Universidad Luterana Salvadoreña

Cátedra: Redes I

Docente: Ing. Manuel de Jesús Flores

Evaluación: Entrega de proyecto final
Fecha de entrega: 30/0/2015

Desarrollar un programa Cliente ï Servidor con Arduino

Estudiantes:

Carnet	**Apellidos**	**Nombres**	**Participación**
PR02110097	Piche Ramirez	Fernando Dagoberto	100%
TN01121295	Trejo Noble	Pedro Antonio	100%
GG202110806	Gámez Gámez	Noris Marbelís	100%

Lugar y fecha

San Salvador 25 de abril de 2015

Contenido

INTRODUCCIÓN

En los últimos años, el avance de la comunicación ha sido espectacular. Hoy en día raro es el caso de una persona que no disponga de un dispositivo de computo, especialmente PC's, y otros utensilios útiles para muchos trabajos y para la vida diaria.

Algo parecido podemos comentar del avance del fenómeno Internet, ya que en pocos años se ha producido un impresionante incremento de su uso, tanto en hogares para uso particular como en oficinas y emplazamientos de trabajo, para uso laboral. Además, se debe tener en cuenta que se ha producido un incremento en la oferta de líneas de alta velocidad (ADSL, cable, y la más reciente 3G), así como un descenso en las tarifas de las mismas, lo que hace que cada vez más Internet sea un servicio al alcance de todos.
La unión de estos dos aspectos ha hecho posible la aparición de dispositivos y tecnologías cada vez más avanzadas, para poder dar cualquier tipo de servicio inimaginable. Uno de ellos es la posibilidad de recibir o mandar ordenes través de un navegador de Internet. Y esto último es lo que pretende este proyecto, poder acceder desde dispositivo PC a un servidor web que proporciona la posibilidad de encender o apagar un led, usando una tecnología específica.
Esta tecnología, como se comentará con detenimiento más adelante, no es más que un conjunto de la programación clásica, pero adaptadas a dispositivos con recursos más limitados que los de un ordenador personal, eliminando clases y librerías complejas y proporcionando sólo las básicas para desarrollar aplicaciones lógicas en este tipo de aparatos: textos, gráficos limitados, conexiones de distintos tipos, seguridad, etc.

Un ejemplo de uso de la aplicación que desarrolla este proyecto, sería ofrecer a un usuario determinado de un switch remoto, por ejemplo, el acceso continuo al switch de un led.
En definitiva, la idea es, usando las tecnologías que hoy en día tenemos a nuestro alcance, llegar a desarrollar una aplicación simple, pero efectiva, que pueda ser usada por cualquier persona y que pueda servir de utilidad para su vida diaria.

OBJETIVOS

General

- Desarrollar un servidor web arduino con ethernet shield el cual contendrá un programa servidor que se conectara con un programa cliente utilizando el protocolo http, dicho programa será capaz de encender y apagar un led.

Específicos

- Desarrollar el programa cliente-servidor utilizando hardware libre.
- Utilizar el protocolo http para poder realizar la comunicación entre el cliente y el servidor.

MARCO TEÓRICO

Modelo cliente servidor

En las redes modernas, un hosts pueden actuar como un cliente, un servidor o ambos. El software instalado en el host determina qué función tiene en la red. Los servidores son hosts que tienen instalado software que les permite proporcionar información y servicios, como correo electrónico o páginas Web, a otros hosts en la red. Los clientes son hosts que tienen instalado un software que les permite solicitar información al servidor y mostrar la información obtenida. Sin embargo, un cliente también se puede configurar como servidor simplemente al instalar software de servidor. El protocolo más utilizado por este modelo es el TCP orientado a las conexiones o peticiones de los clientes hacia el servidor.

Cliente:

Lo clientes son hosts que tienen instalado un software que les permite solicitar información al servidor y mostrar la información obtenida. Es decir que es quien inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

Servidor:

Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LANs o WANs, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes, aplicaciones web, Etc.

Imagen 1 *Ilustra el moldeo Cliente-Servidor donde el cliente envió una petición y el servidor emite una respuesta.*

Arduino

¿Qué es Arduino?

Arduino es una herramienta para hacer que los ordenadores puedan sentir y controlar el mundo físico a través de tu ordenador personal. Es una plataforma de desarrollo de computación física (physical computing) de código abierto, basada en una placa con un sencillo microcontrolador y un entorno de desarrollo para crear software (programas) para la placa.

