

TRABAJO FINAL



Proyecto Fin de Carrera

Diseño de un sistema de riego para optimizar el
recurso hídrico

Matías Leandro Ferraro

Director de tesis: Sergio Hilario Gallina

Año: 2015

Agradecimientos

Agradezco a mis padres, hermanos y abuelos su apoyo moral y económico para concretar lo que era nada más que un sueño.

Agradezco a los docentes de la Facultad de Tecnología y Ciencias Aplicadas por mi formación académica, pero especialmente al Ing. Sergio Hilario Gallina por darme la posibilidad de trabajar junto a él en innumerables proyectos extracurriculares que permitieron formarme en mi profesión, gracias a su apoyo y enseñanzas pude iniciarme en la docencia trabajando junto a él como ayudante alumno en la catedra de sistemas lógicos durante cinco años.

Para la realización de este trabajo final conté con la colaboración del Ing. Luis Nieva para el diseño y construcción de los gabinetes, poniendo a mi disposición su arte y sabiduría, abriéndome las puertas de su taller de forma desinteresada.

A mi padre...

Sus enseñanzas...

¡PIU AVANTI!

No te des por vencido, ni aún vencido,
no te sientas esclavo, ni aún esclavo;
trémulos de pavor, piénsate bravo,
y acomete feroz, ya mal herido.

Ten el tesón del clavo enmohecido
que ya viejo y ruin, vuelve a ser clavo;
no la cobarde estupidez del pavo
que amaina su plumaje al primer ruido.

Procede como Dios que nunca llora;
o como Lucifer, que nunca reza;
o como el robledal, cuya grandeza
necesita del agua, y no la implora...
Que muerda y vocifere vengadora,
ya rodando en el polvo, tu cabeza!

ALMA FUERTE

UNIVERSIDAD NACIONAL DE CATAMARCA



1-INDICE

Diseño de un sistema de riego para optimizar el recurso hídrico

Matías Leandro Ferraro

Año: 2015

Título: Ingeniero Electrónico
Director de tesis: Sergio Hilario Gallina
Departamento: Electrónica

Contenido

1-INDICE	4
2- MARCO TEORICO.....	8
2.1- INTRODUCCION.....	9
2.2-SISTEMAS DE RIEGO AUTOMATIZADOS.....	11
2.3-SISTEMAS DE RIEGO AUTOMATIZADOS Y TELECONTROLADOS.....	12
2.4-NIVELES DE GESTION DE PROCESOS.....	13
2.5-CARACTERISTICAS A EXIGIR A UN SISTEMA TELECONTROLADO AUTOMATICO.....	13
2.6-ELEMENTOS QUE COMPONEN EL SISTEMA DE RIEGO	15
2.6.1-Centro de control de la instalación.....	15
2.6.2-Estaciones concentradoras	15
2.6.3-Estaciones remotas (RTUs)	15
2.6.4-Actuadores	15
2.6.5-Transductores	15
2.7-REDES DE DISTRIBUCION Y SISTEMAS DE ALMACENAMIENTO	19
2.8-FORMAS HABITUALES DE RIEGO.....	19
2.9-SISTEMAS DE RIEGO LOCALIZADO	20
2.10- RIEGO POR GOTEO.....	22
2.10.1-¿Por qué usar riego por goteo?	22
2.10.2-Ventajas del uso de riego por goteo.....	22
2.10.3-Emisores de riego por goteo	23
2.11-OBJETIVO GENERAL.....	24
2.12-ALCANCE	24
3. MEMORIA DESCRIPTIVA.....	26
3.1-PRIMER PROTOTIPO.....	27
3.2-SEGUNDO PROTOTIPO	27
3.3-DESCRIPCION TECNICA DEL SISTEMA DE RIEGO IMPLEMENTADO.....	27
3.3.1-Centro de control de la instalación.....	28
3.3.2-Estacion concentradora	28
3.3.3-Estacion remota (RTU)	28
3.4-HARDWARE DEL SISTEMA DE RIEGO IMPLEMENTADO	33
3.4.1-CPU.....	35
3.4.2-Entradas analógicas	35
3.4.3-Pulsadores.....	39

3.4.4-Reloj de tiempo real.....	41
3.4.5-Puertos de comunicaciones	42
3.4.6-Expansiones	43
3.4.7-Display de LCD.....	44
3.4.8-Leds	45
3.4.9-Salidas digitales.....	47
3.4.10-Teclado analógico	48
3.4.11-Memoria EEPROM	53
3.4.12-Fuente de alimentación	54
3.5-CIRCUITOS IMPRESOS	55
3.6-FIRMWARE DEL SISTEMA DE RIEGO IMPLEMENTADO	56
3.6.1 ¿Qué es un RTOS?	56
3.6.2-Estructura de los comandos enviados sobre el bus RS-485.....	56
3.6.3-Descripcion de las máquinas de estado implementadas.....	58
3.6.4-Maquina de estado utilizada para la comunicación según el protocolo RS-485	58
3.6.5-Maquina de estado utilizada en el teclado analógico.....	64
3.6.6-Maquina de estado para elaborar los reportes de actividades del sistema	67
3.6.7-Maquina de estado para la comunicación bluetooth de la estación concentradora	74
3.6.8-Maquina de estado de la aplicación en las estaciones remotas.....	75
3.6.9- Maquina de estado utilizada para implementar las pantallas de información de salida del sistema en la estación concentradora	78
3.6.10-Funciones en lenguaje C empleadas en el sistema.....	80
3.6.11-Función utilizada para controlar las bombas de agua	80
3.6.12-Función utilizada para armar la trama a transmitir	82
3.6.13-Función empleada para decodificar la trama recibida	85
3.6.14-Función utilizada para la lectura del reloj de tiempo real	87
3.6.15-Funciones para leer y guardar datos en la memoria EEPROM	88
3.6.16-Función de interrupción externa y configuraciones necesarias para generar un puerto de comunicaciones serie por software	90
4-SEÑALES DE VISUALIZACION Y MEDIDAS DE PROTECCION DEL SISTEMA ANTE FALLOS.....	95
5-ASPECTOS ECONOMICOS.....	96
6-ENSAYOS	97
7-CONCLUSIONES Y MEJORAS A FUTURO.....	99
8- REFERENCIAS.....	100
ANEXO I: CIRCUITOS ESQUEMATICOS.....	102
ANEXO II: CIRCUITOS IMPRESOS	118

ANEXO III: CIRCUITOS ELECTRICOS.....	128
ANEXO IV: MANUAL DE INSTALACION RAPIDA.....	132
ANEXO V: CODIGO FUENTE	141
ANEXO VI: FOTOGRAFIAS	201
ANEXO VII: PREMIO GANADO POR EL SISTEMA DE RIEGO	207

UNIVERSIDAD NACIONAL DE CATAMARCA



2- MARCO TEORICO

Diseño de un sistema de riego para optimizar el recurso hídrico

Matías Leandro Ferraro

Año: 2015

Título: Ingeniero Electrónico
Director de tesis: Sergio Hilario Gallina
Departamento: Electrónica

2.1- INTRODUCCION

La escasez de agua a nivel mundial es alarmante como lo expresa la Organización de Naciones Unidas (ONU):

“La escasez de agua constituye uno de los principales desafíos del siglo XXI al que se están enfrentando ya numerosas sociedades de todo el mundo. A lo largo del último siglo, el uso y consumo de agua creció a un ritmo dos veces superior al de la tasa de crecimiento de la población y aunque no se puede hablar de escasez hídrica a nivel global, va en aumento el número de regiones con niveles crónicos de carencia de agua. La escasez de agua es un fenómeno no solo natural sino también causado por la acción del ser humano. Hay suficiente agua potable en el planeta para abastecer a los 7.000 millones de personas que lo habitamos, pero ésta está distribuida de forma irregular, se desperdicia, está contaminada y se gestiona de forma insostenible”. [1]

En el planeta hay ambientes con grandes cantidades de agua, pero en otras zonas con escasez de agua dramática y sequías.

Esto provoca que los gobiernos y empresas privadas hagan un gran esfuerzo para buscar nuevas formas para enfrentarse a los retos de abastecimiento del agua.

En muchas explotaciones agrícolas se hace uso poco racional del agua, regando generalmente por inundación provocando problemas ambientales, sociales y económicos:

Ambientales:

- El mal uso del agua, priva el uso prioritario en centros urbanos.
- La contaminación del agua hace necesario implementar un sistema de reutilización del agua residual.
- Las buenas prácticas agrarias, determinan la conservación de la calidad del suelo y del agua.

Sociales:

- La adaptación a la evolución de los mercados globalizados de productos agrarios con la creciente demanda de alimentos y de energía.
- El asentamiento de las poblaciones
- Las nuevas tecnologías aplicadas a la gestión del agua racionaliza y mejora la creación de empleos de calidad.

Económicos:

- Facilitar alimentos de calidad, seguros y en cantidad.
- Facilitar la capacidad del agricultor de adaptarse a las fluctuaciones del mercado económico.
- Maximización costo/beneficio del agua y agroquímicos
- Alimentos producidos según los estándares internacionales.

Acciones que se deben llevar a cabo para mejorar el uso del recurso hídrico:

- Mejorar las estructuras hídricas existentes para evitar pérdidas de agua (impermeabilización, pérdidas por evaporación, etc.) que permiten el ahorro del agua.
- Modificación de los sistemas de bombeo, transporte y distribución.
- Cambio de los sistemas de aplicación del agua.
- Mejora de la red de drenaje.
- Mejora de la capacidad de regulación, almacenamiento y control del agua.
- Control del consumo del agua. Establecimiento de sistemas de control mediante la instalación de caudalímetros u otros tipos de instrumentos.
- Mejora en la gestión del agua.
- Implantación de tecnologías de comunicaciones vinculadas a la irrigación o a redes de energía.
- Actuaciones en sistemas de depuración. Obras de reutilización del agua regeneradas o desalinizadas para uso agrario para permitir, al menos, una reducción equivalente del suministro tradicional para riego.
- Implantación y mejora de las instalaciones eléctricas vinculadas a los sistemas de riego.

Mediante las acciones enumeradas para mejorar el recurso hídrico se pretende la consecución de los siguientes objetivos:

- Optimizar el agua disponible.
- Mejorar la eficiencia global del sistema de riego.
- Disminuir las demandas.
- Mejorar la rentabilidad de las explotaciones.
- Aplicar nuevas tecnologías.
- Mejorar el nivel de vida de los agricultores y mantener a la población en el medio rural.

- Mejorar la calidad del agua.
- Mejorar la situación ambiental de la explotación.
- Conservación del paisaje próximo al regadío.

Por lo tanto la automatización y control de sistemas de riego se presenta como una herramienta fundamental en el desarrollo de las actuaciones y en consecuencia de los objetivos a llevar a cabo en la planificación de una explotación agraria. [2]

2.2-SISTEMAS DE RIEGO AUTOMATIZADOS

En las explotaciones agrícolas modernas, la automatización de la infraestructura hidráulica destinada a riego puede hacerse a varias escalas o en distintas partes de las instalaciones:

- Puede pretenderse una automatización individual del riego en parcela, donde el objetivo fundamental es determinar el momento adecuado para regar y la cantidad de agua a aplicar, entre otros factores, el estado de humedad del suelo o de la planta y la uniformidad en el reparto de agua del sistema. Para esta escala de automatización normalmente resulta suficiente con una serie de sensores que permitan determinar el momento óptimo de riego, un programador y un conjunto de válvulas hidráulicas o electroválvulas.
- Automatización general de una red de riego: Su principal objetivo es la ejecución, control y verificación de las actuaciones de manejo sobre la red resultante de los sistemas de gestión colectivos de la comunidad de regantes. Normalmente, están constituidas por un ordenador central o centro de control y una red en anillo de unidades de campo que controlan las electroválvulas de riego.
- Regulación y control de instalaciones hidráulicas concretas: Como estaciones de bombeo, que adaptan el caudal y la presión a la demanda de la red, con la consiguiente reducción del costo energético.

El nivel de automatización alcanzado se puede medir según la cantidad de fases del proceso de riego que controla el sistema electrónico. La elección del nivel de automatización debe elegirse según criterios técnico-económicos, según las características de la explotación y los gustos de los agricultores. El mínimo nivel de automatización que se puede alcanzar es la apertura y cierre de las válvulas de riego y el máximo será el control total del proceso de riego, censado de humedad de suelo, censado de humedad del bulbo hídrico, riego por tiempos programados, etc., incluyendo la adquisición de datos estadísticos meteorológicos, de fertilización y la presentación

adecuada de los datos y parámetros de riego, por ejemplo utilizando un servidor web embebido en un microcontrolador. [2]

Según el nivel de automatización implementado será la cualificación del personal profesional que maneje el sistema y la dependencia del personal técnico que solucione los posibles desperfectos del sistema y el mantenimiento preventivo a aplicar.[2]

2.3-SISTEMAS DE RIEGO AUTOMATIZADOS Y TELECONTROLADOS

En los sistemas de riego controlados por sistemas electrónicos los procesos de riego están automatizados y también pueden estar telecontrolados.

La implementación de estas tecnologías a la agroindustria trae asociada beneficios:

- En los regadíos telecontrolados se necesita menos personal dedicada al riego.
- Disminuye los costos económicos, sociales y ambientales del agua.
- Mejora la eficiencia en la distribución del agua, optimizando el momento de disponibilidad del agua y favoreciendo el riego durante las horas de menor demanda de evaporación como en la noche.
- Se pueden implementar planes estatales de manejo del agua y el medio ambiente porque con los sistemas telecontrolados se cuenta con información verídica y en tiempo real del uso que se hace del agua, permitiendo la facturación del agua que se consume.
- Mejora la calidad de vida de los regantes.[2]

La topología de la instalación telecontrolada debe responder a las características de la red hidráulica a monitorear y telecontrolar. Para grandes sistemas de riego se parte de una o varias fuentes de agua (canales, estaciones de bombeo, embalses, etc.) y a través de canales o cañerías llegan a los sectores de riego, desde estos sectores a través de una red secundaria se lleva a los hidrantes de agrupación (sectores de riego con control común de presión y caudal), desde este punto, la red terciaria conduce el agua hasta la entrega en la parcela. [2]

Para una arquitectura de este tipo se hace necesario disponer de sistema de bombeo, sistema de filtrado, válvulas, sensores y actuadores de diversos tipos, entre otros elementos.

2.4-NIVELES DE GESTION DE PROCESOS

La automatización debe llegar a controlar el consumo de agua y las operaciones de riego (apertura y cierre, por actuación directa o programada, con la posibilidad de modificaciones) a nivel de parcela, lo que otorga un beneficio concreto en la implementación del sistema de control. Para alcanzar esto se pueden encontrar tres niveles de gestión de procesos:

- **Primer nivel:** Centro de control de la instalación: Este nivel es el encargado de recibir y almacenar los datos de la totalidad de la red hidráulica (valores de presión, lecturas de control de consumo, caudal, etc.) y captar y mostrar los parámetros de funcionamiento del sistema. [2]
- **Segundo nivel:** Estaciones concentradoras: Estando ubicadas en puntos estratégicos de la red de riego captan la programación de riego enviadas desde el centro de control, actuando como programadores de riego, asimismo captan los datos de la red hidráulica y de funcionamiento propio del sistema para su transmisión al Centro de Control. [2]
- **Tercer nivel:** Estaciones remotas (RTUs): Estas unidades, ubicadas junto a los hidrantes, ejecutan en tiempo real las órdenes recibidas desde la estación concentradora a la cual están conectadas y envían a esta todos los datos nuevos de eventos que se vayan produciendo (Confirmación de ejecución de órdenes, lecturas de humedad, etc.), estas unidades son autónomas, esto es: una vez recibidos los parámetros de la estación concentradora los almacenan y actúan de forma autónoma, esto permite continuar con el proceso de riego ante una avería del centro de control de la instalación o de la estación concentradora a la cual pertenecen o de ambas.

La arquitectura así descripta tiene la ventaja de ser robusta. De esta forma ante una avería del centro de control o de alguna estación concentradora el sistema continua con el plan de riego preestablecido.

2.5-CARACTERISTICAS A EXIGIR A UN SISTEMA TELECONTROLADO AUTOMATICO

Para un adecuado funcionamiento que responda a las expectativas de los usuarios del sistema, es necesario que se cumplan una serie de requisitos:

- **Robustez de los elementos que se van a instalar:** El medio donde se instala el sistema es hostil para los elementos que lo componen (dispositivos

electrónicos, sistemas de comunicaciones, cables, etc.). Los elementos de control deben estar preparados para soportar niveles altos de temperatura, humedad relativa, mojaduras. El viento transporta mucho polvo y arenilla, además los animales pueden dañar el equipo y los cables de conexión.[2]

- **Seguridad de los sistemas de comunicación y procesos de funcionamiento del sistema:** Las instalaciones de telecontrol de riego abarcan grandes superficies, es necesario que los fallos no afecten a la totalidad del sistema sino que sean limitados e identificables, para reducir al máximo su impacto y el tiempo necesario para el restablecimiento de la zona afectada.[2]
- **Anti vandalismo:** Los equipos que se instalan en el campo no están normalmente vigilados, lo que facilita acciones vandálicas o de hurto. Por esto se hace necesario dificultar al máximo estas acciones, protegiendo convenientemente u ocultando los equipos.[2]
- **Autonomía:** El lugar donde se instalan los elementos de control y de actuación no suelen disponer de energía eléctrica cercana y dotarlos de energía desde el punto de distribución más cercana hace inviable el proyecto de alimentación, por este motivo es preciso dotarlos de energía de forma autónoma por ejemplo con paneles solares.[2]
- **Mantenimiento sencillo y económico:** El propio operador del sistema debe ser capaz de realizar las tareas de mantenimiento más habituales del sistema, sin tener que depender de la empresa instaladora. Para ello el manejo debe ser sencillo y amigable y los elementos de campo deben ser de fácil chequeo y sustitución. Además la empresa instaladora debe proveer al operador un manual detallado del sistema y cursos de capacitación.[2]
- **Escalabilidad:** Es muy habitual que se requiera un mayor número de parcelas a automatizar, aumentar el número de hidrantes, etc. Por lo que debe permitirse la ampliación o adaptación del sistema instalado.[2]
- **Uso de estándares comerciales:** Facilita la sustitución por avería o fin de vida útil de elementos por otros iguales o de otras marcas existentes en el mercado. Impide la dependencia absoluta de un único fabricante.[2]

2.6-ELEMENTOS QUE COMPONEN EL SISTEMA DE RIEGO

En un sistema de riego podemos encontrar los siguientes componentes:

2.6.1-Centro de control de la instalación

Generalmente está formado por un servidor con un software SCADA (Supervisory Control and Data Acquisition), encargado de gestionar los procesos y de supervisar, a través de un monitor, el desarrollo de los mismos, también puede estar implementado por un dispositivo móvil como una Tablet o un teléfono inteligente.

2.6.2-Estaciones concentradoras

Realizan la gestión total de las órdenes enviadas por el centro de control, y la adquisición y almacenamiento de datos de la red de riego y de la red de telecontrol del siguiente nivel de gestión, para su posterior transmisión al Centro de Control. Para ello contiene una tarjeta microcontrolada o un PLC, que las dota de suficiente memoria y poder de decisión. Disponen en caso de que la comunicación sea por radio de un sistema propio de alimentación, normalmente con paneles solares de baja potencia, que las dota de autonomía, estas estaciones telecontrolan al siguiente nivel de gestión: las estaciones remotas.

2.6.3-Estaciones remotas (RTUs)

Las estaciones remotas tienen extrema importancia, siendo estas las encargadas de recolectar la información de humedad y gestionar la apertura y cierre de las electroválvulas, estas estaciones además como tienen inteligencia propia, una vez que se establecieron los parámetros de riego suministrados por la estación concentradora los almacena y actúa de forma autónoma, a estas estaciones se les puede conferir la capacidad de contabilizar el consumo de agua para su facturación.

2.6.4-Actuadores

Son dispositivos que modifican la respuesta del sistema, los actuadores más habituales son las electroválvulas de bajo consumo, se deben utilizar electroválvulas robustas y de difícil obstrucción por las impurezas del agua de riego, este inconveniente es frecuente y obliga a la instalación de filtros, otros actuadores son: válvulas motorizadas, motores para el accionamiento de compuertas, etc.[2]

2.6.5-Transductores

Los transductores son dispositivos que transforman el efecto de una causa física, como la presión, la temperatura, la humedad, etc. en otro tipo de señal, normalmente eléctrica,

realizando el censado de la magnitud física y el tratamiento de la señal obtenida: acondicionándola, este acondicionamiento puede ser:

- Amplificación
- Filtrado
- Conversión de niveles
- Conversión de tensión a corriente, y viceversa
- Conversión de tensión a frecuencia, y viceversa
- Linealización

Los transductores que están en contacto directo con la magnitud a medir no deben perturbar el medio donde se realiza la medida. [3]

Los transductores se pueden clasificar de dos maneras:

a)-Teniendo en cuenta la forma de utilización de la energía generada por la magnitud física a medir:

- Activos: El elemento sensor genera la energía necesaria para el funcionamiento del transductor
- Pasivos: Requieren alimentación eléctrica para su funcionamiento

b)-Según el tipo de señal de salida:

- Analógicos: La señal de salida está definida en un intervalo continuo.
- Digitales: La señal de salida se presenta en forma de niveles discretos de tensión a los que se les asigna valores numéricos preestablecidos.

En la tabla 1 se enumeran las magnitudes físicas más comunes que se pueden medir con sensores y el componente más común a medir. [3]

Magnitud	Componente a medir	
Mecánica	Geométrica	Desplazamiento, ángulo, nivel, gradiente.
	Cinemática	Velocidad, aceleración, oscilación, flujo.
	Fuerza	Fuerza, presión, par.
	Características de materiales	Masa, densidad, viscosidad.
	Magnitud acústica	Velocidad, presión, frecuencia.
Térmica	Temperatura	Temp. por contacto, temp. por radiación.
Eléctrica	Variables de estado	Voltaje, corriente, potencia eléctrica.
	Parámetros eléctricos	Resistencia, impedancia, capacitancia, inductancia.
	Variables de campo	Campo magnético, campo eléctrico.
Química y física	Concentración	pH, humedad, conducción de calor.
	Tamaño de partículas	Contenido materiales en suspensión.
	Tipo de moléculas	Gases, fluidos.
	Magnitudes ópticas	Intensidad, longitud de onda, color.

Tabla 1: Tabla de magnitudes a medir

Fuente [3]

Al seleccionar un transductor de forma óptima se deben tener en cuenta las siguientes características:

Características estáticas:

- Rango de medida: Es la diferencia entre el valor máximo y el mínimo que el elemento es capaz de medir, este debe ser lo suficientemente grande para que abarque todas las magnitudes esperadas de la cantidad a ser medida.
- Sensibilidad: Es la razón entre una variación de la magnitud de salida y la correspondiente variación de la magnitud de entrada que la provoca, para obtener datos significativos, el transductor debe producir una señal de salida suficiente por unidad de entrada.
- Resolución: Es la mínima variación detectable de la magnitud de entrada.
- Sobre rango: Es la máxima magnitud de entrada que se puede aplicar al transductor sin causarle daños permanentes.

- Error de medida: Se define como la razón entre el error total y el rango completo de medida, se suele expresar en tanto por ciento. Existen errores que no son atribuibles al transductor, denominados sistemáticos, que son inherentes a la medida, se pueden producir por: vibraciones, defectos en la alimentación, errores de calibración, etc.
- Error de cero (Offset): Es aquel que se produce cuando la magnitud a medir es nula y la señal proporcionada por el transductor no lo es. Este error suele producirse en transductores pasivos, estos pueden proporcionar señal de salida en ausencia de señal de entrada porque necesitan para su funcionamiento de una fuente de alimentación externa al transductor. Este error se corrige introduciendo al sistema una señal de valor constante de magnitud opuesta al error que lo contrarreste.

El transductor está compuesto por tres etapas en la generación de la señal eléctrica de salida en respuesta a la medida física, estas son:

- Sensor: El sensor es el elemento que está en contacto directo con el proceso a medir. Convierte la magnitud a medir en otra que puede ser (eléctrica o no), relaciona la magnitud de entrada y de salida.
- Transductor: Transforma la señal entregada por el sensor en otra eléctrica cuya medida y tratamiento resulten más fáciles.
- Acondicionador: La señal entregada por el sensor puede no ser válida para ser procesada, puede requerir: amplificación, filtrado, etc. [3]

Como sensores y elementos de captación de señal de uso agrario se pueden enumerar:

- **Transductores de presión:** Se utilizan para verificar el correcto funcionamiento de las bombas de agua y las válvulas de riego para detectar fugas o roturas de tuberías.
- **Contadores:** Permiten medir el agua consumida.
- **Caudalímetros:** Empleados para medir el caudal del agua circulante por una cañería.
- **Detectores de lluvia:** Son de tipo on/off y detectan la caída de lluvia.
- **Sensores de humedad:** Son de diferentes principios de funcionamiento, pueden ser capacitivos inductivos o resistivos, se basan en el cambio de una magnitud física con la humedad.
- **pH metro:** Detectan concentraciones de sales. [2]

2.7-REDES DE DISTRIBUCION Y SISTEMAS DE ALMACENAMIENTO

De forma general, una red de riego se compone de una serie de infraestructuras que se pueden enumerar de la siguiente manera:

- **Obras de captación:** Se deben realizar garantizando las necesidades hídricas para la superficie a regar, pueden estar constituidas por acuíferos subterráneos, derivaciones de ríos, canales de riego, etc.
- **Obras de almacenamiento y regulación:** Su objetivo es la de acumular el agua ya sea para almacenamiento o como regulación de los caudales a distribuir en la red de riego.
- **Obras de transporte y distribución:** Para el transporte y la distribución del agua desde los puntos de almacenamiento(o directamente la captación) hasta el pie de la parcela, mediante una red de conducción generalmente a presión. Cuando su función se limita al transporte se denomina aducciones o impulsiones si se requiere instalaciones de elevación. Si además distribuye el agua entre las parcelas se conoce como redes de distribución.
- **Obras de tratamiento y filtrado:** Son obras destinadas a garantizar la calidad del agua, de forma que no se presenten problemas para su aplicación en el riego (plantas depuradoras, sistemas colectivos de filtrado, etc.).
- **Obras de maniobra, protección y automatización:** Son las destinadas a garantizar el correcto funcionamiento del sistema, así como a posibilitar su automatización.
- **Obras de drenaje:** Son las destinadas a evacuar los excesos de humedad del suelo.[2]

2.8-FORMAS HABITUALES DE RIEGO

Los sistemas habituales de riego que se puede encontrar en la bibliografía consultada y de aplicación en la Provincia de Catamarca son de dos tipos, cada uno de ellos con sus ventajas y desventajas que se analizan a continuación:

Por manto o inundación. Para poder aplicar este sistema el terreno debe ser trabajado de forma tal que las áreas a ser irrigadas, o parte de estas, deben ser prácticamente horizontales, rodeadas por pequeños diquecitos que contienen el agua. En esta modalidad, una vez que la parcela se ha llenado de agua, se cierra la entrada a la misma, el agua no circula sobre el suelo, se infiltra o evapora. Este tipo de riego, además de consumir mucha agua tiene también un efecto poco deseable de lixiviación y compactación del suelo. [4]

Riego localizado. Para referirnos a este tipo de sistema se hace necesario realizar una subdivisión, la que se detalla a continuación:

- **Riego por tuberías emisoras:** Se caracteriza por la instalación de tuberías emisoras sobre la superficie del suelo creando una banda continua de suelo humedecido y no en puntos localizados como en el riego por goteo, este tipo de riego tiene una presión de trabajo entre 1 y 5 mca con caudales que oscilan entre 0,5 y 2 l/h
- **Riego por micro aspersión y micro difusión:** En él, el agua se aplica sobre la superficie del suelo en forma de lluvia muy fina, mojando una zona determinada que depende del alcance de cada emisor, este tipo de riego tienen presiones de trabajo de entre 10 y 20 mca y caudales de hasta 300 l/h.
En este tipo de riego se suele llamar micro aspersores a los emisores que llevan algún elemento giratorio que distribuye el agua accionados por la propia presión de ésta, en cuyo caso se suele poner el límite de seis metros de diámetro mojado efectivo, si carece de estos elementos se les llama difusores o micro jets y no se limita el diámetro mojado.
- **Riego por goteo:** El agua circula a presión por la instalación hasta llegar a los goteros, en los que se pierde presión y velocidad, saliendo gota a gota. El riego por goteo suministra agua de manera lenta y uniforme a baja presión a través de mangueras de plástico instaladas dentro o cerca de la zona radicular de las plantas. Es una alternativa a los sistemas de riego por aspersores o surcos, este tipo de riego tiene presiones de trabajo en torno a 10 mca y caudales de entre 1 y 16 l/h.

Estos tipos de riego se dividen también en dos grupos dependiendo del caudal que utilicen:

- **Bajo caudal:** hasta 16 l/h, entrando en este grupo el riego por goteo y el riego por tuberías emisoras.
- **Alto caudal:** mayores de 16 l/h, este grupo lo constituyen el riego por micro aspersión y micro difusión. [4]

2.9-SISTEMAS DE RIEGO LOCALIZADO

Considerando que para la realización del presente trabajo, se ha optado por la implementación de un sistema de riego localizado, se profundizara sobre este tipo de riego.

Las instalaciones constan de un suministro de agua que puede proceder de un grupo de bombeo desde un pozo o depósito, este suministro de agua se conduce a un cabezal de riego que comprende los dispositivos necesarios para controlar toda la instalación e incluye dispositivos de filtrado, de inyección de fertilizantes, de control de presión y de volúmenes aplicados, programador de riego, etc.

La red de distribución parte del cabezal y consta de tuberías primarias que conducen el agua hasta las unidades o sectores de riego y secundarias que llevan el agua desde la primaria hasta las subunidades de riego, con las correspondientes válvulas y acoplos, y las subunidades que están formadas por las tuberías terciarias y los laterales porta emisores. [2]

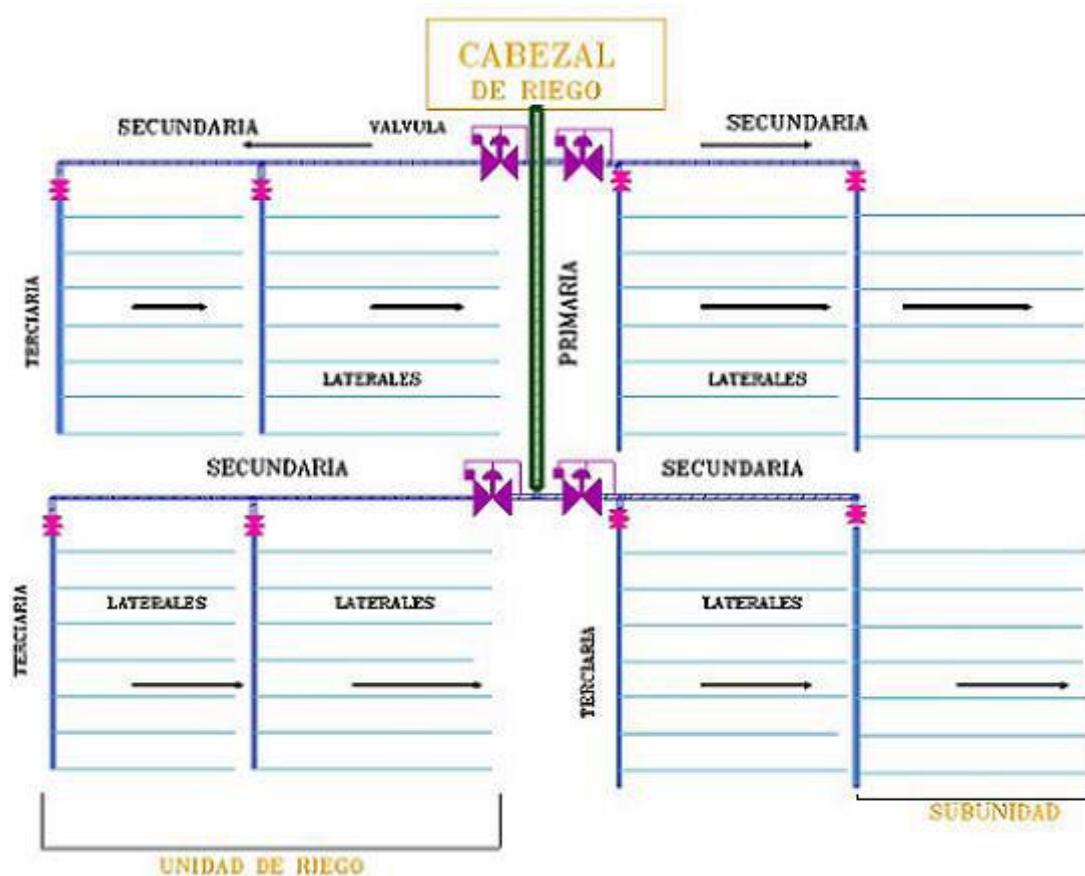


Figura 1: Instalación de riego localizado

Fuente [2]

Según la descripción y la figura 1 se puede definir:

- Unidad o sector de riego: Es la superficie que se riega simultáneamente desde un mismo cabezal e incluye una o más subunidades de riego.

- Subunidad de riego: Es la superficie que se riega simultáneamente desde un mismo punto donde se controla o regula la presión.

2.10- RIEGO POR GOTEO

Dentro de los sistemas de riego localizado se ha adoptado el riego por goteo y es por ello que se explicara más en detalle este sistema y los diversos componentes necesarios.

2.10.1-¿Por qué usar riego por goteo?

El riego por goteo puede reducir el uso de agua. Un sistema de riego por goteo bien diseñado pierde muy poca agua porque hay poco escurrimiento, evaporación o percolación profunda en suelo limoso. Con el riego por goteo hay menos contacto del agua con el follaje, los tallos y los frutos. Por eso, las condiciones son menos favorables para el desarrollo de enfermedades en las plantas. Con un buen programa de riego que cubre las necesidades de las plantas, es posible aumentar el rendimiento y la calidad de la cosecha.

Los agricultores y profesionales a menudo hablan del “riego por goteo sub superficial”, RGS. Si la manguera o cinta de riego está instalada bajo la superficie del suelo, hay menos riesgo de que sea dañada debido a la radiación UV o las operaciones de labranza o eliminación de maleza. Con el RGS, se maximiza la eficiencia del riego porque hay poco escurrimiento y evaporación.

La aplicación de productos químicos agrícolas es más eficiente a través del riego por goteo. Debido a que la aplicación de agua está limitada a la zona radicular, es menos probable que el nitrógeno que se encuentra en el suelo se pierda a través de la percolación profunda (lixiviación). Además, el uso de fertilizante es más eficaz, y a menudo es posible usar menos insecticida. [3]

2.10.2-Ventajas del uso de riego por goteo

El riego por goteo también ofrece las siguientes ventajas:

- Es apropiado para los campos de forma irregular o donde la topografía o textura del suelo no es uniforme. Hay que tener en cuenta estos factores al diseñar el sistema de riego. Los sistemas de riego por goteo también son una buena opción donde hay altas tasas de infiltración, formación de charcos o un exceso de escurrimiento en algunas partes del campo.

- Es útil si el agua es escasa o costosa. Con menos evaporación, escurrimiento y percolación profunda, y con mayor uniformidad de aplicación, no es necesario aplicar un exceso de agua a ciertas áreas del campo para asegurar que otras reciban suficiente agua.
- La aplicación de nutrientes es más precisa, de este modo, se pueden reducir los gastos en fertilizantes y la pérdida de nitratos. Además, se puede escoger el mejor momento para fertilizar y satisfacer las necesidades de las plantas.
- Es posible diseñar el sistema de tal manera que el área entre hileras se mantenga seca, permitiendo así operaciones de tractores en cualquier momento. Esto facilita la aplicación de herbicidas, insecticidas y fungicidas en el momento más oportuno.
- Un aumento en el rendimiento y calidad es posible mediante la programación precisa del riego, la cual se hace posible con el sistema por goteo. Se han observado aumentos en rendimiento y calidad de cebolla, lúpulo, brócoli, coliflor, lechuga, melón, tomate, algodón y otros cultivos.
- Se puede automatizar y telecontrolar. [3]

El riego por goteo permite alta eficiencia en el uso del agua.

2.10.3-Emisores de riego por goteo

Este elemento es el que determina en su mayor parte la capacidad y el funcionamiento del sistema de riego, los emisores se emplean para disipar la presión del agua en la tubería y descargarla con un caudal determinado. En el mercado existen muchos modelos de emisores con distintos diseños, tanto en goteros como micro aspersores y difusores. El orificio por el que sale el agua al exterior se llama punto de emisión y puede ser simple o múltiple.

Se suele diferenciar los emisores en:

Emisores puntuales (point source): Estos están separados más de un metro, suelen emplearse en árboles, viñedos, ornamentales y arbustos.

Emisores continuos (line source): Estos descargan el agua en puntos más cercanos, o continuamente, a lo largo de la línea como los empleados en los cultivos de maíz, algodón, hortícolas, etc. [2]

2.11-OBJETIVO GENERAL

Solucionar la mala gestión del recurso hídrico en explotaciones agrícolas en zonas áridas como la provincia de Catamarca donde el recurso es escaso y con periodos de sequía, generalmente, se riega por inundación desperdiando el agua necesaria también para la producción ganadera y fundamentalmente para la vida humana, aportar un producto al mercado local desarrollado y producido en el país que permita futuras ampliaciones y el mantenimiento por personal capacitado con tecnología argentina.

2.12-ALCANCE

Desarrollar un prototipo de un sistema de riego electrónico para optimizar el recurso hídrico, que permita aportar agua y nutrientes disueltos al pie de la planta de manera localizada sin desperdiciar agua regando zonas innecesarias.

Se pretende desarrollar un prototipo apto para su comercialización a nivel regional con tecnología accesible en el país y de bajo costo que reemplace o mejore la tecnología existente

El prototipo a desarrollar tiene que ser apto para detectar un umbral mínimo de humedad donde comenzar a regar y un umbral máximo de humedad donde cortar el suministro de agua, para esto se debe elegir el sensor de humedad de suelos óptimo.

Además de poseer la capacidad de regar según el parámetro de humedad debe poseer la capacidad de programar el riego por tiempo, activándose el riego a una hora y minutos específicos del día, para esto se debe utilizar un reloj de tiempo real.

Debe poder manejar una bomba de agua apropiada para explotaciones agrícolas y electroválvulas de similares características, diseñándose una interfaz de potencia, utilizando dispositivos semiconductores apropiados para las potencias involucradas.

El sistema de riego debe estar compuesto por una central de comando y como mínimo uno o varios subsistemas de campo interconectados, los subsistemas de campo son circuitos independientes del control central que tienen la capacidad de trabajar de forma autónoma de la central manejando el recurso hídrico habilitando las electroválvulas y censar la humedad del suelo, pero actuar según los límites de humedad y demás parámetros establecidos por la central de comando.

Estos subsistemas de campo deben poder estar alejados de la central de comando una distancia considerable, para esto se debe diseñar la interfaz y protocolo de comunicaciones óptimos.

La central de comando debe permitir la programación de los parámetros de riego y suministrar información sobre las actividades que está desarrollando el sistema.

La central de comando debe poder activar la bomba de agua según los requerimientos solicitados por los subsistemas de campo.

UNIVERSIDAD NACIONAL DE CATAMARCA



3. MEMORIA DESCRIPTIVA

Diseño de un sistema de riego para optimizar el recurso hídrico

Matías Leandro Ferraro

Año: 2015

Título: Ingeniero Electrónico
Director de tesis: Sergio Hilario Gallina
Departamento: Electrónica

3-MEMORIA DESCRIPTIVA

Se desarrollaron tres sistemas de riego, estos son:

3.1-PRIMER PROTOTIPO

Inicialmente se construyó el sistema en forma de maqueta, pero con la electrónica de un sistema de riego real, se podían ver los elementos constituyentes de este sistema sobre un tablero, el objetivo de este desarrollo fue mostrar las posibilidades del sistema de control automático aplicado al riego para una serie de charlas con productores locales. Se pensó que todas las partes intervenientes estén separadas , la etapa de control, potencia y comando se encuentra en un tablero separadas unas de otras, se emplearon tres sectores de riego, dos de los cuales pueden programarse por tiempo y uno por humedad implementados físicamente en tres cajones de madera sellados con pintura, también se construyó un depósito de agua con posibilidad de regular presión y caudal de agua por control mecánico y se utilizó un módulo de electroválvulas, una por cada sector de riego. Se diseñó de forma de utilizar una bomba de reducido caudal, más concretamente una bomba de lavarropas automático. Se formuló para indicar las capacidades de un sistema de control electrónico pero sin perder de vista que era a nivel de maqueta, su principal falencia era el sensor de humedad, este censaba humedad ambiente y mediante este dato se estimaba la humedad de suelo.

3.2-SEGUNDO PROTOTIPO

Consistió en desarrollar un sistema comercial, dentro de un gabinete apto para exteriores, una central de comando y unidades de comando en el campo, las cuales estando conectadas a la central por el protocolo de comunicaciones RS-485 comandan las electro válvulas y la bomba, estas unidades de comando en el campo pueden ser múltiples y distribuirse por sectores sobre el terreno de la explotación agrícola, la mayor ventaja de este desarrollo es la utilización de sensores de humedad aptos para detectar humedad de suelo, lo cual le confiere al desarrollo tecnológico confiabilidad, aumentando su capacidad de comercialización.

3.3-DESCRIPCION TECNICA DEL SISTEMA DE RIEGO IMPLEMENTADO

La tercera etapa del proyecto es el desarrollo que se expone en las siguientes líneas, fruto de la experiencia adquirida en los sistemas de riego realizados con anterioridad obteniendo un desempeño óptimo.

Descripción técnica del sistema de riego implementado:

El sistema implementado está compuesto por los siguientes bloques:

- Centro de control de la instalación
- Estación Concentradora
- Estaciones Remotas (RTUs)

3.3.1-Centro de control de la instalación

Esta unidad esta implementada por una Tablet o un teléfono inteligente donde corre una terminal bluetooth, esta terminal mediante el protocolo de comunicaciones bluetooth se comunica con la estación concentradora e indaga el historial de riego, la estación concentradora registra los parámetros y variables de riego en una memoria EEPROM de un megabyte de capacidad, los parámetros de riego se guardan en la memoria de forma codificada, esto permite a futuro desarrollar un software de aplicación capaz de interpretar los datos

3.3.2-Estacion concentradora

La estación concentradora tiene comunicación bluetooth para transmitir los parámetros y variables de riego al centro de control de la instalación y comunicación por el protocolo RS-485 con las estaciones remotas, además posee una interfaz de programación constituida por un teclado matricial de cuatro columnas y cuatro filas y un display de cuatro líneas y dieciséis caracteres por línea con retroiluminación azul, además cuenta con un reloj de tiempo real y una memoria EEPROM de un megabyte de capacidad para guardar variables y parámetros de riego

3.3.3-Estacion remota (RTU)

El sistema se puede dividir en dos sistemas realimentados:

- Sistema de control de humedad de suelo.
- Sistema de control por tiempo en una franja horaria preestablecida.

Sistema de control de humedad de suelo:

Se entiende por sistema de control de humedad de suelo al que mediante la utilización de un sensor de humedad y la información del estado actual de las válvulas y las bombas mantiene la humedad del suelo en un valor ingresado previamente como consigna al sistema.

La humedad de referencia es ingresada a la estación concentradora por el usuario, esta estación la envía por el bus de comunicaciones a la estación remota, el pedido de activación de la bomba es enviado por la estación remota a la estación concentradora por el bus de comunicaciones.

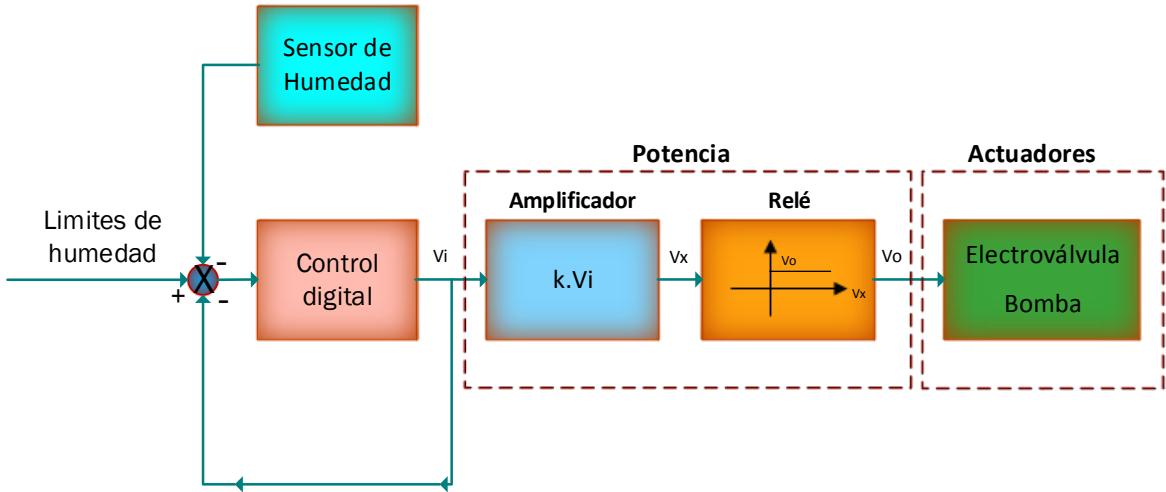


Figura 2: Diagrama en bloques del sistema realimentado de Control de Humedad de Suelo

El sistema de la figura 2 se puede dividir en cinco bloques, el control digital es el encargado de controlar el sistema, está constituido por un microcontrolador con un firmware embebido programado en lenguaje C, este control tiene una doble realimentación negativa una de estas realimentaciones provee información proporcionada por el sensor de humedad y la otra realimentación es una señal que indica el estado de los actuadores, con estas dos señales y la información de los límites de humedad el sistema actúa de forma tal de mantener la humedad dentro de los límites preestablecidos, luego en lazo abierto se modelan tres bloques: el amplificador y el relé que conforman la interfaz de potencia y el bloque de actuadores.

Sistema de control por tiempo en una franja horaria preestablecida:

Se entiende por sistema de control por tiempo en una franja horaria preestablecida al que mediante la utilización de un reloj de tiempo real que proporciona la hora actual y la información del estado actual de las válvulas y las bombas efectúa el riego en la franja horaria preestablecida como consigna.

La franja horaria de riego y la hora actual lo envía la estación concentradora por el bus de comunicaciones a la estación remota y el pedido de activación de la bomba es enviado por la estación remota a la estación concentradora también por el bus de comunicaciones

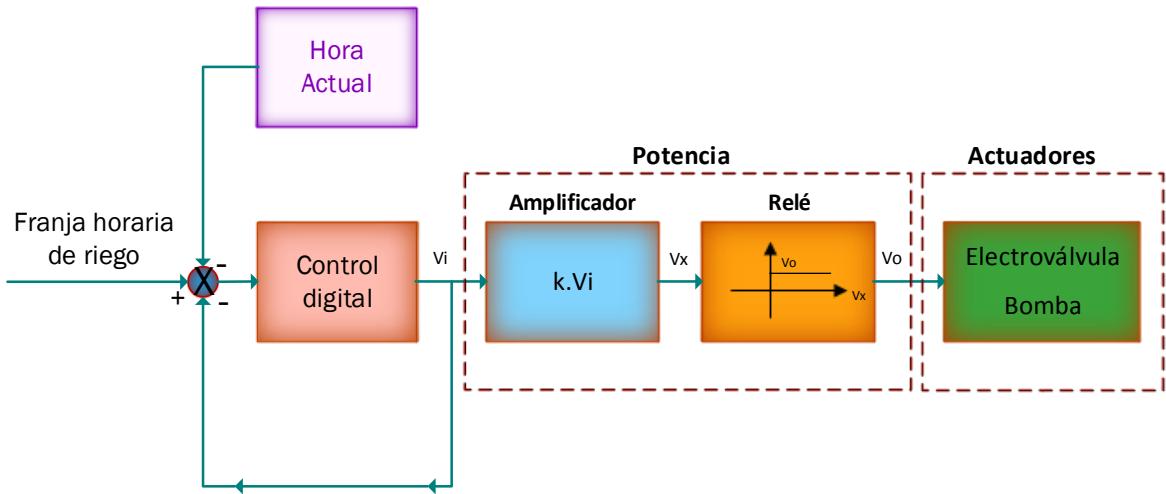


Figura 3: Diagrama en bloques del sistema de control por tiempo en una franja horaria preestablecida

Este sistema es similar al sistema de control de humedad de suelo de la figura 2, pero difiere en que una de las realimentaciones es proporcionada por un reloj de tiempo real que entrega la hora actual y la consigna es la franja horaria de riego como se puede observar en la figura 3.

El control digital de ambos está formado por un microcontrolador PIC18f4550 de 16 bits programado en lenguaje C, trabajando en multitarea, implementando en el firmware embebido el sistema de control de humedad de suelo y el sistema de control por tiempo en una franja horaria preestablecida.

El sistema de control de humedad de suelo realiza las lecturas de humedad del suelo y según esta realimentación y el estado actual de los actuadores decide la apertura y cierre de las electroválvulas y las bombas, la señal de control que activan y cierran las electroválvulas responden a un ciclo de histéresis, el ancho de la ventana de este ciclo de histéresis es variable y responde a los límites máximos y mínimos de humedad ingresados por teclado en la estación concentradora y entregados al sistema de control del nodo remoto como consigna.

La interfaz de potencia se implementó con dos relés con bobinas de 12v manejados con transistores bipolares, estos relés manejan las dos electroválvulas de agua instaladas en el sector de riego.

Se utilizaron sensores de marca Decagon Devies, modelo EC-5, en la fotografía 1 se puede observar su aspecto físico.



Fotografía 1: Fotografía del sensor de humedad EC-5

Fuente [5]

La tensión de alimentación de este sensor de humedad debe estar en el rango de: 2,5V a 3,6V de corriente continua y tiene un consumo de 10mA, entregando un voltaje proporcional a la humedad, esta señal analógica es digitalizada y utilizada en la estación remota y transmitida por el bus RS-485 a la estación concentradora.

El control por tiempo se realizó con el reloj de tiempo real ds1307, el cual dispone de comunicación I2C, alimentación de 5V y una batería externa. Dado el caso en que la tensión de alimentación descienda por debajo de los 3V es alimentado por la batería externa, este tipo de alimentación mantiene el reloj con su hora actualizada, en la figura 4 se puede observar la distribución de pines del reloj de tiempo real.

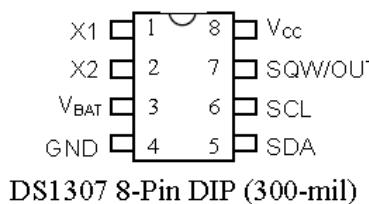


Figura 4: Asignación de pines del DS1307

Fuente[6]

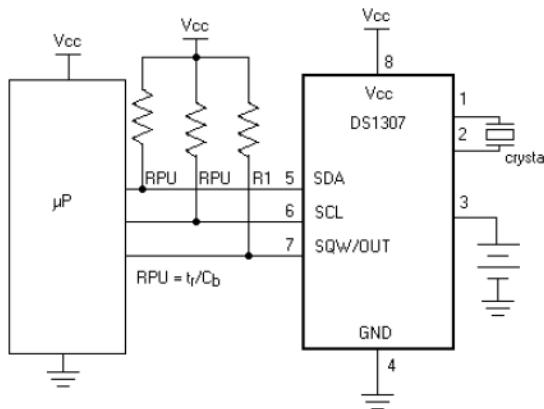


Figura 5: Conexiones del DS1307

Fuente [6]

En la figura 5 se puede observar la conexión del reloj de tiempo real con el microcontrolador y las resistencias de pull up del bus I2C.

Se utilizaron válvulas con control eléctrico/hidráulico de uso agrícola marca Todoriego, el modelo S390, esta electroválvula se alimenta con 24V y consumen 130mA, en la fotografía 2 se puede observar la electroválvula utilizada.



Fotografía 2: Fotografía de la electroválvula de riego utilizada

El principio de funcionamiento es el siguiente:

En la figura 6 se puede observar el funcionamiento de la válvula de riego.

La electroválvula puede trabajar de dos formas:

Cerrada: La restricción interna permite el acceso de la presión en la línea a la cámara de control. El solenoide controla la salida de la cámara de control. El solenoide cerrado hace que la presión se acumule en la cámara de control y cierre la válvula.

Abierta: La apertura del solenoide hace que disminuya la presión acumulada en la cámara de control, y que la presión en la línea que actúa sobre el tapón pase a la válvula

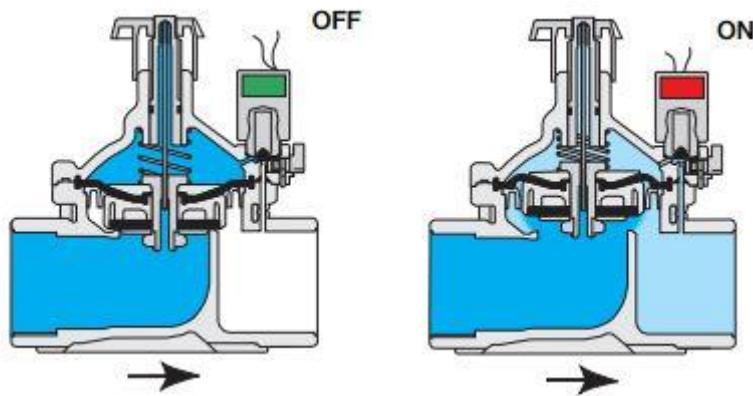


Figura 6: Funcionamiento de la válvula de riego

Fuente [7]

En el sistema de riego a implementar nos encontramos con una instalación preexistente con accionamiento manual, para su automatización fue necesario convertir las válvulas ya instaladas en válvulas con comando eléctrico, mediante la instalación de solenoides accionados por una tensión de corriente continua de 24 V. Es de destacar que el sistema hidráulico fue calculado y desarrollado por el Ing. Agr. Emiliano Gallo, los cálculos hidráulicos no están en los objetivos del proyecto, son incumbencias de otra disciplina.

3.4-HARDWARE DEL SISTEMA DE RIEGO IMPLEMENTADO

En la figura 7 se puede observar el diagrama en bloques de la estación concentradora y en la figura 8 el diagrama en bloques de la estación remota, algunos bloques son comunes a ambas estaciones y otros son propios de cada estación.

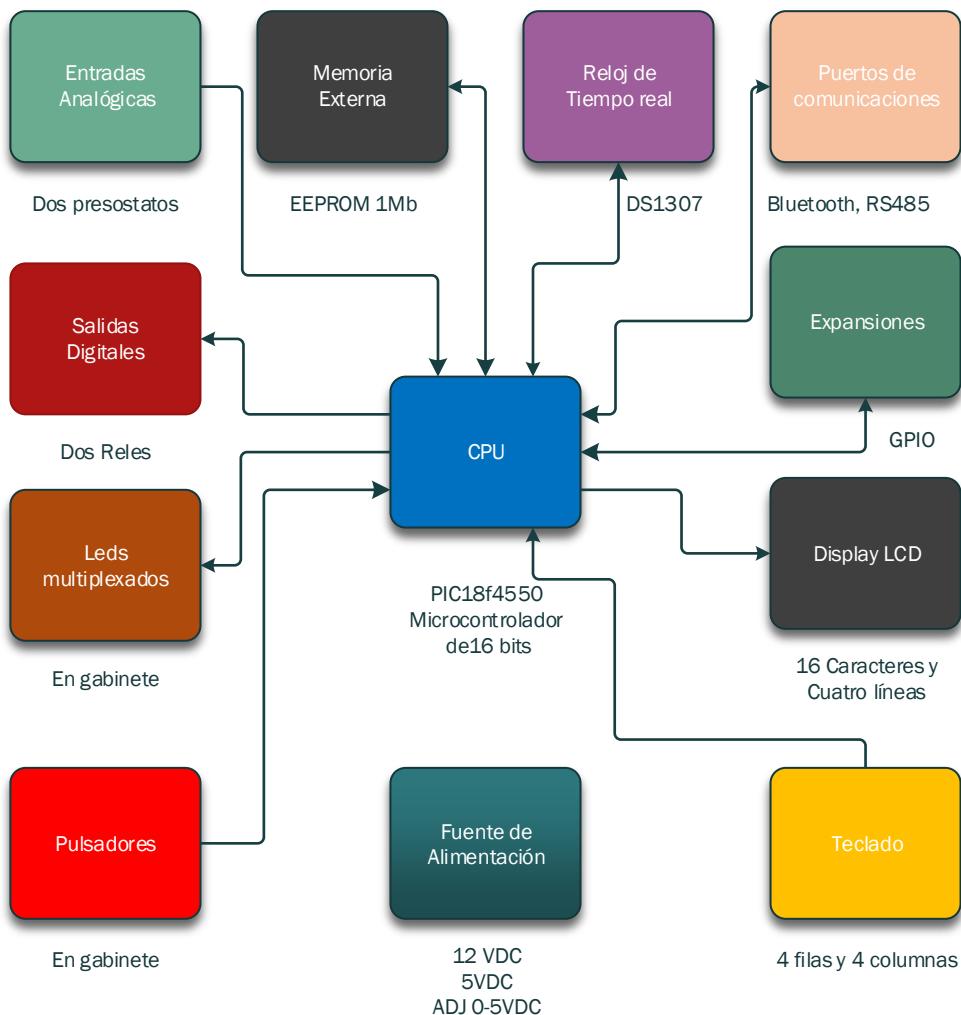


Figura 7: Diagrama en bloques de la estación concentradora

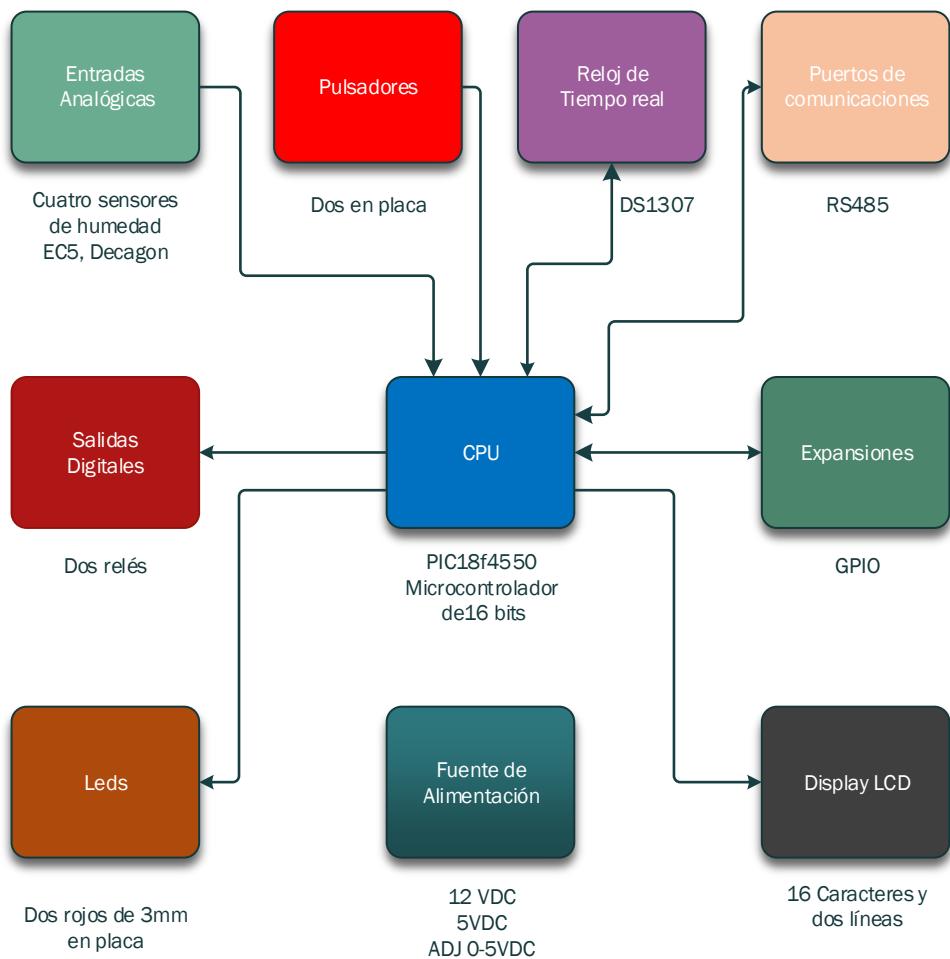


Figura 8: Diagrama en bloques de la estación remota

3.4.1-CPU

El CPU está constituido por un microcontrolador de dieciséis bits del fabricante de semiconductores microchip, el PIC18f4550. Se utilizó este microcontrolador por ser de bajo costo y su encapsulado permite montarlo sobre una placa de circuito impreso de fabricación propia.

3.4.2-Entradas analógicas

Las estaciones remotas poseen cuatro entradas analógicas destinadas a proveer información al sistema de los valores de humedad de suelo, los sensores utilizados son el modelo EC5 del fabricante Decagon Devices, estos entregan una tensión proporcional a la humedad del suelo con un rango desde cero por ciento a la saturación, la saturación

se produce entre el cuarenta y el sesenta por ciento de la tensión de alimentación dependiendo del tipo de suelo, son de tipo capacitivo, el principio físico por el cual registran el valor de humedad es por el cambio de la capacidad eléctrica donde está instalado para registrar la humedad, el cambio en la capacidad eléctrica del suelo es debido al porcentaje de humedad y como en todo capacitor al variar la capacidad eléctrica varia la tensión entre placas.

El sensor EC-5 incorpora una frecuencia de oscilación del voltaje aplicado entre placas del sensor modelado como un capacitor alta que le permite medir con precisión la humedad en cualquier suelo o sustrato independizándose de la textura y conductividad eléctrica del medio.

Se usaron cuatro amplificadores operacionales configurados en modo seguidor de tensión con ganancia unitaria destinados a aislar y adaptar impedancias entre la salida de los sensores y la entrada del conversor analógico digital, el fabricante de los sensores de humedad de suelo recomienda una tensión de alimentación entre 2,5V y 3,6V, se utilizó para alimentarlos una tensión de 2,5V, en la placa de circuito impreso se usaron borneras para PCB de tres contactos, el primer borne suministra la tensión de alimentación para los sensores de 2,5V el segundo borne es la entrada de la tensión analógica al nodo remoto entregada por el sensor de humedad, que previo paso por el amplificador operacional ingresa al conversor analógico digital del microcontrolador, el tercer borne es la masa del sensor.

La resolución indicada por el fabricante del sensor de humedad es de 0.25% de la tensión de saturación dependiente del tipo de suelo, utilizando una tensión de alimentación de 2,5V y suponiendo que se produce la saturación al 60% de la tensión de alimentación:

$$\frac{60\% \times 2,5V}{100\%} = 1,5V$$

Es decir el rango es de 0V a 1,5V, produciéndose a 1,5V la saturación, el valor mínimo de tensión entregado por el sensor de humedad es:

$$\frac{0,25\% \times 1,5V}{100} = 3,75mV$$

Esta tensión mínima que entrega el sensor de humedad equivale a un porcentaje mínimo de humedad:

$$\frac{3,75E - 3V \times 100\%}{1,5V} = 0,25\%$$

El conversor analógico digital utilizado es de 10 bits, por esto el conversor analógico digital del microcontrolador tiene una resolución utilizando como tensión de referencia 2,5V de:

$$\frac{2,5V}{2^{10}} = 2,44mV$$

Como:

$$\frac{3,75mV}{2,44mV} = 1,53$$

Con lo cual: con dos escalones del conversor analógico digital equivalen a un aumento de 0,25% de la humedad de suelo.

