

Benemérita Universidad Autónoma de Puebla



**Benemérita Universidad  
Autónoma de Puebla**  
Facultad de Ciencias  
de la Electrónica

**Asignatura**

Control Digital y Aplicaciones

**Proyecto**

Práctica 1 - Control de Motor

**Estudiantes**

Marlene Del Carmen Ahuactzin Villanueva

Héctor Jonathan Flores Freeman

Héctor José Vargas Ruiz

**Profesor**

Jaime Julián Cid Monjaraz

**Fecha**

19 de Enero de 2015

Control Digital y Aplicaciones

Primavera 2015

## Objetivo

Conocer y aprender de manera práctica el funcionamiento básica de la tarjeta de adquisición de datos Arduino.

## Introducción

Desde hace algunas décadas, ha surgido la gran necesidad de proveer a las industrias y manufactureras una manera simple y efectiva de controlar los sistemas automatizados que utilizan para los procesos, sin embargo, dicho control ha evolucionado de manera lenta pero persistente. En sus inicios todo comenzó por medio de sistemas meramente electromecánicos sumamente complejos y difíciles de modificar. Posteriormente se desarrolló la electrónica digital, permitiendo una mayor facilidad de programación y versatilidad para diseñar sistemas de control.

Actualmente disponemos de tarjetas de adquisición de datos que nos permiten manejar variables de una manera simple y eficiente, logrando así una mayor producción y ahorro de recursos en los procesos.

El presente reporte describe el trabajo realizado durante la práctica 1 que consiste en controlar el giro de un motor utilizando la tarjeta arduino e implementando una etapa de potencia.

Se describe el proceso del trabajo explicando la lógica del código realizado y el circuito implementado para la etapa de potencia, obteniendo así el resultado esperado a partir de varias pruebas donde el usuario puede regular directamente la velocidad del motor.

## Estado del Arte

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo diseñada para facilitar el uso de la electrónica en proyectos multidisciplinarios.



El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Dentro de los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280 y ATmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños.

La modulación por ancho de pulsos (PWM por sus siglas en inglés: Pulse-Width Modulation) de una señal o una fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (senoidal o cuadrada por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga. Finalmente, un motor de corriente continua (DC) es una máquina que convierte la energía eléctrica en mecánica, provocando un movimiento rotatorio, gracias a la acción de un campo magnético.

## Desarrollo

Para llevar a cabo esta práctica, es necesario contar con los siguientes elementos:

- Placa Arduino
- Motor DC
- Cable USB
- Entorno de desarrollo Arduino
- Cables de conexión
- Transistor TIP120
- Diodo 1N4004
- Resistencia de 10K
- Computadora

Para realizar esta práctica se hizo uso de un Arduino MEGA 2560 y un motor DC de DAGU robot DG01D 48:1 de 5V.

Para comenzar es necesario realizar el diagrama de conexión como se muestra a continuación:

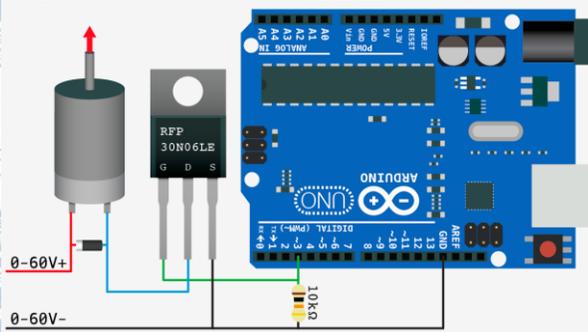


Figura 1 Esquema de conexión

Como se muestra en la figura 1, el motor se conecta directamente a una fuente de alimentación externa para su correcto funcionamiento y a su vez, al emisor del transistor. Asimismo, el motor lleva entre sus terminales un diodo para evitar corrientes de retorno.

Por otra parte, se conecta la resistencia en paralelo con la base del transistor y con la salida de señal PWM de la placa. Finalmente el colector y la fuente de alimentación se conectan en paralelo con la terminal de tierra de la placa.

Una vez realizado este paso, se procede a realizar la conexión vía USB a la computadora y se ejecuta el entorno de desarrollo de arduino.

A continuación, procedemos a diseñar el código que nos permite controlar la velocidad del motor mediante una señal PWM (Modulación de ancho de pulso) mandando el dato desde la computadora.

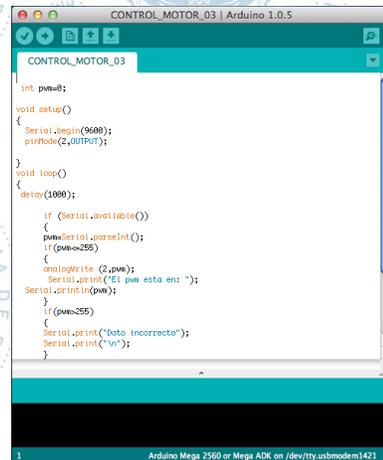


Figura 2 IDE Arduino 1.0.5

Una vez realizado el código en IDE de Arduino se procede a compilarlo y cargarlo en la tarjeta.

El código de Arduino funciona de la siguiente manera:

Lo primero es declarar una variable tipo entero llamada pwm (esta variable permitirá manipular el ancho de pulso).

Para enviar los datos desde la computadora al Arduino es necesario inicializar la comunicación Serial y utilizar una serie de instrucciones que permite escribir desde el monitor Serial el valor de pwm deseado (0 a 255).

Para eso se utiliza la instrucción Serial.parseInt(). Esta instrucción es capaz de mandar cualquier dato tipo entero al Arduino omitiendo caracteres o signos. Finalmente para delimitar el rango de valores de 0 a 255 se hizo uso de la sentencia de if. Esta sentencia da la posibilidad que los únicos datos que se pueden mandar a Arduino es un rango entre 0 y 255.

