

*LABORATORIO DE
DESARROLLO DE HARDWARE
ARDUINO*



MEMORIAS

2016/2017

Rafael Arroyo
Alemán

Ingeniería Informática de
Computadores



Escuela Técnica Superior de
Ingeniería Informática

ÍNDICE

Contenido

Practicas	3
❖ Blink.....	3
➤ Introducción	3
➤ Descripción de la actividad.....	3
➤ Evaluación después de la actividad.....	5
➤ Incidencias y anécdotas.....	5
➤ Valoración personal.....	5
❖ Encendido y apagado de un led mediante comandos a través del puerto serie con lenguaje python	6
➤ Introducción	6
➤ Descripción de la actividad.....	6
➤ Evaluación después de la actividad.....	8
➤ Incidencias y anécdotas.....	8
➤ Valoración personal.....	8
❖ Encendido y apagado de una bombilla mediante un relé por puerto serial.....	9
➤ Introducción	9
➤ Descripción de la actividad.....	9
➤ Evaluación después de la actividad.....	10
➤ Incidencias y anécdotas.....	10
➤ Valoración personal.....	10
❖ Encendido Apagado de un led mediante pulsador por interrupciones	11
➤ Introducción	11
➤ Descripción de la actividad.....	11
➤ Evaluación después de la actividad.....	12
➤ Incidencias y anécdotas.....	12
➤ Valoración personal.....	12
❖ Control del Angulo de posición de un servo motor enviando el ángulo a través del puerto serie	13
➤ Introducción	13
➤ Descripción de la actividad.....	13
➤ Evaluación después de la actividad.....	16
➤ Incidencias y anécdotas.....	16

➤ Valoración personal.....	16
❖ Control de la velocidad de giro de un motor de continua en doble sentido con un potenciómetro “Puente H”	17
➤ Introducción	17
➤ Descripción de la actividad.....	17
➤ Evaluación después de la actividad.....	19
➤ Incidencias y anécdotas.....	19
➤ Valoración personal.....	19
❖ Impresión en una pantalla LCD con caracteres enviados por puerto serial	20
➤ Introducción	20
➤ Descripción de la actividad.....	20
➤ Evaluación después de la actividad.....	24
➤ Incidencias y anécdotas.....	24
➤ Valoración personal.....	24
❖ Termómetro en el Display LCD.....	25
➤ Introducción	25
➤ Descripción de la actividad.....	25
➤ Evaluación después de la actividad.....	27
➤ Incidencias y anécdotas.....	27
➤ Valoración personal.....	27
❖ Contador de 00 a 59 en Display 7-segmentos cada segundo	28
➤ Introducción	28
➤ Descripción de la actividad.....	28
➤ Evaluación después de la actividad.....	31
➤ Incidencias y anécdotas.....	31
➤ Valoración personal.....	31

Practicas

❖ Blink.

➤ Introducción

En esta primera practica se hace una toma de contacto con la plataforma de desarrollo de arduino.

Esta plataforma se trata de un microcontrolador, basados inicialmente en los AVR de 8,128,328,1280 bits, aunque con alguna versión basada en ARM de 32 bits, un pequeño sistema de procesamiento con condición de sistema libre.

En primera instancia tenemos una interfaz de entrada, que puede estar directamente unida a los periféricos, o conectarse a ella por puertos cuyo objetivo es llevar de esa interfaz de entrada la información al microcontrolador que puede variar dependiendo de las características del proyecto al que se desee implementar.

Y por último la interfaz de salida que lleva la información procesada a los periféricos encargados de hacer uso final de esos datos, la cual puede ser otra placa, una pantalla..., lo que podemos decir que Arduino es sinónimo de hardware libre y con el ello podemos hallar toda la complejidad que se desee.

➤ Descripción de la actividad

En primer lugar, antes de poner en práctica el uso de nuestra plataforma Arduino hemos tenido que descargar nuestra plataforma de desarrollo de la página oficial y tras esto vamos a desarrollar algunos sketches de toma de contacto.

Como primera actividad se ha realizado el programa **Blink** sobre un led. En esta práctica hemos realizado una primera toma de contacto con Arduino el cual consiste en apagar y encender un led con un intervalo de tiempo de apagado y encendido de 1 segundo.

Se ha realizado un circuito conectando el pin 8 como salida de 5 voltios y a través de una resistencia de 220ohmios con $\pm 5\%$ la cual se ha calculado mediante la fórmula:

Resistencia = V / I para hallar la resistencia la cual nos da 213.33oh

Potencia = $V * I = I^2 * R = V^2 / R = 0.048W$

Código de blink:

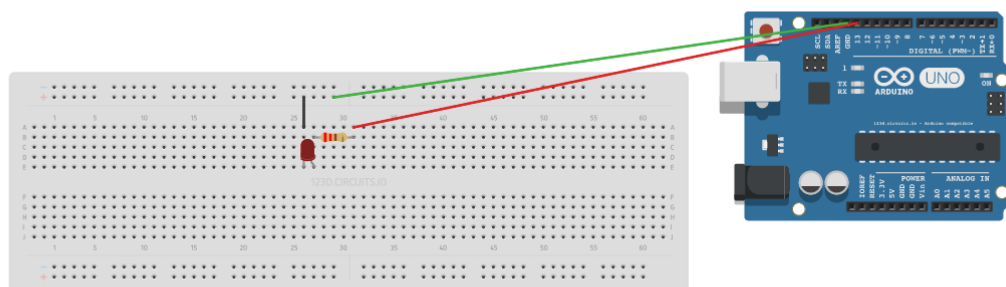
```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(8, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(8, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);            // wait for a second
  digitalWrite(8, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);            // wait for a second
}
```

Aquí una demostración del resultado:

https://1drv.ms/v/s!Ao2_30mhw3bigXdTjIbg1VznrQKd

Esquema del circuito realizado con **Autodesk Circuits** para su simulación:



➤ Evaluación después de la actividad

Tras la realización de la práctica, se ha aprendido a instalar el entorno de desarrollo de arduino , como es una estructura de un programa con los métodos void setup() y void loop() así como el encendido o apagado de un led y la realización de escritura sobre los pines digitales y los retrasos como una primera toma de contacto con la plataforma de desarrollo arduino.

➤ Incidencias y anécdotas

- Ejecución de IDE arduino sobre Linux, descarga extracción del paquete instalador, así como ejecución del programa sobre terminal.
- Bloqueo de terminal al estar ligada a la ejecución del IDE arduino.

➤ Valoración personal

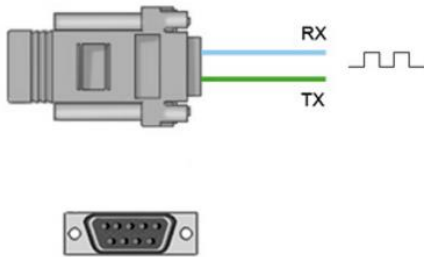
Practica la cual me ha servido mucho como primera toma de contacto tanto como para el conocimiento de la estructura de un programa en arduino así como un primer manejo de ejecución de programas sobre un terminal.

❖ Encendido y apagado de un led mediante comandos a través del puerto serie con lenguaje python

➤ Introducción

Aquí vamos a hacer la ejecución de un programa el cual se encargue de controlar el encendido o apagado de un led desde el pc vía puerto serie, el cual recibirá comandos a través de un programa en Python.

COMUNICACIÓN SERIE



Un puerto serie envía la información mediante una secuencia de bits. Para ello se necesitan al menos dos conectores para realizar la comunicación de datos, RX (recepción) y TX (transmisión). No obstante, pueden existir otros conductores para referencia de tensión, sincronismo de reloj, etc.

