1. CONFIGURACION DE LA RASPBEERY

Este documento va a explicar los pasos a seguir para configurar la raspberry, para poder trabajar de una manera comoda y facil.

1. El primer paso se formatea la SD en fat 32.

2. Se descarga **NOOBS** desde internet (la version de 2GB) y se decomprime en la raiz de la SD

3. Se introduce la SD en la raspberry, se conecta la raspberry a la red electrica, a un montior y a un raton y se inicia.

4. Se instala el sistema operativo Raspbian. Tardara mas o menos media hora

5. Una vez se muestre el escritoria **Raspbian** se apaga la raspberry

6. Vamos otra vez al ordenador y en la SD en el directorio /boot, añadimos un fichero vacio sin extension que se llame "**ssh**" (esto es para activar ssh en el arranque automaticamente)

7. En el directorio /boot creamos un fichero "**wpa_supplicant.conf**" con el siguiente texto:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=ES

network={
        ssid="nombre-de-tu-wifi"
        psk="password-de-tu-wifi"
        key_mgmt=WPA-PSK
}
```

8. Asi la raspberry se conectara al wifi automaticamente al arrancar, y podremos configurarla por ssh. Es recomendable que en ssid y psk pongamos una red creada con el movil para asi poder usarla en todos los sitios de trabajo.

9. Nos conectamos por ssh usando "ssh pi@raspberrypi.local" y como contraseña "raspberry"

2. Activacion del bluetooth para el SERVER

1. Ejecutar los comandos:
   sudo apt-get update
   sudo apt-get install bluetooth
   sudo apt-get install bluez
   sudo apt-get install python-bluez
   sudo apt-get install bluez-utils

2. Procederemos a emparejar el ESP-32 (o movil para pruebas)

sudo bluetoothctl
[bluetooth]# power on
[bluetooth]# agent on
[bluetooth]# discoverable on
[bluetooth]# pairable on
[bluetooth]# scan on
[bluetooth]# pair <MAC_BLUETOOTH_DEL_ESP32>
[bluetooth]# paired-devices ← Comporbamos que esta emparejado

3. Establecemos el canal (puerto)

sudo sdptool add --channel=22 SP
sudo sdptool browse local

- En caso que obtengas este error: *Failed to connect to SDP server on FF:FF:FF:00:00:00:No such file or directory*

sudo nano/lib/systemd/system/bluetooth.service

- Añade **--compat** a esta línea:

ExecStart=/usr/lib/bluetooth/bluetoothd –compat

sudo systemctl daemon-reload
sudo systemctl restart bluetooth

sudo chmod 777 /var/run/sdp

sudo sdptool add --channel=22 SP

http://kio4.com/raspberry/31_bluetooth.htm

3. Configurar script para el arranque

Antes que nada haremos git pull del repositorio para tener los codigos necesarios

Al arrancar la RASPBERRY se pone en modo emparejable y descubrible (esto lo hace el script **arrancarServidor.sh**), esto es que desde el movil por ejemlplo te saldra para vincular (en ajustes) una vez vinculada por ejemplo con el Bluetooth terminal podemos mandarle mensajes, en un futuro en vez de usar el terminal usaremos la app android. **En cambio la ESP32 la tendremos que vincular desde la RASPBERRY,** antes de que puedan comunicar. Para ello haremos estos pasos en la RASPBERRY:

sudo bluetoothctl

[bluetooth]# power on
[bluetooth]# agent on
[bluetooth]# discoverable on

[bluetooth]# pairable on
[bluetooth]# scan on
[bluetooth]# pair <MAC_BLUETOOTH_DEL_ESP32>

Ahora se va a poner en la Raspberry que se ejecute automaticamente al arrancar el script (arrancarServidor.sh) el cual pone el modo discover, activa puerto 22 y arranca el servidor, si **queremos parar el servidor (por ssh matamos el proceso con ps, y kill**), si no pasandole al servidor por **“BluetoothTerminal” el string (“killServer”)** se para tambien. **Para arrancarlo reiniciamos la raspberry o ejecutamos manualmente el script** (arrancarServidor.sh)

Añadiendo siguiente linea al crontab: (comando crontab -e), conseguiremos que el script se ejecute al arrancar el sistema, con un delay de 25 segundos para darle tiempo a arrancar los servicios:

@reboot sleep 25 ; /home/pi/Proyecto/Raspberry/arrancarServer.sh > /home/pi/salida 2> /home/pi/error

4. Configurado log para conocer estado

Se ha añadido un log en la ruta (/home/pi/Proyecto/Raspberry) para ver el log del servidor

Con el comando **tail -f Server.log** vemos el log en tiempo real