De los sensores de humedad de suelo disponibles en el mercado se optó por éste debido a que posee: resolución óptima, es preciso y robusto.

Para el acondicionamiento de la señal se utilizó un circuito integrado basado en el amplificador operacional: LM324N, este posee cuatro amplificadores operacionales en un mismo encapsulado, se agregó una resistencia y un capacitor a la salida del amplificador para mejorar la estabilidad, además en los pines de alimentación se instalaron dos capacitores, uno electrolítico y otro cerámico destinados a filtrar ruido que pudiera ingresar por la línea de alimentación.

Lo descripto anteriormente es para la estación remota, en la estación concentradora se utilizó el mismo circuito esquemático pero se emplearon dos de los cuatro amplificadores operacionales que posee el integrado LM324N, estas entradas analógicas en la estación concentradora están destinadas a los presostatos que miden la presión de la cañería, el sistema posee la capacidad de trabajar con dos bombas de agua y según la medida de presión que entregan los presostatos se activan las bombas necesarias.

En la figura 9 se observa el circuito esquemático de las cuatro entradas analógicas de las estaciones remotas y en la figura 10 se observan las dos entradas analógicas de la estación concentradora para la conexión de dos presostatos.

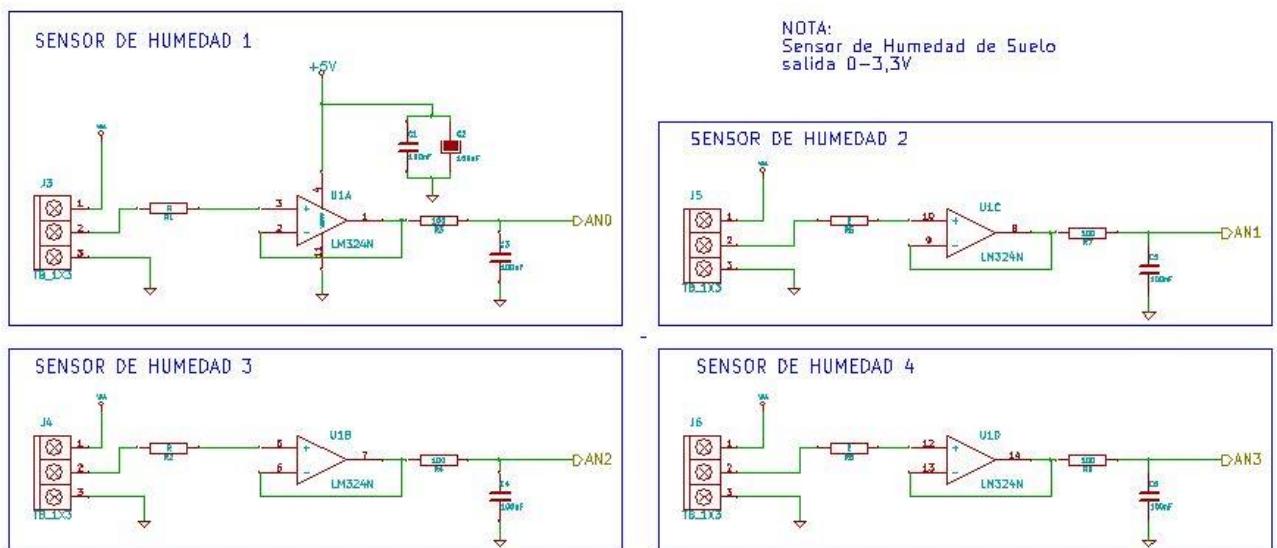


Figura 9: Entradas analógicas utilizadas en las estaciones remotas para los sensores de humedad

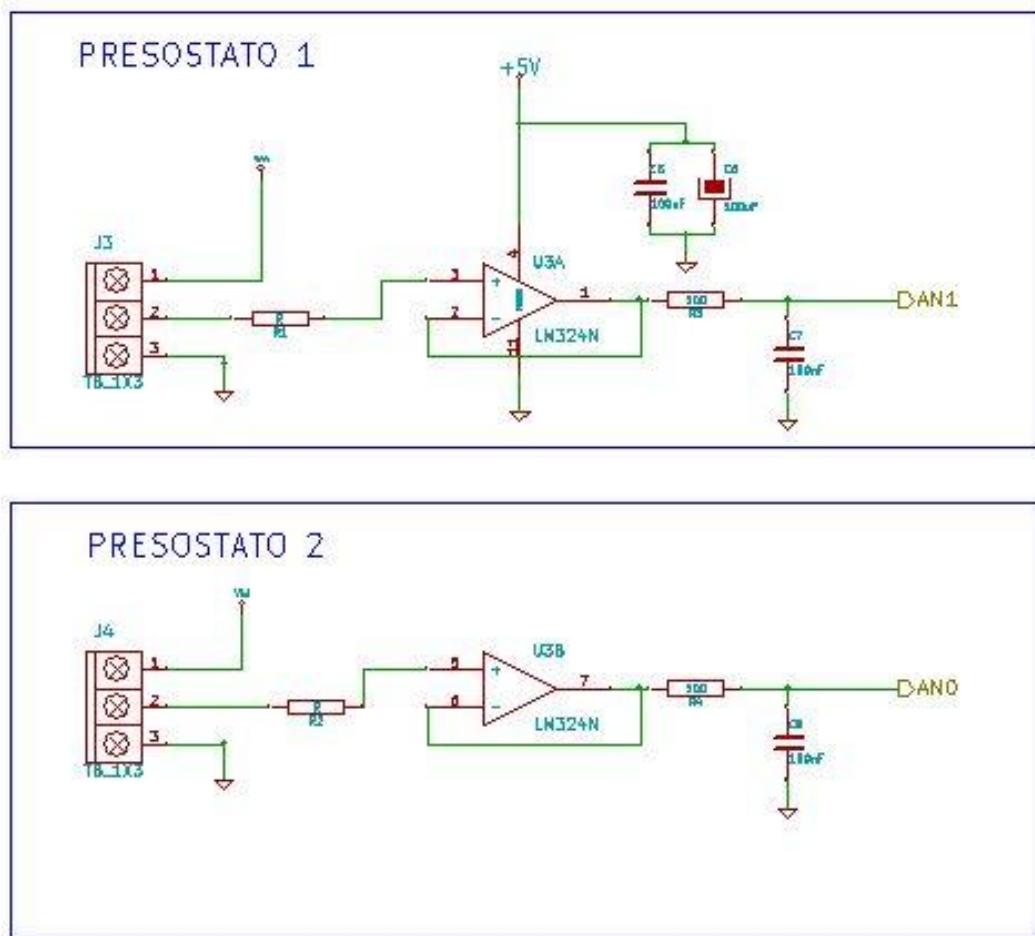


Figura 10: Entradas analógicas utilizadas en la estación concentradora para los presostatos

3.4.3-Pulsadores

En la estación concentradora en su circuito impreso se usó una ficha destinada a conectar tres pulsadores y poderlos instalarlos en el gabinete, luego de programar el firmware se llegó a la conclusión que no son necesarios los pulsadores, con el teclado matricial instalado se consigue un ingreso de información óptimo sin necesidad de utilizarlos quedando la ficha para futuras ampliaciones de la estación.

En la estación remota los pulsadores son destinados a la depuración y testeo del firmware, los dos pulsadores usados están instalados en la placa de circuito impreso que luego en el campo son inaccesibles, porque la placa de circuito impreso está dentro de un gabinete hermético.

Los pulsadores poseen una resistencia de pull down para establecer un nivel lógico bajo en el pin de entrada del microcontrolador cuando no se presiona y un capacitor para eliminar el rebote al presionarlo.

En la figura 11 se pueden observar los dos tipos de configuraciones pull down y pull up.

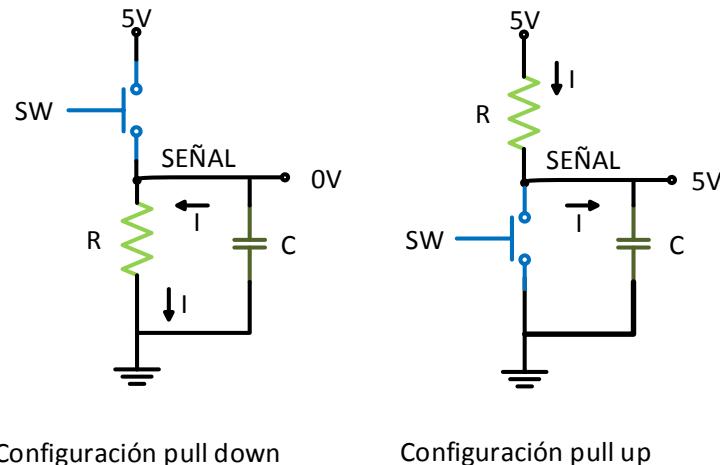


Figura 11: Configuración pull down y pull up

Este tipo de configuraciones establecen un estado lógico a la entrada de un circuito lógico cuando dicho circuito está en reposo, siendo para pull up un estado lógico alto y para pull down un estado lógico bajo. De esta forma, se evita falsos estados lógicos producidos por ruido eléctrico al dejar una entrada con un valor indeterminado.

El cálculo de la resistencia pull down se efectúa con la ley de ohm y extrayendo los siguientes datos del datasheet del microcontrolador:

$$V_{OL}=0.6v$$

$I_{OL}=1\text{mA}$ como máximo

Con lo cual:

Como mínimo el valor de la resistencia R del circuito de pull down se calcula de la siguiente forma:

$$R = \frac{Vol}{Iol} = \frac{0.6v}{1mA} = 600\Omega$$

Se eligió: 1K, cumpliendo con el cálculo anterior.

El valor del capacitor usado es de 100nF por ser un valor óptimo para evitar rebotes en el pulsador, la configuración que genera el capacitor y la resistencia es de un filtro pasa bajo, las oscilaciones de alta frecuencia producidas por el contacto mecánico del pulsador son enviadas a masa.

Se puede calcular la constante de tiempo del filtro pasa bajo, como sigue:

$$\tau = R \cdot C = 1K * 100nF = 100\mu s$$

Este valor de la constante de tiempo es óptimo, si fuera mayor al presionar el pulsador el capacitor quedaría cargado más tiempo del debido produciendo un uno lógico aun cuando el pulsador no esté presionado.

En la figura 12 se puede observar el circuito esquemático de los pulsadores instalados en la estación concentradora y en la figura 13 el circuito esquemático de los pulsadores de la estación remota.

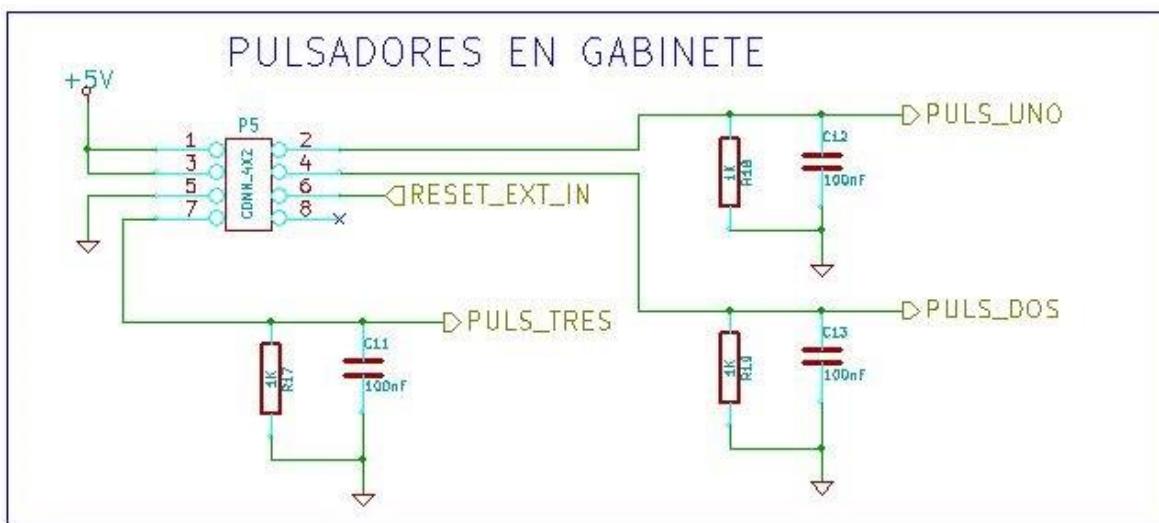


Figura 12: Pulsadores en el gabinete en la estación concentradora

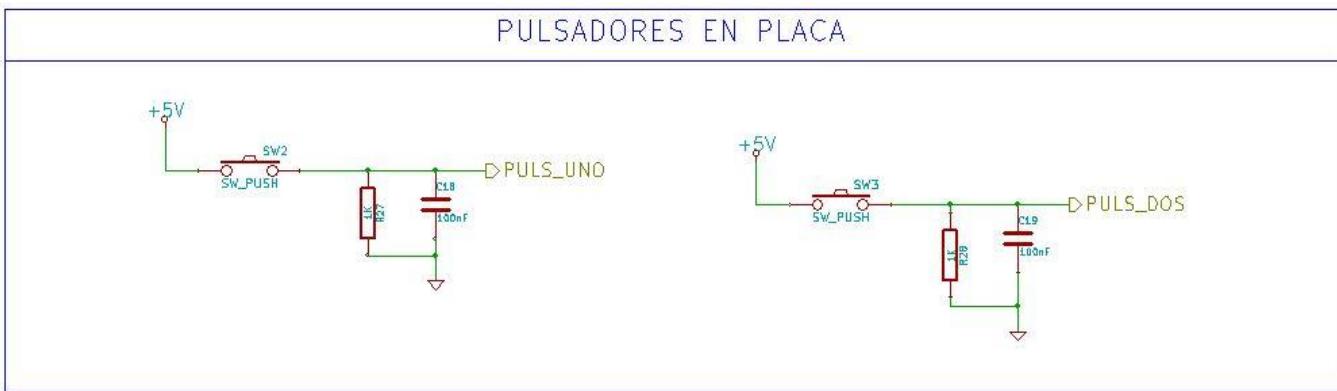


Figura 13: Pulsadores en la placa en la estación remota

3.4.4-Reloj de tiempo real

El sistema se planteó con la capacidad de riego según la variable temporal, para esto se utilizó un reloj de tiempo real, se empleó el circuito integrado DS1307, en la figura 14 se puede observar el circuito esquemático utilizado.

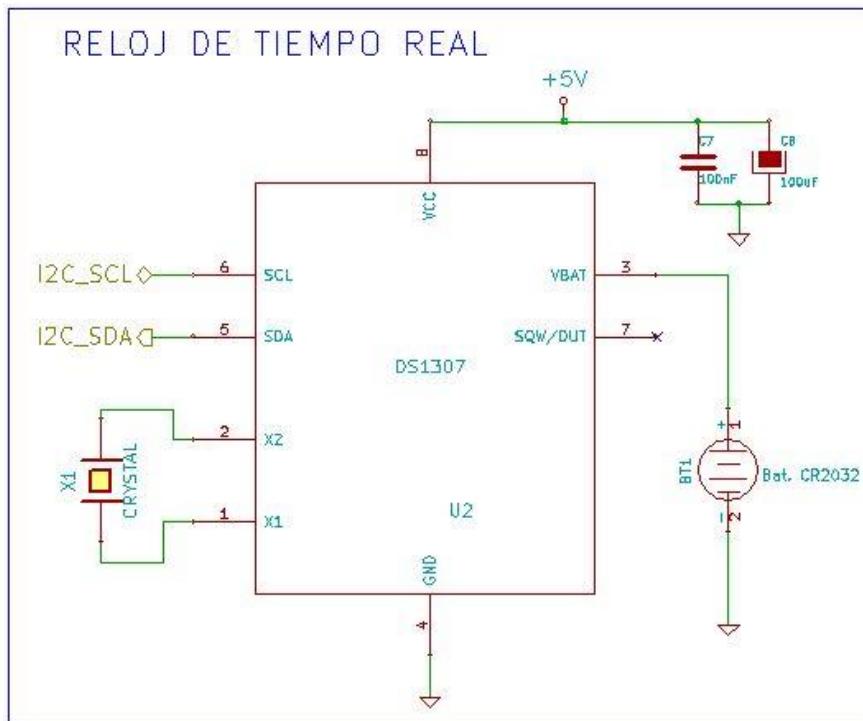


Figura 14: Circuito esquemático del reloj de tiempo real

3.4.5-Puertos de comunicaciones

La comunicación entre la estación concentradora y las estaciones remotas se efectúa por RS-485, se utilizó el circuito integrado SN75176, en la figura 15 se puede observar el circuito esquemático.

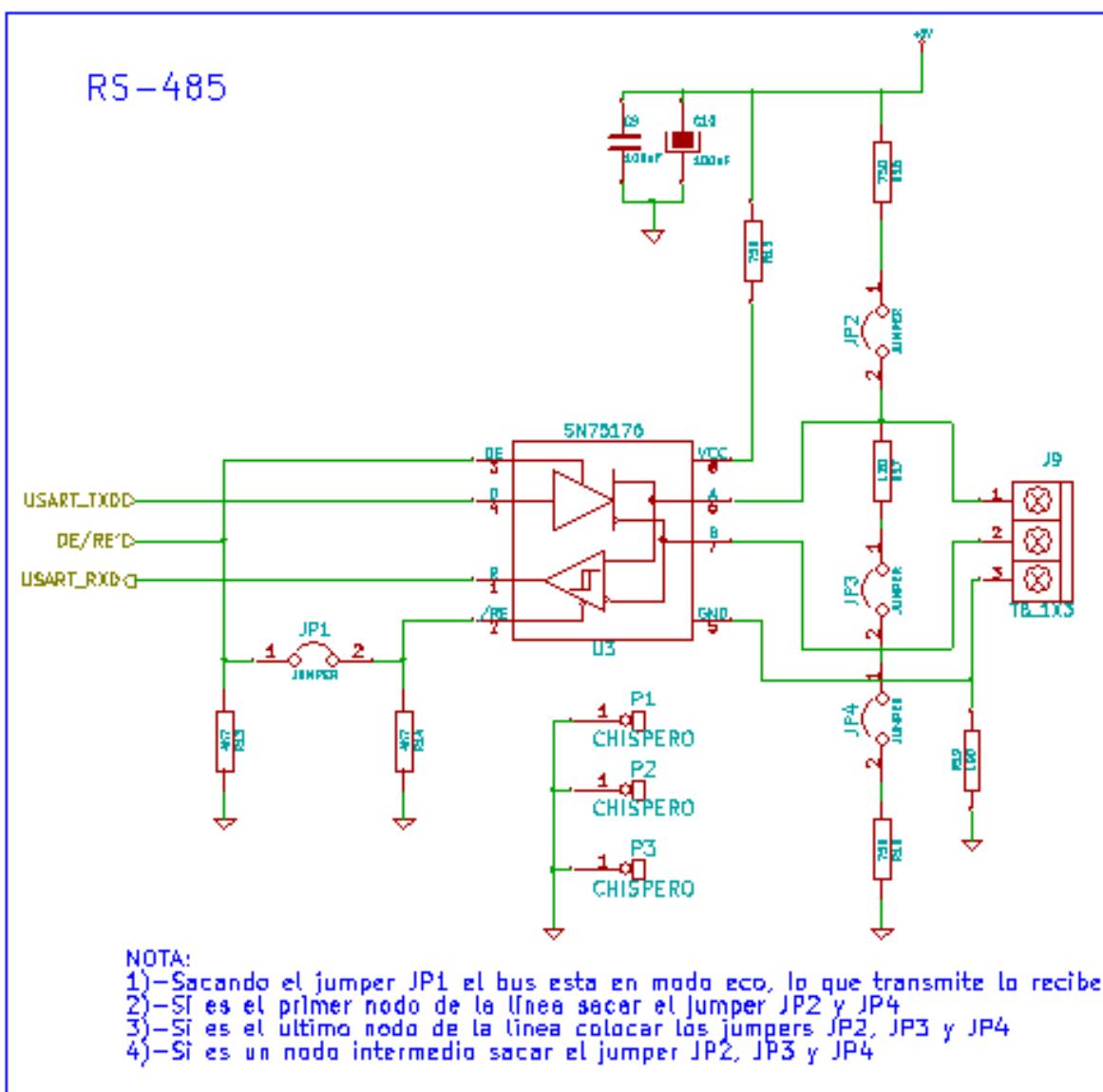


Figura 15: Circuito esquemático del puerto de comunicaciones RS-485

La comunicación entre la estación concentradora y el centro de control de la instalación se efectuó por el protocolo de comunicaciones Bluetooth, se empleó un módulo comercial basado en el circuito integrado HC-05 este circuito trabaja con un nivel de tensión de 3.3V, se utilizó montado sobre una circuito impreso que contiene un regulador de tensión de 3.3V y adaptación de niveles de tensión en las líneas de comunicación, esto permitió trabajar tanto para la alimentación como para la línea de transmisión con

5V que es la tensión de trabajo del microcontrolador, en la estación concentradora se incluyó un conector donde el modulo Bluetooth se inserta, el esquemático de este conector se observa en la figura 16.

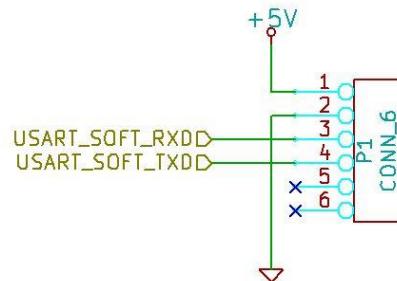


Figura 16: Esquemático del conector del módulo bluetooth

3.4.6-Expansiones

Tanto en el circuito impreso de la estación concentradora como en las estaciones remotas los pines del microcontroladores que no se utilizaron en el sistema se dejaron accesibles mediante una conector, destinados a futuras ampliaciones, la estación concentradora posee seis pines de propósitos generales y la placa de la estación remota posee nueve pines de propósitos generales, en la figura 17 se puede observar el conector de expansión de la estación concentradora y en la figura 18 el conector instalado en la estación remota

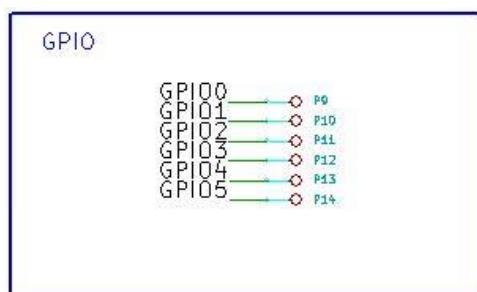


Figura 17: Puerto de propósitos generales de la estación concentradora

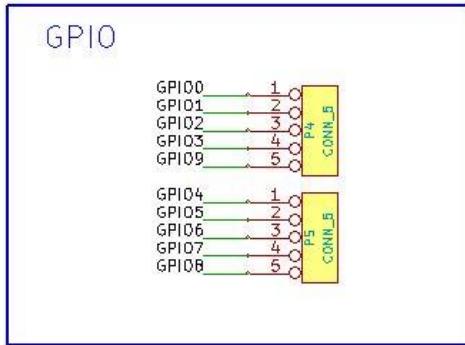


Figura 18: Puerto de propósitos generales de las estaciones remotas

3.4.7-Display de LCD

El display que se utilizó en la estación concentradora es de dieciséis caracteres por línea y cuatro líneas con luz de fondo azul, se instaló en la placa de circuito impreso un preset destinado al ajuste del contraste, se diseñó el sistema utilizando este tipo de display con estas características para hacerlo amigable, mostrando la información de una manera óptima.

En la estación remota se utilizó un display técnico de dieciséis caracteres por línea y dos líneas instalado con un conector en el circuito impreso, destinado a la depuración y testeo, una vez que el sistema pasó la fase de depuración y testeo se retiró, este display posee un preset para regular el contraste.

En la figura 19 se puede observar el conector del display de uso técnico de la estación remota y en la figura 20 el conector del display de la estación concentradora.

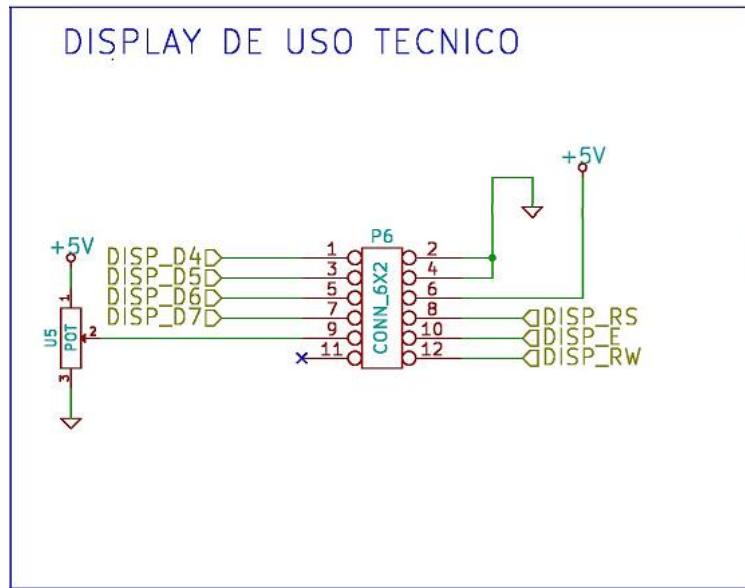


Figura 19: Circuito esquemático del conector para el display de uso técnico en la estación remota

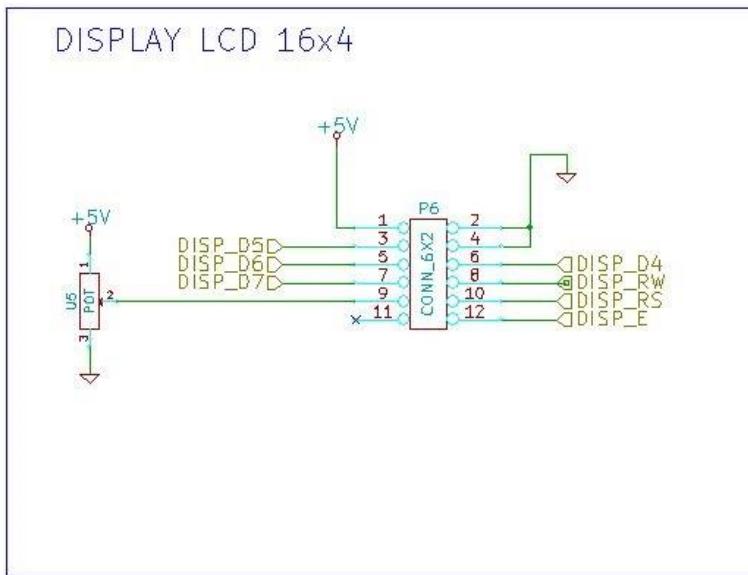


Figura 20: Circuito esquemático del conector del display utilizado en la estación concentradora

3.4.8-Leds

En la estación remota se utilizaron para usos de depuración dos leds de 3mm.

En la estación concentradora los leds se multiplexaron, cuatro pines del microcontrolador permiten manejar doce leds, este tipo de multiplexado esta sugerida en la nota técnica AN234 de microchip. [8]

Luego del diseño del firmware se llegó a la conclusión que con dos leds en la estación concentradora el sistema es amigable y optimo, quedando la posibilidad de conectar los otros leds multiplexados para futuras ampliaciones.

En la figura 21 se puede observar el conector que se utilizó y el circuito extraído de la nota técnica AN234.

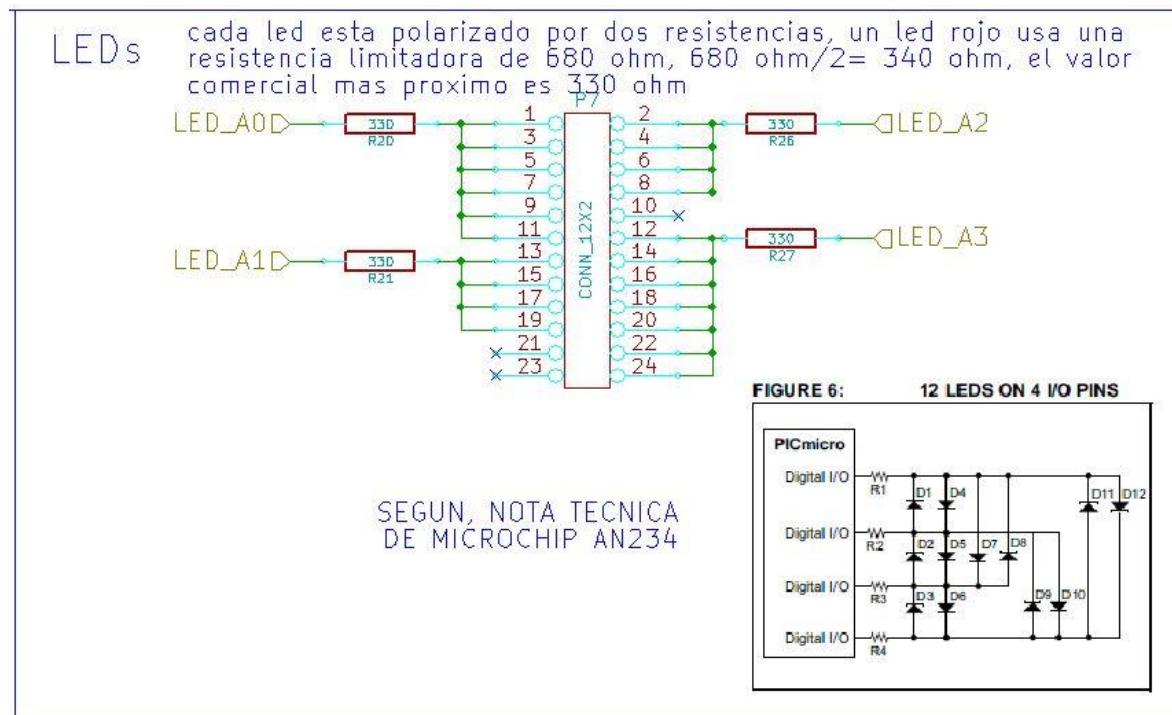


Figura 21: Circuito esquemático del conector utilizado para multiplexar doce leds en la estación concentradora

El esquemático de la figura 22 corresponde a la estación remota donde se observan los dos leds provistos en esta estación con sus resistencias limitadoras, instalados para depurar el sistema.

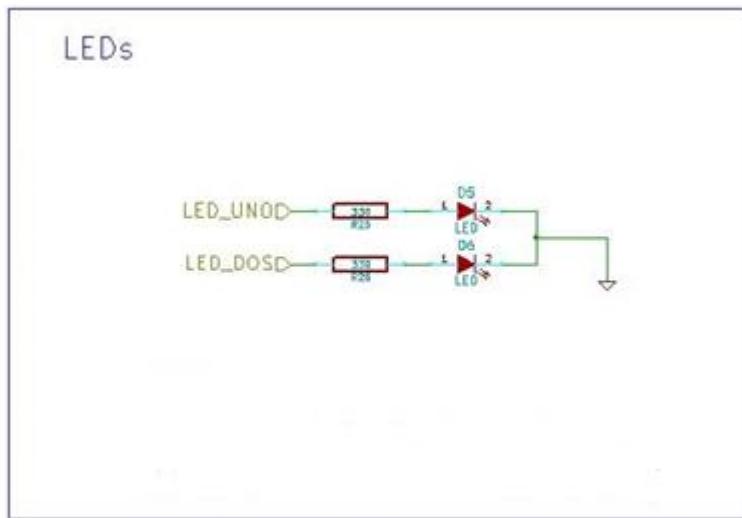


Figura 22: Leds instalados en la estación remota para depuración del sistema

3.4.9-Salidas digitales

Las salidas digitales se implementaron mediante relés comandados por transistores bipolares, estos transistores suministran la corriente necesaria para comandar las bobinas de los relés sin sobrecargar el puerto del microcontrolador, estos transistores trabajan en corte y saturación, en el emisor se conectó un led con su resistencia limitadora para indicar el estado de la válvula en el caso de la estación remota y en la estación concentradora indican el estado de la bomba, este led se montó sobre la placa de circuito impreso siendo inaccesible una vez instalada la placa dentro del gabinete y sirve para fines de mantenimiento.

La estación concentradora y la remota poseen dos de estas salidas digitales cada una, la primera posee la capacidad de comandar dos bombas y la segunda, la capacidad de comandar dos válvulas.

En la figura 23 se puede observar el circuito implementado para el comando del relé.

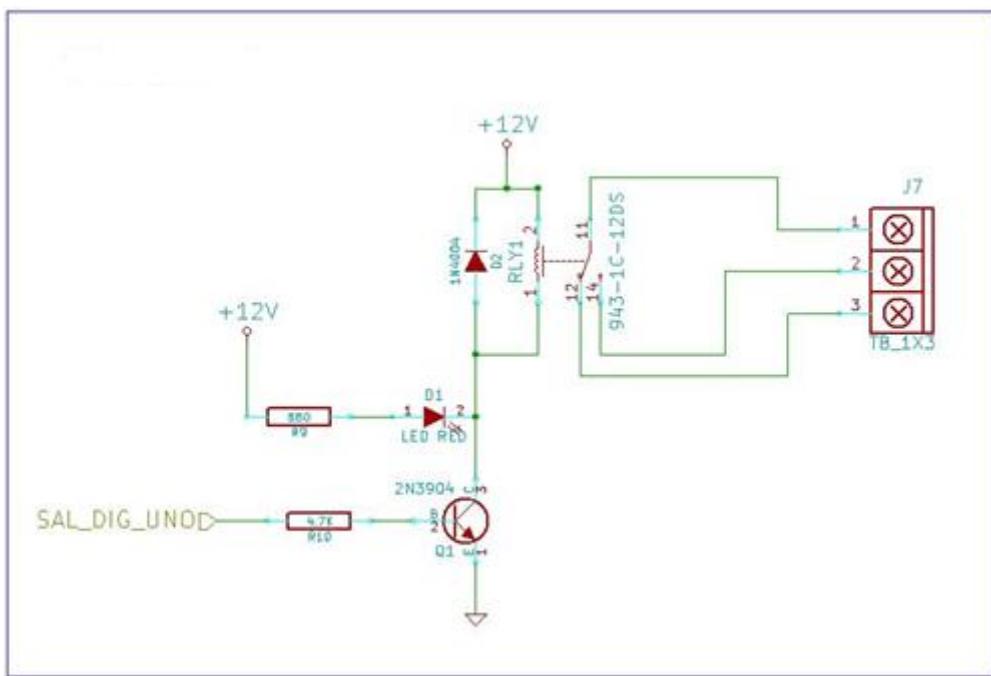


Figura 23: Circuito esquemático de comando del relé

3.4.10-Teclado analógico

Este teclado analógico es recomendado al igual que la multiplexación de los leds usada en la estación concentradora en la nota técnica AN234 de Microchip [8].

En la figura 24 se puede observar el teclado implementado.

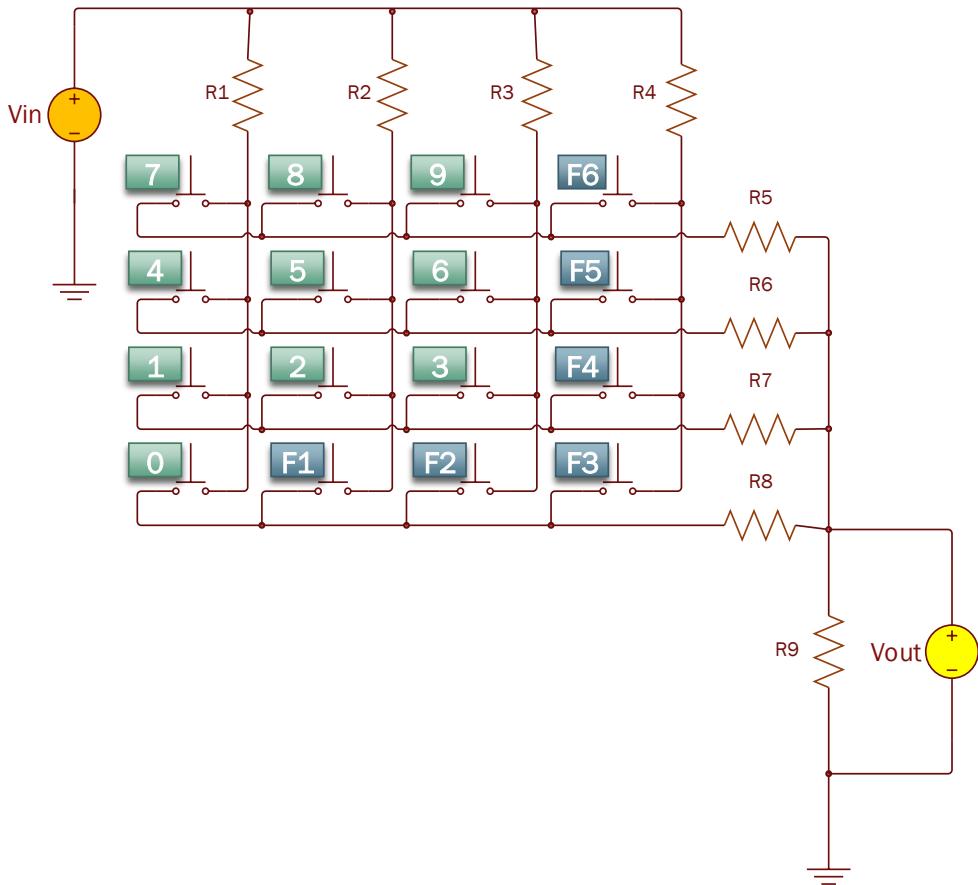


Figura 24: Circuito esquemático del teclado analógico implementado

Este tipo de configuración tiene la ventaja que usa un solo pin del microcontrolador para comandar el teclado, con un teclado controlado de forma digital se necesitarían ocho líneas de comando.

En la tabla 2 se observan los valores de las resistencias utilizadas en el teclado analógico.

Numero de resistencia	Valor de resistencia
R1	0 Ω
R2	470 Ω
R3	1 K
R4	2.2 K
R5	220 Ω
R6	330 Ω
R7	470 Ω
R8	560 Ω
R9	1.2 K

Tabla 2: Resistencias utilizadas en el teclado analógico

En la tabla 3 se puede observar el símbolo de cada tecla y la fila y columna a la cual pertenece.

Fila	Columna			
	uno	dos	tres	cuatro
uno	7	8	9	F6
dos	4	5	6	F5
tres	1	2	3	F4
cuatro	0	F1	F2	F3

Tabla 3: Símbolos de cada tecla según fila y columna

Al presionar cada tecla se produce un divisor de tensión, por ejemplo para la tecla que al ser presionada une la fila cuatro con la columna cuatro, se puede observar en la figura 25 el circuito generado.

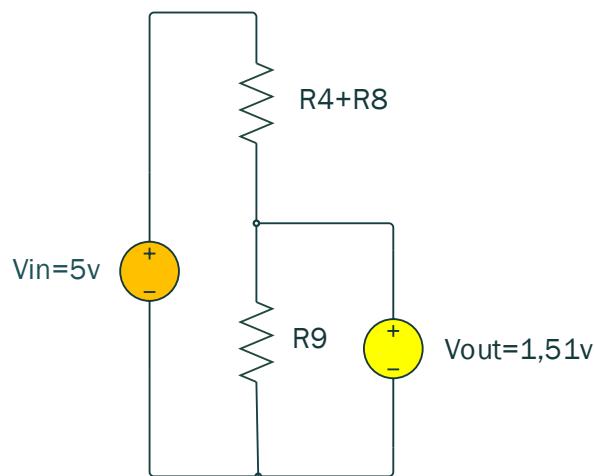


Figura 25: Divisor de tensión formado al presionar la tecla F3

$$V_{out} = \frac{R9}{R4 + R8 + R9} \times V_{in} = \frac{1200\Omega}{2200\Omega + 560\Omega + 1200\Omega} \times 5v = 1,51v$$

En la tabla 4 se volcaron los valores de tensión obtenidos para cada tecla al ser presionada.

Fila	Columna			
	uno	dos	tres	cuatro
uno	4,22V	3,14V	2,47V	1,65V
dos	3,92V	3,00V	2,37V	1,60V
tres	3,59V	2,80V	2,24V	1,55V
cuatro	3,40V	2,69V	2,17V	1,51V

Tabla 4: Valores de tensión obtenidos al presionar cada tecla

Luego de calcular los valores de tensión anteriores se transformaron teniendo en cuenta que se utilizó un conversor analógico digital de 10 bits.

Fila	Columna			
	uno	dos	tres	cuatro
uno	863	642	505	337
dos	802	613	484	327
tres	735	572	458	317
cuatro	695	550	443	308

Tabla 5: Valores leídos por el conversor A/D de diez bits teóricos

Los valores obtenidos en la tabla 5 son teóricos, en la práctica no se cumplieron, se utilizó un teclado de membrana como se muestra en la fotografía 3.



Fotografía 3: Fotografía del teclado de membrana utilizado

Este tipo de teclado presenta resistividad tanto al presionar una tecla como en las conexiones por este motivo se diseñó un software con el cual se leyó el valor analógico que entregaba cada tecla y se volcó en la tabla 6, estos son los valores prácticos utilizados.

Fila	Columna			
	uno	dos	tres	cuatro
uno	181	497	633	335
dos	128	477	601	326
tres	63	454	563	315
cuatro	27	438	540	308

Tabla 6: Valores leídos por el conversor A/D de diez bits prácticos

El circuito esquemático de la figura 26 muestra las resistencias y el conector utilizado.

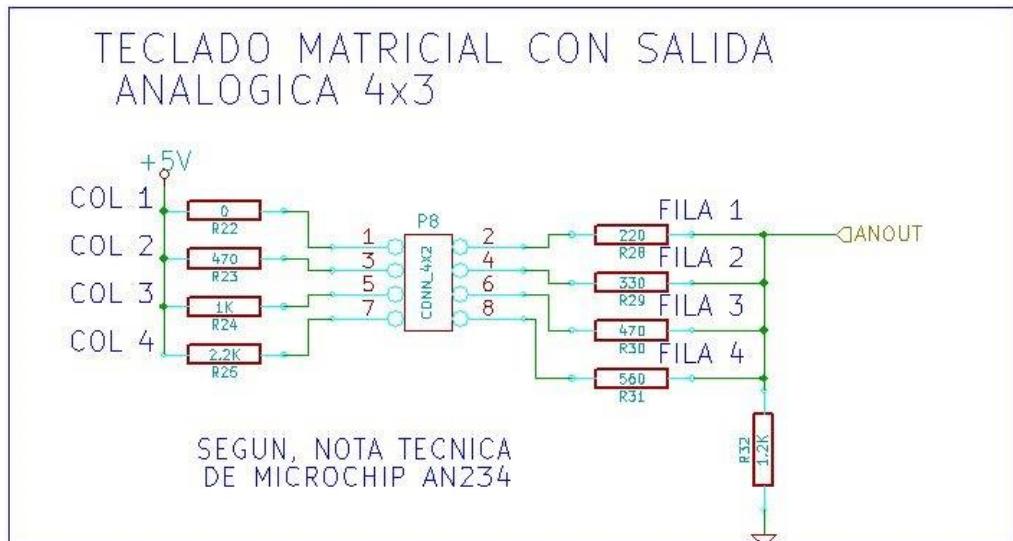


Figura 26: Esquemático del conector utilizado para el teclado analógico en la estación concentradora

En las pruebas de laboratorio se observaron oscilaciones en la entrada del conversor analógico digital producido al presionar las teclas, esto es debido al contacto mecánico de la membrana, este contacto produce el fenómeno conocido como rebote, para atenuar este efecto se modificó la placa de circuito impreso agregando a la salida del teclado una resistencia de 100 ohm y luego de esta un capacitor de 100nF a masa, como se puede observar en la figura 27 , esta configuración de resistencia y capacitor se denomina filtro pasa bajo, el capacitor ante las oscilaciones de alta frecuencia se comporta como un conductor de baja resistividad suprimiendo la oscilación.

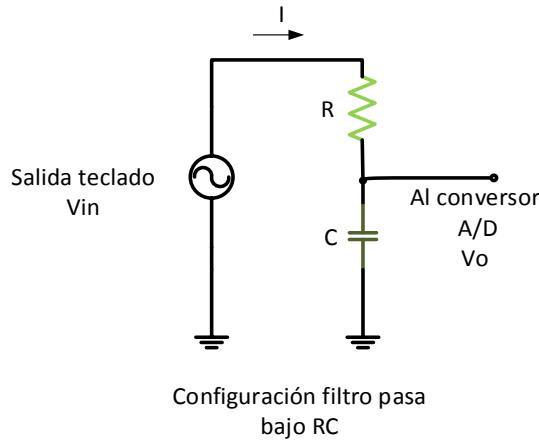


Figura 27: Filtro pasa bajo RC

La frecuencia de corte del filtro es aquella a la cual la ganancia del circuito toma el valor $1/\sqrt{2}$ del valor máximo, cuando la ganancia se ha reducido al 70% y la tensión de salida V_o tiene una amplitud 0.707 veces la de la señal de entrada V_{in} , para el filtro implementado, usando una resistencia de 100 ohm y un capacitor cerámico de 100nF la frecuencia de corte se calcula de la siguiente forma:

$$f_c = \frac{1}{\sqrt{2\pi RC}} = \frac{1}{\sqrt{2 \cdot \pi \cdot 100\text{ohm} \cdot 100E - 9F}} = 126\text{Hz}$$

Con esta frecuencia de corte se atenuaron las oscilaciones a la entrada del conversor analógico digital obteniendo lecturas estables en el tiempo.

3.4.11-Memoria EEPROM

En el circuito esquemático de la figura 28 se puede observar la memoria EEPROM utilizada, con dos capacitores, uno electrolítico de 100uf y otro cerámico de 100nf utilizados para filtrar ruido eléctrico proveniente de la línea de alimentación, es una memoria serie de tipo I2C.

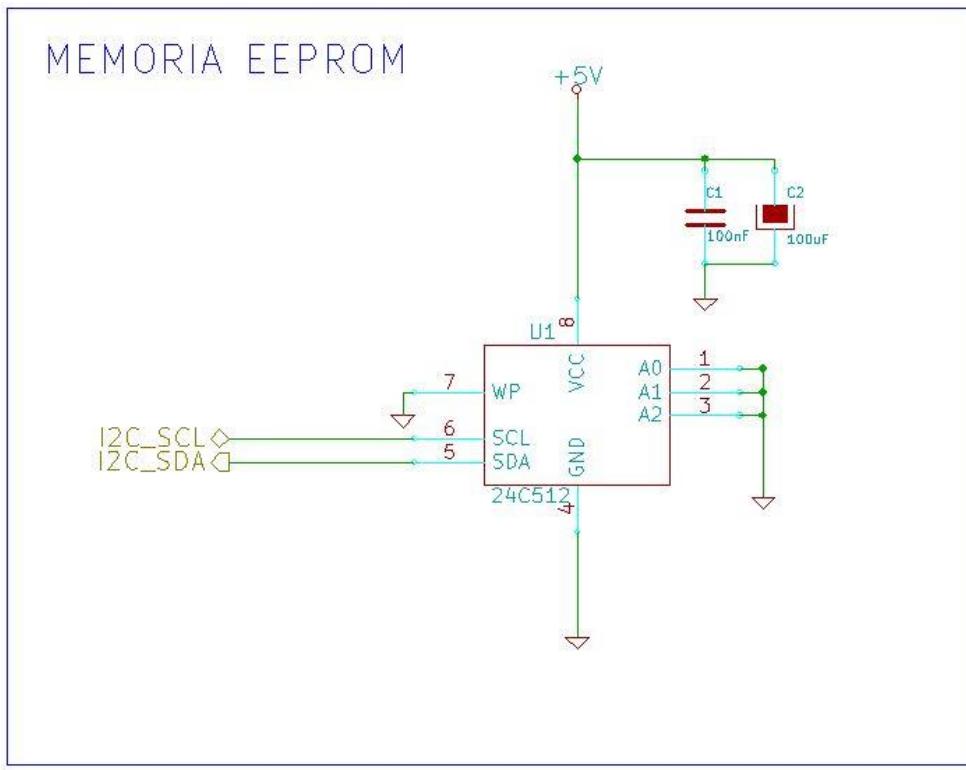


Figura 28: Circuito esquemático de la memoria EEPROM instalada en la estación concentradora

3.4.12-Fuente de alimentación

La fuente de alimentación se construyó en una placa separada y una para cada estación, en todas las estaciones se utilizó el mismo circuito impreso, pero la alimentación de la estación concentradora es de 12V de corriente alterna por lo cual ésta posee el puente rectificador, luego la salida de 12V de corriente continua rectificada de la estación concentradora a través del bus de comunicaciones provee alimentación a las remotas donde no se instaló el puente rectificador.

Esta fuente provee tensiones de salida de: 12V, 3.3V, 5V y una tensión ajustable entre 1.25V y 5V utilizada para fijar la tensión de alimentación de los sensores de humedad y como tensión de referencia del conversor analógico digital.

El esquemático siguiente es el empleado para la fuente de alimentación con el que se diseñó el circuito impreso, al momento de armarla no se consiguió el circuito integrado LM2596 y se sustituyó por el regulador LM7805 adaptando el circuito impreso.

La salida de 3.3V se implementó utilizando dos diodos polarizados de forma directa con 5V.

La salida de tensión ajustable la provee el circuito integrado LM117 que mediante un preset se puede regular la tensión de salida.

En la figura 29, se puede observar el circuito esquemático de la fuente de alimentación.

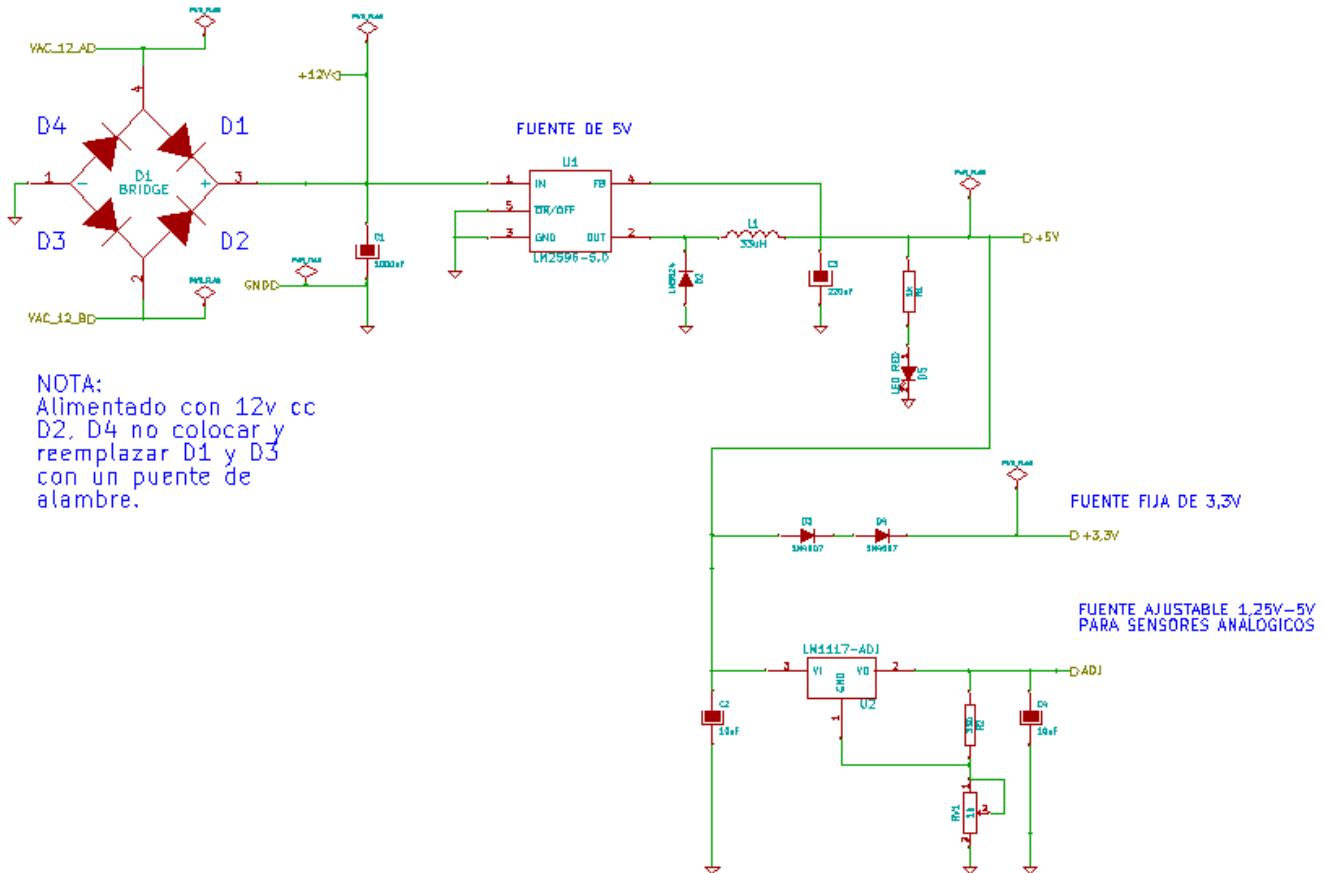


Figura 29: Circuito esquemático de la fuente de alimentación

3.5-CIRCUITOS IMPRESOS

Los esquemáticos y los circuitos impresos se diseñaron con el software KICAD, este entorno de desarrollo es software libre, los circuitos impresos se fabricaron en placa virgen doble faz habiéndose fabricado el prototipo de forma artesanal, el software de diseño usado entrega los archivos necesarios que permiten enviarlo a una empresa de fabricación de circuitos impresos y producir el sistema de riego en serie, a pesar de ser un prototipo y pensando en la posibilidad de diseñarlo para generar un producto comercial y con posibilidad de fabricación en serie se diseñó la placa de circuito impreso del tamaño más reducido posible sobre una placa doble faz, utilizando pistas de ancho 0,3 mm para señal y de 1 mm para alimentación en el anexo II se puede observar la máscara del circuito impreso obtenido, para la implementación del prototipo se utilizó un film fotosensible por permitir este método implementar pistas de reducido grosor, se

estudió todo el proceso necesario para utilizar este método de fabricación de circuitos impresos, luego se vio la necesidad de fabricar una insoladora, con este método se obtuvieron excelentes resultados.

3.6-FIRMWARE DEL SISTEMA DE RIEGO IMPLEMENTADO

El firmware a embeber en el microcontrolador se programó en lenguaje C, se utilizó en ambas estaciones, tanto la estación concentradora como la estación remota un sistema operativo de tiempo real RTOS.

3.6.1 ¿Qué es un RTOS?

Los sistemas embebidos cuentan con recursos muy limitados en comparación con una PC, por esto generalmente no tienen un sistema operativo completo, en algunos casos el sistema operativo no es más que un conjunto de funciones que se ejecutan solo en momentos determinados del programa.

Un RTOS es un conjunto de programas que ayudan al programador de aplicaciones a gestionar los recursos de hardware disponibles, entre ellos el tiempo del procesador y la memoria.

La gestión del tiempo del procesador permite al programador de aplicaciones escribir múltiples subprogramas como si cada uno fuera el único que utiliza la CPU.

Una parte del sistema operativo se encarga de asignar tiempo de ejecución a todos los programas que tiene cargados en base a un juego de reglas conocido de antemano. A estos subprogramas se los llama tareas. [9]

3.6.2-Estructura de los comandos enviados sobre el bus RS-485

En la tabla 7 y en la tabla 8 se indican los comandos implementados, éstos tienen la particularidad de ser de longitud variable terminada la trama con un carácter ‘&’ y cada campo es también de longitud variable separados por un carácter “+”, cada comando posee una cabecera formada por ocho dígitos binarios que indican el destino del comando en el caso de que lo envíe la estación concentradora o de la procedencia del comando en el caso de que lo envíe una estación remota, luego de estos ocho dígitos binarios le sigue un numero entero que indica el comando a transmitir luego la trama continua con un carácter “+”, en el caso de que el comando lo envíe la estación concentradora los dos campos siguientes separados por un carácter “+” corresponden a la hora y minutos actuales determinados por el reloj de tiempo reas DS1307 instalado en la estación concentradora, si bien las remotas poseen en su hardware un reloj de tiempo real se utilizó un solo reloj para sincronizar el sistema instalado en la estación

concentradora no utilizando los relojes instalados en las estaciones remotas para que no se produzcan inestabilidades en el sistema por la posible pérdida de sincronización de los relojes, por esto la estación concentradora transmite con todos sus comandos a través del bus RS-485 dos campos, uno con la hora y el siguiente con los minutos actuales, luego cada comando posee los campos necesarios según el comando implementado.

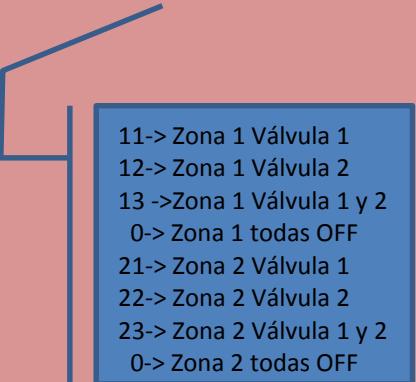
Comando	Función	Trama
1	La central envía los límites de humedad a la placa de campo	000000XX1 + hora + minutos + minima [tx_placa] + maxima [tx_placa] +chk&
2	La central envía los límites horarios a la placa de campo	000000XX2 + hora + minutos + hora_inicio[tx_placa] + minutos_inicio[tx_placa] + hora_fin[tx_placa] + minutos_fin[tx_placa]+chk &
3	Solicita valores de humedad a la placa de campo	000000XX3 + hora + minutos+chk &
4	Activa o desactiva de forma manual una válvula en una placa de campo	000000XX4 + hora + minutos + BB+chk&  <div style="border: 1px solid blue; padding: 5px;"> 11-> Zona 1 Válvula 1 12-> Zona 1 Válvula 2 13 ->Zona 1 Válvula 1 y 2 0-> Zona 1 todas OFF 21-> Zona 2 Válvula 1 22-> Zona 2 Válvula 2 23-> Zona 2 Válvula 1 y 2 0-> Zona 2 todas OFF </div>
8	Indaga estado de las válvulas	000000XX8+hora+minutos+chk&

Tabla 7: Comandos de la estación concentradora

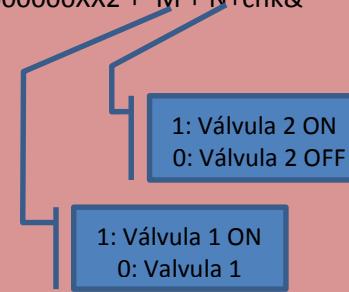
Comando	Función	Trama
1	Envía valores de humedad de los cuatro sensores de la estación remota	000000XX1 + valor_humedad[0] + valor_humedad[1] + valor_humedad[2] + valor_humedad[3]+ chk&
2	Informa estado: ON/OFF de las válvulas de la estación remota	000000XX2 + M+N+chk& 

Tabla 8: Comandos de la estación remota

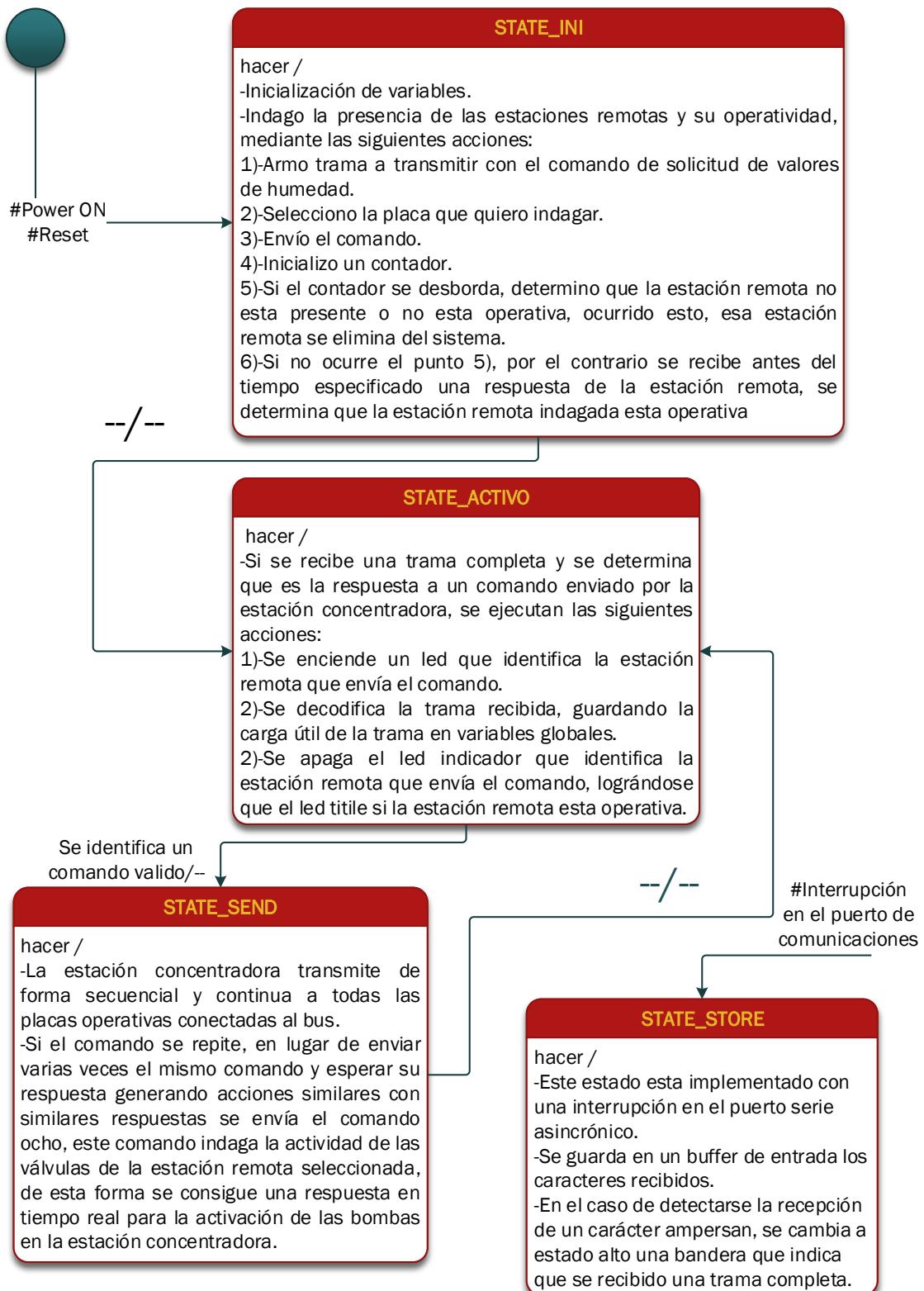
Comando enviado por la estación concentradora	Respuesta de la estación remota
1	1
2	2
3	1
4	2
8	2

Tabla 9: Comandos enviados por la estación concentradora y su respuesta por parte de la estación remota

3.6.3-Descripción de las máquinas de estado implementadas

Se utilizó un sistema operativo de tiempo real RTOS y cada tarea del sistema operativo se implementó mediante una máquina de estado.

