

UACM

Universidad Autónoma
de la Ciudad de México

Nada humano me es ajeno

PLANTEL SAN LORENZO TEZONCO

PROYECTO FINAL

OBJETIVO:

GENERADOR DE SEÑALES CON ARDUINO

Asignatura:

PROCESAMIENTO DIGITAL DE SEÑALES.

Estudiante:

SÁNCHEZ BARROSO ALFONSO

TAPIA DE LA CRUZ JORGE EDUARDO

VALVERDE QUINTANA CHRISTOPHER

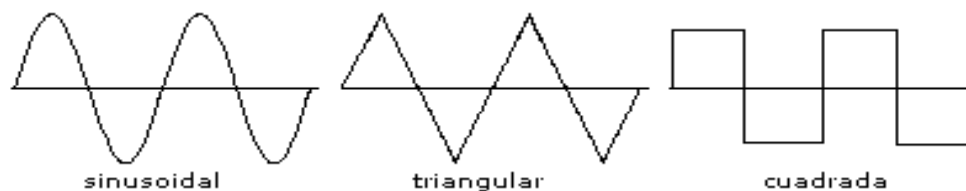
PROFESOR:

VAZQUEZ VILLANUEVA CHRISTIAN AGUSTIN

OBJETIVO:

Se realizara un generador de señales con arduino el cual nos mostrara las señales sinusoidal, triangular y cuadrada, a lo cual en estas señales se podrá modificar manualmente su frecuencia y su amplitud.

Esto se lograra usando un arduino (UNO) con el cual se programara para obtener dichas señales.



INTRODUCCIÓN:

La plataforma Arduino, está compuesta por hardware y software, la cual está basada en un microcontrolador con entradas y salidas, tanto analógicas como digitales. Esta plataforma tiene la característica de ser un sistema abierto, lo que significa que su diseño como su distribución son libres, es decir se puede utilizar sin haber adquirido licencia alguna; así también están disponibles los archivos de diseño (CAD) permitiendo al usuario adaptarlo a necesidades específicas. Otra ventaja de Arduino es que es compatible con Windows, Mac OS y Linux, que a diferencia del entorno de otros microcontroladores están limitados a Windows.

Debido a que existen diversas librerías, Arduino puede personalizarse con nuevas funcionalidades, por lo que esta plataforma facilita el desarrollo de aplicaciones en distintas áreas de la electrónica, tales como: Procesamiento de señales, electrónica de potencia, automatización y control, etc. Actualmente Arduino, ha comenzado a tomar relevancia a nivel mundial, no solo por ser una plataforma abierta, si no porque también está orientado a usuarios no programadores, ya que utiliza el lenguaje "Processing" (el cual ha sido creado para la enseñanza de la programación en un contexto visual) en conjunto con "Wiring" (plataforma de Hardware multipropósito con un ambiente para no programadores).

El arduino Uno es una placa electrónica basada en el ATmega328P. Cuenta con 14 pines digitales de entrada / salida (de los cuales 6 se podrán utilizar como salidas PWM), 6 entradas analógicas, un cristal de cuarzo de 16 MHz, una conexión USB, un conector de alimentación, una jefe de ICSP y un botón de reinicio. Contiene todo lo necesario para apoyar el microcontrolador; basta con conectarlo a un ordenador con un cable USB o la corriente con un adaptador de CA a CC o una batería para empezar. Puede jugar con el UNO sin preocuparse demasiado por hacer algo mal, peor de los casos puede sustituir el saltar por unos pocos dólares y empezar de nuevo.

"UNO" en italiano y fue elegido para celebrar el lanzamiento del software de Arduino (IDE) 1.0. La junta Uno y la versión 1.0 del software de Arduino (IDE) fueron las versiones de referencia de Arduino, ahora evolucionado para nuevos lanzamientos. La junta Uno es el primero de una serie de placas Arduino USB, y el modelo de referencia para la plataforma Arduino; para una extensa lista de las tarjetas actuales, anteriores u obsoletos.

TABLA 1: Especificaciones técnicas

Microcontrolador	ATmega328P
Tensión de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límite)	6-20V
E / S digitales prendedores	14 (de los cuales 6 proporcionan salida PWM)
PWM digital pines I / O	6
Pines de entrada analógica	6
Corriente continua para Pin I / O	20 mA
Corriente CC para Pin 3.3V	50 mA
Memoria flash	32 KB (ATmega328P) de los cuales 0,5 KB utilizado por cargador de arranque
SRAM2 KB	(ATmega328P)

EEPROM	1 KB (ATmega328P)
Velocidad de reloj	16 MHz
Longitud	68,6 mm
Anchura	53,4 mm
Peso	25 g

Los ATmega328 en el Uno viene pre programado con un cargador de arranque que le permite cargar nuevo código a ella sin el uso de un programador de hardware externo. Se comunica utilizando el protocolo original STK500 (referencia, archivos de cabecera C)

Cada uno de los 14 pines digitales en el Uno se puede utilizar como una entrada o salida, utilizando `pinMode ()` , `digitalWrite ()` , y `digitalRead ()` funciones. Operan a 5 voltios. Cada pin puede proporcionar o recibir 20 mA como condición de funcionamiento recomendada y tiene una resistencia de pull-up (desconectada por defecto) de 20-50k ohmios. Un máximo de 40 mA es el valor que no debe superarse en cualquier pin de E / S para evitar daños permanentes en el microcontrolador.

Además, algunos pines tienen funciones especializadas:

De Serie: 0 (RX) y 1 (TX). Se utiliza para recibir (RX) y transmitir datos en serie (TX) TTL. Estos pines están conectados a los pines correspondientes del chip de serie ATmega8U2 de USB a TTL.

Las interrupciones externas: 2 y 3. Estos pines pueden ser configurados para desencadenar una interrupción en un valor bajo, un flanco ascendente o descendente, o un cambio en el valor. Véase la función `attachInterrupt ()` para más detalles.

PWM: 3, 5, 6, 9, 10, y 11. Proporciona una salida de PWM de 8 bits con la función `analogWrite ()`.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Estos pines soportan la comunicación SPI utilizando la librería SPI.

LED: 13. Hay un LED incorporado impulsado por pin digital 13. Cuando el pasador es ALTO, el LED está encendido, cuando el pasador es bajo, es apagado.

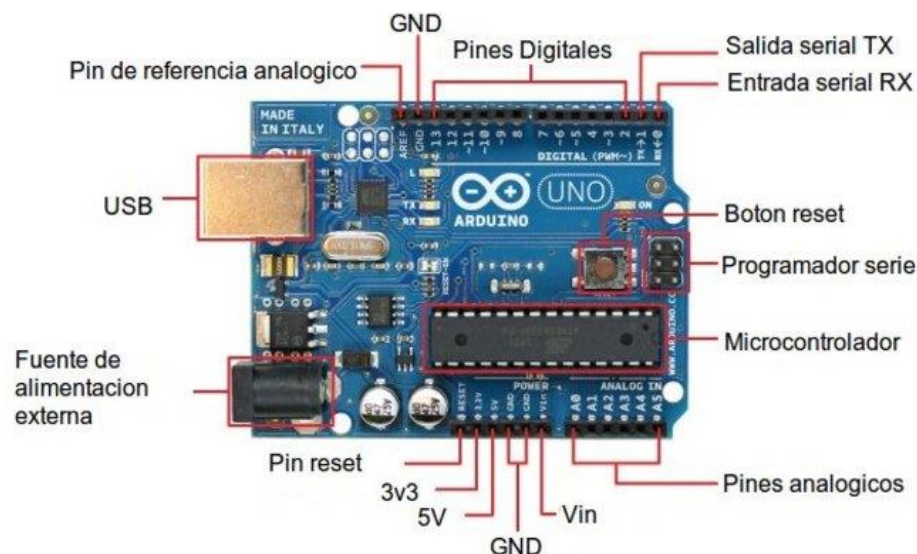
TWI: pin A4 o A5 o SDA y SCL pasador. TWI soporte de comunicación utilizando la biblioteca de alambre.

El Uno tiene 6 entradas analógicas, A0 a A5 marcado, cada uno de los cuales proporcionan 10 bits de resolución (es decir, 1024 valores diferentes). Por defecto se miden desde el suelo a 5 voltios, aunque es posible cambiar el extremo superior de su rango de uso de la función de la `analogReference()` pin AREF y.

