

Proceso de Instalacion y Uso del Dispositivo Raspberry Pi B+ como Reproductor de Sonidos en una red LAN

CONOCIMIENTOS REQUERIDOS

IMPLEMENTOS

INSTALACION Y USO RASPBERRY PI

INSTALACION ARCHIVOS

INSTALACION Y USO SOUND EXCHANGE

INSTALACION Y USO REPRODUCTOR DE SONIDO

SOLUCION DE PROBLEMAS

TRABAJO FUTURO

REFERENCIAS

ANEXOS

CONOCIMIENTOS REQUERIDOS

MINIMOS

Ser entusiasta de uso del dispositivo Raspberry Pi. El proceso de instalación proporciona información de las actividades necesarias de forma que sean claras para toda persona entusiasta de usar el dispositivo para reproducir sonidos en una red local LAN.

DESEABLES

*Uso de sistema operativo Unix *Uso de SSH (Secure Socket Layer) *Programación en C++
*Uso del editor [vi](#) [1]. El proceso de instalación con estos conocimientos se realiza de forma más ágil e intuitiva.

IMPLEMENTOS

BASICOS

*Dispositivo Raspberry Pi B+ *Tarjeta de almacenamiento SD *Computador (Mac OS X, Linux o Windows) *Lector tarjeta SD con conexión USB *Programa [SD Formatter](#)
*Adaptador USB mini USB *Monitor HDMI o adaptador HDMI *Teclado USB *Ratón USB
*Instalados SSH Windows *Archivo socket_server *Carpeta rapidjson *Carpeta de sonidos Digiturno 5

INTERMEDIOS

*Cable Ethernet RJ45 *Programa [SoundExchange SoX](#) *Compilador GNU C *Artefactos programa reproductor de sonido

OPCIONALES

*Archivo socket_server.c *Archivo socket_client.c

INSTALACION Y USO

PREPARACION TARJETA SD

El dispositivo Raspberry Pi B+ requiere para su funcionamiento en computación y almacenamiento una tarjeta mini SD la cual hay que preparar para su uso en el dispositivo. El proceso de preparación consiste en el formateo de la tarjeta y la instalación de los programas y archivos requeridos en instalación y uso.



Tarjetas SD y Mini SD

El formateo de la tarjeta SD consta de los pasos siguientes:

*Conectar la tarjeta SD en el lector de tarjetas.

*Conectar el lector de tarjetas en un computador estándar con puerto USB.


*Desde el computador usar un programa formateador que reconozca la tarjeta y le de formato.

RECOMENDACION: Usar el programa SD Formatter. Si es así hacer los pasos siguientes:

*Instalar y abrir el programa SD Formatter. El programa reconoce la tarjeta en el lector cuando esta conectado al computador.

Format for SD/SDHC/SDXC Card

1 Select Card


Not found any media.

2 Select Format Option

☒ Quick Format

It formats the card quickly without erasing data.
 Data in the card may be retrieved after Quick Format.

☐ Overwrite Format

It takes time depending on the card capacity with overwriting and erasing data in the card.

3 Specify Name of Card

Name :

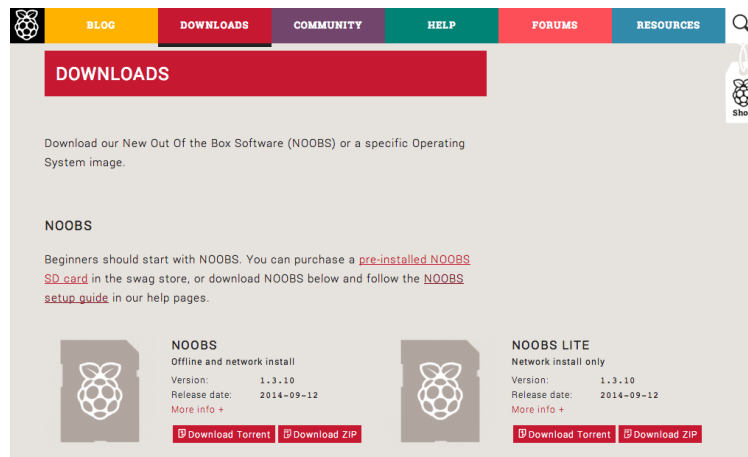
4 Click Format button

Interfaz SD Formatter

*Dar las siguientes opciones: **Overwrite Format** y nombre de la unidad en el campo correspondiente. El formateo se demora de acuerdo a la capacidad del computador donde se ejecuta SD Formatter.

