ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE TELECONTROL Y SEGURIDAD PERIMETRAL DE ESTACIÓN DE BOMBEO DE RIEGO CON INTERFAZ WEB, MEDIANTE USO DE HARDWARE Y SOFTWARE LIBRE"

INFORME DE PROYECTO INTEGRADOR

Previo a la obtención del título de:

INGENIERO EN TELEMÁTICA

DÍAZ REDROBÁN GIANPAOLO DAVID

NAVIA SANTANA FÉLIX EDUARDO

GUAYAQUIL - ECUADOR

Año: 2015

AGRADECIMIENTOS

A Dios

A nuestros padres

A nuestros hermanos

A nuestros profesores

DEDICATORIA

A Dios, quien mediante el espíritu santo me dio fortaleza para la conclusión de mi trabajo final de graduación. A mi Madre, siempre me ha apoyado, mujer valiente que me supo dar la educación, consejos y no permitió que desfallezca. A mi esposa, es mi compañera, mi motivación de vida, quien sin su ayuda no hubiera podido culminar esta meta. A todos les agradezco desde lo más profundo de mi alma.

Gianpaolo Díaz Redrobán

A Dios, quien es la base de mi vida. A mi Madre que me supo guiar y que gracias a sus consejos me formaron para la vida. A mi Padre, quien desde el cielo me dirigió en el camino correcto y me daba fuerzas para seguir luchando. A todas aquellas personas que de una u otra manera intervinieron para lograr este objetivo.

Félix Navia Santana

TRIBUNAL DE EVALUACIÓN

Ing. Marcos Millán

PROFESOR EVALUADOR

Ing. Washington Velásquez

PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

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

Gianpaolo David Díaz Redrobán

Félix Eduardo Navia Santana

RESUMEN

Las estaciones de bombeo de agua son importantes en las zonas agrícolas pero lamentablemente tienen problemas que no son reconocidos y pueden llegar a tener impacto en la naturaleza, para ayudar a resolver estas dificultades se ha diseñado un sistema de automatización usando una tarjeta de desarrollo llamada BeagleBone Black, la cual es considerada una microcomputadora por sus características. Adicionalmente se desarrolló una interfaz web para el acceso remoto y finalmente se añadió seguridad perimetral a la estación para precautelar la integridad del sistema.

El sistema de automatización tiene el deber de controlar los niveles de agua en el reservorio y en la albarrada. Tendrá que detener la bomba cuando el nivel de agua en el reservorio haya sobrepasado su límite, encender la bomba cuando el nivel de agua sea mínimo en el reservorio y detener la bomba cuando en la albarrada no haya agua, así se evita el desperdicio de agua y daños en el sistema. El sistema funciona en modo automático o manual.

El sistema está ubicado en una zona alejada por lo que debe brindársele seguridad, para esto se ha instalado un cerramiento en su perímetro con su respectiva cerca eléctrica, un sensor de presencia que al momento que detecte algún movimiento dará un tiempo adecuado para que el usuario se autentique caso contrario se activará una alarma y enviará una alerta al administrador. Consta con una cámara IP que permite observar que está pasando dentro de la estación en cualquier momento.

ÍNDICE GENERAL

CAPÍTULO 1

1 ANTECEDENTES

1.1 Descripción del problema

El personal de los Campos Experimentales y de Investigación Agropecuaria - ESPOL (CENAE) para encender y apagar la bomba caminan entre la maleza unos 30 minutos de ida e igual tiempo para el regreso, esto puede ser peligroso ya que cuando se dirigen hacia la bomba pueden tener accidentes de consideración y al encontrarse en un lugar aislado no van a poder recibir la ayuda adecuada.

El camino tiene árboles de grandes dimensiones, por esto la señal celular es muy pobre y en caso de algún accidente lo más probable sería que no se pueda comunicar con alguien que lo pueda socorrer.

En ocasiones existe descuido del personal del CENAE, mientras se está realizando el bombeo del agua se llena el reservorio y se reboza el líquido causando una excesiva pérdida de agua hasta caminar a la bomba y apagarla.

La bomba queda en un lugar apartado y solitario esto hace que sea un blanco perfecto para la delincuencia, tiempo atrás ya ha sido sustraída dejando sin funcionamiento por un periodo de tiempo prolongado al CENAE.

1.2 Justificación

Se propone minimizar los riesgos monitoreando y controlando el correcto desempeño de los equipos para el buen funcionamiento de los procesos dentro de una estación de bombeo.

La automatización se ha convertido en una de las principales soluciones a un sinnúmero de problemas en algunas empresas. Desde labores de mantenimiento hasta aumento en la productividad y confiabilidad.

Al implementar un confiable sistema de automatización se permitirá que mediante sensores y una tarjeta de desarrollo se mantenga la estación de bombeo en un estado óptimo y garantizando que los procesos se ejecuten de manera confiable.

Además de esto, el uso de redes de comunicación permitirá que se conozca en tiempo real el estado de variables determinantes tanto de la bomba como de la seguridad donde se centraliza el sistema.

1.3 Objetivo General

Diseñar e implementar un prototipo de telecontrol y seguridad perimetral para una estación de bombeo de riego con interfaz web mediante el uso de hardware y software libre.

1.4 Objetivos Específicos

- Permitir el funcionamiento del sistema en modo manual o automático
- Controlar y vigilar en tiempo real la estación de bombeo
- Brindar seguridad perimetral a la estación de bombeo
- Simular un enlace radial que permita la comunicación entre la estación de bombeo y CENAE
- Desarrollar una aplicación web para el control remoto del sistema

1.5 Metodología

El sistema se implementará utilizando hardware y software libre, debido a su bajo costo y a la compatibilidad con actuadores y sensores fácilmente accesibles en el mercado. Se utilizará la tarjeta BeagleBone Black, ya que cumple con las características necesarias para implementación del sistema.

Se simulará un radioenlace para tener comunicación entre los puntos y poder telecontrolar el sistema.

En la implementación se emplearán diversos tipos de sensores que proporcionarán los datos necesarios para el análisis y seguimiento de variables al momento del uso del sistema.

La seguridad electrónica es indispensable en la implementación del sistema, por ello se han escogido algunos elementos para poder garantizar el correcto funcionamiento de toda la estación de bombeo.

1.6 Resultados Esperados

Al finalizar este proyecto se espera haber logrado un sistema automatizado capaz de ser manipulable remotamente, a su vez tener un sistema seguro que permita

mantener la integridad de los equipos que conforman la estación y evite el daño del sistema por alguna avería o robo.

1.7 Observaciones

Para hacer el radioenlace se tendrían que instalar torres donde se colocarán los equipos de transmisión.

Donde se ubicará la bomba se deberá construir un cerramiento para poder instalar una cerca eléctrica y así brindar más seguridad para el sistema.

CAPÍTULO 2

2 MARCO TEÓRICO

2.1 Telemetría y Telecontrol

La telemetría es una técnica automatizada de las comunicaciones con la ayuda de que las mediciones y selección de datos se realizan en lugares remotos y de transmisión para la vigilancia. Esta técnica utiliza comúnmente transmisión inalámbrica. Los usos más importantes de la telemetría son la recopilación de datos, supervisión de plantas de generación de energía y hacer el seguimiento de vuelos tripulados y no tripulados. [1]

La información que es enviada desde un sistema que se está controlando hasta el receptor, frecuentemente es realizada mediante comunicación inalámbrica, sin embargo puede realizarse por otros medios, como ejemplos están los teléfonos, las redes de computadoras, los enlaces de fibra óptica, etc. Estos sistemas de telemetría captan las instrucciones y la información requerida para poder trabajar mediante una estación de control. [2]

El telecontrol es el envío de indicaciones a distancia mediante un enlace de transmisión, este sea a través de cables, radio, manipulando la información enviada para controlar los sistemas remotos, estos pueden o no estar directamente conectados al equipo desde el cual se mandan las órdenes. [3]

La palabra viene de dos raíces tele = distancia y control = controlar. Los sistemas que necesitan medición remota y reporte de información de interés para el diseñador del sistema o el operador deben usar la contrapartida del telecontrol. El telecontrol se puede llevar a cabo en tiempo real o no dependiendo de las circunstancias. [3]

2.1.1 Funciones

Cuando se habla de telemetría y telecontrol son sistemas que trabajan conjuntamente, son complementarios, estos sistemas están conformados por diferentes equipos, uno de ellos es un transductor, su principal función es el convertir las diferentes mediciones recibidas como una temperatura, vibraciones de una señal, el estado de una máquina, entre otras. Adicional a esto se necesita el medio por donde se transmitirá la información, puede ser aire, cable, fibra. Son muy importantes los dispositivos que procesan la señal y por último los equipos que realizan el procesamiento de los datos,

el trabajo conjunto de estos elementos hacen posible el buen funcionamiento de un sistema de telecontrol y telemetría. [1]

2.1.2 Aplicaciones

Estos sistemas actualmente son muy utilizados en diferentes campos de la ciencia, para poner algunos ejemplos: en la medicina, los aviones, las empresas que distribuyen electricidad, las naves aeroespaciales y en casi todas las actividades diarias, en la Figura 2.1 se muestra el funcionamiento de esta tecnología. [4]

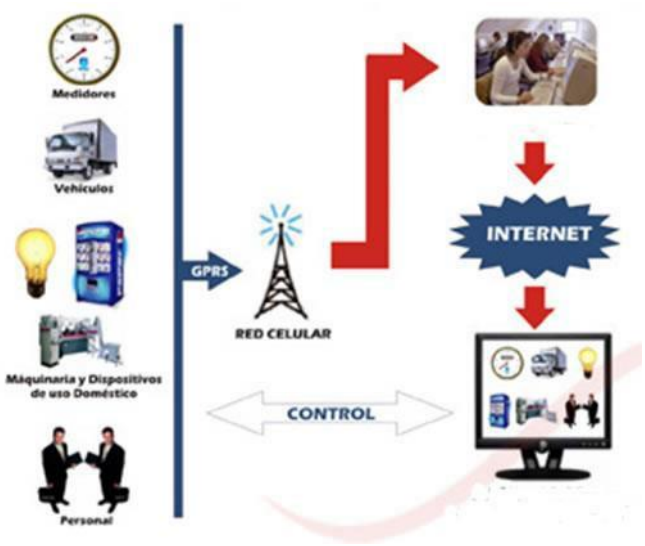

Figura 2.1: Aplicación y telecontrol [5]

Otras aplicaciones son las de:

- Sistema de telemedición y telecontrol
- Sistema de monitoreo remoto
- Automatización de bombeo de agua
- Automatización de procesos

2.2 BeagleBone Black

Es una tarjeta de desarrollo de bajo costo, gracias a la comunidad de desarrolladores y aficionados es posible encontrar información. Arranca Linux en menos de 10 segundos y permite comenzar a trabajar cualquier desarrollo en menos de 3 minutos sólo con el cable USB. [6]

2.2.1 Características

A continuación se muestra en la Tabla 1 las características de la tarjeta BeagleBone Black.