```
pi@raspberrypi:~/Proyecto/Raspberry $ tail -f Server.log
INFO:root:2020-12-05 19:09:00.423605 Raspberry> Mac: b8:27:eb:a6:38:58
INFO:root:2020-12-05 19:09:00.425336 Raspberry> Espero conexion
INFO:root:2020-12-05 19:09:17.692417 Raspberry> Conexion aceptada
INFO:root:2020-12-05 19:09:26.008105 Raspberry> Recibido: hola
INFO:root:2020-12-05 19:10:26.301402 Raspberry> Cierro conexion
INFO:root:2020-12-05 19:10:26.301982 Raspberry> Espero conexion
^C
```

5. Protocolo de comunicación con el servidor

¿Como sera la trama que le enviara la ESP32(servidor) a la Raspberry?

6. API DE GOOGLE (Servidor Google)

Se ha conectado una hoja de excel del google drive con el API de google para poder tenerla a especie de BBDD. El servidor raspberry ira actualizando los datos

Para usar la librería (“GoogleDrive.py”) se necesita instalar en la Raspberry:

pip3 install gspread oauth2client

Ademas se necesita un .json de configuracion, con las claves de autenticacion (solicitar a ximo)

https://www.twilio.com/blog/hojas-de-calculo-de-google-y-python

** FALTA ESTABLECER EL PROTOCOLO Y FORMATO DE LA BBDD

7. LCD

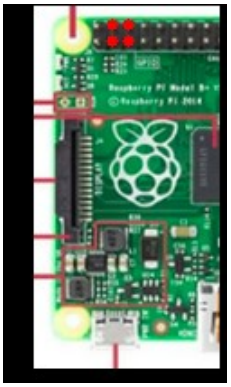
LCD usa:

VCC +5V

GND

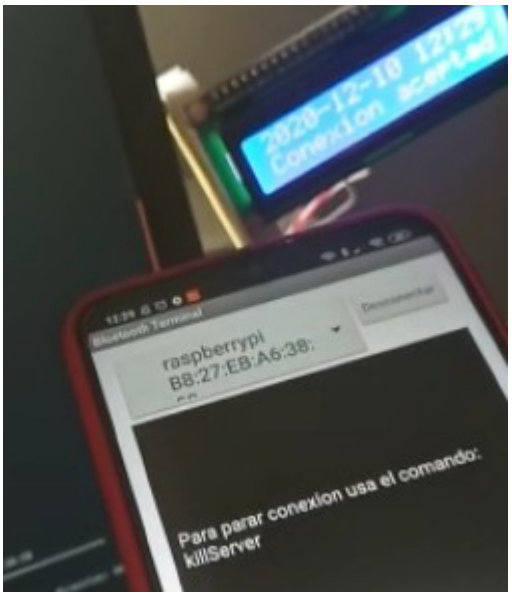
SDA (Gpio 2)

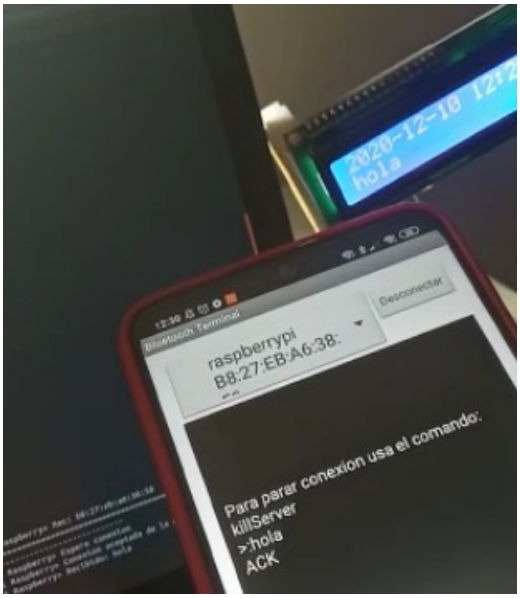
SCL(Gpio 3)



LCD de 2 lineas con shield I2C para usar comunicación serie I2C*

Se ha configurado el LCD para conocer el estado del servidor sin tener que usar el log interno de la raspberry, esto facilitara su uso. De todas formas el log muestra mas informacion que el LCD (por cuestiones fisicas de tamaño de pantalla) En las siguientes figuras se puede ver un ejemplo de su funcionamiento





http://www.multicopterox.es/configurar-un-lcd-16x2-por-i2c-en-la-raspberry-pi/

COSAS POR HACER: (XIMO)

- **Cuando este la trama subir al drive**
- **Almacenar datos en raspberry**
- **App android**