MONITOREO A TRAVÉS DE SISTEMAS EMBEBIDOS A PROCESOS INDUSTRIALES

Equipo de Trabajo

Asesores

Germán Guzmán

Edwin Fernando Giraldo

Integrantes

Jorge Mario Ríos Herrera 200110109010

Juan Carlos Ríos Herrera 200410093010

Marcos Barrientos Botero 200510047010

Historia

Versión	Fecha	Descripción o Cambios introducidos
2.0	07/11/2011	Entrega 2 Proyecto de grado

MONITOREO A TRAVÉS DE SISTEMAS EMBEBIDOS A PROCESOS INDUSTRIALES

Jorge Mario Ríos Herrera

200110109010

Jriosher@eafit.edu.co

034-451-8126

300-288-6764

Juan Carlos Ríos Herrera

200410093010

Jrioshe1@eafit.edu.co

034-451-8126

301-7778657

Marcos Barrientos Botero

200510047010

Mbarrie2@eafit.edu.co

034-268-5771

311-358-1383

Germán Alonso Guzmán Rivera

gguzman@eafit.edu.co

034-261-9291

Edwin Fernando Giraldo

edgiral@eafit.edu.co

034-2619500

UNIVERSIDAD EAFIT

ESCUELA DE INGENIERIA

DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS

2011 – 2

MONITOREO A TRAVÉS DE SISTEMAS EMBEBIDOS A PROCESOS INDUSTRIALES

Trabajo de grado presentado como requisito para optar por el título de Ingeniero de Sistemas

Equipo de Trabajo

Asesores

Germán Guzmán

Edwin Fernando Giraldo

Integrantes

Jorge Mario Ríos Herrera 200110109010

Juan Carlos Ríos Herrera 200410093010

Marcos Barrientos Botero 200510047010

UNIVERSIDAD EAFIT

ESCUELA DE INGENIERIA

DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS

2011 – 2

Nota de aceptación

_____

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

TABLA DE CONTENIDO

LISTA DE IMÁGENES

PAG

1. INTRODUCCIÓN

Actualmente las empresas se dedican a invertir mucho dinero en la automatización de tareas de producción, esto lo hace por medio de maquinaria industrial cada vez más tecnológica, eficiente y moderna.

Por medio de este proyecto de grado, se pretende brindar a las empresas una solución para el rendimiento de la maquinaria industrial, ya que actualmente como se mencionó, las empresas invierten mucho dinero en este tipo de productos y no realizan un respectivo monitoreo acerca de las mismas y en ocasiones estas son subutilizadas.

La finalidad de nuestro proyecto es realizar un prototipo que comunique sistemas embebidos con sensores conectados a las máquinas y que nos ayude con el monitoreo de la maquinaria industrial, esto con el fin de sacarle un mayor provecho con respecto a la producción de estas.

2. DEFINICIÓN DEL PROBLEMA

Uno de los problemas que se tienen hoy en día, es que la mayoría las empresas industriales no tienen mecanismos de monitoreo sobre la maquinaria industrial que tienen en sus instalaciones. Por este motivo nos enfocamos en este proyecto ya que lo visionamos como solución a los problemas de productividad que se tienen en estas empresas debido a no tener control de sus máquinas.

El enfoque que le daremos al proyecto, para solucionar esta clase de problemas, lo definimos globalmente de la siguiente forma:

Se desea implementar una tecnología, para el incremento de la productividad de la maquinaria industrial. Para la implementación del mismo se utilizaran placas CPU compact PCI o SBC con sistemas embebidos y un prototipo de un software desarrollado a la medida que nos permita realizar el monitoreo y arrojar información necesaria a diferentes usuarios (operarios, coordinadores, directivos) con el fin de cumplir el objetivo principal (aumento de la productividad).

La información que es arrojada por el sistema como se mencionó anteriormente, le llega a diferentes tipos de usuarios, los cuales la pueden utilizar de diversas formas. Estos usuarios y la diferente información que reciben son:

- **Operario:** El sistema de acuerdo a diferentes datos arrojados por la máquina, le mostrara a éste si están siendo productivos o no, esto con el fin de que el operario siempre sienta si está haciendo bien el trabajo o no. En caso de que la maquina se encuentre en mantenimiento el operario podrá indicarle al sistema , para que este no lo valore como no productivo
- **Coordinadores:** La información que le arrojara el sistema a este usuario es, primero si el operario está siendo productivo o no, al hacerlo en tiempo real se garantiza que se puedan tomar decisiones al instante, de acuerdo a dicho operario; segundo, muestra la periodicidad con que la maquina entra a mantenimiento, esto con el fin de tomar decisiones sobre dicha maquina o futuras maquinas del mismo tipo que se tengan presupuestado comprar.

- **Directivos:** La información que este tipo de usuario recibe, le ayuda a captar como es la producción de la máquina y la productividad del operario, esto con el fin de tomar decisiones importantes, de acuerdo a dicha información arrojada.

3. JUSTIFICACIÓN

Este trabajo se encuentra bajo el marco teórico de las ciencias de la computación e informática y pretende enfocarse en el uso de sistemas embebidos (SBC: Single Board Computer) que son pequeñas computadoras, que poseen microprocesador, memoria rom y ram, entradas y salidas y muchas de las características de un computador funcional en una sola placa base (Board).

Actualmente este tipo de arquitectura es utilizada en entornos industriales para realizar la comunicación entre las interfaces de las maquinas, desde un solo lugar o sistema. La importancia de este proyecto es realizar una comunicación entre la maquinaria y los directivos de la empresa, esto con el fin de que se tenga control sobre las mismas con el fin de mejorar el rendimiento.

También es importante resaltar la importancia que tiene el proyecto de acuerdo al conocimiento adquirido durante toda la carrera, ya que se va hacer una integración entre hardware y software y aplicar un poco la visión administrativa gerencial.

Las ventajas de realizar este proyecto son, en primer lugar aprovechar la información arrojada por la maquinaria para la tomar decisiones de tipo administrativas; en segundo lugar es darle un mejor aprovechamiento a este tipo de sistemas embebidos, ya que la mayoría de la industria colombiana las utiliza pero solamente como un controlador gráfico de las máquinas.

4. OBJETIVOS

4.1. OBJETIVOS GENERALES

Diseñar e implementar un prototipo de software que se ejecuta bajo sistemas embebidos (CPU CompactPSC o SBC), con el fin de hacer el monitoreo de procesos industriales y así, mejorar de la productividad de la maquinaria industrial.

4.2. OBJETIVOS ESPECIFICOS

- Estudiar los diferentes protocolos empleados en la industria (MODBUS bajo tcp/ip, MODBUS bajo rs232, CANBUS) para acoplar el sistema embebido a la maquinaria y así poder extraer de la mejor manera las variables objeto de estudio.
- Aprender a manejar algunas de las tecnologías asociadas a los artefactos de control y sistemas embebidos (SBC- Single-board computer).
- Implementar el sistema en un proceso controlado en una empresa ubicada en Rio Negro Antioquia, esto con el fin de poder estar monitoreando el proceso y revisar los datos arrojados por el sistema.
- Visualizar los estados del proceso controlado en el sistema embebido para poder realizar un seguimiento más detallado de los posibles paros y determinar cuáles hacen que la productividad disminuya
- Desarrollar mediante librerías QT V. 4.5 (Nokia), una serie de interfaces de usuarios, para que la visualización y manipulación de los datos de la maquina se desarrollen de una forma amigable para cada usuario.
- Proveer a la industria una forma adquisición de datos para realizar informes más relevantes y necesarios para la toma de decisiones durante el proceso.

5. ALCANCE DEL PROYECTO

Este proyecto consta de un alcance que es la implementación de sistemas embebidos en el monitoreo procesos industriales, el cual lo vamos a realizar con el desarrollo de un prototipo que implementaremos e instalaremos en una tecnología especifica (placa CPU CompactPCI o SBC).

Se entregara un documento de instalación y un manual de usuario que guiara en el proceso de cómo utilizar el software con las placas CPU CompactPCI o SBC.

6. IMPORTANCIA

Este proyecto se centra en la implementación de un prototipo de software que comunica las maquinas industriales con uno o varios sistemas embebidos que contiene una tecnología especifica (placa CPU CompactPCI o SBC), además es de gran importancia ya que tiene varias tecnologías que no han sido exploradas porque son novedosas en nuestro entorno y requieren un alto grado de investigación. Así mismo la importancia radica en la posibilidad de medir el nivel de desempeño de la máquina y el operario para poder mantener un nivel alto en la productividad del proceso en cuestión.

7. METODOLOGIA SUGERIDA

Este proyecto se va a realizar en diferentes etapas secuenciales, las cuales son:

- **Etapa 1:** En esta primera etapa, se realizara todo tipo de consultas e investigaciones acerca de las placas CPU CompactPCI o SBC, dispositivos de control y maquinaria industrial, todo esto con el fin de tener una documentación adecuada para el desarrollo del proyecto.
- **Etapa 2:** En esta etapa lo que se desea realizar, es analizar cada una de las necesidades que tendrían los posibles usuarios del sistema. Específicamente en esta etapa, se desea analizar y refinar cada uno de los requisitos del sistema, tanto funcionales como no funcionales.
- **Etapa 3:** En esta etapa se va a realizar el diseño de la aplicación utilizando UML. Se definirán los modelos y se refinarán los mismos a medida que se avance en ella. En esta etapa se van a presentar las interfaces de usuario.
- **Etapa 4:** En esta etapa se va a realizar la implementación del prototipo en los sistemas embebidos que contienen las placas CPU CompactPCI o SBC.
- **Etapa 5:** Finalmente en esta etapa se van a realizar cada una de las pruebas de los requisitos funcionales y no funcionales del prototipo

8. MARCO TEORICO.

A continuación se presentan diferentes conceptos que son importantes para contextualizar al lector en las generalidades del proyecto.