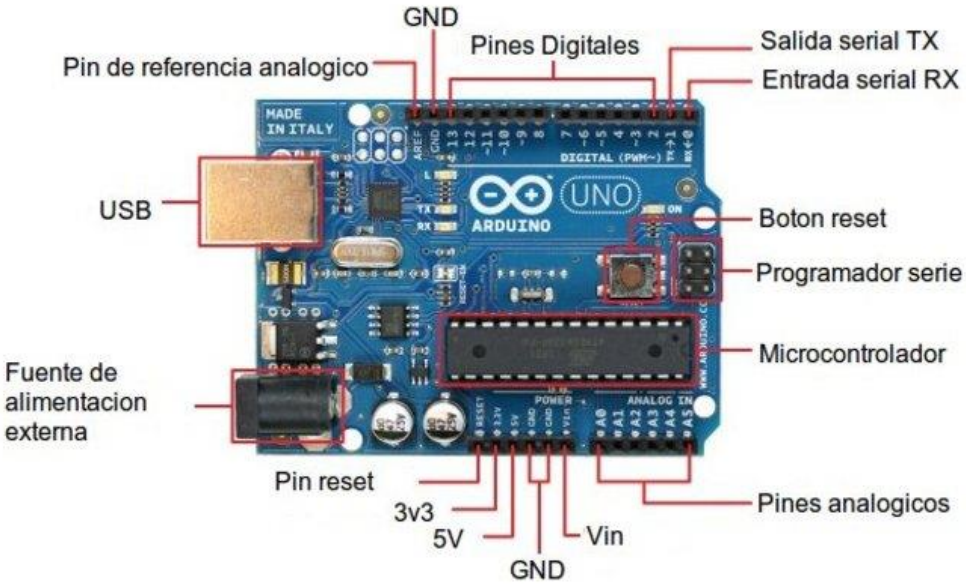


Imagen 2 *Ilustración de la placa arduino señalando la funcionalidad que tiene sus elementos*

Pines de tierra (GND)

Masa del circuito para pines, es decir es la tensión de referencia de 0V.

Pines digitales de entrada y salida

En estos pines conectaremos la patilla de dato del sensor/actuador. Desde ellos podremos leer la información del sensor o activar el actuador.

Hay 14 pines digitales que pueden utilizarse como entrada o salida con las funciones *pinMode(), digitalWrite(), y digitalRead()*. Operan a 5 voltios. Cada pin proporciona o recibe como máximo 40mA y disponen de una resistencia pull-up (desconectada por defecto) de 20-50 kOhmios. Ciertos pines son reservados para determinados usos:

Serie: 0(RX) y 1(TX). Utilizados para recibir (RX) y trasmitir (TX) datos serie. Están directamente conectados a los pines serie del microcontrolador. Utilizando estos pines podremos conectarnos con otras placas.

Interrupciones externas: 2 y 3. Estos pines pueden ser configurados para activar interrupciones.

PWM: 3, 5, 6, 9, 10 y 11. Proporcionan una salida de 8 bits en modo PWM.

SPI: 10-13. Estos pines soportan la librería de comunicación de dispositivos SPI.

LED: 13. Este pin está conectado con un led de la placa. Cuando se le asigne un valor HIGH se encenderá, en cambio si lo dejamos en LOW estará apagado.

Conector USB

Existen varios tipos de conectores USB, en concreto esta placa utiliza el tipo B hembra. Con lo cual se necesitará un cable tipo B macho ï tipo A macho (aunque se pueden utilizar otros este es el más extendido) que deberá conectarse a un conector tipo A hembra (por ejemplo a un ordenador o al cargador de un móvil). La placa se puede alimentar con la tensión de 5V que le proporciona el bus serie USB.

Cuando carguemos un programa a la placa desde el software de Arduino se inyectará el código del ordenador por este bus.

Botón Reset

Utilizando este botón podremos reiniciar la ejecución del código del microcontrolador.

ICSP (In Circuit Serial Programming)

Es un conector utilizado en los dispositivos PIC para programarlos sin necesidad de tener que retirar el chip del circuito del que forma parte.

