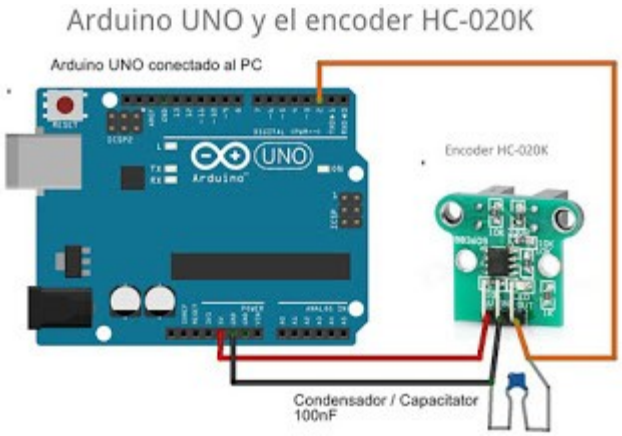
Encoder para robot. Tutorial sobre el encoder fotoeléctrico HC-020K, usado en un robot Arduino.

En este tutorial avanzado de Arduino se enseña como se debe conectar el encoder de barrera fotoeléctrico HC-020K a Arduino para medir las Revoluciones Por Minuto (RPM) y la velocidad de las ruedas de un robot. También se le puede llamar optointerruptor. Se enseña un truco para que el encoder funcione correctamente con Arduino UNO o MEGA. Al final de esta entrada hay dos pequeños programas de Arduino para controlar este encoder. Este encoder es muy económico y se vende para ser montado en los motores de los robots. Yo lo voy a montar en las 4 ruedas del robot Andromina OFF ROAD.

Encoder fotoeléctrico HC-020K

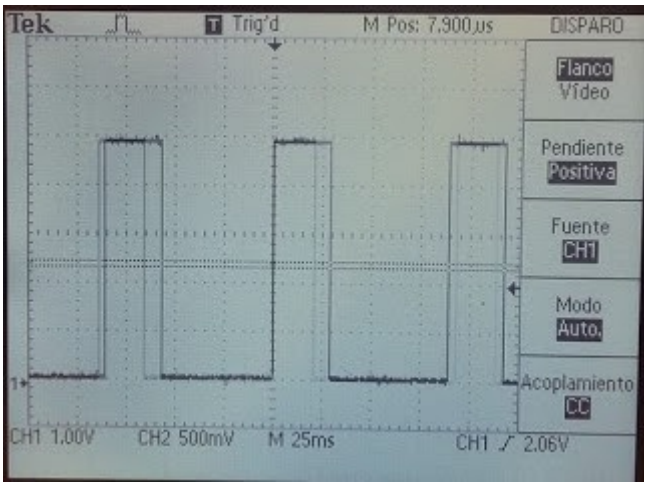
En Internet hay poca información sobre este encoder. También hay mucha desinformación. Lo que ha provocado que no me resultase nada fácil usar este encoder al conectarlo al Arduino UNO o MEGA. En Internet hay más información sobre este encoder pero por mi poca experiencia en encoders diría que no es del todo correcta esta información. Ya que muchas páginas web indican que este encoder es de dos canales A y B. Y yo solo veo un canal en este encode. Si usted tiene experiencia en encoders, agradecería sus opiniones.

En la foto siguiente se aprecia el esquema eléctrico realizado en este tutorial, para que el encoder y Arduino funcionen correctamente.



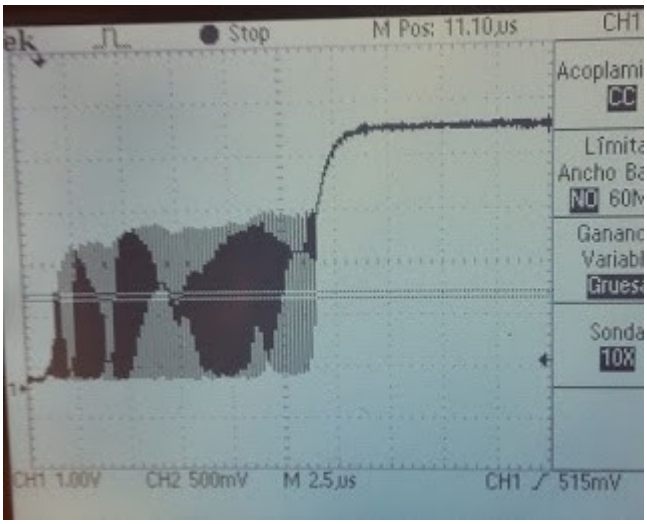
Esquema del encoder HC-020K, con un condesador y Arduino UNO.

