Práctica 1 - Equipo de laboratorio
Interfaces y periféricos para robots

Roberto Cadena Vega

Objetivos

Familiarizarse con el equipo del laboratorio de electrónica y comprender el funcionamiento del lenguaje de programación Wiring-Arduino.

Conocimientos Previos

Lenguajes de programación para microcontroladores

Dentro del encuadre de esta materia se maneja como objetivo de conocimiento el utilizar lenguajes de programación para microcontroladores de alto nivel y de bajo nivel, por lo que resulta bastante conveniente utilizar un ambiente de desarrollo como el de Wiring-Arduino, el cual nos permite empezar a trabajar con un lenguaje de programación de alto nivel, facil de comprender y de dominar, aprendiendo los conceptos importantes del funcionamiento de un μC (microcontrolador) y dejar para prácticas posteriores el uso de lenguajes de programación de bajo nivel, con la ventaja de poder utilizar el mismo μC con las mismas herramientas de software.

Por el momento empecemos a revisar el ambiente de desarrollo de Arduino en la figura 1

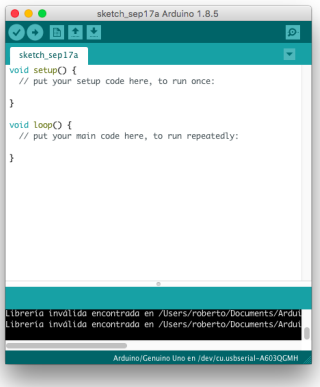

Figura 1: IDE de Arduino

En la parte superior de la ventana se puede encontrar los siguientes botones:

Verificar - Este boton manda la orden de compilar el programa para asegurar que no haya errores de compilación.

Subir - Este boton manda la orden de compular el programa y envar el archivo binario para la escritura dentro de la memoria del μC.

Nuevo - Abre una nueva ventana del IDE con un archivo nuevo.

Abrir - Abre el navegador de archivos para escoger el archivo a abrir.

Salvar - Guarda el archivo de la ventana actual.

En la parte central de la ventana se encuentra el espacio para redactar el programa a ejecutar y la ventana de mensajes en donde se despliega el estado del compilador.

En el borde inferior derecho de la ventana se encuentra especificado el modelo de tarjeta de desarrollo a utilizar por el compilador y el puerto especifico en el que esta conectado. Esta información se puede modificar en el menu principal del IDE, en la opción de Herramientas.

Cuando se abre el IDE de Arduino por primera vez, se tiene el siguiente programa por default:

```
1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
```

En donde podemos ver dos bloques principales para nuestra programación:

setup - En este función existen las indicaciones para inicializar el μC, es decir las instrucciones que se ejecutarán en el μC solo una vez (cuando el μC inicia su operación).

loop - En este función existen las instrucciones que se ejecutarán una y otra vez mientras el μC este funcionando.

Podemos ver un ejemplo de programación en Arduino directamente en los ejemplos del software (en el menu principal, Archivo, Ejemplos, 01.Basics, Blink):

```
1  /*
2    Blink
3
4    Turns an LED on for one second, then off for one second, repeatedly.
5
6    Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
7    it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
8    the correct LED pin independent of which board is used.
9    If you want to know what pin the on-board LED is connected to on your Arduino
10   model, check the Technical Specs of your board at:
11   https://www.arduino.cc/en/Main/Products
12
13   modified 8 May 2014
14   by Scott Fitzgerald
15   modified 2 Sep 2016
16   by Arturo Guadalupi
17   modified 8 Sep 2016
18   by Colby Newman
19
20   This example code is in the public domain.
21
22   http://www.arduino.cc/en/Tutorial/Blink
23 */
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
34   delay(1000);                       // wait for a second
35   digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
36   delay(1000);                       // wait for a second
37 }
```

En las lineas 1 - 23 se tiene un comentario de multiples lineas, en la linea 25 se tiene un comentario de una sola linea.

En la linea 28, adentro de la función setup, se tiene la inicialización de un pin del puerto de salida de la tarjeta de desarrollo, en donde se configura el pin LED_BUILTIN, como un pin de salida, es decir, se configura de tal manera que el pin pueda cambiar el voltaje de acuerdo a la programación a continuación[1].