Procesador	1 GHz Sitara AM3359AZCZ100	
Motor Gráfico	SGX530 3D, 20M Polygons/S	
Memoria SDRAM	512 MB DDR3L 400 MHz	
Flash	2GB,8bit Embedded MMC	
PMIC	TPS65217C PMIC regulator and one additional LDO.	
Alimentación	miniUSB o Jack DC	5VC Via header de expasión
PCB	3.4” x 2.1”, 6 capas	
Indicadores	1 Poder, 2 Ethernet, 4 Leds para control del usuario	
Puerto Serial	UART0 via pin 6, 3.3V TTL.	
Ehternet	10/100, RJ45	
SD/Conector MMC	microSD, 3.3V	
Entradas Usuario	Botón Reset, Botón Boot, Botón Poder	
Salida de Video	16b HDMI, 1280×1024 (MAX)1024×768, 1280×720, 1440x900w/Soporte EDID	
Audio	Via HDMI, Stereo	
Conectores de Expasión	Poder 5V, 3.3V VDD_ADC(1.8V)3.3V I/OMcASP0, SPI1, I2C, GPIO(69), LCD, GPMC, MMC1, MMC2, 7 AIN(1.8 MAX) , 4 Timers, 4 Puertos Seriales, CAN0, EHRPWM(0.2) , Interrupciones XDMA, Botón de Poder	
Consumo	210-460mA @ 5V Dependiendo de la actividad y velocidad del procesador.	
Peso	39.68 gramos	

Tabla 1: Características de BeagleBone Black [7]

2.2.2 Hardware

La BeagleBone Black es considerada una microcomputadora que se puede conectar fácilmente a un televisor o monitor por el puerto HDMI. La tarjeta puede manipularse lógicamente usando un mouse y teclado, otra característica es su acceso al internet y posee pines GPIO que se comunicarán con sensores, placa y circuitos electrónicos. En la Figura 2.2 se muestra el diseño físico de la tarjeta de desarrollo [8]

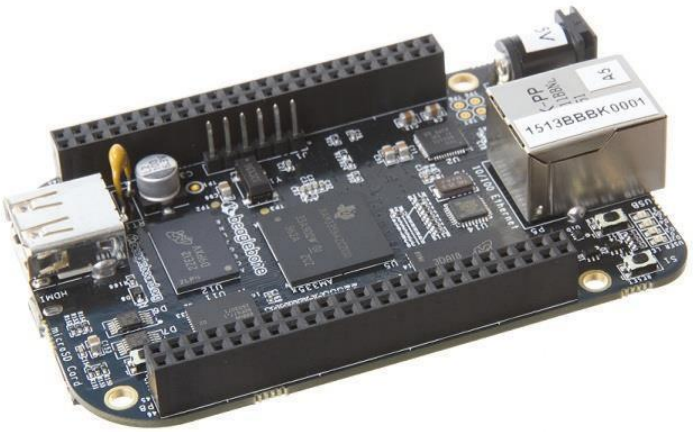

Figura 2.2: Hardware BeagleBone Black [9]

2.2.3 Software

En la tarjeta viene implementada la distribución Linux Angstrom, sin embargo es posible instalar otras distribuciones, como:

- Ubuntu
- Fedora
- Android
- Debian
- Gentoo
- Erlang

Las aplicaciones hacen llamadas a las librerías de tiempo de ejecución que son de código abierto, y estas librerías hacen llamadas a unos drivers de código abierto en el kernel de Linux.

2.3 Sevidor Web

El servidor web es una aplicación informática que resuelve del lado del servidor, sus conexiones pueden ser sincrónicas o asincrónicas con el cliente, siempre genera una respuesta de cualquier aplicación o lenguaje del lado del cliente. Cuando el cliente recibe algún tipo de código generalmente es ejecutado y compilado por un servidor web, cuando se transmiten datos siempre se utiliza algún tipo de protocolo, habitualmente se utiliza el protocolo HTTP para estas clases de comunicaciones. En el modelo OSI esta pertenece a la capa de aplicación. [10]

2.3.1 Nginx

Es un servidor web/proxy completamente inverso, su características más relevante son ser muy liviano y su gran velocidad. Puede trabajar como un proxy para correo electrónico del tipo IMAP/POP3, otra de sus fortalezas es ser software libre listo para trabajar sobre cualquier sistema operativo. [11]

2.3.2 Funcionalidad

Un servidor web se mantiene a la espera de peticiones de ejecución que le hará un cliente o un usuario de la red. El servidor web se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo de acuerdo a los comandos solicitados. [12]

2.3.3 Tipo de comunicación

La comunicación hacia un servidor web debe ser mediante una red, dependiendo donde se encuentre ubicado el servidor esta puede ser una red local (LAN) o el internet, como lo muestra la Figura 2.3. Para acceder al servidor debemos digitar una dirección IP, la cual ya fue configurada con anterioridad en el servidor.

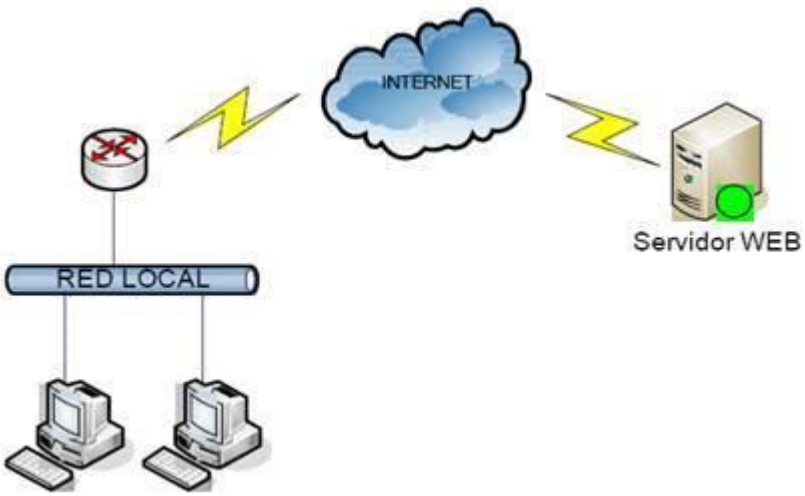

Figura 2.3: Comunicación a servidor web [13]

2.4 Sistemas de Seguridad

Los sistemas de seguridad son la unión de varios dispositivos, instalaciones y equipos que se pueden brindar a personas o los bienes materiales que de alguna manera necesiten ser protegidos de ataques, incendios, agresiones, entre otras. [14]

Cuando suceda algún evento el sistema automáticamente lo detectará, luego lo derivará a su respectiva alarma para finalmente comenzar con acciones guiadas a reducir o eliminar los efectos del siniestro, todo esto alertando mediante mecanismos de extinción, comunicación de radio, cámaras de seguridad, entre otras.

Estos sistemas se distinguen porque pueden servir para diferentes casos o necesidades, por ejemplo la seguridad o métodos que se le brinda a una empresa robusta no será la misma que se le brinde a un alto ejecutivo. Estos casos también dependen del dinero que se esté planeado invertir en seguridad. En la Figura 2.4 se muestran ejemplos de dispositivos de seguridad.

Figura 2.4: Sistemas de seguridad [15]

2.4.1 Seguridad Electrónica

Desde mucho tiempo atrás la seguridad ha sido para los seres humanos una de las mayores preocupaciones. Actualmente gracias a las redes de comunicación y al internet se ha desarrollado una nueva seguridad que tiene que ver con este nuevo mundo virtual. [16]

2.4.2 Características de un Sistema de Seguridad

Los sistemas de seguridad electrónica tiene en común algunas cualidades, que se pueden enumerar en tan sólo cinco ítems:

- Integridad
- Confidencialidad
- Disponibilidad
- Confiabilidad
- Control de Acceso

2.4.3 Arquitectura de un Sistema de Seguridad

La seguridad puede aplicarse a un sin número de escenarios, para alcanzar una óptima seguridad integral es necesario utilizar diferentes medios. Su clasificación según los medios se detalla a continuación:

- Recursos: Están compuestos por la seguridad pública o privada debidamente capacitados.
- Medios Técnicos: Son los equipos que fueron diseñados y construidos para mantener la seguridad de personas o lugares específicos, también llamados pasivos, estos son cerramientos o construcciones, y los activos son los equipos electrónicos.
- Medios Organizativos: Son todos los instrumentos que se utilizan en la coordinación de la manipulación de los recursos, como ejemplo la planificación, normas de seguridad, entre otros.

En la Figura 2.5 se pueden observar algunos componentes de la seguridad electrónica, se podrá observar una idea de los diferentes tipos de medios, con esto tendrá una visión un poco más amplia de los sistemas de seguridad. [16]

Figura 2.5: Arquitectura de sistemas de seguridad [17]

2.5 Proceso de Bombeo

El proceso de bombeo radica en tomar agua de un algún lugar principalmente pueden ser ríos, embalses o lagos y trasladarla o ubicarla hasta los lugares en donde se las van a depositar para darles un uso respectivo. La forma más usual de trasladar el agua desde un lago hasta un reservorio es por medio de bombas centrifugas las cuales obtienen su potencia de motores eléctricos.

2.5.1 Bombas de Agua

Es la máquina eléctrica que transforma energía, aplicándola para transportar el agua. Este traslado, normalmente es ascendente. Existen dos tipos de bombas: volumétricas y turbo-bombas. Todas están diseñadas con un orificio de entrada de aspiración y otro de salida de impulsión.

Cuando la bomba recibe la energía a través de un motor acoplado este sea eléctrico o gasolina, al equipo se le llama moto-bomba. El motor del equipo puede también estar separado de la bomba. Entonces hace falta un elemento que le transmita el movimiento. Puede ser una polea, un eje, entre otras. La Figura 2.6 muestra una bomba de agua eléctrica [18]

Figura 2.6: Bomba de agua [18]

2.5.2 Válvulas

Es un dispositivo que permite regular o controlar un determinado flujo de líquidos o gases. Se puede llegar a determinar que la válvula es una herramienta de control indispensable en cualquier tipo de industria. El funcionamiento y objetivo de las válvulas dependen mayormente como están diseñadas y de que materiales están construidas, estas pueden abrir o cerrar, aislar o regular, conectar o desconectar, resisten casi cualquier tipo de materiales líquidos o gases estos sean tóxicos o corrosivos. En la Figura 2.7 se muestra una válvula de acero. [19]

Figura 2.7: Válvula [27]

2.5.3 Sensores

Los sensores o captadores son unos dispositivos diseñados para obtener información de agentes externos y convertirlas en una magnitud, generalmente eléctrica, para que esta información obtenida sea capaz de cuantificar y poder manipularla. Generalmente este tipo de dispositivos se componen de algunos elementos pasivos tales como resistencias variables, LDR, PTC, NTC, y aquellos elementos que pueden variar su

magnitud o algún tipo de variable, adicional a esto están compuestos de componentes activos. [21]

Figura 2.8: Sensor PIR [22]

2.5.4 Estación de Bombeo

Las estaciones de bombeo son infraestructuras, diseñadas e implementadas para trasladar el líquido del nivel de succión o reservorio a un nivel superior. Estas estaciones son muy importantes para llevar el fluido cuando en la ubicación del esquema final no se pueda disponer de la fuerza gravitatoria.

Los ductos o cañerías como vías libres necesitan tener cierta pendiente que permita el flujo de los líquidos por gravedad, lamentablemente en terrenos planos no se tiene la pendiente necesaria, para estos casos las estaciones de bombeo surgen de manera obligatoria. [23]

La funcionalidad principal de la estación de bombeo es transportar el agua de un punto inicial hasta otro punto final, donde será utilizada para el riego a cosechas. Los componentes de una estación de bombeo son:

- Caseta de bombeo
- Cisterna de bombeo
- Equipo de bombeo
- Grupo generador de energía y fuerza motriz
- Tubería de succión e impulsión
- Interruptores de encendido y apagado

CAPÍTULO 3

3 DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO

En este capítulo se mostrarán los detalles de cómo se desarrolló la automatización del prototipo, los diferentes sensores que se utilizaron y los componente eléctricos que conforman el tablero de control.

3.1 Controlador

Se puede considerar la parte fundamental del sistema, ya que en este se encuentra la programación y la estructura del mismo, lo que brindó una solución confiable para el sistema.

3.1.1 Diseño

El controlador ha sido diseñado por módulos, y cada uno de ellos se encarga de controlar un aspecto del sistema, por ejemplo, el encendido y apagado de la bomba.

