

Practicas LDH. Bloque 1º

21.11.2016

Manuel Jesús Bellido Lama
LDH

Resumen:

En estas prácticas del primer bloque de LDH daremos una serie de funcionalidades a las placas Arduino, Zpuino y raspberry, en este orden, y en este documento se detalla cómo han sido las prácticas 1 a 1 que se han hecho con cada una de las placas.

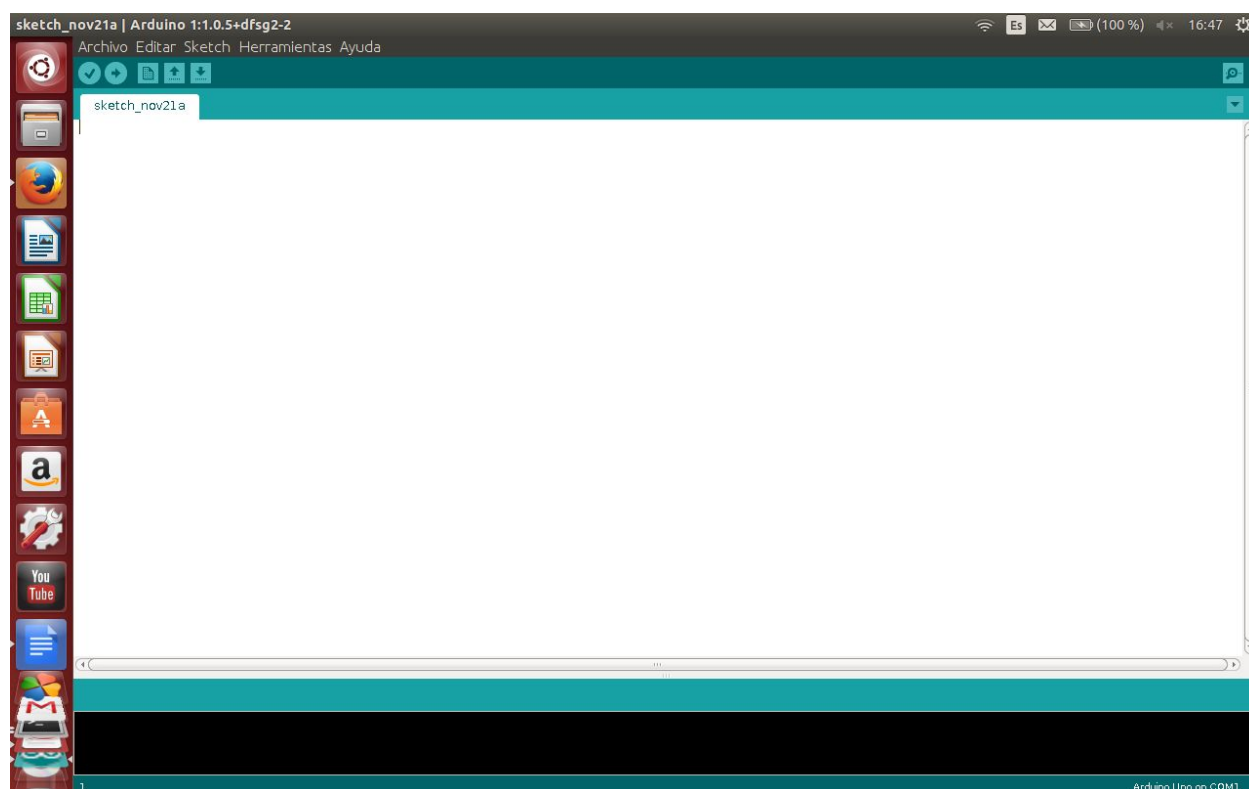
Estas prácticas no llevarán desde los primeros pasos con arduino de circuitos simples con tan solo el montaje y programación software de la placa hasta la raspberry con la que nos podremos proponer proyectos más interesantes instalándole un SO y haciendo que incluso sea la raspberry la que se comunique con arduino para llevar a cabo una funcionalidad mediante ambos como puede ser un servidor web que usa a arduino para determinar la temperatura, nivel de luminosidad, etc y a la raspberry para la creación de la página con el uso de python y html.



Arduino

Para usar la placa de arduino lo primero será descargarnos el IDE de arduino en el cual programaremos la placa y tendremos que configurarlo con el puerto correcto para comunicarnos y el arduino que estemos usando (UNO, Genuino, etc)

El entorno es este:



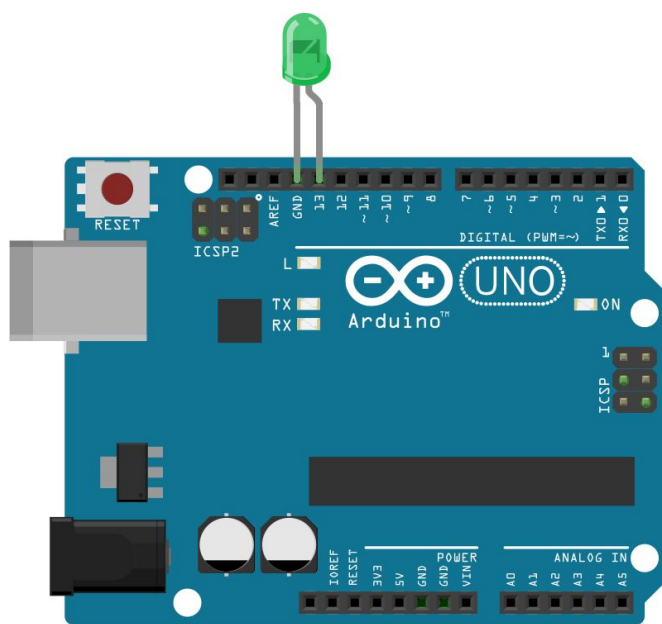
Y en el en las herramientas deberemos configurarlo como se ha dicho, para subir los proyectos e usa flecha, el segundo botón que aparece y en *archivo->ejemplos* podremos encontrar varios archivos de ejemplo para empezar nuestros proyectos sobre una base

Practica 1

La primera de las prácticas que se hacen en arduino son un conjunto en las que tendremos en primer lugar que probar uno de los programas de ejemplo de arduino el programa blink.

Con este programa un led que tenemos en el placa se encenderá y se apagará constantemente cambio de estado cada segundo con lo que estará permanentemente parpadeando, entendiendo el programa podemos cuales son las claves para que esto funcione, en primer lugar crear lo que sera el Led y darle el valor salida { pinMode(led, OUTPUT); }.

Tras esto dentro del bucle while que está por defecto(ya que este codigo se estará ejecutando permanentemente) encenderemos y apagaremos el led {digitalWrite(led, HIGH); digitalWrite(led, LOW)}. Con un tiempo de delay entre cada uno de estos de 1 seg.



