|  |  |  |
| --- | --- | --- |
| Generador de funciones para Qmetro | mayo 25  2017 | |
| Alumno: Trapaglia Mansilla Matías Daniel | | Materia: Medidas Electrónicas II |

# Introducción

Los Qmetros que se encuentran actualmente en el laboratorio, utilizan para oscilar unas válvulas que, pese a su excelente precisión, presentan complicaciones a la hora de buscar repuesto en el caso de un mal funcionamiento por desgaste de las mismas.

De ahí nació la idea de fabricar un oscilador exterior opcional, el cuál se puede utilizar o no según se precise; el mismo aprovecha la tecnología digital existente y un microprocesador ampliamente utilizado, para realizar una función similar, en un tamaño reducido y con componentes accesibles a la hora de intercambiarlos.

# Características

Exteriormente, el dispositivo cuenta con tres bornes de alimentación

Positivo, Neutro y Negativo, para ser conectados a una fuente de continua mayores a 10V, disponibles en el laboratorio.

En la parte superior se encuentran cinco potenciómetros, representando y controlando discretamente cada uno una década distinta (centena de mil, decena de mil, miles, centenas, decenas), y un potenciómetro más, cuyo recorrido es continuo y representa a las unidades, para un ajuste fino.

Se planea agregar un interruptor para manejar frecuencias superiores a 1M, hasta 25M, con la desventaja de una amplitud disminuida.

# Desarrollo

El sistema cuenta con dos módulos, uno correspondiente a la digitalización de los valores de los potenciómetros (conversores A/D), y su posterior procesamiento. Estas tareas las realiza la placa Arduino NANO, cuyas entradas y salidas alcanzan para este fin.

Luego del procesamiento, este dispositivo se comunica con un AD9850, el cual, con los datos recibidos de manera digital, compone una señal sinusoidal de la frecuencia indicada.

A la salida, mediante un capacitor de desacople, se coloca un amplificador operacional en modo seguidor, para incrementar la impedancia de salida, y luego a un atenuador formado por dos resistencias fijas y un potenciómetro, para ajustar la amplitud de salida. Las tres resistencias, en su conjunto, logran traducir la amplitud de 1V fija, a una amplitud variable entre 30mV y 100mV, que utiliza el Qmetro. A la salida del atenuador, se coloca otro seguidor de tensión que cumple la función de buffer de protección en el caso de solicitarle demasiada corriente al sistema.

Las placas son alimentadas con un regulador 7805 y 7905, que reducen la tensión de 10 a 5V.

## AD9850

El integrado AD9850 es un sintetizador de frecuencias digital, más un D/A de alta velocidad con filtrado. La frecuencia se establece mediante una palabra de 32 bits, ingresada en cinco paquetes de 8 bits (los ocho restantes son de control), lo que otorga una resolución de 0,0291 Hz de precisión. La arquitectura permite una frecuencia de salida máxima igual a la mitad de la señal de reloj de referencia (125 MHz / 2 = 62,5 MHz), sin embargo, a partir de los 30 MHz, se introducen armónicos, por lo que la señal de salida sinusoidal será cada vez menos limpia.

## Arduino NANO

El controlador posee 7 entradas analógicas, que son digitalizadas internamente. Las mismas son tomadas y discretizadas (se truncan a 10 valores posibles en todo el rango), obteniendo el dígito correspondiente a la década en cuestión, a excepción del potenciómetro que representa las unidades, el cual posee una gama continua de valores intermedios.

## Notas de Funcionamiento

El proyecto se encuentra funcionando en su primera fase, a ser, con frecuencia máxima de Mhz. El dispositivo cuenta con capacidad para incrementar dicha frecuencia máxima, agregando una funcionalidad accesible con un botón aún no existente en el chasis, que cambia la funcionabilidad de dos de las perillas superiores, la de unidad y decena, para convertirse en unidad de millón y decena de millón, respectivamente. El problema con ésta implementación es la amplitud, la cual decrece considerablemente debido, por un lado, a la falta de correcta adaptación de impedancia entre entrada y salida y al ancho de banda acortado del circuito operacional utilizado de buffer (seguidor de tensión) y de etapa de protección.