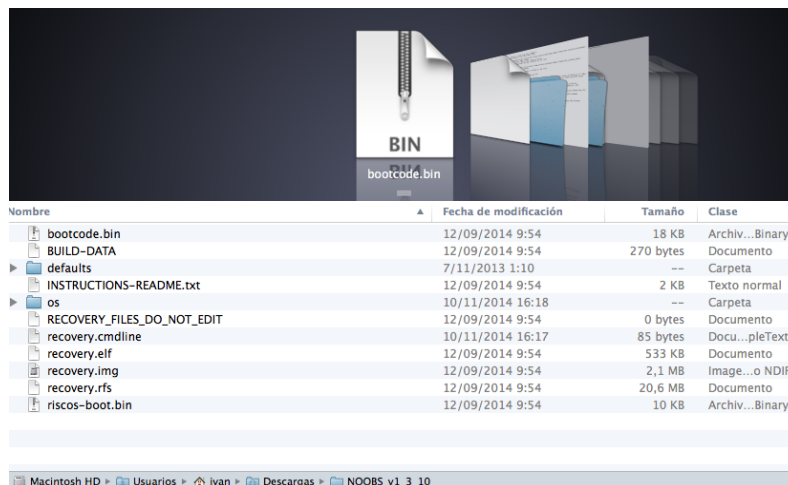
*Después de formatear, descargar e instalar la distribución basada en Linux. Una **distribución** es una solución que contiene los programas y archivos para funcionar como sistema operativo (SO) en un computador. Para el dispositivo se usan distribuciones basadas en el SO Linux. Estas contiene los programas y archivos que se almacenarán en la tarjeta y son usados en el dispositivo

como SO. Se recomienda usar la distribución [NOOBS](#) por su facilidad de uso e instalación. Esta contiene la distribución **Raspian** basada a su vez en la distribución Linux Debian.



Distribuciones disponibles en la página raspberry.org

**Al terminar de descargar la distribución, descomprimir y copiar los archivos en la tarjeta SD. Si al descomprimir se genera una carpeta, copiar los archivos ubicados dentro de esa carpeta al directorio raíz de la tarjeta SD.*



Manejador de archivos (Finder de Mac OS X)

INSTALACION DEL DISPOSITIVO

Después de preparar la tarjeta SD todo está listo para instalar y usar el dispositivo. Se toma como referencia el proceso de instalación dado por raspberry.org [2]. Se siguen los pasos siguientes:

**Conectar la tarjeta en el dispositivo.*

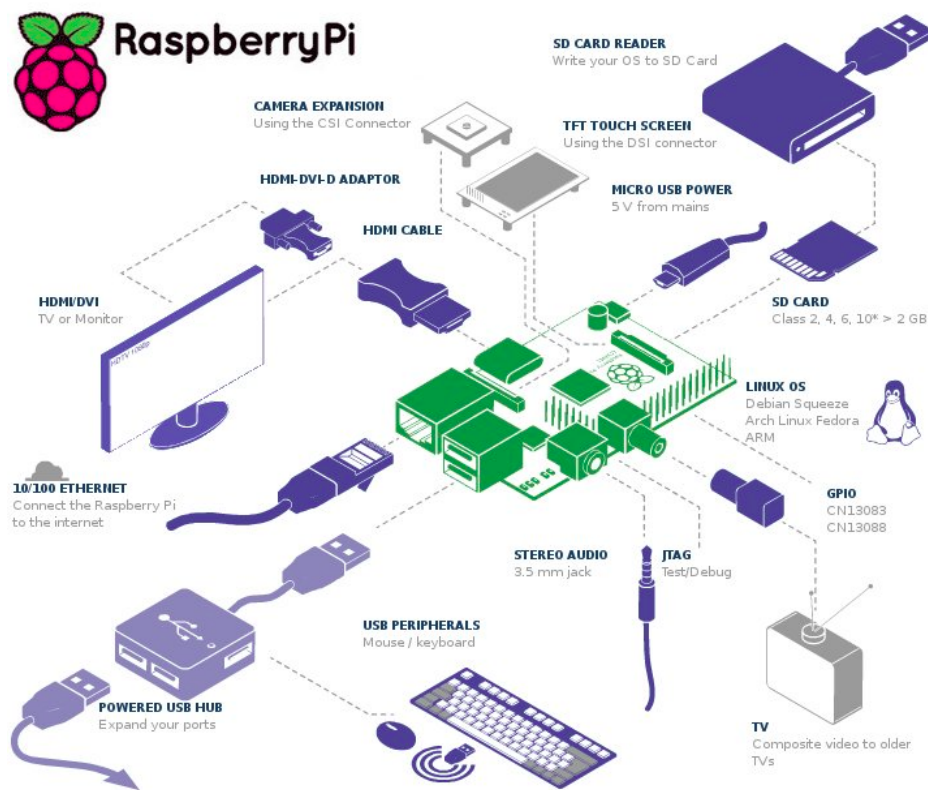
**Conectar monitor, teclado, ratón.*

**Si se desea usar el dispositivo en una red local, conectar cable Ethernet.*

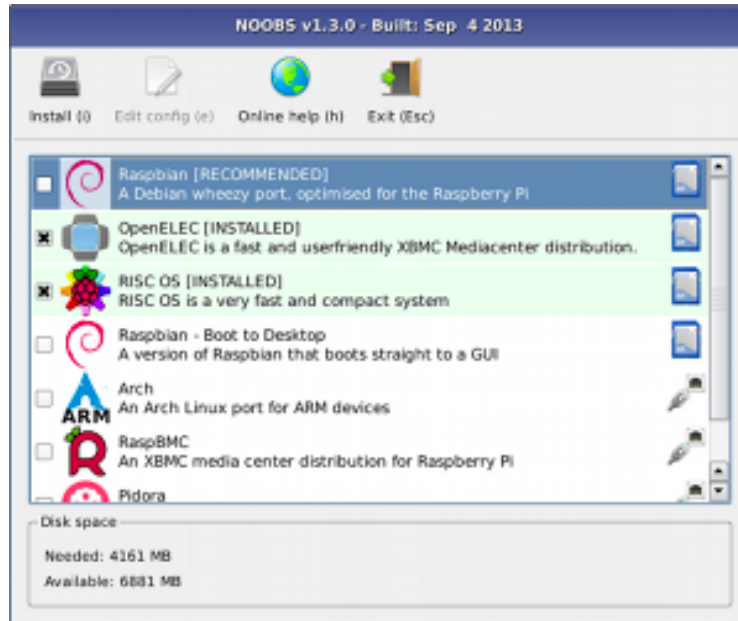
**Conectar el adaptador mini USB a un puerto USB o a una fuente de poder con el adaptador de corriente. Este adaptador de corriente debe generar máximo 1 amperio. Al conectar el adaptador, verificar que brillen unas luces rojo y verde en la tarjeta del dispositivo. En ese momento comienza el proceso de arranque de éste.*

**Verificar que en el monitor conectado se muestra la ventana de seleccionar sistema operativo. Si es así, seleccionar Raspbian y decir instalar. Y en ese momento se realiza el proceso de instalación del sistema operativo en el dispositivo*

