aproximadamente la mitad debido al margen de protección de la electrobomba o también la disposición de otro punto de bombeo.

- La administración de los invernaderos en busca de la mejora de la producción no solo requiere de un suministro adecuado del agua, se presentan muchas otras necesidades que pueden ser solucionadas en la medida que se pueda expandir la cantidad de variables involucradas en el proceso de control automático para llevar a la inclusión de procesos como el fertirriego, control de temperatura, control de plagas, entre otros.

- La aplicación del concepto de cosecha de agua mediante la automatización y bombeo de agua lluvia también requiere de un aumento de variables como el flujo de agua para poder generar información estadística más detallada de este proceso y llevar a la cuantificación de eficiencia y la incentivación de su uso.

- La gran avanzada mundial de internet permite que se puedan conectar a la red dispositivos que antes eran impensables, actualmente se está viviendo la era de internet de todo (IdT) y cada vez se busca más la implementación de sistemas basados en la red que permitan supervisión y control desde cualquier lugar del mundo, por lo cual se recomienda la implementación de una interfaz conectada a la nube que aproveche dispositivos ya instalados y funcionales para la IdT como la FRDM K64 F o la Raspberry Pi 3.

BIBLIOGRAFÍA

- EBEL, F. IDLER, S. PREDE, G. Scholz, D. Fundamentos de la técnica de Automatización. Denkendorf. : Festo Didactic GmbH & Co, 2007. 106p. Referencia 563062.
- CORONA, Leonel G. ABARCA, Griselda S. CARREÑO Jesús M. Sensores y actuadores aplicaciones con Arduino. México D.F.: Grupo Editorial Patria, 2014.304p. ISBN 978-607-438-936-4.
- ALONSO, Nuria O. CASTRO, Manuel A. DIAZ Gabriel. Redes de comunicaciones Industriales. Madrid: UNED. Universidad Nacional de Educación a Distancia, 2013. 485p. ISBN 978-84-362-6549-1.
- DIAZ, Tomás. CARMONA, Guadalupe. Instalaciones solares fotovoltaicas.GM. Madrid: McGraw-Hill Interamericana de España S.L, 2010. 200p. ISBN 978-84-481-7169-8.
- ROLDÁN, José. Instalaciones solares fotovoltaicas. Madrid: S.A. Ediciones Paraninfo, 2010. 408p. ISBN 978-84-283-3203-3.
- ORIOL, José. Elementos de geometría y dibujo lineal para uso de las escuelas. Barcelona: Nabu Press,2013. 156p. ISBN 129-41-0174-9.
- CREUS, Antonio. Instrumentación industrial. México D.F.: Alfaomega Grupo Editor S.A, 2010. 792p. ISBN 978-607-707-042-9.