1-Pulsos malos: Si conectamos directamente el encoder HC-020K a Arduino, tal y como nos indican sus proveedores. En seguida nos damos cuenta que Arduino lee más pulsos de los que realmente se generan. Arduino nos indica una velocidad de rotación mas alta de la que realmente hay en el motor. Este problema es difícil de solventar. Ya que si conectamos la señal del encoder a un osciloscopio nos muestra unos pulsos como los de la foto siguiente:



Vista de los pulsos digitales del encoder HC-020K

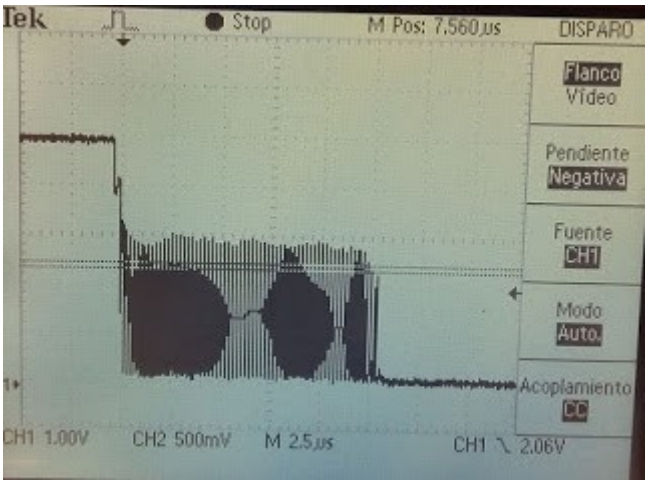
En la foto anterior parece que los pulsos que nos envía el encoder son totalmente correctos. Ya que parecen pulsos totalmente digitales. Se asemeja realmente a una señal TTL digital de 0 a 5 Voltios. Pero si ampliamos mucho la rampa inicial de subida de estos pulsos, nos damos cuenta que en el inicio del pulso existen gran cantidad de rebotes. Esto se puede ver en la foto siguiente:



Vista de los rebotes de la rampa de subida inicial de un pulso.

Para más información sobre rebotes en señales eléctricas leer este **enlace**
Ahora nos hemos dado cuenta que tenemos una señal TTL del encoder muy mala. Por esta razón Arduino lee más pulsos de los que realmente se generan. Ya que Arduino lee el pulso generado y algunos rebotes. Si volvemos a analizar otra vez el mismo pulso. Esta vez en la rampa de descenso, aquí también nos damos cuenta que existen rebotes al final

del pulso. En este punto Arduino también lee más pulsos, generados por los rebotes de la señal. Ver foto siguiente:



Vista de los rebotes de la rampa de baja final de un pulso.

2-Soluciones : Para solucionar este problema tenemos dos opciones, eliminar los rebotes por "Hardware" o por "Software".
Para eliminar los rebotes por "Hardware" tenemos que colocar un condensador que absorba los rebotes iniciales y finales de los pulsos. En el esquema inicial que hay más arriba se muestra como se tiene que conectar el encoder y dicho condensador.
Para eliminar los rebotes por "Software" tenemos que diseñar un "Sketch" de Arduino que no tenga en cuenta los rebotes iniciales y finales y los falsos pulsos. En este tutorial solo se a implementado la solución del condensador.
3-Montaje : En la foto siguiente se muestra el montaje realizado para este tutorial. Inicialmente solo hemos usado un solo encoder HC-020K, un micro motor CC con una reducción de i = 120:i, una placa de prototipo, una fuente de alimentación CC de 3 Voltios (Para alimentar el motor) y un Arduino UNO o MEGA.

