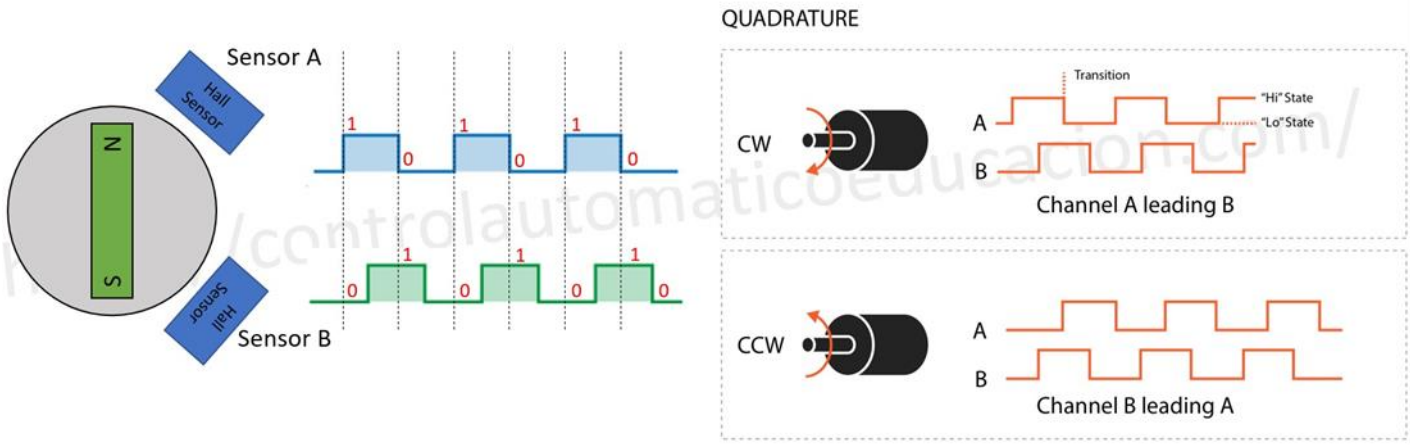
Actividad 3ra Semana

Docentes: Ing. Jorge E. Morales, Téc. Sup. Mecatrónica Gonzalo Vera

Grupo: 8

Actividad 2. A:

"Implemente un control de velocidad, posición y sentido de giro utilizando un motor con encoder incremental, el control del motor se debe realizar con pwm.