Made with  Fritzing.org

Una vez visto y analizado el código que tenemos de ejemplo debemos ser capaces de modificarlo para realizar un par de cosas distintas, en primer lugar cambiar el modo de encender y apagar el led para en lugar de ser manual sea mediante el puerto serie y que nosotros le indiquemos en el ordenador si el led debe encenderse o apagarse, esto sera tan facil como crear un pequeño programa en python para el ordenador que se comunicara con el puerto en el que tenemos conectado el arduino (informacion que deberiamos conocer y poner en el programa a la hora de crear el objeto serial) y tras esto leer un comando desde el terminal que le diga "H" si queremos encender el led o "L" si queremos apagarlo, esta letra la

mandaremos al arduino y alli comprobaremos cual es la entrada de nuestro serial con

un `read.serial()` y en caso de "H" encenderemos y con "L" apagaremos tal y como hemos visto antes analizando el código del programa blink.

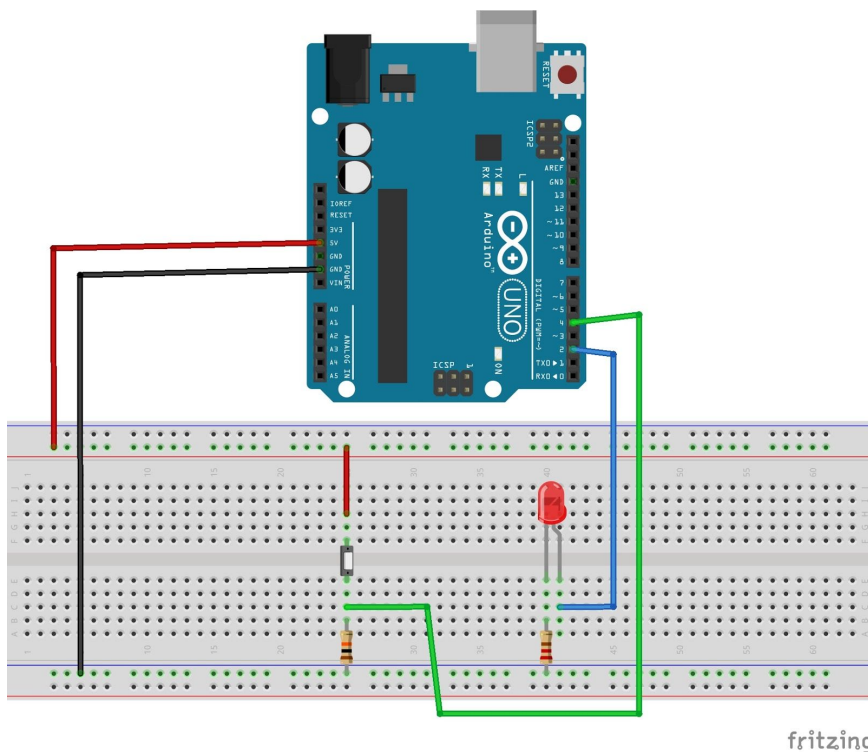
En lo que respecta a conexiones de momento no nos hace falta hacer ninguna y solo con el arduino y el ordenador el programa debería funcionar sin errores, aunque algunos extraños pueden surgir por ejemplo si llamamos `serial.py` al programa en python puede que no sea capaz de encontrar la librería serial que estamos usando para la comunicación y no podamos por tanto comunicarnos con el arduino.

En la página <https://geekytheory.com/arduino-raspberry-pi-raspduino/> se encuentran los tutoriales con todo el código completo e instrucciones para realizar este programa, cambiando que en lugar de un PC de sobremesa utilizan una raspberry.

2ª parte

El segundo cambio para esta pequeña práctica por fin empezaba a dar juegos con las conexiones usando ahora un pulsador para decidir si el led estaría encendido o apagado.


A primera vista es una tarea sencilla ya que de nuevo apenas tenemos que cambiar el código no requerimos esta vez de ningún tipo de comunicaciones y solo conectado los componentes el circuito debería funcionar perfectamente.



El problema es que además había que emplear las interrupciones en arduino con lo que se complicaba la tarea y al no haberlas usado antes no sabía si estaban bien o mal.

Sin embargo la conexión del pulsador resultó en mi caso bastante más engorrosa ya que a diferencia de lo que se ve en esta imagen se usaba un pulsador de 4 patillas que mal

conectado provocaba un cortocircuito y no hacía absolutamente nada, y al no estar seguro sobre las interrupciones parecía más probable que el problema fuera del código que del propio botón y la conexión del circuito.



Pues bien eso ocurrió con lo que el circuito no funcionaba independientemente de la cantidad de retoques en los cables posición del led, código, parecía que nada conseguía arreglarlo, hasta que por fin se conectó bien el pulsador y comenzó a funcionar perfectamente así que hay que tener cuidado en cómo se conectan los pulsadores la orientación de las patillas para que no provoquen un cortocircuito.

Tras esto se pedía un último retoque al circuito el cual solo requería en esta ocasión de retocar las conexiones para que en lugar de encenderse un led se encendieron una bombilla usando un relé para ello.

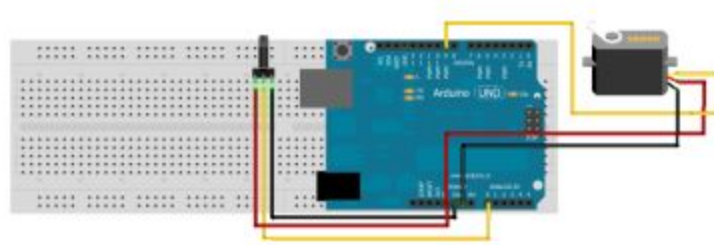
Con el relé hay que tener cuidado de los posibles calambrazos así que mejor no tocarlo demasiado, y basta con hacer exactamente lo mismo haciendo que ahora donde estaba el led este el relé para darle corriente a la bombilla y enchufando esta cualquier enchufe de los que hay disponibles.

El problema con esta práctica era que la bombilla por alguna razón no funcionaba y se podía ver que el relé estaba funcionando y correctamente conectado pero la bombilla no se encendía posiblemente porque estuviera fundida, etc, algunas de las bombillas no funcionan entonces con enseñar solo que el piloto del relé se enciende indicando que el circuito está funcionando como debería aunque la bombilla no, basta para pasar esta práctica.

Como podemos ver con esta serie de prácticas el encendido y apagado de un Led, puede dar para mucho juego y se puede sacar bastante de él con una placa como arduino según como lo queramos implementar necesitaremos una u otra cosa y así será para todo, simplemente un led parece lo más sencillo para empezar y lo que menos complicaciones ofrece a la hora de código y conexiones para poder centrarnos en otro tipo de complicaciones una vez aprendido cómo se maneja un led en Arduino analizando los códigos de partida que obtenemos de los ejemplos.

Practica 2

En la segunda práctica de este bloque de arduino teníamos como objetivo controlar la posición de un servomotor utilizando un potenciómetro con lo que conseguiremos que el servo vaya moviéndose a la par que giremos nuestro potenciómetro a un lado u otro.



Modificado por: [Fotogram](#)

Para esto usaremos además del servomotor y el potenciómetro una nueva librería en nuestro código de arduino la librería <Servo.h>.