ANEXOS

ANEXO A. Programación en estaciones de control

```
/*Programación realizada para la estación A
 * Reservorio
 * Granja La Luz de la esperanza
 * Daniel Linares - Mayerly Narváez
 * Proyecto de grado - USCO
 */

//Inclusión de la librería que permite la utilización de cualquier pin como Rx Tx Serial
#include <SoftwareSerial.h>
//Constructor del objeto max485 para la transmisión de datos
SoftwareSerial max485(10,11);
//Definición de variables
const char direcciona='a';    //Direcciones
const char direccionraspb='r';
char direccionr[2];
const int ReDePin =  2;   //Pin para funcionamiento half duplex
int tiempo, distanciaa;  //Variables para lectura de sensor ultrasonido
int trigger = 3;
int echo = 4;


void setup()
{
  //Configuración de pines e inicialización de los puertos seriales
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
  Serial.begin(9600);
  max485.begin(9600);
  max485.setTimeout(3000);
  pinMode(ReDePin, OUTPUT);
}
```

```
void loop()
{
 //Max485 como RX
 digitalWrite(ReDePin, LOW);
 //Lectura de dirección R (un solo Byte)
 max485.readBytes(direccionr, 1);
 //Inicialización del sistema (la dirección queda almacenada en el buffer serial si no
la vuelve a recibir)
 if(direccionr[0]==direccionraspb){

  //Max485 como TX
  digitalWrite(ReDePin, HIGH);

  //Lectura-Calculo del sensor ultrasonido
  //Pulso de 10 microsegundos
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);
  //Lectura del tiempo que dura en 1 el pin echo
  tiempo = (pulseIn(echo, HIGH));
  //Calculo de distancia
  distanciaa=tiempo/58;

  //Envio de datos
  max485.println(direcciona);
  max485.println(distanciaa);

  //Comprobacion de envio
  Serial.println(distanciaa);
  }
 }


/*Programación realizada para la estación B
 * Punto de bombeo
 * Granja La Luz de la Esperanza
 * Daniel Linares - Mayerly Narváez
 * Proyecto de grado - USCO
 */
```

```
//Inclusión de librería que permite las interrupciones por el Timer1 del microcontrolador
#include <TimerOne.h>
//Inclusión de la librería que permite la utilización de cualquier pin como Rx Tx Serial
#include <SoftwareSerial.h>
//Constructor del objeto max485 para la transmisión de datos
SoftwareSerial max485(10,11);
//Definición de variables
const char direcciona='a';    //Direcciones
const char direccionb='b';
int distanciaarray[2];    //Arreglos para detección y corrección de errores
int distanciabarray[2];
int bandera=0;
int tiempo, distanciaa, distanciab, estadoEB1, estadoNIB1;    //Datos de la trama
int trigger = 3;     //Variables para lectura de sensor ultrasonido
int echo = 4;
int EB1 = 5;     //Variable para electrobomba
int NIB1 = 6;    //Variable para lectura de sensor de nivel interruptor

void setup() {
  //Configuración de pines, inicialización de los puertos seriales y de interrupción para lectura de sensor
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(EB1, OUTPUT);
  pinMode(NIB1, INPUT);
  Serial.begin(9600);
  max485.begin(9600);
  max485.setTimeout(1000);
  Timer1.initialize(500000);
  Timer1.attachInterrupt(lectSensor);
  digitalWrite(EB1,HIGH);
}

void lectSensor(){
  //Función que se activa por interrupción de Timer cada 500 ms
  //Lectura-Cálculo del sensor ultrasonido
```

```
  //Pulso de 10 microsegundos para emitir señal
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);
  //Lectura del tiempo que dura en 1 el pin echo
  tiempo = (pulseIn(echo, HIGH));
  //Calculo de distancia
  distanciab=tiempo/(int)58;
}

void loop() {
  //Lectura de datos
    if (max485.read()==direcciona){
      distanciaa=max485.parseInt();

  /*Detección/Correccion de errores
   *Se detecta que los errores son valores por debajo de 20 que están fuera del rango de lectura del sensor
   *DATOA
    *Si el dato es bueno se guarda en ambos espacios del arreglo se pone la bandera a 1
    *este if sirve como una actualización del último dato bueno*/
    if (distanciaa>20 && bandera==0){
      distanciaarray[0]=distanciaa;
      distanciaarray[1]=distanciaa;
      bandera=1;
     }
   //Si el dato es bueno lo guardo en el segundo espacio del arreglo
      else if(distanciaa>20 && bandera==1){
        distanciaarray[1]=distanciaa;
        bandera=0;
      }
   //Si el dato es un error utilizo el último valor bueno guardado en "memoria"
      else if(distanciaa<20){
        distanciaarray[1]=distanciaarray[0];
      }
  //DATOB
  //Proceso homologo al anterior
    if (distanciab>20 && bandera==0){
```

```
      distanciabarray[0]=distanciab;
      distanciabarray[1]=distanciab;
      bandera=1;
     }
      else if(distanciab>20 && bandera==1){
        distanciabarray[1]=distanciab;
        bandera=0;
      }
      else if(distanciab<20){
        distanciabarray[1]=distanciabarray[0];
      }
  //Lectura de estados logicos
        estadoEB1=digitalRead(EB1);
        estadoNIB1=digitalRead(NIB1);

  //Condicion de control
  //Si el nivel es alto y el reservorio no está lleno enciende EB1
  //El estado EB1 funciona como bandera para ejecutar solo uno de los condicionales cada vez que cambie de estado el actuador
      if (distanciabarray[1]<=(int)41 && estadoEB1==1 && distanciaarray[1]>(int)37){
        digitalWrite(EB1,LOW);
      }
  //Si el nivel es bajo o el reservorio está lleno o se dispara el sensor de protección apaga EB1
      if (distanciabarray[1]>=(int)110 && estadoEB1==0 || distanciaarray[1]<=(int)37 || estadoNIB1==1){
        digitalWrite(EB1,HIGH);
      }

        //Envio de datos
        max485.println(direccionb);
        max485.println(distanciaarray[1]);
        max485.println(distanciabarray[1]);
        max485.println(estadoEB1);
        max485.println(estadoNIB1);

        //Comprobacion de recepcion y envio
        Serial.println(distanciaarray[1]);
        Serial.println(distanciabarray[1]);
```

```
      Serial.println(estadoEB1);
      Serial.println(estadoNIB1);
  }
}


/*Programación realizada para la estación C
 * Punto de almacenamiento
 * Granja La Luz de la Esperanza
 * Daniel Linares - Mayerly Narváez
 * Proyecto de grado - USCO
 */

//Inclusión de la librería que permite la utilización de cualquier pin como Rx Tx Serial
#include <SoftwareSerial.h>
//Inclusión de librería que permite las interrupciones por el Timer1 del microcontrolador
#include <TimerOne.h>
//Constructor del objeto max485 para la transmisión de datos
SoftwareSerial max485(10,11);
//Definición de variables
const char direccionb='b';    //Direcciones
const char direccionc='c';
int distanciacarray[2];    //Arreglo para detección y corrección de errores
int bandera=0;
//Variables datos de trama
int tiempo, distanciaa, distanciab, distanciac, estadoEB1, estadoNIB1, estadoEV1;
//Las mismas variables en string pues la FRDM solo lee valores en este tipo de dato
//Se utilizan arreglos de 16 char para darle espacio suficiente a la variable
char sdistanciaa[16], sdistanciab[16], sdistanciac[16], sestadoEB1[16], sestadoNIB1[16], sestadoEV1[16];
int trigger = 3;    //Variables para lectura de sensor ultrasonido
int echo = 4;
int EV1 = 5;    //Variable para electroválvula

void setup() {
```

```
  //Configuración de pines, inicialización de los puertos seriales y de interrupción para lectura de sensor
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(EV1, OUTPUT);
  Serial.begin(9600);
  max485.begin(9600);
  max485.setTimeout(1000);
  Timer1.initialize(500000);
  Timer1.attachInterrupt(lectSensor);
}

void lectSensor(){
  //Función que se activa por interrupción de Timer cada 500 ms
  //Lectura-Cálculo del sensor ultrasonido
  //Pulso de 10 microsegundos para emitir señal
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);
  //Lectura del tiempo que dura en 1 el pin echo
  tiempo = (pulseIn(echo, HIGH));
  //Calculo de distancia
  distanciac=tiempo/(int)58;
}

void loop() {
    //Lectura de datos
      if (max485.read()==direccionb){
        distanciaa=max485.parseInt();
        distanciab=max485.parseInt();
        estadoEB1=max485.parseInt();
        estadoNIB1=max485.parseInt();

      /*Correccion de dato en sensor ultrasonico (por perturbaciones en superficie del agua)
       *Si el dato es bueno se guarda en ambos espacios del arreglo se pone la bandera a 1
       *Se detecta que los errores son valores por debajo de 20 que están fuera del rango de lectura del sensor
```

```
     *este if sirve como una actualización del último dato bueno*/
    if (distanciac>20 && bandera==0){
      distanciacarray[0]=distanciac;
      distanciacarray[1]=distanciac;
      bandera=1;
    }
    //Si el dato es bueno lo guardo en el segundo espacio del arreglo
    else if(distanciac>20 && bandera==1){
      distanciacarray[1]=distanciac;
      bandera=0;
    }
    //Si el dato es un error utilizo el último valor bueno guardado en "memoria"
    else if(distanciac<20){
      distanciacarray[1]=distanciacarray[0];
    }

  //Lectura de estado de válvula que funciona como bandera y también se usa
para enviar
   estadoEV1=digitalRead(EV1);

  //Condicion de control
  //Si el nivel es bajo enciende la electroválvula para abastecerse
  //El relé de esta estación es activo en bajo
    if (distanciacarray[1]>=(int)90 && estadoEV1==1){
      digitalWrite(EV1, LOW);
    }
  //Si el nivel sube lo suficiente cierra electroválvula
    if (distanciacarray[1]<=(int)58 && estadoEV1==0 ){
      digitalWrite(EV1, HIGH);
    }

      //Conversion de datos de entero a string en base 10
      itoa(distanciaa,sdistanciaa,10);
      itoa(distanciab,sdistanciab,10);
      itoa(distanciacarray[1],sdistanciac,10);
      itoa(estadoEB1,sestadoEB1,10);
      itoa(estadoNIB1,sestadoNIB1,10);
      itoa(estadoEV1,sestadoEV1,10);
```

```
        //Envio de datos
        max485.println(direccionc);
        max485.println(sdistanciaa);
        max485.println(sdistanciab);
        max485.println(sdistanciac);
        max485.println(sestadoEB1);
        max485.println(sestadoNIB1);
        max485.println(sestadoEV1);

        //Comprobacion de lectura y escritura
        Serial.println(sdistanciaa);
        Serial.println(sdistanciab);
        Serial.println(sdistanciac);
        Serial.println(sestadoEB1);
        Serial.println(sestadoNIB1);
        Serial.println(sestadoEV1);
  }
}

/*Programación realizada para la estación D
 * Punto de riego automático temporizado
 * Granja La Luz de la Esperanza
 * Daniel Linares - Mayerly Narváez
 * Proyecto de grado - USCO
 */

#include "mbed.h"
//Constructor de los objetos para transmisión serial via pines Rx Tx y puerto USB
Serial max485(D1,D0);
Serial pc(USBTX, USBRX);
//Definición de variables
DigitalOut EV(D5);    //Válvulas de retorno y riego
DigitalOut EVR(D6);
DigitalOut led(LED1);   //Led indicador de abastecimiento
DigitalIn BtnRiego(D8);    //Botón de riego
//Strings a recibir con la trama de datos
char distanciaa[16], distanciab[16], distanciac[16], estadoEB1[16],
estadoNIB1[16], estadoEV1[16], buffer[32];
const char direccionc='c';    //Direcciones
```

```
const char direcciond='d';
int distanciad, estadoEV2, estadoEVR;    //Datos generados en la estación

/*Definición de la función con retorno de entero donde los argumentos
son el pin Echo y Trigger respectivamente*/

int ultrasonido(PinName Echo, PinName Trigger)
{
   int Correccion=3,Distancia;
   //Instancia del objeto ultra a la clase Timer
   Timer ultra;
   DigitalOut trigger(Trigger);
   DigitalIn echo(Echo);
   trigger = 1;
   ultra.reset();
   //Pulso disparador de 10 us
   wait_us(10.0);
   trigger = 0;
   //Se inicia el Timer cuando el pin Echo deje de estar en 0
   while(echo==0) {};
   ultra.start();
   //Se inicia el Timer cuando el pin Echo deje de estar en 0
   while (echo==1) {};
   ultra.stop();
   //Se lee el tiempo transcurrido el microsegundos y se divide en 58 para obtener
Distancia en cm
   //la corrección es el tiempo muerto en el cual se enciente y detiene el timer
   Distancia = (ultra.read_us()-Correccion)/(float)58;
   return Distancia;
}
//Función sin retorno que contiene la instrucción de control para abastecimiento
void controlestacione()
{
   //Condicional para activar electroválvula de retorno cuando nivel cae a un valor
bajo
   if (distanciad>=(int)136 && EV==1) {
      EV=0;
      led=0;
   }
```

```
    //Condicional para desactivar la electraválvula de retorno cuando alcance nivel máximo
    if (distanciad<=(int)86 && EV==0) {
        EV=1;
        led=1;
    }
}
//Función para el riego automático y temporizado
void riego()
{
    //Uso de la función strncmp para la comparación de variables tipo string
    //Se evaluan los caracteres hasta el digito que representa los minutos para así lograr riegos de 10 minutos
    if (strncmp(buffer,"07:0",4)== 0 || strncmp(buffer,"12:0",4)== 0 || strncmp(buffer,"17:0",4)== 0) {
        //Electrovalvulas de riego activadas
        EVR = 0;
    } else {
        //Electroválvulas de riego desactivadas
        EVR = 1;
    }
}


int main()
{
    //La función set_time configura el RTC a la hora (tiempo unix) ingresada en este caso a las 4 pm
    //La fecha ingresada no corresponde, pero es un dato innecesario
    set_time(64200);
    //Inicializa EV apagada
    EV=1;
    while(1) {
        //Inicialización de puertos seriales y velocidades de transmisión
        max485.baud(9600);
        pc.baud(9600);
        //lectura de dirección
        if (max485.getc()==direccionc) {
```

```
        //LECTURA DE DATOS
        max485.scanf("%s",distanciaa);
        max485.scanf("%s",distanciab);
        max485.scanf("%s",distanciac);
        max485.scanf("%s",estadoEB1);
        max485.scanf("%s",estadoNIB1);
        max485.scanf("%s",estadoEV1);

        //LECTURA DE SENSOR ULTRASONIDO
        distanciad=ultrasonido(D3,D4);//echo, trigger

        //INSTRUCCION DE CONTROL
        controlestacione();

        /*Se define la variable tipo time_t que aumentará su valor conforme pase el tiempo (segundos), la funión time con argumento nulo retorna
        el número de segundos hasta la fecha actual*/
        time_t segundos = time(NULL);
         /*La función strftime copia el formato dado(3er arg)de acuerdo a la estructura tm(4to arg)
        dentro del apuntador(1er arg) y limíta el tamaño(2do arg)*/
        //La función localtime retorna una estructura tm con los valores que corresponden a la hora su unico argumento es un valor tipo time_t
        strftime(buffer, 32, "%R\r\n", localtime(&segundos));

        //Comprobar hora
        pc.printf("%s\r\n",buffer);

        //INSTRUCCION DE RIEGO
        //Si el interruptor se encuentra activado ejecuta la función de riego, de lo contrario apaga las electroválvulas de riego
        if (BtnRiego.read() == 1) {
           riego();
        } else {
           EVR=1;
        }

        //LECTURA DE ESTADOS LOGICOS
        estadoEV2=EV.read();
```

```
            estadoEVR=EVR.read();

            //ENVÍO DE DATOS
            max485.printf("%c\r\n",direcciond);
            max485.printf("%s\r\n",distanciaa);
            max485.printf("%s\r\n",distanciab);
            max485.printf("%s\r\n",distanciac);
            max485.printf("%i\r\n",distanciad);
            max485.printf("%s\r\n",estadoEB1);
            max485.printf("%s\r\n",estadoNIB1);
            max485.printf("%s\r\n",estadoEV1);
            max485.printf("%i\r\n",estadoEV2);
            max485.printf("%i\r\n",estadoEVR);

            //COMPROBAR RECEPCION Y ENVIO
            pc.printf("%s\r\n",distanciaa);
            pc.printf("%s\r\n",distanciab);
            pc.printf("%s\r\n",distanciac);
            pc.printf("%i\r\n",distanciad);
            pc.printf("%s\r\n",estadoEB1);
            pc.printf("%s\r\n",estadoNIB1);
            pc.printf("%s\r\n",estadoEV1);
            pc.printf("%i\r\n",estadoEV2);
            pc.printf("%i\r\n",estadoEVR);

        }
    }
}
```

