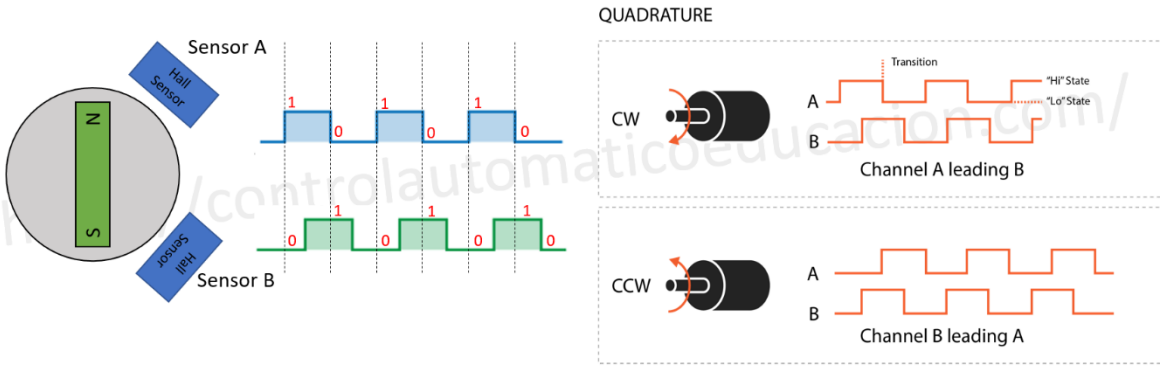
COMO FUNCIONA UN ENCODER DE CUADRATURA DE EFECTO HALL

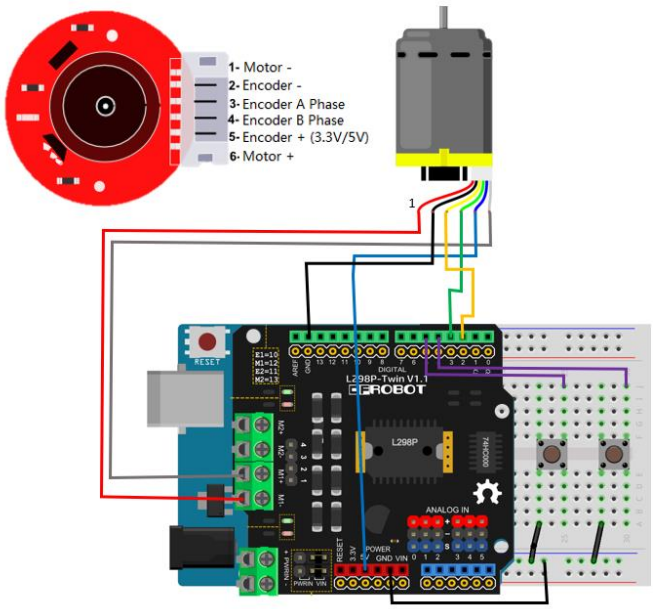
Un Encoder (codificador) funciona a través de la detección de los cambios en el campo magnético creado por un imán conectado al eje del motor. A medida que el motor gira, las salidas del Encoder se dispararán periódicamente.

Generalmente contamos con 2 sensores que transforman el giro del motor (transductor) a 2 señales cuadradas que presentan un desfase de 90°. Gracias a este desfase, a estos codificadores se les denomina como encoders en cuadratura, ocupando un cuadrante del círculo de 360°



Cuando el imán gira en el sentido de las agujas del reloj, la salida del sensor A se activará primero. Cuando se gira en sentido antihorario, por otro lado, la salida del sensor B se activará primero

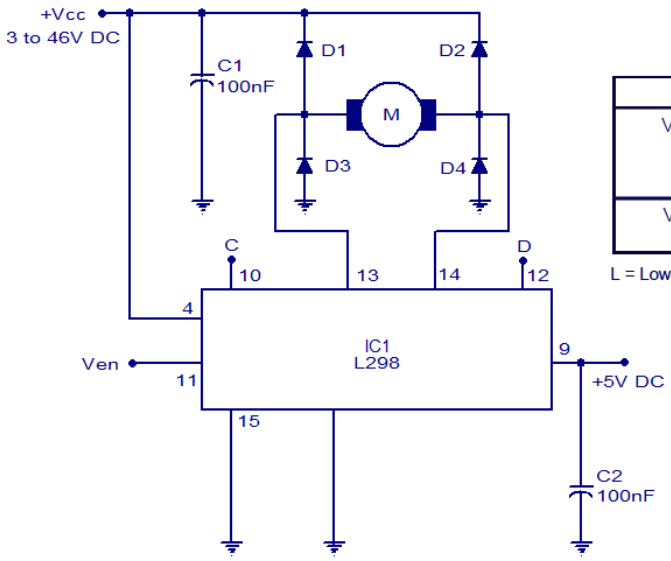
Este efecto lo podemos ver fácilmente en nuestro Arduino, para eso implementamos el siguiente circuito.