El controlador y los módulos que lo conforman han sido escritos en lenguaje Python y se ejecutan cada vez que se inicia la tarjeta BeagleBone (ver Anexo 1). A continuación en la Figura 3.1 se presenta el diagrama de bloques del controlador.

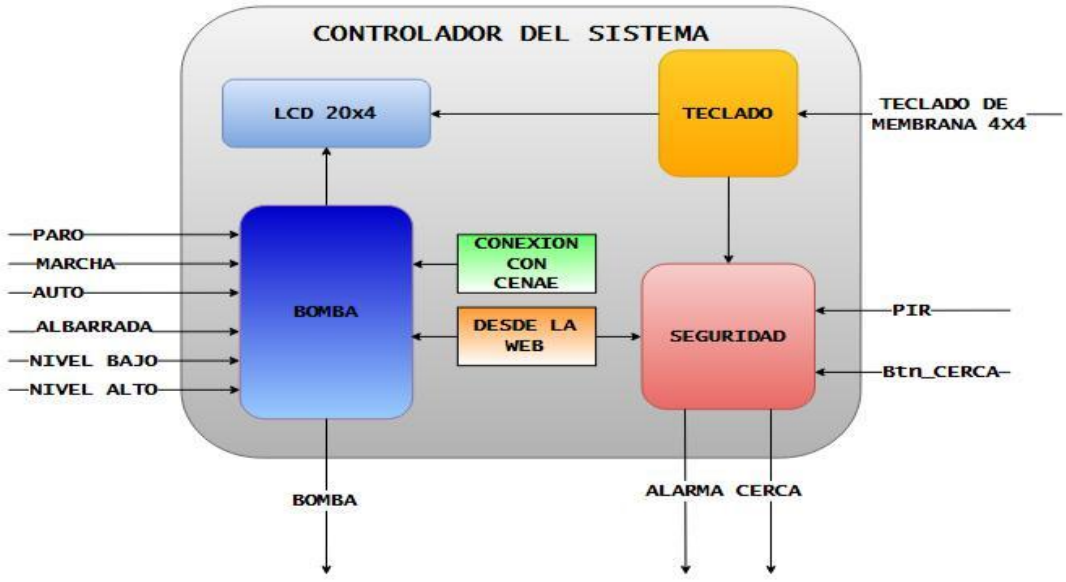

Figura 3.1: Controlador del sistema

3.1.2 Módulo Bomba

Módulo que se encarga del control de la bomba en base a las señales externas que ingresan al controlador, por ejemplo, el presionar un botón, el valor del sensor de nivel, o el comando enviado desde la página web (ver Anexo 2). Este módulo también se encarga de registrar en una base de datos los cambios de estado de la bomba y del nivel de agua en CENAE. En la Figura 3.2 se muestra el diagrama de bloque.

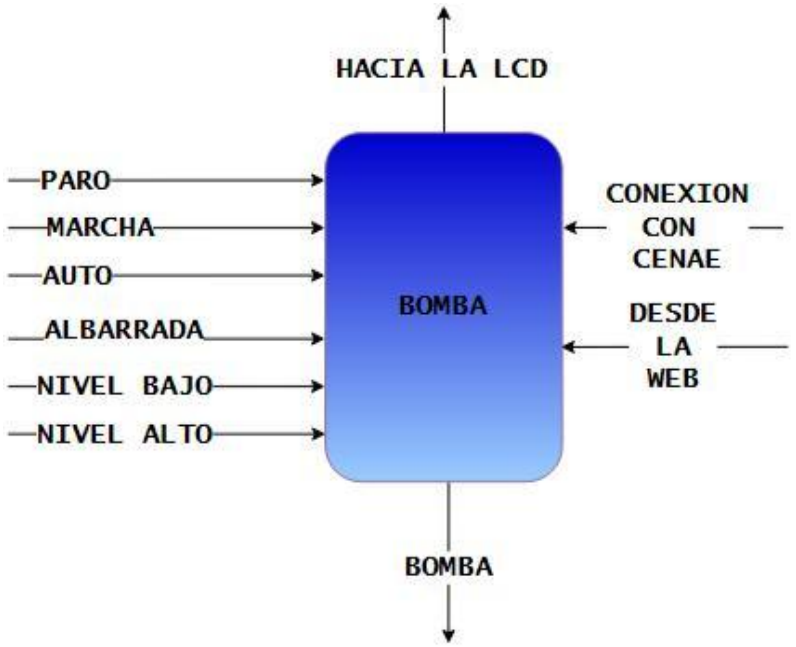

Figura 3.2: Módulo bomba

Las señales de entrada son:

- Paro: apaga la bomba, siempre que no se encuentre en modo automático y el nivel de agua en CENAE sea bajo.
- Marcha: enciende la bomba. Si se encuentra en modo automático, la bomba se apagará cuando el nivel de agua en CENAE sea alto.
- Automático: apaga o enciende la bomba dependiendo del nivel de agua en CENAE (bajo o alto). Si el nivel de agua en CENAE es medio, y la bomba no está encendida, se la puede controlar mediante Paro o Marcha. La importancia de esta señal para este sistema, depende de la conexión entre la estación de bombeo y CENAE, que es donde estarán ubicados los sensores de nivel de agua.

- Albarrada: señal de salida que proporciona el sensor de nivel de agua ubicado en la albarrada. En caso de no alcanzarse el nivel de agua mínimo en la albarrada, esta señal, apagará la bomba.
- Nivel Bajo: señal de salida del sensor de nivel ubicado en la parte inferior del estanque de CENAE, sirve de referencia para el encendido de la bomba cuando se encuentra en modo automático.
- Nivel Alto: señal de salida del sensor de nivel ubicado en la parte superior del estanque de CENAE, sirve de referencia para el apagado de la bomba cuando se encuentra en modo automático.

La señal de salida es:

- Bomba: señal de salida que controla un relé, el cual cierra o abre el circuito para energizar la bobina de control del contactor para la bomba.

El módulo envía el estado de alarmas (estado de bomba, nivel de agua en la albarrada y CENAE) a la LCD para que sean mostradas.

3.1.3 Módulo Seguridad

En la Figura 3.2 se muestra el diagrama de bloque.

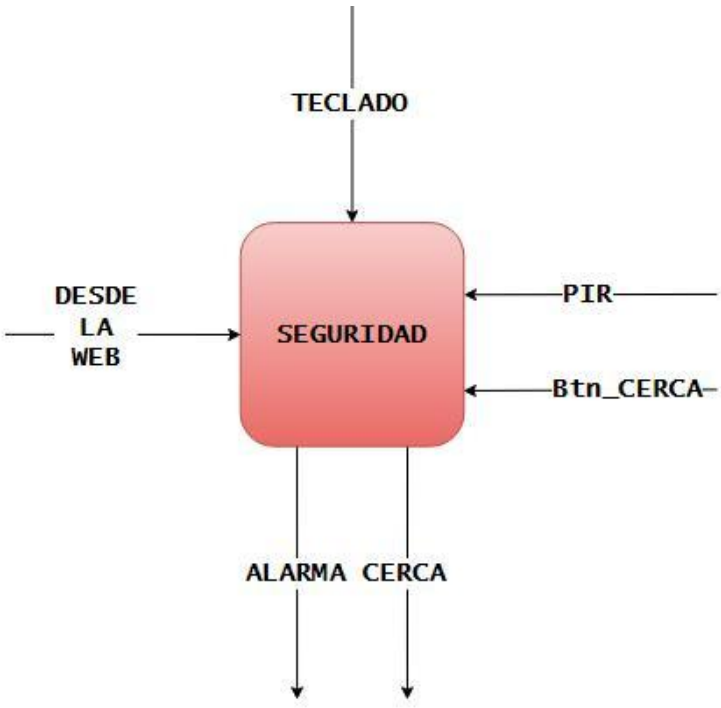

Figura 3.3: Módulo seguridad

Las señales de entrada son:

- PIR: señal de salida del sensor PIR, que alerta al sistema sobre la presencia de una persona en la estación.
- Btn_cerca: señal que se genera al pulsar un botón para encender o apagar la cerca.
- Teclado: señal de salida del módulo teclado, la cual indica que se ha ingresado una clave de acceso correcta.

Las señales de salida son:

- Alarma: señal que controla un relé, el cual cierra o abre el circuito para encender o apagar una alarma.
- Cerca: señal que controla un relé, el cual cierra o abre el circuito para activar o desactivar la alarma.

3.1.4 Módulo Teclado

Basado en un teclado de membrana 4x4, sirve al usuario para el ingreso de la clave de acceso (ver Anexo 4). Luego de verificar la clave ingresada (sea correcta o no), notifica al módulo seguridad sobre el evento.

A través del módulo display, mostrado su módulo en la Figura3.4, se guiará al usuario sobre el ingreso de cada dígito de la clave.

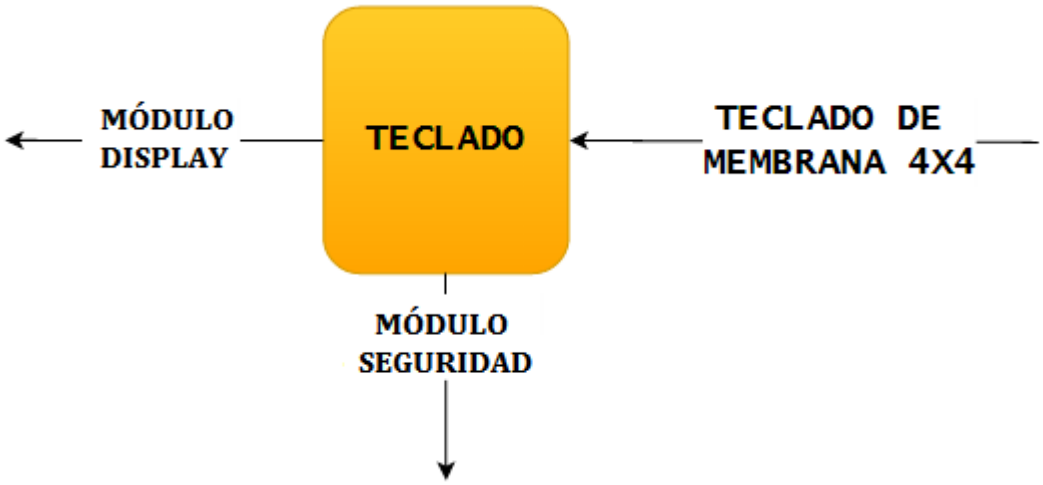

Figura 3.4: Módulo teclado

3.1.5 Módulo Display

Módulo encargado del control de una LCD 20x04. El display indicará alarmas del sistema (estado de la bomba, nivel de agua y conexión con CENAE, nivel de agua en la albarrada) y servirá de guía al usuario, en el momento de la autenticación. Debido a esto, es necesaria la comunicación con los módulos bomba y teclado, mostrados en la Figura 3.5. (Ver Anexo 5)

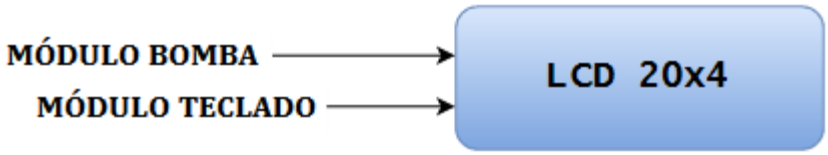

Figura 3.5: Módulo display

3.2 Interfaz Web del Prototipo

El telecontrol del sistema se realizará vía interfaz web, para lo cual se ha embebido un servidor web en la BeagleBone. Se ha escogido a Nginx, dado que es un servidor versátil, bastante ligero y que ofrece las prestaciones necesarias para la aplicación (ver Anexo 2).

