UNIVERSIDAD DE MONTEMORELOS

FACULTAD DE INGENIERÍA Y TECNOLOGÍA

DISEÑO Y CONSTRUCCIÓN DE UNA INTERFAZ PARA COMUNICAR LOS DISPOSITIVOS DE LA RED ELÉCTRICA CON LA UNIDAD TERMINAL REMOTA

Por

Hiram Isaí Hernández González

Asesor: Alejandro Garrido Soto

RESUMEN DE PROYECTO DE INVESTIGACIÓN

Universidad de Montemorelos

Facultad de Ingeniería y Tecnología

Titulo de la investigación parcial:

DISEÑO Y CONSTRUCCIÓN DE UNA INTERFAZ PARA COMUNICAR LOS DISPOSITIVOS DE LA RED ELÉCTRICA CON LA UNIDAD TERMINAL REMOTA

Investigador: Hiram Isaí Hernández González

Asesor: Alejandro Garrido Soto

Fecha de defensa: 8 de Mayo del 2014

Problema

La configuración de las Unidades Terminales Remotas (UTR) comerciales, es muy limitada a las necesidades del usuario, cuentan con un número especifico de puertos de entrada y salida, los cuales no pueden ser modificados a beneficio del usuario, además presenta constantes fallas debido al mal funcionamiento de su sistema de protección contra descargas eléctricas, generando continuas bajas de este dispositivo, por lo tanto pérdidas económicas.

Metodología

Para resolver este problema, se propone crear una UTR configurable y de bajo costo. Para esto es necesario crear la interfaz que comunica a la UTR con los sensores de los dispositivos en la red eléctrica así como un software con interfaz gráfica que permita al operador, interpretar la información recibida en la UTR y enviar pulsos de control a los dispositivos de la red eléctrica. Este proyecto fue realizado con el fin de crear un prototipo de dicha interfaz junto con el software que lo controla. Se utilizó la tarjeta embebida PC-Duino, como CPU del prototipo. Esta tarjeta utiliza el sistema operativo Linux, esta característica, da lugar a la programación con el lenguaje Python. El circuito electrónico fue diseñado con Proteus 8 Professional, el cual es un software especializado en la simulación y creación de tarjetas de circuitos impresos denominados PCB por sus siglas en ingles (Print Circuit Board). El software fue programado con el lenguaje de programación Python utilizando la librería para interfaz gráfica Tkinter.

Resultados

Se creo un prototipo capaz de recibir pulsos digitales y enviar un pulso digital de control, controlado por una interfaz gráfica configurable, que guarda un log de eventos por segundo, programada en Python.

Conclusión

Concluyendo así, que es factible crear una interfaz entre los dispositivos de la red eléctrica y la UTR, que reciba y envíe pulsos digitales, para el control y monitorización de los dispositivos instalados en las redes eléctricas.

Universidad de Montemorelos

Facultad de Ingeniería y Tecnología

Proyecto
presentado en cumplimiento parcial
de los requisitos para el grado de
Ingeniería en Electrónica y Telecomunicaciones

Por

Hiram Isaí Hernández González

Mayo 2014

DEDICATORIA

Todo lo que escrito esta en este documento, no pudo haber sido real si no hubiera sido por Él, esa persona que me formó en el vientre de mi madre, que no me ha abandonado ni un segundo desde que Él me formó, que me ha cuidado y motivado. Gracias Dios, por Tu amor, por permitir que realice cosas que jamás me hubiera imaginado que haría, todo lo que yo haga sea para Tu honra y gloria. A mis padres que siempre me apoyaron, que a pesar que soy una persona adulta, ellos están ahí como si aun fuera un niño, gracias por estar siempre ahí, en los momentos mas difíciles de mi carrera, por brindarme palabras de motivación y confianza. A mi hermano menor, que aunque nos separamos por un período de tiempo, siempre estuvo al pendiente de mí, motivándome y ayudándome, siendo el mi mejor amigo, no pude tener mejor hermano y amigo. Familia, Gracias, por sus esfuerzos, por su sacrificio, por su amor y cariño. No hay forma de agradecer lo que han hecho por mí. Dedico el proyecto que realice a ustedes que lo han hecho posible. Gracias.

Dios

Margarito Hernández

Perlina González

Isaac Hernández

TABLA DE CONTENIDO

TABLA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

Antecedentes

En los años sesenta cada fabricante debía resolver sus problemas de control por sí solo. Normalmente se ocupaba una memoria reducida, por lo cual debían de comunicarse constantemente con su centro de control para enviar los datos. En los años setenta, fabricantes como SIEMENS, Square-D, o Allen-Bradley, implementaron autómatas capaces de controlar grandes cantidades de entradas y salidas. Estos dispositivos eran robustos, no tenían un entorno amigable, estaban diseñados para soportar condiciones severas por lo tanto eran grandes, pesados y muy caros. (Penin, 2012)

Con el pasar de los años los componentes electrónicos fueron reduciendo su tamaño y aumentando su efectividad, gracias a esto se fabricaron dispositivos llamados micro PLC (Programmable Logic Controler). El PLC permitía ser programado para realizar actividades de control inmediatas, tenía un sistema de programación genérico llamado escalera, lo que deparó un éxito inmediato en todo el ámbito industrial. (Penin, 2012)

Sistema SCADA

Las siglas SCADA significan Supervisory Control And Data Acquisition (Control con Supervisión y Adquisición de Datos). El sistema SCADA es un conjunto de aplicaciones que trabajan en el área de control y supervisión en la industria, cuenta con comunicación digital con los instrumentos de medición y actuadores, y una interfaz grafica con la que el usuario puede ver los resultados de los instrumentos de medición y controlar los actuadores. (Jaume Romangoza Cabús, 2004)

Los primeros sistemas SCADA solo proporcionaban al operador reportes periódicos de los instrumentos de medición que estuvieran conectados al Sistema SCADA. Eran simples sistemas de telemetría, simple monitoreo y algunas características de control. El operador interpretaba los

contadores y lámparas detrás de paneles llenos de indicadores. A medida que la tecnología ha avanzado, los ordenadores han tomado la tarea de manejar y administrar los datos, mostrándolos al operador de una forma amigable para su interpretación. (David Chacon, 2001)

Características del Sistema SCADA

El sistema SCADA esta dividido en tres bloques principales (Penin, 2012): (a) bloque del software de Adquisición de datos y control (Unidad Central Maestra), (b) bloque de adquisición y mando (sensores y actuadores), (c) bloque de interconexión (comunicaciones).

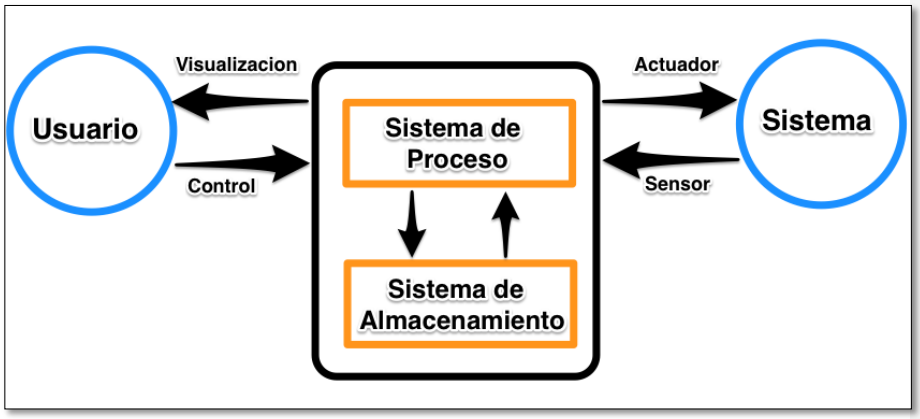

Figura 1 Arquitectura típica de un sistema SCADA

El Sistema de proceso y almacenamiento esta dentro del bloque de interconexión, por medio del software de adquisición de datos y control, el usuario puede visualizar la información obtenida y controlar el bloque de interconexión y el bloque de adquisición y mando. Esta comunicación es realizada a través de redes de comunicación corporativas (Mayormente Ethernet). El Sistema de proceso obtiene datos del sistema a monitorear por medio de los sensores que envían esta información por el bus de campo. El bus de campo son todas las conexiones entre el sistema de proceso y el sistema a monitorear. De la misma forma el Sistema de proceso, envía los comandos de control ingresados por el usuario a los actuadores del sistema. (Penin, 2012)

Los actuadores y sensores son diversos dispositivos que se encuentran en el mercado. Uno de estos dispositivos es la Unidad Terminal Remota, que incluye tanto actuadores como sensores.

