Informe de Laboratorio - Proyecto Osciloscopio y Generador de señales

Diana Cristina Castillo Motta
100618010215
Carlos Ernesto Pitre Prieto
100618010459
Jeison Hair Patiño Cuasapud
100618011326

DOCENTE:
FABIO HERNAN REALPE MARTINEZ

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Popayán, Agosto de 2022

INTRODUCCIÓN

En el presente informe se describe el proceso de desarrollo e implementación del proyecto denominado “Osciloscopio y generador de señales”, seleccionado como proyecto de curso para la asignatura de Laboratorio II de Electrónica.

El objetivo del proyecto es el desarrollo de un dispositivo de bajo costo y con pocos componentes, utilizando la plataforma de arduino, que implemente las funcionalidades básicas de un osciloscopio y de un generador de señales. Con esto, se tiene una herramienta portable y análoga a los equipos de laboratorio, la cual puede ser útil para la lectura y generación de señales en aplicaciones de bajos requerimientos.

Inicialmente se realizó un cronograma de actividades, donde se definen las diferentes etapas necesarias para llevar a cabo satisfactoriamente el desarrollo del proyecto, indicando el número de semanas que cada tarea requiere. Para cada actividad, se describen los avances realizados.

Al final del desarrollo se obtiene un prototipo funcional del dispositivo, sobre el cual se realizan diferentes pruebas para comprobar su funcionamiento.

DESCRIPCIÓN DEL PROYECTO

Como objetivo del proyecto se plantea el desarrollo de un osciloscopio y generador de señales de voltaje en un rango limitado de operación, alrededor de 5 voltios, que es un estándar ampliamente utilizado, por ejemplo en tecnologías como TTL.

Respecto al osciloscopio, se desea poder visualizar cualquier tipo de señal, además de poder obtener distintos parámetros asociados, como voltaje pico a pico, voltaje pico y voltaje rms.

Por otra parte, para el circuito generador de señales se desea poder generar diferentes tipos de señales de uso común (cuadradas, rectangulares, senoidales) y poder modificar sus parámetros de amplitud y frecuencia mediante el uso de una interfaz gráfica de control.

Para el desarrollo del proyecto se elige la plataforma de arduino, dado que es ampliamente utilizada, principalmente por estudiantes, para el aprendizaje de conceptos básicos en el

área de la electrónica, además que sus dispositivos son muy populares, de bajo costo y fácil acceso.

Los códigos desarrollados a lo largo de todo el proyecto están consignados en https://github.com/PitreC1/Proyecto_Osc_Gen.