El lenguaje para la ejecución de scripts del lado del servidor será PHP y del lado del cliente se empleará HTML5, CSS3, Javascript y JQuery.

La comunicación entre la interfaz web y el controlador se realizará mediante el uso de websockets, como lo vemos en la Figura 3.6 y Figura 3.7, que es una tecnología de comunicación bidireccional para aplicaciones de este tipo, la cual permite intercambiar datos entre el servidor y el cliente en tiempo de real, y de una manera rápida.

Figura 3.6: Interfaz web

Figura 3.7: Página web principal

A continuación se describen las secciones de la interfaz web:

- La pestaña de telecontrol: con esta opción se podrá controlar el encendido y apagado de la bomba, ver Figura 3.8, en ella se visualizan los mismos botones y luces piloto que físicamente se encuentran en el panel eléctrico.

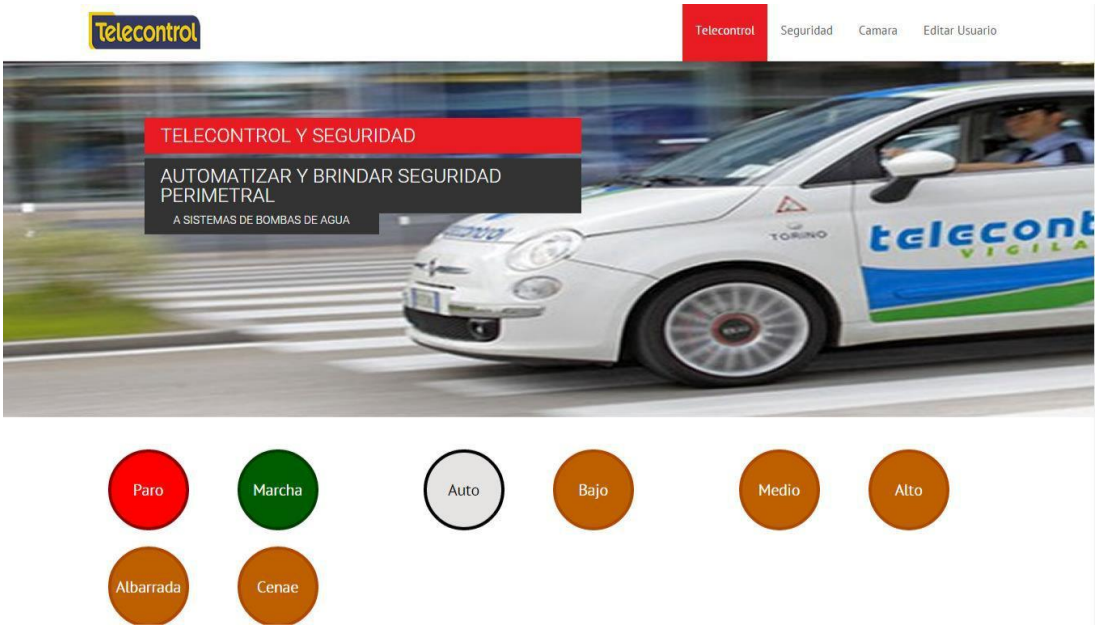

Figura 3.8: Página web telecontrol

- La pestaña de seguridad: con esta opción se podrá controlar el encendido y apagado de la cerca eléctrica, además de poder visualizar el estado de la alarma en la estación de bombeo, ver Figura 3.9.

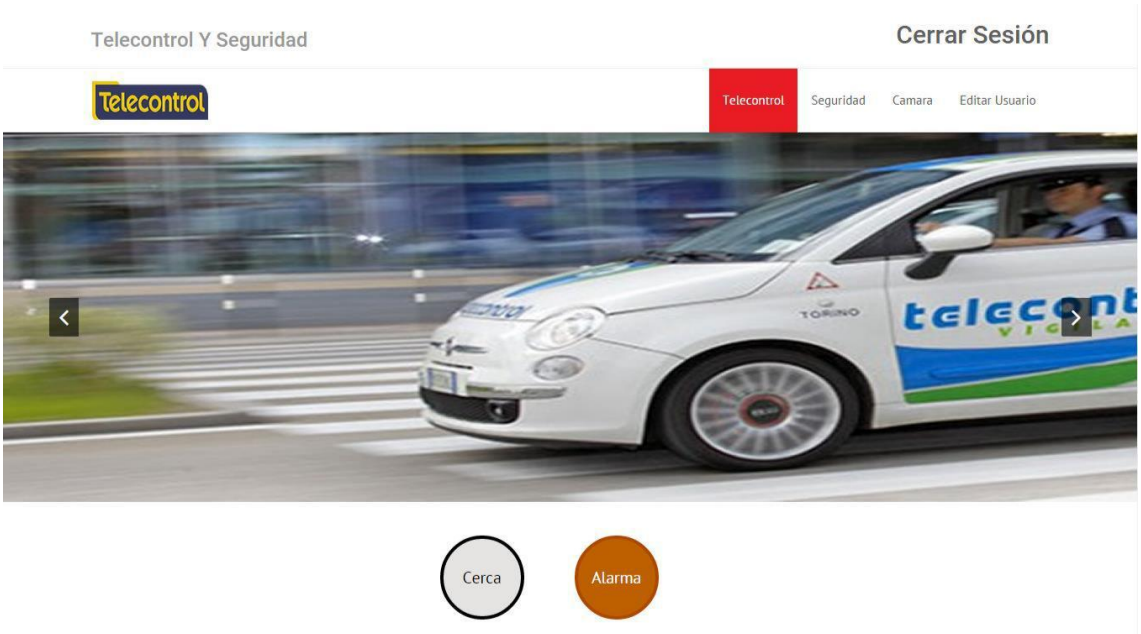

Figura 3.9: Página web seguridad

- La pestaña de cámara: sirve de enlace hacia la cámara IP del sistema
- Las pestaña de usuario: en esta página se podrán ejecutar las tareas básicas para la gestión de usuarios con acceso al sistema, ver Figura 3.10. Es decir, se podrá eliminar, agregar o editar los atributos de cualquier usuario.

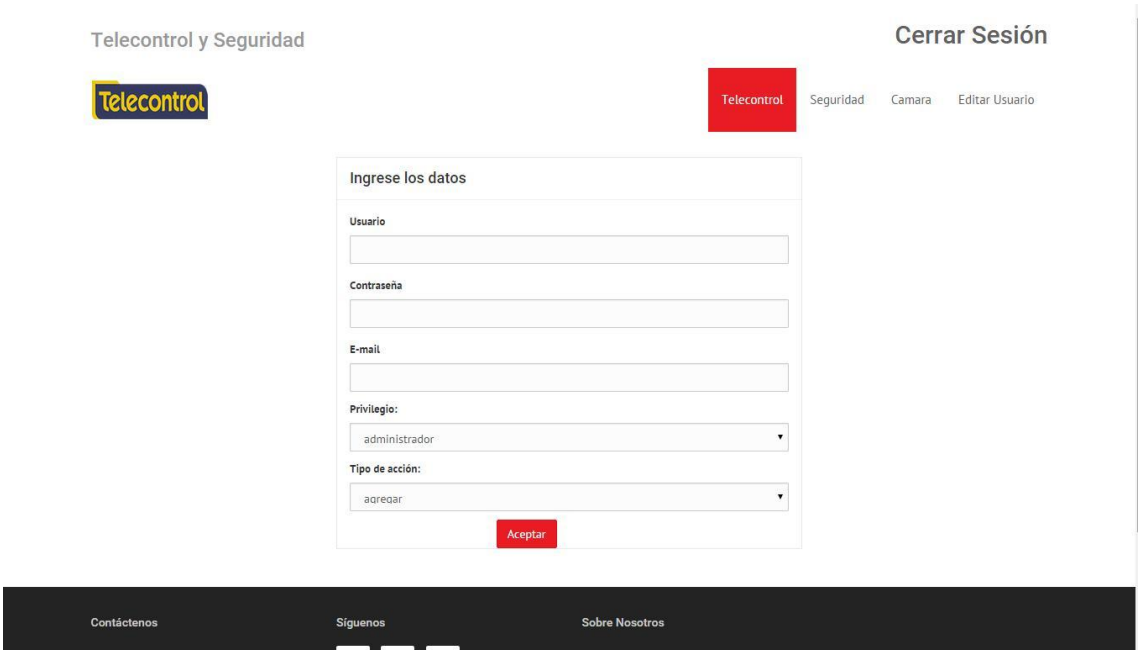

Figura 3.10: Página web usuarios

3.3 Radioenlace

Generalmente las estaciones de bombeo que son utilizadas para el riego están ubicadas en zonas donde hay poca cobertura, sea esta celular o internet. En la estación de bombeo ubicada en CENAE no se tiene cobertura de internet, por esta razón se ha decidido hacer un enlace punto multipunto para poder controlar el sistema de automatización como muestra la Figura 3.11.

Figura 3.11: Imagen satelital FIEC - Bomba - CENAE

El enlace estará constituido por 3 torres, en la FIEC, en la albarrada y en CENAE, la torre ubicada en la FIEC será la que proporcione internet a los demás puntos.

3.3.1 Software Radio Mobile

Es un programa que permite realizar los cálculos y obtener todos los datos necesarios para poder realizar radio enlaces funcionales y excluir la tarea que resulta de hacerlo manualmente, conseguir las ubicaciones geográficas e ir relevando todas las curvas de nivel que atraviesa el enlace, para poder empezar a considerar los demás aspectos operativos para un correcto enlace. El programa usa cartografía y mapas satélites. [24]

El software fue diseñado y desarrollado para darle un uso humanitario o aficionado, aunque después de algunos años de intenso desarrollo desinteresado de su creador Roger Coudé pudo lograr un nivel de excelencia que hasta se llega a comparar con los programas de simulación de radio enlaces de gigantescas marcas conocidas del medio que se estiman en miles de dólares. Con este software es posible ver y ubicar características similares a las del terreno real, por ejemplo si en el terreno hay una elevación en el software también se encontrará esa elevación, el software utiliza la información de la Misión Topográfica Radar Shuttle (SRTM) previamente precargadas, también se pueden añadir nuevas rutas o autopistas con sus características de relieve junto a las curvas de nivel.[25]

Para la simulación se debe tener conocimiento de las coordenadas exactas del lugar donde se van a ubicar las futuras torres para los enlaces. La ubicación se las puede conseguir mediante un GPS, al ubicarse previamente en los lugares de los que se necesite obtener los datos, para el sistema será en la FIEC, la albarrada y el CENAE.