Microcontrolador ATmega328

El microcontrolador es el elemento más importante de la placa. Es donde se instalará y ejecutará el código que se haya diseñado. Ha sido creado por la compañía Atmel, tiene un voltaje operativo de 5V, aunque se recomienda como entrada de 7-12V con un límite de 20V. Contiene 14 pines digitales de entrada y salida, 6 pines analógicos que están conectados directamente a los pines de la placa Arduino comentados anteriormente. Dispone de 32KB de memoria flash (de los cuales 512 bytes son utilizados por el bootloader). En la memoria flash se instalará el programa a ejecutar. El bootloader será el encargado de preparar el microcontrolador para que pueda ejecutar nuestro

programa. También tiene una memoria EEPROM de 1KB que puede ser leída o escrita con la librería EEPROM.

En la parte de procesamiento dispone de un reloj de 16Mhz y 2KB de memoria RAM.

Fuente de alimentación externa

La placa puede ser alimentada también mediante corriente continua suministrada por el conector jack de 3.5mm que podrá recibir entre 7 y 12V.

Pin de Reset

Podemos imitar el funcionamiento del botón reset suministrando un valor LOW(0V) para reiniciar el microcontrolador.

Pin de 3.3V

Desde aquí podremos suministrar 3.3V a los dispositivos que lo necesiten con una corriente máxima de 50mA. Es generada gracias al chip FTDI integrado en la placa.

Pin de 5V

Este pin saca una tensión de 5v del regulador de la placa. El regulador es necesario puesto que puede ser alimentada con distintos voltajes.
Diseño de un sistema de control domótico basado en la plataforma Arduino

Pin de Vin

Es el voltaje de entrada cuando se usa una fuente de alimentación externa (no tiene en cuenta la conexión USB). Se puede proporcionar voltaje a la placa a través de este pin, o en caso de que se esté utilizando una fuente de alimentación externa tomar el valor que está siendo suministrado.

Pines analógicos

Esta placa contiene 6 pines de entrada analógicos. Los elementos que se conecten aquí suelen tener mayor precisión que los digitales pero su uso requiere de una lógica levemente mayor. Más adelante se comentará el uso de un termistor analógico.

Módulo Ethernet

Es una placa que se acopla encima de la Arduino y permite establecer conexiones a internet mediante el estándar Ethernet que utiliza el protocolo TCP/IP. Podemos conectarla a un router utilizando un cable RJ45 y le asignará una dirección IP. Con esta dirección podremos abrir conexiones entre el servidor y la placa o de placa a
placa para enviar flujos de datos. Hay distintos chips y cada uno utiliza sus propias librerías. En nuestro caso hemos trabajado con el chip 28J60 que gasta las librerías

etherShield.h y *ETHER_28J60.h*

***Imagen 3**: muestra l imagen de una placa Ethernet Shield*

Es importante tener en cuenta que en el código de la placa la configuración de la Ethernet Shield ha de ser correcta. Debemos asignarle la dirección IP que le proporcione el router en caso de que utilice DHCP. También hay que poner una dirección MAC única para que el router conozca los distintos dispositivos conectados. Además podremos abrir un puerto mediante el que escuchará peticiones. Esta configuración se ha de realizar en la función *setup(),* es decir en la fase de inicio de la placa.

La función principal de la Ethernet Shield va a ser leer peticiones, en nuestro caso HTTP (puerto 80). Las peticiones en HTTP tienen el siguiente formato: GET /ruta_del_objeto HTTP/1.1 Para tomar la petición nos basaremos en la función *serviceRequest()* que nos devolverá una cadena con toda la parte del protocolo ya tratada, es decir, obtendremos sólo la ruta del objeto.

CREAR SERVIDOR WEB CON ARDRUINO

Materiales

1 Placa Arduino

1 Protoboard

1 LED

1 Ethernet

Alambre de cobre aislado

Cable USB (tipo impresora)

Pinzas para cortar y pelar cable

Maquina con sistema linux debian

Software de Arduino

Primero instalamos Adruino en la plataforma que necesitamos, en este caso lo aremos en debían
Donde utilizamos en comando

sudo apt-get install arduino

Abra el IDE Arduinos , lo podemos hacer de esta manera
arduino &
ó lo buscamos en las aplicaciones de electrónica