Unidad Terminal Remota

Una UTR consiste de un gabinete o panel de terminales para la instrumentación y los cables de control, conectados en un sistema de energía. La posición de los interruptores en el sistema de energía es monitoreada por relevadores auxiliares. Los valores análogos son obtenidos por transformadores de voltaje y transformadores de corriente que son conectados a los buses del sistema de energía. (Smith, 2010)

Los primeros sistemas SCADA, tenían UTRs que no eran inteligentes, el sistema central utilizaba un sistema de interrogación (polling) para tener acceso a sus datos. La UTR nunca hablaba a menos que fuera interrogada. La UTR hacia todo lo necesario para recuperar los últimos datos de sus instrumentos, además de la conversión de señales analógicas a digitales, y después contestar a la petición que se le había hecho. (David Chacon, 2001)

Historia de la UTR en México

Las UTR sirven como los ojos y manos de un sistema de control supervisorio. Los sistemas de control supervisorio están compuestos por las UTR, que obtienen todos los datos por medio de sus sensores y por una unidad central maestra, que es una computadora que recibe todos los datos obtenidos desde las UTR. (Instituto de Investigaciones Electricas (IIE), 1981)

El Instituto de Investigaciones Eléctricas (IIE), comenzó esta investigación en el año de 1977, teniendo como resultado una de las primeras UTR de México en 1981 denominada TRIIE (Terminal Remota del Instituto de Investigaciones Eléctricas). (Instituto de Investigaciones Electricas (IIE), 1981)

LA TRIIE tenía características que la ubicaron a la vanguardia de la tecnología en esa época. Era una UTR Programable, inteligente, modular, diagnosticable y confiable. (Instituto de Investigaciones Electricas (IIE), 1981)

El Primer prototipo de la TRIIE fue instalado en la subestación Hermosillo II, en Hermosillo Sonora, se instaló juntamente con una UTR comercial para realizar pruebas en ambos. Para realizar este logro se contó con la asesoría y la experiencia en la fabricación de equipos electrónicos del ingeniero Herbert red, de la Universidad de Utah. (Instituto de Investigaciones Electricas (IIE), 1981)

Arquitectura de una UTR

La arquitectura de una UTR comprende un CPU, memoria volátil y memoria no volátil para procesar y guardar programas y datos. Esta se comunica con otros dispositivos vía serial o tarjetas de modem con interfaces de entrada y salida. Tiene una fuente de energía con un respaldo de baterías, protección contra sobre tensión y picos de voltaje, reloj de tiempo real, y temporizador de vigilancia para garantizar que se reinicia cuando se opera en modo de reposo. (Francis Enejo Idachaba, 2012)

La siguiente figura muestra un diagrama de bloques de la configuración típica de una UTR. El hardware típico de una UTR incluye un procesador de control, y una memoria asociada, entradas y salidas análogas, entradas de contador, entradas digitales, salidas digitales, interfaces de comunicación y fuente de poder. (Francis Enejo Idachaba, 2012)

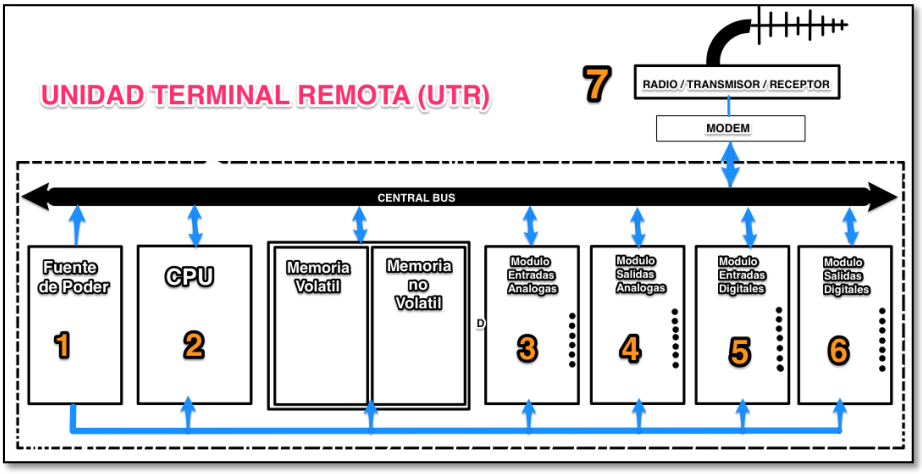

Figura 2 Arquitectura típica de una unidad terminal remota

1. **Módulo Fuente de Poder:** La UTR necesita una continua fuente de poder para funcionar, pero hay situaciones donde la UTR se encuentra lejos de una fuente de electricidad. En este caso, la UTR esta equipada con una fuente de poder alternativa y batería de respaldo para facilitarle energía en caso de pérdida de poder. (Francis Enejo Idachaba, 2012)

2. **Unidad Central de Proceso (CPU):** Los diseños actuales de UTR utilizan un micro procesador de 16 o 32 bits, con una capacidad de memoria total de 256KB ampliable a 4MB. Este sistema es controlado por un Firmware y un reloj de tiempo real con un calendario completo para poder agregar la fecha y la hora de los eventos. (Francis Enejo Idachaba, 2012)

3. **Módulo de Entradas análogas:** Una señal de entrada análoga es generalmente voltaje o corriente que varía sobre un rango de valores, en proporción directa al proceso de medición física, como presión, o temperatura. (Francis Enejo Idachaba, 2012)

4. **Módulo de Salidas Análogas:** La función del módulo de salidas análogas, es convertir un valor digital dado por el CPU a un valor análogo por medio de un convertidor análogo digital. Esta representación análoga puede ser usada por una variable de control de actuadores. (Francis Enejo Idachaba, 2012)

Rudy Dzul Ramírez 19/5/14 22:05
Comentario [1]: Todo lo señalado en color amarillo son faltas de ortografía que debes corregir.

5. **Módulo de entradas digitales:** Este módulo es usado para indicar estados y señales de alarmas. (Francis Enejo Idachaba, 2012)

6. **Módulo de salidas digitales:** Este módulo es usado para controlar una salida de voltaje en el canal apropiado, con 3 enfoques posibles: Triac Switching, Read Relay Switching, TTL voltaje Outputs. (Francis Enejo Idachaba, 2012)

7. **Interface de Comunicación:** Las UTR modernas están diseñadas para ser lo suficientemente flexibles para manejar múltiples medios de comunicación como: (a) RS232/RS442/RS485, (b) Ethernet, (c) Dial Up Telephone lines/ dedicated landlines, (d) Mircrowave/MUX, (e) Satélite, (f) X.25 Packet Protocols, (g) Radio via Trunked/VHF/UHF/900 Mhz. (Francis Enejo Idachaba, 2012)

Interfaz Hombre Máquina

La función de monitorización de estos sistemas se realiza sobre un PC industrial ofreciendo una visión de los parámetros de control sobre la pantalla del ordenador, lo que se denomina interfaz hombre Maquina o HMI por sus siglas en ingles (Human Machine Interface)

Rudy Dzul Ramírez 19/5/14 22:08
Comentario [2]: No es necesario que lo pongas en mayúsculas interfaz hombre maquina

La HMI comprende los sinópticos de control y los sistemas de presentación grafica. La función de un panel sinóptico es la de representar, de forma simplificada, el sistema bajo control. En un principio los paneles sinópticos eran de tipo estático, colocados en grandes paneles plagados de indicadores y luces. Con el tiempo han ido evolucionando, junto con el software, en forma de representaciones graficas en pantallas de visualización. (Penin, 2012).