➤ Descripción de la actividad

En esta prueba sobre el led se ha basado en el mismo circuito que el anterior pero con la diferencia que en lugar de realizar intervalos de tiempo de 1 segundo en encendido y apagado del led éste se apaga y enciende utilizando el puerto serie, donde en el código se inicia la comunicación y se estarán leyendo constantemente los valores que no estén llegando desde Python, en el cual la primera prueba realizada es a través del monitor serie si mando la letra H se enciende el led y si mando la letra L se apaga el led.

Ahora se hace la misma prueba pero a través de una ventana de símbolo del sistema, en este proceso se ha utilizado el Notepad++ para la ejecución del código Python en el cual pues se a preparado un programa en el que primeramente se importa la librería serial para poder usar el puerto serial y se declara una variable llamada Arduino con os datos necesarios para que se pueda realizar la comunicación , datos con el puerto y los baudios, tras eso lo que se hace es pedir un comando la cual es H o L dependiendo de si queremos encender o apagar el led.

Código Arduino:

```
int led = 8;
void setup () {
    pinMode(led, OUTPUT); //LED 8 como salida
    Serial.begin(9600); //Inicializo el puerto serial a 9600 baudios
}

void loop () {
    if (Serial.available()) { //Si está disponible
        char c = Serial.read(); //Guardamos la lectura en una variable char
        if (c == 'H') { //Si es una 'H', enciendo el LED
            digitalWrite(led, HIGH);
        } else if (c == 'L') { //Si es una 'L', apago el LED
            digitalWrite(led, LOW);
        }
    }
}
```

Código Python para el envío de comandos al Arduino:

```
import serial

arduino = serial.Serial('COM3', 9600)

print("Starting!")

while True:
    comando = raw_input('Introduce un comando: ') #Input
    arduino.write(comando) #Mandar un comando hacia Arduino
    if comando == 'H':
        print('LED ENCENDIDO')
    elif comando == 'L':
        print('LED APAGADO')

arduino.close() #Finalizamos la comunicacion
```

Aquí un ejemplo del resultado:

https://1drv.ms/v/s!Ao2_30mhw3biqgcHHpnry7jeaok :

➤ Evaluación después de la actividad

Se ha aprendido a usar el puerto Serial tanto en el programa escrito para arduino como el uso de librería Serial de Python para el mando de comandos por la terminal para el control de encendido y apagado del led

➤ Incidencias y anécdotas

- Ventana de monitor serie puesta a 11500 baudios y programa a 9600 baudios, no se recibían los comandos adecuadamente.
- Incorporación de librería serial a programa Python
- Puerto de arduino mal configurado
- Uso de pines 0 y 1 no se pueden usar mientras se usa el puerto serie

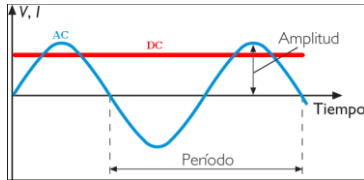
➤ Valoración personal

Practica que me ha servido para conocer el envío de comandos a través del puerto serie utilizando el lenguaje Python desde un pc a una placa arduino así como para conocer el funcionamiento de dicho puerto.

❖ Encendido y apagado de una bombilla mediante un relé por puerto serial.

➤ Introducción

Aquí se desarrolló el mismo ejemplo que el encendido y apagado de un led, pero en lugar de éste con la activación de un relé que esté conectado a una bombilla para así poder encenderla y apagarla.



Corriente para bombilla

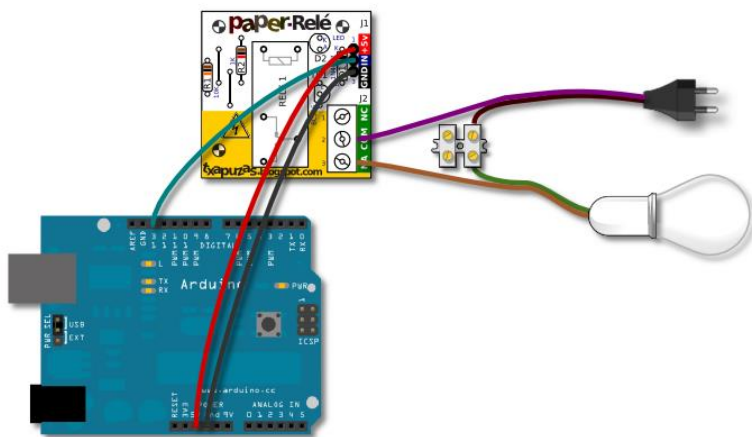
Este tipo de dispositivo “Bombilla” es accionada con corriente alterna pero nosotros no podemos controlar este tipo de corriente de manera directa con nuestro microcontrolador arduino de manera que necesitamos un elemento intermedio el cual manejará corrientes de alto amperaje con activación o desactivación de éste con voltajes pequeños.

➤ Descripción de la actividad

En esta prueba se ha realizado el mismo tipo de circuito elaborado en el ejemplo de encendido y apagado de un led por el puerto serial, pero con una diferencia, y es que, en lugar de ser aplicado sobre un led, lo hemos usado sobre un relé, cuyo funcionamiento es el mismo ‘Activar o desactivar’, por ello lo hemos reutilizado de manera que se le da una alimentación al relé y es la placa arduino quien se encarga de realizar la conmutación de ésta.

Relés utilizados en practicas





Esquema de conexión realizado de una bombilla al relé



Funcionamiento de bombilla conectada a relé

➤ Evaluación después de la actividad

Se ha aprendido a controlar un medio que funciona con un tipo de corriente no continua (Alterna) por medio de un relé el cual tiene la función de conmutar a nuestra orden, la cual es mandada a través de nuestro pc a arduino por medio del puerto serial.

➤ Incidencias y anécdotas

- Colocación incorrecta de la bombilla al relé haciendo una mala conmutación
- Alimentación incorrecta del relé al interruptor

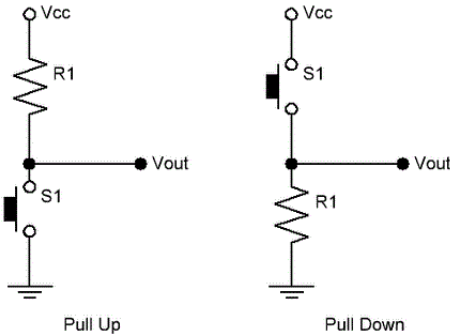
➤ Valoración personal

Observación de aprendizaje para darnos cuenta que se pueden controlar fuentes de corriente muy intensas a través de pequeñas placas de control por medio de comandos a través de nuestro pc por ejemplo el puerto serial.

❖ Encendido Apagado de un led mediante pulsador por interrupciones

➤ Introducción

En este ejemplo se realizará el encendido o apagado de un led que esté conectado a un circuito con una configuración pull-up:



En el cual se encuentra un pulsador que active o desactive una interrupción que encenderá o apagará el led según se pulse.

Este sería el esquema del pin de interrupción del ejercicio expuesto.

➤ Descripción de la actividad

En esta variante de apagado y encendido de un led se va a usar por medio de interrupciones.

En la placa arduino el pin que va a realizar la interrupción es el 2, se activara en el código con el método `attachInterrupt()`.