Hay un par de patas de la placa:

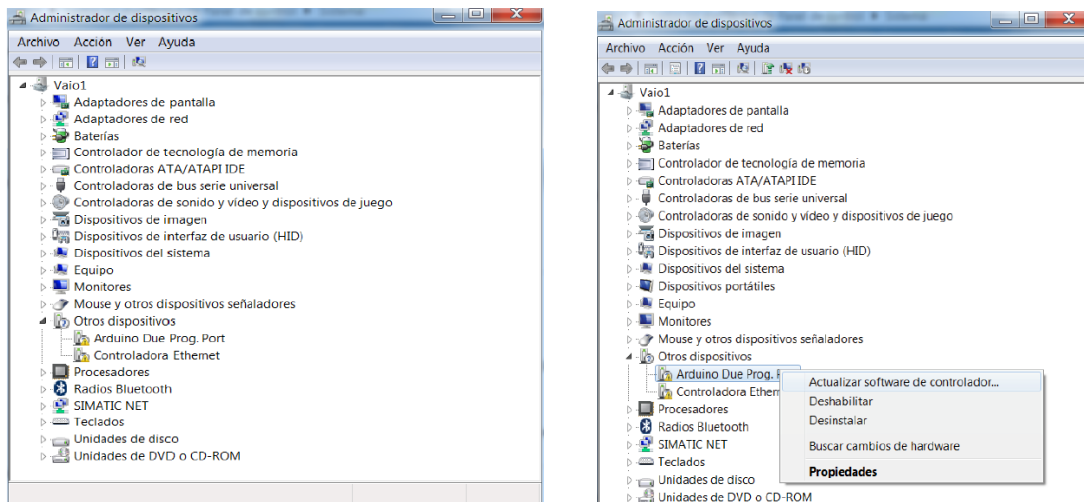
AREF. Voltaje de referencia para las entradas analógicas. Se utiliza con `analogReference()`.

Reiniciar. Llevar esta línea baja para reiniciar el microcontrolador. Normalmente se utiliza para añadir un botón de reinicio para escudos que bloquean la una en la mesa.

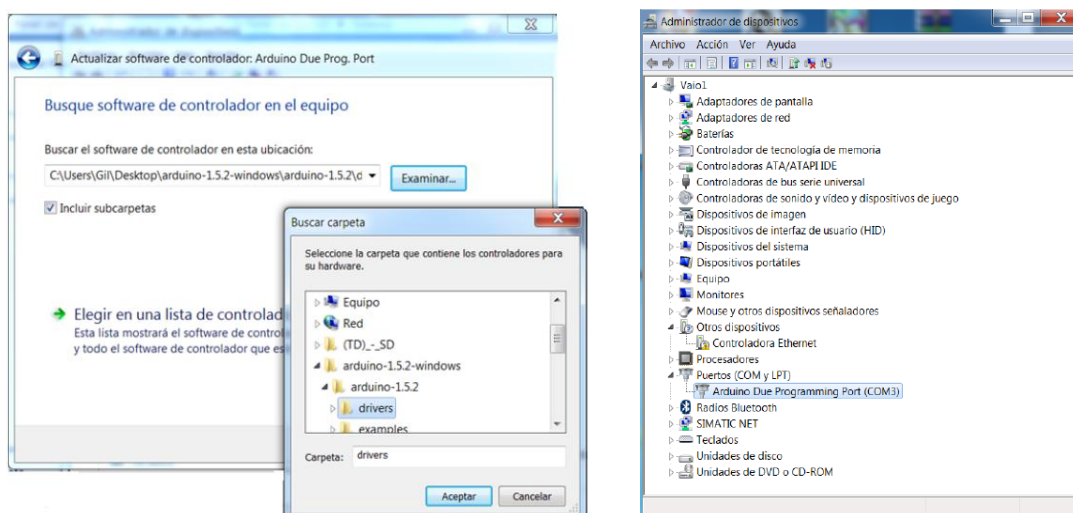


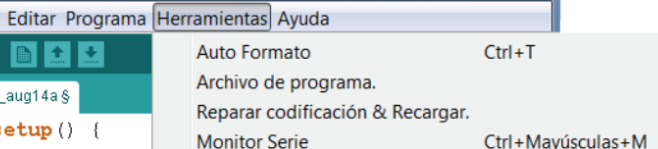
El primer paso para establecer el enlace con Arduino UNO a través de la computadora, es descargar el software desde la página oficial de Arduino.

La instalación del controlador de Arduino se realizará de manera manual, esto con la finalidad de identificar el puerto asignado por la computadora a la plataforma de Arduino, ya que posteriormente en el software de programación se especificará el mismo puerto para poder tener el enlace.



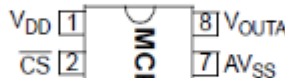
- Primero hay que abrir el administrador de dispositivos.
- Luego se selecciona donde dice arduino, se da clic derecho y damos clic en actualizar controlador



- 
- The screenshot shows the Arduino IDE interface. The 'Herramientas' (Tools) menu is open, and the 'Puerto Serie' (Serial Port) option is selected. The dropdown menu shows 'COM3' as the selected port. The background shows a C++ sketch in the editor.
- ```
void setup() {
 // Escribir el código aquí para
 // ejecutarse una vez
}

void loop() {
 // Escribir el código aquí para
 // ejecutarse repetidamente:
}
```

Es un dispositivo para convertir un código digital (generalmente binario, compuesto de ceros y unos) a una señal analógica (corriente, voltaje o carga eléctrica). Hay distintos componentes que pueden intervenir en este proceso, como interruptores simples, red de resistores, fuentes actuales o condensadores. Un convertidor de analógico a digital (ADC) realiza la operación inversa.

| UNO | MEGA | MCP4921                                                                             | MEGA          | UNO |
|-----|------|-------------------------------------------------------------------------------------|---------------|-----|
|     |      |                                                                                     |               |     |
|     |      | <b>8-Pin PDIP, SOIC, MSOP</b>                                                       |               |     |
| 5V  | 5V   |  | Analog output |     |
| 10  | 53   |                                                                                     | GND           | GND |
| 13  | 52   |                                                                                     | +5V           | +5V |
| 11  | 51   |                                                                                     | GND           | GND |
|     |      |                                                                                     |               |     |

Un sistema de control (como un microcontrolador) no tiene capacidad alguna para trabajar con señales analógicas, de modo que necesita convertirlas en señales digitales para poder trabajar con ellas. Es por ello que esta transformación es requerida para poder trabajar con ella en el ARDUINO.

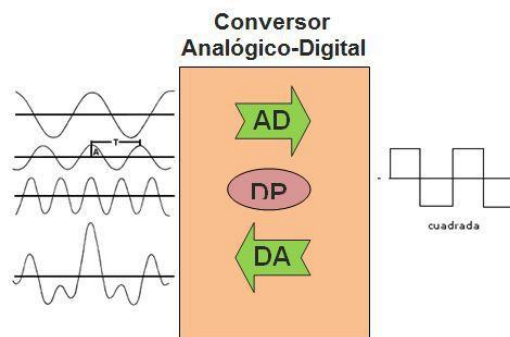
La señal digital obtenida de una analógica tiene dos propiedades fundamentales: Valores.

- Qué valor en voltios define 0 y 1. En nuestro caso es tecnología TTL (0 – 5V)
- Resolución analógica: nº de bits que usamos para representar con una notación digital una señal analógica:

En el caso de un arduino Uno, el valor de 0 voltios analógico es expresado en digital como B0000000000 (0) y el valor de 5V analógico es expresado en digital como B1111111111 (1023). Por lo tanto todo valor analógico intermedio es expresado con un valor entre 0 y 1023, es decir, sumo 1 en binario cada 4,883 mV

- Arduino Uno tiene una resolución de 10 bits, es decir, unos valores entre 0 y 1023.
- Arduino Due tiene una resolución de 12 bits, es decir, unos valores entre 0 y 4095.