Con esta libreria de forma similar a como declarabamos un led podremos declarar un servomotor y una vez creado {Servo Manolo.attach('miservo')}, tan solo tendremos que darle un valor, el valor leído por el potenciómetro que oscila entre 0 y 5 v, y según el valor que leamos moveremos el servo pero ha de ser de forma proporcional no directamente el dato leído ya que el valor sera mucho mas alto del que podamos alcanzar con el servo.

Ejemplo código

```
val = analogRead(pot); //Aquí le decimos que lea el valor del potenciómetro,
valor el cual oscila entre 0 y 1023
val = map(val, 0, 1023, 0, 180); //Traduce la lectura análoga (0, 1023) a grados (0°, 180°)
ServoManolo.write(val); //Mueve el Servo según la lectura análoga
```

Con esto el servomotor funcionará y girará tal y como le indiquemos con nuestro potenciómetro, desde lo que será su punto de origen los 0° hasta los 180°, siguiendo tutoriales de internet la práctica se hace con mucha facilidad ya que hay muchos ejemplos hechos de cosas iguales o similares en las que solo se ha de trastocar un poco

para dar con el correcto funcionamiento de nuestro ejemplo, Aunque después se nos pide añadir una nueva funcionalidad.

Modifiquemos un poco el código y ahora en lugar de el servo moverse como nosotros le indicamos con el potenciómetro se moverá como se lo indiquemos mediante el puerto serie, la idea vuelve a ser la misma que la última vez, cuando encendemos y apagamos el led solo que ahora en lugar de encender y apagar mandaremos un número.

El número que mandamos debe ser un valor comprendido entre 0 y 180 indicando directamente cual es el ángulo al que se debe mover nuestro servomotor, de nuevo lo meteremos por el terminal usando el programa de python, de nuevo tener cuidado y no llamarlo serial.py ya que podría acarrear problemas con la librería que estamos usando de python, en el código del arduino ahora nos deja de hacer falta el potenciómetro y necesitamos hacer como antes la lectura de los datos recibidos por el puerto serie, otra cosa que nos dejara de hacer falta será el mapeo de los valores que teníamos ya que el potenciómetro cogía valores de 0 a 1023 y nosotros teníamos que cambiarlos para que el servo los aceptara como máximo y mínimo de 0 a 180 pero ahora nosotros mandaremos los valores y mandaremos el número exacto, es decir, un número entre 0 y 180, con lo que directamente el número que nosotros estamos seleccionando desde la entrada será el que nos valga para el servomotor.

Una vez hemos realizado los cambios en el programa de python el arduino será igual de sencillo solo debemos en lugar de leer los datos de puerto en el que estuviera conectado el potenciómetro leerlos directamente del puerto serie, y ahora sin necesidad de hacer ni transformación ya tendremos el valor al que nuestro servomotor debe moverse.

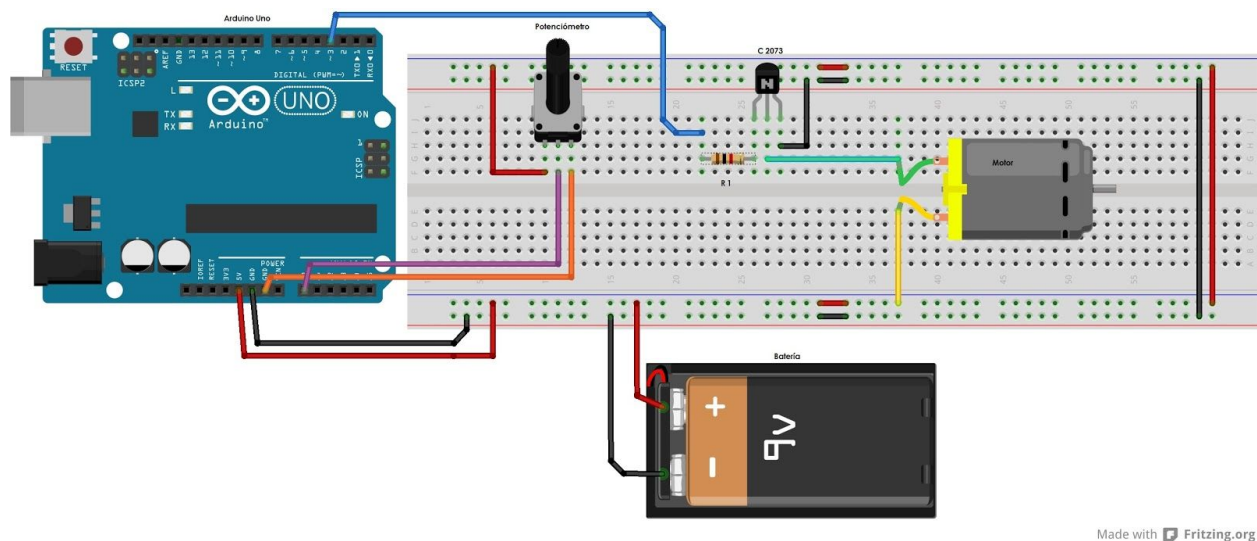
Además será más preciso ya que cambia directamente al ángulo que le decimos ni uno más ni uno menos sin tener que depender del posible error de estar haciendo lo con nuestras manos.

En lo que respecta al circuito ya deja de hacernos falta el potenciómetro para indicar hasta qué ángulo queremos movernos con el servomotor con lo que podemos tranquilamente eliminarlo del circuito y dejar tan solo el servomotor para que las comunicaciones desde el puerto serie se encarguen del resto.

Practica 3

En la tercera Práctica de este bloque de arduino seguimos con motores, aunque ahora no eran servomotores, sino un motor del cual teníamos que controlar su velocidad mediante un potenciómetro de nuevo.

Un concepto muy similar a la práctica anterior solo cambiando el servomotor por el motor, tal y como se puede intuir el diseño será muy parecido al anterior aunque aparecen




Made with Fritzing.org

un nuevo elemento. Ahora tendremos una pila para alimentar al motor para crear la señal del motor.

```
Void Setup(){
  Serial.begin(9600);
}

void loop(){
  float niv=analogRead(A0)/4;
  analogWrite(3, niv);
  //Serial.println (niv);

  delay(1000);
}
```