Luego de obtener los datos mediante un GPS, se ingresa esta información en el software:

- FIEC: se obtuvieron los siguientes datos mediante el GPS: -2,144377 / -79,96748. Se llenan los datos como muestra la Figura 3.12

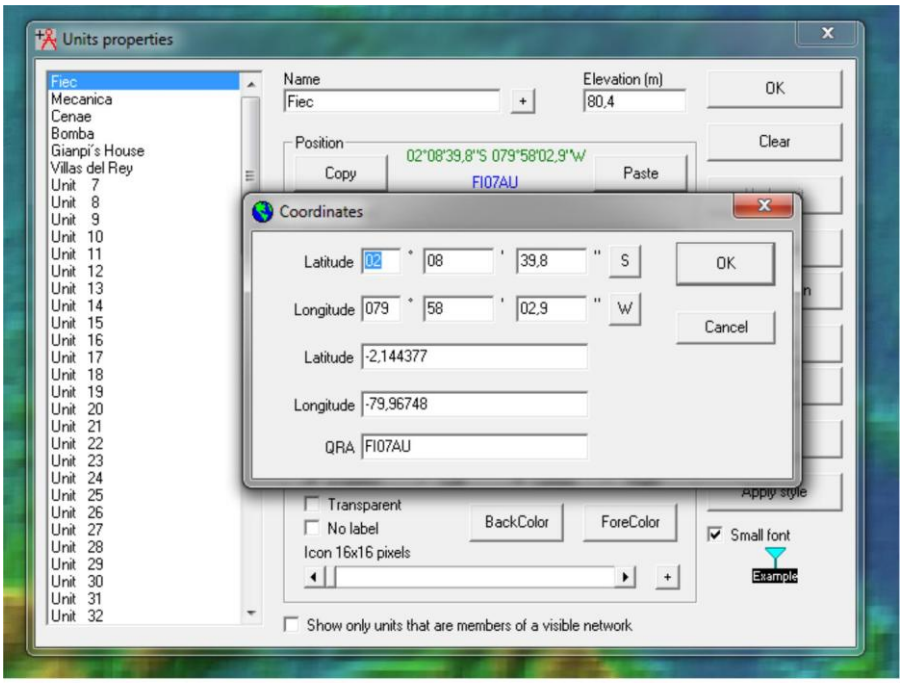

Figura 3.12: Coordenadas FIEC

- Bomba o albarrada: se obtuvieron los siguientes datos mediante el GPS: -2,14369 / -79,96359. Se llenan los datos como muestra la Figura 3.13

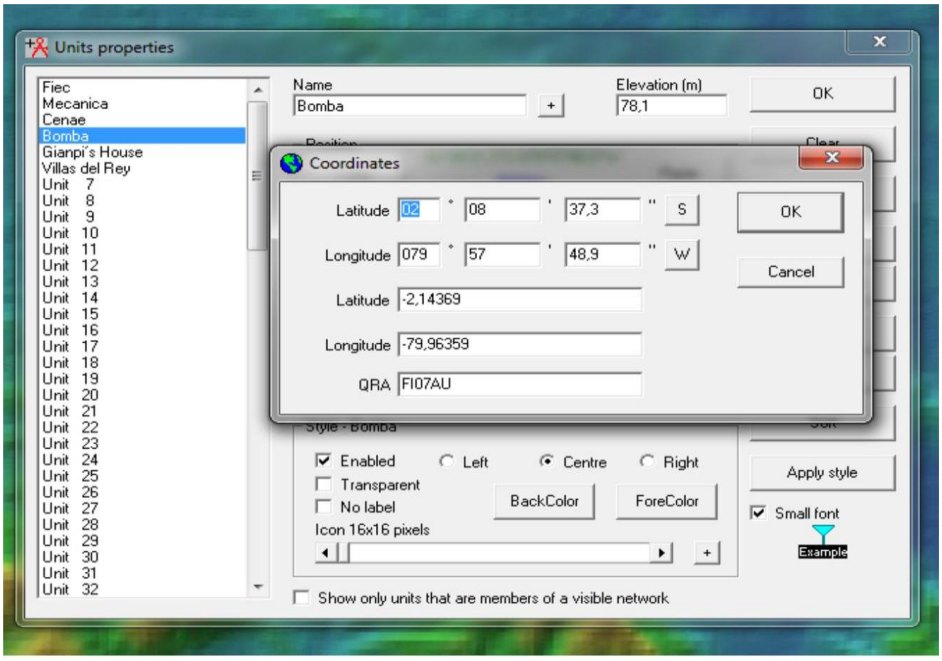

Figura 3.13: Coordenadas albarrada

- CENAE: se obtuvieron los siguientes datos mediante el GPS: -2,14054 / -79,96265. Se llenan los datos como muestra la Figura 3.14

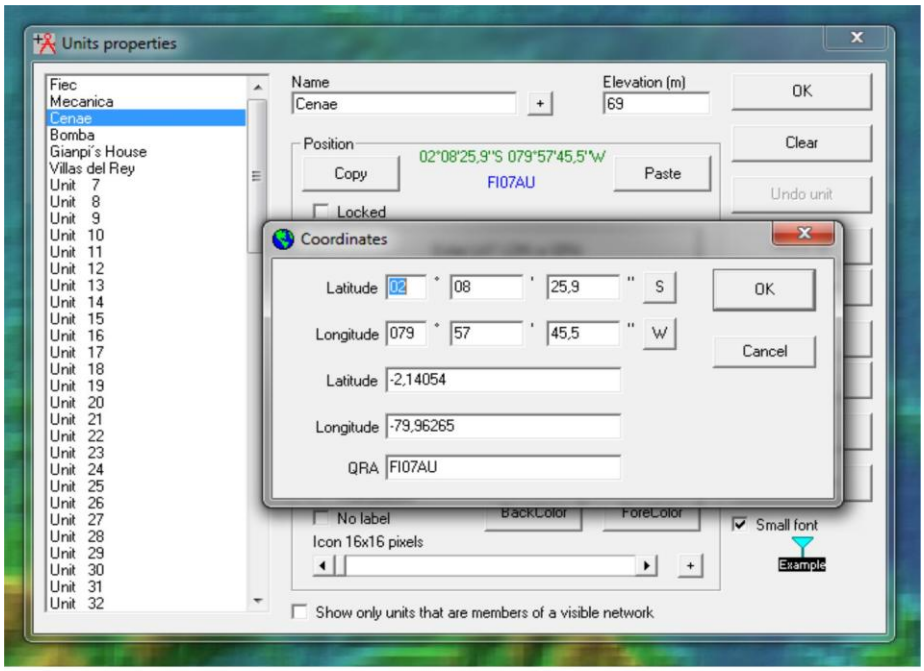

Figura 3.14: Coordenadas CENAE

Ingresados los datos se debe escoger el dispositivo que se utilizará para el radio enlace, para el sistema escogerá un equipo que es accesible y económico en el mercado local, el Nano Station M2, sus especificaciones se muestran en la Figura 3.15.

NanoStationM2 Specifications

Output Power: 28 dBm							
2.4 GHz TX POWER SPECIFICATIONS				2.4 GHz RX POWER SPECIFICATIONS			
	Data Rate/MCS	Avg. TX	Tolerance		Data Rate/MCS	Sensitivity	Tolerance
11b/g	1-24 Mbps	28 dBm	± 2 dB	11b/g	1-24 Mbps	-83 dBm	± 2 dB
	36 Mbps	26 dBm	± 2 dB		36 Mbps	-80 dBm	± 2 dB
	48 Mbps	25 dBm	± 2 dB		48 Mbps	-77 dBm	± 2 dB
	54 Mbps	24 dBm	± 2 dB		54 Mbps	-75 dBm	± 2 dB

Antenna Information	
Gain	10.4-11.2 dBi
Cross-pol Isolation	23 dB Minimum
Max. VSWR	1.6:1
Beamwidth	55° (H-pol) / 53° (V-pol) / 27° (Elevation)

Figura 3.15: Características NanoStation M2 [26]

Escogido el equipo que se utilizará, se ingresan las características del mismo y se realiza la simulación.

- Enlace FIEC - Bomba: como se puede ver en la Figura 3.16 se encuentra el enlace de FIEC con la albarrada, con una torre de 10 metros en cada punto, se tiene línea de vista sin obstáculos y es posible realizar con éxito el enlace

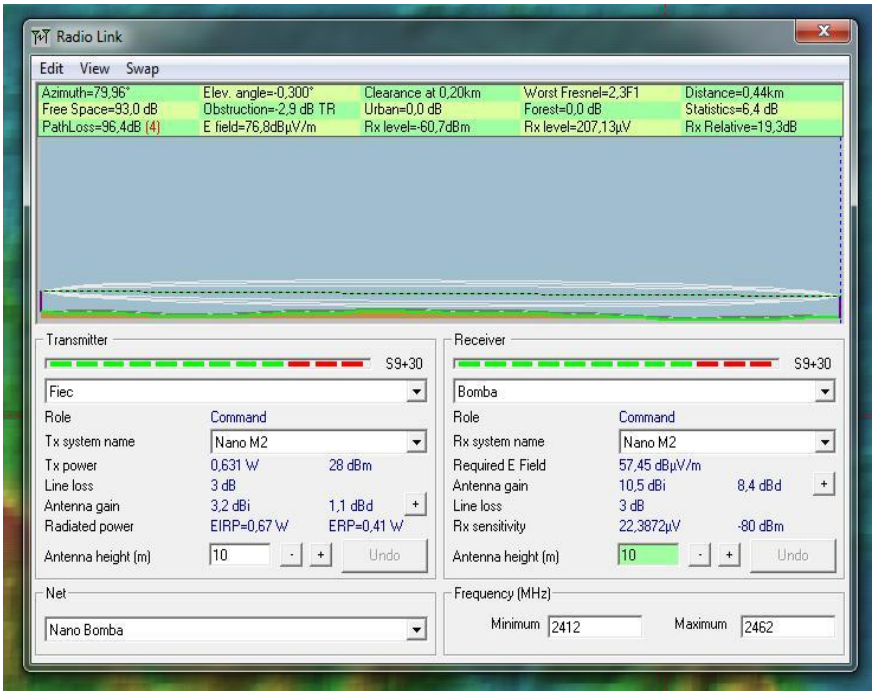

Figura 3.16: Enlace FIEC - Bomba

- Enlace FIEC - CENAE: como se puede ver en la Figura 3.17 se encuentra el enlace de FIEC con CENAE, por una mayor irregularidad en el terreno donde se encuentra CENAE, se necesita una torre de 10 metros en la FIEC y una torre de 20 metros en CENAE, ya con estas torres se tiene línea de vista sin obstáculos y es posible realizar con éxito el enlace

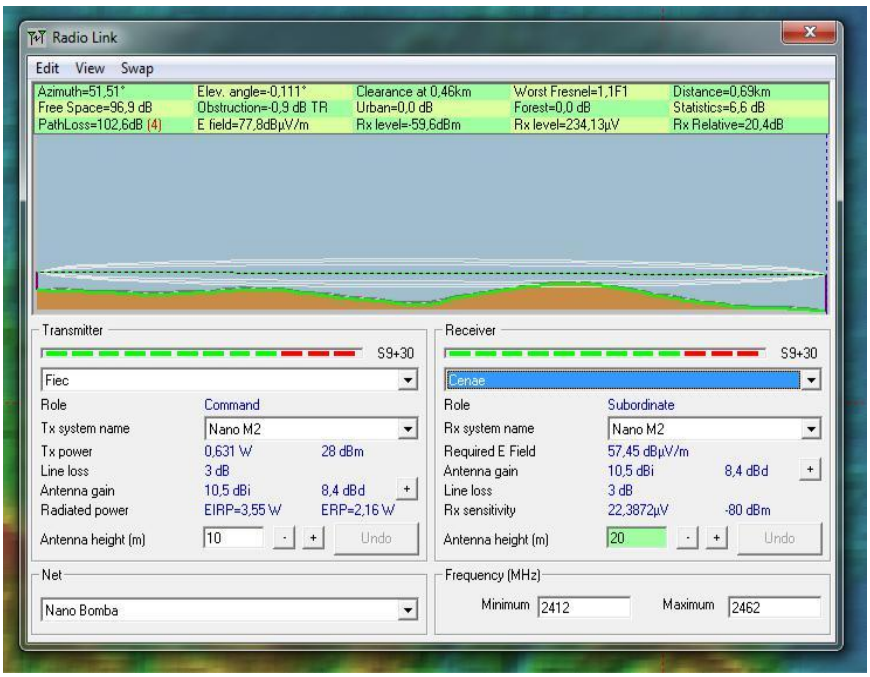

Figura 3.17: Enlace FIEC – CENAE

3.4 Elaboración de Maqueta

Para el desarrollo de la maqueta se utilizó madera para armar las estructuras de la estación de Bombeo, CENAE y la FIEC. Se emplearon recipientes de plástico para la albarrada y el reservorio de agua, y todos los dispositivos eléctricos y de control (incluida la tarjeta BeagleBone) fueron colocados en un panel eléctrico.

