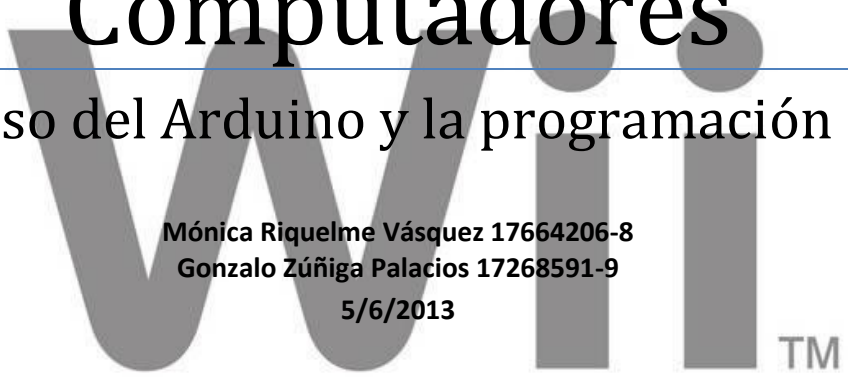
UNIVERSIDAD TECNOLÓGICA METROPOLITANA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INFORMÁTICA Y COMPUTACIÓN

Proyecto Arquitectura de Computadores

Uso del Arduino y la programación



Mónica Riquelme Vásquez 17664206-8
Gonzalo Zúñiga Palacios 17268591-9
5/6/2013

Informe del proyecto de Arquitectura de Computadores, profesora Karina Harispe Aguilar.
Primer semestre del año 2013.

Objetivos

- Explicar en qué consiste un Arduino, sus características de funcionamiento y estructura.
- Como implementar el control de velocidad de un motor en el Arduino.
- Definir las características de la programación en el Arduino.
- Especificar la programación hecha en Arduino para el Javabot.

Arduino

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 multidisciplinares.

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida.[4] Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, ATmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (*boot loader*) que corre en la placa.

Desde octubre de 2012, Arduino se usa también con microcontroladoras CortexM3 de ARM de 32 bits , que coexistirán con las más limitadas, pero también económicas AVR de 8 bits. ARM y AVR no son plataformas compatibles a nivel binario, pero se pueden programar con el mismo IDE de Arduino y hacerse programas que compilen sin cambios en las dos plataformas. Eso sí, las microcontroladoras CortexM3 usan 3.3V, a diferencia de la mayoría de las placas con AVR que usan mayoriamente 5V. Sin embargo ya anteriormente se lanzaron placas Arduino con Atmel AVR a 3.3V como la Arduino Fio y existen clónicos de Arduino Nano y Pro como Meduino en que se puede conmutar el voltaje.

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software del ordenador (por ejemplo: Macromedia Flash, Processing, Max/MSP, Pure Data). Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente.

Al ser open-hardware, tanto su diseño como su distribución es libre. Es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia.

El proyecto Arduino recibió una mención honorífica en la categoría de Comunidades Digital en el Prix Ars Electrónica de 2006.