3.6.4-Maquina de estado utilizada para la comunicación según el protocolo RS-485



Máquina de estado 1: Comunicación RS-485

Esta máquina de estado 1, ante un reset del microcontrolador o una energización del sistema se posiciona en el estado: STATE_INI, a este estado se accede únicamente cuando se producen las dos situaciones mencionadas anteriormente solamente,

estando en este estado se inicializan las variables a utilizar , luego en este proceso de inicialización se transmite a las placas conectadas al bus de forma secuencial y esperando acuse de recibo con un tiempo de espera preestablecido una trama a cada estación remota, con el fin de detectar la integridad del sistema y la posible ausencia intencional o fortuita de las placas remotas del sistema, el proceso mencionado es el siguiente:

- 1)-Se selecciona la placa que se quiere indagar su existencia o integridad en el bus.
- 2)-Se inicializa una variable: dato_completo=0, esta variable es utilizada para indicar si se recibió una trama completa en la estación concentradora, esta situación se produce cuando el último dato ASCII recibido es un Ampersan, al ocurrir esto la variable dato_completo toma el valor uno.
- 3)- Se arma una trama para ser enviada a la estación remota seleccionada, esta trama corresponde a un comando de solicitud de valores de humedad.
- 4)-Se envía la trama.
- 5)-Luego de enviada la trama, se genera un ciclo de espera y se inicia un conteo, pueden ocurrir dos cosas por las cuales se puede salir del ciclo: que se reciba un dato, es decir que la estación remota acuse recibo de la solicitud de humedad y a su vez transmita los valores de humedad a la estación concentradora o que el contador se desborde, si ocurre la primera situación, se determina que la estación remota está conectada al bus y en estado operativa, si ocurre la segunda situación se determina que la estación remota no contesto en el tiempo máximo estipulado, porque no está presente o no está operativa en cuyo caso no se establece más comunicación con ella y para el sistema no existe esta estación remota ajustando los menús de programación de forma tal de no tener acceso y no operar con esta placa .

Este proceso se repite para todas las estaciones remotas del bus, luego de este proceso la máquina de estado cambia al estado: STATE_ACTIVO.

El estado: STATE_ACTIVO, es donde se encuentra habitualmente la máquina de estados, en este se controla si se recibió una trama completa y en el caso de ocurrir esto a que estación remota pertenece, detectando esta ocurrencia se decodifica la trama y se enciende un led indicador de identificación de la procedencia de la trama recibida, aplicando este método el operador de la estación concentradora puede determinar de forma fehaciente la integridad del sistema, ante la ausencia del parpadeo del led que le

correspondiente a una cualquiera de las estaciones remotas el operador puede determinar que esa estación remota no está operativa.

Si ocurre la detección de la trama y luego de desencadenarse los procesos antes mencionados la máquina de estado produce una transición hacia el estado: STATE_SEND.

El estado: STATE_SEND es el encargado de transmitir la trama a la estación remota seleccionada, esta selección se determina de forma ordenada, esto es: en primera instancia se comunica la estación concentradora con una de las estaciones remotas esperando la respuesta, luego la próxima comunicación se produce con otra de las estaciones conectadas al bus y así de forma consecutiva y de forma continua, con cada una de las estaciones conectadas al bus , cada una de estas estaciones están numeradas con un dígito entero y de estación en estación con dígitos consecutivos, los comandos a enviar se cargan en un buffer de salida, cada comando se envía y se espera una respuesta, permanentemente se están enviando comandos y esperando su respuesta, pero no se envían comandos repetidos, es decir: en un instante t se envía un comando pero en el instante t+1 si no se necesita cambiar de comando en vez de repetir el comando enviado en el instante t, se envía el comando ocho hasta que se necesite enviar otro comando, este comando indaga el estado de las válvulas, es necesaria esta comunicación con esa frecuencia alta para detectar el momento en que se produce la apertura de alguna de las válvulas en una o varias de las estaciones remotas, ocurriendo esto la central de comandos activa una o varias bombas según la cantidad de las válvulas activas.

Ante una interrupción debido a la ocurrencia de la recepción de un carácter por el bus RS-485 se ingresa a un estado denominado STATE_STORE, este estado almacena la trama recibida carácter a carácter en un vector hasta que por el bus llega un carácter ampersan cuando ocurre esto se señaliza en una bandera que la trama se recibió completa, las tramas enviadas y recibidas y los campos que componen estas tramas son de longitud variable.

El código siguiente es la implementación en lenguaje C de la máquina de estado diseñada:

```
void maquina_de_estado_de_comunicacion()
{
    switch (estado_siguiente){
        case STATE_INI:
        {
            minima[1]=read_EEPROM (minima_placa_1_e);
            maxima[1]=read_EEPROM (maxima_placa_1_e);
```

```

minima[2]=read_EEPROM (minima_placa_2_e);
maxima[2]=read_EEPROM (maxima_placa_2_e);
for(i=0;i<=2;i++)
{
    salida_1[i]=0;
    salida_2[i]=0;
    salida_3[i]=0;
    salida_4[i]=0;
    sube_1[i]=0;
    sube_2[i]=0;
    sube_3[i]=0;
    sube_4[i]=0;
    baja_1[i]=0;
    baja_2[i]=0;
    baja_3[i]=0;
    baja_4[i]=0;
}
veces_registro=0;
hora_inicio[1] =0;
hora_inicio[2] =0;
minutos_inicio[1] =0;
minutos_inicio[2] =0;
hora_fin[1] =0;
hora_fin[2] =0;
minutos_fin[1] =0;
minutos_fin[2] =0;
manual_anterior[1]=44;
manual_anterior[2]=44;
manual[1]=44;
manual[2]=44;
valor_humedad_uno[1]=1;
valor_humedad_uno[2]=1;
valor_humedad_dos[1]=1;
valor_humedad_dos[2]=1;
valor_humedad_tres[1]=1;
valor_humedad_tres[2]=1;
valor_humedad_cuatro[1]=1;
valor_humedad_cuatro[2]=1;
comando_tx=1;

tx_placa=2;
dato_completo=0;
armar_trama();
envio_dato();
while(dato_completo==0)
{
    if (espera_placa_dos!=max_espera) espera_placa_dos=(espera_placa_dos+1);
    if (espera_placa_dos==max_espera) dato_completo=1;
    delay_ms(10);
}

comando_tx=1;
tx_placa=1;
dato_completo=0;
armar_trama();
envio_dato();
while(dato_completo==0)
{
    if (espera_placa_uno!=max_espera) espera_placa_uno=(espera_placa_uno+1);
    if (espera_placa_uno==max_espera) dato_completo=1;
    delay_ms(10);
}
estado_siguiente=STATE_ACTIVO;
break;
}
case STATE_SEND:
{

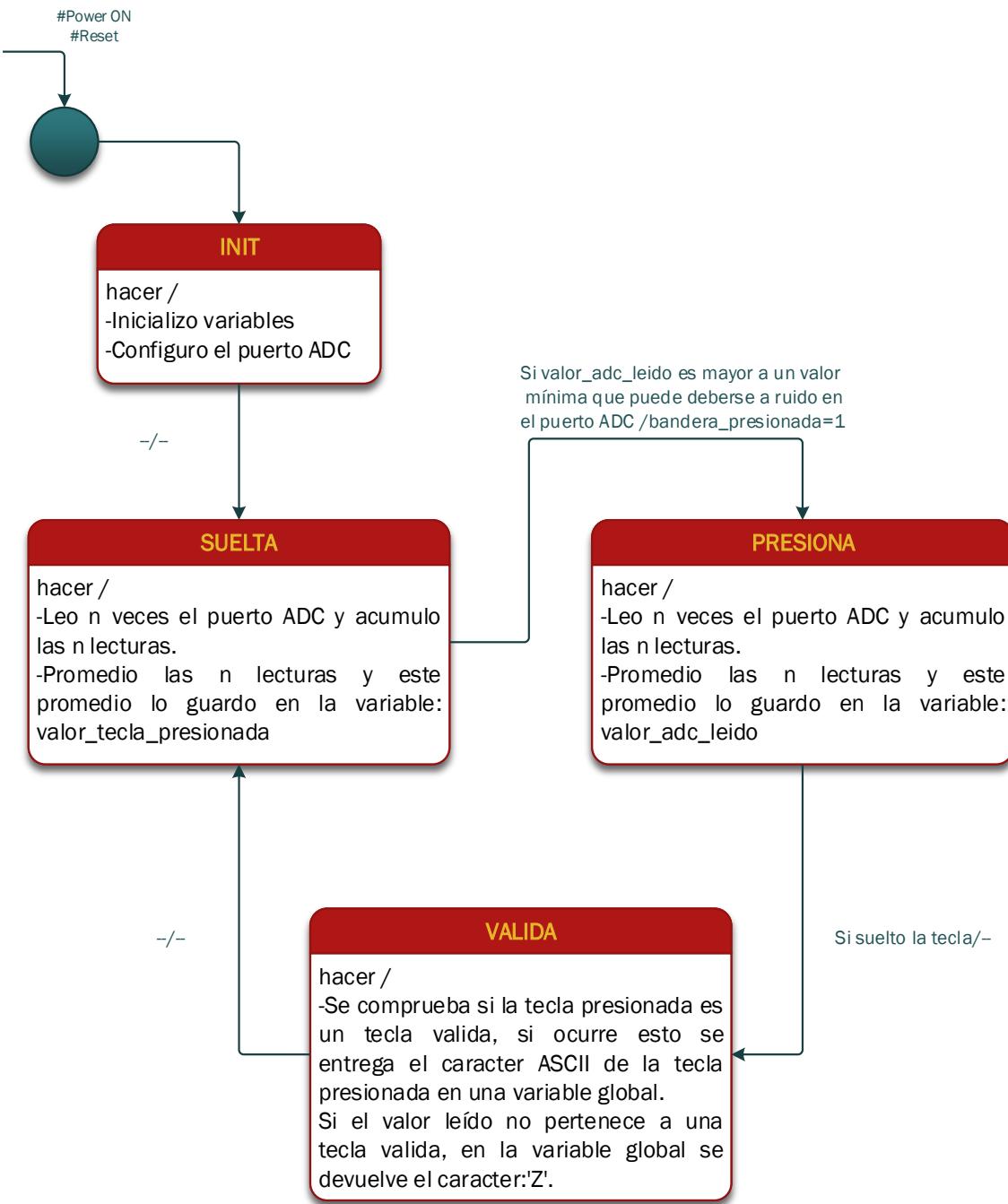
```

```

if(( tx_placa==1)) tx_placa=2; else tx_placa=1;
if ((comando_tx==3)&& (comando_rx==1)) comando_tx=8;
armar_trama();
envio_dato();
if ((comando_tx_anterior==comando_tx)&&(tx_placa_anterior!=tx_placa))comando_tx=8;
comando_tx_anterior=comando_tx;
tx_placa_anterior=tx_placa;
estado_siguiente=STATE_ACTIVO;
break;
}
case STATE_ACTIVO:
{
if ( ((dato_rx[0]=='0') && (dato_rx[1]=='0')&& (dato_rx[2]=='0') && (dato_rx[3]=='0')&&
(dato_rx[4]=='0') && (dato_rx[5]=='0')&& (dato_rx[6]=='0')&&(dato_rx[7]=='1')&&(dato_completo==1)) ||
(((dato_rx[0]=='0') && (dato_rx[1]=='0')&& (dato_rx[2]=='0') && (dato_rx[3]=='0')&&
(dato_rx[4]=='0') && (dato_rx[5]=='0')&& (dato_rx[6]=='1')&&(dato_rx[7]=='0')&&(dato_completo==1))) )
{
deco_tramaRx();
dato_completo=0;
if ((tx_placa==1)&& (espera_placa_uno!=max_espera))encender_led(1);
if ((tx_placa==2)&&(espera_placa_dos!=max_espera))encender_led(2);
estado_siguiente=STATE_SEND;
}
if ((espera_placa_uno==max_espera)&&(tx_placa==1))
{
estado_siguiente=STATE_SEND;
dato_completo=0;encender_led(0);
}
if((espera_placa_dos==max_espera)&&(tx_placa==2))
{
estado_siguiente=STATE_SEND;
dato_completo=0;encender_led(0);
}
break;
}
}
}

```

3.6.5-Maquina de estado utilizada en el teclado analógico



Máquina de estado 2: Teclado analógico

El código siguiente es la implementación en lenguaje C de la máquina de estado 2 diseñada:

```
//////////////////////////////TECLADO ANALOGICO///////////////////////////
int16 valor_adc_leido=0;
int16 valor_adc_tecla_presionada;
char tecla;
int bandera_presionada,i;
```

```

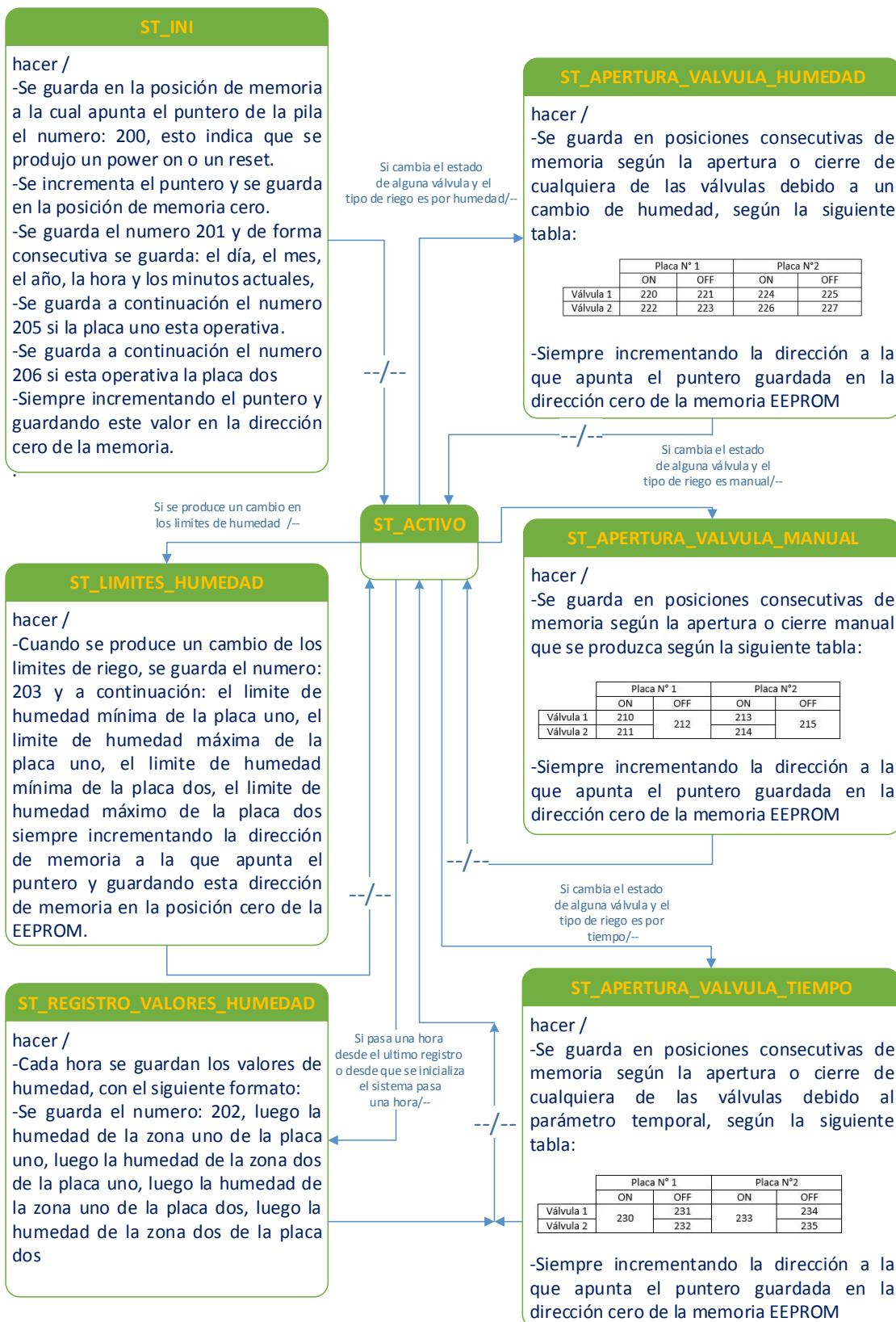
//const int16
valores_adc_teclas[17]={0,308,317,327,337,443,458,484,505,550,572,613,642,695,735,802,863}//valores teoricos
const int16
valores_adc_teclas[17]={0,308,315,326,335,540,563,601,633,438,454,477,497,27,63,126,181}//valores practicos
const char simbolo_tecla[17]={'Z','C','D','E','F','B','3','6','9','A','2','5','8','0','1','4','7'};
const int16 tolerancia=5;
int max_tecla=18;
const int INIT = 0;
const int SUELTA = 1;
const int VALIDA = 2;
const int PRESIONA = 3;
int estado_siguiente_maquina_teclado=0;

void tarea_teclado_analogico(void){
switch(estado_siguiente_maquina_teclado)
{
case INIT:
int i,max=16;
setup_adc_ports(AN0|VSS_VDD);
setup_adc(ADC_CLOCK_DIV_8);
enable_interrupts(global);
bandera_presionada=0;
estado_siguiente_maquina_teclado=SUELTA;
break;
case SUELTA:
valor_adc_leido=0;
set_adc_channel(0); //habilitacion canal 0
delay_ms(2);
valor_adc_leido =0;
for(i=1;i<=100;i++)
{
valor_adc_leido =read_adc()+valor_adc_leido;
if (i==100)valor_adc_leido =(valor_adc_leido/i);
}
max=16;
if((valor_adc_leido>10)&&(bandera_presionada==0))
{
bandera_presionada=1;
estado_siguiente_maquina_teclado=PRESIONA;
valor_adc_tecla_presionada=valor_adc_leido;
}
break;
case PRESIONA:
set_adc_channel(0); //habilitacion canal 0
delay_ms(2);
valor_adc_leido =0;
for(i=1;i<=100;i++)
{
valor_adc_leido =read_adc()+valor_adc_leido;
if (i==100)valor_adc_leido =(valor_adc_leido/i);
}
if((bandera_presionada==1)&&(valor_adc_leido<10))
estado_siguiente_maquina_teclado=VALIDA;
break;
case VALIDA:
max=max_tecla;
for(i=0;i<=max;i++)
{
if((valor_adc_tecla_presionada>(valores_adc_teclas[i]-tolerancia)) &&
(valor_adc_tecla_presionada<(valores_adc_teclas[i]+tolerancia)))
{
max=i;
tecla=simbolo_tecla[i];
}
}
}
}

```

```
if (i>=max_tecla) tecla='Z';
bandera_presionada=0;
estado_siguiente_maquina_teclado=SUELTA;
break;
}
}
```

3.6.6-Maquina de estado para elaborar los reportes de actividades del sistema



Máquina de estado 3: Reportes de actividades

El código siguiente es la implementación en lenguaje C de la máquina de estado 3 diseñada:

```

void maquina_de_estado_reportes()
{
    switch(estado_siguiente_maquina_reportes){
        case ST_INI:
            if (leer_MEM(0)<1022)
            {
                guardar_MEM(200,(int)leer_MEM(0)+1);
                guardar_MEM(200,(int)leer_MEM(0)+1);
                guardar_MEM((int)leer_MEM(0)+1,0);
                guardar_MEM(201,(int)leer_MEM(0)+1);
                guardar_MEM((int)leer_MEM(0)+1,0);
                guardar_MEM((byte)dia,(int)leer_MEM(0)+1);
                guardar_MEM((int)leer_MEM(0)+1,0);
                guardar_MEM((byte)mes,(int)leer_MEM(0)+1);
                guardar_MEM((int)leer_MEM(0)+1,0);
                guardar_MEM((byte)anio,(int)leer_MEM(0)+1);
                guardar_MEM((int)leer_MEM(0)+1,0);
                guardar_MEM((byte)hora,(int)leer_MEM(0)+1);
                guardar_MEM((int)leer_MEM(0)+1,0);
                guardar_MEM((byte)minutos,(int)leer_MEM(0)+1);
                guardar_MEM((int)leer_MEM(0)+1,0);
                if ((espera_placa_uno!=max_espera))
                {
                    guardar_MEM(204,(int)leer_MEM(0)+1);
                    guardar_MEM((int)leer_MEM(0)+1,0);
                    inicializacion_1=1;
                }
                if ((espera_placa_dos!=max_espera))
                {
                    guardar_MEM(205,(int)leer_MEM(0)+1);
                    guardar_MEM((int)leer_MEM(0)+1,0);
                    inicializacion_2=1;
                }
                if ((espera_placa_uno==max_espera)&&(espera_placa_dos==max_espera))
                {
                    guardar_MEM(206,(int)leer_MEM(0)+1);
                    guardar_MEM((int)leer_MEM(0)+1,0);inicializacion_1=1;inicializacion_2=1;
                }
                estado_siguiente_maquina_reportes=ST_LIMITES_HUMEDAD;
                veces_registro=0;
            }
            break;
        case ST_ACTIVO:
            if (((((estado_valvula_1_placa_1!=estado_valvula_1_placa_1_anterior)
            ||(estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior))&&
            ((tipo_riego_placa_1==2)&&(manual[1]!=
            manual_anterior[1]))|||
            ((estado_valvula_1_placa_2!=estado_valvula_1_placa_2_anterior)|||(estado_valvula_2_placa_2!=estado_v
            alvula_2_placa_2_anterior))&&
            ((tipo_riego_placa_2==2)&&(manual[2]!=
            manual_anterior[2]))))&&(leer_MEM(0)<1022))
                estado_siguiente_maquina_reportes=ST_APERTURA_VALVULA_MANUAL;

            if (((((estado_valvula_1_placa_1!=estado_valvula_1_placa_1_anterior)
            ||(estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior))&&(tipo_riego_placa_1==1))|||
            ((estado_valvula_1_placa_2!=estado_valvula_1_placa_2_anterior)|||(estado_valvula_2_placa_2!=estado_
            valvula_2_placa_2_anterior))&&(tipo_riego_placa_2==1))&&
            (leer_MEM(0)<1022))//&&(manual[1]==45)&&(manual[2]==45)&&(manual[1]==46)&&(manual[2]==46)
            estado_siguiente_maquina_reportes=ST_APERTURA_VALVULA_HUMEDAD;
    }
}

```

```

if
(((minutos==0)&&(veces_registro==0))&&(leer_MEM(0)<1022))estado_siguiente_maquina_reportes=ST_
REGISTRO_VALORES_HUMEDAD;
if (minutos!=0) veces_registro=0;
if
(((minima[1]!=minima_anterior[1])||(minima[2]!=minima_anterior[2])||(maxima[1]!=maxima_anterior[1])||(ma
xima[2]!=maxima_anterior[2])) &&(leer_MEM(0)<1022))
    estado_siguiente_maquina_reportes=ST_LIMITES_HUMEDAD;

if (((((estado_valvula_1_placa_1!=estado_valvula_1_placa_1_anterior)
||(estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior))&&
((tipo_riego_placa_1==3)))|
(
(estado_valvula_1_placa_2!=estado_valvula_1_placa_2_anterior)||((estado_valvula_2_placa_2!=estado_v
alvula_2_placa_2_anterior))&&
(
(tipo_riego_placa_2==3)))&&(leer_MEM(0)<1022))
    estado_siguiente_maquina_reportes=ST_APERTURA_VALVULA TIEMPO;
break;
case ST_LIMITES_HUMEDAD:
guardar_MEM(203,(int)leer_MEM(0)+1);
guardar_MEM((int)leer_MEM(0)+1,0);
if ((espera_placa_uno!=max_espera))
{
    guardar_MEM((byte)minima[1],(int)leer_MEM(0)+1);
    guardar_MEM((int)leer_MEM(0)+1,0);
    guardar_MEM((byte)maxima[1],(int)leer_MEM(0)+1);
    guardar_MEM((int)leer_MEM(0)+1,0);
}
if ((espera_placa_dos!=max_espera))
{
    guardar_MEM((byte)minima[2],(int)leer_MEM(0)+1);
    guardar_MEM((int)leer_MEM(0)+1,0);
    guardar_MEM((byte)maxima[2],(int)leer_MEM(0)+1);
    guardar_MEM((int)leer_MEM(0)+1,0);
}
minima_anterior[1]=minima[1];
minima_anterior[2]=minima[2];
maxima_anterior[1]=maxima[1];
maxima_anterior[2]=maxima[2];
estado_siguiente_maquina_reportes=ST_REGISTRO_VALORES_HUMEDAD;
break;
case ST_APERTURA_VALVULA TIEMPO:

if(((estado_valvula_1_placa_1==1)&&(estado_valvula_1_placa_1!=estado_valvula_1_placa_1_anterior)))
||
((estado_valvula_2_placa_1==1)&&(estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior))
{
    guardar_MEM(230,(int)leer_MEM(0)+1);
    guardar_MEM((int)leer_MEM(0)+1,0);
    estado_valvula_1_placa_1_anterior=estado_valvula_1_placa_1;
    estado_valvula_2_placa_1_anterior=estado_valvula_2_placa_1;
}

if(((estado_valvula_1_placa_2==1)&&(estado_valvula_1_placa_2!=estado_valvula_1_placa_2_anterior)))
|
(
((estado_valvula_2_placa_2==1)&&(estado_valvula_2_placa_2!=estado_valvula_2_placa_2_anterior))
{
    guardar_MEM(233,(int)leer_MEM(0)+1);
    guardar_MEM((int)leer_MEM(0)+1,0);
    estado_valvula_1_placa_2_anterior=estado_valvula_1_placa_2;
    estado_valvula_2_placa_2_anterior=estado_valvula_2_placa_2;
}

if(((estado_valvula_1_placa_1==0)&&(estado_valvula_1_placa_1!=estado_valvula_1_placa_1_anterior)))

```

```

{
    guardar_MEM(231,(int)leer_MEM(0)+1);
    guardar_MEM((int)leer_MEM(0)+1,0);
    estado_valvula_1_placa_1_anterior=estado_valvula_1_placa_1;
}
if
(((estado_valvula_2_placa_1==0)&&(estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior))
{
    guardar_MEM(232,(int)leer_MEM(0)+1);
    guardar_MEM((int)leer_MEM(0)+1,0);
    estado_valvula_2_placa_1_anterior=estado_valvula_2_placa_1;
}
if((estado_valvula_1_placa_2==0)&&(estado_valvula_1_placa_2!=estado_valvula_1_placa_2_anterior))
{
    guardar_MEM(234,(int)leer_MEM(0)+1);
    guardar_MEM((int)leer_MEM(0)+1,0);
    estado_valvula_1_placa_2_anterior=estado_valvula_1_placa_2;
}
if ((estado_valvula_2_placa_2==0)&&(estado_valvula_2_placa_2!=estado_valvula_2_placa_2_anterior))
{
    guardar_MEM(235,(int)leer_MEM(0)+1);
    guardar_MEM((int)leer_MEM(0)+1,0);
    estado_valvula_2_placa_2_anterior=estado_valvula_2_placa_2;
}
guardar_MEM((byte)hora,(int)leer_MEM(0)+1);
guardar_MEM((int)leer_MEM(0)+1,0);
guardar_MEM((byte)minutos,(int)leer_MEM(0)+1);
guardar_MEM((int)leer_MEM(0)+1,0);
estado_siguiente_maquina_reportes=ST_ACTIVO;
break;
case ST_REGISTRO_VALORES_HUMEDAD:
if ((valor_humedad_uno[1]!=0)&&(valor_humedad_dos[1]!=0))
humedad_valvula_uno_placa_uno=((valor_humedad_uno[1]+valor_humedad_dos[1])/2);
else if(valor_humedad_uno[1]==0) humedad_valvula_uno_placa_uno=valor_humedad_dos[1];
else if(valor_humedad_dos[1]==0) humedad_valvula_uno_placa_uno=valor_humedad_uno[1];

if ((valor_humedad_tres[1]!=0)&&(valor_humedad_cuatro[1]!=0))
humedad_valvula_dos_placa_uno=((valor_humedad_tres[1]+valor_humedad_cuatro[1])/2);
else if(valor_humedad_tres[1]==0) humedad_valvula_dos_placa_uno=valor_humedad_cuatro[1];
else if(valor_humedad_cuatro[1]==0) humedad_valvula_dos_placa_uno=valor_humedad_tres[1];

if ((valor_humedad_uno[2]!=0)&&(valor_humedad_dos[2]!=0))
humedad_valvula_uno_placa_dos=((valor_humedad_uno[2]+valor_humedad_dos[2])/2);
else if(valor_humedad_uno[2]==0) humedad_valvula_uno_placa_dos=valor_humedad_dos[2];
else if(valor_humedad_dos[2]==0) humedad_valvula_uno_placa_dos=valor_humedad_uno[2];

if ((valor_humedad_tres[2]!=0)&&(valor_humedad_cuatro[2]!=0))
humedad_valvula_dos_placa_dos=((valor_humedad_tres[2]+valor_humedad_cuatro[2])/2);
else if(valor_humedad_tres[2]==0) humedad_valvula_dos_placa_dos=valor_humedad_cuatro[2];
else if(valor_humedad_cuatro[2]==0) humedad_valvula_dos_placa_dos=valor_humedad_tres[2];
guardar_MEM(202,(int)leer_MEM(0)+1);
guardar_MEM((int)leer_MEM(0)+1,0);
if ((espera_placa_uno!=max_espera))
{
    guardar_MEM((byte)humedad_valvula_uno_placa_uno,(int)leer_MEM(0)+1);
    guardar_MEM((int)leer_MEM(0)+1,0);
    guardar_MEM((byte)humedad_valvula_dos_placa_uno,(int)leer_MEM(0)+1);
    guardar_MEM((int)leer_MEM(0)+1,0);
}
if ((espera_placa_dos!=max_espera))
{
    guardar_MEM((byte) humedad_valvula_uno_placa_dos,(int)leer_MEM(0)+1);
    guardar_MEM((int)leer_MEM(0)+1,0);
    guardar_MEM((byte)humedad_valvula_dos_placa_dos,(int)leer_MEM(0)+1);
    guardar_MEM((int)leer_MEM(0)+1,0);
}
estado_siguiente_maquina_reportes=ST_APERTURA_VALVULA_HUMEDAD;

```

```

veces_registro=1;
break;
case ST_APERTURA_VALVULA_HUMEDAD:
if((estado_valvula_1_placa_1==1)
&&((estado_valvula_1_placa_1!=estado_valvula_1_placa_1_anterior)|| ( inicializacion_1==1)))
{
guardar_MEM(220,(int)leer_MEM(0)+1);
guardar_MEM((int)leer_MEM(0)+1,0);
estado_valvula_1_placa_1_anterior=estado_valvula_1_placa_1;
}
if((estado_valvula_1_placa_1==0)
&&((estado_valvula_1_placa_1!=estado_valvula_1_placa_1_anterior)|| ( inicializacion_1==1)))
{
guardar_MEM(221,(int)leer_MEM(0)+1);
guardar_MEM((int)leer_MEM(0)+1,0);
estado_valvula_1_placa_1_anterior=estado_valvula_1_placa_1;
}
if((estado_valvula_2_placa_1==1)
&&((estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior)|| ( inicializacion_1==1)))
{
guardar_MEM(222,(int)leer_MEM(0)+1);
guardar_MEM((int)leer_MEM(0)+1,0);
estado_valvula_2_placa_1_anterior=estado_valvula_2_placa_1;
}

if((estado_valvula_2_placa_1==0)&&((estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior)|| ( inicializacion_1==1)))
{
guardar_MEM(223,(int)leer_MEM(0)+1);
guardar_MEM((int)leer_MEM(0)+1,0);
estado_valvula_2_placa_1_anterior=estado_valvula_2_placa_1;
}

if((estado_valvula_1_placa_2==1)&&((estado_valvula_1_placa_2!=estado_valvula_1_placa_2_anterior)|| ( inicializacion_2==1)))
{
guardar_MEM(224,(int)leer_MEM(0)+1);
guardar_MEM((int)leer_MEM(0)+1,0);
estado_valvula_1_placa_2_anterior=estado_valvula_1_placa_2;
}

if((estado_valvula_1_placa_2==0)&&((estado_valvula_1_placa_2!=estado_valvula_1_placa_2_anterior)|| ( inicializacion_2==1)))
{
guardar_MEM(225,(int)leer_MEM(0)+1);
guardar_MEM((int)leer_MEM(0)+1,0);
estado_valvula_1_placa_2_anterior=estado_valvula_1_placa_2;
}

if((estado_valvula_2_placa_2==1)&&((estado_valvula_2_placa_2!=estado_valvula_2_placa_2_anterior)|| ( inicializacion_2==1)))
{
guardar_MEM(226,(int)leer_MEM(0)+1);
guardar_MEM((int)leer_MEM(0)+1,0);
estado_valvula_2_placa_2_anterior=estado_valvula_2_placa_2;
}

if((estado_valvula_2_placa_2==0)&&((estado_valvula_2_placa_2!=estado_valvula_2_placa_2_anterior)|| ( inicializacion_2==1)))
{
guardar_MEM(227,(int)leer_MEM(0)+1);
guardar_MEM((int)leer_MEM(0)+1,0);
estado_valvula_2_placa_2_anterior=estado_valvula_2_placa_2;
}

guardar_MEM((byte)hora,(int)leer_MEM(0)+1);
guardar_MEM((int)leer_MEM(0)+1,0);
guardar_MEM((byte)minutos,(int)leer_MEM(0)+1);

```

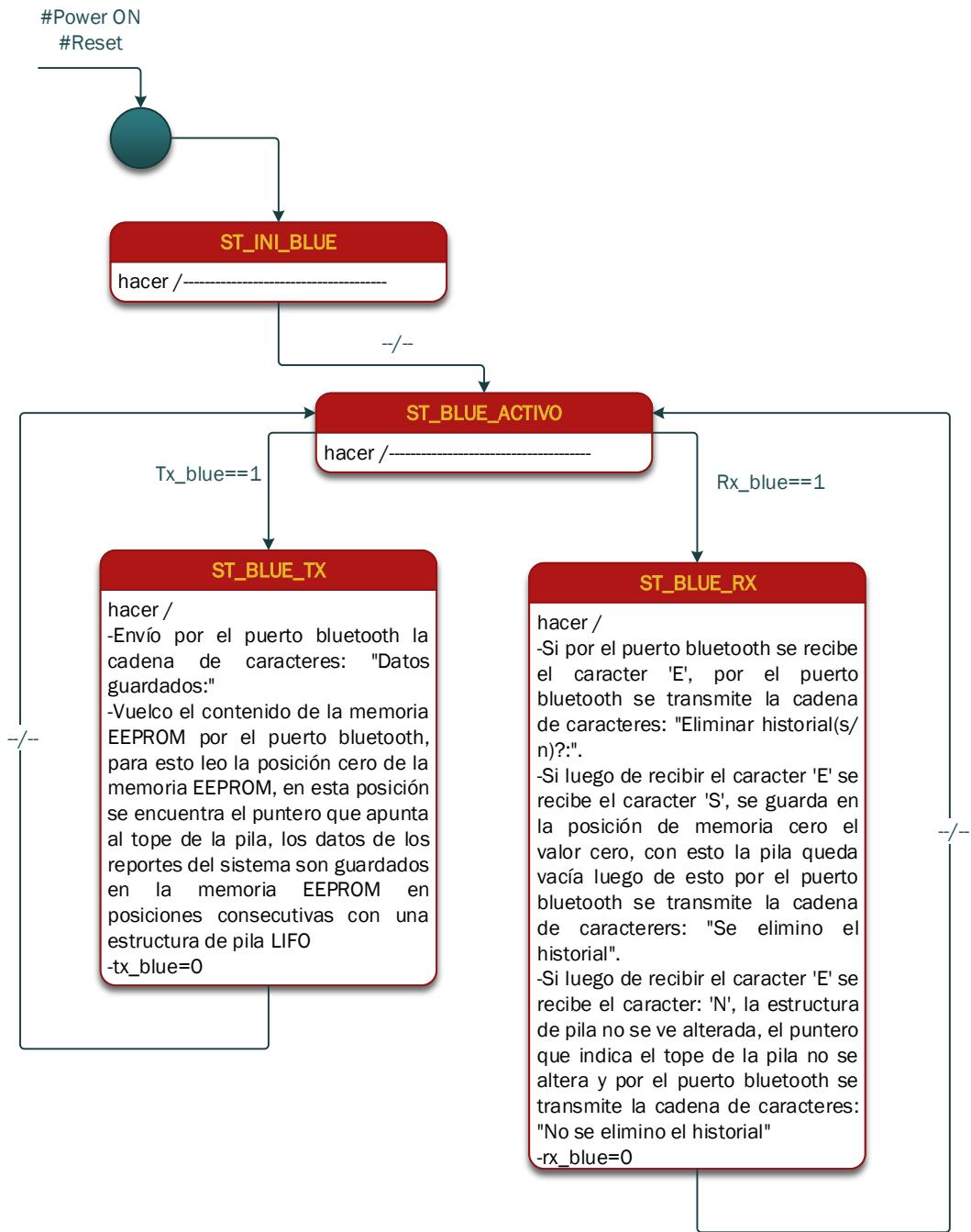
```

guardar_MEM((int)leer_MEM(0)+1,0);
 inicializacion_1=0;
 inicializacion_2=0;
 estado_siguiente_maquina_reportes=ST_ACTIVO;
break;
case ST_APERTURA_VALVULA_MANUAL:
 if ((estado_valvula_1_placa_1!=estado_valvula_1_placa_1_anterior)
||(estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior))
{
 if (manual[1]==11)
 {
 guardar_MEM(210,(int)leer_MEM(0)+1);
 guardar_MEM((int)leer_MEM(0)+1,0);
 }
 if (manual[1]==12)
 {
 guardar_MEM(211,(int)leer_MEM(0)+1);
 guardar_MEM((int)leer_MEM(0)+1,0);
 }
 if (((((manual[1]==44))||((manual[1]==46)))
{
 guardar_MEM(212,(int)leer_MEM(0)+1);
 guardar_MEM((int)leer_MEM(0)+1,0);
 }
 if ((manual[1]==13)&&(manual_anterior[1]==11))
{
 guardar_MEM(211,(int)leer_MEM(0)+1);
 guardar_MEM((int)leer_MEM(0)+1,0);
 }
 manual_anterior[1]=manual[1];
 }
 if
((estado_valvula_1_placa_2!=estado_valvula_1_placa_2_anterior)||((estado_valvula_2_placa_2!=estado_valvula_2_placa_2_anterior))
{
 if ((manual[1]==23)&&(manual_anterior[1]==12))
{
 guardar_MEM(210,(int)leer_MEM(0)+1);
 guardar_MEM((int)leer_MEM(0)+1,0);
 }
 if (manual[2]==21)
{
 guardar_MEM(213,(int)leer_MEM(0)+1);
 guardar_MEM((int)leer_MEM(0)+1,0);
 }
 if (manual[2]==22)
{
 guardar_MEM(214,(int)leer_MEM(0)+1);
 guardar_MEM((int)leer_MEM(0)+1,0);
 }
 if (((manual[2]==44))||((manual[2]==46)))
{
 guardar_MEM(215,(int)leer_MEM(0)+1);
 guardar_MEM((int)leer_MEM(0)+1,0);
 }
 if ((manual[2]==23)&&(manual_anterior[2]==21))
{
 guardar_MEM(214,(int)leer_MEM(0)+1);
 guardar_MEM((int)leer_MEM(0)+1,0);
 }
 if ((manual[2]==23)&&(manual_anterior[2]==22))
{
 guardar_MEM(213,(int)leer_MEM(0)+1);
 guardar_MEM((int)leer_MEM(0)+1,0);
 }
 manual_anterior[2]=manual[2];
}

```

```
guardar_MEM((byte)hora,(int)leer_MEM(0)+1);
guardar_MEM((int)leer_MEM(0)+1,0);
guardar_MEM((byte)minutos,(int)leer_MEM(0)+1);
guardar_MEM((int)leer_MEM(0)+1,0);
estado_siguiente_maquina_reportes=ST_ACTIVO;
break;
}
}
```

3.6.7-Maquina de estado para la comunicación bluetooth de la estación concentradora



Máquina de estado 4: Comunicación bluetooth

El código siguiente es la implementación en lenguaje C de la máquina de estado 4 diseñada:

```

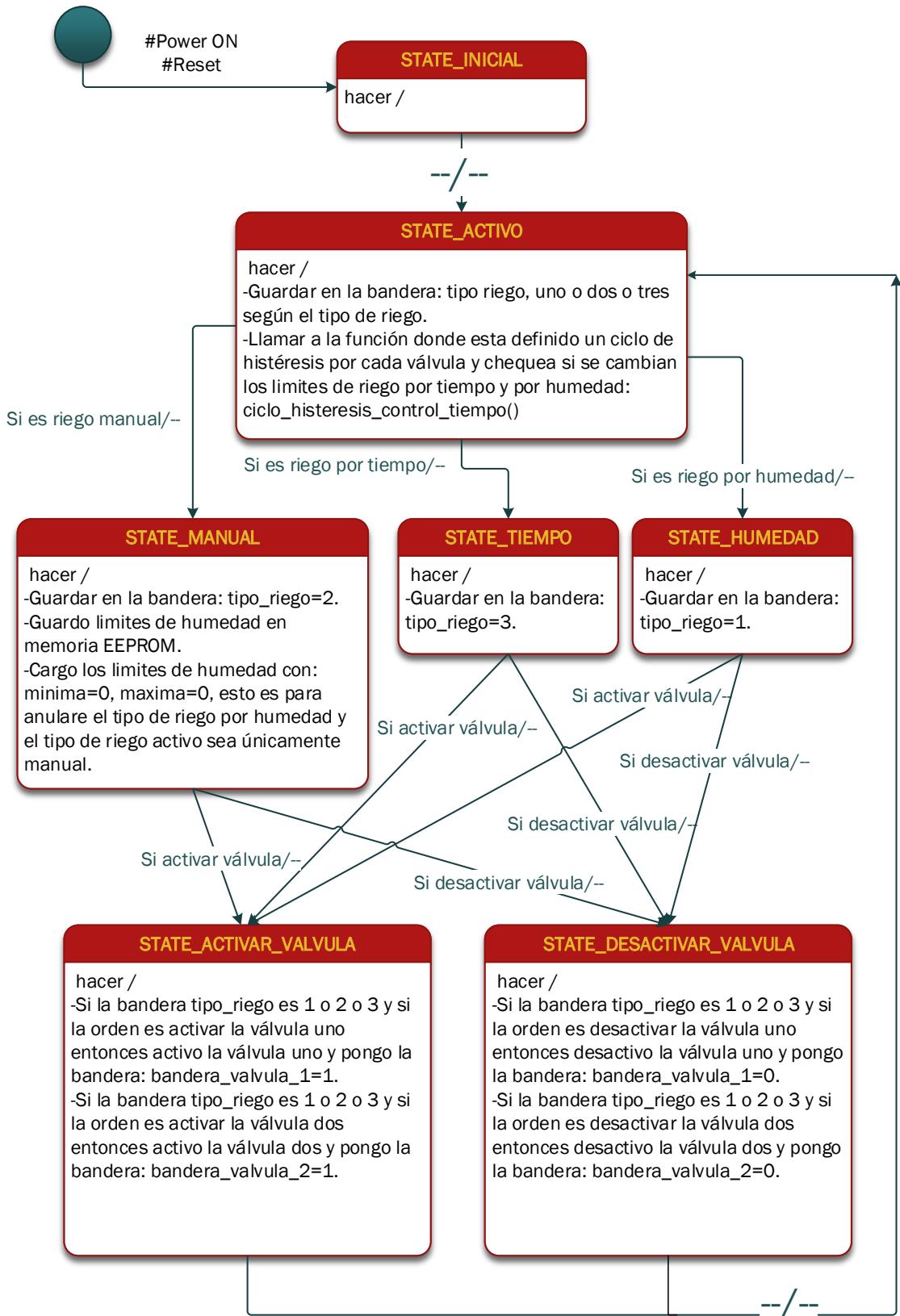
void maquina_de_estado_bluetooth()
{
switch (estado_siguiente_maquina_blue){
case ST_INI_BLUE:
    estado_siguiente_maquina_blue=ST_BLUE_ACTIVO;
break;
  
```

```

case ST_BLUE_ACTIVO:
    if (tx_blue==1) estado_siguiente_maquina_blue=ST_BLUE_TX;
    if (rx_blue==1)estado_siguiente_maquina_blue=ST_BLUE_RX;
break;
case ST_BLUE_TX:
    fprintf(PORT2,"Datos guardados:\n\r");
    for(i=0;i<=(int)leer_MEM(0);i++)
    {
        delay_us(250);
        itoa((int)leer_MEM(i),10, string);
        fprintf(PORT2, string);
        fprintf(PORT2, "\r");
        if ((int)leer_MEM(i+1)>100) fprintf(PORT2, "\n\r");
    }
    fprintf(PORT2, "\n\r");
    tx_blue=0;
    estado_siguiente_maquina_blue=ST_BLUE_ACTIVO;
break;
case ST_BLUE_RX:
    if (caracter_rx_blue=='E')
    {
        caracter_anterior_rx_blue='E';
        fprintf(PORT2,"Eliminar historial(s/n)?:\n");
    }
    if ((caracter_anterior_rx_blue=='E')&&(caracter_rx_blue=='N'))
    {
        caracter_anterior_rx_blue='k';
        fprintf(PORT2,"El historial no se elimino\n\r");
    }
    if ((caracter_anterior_rx_blue=='E')&&(caracter_rx_blue=='S'))
    {
        guardar_MEM(0,0);
        fprintf(PORT2,"Se elimino el historial\n\r");
        estado_siguiente_maquina_reportes =ST_INI;
    }
    rx_blue=0;
    estado_siguiente_maquina_blue=ST_BLUE_ACTIVO;
break;
}
}

```

3.6.8-Maquina de estado de la aplicación en las estaciones remotas



Máquina de estado 5: Aplicación en las estaciones remotas

El código siguiente es la implementación en lenguaje C de la máquina de estado 5 diseñada:

```

void maquina_de_estado_de_aplicacion()
{
    switch (estado_siguiente_riego){
        case STATE_INICIAL:
            estado_siguiente_riego=STATE_ACT;
            break;
        case STATE_ACT :
            ciclo_histeresis_control_tiempo();
            if ((regando_t!=regando_t_anterior)) estado_siguiente_riego=STATE_TIEMPO;
            if
((manual==11)|| (manual==13)|| (manual==12)|| (manual==44)|| (manual==45)&&(entro_manual==0))||((man
ual==46)&&(entro_manual==1))) estado_siguiente_riego=STATE_MANUAL;

            if((regando_1!=regando_1_anterior)|| (regando_2!=regando_2_anterior))estado_siguiente_riego=STATE_H
UMEDAD;
            regando_1_anterior=regando_1;
            regando_2_anterior=regando_2;
            regando_t_anterior=regando_t;
            break;
        case STATE_MANUAL:
            if ((manual==11)|| (manual==13)|| (manual==12)) estado_siguiente_riego=STATE_ACTIVAR_VALVULA;
            if ((manual==44)) estado_siguiente_riego=STATE_DESACTIVAR_VALVULA;
            if ((manual==45)&&(entro_manual==0))
            {
                minima_guardada=minima;
                maxima_guardada=maxima;
                minima=0;
                maxima=0;
                entro_manual=1;
                estado_siguiente_riego=STATE_DESACTIVAR_VALVULA;
            }
            if ((manual==46))
            {
                minima=minima_guardada;
                maxima=maxima_guardada;
                estado_siguiente_riego=STATE_DESACTIVAR_VALVULA;
                entro_manual=0;
            }
            tipo_riego=2;
            tipo_riego_anterior=tipo_riego;
            break;
        case STATE_TIEMPO:
            if (regando_t==1) estado_siguiente_riego=STATE_ACTIVAR_VALVULA;
            if (regando_t==0) estado_siguiente_riego=STATE_DESACTIVAR_VALVULA;
            tipo_riego=3;
            break;
        case STATE_HUMEDAD:
            if((regando_1==1)|| (regando_2==1)) estado_siguiente_riego=STATE_ACTIVAR_VALVULA;

            if((regando_1==0)&&(regando_2==0))estado_siguiente_riego=STATE_DESACTIVAR_VALVULA;//aca&&
            if ( tipo_riego_anterior==2) tipo_riego=2; else tipo_riego=1;
            tipo_riego_anterior=1;
            break;
        case STATE_ACTIVAR_VALVULA:
            if((regando_1==1)|| (regando_t==1)|| (manual==11)|| (manual==13))
            {
                activar_rele(PIN_C1);
                estado_valvula_1_placa_1=1;
            }
            if ((regando_2==1)|| (regando_t==1)|| (manual==12)|| (manual==13))
            {
                activar_rele(PIN_C2);
                estado_valvula_2_placa_1=1;
            }
    }
}

```

```

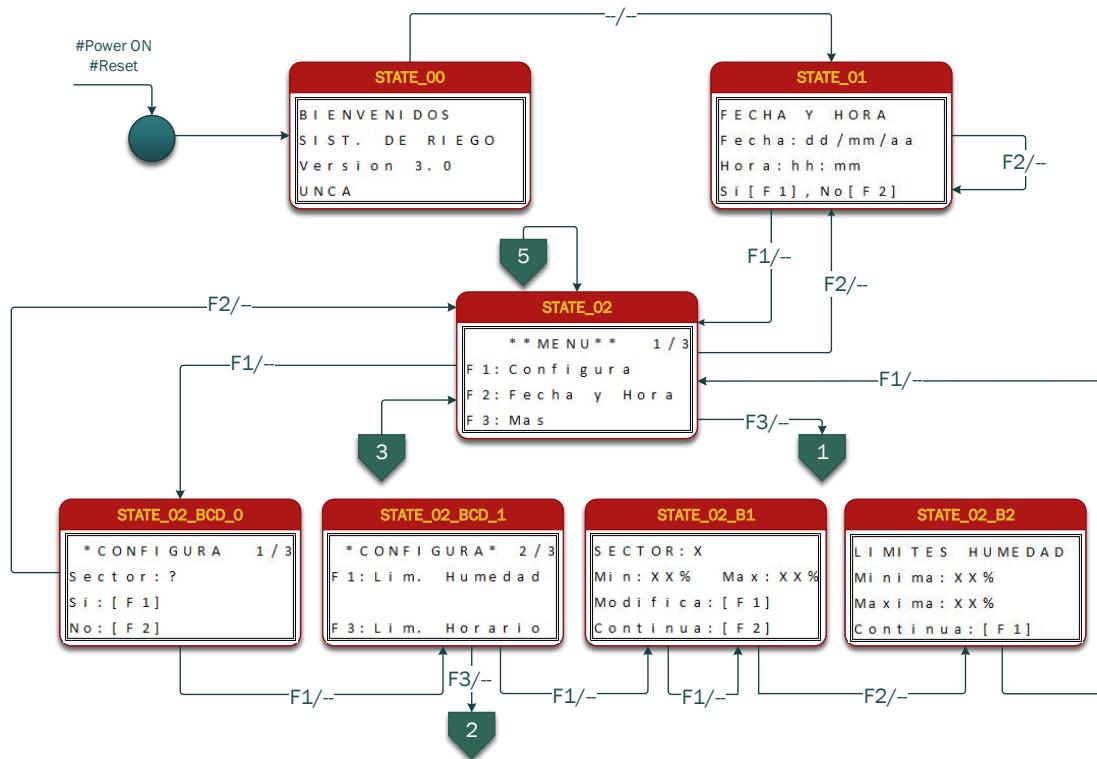
}
if (tipo_riego==1) estado_siguiente_riego=STATE_DESACTIVAR_VALVULA; else
estado_siguiente_riego=STATE_ACT;
break;
case STATE_DESACTIVAR_VALVULA:
if(((regando_1!=1)&&(regando_t!=1)&&(manual!=11)&&(manual!=13))||(manual==44)||((manual==45)))
{
desactivar_rele(PIN_C1);
estado_valvula_1_placa_1=0;
}
if (((regando_2!=1)&&(regando_t!=1)&&(manual!=12)&&(manual!=13))||(manual==44)||((manual==45)))
{
desactivar_rele(PIN_C2);
estado_valvula_2_placa_1=0;
}
estado_siguiente_riego=STATE_ACT;

break;
}

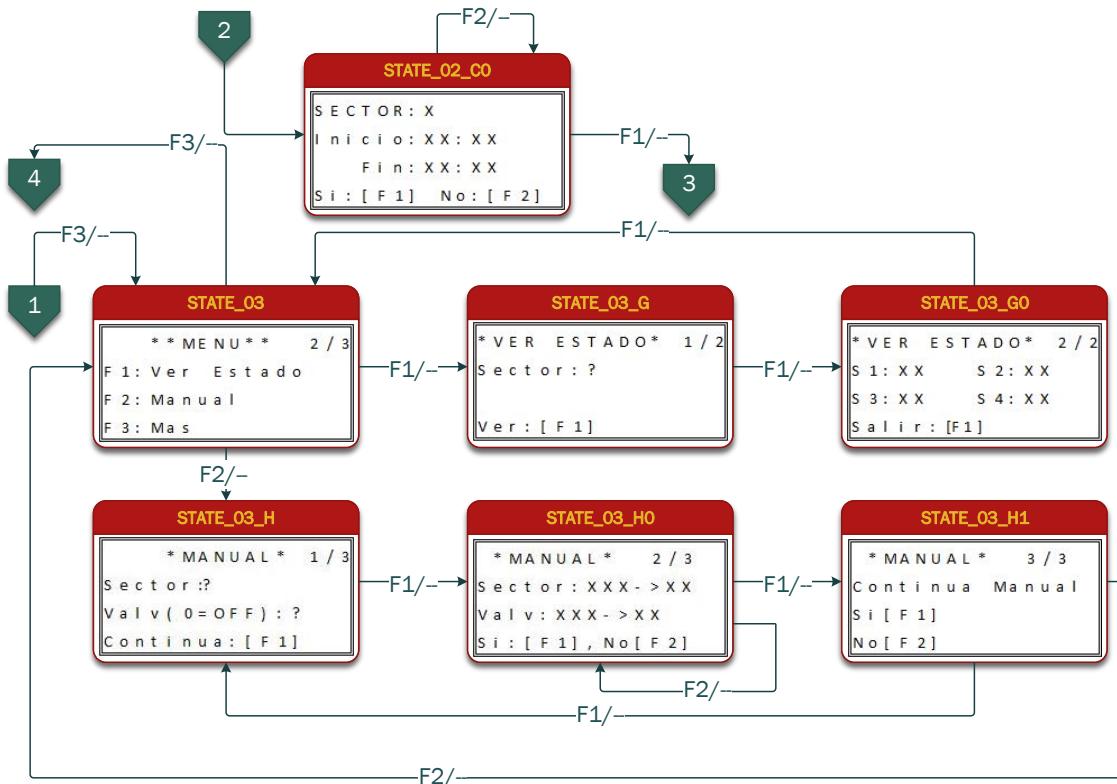
```

3.6.9- Maquina de estado utilizada para implementar las pantallas de información de salida del sistema en la estación concentradora

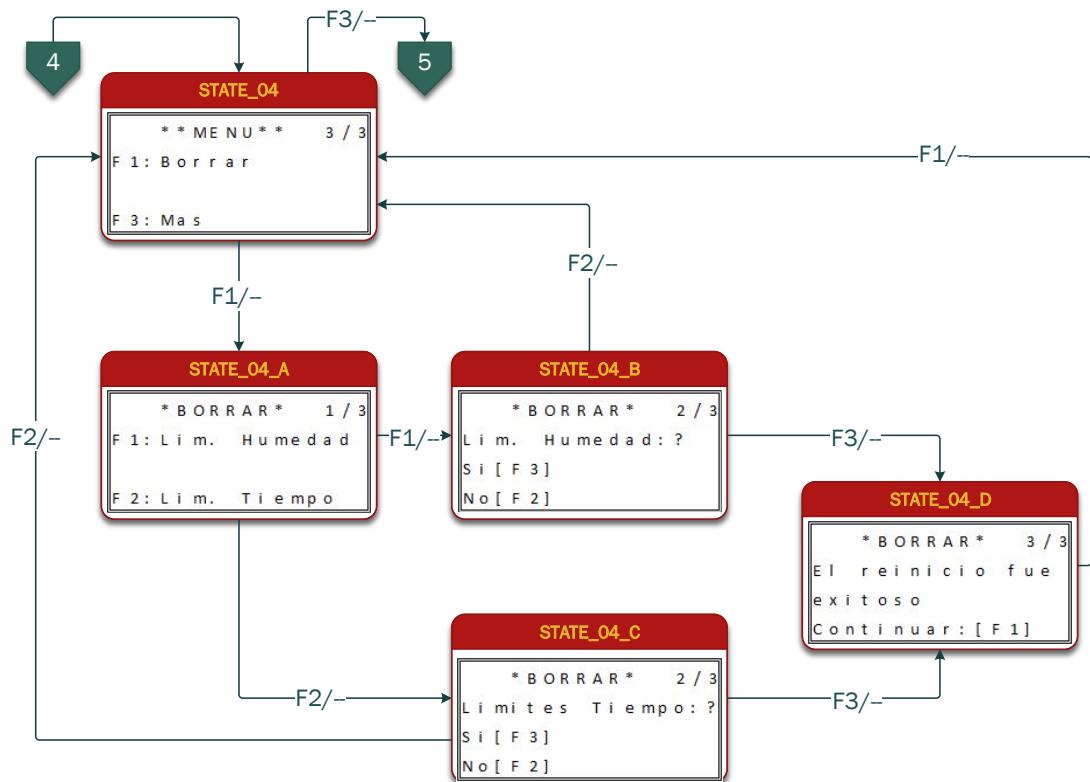
Las pantallas de visualización de la estación concentradora se programaron con una máquina de estado, donde cada estado corresponde a una pantalla, las transiciones se dan según una tecla ingresada por el teclado analógico.



Máquina de estado 6: Pantallas de información de salida del sistema



Máquina de estado 6: Pantallas de información de salida del sistema



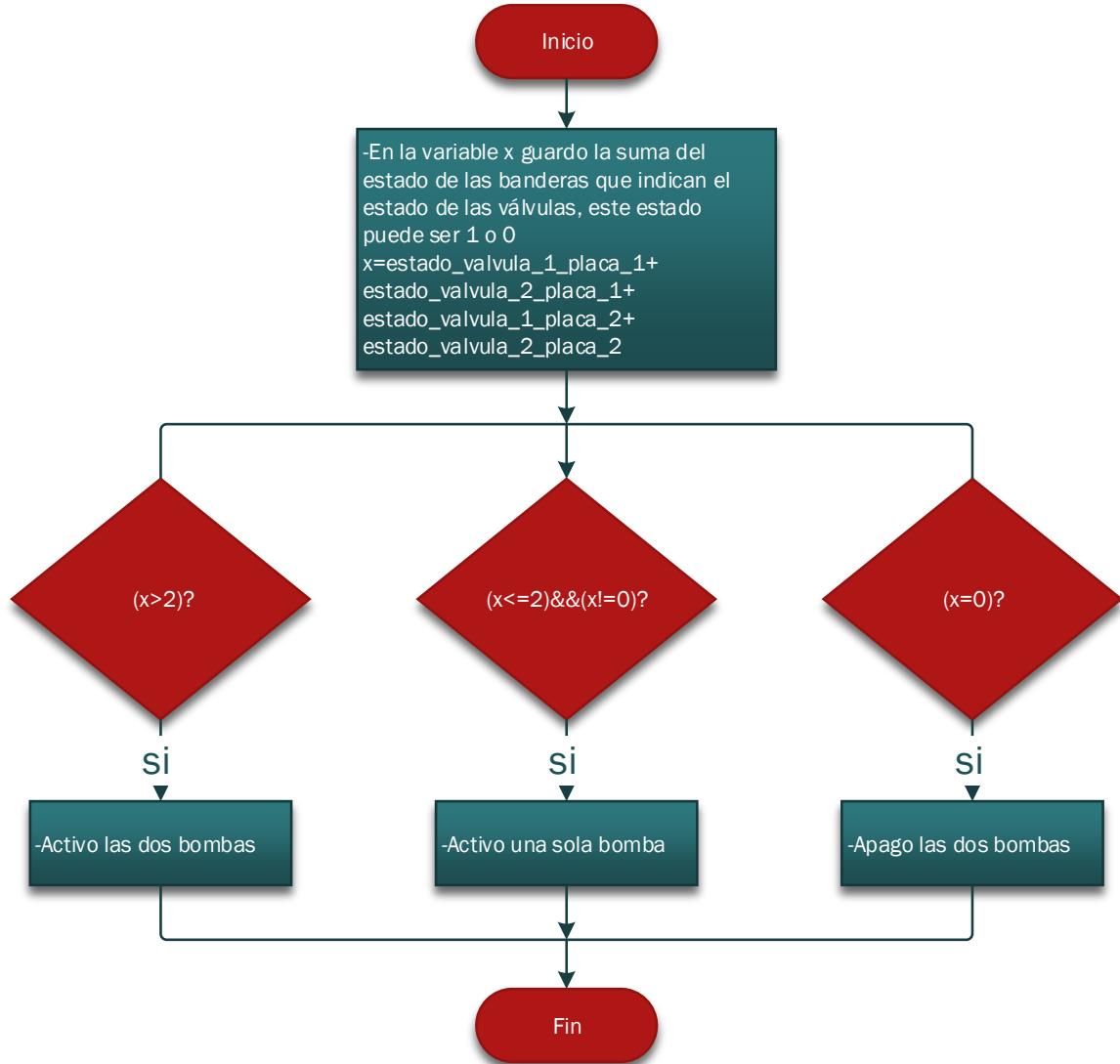
Máquina de estado 6: Pantallas de información de salida del sistema

La implementación de la máquina de estado 6 por su extensión se puede consultar en el anexo V del código fuente.