Estructura de un software HMI

1. **Configuración:** Permite al usuario definir el entorno del trabajo de su aplicación según la disposición de pantallas requeridas, y los niveles de acceso para los distintos usuarios. (Jaume Romangoza Cabús, 2004)

Rudy Dzul Ramírez 19/5/14 22:10
Comentario [3]: Estas hablando de pantallas requeridas corrige ahí por favor

2. **Interfaz grafico del operador:** Proporciona al operador las funciones de control y supervisión de la planta. (Jaume Romangoza Cabús, 2004)

3. **Módulo de proceso:** Ejecuta las acciones de mando pre-programadas a partir de los valores actuales de variables leídas. (Jaume Romangoza Cabús, 2004)

4. **Gestión y archivo de datos:** Se encarga del almacenamiento y procesado ordenada de los datos, según formatos inteligibles para periféricos hardware o software del sistema, de forma que otra aplicación o dispositivo pueda tener acceso a ellos. (Jaume Romangoza Cabús, 2004)

Definición del Problema:

En la empresa Comisión Federal de Electricidad (CFE), se presenta la necesidad de tener un dispositivo configurable de acuerdo a la aplicación, que controle y observe los dispositivos de campo y a su vez envíe la información obtenida hacia una Unidad Central Maestra (UCM).

Rudy Dzul Ramírez 19/5/14 22:14
Comentario [4]: Puedes utilizar estas siglas en adelante

Justificación

La empresa CFE busca aumentar la eficiencia en el monitoreo de las variaciones eléctricas y las condiciones físicas de los dispositivos en la red eléctrica. De esta manera brindar un mejor servicio a sus clientes.

Objetivos

Diseñar un prototipo de interfaz entre la UTR y los dispositivos de la red eléctrica, para la adquisición de los datos digitales generados por una simulación de los sensores de los dispositivos en la red eléctrica.

Diseñar un software, que sea la interfaz HMI entre hombre y UTR, que permita al operador: (a) visualizar los datos digitales generados por el prototipo de interfaz entre la UTR y los dispositivos de la red eléctrica, (b) Guardar un log de eventos por segundo, (c) configurar el panel de alarmas de la

interfaz grafica, (c) enviar un pulso de control al prototipo de interfaz entre la UTR y los dispositivos de la red eléctrica.

Preguntas

¿Cómo diseñar un prototipo, que reciba las señales simuladas de los dispositivos instalados en las redes eléctricas?

¿Cómo enviar las señales obtenidas por un prototipo a un CPU para mostrarse en una interfaz gráfica HMI?

¿Cómo diseñar una interfaz gráfica para controlar y monitorizar un prototipo que adquiere y envía información entre el CPU y los actuadores y sensores simulados de dispositivos instalados en la red eléctrica?

Hipótesis

Es posible construir una UTR a partir software y hardware libre, que permita realizar modificaciones acorde a las necesidades del usuario.

Limitaciones

La adquisición de información tomo más tiempo del esperado, debido a la falta de conocimiento en diversas áreas.

El tiempo establecido para la investigación era corto en comparación con el tiempo requerido para un proyecto de tal amplitud.

Falta de acceso al campo de trabajo para realizar pruebas por ser una zona de alto riesgo.

Falta de interruptores y transformadores de alta potencia para la simulación de las variables a adquirir y controlar.

Delimitaciones

Diseño de prototipo sin tarjeta de acondicionamiento de señal para los sensores de los dispositivos de la red eléctrica reales.

En el prototipo se utiliza una fuente de 12 volts para simular los sensores de los dispositivos de la red eléctrica.

Definición de Términos

Shields: Dispositivos electrónicos creados para la adición de características a tarjetas embebidas ya construidas

Widgets: Herramientas visuales, para las interfaces gráficas

Protoboards: Tablilla para experimentos utilizada para crear pruebas de circuitos

Diagrama esquemático: Diagrama utilizado para el diseño de circuitos electrónicos

APORTE AL PROYECTO

Metodología

La figura 3 nos muestra de forma general las actividades para el desarrollo de un prototipo de interfaz software – hardware. Tomando como punto de partida, la selección de la tarjeta embebida a utilizar, debido a que hay diferentes tarjetas embebidas en el mercado, con características especificas para las necesidades del cliente. Se determinará el lenguaje de programación y el sistema operativo donde se ejecutará el programa. Se definirán las librerías a utilizar para la interfaz gráfica. Se diseñara y construirá tanto el hardware como el software que lo controlara. Se implementará una comunicación entre el software y el hardware y por ultimo se realizaran pruebas de comunicación entre el software y el hardware.

Rudy Dzul Ramírez 19/5/14 22:22
Comentario [5]: Recuerda que todo lo que está en amarillo debes corregir la ortografía

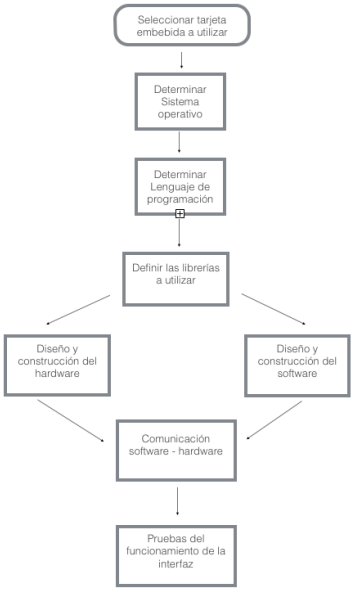

Figura 3 Diagrama de bloques de la metodología utilizada

Debido al tipo de desarrollo e integración del proyecto a realizar se sugiere el uso de una metodología mixta de investigación, en la cual se implica el uso de documentación para el desarrollo del trabajo.

Investigación de las tarjetas embebidas para usar como CPU

La UTR requiere de un CPU, que lleve a cabo todos los procesos del sistema, tanto recibir la información de los sensores, como enviarla, a una unidad central maestra, crear logs de eventos, así mismos utilizar esta información para realizar tareas programadas. Se propone el uso de un hardware libre, que permita la modificación a beneficio del usuario (Fernández, 2012). Se encontró que algunas tarjetas embebidas son de hardware libre, las cuales podríamos utilizar como nuestro CPU. Algunas de estas son: (a) Arduino, (b) FPGA, (c) Raspberry pi. Estas tarjetas embebidas son de hardware libre, lo que quiere decir que se pueden modificar acorde a las necesidades del usuario. El lenguaje de programación que se utiliza en estas tres tarjetas es un lenguaje libre, por lo cual podemos utilizarlo para las necesidades del proyecto. La diferencia entre las 3 tarjetas es que la Raspberry pi utiliza Linux como sistema operativo, y es programable en cualquier lenguaje que Linux soporte. En cambio Arduino no tienen un sistema operativo y el lenguaje de programación que utiliza es un lenguaje basado en C. La FPGA también carece de sistema operativo y es programada en Hardware Description Lenguage (HDL). Por lo que se decidió utilizar una tarjeta embebida que utilizara Linux, ya que Linux es un sistema operativo robusto, que soporta variedad de lenguajes de programación y a su vez es software libre (Stallman, 2004). Se buscaron mas tarjetas embebidas que utilizaran Linux y se encontraron las siguientes: (a) Raspberry Pi, (b) Cubieboard, (c) Rabbit RCM6700, (d) IGEPv2, (e) PC-Duino.

Las características que se toman en cuenta para la selección de la tarjeta son accesibilidad, precio, cantidad de información acerca de ellas, comunidad trabajando con ellas, capacidades físicas. A continuación una tabla hecha para realizar la decisión de las tarjetas que utilizaremos como CPU.

Tabla 1 Comparación de las Tarjetas Embebidas

Caracteristicas	Rasp Berry PI	Cubie Board	Rabbit RCM6700	IGEPv2	PCDUINO
Precio	$35	$44 - $55	$ 99.00	218 Euros con envio a	$60
Procesador	ARM 11 700 MHZ	ARM Cortex A8 1GHZ	RABBBIT 6000 200MHZ	Cortex A8	ARM Cortex A8 1GHZ
RAM	512 MB	512MB a 1GB	1MB	512MB	1GB
Storage	SD	Micro SD	1MB Flash	Micro SD	Micro SD
USB	SI	SI	NO	SI	Si, Micro y USB
Analog IN	SI	NO	SI	SI	12
Analog OUT	SI	SI	SI 32 I/O	SI	6 (PWM)
HDMI	NO	SI	NO	SI	SI
Standares	GPIO, UART, I2C SPI	I2C, SPI, UART	I2C, SERIAL	UART, RS232,RS485	Digital GPIO 14
Hay Informacion	SI mucha	10% de la RBP	NO	Poca	Poca
Se consigue facil	SI	SI (EUA)	SI (EUA)	Medio (SPAIN)	SI
Conexión Camara	NO	SI	NO	SI	SI
Wifi	NO	Con shield	NO	SI	Con shield
TEMP	0 - 70	-15 -70	-40 - 85	-40 a 85	-20 - 80

A partir de esta información, el equipo del proyecto tomo una decisión. Se opto por utilizar la Cubieboard por su compatibilidad con más estándares, mayor memoria RAM, cantidad de información acerca de esta, facilidad de obtenerla, precio y comunidad utilizando esta tarjeta. Para tener un respaldo, en caso que una de las tarjetas fallara, se compró una segunda tarjeta, esta fue la PC-Duino, fue seleccionada por su parecido con el Arduino y así poder utilizar los shields creados para Arduino que se encuentran en el mercado.