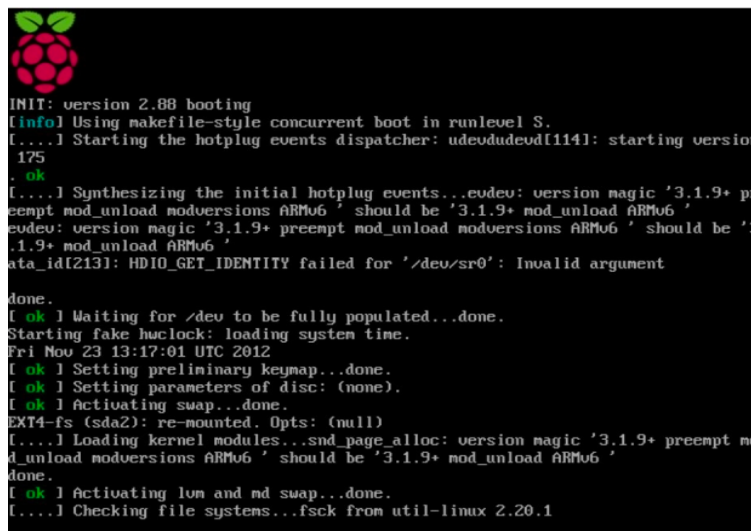
Al terminar la instalación, el dispositivo arranca nuevamente y muestra una consola que a su vez muestra los pasos del proceso de arranque.



Conexiones a un dispositivo Raspberry Pi



Ventana de instalación de Raspbian



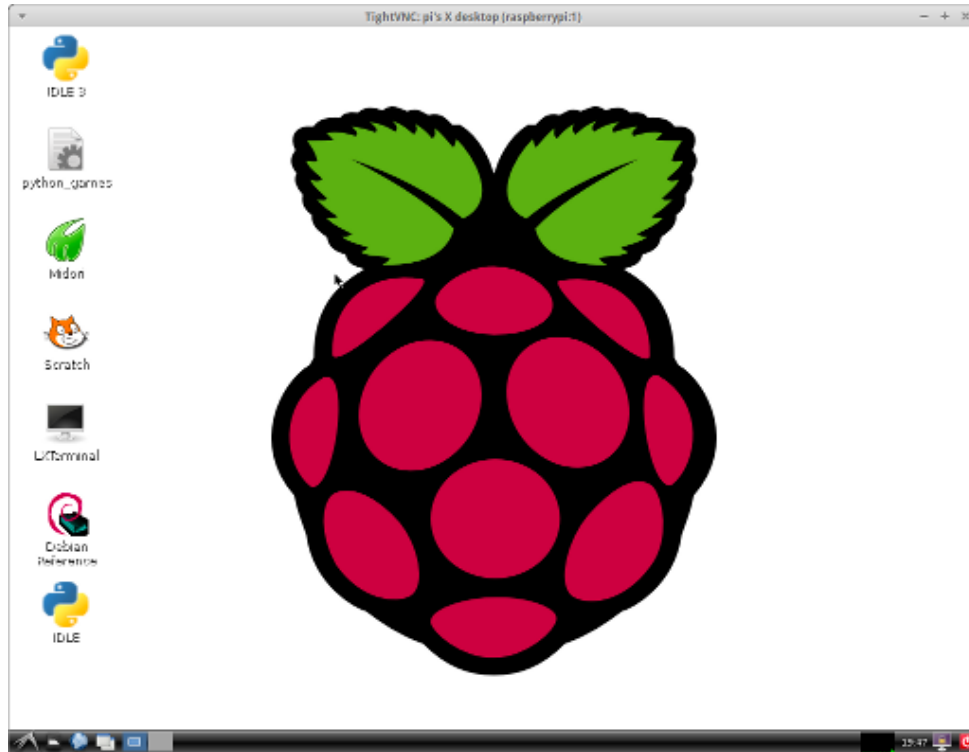
Arranque del dispositivo Raspberry Pi

USO DEL DISPOSITIVO

Al terminar el proceso de arranque después de la instalación y en general cada vez que se conecta el dispositivo a una fuente de poder por USB se solicita un usuario y una clave.