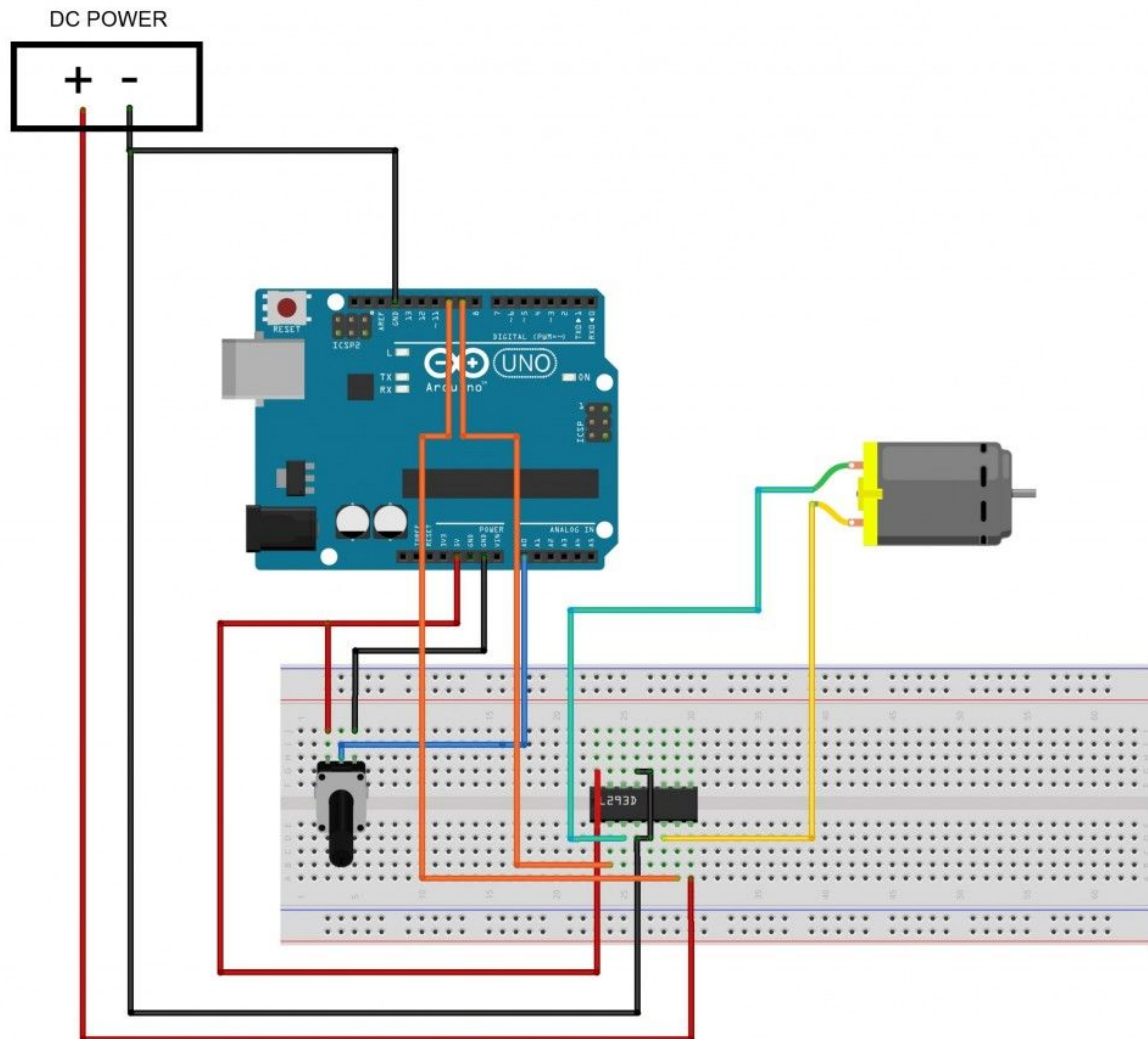
Muy parecido a parte de los nuevos objetos al código anterior, seguimos teniendo que hacer la reducción del valor para hacer que los valores sean correctos a la hora de llegar al motor, que amplificaba la señal para que el motor se mueva, una vez hechos el código y circuito no tiene ningún misterio y con tan solo girar el potenciómetro veremos como el motor empieza a girar cada vez mas fuerte segun giro del potenciómetro. Una vez más tras realizar esto se nos vuelve a pedir como modificación mandar los datos del potenciómetro por el puerto serie, pues bien solo habría que realizar los mismos cambios que antes y con usar el archivo de python que teníamos del anterior debería bastar perfectamente.

Por otra parte el `Serial.println()` que podemos ver comentado es bastante útil a la hora de ver si el programa funciona correctamente ya que es difícil apreciar la velocidad el motor, y este nos da directamente el valor que estamos obteniendo con tan solo abrir la ventana de monitor serie en el IDE de arduino.

Hay que tener en cuenta que ahora no se envían los ángulos sino la velocidad a la que girara el motor y que el rango de valores posibles ahora va desde los 255 de máxima hasta el 0 de mínima.


Practica 4

En esta práctica del bloque de arduino tendremos que volver a el motor, el circuito sigue siendo muy parecido pero para conseguir el giro de doble sentido tendremos que añadir



un nuevo componente el L293D, que sera el que se conecte al motor, y usaremos el pwm para hacer que el circuito funcione correctamente.

```
int pin2=9; //Entrada 2 del L293D  
int pin7=10; //Entrada 7 del L293D
```



```
int pote=A0; //Potenci6metro

int valorpote;      //Variable que recoge el valor del potenci6metro
int pwm1;    //Variable del PWM 1
int pwm2;    //Variable del PWM 2


void setup()
{
    //Inicializamos los pines de salida
    pinMode(pin2,OUTPUT);
    pinMode(pin7, OUTPUT);
}

void loop(){

    //Almacenamos el valor del potenci6metro en la variable
    valorpote=analogRead(pote);

    pwm1 = map(valorpote, 0, 1023, 0, 255);
    pwm2 = map(valorpote, 0, 1023, 255, 0); //El PWM 2 est1 invertido respecto al PWM 1
    //Sacamos el PWM de las dos salidas usando analogWrite(pin,valor)
    analogWrite(pin2,pwm1);
    analogWrite(pin7,pwm2);
}
```

Como podemos ver ahora tenemos que escribir en 2 pines en lugar de solo en uno y un valor distinto en cada uno de los para generar la se1al que necesitaremos, al igual que la primera parte de la pr1ctica anterior solo tenemos que girar el potenci6metro y ver como gira el motor hacia izquierda y derecha seg1n la posici6n del potenci6metro y cambiando la velocidad del giro tambi1n, todo en uno.



Es un circuito que resulta algo más complejo que el anterior hay que pararse para entender lo que se está haciendo en todo el circuito ya que controlamos no solo la velocidad sino también el sentido y metemos un par de componentes nuevos que no habíamos usado antes y se nos pueden hacer difíciles de entender de primeras, recuerda también a la práctica del servomotor ya que usa de nuevo la función map para dar los valores correctos de 0 a 255 en lugar de 1023 y después escribe directamente el dato esta vez no en el dispositivo sino en un pwm.

Practica 5

En esta quinta práctica con la placa de arduino comenzamos a usar un nuevo periférico, una pantalla LCD con la cual podremos ver mensajes que mandamos desde el puerto serie, o que hayamos escrito directamente en el programa, valores que queramos mostrar por la pantalla, etc.

El objetivo de esta primera vez que la usamos sera simplemente imprimir por pantalla nuestro nombre y LDH, para esto tendremos que aprender a usar la pantalla LCD, como inicializarla, escribir en ella, y cómo cambiar entre las 2 lineas que tiene para escribir, ya que en nuestro caso la pantalla LCD tiene solo 2 segmentos en los que podremos poner hasta 12 caracteres en cada uno, también tendremos que ver lo principal que es como imprimir un texto por la pantalla.