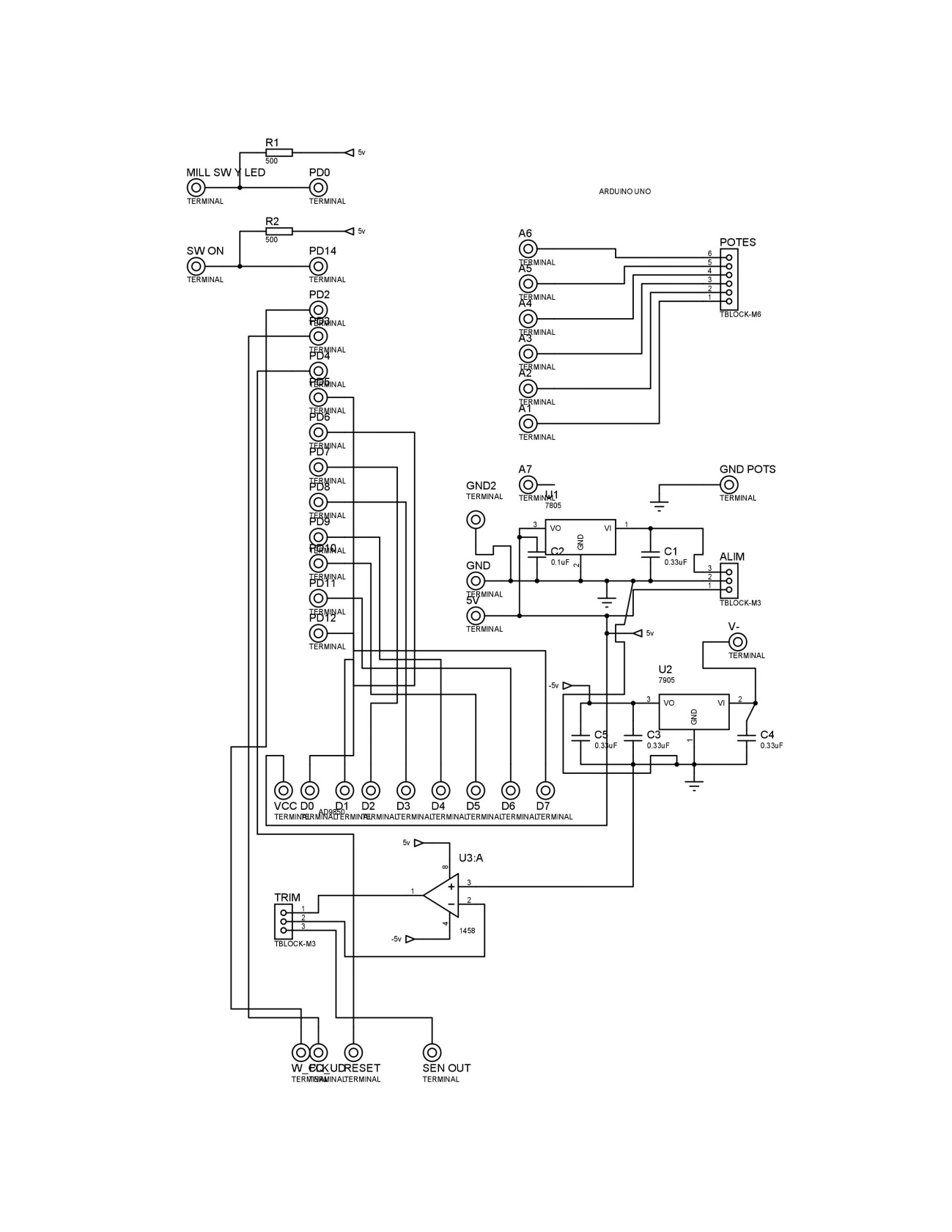
Para aumentar la disponibilidad de frecuencias, se colocó un relé que alterna las resistencias fijas que se colocan al potenciómetro en serie. Estas resistencias cumplen la función de limitar la excursión máxima de amplitud de salida. La bobina del relé es comandado por un transistor funcionando como conmutador, accionado por una corriente de base que provendrá del arduino, ya que posee muchas GPIO (entradas/salidas de propósito general) disponibles. Dichos componentes se encuentran en placa, resta calcular las resistencias que corresponden a los valores de amplitud necesarias.

Originariamente se planeó configurar la placa AD9850 en modo paralelo, pero luego se utilizó sólo un pin como entrada digital, para configurarlo serie. Para que el DDS se inicie en modo serie, se cortocircuita VCC, D0 y D1, y el pin D2 a masa. Esto permitió disponer de más entradas/salidas digitales.