3.6.10-Funciones en lenguaje C empleadas en el sistema

Se implementaron las siguientes funciones en lenguaje C:

3.6.11-Función utilizada para controlar las bombas de agua



Función 1: Función empleada para el control de las bombas de agua

El código siguiente es la implementación en lenguaje C de la función 1 diseñada:

```

void manejo_bombas()
{
    x=estado_valvula_1_placa_1+estado_valvula_2_placa_1+estado_valvula_1_placa_2+estado_valvula_2_placa_2;
    if ((x<=2)&&(x!=0))
    {
        activar_rele(PIN_A2);
        desactivar_rele(PIN_A5);
    }
    if (x>2)
    {
        activar_rele(PIN_A2);
        activar_rele(PIN_A5);
    }
    if (x==0)
    {

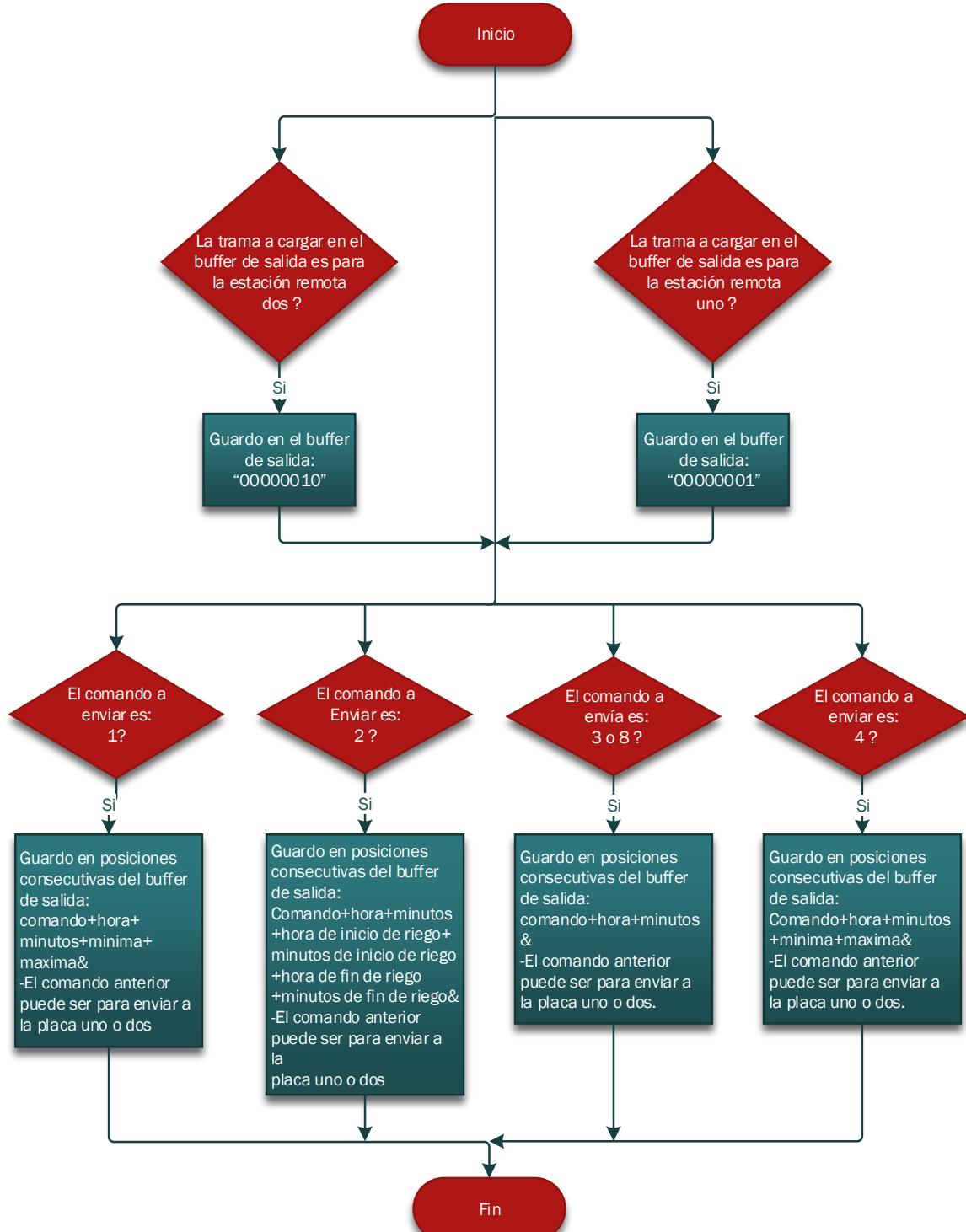
```

```

    desactivar_rele(PIN_A5);
    desactivar_rele(PIN_A2);
}

```

3.6.12-Función utilizada para armar la trama a transmitir



Función 2: Arma la trama a transmitir

El código siguiente es la implementación en lenguaje C de la función 2 diseñada:

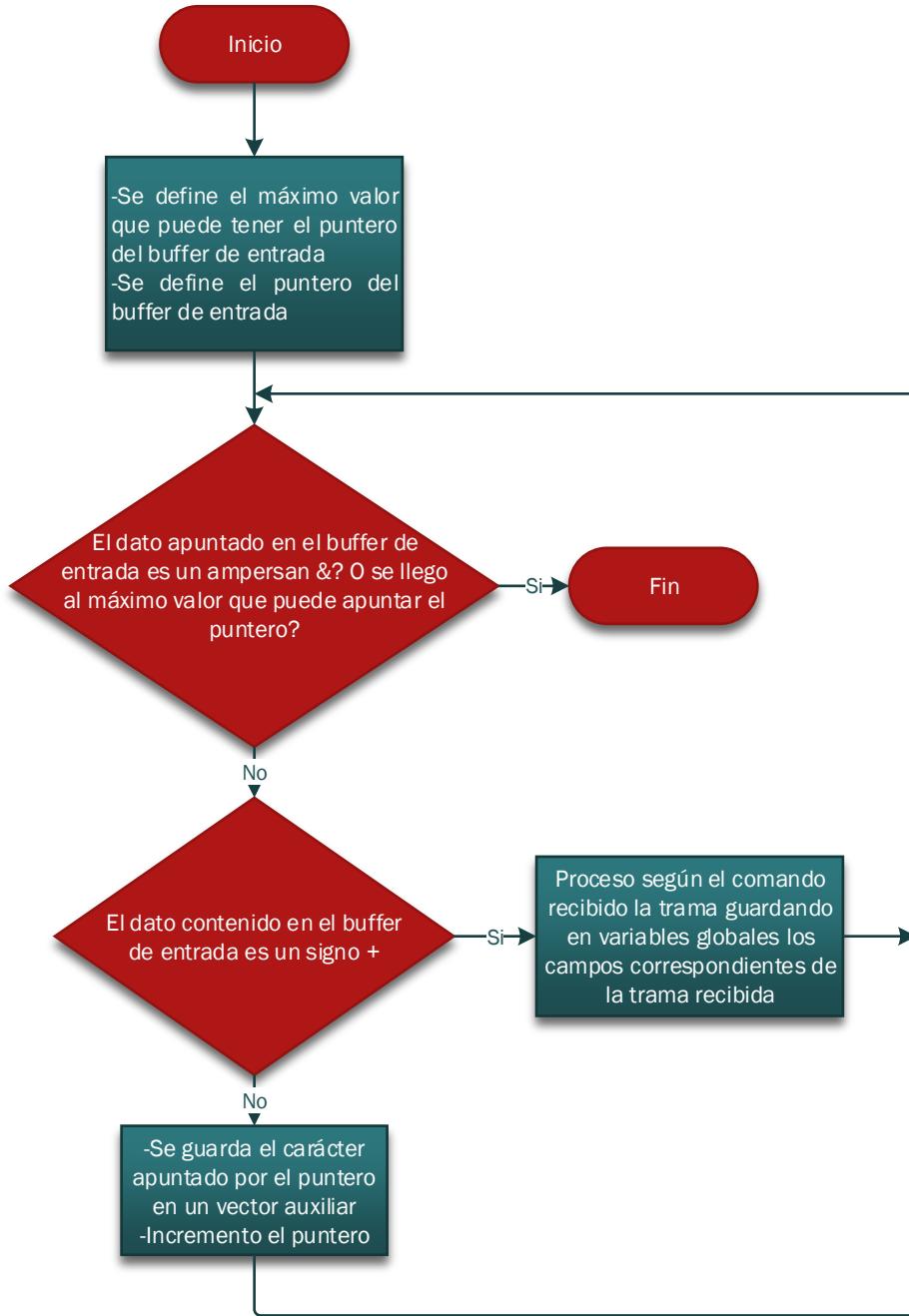
```
void armar_trama()
{
for(i=0;i<=59;i++)
dato_tx[0]='0';
if (tx_placa==1) strcpy(dato_tx,"00000001");
if (tx_placa==2) strcpy(dato_tx,"00000010");
if ((comando_tx==3)||(comando_tx==8))
{
sprintf(cadena,"%lu",comando_tx);
strcat(dato_tx,cadena);
sprintf(cadena,"+");
strcat(dato_tx,cadena);
sprintf(cadena,"%u",hora);
strcat(dato_tx,cadena);
sprintf(cadena,"+");
strcat(dato_tx,cadena);
sprintf(cadena,"%u",minutos);
strcat(dato_tx,cadena);
sprintf(cadena,"& ");
strcat(dato_tx,cadena);
}
if (comando_tx==2)
{
sprintf(cadena,"%lu",comando_tx);
strcat(dato_tx,cadena);
sprintf(cadena,"+");
strcat(dato_tx,cadena);
sprintf(cadena,"%u",hora);
strcat(dato_tx,cadena);
sprintf(cadena,"+");
strcat(dato_tx,cadena);
sprintf(cadena,"%u",minutos);
strcat(dato_tx,cadena);
sprintf(cadena,"+");
strcat(dato_tx,cadena);
sprintf(cadena,"%lu",hora_inicio[tx_placa]);
strcat(dato_tx,cadena);
sprintf(cadena,"+");
strcat(dato_tx,cadena);
sprintf(cadena,"%lu",minutos_inicio[tx_placa]);
strcat(dato_tx,cadena);
sprintf(cadena,"+");
strcat(dato_tx,cadena);
sprintf(cadena,"%lu",hora_fin[tx_placa]);
strcat(dato_tx,cadena);
sprintf(cadena,"+");
strcat(dato_tx,cadena);
sprintf(cadena,"%lu",minutos_fin[tx_placa]);
strcat(dato_tx,cadena);
sprintf(cadena,"& ");
strcat(dato_tx,cadena);
}
if (comando_tx==4)
{
sprintf(cadena,"%lu",comando_tx);
strcat(dato_tx,cadena);
sprintf(cadena,"+");
strcat(dato_tx,cadena);
sprintf(cadena,"%u",hora);
strcat(dato_tx,cadena);
sprintf(cadena,"+");
strcat(dato_tx,cadena);
sprintf(cadena,"%u",minutos);
```

```

strcat(dato_tx,cadena);
sprintf(cadena,"+");
strcat(dato_tx,cadena);
sprintf(cadena,"%lu",manual[tx_placa]);
strcat(dato_tx,cadena);
sprintf(cadena,"& ");
strcat(dato_tx,cadena);
}
if (comando_tx==1)
{
sprintf(cadena,"%lu",comando_tx);
strcat(dato_tx,cadena);
sprintf(cadena,"+");
strcat(dato_tx,cadena);
sprintf(cadena,"%u",hora);
strcat(dato_tx,cadena);
sprintf(cadena,"+");
strcat(dato_tx,cadena);
sprintf(cadena,"%u",minutos);
strcat(dato_tx,cadena);
sprintf(cadena,"+");
strcat(dato_tx,cadena);
sprintf(cadena,"%lu",minima[tx_placa]);
strcat(dato_tx,cadena);
sprintf(cadena,"+");
strcat(dato_tx,cadena);
sprintf(cadena,"%lu",maxima[tx_placa]);
strcat(dato_tx,cadena);
sprintf(cadena,"& ");
strcat(dato_tx,cadena);
}
}

```

3.6.13-Función empleada para decodificar la trama recibida



Función 3: Decodifica la trama recibida

El código siguiente es la implementación en lenguaje C de la función 3 diseñada:

```

void deco_tramaRx()
{
    numero=0;
    fin_pal=50;dat_comp=0;
    for(i=8;i<=fin_pal;i++)
    {
        if((dato_rx[i]=='&')&&(dat_comp==1))
        {
            dat_comp=1;
        }
    }
}
  
```

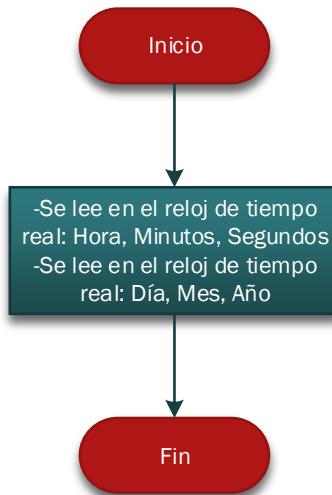
```

fin_pal=i;
}
if ((dato_rx[i]!='+')&& (dato_rx[i]!='&'))
{
dato_aux[j]=dato_rx[i];
j=j+1;dat_comp=0;
}else {
    for(s=0;s<=3;s++) dato_aux2[s]='0';
    for(s=0;s<j;s++) dato_aux2[(4-j)+s]=dato_aux[s];
    j=0;
    if ((numero==0)&&(dat_comp!=1))
    {
        comando_rx=(atoi32(dato_aux2));
        numero=1; dat_comp=1;
    }
    if ((numero==1)&&(dat_comp!=1))
    {
        if (comando_rx==1) valor_humedad_uno[tx_placa]=(atoi32(dato_aux2));
        if ((comando_rx==2)&&(tx_placa==1))
        {
            estado_valvula_1_placa_1_anterior=estado_valvula_1_placa_1;
            estado_valvula_1_placa_1=(atoi32(dato_aux2));
        }
        if ((comando_rx==2)&&(tx_placa==2))
        {
            estado_valvula_1_placa_2_anterior=estado_valvula_1_placa_2;
            estado_valvula_1_placa_2=(atoi32(dato_aux2));
        }
        numero=2; dat_comp=1;
    }
    if ((numero==2)&&(dat_comp!=1))
    {
        if (comando_rx==1) valor_humedad_dos[tx_placa]=(atoi32(dato_aux2));
        if ((comando_rx==2)&&(tx_placa==1))
        {
            estado_valvula_2_placa_1_anterior=estado_valvula_2_placa_1;
            estado_valvula_2_placa_1=(atoi32(dato_aux2));
        }
        if ((comando_rx==2)&&(tx_placa==2))
        {
            estado_valvula_2_placa_2_anterior=estado_valvula_2_placa_2;
            estado_valvula_2_placa_2=(atoi32(dato_aux2));
        }
        numero=3; dat_comp=1;
    }
    if ((numero==3)&&(dat_comp!=1))
    {
        if (comando_rx==1) valor_humedad_tres[tx_placa]=(atoi32(dato_aux2));
        if ((comando_rx==2)&&(tx_placa==1))
        {
            tipo_riego_anterior_placa_1=tipo_riego_placa_1;
            tipo_riego_placa_1=(atoi32(dato_aux2));
        }
        if ((comando_rx==2)&&(tx_placa==2))
        {
            tipo_riego_anterior_placa_2=tipo_riego_placa_2;
            tipo_riego_placa_2=(atoi32(dato_aux2));
        }
        numero=4; dat_comp=1;
    }
    if ((numero==4)&&(dat_comp!=1))
    {
        if (comando_rx==1) valor_humedad_cuatro[tx_placa]=(atoi32(dato_aux2));
        if (comando_rx==2);
        numero=0;
        dat_comp=1;
        fin_pal=i;
    }
}

```

```
    }  
}
```

3.6.14-Función utilizada para la lectura del reloj de tiempo real

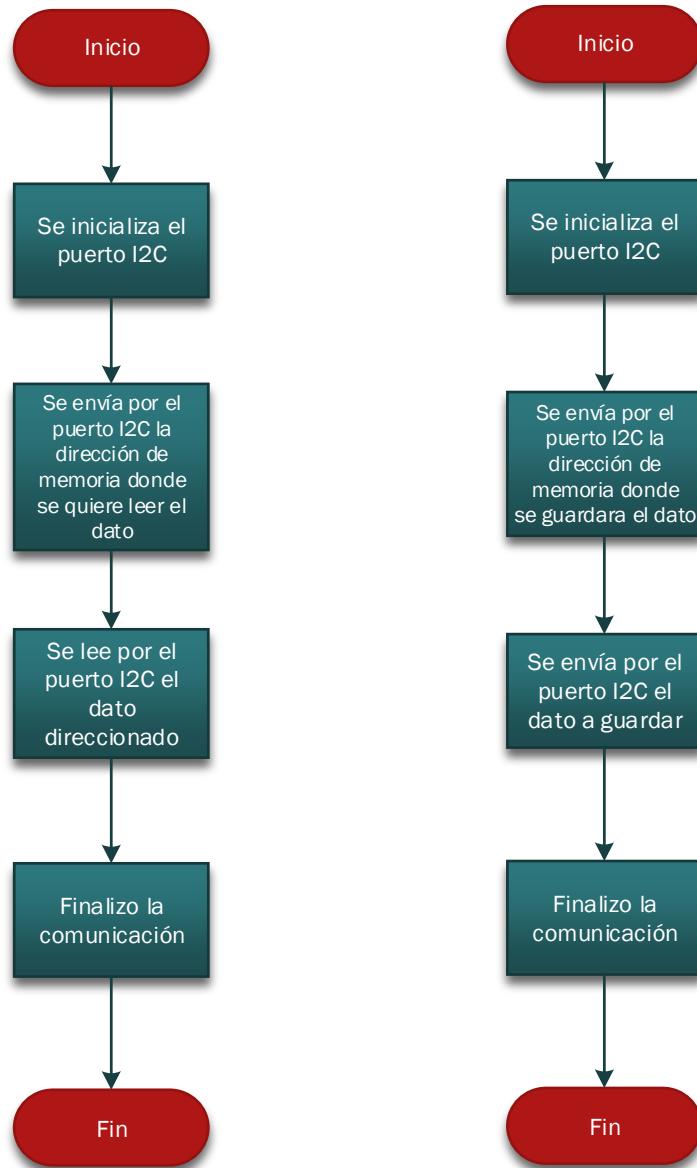


Función 4: Utilizada para leer el reloj de tiempo real

El código siguiente es la implementación en lenguaje C de la función 4 diseñada:

```
void reloj_tiempo_real()  
{  
    delay_ms(15);  
    ds1307_get_time(hora,minutos,segundos);  
    delay_ms(15);  
    ds1307_get_date(dia,mes,anio,dow1);  
    delay_ms(15);  
}
```

3.6.15-Funciones para leer y guardar datos en la memoria EEPROM



Función 5: Leer EEPROM

Función 6: Guardar EEPROM

El código siguiente es la implementación en lenguaje C de la función 5 diseñada para leer en memoria EEPROM, de tipo I2C:

```

BYTE leer_MEM(long int direccion)
{
    byte dato;
    i2c_start();           //inicializa la transmisión
    i2c_write(0xA0);       //escribe la palabra de control (dirección 0h
                          //+ 0 para escritura
    i2c_write(direccion>>8); //parte alta de la dirección a escribir en la
                           //EEPROM
    i2c_write(direccion);   //parte baja de la dirección a escribir en la
                           //EEPROM
    i2c_start();           //reinicio
    i2c_write(0xa1);       //escribe la palabra de control (dirección 0h
                          //+1 para la lectura)
    dato=i2c_read(0);      //lectura del dato
}

```

```

i2c_stop();           //finalización de la transmisión
return(dato);
}

```

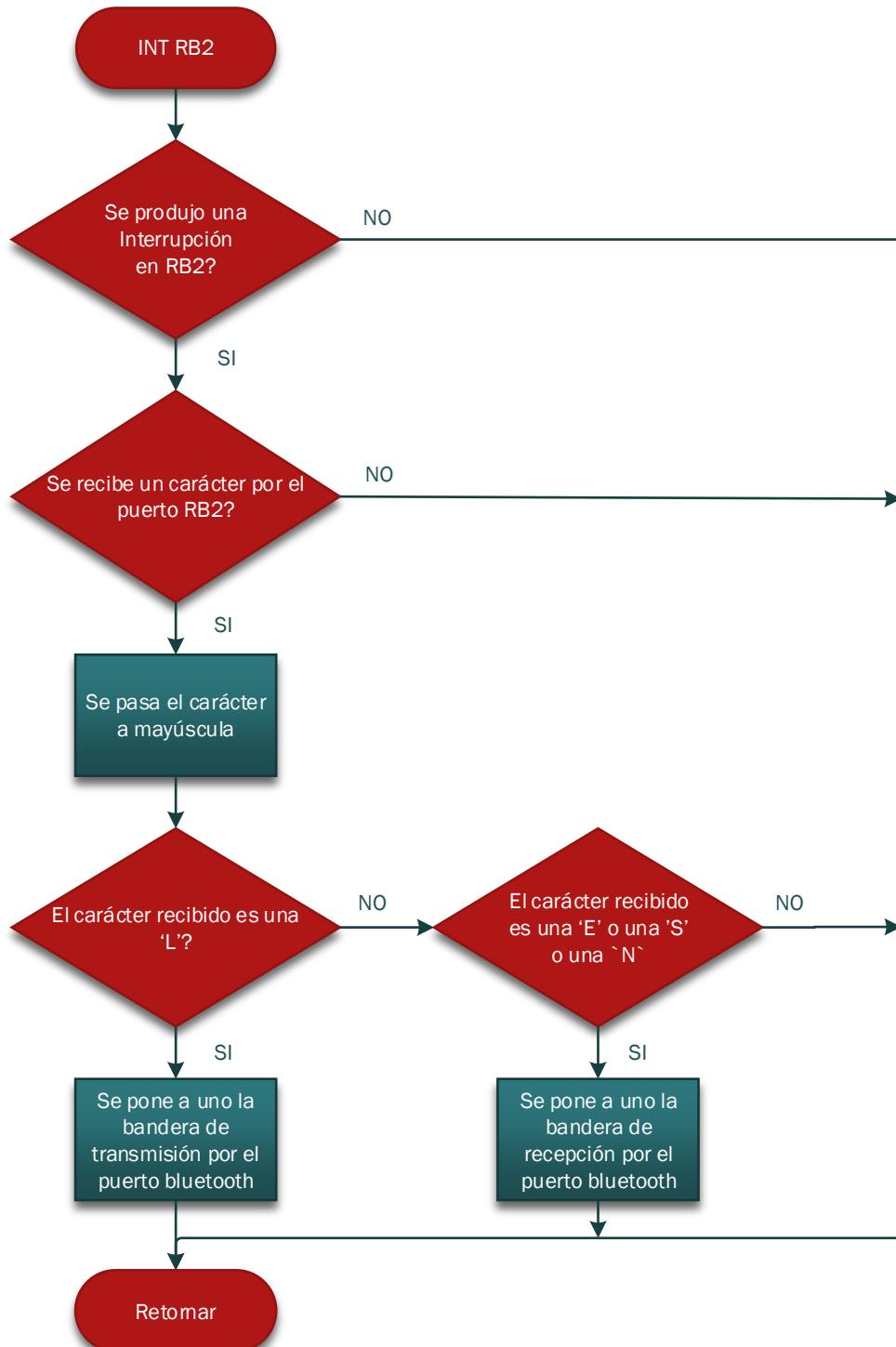
El código siguiente es la implementación en lenguaje C de la función 6 diseñada para guardar en memoria EEPROM, de tipo I2C:

```

void guardar_MEM( byte dato, long int dirección)
{
short int estado;
i2c_start();           //inicializa la transmisión
i2c_write(0xA0);       //escribe la palabra de control(dirección
                      //0h+0 para escritura
i2c_write(dirección>>8); //parte alta de la dirección a escribir en la
                          //EEPROM
i2c_write(dirección);  //parte baja de la dirección a escribir en la
                          //EEPROM
i2c_write(dato);        //dato a escribir
i2c_stop();            //finalización de la transmisión
i2c_start();           //reinicio
estado=i2c_write(0xa0); //lectura del bit ACK, para evitar escrituras
                      //incorrectas
while (estado==1)      //si es 1 espera a que responda el esclavo
{
i2c_start();
estado=i2c_write(0xa0);
}
}

```

3.6.16-Función de interrupción externa y configuraciones necesarias para generar un puerto de comunicaciones serie por software

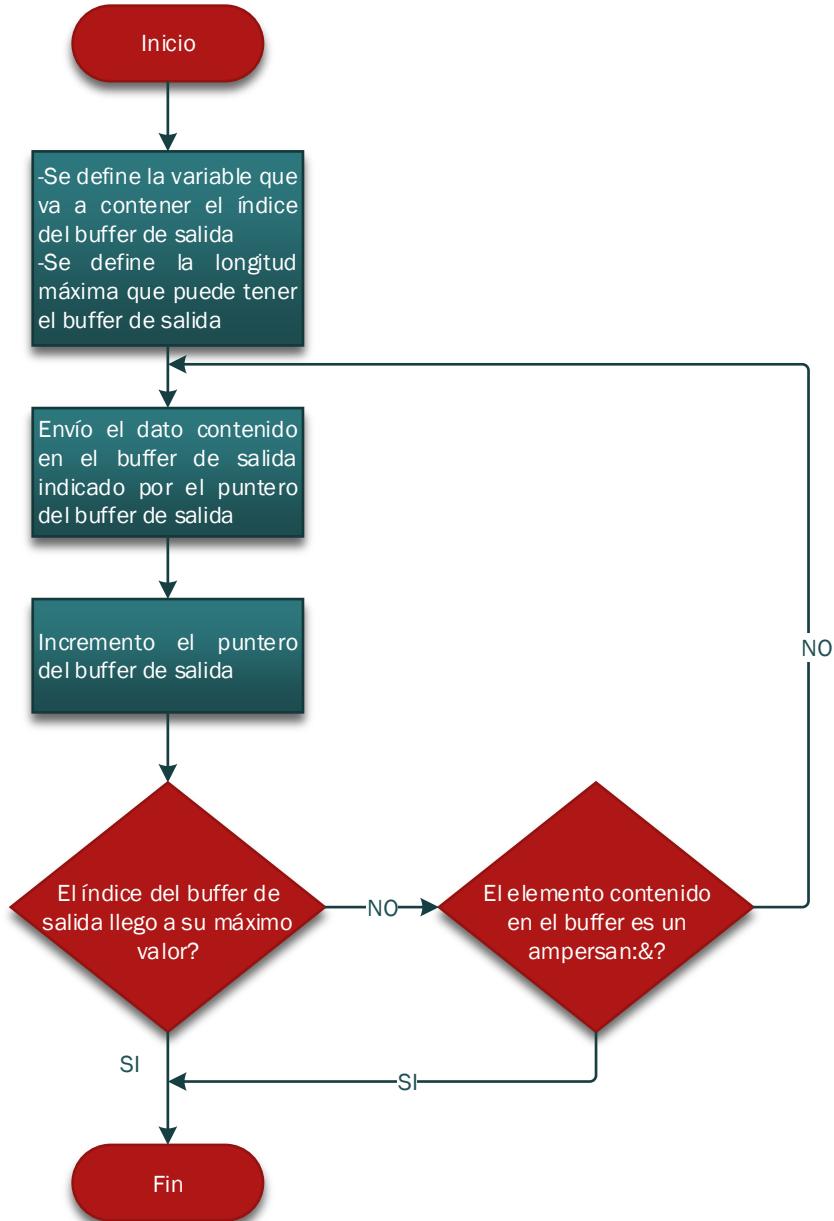


Función 7: Permite comunicaciones serie por software utilizada en el módulo bluetooth

El código siguiente es la implementación en lenguaje C de la función 7 diseñada

```
void interruptb2_init()
{
    enable_interrupts(int_ext2);
    ext_int_edge(2, H_TO_L );
}
#endif
void int_RB2()
{
    if(kbhit(PORT2))
    {
        c=toupper(getc(PORT2));
        if(c=='L')
        {
            tx_blue=1;
            caracter_rx_blue=c;
        }
        if((c=='E')||(c=='S')||(c=='N'))
        {
            rx_blue=1;
            caracter_rx_blue=c;
        }
    }
}
```

3.6.17-Funcion utilizada para transmitir datos a través del puerto serie por hardware del bus RS-485



Función 8: Permite comunicaciones serie por hardware para el bus RS-485

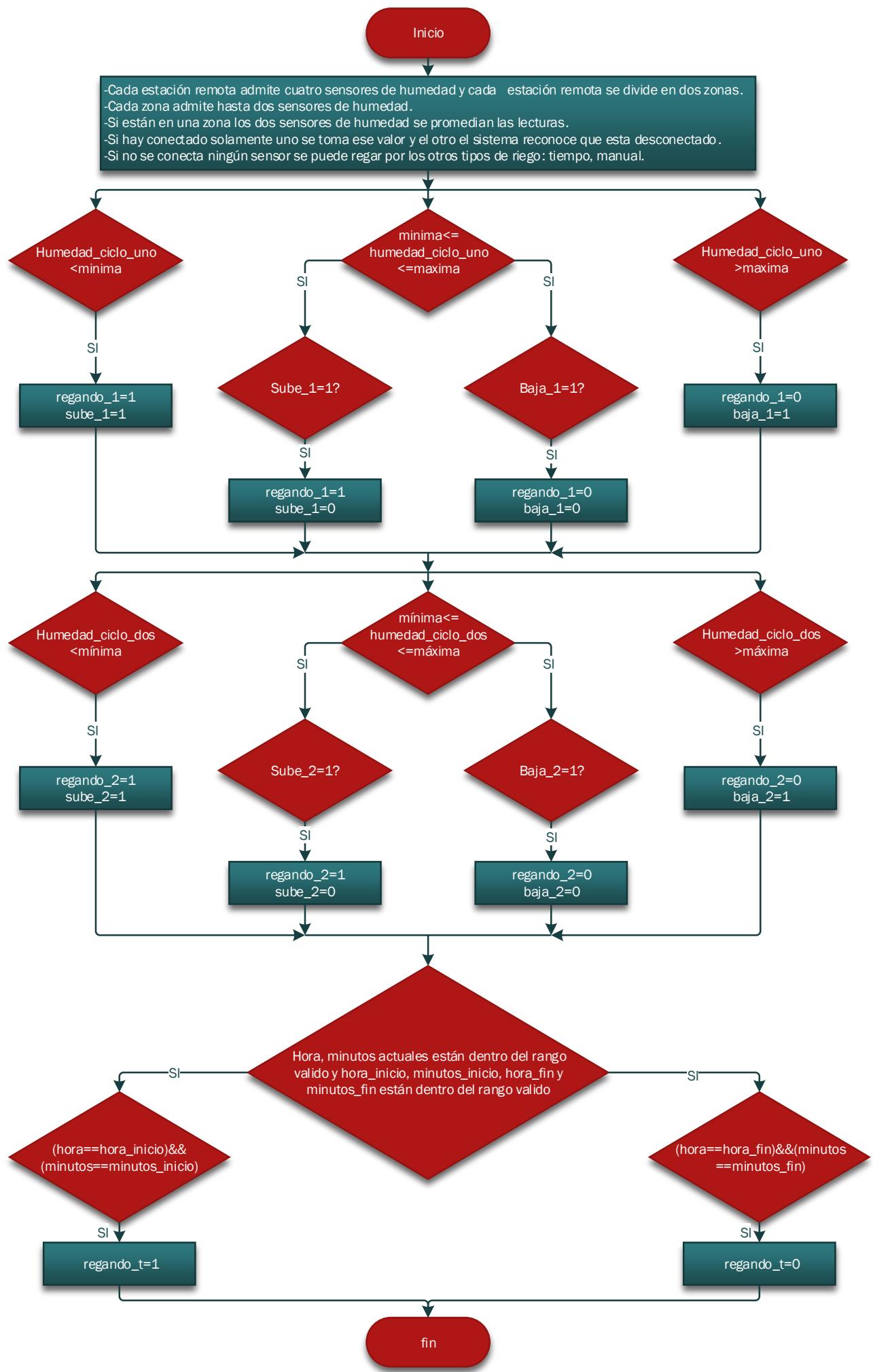
El código siguiente es la implementación en lenguaje C de la función 8 diseñada

```

void envio_dato()
{
int i_tx,fin_dto=60;
for(i_tx=0;i_tx<=fin_dto;i_tx++)
{
putc(data_tx[i_tx]);
if (data_tx[i_tx]=='&') fin_dto=i_tx;
}

```

3.6.8-Función utilizada para generar dos ciclos de histéresis y controlar el riego por tiempo



Función 9: Utilizada para controlar el riego por tiempo y humedad

El código siguiente es la implementación en lenguaje C de la función 9 diseñada:

```
void ciclo_histeresis_control_tiempo()
{
    if ((valor_humedad[0]!=0)&&(valor_humedad[1]!=0))
        humedad_ciclo_uno=((valor_humedad[0]+valor_humedad[1])/2);
    else if(valor_humedad[0]==0) humedad_ciclo_uno=valor_humedad[1];
    else if(valor_humedad[1]==0) humedad_ciclo_uno=valor_humedad[0];

    if ((valor_humedad[2]!=0)&&(valor_humedad[3]!=0))
        humedad_ciclo_dos=((valor_humedad[2]+valor_humedad[3])/2);
    else if(valor_humedad[2]==0) humedad_ciclo_dos=valor_humedad[3];
    else if(valor_humedad[3]==0) humedad_ciclo_dos=valor_humedad[2];
    //////////////CICLO DE HISTERESIS SENSOR UNO///////////////////////////
    if ( humedad_ciclo_uno<minima)
    {
        regando_1=1;
        sube_1=1;
    }
    if ( humedad_ciclo_uno>maxima)
    {
        regando_1=0;
        baja_1=1;
    }
    if (minima<= humedad_ciclo_uno<=maxima)
    {
        if (sube_1==1){regando_1=1; sube_1=0;}
        if (baja_1==1){regando_1=0; ;baja_1=0; }
    }
    //////////////CICLO DE HISTERESIS SENSOR DOS///////////////////////////
    if (humedad_ciclo_dos<minima)
    {
        regando_2=1;
        sube_2=1;
    }
    if ( humedad_ciclo_dos>maxima)
    {hor
        regando_2=0;
        baja_2=1;
    }
    if (minima<= humedad_ciclo_dos<=maxima)
    {
        if (sube_2==1){regando_2=1; sube_2=0;}
        if (baja_2==1){regando_2=0; ;baja_2=0; }
    }
    if (minima==maxima)
    {
        regando_1=0;
        regando_2=0;
    }
    if ((valor_humedad[0]==0)&&(valor_humedad[1]==0))regando_1=0;
    if ((valor_humedad[2]==0)&&(valor_humedad[3]==0))regando_2=0;
    ////////////////////CONTRO POR TIEMPO/////////////////////////////
    if
        (((hora>=0)&&(hora<=23)&&(minutos>=0)&&(minutos<=60)&&(hora_inicio>=0)&&(hora_inicio<=23)&&(minutos_inicio>=0)&&(minutos_inicio<=60)&&
            (hora_fin>=0)&&(hora_fin<=23)&&(minutos_fin>=0)&&(minutos_fin<=60)))
    {
        if ((hora==hora_inicio)&&(minutos==minutos_inicio)) regando_t=1;
        if ((hora==hora_fin)&&(minutos==minutos_fin)) regando_t=0;
    }
    ////////////////////FIN CONTROL TIEMPO/////////////////////////////
}
```

4-SEÑALES DE VISUALIZACION Y MEDIDAS DE PROTECCION DEL SISTEMA ANTE FALLOS

Todo sistema de control debe detectar una anomalía de funcionamiento o fallo del sistema, el implementado posee varios indicadores para detectar un fallo y medidas para proteger la integridad del sistema del mal funcionamiento, los fallos de sistema se pueden deber a las siguientes causas:

1-Por mal funcionamiento de la estación remota o rotura del cable de red:

La estación concentradora permanentemente está comunicándose con las remotas y si ocurre esto titila un led en la estación concentradora, cuando envía una trama apaga el led y cuando recibe respuesta de la estación remota enciende el led, por lo tanto si deja de funcionar una estación remota o se corta el cable de red el led de esa estación remota deja de titilar y la estación concentradora elimina del sistema a la remota defectuosa, deja de comunicarse con ella y la elimina de los menús de configuración, ya no se la puede programar desde la estación concentradora ya esa estación remota no existe más para el sistema, con lo cual si se corta el cable de red o deja de funcionar una estación remota el sistema sigue funcionando sin la estación afectada e indica esta falla dejando de titilar el led correspondiente a la estación remota defectuosa.

2-Por mal funcionamiento de la estación concentradora:

Si todos los leds pertenecientes a las estaciones remotas dejan de titilar es un indicador de mal funcionamiento de la estación concentradora.

3-Por mal funcionamiento de un sensor de humedad:

Los sensores de humedad cuentan en la bornera de conexión con una resistencia de pull-down, esta resistencia lleva a cero voltios la entrada analógica en caso de que se desconecte el cable del sensor o se rompa, el sistema al leer un sensor y detectar cero por ciento de humedad interpreta que el sensor de humedad está desconectado o defectuoso y lo elimina del sistema indicando esto en el display de cristal líquido de la estación concentradora en el menú de vista de los valores de humedad, si se desconectan o se rompen todos los sensores de una zona el sistema permite solamente la programación por franja horaria y por riego manual en la zona afectada, además si el valor de humedad leído se encuentra fuera del rango válido el sistema elimina el sensor por mal funcionamiento y esa zona también admite únicamente riego manual y por franja horaria.

4-Por mal funcionamiento del circuito hidráulico o desperfectos de la bomba o las válvulas:

El sistema hidráulico de la explotación agrícola está dividido en dos, y cada subsistema posee una bomba de agua, por esto la estación concentradora posee dos entradas analógicas acondicionadas con amplificadores operacionales destinadas a la instalación de dos presostatos, estos sensan la presión de la cañería, si aumenta la presión en la cañería es un indicador de que una de las válvulas de agua esta defectuosa y ante una disminución de presión es un síntoma de rotura de la cañería o que una bomba no se puso en servicio, el hardware permite la instalación de dos presostatos pero es una mejora a futuro no se instalaron en el sistema actual por falta de presupuesto propio y falta de interés y presupuesto del dueño de la explotación agrícola.

5-Por datos ingresados, generados, o transmitidos por el sistema fuera de rango:

Todos los datos ingresados, generados o transmitidos tienen control de rango, se corrobora que los datos sean válidos según el rango establecido para el tipo de dato, si el dato es ingresado por teclado y está fuera del rango establecido para ese tipo de dato el sistema muestra un mensaje de error y permite reingresar el dato, si es generado por el propio sistema de riego como por ejemplo la hora y fecha entregada por el reloj de tiempo real, estos datos también se validan y si es erróneo se vuelve a solicitar la hora y la fecha, si el dato es transmitido por el bus de comunicaciones se implementó un checksum que se envía con la trama transmitida y si la estación que recibe el dato mediante el checksum detecta que este dato es erróneo solicita que se reenvíe la trama.

5-ASPECTOS ECONOMICOS

Desde el punto de vista económico los costos de los materiales electrónicos para la construcción del sistema de riego son despreciables frente a los costos de los elementos hidráulicos y eléctricos necesarios para el sistema, dotar de un sistema de control electrónico de riego a una explotación agrícola le confiere ahorros de operatividad importantes, en un sistema de riego tradicional sin control digital las bombas durante el día permanentemente en funcionamiento de forma continua se apagan a la noche cuando hay poca evaporación mientras que con un control digital las bombas se activan en los momentos necesarios con el consiguiente ahorro de energía, además se puede regular el aporte de nutrientes óptimo con el consiguiente aumento de la producción, si el agua es facturada según el volumen utilizado el ahorro es notable.

Pero fundamentalmente es un desarrollo nacional diseñado y construido en el país con lo cual la instalación, el mantenimiento y las posibles ampliaciones pueden ser llevadas a cabo de una forma óptima con personal y recursos propios de la Argentina.

6-ENSAYOS

El sistema está instalado prestando servicios de forma continua.

En las figuras: 30, 31, 32 se observa la humedad registrada por el sistema en función del tiempo entre los días 19/05/15 al 26/05/15, para los dos sensores de humedad de suelo de la estación remota uno y el único sensor instalado en la estación remota dos.

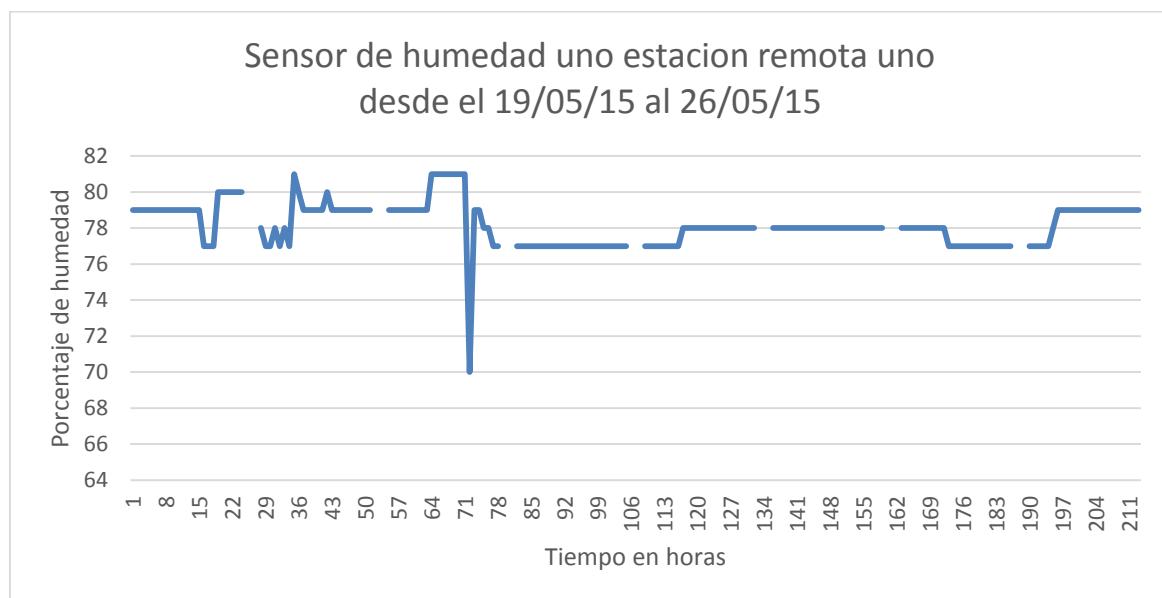


Figura 30: Curva registrada por el sensor uno de la estación remota uno

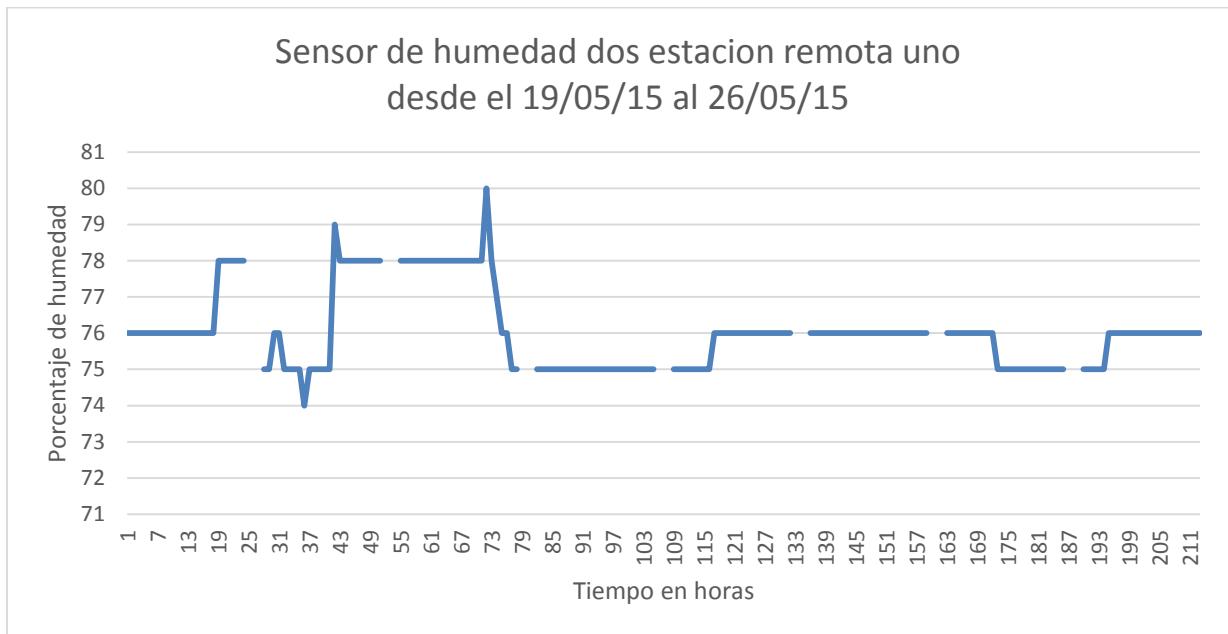


Figura 31: Curva registrada por el sensor de humedad dos de la estación remota uno

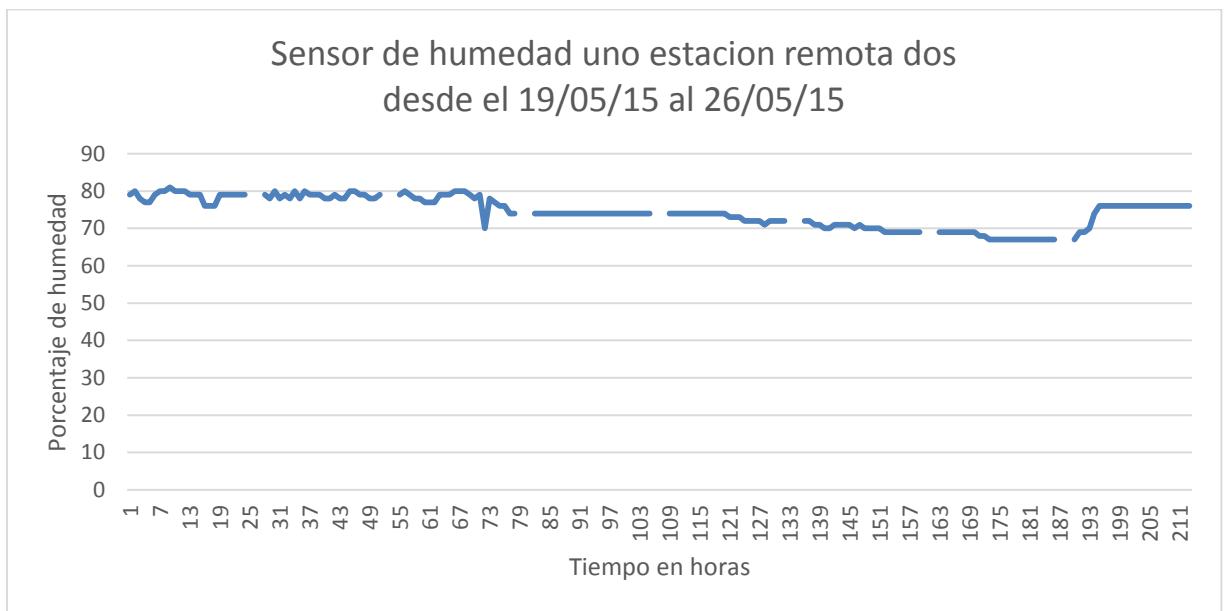


Figura 32: Curva registrada por el sensor de humedad uno de la estación remota dos

En la figura 33 se puede observar la actividad de la válvula uno de la estación remota uno para el día 23/05/15:

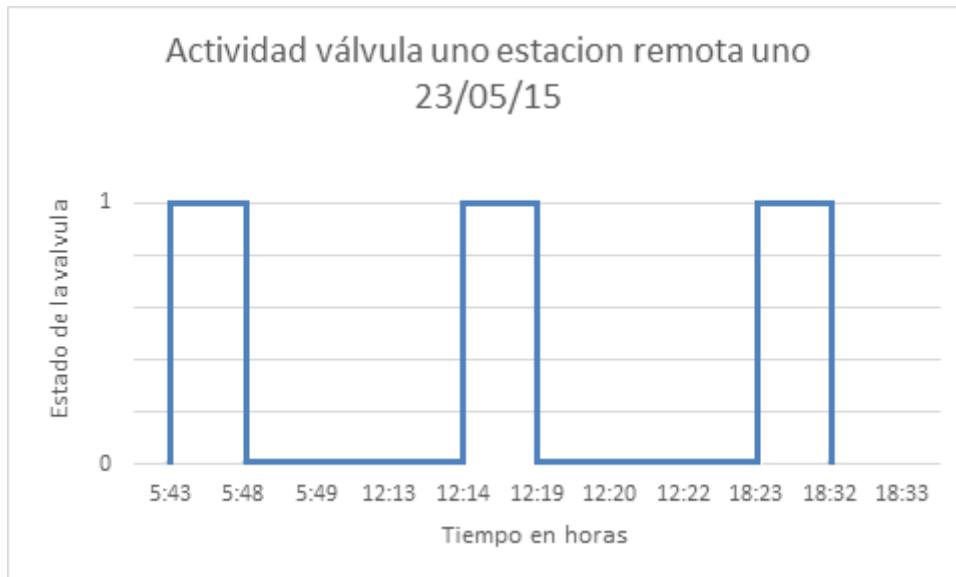


Figura 33: Registro de la actividad de la válvula uno de la estación remota uno

7-CONCLUSIONES Y MEJORAS A FUTURO

En términos generales se ha diseñado e implementado un sistema de control de riego que permite la gestión del recurso hídrico en explotaciones agrícolas tal como fue planteado en los objetivos y alcances, el sistema se desarrolló en base a un concepto de control distribuido consistente en una central de comandos y nodos autónomos distribuidos en la explotación agrícola, lo que permite fácil y rápido crecimiento de la capacidad de riego (agregar mayor superficie regada).

Para lograr el objetivo planteado se han desarrollado tres prototipos de la unidad de control y nodos, el primer y segundo prototipo permitió ensayos en laboratorio detectando y corrigiendo problemas de diseño que fueron subsanados en el tercer prototipo.

El tercer prototipo fue ensayado en campo obteniéndose resultados alentadores y detectando la necesidad de implementar algunas nuevas funciones para dar robustez al sistema de riego (ver mejoras a futuro).

Se han alcanzado los objetivos particulares de control de las bombas de agua, sensores de riego y electroválvulas, para ello fue necesario alcanzar el dominio de diferentes tecnologías como ser: protocolo RS-485, I2C, SPI, bluetooth, manejo de memorias externas en un microcontrolador, reloj de tiempo real, conversor analógico digital y programación avanzada en lenguaje "C".

Mejoras a futuro:

Producto de la experiencia en campo se pueden implementar las siguientes mejoras:

- Construcción de un módulo inalámbrico de bajo consumo alimentado con paneles fotovoltaicos destinado a censar la humedad de forma remota, censando la humedad en el lugar óptimo donde los obstáculos del terreno permitan la comunicación inalámbrica y usando el método desarrollado actualmente cuando los obstáculos del terreno no lo permitan.
- Construcción de un sensor de humedad de la misma calidad de los sensores utilizados, logrando similares o mejores características de resolución, precisión repetitividad de la medida y robustez.
- Implementar una estación meteorológica que permita contar con datos del clima e implementar un modelo matemático de riego.
- Conectar el sistema de riego a internet mediante un servidor web embebido en un microcontrolador que le permita mostrar parámetros y telecontrol del sistema de riego cuando la explotación agrícola posea conexión a internet.
- Las explotaciones agrícolas necesita automatizar otros procesos relacionados con el riego como ser la limpieza de los filtros de agua, este proceso puede ser telecontrolado y gestionado por el sistema de riego, diseñando un software para la central de control que aporte mayores prestaciones.

8- REFERENCIAS

[1]Informe sobre Desarrollo Humano 2006: Más allá de la escasez: Poder, pobreza y crisis mundial del agua. PNUD, 2006 Combatir la escasez de agua. El desafío del Siglo XXI. ONU-Agua, FAO. 2000

[2]AUTOMATIZACION Y TELECONTROL DE SISTEMAS DE RIEGO, AUTOR: Antonio Ruiz Canales, Jose Miguel Molina Martinez, Ed Marcombo 409p.

[3]INSTRUMENTACION ELECTRONICA, AUTOR: Perea M., Alvarez J., Campo J., Ferrero F., Grillo G. 2004., Ed Thompson. 662p

[4]TECNICAS PARA LA AGRICULTURA SOSTENIBLE, EL RIEGO POR GOTEO UNA INTRODUCCION, AUTOR: C. C. Shock y T. Welch., Ed Oregon State University.

[5]EC-20, EC-10 AND EC-5 USER'S MANUAL, Decagon Devices-Hoja de datos

[6]DS1307, I2C REAL TIME CLOCK, MAXIM-Hoja de datos

[7]CATALOGO DEL FABRICANTE DE VALVULAS BERMAD:

http://www.bermad.com/data/uploads/PC2AS11%20IR%202000%20SERIES%20CATALOGUE_SPANISH_FINAL.pdf

[8] NOTA TECNICA AN234, MICROCHIP

[9]SISTEMAS OPERATIVOS DE TIEMPO REAL, Autor: Alejandro Celery:

http://www.sase.com.ar/2011/files/2010/11/SASE2011-Introduccion_RTOS.pdf

UNIVERSIDAD NACIONAL DE CATAMARCA



ANEXO I: CIRCUITOS ESQUEMATICOS

Diseño de un sistema de riego para optimizar el recurso hídrico

Matías Leandro Ferraro

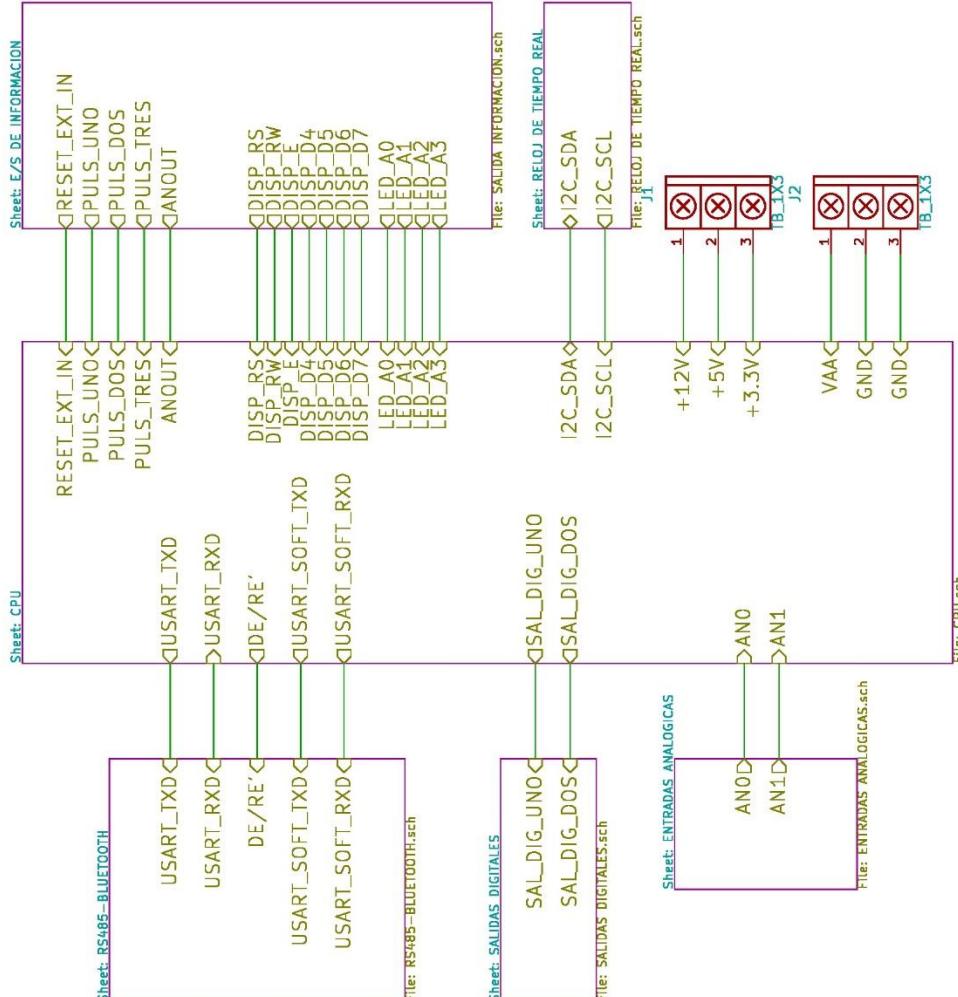
Año: 2015

Título: Ingeniero Electrónico
Director de tesis: Sergio Hilario Gallina
Departamento: Electrónica

Sistema de Riego Estación Concentradora

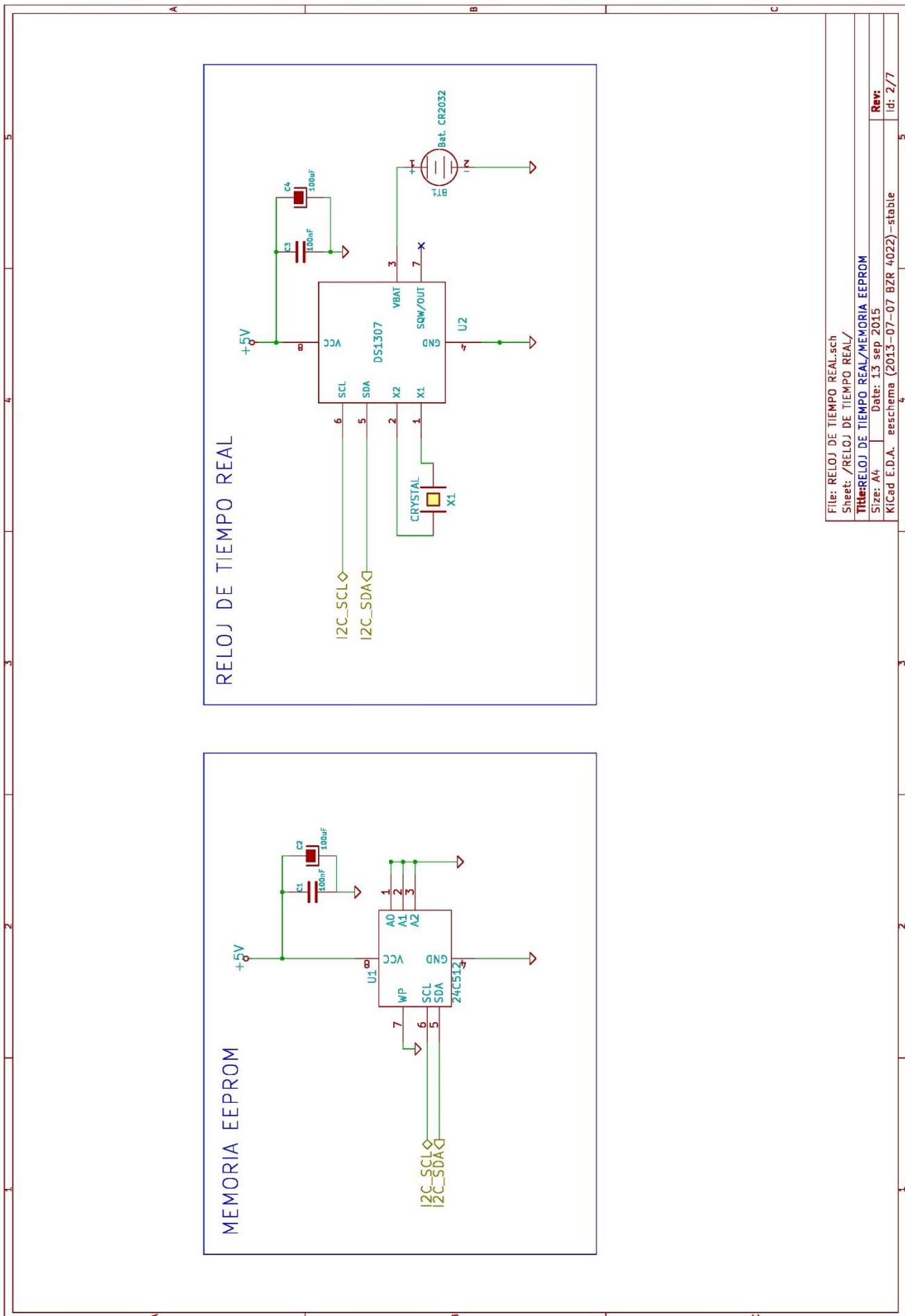
Índice:

1. Esquemático Jerárquico
2. Reloj de Tiempo Real/Memoria Entrada/Salida de Información
3. Entradas Analógicas
4. Entradas Digitales
5. CPU
6. RS485/BLUETOOTH
7. Salidas Digitales

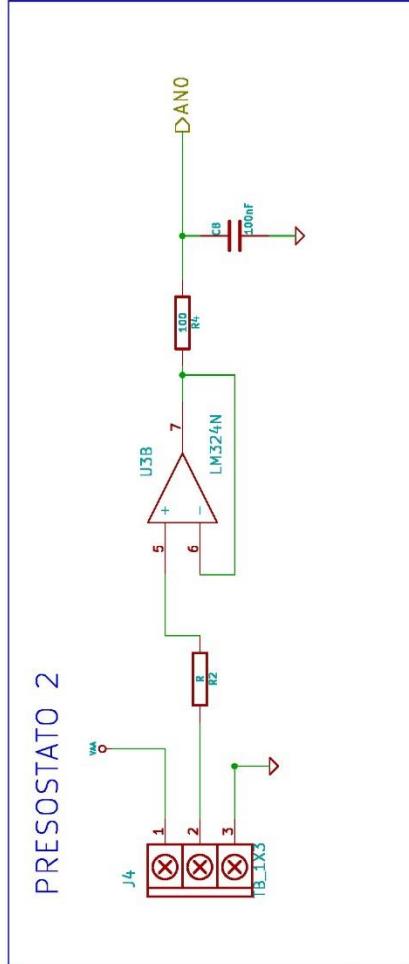
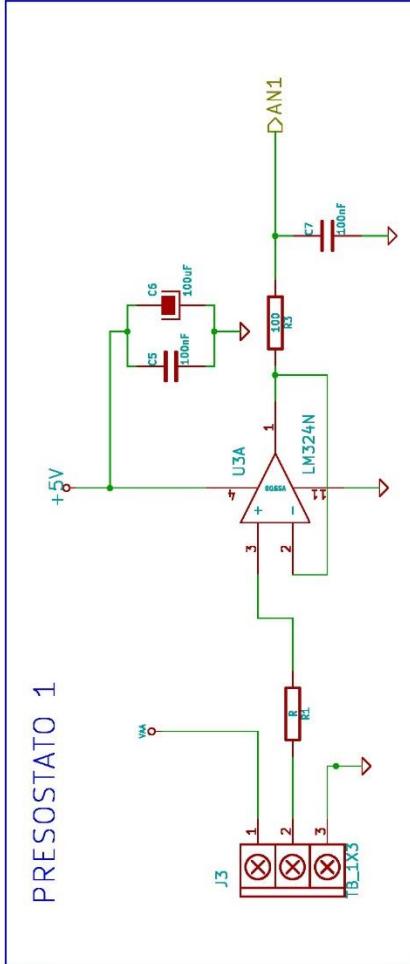


File: sistema_riege.central.sch
Sheet: /
Title: ESQUEMATICO_JERARQUICO
Size: A4
Date: 13 sep 2015
KICad E.D.A. eschema (2013-07-07 BZR 4022)-stable
Rev: 4
Id: 1/1

File: sistema_riege.central.sch	Sheet: /	Title: ESQUEMATICO_JERARQUICO	Rev: 4
		Size: A4	
		Date: 13 sep 2015	
		KICad E.D.A. eschema (2013-07-07 BZR 4022)-stable	
		Rev: 5	
		Id: 1/1	



File: RELOJ DE TIEMPO REAL.sch
 Sheet: /RELOJ DE TIEMPO REAL/
 Title: RELOJ DE TIEMPO REAL/MEMORIA EEPROM
 Size: A4 Date: 13 sep 2015 Rev:
 KiCad E.D.A. eschema (2013-07-07 BZR 4022)-stable Id: 2/7
 5
 4
 3
 2
 1



File: ENTRADAS ANALÓGICAS.sch
 Sheet: /ENTRADAS ANALÓGICAS/
Title: ENTRADAS ANALÓGICAS
 Size: A4 Date: 13 sep
 KiCad E.D.A. eeschema (2013-
 4)