En el circuito anterior se está empleando un módulo Shield L298, se puede usar cualquier otro módulo o montar directamente el circuito.

El L298 es un controlador de puente completo dual que tiene un amplio rango de voltaje operativo y puede manejar corrientes de carga de hasta 3A. El IC también cuenta con voltaje de saturación bajo y protección contra sobre temperatura.

En el circuito, el diodo D1 a D4 son diodos de protección. El condensador C2 es el filtro de fuente de alimentación lógica y el condensador C2 es el filtro de tensión de alimentación. El estado del motor dependerá del nivel lógico de los pines 10, 11, 12 y se describe en la tabla que se muestra debajo del esquema del circuito.



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

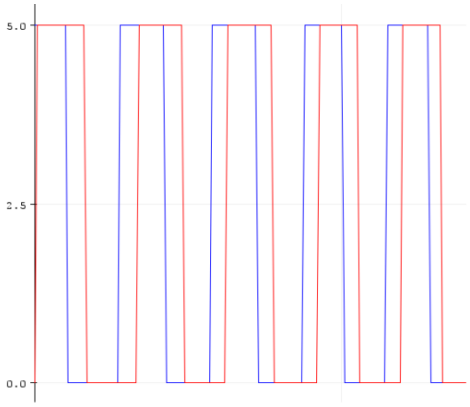
El código en Arduino es:

```
#define ENCODER_A       2 // Amarillo
#define ENCODER_B       3 // Verde
#define BUTTON_FORWARD  4
#define BUTTON_BACKWARD 5

// Pines de Control Shield
const int E1Pin = 10;
const int M1Pin = 12;
const int E2Pin = 11;
const int M2Pin = 13;

typedef struct
{
  byte enPin;
  byte directionPin;
}
Motor;//Creo el motor

const Motor motor = {E1Pin, M1Pin};
```

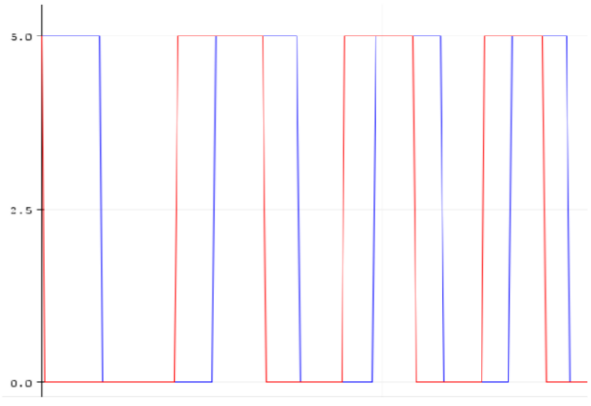
```
const int Forward = LOW;
const int Backward = HIGH;

void setup()
{
  Serial.begin(9600);

  //Encoders como entradas
  pinMode(ENCODER_A, INPUT);
  pinMode(ENCODER_B, INPUT);

  //Pulsadores
  pinMode(BUTTON_FORWARD, INPUT_PULLUP);
  pinMode(BUTTON_BACKWARD, INPUT_PULLUP);

  //Configura Motor
  pinMode(motor.enPin, OUTPUT);
  pinMode(motor.directionPin, OUTPUT);
}

void loop()
{
  // Pulsador hacia adelante
  if(! digitalRead(BUTTON_FORWARD))
  {
    digitalWrite(motor.directionPin, Forward);
    digitalWrite(motor.enPin, HIGH);
    imprimir_cuadratura();
  }
  else if(! digitalRead(BUTTON_BACKWARD))
  {
    digitalWrite(motor.directionPin, Backward);
    digitalWrite(motor.enPin, HIGH);
    imprimir_cuadratura();
  }
  else
  {
    digitalWrite(motor.enPin, LOW);
  }
}
void imprimir_cuadratura()
{
  int a = digitalRead(ENCODER_A);
  int b = digitalRead(ENCODER_B);
  Serial.print(a*5);
```

```
  Serial.print(" ");
  Serial.println(b*5);
}
```

La señal de cuadratura observada por el serial plotter:



BACKWARD

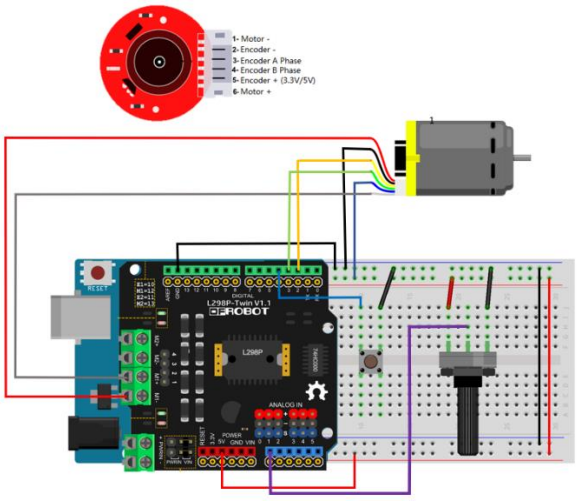


FORWARD

Medir la velocidad y posición motor DC

El hecho de disponer el encoder acoplado al rotor del motor de corriente continua, nos brinda la posibilidad de poder medir su posición y velocidad para ser empleado en muchas aplicaciones de robótica.