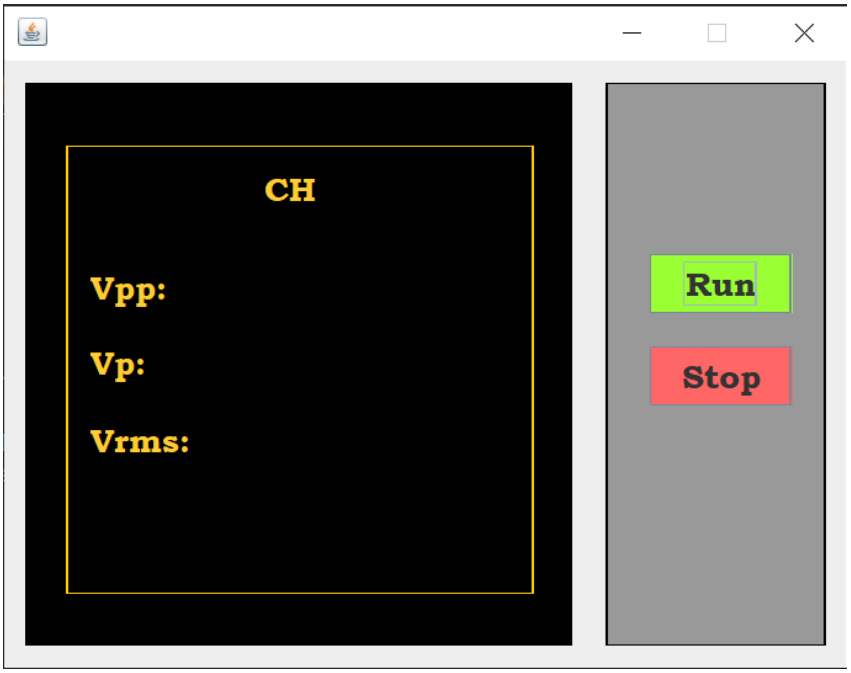
CRONOGRAMA DE ACTIVIDADES

Actividades	Abril					Mayo					Junio			
	1	2	3	4	5	1	2	3	4	5	1	2	3	4
Planificación del proyecto														
Rastreo fundamento teórico														
Establecimiento Interfaz gráfica														
Establecimiento y validación del circuito oscilador														
Simulación del circuito oscilador														
Implementación del circuito														
Pruebas y correcciones del circuito														
Establecimiento y validación del circuito generador														
Simulación del circuito generador														
Implementación del circuito														
Pruebas y correcciones del circuito														
Interconexión con interfaz gráfica														
Pruebas y correcciones del circuito mejorado														
Entrega final del proyecto														

ESTABLECIMIENTO INTERFAZ GRÁFICA:

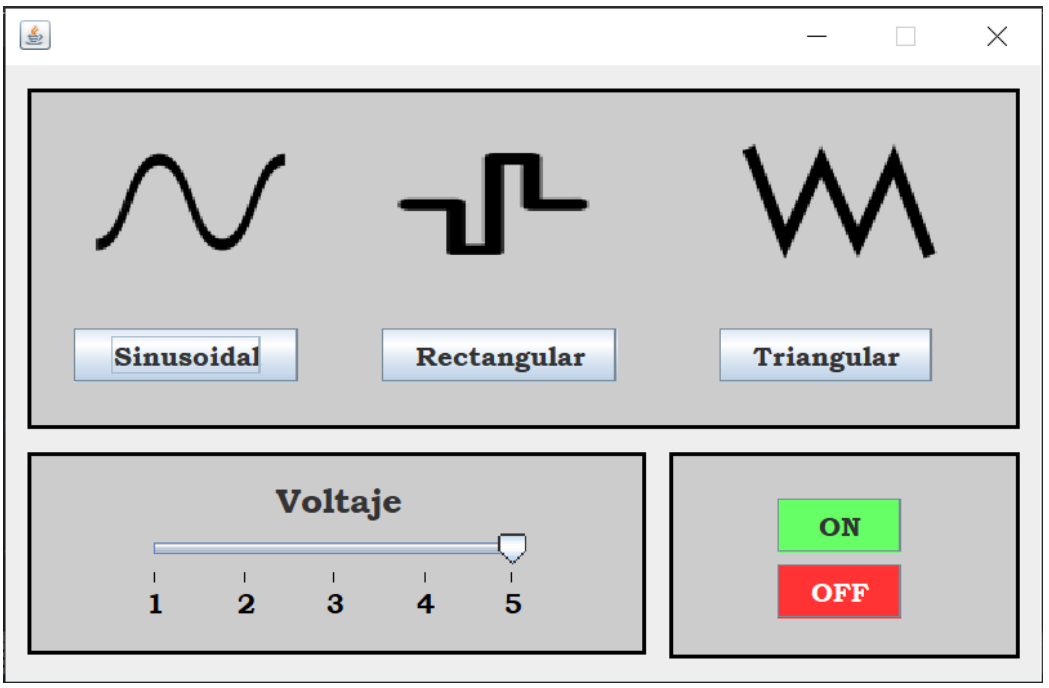
Se diseñaron 2 interfaces gráficas para el proyecto, esto es una para el osciloscopio y otra para el generador de señales. Las interfaces gráficas se desarrollaron en lenguaje Java, el código de las mismas se puede encontrar en el github del proyecto.