8.1. PROTOCOLO MODBUS

Es un protocolo desarrollado para la comunicación basado en la arquitectura Maestro/Esclavo o cliente/Servidor [Imagen 1]. Este protocolo se sitúa en el nivel 7 del modelo OSI (nivel de aplicación). Actualmente es uno de los protocolos más utilizados para establecer comunicaciones entre diferentes dispositivos electrónicos industriales.

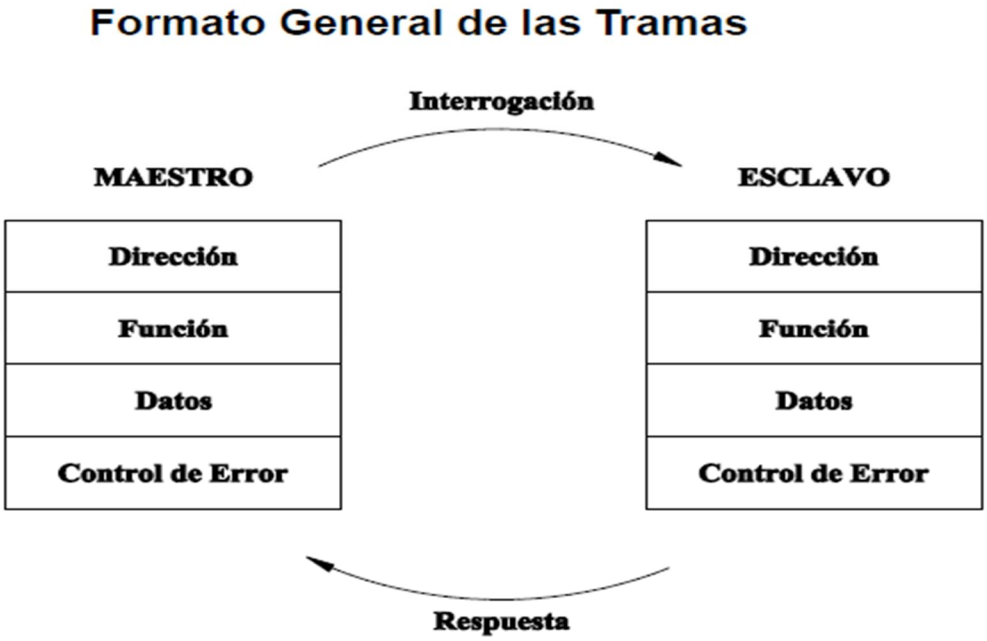


Imagen 1 [Arquitectura MODBUS]

8.1.1. FORMATO ASCII

El protocolo trabaja con un formato de comunicación que contiene formato de trama, secuencias y control de errores. Este primero (formato de trama) tiene dos maneras de implementarse, por medio de del formato ASCII y RUT. Para el proyecto se trabajó con el formato ASCII [Imagen 2].

Modo ASCII

Comienzo de Trama	Dirección	Función	Datos	Control de Errores	Fin de Trama
:	2 bytes	2 bytes	N x 2 bytes	2 bytes	CR + LF

Imagen 2 [Formato Trama ASCII]

8.1.2. COMPONENTES DE LA TRAMA

La trama en el formato ASCII [Imagen 2] está compuesta por:

- **Comienzo de la trama:** Con este indicador se especifica al dispositivo (en este caso un PLC), que va a empezar trama
- **Dirección:** Con este indicador identificamos para cual de los dispositivos va la trama, la dirección es fija y única, esta se ubica en el rango entre 1 y 247. La dirección cero (0) es reservada para enviar mensajes a todos los dispositivos sin esperar respuesta alguna.
- **Función:** Esta identifica que tipo de operación se va a realizar en el dispositivo [Imagen 3].

Código	Acción	Significado
01	Leer Bobinas (0:xxxx)	Obtiene el estado actual ON/OFF de un grupo de bobinas lógicas.
02	Leer Entradas (1:xxxx)	Obtiene el estado actual ON/OFF de un grupo de entradas lógicas.
03	Leer Registros (4:xxxx)	Obtiene el valor binario de uno o más registros de almacenamiento.
04	Leer Registros (3:xxxx)	Obtiene el valor binario de uno o más registros de entrada.
05	Escribir Bobina (0:xxxx)	Fuerza el estado de una bobina.
06	Escribir Registro (4:xxxx)	Escribe el valor binario de un registro de almacenamiento.
15	Escribir Bobinas (0:xxxx)	Fuerza el estado de un grupo de bobinas.
16	Escribir Registros (4:xxxx)	Escribe el valor binario de un grupo de registros de almacenamiento.

Imagen 3 [Funciones]

- **Datos o Subfunciones:** El comando que se debe ingresar para ejecutar la función, por ejemplo en caso de lectura la dirección de memoria donde se encuentra almacenado el registro y en caso de escritura la dirección en memoria donde se desea guardar el registro y los datos a guardar.
- **Control de errores (CheckSum):** Por medio de este método se desea proteger que los datos sean los correctos, verificando que estos no hayan sido corruptos. Para calcular el CheckSum se suman los bytes de la trama en módulos de 256, que es el mismo número pero invertido y se le suma uno (1), luego cuando la trama llega se realiza el mismo procedimiento y el valor resultante se compara con el enviado. Si los datos concuerdan se asumen que llegaron correctos.
- **Fin de la Trama:** Con este identificador especificamos al dispositivo que la trama ha finalizado. Se valida checkSum y final de trama con CRLF (Se refiere a la combinación de dos códigos de control).

8.1.3. RESPUESTA DE ERROR

Cuando el maestro envía una trama para ser procesada por un esclavo, y este último no puede realizar la función requerida, envía una trama de error [Imagen 4]

Dirección	Función	Código de Error	Control de Error
2 bytes	2 bytes	2 bytes	2 bytes

Imagen 4 [Trama de Error]

8.1.4. COMPONENTES DE LA TRAMA DE ERROR

La trama de Error se compone por:

- **Dirección:** Muestra el indicador del dispositivo al cual se había enviado inicialmente la trama
- **Función:** Es el retorno de un valor que indica que ocurrió un error
- Código de Error: Muestra cual fue la causa del error por el cual no se pudo procesar la trama [Imagen 5].

Código	Tipo de Error	Significado
01	Función ilegal	La función recibida no esta permitida en el esclavo.
02	Dirección ilegal	La dirección esta fuera del rango permitido.
03	Dato ilegal	El dato contiene un valor no válido.
04	Falla en el dispositivo	El controlador no responde o ha ocurrido un error.
05	Reconocimiento (ACK)	Se ha aceptado la función y se esta procesando.
06	Ocupado	El mensaje ha sido recibido sin error, pero el dispositivo no puede procesarlo en este momento.
07	Reconocimiento Negativo (NAK)	La función solicitada no puede realizarse en este momento.

Imagen 5 [Códigos de Error]

- **Control de Error:** Igual que en la trama anterior utiliza el CheckSum para validar que la trama llegue correctamente y valida la trama con (CR y LF), esto quiere decir que si la trama finaliza correctamente pero no trae CRLF (Se refiere a la combinación de dos códigos de control) se toma esta como incorrecta.

8.2. LIBRERIAS

8.2.1. QT Nokia

Las librerías QT son una biblioteca multiplataforma utilizadas para el desarrollo de aplicaciones con interfaz gráfica de usuario, estas actualmente cuentan aproximadamente 500 clases, más de 9000 funciones y 500.000 líneas de código, brindando un alto grado de "potencia" aproximándose a lenguajes como C# o Java con la eficiencia de código compilado en C++.

Una de las grandes ventajas que poseen estas librerías es que incluyen soporte para dibujo en 2D, hilos, red, bases de datos, etc. Lo que brinda una gran ventaja en comparación con los otros FrameWorks.

Otras de las grandes ventajas que tienen las librerías QT es que es posible implementar objetos durante el desarrollo, los cuales brindan una ventaja ya que es posible la comunicación y el intercambio de información entre estos objetos.

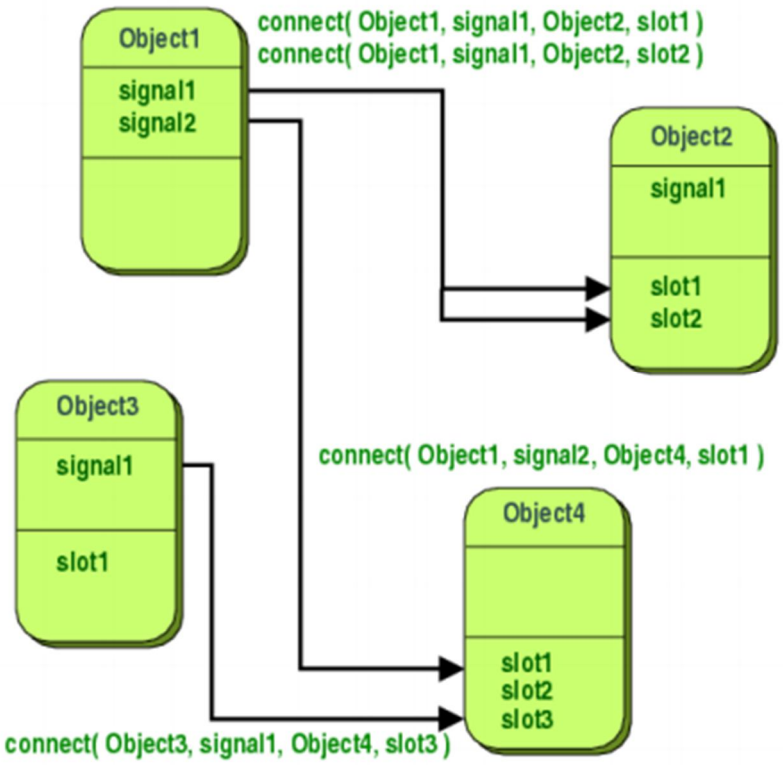


Imagen 6 [Conexión entre objetos]

8.2.2. LIBRERIAS GTK +

GTK+, es un conjunto de bibliotecas multiplataformas, enfocadas a la programación con interfaces de usuario. Inicialmente, fueron creadas únicamente para desarrollar el programa de edición gráfica GIMP de Linux, pero actualmente son muy utilizadas para el desarrollo en los Sistemas Embebidos.