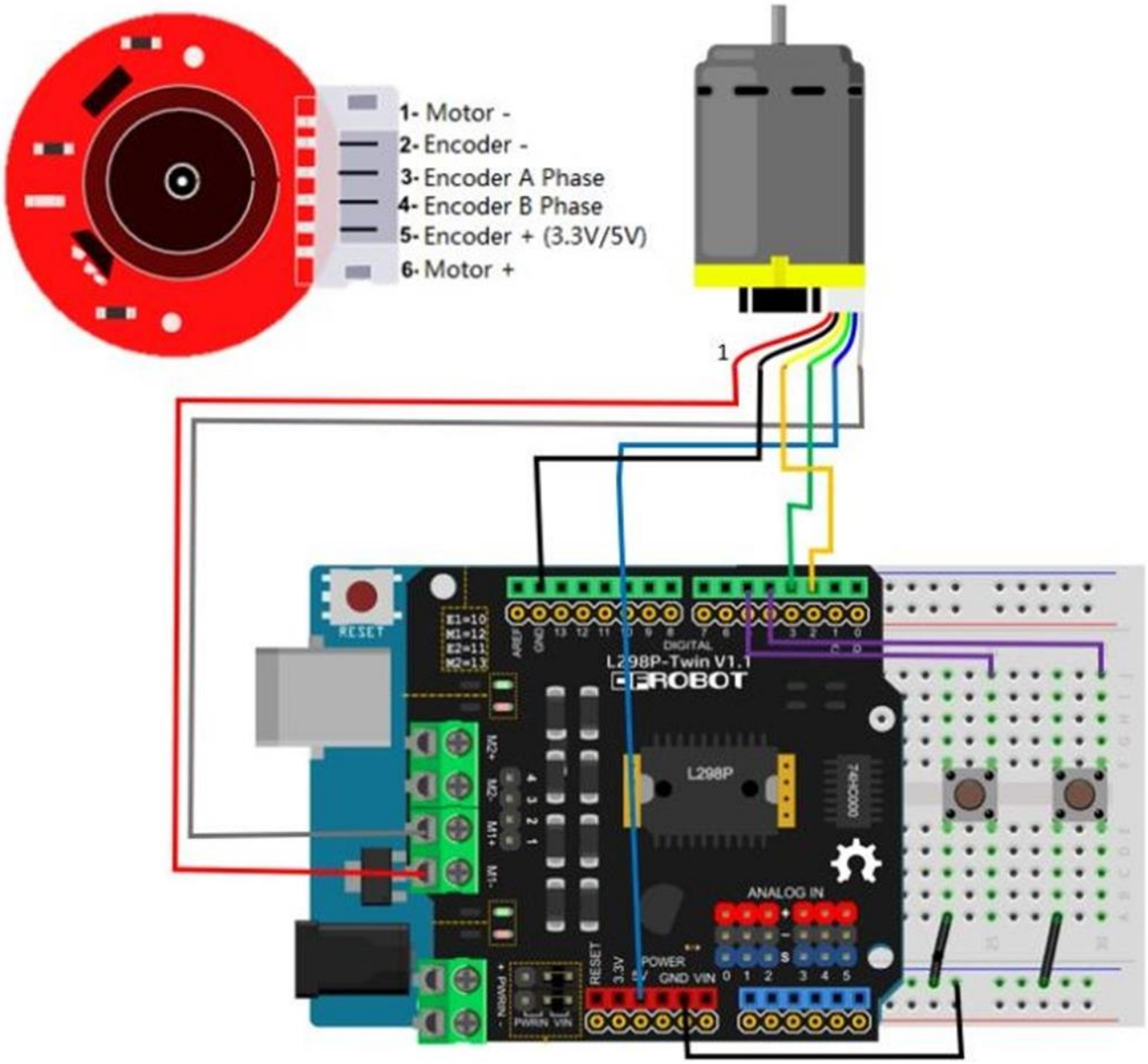


Un Encoder (codificador) funciona a través de la detección de los cambios en el campo magnético creado por un imán conectado al eje del motor. A medida que el motor gira, las salidas del Encoder se dispararán periódicamente.

Generalmente contamos con 2 sensores que transforman el giro del motor (trasductor) 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

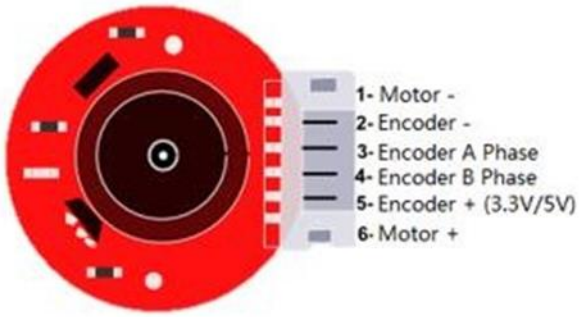
Este efecto lo podemos ver facilmente en nuestro Arduino, para eso implementamos el siguiente circuito.