Conociendo el CPU

La siguiente etapa, fue empezar a conocer la forma de usar las herramientas y características de las tarjetas adquiridas. Se inicio usando la tarjeta Cubieboard, por su variedad de puertos, por su accesibilidad, y por la facilidad de uso de sus puertos GPIO.

La Cubieboard, cuenta con un procesador ARM Cortex 8 a 1Ghz, 1GB de RAM, expansión de memoria con Micro SD, salidas análogas, GPIO, UART, SPI, I2C.

Instalando SO Linux

Esta tarjeta opera con Android nativo, y se le puede instalar Linux en una tarjeta SD, en diferentes versiones especiales para cubieboard.

Algunas de estas versiones son: (a) Cubiezz, (b) Cubian, (c) Cubiuntu, (d) Lubuntu, (e)Linaro, entre otras. Las versiones antes mencionadas fueron probadas y utilizadas para ver su funcionamiento y su estabilidad. Se decidió utilizar Linux en vez de Android porque es un sistema mas completo, ya que Android es un sistema operativo para dispositivos móviles y Linux es más robusto.

La mayoría de las versiones de Linux tenían gran cantidad de errores, entre los principales, no proporcionaban internet. Las dos versiones más estables fueron Lubuntu, y Cubiuntu. Se decidió trabajar con Cubiuntu porque nos entrega internet fácilmente, y la forma de acceder a sus puertos es más accesible que Lubuntu.

Accediendo a los puertos GPIO

Lar Cubieboard cuenta con gran cantidad de puertos, entre los cuales se encuentran: (a) GPIO, (b) UART, (c) SPI, (d) I2C. Para el prototipo se necesita entradas y salidas digitales, por tal motivo utilizamos los puertos GPIO. Las siglas GPIO significan "Entradas y Salidas de Propósito General" (General Purpose Input Output). Estos puertos se pueden configurar como entradas o salidas. Cuando están configurados como salidas, se puede escribir en un determinado registro para que controle la salida de cada pin del puerto. Cuando están configurados como entradas, se puede detectar el estado del pin leyendo un registro interno asociado al puerto. (Guerrero, 2007)

Para acceder a los puertos GPIO de la tarjeta, es necesario modificar el archivo del sistema llamado Script.bin. En algunas versiones de Linux era muy complicado acceder a este archivo, ya que se localizaba en diferentes directorios dependiendo de la versión, además, de que es necesario convertir este archivo de tipo bin a tipo fex para poder modificarlo. En la versión Cubiuntu, no se tuvo este problema, porque el sistema viene con un acceso directo y un script que lo convertía automáticamente para poder modificarlo y lo regresaba a su formato original después de guardar los cambios y reiniciaba el sistema.

En este archivo se encontraban todos los pines de la tarjeta, algunos de estos estaban activos otros no. Se localizaron los puertos GPIO y se dieron de alta algunos puertos como se muestra en la figura 4:

```
gpio_used = 1
gpio_num = 4
gpio_pin_1 = port:PG00<1><default><default><default>
gpio_pin_2 = port:PB19<1><default><default><default>
gpio_pin_3 = port:PG02<0><default><default><default>
gpio_pin_4 = port:PG04<0><default><default><default>
```

Figura 4 Fragmento de código en el archivo Script.bin modificado a Script.fex para activar los puertos GPIO

Con el fin de comprobar que se accedió correctamente a los puertos GPIO, se utilizó el código de la figura 5 para encender y apagar un LED,

```
import wiringpi2

OUTPUT = 1
pin = 2
HIGH = 1
LOW = 0

wiringpi2.wiringPiSetupPhys()
wiringpi2.pinMode(pin,OUTPUT) # se asignaba el pin 2 como salida

while 1:
    wiringpi2.digitalWrite(pin,HIGH) # Escribe 1 en el pin 2
    wiringpi2.delay(1000) # Se espera 1 segundo
    wiringpi2.digitalWrite(pin,LOW)  # Escribe 0 en el pin 2
    wiringpi2.delay(1000) # Se espera 1 segundo
```

Figura 5 Código para encender y apagar un LED con los puertos GPIO.

El LED encendió exitosamente, y se dispuso a crear la interfaz grafica para controlar y obtener datos desde la tarjeta.

Determinar el lenguaje de Programación a utilizar

Se necesitaba un lenguaje de programación que nos permitiera combinar el lenguaje de bajo nivel para controlar hardware, y el lenguaje de alto nivel para crear interfaces graficas, manejar archivos, bases de datos, entre otras tareas. Había varios lenguajes que podíamos utilizar, entre los cuales estaban, C++, JAVA , Python. De estos tres se eligió Python

¿Porque Python?

Python es un lenguaje de programación fácil de aprender y potente. La elegante sintaxis de Python, su gestión de tipos dinámica y su naturaleza interpretada hacen de este, el lenguaje ideal para scripts y desarrollo rápido de aplicaciones, en muchas áreas y en la mayoría de las plataformas. (Rossum, 2000)

El intérprete de Python esta disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que si no se utilizan librerías especificas de cada plataforma, nuestro programa podrá corren en todos estos sistemas sin grandes cambios. (Duque, 2010)

Además de su facilidad de aprender y fácil de manejar, es un lenguaje que se puede usar como lenguaje de bajo nivel tanto como lenguaje de alto nivel. Otra de las ventajas de usar este lenguaje, es que en internet hay mucha información con respecto al control de hardware usando este lenguaje.

Conociendo el Lenguaje Python

Lo primero que se buscó, fue una descripción clara de Python, así que se buscó una guía básica de Python, se encontró: Guía de aprendizaje de Python Release 2.0 escrito por el creador de Python, Guido Van Rossum y el libro Python para Todos escrito por Raúl Gonzalo Duque, de estas dos publicaciones se obtuvo la primera idea de que es Python y como empezar a utilizarlo.

Después de implementar la instalación de Python en un ordenador y en la tarjeta, se prosiguió a aprender la sintaxis, y métodos mas usados.

Interfaz gráfica

Rudy Dzul Ramírez 19/5/14 22:31
Comentario [6]: Viste que aquí si le pusiste acento?

Para el proyecto es necesario tener una interfaz grafica, que cuente con las principales características de un HMI que son: (a) configuración, (b) interfaz grafica del operador, (c) módulo de proceso, (d) gestión y archivo de datos.

Para crear una HMI que se aplicará a nuestro prototipo se utilizó la librería Tkinter, que es una librería nativa de Python a partir de Python 3. Tkinter cuenta con diferentes widgets de los cuales utilizaron: (a) **entry**: Son las cajas de Texto para introducir datos, (b) **label:** Son las etiquetas, son utilizadas para mostrar información de tipo String, (c) **button:** Realizar operaciones, (d) **listbox:** Para que el usuario seleccione una opción desde una lista, (e) **tk:** creación de ventanas, (f) **frame:** creación de un espacio a utilizar dentro de una ventana.

Lógica del software

Anteriormente, se han mencionado las características necesarias para una HMI. En base a estas necesidades se crearon diferentes módulos que suplen cada una de estas necesidades. Los módulos son: (a) alarmas digitales, (b) cambio de nombre, (c) control, (d) gestión de archivos. Estos 4 módulos se inician junto con el programa como se muestra en la figura 6

Rudy Dzul Ramírez 19/5/14 22:34
Comentario [7]: Viste que aquí sí lo acentuaste?