ANEXO B. Programación en estación central de monitoreo

```
"""Programación realizada para la Estacion R
   Granja la Luz de la Esperanza
   Daniel Linares - Mayerly Narváez
   Proyecto de grado - USCO"""

#! usr/bin/env python
# -*- coding: utf-8 -*-

#Se importan los módulos necesarios
import sys
import time
import serial
import os
from interfaz_grafica import * #Contiene la clase con los
métodos de los widgets de interfaz de monitoreo

#Lectura del puerto usado
puerto_usado = os.popen("ls /dev/ttyS*").read()
#Se borra el salto de linea para poder usarlo
puerto = puerto_usado[:-1]
#Instancia del objeto de la clase serial e inicialización del
puerto
PuertoSerie = serial.Serial(puerto,9600)

#Clase de la ventana tipo Dialogo
class InterfazMonitoreo(QtGui.QDialog):
#Método de inicialización, aquí se hace la conección de las
señales generadas el el Thread con los slots(widgets)
    def __init__(self, parent=None):
        super(InterfazMonitoreo,self).__init__(parent)
        self.ui=Ui_Dialog()
        self.ui.setupUi(self)
        self.hiloclase=HiloClase()
        self.hiloclase.start()
        self.connect(self.hiloclase,
QtCore.SIGNAL('INIVELA'), self.mostrarSensorProgress1)
        self.connect(self.hiloclase,
```

```
QtCore.SIGNAL('SNIVELA'), self.mostrarSensor1)
        self.connect(self.hiloclase,
QtCore.SIGNAL('INIVELB'), self.mostrarSensorProgress2)
        self.connect(self.hiloclase,
QtCore.SIGNAL('SNIVELB'), self.mostrarSensor2)
        self.connect(self.hiloclase,
QtCore.SIGNAL('INIVELC'), self.mostrarSensorProgress3)
        self.connect(self.hiloclase,
QtCore.SIGNAL('SNIVELC'), self.mostrarSensor3)
        self.connect(self.hiloclase,
QtCore.SIGNAL('INIVELD'), self.mostrarSensorProgress4)
        self.connect(self.hiloclase,
QtCore.SIGNAL('SNIVELD'), self.mostrarSensor4)
        self.connect(self.hiloclase,
QtCore.SIGNAL('SNIVELINTERRUPTORB'), self.mostrarEstadoNF1)
        self.connect(self.hiloclase,
QtCore.SIGNAL('SELECTROVALVULAD'), self.mostrarEstadoEV3)
        self.connect(self.hiloclase,
QtCore.SIGNAL('SELECTROVALVULAC'), self.mostrarEstadoEV4)
        self.connect(self.hiloclase,
QtCore.SIGNAL('SELECTRORIEGO'), self.mostrarEstadoRiego)
        self.connect(self.hiloclase,
QtCore.SIGNAL('SELECTROBOMBAB'), self.mostrarEstadoEB1)
        self.connect(self.hiloclase,
QtCore.SIGNAL('INIVELALERTA'), self.mostrarAlertaReservorio)

#Métodos o slots donde se visualizan las señales
    def mostrarSensor1(self, nla):
        self.ui.labelNivelSN1.setText(nla)

    def mostrarSensorProgress1(self, ada):
        self.ui.progressBarSN1.setValue(ada)

    def mostrarSensor2(self, nlb):
        self.ui.labelNivelSN2.setText(nlb)

    def mostrarSensorProgress2(self, adb):
        self.ui.progressBarSN2.setValue(adb)
```

```
    def mostrarSensor3(self, nlc):
        self.ui.labelNivelSN3.setText(nlc)

    def mostrarSensorProgress3(self, adc):
        self.ui.progressBarSN3.setValue(adc)

    def mostrarSensor4(self, nld):
        self.ui.labelNivelSN4.setText(nld)

    def mostrarSensorProgress4(self, add):
        self.ui.progressBarSN4.setValue(add)

    def mostrarEstadoEV3(self, ed):
        ied=int(ed)
        if (ied == 1):

self.ui.labelValvularetorno1.setText("Desactivada")
        elif (ied == 0):
            self.ui.labelValvularetorno1.setText("Activada")

    def mostrarEstadoEV4(self, ec):
        iec=int(ec)
        if (iec == 1):

self.ui.labelValvularetorno2.setText("Desactivada")
        elif (iec == 0):
            self.ui.labelValvularetorno2.setText("Activada")

    def mostrarEstadoEB1(self, eb):
        ieb=int(eb)
        if (ieb == 0):
            self.ui.labelbomba1.setText("Desactivada")
        elif (ieb == 1):
            self.ui.labelbomba1.setText("Activada")

    def mostrarEstadoNF1(self, nb):
        inb=int(nb)
        if (inb == 0):
            self.ui.labelAlertaSensor1.setText("Desactivado")
```

```
        elif (inb == 1):
            self.ui.labelAlertaSensor1.setText("Alerta:
Sensor de proteccion B ACTIVADO")

    def mostrarAlertaReservorio(self, ida):
        if (ida >= 170):
            self.ui.labelAlerta.setText("Alerta: Nivel de
agua en reservorio bajo. Encender electrobomba de
nacimiento")
        else:
            self.ui.labelAlerta.setText("Nivel de agua
adecuado en reservorio")

    def mostrarEstadoRiego(self, er):
        ier=int(er)
        if (ier == 1):
            self.ui.labelRiego.setText("Desactivado")
        elif (ier == 0):
            self.ui.labelRiego.setText("Activado")

#Debido al ciclo utilizado se crea una clase hilo (Thread)
por medio del objeto QThread
#Esta clase se inicializa y por medio de la función run se
obtienen las señales
class HiloClase(QtCore.QThread):

    def __init__(self, parent=None):
        super(HiloClase,self).__init__(parent)

    def run(self):
        while True:
#Se envía la dirección de inicialización a la estación A
            direccionrasp='r'
            PuertoSerie.write(direccionrasp)
#Se hace la lectura doble de datos para corregir algunos
errores
            d1 = PuertoSerie.readline()
            da1 = PuertoSerie.readline()
            db1 = PuertoSerie.readline()
```

```
                dc1 = PuertoSerie.readline()
                dd1 = PuertoSerie.readline()
                eb1 = PuertoSerie.readline()
                nb1 = PuertoSerie.readline()
                ec1 = PuertoSerie.readline()
                ed1 = PuertoSerie.readline()
                er1 = PuertoSerie.readline()
                d2 = PuertoSerie.readline()
                da2 = PuertoSerie.readline()
                db2 = PuertoSerie.readline()
                dc2 = PuertoSerie.readline()
                dd2 = PuertoSerie.readline()
                eb1 = PuertoSerie.readline()
                nb1 = PuertoSerie.readline()
                ec1 = PuertoSerie.readline()
                ed1 = PuertoSerie.readline()
                er1 = PuertoSerie.readline()

#Correccion de errores al no ser un digito por falta
sincronia en la trama o realmente llega un error en algun
dato
                if (da1.isdigit == True and da2.isdigit == True):
                    da1=da1
                    da2=da2
                if (db1.isdigit == True and db2.isdigit == True):
                    db1=db1
                    db2=db2
                if (dc1.isdigit == True and dc2.isdigit == True):
                    dc1=dc1
                    dc2=dc2
                if (dd1.isdigit == True and dd2.isdigit == True):
                    dd1=dd1
                    dd2=dd2

                if (eb1.isdigit):
                    eb=eb1
                if (nb1.isdigit):
                    nb=nb1
                if (ec1.isdigit):
```

```
                ec=ec1
            if (ed1.isdigit):
                ed=ed1
            if (er1.isdigit):
                er=er1
#Uso de la función int para llevar los valores a enteros,
poder operarlosy posteriormente llevarlos a las barras de
progreso de cada tanque
            ida1=int(da1)
            ida2=int(da2)
            idb1=int(db1)
            idb2=int(db2)
            idc1=int(dc1)
            idc2=int(dc2)
            idd1=int(dd1)
            idd2=int(dd2)

#Filtro de Correccion de errores por transmision
            if(ida1-ida2 > -5 and ida1-ida2 <5):
                ida=ida1
            if(idb1-idb2 > -5 and idb1-idb2 <5):
                idb=idb1
            if(idc1-idc2 > -5 and idc1-idc2 <5):
                idc=idc1
            if(idd1-idd2 > -5 and idd1-idd2 <5):
                idd=idd1

#Operaciones para convertir distncias a nivel(centimetros)
            ada=177-ida
            adb=171-idb
            adc=120-idc
            add=170-idd
#Operaciones para convertir nivel centimetros a nivel en
litros
#EstacionA
            nla=str((500*400*ada)/1000)+" L"
#EstacionB
            r2b=(0.07425*adb)+60.48
            cb=(adb*3.1416)/3
```

```
nlb=str(int(cb*(60.48*60.48+r2b*r2b+60.48*r2b)/1000))+" L"

#EstacionC
            r2c=0.06007*adc+49.34
            cc=(adc*3.1416)/3

nlc=str(int(6*cc*(49.34*49.34+r2c*r2c+49.34*r2c)/1000))+" L"
#EstacionD
            r2d=0.05453*add+55.07
            cd=(add*3.1416)/3

nld=str(int(4*cd*(55.07*55.07+r2d*r2d+55.07*r2d)/1000))+" L"
#Emisión de las señales para conectarlas a su respectivo slot
en la clase principal
            self.emit(QtCore.SIGNAL('INIVELA'), ada)
            self.emit(QtCore.SIGNAL('SNIVELA'), nla)
            self.emit(QtCore.SIGNAL('INIVELB'), adb)
            self.emit(QtCore.SIGNAL('SNIVELB'), nlb)
            self.emit(QtCore.SIGNAL('INIVELC'), adc)
            self.emit(QtCore.SIGNAL('SNIVELC'), nlc)
            self.emit(QtCore.SIGNAL('INIVELD'), add)
            self.emit(QtCore.SIGNAL('SNIVELD'), nld)
            self.emit(QtCore.SIGNAL('SNIVELINTERRUPTORB'),
nb)
            self.emit(QtCore.SIGNAL('SELECTROVALVULAD'), ed)
            self.emit(QtCore.SIGNAL('SELECTROVALVULAC'), ec)
            self.emit(QtCore.SIGNAL('SELECTRORIEGO'), er)
            self.emit(QtCore.SIGNAL('SELECTROBOMBAB'), eb)
            self.emit(QtCore.SIGNAL('INIVELALERTA'), ida)
#Sintaxis propia de python - Función principal
#Métodos necesarios para la ejecución y visualización de la
ventana inicial
if __name__ == "__main__":
    app = QtGui.QApplication(sys.argv)
    myapp = InterfazMonitoreo()
    myapp.show()
    sys.exit(app.exec_())
```