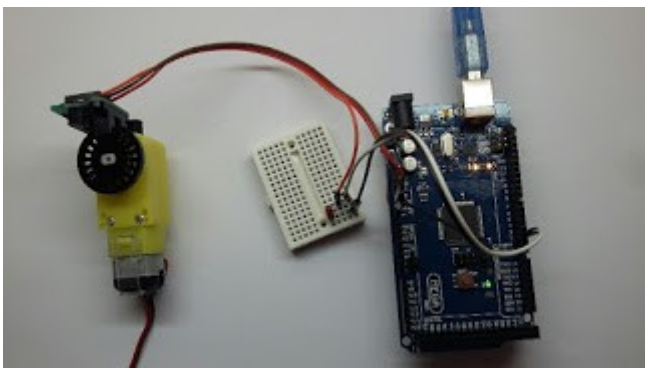
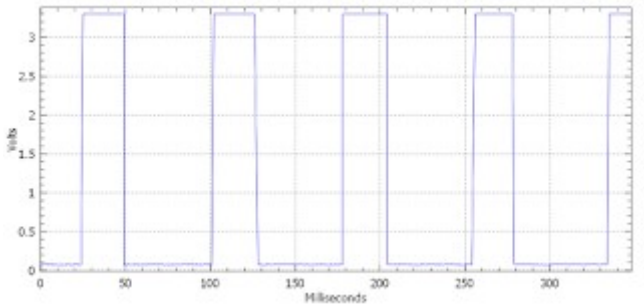


Foto del montaje realizado con el encoder y el Arduino.

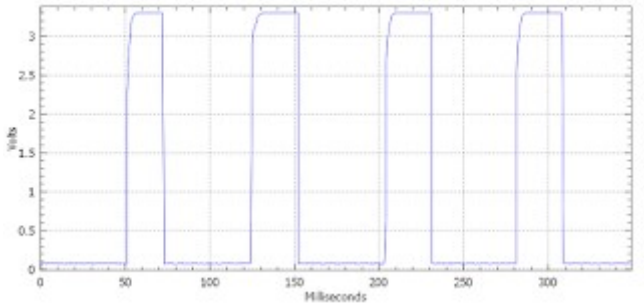
4- El condensador : Para que el encoder funcione correctamente con el Arduino UNO o MEGA se tiene que añadir un condensador de poliéster metalizado de 100nF (104) entre la señal de salida del encoder OUT y GND. Tal como se muestra en el esquema superior y en la foto anterior. He probado con diferentes tamaños de condensador y también funcionan bien. También se podría usar un condensador cerámico, el cual es más económico y pequeño. Yo uso este tipo de condensador por que es el que tengo ahora a mano.

A continuación se puede ver las señales que tenemos en la salida OUT del encoder. Donde se puede ver que la señal del encoder es una onda totalmente cuadrada, aparentemente.



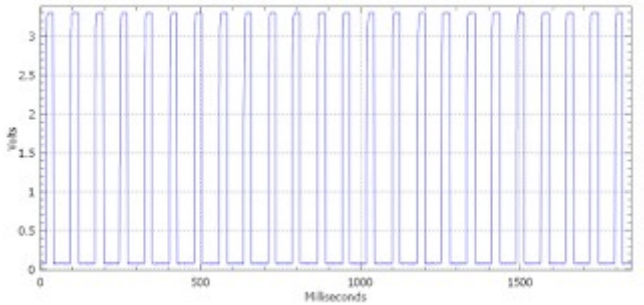
Señal de salida del encoder HC-020K a 36 RPM.

A continuación se puede ver las señales que tenemos en la salida OUT del encoder, usando un condenador cerámico de 100 nFaradios. En este ejemplo hay 36 RPM. Se puede ver que la señal esta un poco deformada, en la rampa de subida. Debido a la carga del condensador. Pero Arduino interpreta perfectamente esta señal. Si se coloca un condensador más pequeño esta deformación es más pequeña y casi no se percibe.



Señal de salida del encoder HC-020K a 36 RPM con condensador.

A continuación se muestran más gráficos con diferentes RPM del motor.



Señal de salida del encoder HC-020K a 36 RPM.

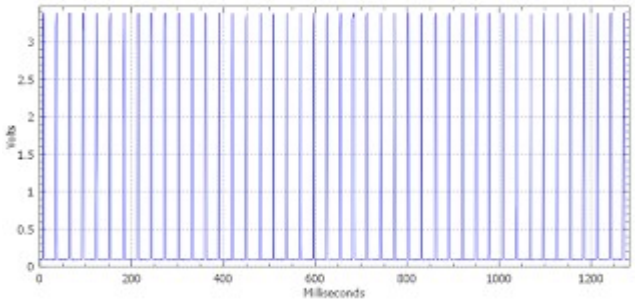
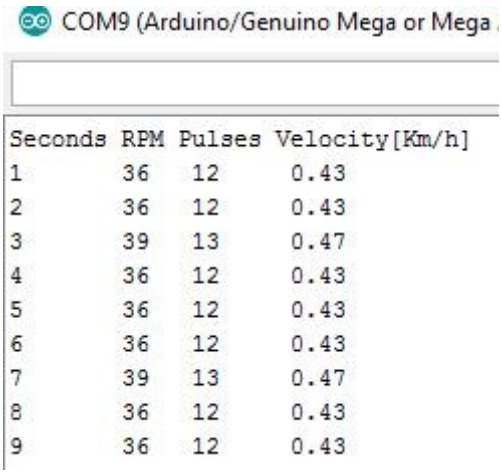


