Universidad de Costa Rica

Grupo de Robótica de la Universidad de Costa Rica

Curso: Introducción al controlador Arduino

I ciclo 2014

Curso: Introducción al controlador Arduino

Laura Fonseca Picado
Jeffry Luque Agüero

Índice

Índice de figuras

1. Conocimiento eléctrico básico antes del uso del Arduino

1.1. Ley de Ohm

La ley de Ohm establece la relación entre la tensión la corriente y la resistencia.

$$V = IR \tag{1}$$

Donde, I es la corriente eléctrica que pasa a través del objeto en amperios, representando la velocidad de los electrones, V es la diferencia de potencial de las terminales del objeto en voltios que es la fuerza que mueve a los electrones, y R es la resistencia en ohmios (Ω) a representan la oposición del material al flujo de electrones. Se puede explicar estos conceptos por medio de la siguiente imagen.

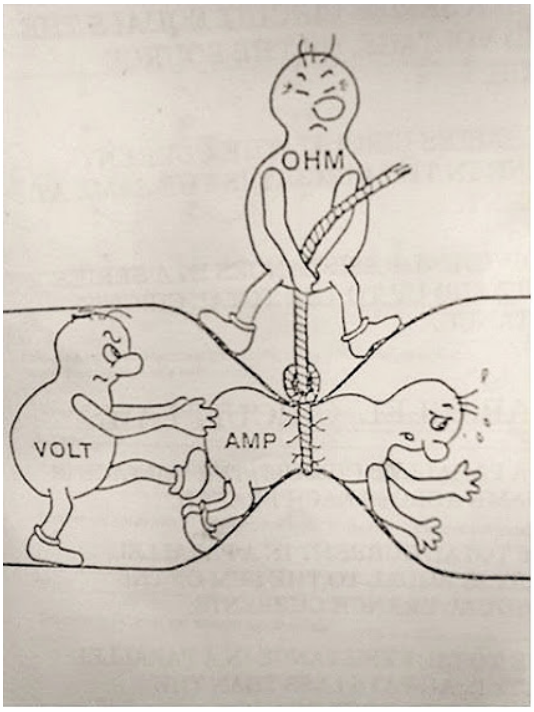

Figura 1: Ley de Ohm

1.2. Circuitos en serie y en paralelo

Serie

El circuito en serie es una configuración de conexión donde las terminales de cada dispositivo o elemento eléctrico (resistencias, condensadores, interruptores, entre otros) se conectan secuencialmente. Es decir la terminal de salida de una elemento se conecta al terminal de entrada del elemento siguiente, un ejemplo es la conexión de las baterías, el terminal positivo se conecta al terminal negativo de la batería siguiente.

La corriente que circula en un circuito en serie es la misma en todos los puntos de circuito. La resistencia equivalente (esto para reducir circuitos) de varias resistencias en serie es la suma de los valores de dichas resistencias.

$$Rt = R1 + R2 + R3..... + Rn \tag{2}$$

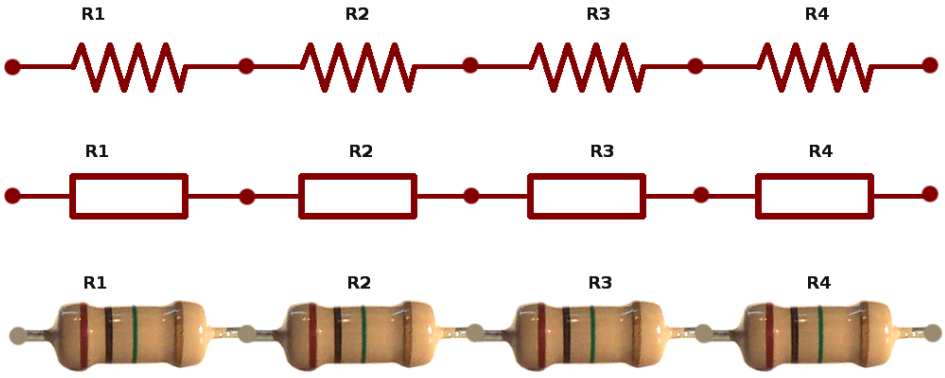

Figura 2: Circuito en serie

Paralelo

El circuito en paralelo es una conexión donde las terminales de entrada de todos los dispositivos o elementos eléctricos del circuito conectados coincidan entre sí, lo mismo con sus terminales de salida. Un ejemplo de un circuito en paralelo son las bombillas de iluminación de una cada, porque aunque se apague una bombillas las demás pueden seguir encendidas.