Vamos a modificar levemente el circuito presentado anteriormente, donde solo dejaremos un solo pulsador y agregaremos un potenciómetro.



De esta manera con el pulsador podemos seleccionar el modo de operación: Modo de regulación de velocidad, o modo de posición. Ambos controlados por la lectura ADC con Arduino del Potenciómetro.

Medición de posición

Para la lectura de la posición, vamos a tener que recurrir a las interrupciones con Arduino, donde escogeremos el Sensor 1 (A) para que active la interrupción con el flanco de subida. En la interrupción se incrementa una variable que llamamos theta (si la salida del encoder B es alta) o se resta uno (si la salida del encoder B es baja).

Para garantizar que la variable theta se almacene de modo que pueda ser leída con precisión por las funciones de loop y de interrupción, debe utilizar el calificador volátil.

Además, se necesita una macro ATOMIC_BLOCK para acceder a la variable de posición. La macro ATOMIC_BLOCK evita que la interrupción cambie parte de la variable theta mientras se está leyendo.

Finalmente se establece una velocidad fija en el motor DC, con un PWM de 200, y se pregunta por la lectura ADC del potenciómetro para poder ejercer un control aproximado de la posición del motor. Este control se logra con dos condicionales donde mantenemos una tolerancia de más o menos 2 grados, con el fin que el rotor pueda detenerse en ese pequeño umbral o threshold.

Medición de velocidad

La medición de velocidad es muy similar a la de posición, solo que, en este caso, se debe configurar el arduino para que quede capturando los datos de cualquiera de los dos sensores del encoder (A o B) por un periodo de tiempo que sea conocido, para realizar posteriormente el cálculo de la velocidad.

Como ya se está leyendo la interrupción con el sensor 1 (A) lo que hacemos es incrementar una variable entera volátil llamada pulsos.

Dentro del void loop, haremos uso de la función millis() del Arduino, para contabilizar de una forma precisa 1 segundo.

Una vez ha transcurrido 1 segundo, entramos en el condicional y dentro de la macro ATOMIC calculamos las RPM del motor con base a 1 segundo, empleando la resolución del encoder (Numero de pulsos por giro) en este caso es de 374.22. Actualizamos la variable timeold para que millis() vuelva y contabilice nuevamente 1 segundo y cerramos el contador pulsos para realizar una nueva medición.

Código de Arduino

El siguiente será el código que vamos a implementar para esta práctica y el cual nos será muy útil para otros proyectos que haremos posteriormente en el sitio web, por lo tanto, es importante que lo entiendan a la perfección.