Gráfico de los pulsos del encoder HC-020K a 102 RPM.

5- Los datos técnicos : Las principales características técnicas a tener en cuenta del encoder HC-020K son:

1. Voltaje de alimentación del encoder entre 4,5 y 5,5 voltios. En este tutorial se usan los 5 voltios.
2. Frecuencia máxima de medición de $100KHZ$, es decir una alta definición a gran velocidad.
3. Resolución de 0,01mm.
4. El disco del encoder tiene 20 muescas.
5. Tamaño del encoder de 2 $cm\ x$ 2 cm ideal para cualquier aplicación.
6. anchura de paso del disco 6 mm.

6- El cálculo : En la foto siguiente se puede ver la salida del puerto serie del IDE de Arduino. Donde en la primera columna se indican los segundos, en la segunda las Revoluciones Por Minuto, en la tercera los impulsos leídos en un segundo y en la cuarta la velocidad de la rueda del robot:



Datos que calcula el Arduino con la señal del encoder.

En el Sketch de Arduino (ver más a bajo) se usa la siguiente formula para calcular las Revoluciones Por Minuto del motor:

$$N = \frac{n*60}{t*S}$$

Fórmula usada para calcular las revoluciones por minuto.

N ; Es la velocidad de rotación, Revoluciones Por Minuto (RPM).

n ; Es el número de pulsos generados. En nuestro caso pulsos en 1 segundo, 1000 mili segundos.

t ; Es el tiempo de generación de los pulsos. En nuestro caso 1 segundo, 1000 mili segundos.

S ; Es la sensibilidad del disco del encoder, Pulsos Por Revolución. En nuestro caso 20 pulsos revolución.

7- El sketch : En el "Sketch" usamos una interrupción de Arduino, en modo FALLING, esto quiere decir que Arduino solo detecta todas las pendientes de bajada de los impulsos.

Para el cálculo de la fórmula de N [RPM] se tiene que ir con mucho cuidado en como se pone el orden de operar de las variables. Ya que durante las operaciones de multiplicación y división que se realizan, en la formula, no podemos sobrepasar el rango de la variable N. Esto puede provocar que los valores calculados sean totalmente erróneos. También debemos de vigilar que no salgan número con decimales en la operación. Esto puede provocar que el valor calculado no sea muy preciso, al eliminar los decimales, ya que las variables están definidas como números enteros y no pueden trabajar en tracciones.

```
/* Programa que cuenta los pulsos y las revoluciones del encoder
foto eléctrico HC-020K con un Arduino MEGA O UNO.
 *////
Variables ////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
//
int encoder_pin = 2;    // Pin 2, donde se conecta el encoder

unsigned int N;         // [RPM] Revoluciones por minuto
calculadas.
volatile byte n;        //  [pulses]Número de pulsos leídos por el
Arduino
unsigned long timeold;  // Tiempo almacenado
unsigned int S = 20;    // Número de muescas que tiene el disco
del encoder.
const int wheel_diameter = 64;   // Diámetro de la rueda
pequeña[mm]
float velocity = 0;              // Velocity // Velocidad en
[Km/h]
//////////////////////////// Función que cuenta los pulsos del
encoder ////////////////////////////////////////////////////
 void counter(){
```

```
      n++;     // Incrementa los impulsos
 }
//// Configuración del
Arduino ////////////////////////////////////////////////////
/////////////////////////////////////////
void setup(){
   Serial.begin(9600);           // Configuración del puerto serie
   pinMode(encoder_pin, INPUT);  // Configuración del pin nº2
   attachInterrupt(0, counter, FALLING); // Configuración de la
interrupción 0, donde esta conectado el encoder HC-020K. FALLING =
la interrupción actua cuando la señal del pin cae : pasa de HIGH a
LOW.
// Inicialización de los parametros
   n = 0;
   N = 0;
   timeold = 0;
   Serial.println("");
   Serial.print("           Wheel  ");Serial.print("Wheel
 ");Serial.println("Wheel");
   Serial.print(" Seconds   ");Serial.print("RPM
 ");Serial.print("Pulses   "); Serial.println("Velocity [Km/h]");
 }
//// Programa
principal ////////////////////////////////////////////////////
/////////////////////////////////////////////////////
 void loop(){
   if (millis() - timeold >= 1000){   //  Se actualiza cada
segundo, [t] es el tiempo de muestreo igual a 1000 milisegundos, 1
segundo.
    detachInterrupt(0);               // Desconectamos la
interrupción para que no actué en esta parte del programa.
    N = ( n * 60 ) / (( millis() - timeold )/ 1000 * S );   //  N
= n x 60 / (t x S) Calculamos las revoluciones por minuto [RPM].
El valor 1000 es para pasar los milisegundos a segundos
    velocity = N * 3.1416 * wheel_diameter * 60 / 1000000;  //
Cálculo de la velocidad en [Km/h]
    Serial.print("   ");
    Serial.print(millis()/1000);Serial.print("       ");
    Serial.print(N,DEC);Serial.print("      ");
    Serial.print(n,DEC);Serial.print("      ");
    Serial.println(velocity,2);
    n = 0;               // Inicializamos los pulsos.
    timeold = millis();  // Almacenamos el tiempo actual.
    attachInterrupt(0, counter, FALLING); // Reiniciamos la
interrupción
   } }
//// Final del programa principal ////
```