Las bibliotecas GTK+, están soportadas por lenguajes como, C, C++, C#, Java, Ruby, Perl, PHP o Python

GTK + está compuesta por unas series de librerías que son:

- GLib: Biblioteca de bajo nivel, utilizada por GTK+ y GNOME. Proporciona manejo de estructura de datos para C, portabilidad, interfaces para funcionalidades de tiempo de ejecución como ciclos, hilos, carga dinámica o un sistema de objetos.
- GTK. Biblioteca la cual realmente contiene los objetos y funciones para crear la interfaz de usuario. Maneja widgets como ventanas, botones, menús, etiquetas, deslizadores, pestañas, etc.
- GDK. Biblioteca que actúa como intermediario entre gráficos de bajo nivel y gráficos de alto nivel.
- ATK. Biblioteca para crear interfaces con características de una gran accesibilidad muy importante para personas discapacitadas o minusválidas. Pueden usarse utilerías como lupas de aumento, lectores de pantalla, o entradas de datos alternativas al clásico teclado ratón.
- Pango. Biblioteca para el diseño y renderizado de texto, hace hincapié especialmente en la internacionalización. Es el núcleo para manejar las fuentes y el texto de GTK+2.
- Cairo. Biblioteca de renderizado avanzado de controles de aplicación.

8.3. QT CREATOR

QT Creator es un IDE multiplataforma que trabaja bajo librerías QT de una manera sencilla y rápida.

8.3.1. VENTAJAS

- Una de sus principales ventajas es que cuenta con un diseñador de formularios (GUI: Interfaces Gráficas de Usuario) integrado.
- Es posible administrar y crear varios proyectos.
- Soporta diferentes lenguajes (Python, Pascal, PHP, ruby, entre otros).
- Contiene un depurador de proyecto visual.
- Ayuda a la autocompletación de código.
- Resalta los errores en el código.
- Posee mucha documentación ONLINE.

8.3.2. DESVENTAJAS

- La depuración del programa no se puede hacer si no se esta conectado al equipo SBC, pues no hay un simulador para los sistemas embebidos.

8.3.3. SISTEMAS OPERATIVOS QUE SOPORTA QT CREATOR

Como se había mencionado anteriormente este IDE es multiplataforma y se puede encontrar para los siguientes sistemas operativos:

- Linux.
- Mac OSX.
- Windows.
- Windows CE.
- Symbian.
- Maemo

8.4. QPORT

QPort es un plugin para las librerías QT de NOKIA, este permite realizar las conexiones con el puerto serial del computador. La ventaja de este plugin es que permite realizar una comunicación de una manera muy sencilla e inclusive visual; lo único que hay que hacer es introducir los parámetros y se encarga del resto, inclusive muestra gráficamente la conexión.

8.5. SISTEMAS EMBEBIDOS

8.5.1. SBC

La SBC son pequeñas computadoras, que poseen microprocesador, memoria rom y ram, entradas y salidas y muchas de las características de un computador funcional en una sola placa base (Board).

Imagen 7 [SBC TS – 7390]

8.5.2. SBC TS – 7390

Las principales características que estábamos buscando y encontramos en esta fueron:

- Pantalla de 7” Touch Screen (buen tamaño para la visualización del operario).
- Procesador ARM9 200 MHz 64 MB SDRAM 512 MB NAND FLASH.
- Sistema operativo Linux Debian.

- Dos puertos USB 2.0.
- Un puerto SD.
- Un puerto RS 232 (serial comunicador con él PLC).
- Un puerto RS 485.
- Entradas y salidas digitales
- Un puerto Ethernet – 10/100.
- Pantalla con diseño industrial.

8.5.3. SBC BL4S200

La segunda opción examinada para la selección del sistema embebido fue la BL4S200 de la empresa Digi. Las principales características que esta ofrece son:

- Utiliza un procesador Rabbit 5000
- Wifi, ZigBee y ethernet integrado.
- Cinco puertos seriales.
- Ocho entradas y Dos salidas análogas
- Entradas y salidas digitales
- Entradas y salidas avanzadas por medio de software que reducen el tiempo de carga del procesador.
- Pantalla de 3.5 pulgadas, tàctil.

8.6. PLC (Programmable logic controller)

El PLC es un equipo electrónico programable, diseñado para controlar procesos secuenciales en tiempo real, en la mayoría de los casos en un ambiente industrial.

Es por medio de un PLC y un cable RS232 conectado al SBC, que se puede controlar la maquina industrial, esto se hace con el protocolo MODBUS (anteriormente explicado) y también es posible recolectar la información que se captura.

El PLC utilizado actualmente por la maquina es el DVP30EX2 [Imagen 8] de la marca delta.

Imagen 8 [PLC]

9. ANALISIS DE INVESTIGACIÒN

9.1. LIBRERIAS

Luego del análisis de las diferentes librerías para el desarrollo del proyecto, se decidió seleccionar las QT NOKIA, gracias a todas las características que ofrece anteriormente mencionadas y por la facilidad que ofrece el IDE para desarrollar en las mismas brindando muy buenos componentes por ejemplo, para el desarrollo gráfico.

9.2. SISTEMAS EMBEBIDOS

En el momento que se realizó la búsqueda del sistema embebido que cumpliera las características del proyecto, se descubrió que la mayoría de estas ofrecen un gran número de utilidades, que no eran necesarias para el desarrollo del proyecto y que hacían que estas tuvieran un precio más costoso.

Es por esto que la SBC seleccionada fue la TS – 7390 [imagen 7], ya que contiene características muy apropiadas para el desarrollo del proyecto y un costo no muy elevado.

10. CONFIGURACIONES

10.1. INSTALACIONES QT CREATOR

Como el proyecto fue desarrollado en LINUX, la instalación que se llevó a cabo fue la siguiente:

1. Descargar el QT Creator de la página oficial (qt.nokia.com).
2. Ingresar a la terminal y dirigirse al directorio donde se encuentra descargado el archivo.
3. Ejecutar el comando shmod u+x nombrearchivo.bin
4. Ejecutar normalmente el nombrearchivo.bin
5. Para poder compilar desde el IDE se debe tener instalado el g++, este se instala desde el gestor de paquetes synaptic.

Luego de realizar estos pasos el programa QT Creator está listo para ser utilizado y manipulado.

10.2. CREACIÓN DE UN NUEVO PROYECTO

Estos son los pasos para empezar con el desarrollo sobre el IDE

1. La pantalla inicial [imagen 9] que aparece al abrir un proyecto muestra los menús de los que se puede hacer en el IDE donde debemos darle menú →nuevo.

Imagen 9 [Pantalla Inicial]

2. La segunda pantalla que nos muestra el programa es la del tipo del proyecto que queremos generar. Como queremos generar un proyecto con interfaz gráfica de usuario, seleccionamos QT4 GUI Application.

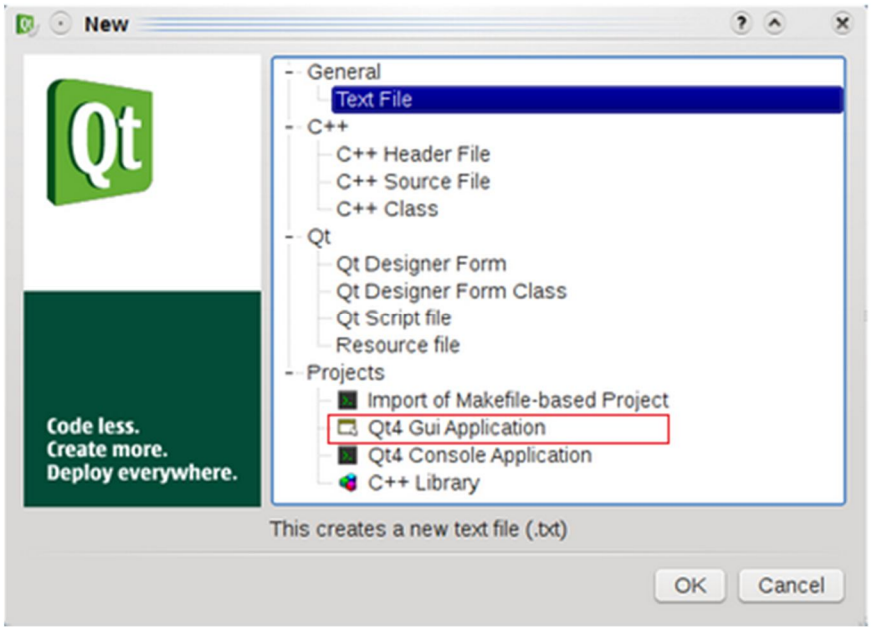


Imagen 10 [Seleccionar tipo de proyecto]