Dar los siguientes usuario y clave: login: pi, password: raspberry

Si se quiere usar el dispositivo con una interfaz gráfica, después de dar usuario y clave, dar el comando **startx**. En ese momento inicia el modo gráfico de Raspbian.



Screenshot del dispositivo con startx

En la interfaz gráfica se realizan las tareas básicas de cómputo como ejecución de programas de administración, terminal, manejador de archivos, navegación en Internet.

Para apagar el dispositivo, desconectar el adaptador mini USB.

USO DEL DISPOSITIVO DESDE UNA RED LOCAL LAN

Si se desea usar el dispositivo sin periféricos (monitor, teclado, ratón) y desde una red local, se realiza mediante el programa SSH (Secure Socket Layer). Este está disponible en las plataformas de las formas siguientes:

***Mac, Linux:** Por terminal mediante el comando `ssh -Y <login>@<IP raspberry Pi>`

***Windows:** Por la aplicación SSH desarrollada para plataformas Windows. Para instalar, en el instalador SSH, en la carpeta `ssh-tectia-client-6.0.6.19-windows-upgrd-eval` ejecutar el programa `ssh-tectia-client-6.0.6.19-windows.msi` y seguir las instrucciones. Por aplicación OpenSSH que se instala y se ejecuta desde línea de comandos. Sirve para todas las plataformas Windows excepto Windows Server

Para usar el dispositivo desde una red local se conectarme a un punto de red con el cable Ethernet. El sistema operativo asigna automáticamente una dirección IP al dispositivo la cual se usa con el usuario y la clave para ingresar al dispositivo.

Para verificar la dirección IP del dispositivo, desde una terminal o línea de comandos en el dispositivo dar el comando **ifconfig**. Entonces se muestra el contenido siguiente:

```
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    options=3<RXCSUM,TXCSUM>
    inet6 ::1 prefixlen 128
    inet 127.0.0.1 netmask 0xff000000
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
    nd6 options=1<PERFORMNUD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=27<RXCSUM,TXCSUM,VLAN_MTU,TSO4>
    ether e8:06:88:df:7a:e2
    inet6 fe80::ea06:88ff:fedf:7ae2%en0 prefixlen 64 scopeid 0x4
    inet 192.168.1.54 netmask 0xfffffc00 broadcast 192.168.3.255
    nd6 options=1<PERFORMNUD>
    media: autoselect (100baseTX <full-duplex,flow-control>)
    status: active
```

Donde dice en0 inet se especifica la dirección IP del dispositivo Raspberry Pi B+.

INSTALACION ARCHIVOS

Para reproducir sonidos debe crearse una estructura de archivos en el dispositivo siguiendo los pasos siguientes:

**Desde terminal en el dispositivo ejecutar `mkdir player`<Enter>*

**Copiar el archivo `socket_server` y la carpeta `rapidjson` en la ruta `/home/pi/player`. Esto se realiza desde el programa `SSH` instalado para Windows y por el programa `sftp` para Mac y Linux*

**Copiar la carpeta de sonidos en la ruta `/home/pi/player` del dispositivo.*

INSTALACION Y USO SOUND EXCHANGE

Para reproducir sonidos en el dispositivo, se usa el programa [Sound Exchange \(SoX\)](#). Este es una solución de distribución libre para los sistemas operativos Mac, Linux, Windows.