Aquí el código utilizado para el uso de esta práctica:

```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

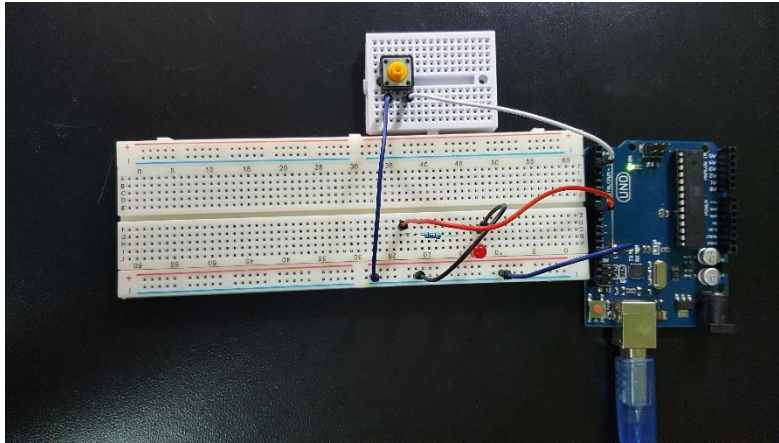
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}

void loop() {
  digitalWrite(ledPin, state);
}

void blink() {
  state = !state;
}
```

Se puede observar cómo hacemos uso del método descrito para las interrupciones, en el cual se ha utilizado ese formato indicando que cuando se activa la interrupción del pin 2 salte al método `blink` y cambie el valor de `state` de bajo a alto.

El segundo dato es utilizado para indicar el método que se debe saltar y el tercer dato para indicar que tipo de cambio se debe hacer el cual en este caso se cambia un valor bajo a alto el cual es un estado que no es guardado en memoria, caso de funcionamiento ram.



Circuito realizado

Aquí un ejemplo del ensayo sobre el circuito:

https://1drv.ms/v/s!Ao2_30mhw3biqibbLKIHFzj8f96N

➤ Evaluación después de la actividad

- A sido una práctica en la que hemos realizado pruebas sobre interrupciones digitales para aplicar sobre el encendido o apagado de un led ante un evento externo, en este caso sobre la aplicación de un uno o un cero sobre la entrada digital 2 de la placa arduino.

➤ Incidencias y anécdotas

- A ser el interruptor un dispositivo mecánico el microcontrolador registra el rebote del interruptor, lo que muchas veces nos lleva a un resultado indeseado, que como en este caso ha sido la..., a veces, mal activación o desactivación del led.

➤ Valoración personal

He realizado mi primera interrupción dándome cuenta de la importancia del registro de interrupciones en el sistema, y de cómo se debe tener cuidado con los dispositivos mecánicos o simplemente como a veces los dispositivos no pueden funcionar como uno desea sin haberlo podido predecir.

❖ Control del Angulo de posición de un servo motor enviando el ángulo a través del puerto serie

➤ Introducción

Se va a realizar un programa el cual se usará para para controlar el Angulo que debe girar el servo, esto se hará enviando los datos por medio del puerto serie con un programa escrito en Python.



Servo utilizado en la practica

➤ Descripción de la actividad

Hemos puesto en práctica conceptos anteriormente usados como uso del puerto serie y métodos nuevos como el uso del `parseInt()` de la librería `Serial` como método de transformación de letra a entero en nuestro programa arduino en la recepción de comandos por el puerto serial.

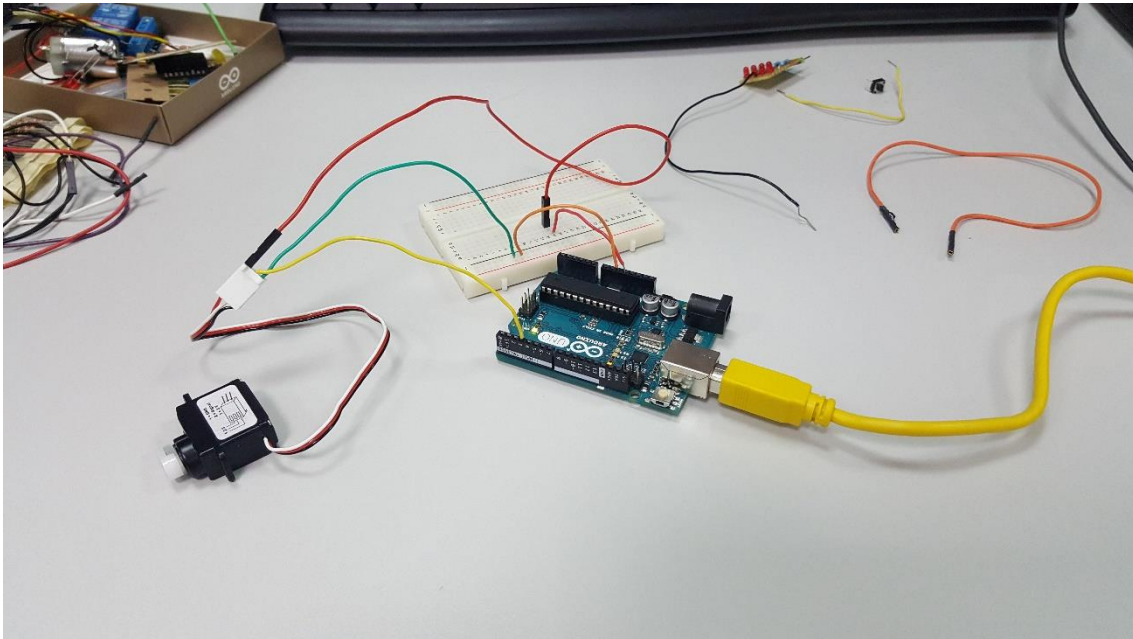
El principio de este motor es que funciona con corriente continua, para controlarlo se le envía pulsos cada 20ms es decir unos 50Hz donde la anchura del pulso es lo que codifica el ángulo de giro, es decir usando el PWM, codificación por ancho de pulso, esta anchura varía dependiendo del servo, pero suele variar entre los 0.5 y 2.5 ms.



Se puede observar el motor, la circuitería de control, un juego de piñones, y la caja. También se pueden ver los **3 cables de conexión externa**:

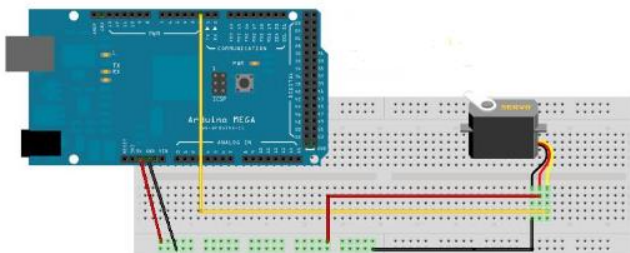
- uno (rojo) es para **alimentación**, Vcc (~ +5volts);
- otro (negro) para conexión a **tierra** (GND);
- el último (blanco o amarillo) es la línea de **control** por la que se le envía la señal codificada para comunicar el ángulo en el que se debe posicionar.

Aquí tenemos el ejemplo de conexión realizado:



- Se puede ver como se conecta el positivo(rojo) ,negativo(verde) y amarillo para mandar los pulsos a través de la arduino.