3. Luego de seleccionar el tipo de proyecto nos va a pedir la ruta y él nombre del proyecto.

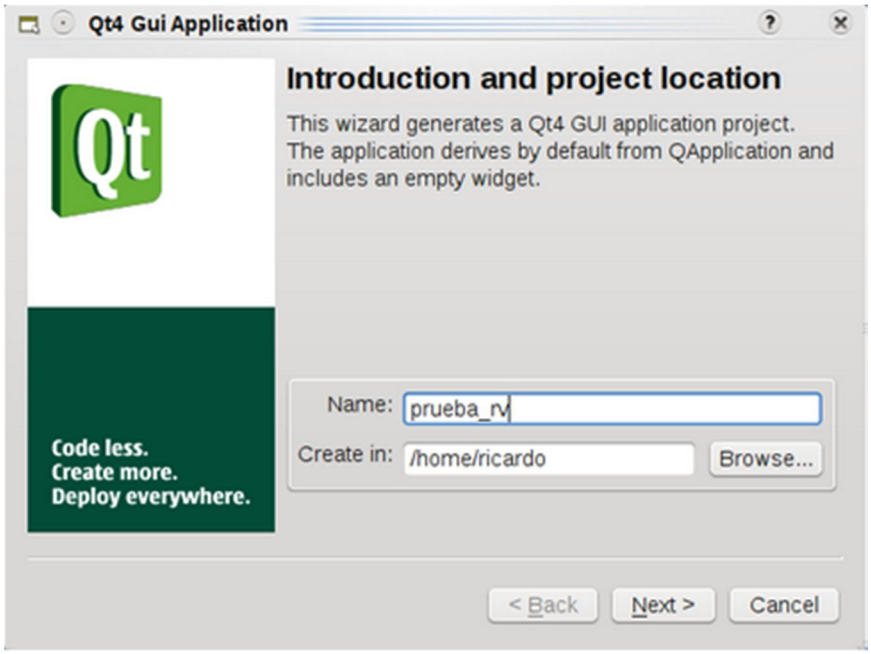


Imagen 11 [Nombre Proyecto]

4. La ventana siguiente es muy importante para la configuración del proyecto ya que nos muestra las librerías que tendremos que importar para trabajar en el mismo. En nuestro caso aparte de las librerías que vienen por defecto seleccionamos las librerías QTSql Module.

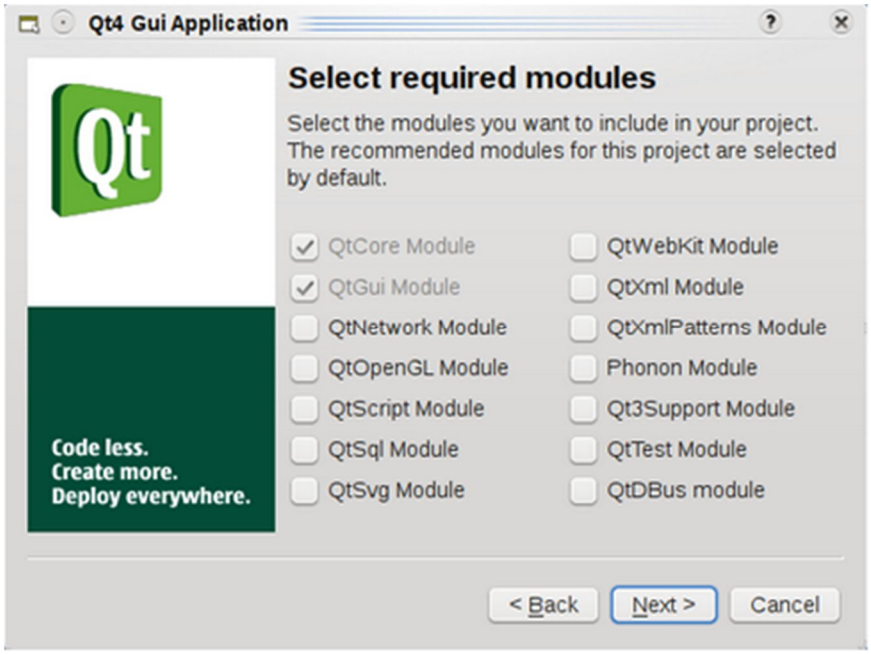


Imagen 12 [Librerías]

5. Luego se desplegara la ventana para ingresar el nombre de la clase principal y la opción de generar un formulario que es la parte gráfica, que queda con la extensión .ui [Imagen 13]. y al darle siguiente se mostrara toda la información que hemos configurado hasta el momento [Imagen 14].

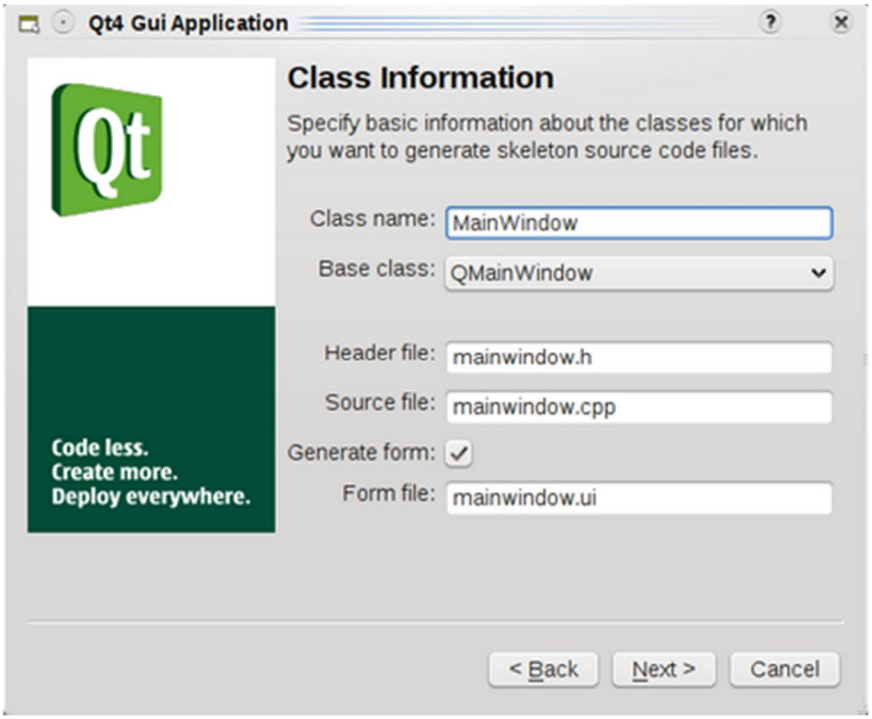


Imagen 13 [Clase principal]

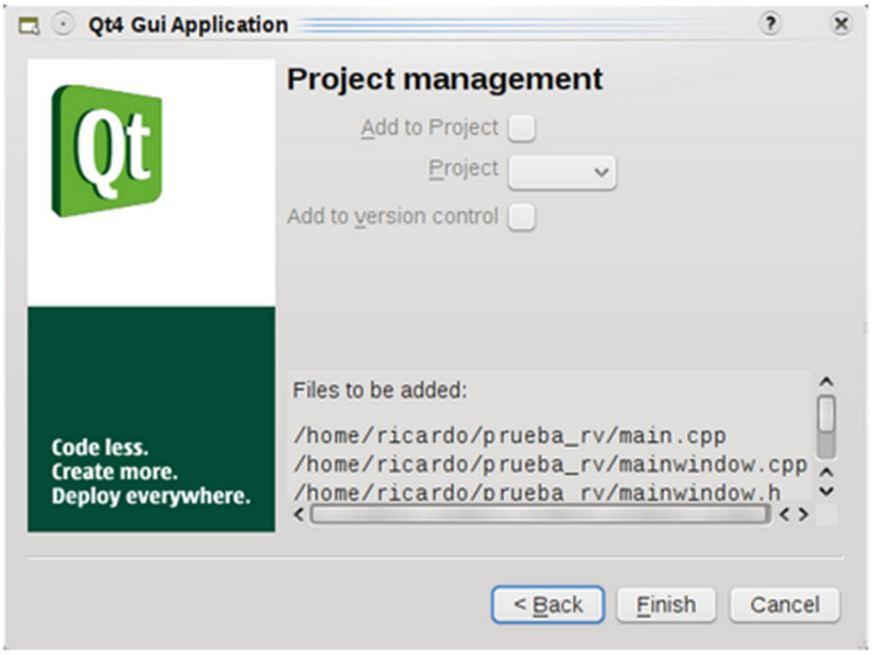


Imagen 14 [Resumen]

6. Finalmente se nos despliega la ventana con las clases (.cpp) [Imagen 15] y la interfaz gráfica (.ui) [Imagen 16]. Es aquí donde se empieza a desarrollar tanto a nivel de código y a nivel de interfaz gráfica.