Una vez miremos algún ejemplo de cómo se inicializa la pantalla y mover el cursor el programa sera bastante facil de hacer, ya que las funciones son muy sencillas e intuitivas, para empezar inicializamos el lcd, luego tenemos que hacer un print directamente de nuestro nombre ya que por defecto el lcd empieza en la posición 0,0 que es la esquina superior izquierda y por tanto donde debemos empezar a escribir, para escribir solamente debemos llamar a la función `print(Texto_a_escribir)`.


Tras esto llamaremos a la función `setCursor(0,1)`. De esta forma cambiaremos la posición de cursor para que este a la izquierda de nuestra 2ª línea, con lo que podremos ver directamente por la pantalla:

Nombre_Usuario

LDH

Una vez entendemos cómo funcionan las pocas funciones que vamos a usar y en qué situación debemos usar cada una debemos modificar un poco el codigo para que ahora en lugar de imprimir simplemente esas 2 cadenas sea capaz de leer una cadena enviada desde el puerto serie con el ordenador (para lo que usaremos de nuevo un programa de python) e imprimirla por la pantalla en la posición 0,0.

Con lo que ya sabemos de esta y anteriores prácticas, esta modificación no tiene ninguna dificultad, simplemente modificamos el programa de python anterior eliminando las comprobaciones de si es uan L o una H y simplemente enviamos el mensaje con un `write`.



Y en el programa de arduino modificamos para que dentro del while haga un read del puerto serie (`Serial.read()`) y guardamos el resultado en un tipo string, después de esto debemos volver a poner la posición del cursor como la (0, 0) para que vuelva a escribir desde el principio de la pantalla siempre en lugar de que los mensajes se vayan acumulando con este dato hacemos una nueva llamada a función `lcd.print` y veremos el resultado por la pantalla LCD, para comprobar que todo funciona ejecutamos el programa de python y subimos el proyecto a arduino y escribimos por el terminal para ver que funciona, tal y como esta los caracteres sobrantes no se borran es decir si escribimos una palabra de 5 caracteres y otra luego de 3 se quedarán los 2 últimos de la anterior para que se borre la pantalla deberemos hacer un borrado de la LCD justo antes de la llamada a `printf` esto se hace con `lcd.clear()`, además con esta función la posición del cursor de nuestra pantalla lcd también vuelve a ser la 0,0 con lo que volvemos a escribir justamente desde la esquina superior izquierda y no necesitamos usar como estábamos haciendo la función `setCursor(0, 0)` para devolver el cursor a su posición original.

Practica 6

En esta sexta y última práctica con arduino tendremos que volver a usar la pantalla lcd, ahora haciendo un contador con ella en el que tendremos que visualizar cada numero nuevo en una de sus filas intercambiando, además el contador deberá llegar contando desde cero hasta cincuenta y nueve y los aumentos deben ser de segundo en segundo.

La práctica no tiene ninguna complicación sabiendo lo que ya sabemos de las anteriores prácticas, solo tendremos que realizar un bucle en el que ir incrementando el valor de un contador de forma que si es mayor de cincuenta y nueve tenga que resetear a cero de nuevo aparte de eso que no resulta complicado usando las funciones setCursor tendremos que hacer que si el numero es par el cursor esté en (0, 0) por ejemplo y si es impar en (0,1) haciendo esto tendremos la practica lista probar y ver cómo funciona.

Si se tiene algún código del primer ejercicio como el de hacer que aparezca el texto LDH que además va antes del bucle tendremos que hacer un clear pero no durante el bucle si no antes, justo antes de entrar, o mas sencillamente un eliminar o comentar el `print("XXXX")` para que no aparezca en nuestra pantalla, también hay que tener cuidado con una cosa mas. Una vez que la función da la vuelta y empieza desde 0 de nuevo los dígitos que había en la segunda columna se quedarán ahí a menos que de nuevo hagamos un clear, así que cuando el contador sea mayor que 59 tendremos que tener cuidado de hacer un clear también para dejar la pantalla lcd limpia.

ZPUino

El entorno de ZPUino es muy parecido al de arduino en principio, para la parte de programación software, sin embargo aquí también tendremos que decidir cómo se comporta internamente la placa de papilio y esto lo haremos en un entorno completamente distinto, Design lab, parecido al usado en cursos anteriores para programar con verilog, y en el que decidiremos los componentes que tendrá el circuito y como estarán estos conectados.


Practica 1

En esta primera práctica del bloque de ZPUino se nos pedirá solo desarrollar un tutorial explicado en la web, en el tutorial nos explican cómo empezar a usar la placa, creando primero el circuito FPGA en el que más tarde cargaremos el programa que en este caso constara simplemente de un led y un botón para encenderlo y apagarlo.

Lo primero que tenemos que hacer es crear un nuevo circuito FPGA, es importante saber que en ZPUino es importante no dejar el nombre que nos aparece por defecto ni poner uno que pueda causar problemas, esto no es difícil pero si no se sabe puede llevar a problemas como me paso a mi, por ejemplo en mi caso no podía subir a la placa el circuito que había diseñado debido a que había dejado el nombre que aparece por defecto.

Una vez creado el new FPGA pulsaremos sobre edit circuit, simplemente siguiendo los pasos del tutorial y llegaremos a tener nuestro circuito hecho, a la hora de coger los WING_AL no es necesario coger los del tutorial lo que si hace falta es saber donde se encuentran los que estamos usando, ya que en estos puertos tendremos que conectar nuestro botón y led además de a un GND.

El inversor que usamos es para que el led se encienda y se apague cuando pulsemos el botón, invirtiendo el valor que tendrá este si es uno a cero y si es cero a uno, así de simple.



Con todo el circuito listo simplemente generamos el fichero de programación, y despues muy importante tenemos que cambiar el nombre de los ficheros, en la carpeta del proyecto que estamos realizando tenemos que eliminar el archivo papilio_one_500k.bit y llamar EXACTAMENTE así al fichero que tenia el mismo nombre pero con mayúsculas, si no hacemos esto el programa no se cargará correctamente en su lugar se estará cargando otro circuito que no hace para nada lo que esperábamos y no nos sirve.

Con el archivo ya cambiado solo nos queda cargar el circuito y terminar probando que el led se enciende y apaga cuando pulsamos el botón tal y como esperábamos para pasar al siguiente tutorial de ZPUino.

Practica 2

En esta practica 2 de el bloque de ZPUino tendremos que no solo usar la creación de un nuevo circuito sino también utilizar como ya hemos hecho en arduino algo de código para darle una funcionalidad a este circuito, en este caso el código que usaremos es de un ejemplo llamado Papilo_QuinckStart, debemos en primer lugar analizar el código del circuito, este circuito enciende y apaga un led cada cierto periodo de tiempo o si conectando un botón de la manera apropiado lo pulsamos queda directamente encendido de forma permanente.


Luego en el circuito, volvemos simplemente a seguir de nuevo los pasos del tutorial para crear el circuito que necesitamos.