Para instalar y usar se puede descargar los artefactos ejecutables y copiar en disco. Sin embargo para el dispositivo que usa la distribución Raspbian basada en Linux Debian, se recomienda hacerlo desde el programa **apt-get**. Los pasos son los siguientes:

**En el dispositivo, desde una terminal ejecutar la instrucción siguiente: `sudo apt-get install sox libsox-fmt-all mc`*

**Si el programa solicita login y clave, dar `pi`, `raspberrypi`*

**Seguir las instrucciones durante la instalación.*

Para probar el programa ejecutar la instrucción siguiente: `play /home/pi/player/sounds/Neko_plays.mp3` Previamente hay que conectar un dispositivo de escuchar a la salida de audio si es un amplificador o audífonos. Al ejecutar puede escucharse el sonido en el dispositivo de audio y en la terminal se muestra el texto siguiente:

`/home/pi/player/sounds/Neko_plays.mp3:`

```
File Size: 148k      Bit Rate: 132k
Encoding: MPEG audio  Info: 2012-02-06T22:37
Channels: 2 @ 16-bit
Samplerate: 44100Hz   Album: Álbum de Juan Manuel Moreno
Replaygain: off       Artist: Juan Manuel Moreno
Duration: 00:00:08.99 Title: Neko_plays
```

```
In:99.7% 00:00:08.96 [00:00:00.03] Out:395k [      |      ] Hd:1.0 Clip:0
Done.
```

INSTALACION Y USO REPRODUCTOR DE SONIDO

Para conseguir un prototipo de reproducción de sonido en una red LAN se creó un programa reproductor de sonido **socket servidor** que usa los artefactos de Sound Exchange para reproducir sonidos ejecutándose desde un punto diferente de la red local. Un **socket** es una solución distribuida que consta de un programa cliente que se comunica con un programa servidor en otro computador que escucha peticiones del cliente para intercambiar información o ejecutar otros programas en forma distribuida [3]. Para esta solución se creó un socket servidor que se copia y ejecuta en el dispositivo y recibe

peticiones de forma que al recibir una petición reproduce un sonido con el programa Sound Exchange.

El socket servidor es un programa ejecutable creado en lenguaje C++ llamado **socket_server**. También está disponible el código fuente en el archivo llamado **socket_server.c**

Para ejecutar el programa se siguen los pasos siguientes:

**Instalar en el dispositivo el [compilador](#) de C++ GNU [4]. Para esto ejecutar en terminal el comando siguiente: apt-get install gcc*

**Seguir las instrucciones. Si pide login y clave, dar pi, raspberry.*

Para probar, ejecutar en terminal en la carpeta /home/pi/player el comando ./socket_server Al ejecutar muestra en terminal:

```
using port #8004
waiting for new client...
```

Si queremos hacer cambios al ejecutable, escribir los cambios en socket_server.c [5] y al final desde terminal ejecutar el comando siguiente para compilar y generar el nuevo ejecutable: g++ -o socket_server socket_server.c. Si los cambios se hicieron correctamente, no debe mostrar error.

Para probar que el socket servidor funciona, desde cualquier punto de la red LAN, desde otro computador de la red donde sea posible escribir programas, crear un socket cliente y ejecutar pasándole como parámetros la dirección IP del dispositivo, el puerto 8004 y como parámetro el texto siguiente:

```
{"NumeroReproducciones":3,"TiempoEntreReproducciones":"00:00:01","TonoReproduccion":"Neko_plays.mp3"}
```

Al ejecutar el cliente, debe mostrarse la traza siguiente en el dispositivo:

```
opened new communication with client
got
{"NumeroReproducciones":3,"TiempoEntreReproducciones":"00:00:01","TonoReproduccion":"Neko_plays.mp3"}
sending back 1
play WARN alsa: can't encode 0-bit Unknown or not applicable
/home/pi/player/sounds/Neko_plays.mp3:
```

File Size: 148k Bit Rate: 132k
Encoding: MPEG audio Info: 2012-02-06T22:37
Channels: 2 @ 16-bit
Samplerate: 44100Hz Album: Álbum de Juan Manuel Moreno
Replaygain: off Artist: Juan Manuel Moreno
Duration: 00:00:08.99 Title: Neko_plays

In:99.7% 00:00:08.96 [00:00:00.03] Out:395k [|] Hd:1.0 Clip:0
Done.