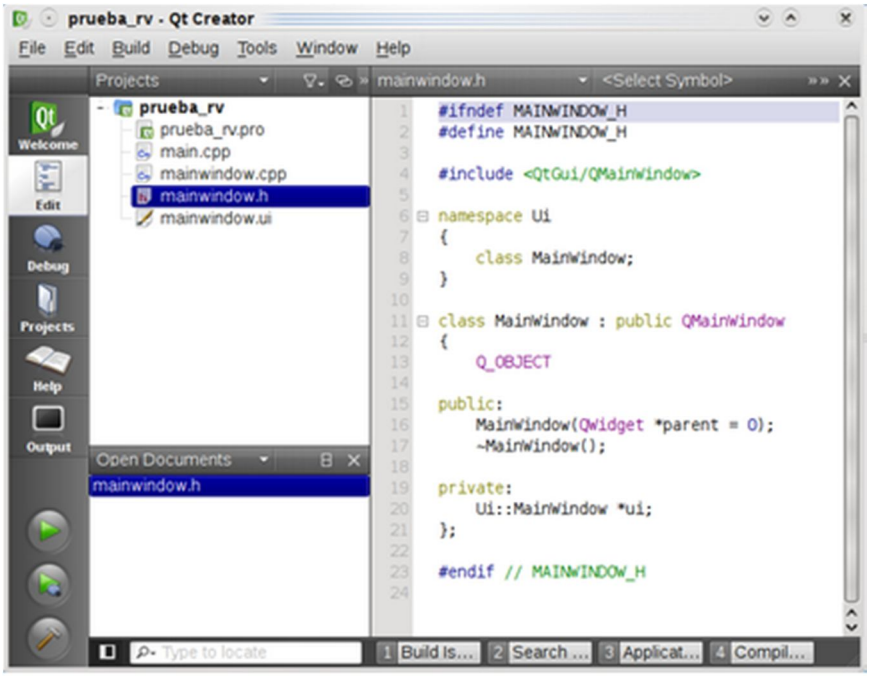


Imagen 15 [clases .cpp]

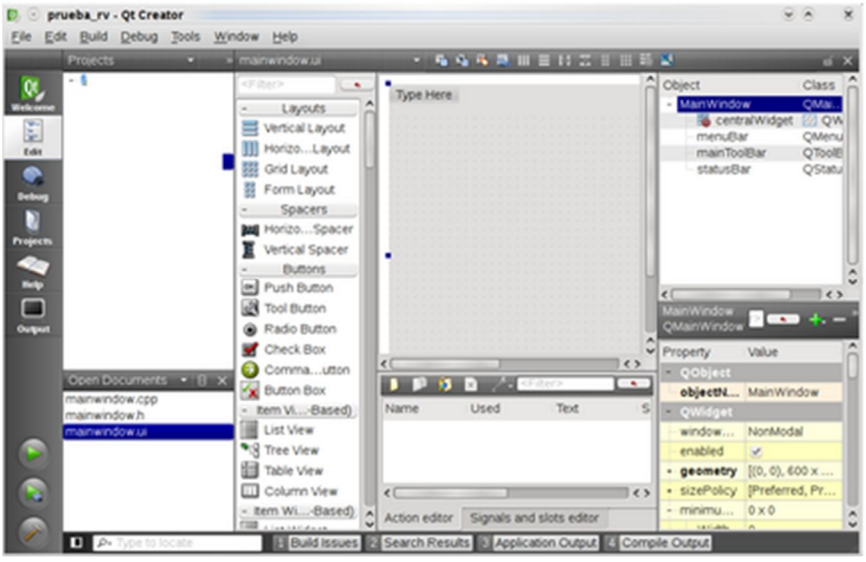


Imagen 16 [interfaz .ui]

7. Finalmente cuando el proyecto esté listo o se quieran ir probando avances, se puede dar clic en el botón ejecutar, esto con el fin de visualizar la construcción que llevamos del proyecto.

10.3. CONFIGURACIÓN DEL QPORT

10.3.1. INSTALACIÓN CON LINUX

1. Lo primero que debemos de hacer es copiar el archivo "libqport.so" en la ubicación de las librerías "/usr/lib" .
2. Debemos copiar los archivos "ManageSerialPort.h", "posix_qextserialport.h", "qextserialbase.h", "qextserialport.h", "ui_qport.h" en la carpeta de los includes de Qt4, que por defecto se encuentra en "/usr/include/qt4/".
3. Crear ahí dentro una carpeta llamada qport y copiar todos los archivos anteriores.
4. Finalmente incluir unos parámetros al archivo del proyecto.

10.4. INSTALACIÓN DEL PROGRAMA EN LA SBC

10.4.1. ACTUALIZACIÓN DE LA DISTRIBUCIÓN

Antes de las instalaciones se debe actualizar nuestro DEBIAN a la versión estable, ya que se debe instalar algunos paquetes que necesitan de muchas dependencias que solo la poseen las distribuciones estables, esto se consigue mediante el comando apt-get dist-upgrade pero si ya se tiene una estable nos basta con apt-get upgrade.

10.4.2. PREPARAR LA MÁQUINA PARA LA COMPILACIÓN CRUZADA

Descomprimir el archivo crosstool-linux-gcc-3.3.4-glibc-2.3.2-0.28rc39.tar.bz2. en la raíz del sistema de archivos.

Verificar que se encuentren en la carpeta los archivos para la compilación como arm-linux-g++, arm-linux-c, arm-linux-ld, etc. la carperta se encuentra en la siguiente ubicación /usr/local/opt/crosstool/arm-linux/gcc-3.3.4-glibc-2.3.2/bin.

Realizar la siguiente acción

debian:/home/nombreequipo#

PATH=$PATH:/usr/local/opt/crosstool/arm-linux/gcc-3.3.4-glibc-2.3.2/bin

Para poder invocar al compilador desde cualquier parte del sistema.

Para descomprimir este archivo realizamos el siguiente comando

debian:/home/nombreequipo# tar xvjf crosstool-linux-gcc-3.3.4-glibc-2.3.2-0.28rc39.tar.bz2

10.4.3. INSTALACIÓN DE QT/EMBEDDED PARA LINUX

Antes que nada se debe tener el sistema actualizado con una distribución estable de debian. Por facilidad abrir una consola de root para evitar el tema de los permisos.

cd ~

Crear estos tres directorios.

$ mkdir -p project/downloads/qt_embedded

$ mkdir -p project/sysapps/host

$ mkdir project/sysapps/device

$ cd ~/project/downloads/qt_embedded/

Se debe descargar estos dos archivos

$ wget ftp://ftp.trolltech.com/qt/source/qt-embedded-linux-opensource-src-4.4.3.tar.gz

$ wget ftp://ftp.trolltech.com/qt/source/qt-x11-opensource-src-4.4.3.tar.gz

$ cd ~/project/sysapps/host

Se descomprime

$ tar -xzvf ~/project/downloads/qt_embedded/qt-embedded-linux-opensource-src-4.4.3.tar.gz

$ tar -xzvf ~/project/downloads/qt_embedded/qt-x11-opensource-src-4.4.3.tar.gz

$ export QT4DIR=~/project/sysapps/host/qt-x11-opensource-src-4.4.3

$ export QTEDIR=~/project/sysapps/host/qt-embedded-linux-opensource-src-4.4.3

$ sudo apt-get install libx11-dev libxmu-dev libpng12-dev

$ sudo apt-get install libxtst-dev

$ sudo apt-get install build-essential

$ cd $QT4DIR

$ export QTDIR=$QT4DIR

$ export PATH=$QTDIR/bin:$PATH

$ export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH

```
$ echo yes | ./configure -prefix /usr/local/qt4
$ make
$ sudo make install
$ mkdir $QTEDIR/bin
$ cp bin/uic $QTEDIR/bin/
$ export TMAKEPATH=$TMAKEDIR/lib/linux-g++
$ export QMAKESPEC=$QTDIR/mkspecs/linux-g++
$ export PATH=$PATH:/usr/local/qt4/bin
$ cd ~/project/sysapps/host/qt-x11-opensource-src-4.4.3/tools/qvfb
$ qmake
$ make
$ sudo cp ../../bin/qvfb /usr/local/qt4/bin
=========
se cierra la terminal para limpiar lo creado anteriormente
se abre una nueva terminal
==========
$ export QTEDIR=/root/project/sysapps/host/qt-embedded-linux-
opensource-src-4.4.3
$ cd $QTEDIR
$ export QTDIR=$QTEDIR
$ export PATH=$PATH:$QTDIR/bin
$ export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
$ echo yes | ./configure -prefix /usr/local/qte4 -qvfb
WARNING: Failure to find: ../skin.cpp
WARNING: Failure to find: ../skin.h
  for /root/project/sysapps/host/qt-x11-opensource-src-
4.4.3/tools/xmlpatterns/xmlpatterns.pro
Qt queda onfigurado
Ejecutar 'make'
Cuando esto se hla hecho ejecutar 'make install'.
Qt sera instalado en la ruta /usr/local/qt4
```

Para reconfigurar, ejecutar 'make confclean' y 'configure'.