Veremos que esta vez ya hay un circuito creado y que nosotros solo debemos continuar el circuito modificando un par de detalles, tras esto, tan solo volvemos a seguir los pasos anteriores de generar el archivo y cambiarle el nombre, muy importante esto se hace siempre al igual que lo de no dejar los nombres por defecto ya que no funcionara nada con cualquiera de estos pequeños fallos, que aparentemente no tienen importancia para alguien que acaba de empezar.

Una finalizado y comprobado que funciona correctamente conectado un botón y un led a los puertos correctos podemos pasar a un par de modificaciones que hay que hacer.

La primera modificación es la más simple y consiste solamente en modificar el archivo para que cuando se pulse el botón y el led debe quedar parpadeando quede permanentemente apagado, esto es tan fácil como irnos a la parte de código en la que seleccionamos las acciones que tomamos según el valor del "botón" y cambiar en la que el led parpadea dejando solamente la línea en la que el led queda a LOW. con esto ya deberíamos haber terminado y de nuevo cargamos el programa y comprobamos que funciona.

Para hacer este cambio y el siguiente de forma rapido es importante previamente haber entendido que hace el código en cada parte, tal y como se especifica, un análisis



previo del código de ejemplo nos ayudará a entender mejor lo que pasa evitarnos quebraderos de cabeza y hacer los siguientes pasos más rápidos y provechosos para nosotros.

La segunda modificación es una que hemos hecho ya muchas veces y a estas alturas resulta muy fácil ya que tenemos el programa y deberíamos saber cómo funciona, es de nuevo el manejar el encendido y apagado del led vía puerto serie en lugar de hacerlo por hacerlo mediante el botón.

Una vez más tendremos que usar el programa de python en nuestro equipo para mediante un terminal pasar los datos a la placa de papilio, el código debería ser prácticamente igual que el de la primera vez que lo hicimos con arduino y el led, cambiando tal vez el puerto, y en el caso de la placa el código es también muy parecido.

De nuevo eliminamos la parte del botón y en su lugar usamos un `serial.read()`. Si el valor de este es "L" tendremos que hacer que el valor del led sea LOW si en lugar de "L" es "H" encenderemos el led haciendo que su valor sea HIGH y listo.

De nuevo comprobamos por última vez ahora que todo funciona correctamente haciendo unas cuantas pruebas en el terminal y observando los resultados en el led que tenemos conectado, una vez que tenemos el visto bueno de que todo funciona correctamente ya estamos listo para la siguiente práctica de ZPUino.

Practica 3

La tercera práctica de este bloque de ZPUino consiste en añadir un periférico al Soc de ZPUino y utilizarlo posteriormente con un sketch.



Para esto seguiremos un tutorial para crear un nuevo SoC para ZPUino en el que se explicara paso a paso lo que debemos hacer. El tutorial hay que seguirlo paso a paso y sin prisas, varias personas se saltaban el primer paso pensando que era una introducción a la práctica, hay que hacerlo todo para que la práctica funcione como esperamos.

Siguiendo la práctica llegaremos a editar el circuito y una vez aquí veremos lo que vamos a añadir al ZPUino

que sera un UART con lo que nos podremos conectar a el con el cable que vemos en la imagen conectado de forma apropiada la cual se nos indicará también un pdf que podemos encontrar en la práctica, con este podremos conectarnos a la placa usando un cable distinto conectado por USB al PC tambien y a pesar de no poder cargar codigo en la placa podremos comunicarnos con el PC y ver que la comunicación funciona.

En la UART simplemente elegimos lo que seran las entradas y salidas en nuestro caso serán WING_AL 0 y WING_AL 1.



Una vez más generamos el bit File y añadimos ciertas líneas de código que aparecen en el tutorial tras esto esta completo.

Antes de cambiar el cable por el nuevo debemos hacer tal y como nos ponen en la práctica el sketch de la primera práctica en el que nos comunicamos mediante el PC con el programa de python para controlar el encendido/apagado de un LED, con esto podremos comprobar si funciona adecuadamente la UART.

Cuando tenemos esto completado solo falta poner en práctica la nueva conexión a ver si funciona, para esto simplemente cambiamos el cable usb con el que estamos conectados por el cable de la conexión UART y eso si, tenemos que cambiar el puerto en el que estamos conectados para que funcione ya que habremos cambiado de puerto con el nuevo cable y tenemos que configurar el IDE con la información correcta.

Una vez esta todo conectado simplemente hacemos funcionar el programa de python y comprobamos que todo está en orden, en ZPUino también debemos tener cuidado aun con el nombre que le damos al archivo de python para que no genere problemas, una vez todo funcione como deseamos la práctica estará concluida sin ninguna modificación que hacer.

Raspberry pi

Para comenzar a trabajar con Raspberry pi lo primero será instalarle su propio sistema operativo ya que tendremos ante nosotros a un mini PC, que con una tarjeta SD será capaz de correr el sistema operativo que le pondremos (en este caso ubuntu MATE) y una vez hecho esto podremos desde un terminal decidir qué hará nuestra placa, comunicarnos desde otro ordenador mediante una conexión SH e incluso crear un servidor Web con cierta información, objetivos que serán todos realizados en las prácticas.

En primer lugar para instalar el sistema operativo tendremos que nuestra tarjeta SD en la que cargaremos el programa, seguiremos los pasos de los tutoriales dispuestos en la guía, con cuidado a la hora de poner en el terminal el nombre de la tarjeta "sdb" por ejemplo, ya que a veces hay que ponerle un 1 o 2 y otras no, según nos refiramos a una partición o toda la tarjeta, y será clave tenerlo bien escrito, una vez cargado el programa en la tarjeta podremos conectarla a la raspberry junto con pantalla, ratón, teclado, etc. y listo, nuestro equipo se instalará y solo nos quedará de nuevo hacer un par de configuraciones para que internet funcione correctamente.

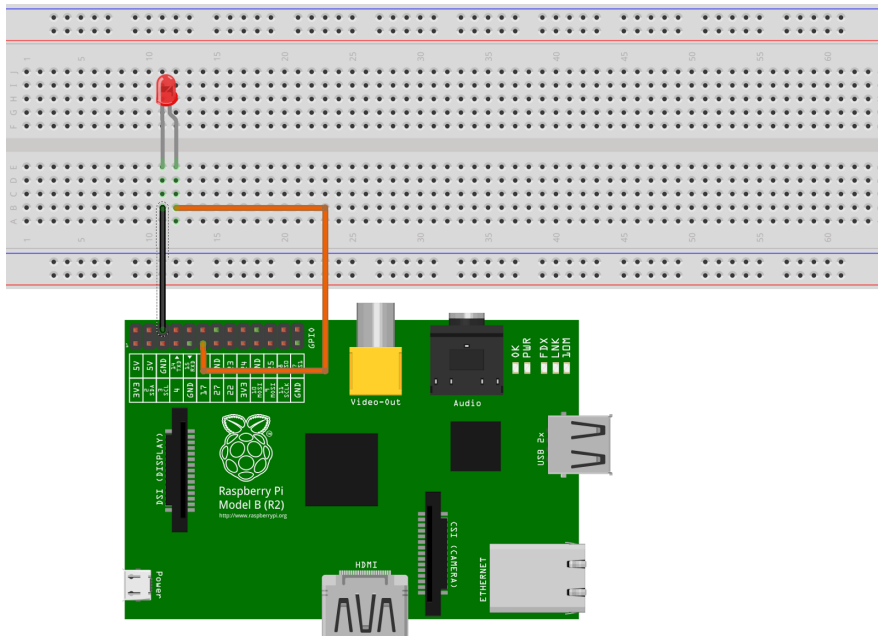
Practica 1

En esta primera práctica de la raspberry pi tendremos que hacer una tarea muy sencilla que en mi caso la hice junto con la tarea tres directamente, esto es, en lugar de usar el

terminal de la propia raspberry me conecte mediante ssh desde otro ordenador y desde hay accedemos al terminal de la raspberry y seguimos los pasos del tutorial.