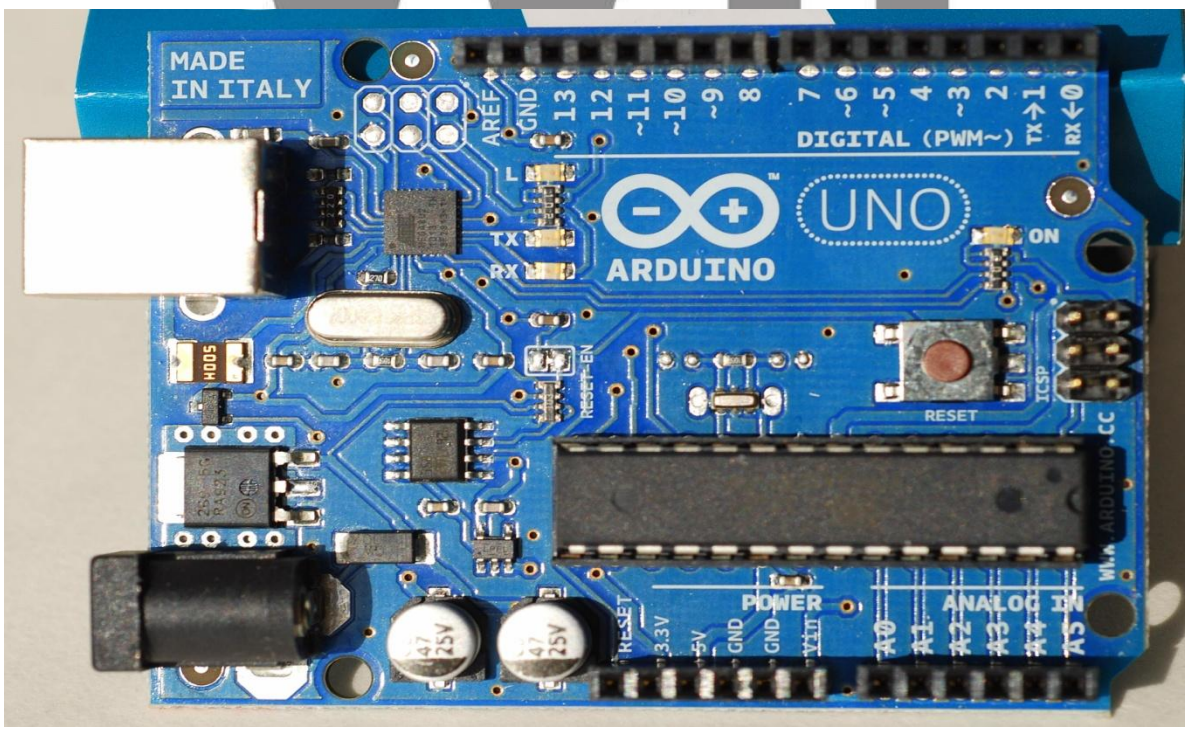
En nuestro caso utilizaremos el Arduino UNO ya que es que el tenemos a disposición y es el más básico de los hardwares.

Cuadro informativo

ITEM	Arduino Uno (retired)
uC	ATmega328
INPUT VOLTAGE	5-12V
SYSTEM VOLTAGE	5V
CLOCK SPEED	16MHz
DIGITAL I/O	14
ANALOG INPUTS	6
PWM	6
UART	1
FLASH SPACE	32Kb
BOOTLOADER	Optiboot
PROGRAMMING INTERFACE	USB via ATMega8U2

Esquema de pines: entradas y salidas

Arduino UNO, consta de 14 entradas digitales configurables entrada y/o salidas que operan a 5 voltios. Cada pin puede proporcionar o recibir como máximo 40 mA. Los pines 3, 5, 6, 9, 10 y 11 pueden proporcionar una salida PWM (Pulse Width Modulation). Si se conecta cualquier cosa a los pines 0 y 1, eso interferirá con la comunicación USB. UNO también tiene 6 entradas analógicas que proporcionan una resolución de 10 bits. Por defecto miden de 0 voltios (masa) hasta 5 voltios, aunque es posible cambiar el nivel más alto, utilizando el pin Aref y algún código de bajo nivel.



Fue a partir de el informe anterior que decidimos utilizar las salidas PWM ya que ofrecen un mejor control del ciclo de trabajo del motor con menos potencia; teniendo en cuenta que la otra opción habría sido variar los voltajes entre 0-12V.

La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de *pulse-width modulation*) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una 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.

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período. Expresado matemáticamente:

$$D = \frac{\tau}{T}$$

D es el ciclo de trabajo
τ es el tiempo en que la función es positiva (ancho del pulso)
T es el período de la función

La construcción típica de un circuito PWM se lleva a cabo mediante un comparador con dos entradas y una salida. Una de las entradas se conecta a un oscilador de onda dientes de sierra, mientras que la otra queda disponible para la señal moduladora. En la salida la frecuencia es generalmente igual a la de la señal dientes de sierra y el ciclo de trabajo está en función de la portadora.