$ make

$ sudo make install

//test framebuffer:

$ /usr/local/qt4/bin/qvfb

==========

crear una nueva ventana de consola / terminal

==========

$ cd ~project/sysapps/qt_embedded/host/qt-embedded-linux-opensource-src-4.4.3/examples/draganddrop/draggibleicons

$ ./draggableicons -qws //QTE necesita un servidor, -qws quiere decir que esta aplicaciòn es el servidor

==========Compilaciòn cruzada de QT/E==========

nueva terminal

==========

paso 1: para soporte de tslib y autoconf se descarga y se instala en el computador propio

$ cd ~/project/downloads

$ wget http://ftp.gva.es/mirror/debian2/pool/main/t/tslib/tslib_1.0.orig.tar.gz

$ wget ftp://ftp.gnu.org/gnu/autoconf/autoconf-2.63.tar.gz

$ cd ~/project/sysapps/

$ mkdir tslib

$ mkdir autoconf

$ cd autoconf

$ tar xzvf ~/project/downloads/autoconf-2.63.tar.gz

$ cd autoconf-2.63

$ ./configure

$ make && sudo make install

$ cd ../../tslib/

$tar xzvf ~/project/downloads/tslib_1.0.orig.tar.gz

$ cd tslib-1.0

$ sudo apt-get install automake //al parecer es necesario

$ sudo apt-get install libtool // al parecer es necesario

$ sudo ./autogen.sh

$ ./configure CC=arm-linux-gcc CXX=arm-linux-g++ PLUGIN_DIR=/usr/local/linux-arm/plugins -prefix=/usr/local/linux-arm -host=arm-linux

======

======

$ edit config.h and comment out #define malloc rpl_malloc

$ make

$ sudo make install

==========

asegurese de que el compilador cruzado sea localizable

try:

$ arm-linux-g++

si no se encuentra se configura

PATH=/path/to/cross/compiler:$PATH”

==========

==========

utilice Qmake para especificar la ruta del make del proyecto

==========

$ cd ~/project/sysapps/device

$ tar -xzvf ~/project/downloads/qt_embedded/qt-embedded-linux-opensource-src-4.4.3.tar.gz

$ cd qt-embedded-linux-opensource-src-4.4.3

$ export QTDIR=$PWD

$ export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH

$ echo yes | ./configure -embedded arm -xplatform qws/linux-arm-g++ -no-qvfb -depths all -qt-mouse-tslib -qt-kbd-usb -I /usr/local/linux-arm/include -L /usr/local/linux-arm/lib

$ make

$ sudo make install //Installs in /usr/local/Trolltech/QT-Embedded-4.4.3-arm/

PATH=$PATH:/usr/local/opt/crosstool/arm-linux/gcc-3.3.4-glibc-2.3.2/bin

export QT4DIR=~/project/sysapps/host/qt-x11-opensource-src-4.4.3/

export QTEDIR=~/project/sysapps/host/qt-embedded-linux-opensource-src-4.4.3

export QTDIR=$QT4DIR

export PATH=$QTDIR/bin:$PATH

export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH

export TMAKEPATH=$TMAKEDIR/lib/linux-g++

export QMAKESPEC=$QTDIR/mkspecs/linux-g++

export PATH=$PATH:/usr/local/qt4/bin

export QTDIR=$QTEDIR

export PATH=$PATH:$QTDIR/bin

export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH

export QTDIR=$PWD

export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH

restaurar conexión de red

/etc/init.d/networking restart

debian:/home/jorge# dd if=ImagenSBC7390ConQTE of=/dev/sdf

cambiar la fecha

ts7000:/home/eclipse# hwclock --set --date="2009-06-04 18:51"

ts7000:/home/eclipse# date -s "2009-06-04 18:51"

para cambiar la secuencia de inicio entrar como root y hacer la modificacion en linuxrc

rc.local para iniciar programa

nice -n -18 ./serialEXT -qws para ejecutar el programa.

11. DESARROLLO DE LA SOLUCIÓN

11.1. CONCEPTUALIZACIÓN

Antes de empezar a desarrollar o diseñar la solución del problema, fue necesario realizar una amplia investigación sobre los dispositivos necesarios para la comunicación con la maquinaria industrial. Luego de esta demostración se llegaron a las conclusiones que se encuentran plasmadas en el marco teórico.

11.1.1. INVESTIGACIÓN DEL ESTADO DEL ARTE

Cuando se investigó a cerca de los sistemas embebidos (SBC), se encontró diferentes tipos de dispositivos, anteriormente mencionados, en especial los que ofrece la empresa Technology Systems, ya que son SBC muy completos de unos muy buenos tamaños con pantalla táctil que facilitan la interacción entre el operario y la máquina, y cumplen las características necesarias para ejecutar el software que se desea desarrollar. Se escogió el dispositivo que cumplía las características necesarias para el desarrollo que se necesitaba. La segunda investigación que se realizó, fue con qué tipo de lenguaje se iba a realizar este desarrollo, y se encontró que lo más utilizado actualmente son las librerías QT Nokia que son soportadas por C++ y C#, por lo cual la investigación se centró en estas.

11.1.2. ANALISIS DEL PROBLEMA

Luego de realizar todas las investigaciones pertinentes para el desarrollo del problema y revisando la maquina [Imagen 17] que nos iba a servir como prototipo del proyecto, seleccionamos como SBC la TS 7390, por las características que presentaba, como se mencionó anteriormente en el estado del arte. Y como lenguaje de desarrollo C++ soportado por las librerías QT Nokia, todo esto apoyado en el IDE de desarrollo QT Creator que unas de sus principales características que nos brindaba para el proyecto, es que trae incorporado un editor gráfico para realizar la interfaz de usuario, lo cual nos proporcionaba

una excelente “mezcla” entre la parte visual que puede ser desarrollada y una pantalla táctil donde se mostraba el desarrollo.

Imagen 17 [Maquina (1)]

11.1.3. ANALISIS DE RIESGO

El principal riesgo sobre el cual no se tiene ningún tipo de control es un daño en la parte eléctrica de la máquina, ya que el sistema se encarga de comunicarse con el PLC por medio de señales, y este a su vez responde con una señal para mostrar que se estableció conexión exitosamente; debido a esto, cualquier daño en un sensor, el motor o incluso el mismo PLC, detendría el sistema y solamente se podría reanudar en el momento en que el daño esté solucionado. Esto implicaría que la información dejaría de llegar a la base de datos y se dejarían de enviar los mails.

11.2. ACLARACIONES

Debido a que el proyecto que se desea presentar no es únicamente para una máquina, sino para aumentar el rendimiento de cualquier tipo de maquina utilizada en la industria, todos los ITEMS siguientes variaran de acuerdo a la maquina utilizada. A continuación se mostraran los del prototipo desarrollado.

11.3. ELABORACIÓN

11.3.1. DISEÑO DE LA SOLUCIÓN

La idea con este proyecto, es medir ciertas variables que pueden ser analizadas de acuerdo al funcionamiento de la máquina, como por ejemplo, tiempo en movimiento, detenida, entre otras. Por lo cual el diseño de la solución fue relativamente sencillo ya que simplemente se necesitaba controlar este tipo de eventos.

Se implementó una interfaz gráfica [Imagen 18] que tuviera cuatro botones que registraran en una base de datos cada uno de los eventos que puedan ocurrir durante el funcionamiento de la máquina.

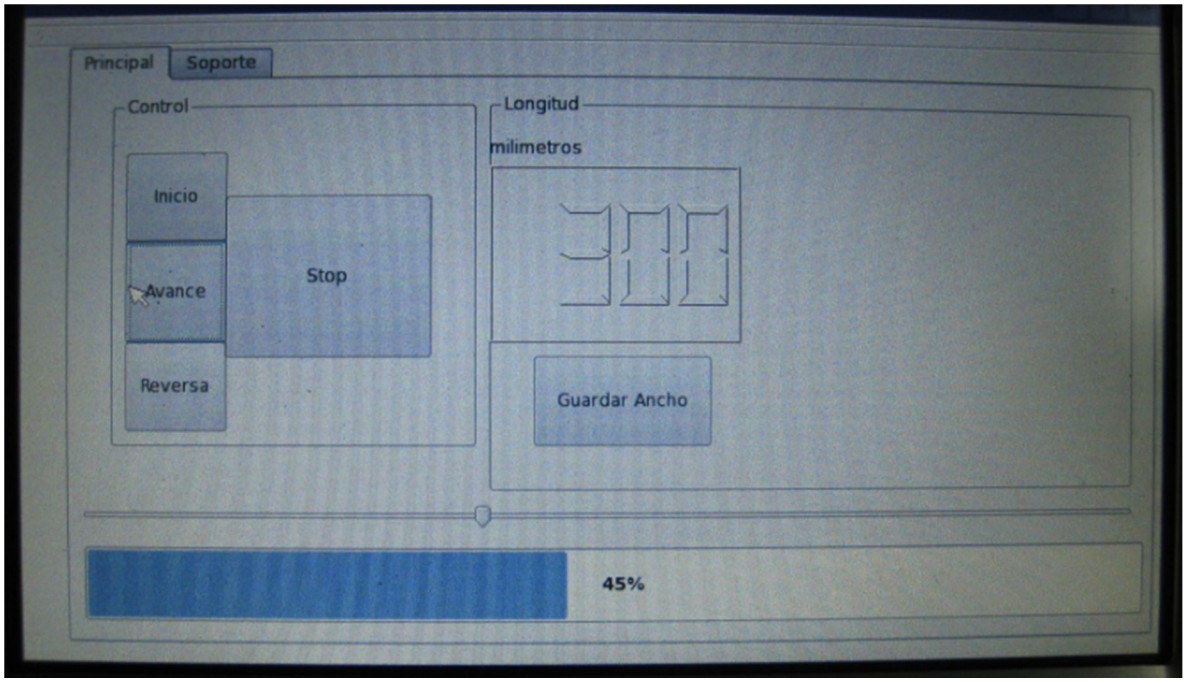


Imagen 18 [Prototipo en la SBC]

11.3.2. ANALISIS DE REQUISITOS FUNCIONALES

Los requisitos funcionales identificados fueron todos aquellos que suplieron con la necesidad del desarrollo.

- El sistema debe enviar una señal de iniciar al PLC, cuando se presiona el botón “inicio”.
- El sistema debe enviar una señal de avance al PLC, cuando se presiona el botón “avance”.
- El sistema debe enviar una señal de stop al PLC, cuando se presiona el botón “stop”.
- El sistema debe enviar una señal de reversa al PLC, cuando se presiona el botón “reversa”.
- El sistema debe enviar una variable al PLC para que la guarde, cuando se oprime el botón “guardar archivo”
- El sistema debe de guardar en la base de datos el tiempo en que la maquina esta en stop.
- El sistema debe de guardar en la base de datos el tiempo en que la maquina está en funcionamiento.
- El sistema debe de enviar un mail en el momento en que se pare la máquina.