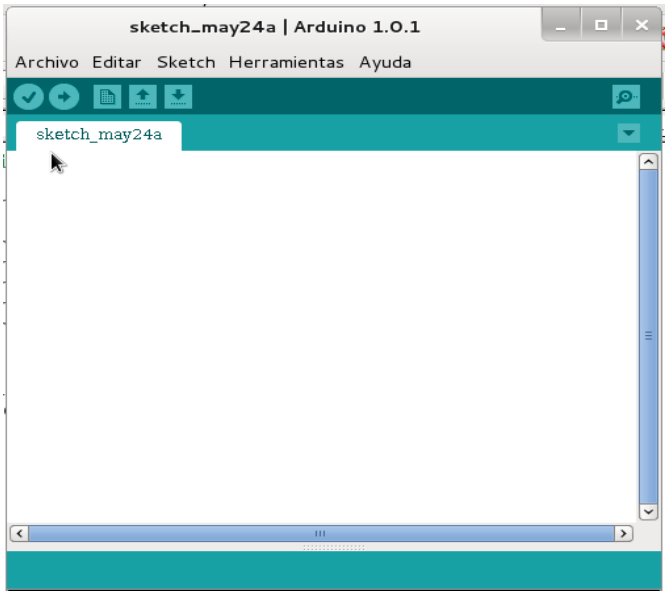


Imagen 4 *muestra la ventana de IDE de programación de ardruino*

Menú

La parte más importante se encuentra en Herramientas. Desde aquí podremos configurar el programa para que pueda comunicarse con la placa Arduino.

Pasando el ratón por Tarjeta aparecerá una lista con los tipos de placa Arduino que el programa comprende. Aquí seleccionaremos Arduino Uno o Diecimilia (para la Seeeduino) dependiendo de con cual estemos trabajando.

En el campo Puerto Serial seleccionaremos el que corresponda a nuestra placa que conectaremos mediante USB. Si utilizamos Windows el puerto tendrá un nombre del estilo COMx pero en Linux será /dev/ttyUSBx donde x es un número. En caso de que

aparezcan varios puertos serie y no sepamos cual es el de nuestra placa procederemos a desconectarla, anotamos los puertos que aparecen, reconectamos la placa y volvemos a mirar la lista de puertos. El nuevo puerto que haya aparecido será el de nuestra placa.

Botones comunes

Estos botones son accesos rápidos a ciertas acciones que también están disponibles mediante el menú. Los botones son los siguientes:

Verificar: comprueba y compila el código.

Cargar: además de compilar el código lo inyecta en la placa.

Nuevo: crea un nuevo sketch.

Abrir: abre un sketch previamente guardado.

Guardar: almacena en disco los cambios realizados en el sketch.

Monitor Serial: abre una nueva ventana desde la que podremos comunicarnos bidireccionalmente vía serie con la placa, es decir, podremos leer la información que nos envía o proporcionarla nosotros.

imagen5 *muestra la ventana donde aparece la información que se ha procesado.*

Editor de texto

En este área escribiremos la implementación (denominada por el programa sketch) para poder cargarla en la placa Arduino. El programa tiene 3 partes. La primera consiste en la inclusión de librerías y la declaración de constantes o variables globales que se podrán utilizar en cualquier función del programa. La segunda es el método *setup()* que es el encargado de inicializar los dispositivos conectados a la placa y será ejecutado solo al iniciar el sistema. La tercera parte consiste en el método *loop()* que ejecutará su código continuamente, es decir, en modo bucle. Aquí es donde se escribirá la lógica de la placa Arduino. Como el lenguaje es muy similar a C es posible crear

otros métodos para separar bloques funcionales y dejar ordenado el programa.

Área de mensajes

Muestra la situación del programa al haber utilizado uno de los botones comunes.

Consola de texto

Aquí aparecerán con mayor detalle los eventos del área de mensajes.

Empezando a programar

Estructura
El código tiene 3 partes principales:
ŏ La zona global
ŏ La función void setup()
ŏ La función void loop()

Zona global

Aquí será donde indicaremos a arduino los nombres de los pines y donde crearemos aquellas variables que queramos que existan en todo el programa. Aunque comprende todo lo que está fuera de las otras dos zonas, es recomendable agruparlo todo en la parte superior del código.

```
/*
fernando dagoberto piche ramirez
pedro antonio trejo noble
noris gamez
*/
#include <SPI.h>
#include <Ethernet.h>
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1,17);
EthernetServer server(80);
int led=6;
String estado="OFF";
```