El voltaje que circula en una circuito en paralelo entre dos terminales es el mismo a otras dos terminales que se encuentran en paralelo. Por lo que la resistencia equivalente de varias resistencias en paralelo es el recíproco matemático de la suma de los recíprocos de las resistencias.

$$\frac{1}{Rt} = \frac{1}{R1} + \frac{1}{R2} + ...\frac{1}{RN} \tag{3}$$

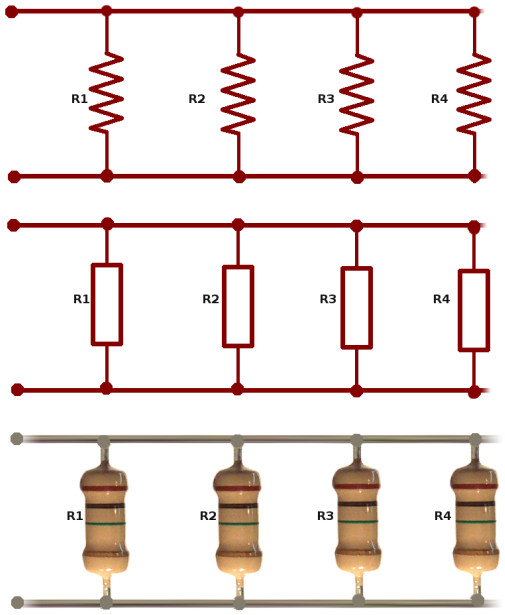

Figura 3: Circuito en paralelo

1.3. Uso del protoboard

Es una tabla que permite la interconección de componentes electrónicos sin necesidad de soldarlos. Así, se puede realizar ejemplos experimentales de manera fácil y ágil. Es una tabla con orificios los cuales están conectados entre sí en un orden coherente. Una protoboard tiene el siguiente aspecto:

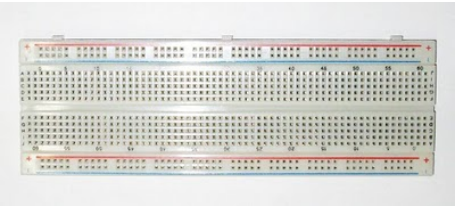

Figura 4: Protoboard

Como se puede apreciar, consta de un gran número de agujeros, que nos permitirá insertar los elementos electrónicos en él. Los agujeros centrales, se utilizan para armar los circuitos, en los agujeros que forman filas y están en la parte superior e inferior se puede ver una linea roja y una linea azul, normalmente se utiliza el rojo para VCC y el azul para GND,para tener acceso a estos a lo largo de toda la tarjeta, es solo una referencia, pero se puede utilizar de cualquier forma.

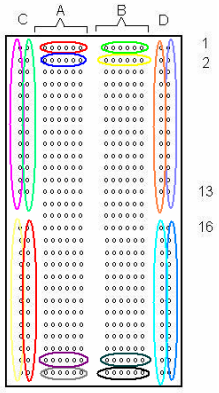

Figura 5: Nodos protoboard

En la Figura se muestran de diferentes colores los conjuntos de orificios que pertenen al mismo nodo y por ende poseen la misma tensión.

1.4. Dividor de tensiones y de corriente

Divisor de tensión

Muchas veces se tienen componentes que funcionan a diferentes tensiones de los 3.3V y los 5V que tiene de salida el Arduino. Un divisor de tensión permite por medio de dos resitencias obtener otros niveles de tensiones. La configuración lo muestra el siguiente circuito:

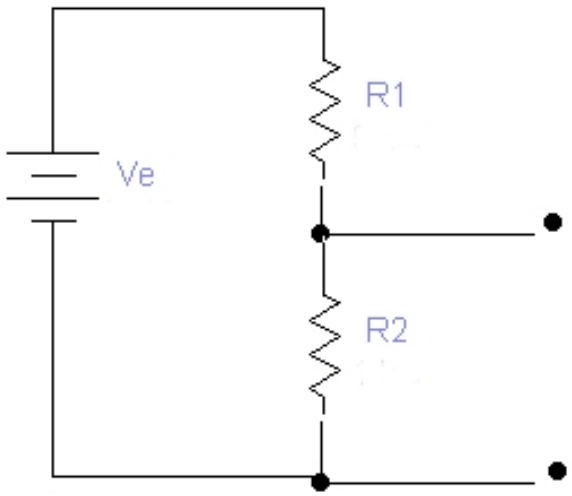

Figura 6: Divisor de tensión

$$V_{R1} = \frac{V_e R1}{R1 + R2} \tag{4}$$

$$V_{R2} = \frac{V_e R2}{R1 + R2} \tag{5}$$

Divisor de corriente

Siempre en todo circuito se debe de tener un control de las corrientes ya que si estás son muy grandes pueden dañar los componentes. Si se tienen varios elementos en paralelo se puede determinar la corriente en cada rama.