INTERFAZ OSCILOSCOPIO



Al conectar Arduino mediante puerto serial al computador, se inicia la interfaz gráfica en Java y cuando se da click en el botón Run se muestran los valores de voltaje que está leyendo el arduino, adicionalmente se da una representación de la señal en la pantalla OLED. La funcionalidad del botón Stop es detener esta lectura de datos lo cual permite observar la forma de onda en ese instante en la pantalla OLED y en la interfaz gráfica el valor leído con mayor precisión. En este caso se tiene un envío y recepción de datos entre la interfaz y el arduino.

INTERFAZ GENERADOR



La interfaz del generador permite cambiar entre las diferentes formas de ondas disponibles y el valor de voltaje que deseemos. En un principio, se selecciona una de las tres formas de onda y se coloca el Slider en el valor de voltaje deseado, luego cuando se da click en el botón ON se envían los datos de selección al arduino y se genera la onda como hemos especificado en la interfaz gráfica. El botón OFF permite colocar la salida del generador en 0V, de manera que tenemos control y evitamos dañar innecesariamente algún circuito con el que se experimente.

ESTABLECIMIENTO Y VALIDACIÓN DEL CIRCUITO OSCILOSCOPIO:

En esta etapa se establece el circuito del osciloscopio, se hace uso del Arduino Mega en este debido a su mayor capacidad y la posibilidad de usar diferentes pines Seriales indispensables para el funcionamiento del Osciloscopio, esto ya que, se tiene una pantalla OLED de 128x64 la cual funciona con el bus I2C. Para la lectura de señales provenientes de otros circuitos se implementó un circuito divisor de tensión.

Al realizar el circuito se conectan los pines del arduino mega de la siguiente forma:

OLED 128x64	ARDUINO MEGA
SCL	SCL

SDA	SDA
VCC	5V
GND	GND

Para lograr la lectura de una señal de otro circuito se realiza adicionalmente la conexión a través del pin A0 del arduino mega.

Implementación del Circuito Oscilador:

Código de arduino:

```
#include <Adafruit_SSD1306.h>  // incluye la libreria de adafruit para el driver del display OLED
#include <Adafruit_GFX.h>  // incluye la libreria de adafruit para gestionar los efectos graficos de manera facil
Adafruit_SSD1306 display(128, 64);  // declara la resolucion del display



int estado = 0;
int valorAnalogico = 0;
int x[128];              //buffer de la grafica
int y[128];              //buffer secundario de la grafica
char estadoChar = '0';     //Maneja interrupciones Puerto Serie
float graficaVoltaje = 0;



void funcion1() {

  if (estado == 0) {
    estado = 1;
  } else {
    estado = 0;
  }
}

void setup() {
```

```
  Serial.begin(9600);
   display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // inicia la comunicacion I2C con el display que tiene la direccion 0x3C
  display.setRotation(0);  // se escoje la orientacion del display puede ser 0 o 2
  display.dim(true); //dejamos el brillo en maximo
  display.setTextColor(WHITE);
  display.setTextSize(1);  // ajusta el tamaño de texto en el minimo valor

  //llenamos las matrices con un valor fuera del rango de medicion >1023
  for (int i = 127; i >= 0; i--) {
    x[i] = 9999;
  }
  for (int i = 127; i >= 0; i--) {
    y[i] = 9999;
  }

  attachInterrupt(digitalPinToInterrupt(2), funcion1, CHANGE);
}


void loop() {

  if (estado == 0) {

    display.clearDisplay(); //limpia el buffer del display

    //dibuja escala
    display.setCursor(0, 0);
    display.print(F("5V"));
    display.setCursor(0, 11);
    display.print(F("4V"));
    display.setCursor(0, 22);
    display.print(F("3V"));
    display.setCursor(0, 32);
    display.print(F("2V"));
    display.setCursor(0, 43);
    display.print(F("1V"));
```

```
    display.drawLine(15, 0, 25, 0, WHITE);
    display.drawLine(15, 11, 25, 11, WHITE);
    display.drawLine(15, 22, 25, 22, WHITE);
    display.drawLine(15, 32, 25, 32, WHITE);
    display.drawLine(15, 43, 25, 43, WHITE);
    //dibuja eje X y Y
    display.drawLine(0, 53, 127, 53, WHITE);
    display.drawLine(25, 53, 25, 0, WHITE);

    valorAnalogico = analogRead(A0); //lee el valor analogico del pin A0
    Serial.println(valorAnalogico);
    delay(100);

     graficaVoltaje = map(valorAnalogico, 0, 1023, 53, 0); //escala el valor analogico a un pixel
imprimible en pantalla

    x[127] = graficaVoltaje; //asigna el valor escalado a el ultimo dato de la matriz

    for (int i = 127; i >= 25; i--) {
      display.drawPixel(i, x[i], WHITE); //dibuja punto a punto el contenido de x
      y[i - 1] = x[i]; //guarda la informacion desplazada una posicion temporalmente en y
    }

    display.display(); //despliega la informacion del buffer en la pantalla

    for (int i = 127; i >= 0; i--) {
      x[i] = y[i]; //envia los datos desplazados de vuelta a la variable x
    }
  }
}

void serialEvent() {

  estadoChar = Serial.read();

  if (estadoChar == '1') {
```

```
    digitalWrite(2, HIGH);
  }if (estadoChar == '0'){
    digitalWrite(2, LOW);
  }
}
```