Esta función se ejecuta cada vez que se inicia Arduino (incluyendo al pulsar RESET). Una de las operaciones que se realiza en void setup() es la de configurar de los pines que vamos a utilizar.

```
void setup()
{
Serial.begin(9600);
// Inicializamos la comunicación Ethernet y el servidor
Ethernet.begin(mac, ip);
server.begin();
Serial.print("server is at ");
Serial.println(Ethernet.localIP());
pinMode(led,OUTPUT);
}
```

Esta función es el corazón de los programas creados con arduino. Es una función que permanece en ejecución en forma de bucle infinito. Esto quiere decir que se ejecuta de comienzo a fin, de forma repetida, siempre.

```
void loop()
{
EthernetClient client = server.available();

if (client) {
Serial.println("new client");
boolean currentLineIsBlank = true;
String cadena="";
while (client.connected()) {
if (client.available()) {
char c = client.read();
Serial.write(c);
cadena.concat(c);

int posicion=cadena.indexOf("LED=");
if(cadena.substring(posicion)=="LED=ON")
{
digitalWrite(led,HIGH);
estado="ON";
}
if(cadena.substring(posicion)=="LED=OFF")
{
digitalWrite(led,LOW);
estado="OFF";
}

if (c == '\n' && currentLineIsBlank) {
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println();
```

```
//Página web en formato HTML
client.println("<html>");
client.println("<head>");
client.println("</head>");
client.println("<body>");
client.println
("<h1 align='center'>Servidor web Arduino</h1><h3 align='center'>LED controlado por Servidor
Web con Arduino y Ethernet Shield</h3>");
//Creamos los botones. Para enviar parametres a través de HTML se utiliza el metodo URL encode.
//Los parámetros se envian a través del símbolo '?'
client.println
("<div style='text-align:center;'>");
client.println
("<button onClick=location.href='./?LED=ON\' style='margin:auto;background-color:
#84B1FF;color: snow;padding: 10px;border: 1px solid #3F7CFF;width:65px;'>");
client.println("ON");
client.println("<b>LED = ");
client.print(estado);
client.println("</b><br />");
client.println("</b></body>");
client.println("</html>");
break;
}
if (c == '\n') {
currentLineIsBlank = true;
}
else if (c != '\r') {
currentLineIsBlank = false;
}
}
}
//Dar tiempo al navegador para recibir los datos
delay(1);
client.stop();// Cierra la conexión
}
}
```