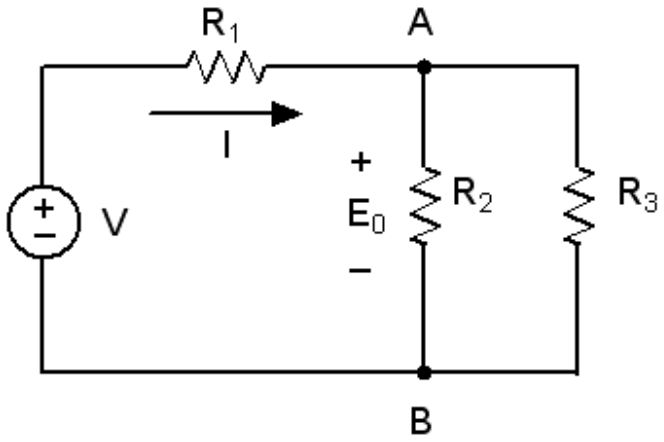

Figura 7: Divisor de corriente

$$I_{R2} = \frac{IR3}{R2 + R3} \tag{6}$$

$$I_{R3} = \frac{IR2}{R3 + R2} \tag{7}$$

2. Especificaciones de la tarjeta arduino

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos. [1]

Este se muestra en la Figura 8.

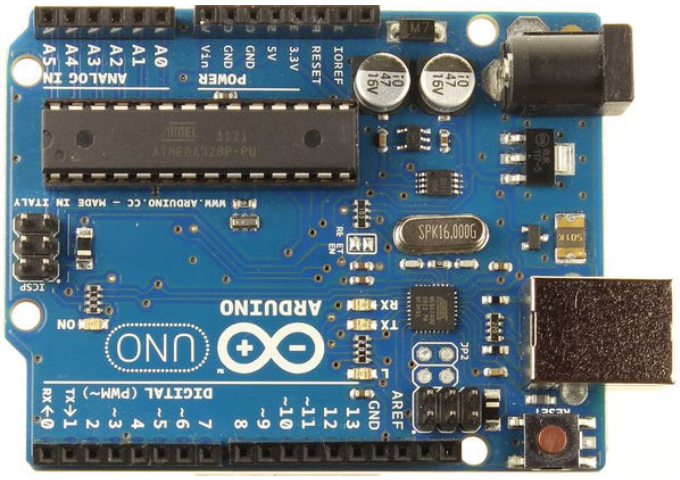

Figura 8: Arduino

Arduino UNO es una plataforma que contiene un microcontrolador ATmega328, permite el uso de entradas analógicas, entradas y salidas digitales y salidas PWM (Pulse width modulation).

2.1. Caracteristicas eléctricas

La Arduino UNO posee todo lo que se necesita para manejar el controlador, simplemente se conecta a un computador por medio del cable USB o se puede alimentar utilizando una batería o un adaptador AC-DC.

- Microcontrolador: ATmega328
- Tensión de operación: 5V
- Tensión recomendada: 7-12 V
- Tensión limite: 6-20V

- Corriente DC por pin I/O: 40mA
- Corriente DC por pin 3.3V: 50mA
- Memoria Flash: 32 KB (0.5 KB usados por bootloader)
- SRAM: 2 KB
- EEPROM: 1 KB
- Frecuencia de reloj:16 MHz

2.2. Puertos de potencia

- **Vin:** Si no se usa el cable USB, se puede conectar alimentación directa a este puerto.
- **GND:** Tierra
- **5V:** Tiene una salida de 5V.
- **3.3V:** Tiene una salida de 3.3V.
- **RESET:** En bajo se ha reiniciado la tarjeta.
- **aref:** Permite modificar el rango en tensión de entrada a los pines análogicos.

2.3. Puertos digitales

- Posee 14 puertos para salidas y entradas digitales.
- 6 de los puertos digitales se pueden usar como PWM.
- Los pines 0 y 1 se pueden usar como TX y RX de una conexión serial
- Estos pines funcionan bajo 5V de tensión y no sobrepasan los 40mA de corriente.

2.4. Puertos análogicos

- Poseen una resolución de 10 bits, valores desde 0-1023
- Rango de tensión de 0-5V pero se puede cambiar el rango por el puerto aref.
- Los pines A4 (SDA pin) y A5 (SCL) pin, también soportan comunicación TWI.

3. IDE

Para empezar a programar la placa es necesario descargarse el entorno de desarrollo (IDE), así como los diferentes drivers necesarios para el buen funcionamiento.