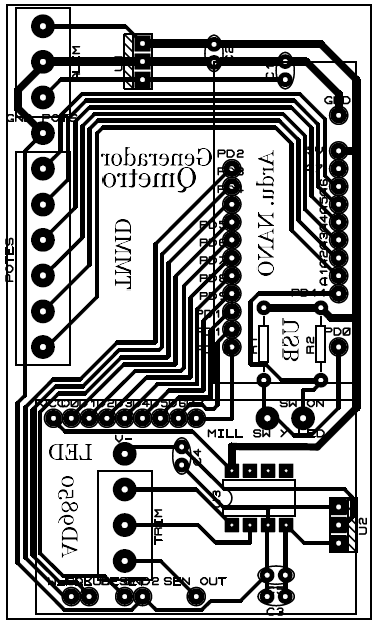
El conexionado de la placa de amplitud se modificó, como se muestra en los diagramas, debido a que inicialmente fue ideado a modo de prueba, y luego reutilizado.

# Circuitos en placa

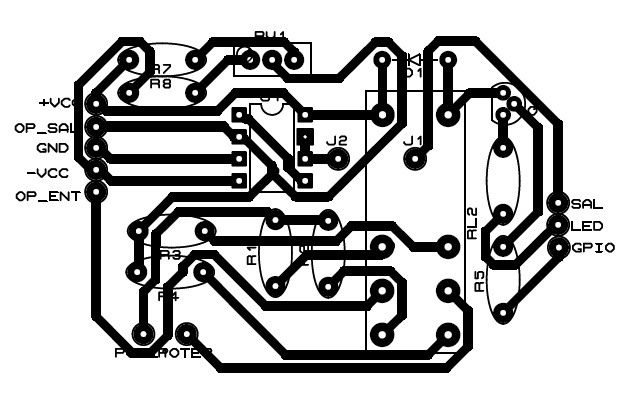
Esquemático Placa base para Arduino y DDS