Procedemos a conectar la placa arduino

Podemos utilizar fuente de energía como una batería ó la pc

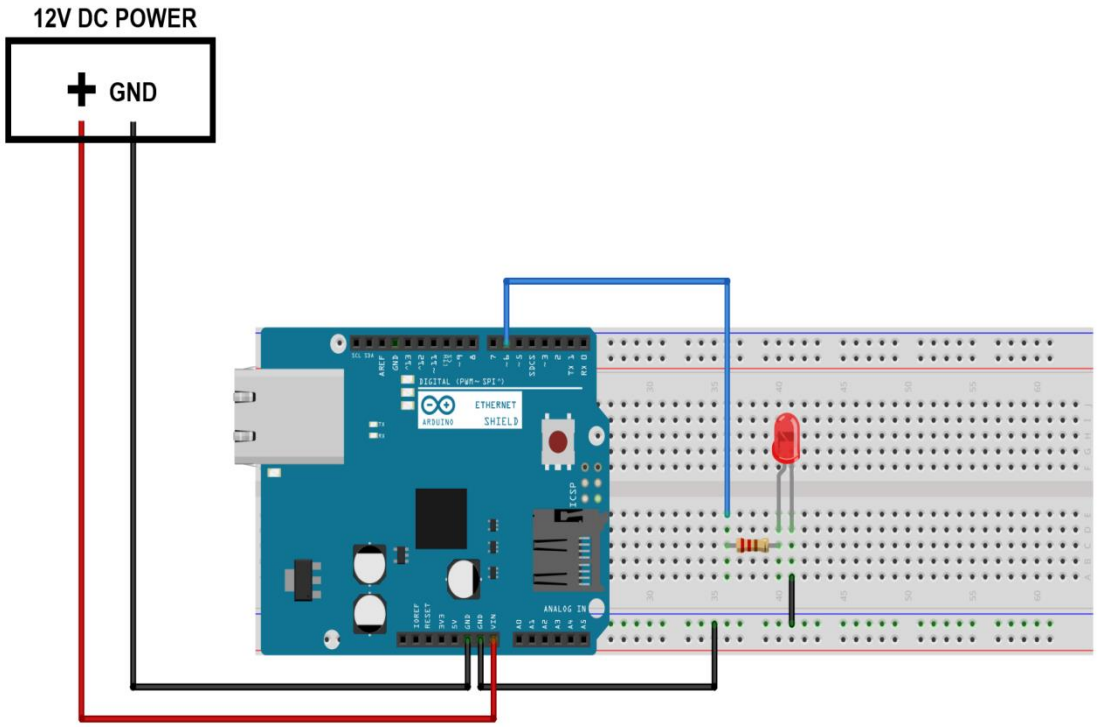


Imagen6 Ilustra la conexión de la placa con el protoboar y los cables de alimentación.

AHORA VAMOS A REALIZAR EL PROGRAMA

Primero debemos de instalar python

sudo apt-get install python2.7

Si no vemos si está instalado solo escribiendo la palabra python en la terminal

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
noris@Noris:~$ python
Python 2.7.3 (default, Mar 13 2014, 11:03:55)
[GCC 4.7.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Ahora en cualquier editor de texto empezamos a programar

```
        self.load = QtGui.QLabel('Loading...')
        self.status.addWidget(self.load)
        self.status.addWidget(self.prog)

        #Add widgets and layout to window
        v_box.addLayout(h_bar)
        v_box.addWidget(self.web)
        v_box.addWidget(self.status)
        #Shortcut
        self.short = QtGui.QShortcut(QtGui.QKeySequence(QtCore.Qt.CTRL + QtCore.Qt.Key_J), self.url)

        self.connect(self.previous, QtCore.SIGNAL("clicked()"), self.web.back)
        self.connect(self.next, QtCore.SIGNAL("clicked()"), self.web.forward)
        self.connect(self.refresh, QtCore.SIGNAL("clicked()"), self.web.reload)
        self.connect(self.stop, QtCore.SIGNAL("clicked()"), self.web.stop)
        self.connect(self.web, QtCore.SIGNAL("loadProgress(int)"), self.progress)
        self.connect(self.web, QtCore.SIGNAL("loadFinished(bool)"), self.loadComplete)
        self.connect(self.web, QtCore.SIGNAL("loadStarted()"), self.status.show)
        self.connect(self.short, QtCore.SIGNAL("activated()"), self.url.setFocus)

    def progress(self, porc):
        self.prog.setValue(porc)

    def openUrl(self, text):
        self.web.setFocus()
        self.web.load(QtCore.QUrl(text))

    def loadComplete(self):
        self.url.setText(self.web.url().toString())
        self.status.hide()
import sys, re
from PyQt4 import QtGui, QtCore, QtWebKit

class PyBrowser(QtGui.QWidget):

    def __init__(self):
        QtGui.QWidget.__init__(self)
        self.setWindowTitle('App WebArduino Redes I')

        v_box = QtGui.QVBoxLayout(self)
        #Navigation Bar
        h_bar = QtGui.QHBoxLayout()
        self.previous = QtGui.QPushButton(self.style().standardIcon(QtGui.QStyle.SP_ArrowLeft), '')
        self.next = QtGui.QPushButton(self.style().standardIcon(QtGui.QStyle.SP_ArrowRight), '')
        self.refresh = QtGui.QPushButton(self.style().standardIcon(QtGui.QStyle.SP_BrowserReload), '')
        self.stop = QtGui.QPushButton(self.style().standardIcon(QtGui.QStyle.SP_BrowserStop), '')
        self.url = QtGui.QLineEdit('')
        self.okUrl = QtGui.QPushButton(self.style().standardIcon(QtGui.QStyle.SP_DialogOkButton), '')
        self.okUrl.setFlat(True)
        h_bar.addWidget(self.previous)
        h_bar.addWidget(self.next)
        h_bar.addWidget(self.refresh)
        h_bar.addWidget(self.stop)
        h_bar.addWidget(self.okUrl)
        #Page Frame
        self.web = QtWebKit.QWebView()
        self.web.load(QtCore.QUrl('http://192.168.1.17'))
        #Status Bar
        self.status = QtGui.QStatusBar()
        self.prog = QtGui.QProgressBar()
```

```
app = QtGui.QApplication(sys.argv)
pybrowser = PyBrowser()
pybrowser.show()

sys.exit(app.exec_())
```