El entorno de código abierto Arduino hace fácil escribir código y cargarlo a la placa. Funciona en Windows, Mac OS X y Linux. El entorno está escrito en Java y basado en Processing, avr-gcc y otros programas también de código abierto.[2] Al encontrase desarrollado en Java este debe de estar instalado en el sistema para poder ejecutarse. El lenguaje de programación de arduino tiene grandes similitudes con el lenguaje C++.

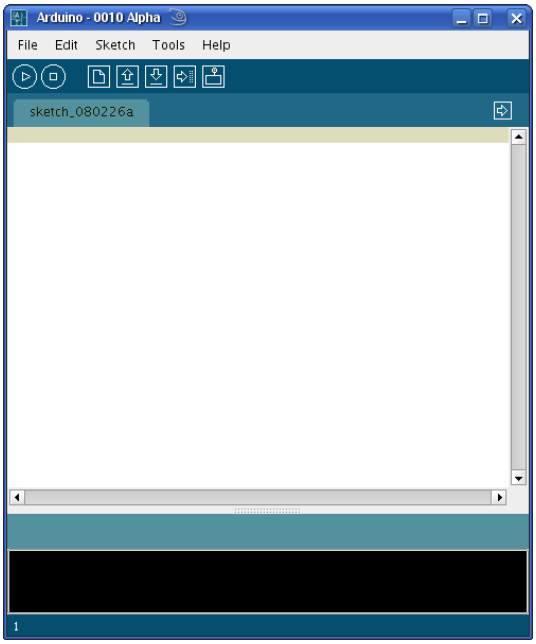

Figura 9: IDE de Arduino

3.1. Instalación Windows 7

Se descarga el programa desde el link `<http://arduino.cc/en/Main/Software>`

IMPORTANTE: es necesario descargar la versión estable más nueva para asegurarse de que la placa sea reconocida por la computadora. Aunado, para que la computadora reconozca la placa de arduino, es necesario instalar los drivers que trae la carpeta descargada. Los pasos realizados para esto fueron los siguientes:

- 1. Se conectó la placa a la computadora. Luego de esto, el sistema operativo intentó instalar automáticamente los drivers y falló.
- 2. En el panel de control, se ingresó a "dispositivos e impresoras", en donde apareció la placa de arduino como 'dispositivo desconocido'. Se picó dos veces y apareció una ventana.
- 3. En la pestaña de 'Hardware', se accedió a 'propiedades', y en la nueva vetana a 'cambiar configuración'.
- 4. En la ventana que se abrió al hacer click en 'cambiar configuración', se accedió a la pestaña de 'driver' y se hizo click en actualizar driver (o update driver).
- 5. En la nueva ventana que se abrió, se seleccionó 'buscar en mi computadora por el driver (o browse my computer for driver software)', y se buscó dentro de la carpeta de arduino -descomprimida del archivo ZIP descargado- la carpeta 'driver'. Se seleccionó esta (completa, no solamente la otra carpeta interna), y luego de hizo click en la opción de 'next'.
- 6. El proceso de instalación del driver se llevó a cabo correctamente. Se verificó esto en 'dispositivos e impresoras' en el panel de control, en donde lo que antes era 'dispositivo desconocido' se convirtió en 'Arduino Uno (COM3)'. Nótese que COM3 puede variar por COM1, COM2, COM4... Lo cual es importante reconocer para la escogencia del puerto serial (serial port) en el software de arduino.

3.2. Instalación Linux

Arduino corre en una maquina de java, por ello es necesario instalarla

```
sudo apt-get install openjdk-6-jdk
```

Si se tiene Ubuntu se puede correr el siguiente comando para descargar e instalar el IDE.

```
sudo apt-get update sudo apt-get install arduino arduino-core
```

o

```
sudo apt-get install arduino
```

También se puede descargar el programa desde el link `<http://arduino.cc/en/Main/Software>`

Después de descomprimir la carpeta, desplazarse hasta donde está el ejecutable y darle permisos, para poder utilizar los puertos.

```
sudo chmod 777 arduino
```

3.3. Estructura básica de un programa

La programación en Arduino es bastante simple y el programa principal se divide en dos partes, Setup() que contiene la preparación del programa y loop() que contiene el contenido a ejecutar. En la función Setup() se incluye la declaración de variables globales, apertura de puertos y definición de pines. Es la primera función que se ejecuta y se hace solo una vez. La función loop() incluye el código principal, este se mantiene corriendo una y otra vez hasta que se resetee por medio del botón o hasta que se desconecte el arduino de la alimentación.

```
1  //Declaración de variables gobales
2
3  void setup() {
4    //Apertura de puertos
5    //Definición de puertos
6  }
7
8  void loop() {
9    //variables locales
10   //lectura y escritura de datos
11   //uso de sensores y actuadores etc..
12 }
```