5

4

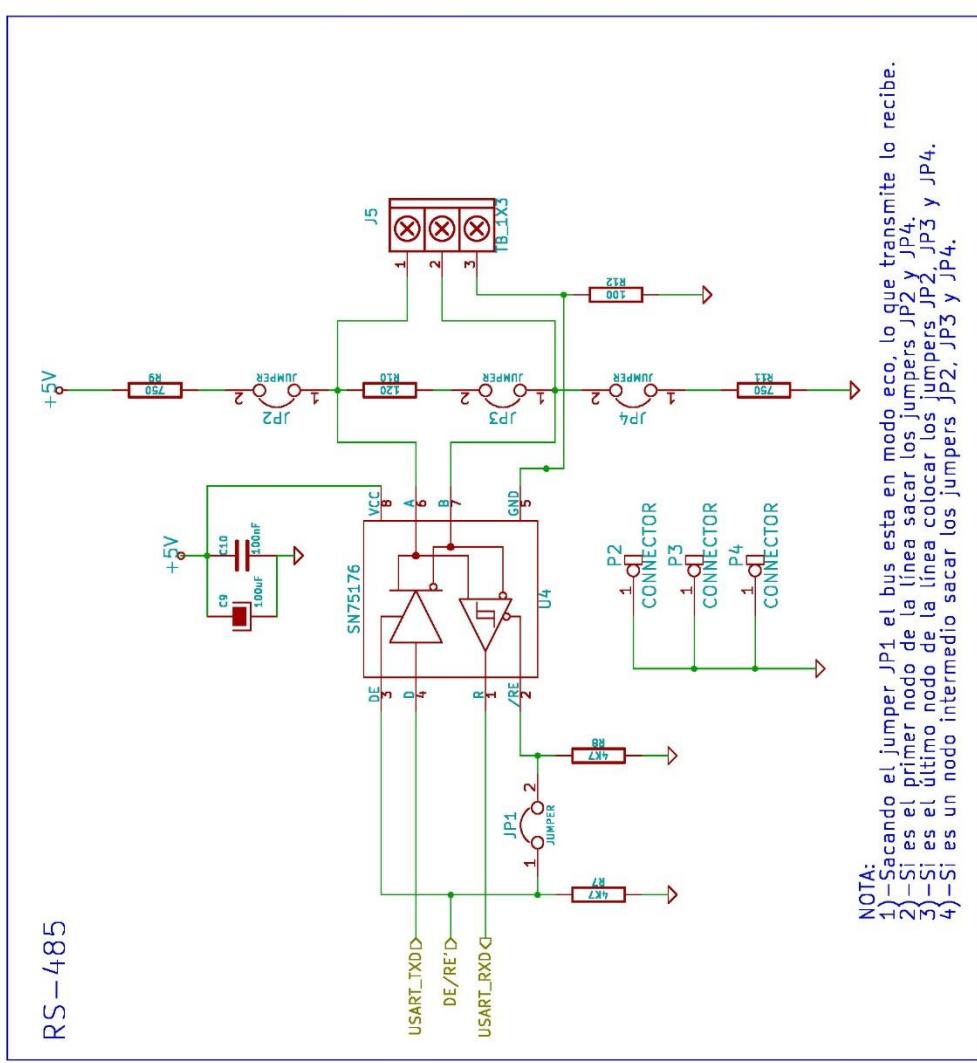
3

2

1

RS-485

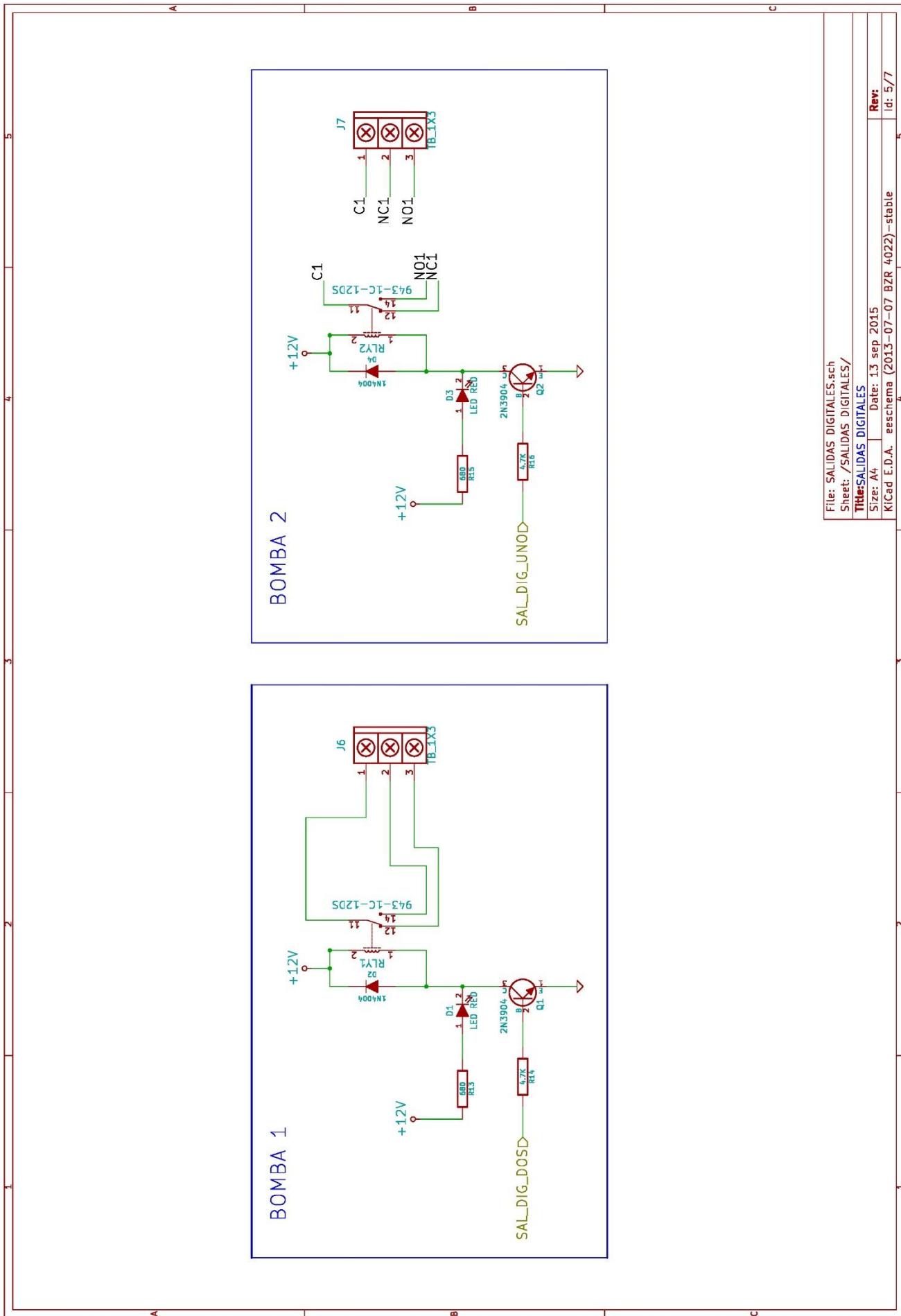
BLUETOOTH

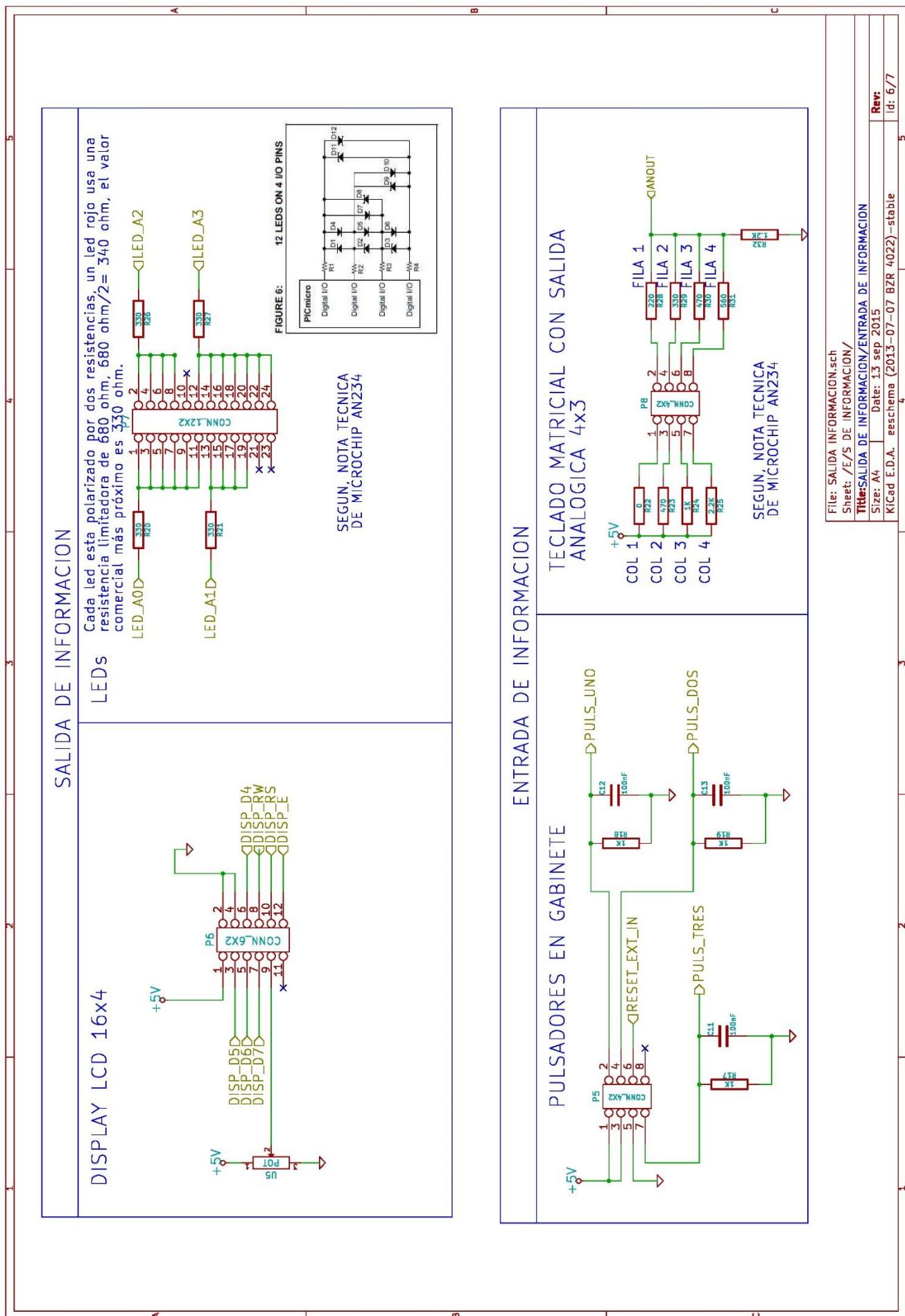


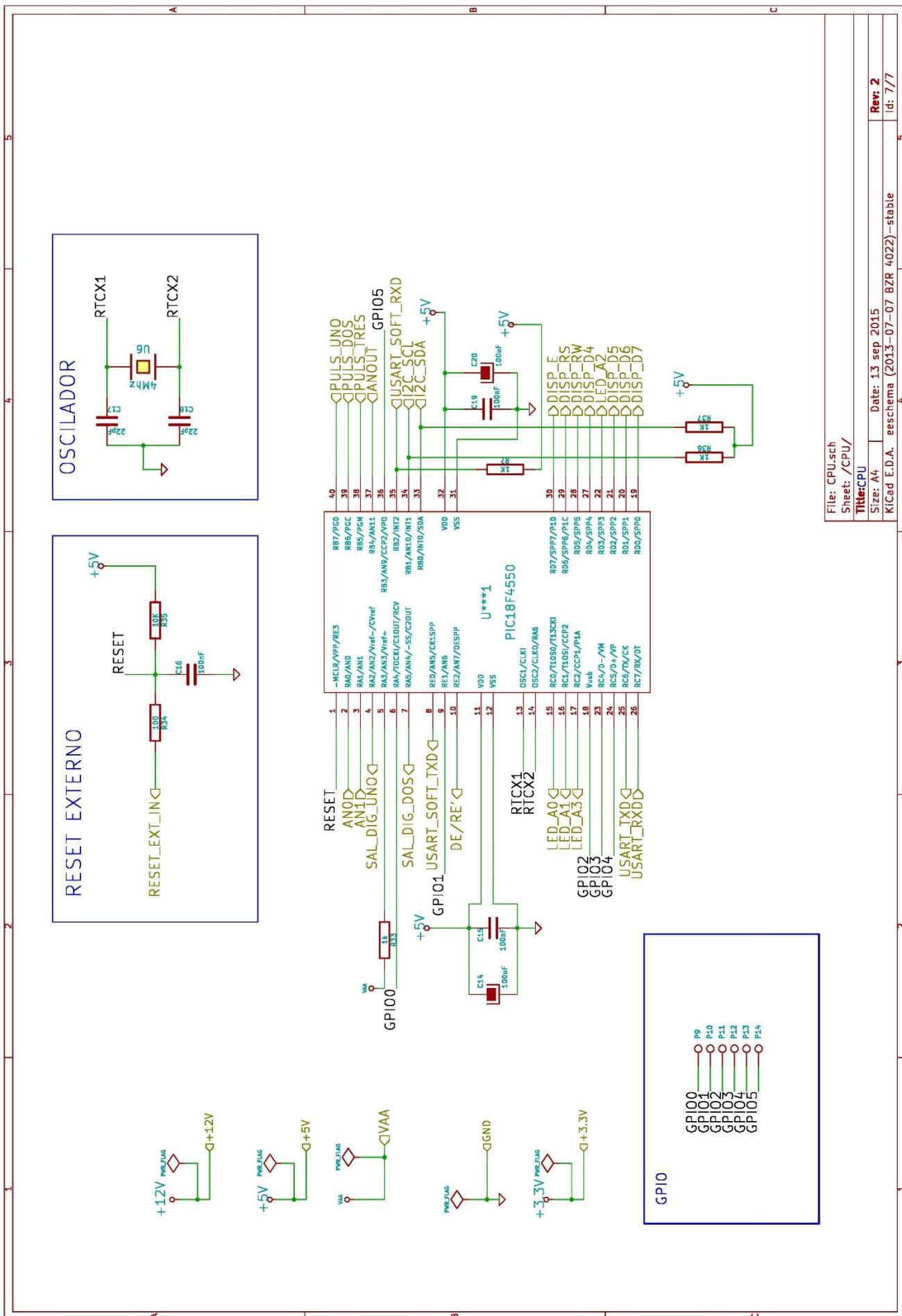
NOTA:

- 1)-Sacando el jumper JP1 el bus esta en modo eco, lo que transmite lo recibe.
- 2)-Si es el primer nodo de la linea sacar los jumpers JP2 y JP4.
- 3)-Si es el ultimo nodo de la linea colocar los jumpers JP2,JP3 y JP4.
- 4)-Si es un nodo intermedio sacar los jumpers JP2, JP3 y JP4.

File: RS485-BLUETOOTH.sch
Sheet: /RS485-BLUETOOTH/
Title: RS485-BLUETOOTH
Size: A4 Date: 13 sep 2015
KICad E.D.A. eeschema (2013-07-07 BZR 4022)-stable Rev: 4/7
Id: 4/7



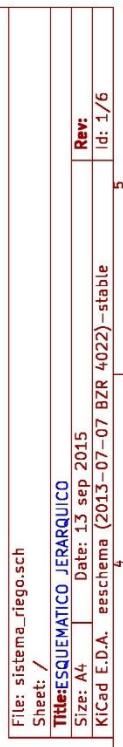
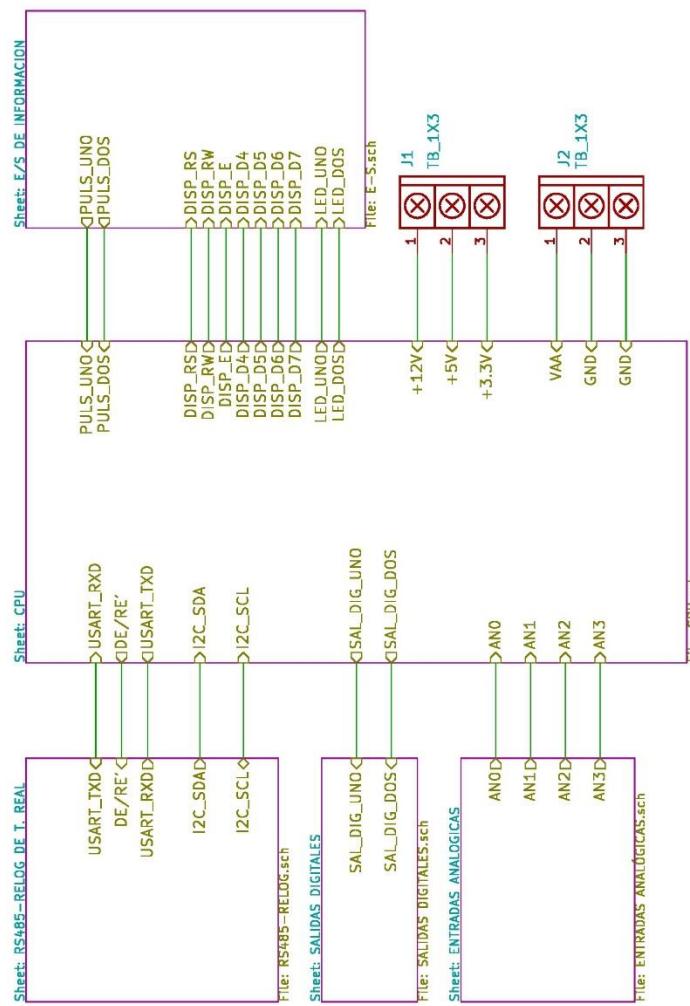


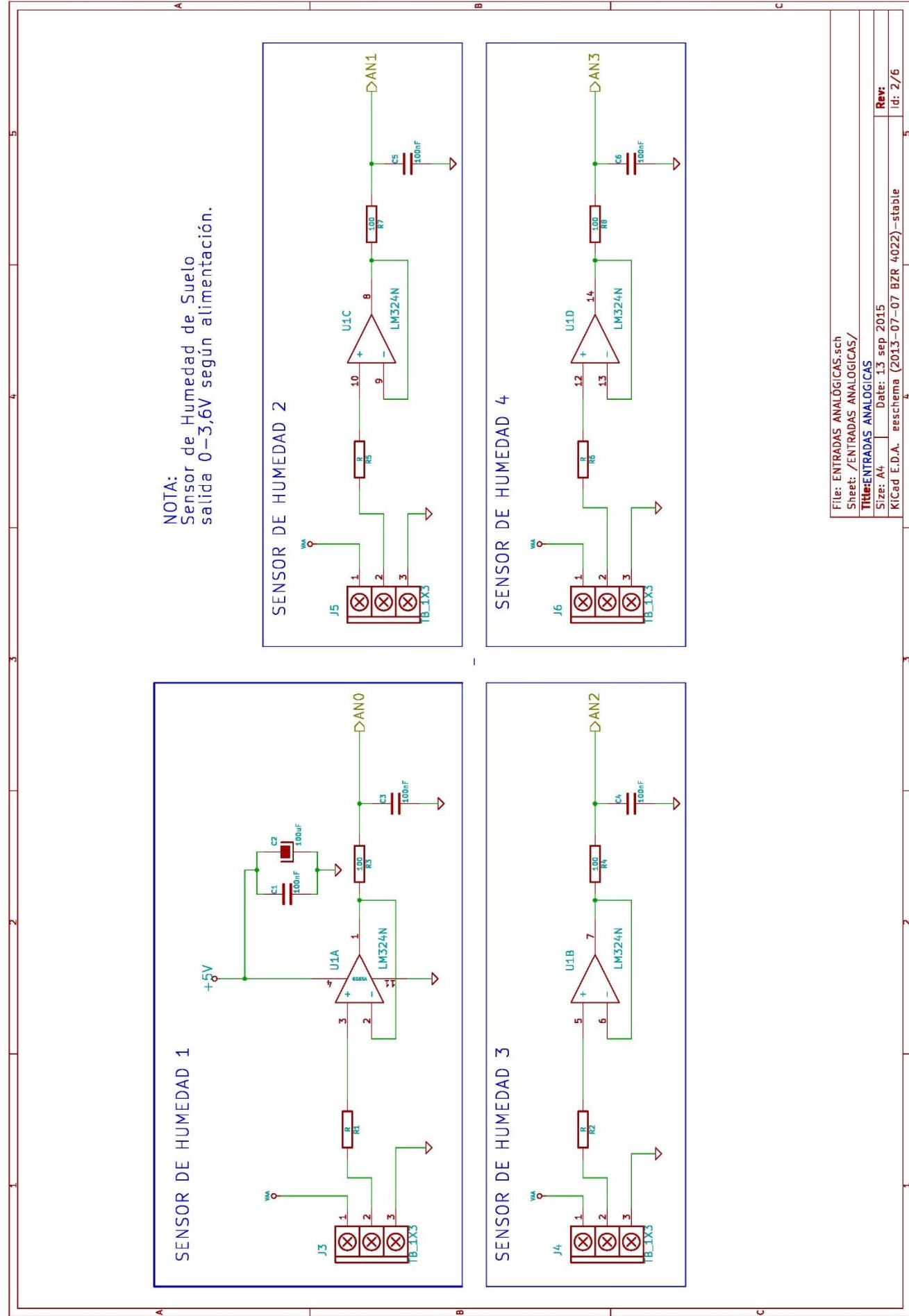


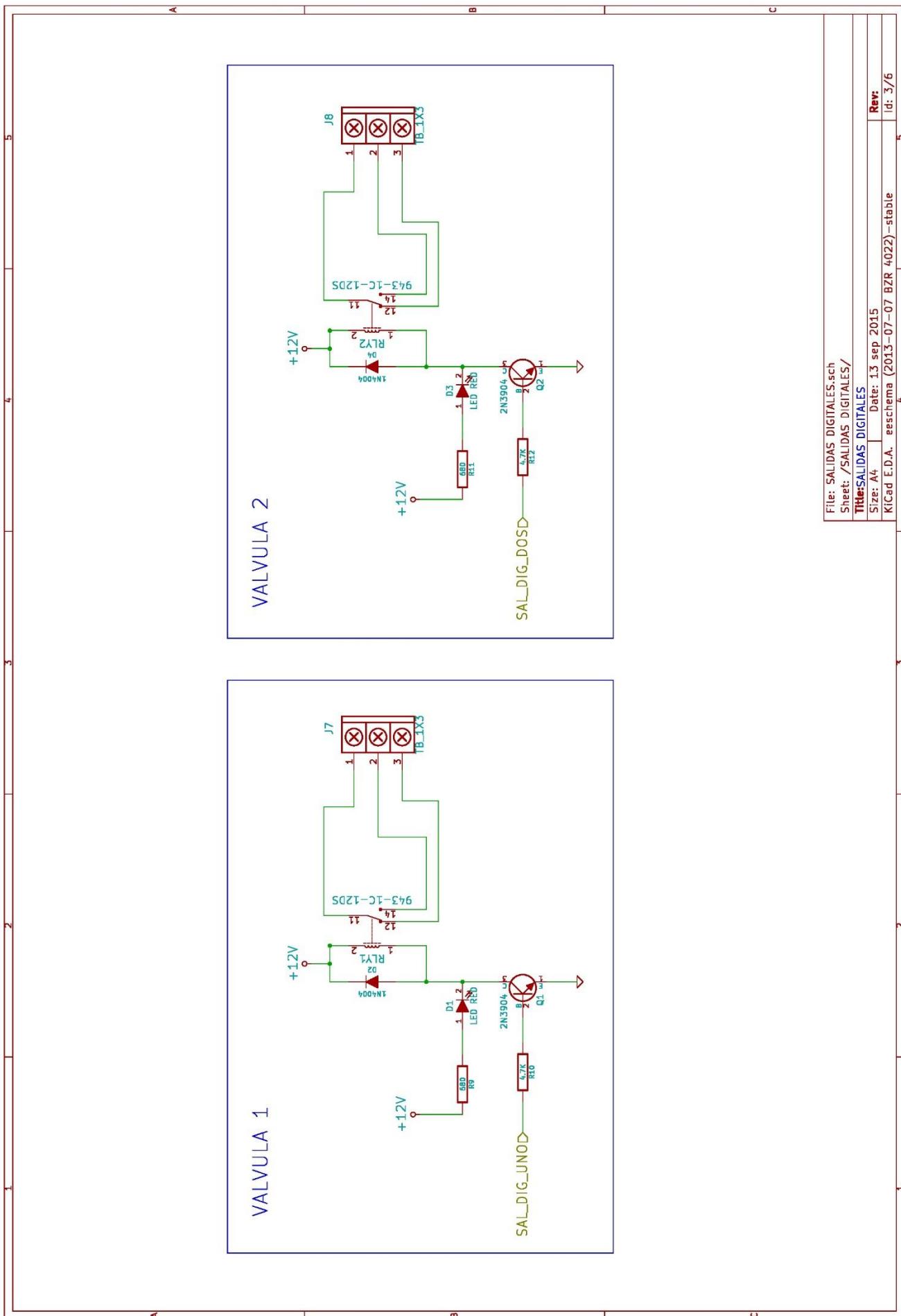
Sistema de Riego Estación Remota

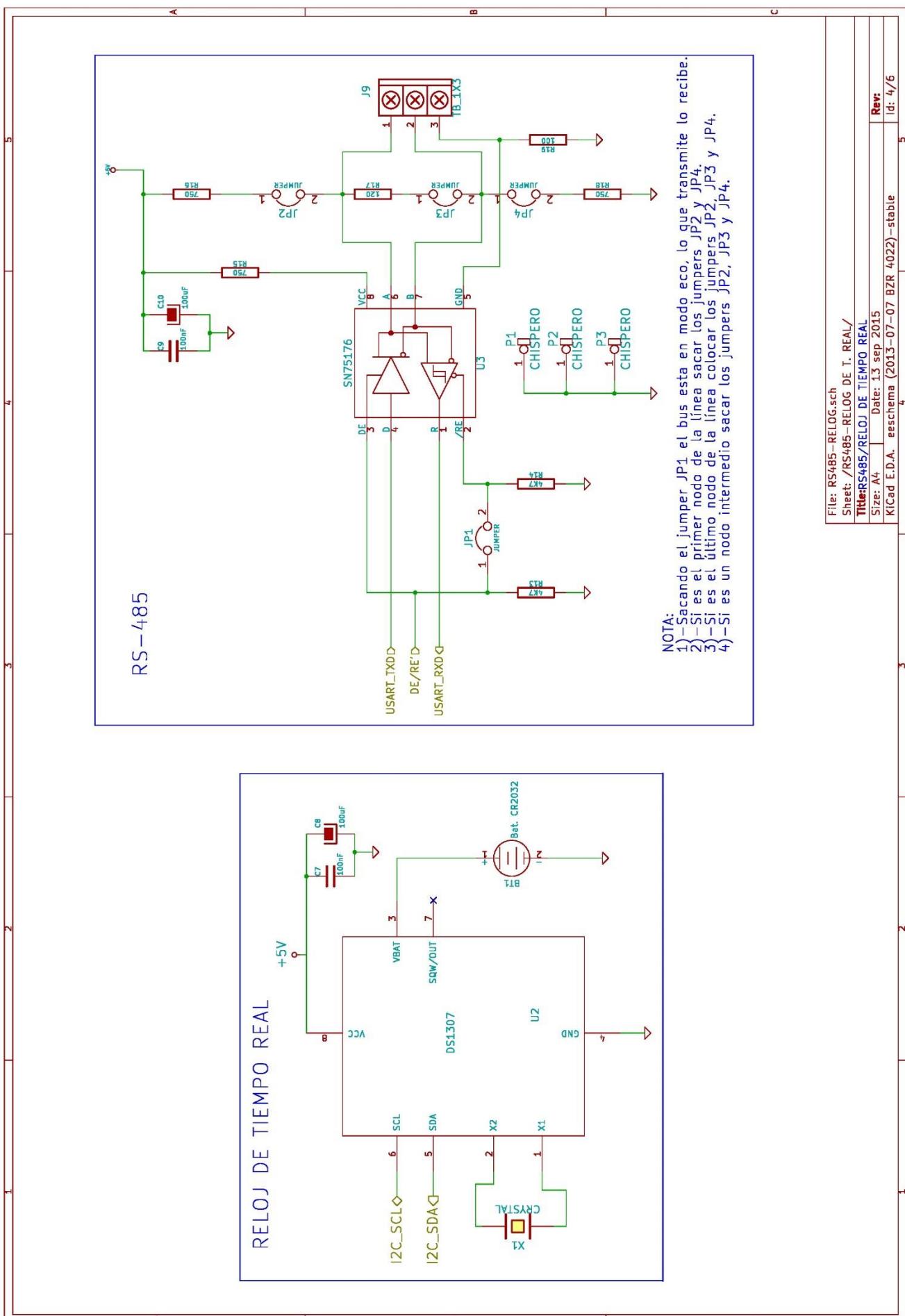
Índice:

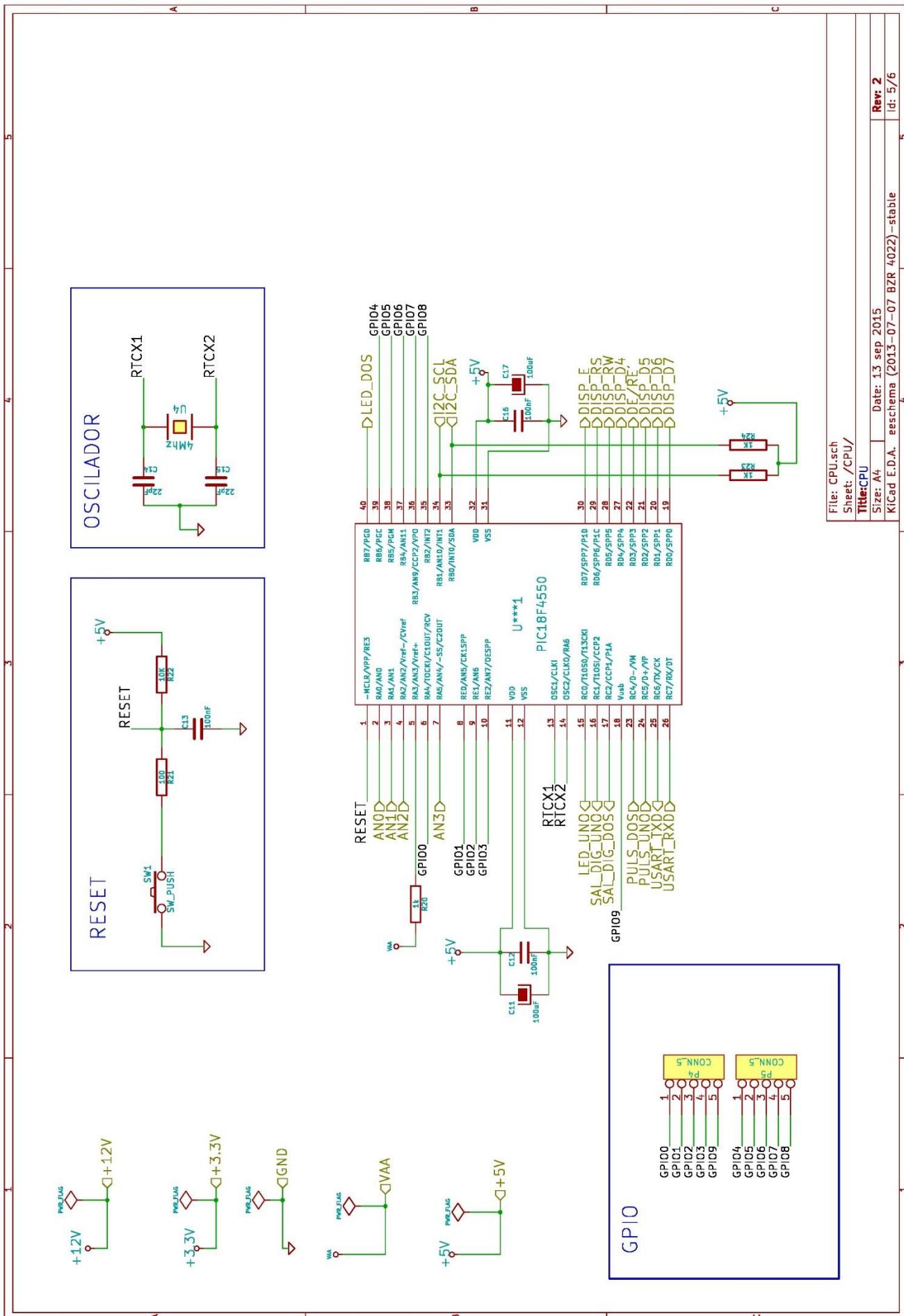
1. Esquemático Jerárquico
2. Entradas Analógicas
3. Salidas Digitales
4. RS485/Reloj de Tiempo Real
5. CPU
6. Entrada/Salida de Información

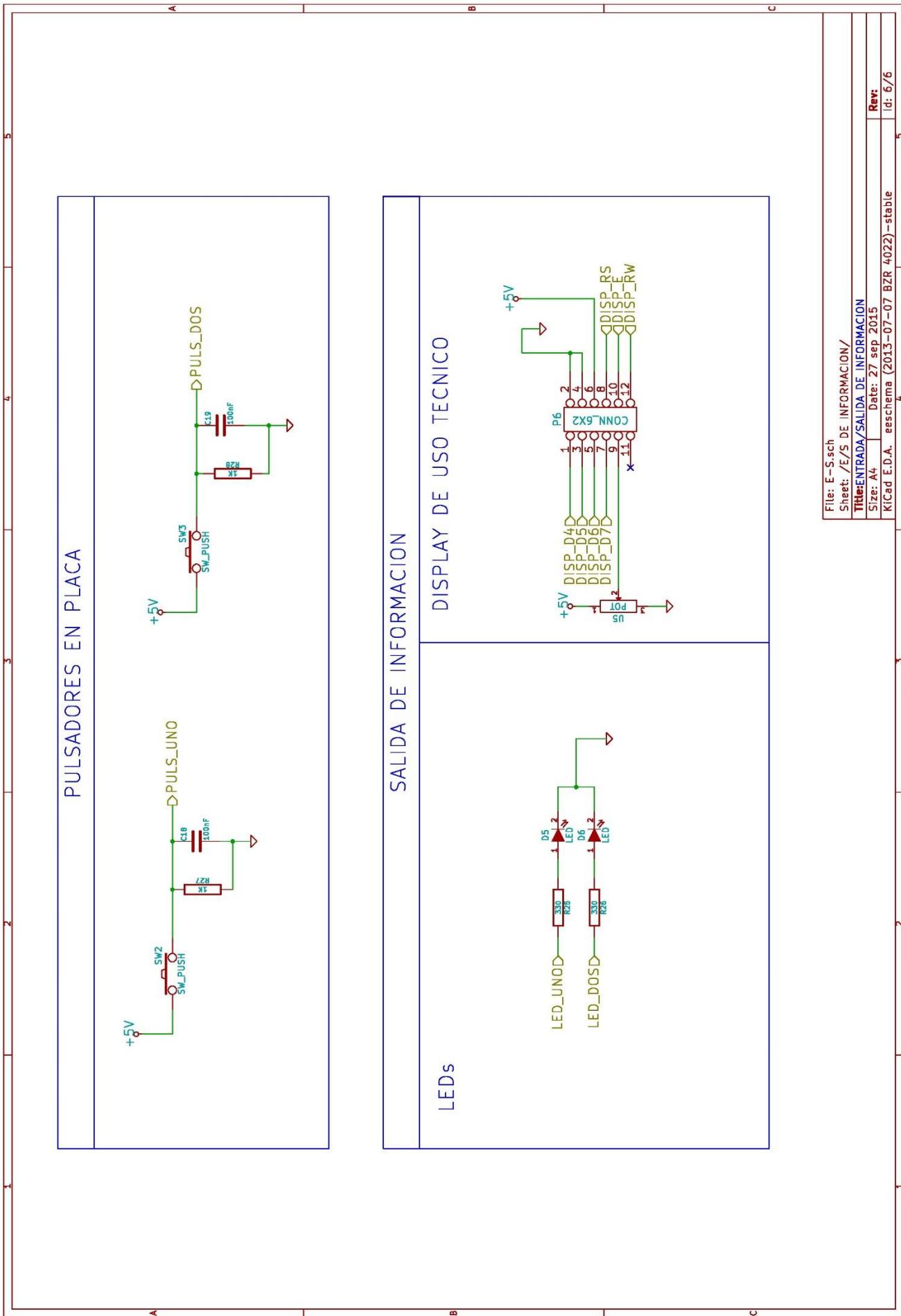










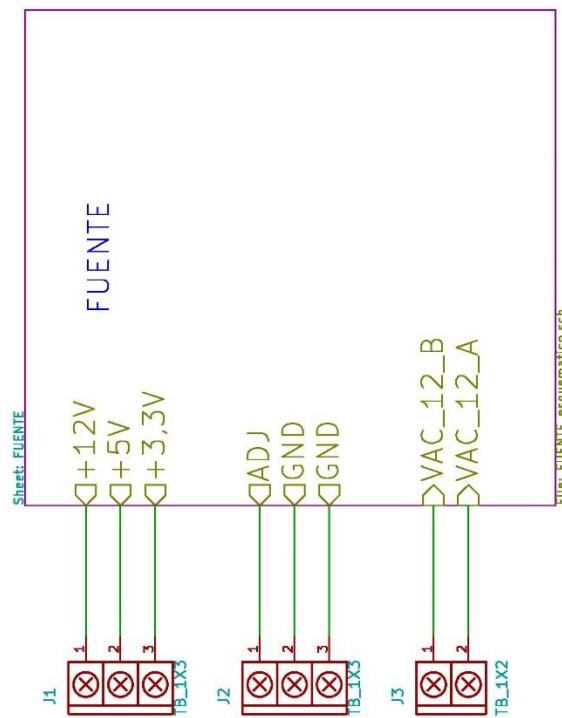


Sistema de Riego

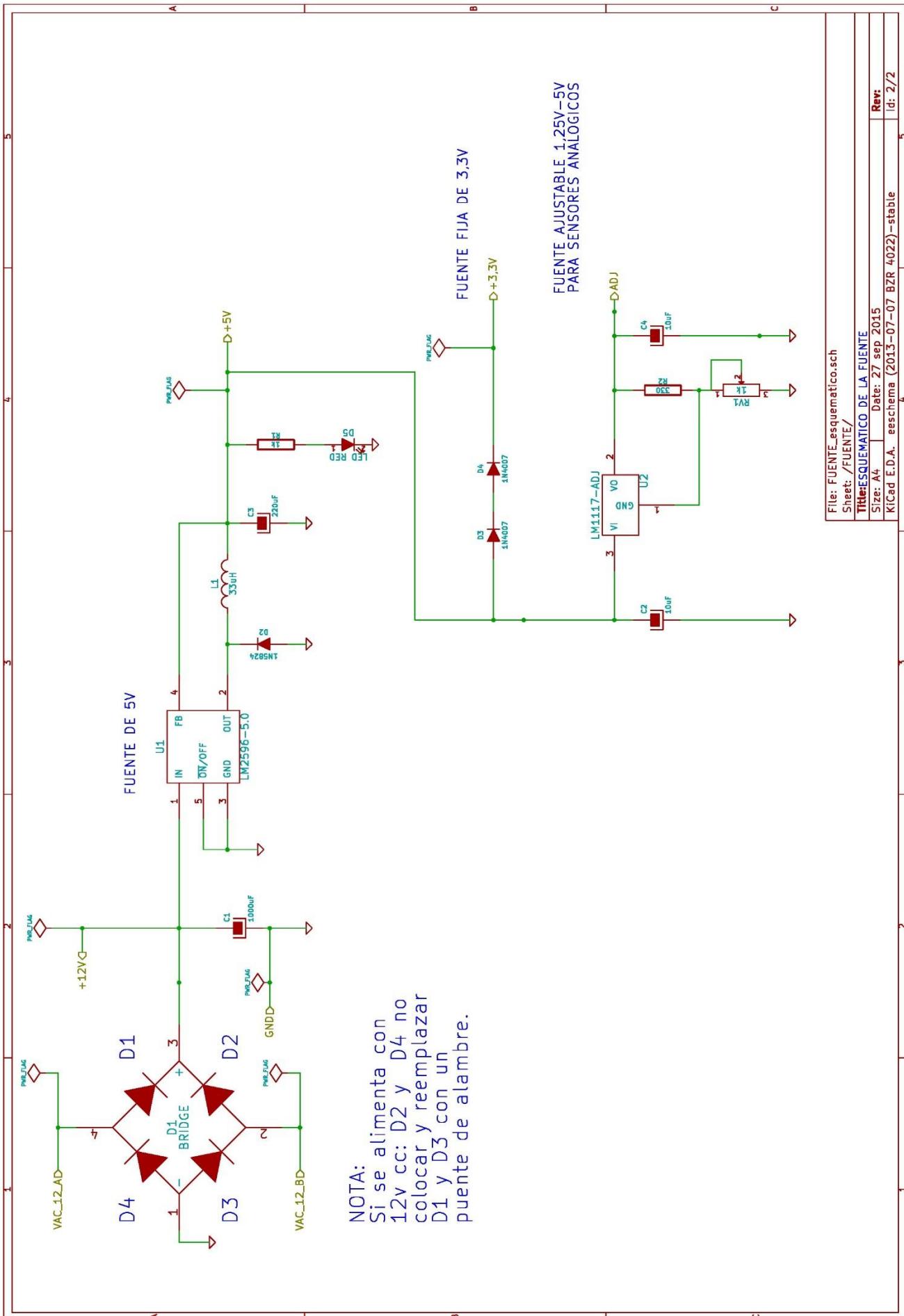
Fuente:

Índice:

1. Esquemático Jerárgico
2. Fuente



File: fuente.sch			
Sheet: /			
Title:ESQUEMATICO_JERARQUICO			
Size: A4	Date: 27 sep 2015	Rev:	
KiCad EDA	eeschema (2013-07-07 BZR 4022)-stable	Id: 1/2	5
			4
			3
			2
			1



UNIVERSIDAD NACIONAL DE CATAMARCA



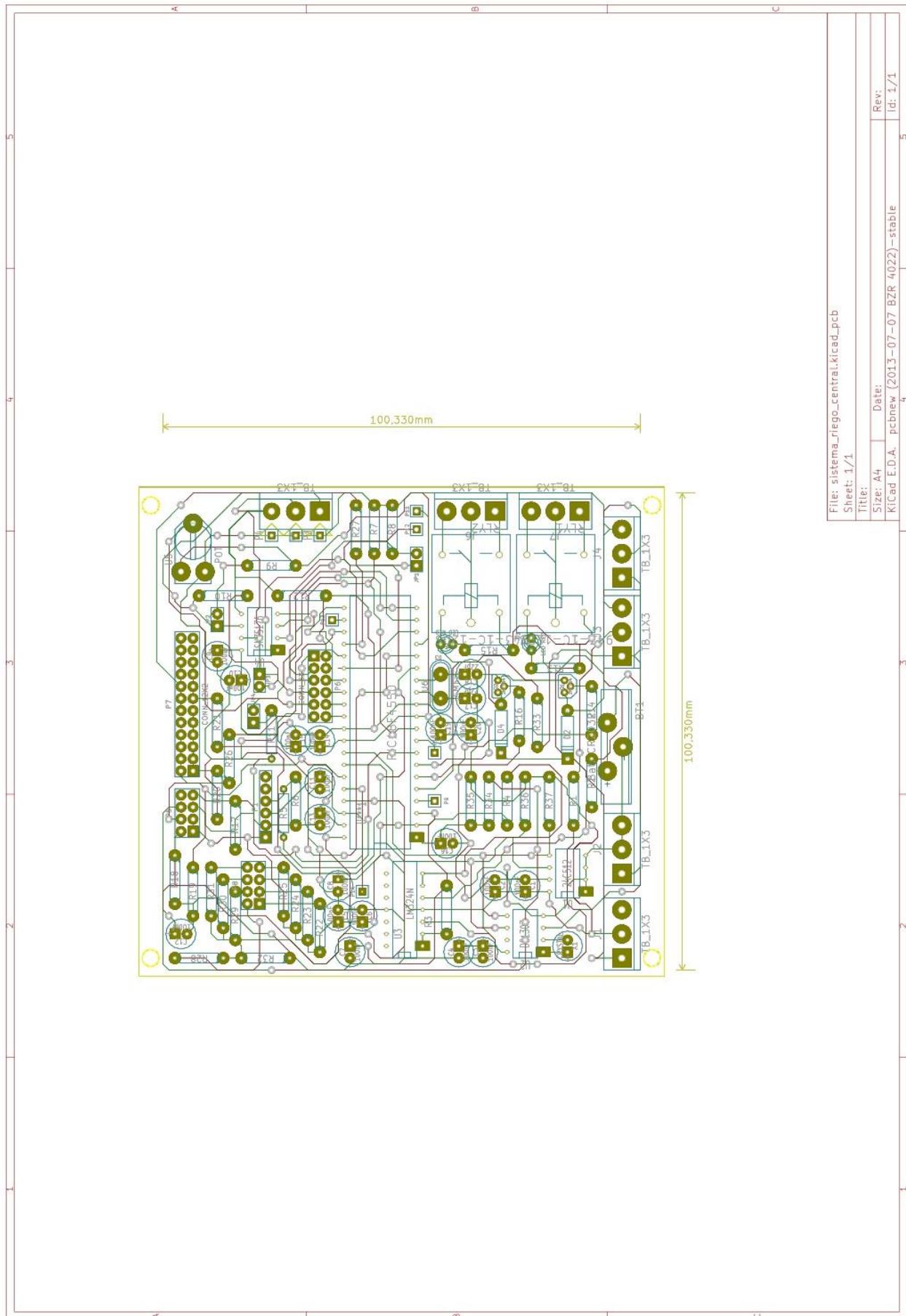
ANEXO II: CIRCUITOS IMPRESOS

Diseño de un sistema de riego para optimizar el recurso hídrico

Matías Leandro Ferraro

Año: 2015

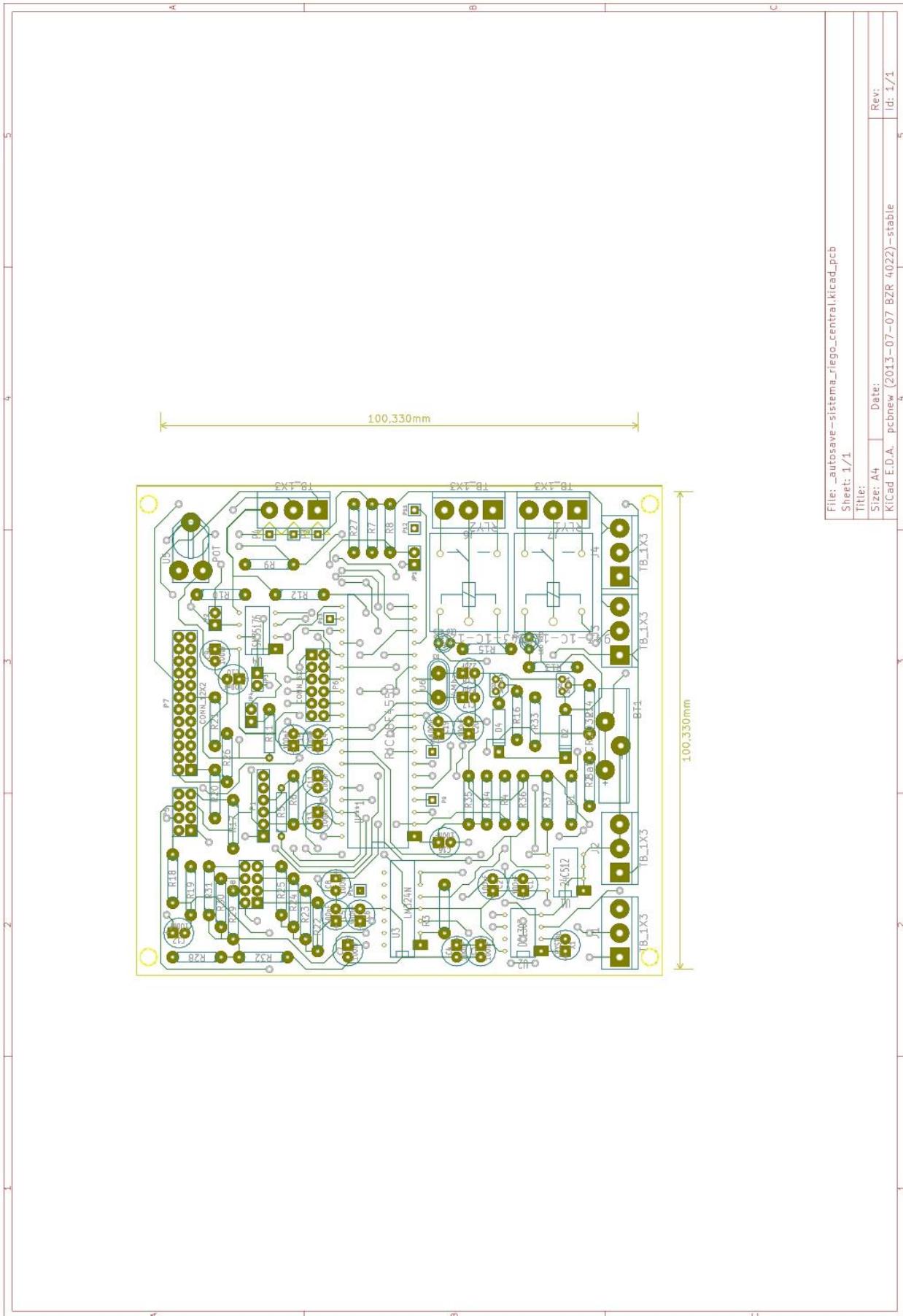
Título: Ingeniero Electrónico
Director de tesis: Sergio Hilario Gallina
Departamento: Electrónica



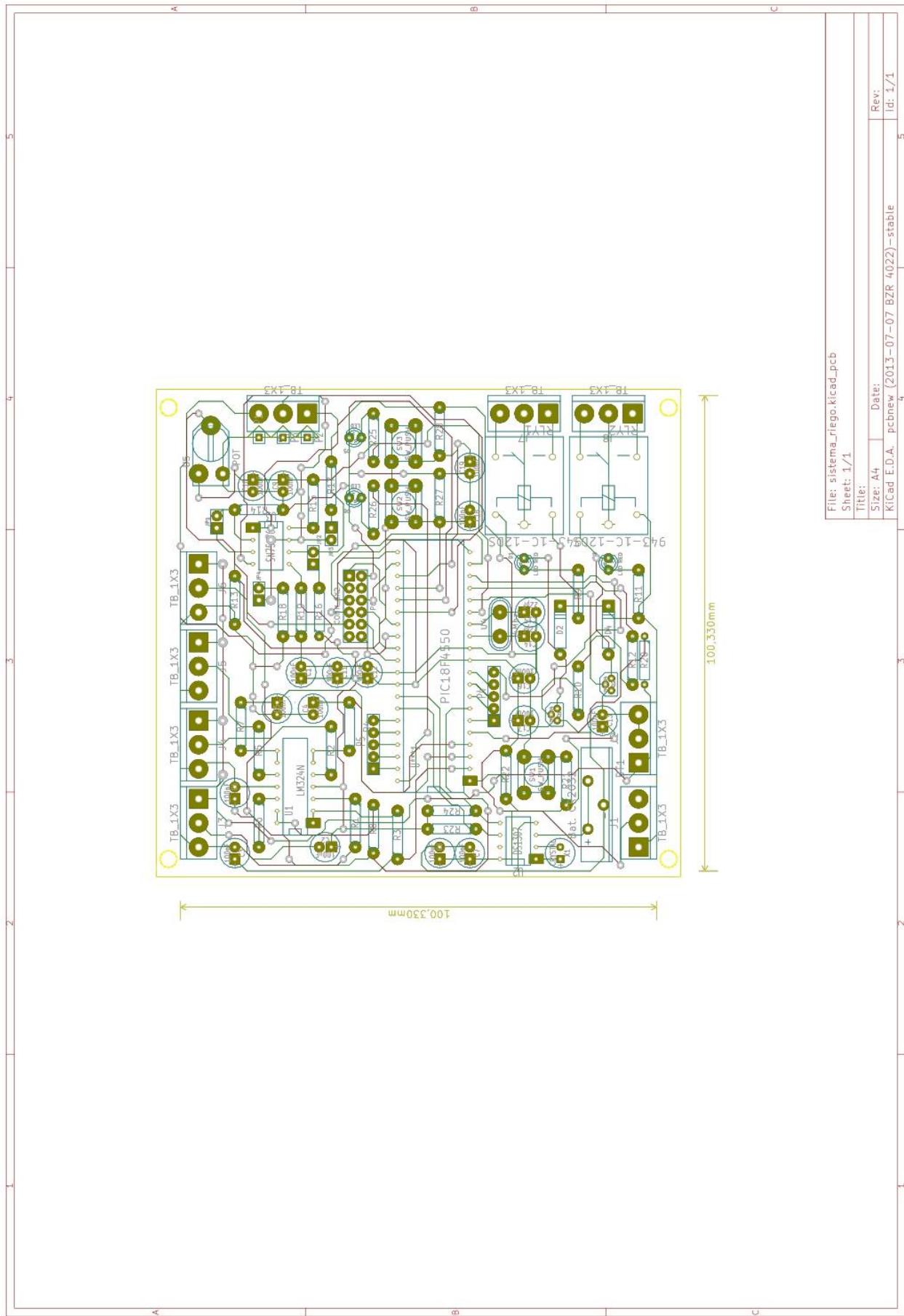
Plano 1: Circuito impreso de la estación concentradora de ambos lados



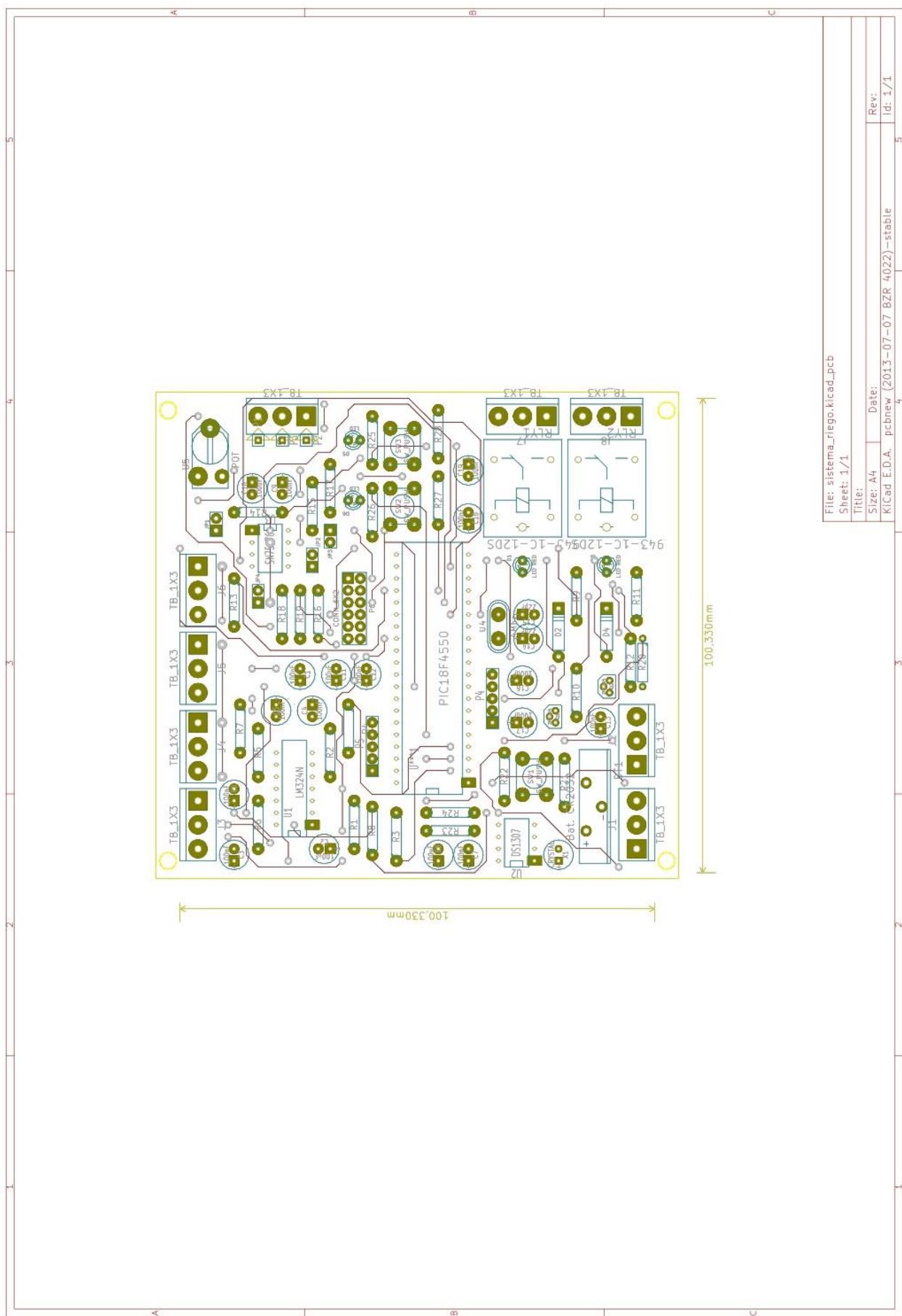
Plano 2: Circuito impreso de la estación concentradora del lado de los componentes



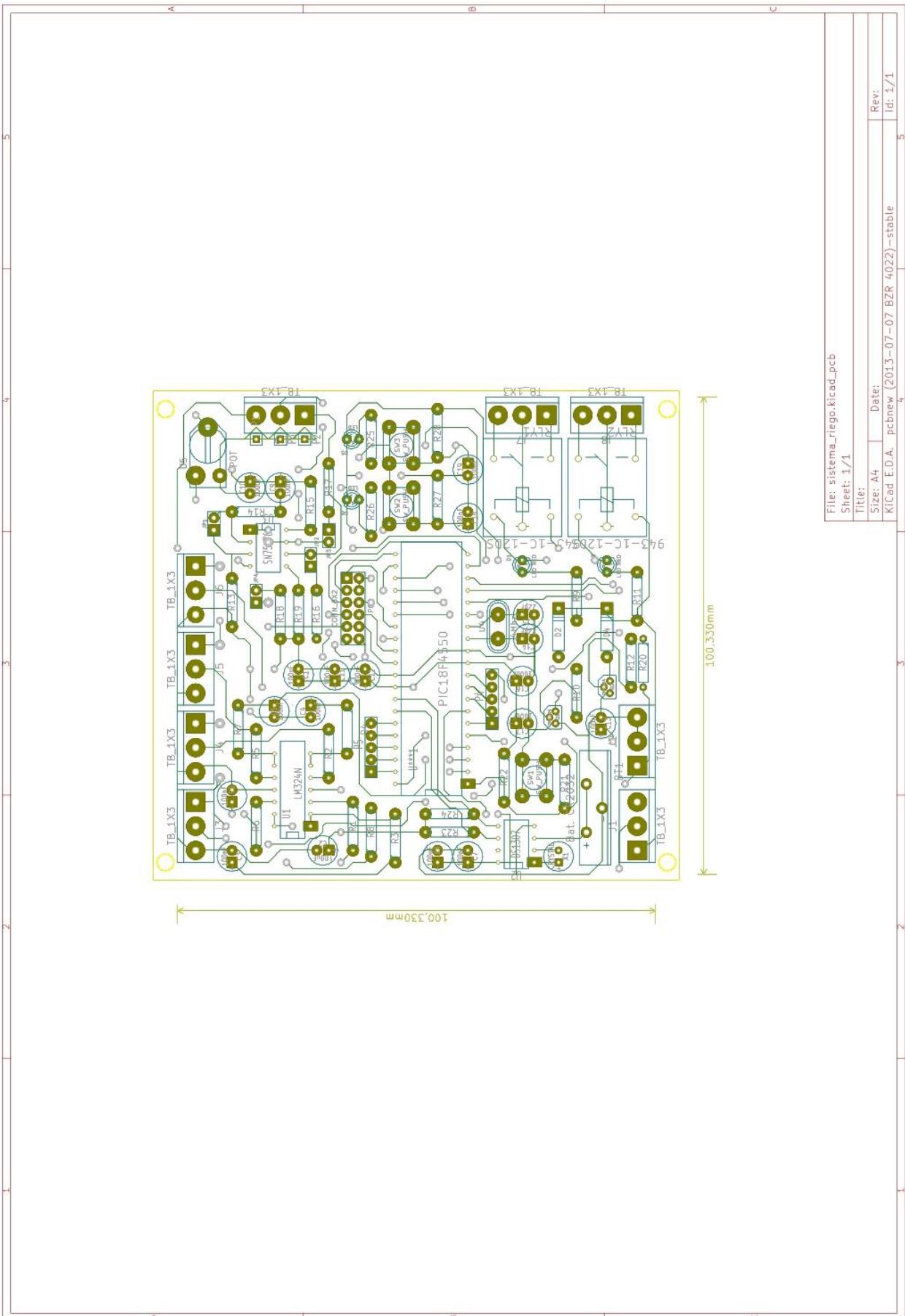
Plano 3: Circuito impreso de la estación concentradora del lado de abajo.