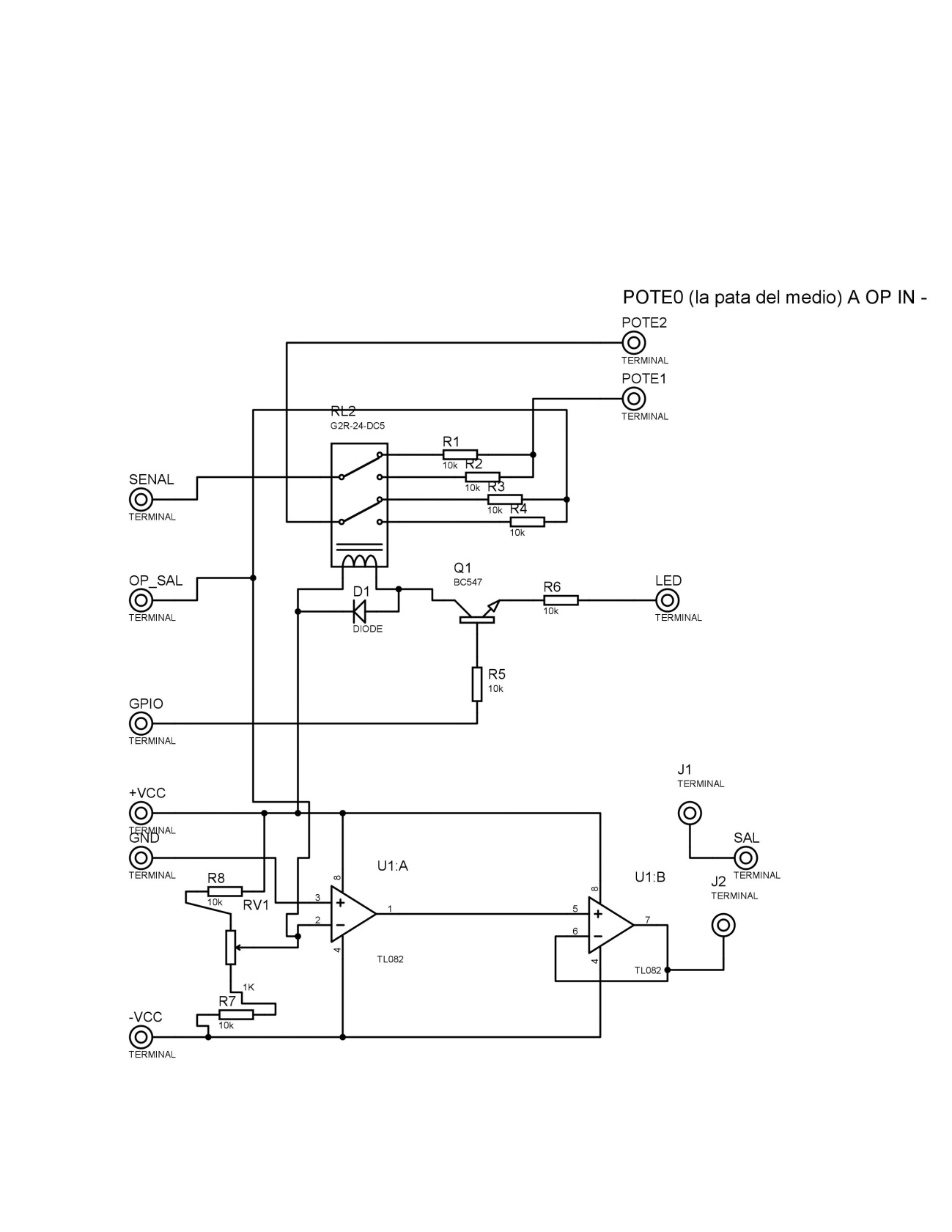
PCB Placa base para Arduino y DDS



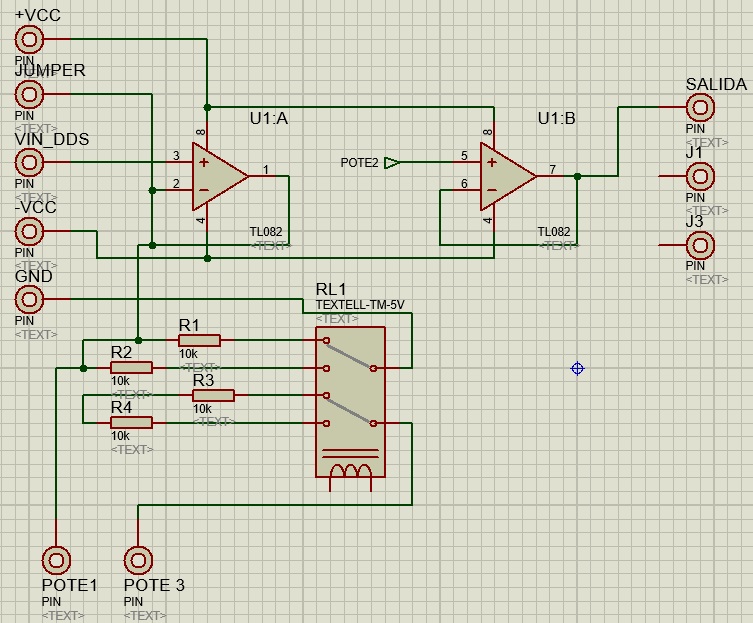
Circuito de manejo de amplitud



Esquemático de placa de amplitud original



Esquemático de placa de amplitud modificado (parte utilizada de la placa)



## Programa Arduino embebido

/\* A través del presente software, se pretende controlar un oscilador digital, con frecuencias del rango

\* de 0 a 25MHz, construido para utilizarse en el Laboratorio de Electrónica, de la facultad UTN de Bahía Blanca.

\*

\* Su realización fue parte de la materia Medidas Electrónicas II, a cargo del profesor Miguel Banchieri y

\* Oscar Rodriguez

\*

\* Autor: Trapaglia Mansilla Matías Daniel

\*/

#include "AD9850.h" //librería de manejo del sintetizador digital (DDS), disponible en https://github.com/F4GOJ/AD9850

int W\_CLK = 2; //coneccionado de pines de control del DDS

int FQ\_UP = 3;

int RESET = 4;

int Dout = 5;

double frec;

double lectura;

int POTE1 = A1, POTE2 = A2, POTE3 = A3, POTE4 = A4, POTE5 = A5, POTE6 = A6; //puerto analógico para cada uno de los //Potenciometros perilla

int retardo = 8 ;

const float resolucion = 0.0291;

bool millon;

enum Posicion { Unidad, Decena, Centena, UMil, DMil, CMil, UMillon, DMillon };

void setup() {

//Serial.begin(9800); // a modo de debug

pinMode(W\_CLK, OUTPUT);

pinMode(FQ\_UP, OUTPUT);

pinMode(RESET, OUTPUT);

pinMode(Dout, OUTPUT);

// pinMode(SW, INPUT); //pensados para el interruptor e indicador de función f>1MHz

// pinMode(LED\_M, OUTPUT);

DDS.begin(W\_CLK,FQ\_UP,Dout,RESET); // inicializamos el hardware DDS

DDS.calibrate(124950000); // calibración fina del cristal del DDS

digitalWrite(W\_CLK, LOW);

digitalWrite(FQ\_UP, LOW);

digitalWrite(RESET, LOW);