- La estación de bombeo: ubicada junto a la albarrada, consta de una caseta de madera con una cerca eléctrica para brindar seguridad perimetral, incorpora un sensor PIR que detecta presencia, una cámara IP para el monitoreo de la actividad dentro de la estación y finalmente una antena para la comunicación con el edificio de la FIEC y CENAE. Se ha empleado un recipiente de plástico para almacenar el agua que será llevada hacia CENAE, en la tapa del recipiente se ha colocado un sensor de nivel que indica al sistema si el nivel de agua en la albarrada es suficiente para poder encender la bomba, de no serlo, la bomba se apaga. En la Figura 3.18 se muestra su maqueta.

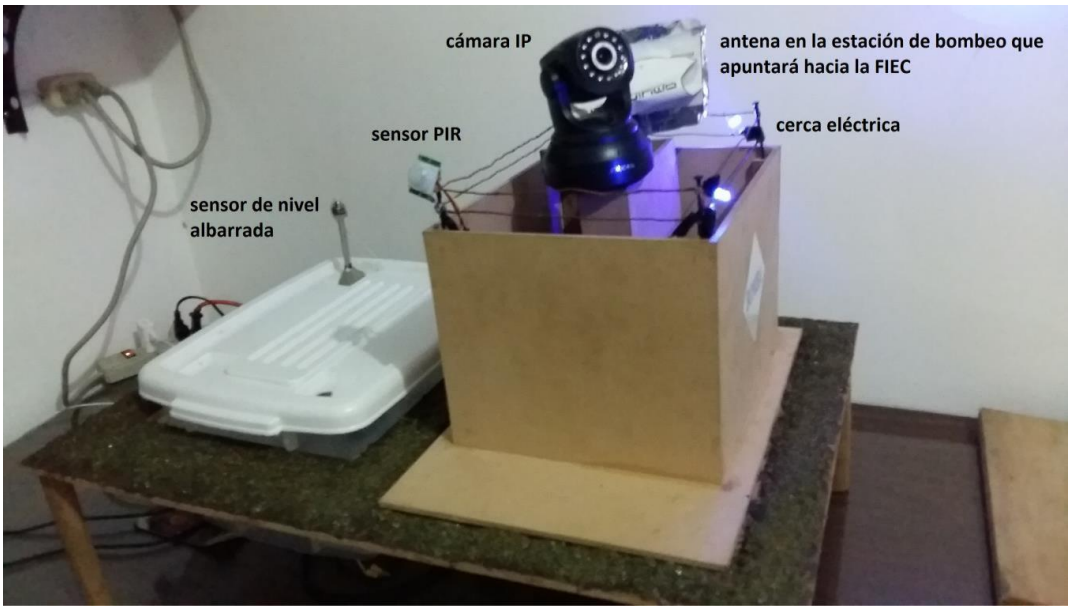

Figura 3.18: Maqueta Estación de Bombeo

- CENAE: Ubicado junto al reservorio de agua, consta de una caseta donde estará ubicada la antena que permitirá la comunicación con la FIEC y con la estación de bombeo. Se ha empleado un recipiente de plástico para almacenar el agua proveniente de la albarrada, en la tapa de este

recipiente se colocaron dos sensores de nivel que indican al sistema el nivel de agua en CENAE (bajo o alto) y en base a ello encender o apagar la bomba cuando el modo automático se encuentre activado. En la Figura 3.19 se muestra su maqueta.

Figura 3.19: Maqueta de CENAE

- Panel eléctrico: En él se colocarán todos los dispositivos para el control del sistema, incluida la tarjeta BeagleBone. Dado que en sistemas de bombeo de agua son muy utilizadas bombas trifásicas se ha decido que los dispositivos eléctricos de protección (breaker) y control (contactor) sean de tres polos. Adicionalmente, se incorpora una fuente DC de 12 V para el encendido de las luces piloto, y una fuente DC de 5V para la alimentación de la tarjeta BeagleBone. Debido a que el máximo voltaje que puede proporcionar la tarjeta es de 3.3 V, se ha utilizado una tarjeta de relés de 5V para el control de la bomba, la cerca y la alarma. En la Figura 3.20 se muestra su maqueta.

Figura 3.20: Panel eléctrico

- Radioenlace: Para la demostración del Radioenlace, se ha formado una pequeña red entre tres puntos: CENAE, Estación de Bombeo y la FIEC. Para ello se ha configurado un radioenlace punto (FIEC) - multipunto (BOMBA y CENAE). En la Figura 3.21 se muestra su maqueta.

Figura 3.21: Maqueta de Radioenlace

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. Para la instalación del servidor Web en la tarjeta de desarrollo no se presentaron mayores inconvenientes, en Internet existen foros donde se pueden seguir los pasos para su instalación.
2. Al realizar la simulación del enlace radial se determinó que son necesarias torres mayores o iguales a 10 metros para obtener una línea de vista sin obstáculos, que garantice la transmisión y recepción de las señales, esto gracias al software libre Radio Mobile. El beneficio del enlace radial es poder monitorear el sistema en tiempo real desde cualquier parte del mundo, siempre que se tenga acceso a Internet.
3. Debido a que la tarjeta de desarrollo no ofrece voltajes altos se tuvo que instalar una fuente adicional de 12 voltios para poder energizar componentes externos.
4. El sistema de seguridad es muy confiable ya que no realiza alertas falsas y todos los componentes de la parte de seguridad perimetral están correctamente configurados.
5. Es posible automatizar el funcionamiento de la estación de bombeo y brindar seguridad perimetral, mediante el uso de hardware y software libre de bajo costo.

Recomendaciones

1. La BeagleBone Black es una tarjeta de desarrollo que tiene ciertas limitaciones pero a su vez es indudablemente capaz de gestionar o realizar la mayoría de las labores de cualquier ordenador actual, puede llegar a ser considerada una minicomputadora, para esto el desarrollador le debe dar el mantenimiento y uso adecuado. A pesar de que es una tarjeta joven existen algunas actualizaciones, hardware y software necesarios que son open source. Por ser una tarjeta relativamente joven, comparada con sus competencias, hay poca información en el Internet lo que implica hacer una búsqueda un poco más exhaustiva.
2. Para validar la factibilidad de un enlace radial es posible realizar simulaciones utilizando software libre, por ejemplo Radio Mobile, este software permitió analizar y planificar el completo funcionamiento de los enlaces radiales del sistema utilizando un mapa con datos digitales de la elevación de los terrenos, junto con los datos de los equipos utilizados en el enlace de radio, realizan el modelo de

propagación de radio. Después de hacer todos estos análisis se pudo verificar que existe la llamada “línea de vista” y es factible el enlace punto-multipunto.

3. Se escogió el lenguaje multiplataforma Python para el desarrollo del sistema, debido a la gran cantidad de librerías e información disponibles en el Internet, adicional a esto es software libre. Debido a que no necesita compilar para ejecutarse es llamado “lenguaje interpretado”.
4. Para la comunicación directa entre el servidor web y la ejecución de los scripts se utilizó la tecnología websocket, ya que brinda un canal de comunicación bidireccional, full-duplex a través de un único socket TCP, permitiendo el monitoreo y la ejecución de comandos en el sistema en tiempo real.
5. Se deben verificar las especificaciones eléctricas de la tarjeta y demás componentes, para evitar sobrecargas que puedan causar un daño o un mal funcionamiento.
6. Nginx es actualmente considerado como uno de los mejores servidores web, debido a que es ligero y a su alto rendimiento. Adicionalmente es software libre y multiplataforma.
7. En caso de utilizar sensores con voltaje de operación de 5 V, se recomienda el uso de convertidores lógicos, ya que la BeagleBone opera únicamente a 3.3 V.
8. Debido a las fuertes condiciones ambientales (temperatura, polvo, humedad) a la que se expone el sistema, debe brindarse una protección y ventilación adecuada para todos los elementos eléctricos y electrónicos que lo conforman.

BIBLIOGRAFÍA

[1] Radiocomunicaciones, http://www.radiocomunicaciones.net/telemetria.html, fecha de consulta junio 2015

[2] Repositorio Unemi, http://repositorio.unemi.edu.ec/xmlui/handle/123456789/1523, fecha de consulta junio 2015

[3] Wikipedia, https://es.wikipedia.org/wiki/Telecontrol, fecha de consulta junio 2015

[4] Wikipedia, https://es.wikipedia.org/wiki/Telemetr%C3%ADa, fecha de consulta junio 2015

[5] Max4systems, http://www.max4systems.com/telemetria.html, fecha de consulta junio 2015

[6] Beagleboard, http://beagleboard.org/BLACK, fecha de consulta junio 2015

[7] Electronilab, http://electronilab.co/tienda/BeagleBone-black-arm-cortex-a8-1ghz/, fecha de consulta junio 2015.

[8] Engadget, http://es.engadget.com/2013/04/22/BeagleBone-black-1ghz-45-dolares/], fecha de consulta julio 2015.

[9] Elinux, http://elinux.org/Beagleboard:BeagleBoneBlack, fecha de consulta julio 2015.

[10] Ecured, http://www.ecured.cu/index.php/Servidor_Web, fecha de consulta julio 2015.

[11] Culturacion, http://culturacion.com/que-es-nginx/, fecha de consulta julio 2015.

[12] Redes Informáticas Grupo 2,http://redesinformaticasgrupo2.blogspot.com/p/servidor-web.html, fecha de consulta julio 2015.

[13] Monografias, http://www.monografias.com/trabajos75/servidores-web/servidores-web2.shtml, fecha de consulta julio 2015.

[14] Seguvigi, http://www.seguvigi.com/seguridad-y-vigilancia-Guayaquil.html, fecha de consulta julio 2015.

[15] Camaras Seguridad Bogota, http://camarasseguridadbogota.com.co/sistemas_de_seg uridad_bogota.html, fecha de consulta julio 2015.

[16] Informeticplus, http://www.informeticplus.com/que-es-la-seguridad-electronica, fecha de consulta agosto 2015.

[17] Alarmados, http://www.alarmados.es/grados-de-seguridad-alarmas/, fecha de consulta agosto 2015.

[18] Igme, http://www.igme.es/ZonaInfantil/MateDivul/guia_didactica/pdf_carteles/cartel4/CARTEL%204_4-4.pdf, fecha de consulta agosto 2015.

[19] Tecval, http://www.tecval.cl/que_son_las_valvulas.html, fecha de consulta agosto 2015.

[20] TLV, http://www.tlv.com/global/LA/steam-theory/types-of-valves.html, fecha de consulta agosto 2015.

[21] Definición, http://definicion.de/sensor/, fecha de consulta agosto 2015

[22] Tecnoseguro, http://www.tecnoseguro.com/faqs/alarma/que-es-un-detector-de-movimiento-pasivo-o-pir.html, fecha de consulta agosto 2015.

[23] Wikipedia, https://es.wikipedia.org/wiki/Estaci%C3%B3n_de_bombeo, fecha de consulta agosto 2015.

[24] Scribd, http://es.scribd.com/doc/94986345/Que-Es-Radio-Mobile#scribd, fecha de consulta agosto 2015.

[25] Ayuda Electrónica, http://ayudaelectronica.com/radio-mobile-software-radio-enlaces/, fecha de consulta agosto 2015.

[26] UBNT, https://dl.ubnt.com/datasheets/nanostationm/nsm_ds_web.pdf, fecha de consulta septiembre 2015.