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file
'interfaz_grafica.ui'
#
# Created: Tue Oct 10 16:19:14 2017
#      by: PyQt4 UI code generator 4.11.2
#
# WARNING! All changes made in this file will be lost!

from PyQt4 import QtCore, QtGui

try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    def _fromUtf8(s):
        return s

try:
    _encoding = QtGui.QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text,
disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text,
disambig)

class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName(_fromUtf8("Dialog"))
        Dialog.resize(1478, 999)
        Dialog.setStyleSheet(_fromUtf8("#Dialog{\n"
"        background-
image:url(/home/pi/Documents/graficatotal2.png);\n"
"    }QProgressBar {border: 1px solid grey; border-
radius:4px; padding:0pxn;background: rgb(255, 255,
255);width: 15px; width: 15px} QProgressBar::chunk {border:
```

```
1px solid grey;background: QLinearGradient(spread:pad, x1:0,
y1:1,stop:0 #006bb3,stop:0.35 #0099ff , stop:0.70 #4db8ff ,
stop:1 #e6f5ff);}\n"
"\n"
"\n"
"#labelInterfaz{\n"
"        font-size:20px;\n"
"        font-weight: 900;\n"
"        font-family: \'Arial\';\n"
"        color:#000080;\n"
"    }\n"
"#labelFinca{\n"
"        font-size:18px;\n"
"        font-weight: 900;\n"
"        font-family: \'Arial\';\n"
"        color:#000080;\n"
"    }\n"
"#labelAlerta{\n"
"        font-size:14px;\n"
"        font-weight: 600;\n"
"        font-family: \'Arial\';\n"
"        color:#FF0000;\n"
"    }\n"
"#labelAlertaSensor2{\n"
"        font-size:14px;\n"
"        font-weight: 600;\n"
"        font-family: \'Arial\';\n"
"        color:#FF0000;\n"
"    }\n"
"#labelAlertaSensor1{\n"
"        font-size:14px;\n"
"        font-weight: 600;\n"
"        font-family: \'Arial\';\n"
"        color:#FF0000;\n"
"    }\n"
"QLabel{\n"
"        font-weight: 100;\n"
"        font-family: \'Arial\';\n"
"        font-size:16px;\n"
```