Y el modelo realizado en **Autodesk Circuits**:



En el código se ha hecho uso de la librería Servo.h el cual se encuentra ya en el IDE de arduino, en este ejemplo se usa el pin 3 el cual nos da una salida de pwm (señal de ancho por pulsos). Este estará a la espera de recibir un dato por parte de nuestro programa escrito en python por el puerto serial de manera que dependiendo el dato recibido se procese un ángulo de giro que será dado por nosotros que venga desde 0º a 180º .

```
#include <Servo.h>

Servo servo1; // Crea un Objeto servo
int posicion; // Variable de la posicion del servo

void setup()
{
  servo1.attach(3); // Seleccionamos el pin 2 como el pin de control para el servo
  Serial.begin(9600);
}

void loop()
{
  if(Serial.available()){
    posicion = Serial.parseInt();

    servo1.write(posicion); // Escribimos la posicion con el mapa de valores al servo
  }
}
```

El código **Python** que mandara datos por el puerto serie:

```
import serial

arduino = serial.Serial('/dev/ttyACM0', 9600)

print("Starting!")

while True:
    comando = raw_input('Introduce un comando: ') #Input
    arduino.write(comando) #Mandar un comando hacia Arduino
    |

arduino.close() #Finalizamos la comunicacion
```

Aquí un ejemplo de funcionamiento de la implementación:

https://1drv.ms/v/s!Ao2_30mhw3biqhlxoEOk0T251A6v

➤ Evaluación después de la actividad

Toma de contacto con un motor servo en el que se usó el pin de salida PWM que emula una señal analógica por medio del control de anchura de pulso.

Envío de comandos a través del puerto serie y su previa transformación de ángulo de giro para mandárselo al servo.

➤ Incidencias y anécdotas

- Tiempo de actuación del servo lento debido al procesamiento del `parseInt()` frente a una posible mejora con el tratamiento letra a letra.

➤ Valoración personal

Uso de Servos como control del movimiento del ángulo de giro para uso en robótica donde el uso de movimiento controlado es necesario, también he aprendido el uso del pwm de como controlar el ancho de pulso.

Aunque no se haya llegado a usar también he aprendido a usar el mapeo de datos con el método `map()`.

❖ Control de la velocidad de giro de un motor de continua en doble sentido con un potenciómetro “Puente H”

➤ Introducción

En esta práctica vamos a hacer uso del puente H el cual puede cambiar la polaridad en señales de salida o entrada, el cual en este caso se va a usar el L293DNE.

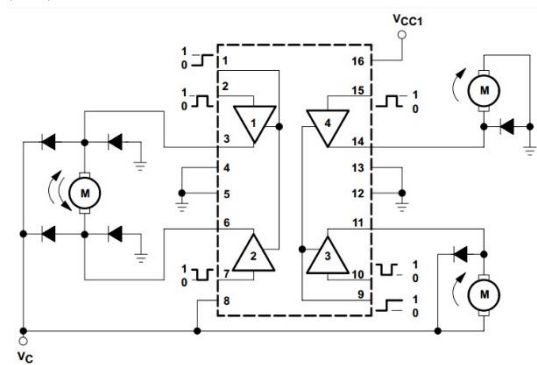


Se aprovechará la capacidad del puente H para usarlo en un motor de corriente continua el cual dependiendo de la polaridad recibida girará en un sentido u en otro.

➤ Descripción de la actividad

En este ejemplo se ha controlado la velocidad y giro del motor por medio de un potenciómetro el cual a través del puente H que ha sido controlado por un programa que según los datos recogidos por una entrada analógica y con sus datos mapeados en el que según sus valores el motor girará en un sentido u en otro.

En el siguiente esquema se ve el datasheet del puente H:



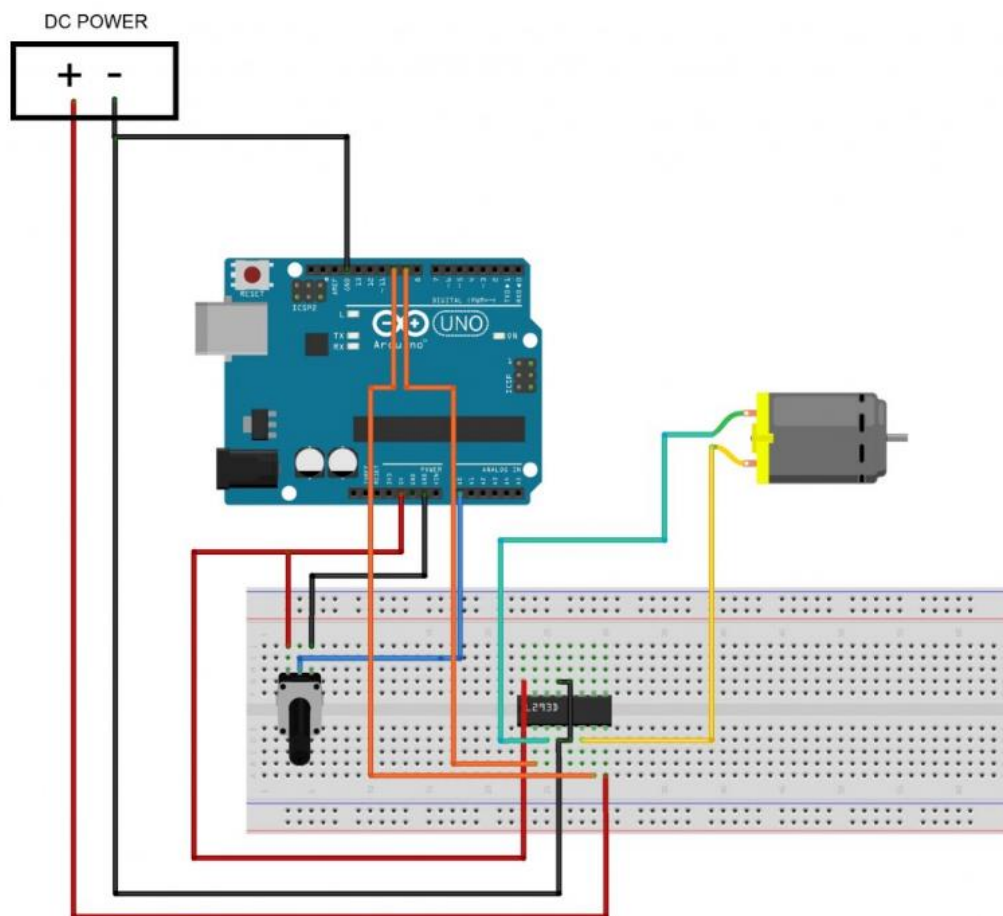
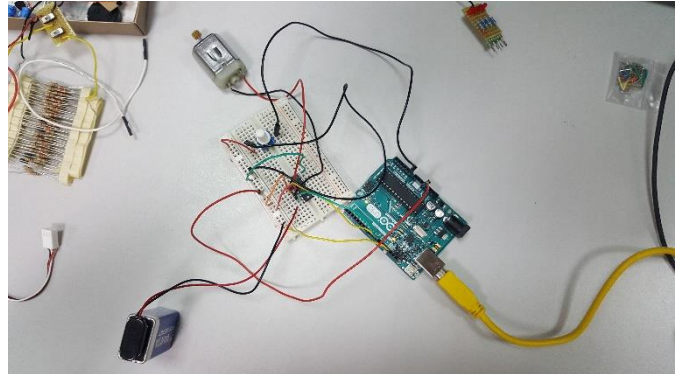
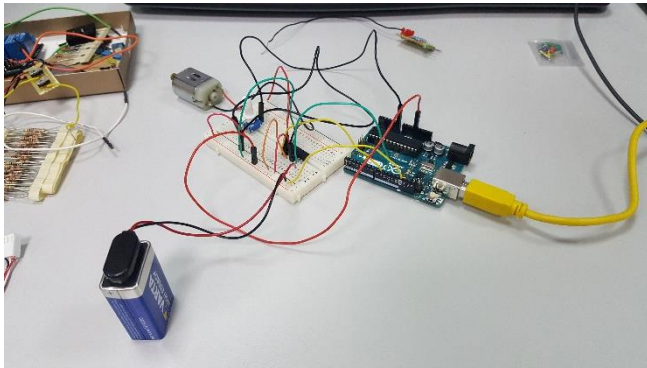
Nosotros hemos usado la parte de la izquierda, los pins 3 y 6 son las salidas y se conectan a los ornes del motor, así como los pins 2 y 7 son las entradas donde conectaremos las salidas del arduino.