Los pasos de este tutorial son realmente sencillos, tan solo un par de comandos escritos en el terminal para ver cómo funciona y listo.

Montamos el pequeño circuito, un led conectado a uno de los GPIO de la raspberry, y la idea será encenderlo y apagarlo a voluntad mediante el terminal.



Pues bien el circuito no tiene ningún misterio, y ahora para seguir basta con copiar y pegar los comandos escritos en el enlace que se nos da en la práctica, y digo copiar y pegar ya que de no ser así podríamos tener algún fallo minúsculo que hiciera que no funcionara.

En mi caso por ejemplo faltaba en el primer comando un espacio de separación

fritzing

entre el "17" y ">" (echó `17 > /sys/class/gpio/export`) con esto que puede parecer que no debe afectar en absoluto el mensaje que enviamos no funcionará en absoluto nada de lo que estamos haciendo en esta tarea, nos dará un error en la segunda o tercera línea que ejecutemos y sabremos que nos hemos equivocado en algo, incluso si miramos detenidamente puede que nos cueste cerciorarnos de fallo o lo consideremos insignificante como fue mi caso y pensemos que el problema debe ser de otra cosa, que algo no ha funcionado bien cuando es tan simple como un carácter que nos falta.

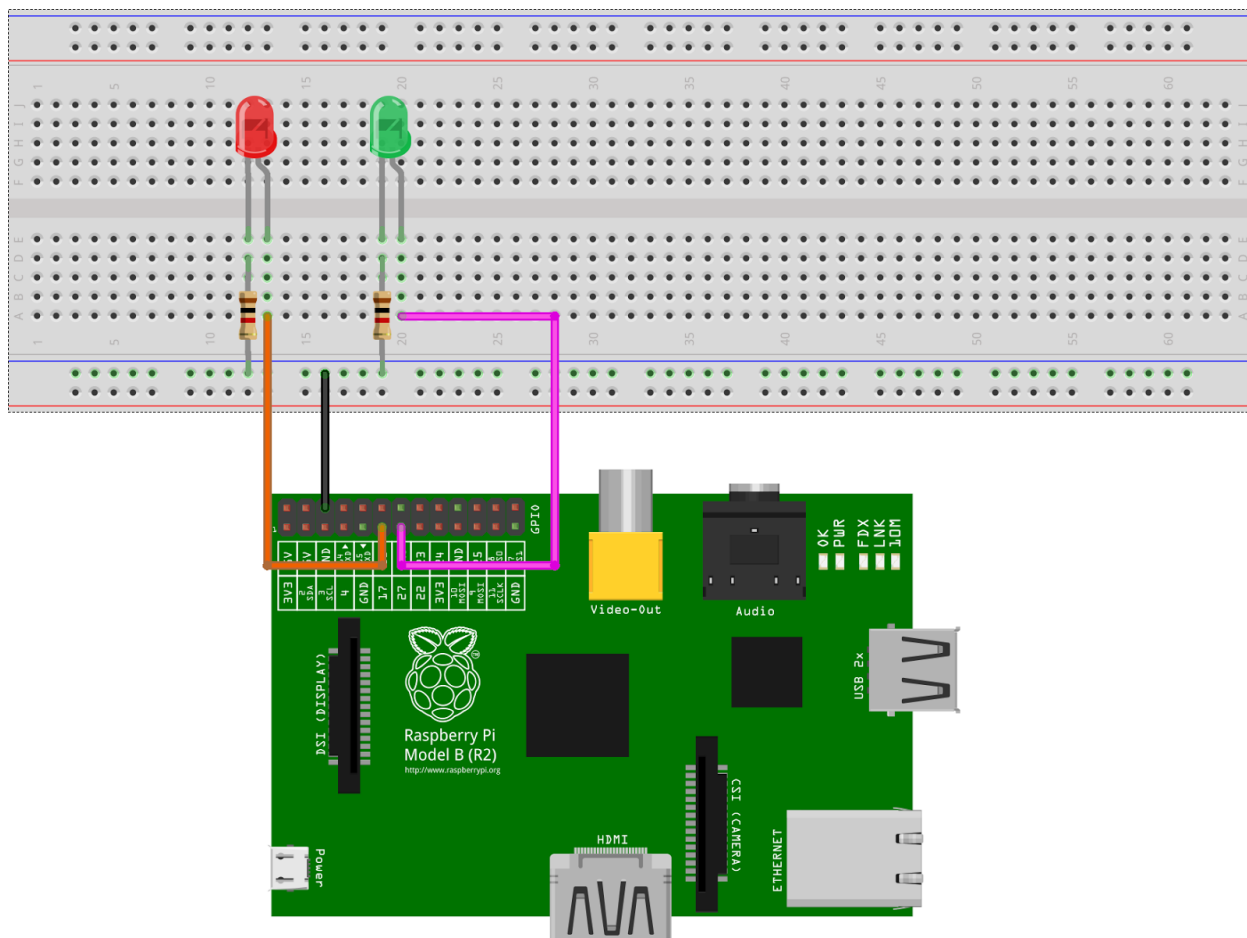
Una vez copiados y pegados todos los comandos correctamente el led debería encenderse y apagarse sin más complicaciones y podremos pasar a la segunda práctica.

Para hacer todo esto mediante ssh, cosa que tendríamos que empezar a hacer en la práctica 3 tan solo hay que irse a otro ordenador, desde este escribir en el terminal `ssh -X <usuario>@<IP address asp>`, y poner después la contraseña que dimos en la fase de configuración a nuestra raspberry después de esto como nos pedirán en el terminal, un problema muy frecuente que se dio en muchos casos es el de que los ordenadores tenían ya una conexión ssh guardada o otra raspberry y no eran por tanto capaces de

conectarse a la que estabamos intentando conectarnos ahora, para solucionar este fallo tendremos que borrar de la memoria de nuestro ordenador la información que concernia a esa conexión ssh con la otra raspberry y tras esto tan solo ya nos podremos conectar perfectamente a nuestro dispositivo tal y como se ha descrito anteriormente.

Practica 2

En esta segunda práctica con la raspberry pi empezaremos a sacarle algo más de jugo a la raspberry y usaremos otra de sus funcionalidades, ya que como buen pc en miniatura que es, es capaz de ejecutar un programa en python y con esto también podremos acceder al gpio y encender y apagar leds, de una forma más interesante que tener que hacerlo desde el terminal y que nos dara mucho más juego y posibilidades.