Listing 1: Estructura programa principal

Por ejemplo:

```
1  int pin=4;          //Declaración de variables
2  void setup() {
3    inMode(pin,OUTPUT); // Establece 'pin' como salida digital .
4  }
5
6  void loop() {
7    digitalWrite(pin, HIGH);  // Pone pin en alto
8    delay(1000);              // Pausa un segundo
9    digitalWrite(pin, LOW);   // Pone pin en bajo
10   delay(1000);              // Pausa un segundo
11 }
```

Listing 2: Ejemplo estructura prinicipal

3.4. Variables

Cada variable debe ser previamente declarada y opcionalmete asignada a un determinado valor antes de ser llamada. Cuando se declara se debe indicar el tipo de datos que almacenará(int, float,long).

```
float nombreDeLaVariable = 0.5;
```

Hay diferentes espacios donde se puede declarar una variable. Se pueden declarar al inicio del programa, incluso antes de la función Setup(), localmente en una función determinada, y además dentro de cualquier bloque, como por ejemplo un bucle. Dependiendo de donde es declarada la variabla así será el ámbito de utilización permitido. Una variable global puede ser utilizada en todo el programa, cualquier función que la llame.

```
int variable;    // 'variable' es visible en todo el programa .
void setup() { }  // no se requiere setup .
void loop() {
  for (int j=0; j<20;){ // 'j' es visible solo en el bucle
    j++;
    }
  float f; // 'f' es visible únicamente en la función loop()
}
```

Listing 3: Variables

3.5. Funciones

Cada función, un bloque de código, se identifica por un nombre y esta se ejecuta cuando es llamada por su respectivo nombre. Cuando se declara una función se debe incluir que tipo de datos es el que devuelve. (Ej. int si lo que devuelve es un entero). Una vez establecido el tipo de dato se especifica el nombre de la función y los parámetros de la misma.

```
int nombreDeLaFuncion(parametro)
```

4. Pines digitales

4.1. Descripción

Las señales digitales eléctricas son variables con dos niveles o estados bien definidos, permitiendo transmitir información según un código establecido previamente acordado. Cada nivel representa un 1 o un 0, un falso o un verdadero. Por ejemplo si se usan dispositivos de la familia lógica TTL se usan estados lógicos de 0V y 5V, con cierto rango de incertidumbre aceptado. La familia lógica CMOS tiene sus estados dependiendo de su alimentación. Al tener solo dos estados estas señales son más inmunes al ruido. [3]

Una señal digital podría ser, un mensaje en código morse o el encendido y apagado de un bombillo.

En el caso de los pines del arduino, el 1 se ve representado por 5V y el 0 por 0V. Si se va a utilizar algún sensor o componente es necesario verificar que sus niveles funcionan de forma analóga.

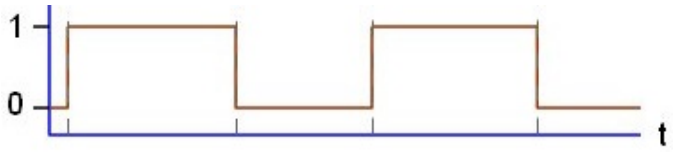

Figura 10: Señal digital

4.2. Funciones

- **pinMode(pin, mode):** Está función es usada en la función Setup() para determinar el comportamiento de algún pin dado. Puede comportarse de dos maneras INPUT o OUTPUT. Ejemplo: pinMode(13, OUTPUT); configura el pin número 13 como salida. Cada pin de arduino se establece por defecto como entrada, por lo que no es necesario declararlos como entradas utilizando pinMode().
- **digitalWrite(pin, value):** Lee el valor desde un pin digital específico. Esta función devuelve un valor HIGH o LOW. La palabra ´pin´ hace referencia a una variable númerica o una constante (0-13). Ejemplo v=digitalRead(5). Esto por que los pines digitales van del 0 al 13.

- **digitalRead(pin):** Se introduce un nivel alto (HIGH) o bajo (LOW) en algún pin digital específico. Ejemplo digitalWrite(13, HIGH).

4.3. Ejemplo led

```
int pin=4;          //Declaración de variables
void setup() {
  inMode(pin,OUTPUT); // Establece 'pin' como salida digital .
}

void loop() {
  digitalWrite(pin, HIGH);  // Pone pin en alto
  delay(1000);          // Pausa un segundo
  digitalWrite(pin, LOW);   // Pone pin en bajo
  delay(1000);          // Pausa un segundo
}
```

Listing 4: Ejemplo pin digital-led

5. Pines analógicos

5.1. Descripción

Las señales analógicas son variables eléctricas que evolucionan en el tiempo de manera continua entre dos límites, representan alguna variable física directamente. [3]

Una señal análogica podría ser la temperatura de un cuarto, la distancia entre dos puntos, la cantidad de luz en el ambiente. Por medio de un transductor estos valores se pasan a señales eléctricas como corrientes y tensiones.