```
"         font-weight: 700;\n"
"         color: #000000;\n"
"     }\n"
"\n"
""))
        self.labelInterfaz = QtGui.QLabel(Dialog)
        self.labelInterfaz.setGeometry(QtCore.QRect(220, 10,
991, 20))

self.labelInterfaz.setAlignment(QtCore.Qt.AlignCenter)

self.labelInterfaz.setObjectName(_fromUtf8("labelInterfaz"))
        self.progressBarSN2 = QtGui.QProgressBar(Dialog)
        self.progressBarSN2.setGeometry(QtCore.QRect(60, 350,
191, 261))
        self.progressBarSN2.setMaximum(171)
        self.progressBarSN2.setProperty("value", 50)

self.progressBarSN2.setOrientation(QtCore.Qt.Vertical)
        self.progressBarSN2.setFormat(_fromUtf8(""))

self.progressBarSN2.setObjectName(_fromUtf8("progressBarSN2")
)
        self.progressBarSN3 = QtGui.QProgressBar(Dialog)
        self.progressBarSN3.setGeometry(QtCore.QRect(580,
350, 191, 261))
        self.progressBarSN3.setMaximum(120)
        self.progressBarSN3.setProperty("value", 50)

self.progressBarSN3.setOrientation(QtCore.Qt.Vertical)
        self.progressBarSN3.setFormat(_fromUtf8(""))

self.progressBarSN3.setObjectName(_fromUtf8("progressBarSN3")
)
        self.progressBarSN1 = QtGui.QProgressBar(Dialog)
        self.progressBarSN1.setGeometry(QtCore.QRect(1000,
280, 281, 381))
        self.progressBarSN1.setStyleSheet(_fromUtf8(""))
        self.progressBarSN1.setMaximum(177)
```

```
        self.progressBarSN1.setProperty("value", 50)

self.progressBarSN1.setOrientation(QtCore.Qt.Vertical)

self.progressBarSN1.setTextDirection(QtGui.QProgressBar.TopTo
Bottom)
        self.progressBarSN1.setFormat(_fromUtf8(""))

self.progressBarSN1.setObjectName(_fromUtf8("progressBarSN1")
)
        self.progressBarSN4 = QtGui.QProgressBar(Dialog)
        self.progressBarSN4.setGeometry(QtCore.QRect(320,
350, 191, 261))
        self.progressBarSN4.setMaximum(160)
        self.progressBarSN4.setProperty("value", 50)

self.progressBarSN4.setOrientation(QtCore.Qt.Vertical)

self.progressBarSN4.setTextDirection(QtGui.QProgressBar.TopTo
Bottom)
        self.progressBarSN4.setFormat(_fromUtf8(""))

self.progressBarSN4.setObjectName(_fromUtf8("progressBarSN4")
)
        self.labelFinca = QtGui.QLabel(Dialog)
        self.labelFinca.setGeometry(QtCore.QRect(520, 40,
381, 21))
        self.labelFinca.setAlignment(QtCore.Qt.AlignCenter)

self.labelFinca.setObjectName(_fromUtf8("labelFinca"))
        self.labelNivelSN1 = QtGui.QLabel(Dialog)
        self.labelNivelSN1.setGeometry(QtCore.QRect(1080,
720, 131, 31))
        self.labelNivelSN1.setText(_fromUtf8(""))

self.labelNivelSN1.setAlignment(QtCore.Qt.AlignCenter)

self.labelNivelSN1.setObjectName(_fromUtf8("labelNivelSN1"))
        self.labelNivelSN2 = QtGui.QLabel(Dialog)
```

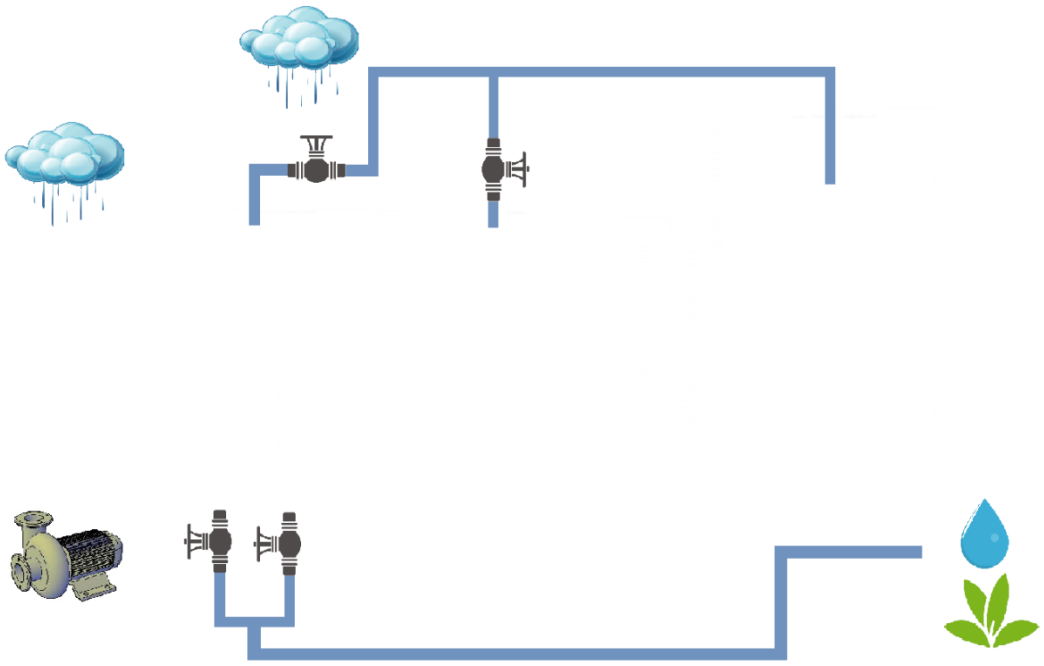
```
        self.labelNivelSN2.setGeometry(QtCore.QRect(90, 690,
131, 31))
        self.labelNivelSN2.setText(_fromUtf8(""))

self.labelNivelSN2.setAlignment(QtCore.Qt.AlignCenter)

self.labelNivelSN2.setObjectName(_fromUtf8("labelNivelSN2"))
        self.labelNivelSN3 = QtGui.QLabel(Dialog)
        self.labelNivelSN3.setGeometry(QtCore.QRect(610, 690,
131, 31))
        self.labelNivelSN3.setText(_fromUtf8(""))

self.labelNivelSN3.setAlignment(QtCore.Qt.AlignCenter)

self.labelNivelSN3.setObjectName(_fromUtf8("labelNivelSN3"))
        self.labelNivelSN4 = QtGui.QLabel(Dialog)
        self.labelNivelSN4.setGeometry(QtCore.QRect(350, 690,
131, 31))
        self.labelNivelSN4.setText(_fromUtf8(""))

self.labelNivelSN4.setAlignment(QtCore.Qt.AlignCenter)

self.labelNivelSN4.setObjectName(_fromUtf8("labelNivelSN4"))
        self.labelValvularetorno1 = QtGui.QLabel(Dialog)

self.labelValvularetorno1.setGeometry(QtCore.QRect(400, 220,
131, 31))
        self.labelValvularetorno1.setText(_fromUtf8(""))

self.labelValvularetorno1.setObjectName(_fromUtf8("labelValvu
laretorno1"))
        self.labelValvularetorno2 = QtGui.QLabel(Dialog)

self.labelValvularetorno2.setGeometry(QtCore.QRect(760, 280,
131, 31))
        self.labelValvularetorno2.setText(_fromUtf8(""))

self.labelValvularetorno2.setObjectName(_fromUtf8("labelValvu
laretorno2"))
```

```
        self.labelvalvuladeriego = QtGui.QLabel(Dialog)

self.labelvalvuladeriego.setGeometry(QtCore.QRect(470, 760,
131, 31))
        self.labelvalvuladeriego.setText(_fromUtf8(""))

self.labelvalvuladeriego.setObjectName(_fromUtf8("labelvalvul
aderiego"))
        self.labelB1 = QtGui.QLabel(Dialog)
        self.labelB1.setGeometry(QtCore.QRect(70, 880, 171,
21))
        self.labelB1.setAlignment(QtCore.Qt.AlignCenter)
        self.labelB1.setObjectName(_fromUtf8("labelB1"))
        self.labelbomba1 = QtGui.QLabel(Dialog)
        self.labelbomba1.setGeometry(QtCore.QRect(90, 930,
131, 31))
        self.labelbomba1.setText(_fromUtf8(""))
        self.labelbomba1.setAlignment(QtCore.Qt.AlignCenter)

self.labelbomba1.setObjectName(_fromUtf8("labelbomba1"))
        self.labelr1 = QtGui.QLabel(Dialog)
        self.labelr1.setGeometry(QtCore.QRect(1290, 680, 81,
21))
        self.labelr1.setAlignment(QtCore.Qt.AlignCenter)
        self.labelr1.setObjectName(_fromUtf8("labelr1"))
        self.labelRiego = QtGui.QLabel(Dialog)
        self.labelRiego.setGeometry(QtCore.QRect(1260, 920,
131, 31))
        self.labelRiego.setText(_fromUtf8(""))
        self.labelRiego.setAlignment(QtCore.Qt.AlignCenter)