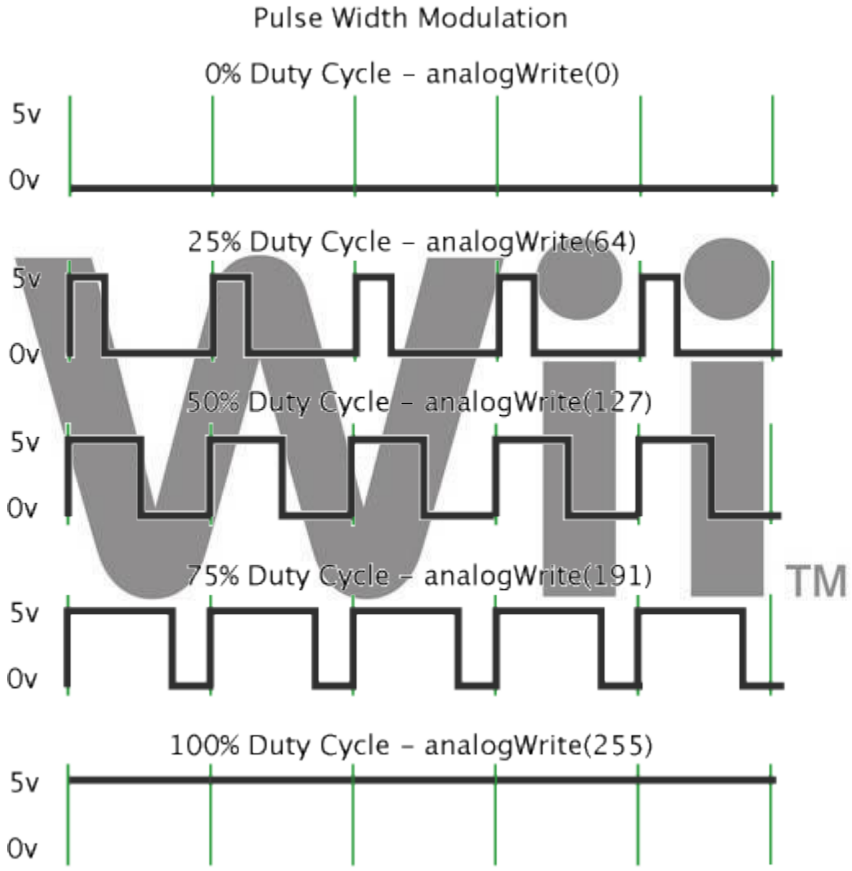
La modulación por ancho de pulsos es una técnica utilizada para regular la velocidad de giro de los motores eléctricos de inducción o asíncronos. Mantiene el par motor constante y no supone un desaprovechamiento de la energía eléctrica. Se utiliza tanto en corriente continua como en alterna, como su nombre lo indica, al controlar: un momento alto (encendido o alimentado) y un momento bajo (apagado o desconectado), controlado normalmente por relevadores (baja frecuencia) o MOSFET o tiristores (alta frecuencia).

Otros sistemas para regular la velocidad modifican la tensión eléctrica, con lo que disminuye el par motor; o interponen una resistencia eléctrica, con lo que se pierde energía en forma de calor en esta resistencia.Otra forma de regular el giro del motor es variando el tiempo entre pulsos de duración constante, lo que se llama modulación por frecuencia de pulsos.

La Modulación simula una salida analógica con una salida digital. El control digital se usa para crear una onda cuadrada, una señal que conmuta constantemente entre encendido y apagado. Este patrón de encendido-apagado puede simular voltajes entre 0 (siempre apagado) y 5 voltios (siempre encendido) simplemente variando la proporción de tiempo entre encendido y apagado. A la duración del tiempo de encendido (ON) se le llama Ancho de Pulso (pulse width). Para variar el valor analógico cambiamos, o modulamos, ese ancho de pulso. Si repetimos este patrón de encendido-apagado lo suficientemente rápido por

ejemplo con un LED el resultado es como si la señal variara entre 0 y 5 voltios controlando el brillo del LED.

En el grafico de abajo las líneas verdes representan un periodo regular. Esta duración o periodo es la inversa de la frecuencia del PWM. En otras palabras, con la Arduino la frecuencia PWM es bastante próxima a 500Hz lo que equivale a periodos de 2 milisegundos cada uno. La llamada a la función analogWrite() debe ser en la escala desde 0 a 255, siendo 255 el 100% de ciclo (siempre encendido), el valor 127 será el 50% del ciclo (la mitad del tiempo encendido), etc.