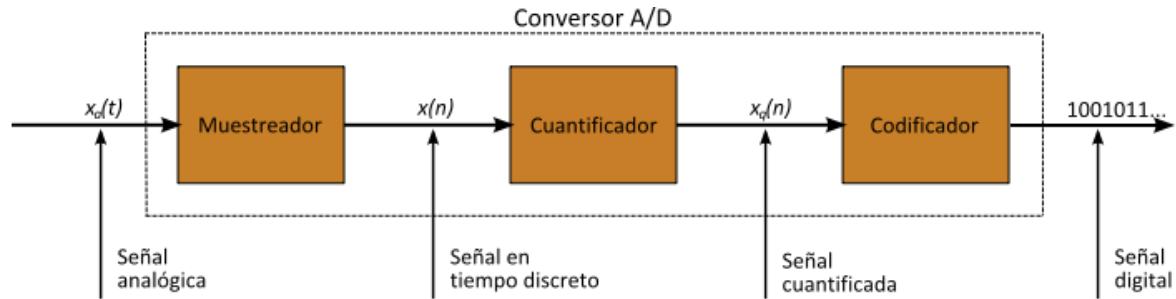
Diferencia entre señales analógicas y digitales:



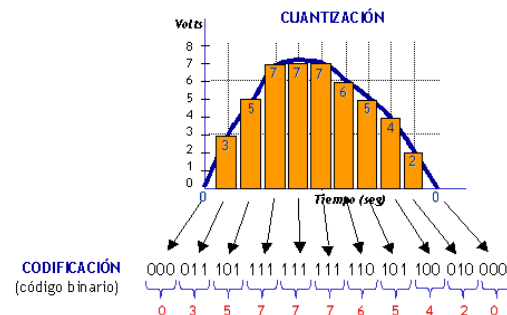
Un microcontrolador solo entiende señales digitales (1's y 0's), por lo tanto para poder leer señales analógicas necesitamos los convertidores Analógico a Digital (ADC).



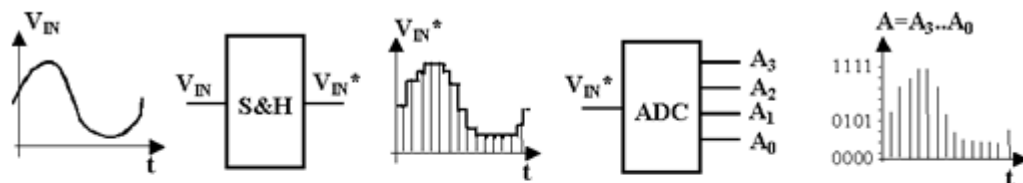
Como funciona un conversor analógico a digital:



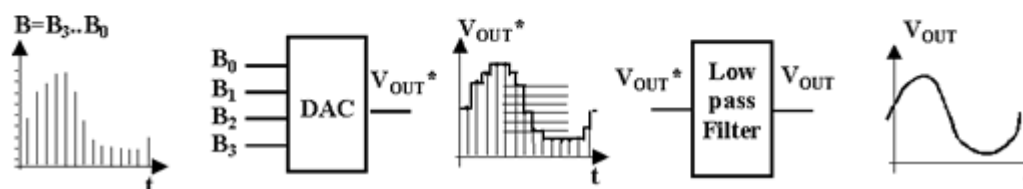
- Como vemos en la imagen hay una señal de entrada la cual va a ser muestreada esto es que se tomaran puntos de la señal mientras más muestras se tomen será más cercana a la señal original (analógica).
- En la conversión de señal digital a analógica y esto es debido a que se truncan los valores entre 2 niveles de cuantificación, mientras mayor cantidad de bits mayor resolución y por lo tanto menor información pérdida.



Analog to Digital Converter converts an analog input to a digital output



Digital to Analog Converter converts a digital signal to an analog output



## **SISTEMAS DISCRETOS**

**Los sistemas de control de tiempo discreto (STD) son sistemas dinámicos para los cuales una ó más de sus variables solamente son conocidas en ciertos instantes.**

Por lo tanto, son aquellos que manejan señales discretas, a diferencia de los sistemas de tiempo continuo (STC) en los cuales sus variables son conocidas en todo momento.

El hecho de que algunas funciones del tiempo propias del STD varíen en forma discreta, puede provenir de una característica inherente al sistema, como en el caso de aquellos que trabajan con algún tipo de barrido, por ejemplo: un sistema de radar.

La otra posibilidad es que la variación discreta provenga de un proceso de muestreo de alguna señal, y estos últimos son los que interesan en este estudio.

**Este proceso de muestreo, que convierte una señal analógica o de tiempo continuo en una señal discreta o muestreada, podría hacerse a un ritmo *constante*, *variable* según alguna ley de variación o *aleatorio*.**

La discretización de una señal es el paso previo para su digitalización, proceso que agrega una determinada codificación a la señal muestreada.

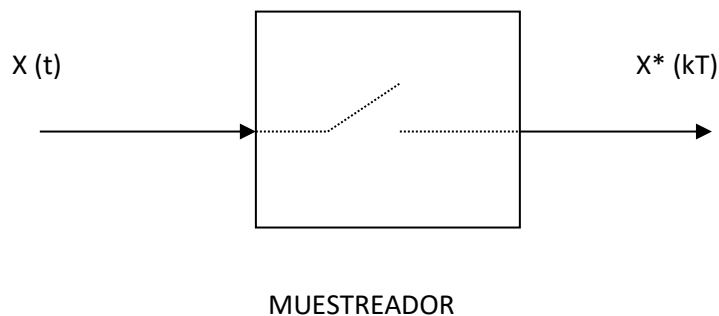
La digitalización de las señales es un proceso imprescindible para poder procesar las mismas en computadoras digitales, y presenta además las ventajas de permitir su transmisión con una mayor densidad y velocidad en la información, además de reducir el costo y volumen de los equipos debido a que se requieren magnitudes de energía significativamente más bajas, etc.

## EL MUESTREO

El proceso de muestreo consiste básicamente en producir una alternancia entre los intervalos de presencia de señal y ausencia de la misma, como si se “tomaran muestras” de la señal analógica.

Es equivalente al trabajo de una llave o interruptor electrónico que se abre y se cierra alternadamente, permitiendo el paso de la señal sólo en determinados instantes, que se denominan instantes de muestreo.

A la entrada del dispositivo de muestreo, llamado “muestreador”, aparece una señal (función del tiempo) analógica y a la salida resultará muestreada o discretizada, lo que se indica con un asterisco.



Así mismo, la señal de salida que está muestreada o discretizada  $[x^*]$ , no será función del tiempo continuo, sino precisamente función de los instantes discretos para los cuales ha quedado definida, que se indican como  $kT$ , donde  $k=1,2,3,4,5,\dots$ ; y  $T$  es el intervalo o periodo de muestreo.

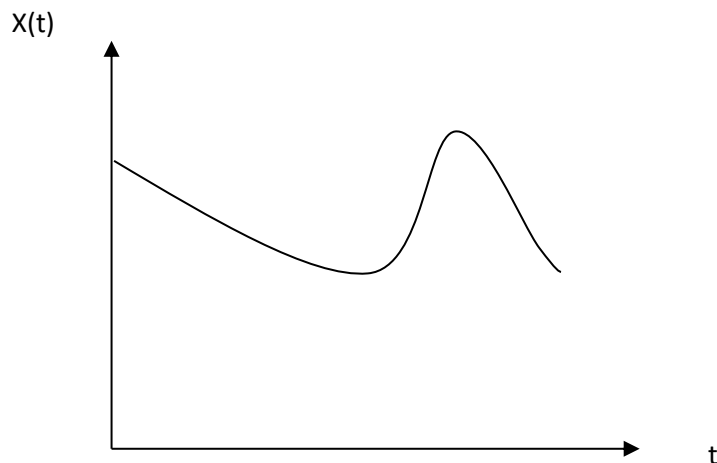
## **PROCESO DE MUESTREO**

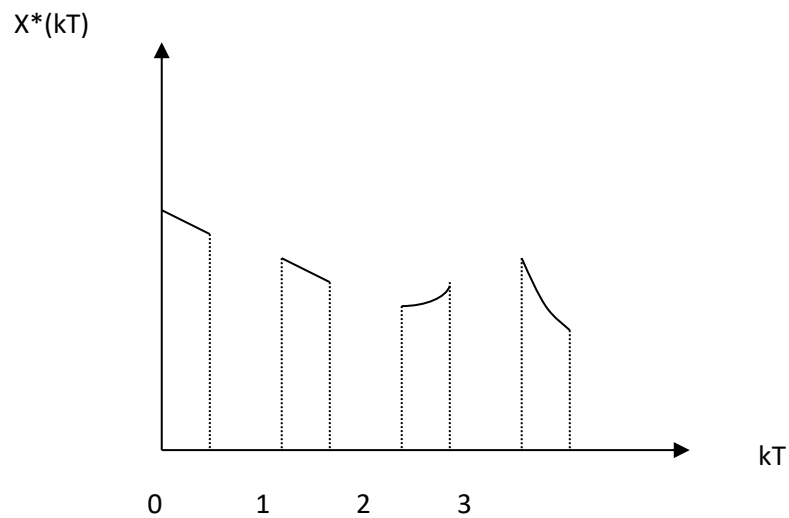
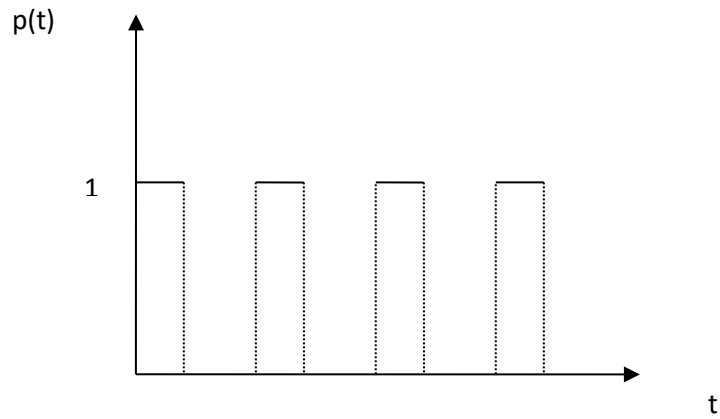
En un sistema de control, el muestreo puede tener lugar en uno ó más puntos del sistema y el muestreador se representa simbólicamente por un interruptor intercalado en el lugar donde se produce el proceso de muestreo.