self.labelRiego.setObjectName(_fromUtf8("labelRiego"))
        self.labelAlerta = QtGui.QLabel(Dialog)
        self.labelAlerta.setGeometry(QtCore.QRect(1050, 90,
411, 41))
        self.labelAlerta.setText(_fromUtf8(""))
        self.labelAlerta.setAlignment(QtCore.Qt.AlignCenter)

self.labelAlerta.setObjectName(_fromUtf8("labelAlerta"))
```

```
        self.labelAlertaSensor1 = QtGui.QLabel(Dialog)
        self.labelAlertaSensor1.setGeometry(QtCore.QRect(590,
90, 411, 41))
        self.labelAlertaSensor1.setText(_fromUtf8(""))

self.labelAlertaSensor1.setAlignment(QtCore.Qt.AlignCenter)

self.labelAlertaSensor1.setObjectName(_fromUtf8("labelAlertaS
ensor1"))
        self.LaberTanqueHuertrta = QtGui.QLabel(Dialog)
        self.LaberTanqueHuertrta.setGeometry(QtCore.QRect(90,
630, 131, 31))

self.LaberTanqueHuertrta.setAlignment(QtCore.Qt.AlignCenter)

self.LaberTanqueHuertrta.setObjectName(_fromUtf8("LaberTanque
Huertrta"))
        self.LaberTanqueRiego = QtGui.QLabel(Dialog)
        self.LaberTanqueRiego.setGeometry(QtCore.QRect(300,
630, 241, 31))

self.LaberTanqueRiego.setAlignment(QtCore.Qt.AlignCenter)

self.LaberTanqueRiego.setObjectName(_fromUtf8("LaberTanqueRie
go"))
        self.LaberTanqueReservorio = QtGui.QLabel(Dialog)

self.LaberTanqueReservorio.setGeometry(QtCore.QRect(1080,
670, 131, 31))

self.LaberTanqueReservorio.setAlignment(QtCore.Qt.AlignCenter
)

self.LaberTanqueReservorio.setObjectName(_fromUtf8("LaberTanq
ueReservorio"))
        self.LaberTanqueRiego2 = QtGui.QLabel(Dialog)
        self.LaberTanqueRiego2.setGeometry(QtCore.QRect(570,
630, 221, 31))
```

```
self.LaberTanqueRiego2.setAlignment(QtCore.Qt.AlignCenter)

self.LaberTanqueRiego2.setObjectName(_fromUtf8("LaberTanqueRi
ego2"))

        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi(self, Dialog):
        Dialog.setWindowTitle(_translate("Dialog", "Dialog",
None))
        self.labelInterfaz.setText(_translate("Dialog",
"INTERFAZ MONITOREO SISTEMA DE ABASTECIMIENTO Y BOMBEO DE
AGUA LLUVIA", None))
        self.labelFinca.setText(_translate("Dialog", "FINCA
LA LUZ DE LA ESPERANZA", None))
        self.labelB1.setText(_translate("Dialog",
"ELECTROBOMBA", None))
        self.labelr1.setText(_translate("Dialog", "RIEGO",
None))
        self.LaberTanqueHuertrta.setText(_translate("Dialog",
"Tanque Huerta", None))
        self.LaberTanqueRiego.setText(_translate("Dialog",
"Tanque Riego Temporizado", None))