En la linea 33, adentro de la funcion loop, se cambia el estado del pin LED_BUILTIN a alto, lo que hará que el pin físico del μC asociado a LED_BUILTIN[2] tenga un voltaje de $5V$, en la linea 34 se llama a la función delay con un argumento de 1000, la cual hará que el μC espere $1000ms$ o bien $1s$, en la linea 35 se cambia el estado del pin LED_BUILTIN a bajo, con lo que el pin físico del μC asociado a LED_BUILTIN tenga un voltaje de $0V$ y en la linea 36 se llama a la función delay para esperar $1s$.

[1] Lo contrario sería configurar el pin como entrada, lo cual dejaría al programa sin la capacidad de modificar el voltaje en este, pero con la capacidad de leer un voltaje causado por una fuente externa.

[2] En las lineas iniciales de comentarios se describe a LED_BUILTIN como el pin 13 para las tarjetas de desarrollo más comúnes de Arduino, por lo que si se reemplaza LED_BUILTIN con 13, el funcionamiento de este programa será el mismo.

El resultado final de este programa será que el μC encenderá y apagará el LED instalado en la misma placa de desarrollo con intervalos de $1s$.

En el siguiente ejemplo se describe como leer el voltaje del pin $A0$ y utilizar este valor para encender y apagar un LED con intervalos basados en este valor.

```
int sensorPin = A0;     // select the input pin for the potentiometer
int ledPin = 13;       // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the sensor

void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
}
```

En el siguiente ejemplo se decribe como encender un LED con una intensidad visual que sube y baja.

```
int ledPin = 9;     // LED connected to digital pin 9

void setup() {
  // nothing happens in setup
}

void loop() {
  // fade in from min to max in increments of 5 points:
  for (int fadeValue = 0 ; fadeValue <= 255; fadeValue += 5) {
    // sets the value (range from 0 to 255):
    analogWrite(ledPin, fadeValue);
    // wait for 30 milliseconds to see the dimming effect
    delay(30);
  }

  // fade out from max to min in increments of 5 points:
  for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -= 5) {
    // sets the value (range from 0 to 255):
    analogWrite(ledPin, fadeValue);
    // wait for 30 milliseconds to see the dimming effect
    delay(30);
  }
}
```

Por el momento estos son todos los ejemplos que se van aanalizar, sin embargo tienes la libertad de revisar los ejemplos incluidos dentro del IDE de Arduino para aprender diferentes funciones.

EQUIPO

El siguiente equipo será proporcionado por el laboratorio, siempre y cuando lleguen en los primeros 15 minutos de la práctica, y hagan el vale conteniendo el siguiente equipo (exceptuando las pinzas).

- Fuente de Alimentación
- Osciloscopio
- Cables de alimentación
- Pinzas

Materiales

- Protoboard
- LED (no importa el color, aunque los difusos son más fáciles de ver en las condiciones de iluminación del laboratorio)
- Resistencias
  - 120Ω
  - 180Ω
  - 220Ω
- Cables

Desarrollo

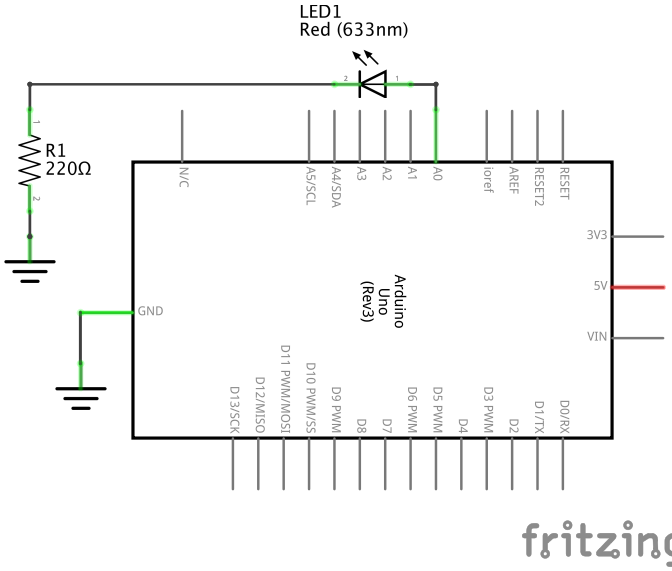

Figura 2: Diagrama esquemático del circuito a ensamblar.