**Básicamente, se ingresa al muestreador con una señal analógica, de tiempo continuo y a la salida se obtiene una señal discreta.**

**En definitiva el proceso opera como una modulación en amplitud que produce la señal continua sobre un tren de pulsos  $p(t)$ . Este tren de pulsos es provocado por la apertura y cierre del interruptor de muestreo a intervalos regulares.**

Así como en los STC se describe su comportamiento por medio de las ecuaciones diferenciales correspondientes, en los STD se utilizan ecuaciones de diferencias para representarlos matemáticamente.

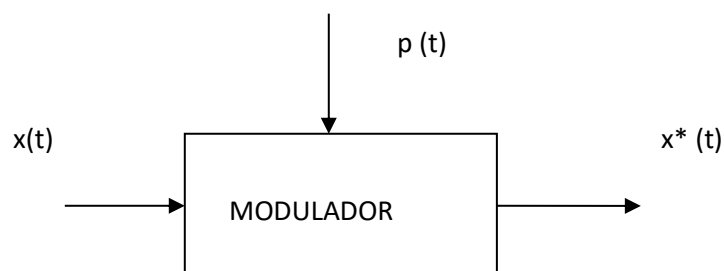




La señal muestreada puede considerarse como el producto del tren de pulsos por la señal analógica, como en toda modulación de amplitud, y para mayor comodidad se supone que los pulsos tienen amplitud unitaria, por lo tanto no se necesita ningún factor de escala:

$$x^*(t) = x(t) p(t)$$

lo que corresponde al siguiente esquema:



**El tren de pulsos evidentemente es una función periódica del tiempo, por lo tanto admite un desarrollo en serie de Fourier:**

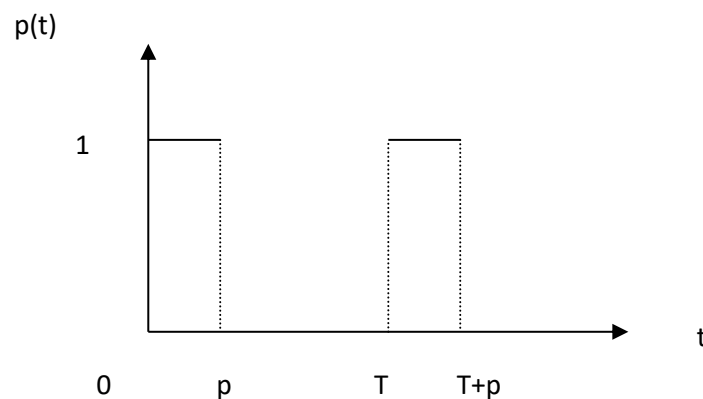
$$p(t) = \sum_{n=-\infty}^{\infty} c_n e^{jn\omega_s t}$$

Donde la frecuencia  $\omega_s$  es la frecuencia de muestreo, tal que  $\omega_s = 2\pi/T$  siendo, obviamente,  $T$  el periodo de muestreo, que será el mismo para el tren de pulsos, como para la función muestreada, que resulta periódica del mismo periodo. (El subíndice corresponde al término inglés *sampling*: muestreo).

El coeficiente de Fourier correspondiente,  $c_n$  se puede calcular del siguiente modo:

$$c_n = 1/T \int_0^T p(t) e^{-jn\omega_s t} dt$$

Pero, teniendo en cuenta la duración del pulso, que llamaremos  $p$ , resulta:



Por lo tanto, la integral sólo tiene sentido entre  $t = 0$  y  $t = p$ ; y en ese intervalo  $p(t) = 1$  de manera que se reduce a:

$$c_n = 1/T \int_0^p 1 e^{-jn\omega_s t} dt = 1/T \frac{1}{(-jn\omega_s)} [e^{-jn\omega_s t}]_0^p = -1/(Tjn\omega_s) [e^{-jn\omega_s p} - 1]$$

Finalmente:

$$c_n = 1/(Tj\omega_s) [1 - e^{-jn\omega_s p}]$$

Esta expresión conviene llevarla a la forma  $(\sin \alpha)/\alpha$  o sea la función sinc  $\alpha$ , para ello se supone que:

$$1 = e^{-jn\omega_s p/2} e^{jn\omega_s p/2}$$

$$e^{-jn\omega_s p} = e^{-jn\omega_s p/2} e^{-jn\omega_s p/2}$$

Haciendo los reemplazos correspondientes, resulta:

$$c_n = 1/(Tj\omega_s) [e^{-jn\omega_s p/2} e^{jn\omega_s p/2} - e^{-jn\omega_s p/2} e^{-jn\omega_s p/2}]$$

y sacando factor común  $e^{-jn\omega_s p/2}$

Resultará:

$$c_n = 1/(Tj\omega_s) e^{-jn\omega_s p/2} [e^{jn\omega_s p/2} - e^{-jn\omega_s p/2}]$$

Si además se multiplica y divide toda la expresión por  $2p$  quedará:

$$c_n = 1/(Tj\omega_s) e^{-jn\omega_s p/2} [e^{jn\omega_s p/2} - e^{-jn\omega_s p/2}] 2p/2p =$$

$$= 2p/(Tpn\omega_s) e^{-jn\omega_s p/2} \{[e^{jn\omega_s p/2} - e^{-jn\omega_s p/2}] / 2j\}$$

Y la última parte de esta expresión, indicada entre llaves, corresponde a la definición de la función seno por fórmulas de Euler, luego:

$$c_n = 2p/(Tpn\omega_s) e^{-jn\omega_s p/2} [\sin (n\omega_s p/2)] = p/T e^{-jn\omega_s p/2} [\sin (n\omega_s p/2) / p/2n\omega_s] =$$

$$c_n = p/T e^{-jn\omega_s p/2} \text{sinc} (n\omega_s p/2)$$

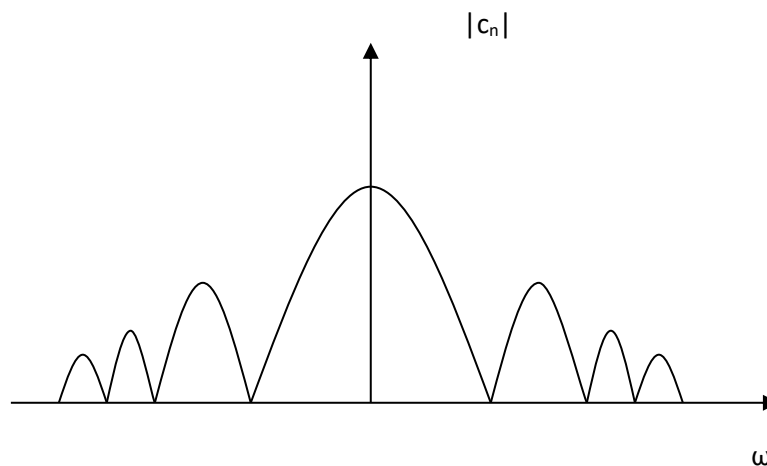
Así mismo, la función muestreada se puede expresar teniendo en cuenta el desarrollo de Fourier del tren de pulsos, como:

$$x^*(t) = x(t) \sum_{n=-\infty}^{\infty} c_n e^{jn\omega_s t} = \sum_{n=-\infty}^{\infty} x(t) c_n e^{jn\omega_s t}$$

por lo tanto  $c_n$  es también el coeficiente de Fourier de la señal discreta, que tiene el desarrollo en serie con el mismo periodo  $T$ .