Dependiendo del valor que ponemos entre los pins 2 y 7 el motor realizara el giro en un sentido u otro.

Como hemos dicho la velocidad de giro será controlado por un potenciómetro para ello se hace uso de la señal PWM.

Como hay que mandar la señal a los pins 2 y 7 las cuales invierten el sentido de giro lo hacemos mediante la señal generada por arduino de manera que:
Si la señal PWM1 está en alta el PWM2 estará en baja y viceversa.

Esquema del circuito:



Aquí tenemos un ejemplo del código usado para la implementación del circuito:

```
int pin2=9;    //Entrada 2 del L293D
int pin7=10;   //Entrada 7 del L293D
int pote=A0;   //Potenciómetro

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

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

void loop()
{
  //Almacenamos el valor del potenciómetro en la variable
  valorpote=analogRead(pote);

  //Como la entrada analógica del Arduino es de 10 bits, el rango va de 0 a 1023.
  //En cambio, la salidas del Arduino son de 8 bits, quiere decir, rango entre 0 a 255.
  //Por esta razón tenemos que mapear el número de un rango a otro usando este código.
  pwm1 = map(valorpote, 0, 1023, 0, 255);
  pwm2 = map(valorpote, 0, 1023, 255, 0); //El PWM 2 esta invertido respecto al PWM 1

  //Sacamos el PWM de las dos salidas usando analogWrite(pin,valor)
  analogWrite(pin2,pwm1);
  analogWrite(pin7,pwm2);
}
```

Y un ejemplo de funcionamiento:

https://1drv.ms/v/s!Ao2_30mhw3bighpoDfvVN9XcXP-5

➤ Evaluación después de la actividad

Una práctica intuitiva y a la vez compleja en el entendimiento del uso del puente H, así como la captura de las señales analógicas para su mapeo e inversión de las señales para poner en funcionamiento del motor continuo en un sentido u otro.

➤ Incidencias y anécdotas

- Mal conexionado del Puente H con olvido de alimentación en los dos bornes a 5V

➤ Valoración personal

He aprendido a usar un inversor de polaridad de señales, así como su aplicación en motores de corriente continua.

Relacionar conceptos de captura de señales analógicas inversas para su uso en el L293DNE

Buena realización de la practica

❖ Impresión en una pantalla LCD con caracteres enviados por puerto serial

➤ Introducción

En este programa se va a realizar el uso de una pantalla lcm1602C



Se va a realizar una impresión en dos filas con desplazamiento a la izquierda:

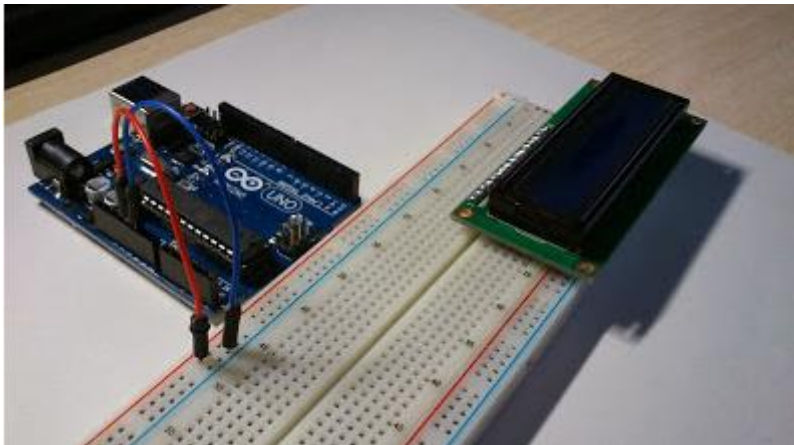
En la primera fila el nombre de la asignatura y en la segunda fila el nombre del alumno, una vez se haya conseguido esto se va a imprimir en la pantalla, pero esta vez con la

información que nosotros hayamos mandado desde el pc vía serie.

➤ Descripción de la actividad

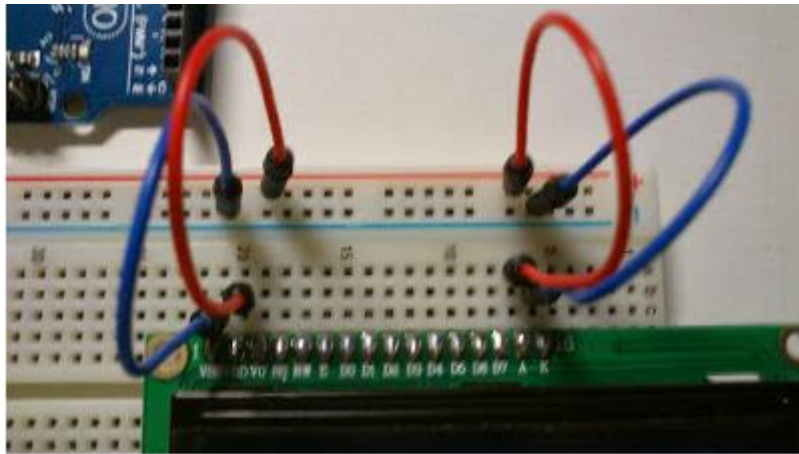
Se ha realizado un circuito en el cual nosotros podemos controlar tanto el contraste del lcd como la información que se muestra en la pantalla.

Lo primero que se ha hecho es conectar la pantalla a nuestra protoboard y alimentar las líneas de positivo +5V y tierra

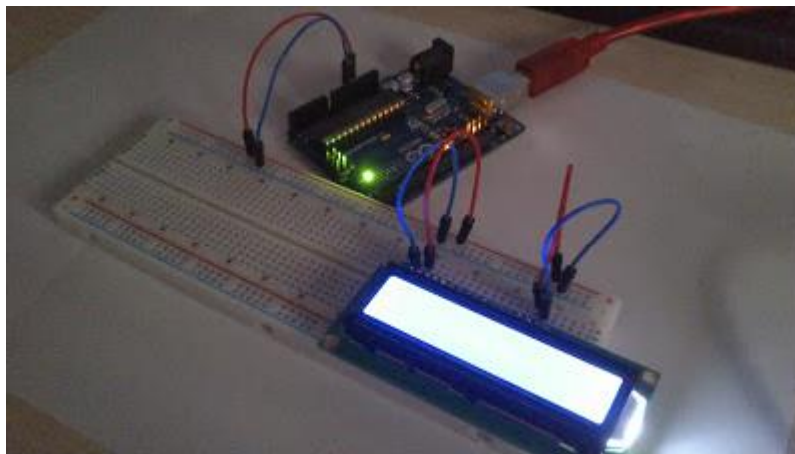


Conexiones:

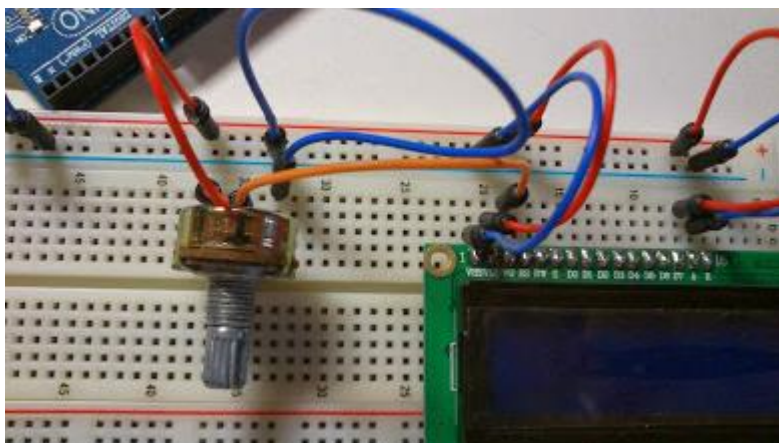
- Pin 1 y Pin 16 a GND
- Pin 2 y pin 15 a 5v



Una vez hecho este paso podemos comprobar que la pantalla se enciende, esto lo hacemos conectando nuestra arduino al usb del pc para darle corriente.