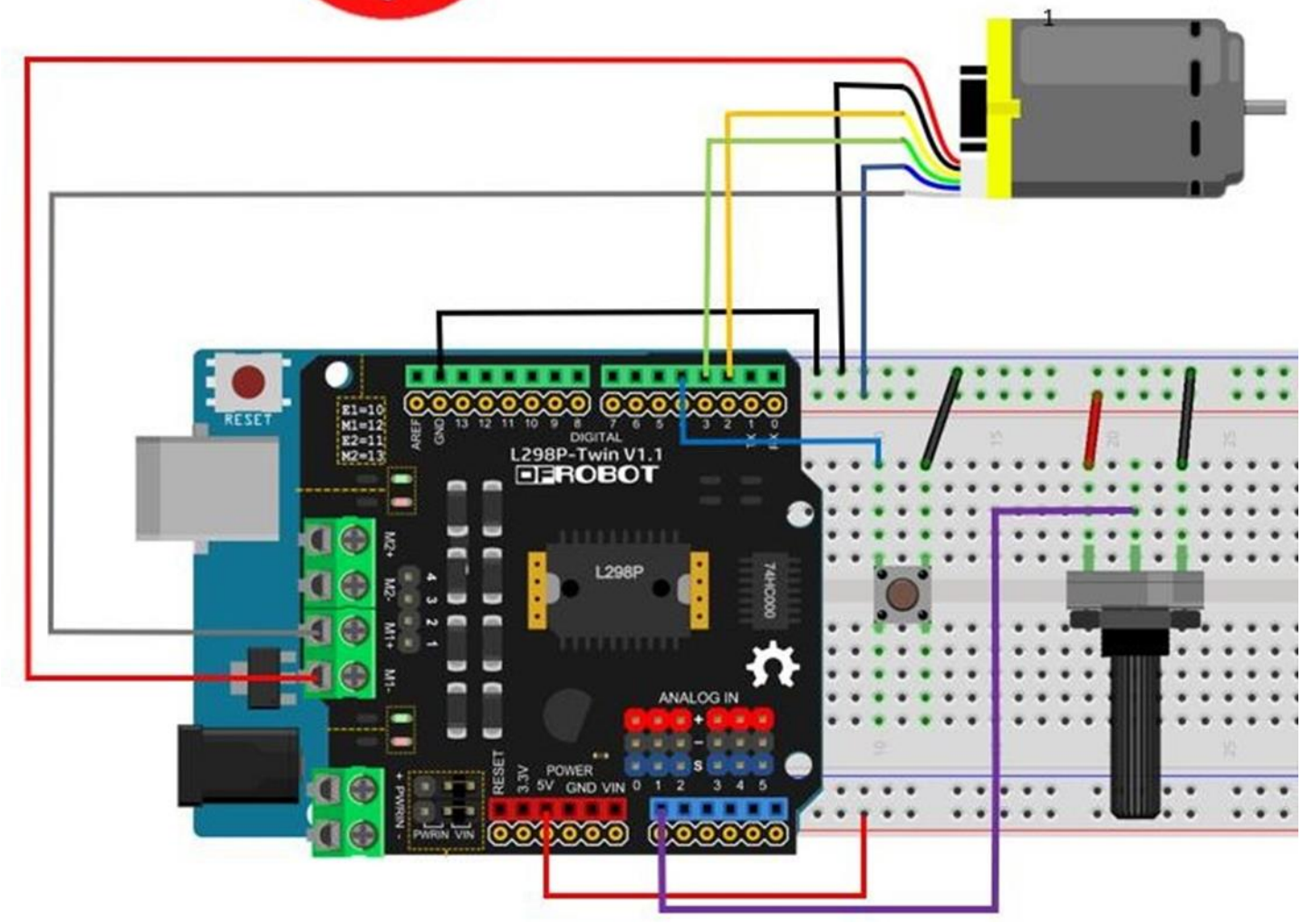


Medir la velocidad y posición motor DC

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.

Para esta práctica, vamos a modificar levemente el circuito presentado anteriormente, donde solo dejaremos un solo pulsador y agregaremos un potenciometro.





De esta manera con el pulsador podemos seleccionar el modo de operación: Modo de regulación de velocidad, o modo de posición.

```
#include <util/atomic.h>

#define ENCODER_A      2 // Amarillo
#define ENCODER_B      3 // Verde
#define BUTTON_MOD     4


// Pin del Potenciometro
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--;
    }
  }
}
```