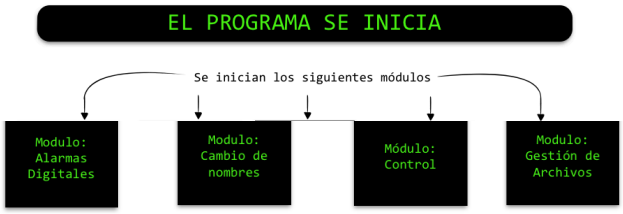

Figura 6 Diagrama de flujo del software

Descripción de los módulos

Módulo Alarmas Digitales

En este modulo, se muestra un panel sinóptico, el cual contiene 5 alarmas digitales. Estas alarmas muestran lo que esta sucediendo en los pines gpio0 – gpio4 de la tarjeta PCduino. Cuando no hay ninguna alarma activa, se muestran los cuadros en color verde, indicando que todo esta bien. Cuando se

detona una alarma los cuadros verdes cambian a color rojo. Los nombres de las alarmas son configurables por el usuario desde el modulo cambio de nombres.

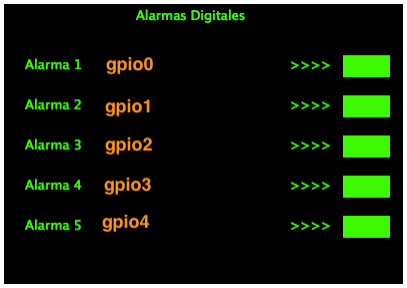

Figura 7 Modulo de alarmas digitales

La lógica de este modulo inicia leyendo los pines gpio0-gpio4, mientras los va leyendo por medio de un if-elif compara si lee un valor 1, esto activa un método llamado set_color_red(num) y se le pasa como parámetro el numero de la alarma, este método cambia de color verde a rojo la alarma donde se detectó el valor 1. Este procedimiento lo podemos ver en el diagrama de flujo de la figura 8.

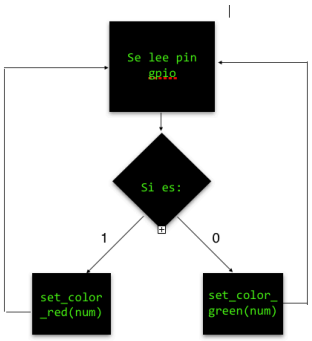

Figura 8 Diagrama de flujo módulo de alarmas digitales

Módulo cambio de nombre

Este módulo es para la configuración del sistema. Aquí el usuario puede darle nombre a las alarmas digitales. El usuario selecciona una alarma de la lista de alarmas, y en el campo de texto escribe el nombre que le quiere dar a la alarma seleccionada, presiona el botón aceptar y el nombre de la alarma seleccionada se cambiará. Podemos ver el funcionamiento de este módulo en el cuadro de flujo de la figura 9.

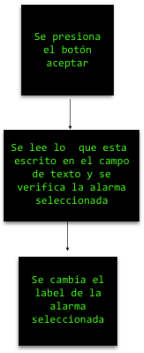

Figura 9 Diagrama de flujo del módulo cambio de nombre

Módulo de Control

Este módulo tiene la capacidad de enviar un pulso desde la UTR, que puede controlar un dispositivo externo. Su salida es de 12volts, con este voltaje se puede encender cualquier dispositivo que requiera este voltaje, o activar un relevador que su bobina sea de 12 volts, para poder encender algún dispositivo mas grande.

Figura 10 Modulo de control

Cuando se da click en el botón cerrar, el switch dejará pasar el voltaje de 12v a las terminales de salida y el módulo cambiara como en la Figura 11.

Figura 11 Modulo de control con el switch cerrado

Este funcionamiento está descrito en el diagrama de flujo de la figura 12.

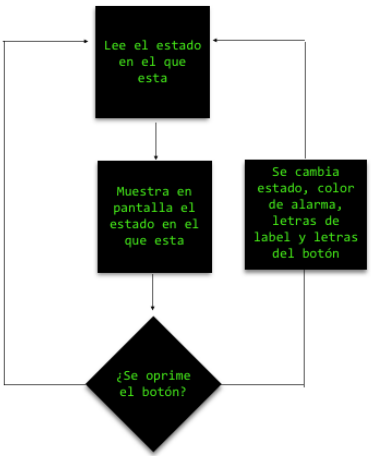

Figura 12 Diagrama de flujo del modulo de control

Módulo gestión de archivos

Este módulo es el más complejo, debido a que realizamos escritura en un archivo desde el programa. Desde que se inicia el programa, este módulo empieza a tomar lectura de todas las alarmas, estas son escritas en un archivo, el cual se crea al momento de iniciar el programa. El nombre del archivo es determinado por la fecha y la hora en que el programa se inicio. Por ejemplo, si el programa se inicio el 23 de abril del 2014 a las 10:25 el nombre del archivo será "Dia_04-23-14_Start_time_10-25-04_log.txt"

Donde 04 es el mes de abril, 23 es el día, 14 es el año 2014, después de Start_time_ encontramos un 10 que es la hora, 25 son los minutos, y 04 son los segundos. En este archivo se van guardando el estado de todas las alarmas incluyendo el estado del switch, cada segundo se hace una nueva lectura y escritura de las alarmas. El funcionamiento de este módulo esta descrito por el diagrama de flujo en la figura 13.

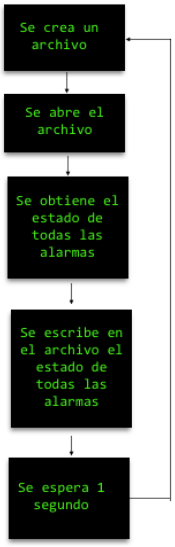

Figura 13 Diagrama de flujo modulo de gestión de archivos

Prueba de Interfaz gráfica en Cubieboard

Mientras se realizaba la interfaz grafica, se realizaban pruebas en la Cubieboard, para ver como funcionaba. Debido a la durabilidad de la Cubieboard, no se prosiguió con las pruebas en esta tarjeta.

Tarjeta embebida PC-Duino

La tarjeta PC-Duino, fue la segunda opción de tarjetas embebidas que adquirimos. Esta cuenta con las siguientes características: (a) procesador ARM Cortex A8 a 1GHz, (b) memoria RAM de 1 GB, (c) almacenamiento interno de 2GB, (d) salida de video HDMI, (e) Ubuntu 12.04 por default, (f) conexión a internet alámbrica, (g) puertos: UART, ADC, PWM, GPIO, I2C y SPI.

Gracias a la decisión tomada de usar Python como lenguaje de programación. La interfaz gráfica hecha para la Cubieboard, se puede usar también en la tarjeta PC-Duino, aunque con capacidades menores a la Cubieboard, el programa funciona también en la PC-Duino.

Pruebas de software

Para realizar las pruebas del software, se utilizaron circuitos sencillos construidos en protoboards. Utilizando las fuentes de voltaje que la PC-Duino tiene incluidas. En la figura 14, se puede apreciar el circuito utilizado para la prueba de las entradas digitales, este circuito nos entregaba un voltaje de 3.3 volts para poder visualizar en el software creado, que se detectaba este pulso de entrada digital.

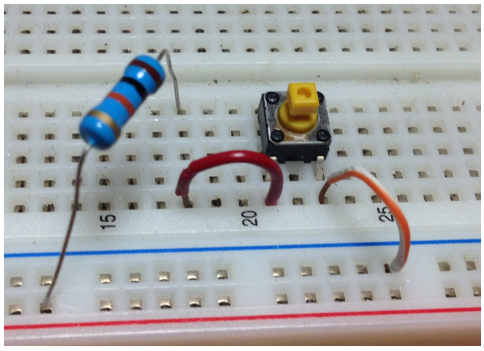

Figura 14 Circuito de prueba entrada digital

En la figura 15 se puede ver un circuito utilizado para probar las salidas digitales, se aprecia un transistor 2N3904, que realiza el rol de switch para dejar pasar un voltaje mas grande que pueda activar los relevadores de salida.

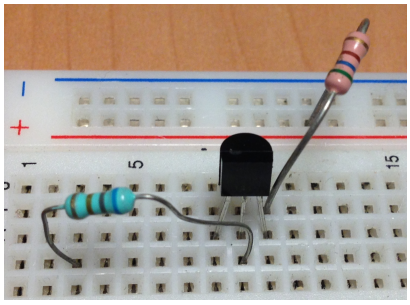

Figura 15 Circuito de prueba para salidas digitales

El software supero las primeras pruebas con éxito, pero todavía faltan las pruebas hechas con el prototipo que se creo simultáneamente a este software.