11.3.3. ANALISIS DE REQUISITOS NO FUNCIONALES.

Los requisitos no funcionales identificados fueron todas las cosas que debe tener el desarrollo que no tiene que ver con la funcionalidad del sistema, los cuales fueron:

- El sistema debe estar corriendo de manera correcta, todo el tiempo que la máquina se encuentre en funcionamiento.
- El sistema se debe acoplar fácilmente a futuras necesidades que tenga la empresa.
- El sistema debe ser fácil de manejar.
- El sistema debe ser escalable y acoplarse a nuevas funcionalidades que pueda tener la máquina.
- El sistema debe ser multiplataforma.

11.3.4. SELECCIÓN DE TECNOLOGIAS Y ARQUITECTURA

Como se mencionó anteriormente en el análisis del problema, se seleccionó la SBC TS 7390, por las características que presentaba.

Como arquitectura al utilizar el protocolo MODBUS, se implementó una arquitectura cliente/servidor (maestro/esclavo), ya que este protocolo trabaja sobre la misma; donde el esclavo es el PLC y el maestro la SBC.

11.3.5. DISEÑO DE LA INTERFAZ GRAFICA

El diseño de la interfaz gráfica [Imagen 18] se hizo con el QT Creator (mencionado anteriormente), y se desarrolló de acuerdo a los requisitos funcionales del sistema.

La interfaz gráfica cuenta con 5 botones y un scrollbar, los cuales son:

- Inicio: Lleva la maquina a su posición inicial.
- Stop: Para la maquina en el punto donde se encuentre.
- Avance: Inicia el ciclo de la máquina.
- Guardar Archivo: Cuando ya se ha modificado el scrollbar, guarda el valor.
- Reversa: Comienza el ciclo en sentido contrario.
- Scrollbar: Indica el número de milímetros.

11.4. PRUEBAS

Las diferentes pruebas que se realizaron fueron:

- Las primeras pruebas fueron de comunicación entre el SBC y el PLC, ya que necesitábamos saber si estos dos si se estaban encontrando y estaban llegando los datos correctamente. Luego de que estas pruebas fueron superadas pasamos a las pruebas con la máquina.
- Al realizar pruebas con las maquinas se analizaron cada una de las funcionalidades del SBC con la máquina. La forma como las abordamos, fue, ejecutar botón por botón y revisar que la comunicación y lo que estaba haciendo la maquina fuera correcta.
- Luego se realizaron pruebas con la base de datos, estas consistían en enviarle datos cada que ocurriera un evento que quisiéramos que estuviera almacenado y luego revisábamos que efectivamente los valores si se hubieran almacenado.
- Finalmente se realizaron pruebas con los correos electrónicos, revisando que si se enviara cuando fuera necesario.

Después de que se realizaron todas estas pruebas concluimos que el sistema estaba estable y que estaba haciendo efectivamente lo que tenía que hacer.

12. TRABAJOS FUTUROS

Gracias al enfoque que se le dio desde un principio a este proyecto de grado, el campo de acción que tenemos luego de realizar el mismo es muy amplio, ya que cualquier máquina que se utilice en la industria, que comunique el PLC de la misma con una SBC, es posible “intervenir” el programa del sistema embebido para implementar mejoras en este que ayuden al mejoramiento de rendimiento de la máquina.

Inclusive, en máquinas en las cuales ya existe un SBC implementado, es posible “intervenirlo”, para medir las diferentes variables y analizar los datos.

13. PRESUPUESTO

A continuación se presentaran los costos que aproximadamente tendrá todo el desarrollo de la aplicación desde el diseño hasta la implementación de este.

13.1. RECURSOS HUMANOS

A continuación se mostrarán las personas que participaran en el proyecto

Nombre	**Rol**	**Dedicación Semanal**	**Valor Hora**	**Subtotal**
Jorge Mario Ríos	Investigador, desarrollador	25 horas	20000	$ 500.000
Juan Carlos Ríos	Investigador, desarrollador	25 horas	20000	$ 500.000
Marcos Barrientos	Investigador, desarrollador	25 horas	20000	$ 500.000
Germán Guzmán	Asesor	4 Horas	35000	$ 140.000
Edwin	Asesor	4 Horas	35000	$ 140.000
Subtotal Semanal				$ 1.780.000

13.2. HARDWARE Y SOFTWARE

Acá mostraremos el costo del hardware y software que utilizaremos en el proyecto

Producto	Descripción	Cantidad	Licencias	Subtotal
Computador	Procesador i3 intel, memoria ram dr3 2GB	1		$ 1.500.000
SBC	Pantalla táctil con debian embebido	1		$ 1.400.000
PLC	Control Lógico Programable	1	1	$ 1.200.000
Red Interna	Dos o más redes o segmentos de la red conectados a dispositivos	1		$ 200.000
Sensores	Productos	n	0	$ 400.000
Librerias QT Ver.4.5	Es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica como herramientas de la consola y los servidores	1	0	$ 0

	$ 4.700.000
Total	

14. GLOSARIO DE TERMINOS

- Automatización: Es el uso de sistemas o elementos computarizados y electromecánicos para controlar maquinarias y/o procesos industriales.
- ASCII: Código estándar americano para el intercambio de información.
- Board: Placa base.
- Librerías QT: Son una biblioteca multiplataforma utilizadas para el desarrollo de aplicaciones con interfaz gráfica de usuario, estas actualmente cuentan aproximadamente 500 clases, más de 9000 funciones y 500.000 líneas de código, brindando un alto grado de "potencia" aproximándose a lenguajes como C# o Java con la eficiencia de código compilado en C++.
- Monitoreo: El monitoreo, a rasgos generales, consiste en la observación del curso de uno o más parámetros para detectar eventuales anomalías.
- PLC: Controlador lógico programable.
- Protocolo MODBUS: Es un protocolo desarrollado para la comunicación basado en la arquitectura Maestro/Esclavo o cliente/Servidor [Imagen 1]. Este protocolo se sitúa en el nivel 7 del modelo OSI (nivel de aplicación). Actualmente es uno de los protocolos más utilizados para establecer comunicaciones entre diferentes dispositivos electrónicos industriales.
- Prototipo: Primer entregable del software, que se va a utilizar para la SBC.
- QPORT: Es un plugin para las librerías QT de NOKIA, este permite realizar las conexiones con el puerto serial del computador. La ventaja de este plugin es que permite realizar una comunicación de una manera muy sencilla e inclusive visual; lo único que hay que hacer es introducir los parámetros y él se encarga del resto, inclusive muestra gráficamente la conexión.

- QT CREATOR: Es un IDE multiplataforma que trabaja bajo librerías QT de una manera sencilla y rápida.

- RAM: Memoria de acceso aleatorio.

- REQUISITOS FUNCIONALES: Un requisito funcional define el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán a la práctica

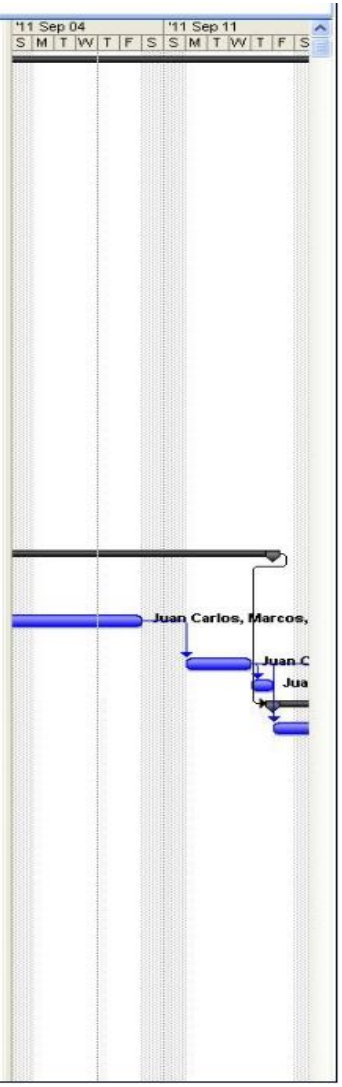
- REQUISITOS NO FUNCIONALES: Un requisito no funcional o atributo de calidad es, en la ingeniería de sistemas y la ingeniería de software, un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales.

- SBC: Single board computer.

- CRLF: Se refiere a la combinación de dos códigos de control: CR (retorno de carro) y LF (salto de línea), uno detrás del otro; normalmente con el objetivo de crear una nueva línea.