Lo primero que tenemos que hacer es realizar el circuito eléctrico en el protoboard. El circuito lo podemos ver en la figura 3; este equema es una representación pictográfica bastante realista del circuito a realizar, sin embargo, en ocasiones puede resultar demasiada información redundante, sobre todo cuando ya se tiene experiencia realizando circuitos eléctricos, por lo que tambien podemos verlo como en la figura 2 y obtener la misma información.

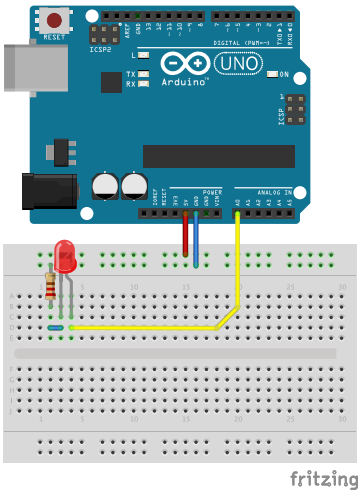

Figura 3: Diagrama representativo del circuito a ensamblar.

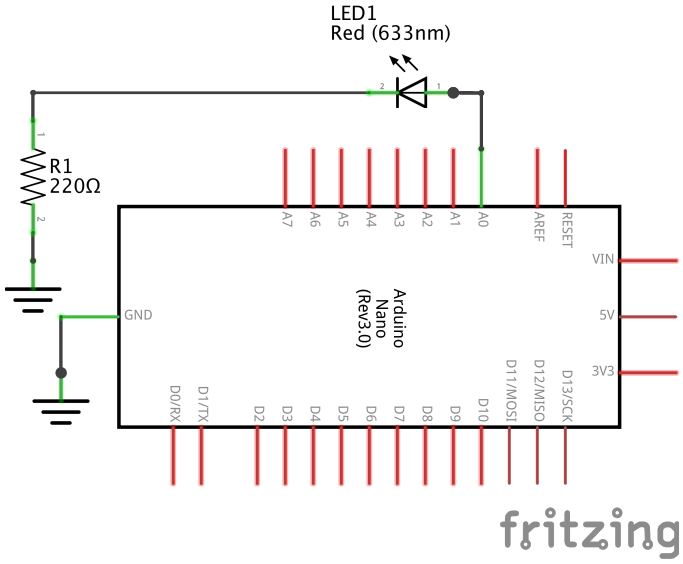

Figura 4: Diagrama esquemático alternativo del circuito a ensamblar.

Si bien la manera mas facil y simple de seguir estas prácticas es con un Arduino UNO, tambien se puede utilizar cualquier otra tarjeta de desarrollo que utilice Wiring-Arduino, por lo que ponemos también un ejemplo con la tarjeta de desarrollo Arduino NANO en la figura 5, así como su diagrama esquemático correspondiente en la figura 4.

Nota que los diagramas especifican el cable de conexion al arduino en *A0*, sin embargo, de acuerdo a los ejemplos es necesario conectar el LED en diferentes pines del Arduino y en algunos casos multiples LED.

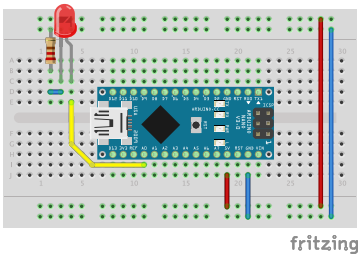

Figura 5: Diagrama representativo alternativo del circuito a ensamblar.

Conclusiones

El alumno deberá describir sus conclusiones al final de su reporte de práctica.

Hoja de Anotaciones

Anota los pasos a seguir para utilizar correctamente la fuente de alimentación.

Anota los pasos a seguir para utilizar correctamente el multímetro como Vóltmetro.

Carga el ejemplo del IDE de Arduino ubicado en Menu principal, Archivo, Ejemplos, 03. Analog, Fading y modifica el diagrama esquemático de la figura 2 para que el ejemplo funcione.

Diseña un circuito eléctrico en el cual 3 LEDs esten conectados a 3 pines del Arduino y se enciendan cuando estos pines del Arduino tengan un voltaje de $5V$.

Diseña un programa de Arduino que prenda secuencialmente los 3 LEDs dejandolos encendidos $1s$, $1s$, y $2s$

Integrantes del equipo: Revisó:

_____

_____ _____