El arduino acepta valores análogicos de tensiones desde 0V a 5V, toma el dato y lo pasa por un convertidor analogico/digital, que obtiene la representación digital del dato con una resolución de 10 bits.

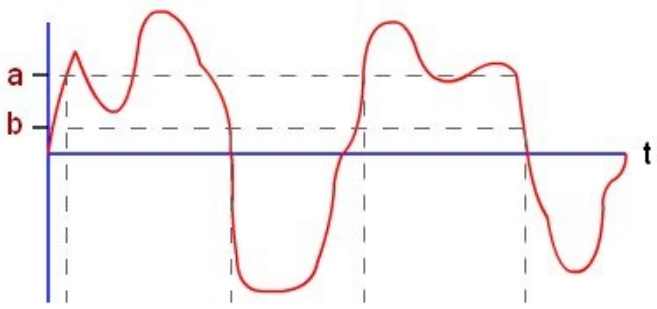

Figura 11: Señal analógica

5.2. Funciones

- **analogRead(pin):** Lee el valor que determina el pin analógico específico. Esta función solo funciona con los pines analógicos, por lo tanto la palabra pin hace referencia a una variable numérica o constante (0-5). El valor resultante es un entero de 0 a 1023. Los pines analógicos no se necesitan declarar previamente como OUTPUT o INPUT.

5.3. Ejemplo potenciometro 1

Se controlorá la frecuencia de parpadeo de un LED. En la Figura 12, se observa como colocar el potenciómetro y el LED en la protoboard. Es necesario poner una resistencia que permita que no se sobrepase la corriente límite del LED. Imagen tomada de [4]

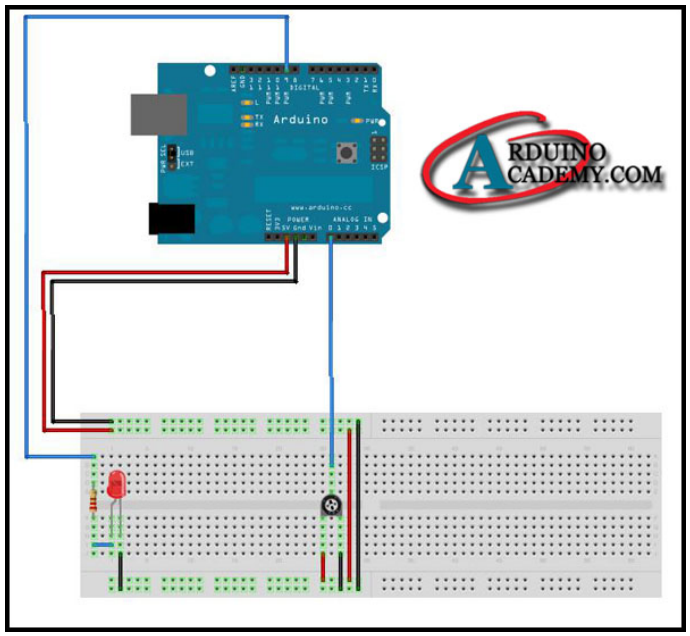

Figura 12: Circuito potenciometro-led

```
int potPin = 0; // Pin de entrada para el potenciómetro
int ledPin = 9; // Pin de salida para el LED
void setup() {
  pinMode(ledPin, OUTPUT); // Declara el pin del LED como de salida
}
void loop() {
  digitalWrite(ledPin, HIGH); // Enciende el LED
  delay(analogRead(potPin)); // Lee el valor del potenciómetro
  digitalWrite(ledPin, LOW); // Apaga el LED
  delay(analogRead(potPin)); //Según lo leído es el tiempo de espera.
}
```

Listing 5: Ejemplo Potenciometro 1

5.4. Ejemplo potenciometro 2

Se tomarán datos de la entrada analógica A0 para variar la frecuencia de parpadeo de un diodo LED conectado a una salida digital cualquiera.

```
int pinSensor = A0; // Entrada para el potenciómetro.
int pinLed = 9 ; // Seleccionamos pin para el Led.
int valorSensor = 0; // variable para el valor del sensor.
void setup() {
  // Declaramos el pin del Led como salida:
  pinMode(pinLed, OUTPUT);
}
void loop() {
  // Leemos el valor del sensor y lo almacenamos:
  valorSensor = analogRead(pinSensor);
  // encendemo el diodo LED:
  digitalWrite(pinLed, HIGH);
  // Detenemos el programa durante <valorSensor> milisegundos:
  delay(valorSensor);
  // Apagamos el diodo Led:
  digitalWrite(pinLed, LOW);
  // Detenemos el programa durante <valorSensor> milisegundos:
  delay(valorSensor);
}
```