El coeficiente  $c_n$ , como es sabido, estará relacionado con la transformada de Fourier de la señal discreta  $X^*(j\omega)$ , que dará su espectro de frecuencias. Si se toma el módulo del coeficiente:

$$|c_n| = p/T |\text{sinc}(n\omega_{sp}/2)|$$



## **ESPECTRO DEL MUESTREADOR**

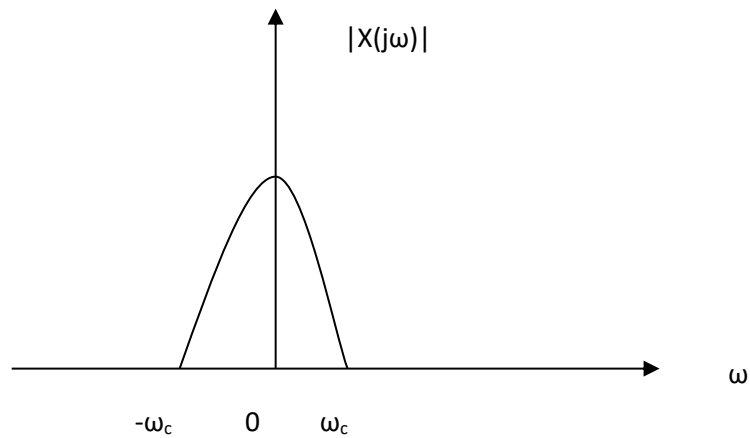
Por otra parte, la señal analógica o de tiempo continuo, tendrá a su vez su propio espectro de frecuencias, dado por su respectiva transformada de Fourier  $X(j\omega)$ , que irá desde  $\omega = 0$  hasta una frecuencia máxima  $\omega = \omega_c$  que llamaremos “máxima frecuencia de la señal continua o analógica”

Como se verá más adelante en el desarrollo del tema, es muy importante que la señal analógica tenga un ancho de banda limitado para que sea posible discretizarla.

Si esta frecuencia máxima fuera muy alta la discretización se complica y si el ancho de banda de la señal analógica fuera ilimitado, entonces resultaría directamente imposible discretizarla.



El espectro de frecuencias de la señal analógica o continua tiene un aspecto como el que sigue:



Si ahora se calcula la transformada de Fourier de la señal discreta, se obtendrá, a partir de su expresión:

$$x^*(t) = \sum_{n=-\infty}^{\infty} c_n x(t) e^{jn\omega_s t}$$

la transformada como:

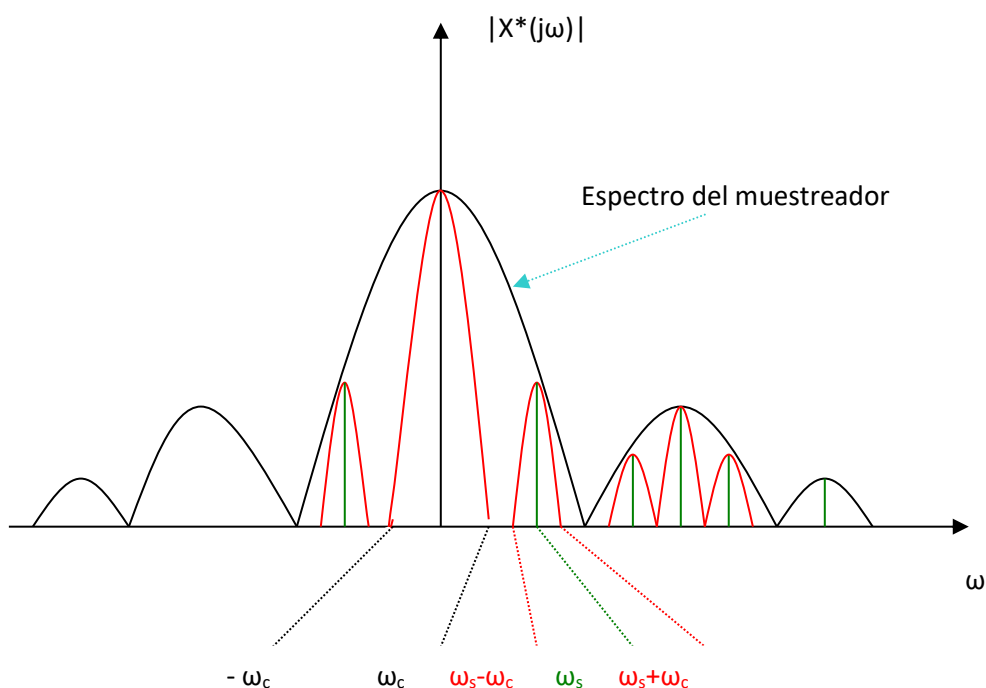
$$X^*(j\omega) = \int_{-\infty}^{\infty} [\sum c_n x(t) e^{jn\omega_s t}] e^{-j\omega t} dt$$

$$= \sum c_n \int x(t) e^{-(j\omega - jn\omega_s)t} dt$$

y por aplicación del Teorema del Desplazamiento en Transformada de Fourier, resulta:

$$\sum c_n \int x(t) e^{-(j\omega - jn\omega_s)t} dt = \sum c_n X(j\omega - j\omega_s)$$

Cuya representación es:

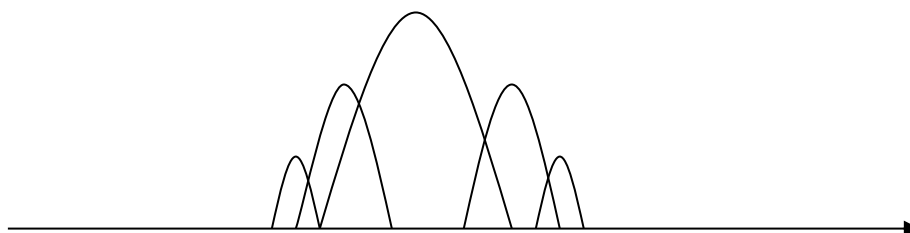


Las ordenadas dibujadas representan los distintos coeficientes  $c_n$  y todos los valores son positivos porque se ha tomado el módulo de la Transformada de Fourier de la señal discreta.

El muestreo produce un espectro muy parecido al de la señal continua, pero por efecto del Teorema del Desplazamiento se despliega en una colección de réplicas del espectro fundamental, desplazadas a intervalos que son múltiplos de la frecuencia de muestreo,  $[n \omega_s]$ . A su vez, todos los espectros adicionales resultan modulados en amplitud por el espectro del muestreador.

Si la frecuencia de muestreo es suficientemente elevada respecto de la frecuencia máxima de la señal analógica, no habrá superposición entre los espectros adicionales, lo cual es deseable.

Si esta frecuencia  $\omega_s$  no es suficientemente elevada, se producirá superposición del siguiente modo, lo cual redundará en pérdida de información del modo en que aprecia a continuación:



## DESARROLLO

Una vez visto las características del ARDUINO y el funcionamiento del circuito MCP4021, así como la forma en que se tiene que hacer la conversión de una señal analógica a digital para poder trabajar con el arduino, se comenzó a programar para principalmente obtener las señales sinusoidal, triangular y cuadrada.