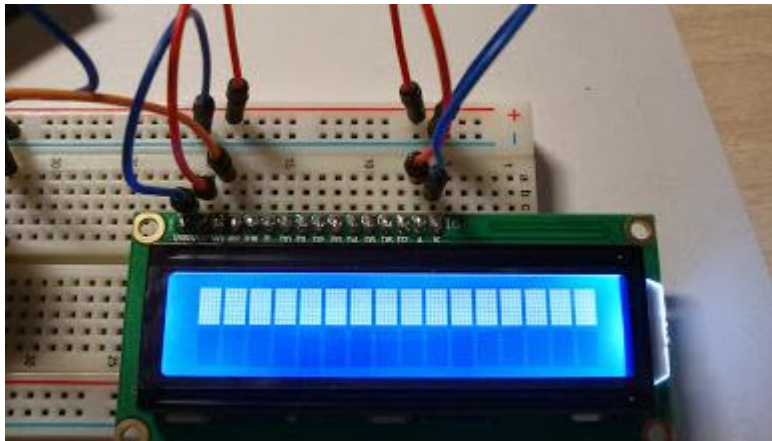
Ya una vez hecho esto procederemos a controlar el contraste de la pantalla, la cual se hará con un potenciómetro.



El esquema a seguir será colocarlo en cualquier parte libre de la protoboard y alimentar su borne izquierdo a +5v y su borne derecho a GND.

Por último nos quedará el borne central del que se obtendrá la diferencia de tensión según se aplique.

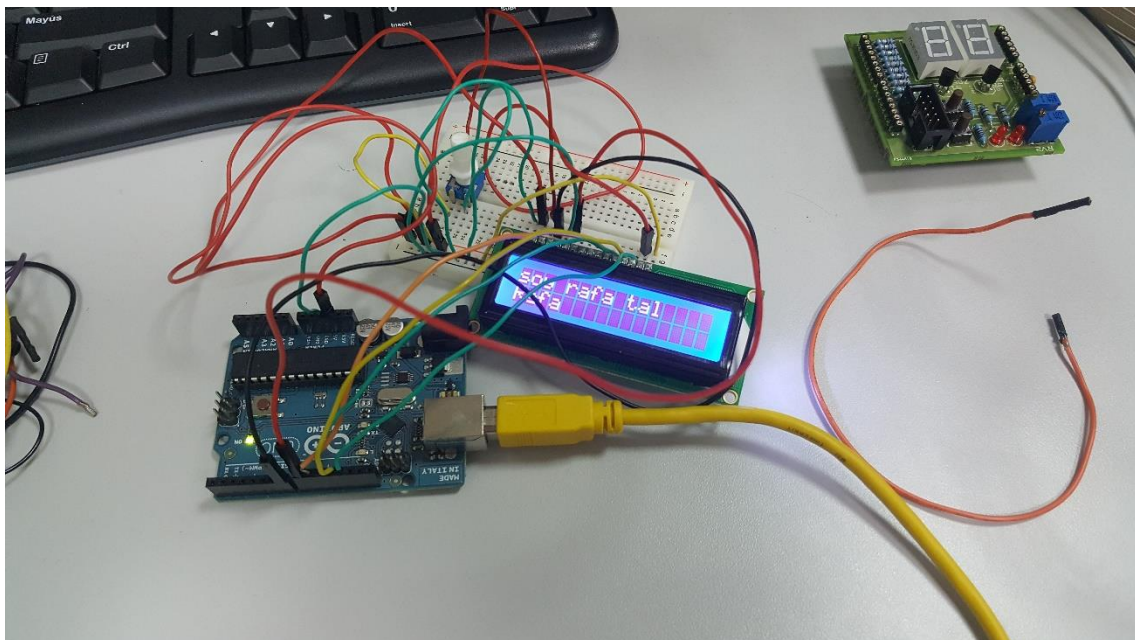
Una vez que hayamos hecho esto podemos darle diferentes tensiones hasta que se logre diferenciar los cuadraditos donde irán los caracteres.



Este sería el resultado de aplicar el potenciómetro.

Ya por ultimo vamos a conectar los pines a arduino para que podamos enviar información a la pantalla.

- Pin 4 del LCD "RS" → pin 7 de arduino (digital y PWM)
- Pin 5 del LCD "RW" → GND
- Pin 6 de la LCD "E" → pin 8 de arduino (PWM)
- Pin 11 del LCD "D4" → pin 9 de arduino (PWM)
- Pin 12 de LCD "D5" → pin 10 de arduino (PWM)
- Pin 13 de LCD "D6" → pin 11 de arduino (PWM)
- Pin 14 de LCD "D7" → pin 12 de arduino (PWM)



Aquí el resultado del cableado

Por último el código de Arduino en el cual se hace uso de `Serial.readString()` de la librería `Serial`.

```
#include <LiquidCrystal.h>
String string="";
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.setCursor(0,0);
  lcd.write("LDH");
  lcd.setCursor(0,1);
  lcd.write("Rafael Arroyo Aleman ");
}

void loop() {
  Serial.available() {
    lcd.setCursor(0,0);
    string = Serial.readString();
    lcd.write(string);
  }
}
```

Y el uso el código Python para el envío del texto vía `Serial`:

```
import serial

arduino = serial.Serial('/dev/ttyACM0', 9600)

print("Starting!")

while True:
    comando = raw_input('Introduce una frase: ') #Input
    arduino.write(comando) #Mandar un comando hacia Arduino
    print('Comando mandado')

arduino.close() #Finalizamos la comunicacion
```

Por ultimo un ejemplo del circuito montado y funcionando:

https://1drv.ms/v/s!Ao2_30mhw3birnmExaGvwGRcbLSG

➤ Evaluación después de la actividad

Se ha desarrollado un ejemplo de conexión de un lcd con la placa arduino, en esta actividad se ha aprendido a utilizar la librería liquidCristal de arduino, a realizar conexiones un poco más complejo, como utilizar el potenciómetro para usar una señal analógica para el control del contraste del lcd y el uso del puerto Serial para el envío de texto con el uso de `Serial.readString()`.

➤ Incidencias y anécdotas

- Mal conexionado del potenciómetro al lcd en pin incorrecto
- Mal desplazamiento del texto introducido en la visualización del lcd
- Mal muestreo del texto en el lcd
- No se estaba controlando la línea "Arriba o abajo" de escritura del texto

➤ Valoración personal

Práctica muy didáctica y estimuladora tanto a nivel visual como a nivel de conceptos aplicables al circuito en el uso de señales analógicas para control de contraste y uso del pc para envío de texto a arduino para su muestra en el lcd.