También está disponible el código fuente del socket cliente. Está en socket_client.c listo para modificar y compilar y generar un programa ejecutable de la misma forma que se hace con el socket servidor.

LANZAR EL SOCKET SERVIDOR AL INICIAR EL DISPOSITIVO

En la solución se espera que el socket servidor esté disponible en todo momento. Aunque ocurra un inconveniente como que no hay fluído eléctrico o el flujo de red fue interrumpido, aunque también hay que reiniciar el dispositivo, se espera que al reiniciarlo arranque el socket servidor también.

Para hacer posible que el socket servidor arranque al iniciar el dispositivo se siguen los pasos siguientes:

**Desde una terminal en el dispositivo ejecutar el comando siguiente: `sudo vi /etc/rc.local`*

**Ubicar el cursor antes de la línea `exit 0` con las teclas de flechas.*

**Digitar 'i'. En ese momento el editor se activa de forma que podemos agregar texto.*

**Agregar las líneas siguientes:*

`#socket`

`/home/pi/player/socket_server`

**Digitar ESC, y digitar `:wq<Enter>`. El cursor queda abajo a la izquierda de la pantalla. Este comando guarda el contenido y cierra el editor.*

**Reiniciar el dispositivo.*

Para probar que el socket servidor está arriba, después de reiniciar, en terminal, ejecutar el comando `ps -xa`. Se muestra la lista de procesos ejecutándose actualmente en el SO del dispositivo. Para los que usan Windows, este comando es el equivalente al programa de Windows **Administrador de Tareas**. En la lista verificar que salga en la lista el texto `/home/pi/player/socket_server`

SOLUCION DE PROBLEMAS

****La tarjeta SD se formateó pero al instalar la distribución en el dispositivo dice available 0mb en la ventana de seleccionar Raspbian.***

Después de formatear hay que ir a un computador, conectar la tarjeta en el lector de tarjetas en el computador y ejecutar una utilidad de discos con la opción de reparar unidades de discos.

En Mac hay un programa llamado Utilidad de Discos el cual permite revisar y reparar las unidades de disco conectadas al computador.

Al reparar la tarjeta SD y volver a conectar al dispositivo, debe mostrar available xxxmb .

****Al ejecutar startx desde red local me muestra:***

FATAL: Module g2d_23 not found.

^Cxinit: connection to X server lost

waiting for X server to shut down Server terminated successfully (0). Closing log file.

startx por el momento no se puede ejecutar desde consola por ssh

TRABAJO FUTURO

****Desde una plataforma diferente a Mac OS X formatear y reparar la tarjeta SD.***

****Ejecutar startx remotamente.***

****Asignar una IP fija al dispositivo sin necesidad que sea asignada automáticamente al conectar a una LAN y que sea su dirección fija oficial en la LAN.***

****Usar el dispositivo desde una red local sin autenticación. Esto es posible creando manualmente las llaves públicas y privadas que son usadas por el programa SSH.***

REFERENCIAS

[1] Editor vi <http://www.cs.colostate.edu/helpdocs/vi.html>

[2] Como instalar y configurar un Raspberry Pi
<http://www.raspberrypi.org/help/noobs-setup/>

[3] Sockets de red <http://sopa.dis.ulpgc.es/ii-dso/leclinux/ipc/sockets/sockets.pdf>

[4] Como instalar un framework de C++ en Raspberry Pi
<http://raspberrypi.stackexchange.com/questions/4813/how-to-install-c-compiler-on-raspberry-pi>

[5] Cliente Servidor en Raspberry Pi
http://cs.smith.edu/dftwiki/index.php/Tutorial:_Client/Server_on_the_Raspberry_Pi