Listing 6: Ejemplo Potenciometro 2

5.5. Ejemplo potenciometro 3

Con el mismo circuito se realizará otra programación, ahora el LED variará su intensidad lumínica en función del valor que aporte el potenciómetro a nuestra entrada analógica. Para este ejemplo es sumamento importante tener conectado en algún pin analógico.

```
int pinSensor = A0; // Entrada para el potenciómetro.
int pinLed = 9; // Seleccionamos pin para el Led.
int valorSensor = 0; // variable para el valor del sensor.
void setup() {
  pinMode(pinLed, OUTPUT); // Establecemos el pin como salida.
}
void loop() {
  // Leemos el valor del sensor y lo almacenamos:
  valorSensor = analogRead(pinSensor);
  // Establecemos el valor analógico para la salida PWM
```

```
11    analogWrite(pinLed, valorSensor / 4);
      // Detenemos el programa durante 30 milisegundos:
13    delay(30);
}
```

Listing 7: Ejemplo Potenciometro 3

6. Pines PWM

6.1. Descripción

Las siglas PWM vienen de Pulse Width Modultation, o Modulación de Ancho de Pulso. Lo que hace este tipo de señal es emitir, en lugar de una señal continua en nuestra salida, una serie de pulsos (altos y bajos de señales digitales) que podremos variar su duración a una frecuencia especifica.

La frecuencia a la que se generan estos pulsos es tan grande, que permite que actuadores como los motores y diodos como los leds, realicen acciones proporcionales al promedio resultante de la señal, controlando así la velocidad del motor o la intensidad de luz. El ciclo de trabajo D se define como.

$$D = \frac{t}{T} \tag{8}$$

Con t= tiempo del ciclo en alto y T= tiempo total del ciclo.

Conforme se aúmenta el ciclo de trabajo la acción aumenta ya que el promedio aumenta. Conforme se disminuye la frecuencia es más notorio que son señales digitales y no analógicas, hasta el punto en que el motor y led se enciende y apaga a un nivel percibible.

En un arduino se puede determinar el ciclo de trabajo, más la frecuencia es fija y de un valor de 490Hz.

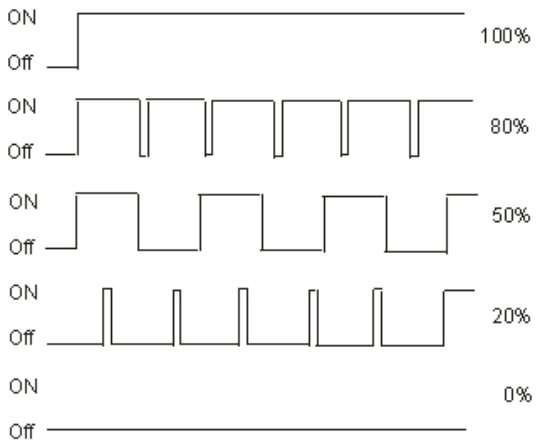

Figura 13: Señales PWM

6.2. Funciones

- **analogWrite(pin, value):** Escribe un valor speudo-analógico usando modulación por ancho de pulso (PWM). Esta función solo puede ser utilizada en los pines 3,5,6,9,10,11. (Ver Ilustración 1). Ejemplo analogWrite(6, v). Puede especificarse un valor de 0-255. Es decir un valor 0 genera 0 V y 255 genera 5 V en el pin especificado.

6.3. Ejemplo led

Un ejemplo para emplear el PWM es establecerm el brillo de un LED. A continuación se aumentará el brillo del LED en el pin 9 mediante PWM y luego se disminuirá.

```
int pinLed = 9; // Pin conttrolado por PWM
void setup(){}
void loop() {
  for (int i=0; i<=255; i++){
    analogWrite(ledPin, i); // Establece el brillo a i.
    delay(10);
  }
  for (int i=255; i>=0; i--) {
    analogWrite(ledPin, i);
    delay(10);
  }
}
```

Listing 8: Ejemplo PWM

7. Comunicación serial

7.1. Descripción

Una comunicación serial, implica que los datos se van pasando en cola, uno tras de otro, entre dos dispositivos. Para ellos se requiere un protocolo que marque el inicio y fin de los datos y la velocidad a los que se envian y cual de los dos puertos RX o TX tiene permiso de ser usado. Cada Intervalo de tiempo representa un bit de información.