15. CRONOGRAMA DE ACTIVIDADES

	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	**PROYECTO MONITOREO EN TIEMPO REAL MAQUINARIA INDUSTRIAL**	**171 days**	**Mon 11/03/07**	**Mon 11/10/31**		
2	**1- Levantamiento de Requisitos**	**12 days**	**Mon 11/03/07**	**Tue 11/03/22**		
3	1.1- Análisis Requerimiento	8 days	Mon 11/03/07	Wed 11/03/16		Juan Carlos, Marcos, Jorge Mario
4	1.2- Reunión de Contextualización (Asesores)	1 day	Thu 11/03/17	Thu 11/03/17	3	Juan Carlos, Marcos, Jorge Mario, German, Edwin
5	1.3- Retroalimentación Requerimientos	2 days	Mon 11/03/21	Tue 11/03/22	4	Juan Carlos, Marcos, Jorge Mario
6	**2- Análisis**	**16 days**	**Mon 11/03/28**	**Mon 11/04/18**	**2**	
7	2.1- Realización Cronograma	2 days	Mon 11/03/28	Tue 11/03/29		Juan Carlos Ríos Herrera
8	2.2- Casos de Uso	5 days	Mon 11/04/04	Fri 11/04/08	7	Juan Carlos, Marcos, Jorge Mario
9	2.3- Necesidades y/o Objetivos	5 days	Mon 11/04/11	Fri 11/04/15	8	Marcos Barrientos Botero, Jorge Mario
10	2.4- Reunión de Contextualización (Asesores)	1 day	Mon 11/04/18	Mon 11/04/18	9	Juan Carlos, Marcos, Jorge Mario, German, Edwin
11	**3- Diseño**	**35 days**	**Tue 11/04/19**	**Mon 11/06/06**	**6**	
12	**3.1- Diccionarios**	**6 days**	**Tue 11/04/19**	**Tue 11/04/26**		
13	3.1.1- Diccionario Técnico	6 days	Tue 11/04/19	Tue 11/04/26		Jorge Mario Ríos
14	3.1.2- Diccionario de Usuario	6 days	Tue 11/04/19	Tue 11/04/26		Marcos Barrientos Botero
15	3.2- Diagrama de Procesos	8 days	Wed 11/04/27	Fri 11/05/06	12	Juan Carlos, Marcos, Jorge Mario
16	3.3- Diagrama de Secuencia	8 days	Mon 11/05/09	Wed 11/05/18	15	Juan Carlos, Marcos, Jorge Mario
17	3.4- Diagrama de Flujo	8 days	Thu 11/05/19	Mon 11/05/30	16	Juan Carlos, Marcos, Jorge Mario
18	3.5- Reunión de Contextualización (Asesores)	1 day	Tue 11/05/31	Tue 11/05/31	17	Juan Carlos, Marcos, Jorge Mario, German, Edwin
19	**3.6- Arquitectura**	**4 days**	**Wed 11/06/01**	**Mon 11/06/06**	**18**	
20	3.6.1- Definición Modelo de Datos	2 days	Wed 11/06/01	Thu 11/06/02		Juan Carlos, Marcos, Jorge Mario
21	3.6.2- Definición Framework	1 day	Fri 11/06/03	Fri 11/06/03	20	Juan Carlos, Marcos, Jorge Mario
22	3.6.3- Definición IDE	1 day	Mon 11/06/06	Mon 11/06/06	21	Juan Carlos, Marcos, Jorge Mario
23	3.6.4- Reunión de Contextualización (Asesores)	1 day	Mon 11/06/06	Mon 11/06/06	21	Juan Carlos, Marcos, Jorge Mario, German, Edwin
24	**4- Construcción**	**73 days**	**Tue 11/06/07**	**Thu 11/09/15**	**11**	
25	4.1- Creación y construcción Base de datos (Modelo)	15 days	Tue 11/06/07	Mon 11/06/27		Juan Carlos, Marcos, Jorge Mario
26	4.2- Creación y construcción interfaces de Usuario	30 days	Tue 11/06/28	Mon 11/08/08	25	Juan Carlos, Marcos, Jorge Mario
27	4.3- Creación y construcción lógica Funcional y No Funcional	45 days	Mon 11/07/11	Fri 11/09/09		Juan Carlos, Marcos, Jorge Mario
28	4.4- Creación y construcción de circuitos electrónicos	15 days	Mon 11/07/25	Fri 11/08/12		Jorge Mario Ríos
29	4.5- Prueba de par	3 days	Mon 11/09/12	Wed 11/09/14	27	Juan Carlos, Marcos, Jorge Mario
30	4.6- Reunión de Contextualización (Asesores)	1 day	Thu 11/09/15	Thu 11/09/15	29	Juan Carlos, Marcos, Jorge Mario, German, Edwin
31	**5- Implantación Ambiente de pruebas**	**7 days**	**Fri 11/09/16**	**Mon 11/09/26**	**24**	
32	5.1- Implantación modelo de datos ambiente de pruebas	2 days	Fri 11/09/16	Mon 11/09/19	29	Juan Carlos, Marcos, Jorge Mario
33	5.2- Implantación Interfaces de usuario ambiente pruebas	2 days	Mon 11/09/19	Tue 11/09/20	29	Juan Carlos, Marcos, Jorge Mario
34	5.3- Implantación ambiente pruebas lógica Funcional y no Funcional	2 days	Tue 11/09/20	Wed 11/09/21	29	Juan Carlos, Marcos, Jorge Mario
35	5.4- Implantación Circuitos Electrónicos Ambiente Pruebas	2 days	Thu 11/09/22	Fri 11/09/23	34	Jorge Mario Ríos
36	5.5- Reunión de Contextualización (Asesores)	1 day	Mon 11/09/26	Mon 11/09/26	35	Juan Carlos, Marcos, Jorge Mario, German, Edwin
37	**6- Pruebas Funcionales**	**20 days**	**Tue 11/09/27**	**Mon 11/10/24**	**31**	
38	6.1- Despliegue Pruebas Usuario	2 days	Tue 11/09/27	Wed 11/09/28	36	Juan Carlos, Marcos Barrientos
39	6.2- Ayuda Pruebas Usuario	2 days	Thu 11/09/29	Fri 11/09/30	38	Juan Carlos, Marcos Barrientos
40	6.3- Corrección de bugs(errores)	5 days	Mon 11/10/03	Fri 11/10/07	39	Juan Carlos, Marcos, Jorge Mario
41	6.4- Definición y construcción de controles de cambios	10 days	Mon 11/10/10	Fri 11/10/21	40	Juan Carlos, Marcos, Jorge Mario
42	6.5- Reunión de Contextualización (Asesores)	1 day	Mon 11/10/24	Mon 11/10/24	41	Juan Carlos, Marcos, Jorge Mario, German, Edwin
43	**7- Implantación Ambiente Producción**	**5 days**	**Tue 11/10/25**	**Mon 11/10/31**	**37**	
44	7.1- Implantación modelo de datos ambiente de producción	1 day	Tue 11/10/25	Tue 11/10/25	41	Juan Carlos, Marcos, Jorge Mario
45	7.2- Implantación Interfaces de usuario ambiente de producción	1 day	Wed 11/10/26	Wed 11/10/26	44	Juan Carlos, Marcos, Jorge Mario
46	7.3- Implantación producción lógica Funcional y no Funcional	1 day	Thu 11/10/27	Thu 11/10/27	45	Juan Carlos, Marcos, Jorge Mario
47	7.4- Implantación Circuitos Electrónicos Ambiente Producción	1 day	Fri 11/10/28	Fri 11/10/28	46	Juan Carlos, Marcos, Jorge Mario
48	7.5- Reunión de Contextualización (Asesores)	1 day	Mon 11/10/31	Mon 11/10/31	47	Juan Carlos, Marcos, Jorge Mario, German, Edwin



16. CONCLUSIONES

- Luego del análisis de las posibles SBC, protocolos y librerías, se puede concluir que es posible desarrollar un sistema que se comunique con la maquina y a su vez este en constante monitoreo de cada uno de los procesos que realiza la misma y de esta manera tener datos para hacer el análisis de la productividad de dicha maquina.
- Al estudiar los diferentes protocolos que se utilizan en la industria para la comunicación entre el PLC y la SBC, se seleccionó el protocolo MODBUS, ya que de acuerdo a las características que presenta éste, es posible realizar una comunicación con el PLC de forma más fácil.
- Se comprendió el valor que tienen las tecnologías embebidas aplicadas a sistemas de control y el potencial que tienen las mismas.
- Luego de realizar el monitoreo de los procesos de una maquina, se concluyo que es posible analizar el rendimiento de un operario con diferentes respuestas arrojadas por dicha maquina.
- Después de un estudio a las diferentes librerías con las que se podía desarrollar el proyecto, se seleccionaron las librerías QT de NOKIA ya que estas proveen una interfaz de desarrollo, amigable y fácil de utilizar, disminuyendo así el tiempo del desarrollo.
- Luego de monitorear los procesos de la maquina, se determinó que es muy importante estar en constante almacenamiento de datos para poder mostrarlos de una forma en la que se pueda tomar decisiones tanto de la maquina como del operario.

17. REFERENCIAS

REFERENCIAS DIGITALES

1. http://automon.donaloconnor.net/
   [Citado en Septiembre 5]
2. http://www.embeddedarm.com/products/board-detail.php?product=TS-TPC-7390
   [Citado en Junio 13]
3. http://www.freewebs.com/jojaqui/qt4eclipse.pdf
   [Citado en junio 17]
4. http://www.rtaautomation.com/modbusrtu/
   [Citado en agosto 9]
5. http://www.modbus.org/
   [Citado en Agosto 11]
6. http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/
   [Citado en agosto 29]

REFERENCIAS IMÁGENES

1. Image 1 - http://www.modbus.org/
2. Image 2 - http://www.modbus.org/
3. Image 3 - http://www.modbus.org/
4. Image 4 - http://www.modbus.org/
5. Image 5 - http://www.modbus.org/
6. Image 6 - http://www.freewebs.com/jojaqui/qt4eclipse.pdf
7. Image 9 - http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/
8. Image 10 - http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/
9. Image 11 - http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/

10. Image 12 - http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/
11. Image 13 - http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/
12. Image 14 - http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/
13. Image 15 - http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/
14. Image 16 - http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/

Asesor Germán Guzmán

_____

Asesor Edwin

_____