❖ Termómetro en el Display LCD

➤ Introducción

En este ejemplo se va a reutilizar el conexionado anteriormente usado y el código, esto es, la base de esta práctica con respecto al LCD es la misma, pero con la diferencia que en lugar de mandar texto vía serial vamos a hacer ese envío mediante la toma de datos por parte del sensor de temperatura que irá tomando datos y actualizando la información en tiempo real en los datos mostrados.



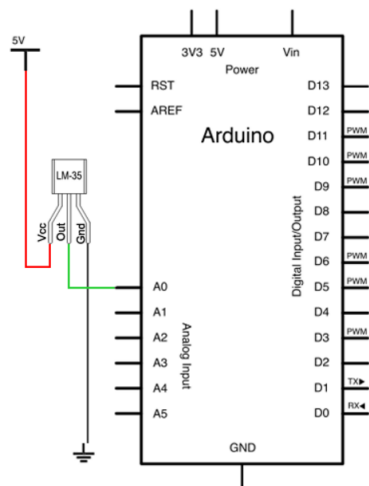
Sensor de temperatura LM35 usado en la práctica

➤ Descripción de la actividad

El sensor LM35:

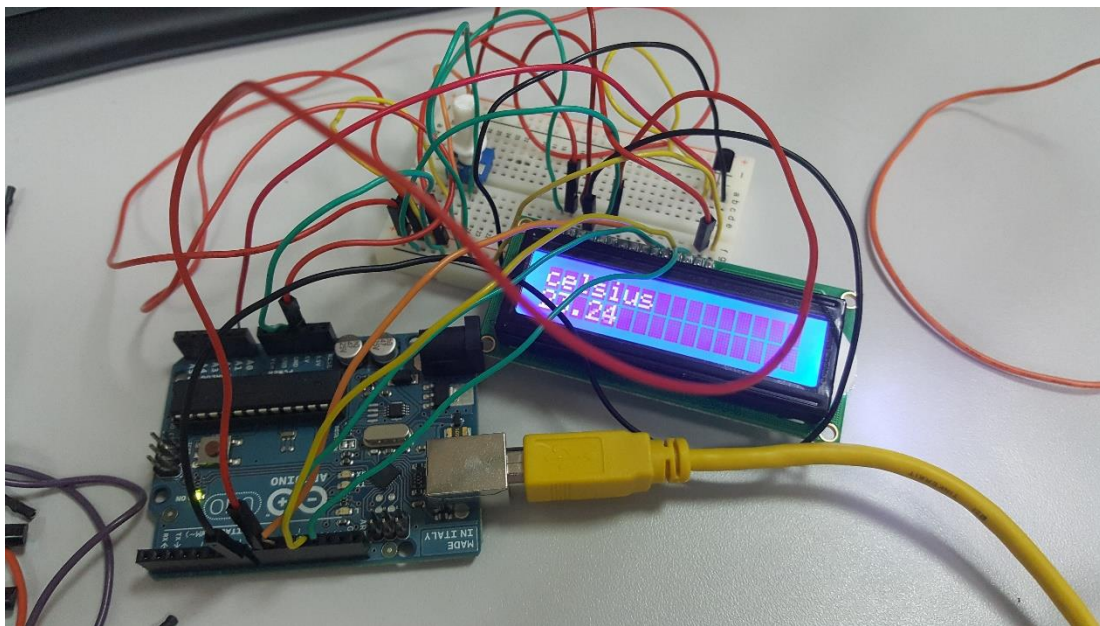


Los pines externos son para la alimentación mientras que el pin central proporciona la medición en una referencia de tensión, a razón de 10 mV/°C



Esquema eléctrico añadido al circuito del LCD con entrada de datos analógica en el pin A0 de arduino.

Resultado del circuito ya montado:



Código:

```
#include <LiquidCrystal.h>
float temperatura;
int analog_pin = 0;
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.setCursor(0,0);
  lcd.write("LDH");
  lcd.setCursor(0,1);
  lcd.write("Rafael Arroyo Aleman ");
}

void loop() {

  temperatura = analogRead(analog_pin);
  temperatura = 5.0*temperatura*100.0/1024.0;
  lcd.setCursor(0,0);
  lcd.write(Celsius);
  lcd.setCursor(0,1);
  lcd.write(temperatura);
}
```

El código lee el valor de tensión mediante la entrada analógica, y traducimos el valor tomado a grados centígrados usando la relación de 10mV/°C.

Una vez se haya tomado el valor se manda el contenido para que se muestre en la fila de abajo del LCD.

Ejemplo de uso:

https://1drv.ms/v/s!Ao2_30mhw3birnScA_6-fop8tOWf

➤ Evaluación después de la actividad

Tras la realización de la práctica se pudo comprobar el correcto funcionamiento de la lectura de datos a través de la medición del sensor de temperatura el cual cuando se acercaba a una fuente de calor se iban modificando los valores de la resistencia interna de este conllevando esto a una diferencia de tensión proporcional, estos datos se recopilan en la arduino por la entrada analógica con 1024 valores de resolución y tras esto se realizaban las operaciones oportunas para adaptarlo a la información que queríamos representar , en este caso , mostrar los grados en centígrados.

➤ Incidencias y anécdotas

- Al hacer el tratamiento de los datos recopilados por el sensor de temperatura no se mostraban los datos en centígrados debido a que se estaba adaptando de forma indeseada.

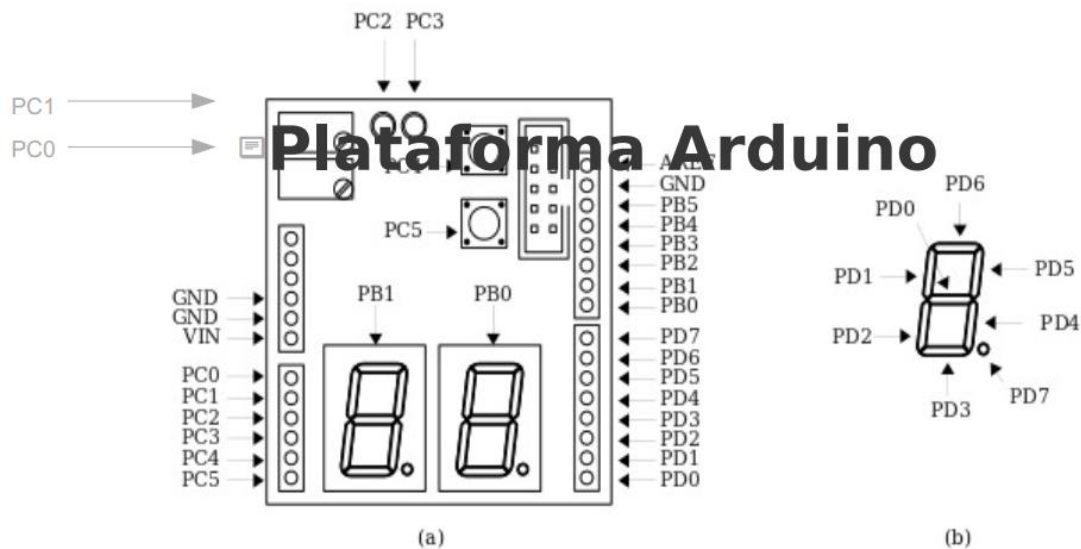
➤ Valoración personal

A sido una práctica instructiva desde el punto de vista de toma de datos en tiempo real con visualización.

❖ Contador de 00 a 59 en Display 7-segmentos cada segundo

➤ Introducción

En esta práctica se va a realizar un contador de 00 a 59 segundos en la cual se utilizarán dos displays 7-segmentos para ello que ya se encuentran implementados sobre una Shell (Placa de expansión).