**ESQUEMA:**

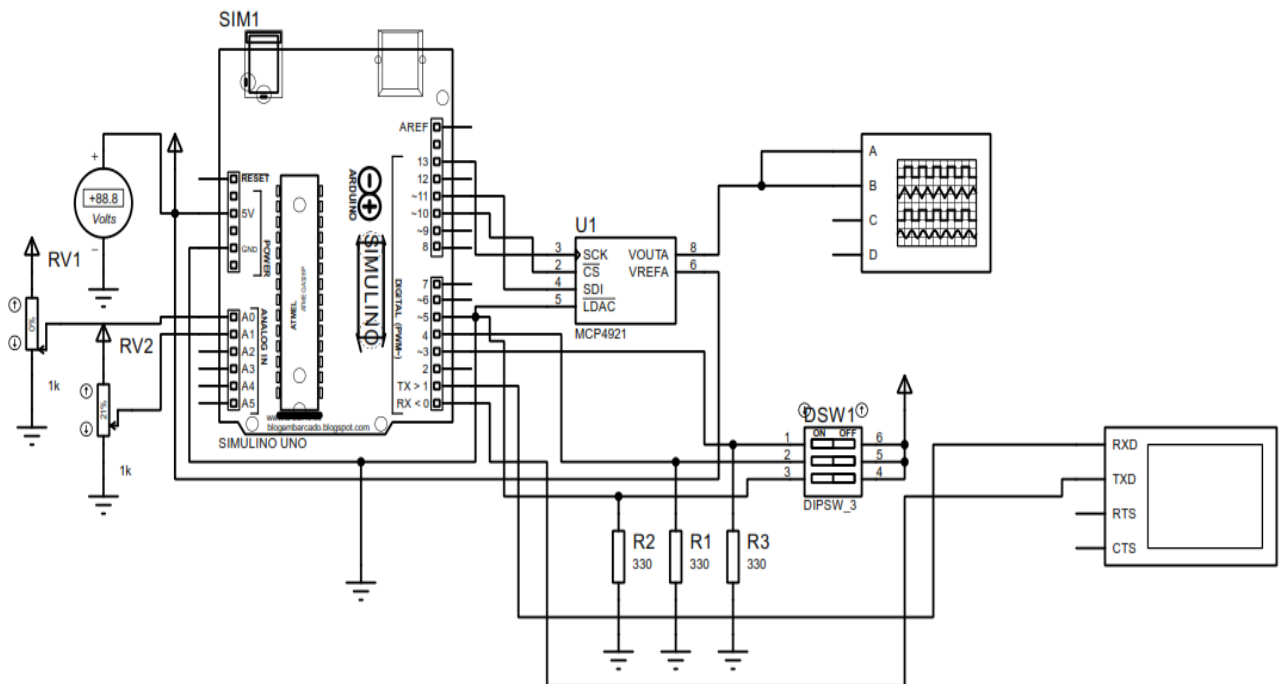


Fig.: 1 Captura del esquemático en Proteus

## MATERIALES

## ARDUINO UNO

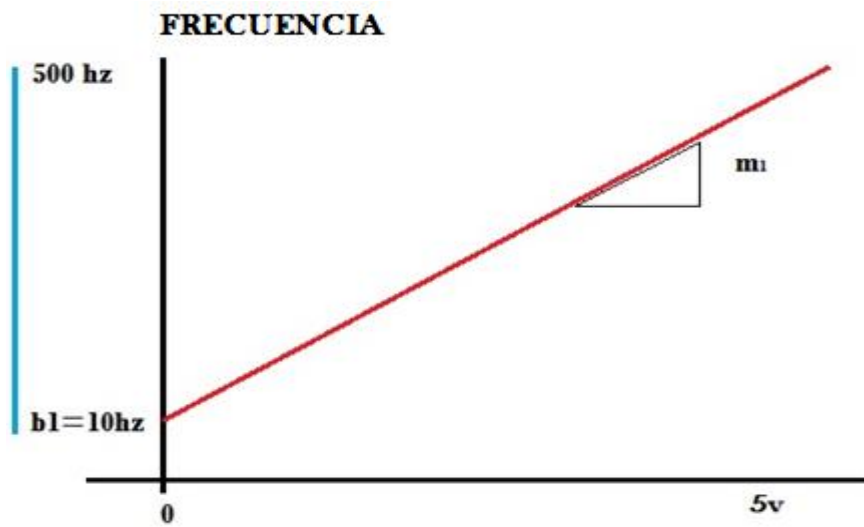
3 RESISTENCIAS 330Ω

2 RESISTENCIA VARIABLE 5K  $\Omega$ , 10K  $\Omega$

## 1 DIP SWITCH

DAC MCP492

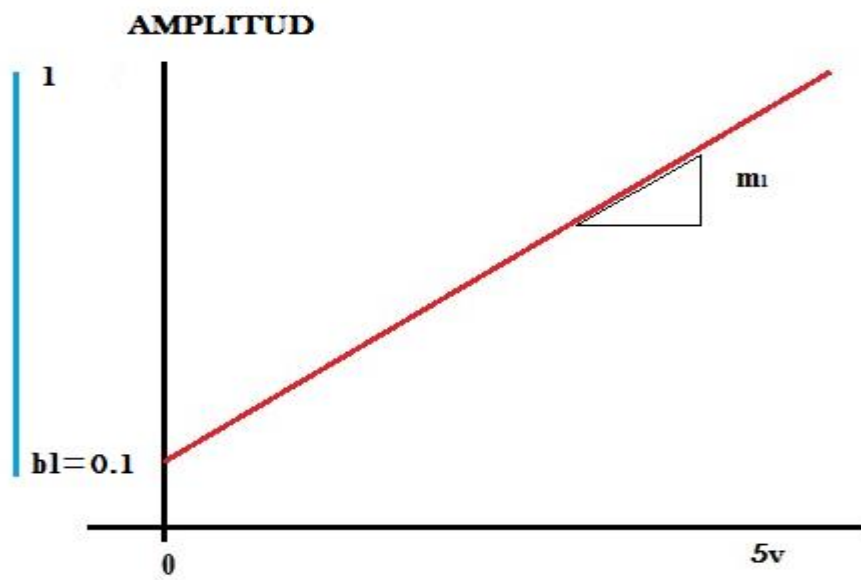
## 1 OSCILOSCOPIO



$$F = m_1 V_1 + b_1$$

$$F = \frac{500 \text{ h z} - 10 \text{ h z}}{5 \text{ v}} + 10 \text{ h z}$$

$$F = 98 + 10 \text{ Hz}$$



$$A = m_2 V_2 + b_2$$

$$A = \frac{1 - 0.1}{5 \text{ v}} + 0.1$$

$$A = 0.18 + 0.1$$

## CÓDIGO

```
#include <TimerOne.h>
#include <SPI.h>
#include <DAC_MCP49xx.h>
#define SS_PIN 10

DAC_MCP49xx dac(DAC_MCP49xx::MCP4921, SS_PIN);

int msen = 3; //señal senoidal
int mcua = 4; // señal cuadrada
int mtri = 5; //señal triangular

int bandera;
int x;
int yf;
int muestra=0; // numero de muestra
float frecuencia; // frecuencia de senal a generar
float frecuenciadigital; // frecuencia digital
float frecuenciamuestreo=1000; // 4 K Hz
float amplitud;
float amplitudx;
float xanterior=0; // declaramos bandera
float yanterior=0; // declaramos bandera

void ISR_timer(){

 bandera=1;
}

void setup(){

 pinMode(msen,INPUT); // se define entrada digital
 pinMode(mcua,INPUT);
 pinMode(mtri,INPUT);

 Serial.begin(9600); // inicializa comunicaciones serial
 Timer1.initialize(250); // Dispara cada 1 ms f=1/10ms = 1kHz
 Timer1.attachInterrupt(ISR_timer); // Activa la interrupcion y la asocia a ISR_timer
 dac.setSPIDivider(SPI_CLOCK_DIV16);
 dac.setPortWrite(true);
}

void loop(){
//Sin Signal

if (digitalRead(msen) == true){

if (bandera== 1){

 frecuencia =floor(98.0*((analogRead(1)*5.0)/(1024.0))+10.0); //frecuencia A0 potenciómetro0
 frecuenciadigital = frecuencia/frecuenciamuestreo;
 amplitud=0.18*((analogRead(0)*5.0)/(1024.0))+0.1; //amplitud A1 potenciómetro
 bandera=0;
 Serial.println(frecuencia);

 Serial.println(amplitud);

 x=(amplitud*sin(2*3.1416*muestra*frecuenciadigital))*2048+2048; //calculo seno digital de 12 bits // calculo seno digital de 12 bits; // calculo seno digital de 12 bits

 yf=0.29291*x +0.2929*xanterior+ 0.4142*yanterior;

 xanterior=x;

 yanterior=yf;

 int yf2 = floor(yf);
```

```

 muestra=muestra+1;
 dac.output(yf2);
}
}

// Square signal
else if (digitalRead(mcu) == true){
 frecuencia =floor(98.0*((analogRead(1)*5.0)/(1024.0))+10.0);
 amplitud = 0;
 for (int i=0; i< frecuencia; i++)
 dac.output(amplitud);
 amplitudx=0.18*((analogRead(0)*5.0)/(1024.0))+0.1;
 amplitud = amplitudx*4095;
 for (int i=0; i< frecuencia; i++)
 dac.output(amplitud);
}

// Triangle wave
else if (digitalRead(mtri) == true){
 /*amplitudx=0.18*((analogRead(0)*5.0)/(1024.0))+0.1;
 amplitud = amplitudx*4095;

 for (int i=frecuencia; i<25; i++)
 {
 dac.output(amplitud);
 }

 frecuencia =floor(98.0*((analogRead(1)*5.0)/(1024.0))+10.0);
 amplitud = 0;

 for (int i=24; i>=frecuencia; i--)
 {
 dac.output(i*100);
 }
 }

 */

// triangle wave
for (int i=0; i<25; i++)
{
 dac.output(i*100);
}
for (int i=24; i>=0; i--)
{
 dac.output(i*100);
}

}
}