[27] Valvulas y Accesorios, http://valvulasyaccesorios.mex.tl/, fecha de consulta septiembre 2015.

ANEXOS

Anexo 1

Script del controlador del sistema

```
from time import * import tornado.ioloop import tornado.websocket

from display_module import display from pump_module import pump from
security_module import security from keypad_module import keypad

from connection_module import connection

class PumpStation(): def __init__(self):

self.bomba = pump.Pump() self.seguridad = security.Security() self.teclado =
keypad.Keypad() self.display = display.Display()

self.conexion = connection.Connection("192.168.1.6",5)

class WSHandler(tornado.websocket.WebSocketHandler): def check_origin(self,
origin):

return True

def open(self): clients.append(self) #Pump signals

for p in ps.bomba.to_WSClient(False): self.write_message(p)

for s in ps.seguridad.to_WSClient(False): self.write_message(s)

def on_message(self, message):

message = str(message) print message

message = message.split(',') state = None

if len(message) > 1: if message[1] == '1':

state = True else:

state = False ps.bomba.from_WSClient(message[0],state)
ps.seguridad.from_WSClient(message[0],state)

def on_close(self): clients.remove(self)

#Instanciar la clase PumpStation(Estación de bombeo) ps = PumpStation()

clients = []

app = tornado.web.Application(handlers=[(r"/", WSHandler)]) server =
tornado.httpserver.HTTPServer(app) server.listen(8000)
```

```
#Función que controla el sistema

def PumpStation_Control_and_Security(): ps.bomba.controlPump()
ps.seguridad.checkSecurity() ps.display.keypad = ps.teclado.to_lcd
ps.teclado.from_lcd = ps.display.to_keypad

ps.bomba.cenae_connection = ps.conexion.connection_ok ps.seguridad.AuthN =
ps.teclado.AuthN

ps.teclado.PIR = ps.seguridad.PIR ps.display.PIR = ps.seguridad.PIR for p in
ps.bomba.to_WSClient(True):

for c in clients: c.write_message(p)

for s in ps.seguridad.to_WSClient(True): for c in clients:

c.write_message(s)

#Función que envia datos de la bomba a la LCD def fromPump_toLCD():

ps.display.pump = ps.bomba.getValue("pump") ps.display.albarrada = not
ps.bomba.getValue("albarrada") ps.display.cenae = ps.bomba.cenae_tolcd
ps.display.cenae_connection = ps.conexion.connection_ok

mainLoop = tornado.ioloop.IOLoop.instance() pumpstation_scheduler =

tornado.ioloop.PeriodicCallback(PumpStation_Control_and_Security,10, io_loop =
mainLoop)

tolcd_scheduler = tornado.ioloop.PeriodicCallback(fromPump_toLCD,1000, io_loop =
mainLoop)

pumpstation_scheduler.start() tolcd_scheduler.start() mainLoop.start()
```

Anexo 2

Script del Módulo Bomba

```
import Adafruit_BBIO.GPIO as GPIO from time import *

from signal_module import signal import Queue

from database_module import db_interaction from threading import Thread

class Pump():

def __init__(self):

self.signals = []

#Seniales para controlar la bomba

self.signals.append(signal.Signal(    'stop', 'P8_27', 'in', 'momentary button', None))

self.signals.append(signal.Signal(    'start', 'P8_28', 'in', 'momentary button', None))

self.signals.append(signal.Signal(    'auto', 'P8_29', 'in', 'latching button', 'P9_23'))

self.signals.append(signal.Signal( 'albarrada', 'P8_30', 'in', 'sensor', 'P9_28'))

self.signals.append(signal.Signal( 'low_level',        None, 'in',       'sensor',        None))

self.signals.append(signal.Signal('high_level',        None, 'in',       'sensor',        None))

self.signals.append(signal.Signal(    'pump', 'P8_37','out',  'relay',  None))

#Luces piloto indicadoras del nivel de agua en CENAE self.low_level = 'P9_24'
GPIO.setup(self.low_level,GPIO.OUT) GPIO.output(self.low_level,GPIO.LOW)
self.medium_level = 'P9_25' GPIO.setup(self.medium_level,GPIO.OUT)
GPIO.output(self.medium_level,GPIO.LOW) self.high_level = 'P9_26'
GPIO.setup(self.high_level,GPIO.OUT) GPIO.output(self.high_level,GPIO.LOW)
self.conn_status = 'P9_30' GPIO.setup(self.conn_status,GPIO.OUT)
GPIO.output(self.conn_status,GPIO.LOW)

#Luces piloto indicadoras del estado de la bomba self.paro = 'P9_21'
GPIO.setup(self.paro,GPIO.OUT) GPIO.output(self.paro,GPIO.LOW)

self.marcha = 'P9_22' GPIO.setup(self.marcha,GPIO.OUT)
GPIO.output(self.marcha,GPIO.LOW)

#Variables tipo boolean para validar el nivel de agua y la conexion con CENAE
self.cenae = False

self.filling = False self.from_cenae = False self.cenae_connection = False
self.pump_mode = "paro" self.pump_state = False self.pump_laststate = False
```

```
self.setValue("stop","on") self.cenae_tolcd = "BAJO"

def getValue(self,name): for i in self.signals:

if i.name == name: return i.getValue()

def setValue(self,name,value): for i in self.signals:

if i.name == name: i.setValue(value)

def getChange(self,name): for i in self.signals:

if i.name == name: return i.change

def low_level_albarrada(self): if not self.getValue("stop"): self.setValue("stop","on")

if self.getValue("start"): self.setValue("start","off")

if self.getValue("pump"): self.setValue("pump","off")

def stop_pressed(self): self.pump_mode = "paro" if not self.getValue("auto"):

if self.getValue("pump"): self.setValue("pump","off") self.setValue("start","off")

else:

if self.getValue("low_level"):

self.setValue("stop","off")

else:

if self.getValue("pump"): self.setValue("pump","off") self.setValue("start","off")

def automatic_pressed(self): self.pump_mode = "auto"

if self.getValue("low_level"): if not self.getValue("pump"):

self.setValue("pump","on")

self.setValue("start","on")

self.setValue("stop","off")

else:

if not self.getValue("high_level"): self.setValue("pump","off") self.setValue("start","off")
self.setValue("stop","on")

def start_pressed(self): self.pump_mode = "marcha"

if self.cenae_connection and self.from_cenae:

if self.getValue("low_level") or self.getValue("high_level"): if not self.getValue("pump"):
```

```
self.setValue("pump","on")
self.setValue("stop","off")
else:
if self.getValue("pump"): self.setValue("pump","off") self.setValue("start","off")
self.setValue("stop","on")
else:
print "START PRESSED - PUMP SHOULD TURN ON" if not self.getValue("pump"):
self.setValue("pump","on")
self.setValue("stop","off")
def cenae_water_level(self):
if self.cenae_connection and self.from_cenae: if not self.getValue("high_level"):
self.cenae = True self.cenae_tolcd = "ALTO"
if not GPIO.input(self.high_level): GPIO.output(self.high_level,GPIO.HIGH)
if GPIO.input(self.medium_level): GPIO.output(self.medium_level,GPIO.LOW)
if GPIO.input(self.low_level): GPIO.output(self.low_level,GPIO.LOW)
elif self.getValue("low_level"): self.cenae = False self.cenae_tolcd = "BAJO"
if not GPIO.input(self.low_level): GPIO.output(self.low_level,GPIO.HIGH)
if GPIO.input(self.high_level): GPIO.output(self.high_level,GPIO.LOW)
if GPIO.input(self.medium_level): GPIO.output(self.medium_level,GPIO.LOW)
else:
self.cenae_tolcd = "MEDIO"
if not GPIO.input(self.medium_level): GPIO.output(self.medium_level,GPIO.HIGH)
if GPIO.input(self.low_level): GPIO.output(self.low_level,GPIO.LOW)
if GPIO.input(self.high_level): GPIO.output(self.high_level,GPIO.LOW)
if not GPIO.input(self.conn_status): GPIO.output(self.conn_status,GPIO.HIGH)
else:
self.cenae_tolcd = "****" if self.getValue("auto"):
self.setValue("auto","off") if self.getValue("high_level"):
```

```
self.setValue("high_level","off") if self.getValue("low_level"):
self.setValue("low_level","off") if GPIO.input(self.conn_status):
GPIO.output(self.conn_status,GPIO.LOW) if GPIO.input(self.high_level):
GPIO.output(self.high_level,GPIO.LOW) if GPIO.input(self.medium_level):
GPIO.output(self.medium_level,GPIO.LOW) if GPIO.input(self.low_level):
GPIO.output(self.low_level,GPIO.LOW)
def setPumpPilotLight(self): if self.getValue("stop"):
GPIO.output(self.paro,GPIO.HIGH)
GPIO.output(self.marcha,GPIO.LOW)
else:
GPIO.output(self.paro,GPIO.LOW)
GPIO.output(self.marcha,GPIO.HIGH)
def controlPump(self):
#Nivel de agua bajo en la albarrada if self.getValue("albarrada"):
self.low_level_albarrada() self.pilotLights("albarrada")
else:
self.pilotLights("albarrada") #Boton de paro presionado
if self.getValue("stop"): self.stop_pressed()
#Si existe comunicacion con CENAE
if self.cenae_connection and self.from_cenae: #Boton de automatico presionado
if self.getValue("auto"): self.automatic_pressed()
self.pilotLights("auto")
#Boton de marcha presionado if self.getValue("start"):
self.start_pressed()
#Nivel de agua bajo o alto en CENAE else:
if self.getValue("start"):
#Boton de marcha presionado self.start_pressed()
self.cenae_water_level() self.PumpRegister() self.setPumpPilotLight()
```

```
if not self.cenae_connection: self.from_cenae = False

def PumpRegister(self):

self.pump_state = self.getValue("pump") if self.pump_laststate != self.pump_state:
self.pump_laststate = self.pump_state db = db_interaction.DB("Telecontrol")

try:

pump_thread = Thread(target=db.RecordPump, args=(strftime("%Y-%m-
%d"),strftime("%H:%M:%S"),self.pump_state,self.cenae,self.pump_mode,))

pump_thread.start() except e:

print e

def pilotLights(self,name): for i in self.signals:

if i.name == name: i.setPilotLight()

def to_WSClient(self,change): WSClients = []

if change:

for i in self.signals: if i.change:

WSClients.append(i.to_WS())

if self.cenae_connection and self.from_cenae: WSClients.append("cenae,1")

else:

WSClients.append("cenae,0")

if GPIO.input(self.medium_level): WSClients.append("medium_level,1")

else: WSClients.append("medium_level,0")

else:

for i in self.signals: WSClients.append(i.to_WS())

if self.cenae_connection and self.from_cenae: WSClients.append("cenae,1")

else:

WSClients.append("cenae,0") if GPIO.input(self.medium_level):

WSClients.append("medium_level,1") else:

WSClients.append("medium_level,0") return WSClients

def from_WSClient(self,name,value): for i in self.signals:

if i.name == name: i.from_WS(value) if i.name == "stop":
```

```
self.stop_pressed() elif i.name == "start": self.start_pressed()

elif i.name == "automatic": self.automatic_pressed()

elif name == "cenae":

print "CENAE CONNECTION IS OK" self.from_cenae = True

if __name__ == '__main__': pump = Pump()

while 1: pump.controlPump()
```