8-El entorno gráfico de Arduino : La versión IDE 1.6.8 de Arduino ya lleva incorporado una opción (en herramientas\serial plotter\ ) que nos permite ver la señal (voltaje) procedente del Arduino. A continuación se muestra el gráfico que nos permite ver el IDE de Arduino si cargamos el programa siguiente.

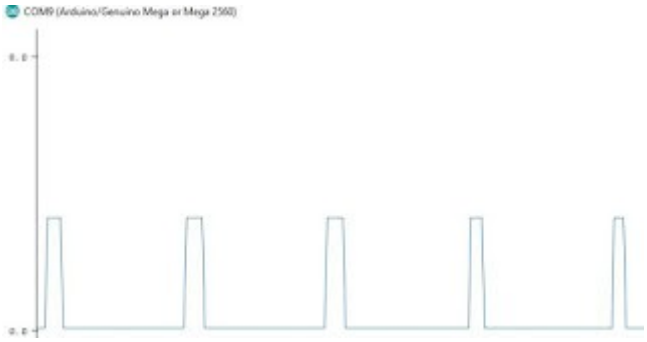


Gráfico de la señal que nos muestra el IDE 1.6.8 de Arduino.

```
/* Programa que muestra por el puerto serie el voltaje (entre 0 y
5 voltios) que entra por el pin 0A Analógico del Arduino.
 *  Se tiene que conectar el cable directamente al pin 0
analógico. Funciona correctamente y con muy buena resolución.
 *  Ver el libro Arduino Cookbook página 178. Funciona bien,
modificado.
*///// Declaración de
variables//////////////////////////////////////////////////////////
////////////////////////////////
int val;
float volts;
const float referenceVolts = 5; // Definición del voltaje de
referencia 5-voltios para la placa UNO y MEGA
const float resistorFactor = 1550; // Valor determinado solo para
valores entre 0 y 5 voltios.
const int batteryPin = 0; // Pin 0 Analógico donde se conecta la
entra de señal.
//// Configuración del
Arduino ///////////////////////////////////////////////////////////
////////////////////////////////
void setup() {
  Serial.begin(9600); // Velocidad de transmisión del puerto serie
IDE 1.6.8.
  Serial.println("Volts");
}
//// Inicio del
programa ///////////////////////////////////////////////////////////
//////////////////////////////////////////////
void loop() {
  val = analogRead(batteryPin); // Lee el valor del sensor en el
pin analógico
```

```
  volts = (val / resistorFactor) * referenceVolts ; // Calcula el
voltage de entrada
  Serial.println(volts ); // Imprime los valores en voltios en el
puerto serie.
}
//// Fin del
programa ////
```

http://androminarobot.blogspot.com/2016/04/tutorial-sobre-el-encoder-fotoelectrico.html