Una vez cargado y ejecutado el ejemplo mueve la arduino de un lado a otro, lo que ves es esencialmente un mapeado del tiempo a lo largo del espacio. A nuestros ojos el movimiento difumina cada parpadeo del LED en una linea. A medida que la luminosidad del LED se incrementa o atenua esas pequeñas lineas crecen o se reducen. Ahora estas viendo el ancho de pulso (pulse width).

analogWrite()

Descripción: Escribe un valor analógico (PWM) en un pin. Puede ser usado para controlar la luminosidad de un LED o la velocidad de un motor. Después de llamar a la función analogWrite(), el pin generará una onda cuadrada estable con el ciclo de trabajo especificado hasta que se vuelva a llamar a la función analogWrite() (o una llamada a las funciones digitalRead() o digitalWrite() en el mismo pin). La frecuencia de la señal PWM sera de aproximadamente 490 Hz.

En la mayoría de las placas Arduino (aquellas con el ATmega168 o ATmega328), se podrá generar señales PWM en los pines 3, 5, 6, 9, 10, y 11. No hace faltar configurar el pin como salida para poder usar la función analogWrite(). La función *analogWrite* no tienen ninguna relación con los pines de entrada analógicos ni con la función *analogRead*.

Sintaxis: analogWrite(pin, valor)

Parámetros:

- Pin: Es el pin en el cual se quiere generar la señal PWM.
- Valor: El ciclo de trabajo deseado comprendido entre 0 (siempre apagado) y 255 (siempre encendido).

Devuelve: Nada

Notas y problemas conocidos: Las señales PWM generadas en los pines 5 y 6 poseerán ciclos de trabajo superiores a lo esperado. Esto es así por que para esos dos pines se utiliza el mismo temporizador que se utiliza en las funciones millis() y delay(). Este efecto se notará mucho más en ciclos de trabajo bajos (por ejemplo de 0 a 10) y puede ser que aunque configuremos esos pines con una señal de ciclo de trabajo cero no llegue a ser verdaderamente 0.

Ejemplo: Produce una señal donde conectamos el LED, cuyo ciclo de trabajo es proporcional a la tensión leída en el potenciómetro.

```
int ledPin = 9;       // LED conectado al pin digital 9
int analogPin = 3;   // potenciómetro conectado al pin 3
int val = 0;          // variable en el que se almacena el dato leído

void setup()
{
  pinMode(ledPin, OUTPUT);   // sets the pin as output
}
```

```
void loop()
{
  val = analogRead(analogPin);   // lee la tensión en el pin
  analogWrite(ledPin, val / 4);  // los valores de analogRead van desde 0 a 1023 y los
                                    valores de analogWrite values van
                                    desde 0 a 255, por eso ajustamos el ciclo de trabajo a el
                                    valor leído dividido por 4.
}
```

Programación

A continuación se muestra la programación hecha al Arduino para controlar el motor del Javabot

```
#include <math.h>

Void setup()

Int MotorI = 9;      // motor Izquierdo conectado al pin digital 9

Int MotorD = 10;     // motor derecho conectado al pin digital 10

Int dir1_I = 2;

Int dir2_I = 3;

Int dir1_D = 4;

Int dir2_D = 5;

Void loop()

analogWrite(MotorI, 255);

analogWrite(MotorD, 255);     // valor 0-255 del cual depende de la velocidad a la que

                                    girará el motor
```

Hitos del Proyecto

Hitos	Avance Porcentual	Estado Actual
Descripción del proyecto y sus hitos	100%	Completado
Estudio del Motor cc y driver l293b	100%	Completado
Uso del Arduino y la programación	100%	Completado
Conexión Arduino-Nunchuk	20%	En curso
Implementación y corrección	0%	Pendiente
Informe Final	50%	En curso

Bibliografía

http://www.arduino.cc/es/

http://es.wikipedia.org/wiki/Arduino