self.LaberTanqueReservorio.setText(_translate("Dialog",
"Reservorio", None))
        self.LaberTanqueRiego2.setText(_translate("Dialog",
"Tanque Riego Manual", None))
```

ANEXO C. Imagen de fondo utilizada para Interfaz Gráfica



ANEXO D. Hoja de datos MAX485

19-0132; Rev 8; 10/03

MAXIM

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

General Description

The MAX481, MAX483, MAX485, MAX487–MAX491, and MAX1487 are low-power transceivers for RS-485 and RS-422 communication. Each part contains one driver and one receiver. The MAX483, MAX487, MAX488, and MAX489 feature reduced slew-rate drivers that minimize EMI and reduce reflections caused by improperly terminated cables, thus allowing error-free data transmission up to 250kbps. The driver slew rates of the MAX481, MAX485, MAX490, MAX491, and MAX1487 are not limited, allowing them to transmit up to 2.5Mbps.

These transceivers draw between 120µA and 500µA of supply current when unloaded or fully loaded with disabled drivers. Additionally, the MAX481, MAX483, and MAX487 have a low-current shutdown mode in which they consume only 0.1µA. All parts operate from a single 5V supply.

Drivers are short-circuit current limited and are protected against excessive power dissipation by thermal shutdown circuitry that places the driver outputs into a high-impedance state. The receiver input has a fail-safe feature that guarantees a logic-high output if the input is open circuit.

The MAX487 and MAX1487 feature quarter-unit-load receiver input impedance, allowing up to 128 MAX487/MAX1487 transceivers on the bus. Full-duplex communications are obtained using the MAX488–MAX491, while the MAX481, MAX483, MAX485, MAX487, and MAX1487 are designed for half-duplex applications.

Applications

Low-Power RS-485 Transceivers

Low-Power RS-422 Transceivers

Level Translators

Transceivers for EMI-Sensitive Applications

Industrial-Control Local Area Networks

Next Generation Device Features

- **For Fault-Tolerant Applications**
  MAX3430: ±80V Fault-Protected, Fail-Safe, 1/4 Unit Load, +3.3V, RS-485 Transceiver
  MAX3440E–MAX3444E: ±15kV ESD-Protected, ±60V Fault-Protected, 10Mbps, Fail-Safe, RS-485/J1708 Transceivers
- **For Space-Constrained Applications**
  MAX3460–MAX3464: +5V, Fail-Safe, 20Mbps, Profibus RS-485/RS-422 Transceivers
  MAX3362: +3.3V, High-Speed, RS-485/RS-422 Transceiver in a SOT23 Package
  MAX3280E–MAX3284E: ±15kV ESD-Protected, 52Mbps, +3V to +5.5V, SOT23, RS-485/RS-422, True Fail-Safe Receivers
  MAX3293/MAX3294/MAX3295: 20Mbps, +3.3V, SOT23, RS-855/RS-422 Transmitters
- **For Multiple Transceiver Applications**
  MAX3030E–MAX3033E: ±15kV ESD-Protected, +3.3V, Quad RS-422 Transmitters
- **For Fail-Safe Applications**
  MAX3080–MAX3089: Fail-Safe, High-Speed (10Mbps), Slew-Rate-Limited RS-485/RS-422 Transceivers
- **For Low-Voltage Applications**
  MAX3483E/MAX3485E/MAX3486E/MAX3488E/MAX3490E/MAX3491E: +3.3V Powered, ±15kV ESD-Protected, 12Mbps, Slew-Rate-Limited, True RS-485/RS-422 Transceivers

Ordering Information appears at end of data sheet.

Selection Table

PART NUMBER	HALF/FULL DUPLEX	DATA RATE (Mbps)	SLEW-RATE LIMITED	LOW-POWER SHUTDOWN	RECEIVER/ DRIVER ENABLE	QUIESCENT CURRENT (µA)	NUMBER OF TRANSMITTERS ON BUS	PIN COUNT
MAX481	Half	2.5	No	Yes	Yes	300	32	8
MAX483	Half	0.25	Yes	Yes	Yes	120	32	8
MAX485	Half	2.5	No	No	Yes	300	32	8
MAX487	Half	0.25	Yes	Yes	Yes	120	128	8
MAX488	Full	0.25	Yes	No	No	120	32	8
MAX489	Full	0.25	Yes	No	Yes	120	32	14
MAX490	Full	2.5	No	No	No	300	32	8
MAX491	Full	2.5	No	No	Yes	300	32	14
MAX1487	Half	2.5	No	No	Yes	230	128	8

ANEXO E. Hoja de datos Microcontrolador Kinetis K64F

NXP Semiconductors
Data Sheet: Technical Data

K64P144M120SF5
Rev. 7, 11/2016

Kinetis K64F Sub-Family Data Sheet

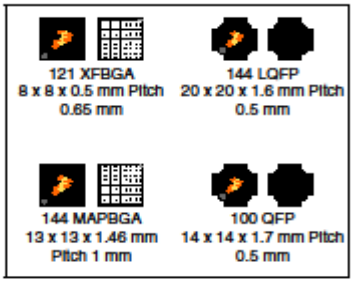
120 MHz ARM® Cortex®-M4-based Microcontroller with FPU

MK64FN1M0Vxx12
MK64FX512Vxx12

The K64 product family members are optimized for cost-sensitive applications requiring low-power, USB/Ethernet connectivity, and up to 256 KB of embedded SRAM. These devices share the comprehensive enablement and scalability of the Kinetis family.

This product offers:

- Run power consumption down to 250 µA/MHz. Static power consumption down to 5.8 µA with full state retention and 5 µs wakeup. Lowest Static mode down to 339 nA
- USB LS/FS OTG 2.0 with embedded 3.3 V, 120 mA LDO Vreg, with USB device crystal-less operation
- 10/100 Mbit/s Ethernet MAC with MII and RMII interfaces



Performance
- Up to 120 MHz ARM® Cortex®-M4 core with DSP instructions and floating point unit

Memories and memory interfaces
- Up to 1 MB program flash memory and 256 KB RAM
- Upto 128 KB FlexNVM and 4 KB FlexRAM on devices with FlexMemory
- FlexBus external bus interface

System peripherals
- Multiple low-power modes, low-leakage wake-up unit
- Memory protection unit with multi-master protection
- 16-channel DMA controller
- External watchdog monitor and software watchdog

Security and integrity modules
- Hardware CRC module
- Hardware random-number generator
- Hardware encryption supporting DES, 3DES, AES, MD5, SHA-1, and SHA-256 algorithms
- 128-bit unique identification (ID) number per chip

Analog modules
- Two 16-bit SAR ADCs
- Two 12-bit DACs
- Three analog comparators (CMP)
- Voltage reference

Communication interfaces
- Ethernet controller with MII and RMII interface
- USB full-/low-speed On-the-Go controller
- Controller Area Network (CAN) module
- Three SPI modules
- Three I2C modules. Support for up to 1 Mbit/s
- Six UART modules
- Secure Digital Host Controller (SDHC)
- I2S module

Timers
- Two 8-channel Flex-Timers (PWM/Motor control)