```

## FOTOS DEL PROCESO DE MONTAJE

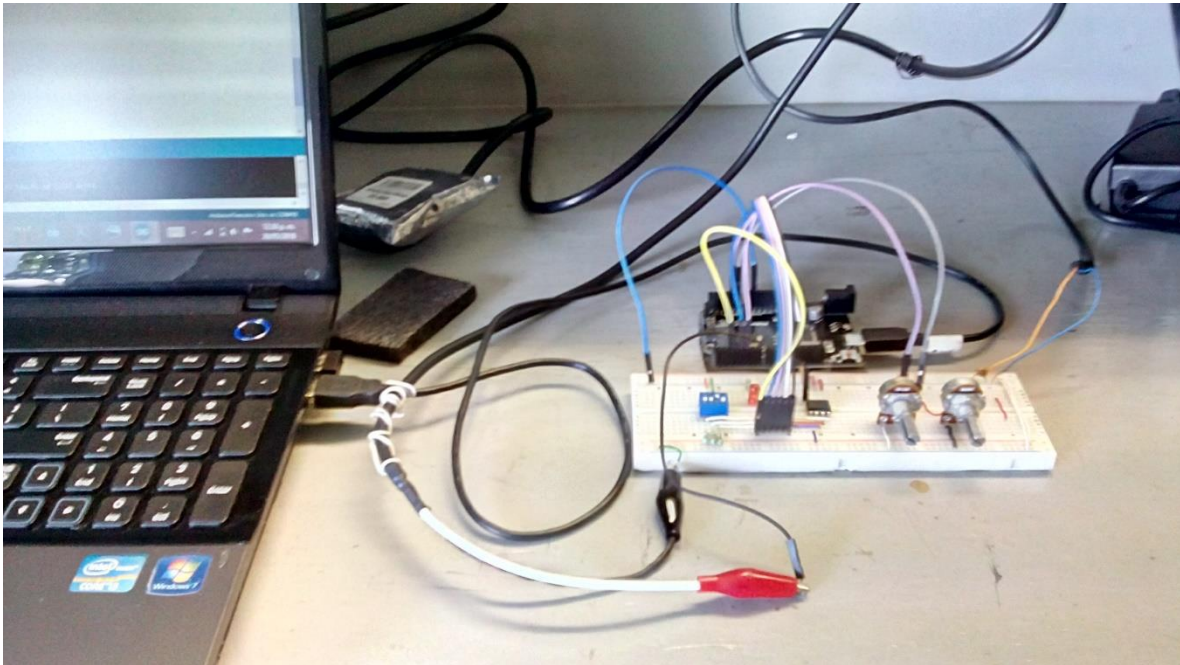
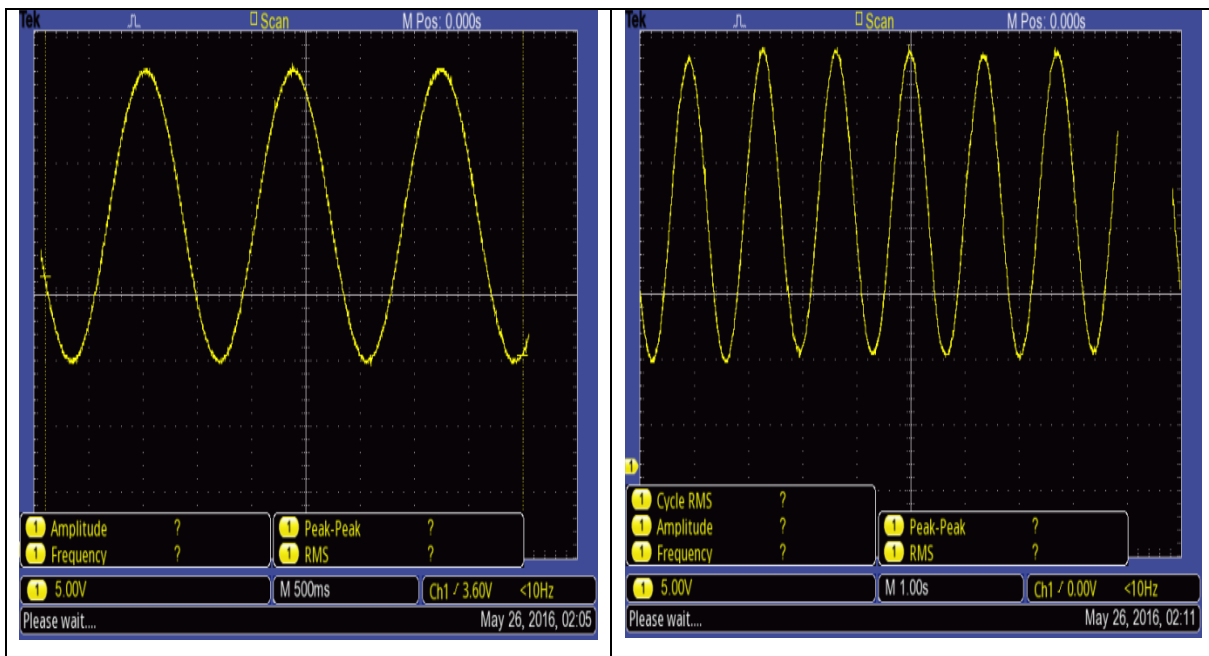
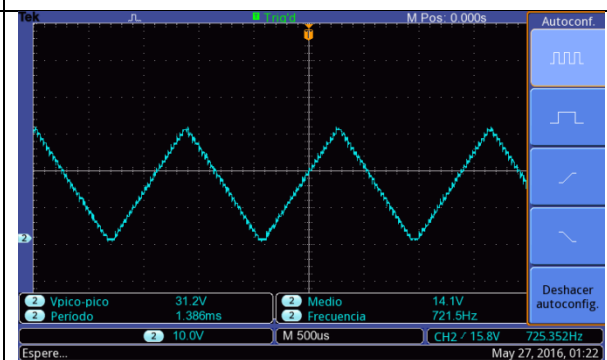
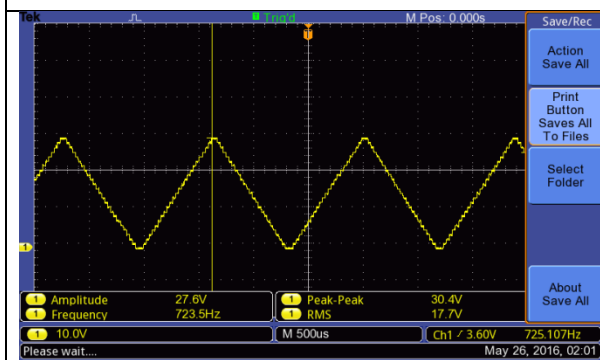
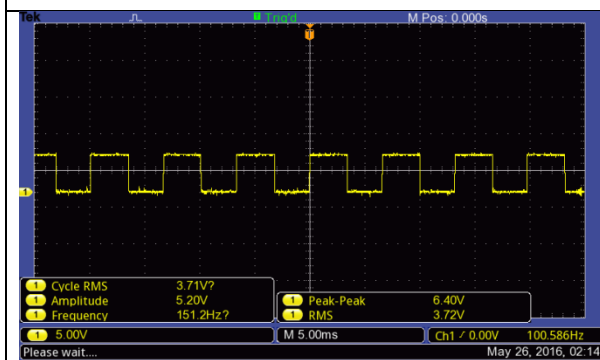
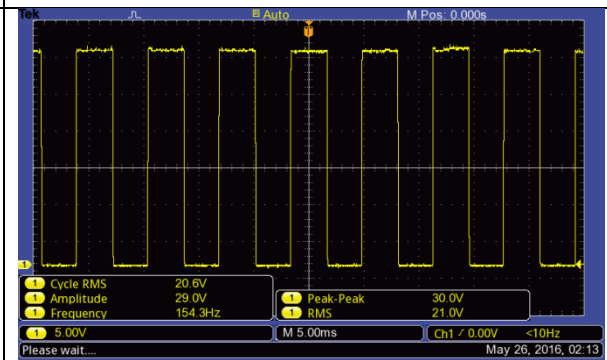
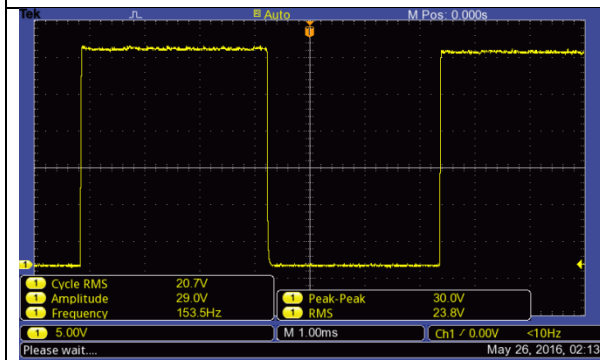
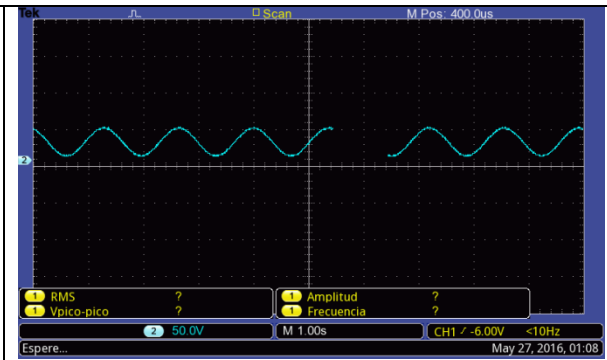
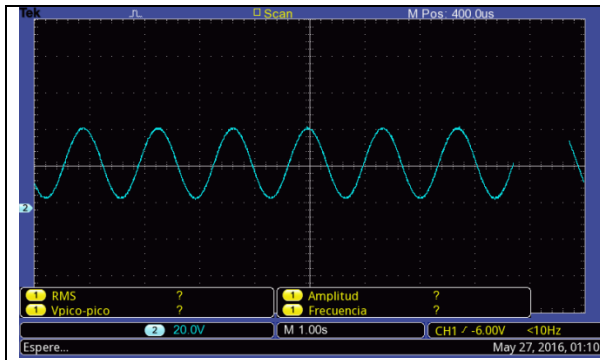