Esta la comunicación serial asincrona, donde la velocidad de intercambio de bits es acordada de forma mutua y cronometrada de forma independiente entre el Transmisor y el Receptor. O la sincrona donde el tiempo de intercambio está estipulado previamente y los dos se rigen bajo el mismo cronometro.

Arduino posee como principal característica la habilidad de comunicarse con nuestra computadora a través del puerto serie. Esto se conoce como comunicación serial. Debido a que el uso de este puerto ha quedado un poco en desuso a favor de la tecnología USB, Arduino cuenta con un convertidor de Serial a USB que permite a nuestra placa ser reconocida por nuestra computadora como un dispositivo conectado a un puerto COM aún cuando la conexión física sea mediante USB.

Arduino se comunica a través de los pines digitales 0 (RX) y 1 (TX), así como con el ordenador mediante USB. Por lo tanto, si utilizas estas funciones, no puedes usar los pines 0 y 1 como entrada o salida digital.

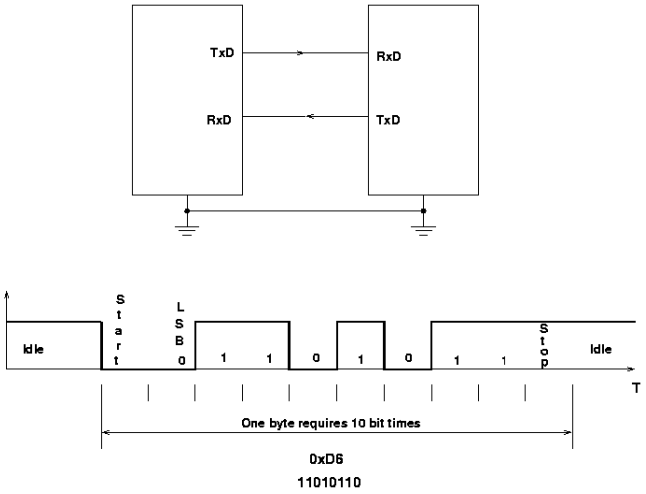

Figura 14: Señal serial

8. Arduino para sensores y actuadores específicos

8.1. Sensor Ultrasónico

Figura 15: Sensor ultrasónico

Especificaciones

- Voltaje de alimentación: 5V DC.
- Rango de medición: 2cm - 500cm.
- Resolución: 0,3 cm.
- 4 pines: VCC, Trig, Echo, GND.

Ejemplo

- Trig——>Pin 12.
- VCC——>5V.
- Echo——>Pin 13.
- GND——>GND

```
#define PIN_TRIG 12
// Se declaran las constantes correspondientes.
#define PIN_ECO 13 //Las conexiones previamente hechas.
void setup() {
  // Inicializacion de la comunicacion serial
  Serial.begin (9600);
  // Inicializacion de pines digitales
  pinMode(PIN_TRIG, OUTPUT);
  pinMode(PIN_ECO, INPUT);
}

void loop() {
  long duracion, distancia; // Variables a utilizar.
  /* Hacer el disparo*/
  digitalWrite(PIN_TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(PIN_TRIG, HIGH); // Flanco ascendente
  delayMicroseconds(10);
  // Duracion del pulso
  digitalWrite(PIN_TRIG, LOW); // Flanco descendente
  /* Recepcion del eco de respuesta */
  duracion = pulseIn(PIN_ECO, HIGH);
  /* Calculo de la distancia efectiva */
  distancia = (duracion/2) / 29;
  /* Imprimir resultados a la terminal serial */
  if (distancia >= 500 || distancia <= 0){
    Serial.println("Fuera de rango");
  }
  else {
    Serial.print(distancia);
    Serial.println(" cm");
  }
// Retardo para disminuir la frecuencia de las lecturas
delay(500);
}
```

Listing 9: Ejemplo Potenciometro 3

Se verifica el programa, se carga en la placa arduino. Y por último se abre el "monitor serial" desde herramientas y observar las mediciones.

Referencias

[1] Arduino. http://www.arduino.cc/es/.U1GyWvl5OAU, 2014.

[2] Arduino. http://arduino.cc/es/main/software.U1HOxPl5OAU, 2014.

[3] Federico Miraya. Convertidores d/a y a/d. http://www.fceia.unr.edu.ar/enica3/da-ad.pdf, 2004.

[4] Pablo Murillo. http://www.arduteka.com/2011/11/tutorial-arduino-0003-entrada-analogica-y-salida-pwm/, 2011.