- Two 2-channel FlexTimers (PWM/Quad decoder)
- IEEE 1588 timers
- 32-bit PITs and 16-bit low-power timers
- Real-time clock
- Programmable delay block

Clocks
- 3 to 32 MHz and 32 kHz crystal oscillator
- PLL, FLL, and multiple internal oscillators
- 48 MHz Internal Reference Clock (IRC48M)

Operating Characteristics
- Voltage range: 1.71 to 3.6 V
- Flash write voltage range: 1.71 to 3.6 V
- Temperature range (ambient): –40 to 105°C

ANEXO F. Hoja de datos Microcontrolador Atmega328P

8-bit AVR Microcontrollers

ATmega328/P

DATASHEET COMPLETE

Introduction

The Atmel® picoPower® ATmega328/P is a low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega328/P achieves throughputs close to 1MIPS per MHz. This empowers system designer to optimize the device for power consumption versus processing speed.

Feature

High Performance, Low Power Atmel®AVR® 8-Bit Microcontroller Family

- Advanced RISC Architecture
  - 131 Powerful Instructions
  - Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 32KBytes of In-System Self-Programmable Flash program Memory
  - 1KBytes EEPROM
  - 2KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data Retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Atmel® QTouch® Library Support
  - Capacitive Touch Buttons, Sliders and Wheels
  - QTouch and QMatrix® Acquisition
  - Up to 64 sense channels

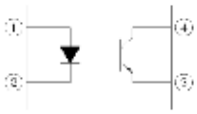
ANEXO G. Hoja de datos optoacoplador 317C

EVERLIGHT

DATASHEET

4 PIN DIP PHOTOTRANSISTOR PHOTOCOUPLER
EL817 Series

Schematic



Features:

- Current transfer ratio
(CTR: 50~600% at I_F = 5mA, V_{CE} = 5V)
- High isolation voltage between input
and output (Viso = 5000Vrms)
- Creepage distance > 7.62mm
- Operating temperature up to +110°C
- Compact small outline package
- The product itself will remain within RoHS compliant version
- Compliance with EU REACH
- UL and cUL approved(No.E214129)
- VDE approved (No. 132249)
- SEMKO approved
- NEMKO approved
- DEMKO approved
- FIMKO approved
- CQC approved

Pin Configuration
1. Anode
2. Cathode
3. Emitter
4. Collector

Description

The EL817 series of devices each consist of an infrared emitting diodes,
optically coupled to a phototransistor detector.
They are packaged in a 4-pin DIP package and available in wide-lead spacing and SMD option.

Applications

- Programmable controllers
- System appliances, measuring instruments
- Telecommunication equipments
- Home appliances, such as fan heaters, etc.
- Signal transmission between circuits of different potentials and impedances

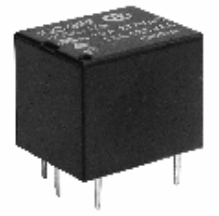
ANEXO H. Hoja de datos relé JQC-3FF

JQC-3FF SUBMINIATURE HIGH POWER RELAY

File No.:R50034671

File No.:E133481

File No.:CQC02001001953



Features

- Extremely low cost
- SPST-NO & SPDT configuration
- Subminiature, standard PCB layout
- Sealed IP67 and Flux proof types available

CONTACT DATA

Contact Arrangement	1A	1C
Initial Contact Resistance Max.		100mΩ (at 1A 6VDC)
Contact Material		Silver Alloy
Contact Rating (Res. Load)	10A 277VAC	7A 250VDC 10A 277VAC
Max. switching voltage		277VAC/30VDC
Max. switching current	15A	10A
Max. switching power		2770VA 210W
Mechanical life		1 x 10^7OPS
Electrical life		1 x 10^5OPS

CHARACTERISTICS

Initial Insulation Resistance		100MΩ ,500VDC
Dielectric Strength	Between coil and contacts	1500VAC, 1min
	Between open contacts	750VAC, 1min
Operate time (at nomi. Volt.)		Max. 10ms
Release time (at nomi. Volt.)		Max. 5ms
Temperature rise (at nomi. Volt.)		Max. 60℃
Shock Resistance	Functional	98 m/s²(10g)
	Destructive	980 m/s²(100g)
Vibration Resistance		1.5mm, 10 to 55Hz
Humidity		35% to 85%RH
Ambient temperature		-40℃ to +85℃
Termination		PCB
Unit weight		Approx. 10g
Construction		Sealed IP67 & Flux proof

COIL

Coil power	0.36W*48VDC : 0.51W*

COIL DATA

Nominal Voltage VDC	Pick-up Voltage VDC	Drop-out Voltage VDC	Max. allowable Voltage VDC(at 25℃)	Coil Resistance Ω
5	3.80	0.5	6.5	70 ± 10%
6	4.50	0.6	7.8	100 ± 10%
9	6.80	0.9	11.7	225 ± 10%
12	9.00	1.2	15.6	400 ± 10%
18	13.5	1.8	23.4	900 ± 10%
24	18.0	2.4	31.2	1600 ± 10%
48	36.0	4.8	62.4	4500 ± 10%

SAFETY APPROVAL RATINGS

UL	1 Form C	10A 277 VAC 10A 120VAC 1/2 HP 125/250VAC
	1 Form A	10A 277VAC TV-5 120VAC 15A 125VAC 120VAC 125VAC 1/2hp,125VAC
TÜV	1 Form C	8A 250VAC 12A 125VAC cos phi=1 5A 250VAC cos phi=1
	1 Form A	10A 277VAC 12A 125VAC cos phi=1 5A 250VAC cos phi=1

ANEXO I. Hoja de datos relé SLA-05VDC-SLC

SONGLE RELAY

松乐继电器 SONGLE RELAY	RELAY ISO9002	SLA

1. MAIN FEATURES

☐ Up to 30A switching in SPST and 20A switching in spot arrangements.

☐ Available as an open-frame relay, with a snap-on dust cover or with an immersion cleanable, plastic sealed case.

2. APPLICATIONS

☐ Used for power switching, Electrical Heater, ventilator, Air conditioning, Refrigerating, Automobile and House-hold Appliance.

3. ORDERING INFORMATION

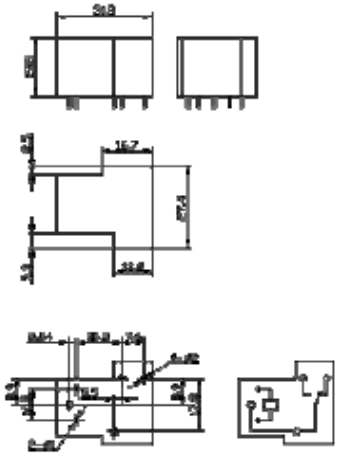
SLA	XX VDC	S	L	C
Model of relay	Nominal coil voltage	Structure	Coil sensitivity	Contact form
SLA	05☐06☐09☐12☐18☐ 24☐48☐110VDC	S:Sealed type	L:0.93W	A:1 form A
		F:Flux free type	D:Special	B:1 form B
				C:1 form C

4. RATING

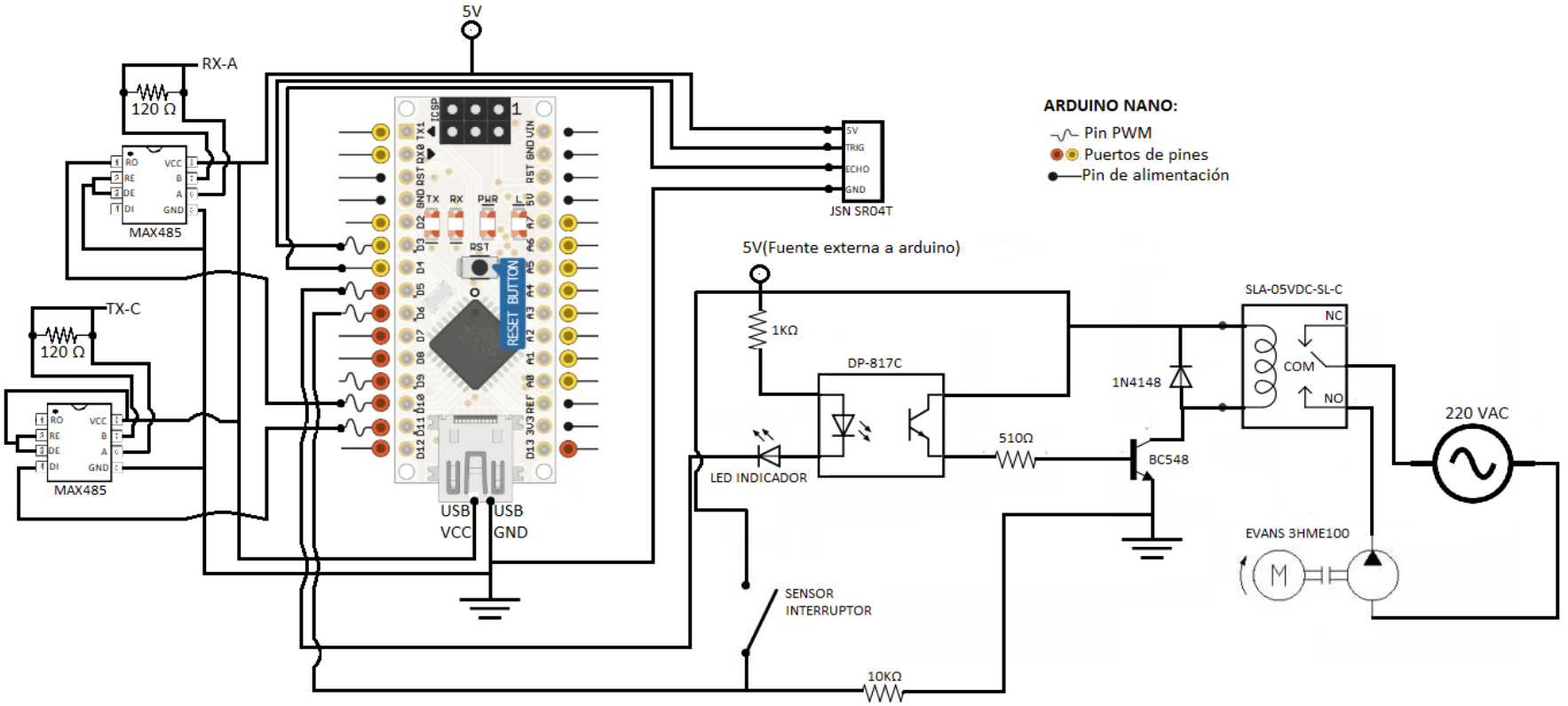
20A/28VDC 250VAC 30A/30VDC 250VAC

CCC	FILE NUMBER:CQC03001003728	30A/250VDC
UL/CUL	FILE NUMBER: E179944	NO:20A/240VAC 28VDC NC:10A/240VAC 28VDC
TUV	FILE NUMBER: R50056114	NO:30A/250VAC 30VDC NC:20A/250VAC 30VDC

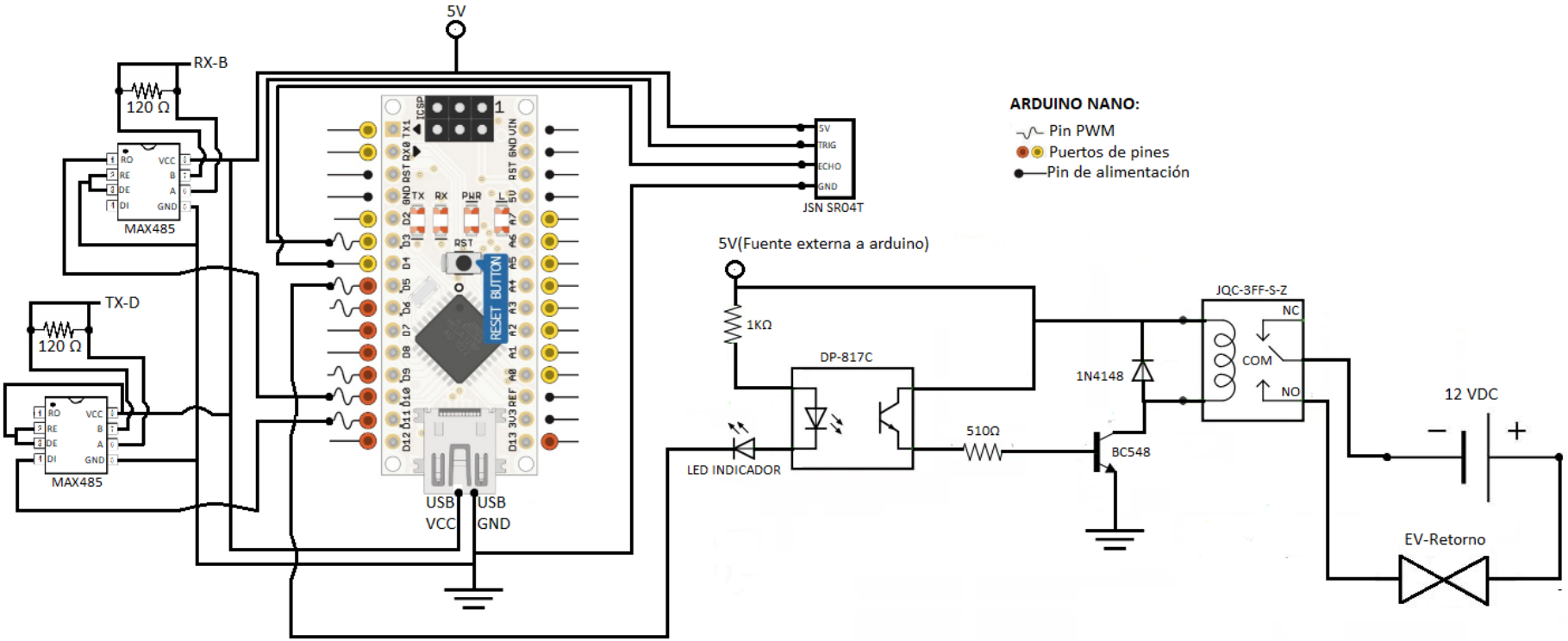
5. DIMENSION(unit:mm) DRILLING(unit:mm) WIRING DIAGRAM



ANEXO J. Representación gráfica del circuito en estación B



ANEXO K. Representación gráfica del circuito en estación C



ANEXO L. Representación gráfica del circuito en estación D