ANEXOS

CODIGO FUENTE DE REFERENCIA SOCKET SERVIDOR

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>

void error( char *msg ) {
    perror( msg );
    exit(1);
}

int func( int a ) {
    return 2 * a;
}

void sendData( int sockfd, int x ) {
    int n;

    char buffer[32];
    sprintf( buffer, "%d\n", x );
    if ( (n = write( sockfd, buffer, strlen(buffer) )) < 0 )
        error( const_cast<char *>( "ERROR writing to socket" ) );
    buffer[n] = '\0';
}

int getData( int sockfd ) {
    char buffer[32];
    int n;

    if ( (n = read(sockfd,buffer,31) ) < 0 )
        error( const_cast<char *>( "ERROR reading from socket" ) );
    buffer[n] = '\0';
    return atoi( buffer );
}

int main(int argc, char *argv[]) {
    int sockfd, newsockfd, portno = 51717, clilen;
    char buffer[256];
    struct sockaddr_in serv_addr, cli_addr;
    int n;
```

```

int data;

printf( "using port %d\n", portno );

sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0)
    error( const_cast<char *>("ERROR opening socket") );
bzero((char *) &serv_addr, sizeof(serv_addr));

serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons( portno );
if (bind(sockfd, (struct sockaddr *) &serv_addr,
    sizeof(serv_addr)) < 0)
    error( const_cast<char *>( "ERROR on binding" ) );
listen(sockfd,5);
clilen = sizeof(cli_addr);

//--- infinite wait on a connection ---
while ( 1 ) {
    printf( "waiting for new client...\n" );
    if ( ( newsockfd = accept( sockfd, (struct sockaddr *) &cli_addr, (socklen_t*) &clilen)
) < 0 )
        error( const_cast<char *>("ERROR on accept") );
    printf( "opened new communication with client\n" );
    while ( 1 ) {
        //--- wait for a number from client ---
        data = getData( newsockfd );
        printf( "got %d\n", data );
        if ( data < 0 )
            break;

        data = func( data );

        //--- send new data back ---
        printf( "sending back %d\n", data );
        sendData( newsockfd, data );
    }
    close( newsockfd );

    //--- if -2 sent by client, we can quit ---
    if ( data == -2 )
        break;
}
return 0;
}

```

CODIGO FUENTE DE REFERENCIA SOCKET CLIENTE (Este es un código escrito para ejecutarse en un equipo Mac OS X)

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <netdb.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>

void error(char *msg) {
    perror(msg);
    exit(0);
}

void sendData( int sockfd, int x ) {
    int n;

    char buffer[32];
    sprintf( buffer, "%d\n", x );
    if ( (n = write( sockfd, buffer, strlen(buffer) ) ) < 0 )
        error( const_cast<char *>( "ERROR writing to socket" ) );
    buffer[n] = '\0';
}

int getData( int sockfd ) {
    char buffer[32];
    int n;

    if ( (n = read(sockfd,buffer,31) ) < 0 )
        error( const_cast<char *>( "ERROR reading from socket" ) );
    buffer[n] = '\0';
    return atoi( buffer );
}

int main(int argc, char *argv[])
{
    int sockfd, portno = 51717, n;
    char serverIp[] = "169.254.0.2";
    struct sockaddr_in serv_addr;
    struct hostent *server;
    char buffer[256];
    int data;
```



```

if (argc < 3) {
    // error( const_cast<char *>( "usage myClient2 hostname port\n" ) );
    printf( "contacting %s on port %d\n", serverIp, portno );
    // exit(0);
}
if ( ( sockfd = socket(AF_INET, SOCK_STREAM, 0) ) < 0 )
    error( const_cast<char *>( "ERROR opening socket" ) );

if ( ( server = gethostbyname( serverIp ) ) == NULL )
    error( const_cast<char *>( "ERROR, no such host\n" ) );

bzero( (char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
bcopy( (char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr, server->h_length);
serv_addr.sin_port = htons(portno);
if ( connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr)) < 0)
    error( const_cast<char *>( "ERROR connecting" ) );

for ( n = 0; n < 10; n++ ) {
    sendData( sockfd, n );
    data = getData( sockfd );
    printf("%d -> %d\n",n, data );
}
sendData( sockfd, -2 );

close( sockfd );
return 0;
}

```