**Resultados**

Después de haber cargado el programa, se procedió a conectar el motor a la fuente externa de alimentación y después se inició la comunicación serial.

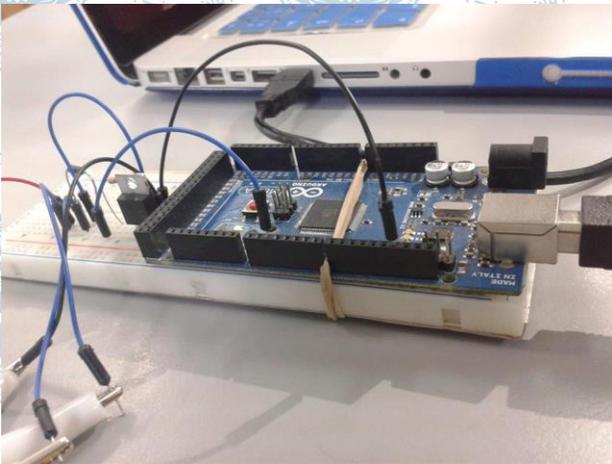


Figura 3 Conexión física de circuito

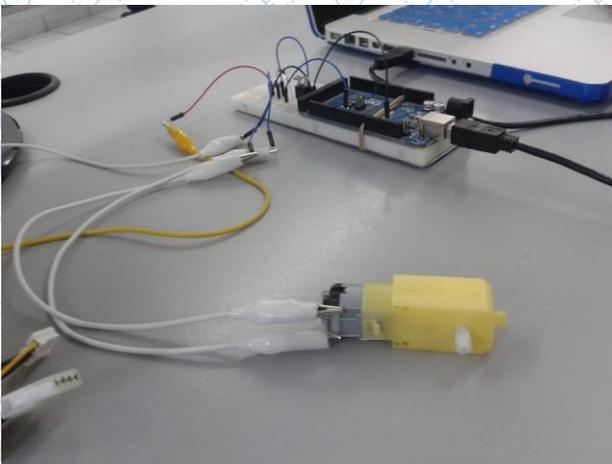


Figura 4 Motor conectado a Arduino

Dentro de la ventana de monitor serial de arduino, se mandaron datos numéricos que van desde 0 hasta 255, esto es debido a que la variable que nos permite hacer tal cosa es de tipo int y la instrucción Serial.parseInt() como se explicó anteriormente. Algo que hay que tomar en cuenta es que los valores de voltaje que recibirá el motor dependerán del dato a enviar, donde 0 es apagado y 255 es el voltaje nominal.

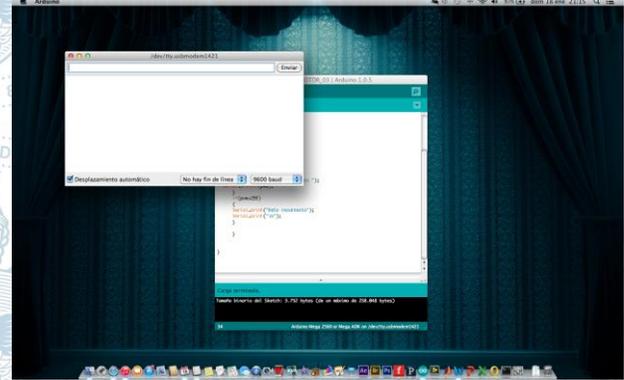


Figura 5 Vista completa de IDE y monitor serial

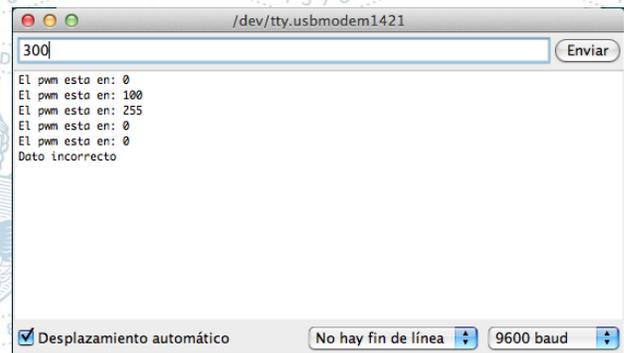


Figura 6 Envío de datos por monitor serial

### Conclusiones

Como práctica de introducción, podemos observar la capacidad y los beneficios que la tarjeta Arduino ofrece, facilitando los procesos de programación y optimizando recursos para así, proporcionar un control adecuado y simple al momento de llevar a cabo proyectos que requieran de dichos recursos.

### Bibliografía

- <http://es.wikipedia.org/wiki/Arduin>
- [http://es.wikipedia.org/wiki/Modulaci3n\\_por\\_ancho\\_de\\_pulsos](http://es.wikipedia.org/wiki/Modulaci3n_por_ancho_de_pulsos)
- [http://es.wikipedia.org/wiki/Motor\\_de\\_corriente\\_continua](http://es.wikipedia.org/wiki/Motor_de_corriente_continua)
- <http://bildr.org/2012/03/rfp30n061e-arduino/>

## Anexos

### Código Arduino:

```
int pwm=0;

void setup()
{
  Serial.begin(9600);
  pinMode(2,OUTPUT);
}

void loop()
{
  delay(1000);

  if (Serial.available())
  {
    pwm=Serial.parseInt();
    if(pwm<=255)
    {
      analogWrite (2,pwm);
      Serial.print("El pwm esta en: ");
      Serial.println(pwm);
    }
    if(pwm>255)
    {
      Serial.print("Dato incorrecto");
      Serial.print("\n");
    }
  }
}
```