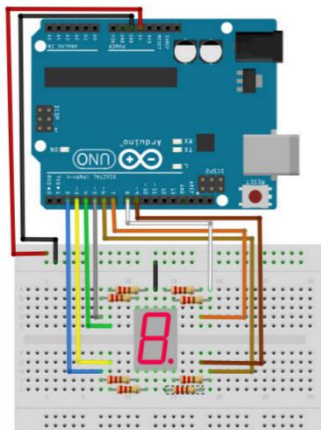


➤ Descripción de la actividad

Aquí se puede observar como PB0 y PB1 son entradas que sirven para indicar cuál va a ser el dispositivo a visualizar si el 7-segmentos de la izquierda o el de la derecha.

También tenemos que para manejar la información que se va a transmitir a PB0 o PB1 solo tenemos 8 entradas del PD0 → PD7, con lo que si queremos mostrar los dos displays simultáneamente tenemos que ir alternando la visualización de uno y su actualización con la visualización del otro y su actualización.

De manera que esta placa de expansión se coloca sobre los pines de arduino y una vez que este alimentada se procede a la implementación del código.



Conexionado de un dígito 7-segmentos

Código:

```
int segPins[] = {
  0, 1, 2, 3, 4, 5, 6, 7};

int tiempototal= 1000;

int j = 40;

int disp1 =8;
int disp2= 9;

int dat1 = 0;
int dat0 = 0;

void setup() {
  // put your setup code here, to run once:
  // loop over the pin array and set them all to output:
  for (int thisseg = 0; thisseg < 8; thisseg++) {
    pinMode(segPins[thisseg], OUTPUT);
  }
  pinMode(disp1, OUTPUT);
  pinMode(disp2, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  //Llamada a función refresh pasando los datos correctos

  refresh(dat1,dat0);

  // Cálculo correcto de los datos dat1, dat0 --> desde 0 0 hasta 5 9

  dat1 = dat1 + 1;
  if(dat1 == 10){
    dat0 = dat0 + 1;
    dat1 = 0;
  }
  if(dat0 == 6){
    dat0 = 0;
    dat1 = 0;
  }
}

// Función refresh: Duración total de ejecución de refresh: tiempototal.
//Se van intercambiando los displays (disp1 con dat0 y disp2 con dat1) a una frecuencia que evite el parpadeo (j--> numero de veces que se activan ambos displays)
void refresh( int dat1, int data0) {
  int tiempo_refresco = tiempototal/(2*j);

  // Bucle de activación de los displays. El bucle se ejecuta j veces. Para escribir en los displays se llama a la función write_data (dato)

  for(int i = 0; i <= j;i++){
    digitalWrite(disp1,0);
    digitalWrite(disp2,1);
    write_data(dat1);
    delay(tiempo_refresco);
    digitalWrite(disp1,1);
    digitalWrite(disp2,0);
    write_data(dat0);
    delay(tiempo_refresco);
  }
}

// Función write_data(dato): Transforma dato en su código siete segmentos para escribirlo en el display

void write_data (int arg) {

  switch (arg) {
    case 0:
      //do something when var equals 1
      write7seg(0x7e);
      break;
    case 1:
      //do something when var equals 2
      write7seg(0x30);
      break;
    case 2:
      //do something when var equals 1
      write7seg(0x6d);
      break;
    case 3:
      //do something when var equals 2
      write7seg(0x79);
      break;
    case 4:
```

```

        //do something when var equals 1
        write7seg(0x33);
        break;
    case 5:
        //do something when var equals 2
        write7seg(0x5b);
        break;
    case 6:
        //do something when var equals 1
        write7seg(0x1f);
        break;
    case 7:
        //do something when var equals 2
        write7seg(0x70);
        break;
    case 8:
        //do something when var equals 1
        write7seg(0x7f);
        break;
    case 9:
        //do something when var equals 1
        write7seg(0x73);
        break;
}

// Función write7seg(data_7seg): escribe el valor data_7seg en el dsplay

void write7seg (unsigned char arg) {
    unsigned char segmen = 0x01;
    unsigned char display1;
    display1 = arg;

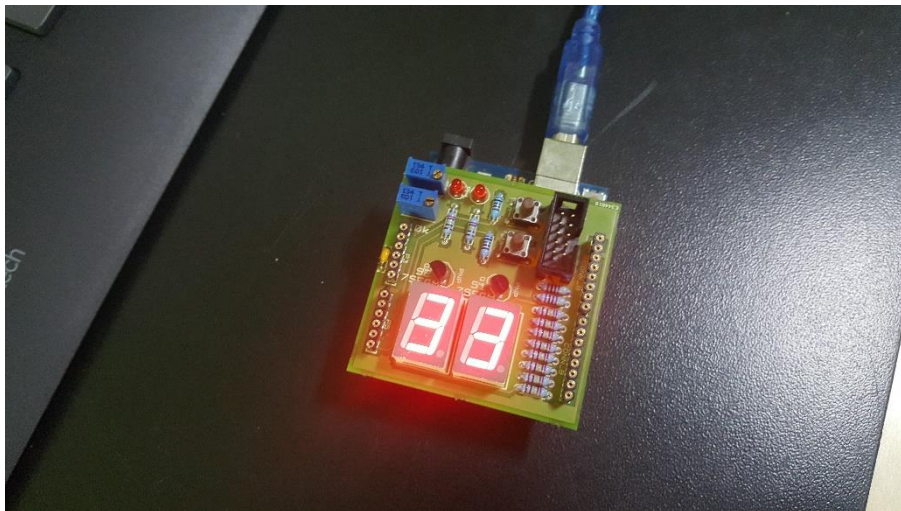
    for (int i = 0; i < 8; i++) {
        if ((display1 & segmen) == 0x00)
            digitalWrite(i, LOW);
        else
            digitalWrite(i, HIGH);

        segmen <<= 1;
    }
}

```

El código esta de manera que dat1 son las unidades de segundo y dat0 son las decenas de segundo de manera que cada 9 dat0 se incrementa en 1 y cuando dat1 llega a 6 se ponen dat0 y dat1 a 0.

Estos datos se van actualizando de manera simultánea cada segundo de manera que si tenemos que ir cambiando la actualización de los dos displays en cada segundo lo hagamos un



j veces cambio de display.

Por lo que si nosotros cambiamos el valor de j, que es el valor de actualización de los displays en un segundo podremos o no ver el parpadeo.

Demostración del resultado:

https://1drv.ms/v/s!Ao2_30mhw3birnbqkr0skM0JAGpl

➤ Evaluación después de la actividad

Practica en la que se ha puesto en práctica el uso de un contador en bcd, se ha tenido que aprender a usar dos dígitos con un solo bus de información, el manejo de ese bus con velocidades altas inapreciables a la vista.

Manejo de métodos que transformen el número a bcd con subrutinas de escritura del número.

➤ Incidencias y anécdotas

- Actualización de los números, pero ambos displays a la vez
- Uso de delay () incorrecto
- Mal entendimiento de división de 1 segundo por j veces por ambos displays.
- Actualización incorrecta de ambos displays en el contador ascendente, no cambio de decena de segundo a 0 cuando se llegaba a 59.

➤ Valoración personal

Practica intuitiva, pero en la que se debe de tener los conceptos claros, tanto en el manejo de escritura de un dígito bcd como la alternancia de actualización de ambos displays simultánea.

Me ha costado entender el concepto de frecuencia de refresco.