Ahora ejecutamos el archivo por medio de la terminal

Nos movemos donde está el archivo con el comando

```
noris@Noris:~$ cd Documentos
noris@Noris:~/Documentos$ ls
diseño  PowerD  proy  python.odt  redes  saber .odt  webardu.py
noris@Noris:~/Documentos$ python webardu.py
```

Imagen7 *Muestra cómo debemos ejecutar el documento*

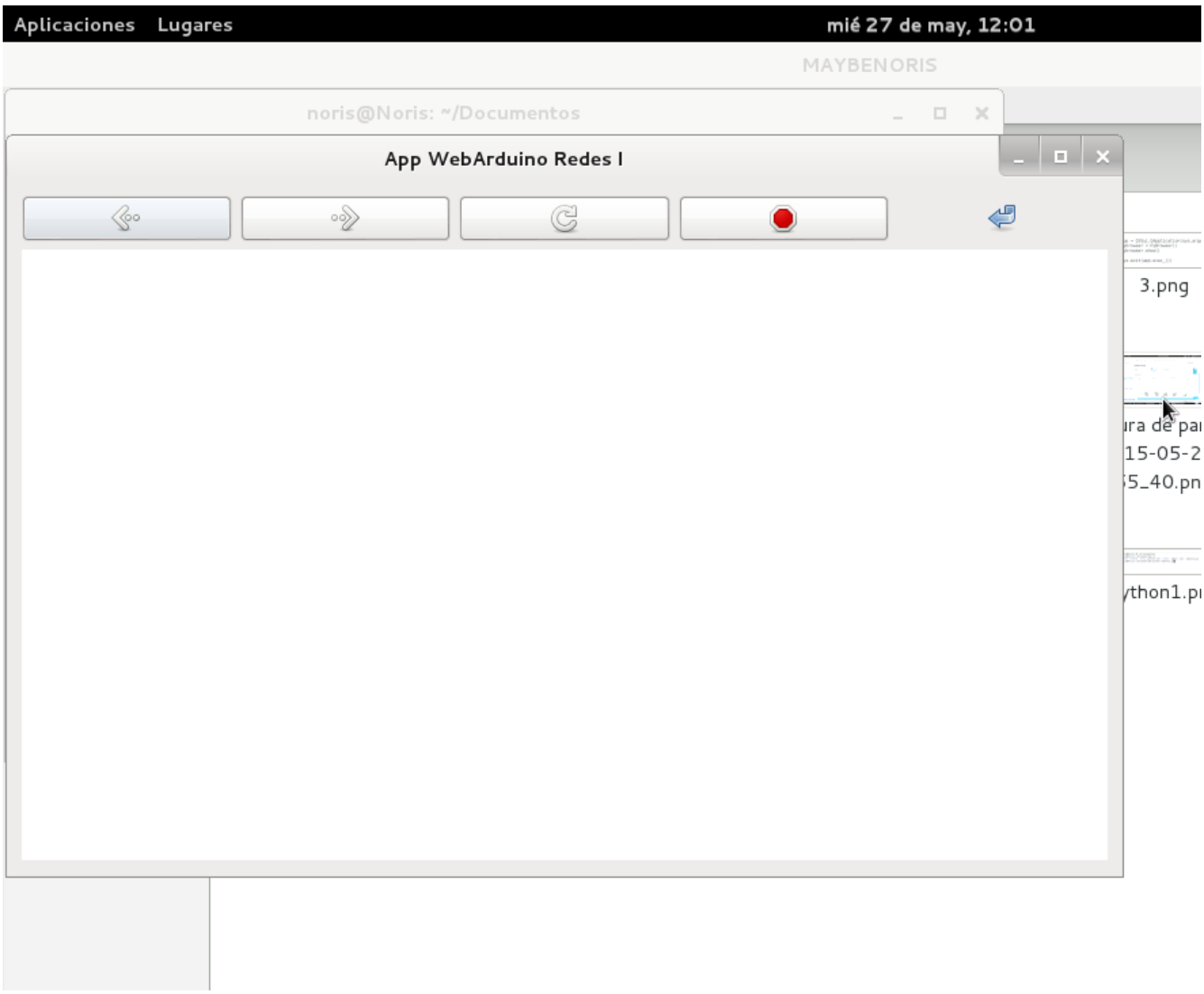
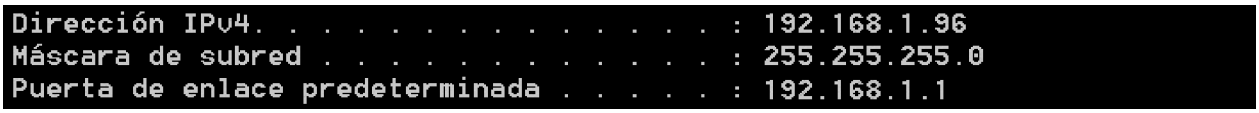