Anexo 3

Script del Módulo Seguridad

```
from time import *

import Adafruit_BBIO.GPIO as GPIO from threading import Thread

from matrix_keypad import BBb_GPIO from signal_module import signal from
timer_module import timer

from keypad_module import keypad

from database_module import db_interaction

class Security(): def __init__(self):

#Security GPIOs Setup self.signals = []

self.signals.append(signal.Signal(    'pirPS', 'P8_35', 'in',    'pir',None))

self.signals.append(signal.Signal(    'ef_pb', 'P8_36', 'in','latching button',None))

self.signals.append(signal.Signal(    'alarm', 'P8_38','out',  'relay','P9_29'))

self.signals.append(signal.Signal('electric_fence', 'P8_39','out',     'relay','P9_27'))

#Pump Station presence detection variable

self.PIR          = False

#Authentication variables

self.AuthN              = False

self.AuthN_timer        = False

self.AuthN_thread   = None

self.AuthN_time         = 30

#Inactivity variables

self.IA_time            = 600

self.IA_timer           = False

self.IA_thread          = None

#Mail notification variables

self.db = db_interaction.DB("Telecontrol")

self.mail_send          = False

self.mail_thread        = None
```

```
self.emailSubject       = "ALERTA DE SEGURIDAD!!!"

self.emailBody                  = "***** ALERTA DE SEGURIDAD - ESTACION DE
BOMBEO

*****\n\nTIPO DE ALERTA: presencia detectada sin autenticarse\n"

def getValue(self,name): for i in self.signals:

if i.name == name: return i.getValue()

def setValue(self,name,value): for i in self.signals:

if i.name == name: i.setValue(value)

def getChange(self,name):

for i in self.signals:

if i.name == name: return i.change

def pilotLights(self): for i in self.signals:

i.setPilotLight()

def to_WSClient(self,change): WSClients = []

if change:

for i in self.signals: if i.change:

WSClients.append(i.to_WS())

else:

for i in self.signals: WSClients.append(i.to_WS())

return WSClients

def from_WSClient(self,name,value): for i in self.signals:

if i.name == name: i.from_WS(value)

if name == "AuthN": self.AuthN = value

def checkSecurity(self): self.pilotLights() #Electric Fence control if
self.getValue("ef_pb"):

if not self.getValue("electric_fence"): self.setValue("electric_fence","on")

else:

if self.getValue("electric_fence"):

self.setValue("electric_fence","off") #Pump Station presence detection
```

```
if not self.PIR:
if self.getValue("pirPS"): self.PIR = True
elif self.PIR:
if not self.AuthN:
if not self.AuthN_timer:
#Inicia el timer para la autenticacion de usuario self.AuthN_timer = True
self.AuthN_thread = timer.Timer(self.AuthN_time,"AuthN")
elif self.AuthN_thread.timeout:
#Tiempo de autenticacion terminado, se enciende la alarma y se envia mail
if not self.getValue("alarm"): self.setValue("alarm","on")
#Se envia mail de notificacion if not self.mail_send:
self.mail_thread = Thread(target=self.db.SendEmail,
args=(self.db.GetList("email"),self.emailSubject,self.emailBody,))
self.mail_thread.start() self.mail_send = True self.AuthN_timer = False
self.AuthN_thread.timeout = False
else:
if self.getValue("alarm"): self.setValue("alarm","off")
if self.AuthN_timer: self.AuthN_timer = False
if not self.IA_timer: self.IA_timer = True
self.IA_thread = timer.Timer(self.IA_time,"ia") elif self.IA_thread.timeout:
self.PIR = False self.IA_timer = False self.AuthN = False self.mail_send = False
self.IA_thread.pirPS = self.getValue("pirPS") if __name__ == '__main__':
security = Security('P8_34','P8_35','P8_36','P8_27','P8_38','P9_29','P8_39','P9_30',
['P8_11','P8_13','P8_15','P8_17'],
['P8_12','P8_14','P8_16','P8_18'])
while 1: security.checkSecurity()
```

Anexo 4

Script del Módulo Teclado

```
from time import *
import Adafruit_BBIO.GPIO as GPIO from threading import Thread
from matrix_keypad import BBb_GPIO
from database_module import db_interaction from signal_module import signal
from timer_module import timer import Queue
class Keypad():
def __init__(self):
self.rowPins    = ['P8_11','P8_12','P8_13','P8_14']
self.columnPins         = ['P8_15','P8_16','P8_17','P8_18']
self.keypad     = BBb_GPIO.keypad(4,self.rowPins,self.columnPins)
self.to_lcd       = Queue.Queue()
self.db = db_interaction.DB("Telecontrol")
self.AuthN       = False
self.PIR          = False
self.timeout     = False
self.timeout_thread = None
self.key_pressed       = False
self.key                    = ''
self.password = ''
self.menu_option  = ''
self.usuario     = "no_user"
self.from_lcd   = Queue.Queue()
self.isready = "no_ready"
thread = Thread(target=self.run, args=())
thread.start()
def run(self): while 1:
```

while not self.PIR: if self.AuthN:

self.AuthN = False if self.key_pressed:

self.key_pressed = False

while not self.key_pressed and not self.AuthN: if self.keypad.digit():

sleep(0.2) self.key_pressed = True

if not self.AuthN: self.to_lcd.put("auth_menu") sleep(1)

while len(self.password) < 4: self.key = self.keypad.digit() self.password += str(self.key) while self.from_lcd.empty():

sleep(0.2)

self.isready = self.from_lcd.get() self.to_lcd.put(self.password)

sleep(0.2)

emailSubject = "ALERTA DE SEGURIDAD!!!" self.usuario = self.db.CheckUser(self.password) if self.usuario != "no_user":

self.AuthN = True

while self.from_lcd.empty(): sleep(0.2)

self.isready = self.from_lcd.get() self.to_lcd.put("granted") sleep(2)

emailBody = "***** ALERTA DE SEGURIDAD - ESTACION DE BOMBEO

*****\n\nTIPO DE ALERTA: acceso AUTORIZADO a la Estación de Bombeo, la clave ingresada es CORRECTA\n"

else:

self.AuthN = False

while self.from_lcd.empty(): sleep(0.2)

self.isready = self.from_lcd.get() self.to_lcd.put("denied") sleep(2)

emailBody = "***** ALERTA DE SEGURIDAD - ESTACION DE BOMBEO

*****\n\nTIPO DE ALERTA: acceso DENEGADO a la Estación de Bombeo, la clave ingresada es INCORRECTA\n"

self.key_pressed = False self.timeout = False self.key = '' self.password = ''

mail_thread = Thread(target=self.db.SendEmail, args=(self.db.GetList("email"),emailSubject,emailBody,))

mail_thread.start() if self.AuthN:

access_record = Thread(target=self.db.RecordAccess, args=(strftime("%Y-%m-%d"),strftime("%H:%M:%S"),self.usuario,))

access_record.start() self.usuario = "no_user"

if not self.from_lcd.empty(): while not self.from_lcd.empty():

self.from_lcd.get()

if __name__ == '__main__':

k = Keypad(['P8_11','P8_13','P8_15','P8_17'],['P8_12','P8_14','P8_16','P8_18'])

Anexo 5

Script del Módulo Display

```
from time import *

import Adafruit_BBIO.GPIO as GPIO from display_20x4 import i2c_lcd from threading import Thread

import Queue

class Display(): def __init__(self):

self.lcd = i2c_lcd.lcd() self.display = "title" self.lastdisplay = None self.lcd.lcd_clear() self.pump = False self.lastpump = False self.albarrada = False self.last_albarrada = False self.cenae = "BAJO" self.last_cenae = "BAJO"

self.cenae_connection = False self.last_cenae_connection = False self.keypad = Queue.Queue()

self.from_keypad = "" self.to_keypad = Queue.Queue() self.PIR = False

self.display_thread = Thread(target=self.set_display, args=()) self.display_thread.start()

def set_display(self): while 1:

if not self.PIR: self.display = "title"

else:

if self.display != "keypad": self.display = "system_alarms"

if self.display == "title":

if self.lastdisplay != self.display: self.lcd.lcd_clear()

sleep(0.8)

self.lcd.lcd_display("Telecontrol y Seguridad de una Estacion de Bombeo") self.lastdisplay = self.display

if not self.keypad.empty(): self.from_keypad = self.keypad.get() if self.from_keypad == "auth_menu":

self.display = "keypad"

if self.lastdisplay != self.display: self.lastdisplay = self.display self.lcd.lcd_clear()

sleep(0.8) self.lcd.lcd_display_string("Autenticacion", 1, 4) self.lcd.lcd_display_string("Ingrese la clave:", 2, 2) self.lcd.lcd_display_string("----",3,9)

print "LCD READY" self.to_keypad.put("is ready")
```

```
elif self.from_keypad == "granted": self.lcd.lcd_clear()
sleep(0.8) self.lcd.lcd_display("clave correcta!") sleep(1)
self.lcd.lcd_clear() sleep(0.8)
self.lcd.lcd_display("acceso autorizado!") sleep(1)
print "LCD READY" self.to_keypad.put("is ready") self.display = "system_alarms"
elif self.from_keypad == "denied": self.lcd.lcd_clear()
sleep(0.8)
self.lcd.lcd_display("clave incorrecta!") sleep(1)
self.lcd.lcd_clear() sleep(0.8)
self.lcd.lcd_display("acceso denegado!") sleep(1)
self.to_keypad.put("is ready") self.display = "system_alarms"
else:
if len(self.from_keypad) == 1: self.lcd.lcd_display_string("*---",3,9)
elif len(self.from_keypad) == 2: self.lcd.lcd_display_string("**--",3,9)
elif len(self.from_keypad) == 3: self.lcd.lcd_display_string("***-",3,9)
elif len(self.from_keypad) == 4:
self.lcd.lcd_display_string("****",3,9) print self.from_keypad
sleep(1) self.to_keypad.put("is ready")
if self.display == "system_alarms": if self.lastdisplay != self.display: self.lastdisplay =
self.display
self.lcd.lcd_clear() sleep(0.8)
if self.pump:
self.lcd.lcd_display_string("bomba: ENCENDIDA",1,1) else:
self.lcd.lcd_display_string("bomba: APAGADA ",1,1) if self.albarrada:
self.lcd.lcd_display_string("nivel albarrada:ALTO",2,1) else:
self.lcd.lcd_display_string("nivel albarrada:BAJO",2,1)
if self.cenae == "ALTO": self.lcd.lcd_display_string("nivel cenae: ALTO ",3,1)
elif self.cenae == "BAJO": self.lcd.lcd_display_string("nivel cenae: BAJO ",3,1)
```

```
else:
self.lcd.lcd_display_string("nivel cenae: MEDIO",3,1)
if self.cenae_connection: self.lcd.lcd_display_string("conexion cenae: SI",4,1)
else:
self.lcd.lcd_display_string("conexion cenae: NO",4,1) else:
if self.lastpump != self.pump: self.lastpump = self.pump
if self.pump: self.lcd.lcd_display_string("ENCENDIDA",1,8)
else:
self.lcd.lcd_display_string("APAGADA ",1,8)
if self.last_albarrada != self.albarrada: self.last_albarrada = self.albarrada if self.albarrada:
self.lcd.lcd_display_string("ALTO",2,17) else:
self.lcd.lcd_display_string("BAJO",2,17) if self.last_cenae != self.cenae:
self.last_cenae = self.cenae if self.cenae == "ALTO":
self.lcd.lcd_display_string("ALTO ",3,14) elif self.cenae == "BAJO":
self.lcd.lcd_display_string("BAJO ",3,14) else:
self.lcd.lcd_display_string("MEDIO",3,14)
if self.last_cenae_connection != self.cenae_connection: self.last_cenae_connection = self.cenae_connection if self.cenae_connection:
self.lcd.lcd_display_string("SI",4,17) else:
self.lcd.lcd_display_string("NO",4,17)
if __name__ == '__main__': print "Display module!"
```

ANEXO 6

ÍNDICE DE FIGURAS