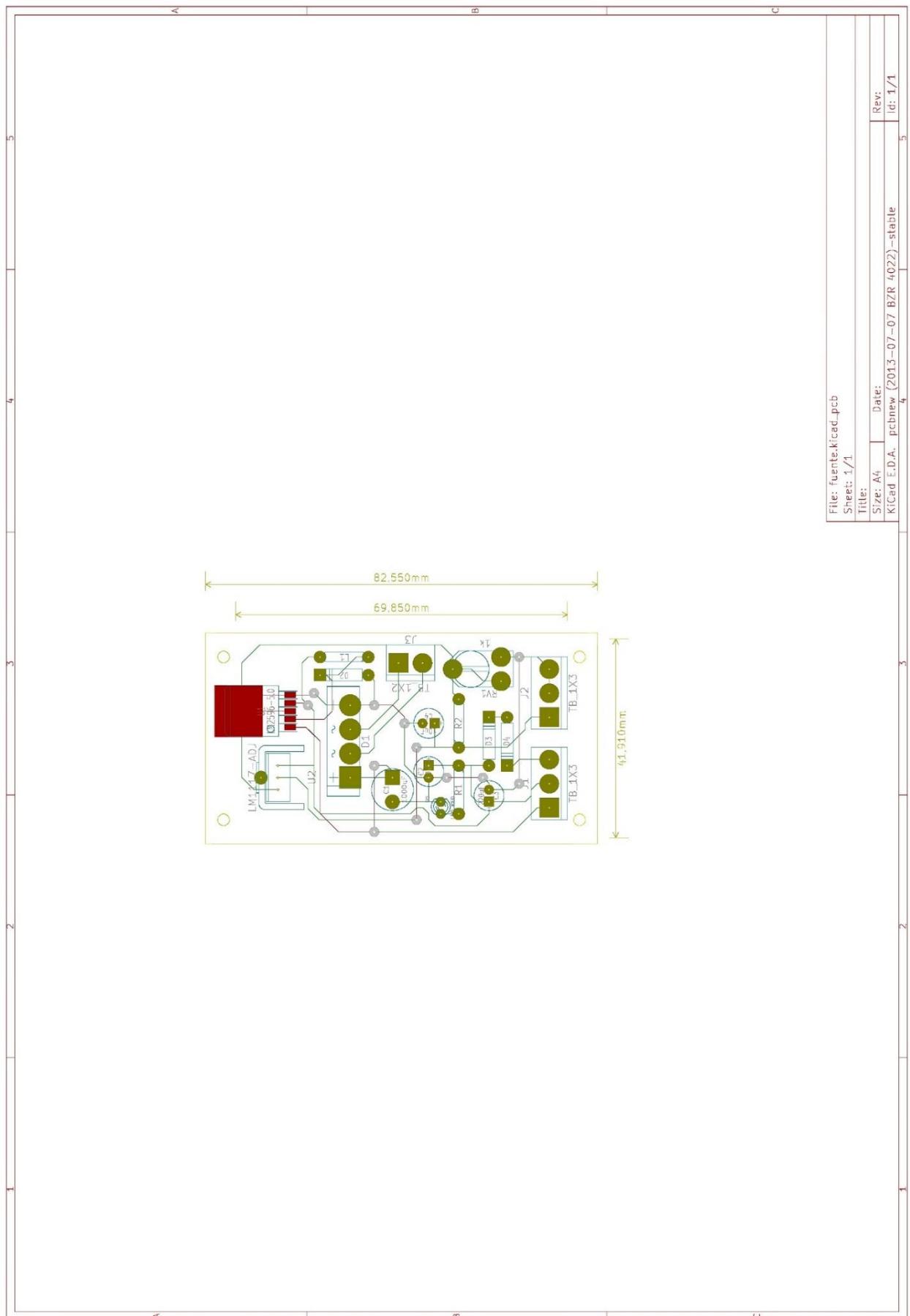
Plano 4: Circuito impreso del nodo remoto de ambos lados



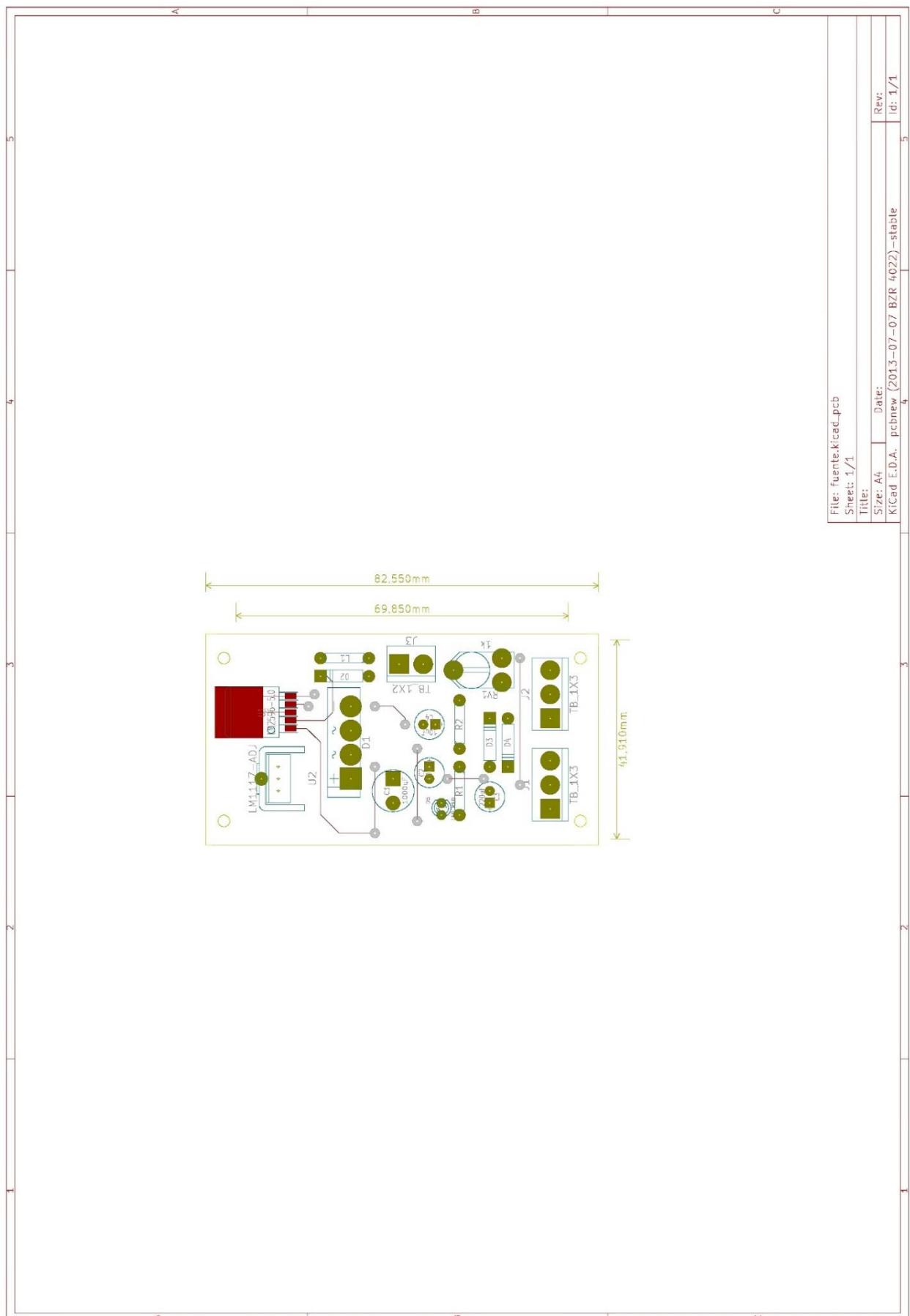
Plano 5: Circuito impreso del nodo remoto del lado de los componentes.



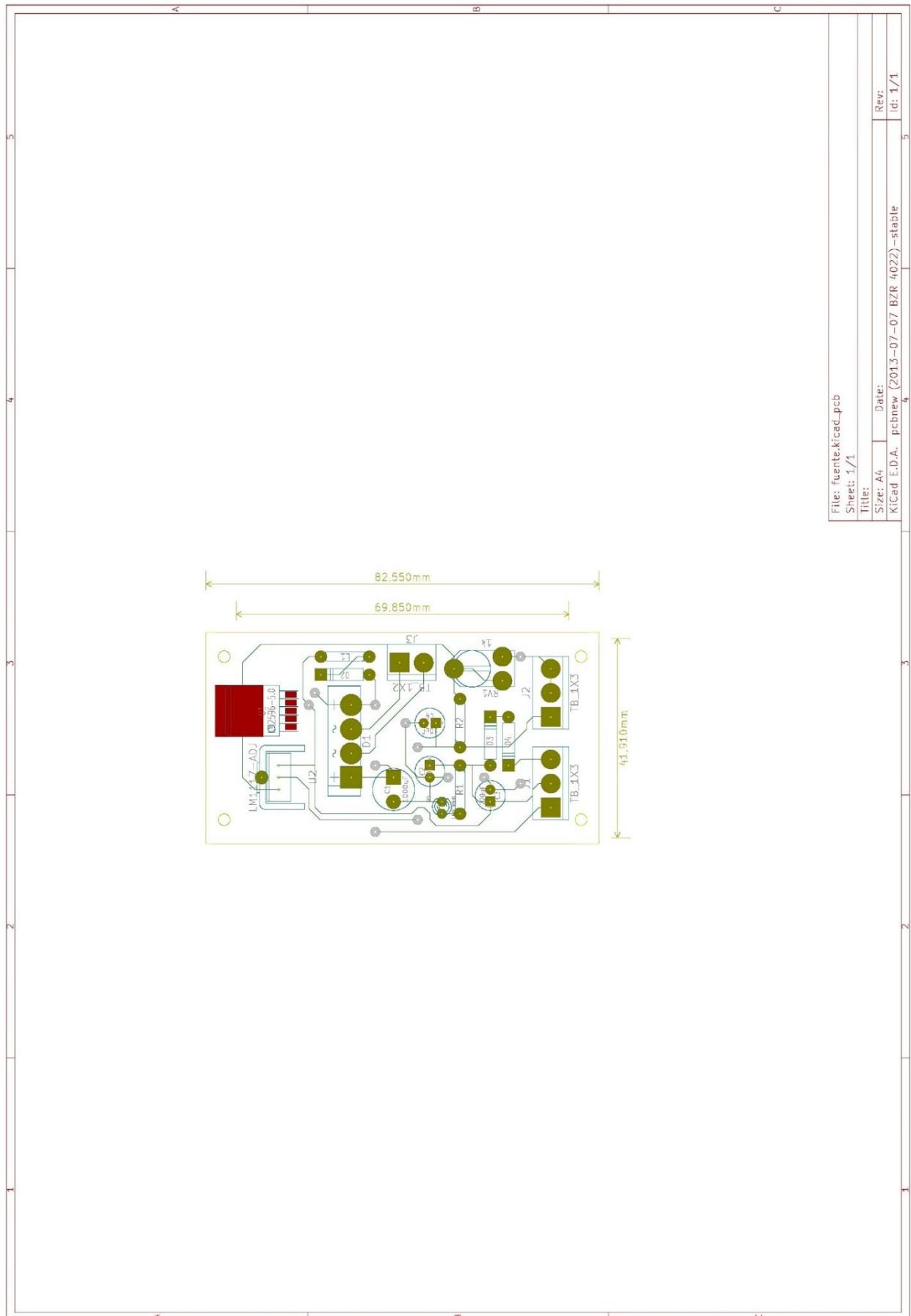
Plano 6: Circuito impreso del nodo remoto del lado de abajo.



Plano 7: Circuito impreso de la fuente de ambos lado.



Plano 8: Circuito impreso de la fuente del lado de los componentes.



Plano 9: Circuito impreso de la fuente del lado de abajo.

UNIVERSIDAD NACIONAL DE CATAMARCA



ANEXO III: CIRCUITOS ELECTRICOS

Diseño de un sistema de riego para optimizar el recurso hídrico

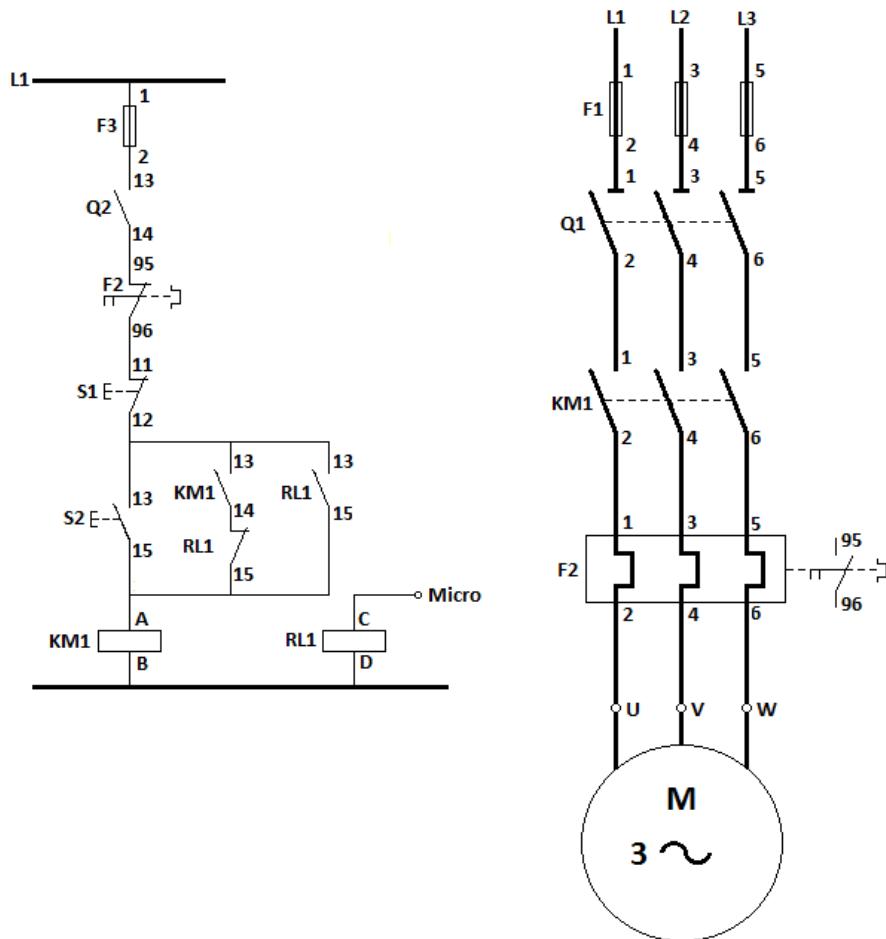
Matías Leandro Ferraro

Año: 2015

Título: Ingeniero Electrónico
Director de tesis: Sergio Hilario Gallina
Departamento: Electrónica

Para el diseño del sistema eléctrico general se ha relevado la instalación existente y se ha intentado incorporar el menor cambio posible en la misma.

La explotación agrícola posee dos bombas propulsadas por motores trifásicos asincrónicos con rotor en conexión jaula de ardilla de 1Hp, en el esquema eléctrico 1 se puede observar el esquema de maniobra y potencia del circuito eléctrico implementado:



Esquema eléctrico 1: Esquema de potencia y maniobra de la bomba de agua

El motor está comandado por un contactor y un relé, el relé está instalado en la placa de la estación concentradora y el contactor en el tablero de comando, además se dispone de un pulsador de marcha y otro de parada para activar la bomba de forma manual.

El esquema de potencia dispone de:

- Tres fusibles: F1.
- Seccionador trifásico: Q1.
- Contactor trifásico: KM1.
- Relé térmico de protección: F2.

El esquema de maniobra dispone de:

- Interruptor bipolar: Q2.
- Fusible: F3.
- Contacto cerrado del relé térmico: F2.
- Pulsador de paro: S1.
- Pulsador de marcha: S2.
- Contacto auxiliar del contactor KM1
- Contacto normalmente abierto del relé RL1.
- Contacto normalmente cerrado del relé RL1.
- Bobina KM1 correspondiente al circuito electromagnético del contactor KM1.
- Bobina RL1 del circuito electromagnético del relé RL1.

Funcionamiento:

El motor se puede poner en servicio de dos formas:

Accionando el pulsador S2:

Se implementó esta forma de riego mediante un pulsador de marcha y parada para permitir el riego que ya existía en la explotación agrícola. Al pulsar S2 se energiza la bobina KM1 del contactor, provocando que se cierre el contacto KM1 NA, al estar el sistema de riego electrónico inactivo el contacto NC del relé RL1 está cerrado provocando que se encalte el contactor, si se presiona S1 la bobina de KM1 se desenergiza con lo cual el contacto KM1 NA abre y la bomba se detiene.

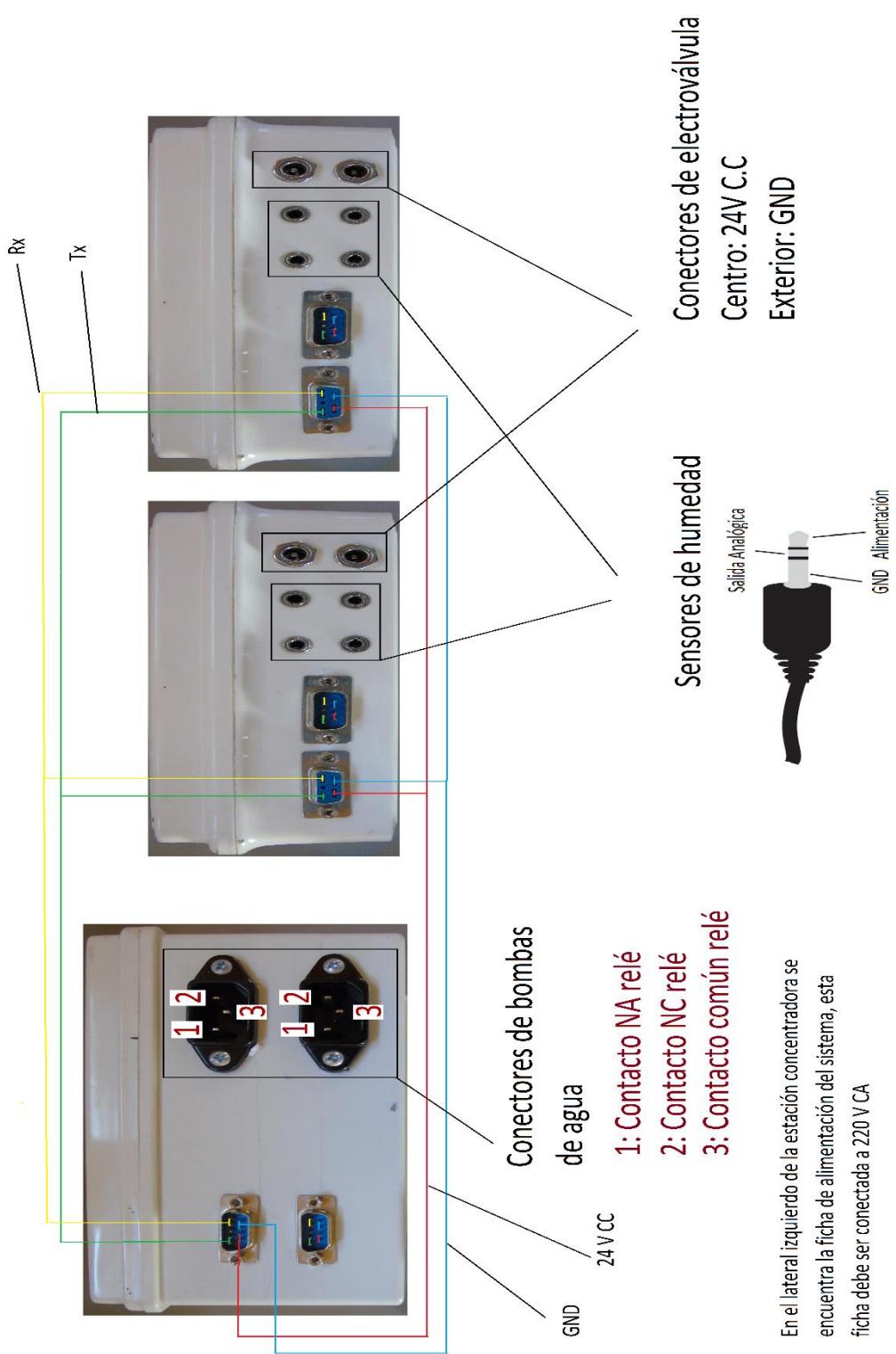
Energizando la bobina de RL1:

Al enviar el microcontrolador una señal de activación de la bobina del relé RL1, este cierra el contacto NA, con lo cual se energiza KM1, la señal que entrega el microcontrolador al relé permanece en alto el tiempo que el sistema electrónico necesita tener activa la bomba de agua.

Si debido a una sobre intensidad el relé térmico F2 acciona su contacto, abriría el circuito de maniobra desconectando la bobina KM1 y en consecuencia parando el motor.

Ante la necesidad de parar de emergencia la bomba estando controlada por el sistema electrónico de riego siempre es posible abrir el interruptor bipolar Q2.

En el esquema eléctrico 2 se puede observar las conexiones generales del sistema implementado.



Esquema eléctrico 2: Conexiones generales del sistema implementado

UNIVERSIDAD NACIONAL DE CATAMARCA



ANEXO IV: MANUAL DE INSTALACION RAPIDA

Diseño de un sistema de riego para optimizar el recurso hídrico

Matías Leandro Ferraro

Año: 2015

Título: Ingeniero Electrónico
Director de tesis: Sergio Hilario Gallina
Departamento: Electrónica

MANUAL DE INSTALACION RAPIDA

En el gráfico 1 se puede observar el esquema general de conexiones del sistema de riego a instalar.

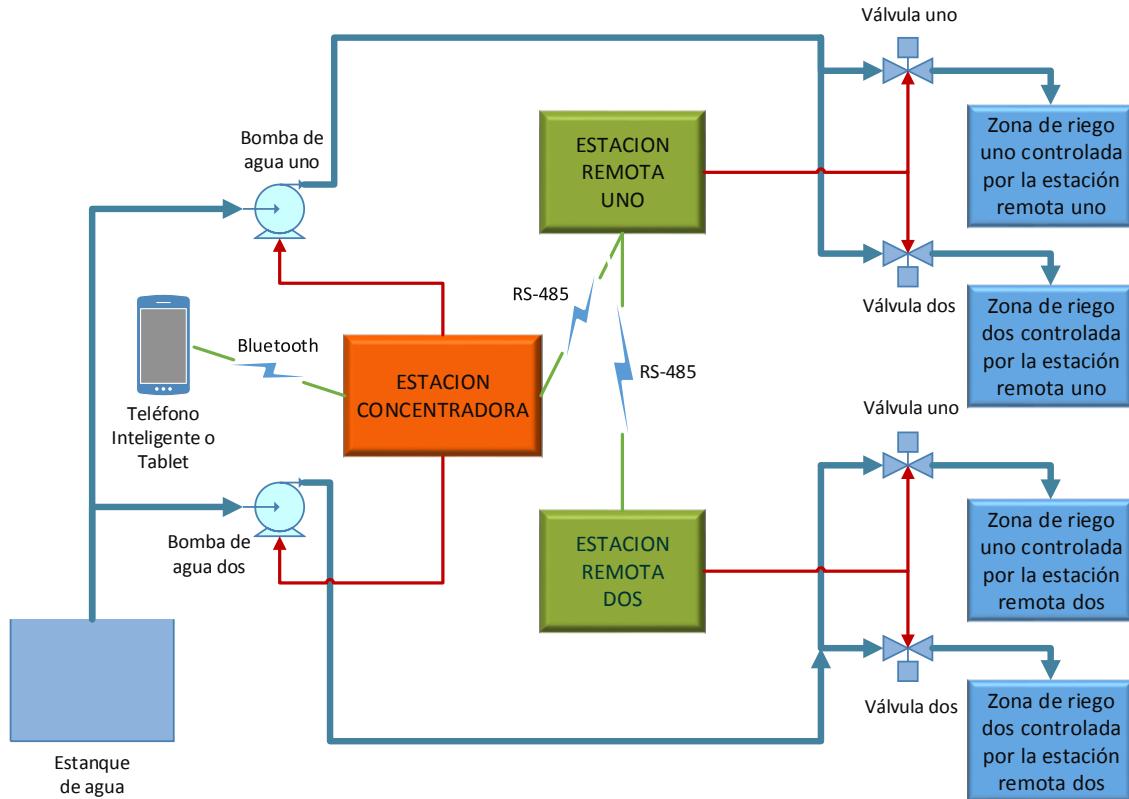


Gráfico 1: Esquema general de conexiones

El sistema de riego está compuesto por tres gabinetes plásticos, uno es la estación concentradora y los otros dos son las estaciones remotas, en la fotografía 1 se puede observar el frente de la estación concentradora y en la fotografía 2 el frente de la estación remota.



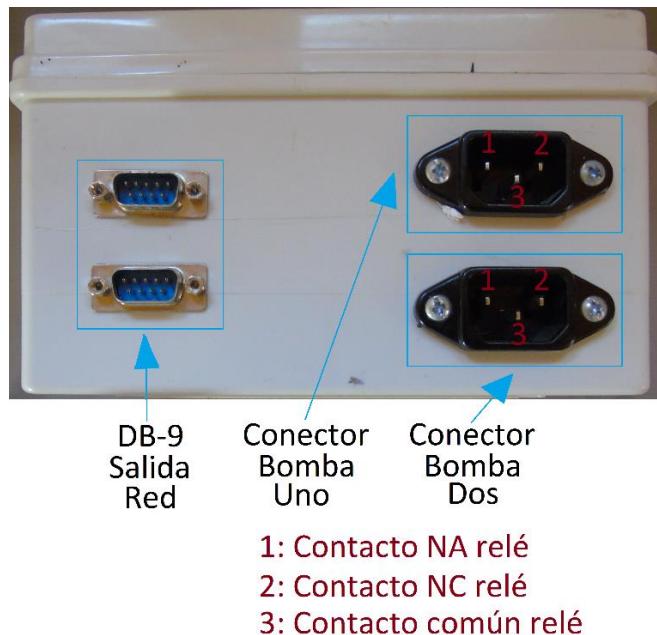
Fotografía 1: Frente de la estación concentradora



Fotografía 2: Frente de la estación remota

La estación concentradora posee dispositivos de entrada y de salida de información: un teclado matricial de dieciséis teclas, un display de cristal líquido con iluminación de fondo azul, dos leds indicadores de la operatividad de las estaciones remotas, un botón de encendido, un puerto de comunicaciones bluetooth que le permite conectarse con un teléfono inteligente o una Tablet para acceder a los parámetros de riego y programación del sistema, en la fotografía 1 se puede observar los elementos que componen la estación concentradora.

Mediante un bus de comunicaciones RS-485 se conecta la estación concentradora con las estaciones remotas utilizando una ficha DB-9,ademas la estación concentradora cuenta con dos enchufes macho para el manejo de dos bombas de agua, cada uno provee las señales necesarias para el control de cada bomba, cada enchufe posee tres terminales, estos terminales son las salidas del relé de manejo de la bomba, uno de estos terminales está conectado con el contacto normal abierto otro al contacto normal cerrado y el tercero al contacto común del relé de manejo de la bomba de agua, en la fotografía 3 se pueden observar los conectores.



Fotografía 3: Vista inferior de la estación concentradora

En el Esquema eléctrico 1 del Anexo III se describe la forma de conexión de potencia y maniobra de la bomba de agua, las bombas de agua a utilizar deben poseer como medio de propulsión un motor trifásico asincrónico con rotor en conexión jaula de ardilla de 1Hp.

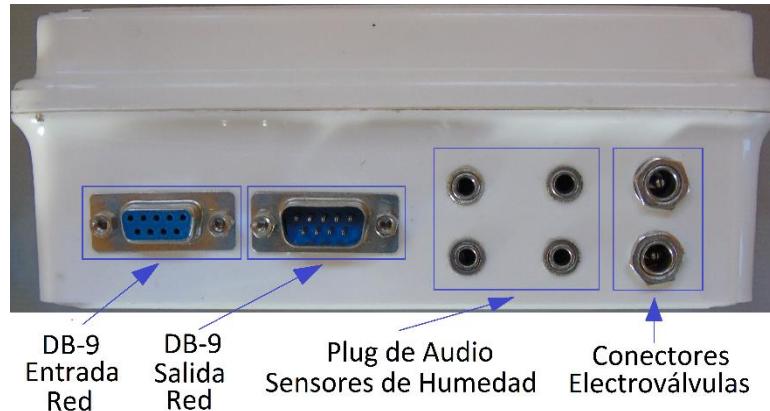
En la fotografía 4 se puede observar la ficha macho instalada en el gabinete de la estación concentradora utilizada para proveer de alimentación al sistema, la que se debe conectar a 220 V de corriente alterna mediante el cable suministrado y esta estación genera las tensiones necesarias para el correcto funcionamiento del sistema de riego.



Fotografía 4: Vista lateral izquierda de la estación concentradora

Las estaciones remotas poseen cuatro conectores, estos conectores son de tipo plug de audio y permiten conectar como máximo cuatro sensores de humedad analógicos de la marca Decagon Devices el modelo EC5, dos sensores por cada zona como máximo, pero puede conectarse un sensor de humedad por zona o ninguno por zona, en el caso de no tener ningún sensor instalado en una zona, esa zona admite solamente el riego por franja horaria y de forma manual, el sistema permite la desconexión de sensores con el sistema en funcionamiento, provocado de forma intencional o fortuita por la rotura del sensor o del cable de conexión, además posee dos fichas destinadas a la conexión de dos electroválvulas, una por cada zona y dos fichas DB-9 una hembra y la otra macho, en la ficha DB-9 hembra se conecta el bus proveniente de la estación anterior y la ficha DB-9 macho permite conectar la estación siguiente, las estaciones se pueden conectadas en serie o con un cable individual desde la estación concentradora a cada

estación remota , en la fotografía 5 se pude observar la disposición de los conectores en la estación remota.



Fotografía 5: Vista inferior de la estación remota

El bus de comunicaciones se puede conectar de dos formas, en conexión serie o con un cable individual desde la estación concentradora a cada una de las estaciones remotas, en el gráfico 3 se puede observar la conexión serie y en el gráfico 4 la conexión individual.

Cada tramo de cable de red debe poseer un conector hembra y otro macho DB-9 provisto con el sistema de riego, este cable debe estar protegido de las inclemencias del tiempo y los roedores, el cable de las electroválvulas debe ser protegido de la misma forma.



Gráfico 3: Conexión del bus de comunicaciones RS-485 en serie

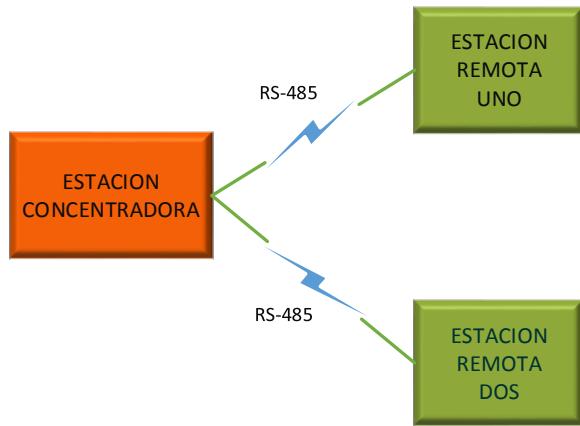


Gráfico 4: Conexión del bus de comunicaciones RS-485 con un cable por estación remota

Cada estación remotas se proveen con una casilla metálica para protegerlas de las inclemencias del tiempo, dotarlas de robustez y protección contra el vandalismo, están provistas de una estaca para fijación, en la fotografía 6 se puede observar la casilla metálica suministrada.



Fotografía 6: Casilla metálica de protección de la estación remota

PASOS A SEGUIR PARA LA INSTALACION DEL SISTEMA (CONEXIÓN SERIE)

1)-Instale la estación concentradora en la sala de bombas, en un lugar protegido de las inclemencias del tiempo y de los roedores ya que posee elementos que se pueden deteriorar.

2)-Instale las casillas metálicas en la zona de riego lo más cercana posible de las electroválvulas.

3)-Coloque las estaciones remotas en las casillas y ajuste los tornillos de anclaje de las estaciones remotas a las casillas.

4)-Instale el ducto por el cual pasara el cable de red, el ducto debe partir de la sala de bombas y llegar a la primera estaciones remota y luego otro tramo desde la primera estación remota hasta la segunda estación remota.

5)-El sistema se provee con dos tramos de cable de red, por el ducto enterrado pasé el cable de red provisto con sus dos conectores DB-9 por tramo, uno hembra y otro macho, en la sala de bombas debe quedar un conector DB-9 hembra y debe pasar por el ducto el cable y llegar a la primera estación remota de la red con un conector DB-9 macho y de esta estación debe partir el otro tramo de cable provisto dentro del otro ducto quedando en esta primera estación remota el conector DB-9 hembra y terminando en la segunda estación remota con un conector DB-9 macho, el esquema de conexión es el mostrado en el gráfico 3.

6)-Conecte en la sala de bombas el conector de red DB-9 hembra.

7)-Conecte la ficha DB-9 hembra y la DB-9 macho a la estación remota uno.

8)- En la segunda estación remota de la red conecte la ficha DB-9 macho.

9)- Instale un ducto entre las estaciones remotas y las electroválvulas destinado a proteger el cable de conexión de las electroválvulas.

10)-Instale las electroválvulas provistas con el sistema pasando el cable por el ducto instalado en el punto 9).

11)-Instale un ducto para el cable de los sensores partiendo de la estación remota uno y llegando al lugar donde se quiere controlar el riego por humedad, de similar forma proceda con la estación remota dos.

12)-Pase por los ductos instalados en el punto 11) los cables de los sensores y conecté el plug de audio que posee cada sensor a la estación remota correspondiente.

13)-Realice las conexiones del circuito de maniobra y potencia según los esquemas eléctricos suministrado en el esquema eléctrico 1 del Anexo III.

14)-Conecte en la sala de bombas el cable de alimentación de la estación concentradora y presione el botón de encendido.

PASOS A SEGUIR PARA LA INSTALACION DEL SISTEMA (CONEXIÓN CON UN CABLE INDIVIDUAL PARA CADA ESTACION REMOTA):

Repita los pasos 1, 2, 3 de la instalación del sistema (conexión serie).

4)-Instale el ducto por el cual pasara el cable de red, deben ser dos tramos, los dos parten de la sala de bombas y llegan a cada una de las estaciones remotas.

5)-El sistema se provee con dos tramos de cable de red, por los dos ductos enterrados pase un tramo de cable provisto por ducto, en la sala de bombas deben quedar dos fichas DB-9 hembra y en cada una de las estaciones remotas una ficha DB-9 macho, el esquema de conexión es el mostrado en el gráfico 4.

6)-Conecte en la sala de bombas los dos conectores de red DB-9 hembra.

7)-Conecte la ficha DB-9 macho a la estación remota uno.

8)-Conecte la ficha DB-9 macho a la estación remota dos.

Repita los pasos 9, 10, 11, 12, 13, 14 de los pasos a seguir para la instalación del sistema (conexión serie).

UNIVERSIDAD NACIONAL DE CATAMARCA



ANEXO V: CODIGO FUENTE

Diseño de un sistema de riego para optimizar el recurso hídrico

Matías Leandro Ferraro

Año: 2015

Título: Ingeniero Electrónico
Director de tesis: Sergio Hilario Gallina
Departamento: Electrónica

Código fuente:

Código del software embebido en el microcontrolador de la estación concentradora:

```
C:\código_estacion_concentradora\código_estacion_concentradora.c
1: #include <18F4550.h>
2: #device adc=10
3: #FUSES NODEBUG
4: #FUSES VREGEN
5: #FUSES NOPBADEN
6: #FUSES NOMCLR
7: #FUSES PUT
8: #FUSES NOWDT
9: #FUSES WDT128
10: #FUSES PLL1
11: #FUSES CPUDIV1
12: #FUSES NOUSBDIV
13: #FUSES HS
14: #FUSES BROWNOUT
15: #FUSES NOLVP
16: #FUSES NOXINST
17: #use delay(clock=20000000)
18: #use rs232(baud=9600,parity=N,xmit=PIN_C6,
19:           rcv=PIN_C7,bits=8 ,stop=2,stream=PORT1,enable=PIN_E2)
20: #use rs232(baud=9600,parity=N,xmit=PIN_E0,
21:           rcv=PIN_B2,bits=8 ,stop=1, STREAM=PORT2,FORCE_SW)
22: #use rtos(timer = 0, minor_cycle =5ms)
23: #use standard_io (a)
24: #use standard_io (b)
25: #use standard_io (c)
26: #use standard_io (d)
27: #use standard_io (e)
28: #define RTC_SDA PIN_B0
29: #define RTC_SCL PIN_B1
30: #include <lcd_16x4.c>
31: #include <teclado_adc.c>
32: #include <_ds1307.c>
33: #include <string.h>
34: #include <stdlib.h>
35: #byte UCON=0xF6D
36: #bit USBEN=UCON.3
37: #byte UCFG = 0xF6F
38: #bit UTRDIS = UCFG.3
39: #include <manejo_leds.c>
40: #include <manejo_pulsadores_reles.c>
41: ///////////////////////////////////////////////////////////////////
42: #define minima_placa_1_e 0 // Location in EEPROM
43: #define maxima_placa_1_e 1 // Location in EEPROM
44: #define minima_placa_2_e 2 // Location in EEPROM
45: #define maxima_placa_2_e 3 // Location in EEPROM
46: ///////////////////////////////////////////////////////////////////
47: byte segundos;
48: byte minutos;
49: byte hora;
50: byte dia;
51: byte mes;
52: byte anio;
53: byte dow1;
54:
55: int16 valor_humedad_uno[3],valor_humedad_dos[3],valor_humedad_tres[3],
56:                               valor_humedad_cuatro[3];
57: int16 minima[3], maxima[3];
58: int16 minima_anterior[3], maxima_anterior[3];
59: int16 manual[3];
60: int16 manual_anterior[3];
61: int16 hora_inicio[3],hora_fin[3],minutos_inicio[3],minutos_fin[3];
62: int16 valor_uno, valor_dos;
63: int j=0,numero=0,s;
64: int i_rx=0;
65: int dato_completo=0,dat_comp,fin_pal;
66: int tx_placa=1,tx_placa_anterior=1;
67:
```

1

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
68: char dato_tx[30];
69: char dato_rx[40];
70: char dato_aux[5],dato_aux2[5];
71: char cadena[6];
72:
73: const int STATE_INI      = 0;
74: const int STATE_SEND    = 1;
75: const int STATE_ACTIVO = 2;
76:
77: const int          STATE_00      =0;
78: const int          STATE_01      =1;
79: const int          STATE_02      =2;
80: const int          STATE_03      =3;
81: const int          STATE_04      =4;
82: const int          STATE_01_A   =5;
83: const int          STATE_02_BCD_0=6;
84: const int          STATE_02_BCD_1=7;
85: const int          STATE_02_B1   =8;
86: const int          STATE_02_B2   =9;
87: const int          STATE_02_C0   =10;
88: const int          STATE_03_G   =11;
89: const int          STATE_03_G0  =12;
90: const int          STATE_03_H   =13;
91: const int          STATE_03_H0  =14;
92: const int          STATE_03_H1  =15;
93: const int          STATE_04_A   =16;
94: const int          STATE_04_B   =17;
95: const int          STATE_04_C   =18;
96: const int          STATE_04_D   =19;
97:
98: char dato,dato1[3],dato2[3],dato3[3],dato4[3],dato5[3];
99: byte hora_riego_inicio=0,minutos_riego_inicio=0;
100: byte hora_riego_fin=0,minutos_riego_fin=0;
101: int16 comando_tx=1, comando_tx_anterior=0;
102:
103: int bandera_modifica=0;
104: int estado_siguiente=0;
105: int estado_siguiente_maquina_lcd=0;
106: int resultado=0;
107: int num_veces=0, max_nodo=2;
108: int ind1=0,ind2=0,ind3=0,ind4=0,ind5=0;
109: int sem1;
110: int max_valvula = 2;
111: int resultado1=0, resultado2=0;
112: int placa=1;
113: int salida_1[3],salida_2[3],salida_3[3],salida_4[3];
114: int sube_1[3],sube_2[3],sube_3[3],sube_4[3];
115: int baja_1[3],baja_2[3],baja_3[3],baja_4[3];
116: int x;
117: int comando_rx=1;
118: int estado_valvula_1_placa_1=0;
119: int estado_valvula_2_placa_1=0;
120: int estado_valvula_1_placa_2=0;
121: int estado_valvula_2_placa_2=0;
122: int estado_valvula_1_placa_1_anterior=0;
123: int estado_valvula_2_placa_1_anterior=0;
124: int estado_valvula_1_placa_2_anterior=0;
125: int estado_valvula_2_placa_2_anterior=0;
126: int tipo_riege_placa_1=0;
127: int tipo_riege_anterior_placa_1=0;
128: int tipo_riege_placa_2=0;
129: int tipo_riege_anterior_placa_2=0;
130:
131: char string[5];
132: int estado_siguiente_maquina_reportes=0;
133: const int ST_INI           =0;
134: const int ST_ACTIVO        =1;

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
135: const int ST_APERTURA_VALVULA_MANUAL =2;
136: const int ST_APERTURA_VALVULA_HUMEDAD =3;
137: const int ST_REGISTRO_VALORES_HUMEDAD =4;
138: const int ST_LIMITES_HUMEDAD =5;
139: const int ST_APERTURA_VALVULA TIEMPO =6;
140:
141: int veces_registro=0;
142: int humedad_valvula_une_placa_une=0;
143: int humedad_valvula_dos_placa_une=0;
144: int humedad_valvula_une_placa_dos=0;
145: int humedad_valvula_dos_placa_dos=0;
146: int estado_siguiente_maquina_blue = 0;
147: const int ST_INI_BLUE =0;
148: const int ST_BLUE_ACTIVO =1;
149: const int ST_BLUE_TX =2;
150: const int ST_BLUE_RX =3;
151: char caracter_rx_blue;
152: char caracter_anterior_rx_blue;
153: int tx_blue,rx_blue;
154: char c;
155: int leo_dato=0;
156: int veces_ver_humedad=0;
157: int espera_placa_une=0;
158: int espera_placa_dos=0;
159: int max_espera=200;
160: int inicializacion_1=0;
161: int inicializacion_2=0;
162:
163: void guardar_MEM( byte dato, long int direccion)
164: {
165:     short int estado;
166:     i2c_start();
167:     i2c_write(0xA0);
168:
169:     i2c_write(direccion>>8);
170:
171:     i2c_write(direccion);
172:
173:     i2c_write(dato);
174:     i2c_stop();
175:     i2c_start();
176:     estado=i2c_write(0xa0);
177:
178:     while (estado==1)
179:     {
180:         i2c_start();
181:         estado=i2c_write(0xa0);
182:     }
183: }
184: BYTE leer_MEM(long int direccion)
185: {
186:     byte dato;
187:     i2c_start();
188:     i2c_write(0xA0);
189:
190:     i2c_write(direccion>>8);
191:
192:     i2c_write(direccion);
193:
194:     i2c_start();
195:     i2c_write(0x1);
196:
197:     dato=i2c_read(0);
198:     i2c_stop();
199:     return(dato);
200: }
201:
```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
202: #int_ext2
203: void int_RB2()
204: {
205:     if(kbhit(PORT2))
206:     {
207:         c=toupper(getc(PORT2));
208:         if(c=='L')
209:         {
210:             tx_blue=1;
211:             caracter_rx_blue=c;
212:         }
213:         if((c=='E')||(c=='S')||(c=='N'))
214:         {
215:             rx_blue=1;
216:             caracter_rx_blue=c;
217:         }
218:     }
219: }
220:
221: void interruptrb2_init()
222: {
223:     enable_interrupts(int_ext2);
224:     ext_int_edge(2, H_TO_L );
225: }
226: #int_rda
227: STATE_STORE()
228: {
229:     dato_rx[i_rx]=getc();
230:     if (dato_rx[i_rx]=='&')
231:     {
232:         dato_completo=1;
233:         i_rx=0;
234:     }
235:     else
236:     {
237:         dato_completo=0;
238:         i_rx++;
239:     }
240: }
241: void envio_dato()
242: {
243:     int i_tx,fin(dto=60;
244:     for(i_tx=0;i_tx<=fin(dto;i_tx++)
245:     {
246:         putc(dato_tx[i_tx]);
247:         if (dato_tx[i_tx]=='&') fin(dto=i_tx;
248:     }
249: }
250:
251: void armar_trama()
252: {
253:     for(i=0;i<=59;i++)
254:     dato_tx[0]='0';
255:     if (tx_placa==1) strcpy(dato_tx,"00000001");
256:     if (tx_placa==2) strcpy(dato_tx,"00000010");
257:     if ((comando_tx==3)|| (comando_tx==8))
258:     {
259:         sprintf(cadena,"%lu",comando_tx);
260:         strcat(dato_tx,cadena);
261:         sprintf(cadena,"+");
262:         strcat(dato_tx,cadena);
263:         sprintf(cadena,"%u",hora);
264:         strcat(dato_tx,cadena);
265:         sprintf(cadena,"+");
266:         strcat(dato_tx,cadena);
267:         sprintf(cadena,"%u",minutos);
268:         strcat(dato_tx,cadena);

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
269:     sprintf(cadena,"%&"); 
270:     strcat(dato_tx,cadena); 
271: } 
272: if (comando_tx==2) { 
274:     sprintf(cadena,"%lu",comando_tx); 
275:     strcat(dato_tx,cadena); 
276:     sprintf(cadena,"+"); 
277:     strcat(dato_tx,cadena); 
278:     sprintf(cadena,"%u",hora); 
279:     strcat(dato_tx,cadena); 
280:     sprintf(cadena,"+"); 
281:     strcat(dato_tx,cadena); 
282:     sprintf(cadena,"%u",minutos); 
283:     strcat(dato_tx,cadena); 
284:     sprintf(cadena,"+"); 
285:     strcat(dato_tx,cadena); 
286:     sprintf(cadena,"%lu",hora_inicio[tx_placa]); 
287:     strcat(dato_tx,cadena); 
288:     sprintf(cadena,"+"); 
289:     strcat(dato_tx,cadena); 
290:     sprintf(cadena,"%lu",minutos_inicio[tx_placa]); 
291:     strcat(dato_tx,cadena); 
292:     sprintf(cadena,"+"); 
293:     strcat(dato_tx,cadena); 
294:     sprintf(cadena,"%lu",hora_fin[tx_placa]); 
295:     strcat(dato_tx,cadena); 
296:     sprintf(cadena,"+"); 
297:     strcat(dato_tx,cadena); 
298:     sprintf(cadena,"%lu",minutos_fin[tx_placa]); 
299:     strcat(dato_tx,cadena); 
300:     sprintf(cadena,"%&"); 
301:     strcat(dato_tx,cadena); 
302: } 
303: if (comando_tx==4) { 
304:     sprintf(cadena,"%lu",comando_tx); 
305:     strcat(dato_tx,cadena); 
307:     sprintf(cadena,"+"); 
308:     strcat(dato_tx,cadena); 
309:     sprintf(cadena,"%u",hora); 
310:     strcat(dato_tx,cadena); 
311:     sprintf(cadena,"+"); 
312:     strcat(dato_tx,cadena); 
313:     sprintf(cadena,"%u",minutos); 
314:     strcat(dato_tx,cadena); 
315:     sprintf(cadena,"+"); 
316:     strcat(dato_tx,cadena); 
317:     sprintf(cadena,"%lu",manual[tx_placa]); 
318:     strcat(dato_tx,cadena); 
319:     sprintf(cadena,"%&"); 
320:     strcat(dato_tx,cadena); 
321: } 
322: if (comando_tx==1) { 
323:     sprintf(cadena,"%lu",comando_tx); 
324:     strcat(dato_tx,cadena); 
326:     sprintf(cadena,"+"); 
327:     strcat(dato_tx,cadena); 
328:     sprintf(cadena,"%u",hora); 
329:     strcat(dato_tx,cadena); 
330:     sprintf(cadena,"+"); 
331:     strcat(dato_tx,cadena); 
332:     sprintf(cadena,"%u",minutos); 
333:     strcat(dato_tx,cadena); 
334:     sprintf(cadena,"+"); 
335:     strcat(dato_tx,cadena);

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
336:     sprintf(cadena,"%lu",minima[tx_placa]);
337:     strcat(dato_tx,cadena);
338:     sprintf(cadena,"+");
339:     strcat(dato_tx,cadena);
340:     sprintf(cadena,"%lu",maxima[tx_placa]);
341:     strcat(dato_tx,cadena);
342:     sprintf(cadena,"&");
343:     strcat(dato_tx,cadena);
344: }
345: }
346:
347: void deco_tramaRx()
348: {
349:     numero=0;
350:     fin_pal=50;dat_comp=0;
351:     for(i=8;i<=fin_pal;i++)
352:     {
353:         if((dato_rx[i]=='&') && (dat_comp==1))
354:         {
355:             dat_comp=1;
356:             fin_pal=i;
357:         }
358:         if ((dato_rx[i]!='+') && (dato_rx[i]!='&'))
359:         {
360:             dato_aux[j]=dato_rx[i];
361:             j=j+1;dat_comp=0;
362:         }else{
363:             for(s=0;s<=3;s++) dato_aux2[s]='0';
364:             for(s=0;s<j;s++) dato_aux2[(4-j)+s]=dato_aux[s];
365:             j=0;
366:             if ((numero==0)&&(dat_comp!=1))
367:             {
368:                 comando_rx=(atoi32(dato_aux2));
369:                 numero=1; dat_comp=1;
370:             }
371:             if ((numero==1)&&(dat_comp!=1))
372:             {
373:                 if (comando_rx==1) valor_humedad_uno[tx_placa]=
374:                     (atoi32(dato_aux2));
375:                 if ((comando_rx==2)&&(tx_placa==1))
376:                 {
377:                     estado_valvula_1_placa_1_anterior=estado_valvula_1_placa_1;
378:                     estado_valvula_1_placa_1=(atoi32(dato_aux2));
379:                 }
380:                 if ((comando_rx==2)&&(tx_placa==2))
381:                 {
382:                     estado_valvula_1_placa_2_anterior=estado_valvula_1_placa_2;
383:                     estado_valvula_1_placa_2=(atoi32(dato_aux2));
384:                 }
385:                 numero=2; dat_comp=1;
386:             }
387:             if ((numero==2)&&(dat_comp!=1))
388:             {
389:                 if (comando_rx==1) valor_humedad_dos[tx_placa]=
390:                     (atoi32(dato_aux2));
391:                 if ((comando_rx==2)&&(tx_placa==1))
392:                 {
393:                     estado_valvula_2_placa_1_anterior=estado_valvula_2_placa_1;
394:                     estado_valvula_2_placa_1=(atoi32(dato_aux2));
395:                 }
396:                 if ((comando_rx==2)&&(tx_placa==2))
397:                 {
398:                     estado_valvula_2_placa_2_anterior=estado_valvula_2_placa_2;
399:                     estado_valvula_2_placa_2=(atoi32(dato_aux2));
400:                 }
401:                 numero=3; dat_comp=1;
402:             }
}

```

```

C:\código_estacion concentradora\código_estacion_concentrador.c
403:         if ((numero==3)&&(dat_comp!=1))
404:         {
405:             if (comando_rx==1)valor_humedad_tres[tx_placa]=
406:                                         (atoi32(dato_aux2));
407:             if ((comando_rx==2)&&(tx_placa==1))
408:             {
409:                 tipo_riego_anterior_placa_1=tipo_riego_placa_1;
410:                 tipo_riego_placa_1=(atoi32(dato_aux2));
411:             }
412:             if ((comando_rx==2)&&(tx_placa==2))
413:             {
414:                 tipo_riego_anterior_placa_2=tipo_riego_placa_2;
415:                 tipo_riego_placa_2=(atoi32(dato_aux2));
416:             }
417:             numero=4; dat_comp=1;
418:         }
419:         if ((numero==4)&&(dat_comp!=1))
420:         {
421:             if (comando_rx==1)valor_humedad_cuatro[tx_placa]=
422:                                         (atoi32(dato_aux2));
423:             if (comando_rx==2);
424:             numero=0;
425:             dat_comp=1;
426:             fin_pal=i;
427:         }
428:     }
429: }
430: }
431:
432: #task(rate = 35ms)
433: void maquina_de_estado_de_comunicacion();
434: #task(rate = 50ms),
435: void reloj_tiempo_real();
436: #task(rate = 5ms)
437: void maquina_de_estado_pantallas();
438: #task(rate = 50ms)
439: void manejo_bombas();
440: #task(rate = 5ms)
441: void maquina_de_estado_teclado_analogico();
442: #task(rate = 55ms)
443: void maquina_de_estado_reportes();
444: #task(rate = 255ms)
445: void maquina_de_estado_bluetooth();
446: void main()
447: {
448:     delay_ms(150);
449:     UTRDIS = 1;
450:     desactivar_rele(PIN_A5);
451:     desactivar_rele(PIN_A2);
452:     setup_timer_3(T3_DISABLED | T3_DIV_BY_1);
453:     lcd_init();
454:     setup_adc_ports(NO_ANALOGS);
455:     setup_adc(ADC_OFF);
456:     setup_psp(PSP_DISABLED);
457:     setup_spi(FALSE);
458:     SETUP_SPI(SPI_DISABLED);
459:     SETUP_SPI(SPI_SS_DISABLED);
460:     enable_interrupts(global);
461:     enable_interrupts(int_rda);
462:     interruptrb2_init();
463:     setup_counters(RTCC_INTERNAL,RTCC_DIV_32);
464:     ds1307_init(DS1307_OUT_ON_DISABLED_HIHG | DS1307_OUT_ENABLED |
465:                                         DS1307_OUT_32_KHZ);
466:     delay_us(10);
467:     rtos_run();
468: }
469:

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
470: void manejo_bombas()
471: {
472:     x=estado_valvula_1_placa_1+estado_valvula_2_placa_1+
473:                     estado_valvula_1_placa_2+estado_valvula_2_placa_2;
474:     if ((x<=2)&&(x!=0))
475:     {
476:         activar_rele(PIN_A2);
477:         desactivar_rele(PIN_A5);
478:     }
479:     if (x>2)
480:     {
481:         activar_rele(PIN_A2);
482:         activar_rele(PIN_A5);
483:     }
484:     if (x==0)
485:     {
486:         desactivar_rele(PIN_A5);
487:         desactivar_rele(PIN_A2);
488:     }
489: }
490:
491: void maquina_de_estado_de_comunicacion()
492: {
493:     switch (estado_siguiente){
494:         case STATE_INI:
495:         {
496:             minima[1]=read_EEPROM (minima_placa_1_e);
497:             maxima[1]=read_EEPROM (maxima_placa_1_e);
498:             minima[2]=read_EEPROM (minima_placa_2_e);
499:             maxima[2]=read_EEPROM (maxima_placa_2_e);
500:             for(i=0;i<=2;i++)
501:             {
502:                 salida_1[i]=0;
503:                 salida_2[i]=0;
504:                 salida_3[i]=0;
505:                 salida_4[i]=0;
506:                 sube_1[i]=0;
507:                 sube_2[i]=0;
508:                 sube_3[i]=0;
509:                 sube_4[i]=0;
510:                 baja_1[i]=0;
511:                 baja_2[i]=0;
512:                 baja_3[i]=0;
513:                 baja_4[i]=0;
514:             }
515:             veces_registro=0;
516:             hora_inicio[1]      =0;
517:             hora_inicio[2]      =0;
518:             minutos_inicio[1]   =0;
519:             minutos_inicio[2]   =0;
520:             hora_fin[1]        =0;
521:             hora_fin[2]        =0;
522:             minutos_fin[1]     =0;
523:             minutos_fin[2]     =0;
524:             manual_anterior[1]=44;
525:             manual_anterior[2]=44;
526:             manual[1]=44;
527:             manual[2]=44;
528:             valor_humedad_uno[1]=1;
529:             valor_humedad_uno[2]=1;
530:             valor_humedad_dos[1]=1;
531:             valor_humedad_dos[2]=1;
532:             valor_humedad_tres[1]=1;
533:             valor_humedad_tres[2]=1;
534:             valor_humedad_cuatro[1]=1;
535:             valor_humedad_cuatro[2]=1;
536:             comando_tx=1;
}

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
537:
538:     tx_placa=2;
539:     dato_completo=0;
540:     armar_trama();
541:     envio_dato();
542:     while(dato_completo==0)
543:     {
544:         if (espera_placa_dos!=max_espera)   espera_placa_dos=
545:                                         (espera_placa_dos+1);
546:         if (espera_placa_dos==max_espera)  dato_completo=1;
547:         delay_ms(10);
548:     }
549:
550:     comando_tx=1;
551:     tx_placa=1;
552:     dato_completo=0;
553:     armar_trama();
554:     envio_dato();
555:     while(dato_completo==0)
556:     {
557:         if (espera_placa_uno!=max_espera)   espera_placa_uno=
558:                                         (espera_placa_uno+1);
559:         if (espera_placa_uno==max_espera)  dato_completo=1;
560:         delay_ms(10);
561:     }
562:     estado_siguiente=STATE_ACTIVO;
563:     break;
564: }
565: case STATE_SEND:
566: {
567:     if(( tx_placa==1)) tx_placa=2; else tx_placa=1;
568:     if ((comando_tx==3)&& (comando_rx==1)) comando_tx=8;
569:     armar_trama();
570:     envio_dato();
571:     if ((comando_tx_anterior==comando_tx)&&
572:             (tx_placa_anterior!=tx_placa)) comando_tx=8;
573:     comando_tx_anterior=comando_tx;
574:     tx_placa_anterior=tx_placa;
575:     estado_siguiente=STATE_ACTIVO;
576:     break;
577: }
578: case STATE_ACTIVO:
579: {
580:     if ( ((dato_rx[0]=='0') && (dato_rx[1]=='0') && (dato_rx[2]=='0') &&
581:             (dato_rx[3]=='0') && (dato_rx[4]=='0') && (dato_rx[5]=='0') &&
582:             (dato_rx[6]=='0') && (dato_rx[7]=='1')) && ( dato_completo==1)) ||
583:             (((dato_rx[0]=='0') && (dato_rx[1]=='0') && (dato_rx[2]=='0') &&
584:                 (dato_rx[3]=='0') && (dato_rx[4]=='0') && (dato_rx[5]=='0') &&
585:                 (dato_rx[6]=='1')) && (dato_rx[7]=='0')) && ( dato_completo==1)) )
586:     {
587:         deco_tramaRx();
588:         dato_completo=0;
589:         if ((tx_placa==1)&& (espera_placa_uno!=max_espera)) encender_led(1);
590:         if ((tx_placa==2)&& (espera_placa_dos!=max_espera)) encender_led(2);
591:         estado_siguiente=STATE_SEND;
592:     }
593:     if ((espera_placa_uno==max_espera)&&(tx_placa==1))
594:     {
595:         estado_siguiente=STATE_SEND;
596:         dato_completo=0;encender_led(0);
597:     }
598:     if((espera_placa_dos==max_espera)&&(tx_placa==2))
599:     {
600:         estado_siguiente=STATE_SEND;
601:         dato_completo=0;encender_led(0);
602:     }
603: }

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
604:     }
605:     break;
606:   }
607: }
608: }
609:
610: void maquina_de_estado_pantallas()
611: {
612:   switch (estado_siguiente_maquina_lcd) {
613:     case STATE_00:
614:       lcd_putc('\f');
615:       lcd_gotoxy(1,1);
616:       lcd_putc("BIENBENIDOS");
617:       lcd_gotoxy(1,2);
618:       lcd_putc("SIST. DE RIEGO");
619:       lcd_gotoxy(1,3);
620:       lcd_putc("Version 3.0");
621:       lcd_gotoxy(1,4);
622:       lcd_putc("UNCA");
623:       delay_ms(1050);
624:       estado_siguiente_maquina_lcd=STATE_01;
625:       num_veces=0;
626:       tecla='Z';
627:       comando_tx=8;
628:     break;
629:     case STATE_01:
630:       if (num_veces==0)
631:       {
632:         lcd_putc('\f');
633:         lcd_gotoxy(1,1);
634:         lcd_putc(" *FECHA Y HORA* ");
635:         lcd_gotoxy(1,4);
636:         lcd_putc("Si[F1],No[F2]");
637:         ind1=0;
638:         ind2=0;
639:         ind3=0;
640:         ind4=0;
641:         ind5=0;
642:         num_veces=1;  tecla='Z';
643:         for(i=0;i<=2;i++) dato1[i]='0';
644:         for(i=0;i<=2;i++) dato2[i]='0';
645:         for(i=0;i<=2;i++) dato3[i]='0';
646:         for(i=0;i<=2;i++) dato4[i]='0';
647:         for(i=0;i<=2;i++) dato5[i]='0';
648:         bandera_modifica=0;
649:         num_veces=1;  tecla='Z';
650:       }
651:       if (bandera_modifica==0)
652:       {
653:         lcd_gotoxy(1,2);
654:         lcd_putc("          ");
655:         lcd_gotoxy(1,2);
656:         if (dia<10) printf(lcd_putc,"Fecha:0%u",dia);
657:         else printf(lcd_putc,"Fecha:%u",dia);
658:         lcd_gotoxy(9,2);
659:         if (mes<10) printf(lcd_putc,"/0%u",mes);
660:         else printf(lcd_putc,"/%u",mes);
661:         lcd_gotoxy(12,2);
662:         if (anio<10) printf(lcd_putc,"/0%u",anio);
663:         else printf(lcd_putc,"/%u",anio);
664:         lcd_gotoxy(1,3);
665:         lcd_putc("          ");
666:         lcd_gotoxy(1,3);
667:         if (hora<10) printf(lcd_putc,"Hora:0%u:",hora);
668:         else printf(lcd_putc,"Hora:%u:",hora);
669:         lcd_gotoxy(9,3);
670:         if (minutos<10) printf(lcd_putc,"0%u",minutos);

```

10

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
671:         else printf(lcd_putc, "%u", minutos);
672:     }
673:     if (tecla=='B')
674:     {
675:         bandera_modifica=1;
676:         lcd_gotoxy(1,2);
677:         lcd_putc("                ");
678:         lcd_gotoxy(1,2);
679:         lcd_putc("Fecha:??");
680:         lcd_putc("/??");
681:         lcd_putc("/??");
682:         lcd_gotoxy(1,3);
683:         lcd_putc("                ");
684:         lcd_gotoxy(1,3);
685:         lcd_putc("Hora:?:?:??");
686:         ind1=0;
687:         ind2=0;
688:         ind3=0;
689:         ind4=0;
690:         ind5=0;
691:         tecla='Z';
692:     }
693:     if (bandera_modifica==1)
694:     {
695:         ////////////////////////////////FECHA///////////////////
696:         if ((tecla!='Z')&&(ind3<=1)&&(tecla!='A')&&
697:             (tecla!='B')&&(tecla!='C')&&(tecla!='D')&&(tecla!='E')&&(tecla!='F'))
698:         {
699:             dato3[ind3]=tecla;
700:             lcd_gotoxy(7+ind3,2);
701:             lcd_putc(tecla);
702:             ind3++;
703:             tecla='Z';
704:             if (ind3==1)  dato3[2]=0;
705:         }
706:         if ((tecla!='Z')&&(ind4<=1)&&(tecla!='A')&&
707:             (tecla!='B')&&(tecla!='C')&&(tecla!='D')&&(tecla!='E')&&(tecla!='F'))
708:         {
709:             dato4[ind4]=tecla;
710:             lcd_gotoxy(10+ind4,2);
711:             lcd_putc(tecla);
712:             ind4++;
713:             tecla='Z';
714:             if (ind4==1)  dato4[2]=0;
715:         }
716:         if ((tecla!='Z')&&(ind5<=1)&&(tecla!='A')&&(tecla!='B')&&(tecla!='C')
717:             &&(tecla!='D')&&(tecla!='E')&&(tecla!='F'))
718:         {
719:             dato5[ind5]=tecla;
720:             lcd_gotoxy(13+ind5,2);
721:             lcd_putc(tecla);
722:             ind5++;
723:             tecla='Z';
724:             if (ind5==1)  dato5[2]=0;
725:         }
726:         ////////////////////////////////HORA/////////////////
727:         if ((tecla!='Z')&&(ind1<=1)&&(tecla!='A')&&(tecla!='B')&&(tecla!='C')
728:             &&(tecla!='D')&&(tecla!='E')&&(tecla!='F'))
729:         {
730:             dato1[ind1]=tecla;
731:             lcd_gotoxy(6+ind1,3);
732:             lcd_putc(tecla);
733:             ind1++;
734:             tecla='Z';
735:             if (ind1==1)  dato1[2]=0;
736:         }
737:         if ((tecla!='Z')&&(ind2<=1)&&(tecla!='A')&&(tecla!='B')&&(tecla!='C')

```

11

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
738:         && (tecla!='D')&&(tecla!='E')&&(tecla!='F'))
739:     {
740:         dato2[ind2]=tecla;
741:         lcd_gotoxy(9+ind2,3);
742:         lcd_putc(tecla);
743:         ind2++;
744:         tecla='Z';
745:         if (ind2==1)  dato2[2]=0;
746:     }
747: }
748: estado_siguiente_maquina_lcd=STATE_01;
749: ///////////////////////////////////////////////////////////////////
750: if ((tecla=='A')&&(bandera_modifica==1))
751: {
752:     hora =atoi(dato1);
753:     minutos =atoi(dato2);
754:     dia = atoi(dato3);
755:     mes = atoi(dato4);
756:     anio = atoi(dato5);
757:     dow1 = 2;
758:     segundos = 0;
759:     if ((hora>=0)&&(hora<=23)&&(minutos>=0)&&(minutos<=60)&&
760:         (dia>=1)&&(dia<=31)&&(mes>=1)&&(mes<=12)&&(anio>=0)&&(anio<=99))
761:     {
762:         bandera_modifica=0;
763:         delay_ms(15);
764:         ds1307_set_date_time(dia,mes,anio,dow1,hora,minutos,segundos);
765:         delay_ms(15);
766:         estado_siguiente_maquina_lcd=STATE_02;
767:         num_veces=0;tecla='Z';
768:     } else
769:     {
770:         lcd_gotoxy(1,3);
771:         lcd_putc("          ");
772:         lcd_gotoxy(1,3);
773:         lcd_putc("error");
774:         delay_ms(500);
775:         lcd_gotoxy(1,3);
776:         lcd_putc("Hora:???:??");
777:         lcd_gotoxy(1,2);
778:         lcd_putc("          ");
779:         lcd_gotoxy(1,2);
780:         lcd_putc("error");
781:         delay_ms(500);
782:         lcd_gotoxy(1,2);
783:         lcd_putc("Fecha:??");
784:         lcd_putc("/??");
785:         lcd_putc("//??");
786:         ind1=0;
787:         ind2=0;
788:         ind3=0;
789:         ind4=0;
790:         ind5=0;
791:         tecla='Z';
792:         bandera_modifica=1;
793:     }
794: }
795: if ((tecla=='A')&&(bandera_modifica==0))
796: {
797:     estado_siguiente_maquina_lcd=STATE_02;
798:     tecla='Z'; num_veces=0;
799: }
800: break;
801: case STATE_02:
802:     if (num_veces==0)
803:     {
804:         lcd_putc('\f');