El circuito que montaremos será bastante parecido al anterior sólo cambiaremos que en lugar de tener un único led usaremos un par de ellos y en lugar encenderse a nuestra voluntad tendrán que estar parpadeando constantemente.


Lo primero es lo primero, antes de nada tendremos que descargar python para ser capaces de ejecutar este programa que vamos a crear tal y como nos explican en el tutorial, una vez completado esta parte pasamos a escribir el código y montar el circuito propuesto.

Para escribir el código yo recomendaría crear un archivo con "sudo pluma blink.py"

Que nos crearia el archivo blink.py y lo abriria con la herramienta pluma, que es algo así como gedit en ubuntu un simple y eficaz editor de textos con el que podremos escribir nuestro código, una vez hecho esto copiamos y pegamos el código del ejemplo.

```
def blink():  
    print "Ejecución iniciada..."  
    iteracion = 0  
    while iteracion < 30: ## Segundos que duraba la función  
        GPIO.output(17, True) ## Enciendo el 17  
        GPIO.output(27, False) ## Apago el 27  
        time.sleep(1) ## Esperamos 1 segundo  
        GPIO.output(17, False) ## Apago el 17  
        GPIO.output(27, True) ## Enciendo el 27  
        time.sleep(1) ## Esperamos 1 segundo  
        iteracion = iteracion + 2 ## Sumo 2 porque he hecho dos parpadeos  
    print "Ejecución finalizada"  
    GPIO.cleanup() ## Hago una limpieza de los GPIO  
    blink()
```

Analizamos qué es lo que hace, en este caso enciende y apaga cada segundo los leds, estando siempre uno encendido cuando el otro se apaga y viceversa, en python es importante a diferencia de otros lenguajes el indexado de cada línea ya que eso indicará el orden de ejecución del código, que está dentro de un if o un for por ejemplo y hay que tener mucho cuidado con esto o con el uso del tabulador y los espacios de forma diferente.



Ejecutamos la función como hemos hecho otras veces para comunicarnos con arduino por ejemplo `sudo blink.py`, y comprobamos que los leds se encienden y apagan uno al contrario que el otro cada segundo. Una vez todo listo podemos pasar directamente a la siguiente práctica, que sera la 4 ya que todo esto lo hemos hecho con ssh y ya hemos explicado cómo se realiza esa conexión.

Practica 4

En esta cuarta práctica en el bloque de raspberry pi se nos propone antes de empezarla seguir un tutorial en el que veremos como crear un servidor web usando la raspberry pi, con este tutorial crearemos una página web en el puerto 8080 en la que imprimiremos la frase "Hello World" en el documento .html con el que se genera la pagina.

Para hacer esto usaremos python para comunicarnos con un template html con lo que de una forma muy rápida podremos pasar la informacion que conseguimos mediante el programa de python y transmitirla al archivo html para verlo en nuestro servidor web.

Una vez completado este tutorial se nos pide ahora hacer otro algo más complejo en el que tendremos dos leds conectados a las raspberry y nuestro objetivo sera, mediante el servidor web en comunicación con un programa python que manejara la raspberry ser capaces de encender los leds y anunciar de estos eventos en pantalla.

De nuevo todo lo que hay que hacer es en principio copiar y pegar todo el documento y una vez hecho cambiar los nombres de los componentes "coffee maker" y "Lamp" por "Led1" y "Led2" respectivamente, con esto el programa estará listo.

Podremos ver al acceder a nuestro servidor que tenemos esta web que nos aparece a la izquierda, en la que si pulsamos sobre los enlaces turn on o turn off aparecerá un mensaje de aviso de que hemos apagado o encendido el led correspondiente y este se apagará o encenderá en la raspberry pi, para esto creamos una funciones en el programa de python que son las que se ejecutan



al pulsar en cualquiera de los enlaces con la información de el led correspondiente.

En nuestra web por supuesto debería leerse Led1 y Led2 en lugar de coffe maker y Lamp, una vez terminado este tutorial se nos pide hacer una pequeña variación para la siguiente práctica, ahora hemos visto cómo desde el servidor Web podemos cambiar los componentes de las raspberry pi, pero cómo podemos hacer lo inverso, es decir con la información que leamos o consigamos desde el programa en python o la raspbeery, cambiar la informacion que nos aparece en el servidor.

Practica 5


En esta quinta práctica del bloque de raspberry pi, nos conectaremos con la placa arduino a la raspberry y en esta conectaremos un sensor de temperatura TMP36GZ, la temperatura que consigamos con este sensor deberemos imprimirla en el servidor web y no solo eso sino crear un nuevo botón, actualizar, que llame a otra función para que la informacion del sensor de temperatura se actualiza cada vez que se pulse.

Empecemos entonces por lo más sencillo ya que lo hemos hecho varias veces implementar la comunicación entre la raspberry y el arduino, simplemente en el programa python deberemos hacer un `Serial.readln()`, y debe ser `readln()` ya que si no solo leerá el primer dígito de la temperatura, con esto conseguiremos la temperatura ya que en arduino, tras crear el sencillo programa para conseguir la temperatura a partir del sensor la mandaremos mediante un `serial.println()`, de momento bastante facil ya que programas para leer la temperatura del sensor que es lo único nuevo se pueden encontrar muchos por internet.

```
void loop ()
{
    int lectura = analogRead(Sensor);
    float voltage = 5.0 / 1024 * lectura ; // Atencion aqui
    float temp = voltage * 100 - 50 ;
    Serial.println(temp) ;
}
```

Este es un ejemplo sin ir más lejos de como podría ser nuestro programa en arduino.

Seguimos adelante y ahora tenemos que crear la función en el programa de python, fijándonos en cómo son las otras funciones esto no debería ser muy complicado ya que en esta función sólo queremos leer el dato (`serial.readln()`) actualizando el valor de la variable en la que guardamos esa informacion (`temp=serial.readline()`) tras esto enviamos la informacion al template, eso si junto con la informacion de los leds, ya que si no al pulsar el botón de la función la pantalla quedara solo con la informacion de la temperatura y el botón actualizar, pero no aparece nada sobre los leds, ni podremos volver a conseguir su informacion ni funciones a menos que recarguemos la pagina inicial.



Con esto ya hemos casi acabado solo nos faltan un par de cosas, hacer una lectura inicial de la temperatura, ya que al cargar la pagina tambien queremos que aparezca no solo al pulsar en el botón asociado a la función de forma que volvemos leer la informacion del arduino y al final la mandamos al template, y por último en el template tendremos que hacerle su propio hueco a la informacion de la temperatura, esto es bastante fácil, solo tenemos que fijarnos en cómo está hecho para los leds, en html usamos "`<p> linea de texto</p>`" para añadir una línea de texto con que los usamos y para coger la informacion del template solo hay que ponerlo entre doble corchete `{{}}`, una vez completado esto podemos ver nuestra web y comprobaremos que ahora cargamos también la temperatura y podemos ir recargando.