Imagen8 *Muestra la pantalla de los botones programados en donde encenderemos y apagaremos el LED*

Ahora para que el servidor de se conecte a tenemos que asignar una ip

IP tenemos que asignarle al Arduino, sabiendo que tiene que estar dentro del mismo rango que la IP de la puerta de enlace (Getaway). Para saber esto, en Linux nos vamos a la consola, escribimos Aquí nos pondrá la IP asignada del ordenador, la máscara subred y la puerta de enlace.

Hacemos ñ*ifconfig***ò** En nuestra terminal
Agregando la ip de arduino que en este caso sería:



En nuestro caso la puerta de enlace es 192.168.1.1, por lo tanto asignaré al Arduino la IP 192.168.1.100. Para comprobar si esta IP no está asignada a ningún otro equipo, haremos un PING a esta dirección. Si al hacer PING no obtenemos respuesta, quiere decir que la dirección IP está libre.

Recomendaciones

Un consejo: desconectar el cable USB del Arduino cuando no se use la comunicación serie porque pueden ocurrir fallos. Alimentar el Arduino con una fuente de alimentación a través del pin Vin o con el conector jack.

Conclusiones

Arduino es una herramienta muy popular que podemos utilizar para poder hacer proyectos electrónicos innovadores como la robótica , domótica, etc. Además una de las buenas características que tiene arduino es que es un hardware de código abierto donde podemos poner a prueba nuestra creatividad en el proyecto que elijamos realizar con esta herramienta. A diferencia de la Raspberry Pi que es una computadora funcional el arduino es una placa microcontroladora que sirve como opción de enseñanza en los centros escolares.

Bibliografía

TITULO: Arduino uno Desconocido
Autor: Desconocido
URL: http://arduino.cc/en/main/arduinoBoardUno
FECHA DE CONSULTA: 03/03/2015

TITULO: Arduino Ethernet Shield
Autor: Desconocido
URL: http://arduino.cc/en/Main/ArduinoEthernetShield
FECHA DE CONSULTA: 03/04/2015

TITULO: Referencia del lenguaje
.**Autor:** Desconocido
URL: http://arduino.cc/en/pmwiki.php?n=Reference/HomePage

FECHA DE CONSULTA: 15/03/2015

TITULO: Referencia del lenguaje
Autor: Desconocido
URL: http://arduino.cc/en/pmwiki.php?n=Reference/HomePage

FECHA DE CONSULTA: 15/03/2015

TITULO: Arquitectura cliente-servido
Autor: Desconocido
URL: http://es.wikipedia.org/wiki/Cliente-servidor
FECHA DE CONSULTA: 15/03/2015

Glosario:

ATmega8U2: Microcontrolador usado en las placas Arduino para establecer la conexión USB

Microcontrolador :es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria

Bootloader: Es un gestionador de arranque. Es una programación de que disponen todos los micros de Arduino para facilitar la comunicación con el PC y la programación.

Clavija: Parte macho de un conector.

Conversor A/D: Dispositivo electrónico encargado de convertir una señal analógica en una sucesión de bits o señal digital.

Firmware: Programación interna de un dispositivo que le permite controlar su propio hardware.

Pines: también llamados **terminales**. Son los contactos terminales de un conector o componente electrónico, fabricados de un material conductor de la electricidad. Estos se utilizan para poder realizar una conexión sin soldar.

Puerto COM: Un puerto serie o puerto serial es una interfaz de comunicaciones de datos digitales, frecuentemente utilizada por computadoras y periféricos, donde la información es transmitida bit a bit enviando un solo bit a la vez. Puerto muy usado hasta la aparición del USB (que es una versión del COM).

Sketch: Un sketch es el nombre que usa Arduino para un programa. Es la unidad de código que se sube y ejecuta en la placa Arduino.