```

12

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
805:         lcd_gotoxy(1,1);
806:         lcd_putc(" **MENU**  1/3");
807:         lcd_gotoxy(1,2);
808:         lcd_putc("F1:Configura");
809:         lcd_gotoxy(1,3);
810:         lcd_putc("F2:Fecha y Hora");
811:         lcd_gotoxy(1,4);
812:         lcd_putc("F3:Mas");
813:         num_veces=1; tecla='Z';
814:     }
815:     if (tecla=='A')
816:     {
817:         estado_siguiente_maquina_lcd=STATE_02_BCD_0;
818:         tecla='Z';
819:         num_veces=0;
820:     }
821:     if (tecla=='B')
822:     {
823:         estado_siguiente_maquina_lcd=STATE_01;
824:         tecla='Z';
825:         num_veces=0;
826:     }
827:     if (tecla=='C')
828:     {
829:         estado_siguiente_maquina_lcd=STATE_03;
830:         tecla='Z';
831:         num_veces=0;
832:     }
833:     break;
834: case STATE_03:
835:     if (num_veces==0)
836:     {
837:         lcd_putc('\f');
838:         lcd_gotoxy(1,1);
839:         lcd_putc(" **MENU**  2/3");
840:         lcd_gotoxy(1,2);
841:         lcd_putc("F1:Ver Estado");
842:         lcd_gotoxy(1,3);
843:         lcd_putc("F2:Manual ");
844:         lcd_gotoxy(1,4);
845:         lcd_putc("F3:Mas");
846:         num_veces=1; tecla='Z';
847:     }
848:     estado_siguiente_maquina_lcd=STATE_03;
849:     if (tecla=='A')
850:     {
851:         estado_siguiente_maquina_lcd=STATE_03_G;
852:         num_veces=0;
853:         tecla='Z';
854:     }
855:     if (tecla=='B')
856:     {
857:         estado_siguiente_maquina_lcd=STATE_03_H;
858:         num_veces=0;
859:         tecla='Z';
860:         manual[1]=45;manual[2]=45;
861:         comando_tx=4;
862:     }
863:     if (tecla=='C')
864:     {
865:         estado_siguiente_maquina_lcd=STATE_04;
866:         num_veces=0;
867:         tecla='Z';
868:     }
869:     break;
870: case STATE_04:
871:     if (num_veces==0)

```

13

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
872:    {
873:        lcd_putc('\f');
874:        lcd_gotoxy(1,1);
875:        lcd_putc(" **MENU**  3/3");
876:        lcd_gotoxy(1,2);
877:        lcd_putc("F1:Borrar");
878:        lcd_gotoxy(1,4);
879:        lcd_putc("F3:Mas");
880:        num_veces=1; tecla='Z';
881:    }
882:    estado_siguiente_maquina_lcd=STATE_04;
883:    if (tecla=='A')
884:    {
885:        estado_siguiente_maquina_lcd= STATE_04_A;
886:        tecla='Z';
887:        num_veces=0;
888:    }
889:    if (tecla=='C')
890:    {
891:        estado_siguiente_maquina_lcd=STATE_02;
892:        tecla='Z';
893:        num_veces=0;
894:    }
895:    break;
896: case STATE_02_BCD_0:
897:     if (num_veces==0)
898:     {
899:         lcd_putc('\f');
900:         lcd_gotoxy(1,1);
901:         lcd_putc(" *CONFIGURA* 1/3");
902:         lcd_gotoxy(1,2);
903:         lcd_putc("Sector:?:");
904:         lcd_gotoxy(1,3);
905:         lcd_putc("Sí:[F1]");
906:         lcd_gotoxy(1,4);
907:         lcd_putc("No:[F2]");
908:         ind1=0;
909:         num_veces=1;
910:         tecla='Z';
911:         valor_uno=0;
912:         for(i=0;i<=2;i++) dato1[i]='0';
913:     }
914:     if ((tecla!='Z')&&(ind1==0)&&(tecla!='A')&&(tecla!='B')&&
915:         (tecla!='C')&&(tecla!='D')&&(tecla!='E')&&(tecla!='F'))
916:     {
917:         dato1[ind1]=tecla;
918:         lcd_gotoxy(8,2);
919:         lcd_putc(tecla);
920:         dato1[1]=0;
921:         ind1=1;
922:         tecla='Z';
923:     }
924:     estado_siguiente_maquina_lcd=STATE_02_BCD_0;
925:     if (tecla=='A')
926:     {
927:         valor_uno=atoi(dato1);
928:         if ((valor_uno>0)&&(valor_uno<=max_nodo)&&
929:             (((espera_placa_uno!=max_espera)&&(valor_uno!=2)) ||
930:              ((espera_placa_dos!=max_espera)&&valor_uno!=1)))
931:         {
932:             placa=valor_uno;
933:             estado_siguiente_maquina_lcd=STATE_02_BCD_1;
934:             num_veces=0; ind1=0;
935:         } else
936:         {
937:             lcd_gotoxy(1,2);

```

14

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
939:         lcd_putc("          ");
940:         lcd_gotoxy(1,2);
941:         lcd_putc("error");
942:         delay_ms(500);
943:         lcd_gotoxy(1,2);
944:         lcd_putc("          ");
945:
946:         lcd_gotoxy(1,2);
947:         lcd_putc("Sector:?");
948:         ind1=0; resultado=5;
949:     }
950: }
951: if (tecla=='B')
952: {
953:     estado_siguiente_maquina_lcd=STATE_02;
954:     num_veces=0;
955:     tecla='Z';
956: }
957: tecla='Z';
958: break;
959: case STATE_02_BCD_1:
960:     if (num_veces==0)
961:     {
962:         lcd_putc('\f');
963:         lcd_gotoxy(1,1);
964:         lcd_putc('f');
965:         lcd_gotoxy(1,1);
966:         lcd_putc(" *CONFIGURA* 2/3");
967:         lcd_gotoxy(1,2);
968:         lcd_putc("F1:Lim. Humedad");
969:         lcd_gotoxy(1,4);
970:         lcd_putc("F2:Lim. Horario ");
971:         num_veces=1;
972:         tecla='Z';
973:     }
974:     estado_siguiente_maquina_lcd=STATE_02_BCD_1;
975:     if (tecla=='A')
976:     {
977:         estado_siguiente_maquina_lcd=STATE_02_B1;
978:         num_veces=0;
979:         tecla='Z';
980:     }
981:     if (tecla=='B')
982:     {
983:         estado_siguiente_maquina_lcd=STATE_02_C0;
984:         num_veces=0;
985:         tecla='Z';
986:     }
987: break;
988: case STATE_02_B1:
989:     if (num_veces==0)
990:     {
991:         lcd_putc('f');
992:         lcd_gotoxy(1,1);
993:         printf(lcd_putc,"SECTOR:%d",placa);
994:         lcd_gotoxy(1,2);
995:         printf(lcd_putc,"Min:%lu", minima[placa]);
996:         lcd_gotoxy(7,2);
997:         lcd_putc("%");
998:         lcd_gotoxy(10,2);
999:         printf(lcd_putc,"Max:%lu", maxima[placa]);
1000:        lcd_gotoxy(16,2);
1001:        lcd_putc("%");
1002:        lcd_gotoxy(1,3);
1003:        lcd_putc("Modifica:[F1]");
1004:        lcd_gotoxy(1,4);
1005:        lcd_putc("Continua:[F2]");

```

15

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
1006:     ind1=0;
1007:     ind2=0;
1008:     num_veces=1;
1009:     tecla='Z';
1010:     for(i=0;i<=2;i++) dato1[i]='0';
1011:     for(i=0;i<=2;i++) dato2[i]='0';
1012:     bandera_modifica=0;valor_uno=0;valor_dos=0;
1013: }
1014: if (tecla=='A')
1015: {
1016:     bandera_modifica=1;
1017:     lcd_gotoxy(1,2);
1018:     lcd_putc("      ");
1019:     lcd_gotoxy(1,2);
1020:     lcd_putc("Min:??");
1021:     lcd_gotoxy(10,2);
1022:     lcd_putc("      ");
1023:     lcd_gotoxy(10,2);
1024:     lcd_putc("Max:??");
1025:     ind1=0;
1026:     ind2=0;
1027:     tecla='Z';
1028: }
1029: if (bandera_modifica==1)
1030: {
1031:     if ((tecla!='Z')&&(ind1<=1)&&(tecla!='A')&&(tecla!='B')&&
1032:         (tecla!='C')&&(tecla!='D')&&(tecla!='E')&&(tecla!='F'))
1033:     {
1034:         dato1[ind1]=tecla;
1035:         lcd_gotoxy(5+ind1,2);
1036:         lcd_putc(tecla);
1037:         ind1++;
1038:         tecla='Z';
1039:         if (ind1==1) dato1[2]=0;
1040:     }
1041:     if ((tecla!='Z')&&(ind2<=1)&&(tecla!='A')&&(tecla!='B')&&
1042:         (tecla!='C')&&(tecla!='D')&&(tecla!='E')&&(tecla!='F'))
1043:     {
1044:         dato2[ind2]=tecla;
1045:         lcd_gotoxy(14+ind2,2);
1046:         lcd_putc(tecla);
1047:         ind2++;
1048:         tecla='Z';
1049:         if (ind2==1) dato2[2]=0;
1050:     }
1051: }
1052: estado_siguiente_maquina_lcd=STATE_02_B1;
1053: if (tecla=='B')
1054: {
1055:     valor_uno=atoi32(dato1);
1056:     valor_dos=atoi32(dato2);
1057:     if (((valor_uno>0)&&(valor_uno<=99)&&(valor_dos>=0)&&
1058:          (valor_uno<=99)&&(valor_uno<valor_dos))||(valor_uno==valor_dos))
1059:     {
1060:         estado_siguiente_maquina_lcd=STATE_02_B2;
1061:         num_veces=0;
1062:         tecla='Z';
1063:         if (bandera_modifica==1)
1064:         {
1065:             minima_anterior[placa]= minima[placa];
1066:             maxima_anterior[placa]= maxima[placa];
1067:             minima[placa]=valor_uno;
1068:             maxima[placa]=valor_dos;
1069:             comando_tx=1;//envio limites de humedad
1070:             write_eeprom(minima_placa_1_e, minima[1]);
1071:             write_eeprom(maxima_placa_1_e, maxima[1] );
1072:             write_eeprom(minima_placa_2_e, minima[2]);

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
1073:         write_eeprom(maxima_placa_2_e, maxima[2]);
1074:     }
1075:     if ((ind1==1)&&(ind2==1))bandera_modifica=0;
1076:   }
1077:   else
1078:   {
1079:     lcd_gotoxy(1,2);
1080:     lcd_putc("      ");
1081:     lcd_gotoxy(1,2);
1082:     lcd_putc("error");
1083:     delay_ms(500);
1084:     lcd_gotoxy(1,2);
1085:     lcd_putc("Min:??");
1086:     lcd_gotoxy(10,2);
1087:     lcd_putc("      ");
1088:     lcd_gotoxy(10,2);
1089:     lcd_putc("error");
1090:     delay_ms(500);
1091:     lcd_gotoxy(10,2);
1092:     lcd_putc("Max:??");
1093:     ind1=0;
1094:     ind2=0;
1095:     tecla='Z';
1096:     bandera_modifica=1;
1097:   }
1098: }
1099: tecla='Z';
1100: break;
1101: case STATE_02_B2:
1102:   if (num_veces==0)
1103:   {
1104:     lcd_putc('\f');
1105:     lcd_gotoxy(1,1);
1106:     lcd_putc("LIMITES HUMEDAD");
1107:     lcd_gotoxy(1,2);
1108:     printf(lcd_putc,"Minima:%lu",minima[placa]);
1109:     lcd_gotoxy(10,2);
1110:     lcd_putc('%');
1111:     lcd_gotoxy(1,3);
1112:     printf(lcd_putc,"Maxima:%lu",maxima[placa]);
1113:     lcd_gotoxy(10,3);
1114:     lcd_putc('%');
1115:     lcd_gotoxy(1,4);
1116:     lcd_putc("Continua:[F1]");
1117:     num_veces=1; tecla='Z';
1118:   }
1119: estado_siguiente_maquina_lcd=STATE_02_B2;
1120: if (tecla=='A')
1121: {
1122:   estado_siguiente_maquina_lcd=STATE_02;
1123:   tecla='Z';
1124:   num_veces=0;
1125: }
1126: break;
1127: case STATE_02_C0:
1128:   if (num_veces==0)
1129:   {
1130:     lcd_putc('\f');
1131:     lcd_gotoxy(1,1);
1132:     printf(lcd_putc,"Sector:%d",placa);
1133:     lcd_gotoxy(1,2);
1134:     printf(lcd_putc,"Inicio:%ld:%ld",hora_inicio[placa],
1135:           minutos_inicio[placa]);
1136:     lcd_gotoxy(1,3);
1137:     printf(lcd_putc,"Fin:%ld:%ld",hora_fin[placa],minutos_fin[placa]);
1138:     lcd_gotoxy(1,4);
1139:     lcd_putc("Si:[F1]");

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
1140:     lcd_gotoxy(9,4);
1141:     lcd_putc("No:[F2]");
1142:     ind1=0;
1143:     ind2=0;
1144:     ind3=0;
1145:     ind4=0;
1146:     num_veces=1;
1147:     tecla='Z';
1148:     for(i=0;i<=2;i++) dato1[i]='0';
1149:     for(i=0;i<=2;i++) dato2[i]='0';
1150:     for(i=0;i<=2;i++) dato3[i]='0';
1151:     for(i=0;i<=2;i++) dato4[i]='0';
1152:     bandera_modifica=0;
1153: }
1154: if (tecla=='B')
1155: {
1156:     bandera_modifica=1;
1157:     lcd_gotoxy(1,2);
1158:     lcd_putc("          ");
1159:     lcd_gotoxy(1,2);
1160:     lcd_putc("Inicio:????");
1161:     lcd_gotoxy(1,3);
1162:     lcd_putc("          ");
1163:     lcd_gotoxy(1,3);
1164:     lcd_putc("Fin:????");
1165:     ind1=0;
1166:     ind2=0;
1167:     ind3=0;
1168:     ind4=0;
1169:     tecla='Z';
1170: }
1171: if (bandera_modifica==1)
1172: {
1173: ////////////////////////////////////////////////////////////////////HORA INICIO///////////////
1174: if ((tecla!='Z')&&(ind1<=1)&&(tecla!='A')&&(tecla!='B')&&
1175:      (tecla!='C')&&(tecla!='D')&&(tecla!='E')&&(tecla!='F'))
1176: {
1177:     dato1[ind1]=tecla;
1178:     lcd_gotoxy(8+ind1,2);
1179:     lcd_putc(tecla);
1180:     ind1++;
1181:     tecla='Z';
1182:     if (ind1==1) dato1[2]=0;
1183: }
1184: if ((tecla=='Z')&&(ind2<=1)&&(tecla!='A')&&(tecla!='B')&&
1185:      (tecla!='C')&&(tecla!='D')&&(tecla!='E')&&(tecla!='F'))
1186: {
1187:     dato2[ind2]=tecla;
1188:     lcd_gotoxy(11+ind2,2);
1189:     lcd_putc(tecla);
1190:     ind2++;
1191:     tecla='Z';
1192:     if (ind2==1) dato2[2]=0;
1193: }
1194: if ((ind1==1)&&(ind2==1)&&(ind3==1)&&(ind4==1)&&
1195:      (ind5==1))bandera_modifica=0;
1196: estado_siguiente_maquina_lcd=STATE_01;
1197: ////////////////////////////////////////////////////////////////////HORA FIN///////////////
1198: if ((tecla=='Z')&&(ind3<=1)&&(tecla!='A')&&(tecla!='B')&&
1199:      (tecla!='C')&&(tecla!='D')&&(tecla!='E')&&(tecla!='F'))
1200: {
1201:     dato3[ind3]=tecla;
1202:     lcd_gotoxy(8+ind3,3);
1203:     lcd_putc(tecla);
1204:     ind3++;
1205:     tecla='Z';
1206: }

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
1207:         if (ind3==1)  dato3[2]=0;
1208:     }
1209:     if ((tecla!='Z')&&(ind4<=1)&&(tecla!='A')&&(tecla!='B')&&
1210:             (tecla!='C')&&(tecla!='D')&&(tecla!='E')&&(tecla!='F'))
1211:     {
1212:         dato4[ind4]=tecla;
1213:         lcd_gotoxy(11+ind4,3);
1214:         lcd_putc(tecla);
1215:         ind4++;
1216:         tecla='Z';
1217:         if (ind4==1)  dato4[2]=0;
1218:     }
1219: }
1220: estado_siguiente_maquina_lcd=STATE_02_C0;
1221: /////////////////////////////////
1222: estado_siguiente_maquina_lcd=STATE_02_C0;
1223: if (tecla=='A')
1224: {
1225:     if (bandera_modifica==1)
1226:     {
1227:         hora_riego_inicio =atoi(dato1);
1228:         minutos_riego_inicio=atoi(dato2);
1229:         hora_riego_fin =atoi(dato3);
1230:         minutos_riego_fin =atoi(dato4);
1231:     }
1232:     if ((ind1==1)&&(ind2==1)&&(ind3==1)&&(ind4==1))bandera_modifica=0;
1233:     if (((hora_riego_inicio>=0)&&(hora_riego_inicio<=23)&&
1234:             (minutos_riego_inicio>=0)&&(minutos_riego_inicio<=60))&&
1235:                 ((hora_riego_fin>=0)&&(hora_riego_fin<=23)&&
1236:                     (minutos_riego_fin>=0)&&(minutos_riego_fin<=60))&&
1237:                         ((hora_riego_inicio<hora_riego_fin)||
1238:                             ((hora_riego_inicio==hora_riego_fin)&&
1239:                                 (minutos_riego_inicio<=minutos_riego_fin))))
1240:     {
1241:         hora_inicio[placa] = hora_riego_inicio;
1242:         hora_fin[placa] = hora_riego_fin;
1243:         minutos_inicio[placa] = minutos_riego_inicio;
1244:         minutos_fin[placa] = minutos_riego_fin;
1245:         comando_tx=2;
1246:         estado_siguiente_maquina_lcd=STATE_02;
1247:         num_veces=0;
1248:         tecla='Z';
1249:     } else
1250:     {
1251:         lcd_gotoxy(1,2);
1252:         lcd_putc("      ");
1253:         lcd_gotoxy(1,2);
1254:         lcd_putc("error");
1255:         delay_ms(500);
1256:         lcd_gotoxy(1,2);
1257:         lcd_putc("Inicio:?:?:");
1258:         lcd_gotoxy(1,3);
1259:         lcd_putc("      ");
1260:         lcd_gotoxy(1,3);
1261:         lcd_putc("error");
1262:         delay_ms(500);
1263:         lcd_gotoxy(1,3);
1264:         lcd_putc("    Fin:?:?:");
1265:         ind1=0;
1266:         ind2=0;
1267:         ind3=0;
1268:         ind4=0;
1269:         tecla='Z';
1270:         bandera_modifica=1;
1271:     }
1272: }
1273: tecla='Z';

```

```

C:\código_estacion_concentradora\código_estacion_concentradora.c
1274:     break;
1275: case STATE_03_G:
1276:     if (num_veces==0)
1277:     {
1278:         lcd_putc('\f');
1279:         lcd_gotoxy(1,1);
1280:         lcd_putc("*VER ESTADO* 1/2");
1281:         lcd_gotoxy(1,2);
1282:         lcd_putc("Sector:?");
1283:         lcd_gotoxy(1,4);
1284:         lcd_putc("Ver:[F1]");
1285:         ind1=0;
1286:         num_veces=1;
1287:         tecla='Z';
1288:         for(i=0;i<=2;i++) datol[i]='0';
1289:     }
1290:     if ((tecla!='Z')&&(ind1==0)&&(tecla!='A')&&(tecla!='B')&&
1291:         (tecla!='C')&&(tecla!='D')&&(tecla!='E')&&(tecla!='F'))
1292:     {
1293:         datol[ind1]=tecla;
1294:         lcd_gotoxy(8,2);
1295:         lcd_putc(tecla);
1296:         datol[1]=0;
1297:         ind1=1;
1298:         tecla='Z';
1299:     }
1300:     estado_siguiente_maquina_lcd=STATE_03_G;
1301:     if (tecla=='A')
1302:     {
1303:         resultado=atoi(datol);
1304:         lcd_gotoxy(1,1);
1305:         if ((resultado>=1)&&(resultado<=max_nodo)&&
1306:             ((espera_placa_uno!=max_espera)&&(resultado!=2)) ||
1307:             ((espera_placa_dos!=max_espera)&&(resultado!=1)))
1308:         {
1309:             placa=resultado;
1310:             estado_siguiente_maquina_lcd=STATE_03_G0;
1311:             num_veces=0;ind1=0;
1312:         } else
1313:         {
1314:             lcd_gotoxy(1,2);
1315:             lcd_putc("      ");
1316:             lcd_gotoxy(1,2);
1317:             lcd_putc("error");
1318:             delay_ms(500);
1319:             lcd_gotoxy(1,2);
1320:             lcd_putc("      ");
1321:             lcd_gotoxy(1,2);
1322:             lcd_putc("Sector:?");
1323:             ind1=0;
1324:             resultado=5;
1325:         }
1326:     }
1327:     tecla='Z';
1328:     break;
1329: case STATE_03_G0:
1330:     if (num_veces==0)
1331:     {
1332:         lcd_putc('\f');
1333:         lcd_gotoxy(1,1);
1334:         lcd_putc("*VER ESTADO* 2/2");
1335:         lcd_gotoxy(1,4);
1336:         lcd_putc("Saliri:[F1]");
1337:         lcd_gotoxy(1,2);
1338:         lcd_putc("S1:");
1339:         lcd_gotoxy(9,2);
1340:         lcd_putc("S2:");

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
1341:     lcd_gotoxy(1,3);
1342:     lcd_putc("S3:");
1343:     lcd_gotoxy(9,3);
1344:     lcd_putc("S4:");
1345:     lcd_gotoxy(6,2);
1346:     lcd_putc("%");
1347:     lcd_gotoxy(14,2);
1348:     lcd_putc("%");
1349:     lcd_gotoxy(6,3);
1350:     lcd_putc("%");
1351:     lcd_gotoxy(14,3);
1352:     lcd_putc("%");
1353:     num_veces=2;
1354:     tecla='Z';
1355: }
1356: lcd_gotoxy(4,2);
1357: lcd_putc(" ");
1358: lcd_gotoxy(4,2);
1359: if (valor_humedad_uno[placa]==0) lcd_putc("OFF");
1360: else
1361: {
1362:     printf(lcd_putc,"%2lu",valor_humedad_uno[placa]-1);
1363:     lcd_gotoxy(6,2);
1364:     lcd_putc("%");
1365: }
1366: lcd_gotoxy(12,2);
1367: lcd_putc(" ");
1368: lcd_gotoxy(12,2);
1369: if (valor_humedad_dos[placa]==0) lcd_putc("OFF");
1370: else
1371: {
1372:     printf(lcd_putc,"%2lu",valor_humedad_dos[placa]-1);
1373:     lcd_gotoxy(14,2);
1374:     lcd_putc("%");
1375: }
1376: lcd_gotoxy(4,3);
1377: lcd_putc(" ");
1378: lcd_gotoxy(4,3);
1379: if (valor_humedad_tres[placa]==0) lcd_putc("OFF");
1380: else
1381: {
1382:     printf(lcd_putc,"%2lu",valor_humedad_tres[placa]-1);
1383:     lcd_gotoxy(6,3);
1384:     lcd_putc("%");
1385: }
1386: lcd_gotoxy(12,3);
1387: lcd_putc(" ");
1388: lcd_gotoxy(12,3);
1389: if (valor_humedad_cuatro[placa]==0) lcd_putc("OFF");
1390: else
1391: {
1392:     printf(lcd_putc,"%2lu",valor_humedad_cuatro[placa]-1);
1393:     estado_siguiente_maquina_lcd=STATE_03_G0;
1394:     lcd_gotoxy(14,3);
1395:     lcd_putc("%");
1396: }
1397: if (tecla=='A')
1398: {
1399:     estado_siguiente_maquina_lcd=STATE_03;
1400:     tecla='Z';
1401:     num_veces=0;
1402: }
1403: comando_tx=3;//solicito valores de humedad
1404: break;
1405: case STATE_03_H:
1406: if (num_veces==0)
1407: {

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
1408:     lcd_putc('\f');
1409:     lcd_gotoxy(1,1);
1410:     lcd_putc("      *MANUAL* 1/3");
1411:     lcd_gotoxy(1,2);
1412:     lcd_putc("Sector:?:");
1413:     lcd_gotoxy(1,3);
1414:     lcd_putc("Valv(0=OFF):?:");
1415:     lcd_gotoxy(1,4);
1416:     lcd_putc("Continua:[F1]");
1417:     ind1=0;
1418:     ind2=0;
1419:     num_veces=1;
1420:     tecla='Z';
1421:     for(i=0;i<=2;i++) dato1[i]='0';
1422:     for(i=0;i<=2;i++) dato2[i]='0';
1423:   }
1424:   if ((tecla!='Z')&&(ind1==0)&&(tecla!='A')&&(tecla!='B')&&
1425:       (tecla!='C')&&(tecla!='D')&&(tecla!='E')&&(tecla!='F'))
1426:   {
1427:     dato1[ind1]=tecla;
1428:     lcd_gotoxy(8,2);
1429:     lcd_putc(tecla);
1430:     dato1[1]=0;
1431:     ind1=1;
1432:     tecla='Z';
1433:   }
1434:   if ((tecla!='Z')&&(ind2==0)&&(tecla!='A')&&(tecla!='B')&&
1435:       (tecla!='C')&&(tecla!='D')&&(tecla!='E')&&(tecla!='F'))
1436:   {
1437:     dato2[ind2]=tecla;
1438:     lcd_gotoxy(13,3);
1439:     lcd_putc(tecla);
1440:     dato2[1]=0;
1441:     ind2=1;
1442:     tecla='Z';
1443:   }
1444: estado_siguiente_maquina_lcd=STATE_03_H;
1445: if (tecla=='A')
1446: {
1447:   tecla='Z';
1448:   resultado1=atoi(dato1);
1449:   resultado2=atoi(dato2);
1450:   if (((resultado1>=1)&&(resultado1<=max_nodo))&&((resultado2>=0)
1451:       &&(resultado2<=max_valvula))&&
1452:       ((espera_placa_uno!=max_espera)&&(resultado1!=2))||
1453:       ((espera_placa_dos!=max_espera)&&(resultado1!=1))))
1454:   {
1455:     manual_anterior[resultado1]=manual[resultado1];
1456:     if (((manual[resultado1]==21)&&(resultado2==2)) ||
1457:         ((manual[resultado1]==22)&&(resultado2==1)) ||
1458:         ((manual[resultado1]==22)&&(resultado2==2)) ||
1459:         ((manual[resultado1]==22)&&(resultado2==1)) ||
1460:         ((manual[resultado1]==23)&&(resultado2==2)) ||
1461:         ((manual[resultado1]==23)&&(resultado2==1)) ||
1462:         ((manual[resultado1]==11)&&(resultado2==2)) ||
1463:         ((manual[resultado1]==12)&&(resultado2==1)) ||
1464:         ((manual[resultado1]==12)&&(resultado2==2)) ||
1465:         ((manual[resultado1]==12)&&(resultado2==1)) ||
1466:         ((manual[resultado1]==13)&&(resultado2==2)) ||
1467:         ((manual[resultado1]==13)&&(resultado2==1)))
1468:     {
1469:       if (resultado1==1) manual[resultado1]=13;
1470:       if (resultado1==2) manual[resultado1]=23;
1471:     }
1472:     else if (resultado2==0)manual[resultado1]=44;
1473:     else manual[resultado1]=((resultado1*10)+resultado2);
1474:   estado_siguiente_maquina_lcd=STATE_03_H0;

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
1475:     num_veces=0;
1476:     ind2=0;
1477:     ind1=0;
1478:     comando_tx=4; //envio comando manual
1479: } else
1480: {
1481:     lcd_gotoxy(8,2);
1482:     lcd_putc("      ");
1483:     lcd_gotoxy(8,2);
1484:     lcd_putc("error");
1485:     delay_ms(500);
1486:     lcd_gotoxy(8,2);
1487:     lcd_putc("      ");
1488:     lcd_gotoxy(1,2);
1489:     lcd_putc("Sector:?");
1490:     lcd_gotoxy(13,3);
1491:     lcd_putc("      ");
1492:     lcd_gotoxy(12,3);
1493:     lcd_putc("error");
1494:     delay_ms(500);
1495:     lcd_gotoxy(12,3);
1496:     lcd_putc("      ");
1497:     lcd_gotoxy(1,3);
1498:     lcd_putc("Valv(0=OFF) :? ");
1499:     ind1=0;
1500:     resultado1=5;
1501:     ind2=0;
1502:     resultado2=5;
1503: }
1504: }
1505: break;
1506: case STATE_03_H0:
1507: if (num_veces==0)
1508: {
1509:     lcd_putc('\f');
1510:     lcd_gotoxy(1,1);
1511:     lcd_putc(" *MANUAL*  2/3  ");
1512:     lcd_gotoxy(1,2);
1513:     lcd_putc("Sector:");
1514:     lcd_gotoxy(1,3);
1515:     lcd_putc("Valv:");
1516:     lcd_gotoxy(1,4);
1517:     lcd_putc("Si[F1], No[F2] ");
1518:     num_veces=1;
1519: }
1520: //////////PLACA UNO/////////
1521: if ((manual[resultado1]==0)&&(resultado1==1))
1522: {
1523:     lcd_gotoxy(8,2);
1524:     lcd_putc("UNO->OFF");
1525:     lcd_gotoxy(6,3);
1526:     lcd_putc("TODAS->OFF");
1527: }
1528: if (manual[resultado1]==11)
1529: {
1530:     lcd_gotoxy(8,2);
1531:     lcd_putc("UNO->ON");
1532:     lcd_gotoxy(6,3);
1533:     lcd_putc("UNO->ON");
1534: }
1535: if (manual[resultado1]==12)
1536: {
1537:     lcd_gotoxy(8,2);
1538:     lcd_putc("UNO->ON");
1539:     lcd_gotoxy(6,3);
1540:     lcd_putc("DOS->ON");
1541: }

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
1542:     if (manual[resultado1]==13)
1543:     {
1544:         lcd_gotoxy(8, 2);
1545:         lcd_putc("UNO->ON");
1546:         lcd_gotoxy(6, 3);
1547:         lcd_putc("TODAS->ON");
1548:     }
1549:     if (manual[resultado1]==44)
1550:     {
1551:         lcd_gotoxy(8, 2);
1552:         lcd_putc("UNO->OFF");
1553:         lcd_gotoxy(6, 3);
1554:         lcd_putc("TODAS->OFF");
1555:     }
1556: //////////////PLACA DOS///////////////
1557: if ((manual[resultado1]==0)&&(resultado1==2))
1558: {
1559:     lcd_gotoxy(8, 2);
1560:     lcd_putc("DOS->OFF");
1561:     lcd_gotoxy(6, 3);
1562:     lcd_putc("TODAS->OFF");
1563: }
1564: if (manual[resultado1]==21)
1565: {
1566:     lcd_gotoxy(8, 2);
1567:     lcd_putc("DOS->ON");
1568:     lcd_gotoxy(6, 3);
1569:     lcd_putc("UNO->ON");
1570: }
1571: if (manual[resultado1]==22)
1572: {
1573:     lcd_gotoxy(8, 2);
1574:     lcd_putc("DOS->ON");
1575:     lcd_gotoxy(6, 3);
1576:     lcd_putc("DOS->ON");
1577: }
1578: if (manual[resultado1]==23)
1579: {
1580:     lcd_gotoxy(8, 2);
1581:     lcd_putc("DOS->ON");
1582:     lcd_gotoxy(6, 3);
1583:     lcd_putc("TODAS->ON");
1584: }
1585: if (manual[resultado1]==44)
1586: {
1587:     lcd_gotoxy(8, 2);
1588:     lcd_putc("DOS->OFF");
1589:     lcd_gotoxy(6, 3);
1590:     lcd_putc("TODAS->OFF");
1591: }
1592: if (tecla=='A')
1593: {
1594:     estado_siguiente_maquina_lcd=STATE_03_H1;
1595:     tecla='Z';
1596:     num_veces=0;
1597: }
1598: if (tecla=='B')
1599: {
1600:     estado_siguiente_maquina_lcd=STATE_03_H;
1601:     tecla='Z';//manual[resultado1]=11;
1602:     num_veces=0;
1603: }
1604: break;
1605: case STATE_03_H1:
1606: if (num_veces==0)
1607: {
1608:     lcd_putc('\f');

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
1609:     lcd_gotoxy(1,1);
1610:     lcd_putc(" *MANUAL*  3/3  ");
1611:     lcd_gotoxy(1,2);
1612:     lcd_putc("Continua Manual");
1613:     lcd_gotoxy(1,3);
1614:     lcd_putc("Si: [F1] ");
1615:     lcd_gotoxy(1,4);
1616:     lcd_putc("No: [F2] ");
1617:     num_veces=1;
1618: }
1619: estado_siguiente_maquina_lcd=STATE_03_H1;
1620: if (tecla=='A')
1621: {
1622:     estado_siguiente_maquina_lcd=STATE_03_H;
1623:     tecla='Z';
1624:     num_veces=0;
1625: }
1626: if (tecla=='B')
1627: {
1628:     estado_siguiente_maquina_lcd=STATE_03;
1629:     tecla='Z';
1630:     num_veces=0;
1631:     manual[1]=46;
1632:     manual[2]=46;
1633:     comando_tx=4;
1634: }
1635: break;
1636: case STATE_04_A:
1637: if (num_veces==0)
1638: {
1639:     lcd_putc('\f');
1640:     lcd_gotoxy(1,1);
1641:     lcd_putc(" *BORRAR*  1/3");
1642:     lcd_gotoxy(1,2);
1643:     lcd_putc("F1:Lim. Humedad");
1644:     lcd_gotoxy(1,4);
1645:     lcd_putc("F2:Lim. Tiempo");
1646:     num_veces=1;
1647: }
1648: if (tecla=='A')
1649: {
1650:     estado_siguiente_maquina_lcd=STATE_04_B;
1651:     num_veces=0;
1652: }
1653: if (tecla=='B')
1654: {
1655:     estado_siguiente_maquina_lcd=STATE_04_C;
1656:     tecla='Z';
1657:     num_veces=0;
1658: }
1659: break;
1660: case STATE_04_B:
1661: if (num_veces==0)
1662: {
1663:     lcd_putc('\f');
1664:     lcd_gotoxy(1,1);
1665:     lcd_putc(" *BORRAR*  2/3");
1666:     lcd_gotoxy(1,2);
1667:     lcd_putc("Lim. Humedad:?");
1668:     lcd_gotoxy(1,3);
1669:     lcd_putc("Si: [F3] ");
1670:     lcd_gotoxy(1,4);
1671:     lcd_putc("No: [F2] ");
1672:     num_veces=1;
1673: }
1674: if (tecla=='C')
1675: {

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
1676:     minima[1]=0;
1677:     maxima[1]=0;
1678:     minima[2]=0;
1679:     maxima[2]=0;
1680:     write_eeprom(minima_placa_1_e, minima[1]);
1681:     write_eeprom(minima_placa_2_e, minima[2]);
1682:     write_eeprom(maxima_placa_1_e, maxima[1] );
1683:     write_eeprom(maxima_placa_2_e, maxima[2]);
1684:     estado_siguiente_maquina_lcd=STATE_04_D;
1685:     comando_tx=1;
1686:     num_veces=0;
1687: }
1688: if (tecla=='B')
1689: {
1690:     estado_siguiente_maquina_lcd=STATE_04;
1691:     tecla='Z';
1692:     num_veces=0;
1693: }
1694: break;
1695: case STATE_04_C:
1696: if (num_veces==0)
1697: {
1698:     lcd_putc('\f');
1699:     lcd_gotoxy(1,1);
1700:     lcd_putc("*BORRAR*  2/3");
1701:     lcd_gotoxy(1,2);
1702:     lcd_putc("Limites Tiempo?:?");
1703:     lcd_gotoxy(1,3);
1704:     lcd_putc("Si: [F3]");
1705:     lcd_gotoxy(1,4);
1706:     lcd_putc("No: [F2]");
1707:     num_veces=1;
1708: }
1709: if (tecla=='C')
1710: {
1711:     hora_inicio[1] =0;
1712:     hora_inicio[2] =0;
1713:     minutos_inicio[1] =0;
1714:     minutos_inicio[2] =0;
1715:     hora_fin[1] =0;
1716:     hora_fin[2] =0;
1717:     minutos_fin[1] =0;
1718:     minutos_fin[2] =0;
1719:     comando_tx=2;
1720:     estado_siguiente_maquina_lcd=STATE_04_D;
1721:     num_veces=0;
1722: }
1723: if (tecla=='B')
1724: {
1725:     estado_siguiente_maquina_lcd=STATE_04;
1726:     tecla='Z';
1727:     num_veces=0;
1728: }
1729: break;
1730: case STATE_04_D:
1731: if (num_veces==0)
1732: {
1733:     lcd_putc('\f');
1734:     lcd_gotoxy(1,1);
1735:     lcd_putc("*BORRAR*  3/3");
1736:     lcd_gotoxy(1,2);
1737:     lcd_putc("El reinicio fue");
1738:     lcd_gotoxy(1,3);
1739:     lcd_putc("exitoso");
1740:     lcd_gotoxy(1,4);
1741:     lcd_putc("Continuar:[F1]");
1742:     num_veces=1;

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
1743:     }
1744:     estado_siguiente_maquina_lcd=STATE_04_D;
1745:     if (tecla=='A')
1746:     {
1747:         estado_siguiente_maquina_lcd=STATE_04;
1748:         tecla='Z';
1749:         num_veces=0;
1750:     }
1751: }
1752: }
1753:
1754: void maquina_de_estado_teclado_analogico()
1755: {
1756:     tarea_teclado_analogico();
1757:     setup_adc_ports(NO_ANALOGS);
1758: }
1759:
1760: void reloj_tiempo_real()
1761: {
1762:     delay_ms(15);
1763:     ds1307_get_time(hora,minutos,segundos);
1764:     delay_ms(15);
1765:     ds1307_get_date(dia,mes,año,dow1);
1766:     delay_ms(15);
1767: }
1768: void maquina_de_estado_reportes()
1769: {
1770:     switch(estado_siguiente_maquina_reportes){
1771:         case ST_INI:
1772:             if (leer_MEM(0)<1022)
1773:             {
1774:                 guardar_MEM(200,(long int)leer_MEM(0)+1);
1775:                 guardar_MEM(200,(long int)leer_MEM(0)+1);
1776:                 guardar_MEM((long int)leer_MEM(0)+1,0);
1777:                 guardar_MEM(201,(long int)leer_MEM(0)+1);
1778:                 guardar_MEM((long int)leer_MEM(0)+1,0);
1779:                 guardar_MEM((byte)dia,(long int)leer_MEM(0)+1);
1780:                 guardar_MEM((long int)leer_MEM(0)+1,0);
1781:                 guardar_MEM((byte)mes,(long int)leer_MEM(0)+1);
1782:                 guardar_MEM((long int)leer_MEM(0)+1,0);
1783:                 guardar_MEM((byte)año,(long int)leer_MEM(0)+1);
1784:                 guardar_MEM((long int)leer_MEM(0)+1,0);
1785:                 guardar_MEM((byte)hora,(long int)leer_MEM(0)+1);
1786:                 guardar_MEM((long int)leer_MEM(0)+1,0);
1787:                 guardar_MEM((byte)minutos,(long int)leer_MEM(0)+1);
1788:                 guardar_MEM((long int)leer_MEM(0)+1,0);
1789:                 if ((espera_placa_uno!=max_espera))
1790:                 {
1791:                     guardar_MEM(204,(long int)leer_MEM(0)+1);
1792:                     guardar_MEM((long int)leer_MEM(0)+1,0);
1793:                     inicializacion_1=1;
1794:                 }
1795:                 if ((espera_placa_dos!=max_espera))
1796:                 {
1797:                     guardar_MEM(205,(long int)leer_MEM(0)+1);
1798:                     guardar_MEM((long int)leer_MEM(0)+1,0);
1799:                     inicializacion_2=1;
1800:                 }
1801:                 if ((espera_placa_uno==max_espera)&&(espera_placa_dos==max_espera))
1802:                 {
1803:                     guardar_MEM(206,(long int)leer_MEM(0)+1);
1804:                     guardar_MEM((long int)leer_MEM(0)+1,0);
1805:                     inicializacion_1=1;
1806:                     inicializacion_2=1;
1807:                 }
1808:             estado_siguiente_maquina_reportes=ST_LIMITES_HUMEDAD;
1809:             veces_registro=0;

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
1810: }
1811: break;
1812: case ST_ACTIVO:
1813: if (((((estado_valvula_1_placa_1!=estado_valvula_1_placa_1_anterior) ||
1814: (estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior))&&
1815: ((tipo_rieo_placa_1==2)&&(manual[1]!= manual_anterior[1]))|||
1816: (((estado_valvula_1_placa_2!=estado_valvula_1_placa_2_anterior)|||
1817: (estado_valvula_2_placa_2!=estado_valvula_2_placa_2_anterior))&&
1818: ((tipo_rieo_placa_2==2)&&(manual[2]!= manual_anterior[2]))))&&
1819: (leer_MEM(0)<1022))
1820: estado_siguiente_maquina_reportes=ST_APERTURA_VALVULA_MANUAL;
1821:
1822: if (((((estado_valvula_1_placa_1!=estado_valvula_1_placa_1_anterior) ||
1823: (estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior))&&
1824: (tipo_rieo_placa_1==1))|||
1825: (((estado_valvula_1_placa_2!=estado_valvula_1_placa_2_anterior)|||
1826: (estado_valvula_2_placa_2!=estado_valvula_2_placa_2_anterior))&&
1827: (tipo_rieo_placa_2==1)))&&
1828: (leer_MEM(0)<1022))
1829: estado_siguiente_maquina_reportes=ST_APERTURA_VALVULA_HUMEDAD;
1830:
1831: if (((minutos==0)&&(veces_registro==0))&&(leer_MEM(0)<1022))
1832: estado_siguiente_maquina_reportes=ST_REGISTRO_VALORES_HUMEDAD;
1833: if (minutos!=0) veces_registro=0;
1834: if (((minima[1]!=minima_anterior[1])||(minima[2]!=minima_anterior[2]))||
1835: (maxima[1]!=maxima_anterior[1])||(maxima[2]!=maxima_anterior[2])) &&
1836: (leer_MEM(0)<1022))
1837: estado_siguiente_maquina_reportes=ST_LIMITES_HUMEDAD;
1838:
1839: if (((((estado_valvula_1_placa_1!=estado_valvula_1_placa_1_anterior) ||
1840: (estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior))&&
1841: ((tipo_rieo_placa_1==3)))|||
1842: (( (estado_valvula_1_placa_2!=estado_valvula_1_placa_2_anterior)|||
1843: (estado_valvula_2_placa_2!=estado_valvula_2_placa_2_anterior))&&
1844: ((tipo_rieo_placa_2==3))))&&(leer_MEM(0)<1022))
1845: estado_siguiente_maquina_reportes=ST_APERTURA_VALVULA TIEMPO;
1846: break;
1847: case ST_LIMITES_HUMEDAD:
1848: guardar_MEM(203,(long int)leer_MEM(0)+1);
1849: guardar_MEM((long int)leer_MEM(0)+1,0);
1850: if ((espera_placa_uno!=max_espera))
1851: {
1852: guardar_MEM((byte)minima[1],(long int)leer_MEM(0)+1);
1853: guardar_MEM((long int)leer_MEM(0)+1,0);
1854: guardar_MEM((byte)maxima[1],(long int)leer_MEM(0)+1);
1855: guardar_MEM((long int)leer_MEM(0)+1,0);
1856: }
1857: if ((espera_placa_dos!=max_espera))
1858: {
1859: guardar_MEM((byte)minima[2],(long int)leer_MEM(0)+1);
1860: guardar_MEM((long int)leer_MEM(0)+1,0);
1861: guardar_MEM((byte)maxima[2],(long int)leer_MEM(0)+1);
1862: guardar_MEM((long int)leer_MEM(0)+1,0);
1863: }
1864: minima_anterior[1]=minima[1];
1865: minima_anterior[2]=minima[2];
1866: maxima_anterior[1]=maxima[1];
1867: maxima_anterior[2]=maxima[2];
1868: estado_siguiente_maquina_reportes=ST_REGISTRO_VALORES_HUMEDAD;
1869: break;
1870: case ST_APERTURA_VALVULA TIEMPO:
1871: if(((estado_valvula_1_placa_1==1)&&
1872: (estado_valvula_1_placa_1!=estado_valvula_1_placa_1_anterior)) ||
1873: ((estado_valvula_2_placa_1==1)&&
1874: (estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior)))
1875: {
1876: guardar_MEM(230,(long int)leer_MEM(0)+1);

```

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
1877:     guardar_MEM((long int)leer_MEM(0)+1,0);
1878:     estado_valvula_1_placa_1_anterior=estado_valvula_1_placa_1;
1879:     estado_valvula_2_placa_1_anterior=estado_valvula_2_placa_1;
1880: }
1881: if(((estado_valvula_1_placa_2==1)&&
1882: ((estado_valvula_1_placa_2!=estado_valvula_1_placa_2_anterior))|||
1883:     ((estado_valvula_2_placa_2==1))&&
1884:     (estado_valvula_2_placa_2!=estado_valvula_2_placa_2_anterior)))
1885: {
1886:     guardar_MEM(233,(long int)leer_MEM(0)+1,0);
1887:     guardar_MEM((long int)leer_MEM(0)+1,0);
1888:     estado_valvula_1_placa_2_anterior=estado_valvula_1_placa_2;
1889:     estado_valvula_2_placa_2_anterior=estado_valvula_2_placa_2;
1890: }
1891: if(((estado_valvula_1_placa_1==0)&&(estado_valvula_1_placa_1!=
1892:                                     estado_valvula_1_placa_1_anterior)))
1893: {
1894:     guardar_MEM(231,(long int)leer_MEM(0)+1,0);
1895:     guardar_MEM((long int)leer_MEM(0)+1,0);
1896:     estado_valvula_1_placa_1_anterior=estado_valvula_1_placa_1;
1897: }
1898: if (((estado_valvula_2_placa_1==0))&&
1899:         (estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior))
1900: {
1901:     guardar_MEM(232,(long int)leer_MEM(0)+1,0);
1902:     guardar_MEM((long int)leer_MEM(0)+1,0);
1903:     estado_valvula_2_placa_1_anterior=estado_valvula_2_placa_1;
1904: }
1905: if((estado_valvula_1_placa_2==0)&&
1906:     (estado_valvula_1_placa_2!=estado_valvula_1_placa_2_anterior))
1907: {
1908:     guardar_MEM(234,(long int)leer_MEM(0)+1,0);
1909:     guardar_MEM((long int)leer_MEM(0)+1,0);
1910:     estado_valvula_1_placa_2_anterior=estado_valvula_1_placa_2;
1911: }
1912: if ((estado_valvula_2_placa_2==0)&&
1913:         (estado_valvula_2_placa_2!=estado_valvula_2_placa_2_anterior))
1914: {
1915:     guardar_MEM(235,(long int)leer_MEM(0)+1,0);
1916:     guardar_MEM((long int)leer_MEM(0)+1,0);
1917:     estado_valvula_2_placa_2_anterior=estado_valvula_2_placa_2;
1918: }
1919: guardar_MEM((byte)hora,(long int)leer_MEM(0)+1);
1920: guardar_MEM((long int)leer_MEM(0)+1,0);
1921: guardar_MEM((byte)minutos,(long int)leer_MEM(0)+1);
1922: guardar_MEM((long int)leer_MEM(0)+1,0);
1923: estado_siguiente_maquina_reportes=ST_ACTIVO;
1924: break;
1925: case ST_REGISTRO_VALORES_HUMEDAD:
1926:     if ((valor_humedad_uno[1]!=0)&&(valor_humedad_dos[1]!=0))
1927:         humedad_valvula_uno_placa_uno=
1928:             ((valor_humedad_uno[1]+valor_humedad_dos[1])/2);
1929:         else if(valor_humedad_uno[1]==0)
1930:             humedad_valvula_uno_placa_uno=valor_humedad_dos[1];
1931:         else if(valor_humedad_dos[1]==0)
1932:             humedad_valvula_uno_placa_uno=valor_humedad_uno[1];
1933:     if ((valor_humedad_tres[1]!=0)&&(valor_humedad_cuatro[1]!=0))
1934:         humedad_valvula_dos_placa_uno=
1935:             ((valor_humedad_tres[1]+valor_humedad_cuatro[1])/2);
1936:         else if(valor_humedad_tres[1]==0)
1937:             humedad_valvula_dos_placa_uno=valor_humedad_cuatro[1];
1938:         else if(valor_humedad_cuatro[1]==0)
1939:             humedad_valvula_dos_placa_uno=valor_humedad_tres[1];
1940:     if ((valor_humedad_uno[2]!=0)&&(valor_humedad_dos[2]!=0))
1941:         humedad_valvula_uno_placa_dos=

```

```

C:\código_estacion concentradora\código_estacion_concentrador.c
1944:           ((valor_humedad_uno[2]+valor_humedad_dos[2])/2);
1945:     else if(valor_humedad_uno[2]==0)
1946:         humedad_valvula_uno_placa_dos=valor_humedad_dos[2];
1947:     else if(valor_humedad_dos[2]==0)
1948:         humedad_valvula_uno_placa_dos=valor_humedad_uno[2];
1949:
1950: if ((valor_humedad_tres[2]!=0)&&(valor_humedad_cuatro[2]!=0))
1951:     humedad_valvula_dos_placa_dos=
1952:           ((valor_humedad_tres[2]+valor_humedad_cuatro[2])/2);
1953:     else if(valor_humedad_tres[2]==0)
1954:         humedad_valvula_dos_placa_dos=valor_humedad_cuatro[2];
1955:     else if(valor_humedad_cuatro[2]==0)
1956:         humedad_valvula_dos_placa_dos=valor_humedad_tres[2];
1957: guardar_MEM(202,(long int)leer_MEM(0)+1);
1958: guardar_MEM((long int)leer_MEM(0)+1,0);
1959: if ((espera_placa_uno!=max_espera))
1960:
1961:     guardar_MEM((byte)humedad_valvula_uno_placa_uno,(long int)leer_MEM(0)+1);
1962:     guardar_MEM((long int)leer_MEM(0)+1,0);
1963:     guardar_MEM((byte)humedad_valvula_dos_placa_uno,(long int)leer_MEM(0)+1);
1964:     guardar_MEM((long int)leer_MEM(0)+1,0);
1965:
1966: if ((espera_placa_dos!=max_espera))
1967:
1968:     guardar_MEM((byte) humedad_valvula_uno_placa_dos,(long int)leer_MEM(0)+1);
1969:     guardar_MEM((long int)leer_MEM(0)+1,0);
1970:     guardar_MEM((byte)humedad_valvula_dos_placa_dos,(long int)leer_MEM(0)+1);
1971:     guardar_MEM((long int)leer_MEM(0)+1,0);
1972:
1973: estado_siguiente_maquina_reportes=ST_APERTURA_VALVULA_HUMEDAD;
1974: veces_registro=1;
1975: break;
1976: case ST_APERTURA_VALVULA_HUMEDAD:
1977: if((estado_valvula_1_placa_1==1) &&
1978: (estado_valvula_1_placa_1!=estado_valvula_1_placa_1_anterior) ||
1979: (inicializacion_1==1))
1980:
1981:     guardar_MEM(220,(long int)leer_MEM(0)+1);
1982:     guardar_MEM((long int)leer_MEM(0)+1,0);
1983:     estado_valvula_1_placa_1_anterior=estado_valvula_1_placa_1;
1984:
1985: if((estado_valvula_1_placa_1==0) &&
1986: (estado_valvula_1_placa_1!=estado_valvula_1_placa_1_anterior) ||
1987: (inicializacion_1==1))
1988:
1989:     guardar_MEM(221,(long int)leer_MEM(0)+1);
1990:     guardar_MEM((long int)leer_MEM(0)+1,0);
1991:     estado_valvula_1_placa_1_anterior=estado_valvula_1_placa_1;
1992:
1993: if((estado_valvula_2_placa_1==1) &&
1994: ((estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior) ||
1995: (inicializacion_1==1)))
1996:
1997:     guardar_MEM(222,(long int)leer_MEM(0)+1);
1998:     guardar_MEM((long int)leer_MEM(0)+1,0);
1999:     estado_valvula_2_placa_1_anterior=estado_valvula_2_placa_1;
2000:
2001: if((estado_valvula_2_placa_1==0)&&
2002: ((estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior) ||
2003: (inicializacion_1==1)))
2004:
2005:     guardar_MEM(223,(long int)leer_MEM(0)+1);
2006:     guardar_MEM((long int)leer_MEM(0)+1,0);
2007:     estado_valvula_2_placa_1_anterior=estado_valvula_2_placa_1;
2008:
2009: if((estado_valvula_1_placa_2==1)&&((estado_valvula_1_placa_2!=
2010:     estado_valvula_1_placa_2_anterior)|| (inicializacion_2==1)))

```

30

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
2011: {
2012:     guardar_MEM(224, (long int) leer_MEM(0)+1);
2013:     guardar_MEM((long int) leer_MEM(0)+1, 0);
2014:     estado_valvula_1_placa_2_anterior=estado_valvula_1_placa_2;
2015: }
2016: if((estado_valvula_1_placa_2==0)&&
2017: ((estado_valvula_1_placa_2!=estado_valvula_1_placa_2_anterior)||
2018: (inicializacion_2==1)))
2019: {
2020:     guardar_MEM(225, (long int) leer_MEM(0)+1);
2021:     guardar_MEM((long int) leer_MEM(0)+1, 0);
2022:     estado_valvula_1_placa_2_anterior=estado_valvula_1_placa_2;
2023: }
2024: if((estado_valvula_2_placa_2==1)&&
2025: ((estado_valvula_2_placa_2!=estado_valvula_2_placa_2_anterior)||
2026: (inicializacion_2==1)))
2027: {
2028:     guardar_MEM(226, (long int) leer_MEM(0)+1);
2029:     guardar_MEM((long int) leer_MEM(0)+1, 0);
2030:     estado_valvula_2_placa_2_anterior=estado_valvula_2_placa_2;
2031: }
2032: if((estado_valvula_2_placa_2==0)&&
2033: ((estado_valvula_2_placa_2!=estado_valvula_2_placa_2_anterior)||
2034: (inicializacion_2==1)))
2035: {
2036:     guardar_MEM(227, (long int) leer_MEM(0)+1);
2037:     guardar_MEM((long int) leer_MEM(0)+1, 0);
2038:     estado_valvula_2_placa_2_anterior=estado_valvula_2_placa_2;
2039: }
2040: guardar_MEM((byte)hora, (long int) leer_MEM(0)+1);
2041: guardar_MEM((long int) leer_MEM(0)+1, 0);
2042: guardar_MEM((byte)minutos, (long int) leer_MEM(0)+1);
2043: guardar_MEM((long int) leer_MEM(0)+1, 0);
2044: inicializacion_1=0;
2045: inicializacion_2=0;
2046: estado_siguiente_maquina_reportes=ST_ACTIVO;
2047: break;
2048: case ST_APERTURA_VALVULA_MANUAL:
2049: if ((estado_valvula_1_placa_1!=estado_valvula_1_placa_1_anterior) ||
2050: (estado_valvula_2_placa_1!=estado_valvula_2_placa_1_anterior))
2051: {
2052: if (manual[1]==11)
2053: {
2054:     guardar_MEM(210, (long int) leer_MEM(0)+1);
2055:     guardar_MEM((long int) leer_MEM(0)+1, 0);
2056: }
2057: if (manual[1]==12)
2058: {
2059:     guardar_MEM(211, (long int) leer_MEM(0)+1);
2060:     guardar_MEM((long int) leer_MEM(0)+1, 0);
2061: }
2062: if (((manual[1]==44)) || ((manual[1]==46)))
2063: {
2064:     guardar_MEM(212, (long int) leer_MEM(0)+1);
2065:     guardar_MEM((long int) leer_MEM(0)+1, 0);
2066: }
2067: if ((manual[1]==13) && (manual_anterior[1]==11))
2068: {
2069:     guardar_MEM(211, (long int) leer_MEM(0)+1);
2070:     guardar_MEM((long int) leer_MEM(0)+1, 0);
2071: }
2072: manual_anterior[1]=manual[1];
2073: }
2074: if ((estado_valvula_1_placa_2!=estado_valvula_1_placa_2_anterior)||
2075: (estado_valvula_2_placa_2!=estado_valvula_2_placa_2_anterior))
2076: {
2077: if ((manual[1]==23) && (manual_anterior[1]==12))

```

31

```

C:\código_estacion_concentrador\código_estacion_concentrador.c
2078: {
2079:     guardar_MEM(210, (long int)leer_MEM(0)+1);
2080:     guardar_MEM((long int)leer_MEM(0)+1, 0);
2081: }
2082: if (manual[2]==21)
2083: {
2084:     guardar_MEM(213, (long int)leer_MEM(0)+1);
2085:     guardar_MEM((long int)leer_MEM(0)+1, 0);
2086: }
2087: if (manual[2]==22)
2088: {
2089:     guardar_MEM(214, (long int)leer_MEM(0)+1);
2090:     guardar_MEM((long int)leer_MEM(0)+1, 0);
2091: }
2092: if (((manual[2]==44)) || ((manual[2]==46)))
2093: {
2094:     guardar_MEM(215, (long int)leer_MEM(0)+1);
2095:     guardar_MEM((long int)leer_MEM(0)+1, 0);
2096: }
2097: if ((manual[2]==23)&&(manual_anterior[2]==21))
2098: {
2099:     guardar_MEM(214, (long int)leer_MEM(0)+1);
2100:     guardar_MEM((long int)leer_MEM(0)+1, 0);
2101: }
2102: if ((manual[2]==23)&&(manual_anterior[2]==22))
2103: {
2104:     guardar_MEM(213, (long int)leer_MEM(0)+1);
2105:     guardar_MEM((long int)leer_MEM(0)+1, 0);
2106: }
2107: manual_anterior[2]=manual[2];
2108: }
2109: guardar_MEM((byte)hora, (long int)leer_MEM(0)+1);
2110: guardar_MEM((long int)leer_MEM(0)+1, 0);
2111: guardar_MEM((byte)minutos, (long int)leer_MEM(0)+1);
2112: guardar_MEM((long int)leer_MEM(0)+1, 0);
2113: estado_siguiente_maquina_reportes=ST_ACTIVO;
2114: break;
2115: }
2116: }
2117:
2118: void maquina_de_estado_bluetooth()
2119: {
2120:     switch (estado_siguiente_maquina_blue){
2121:         case ST_INI_BLUE:
2122:             estado_siguiente_maquina_blue=ST_BLUE_ACTIVO;
2123:             break;
2124:         case ST_BLUE_ACTIVO:
2125:             if (tx_blue==1) estado_siguiente_maquina_blue= ST_BLUE_TX;
2126:             if (rx_blue==1)estado_siguiente_maquina_blue=ST_BLUE_RX;
2127:             break;
2128:         case ST_BLUE_TX:
2129:             fprintf(PORT2,"Datos guardados:\n\r");
2130:             for(i=0; i<=(long int)leer_MEM(0); i++)
2131:             {
2132:                 delay_us(650);
2133:                 itoa((int)leer_MEM(i),10, string);
2134:                 fprintf(PORT2, string) ;
2135:                 fprintf(PORT2, ";") ;
2136:                 if ((int)leer_MEM(i+1)>100) fprintf(PORT2, "\n\r");
2137:             }
2138:             fprintf(PORT2, "\n\r");
2139:             tx_blue=0;
2140:             estado_siguiente_maquina_blue=ST_BLUE_ACTIVO;
2141:             break;
2142:         case ST_BLUE_RX:
2143:             if (caracter_rx_blue=='E')
2144:             {

```

Librerías utilizadas en la estación concentradora:

Librería implementada para manejar el teclado analógico:

```
C:\código_teclado_analogico\teclado_adc.c
1: //////////////////////////////////////////////////////////////////// TECLADO ANALOGICO/////////////////////////////////////////////////////////////////
2: int16 valor_adc_leido=0;
3: int16 valor_adc_tecla_presionada;
4: char tecla;
5: int bandera_presionada,i;
6: //const int16 valores_adc_teclas[17]={0,308,317,327,337,443,458,484,
7: //505,550,572,613,642,695,735,802,863};//valores teoricos
8: const int16 valores_adc_teclas[17]={0,308,315,326,335,540,563,601,
9: //633,438,454,477,497,27,63,126,181};//valores practicos
10: const char simbolo_tecla[17]={'Z','C','D','E','F','B','3','6','9',
11: //A,'2','5','8','0','1','4','7'};
12: const int16 tolerancia=5;
13: int max_tecla=18;
14: const int INIT = 0;
15: const int SUELTA = 1;
16: const int VALIDA = 2;
17: const int PRESIONA = 3;
18: int estado_siguiente_maquina_teclado=0;
19:
20: void tarea_teclado_analogico(void){
21: switch(estado_siguiente_maquina_teclado)
22: {
23: case INIT:
24: int i,max=16;
25: setup_adc_ports(ANO|VSS_VDD);
26: setup_adc(ADC_CLOCK_DIV_8);
27: enable_interrupts(global);
28: bandera_presionada=0;
29: estado_siguiente_maquina_teclado=SUELTA;
30: break;
31: case SUELTA:
32: valor_adc_leido=0;
33: set_adc_channel(0); //habilitacion canal 0
34: delay_ms (2);
35: valor_adc_leido =0;
36: for(i=1;i<=100;i++)
37: {
38: valor_adc_leido =read_adc()+valor_adc_leido;
39: if (i==100)valor_adc_leido =(valor_adc_leido/i);
40: }
41: max=16;
42: if((valor_adc_leido>10)&&(bandera_presionada==0))
43: {
44: bandera_presionada=1;
45: estado_siguiente_maquina_teclado=PRESIONA;
46: valor_adc_tecla_presionada=valor_adc_leido;
47: }
48: break;
49: case PRESIONA:
50: set_adc_channel(0); //habilitacion canal 0
51: delay_ms (2);
52: valor_adc_leido =0;
53: for(i=1;i<=100;i++)
54: {
55: valor_adc_leido =read_adc()+valor_adc_leido;
56: if (i==100)valor_adc_leido =(valor_adc_leido/i);
57: }
58: if((bandera_presionada==1)&&(valor_adc_leido<10))
59: estado_siguiente_maquina_teclado=VALIDA;
60: break;
61: case VALIDA:
62: max=max_tecla;
63: for(i=0;i<=max;i++)
64: {
65: if((valor_adc_tecla_presionada>(valores_adc_teclas[i]-tolerancia))&&
66: (valor_adc_tecla_presionada<(valores_adc_teclas[i]+tolerancia)))
67: {
```