}

double convertirValor(Posicion perilla, double valor) {

switch (perilla) {

case Unidad:

valor = valor\*10/1024; break;

case Decena:

if (valor<7) valor=0;

else if (valor<125) valor = 10;

else if (valor<234) valor = 20;

else if (valor<352) valor = 30;

else if (valor<463) valor = 40;

else if (valor<578) valor = 50;

else if (valor<696) valor = 60;

else if (valor<812) valor = 70;

else if (valor<925) valor = 80;

else valor = 90; break;

case Centena:

if (valor<4) valor=0;

else if (valor<103) valor = 100;

else if (valor<215) valor = 200;

else if (valor<336) valor = 300;

else if (valor<462) valor = 400;

else if (valor<598) valor = 500;

else if (valor<724) valor = 600;

else if (valor<842) valor = 700;

else if (valor<942) valor = 800;

else valor = 900; break;

case UMil:

if (valor<2) valor=0;

else if (valor<85) valor = 1000;

else if (valor<212) valor = 2000;

else if (valor<342) valor = 3000;

else if (valor<459) valor = 4000;

else if (valor<571) valor = 5000;

else if (valor<692) valor = 6000;

else if (valor<811) valor = 7000;

else if (valor<933) valor = 8000;

else valor = 9000; break;

case DMil:

if (valor<10) valor=0;

else if (valor<132) valor = 10000;

else if (valor<251) valor = 20000;

else if (valor<377) valor = 30000;

else if (valor<485) valor = 40000;

else if (valor<607) valor = 50000;

else if (valor<731) valor = 60000;

else if (valor<845) valor = 70000;

else if (valor<969) valor = 80000;

else valor = 90000; break;

case CMil:

if (valor<2) valor=0;

else if (valor<98) valor = 100000;

else if (valor<209) valor = 200000;

else if (valor<331) valor = 300000;

else if (valor<449) valor = 400000;

else if (valor<586) valor = 500000;

else if (valor<705) valor = 600000;

else if (valor<818) valor = 700000;

else if (valor<934) valor = 800000;

else valor = 900000; break;

case UMillon:

if (valor<3) valor=0;

else if (valor<96) valor = 1000000;

else if (valor<209) valor = 2000000;

else if (valor<322) valor = 3000000;

else if (valor<440) valor = 4000000;

else if (valor<566) valor = 5000000;

else if (valor<692) valor = 6000000;

else if (valor<805) valor = 7000000;

else if (valor<928) valor = 8000000;

else valor = 9000000; break;

case DMillon:

if (valor<7) valor=0;

else if (valor<125) valor = 10000000;

else if (valor<234) valor = 20000000;

else if (valor<352) valor = 30000000;

else if (valor<463) valor = 40000000;

else if (valor<578) valor = 50000000;

else if (valor<696) valor = 60000000;

else if (valor<812) valor = 70000000;

else if (valor<925) valor = 80000000;

else valor = 90000000; break;

}

return (valor);

}

void loop() {

//millon = digitalRead(SW); // leo el estado del interruptor frontal de Millones

//digitalWrite(LED\_M, millon); // paso dicho parámetro al led y a los relays

millon = LOW;

delay(retardo);

lectura = analogRead(POTE1);

//lectura = constrain(lectura, 10, 1010);

if(millon == HIGH)

frec = convertirValor(UMillon, lectura);

else frec = convertirValor(Unidad, lectura);

//Serial.print("U: ");Serial.print(lectura); Serial.print("\t"); //salidas para debug

delay(retardo);

lectura = analogRead(POTE2);

if(millon == HIGH)

frec += convertirValor(DMillon, lectura);

else frec += convertirValor(Decena, lectura);

//Serial.print("D: ");Serial.print(lectura); Serial.print("\t"); //salidas para debug

delay(retardo);

lectura = analogRead(POTE3);

frec += convertirValor(Centena, lectura);

//Serial.print("C: ");Serial.print(lectura); Serial.print("\t"); //salidas para debug

delay(retardo);

lectura = analogRead(POTE6);

frec += convertirValor(UMil, lectura);

//Serial.print("UM: ");Serial.print(lectura); Serial.print("\t"); //salidas para debug

delay(retardo);

lectura = analogRead(POTE5);

frec += convertirValor(DMil, lectura);

//Serial.print("DM: ");Serial.print(lectura); Serial.print("\t"); //salidas para debug

delay(retardo);

lectura = analogRead(POTE4);

frec += convertirValor(CMil, lectura);

//Serial.print("CM: ");Serial.print(lectura); Serial.println("\t"); //salidas para debug

DDS.setfreq(frec,0); // paso al DDS el valor acumulado de frecuencia.

delay(80);

}