Figura 3 Generador de funciones con arduino uno (dac mcp492).

## SEÑAL SENOIDAL ARROJADA CON OSCILOSCOPIO.







## ESQUEMA FIN DEL GENERADOR DE FUNCIONES

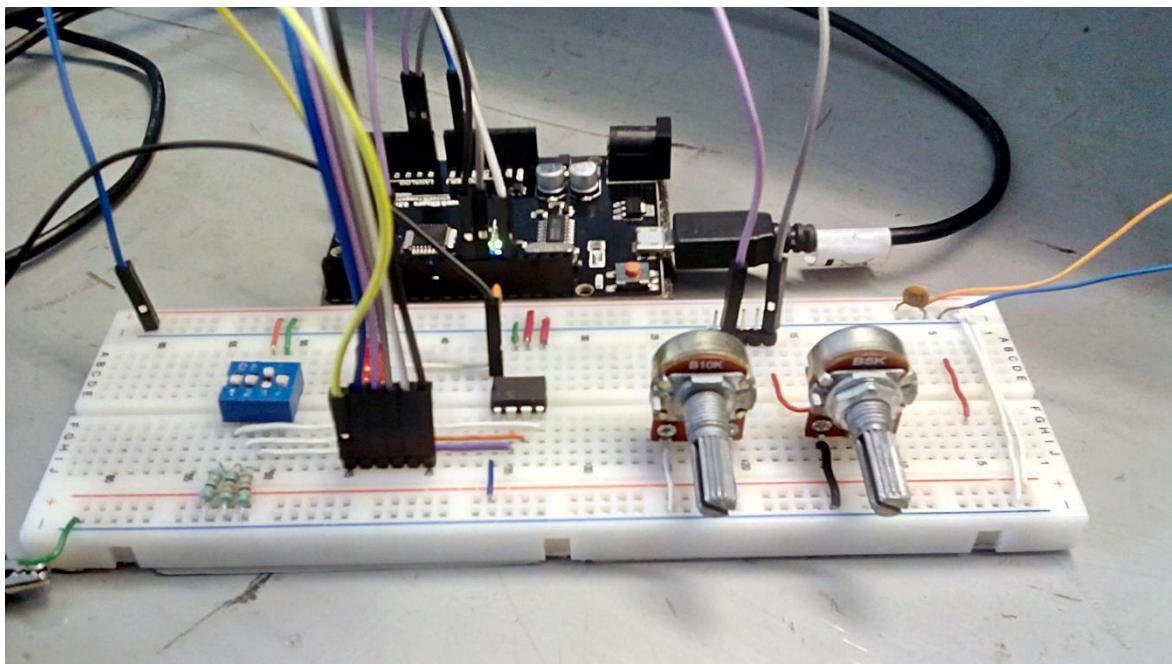
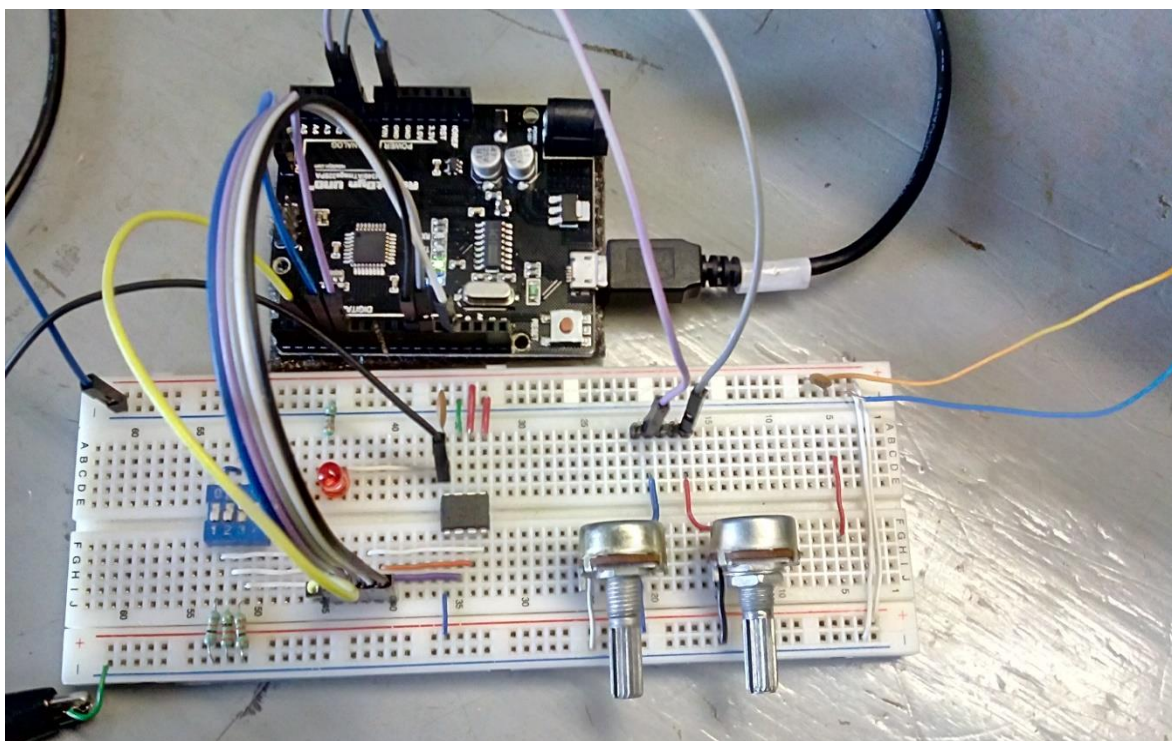


Foto 4. Circuito final "Generador de funciones".



## **CONCLUSIONES:**

Existen dos objetivos principales que se cumplieron en este trabajo. El primero de ellos es el diseño y desarrollo de un generador de ondas virtual, utilizando DAC mcp492 y Arduino Uno, y comparar sus características con un generador de ondas físico.

El segundo objetivo, entender mejor los conceptos vistos en clase con la práctica. Esto es una importante ventaja de este diseño. Y aunque con este prototipo no se ha conseguido un excelente funcionamiento, sí que ha servido para conocer algunas limitaciones del uso de DAC mcp492 y Arduino Uno para la transmisión y el tratamiento de datos.

## **BIBLIOGRAFÍA:**

[1] Procesamiento Digital de Señales EL-5805 Dr. José Pablo Alvarado Moya

[2] Bosch, J. and Carmona, M. ." Instrumentación Electrónica Avanzada. Instrumentación Inteligente. Departamento de Electrónica", Universidad de Barcelona, (2012)

[3] "Funcionamiento De Un Generador De Funciones."

<http://ingeniatic.net/index.php/tecnologias/item/465-generador-de-funciones>

[4] Analog Write Resolution()

<https://www.arduino.cc/en/Reference/AnalogWriteResolution>

[5] Aprendiendo con arduino: Entradas y salidas analógicas, PWM.

<https://aprendiendoarduino.wordpress.com/2015/03/30/entradas-y-salidas-analogicas-pwm/>