Prototipo de hardware

La tarjeta embebida PC-Duino, necesita un circuito el cual adapte el voltaje recibido por los sensores a un voltaje que no dañe la tarjeta embebida, de esta forma la tarjeta embebida puede manejar esta información y mostrarla en la interfaz gráfica de usuario.

Este circuito debe ser capaz de tanto de recibir como de enviar un voltaje de 120 VDC (Volts de Corriente Directa). El prototipo construido maneja un voltaje de 12 volts como entrada y 120 VAC (Volts de Corriente Alterna) como salida.

Diseño del prototipo

Para diseñar el circuito se utilizó el software Proteus Professional 8, que cuenta con dos herramientas principales ISIS, que es un simulador de circuitos, y la herramienta ARES, que es un diseñador de PCB (Print Board Circuit). Para el prototipo se utilizo el siguiente material: (a) placa de cobre 15x20 cm, (b) placa de cobre 10 x 5 cm, (c) 5 Resistencias 10 KΩ, (d) 6 diodos 1N4001, (e) Relevador RAS-0510, (f) 6 Relevadores KUP-11D15-12, (g) 2 Potenciómetro de 5KΩ, (h) 80 headers DIL, (i) kit Headers para Arduino, (j)2 Terminales de 8 entradas tipo Tblocks, (k) cable IDE de 40 Pines, (l) Bloques de lego (utilizados para el chasis).

Descripción del circuito

En la figura 16, encontramos las partes del circuito enumeradas de la siguiente forma: (a) relevadores de alarmas, (b) Relevador de control, (c) diodo de protección, (d) resistencias pull up, (e) header conector con PC-duino, (f) terminales de entradas.

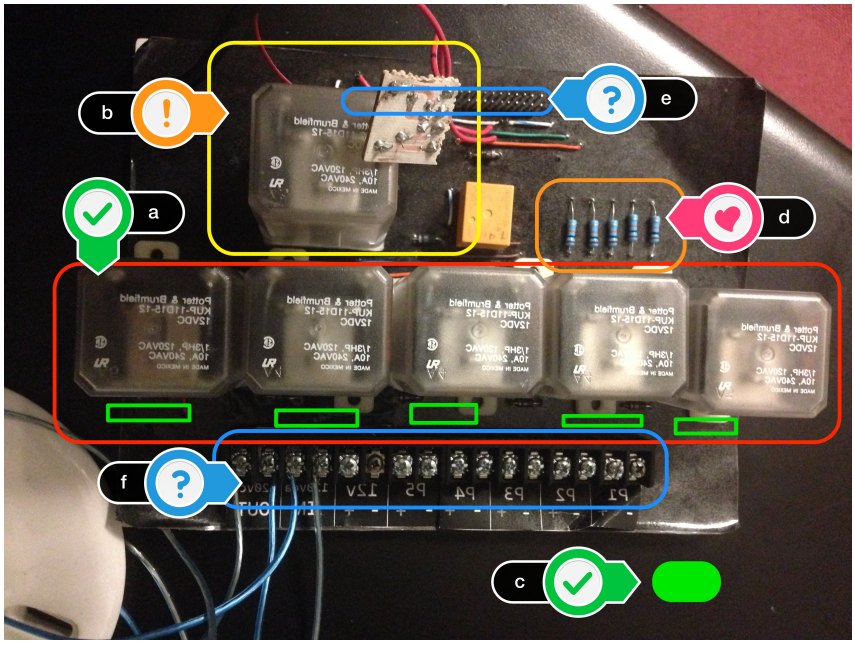

Figura 16 Descripción del prototipo

Funcionamiento

A continuación en las figuras 17 y 18 se muestran los diagramas esquemáticos bases para nuestro circuito los cuales se explican a continuación. Para realizar estos diagramas esquemáticos se utilizó la herramienta Multisim 11, el cual también sirvió como simulador de circuitos.

Entrada GPIO

En la figura 17 la fuente de voltaje de corriente directa (VDC), esta simulando un sensor, esta fuente polariza la bobina del relevador KUP-11D15-12, el cual cierra sus terminales, y deja pasar el común de 3.3v a la entrada del GPIO. La configuración de la resistencia, es pull up, esto quiere decir que cuando el común del voltaje de 3.3v está en contacto con la entrada GPIO, la resistencia genera un pulso alto, de 3.3v, un valor 1 en la entrada GPIO. Para proteger los circuitos digitales, se coloco

un diodo de protección, conectado en paralelo a la bobina. (A. Cuenca, 2002). Este circuito es utilizado en cada entrada digital del prototipo.

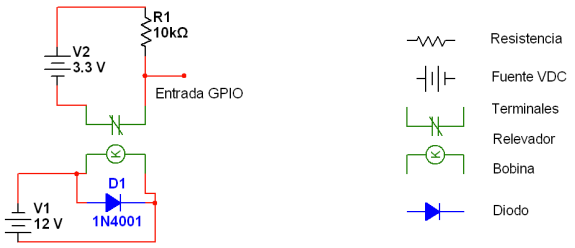

Figura 17 Diagrama esquemático entrada digital en el puerto GPIO

Salida GPIO

La salida en el puerto GPIO, va a generar una salida de 120 VCA, la cual simula un actuador que se activa con un voltaje de 120VDC. Para realizar esta operación es necesario utilizar 2 relevadores. El primero en activarse será el relevador de 5v RAS-0510, este relevador se activara por medio del circuito con el transistor 2N3904, se enviará el pulso de control desde la tarjeta PC-Duino a la base del transistor 2N3904 el cual genera un corto entre sus terminales colector y emisor, de esta forma polariza la bobina del relevador 5v RAS-0510, a su vez, este relevador polariza el relevador KUP-11D15-12 para dejar pasar el voltaje de 120VAC.

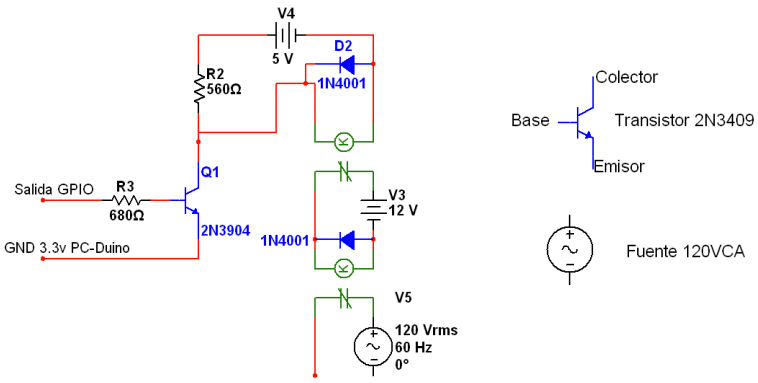

Figura 18 Diagrama esquemático de las salidas de los puertos GPIO

Descripción de los símbolos

A continuación se describen los símbolos utilizados tanto en el diagrama de bloques del circuito, como en el circuito impreso, es decir el circuito físico, estos dos fueron realizados con la herramienta Proteus 8 Professional.

Relevador KUP – 11D15-12

El relevador KUP-11-D15-12 cuenta con 2 polos, cada polo tiene sus terminales Normalmente cerrado y normalmente abierto, y sus dos terminales de bobina.

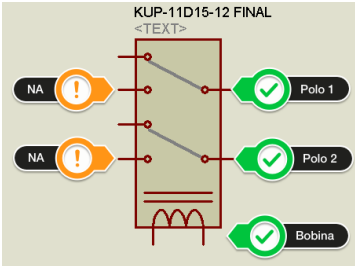

Figura 19 Diagrama de Bloques de un Relevador, no se especifica la terminal normalmente cerrada por que se omite su uso para este proyecto.