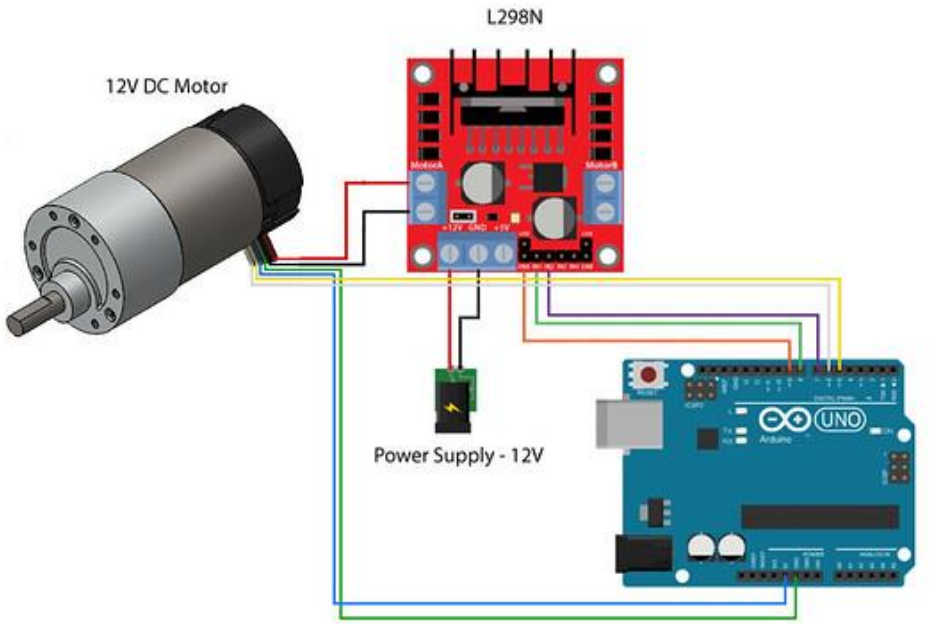
```
#include <util/atomic.h>
#define ENCODER_A       2 // Amarillo
#define ENCODER_B       3 // Verde
#define BUTTON_MOD      4

// Pin del Potenciómetro
const int pot = A0;

// Pines de Control Shield
const int E1Pin = 10;
const int M1Pin = 12;
const int E2Pin = 11;
const int M2Pin = 13;
```

```
//Variable global de posición compartida con la interrupción
volatile int theta = 0;

//Variable global de pulsos compartida con la interrupción
volatile int pulsos = 0;
unsigned long timeold;
float resolution = 374.22;

//Variable Global Velocidad
int vel = 0;

//Variable Global Posicion
int ang = 0;

//Variable Global MODO
bool modo = false;

//Estructura del Motor
typedef struct
{
  byte enPin;
  byte directionPin;
}
Motor;

//Creo el motor
const Motor motor = {E1Pin, M1Pin};

//Constantes de dirección del Motor
const int Forward = LOW;
const int Backward = HIGH;
void setup()
{
  // set timer 1 divisor to  1024 for PWM frequency of 30.64 Hz
  TCCR1B = TCCR1B & B11111000 | B00000101;
  Serial.begin(9600);

  //Encoders como entradas
  pinMode(ENCODER_A, INPUT);
  pinMode(ENCODER_B, INPUT);

  //Pulsadores
  pinMode(BUTTON_MOD, INPUT_PULLUP);
```

```
  //Configura Motor
  pinMode(motor.enPin, OUTPUT);
  pinMode(motor.directionPin, OUTPUT);

  //Configurar Interrupción
  timeold = 0;
  attachInterrupt(digitalPinToInterrupt(ENCODER_A),leerEncoder,RISING);
}

void loop()
{
  float posicion;
  float rpm;
  int value,dir=true;

  //Lee el Valore del Potenciometro
  value = analogRead(pot);

  //Cambia de Modo Velociadad o Posición
  if(debounce(BUTTON_MOD))
  {
    modo = !modo;
    theta = 0;
  }
  if(modo)
  {
    //Transforma el valor del Pot a velocidad
    vel = map(value,0,1023,0,255);

    //Activa el motor dirección Forward con la velocidad
    setMotor(motor, vel, false);

    //Espera un segundo para el calculo de las RPM
    if (millis() - timeold >= 1000)
    {
      //Modifica las variables de la interrupción forma atómica
      ATOMIC_BLOCK(ATOMIC_RESTORESTATE)
      {
        //rpm = float(pulsos * 60.0 / 374.22); //RPM
        rpm = float((60.0 * 1000.0 / resolution ) / (millis() - timeold) *
pulsos);
        timeold = millis();
        pulsos = 0;
      }
      Serial.print("RPM: ");
```

```
      Serial.println(rpm);
      Serial.print("PWM: ");
      Serial.println(vel);
    }
  }

  else
  {
    //Transforma el valor del Pot a ángulo
    ang = map(value,0,1023,0,360);

    //Modifica las variables de la interrupción forma atómica
    ATOMIC_BLOCK(ATOMIC_RESTORESTATE)
    {
      posicion = (float(theta * 360.0 /resolution));
    }

    //Posiciona el ángulo con tolerancia +- 2
    if(ang > posicion+2)
    {
      vel = 200;
      dir = true;
    }
    else if(ang < posicion-2)
    {
      vel = 200;
      dir = false;
    }
    else
    {
      vel = 0;
    }
    setMotor(motor, vel, dir);
  }
}

//Función para dirección y velocidad del Motor
void setMotor(const Motor motor, int vel, bool dir)
{
  analogWrite(motor.enPin, vel);
  if(dir)
    digitalWrite(motor.directionPin, Forward);
  else
    digitalWrite(motor.directionPin, Backward);
}
```

```
//Función anti-rebote
bool debounce(byte input)
{
  bool state = false;
  if(! digitalRead(input))
  {
    delay(200);
    while(! digitalRead(input));
    delay(200);
    state = true;
  }
  return state;
}

//Función para la lectura del encoder
void leerEncoder()
{
  //Lectura de Velocidad
  if(modo)
    pulsos++; //Incrementa una revolución

  //Lectura de Posición
  else
  {
    int b = digitalRead(ENCODER_B);
    if(b > 0)
    {
      //Incremento variable global
      theta++;
    }
    else
    {
      //Decremento variable global
      theta--;
    }
  }
}
```


L298N
12V DC Motor
Power Supply - 12V