1

```
C:\código_teclado_analogico\teclado_adc.c
68:     max=i;
69:     tecla=simbolo_tecla[i];
70: }
71: }
72: if (i>=max_tecla) tecla='Z';
73: bandera_presionada=0;
74: estado_siguiente_maquina_teclado=SUELTA;
75: break;
76: }
77: }
```

Librería implementada en la estación concentradora para manejar los relés y los pulsadores:

```
C:\pulsadores_reles\manejo_pulsadores_reles.c
1: int lee_pulsador(byte pin)
2: {
3:     if (input(pin)==1) return(1); else return (0);
4: }
5: void activar_rele(byte pin)
6: {
7:     output_high(pin);
8: }
9: void desactivar_rele(byte pin)
10: {
11:     output_low(pin);
12: }
13: }
```

Librería utilizada en la estación concentradora para el manejo de los leds multiplexados:

```
C:\leds\manejo_leds.c
1: void encender_led(int led_numero)
2: {
3:     if (led_numero==0)
4:     {
5:         output_low(PIN_C0);
6:         output_low(PIN_C1);
7:         output_low(PIN_D3);
8:         output_low(PIN_C2);
9:     }
10:    if (led_numero==1)
11:    {
12:        output_low(PIN_C0);
13:        output_high(PIN_C1);
14:        output_float(PIN_D3);
15:        output_float(PIN_C2);
16:    }
17:    if (led_numero==2)
18:    {
19:        output_float(PIN_C0);
20:        output_low(PIN_C1);
21:        output_high(PIN_D3);
22:        output_float(PIN_C2);
23:    }
24:    if (led_numero==3)
25:    {
26:        output_float(PIN_C0);
27:        output_float(PIN_C1);
28:        output_low(PIN_D3);
29:        output_high(PIN_C2);
30:    }
31:    if (led_numero==4)
32:    {
33:        output_high(PIN_C0);
34:        output_low(PIN_C1);
35:        output_float(PIN_D3);
36:        output_float(PIN_C2);
37:    }
38:    if (led_numero==5)
39:    {
40:        output_float(PIN_C0);
41:        output_high(PIN_C1);
42:        output_low(PIN_D3);
43:        output_float(PIN_C2);
44:    }
45:    if (led_numero==6)
46:    {
47:        output_float(PIN_C0);
48:        output_float(PIN_C1);
49:        output_high(PIN_D3);
50:        output_low(PIN_C2);
51:    }
52:    if (led_numero==7)
53:    {
54:        output_high(PIN_C0);
55:        output_float(PIN_C1);
56:        output_low(PIN_D3);
57:        output_float(PIN_C2);
58:    }
59:    if (led_numero==8)
60:    {
61:        output_low(PIN_C0);
62:        output_float(PIN_C1);
63:        output_high(PIN_D3);
64:        output_float(PIN_C2);
65:    }
66:    if (led_numero==9)
67:    {
```

```
C:\leds\manejo_leds.c
68:     output_float(PIN_C0);
69:     output_low(PIN_C1);
70:     output_float(PIN_D3);
71:     output_high(PIN_C2);
72: }
73: if (led_numero==10)
74: {
75:     output_float(PIN_C0);
76:     output_high(PIN_C1);
77:     output_float(PIN_D3);
78:     output_low(PIN_C2);
79: }
80: if (led_numero==11)
81: {
82:     output_low(PIN_C0);
83:     output_float(PIN_C1);
84:     output_float(PIN_D3);
85:     output_high(PIN_C2);
86: }
87: if (led_numero==12)
88: {
89:     output_high(PIN_C0);
90:     output_float(PIN_C1);
91:     output_float(PIN_D3);
92:     output_low(PIN_C2);
93: }
94: }
```

Librería utilizada para programar el reloj de tiempo real, DS1307:

```

C:\reloj de tiempo real\ds1307.c
1: /////////////////////////////////////////////////////////////////////
2: /// DS1307.C
3: /// Driver for Real Time Clock
4: /// modified by Redpic 08/2006
5: /// http://picmania.garcia-cuervo.com
6: ///
7: /// void ds1307_init(val)
8: /// - Enable oscillator without clearing the seconds register
9: /// used when PIC loses power and DS1307 run from 3V BAT
10:/// - Config Control Register with next parameters:
11: DS1307_ALL_DISABLED All disabled
12: DS1307_OUT_ON_DISABLED_HIHG Out to Hight on Disable Out
13: DS1307_OUT_ENABLED Out Enabled
14: DS1307_OUT_1_HZ Freq. Out to 1 Hz
15: DS1307_OUT_4_KHZ Freq. Out to 4.096 Khz
16: DS1307_OUT_8_KHZ Freq. Out to 8.192 Khz
17: DS1307_OUT_32_KHZ Freq. Out to 32.768 Khz
18: ///
19: /// Example init:
20: ds1307_init(DS1307_ALL_DISABLED);
21: ds1307_init(DS1307_OUT_ENABLED | DS1307_OUT_1_HZ);
22: ///
23: void ds1307_set_date_time(day,mth,year,dow,hour,min,sec)
24: - Set the date/time
25: ///
26: void ds1307_get_date(day,mth,year,dow) - Get the date
27: ///
28: void ds1307_get_time(hr,min,sec) - Get the time
29: ///
30: char ds1307_read_nvram_byte(char addr) - Read byte in address
31: ///
32: void ds1307_write_nvram_byte(char addr, char value) - Write byte in address
33: ///
34: ///
35: void ds1307_get_day_of_week(char* ptr) - Get string Day Of Week
36: ///
37: If defined USE_INTERRUPTS all functions disable Global
38: Interrupts on starts and
39: enable Global on ends else usar can do it himself
40: ///
41: /////////////////////////////////////////////////////////////////////
42: #
43: #ifndef RTC_SDA
44: #define RTC_SDA PIN_C4
45: #define RTC_SCL PIN_C3
46: #endif
47: #
48: #use i2c(master, sda=RTC_SDA, scl=RTC_SCL)
49: #
50: #define DS1307_ALL_DISABLED 0b00000000 // All disabled
51: #define DS1307_OUT_ON_DISABLED_HIHG 0b10000000 // Out to Hight on Disable Out
52: #define DS1307_OUT_ENABLED 0b00010000 // Out Enabled
53: #define DS1307_OUT_1_HZ 0b00000000 // Freq. Out to 1 Hz
54: #define DS1307_OUT_4_KHZ 0b00000001 // Freq. Out to 4.096 Khz
55: #define DS1307_OUT_8_KHZ 0b00000010 // Freq. Out to 8.192 Khz
56: #define DS1307_OUT_32_KHZ 0b00000011 // Freq. Out to 32.768 Khz
57: #
58: #define Start_user_address_nvram 0x08
59: #define End_user_address_nvram 0x3f
60: #
61: char days_of_week[7][11]={"Lunes\0","Martes\0","Miércoles\0","Jueves\0",
62: "Viernes\0","Sábado\0","Domingo\0"};
63: #
64: byte ds1307_bin2bcd(byte binary_value);
65: byte ds1307_bcd2bin(byte bcd_value);
66: #
67: void ds1307_init(int val){

```

```

C:\reloj de tiempo real\_ds1307.c
68:
69:     byte seconds = 0;
70:
71: #ifndef USE_INTERRUPTS
72:     disable_interrupts(global);
73: #endif
74:
75:     i2c_start();
76:     i2c_write(0xD0);
77:     i2c_write(0x00);
78:     i2c_start();
79:     i2c_write(0xD1);
80:     seconds = ds1307_bcd2bin(i2c_read(0));
81:     i2c_stop();
82:     seconds &= 0x7F;
83:
84:     delay_us(3);
85:
86:     i2c_start();
87:     i2c_write(0xD0);
88:     i2c_write(0x00);
89:     i2c_write(ds1307_bin2bcd(seconds));
90:     i2c_start();
91:     i2c_write(0xD0);
92:     i2c_write(0x07);
93:     i2c_write(val);
94:     i2c_stop();
95:
96: #ifndef USE_INTERRUPTS
97:     enable_interrupts(global);
98: #endif
99:
100: }
101:
102: void ds1307_set_date_time(byte day, byte mth, byte year, byte dow, byte hr,
103:                             byte min, byte sec){
104:
105: #ifndef USE_INTERRUPTS
106:     disable_interrupts(global);
107: #endif
108:
109:     sec &= 0x7F;
110:     hr &= 0x3F;
111:
112:     i2c_start();
113:     i2c_write(0xD0);
114:     i2c_write(0x00);
115:     i2c_write(ds1307_bin2bcd(sec));
116:     i2c_write(ds1307_bin2bcd(min));
117:     i2c_write(ds1307_bin2bcd(hr));
118:     i2c_write(ds1307_bin2bcd(dow));
119:     i2c_write(ds1307_bin2bcd(day));
120:     i2c_write(ds1307_bin2bcd(mth));
121:     i2c_write(ds1307_bin2bcd(year));
122:     i2c_stop();
123:
124: #ifndef USE_INTERRUPTS
125:     enable_interrupts(global);
126: #endif
127:
128: }
129:
130: void ds1307_get_date(byte &day, byte &mth, byte &year, byte &dow){
131:
132: #ifndef USE_INTERRUPTS
133:     disable_interrupts(global);
134: #endif

```

```

C:\reloj de tiempo real\ds1307.c
135:
136:     i2c_start();
137:     i2c_write(0xD0);
138:     i2c_write(0x03);
139:     i2c_start();
140:     i2c_write(0xD1);
141:     dow = ds1307_bcd2bin(i2c_read() & 0x7f);
142:     day = ds1307_bcd2bin(i2c_read() & 0x3f);
143:     mth = ds1307_bcd2bin(i2c_read() & 0x1f);
144:     year = ds1307_bcd2bin(i2c_read(0));
145:     i2c_stop();
146:
147: #ifndef USE_INTERRUPTS
148:     enable_interrupts(global);
149: #endif
150:
151: }
152:
153: void ds1307_get_time(byte &hr, byte &min, byte &sec){
154:
155: #ifndef USE_INTERRUPTS
156:     disable_interrupts(global);
157: #endif
158:
159:     i2c_start();
160:     i2c_write(0xD0);
161:     i2c_write(0x00);
162:     i2c_start();
163:     i2c_write(0xD1);
164:     sec = ds1307_bcd2bin(i2c_read() & 0x7f);
165:     min = ds1307_bcd2bin(i2c_read() & 0x7f);
166:     hr = ds1307_bcd2bin(i2c_read(0) & 0x3f);
167:     i2c_stop();
168:
169: #ifndef USE_INTERRUPTS
170:     enable_interrupts(global);
171: #endif
172:
173: }
174:
175:
176: char ds1307_read_nvram_byte(char addr){
177:
178:     char retval;
179:
180: #ifndef USE_INTERRUPTS
181:     disable_interrupts(global);
182: #endif
183:
184:     i2c_start();
185:     i2c_write(0xD0);
186:     i2c_write(addr);
187:
188:     i2c_start();
189:     i2c_write(0xD1);
190:     retval = i2c_read(0);
191:     i2c_stop();
192:
193:     return(retval);
194:
195: #ifndef USE_INTERRUPTS
196:     enable_interrupts(global);
197: #endif
198:
199: }
200:
201: void ds1307_write_nvram_byte(char addr, char value){

```

```

C:\reloj de tiempo real\_ds1307.c
202:
203: #ifndef USE_INTERRUPTS
204:     disable_interrupts(global);
205: #endif
206:
207:     i2c_start();
208:     i2c_write(0xD0);
209:     i2c_write(addr);
210:     i2c_write(value);
211:     i2c_stop();
212:
213: #ifndef USE_INTERRUPTS
214:     enable_interrupts(global);
215: #endif
216:
217: }
218:
219: void ds1307_get_day_of_week(char* ptr){
220:
221:     byte lday;
222:     byte lmonth;
223:     byte lyr;
224:     byte ldow;
225:     ds1307_get_date(lday,lmonth,lyr,ldow);
226:     sprintf(ptr,"%s",days_of_week[ldow]);
227: }
228:
229: ///////////////////////////////////////////////////
230:
231: byte ds1307_bin2bcd(byte binary_value){
232:
233:     byte temp;
234:     byte retval;
235:
236:     temp = binary_value;
237:     retval = 0;
238:     while(1){
239:         if(temp >= 10){
240:             temp -= 10;
241:             retval += 0x10;
242:         }else{
243:             retval += temp;
244:             break;
245:         }
246:     }
247:     return(retval);
248: }
249:
250: byte ds1307_bcd2bin(byte bcd_value){
251:
252:     byte temp;
253:
254:     temp = bcd_value;
255:     temp >= 1;
256:     temp &= 0x78;
257:     return(temp + (temp >> 2) + (bcd_value & 0x0f));
258: }
259: ///////////////////////////////////////////////////

```

Librería utilizada en la estación concentradora para comandar el display de LCD:

```
C:\display\lcd_16x4.c
1: // Flex_LCD416.c
2:
3: // These pins are for my Microchip PicDem2-Plus board,
4: // which I used to test this driver.
5: // An external 20x4 LCD is connected to these pins.
6: // Change these pins to match your own board's connections.
7:
8:
9:
10: #define LCD_DB4    PIN_D4
11: #define LCD_DB5    PIN_D2
12: #define LCD_DB6    PIN_D1
13: #define LCD_DB7    PIN_D0
14: #define LCD_RS     PIN_D6
15: #define LCD_RW     PIN_D5
16: #define LCD_E      PIN_D7
17:
18:
19: /*
20: // To prove that the driver can be used with random
21: // pins, I also tested it with these pins:
22: #define LCD_DB4    PIN_D4
23: #define LCD_DB5    PIN_B1
24: #define LCD_DB6    PIN_C5
25: #define LCD_DB7    PIN_B5
26:
27: #define LCD_RS     PIN_E2
28: #define LCD_RW     PIN_B2
29: #define LCD_E      PIN_D6
30: */
31:
32: // If you want only a 6-pin interface to your LCD, then
33: // connect the R/W pin on the LCD to ground, and comment
34: // out the following line. Doing so will save one PIC
35: // pin, but at the cost of losing the ability to read from
36: // the LCD. It also makes the write time a little longer
37: // because a static delay must be used, instead of polling
38: // the LCD's busy bit. Normally a 6-pin interface is only
39: // used if you are running out of PIC pins, and you need
40: // to use as few as possible for the LCD.
41: #define USE_RW_PIN  1
42:
43:
44: // These are the line addresses for most 4x20 LCDs.
45: #define LCD_LINE_1_ADDRESS 0x00
46: #define LCD_LINE_2_ADDRESS 0x40
47: #define LCD_LINE_3_ADDRESS 0x10
48: #define LCD_LINE_4_ADDRESS 0x50
49:
50: // These are the line addresses for LCD's which use
51: // the Hitachi HD66712U controller chip.
52: /*
53: #define LCD_LINE_1_ADDRESS 0x00
54: #define LCD_LINE_2_ADDRESS 0x20
55: #define LCD_LINE_3_ADDRESS 0x40
56: #define LCD_LINE_4_ADDRESS 0x60
57: */
58:
59:
60: //=====
61:
62: #define lcd_type 2    // 0=5x7, 1=5x10, 2=2 lines(or more)
63:
64: int8 lcd_line;
65:
66: int8 const LCD_INIT_STRING[4] =
67: {
```

```

C:\display\lcd_16x4.c
68: 0x20 | (lcd_type << 2), // Set mode: 4-bit, 2+ lines, 5x8 dots
69: 0xc, // Display on
70: 1, // Clear display
71: 6 // Increment cursor
72: };
73:
74:
75: //-----
76: void lcd_send_nibble(int8 nibble)
77: {
78: // Note: !! converts an integer expression
79: // to a boolean (1 or 0).
80: output_bit(LCD_DB4, !(nibble & 1));
81: output_bit(LCD_DB5, !(nibble & 2));
82: output_bit(LCD_DB6, !(nibble & 4));
83: output_bit(LCD_DB7, !(nibble & 8));
84:
85: delay_cycles(1);
86: output_high(LCD_E);
87: delay_us(2);
88: output_low(LCD_E);
89: }
90:
91: //-----
92: // This sub-routine is only called by lcd_read_byte().
93: // It's not a stand-alone routine. For example, the
94: // R/W signal is set high by lcd_read_byte() before
95: // this routine is called.
96:
97: #ifdef USE_RW_PIN
98: int8 lcd_read_nibble(void)
99: {
100: int8 retval;
101: // Create bit variables so that we can easily set
102: // individual bits in the retval variable.
103: #bit retval_0 = retval.0
104: #bit retval_1 = retval.1
105: #bit retval_2 = retval.2
106: #bit retval_3 = retval.3
107:
108: retval = 0;
109:
110: output_high(LCD_E);
111: delay_us(1);
112:
113: retval_0 = input(LCD_DB4);
114: retval_1 = input(LCD_DB5);
115: retval_2 = input(LCD_DB6);
116: retval_3 = input(LCD_DB7);
117:
118: output_low(LCD_E);
119: delay_us(1);
120:
121: return(retval);
122: }
123: #endif
124:
125: //-----
126: // Read a byte from the LCD and return it.
127:
128: #ifdef USE_RW_PIN
129: int8 lcd_read_byte(void)
130: {
131: int8 low;
132: int8 high;
133:
134: output_high(LCD_RW);

```

```

C:\display\lcd_16x4.c
135: delay_cycles(1);
136:
137: high = lcd_read_nibble();
138:
139: low = lcd_read_nibble();
140:
141: return( (high<<4) | low);
142: }
143: #endif
144:
145: //-----
146: // Send a byte to the LCD.
147: void lcd_send_byte(int8 address, int8 n)
148: {
149: output_low(LCD_RS);
150:
151: #ifdef USE_RW_PIN
152: while(bit_test(lcd_read_byte(),7)) ;
153: #else
154: delay_us(60);
155: #endif
156:
157: if(address)
158:     output_high(LCD_RS);
159: else
160:     output_low(LCD_RS);
161:
162: delay_cycles(1);
163:
164: #ifdef USE_RW_PIN
165: output_low(LCD_RW);
166: delay_cycles(1);
167: #endif
168:
169: output_low(LCD_E);
170:
171: lcd_send_nibble(n >> 4);
172: lcd_send_nibble(n & 0xf);
173: }
174: //-----
175:
176: void lcd_init(void)
177: {
178: int8 i;
179:
180: lcd_line = 1;
181:
182: output_low(LCD_RS);
183:
184: #ifdef USE_RW_PIN
185: output_low(LCD_RW);
186: #endif
187:
188: output_low(LCD_E);
189:
190: // Some LCDs require 15 ms minimum delay after
191: // power-up. Others require 30 ms. I'm going
192: // to set it to 35 ms, so it should work with
193: // all of them.
194: delay_ms(35);
195:
196: for(i=0 ;i < 3; i++)
197: {
198:     lcd_send_nibble(0x03);
199:     delay_ms(5);
200: }
201:
```

```

C:\display\lcd_16x4.c
202: lcd_send_nibble(0x02);
203:
204: for(i=0; i < sizeof(LCD_INIT_STRING); i++)
205: {
206:     lcd_send_byte(0, LCD_INIT_STRING[i]);
207:
208:     // If the R/W signal is not used, then
209:     // the busy bit can't be polled. One of
210:     // the init commands takes longer than
211:     // the hard-coded delay of 50 us, so in
212:     // that case, lets just do a 5 ms delay
213:     // after all four of them.
214: #ifndef USE_RW_PIN
215:     delay_ms(5);
216: #endif
217: }
218:
219: }
220:
221: //-----
222:
223: void lcd_gotoxy(int8 x, int8 y)
224: {
225:     int8 address;
226:
227:
228:     switch(y)
229:     {
230:         case 1:
231:             address = LCD_LINE_1_ADDRESS;
232:             break;
233:
234:         case 2:
235:             address = LCD_LINE_2_ADDRESS;
236:             break;
237:
238:         case 3:
239:             address = LCD_LINE_3_ADDRESS;
240:             break;
241:
242:         case 4:
243:             address = LCD_LINE_4_ADDRESS;
244:             break;
245:
246:         default:
247:             address = LCD_LINE_1_ADDRESS;
248:             break;
249:
250:     }
251:
252:     address += x-1;
253:     lcd_send_byte(0, 0x80 | address);
254: }
255:
256: //-----
257: void lcd_putc(char c)
258: {
259:     switch(c)
260:     {
261:         case '\f':
262:             lcd_send_byte(0, 1);
263:             lcd_line = 1;
264:             delay_ms(2);
265:             break;
266:
267:         case '\n':
268:             lcd_gotoxy(1, ++lcd_line);

```

```

C:\display\lcd_16x4.c
269:         break;
270:
271:     case '\b':
272:         lcd_send_byte(0, 0x10);
273:         break;
274:
275:     default:
276:         lcd_send_byte(1, c);
277:         break;
278:     }
279: }
280:
281: //-----
282: #ifndef USE_RW_PIN
283: char lcd_getc(int8 x, int8 y)
284: {
285:     char value;
286:
287:     lcd_gotoxy(x, y);
288:
289:     // Wait until busy flag is low.
290:     while(bit_test(lcd_read_byte(), 7));
291:
292:     output_high(LCD_RS);
293:     value = lcd_read_byte();
294:     output_low(LCD_RS);
295:
296:     return(value);
297: }
298: #endif
299:
300: void lcd_setcursor_vb(short visible, short blink) {
301:     lcd_send_byte(0, 0xC | (visible<<1)|blink);
302: }

```

Software embebido en el microcontrolador de la estación remota:

```
C:\código estacion remota\codigo_estacion_remota.c
1: #include <18F4550.h>
2: #device adc=10
3: #FUSES NOMCLR
4: #FUSES PUT
5: #FUSES NOWDT
6: #FUSES WDT128
7: #FUSES PLL1
8: #FUSES CPUDIV1
9: #FUSES NOUSBDIV
10: #FUSES HS
11: #FUSES NOBROWNOUT
12: #FUSES NOLVP
13: #FUSES NOXINST
14: #use delay(clock=20000000)
15: #use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,
               bits=8, stop=2, stream=PORT1,enable=PIN_D3)
16: #include <adc.c>
17: #use standard_io (a)
18: #use standard_io (b)
19: #use standard_io (c)
20: #use standard_io (d)
21: #use standard_io (e)
22: #use standard_io (e)
23: //##define LCD_ENABLE_PIN PIN_D7
24: //##define LCD_RS_PIN PIN_D6
25: //##define LCD_RW_PIN PIN_D5
26: //##define LCD_DATA4 PIN_D4
27: //##define LCD_DATA5 PIN_D2
28: //##define LCD_DATA6 PIN_D1
29: //##define LCD_DATA7 PIN_D0
30: //##include <LCD_nuevo.C>
31: #define char_int(c) ((int)((c)-48))
32: #byte UCON=0xF6D
33: #bit USBEN=UCON.3
34: #byte UCFG = 0xF6F
35: #bit UTRDIS = UCFG.3
36: #INCLUDE <string.h>
37: #INCLUDE <stdlib.h>
38:
39: #use rtos(timer = 0, minor_cycle =5ms)
40: int16 valor_humedad[4];
41: int16 hum_i_1,hum_i_2,hum_i_3,hum_i_4;
42: int16 hum_f_1,hum_f_2,hum_f_3,hum_f_4;
43: int16 item;
44: int16 n_menus;
45: int16 i_rx=0;
46: int16 dato_completo=0;
47: int regando_1=0,regando_2=0,regando_3=0,regando_4=0,regando_t=0;
48: int sube_1=0,sube_2=0,sube_3=0,sube_4=0;
49: int baja_1=0,baja_2=0,baja_3=0,baja_4=0;
50: char dato_tx[30];
51: char dato_rx[30];
52: char cadena[6];
53: char dato_aux[5],dato_aux2[5];
54: int numero=0,fin_pal=0,dato_comp=0,j=0,s=0,m=0;
55: const int STATE_INI = 1;
56: const int STATE_SEND = 2;
57: const int STATE_ACTIVO = 3;
58: int estado_siguiente=1;
59: const int STATE_INICIAL = 0;
60: const int STATE_ACT = 1;
61: const int STATE_MANUAL = 2;
62: const int STATE TIEMPO = 3;
63: const int STATE_HUMEDAD = 4;
64: const int STATE_ACTIVAR_VALVULA = 5;
65: const int STATE_DESACTIVAR_VALVULA = 6;
66: int estado_siguiente_riege=0;
67: int hora=0, minutos=0;
```

```

C:\código estacion remota\codigo_estacion_remota.c
68: int minima=0, maxima=0, hora_inicio=0, minutos_inicio=0;
69: int hora_fin=0, minutos_fin=0;
70: int salida[4];
71: int manual=0;
72: int16 comando_rx=10;
73: int16 comando_tx=1;
74: int16 estado_valvula_1_placa_1=0;
75: int16 estado_valvula_2_placa_1=0;
76: int16 humedad_ciclo_uno=0;
77: int16 humedad_ciclo_dos=0;
78: int tipo_riego=0;
79: int tipo_riego_anterior =0;
80: int minima_guardada=0;
81: int maxima_guardada=0;
82: int entro_manual=0;
83: int regando_1_anterior=0;
84: int regando_2_anterior=0;
85: int regando_t_anterior=0;
86: #int_rda
87: STATE_STORE()
88: {
89:     dato_rx[i_rx]=getc();
90:     if (dato_rx[i_rx]=='&')
91:     {
92:         dato_completo=1;
93:         i_rx=0;
94:     }
95:     else
96:     {
97:         dato_completo=0;
98:         i_rx++;
99:     }
100: }
101:
102: int lee_pulsador(byte pin)
103: {
104:     if (input(pin)==1) return(1); else return (0);
105: }
106: void activar_rele(byte pin)
107: {
108:     output_high(pin);
109: }
110: void desactivar_rele(byte pin)
111: {
112:     output_low(pin);
113: }
114: void encender_led(byte pin)
115: {
116:     output_high(pin);
117: }
118: void apagar_led(byte pin)
119: {
120:     output_low(pin);
121: }
122: void envio_dato()
123: {
124:     int i_tx,fin_dto=40;
125:     for(i_tx=0;i_tx<=fin_dto;i_tx++)
126:     {
127:         putc(dato_tx[i_tx]);
128:         if (dato_tx[i_tx]=='&') fin_dto=i_tx;
129:     }
130: }
131: void armar_trama()
132: {
133:     dato_tx[0]='0';
134:     strcpy(dato_tx,"00000001");

```

```

C:\código estacion remota\codigo_estacion_remota.c
135: sprintf(cadena,"%lu",comando_tx);
136: strcat(dato_tx,cadena);
137: sprintf(cadena,"+");
138: strcat(dato_tx,cadena);
139: if (comando_tx==1)
140: {
141:     sprintf(cadena,"%lu",valor_humedad[0]);
142:     strcat(dato_tx,cadena);
143:     sprintf(cadena,"+");
144:     strcat(dato_tx,cadena);
145:     sprintf(cadena,"%lu",valor_humedad[1]);
146:     strcat(dato_tx,cadena);
147:     sprintf(cadena,"+");
148:     strcat(dato_tx,cadena);
149:     sprintf(cadena,"%lu",valor_humedad[2]);
150:     strcat(dato_tx,cadena);
151:     sprintf(cadena,"+");
152:     strcat(dato_tx,cadena);
153:     sprintf(cadena,"%lu",valor_humedad[3]);
154:     strcat(dato_tx,cadena);
155:     sprintf(cadena,"&");
156:     strcat(dato_tx,cadena);
157: }
158: if (comando_tx==2)
159: {
160:     sprintf(cadena,"%lu",estado_valvula_1_placa_1);
161:     strcat(dato_tx,cadena);
162:     sprintf(cadena,"+");
163:     strcat(dato_tx,cadena);
164:     sprintf(cadena,"%lu",estado_valvula_2_placa_1);
165:     strcat(dato_tx,cadena);
166:     sprintf(cadena,"+");
167:     strcat(dato_tx,cadena);
168:     sprintf(cadena,"%u",tipo_riego);
169:     strcat(dato_tx,cadena);
170:     sprintf(cadena,"&");
171:     strcat(dato_tx,cadena);
172:     tipo_riego_anterior=0;
173: }
174: }
175: void deco_tramaRx()
176: {
177:     numero=0;
178:     fin_pal=65;
179:     dat_comp=0;
180:     j=0;
181:     for (m=8;m<=fin_pal;m++)
182:     {
183:         if ((dato_rx[m]=='&') && (dat_comp==1))
184:         {
185:             dat_comp=1;
186:             fin_pal=m;
187:         }
188:         if ((dato_rx[m]!='+') && (dato_rx[m]!='&'))
189:         {
190:             dato_aux[j]=dato_rx[m];
191:             j=j+1;
192:             dat_comp=0;
193:         } else {
194:             for (s=0;s<=2;s++) dato_aux2[s]='0';
195:             for (s=0;s<j;s++) dato_aux2[(4-j)+s]=dato_aux[s];
196:             j=0;
197:             if ((numero==0) && (dat_comp!=1))
198:             {
199:                 comando_rx=atoi32(dato_aux2);
200:                 numero=1; dat_comp=1;
201:             }
}

```

```

C:\código estacion remota\codigo_estacion_remota.c
202:         if ((numero==1)&&(dat_comp!=1))
203:         {
204:             if (comando_rx==1) hora=atoi32(dato_aux2);
205:             if (comando_rx==2) hora=atoi32(dato_aux2);
206:             if (comando_rx==3) hora=atoi32(dato_aux2);
207:             if (comando_rx==4) hora=atoi32(dato_aux2);
208:             if (comando_rx==8) hora=atoi32(dato_aux2);
209:             numero=2;
210:             dat_comp=1;
211:         }
212:         if ((numero==2)&&(dat_comp!=1))
213:         {
214:             if (comando_rx==1) minutos=atoi32(dato_aux2);
215:             if (comando_rx==2) minutos=atoi32(dato_aux2);
216:             if (comando_rx==3) minutos=atoi32(dato_aux2);
217:             if (comando_rx==4) minutos=atoi32(dato_aux2);
218:             if (comando_rx==8) minutos=atoi32(dato_aux2);
219:             numero=3;
220:             dat_comp=1;
221:         }
222:         if ((numero==3)&&(dat_comp!=1))
223:         {
224:             if (comando_rx==1) minima=atoi32(dato_aux2);
225:             if (comando_rx==2) hora_inicio=atoi32(dato_aux2);
226:             if (comando_rx==3);
227:             if (comando_rx==4) manual=atoi32(dato_aux2);
228:             if (comando_rx==8);
229:             numero=4;
230:             dat_comp=1;
231:         }
232:         if ((numero==4)&&(dat_comp!=1))
233:         {
234:             if (comando_rx==1) maxima=atoi32(dato_aux2);
235:             if (comando_rx==2) minutos_inicio=atoi32(dato_aux2);
236:             if (comando_rx==3);
237:             if (comando_rx==4);
238:             if (comando_rx==8);
239:             numero=5;
240:             dat_comp=1;
241:         }
242:         if ((numero==5)&&(dat_comp!=1))
243:         {
244:             if (comando_rx==1);
245:             if (comando_rx==2) hora_fin=atoi32(dato_aux2);
246:             if (comando_rx==3);
247:             if (comando_rx==4);
248:             if (comando_rx==8);
249:             numero=6;
250:             dat_comp=1;
251:         }
252:         if ((numero==6)&&(dat_comp!=1))
253:         {
254:             if (comando_rx==1);
255:             if (comando_rx==2) minutos_fin=atoi32(dato_aux2);
256:             if (comando_rx==3);
257:             if (comando_rx==4);
258:             if (comando_rx==8);
259:             numero=0;
260:             dat_comp=1; fin_pal=m;
261:         }
262:     }
263: }
264: }
265:
266: void ciclo_histeresis_control_tiempo()
267: {
268:     if ((valor_humedad[0]!=0)&&(valor_humedad[1]!=0))

```

```

C:\código estacion remota\codigo_estacion_remota.c
269:         humedad_ciclo_uno=((valor_humedad[0]+valor_humedad[1])/2);
270:     else if(valor_humedad[0]==0) humedad_ciclo_uno=valor_humedad[1];
271:     else if(valor_humedad[1]==0)
272:             humedad_ciclo_uno=valor_humedad[0];
273:
274:     if ((valor_humedad[2]!=0)&&(valor_humedad[3]!=0))
275:             humedad_ciclo_dos=((valor_humedad[2]+valor_humedad[3])/2);
276:     else if(valor_humedad[2]==0) humedad_ciclo_dos=valor_humedad[3];
277:             else if(valor_humedad[3]==0)
278:                     humedad_ciclo_dos=valor_humedad[2];
279: //////////////CICLO DE HISTERESIS SENSOR UNO///////////////////////////
280:
281:     if ( humedad_ciclo_uno<minima)
282:     {
283:         regando_1=1;
284:         sube_1=1;
285:     }
286:     if ( humedad_ciclo_uno>maxima)
287:     {
288:         regando_1=0;
289:         baja_1=1;
290:     }
291:     if (minima<= humedad_ciclo_uno<=maxima)
292:     {
293:         if (sube_1==1){regando_1=1; sube_1=0; }
294:         if (baja_1==1){regando_1=0; ;baja_1=0; }
295:     }
296: //////////////CICLO DE HISTERESIS SENSOR DOS///////////////////////////
297:     if (humedad_ciclo_dos<minima)
298:     {
299:         regando_2=1;
300:         sube_2=1;
301:     }
302:     if ( humedad_ciclo_dos>maxima)
303:     {
304:         regando_2=0;
305:         baja_2=1;
306:     }
307:     if (minima<= humedad_ciclo_dos<=maxima)
308:     {
309:         if (sube_2==1){regando_2=1; sube_2=0; }
310:         if (baja_2==1){regando_2=0; ;baja_2=0; }
311:     }
312:     if (minima==maxima)
313:     {
314:         regando_1=0;
315:         regando_2=0;
316:     }
317:     if ((valor_humedad[0]==0)&&(valor_humedad[1]==0)) regando_1=0;
318:     if ((valor_humedad[2]==0)&&(valor_humedad[3]==0)) regando_2=0;
319: ////////////////////////////////CONTRO POR TIEMPO///////////////////////
320:     if (((hora>=0)&&(hora<=23)&&(minutos>=0)&&(minutos<=60)&&
321:             (hora_inicio>=0)&&(hora_inicio<=23)&&
322:             (minutos_inicio>=0)&&(minutos_inicio<=60)&&
323:             (hora_fin>=0)&&(hora_fin<=23)&&(minutos_fin>=0)&&
324:             (minutos_fin<=60)))
325:     {
326:         if ((hora==hora_inicio)&&(minutos==minutos_inicio)) regando_t=1;
327:         if ((hora==hora_fin)&&(minutos==minutos_fin)) regando_t=0;
328:     }
329: ///////////////////////////////////////////////////////////////////////////////////
330: }
331: //#task(rate = 250ms)
332: //void maquina_de_estado_display();
333: #task(rate = 5ms)
334: void maquina_de_estado_de_aplicacion();
335: #task(rate = 5ms)

```

```

C:\código estacion remota\codigo_estacion_remota.c
336: void maquina_de_estado_de_comunicacion();
337: void main()
338: {
339:     enable_interrupts(INT_RDA);
340:     enable_interrupts(GLOBAL);
341:     desactivar_rele(PIN_C1);
342:     desactivar_rele(PIN_C2);
343:     setup_counters(RTCC_INTERNAL, RTCC_DIV_32);
344:     delay_us(10);
345:     rtos_run();
346: }
347: void maquina_de_estado_de_comunicacion()
348: {
349:     switch (estado_siguiente){
350:         case STATE_INI:
351:             delay_ms(150);
352:             UTRDIS = 1;
353:             adc_init();
354:             //lcd_init();
355:             dato_rx[0]='1';
356:             dato_rx[1]='1';
357:             dato_rx[2]='1';
358:             dato_rx[3]='1';
359:             dato_rx[4]='1';
360:             dato_rx[5]='1';
361:             dato_rx[6]='1';
362:             dato_rx[7]='1';
363:             AN_0=10; AN_1=20; AN_2=30; AN_3=40;
364:             AN0_V=0; AN1_V=0; AN2_V=0; AN3_V=0;
365:             hum_i_1=0; hum_i_2=0; hum_i_3=0; hum_i_4=0;
366:             hum_f_1=0; hum_f_2=0; hum_f_3=0; hum_f_4=0;
367:             regando_1=0; regando_2=0; regando_3=0; regando_4=0; regando_t=0;
368:             sube_1=0; sube_2=0; sube_3=0; sube_4=0;
369:             baja_1=0; baja_2=0; baja_3=0; baja_4=0;
370:             item=0;
371:             n_menus =4;
372:             estado_siguiente=STATE_ACTIVO;
373:             salida[0]=0; salida[1]=0; salida[2]=0; salida[3]=0;
374:             minima=0; maxima=0;
375:             valor_humedad[0]=0;
376:             valor_humedad[1]=0;
377:             valor_humedad[2]=0;
378:             valor_humedad[3]=0;
379:             break;
380:         case STATE_SEND:
381:             dato_completo=0;
382:             envio_dato();
383:             while(dato_completo!=1)
384:             {}
385:             deco_tramaRx();
386:             if (comando_rx==3) comando_tx=1;
387:             if (comando_rx==1) comando_tx=1;
388:             if (comando_rx==8) comando_tx=2;
389:             dato_completo=0;
390:             estado_siguiente=STATE_ACTIVO;
391:             break;
392:         case STATE_ACTIVO:
393:             leer_ADC0_ADC1_ADC2_ADC3();
394:             convertir_ADC0_ADC1_ADC2_ADC3_a_tension();
395:             convertir_ADC0_ADC1_ADC2_ADC3_a_porcentaje();
396:             valor_humedad[0] = AN_0_porcentaje;
397:             valor_humedad[1] = AN_1_porcentaje;
398:             valor_humedad[2] = AN_2_porcentaje;
399:             valor_humedad[3] = AN_3_porcentaje;
400:             if ((dato_rx[0]=='0') && (dato_rx[1]=='0') && (dato_rx[2]=='0') &&
401:                 (dato_rx[3]=='0') && (dato_rx[4]=='0') && (dato_rx[5]=='0') &&
402:                 (dato_rx[6]=='0') && (dato_rx[7]=='1') && (dato_completo==1) )

```

Librería utilizada para el conversor analógico digital en la estación remota para el control de los sensores:

```
C:\conversor analogico, digital\adc.c
1: int16 AN_0,AN_1,AN_2,AN_3;
2: float AN0_V,AN1_V,AN2_V,AN3_V;
3: int AN_0_porcentaje,AN_1_porcentaje,AN_2_porcentaje,AN_3_porcentaje;
4: adc_init()
5: {
6:     setup_adc_ports(AN0_TO_AN4|VSS_VREF);
7:     setup_adc(ADC_CLOCK_INTERNAL);
8:     enable_interrupts(global);
9: }
10: void leer_ADC0_ADC1_ADC2_ADC3()
11: {
12:     set_adc_channel(0);
13:     delay_ms(2);
14:     AN_0=read_adc();
15:
16:     set_adc_channel(1);
17:     delay_ms(2);
18:     AN_1=read_adc();
19:
20:     set_adc_channel(2);
21:     delay_ms(2);
22:     AN_2=read_adc();
23:
24:     set_adc_channel(4);
25:     delay_ms(2);
26:     AN_3=read_adc();
27: }
28: void convertir_ADC0_ADC1_ADC2_ADC3_a_tension()
29: {
30:     AN0_V=2.5*AN_0 /1023.0;      //conversion a tension
31:     AN1_V=2.5*AN_1 /1023.0;      //conversion a tension
32:     AN2_V=2.5*AN_2 /1023.0;      //conversion a tension
33:     AN3_V=2.5*AN_3 /1023.0;      //conversion a tension
34: }
35: void convertir_ADC0_ADC1_ADC2_ADC3_a_porcentaje()
36: {
37:     AN_0_porcentaje=(int)((AN0_V*100)/0.58);
38:     AN_1_porcentaje=(int)((AN1_V*100)/0.58);
39:     AN_2_porcentaje=(int)((AN2_V*100)/0.58);
40:     AN_3_porcentaje=(int)((AN3_V*100)/0.58);
41: }
```

Librería utilizada para comandar el display de LCD en la estación remota:

```
C:\display nodo remoto\LCD_nuevo.c
1: /////////////////////////////////////////////////////////////////////
2: /////////////////////////////////////////////////////////////////// LCD.C
3: /////////////////////////////////////////////////////////////////// Driver for common LCD modules
4: ///////////////////////////////////////////////////////////////////
5: ////////////////////////////////////////////////////////////////// lcd_init() Must be called before any other function.
6: ///////////////////////////////////////////////////////////////////
7: ////////////////////////////////////////////////////////////////// lcd_putc(c) Will display c on the next position of the LCD.
8: ////////////////////////////////////////////////////////////////// \a Set cursor position to upper left
9: ////////////////////////////////////////////////////////////////// \f Clear display, set cursor to upper left
10: ////////////////////////////////////////////////////////////////// \n Go to start of second line
11: ////////////////////////////////////////////////////////////////// \b Move back one position
12: ////////////////////////////////////////////////////////////////// If LCD_EXTENDED_NEWLINE is defined, the \n character
13: ////////////////////////////////////////////////////////////////// will erase all remaining characters on the current
14: ////////////////////////////////////////////////////////////////// line, and move the cursor to the beginning of the next
15: ////////////////////////////////////////////////////////////////// line.
16: ////////////////////////////////////////////////////////////////// If LCD_EXTENDED_NEWLINE is defined, the \r character
17: ////////////////////////////////////////////////////////////////// will move the cursor to the start of the current
18: ////////////////////////////////////////////////////////////////// line.
19: //////////////////////////////////////////////////////////////////
20: ////////////////////////////////////////////////////////////////// lcd_gotoxy(x,y) Set write position on LCD (upper left is 1,1)
21: //////////////////////////////////////////////////////////////////
22: ////////////////////////////////////////////////////////////////// lcd_getc(x,y) Returns character at position x,y on LCD
23: //////////////////////////////////////////////////////////////////
24: ////////////////////////////////////////////////////////////////// CONFIGURATION
25: ////////////////////////////////////////////////////////////////// The LCD can be configured in one of two ways: a.) port access or
26: ////////////////////////////////////////////////////////////////// b.) pin access. Port access requires the entire 7 bit interface
27: ////////////////////////////////////////////////////////////////// connected to one GPIO port, and the data bits (D4:D7 of the LCD)
28: ////////////////////////////////////////////////////////////////// connected to sequential pins on the GPIO. Pin access
29: ////////////////////////////////////////////////////////////////// has no requirements, all 7 bits of the control interface can
30: ////////////////////////////////////////////////////////////////// can be connected to any GPIO using several ports.
31: //////////////////////////////////////////////////////////////////
32: ////////////////////////////////////////////////////////////////// To use port access, #define LCD_DATA_PORT to the SFR location of
33: ////////////////////////////////////////////////////////////////// of the GPIO port that holds the interface, -AND- edit LCD_PIN_MAP
34: ////////////////////////////////////////////////////////////////// of this file to configure the pin order. If you are using a
35: ////////////////////////////////////////////////////////////////// baseline PIC (PCB), then LCD_OUTPUT_MAP and LCD_INPUT_MAP also must
36: ////////////////////////////////////////////////////////////////// be defined.
37: //////////////////////////////////////////////////////////////////
38: ////////////////////////////////////////////////////////////////// Example of port access:
39: ////////////////////////////////////////////////////////////////// #define LCD_DATA_PORT getenv("SFR:PORTD")
40: //////////////////////////////////////////////////////////////////
41: ////////////////////////////////////////////////////////////////// To use pin access, the following pins must be defined:
42: ////////////////////////////////////////////////////////////////// LCD_ENABLE_PIN
43: ////////////////////////////////////////////////////////////////// LCD_RS_PIN
44: ////////////////////////////////////////////////////////////////// LCD_RW_PIN
45: ////////////////////////////////////////////////////////////////// LCD_DATA4
46: ////////////////////////////////////////////////////////////////// LCD_DATA5
47: ////////////////////////////////////////////////////////////////// LCD_DATA6
48: ////////////////////////////////////////////////////////////////// LCD_DATA7
49: //////////////////////////////////////////////////////////////////
50: ////////////////////////////////////////////////////////////////// Example of pin access:
51: ////////////////////////////////////////////////////////////////// #define LCD_ENABLE_PIN PIN_E0
52: ////////////////////////////////////////////////////////////////// #define LCD_RS_PIN PIN_E1
53: ////////////////////////////////////////////////////////////////// #define LCD_RW_PIN PIN_E2
54: ////////////////////////////////////////////////////////////////// #define LCD_DATA4 PIN_D4
55: ////////////////////////////////////////////////////////////////// #define LCD_DATA5 PIN_D5
56: ////////////////////////////////////////////////////////////////// #define LCD_DATA6 PIN_D6
57: ////////////////////////////////////////////////////////////////// #define LCD_DATA7 PIN_D7
58: //////////////////////////////////////////////////////////////////
59: ////////////////////////////////////////////////////////////////// (C) Copyright 1996,2010 Custom Computer Services
60: ////////////////////////////////////////////////////////////////// This source code may only be used by licensed users of the CCS C
61: ////////////////////////////////////////////////////////////////// compiler. This source code may only be distributed to other
62: ////////////////////////////////////////////////////////////////// licensed users of the CCS C compiler. No other use, reproduction
63: ////////////////////////////////////////////////////////////////// or distribution is permitted without written permission.
64: ////////////////////////////////////////////////////////////////// Derivative programs created using this software in object code
65: ////////////////////////////////////////////////////////////////// form are not restricted in any way.
66: //////////////////////////////////////////////////////////////////
67: //////////////////////////////////////////////////////////////////
```

```

C:\display nodo remoto\LCD_nuevo.c
68:
69: // define the pinout.
70: // only required if port access is being used.
71: typedef struct
72: {                                     // This structure is overlayed
73:     BOOLEAN enable;                  // on to an I/O port to gain
74:     BOOLEAN rs;                     // access to the LCD pins.
75:     BOOLEAN rw;                    // The bits are allocated from
76:     BOOLEAN unused;                // low order up.  ENABLE will
77:     int data : 4;                 // be LSB pin of that port.
78:     #if defined(__PCD__)
79:     int reserved: 8;              // The port used will be LCD_DATA_PORT.
80:     #endiff
81: } LCD_PIN_MAP;
82:
83: // this is to improve compatibility with previous LCD drivers that accepted
84: // a define labeled 'use_portb_lcd' that configured the LCD onto port B.
85: #if ((defined(use_portb_lcd)) && (use_portb_lcd==TRUE))
86: #define LCD_DATA_PORT getenv("SFR:PORTB")
87: #endiff
88:
89: #if defined(__PCB__)
90:     // these definitions only need to be modified for baseline PICs.
91:     // all other PICs use LCD_PIN_MAP or individual LCD_xxx pin definitions.
92:     /*                                         EN, RS,      RW,      UNUSED,   DATA */
93:     const LCD_PIN_MAP LCD_OUTPUT_MAP = {0, 0, 0, 0, 0};
94:     const LCD_PIN_MAP LCD_INPUT_MAP = {0, 0, 0, 0, 0xF};
95: #endiff
96:
97: ////////////////////////////// END CONFIGURATION //////////////////////////////
98:
99: #ifndef LCD_ENABLE_PIN
100:    #define lcd_output_enable(x) lcdlat.enable=x
101:    #define lcd_enable_tris()    lcdtris.enable=0
102: #else
103:    #define lcd_output_enable(x) output_bit(LCD_ENABLE_PIN, x)
104:    #define lcd_enable_tris()    output_drive(LCD_ENABLE_PIN)
105: #endiff
106:
107: #ifndef LCD_RS_PIN
108:    #define lcd_output_rs(x) lcdlat.rs=x
109:    #define lcd_rs_tris()    lcdtris.rs=0
110: #else
111:    #define lcd_output_rs(x) output_bit(LCD_RS_PIN, x)
112:    #define lcd_rs_tris()    output_drive(LCD_RS_PIN)
113: #endiff
114:
115: #ifndef LCD_RW_PIN
116:    #define lcd_output_rw(x) lcdlat.rw=x
117:    #define lcd_rw_tris()    lcdtris.rw=0
118: #else
119:    #define lcd_output_rw(x) output_bit(LCD_RW_PIN, x)
120:    #define lcd_rw_tris()    output_drive(LCD_RW_PIN)
121: #endiff
122:
123: // original version of this library incorrectly labeled LCD_DATA0 as LCD_DATA4,
124: // LCD_DATA1 as LCD_DATA5, and so on.  this block of code makes the driver
125: // compatible with any code written for the original library
126: #if (defined(LCD_DATA0) && defined(LCD_DATA1) && defined(LCD_DATA2) &&
127:      defined(LCD_DATA3) && !defined(LCD_DATA4) && !defined(LCD_DATA5) &&
128:      !defined(LCD_DATA6) && !defined(LCD_DATA7))
129:     #define LCD_DATA4    LCD_DATA0
130:     #define LCD_DATA5    LCD_DATA1
131:     #define LCD_DATA6    LCD_DATA2
132:     #define LCD_DATA7    LCD_DATA3
133: #endiff
134:
```

```

C:\display nodo remoto\LCD_nuevo.c
135: #ifndef LCD_DATA4
136: #ifndef LCD_DATA_PORT
137:     #if defined(__PCB__)
138:         #define LCD_DATA_PORT      0x06      //portb
139:         #define set_tris_lcd(x)    set_tris_b(x)
140:     #else
141:         #if defined(PIN_D0)
142:             #define LCD_DATA_PORT      getenv("SFR:PORTD")      //portd
143:         #else
144:             #define LCD_DATA_PORT      getenv("SFR:PORTB")      //portb
145:         #endif
146:     #endif
147: #endif
148:
149: #if defined(__PCB__)
150:     LCD_PIN_MAP lcd, lcdlat;
151:     #byte lcd = LCD_DATA_PORT
152:     #byte lcdlat = LCD_DATA_PORT
153: #elif defined(__PCM__)
154:     LCD_PIN_MAP lcd, lcdlat, lcdtris;
155:     #byte lcd = LCD_DATA_PORT
156:     #byte lcdlat = LCD_DATA_PORT
157:     #byte lcdtris = LCD_DATA_PORT+0x80
158: #elif defined(__PCH__)
159:     LCD_PIN_MAP lcd, lcdlat, lcdtris;
160:     #byte lcd = LCD_DATA_PORT
161:     #byte lcdlat = LCD_DATA_PORT+9
162:     #byte lcdtris = LCD_DATA_PORT+0x12
163: #elif defined(__PCD__)
164:     LCD_PIN_MAP lcd, lcdlat, lcdtris;
165:     #word lcd = LCD_DATA_PORT
166:     #word lcdlat = LCD_DATA_PORT+2
167:     #word lcdtris = LCD_DATA_PORT-0x02
168: #endif
169: #endif //LCD_DATA4 not defined
170:
171: #ifndef LCD_TYPE
172:     #define LCD_TYPE 2           // 0=5x7, 1=5x10, 2=2 lines
173: #endif
174:
175: #ifndef LCD_LINE_TWO
176:     #define LCD_LINE_TWO 0x40    // LCD RAM address for the second line
177: #endif
178:
179: #ifndef LCD_LINE_LENGTH
180:     #define LCD_LINE_LENGTH 20
181: #endif
182:
183: BYTE const LCD_INIT_STRING[4] = {0x20 | (LCD_TYPE << 2), 0xc, 1, 6};
184:                                         // These bytes need to be sent to the LCD
185:                                         // to start it up.
186: BYTE lcd_read_nibble(void);
187: BYTE lcd_read_byte(void)
188: {
189:     BYTE low,high;
190:     #if defined(__PCB__)
191:         set_tris_lcd(LCD_INPUT_MAP);
192:     #else
193:         #if (defined(LCD_DATA4) && defined(LCD_DATA5) &&
194:             defined(LCD_DATA6) && defined(LCD_DATA7))
195:             output_float(LCD_DATA4);
196:             output_float(LCD_DATA5);
197:             output_float(LCD_DATA6);
198:             output_float(LCD_DATA7);
199:         #else
200:             lcdtris.data = 0xF;
201:         #endif

```

```

C:\display nodo remoto\LCD_nuevo.c
202: #endif
203:     lcd_output_rw(1);
204:     delay_cycles(1);
205:     lcd_output_enable(1);
206:     delay_cycles(1);
207:     high = lcd_read_nibble();
208:     lcd_output_enable(0);
209:     delay_cycles(1);
210:     lcd_output_enable(1);
211:     delay_us(1);
212:     low = lcd_read_nibble();
213:     lcd_output_enable(0);
214: #if defined(__PCB__)
215:     set_tris_lcd(LCD_OUTPUT_MAP);
216: #else
217: #if (defined(LCD_DATA4) && defined(LCD_DATA5) && defined(LCD_DATA6) &&
218: defined(LCD_DATA7))
219:     output_drive(LCD_DATA4);
220:     output_drive(LCD_DATA5);
221:     output_drive(LCD_DATA6);
222:     output_drive(LCD_DATA7);
223: #else
224:     lcdtris.data = 0x0;
225: #endif
226: #endif
227: return( (high<<4) | low);
228: }
229: BYTE lcd_read_nibble(void)
230: {
231: #if (defined(LCD_DATA4) && defined(LCD_DATA5) && defined(LCD_DATA6) &&
232: defined(LCD_DATA7))
233:     BYTE n = 0x00;
234: /*Read the data port */
235:     n |= input(LCD_DATA4);
236:     n |= input(LCD_DATA5) << 1;
237:     n |= input(LCD_DATA6) << 2;
238:     n |= input(LCD_DATA7) << 3;
239:     return(n);
240: #else
241:     return(lcd.data);
242: #endif
243: }
244: void lcd_send_nibble(BYTE n)
245: {
246: #if (defined(LCD_DATA4) && defined(LCD_DATA5) && defined(LCD_DATA6) &&
247: defined(LCD_DATA7))
248: /* Write to the data port */
249:     output_bit(LCD_DATA4, bit_test(n, 0));
250:     output_bit(LCD_DATA5, bit_test(n, 1));
251:     output_bit(LCD_DATA6, bit_test(n, 2));
252:     output_bit(LCD_DATA7, bit_test(n, 3));
253: #else
254:     lcdlat.data = n;
255: #endif
256:     delay_cycles(1);
257:     lcd_output_enable(1);
258:     delay_us(2);
259:     lcd_output_enable(0);
260: }
261: void lcd_send_byte(BYTE address, BYTE n)
262: {
263: #if defined(__PCB__)
264:     set_tris_lcd(LCD_OUTPUT_MAP);
265: #else
266:     lcd_enable_tris();
267:     lcd_rs_tris();
268:     lcd_rw_tris();

```

```

C:\display nodo remoto\LCD_nuevo.c
269: #endif
270: lcd_output_rs(0);
271: while ( bit_test(lcd_read_byte(), 7) ) ;
272: lcd_output_rs(address);
273: delay_cycles(1);
274: lcd_output_rw(0);
275: delay_cycles(1);
276: lcd_output_enable(0);
277: lcd_send_nibble(n >> 4);
278: lcd_send_nibble(n & 0xf);
279: }
280: #if defined(LCD_EXTENDED_NEWLINE)
281: unsigned int8 g_LcdX, g_LcdY;
282: #endif
283: void lcd_init(void)
284: {
285: BYTE i;
286: #if defined(__PCB__)
287: set_tris_lcd(LCD_OUTPUT_MAP);
288: #else
289: #if (defined(LCD_DATA4) && defined(LCD_DATA5) && defined(LCD_DATA6) &&
290: defined(LCD_DATA7))
291: output_drive(LCD_DATA4);
292: output_drive(LCD_DATA5);
293: output_drive(LCD_DATA6);
294: output_drive(LCD_DATA7);
295: #else
296: lcdtris.data = 0x0;
297: #endif
298: lcd_enable_tris();
299: lcd_rs_tris();
300: lcd_rw_tris();
301: #endif
302: lcd_output_rs(0);
303: lcd_output_rw(0);
304: lcd_output_enable(0);
305: delay_ms(15);
306: for(i=0;i<=3;++i)
307: {
308:     lcd_send_nibble(3);
309:     delay_ms(5);
310: }
311: lcd_send_nibble(2);
312: delay_ms(5);
313: for(i=0;i<=3;++i)
314:     lcd_send_byte(0,LCD_INIT_STRING[i]);
315: #if defined(LCD_EXTENDED_NEWLINE)
316: g_LcdX = 0;
317: g_LcdY = 0;
318: #endif
319: }
320: void lcd_gotoxy(BYTE x, BYTE y)
321: {
322: BYTE address;
323: if(y!=1)
324:     address=LCD_LINE_TWO;
325: else
326:     address=0;
327: address+=x-1;
328: lcd_send_byte(0,0x80|address);
329: #if defined(LCD_EXTENDED_NEWLINE)
330: g_LcdX = x - 1;
331: g_LcdY = y - 1;
332: #endif
333: }
334: void lcd_putc(char c)
335: {

```

```

C:\display nodo remoto\LCD_nuevo.c
336:     switch (c)
337:     {
338:         case '\a' : lcd_gotoxy(1,1); break;
339:
340:         case '\f' : lcd_send_byte(0,1);
341:                     delay_ms(2);
342:                     #if defined(LCD_EXTENDED_NEWLINE)
343:                         g_LcdX = 0;
344:                         g_LcdY = 0;
345:                     #endif
346:                     break;
347:                     #if defined(LCD_EXTENDED_NEWLINE)
348:                     case '\r' : lcd_gotoxy(1, g_LcdY+1);break;
349:                     case '\n' :
350:                         while (g_LcdX++ < LCD_LINE_LENGTH)
351:                         {
352:                             lcd_send_byte(1, ' ');
353:                         }
354:                         lcd_gotoxy(1, g_LcdY+2);
355:                         break;
356:                     #else
357:                     case '\n' : lcd_gotoxy(1,2);break;
358:                     #endif
359:                     case '\b' : lcd_send_byte(0,0x10);break;
360:                     #if defined(LCD_EXTENDED_NEWLINE)
361:                         default :
362:                             if (g_LcdX < LCD_LINE_LENGTH)
363:                             {
364:                                 lcd_send_byte(1, c);
365:                                 g_LcdX++;
366:                             }
367:                             break;
368:                         #else
369:                         default : lcd_send_byte(1,c);      break;
370:                         #endif
371:                     }
372:     }
373: char lcd_getc(BYTE x, BYTE y)
374: {
375:     char value;
376:     lcd_gotoxy(x,y);
377:     while ( bit_test(lcd_read_byte(),7) ); // wait until busy flag is low
378:     lcd_output_rs(1);
379:     value = lcd_read_byte();
380:     lcd_output_rs(0);
381:     return(value);
382: }

```

UNIVERSIDAD NACIONAL DE CATAMARCA



ANEXO VI: FOTOGRAFIAS

Diseño de un sistema de riego para optimizar el recurso hídrico

Matías Leandro Ferraro

Año: 2015

Título: Ingeniero Electrónico
Director de tesis: Sergio Hilario Gallina
Departamento: Electrónica



Fotografía 1: Estación remota



Fotografía 2: Estación remota y depósito de agua



Fotografía 3: Estación remota donde se pueden observar los conectores y la casilla metálica de protección



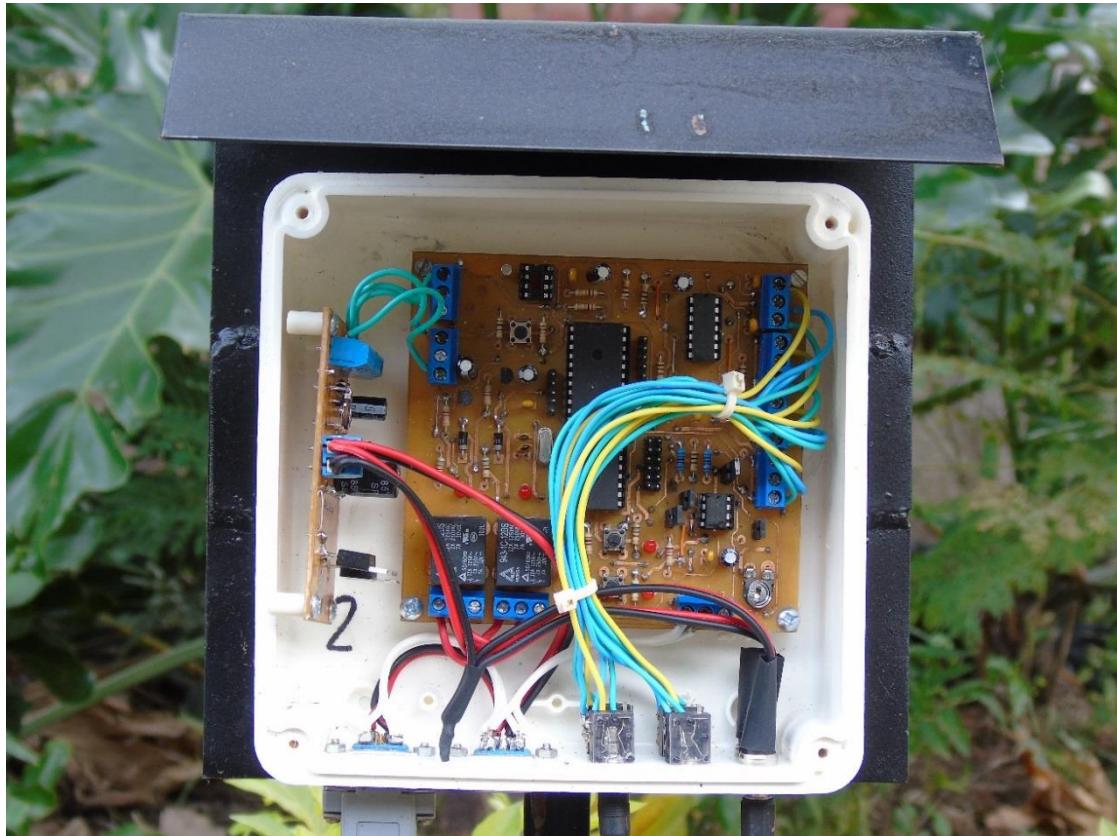
Fotografía 4: Estación remota dos



Fotografía 5: Estación remota dos y vista de la manguera de riego



Fotografía 6: Vista del frente de la estación remota



Fotografía 7: Vista interna de la estación remota



Fotografía 8: Vista de la estación concentradora

UNIVERSIDAD NACIONAL DE CATAMARCA



ANEXO VII: PREMIO GANADO POR EL SISTEMA DE RIEGO

Diseño de un sistema de riego para optimizar el recurso hídrico

Matías Leandro Ferraro

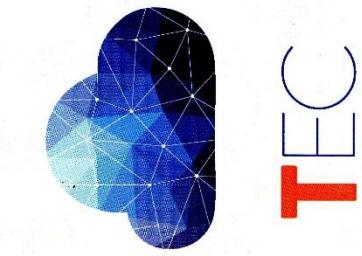
Año: 2015

Título: Ingeniero Electrónico
Director de tesis: Sergio Hilario Gallina
Departamento: Electrónica

Premio ganado por el sistema de riego



**FESTIVAL
DEL DESARROLLO
TECNOLÓGICO**



Microsoft certifica a

MATIAS LEANDRO FERRARO

como ganador provincial de **Tecnotour Argentina 2015**, la hackatón más grande del país.

José Antonio Scioli
Presidente
Desarrollo Argentino

Alejandro Anderlic
Director de Legales & Asuntos Corporativos
Microsoft Argentina