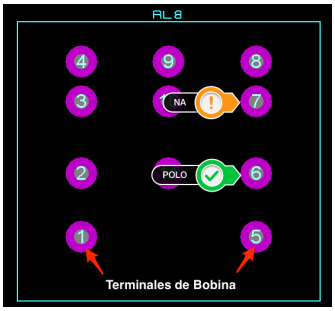

Figura 20 Footprint del relevador KUP-11D15-12

Terminales de entrada y salida

La terminal de entrada y salida cuenta con 8 pines. En esta terminal se conectan las alarmas, en el pin 8 se conecta el negativo de la alarma 1, en la terminal 7 el positivo de la alarma 1. En la tabla 2 se muestra como se conectan los relevadores y las alarmas en la terminal de sensores. El diagrama de bloques se muestra en la figura 21.

Tabla 2 Conexión de los relevadores y alarmas en la terminal de entrada y salida

Dispositivo	Alarma Digital	PINES	Terminal de Sensores
Relevador 1	1	8 y 7	1
Relevador 2	2	6 y 5	1
Relevador 3	3	4 y 3	1
Relevador 4	4	2 y 1	1
Relevador 5	5	8 y 7	2
12 VDC	-	6 y5	2
120VAC IN	-	4 y 3	2
120 VAC OUT	-	2 y 1	2

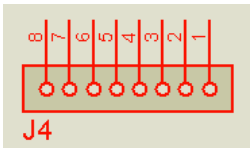

Figura 21 Diagrama de bloques terminales de entrada y salida

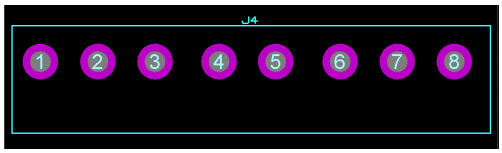

Figura 22 Footprint de las terminales de entrada y salida

Conector PC-DUINO – Interfaz hardware

El conector que comunica el PC-Duino con la interfaz hardware cuenta con 40 pines, de los cuales solo 30 son utilizados. Podremos ver en la tabla tal, las conexiones que se hicieron entre el PC-Duino y el prototipo.

Tabla 3 Conexión de pines del conector 40 DIL y los pines del PC-Duino

Pin del Conector	PIN del PC-Duino
1,3,5,7,9,11,13	Gpio0 –Gpio6
34	- 5v (GND)
32	- 3.3v (GND)
30	+5v
28	+3.3v

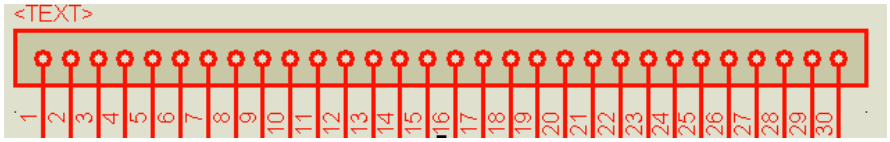

Figura 23 Diagrama de bloques conector PC-Duino – Interfaz Hardware

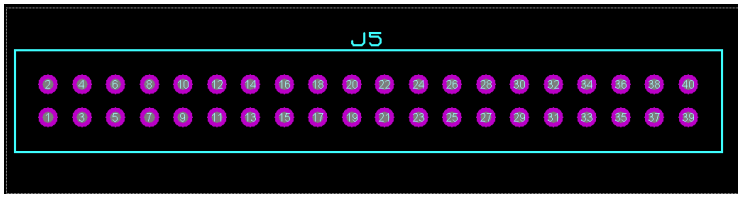

Figura 24 Footprint conector PC- Duino Interfaz Hardware

Diagrama de bloques

Para diseñar la placa física de este circuito, es necesario crear un diagrama de bloques en el software Proteus 8 Profesional. Anteriormente se explicó la simbología del diagrama de bloques y se mostraron las footprints de cada componente. A continuación en la figura 25, el diagrama de bloques de todo el circuito. Donde podemos apreciar el circuito de entrada de los puertos GPIO, y el circuito de salida de los puertos GPIO.

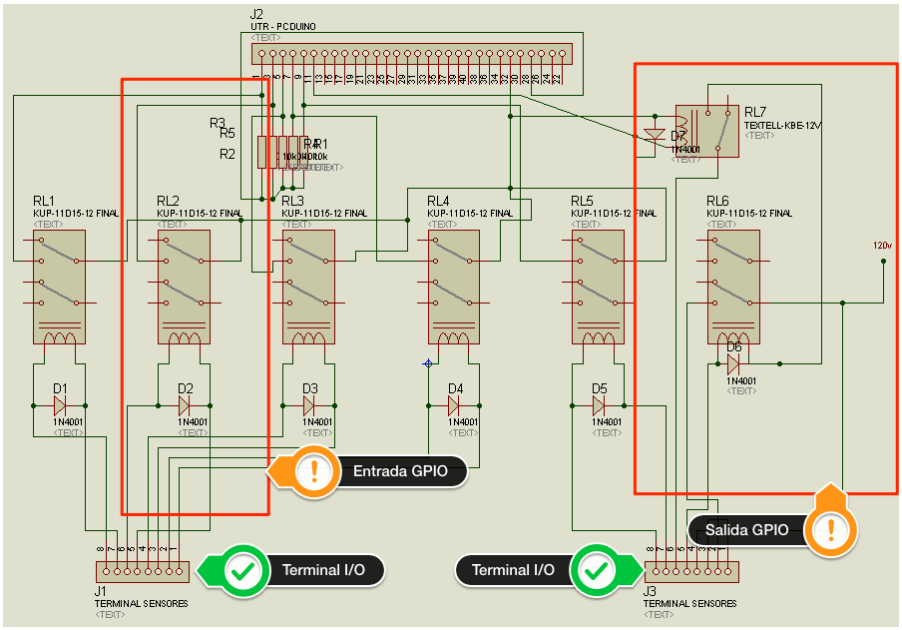

Figura 25 Diagrama de bloques del prototipo

Como se puede apreciar en el diagrama, el circuito de entrada de GPIO se repite 5 veces, para las 5 entradas digitales que se tienen en el prototipo, y el circuito de salida GPIO se presenta 1 sola vez ya que se presenta solo una salida digital.

Circuito Físico

Al crear el diagrama de bloques se tiene definido lo que se requiere que el circuito realice. Incluso con esta herramienta de software se puede simular el circuito para poder analizar su funcionamiento. Para realizar una placa, física, en la cual podamos conectar nuestros componentes físicos y poder utilizar el circuito se utiliza la herramienta ARES, incluida en el mismo software Proteus 8 professional. En esta se incluyen todos los componentes utilizados en el diagrama de bloques. Se van insertando uno por uno y se van uniendo por pistas de cobre.

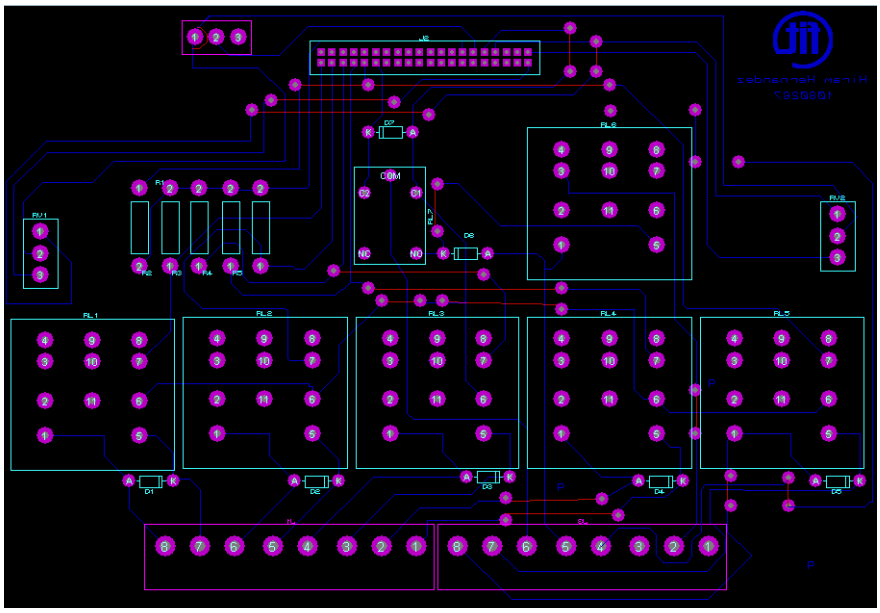

Figura 26 Footprint del circuito del prototipo

En la figura anterior, las líneas color azul rey, representan las pistas de cobre del circuito físico, los círculos morados, representan las entradas de los pines de cada componente electrónico, las líneas rojas, representan los puentes utilizados, anteriormente ya se había mostrado el footprint de cada componente los cuales podemos apreciar en esta figura.

Pruebas del prototipo

Prueba de funcionamiento

Para realizar estas pruebas se realizo una serie de conexiones con relevadores activados por un arduino. Esto para simular varias entradas en las terminales de sensores. Durante 30 minutos se activaron y desactivaron las alarmas, en intervalos de 1 segundo. Dando como resultado que todas las alarmas se registraron en el log de eventos correctamente.

Prueba de retardo en el tiempo

Para esta prueba se genero un log de una alarma el cual cada línea representa 100 milisegundos (ms), de esta forma cada 10 segundo se activaba una alarma, después de que la alarma es activada transcurren 5 segundos y la alarma se desactiva. Del log de eventos se obtuvo la tabla 4, tomando 10 muestras de activación:

Tabla 4 Activación y desactivación de una alarma y sus respectivos retardos en ms

Hora	Retardo activado en ms	Retardo desactivado en ms
16:07:10	100	100
16:07:20	0	100
16:07:30	100	100
16:07:40	100	0
16:07:50	100	0
16:08:00	0	0
16:08:10	0	0
16:08:20	0	100
16:08:30	0	0
16:08:40	0	0

En la tabla 4 encontramos que existe un retardo mínimo, tan pequeño que es despreciable, pero aun así se profundizo para ver el retardo en 3 pruebas, en un log donde cada línea equivale a 1 ms. La tabla 5 nos muestra la segunda prueba que se realizo.

Tabla 5 Prueba de retardo, donde cada línea equivale 1 ms

Hora	linea de activacion	B
17:03:20	1	16
17:03:30	-12	29
17:03:40	-21	28

La prueba comenzó en el primer milisegundo del segundo 17:03:20. La alarma se desactiva a los 5 segundos, por lo que la desactivación ocurrirá en el segundo 17:03:25. La columna B es la cantidad de milisegundos anticipados antes del segundo donde la alarma se debía desactivar. En otras palabras la celda B1, indica que la alarma se desactivó 16 ms antes. De esta forma vemos que el retardo es muy despreciable, y que no afecta a la aplicación de estas alarmas.

Prueba de Switch

Se realizaron pruebas de tiempo al switch, el tiempo en que tarda la señal en llegar al prototipo, para realizar esta prueba se tomó video, del prototipo encendiendo un led rojo y uno verde, y los clicks que se daban en la interfaz gráfica. Este video fue abierto en un software para edición de video, llamado Final Cut Pro, para poder acceder al zoom de tiempo y ver exactamente el momento en que se presionaba el botón en el módulo de control y el momento en que encendían los leds físicamente.

La interfaz gráfica está diseñada con botones con configuración realease, esto quiere decir que cuando el botón es presionado con el mouse, no sucede nada hasta que el botón del mouse se suelta. Teniendo en cuenta esta configuración al momento de revisar el video, nos dimos cuenta que el botón tiene 34 ms de retardo. Ya que teóricamente al soltar el botón, los leds deben encender, pero no fue hasta 34 ms después que los leds encendían. Es un tiempo demasiado pequeño, y para el prototipo es despreciable.

CONCLUSIÓN

Conclusiones

EL prototipo diseñado, funciono con éxito. Las alarmas son detectadas por el HMI, al igual que el pulso de control es enviado correctamente hasta el prototipo físico. Se comprobó, que debido al uso de software y hardware libre, hemos podido modificar tanto el prototipo como el software que lo controla, de esta forma, podemos realizar los cambios necesarios para la interfaz real, y adaptarla a la necesidad del usuario. El prototipo construido aporta la siguiente información a la investigación mayor:

1. La forma de construir un prototipo de interfaz entre la UTR y los dispositivos de la red eléctrica, es muy similar a la forma de construir la interfaz real.
2. Se requiere de otros relevadores que soporten la cantidad de energía que envían los sensores de los dispositivos de la red eléctrica.
3. Ahora sabemos, que Python, es un lenguaje de programación que se puede aplicar perfectamente para controlar la interfaz prototipo y la interfaz real.
4. El hardware y software libre que se utiliza, reducirá los costos, de la UTR.
5. La selección de CPU para la UTR, fueron las tarjetas embebidas PC-Duino y Cubieboard, quedando como principal tarjeta PC-Duino, debido a la poca durabilidad de la tarjeta Cubieboard.
6. Se creó un shield para controlar la PC Duino, desde un cable IDE de 40 pines.
7. El prototipo físico creado, puede ser utilizado para realizar pruebas en el futuro.

Reflexión

El prototipo construido de la interfaz física entre UTR y los dispositivos de la red eléctrica, y el software que controla y monitorea este prototipo, ha brindado un gran avance en el proyecto de la UTR. Ya se tiene una base sobre la cual trabajar, e ir madurando el proyecto. Se encontró información importante para aplicar en el proyecto mayor. En lo personal este proyecto sin darme cuenta ha realizado uno de los sueño que he tenido desde pequeño, controlar algo físico desde la computadora. En mi vida profesional este proyecto a ayudado a madurar mis conocimientos sobre electrónica digital, y programación.

Recomendaciones

Gracias a la información que este proyecto ha brindado, se hacen las siguientes recomendaciones, tanto para mejorar el prototipo, como para aplicar en la interfaz real.

1. Seguir utilizando Python como lenguaje de programación, además de ser un lenguaje multiplataforma, Python puede ser utilizado en la web, así se puede crear una interfaz gráfica en la web, y controlar y monitorizar la UTR desde la nube.
2. Para los relevadores de la interfaz real se recomienda utilizar el relevador KUP-14D15-110
3. Se recomienda altamente el uso del transistor 2N3904 para funciones de salidas digitales, en las salidas del PC-Duino.
4. Utilizar una fuente de voltaje externa al PC-Duino, de esta forma evitar la pérdida de energía en la tarjeta PC-Duino.

Futuros Aportes

Para mejorar este prototipo, es importante incluir alarmas análogas, las cuales nos darán una medición exacta o muy aproximada de los sensores a los cuales se conecta. Para esto es necesario otro

hilo de investigación, que se encargue de resolver las preguntas, ¿Cómo detectar el voltaje de los sensores?, ¿Cómo interpretar el voltaje con la tarjeta PC- Duino mediante un software?

REFERENCIAS

Rudy Dzul Ramírez 19/5/14 22:53
Comentario [8]: Sólo espero que este de acuerdo al formato APA.

A. Cuenca, A. (2002). Sistematización de un experimento de difraccion de luz. *Revista Colombiana de Física*, 123 - 127.

David Chacon, O. D. (2001). *Supervisión y control de procesos*.

Duque, R. G. (2010). *Python para todos*.

Fernández, H. (2012). *Open Hardware*.

Francis Enejo Idachaba, A. O. (Agosto de 2012). Review of Remote Terminal Unit (RTU) and Gateways for Digital Oilfield delpoyments. *IJACSA (International Journal of Advanced Computer Science and Applications, 3*(8), 157-170.

Guerrero, S. A. (Septiembre de 2007). Diseño e implementacion de una plataforma de desarrollo, aplicada a un prototipo de maquina fresadora CNC. Santiago de Chile, Chile.

Instituto de Investigaciones Electricas (IIE). (Agosto de 1981). TRIIE, La unidad terminal remota diseñada por el IIE. *Boletin IIE, 5*(8), 284-291.

Jaume Romangoza Cabús, D. G. (2004). *Miniproyecto Automatizacion Industrial*. Escola Politècnica Superior d'Enginyeria de Vilanova i la Geltrú.

Penin, A. R. (2012). Sistemas SCADA. En A. R. Penin, & MARCOCOMBO (Ed.), *Sistemas SCADA* (3ª Edicion ed.).

Rossum, G. V. (2000). *Guia de aprendizaje de Python*.

Smith, H. L. (April de 2010). A brief History of Electric Utility Automation Systems. *Electric Energy T&D Magazine, 14*, 39-44.

Stallman, R. (2004). *Software libre para una sociedad libre*. Madrid: Traficantes de sueños.