ESTABLECIMIENTO Y VALIDACIÓN DEL CIRCUITO GENERADOR:

En esta etapa se inicia el establecimiento del circuito generador de señales. Dado que las placas arduino convencionales no son capaces de generar por sí mismas señales de voltaje analógicas, se opta por utilizar un conversor digital a analógico (DAC) el cual es controlado por las salidas digitales de la placa.

Específicamente, se elige el módulo MCP4725. Este módulo es un DAC de bajo consumo y alta precisión, el cual posee las siguientes características:

- Interfaz de comunicación mediante el protocolo I2C
- 12 bits de resolución
- Voltaje de operación: 2.7V - 5V
- Tiempo de establecimiento: 6μs
- Dimensiones: 14.9 mm x 15.2 mm.

Para el control de este dispositivo desde la plataforma de arduino, se utiliza la librería "*Adafruit_MCP4725*". Esta librería permite operar fácilmente con el módulo dentro del código de arduino mediante la creación de un objeto de tipo *Adafruit_MCP4725;* inicializando el objeto en el setup mediante la función *begin* pasando como parámetro la dirección correspondiente con el modelo del DAC (en este caso para el modelo MCP4725A0 se pasa la dirección 0x60); y finalmente invocando la función *setVoltage,* que recibe como parámetro un entero entre 0 y 4095, donde 0 corresponde a 0 voltios, 4095 corresponde al voltaje de alimentación, y los valores intermedios corresponden a un voltaje intermedio cuyo valor se puede calcular mediante la expresión:

$$V_o = \frac{V_{supply} * n}{4095}$$

Para la generación de señales se recorre una *Lookup Table*, que son estructuras de datos -en este caso vectores- que almacenan los diferentes valores que adquiere la señal

deseada durante un periodo; y durante cada iteración se invoca la función *setVoltage* para generar el voltaje correspondiente. La amplitud de la señal generada depende del valor máximo que contenga la Lookup Table, la forma de la señal depende de cómo están distribuidos los datos a lo largo del vector, y la frecuencia depende del número de datos que contiene.

De esta forma, se entiende que para variar los parámetros de la señal generada, es necesario cambiar la Lookup Table que se esté leyendo durante la ejecución del programa. Esta funcionalidad se realiza desde la interfaz gráfica que controla el circuito generador de señales.

SIMULACIÓN DEL CIRCUITO GENERADOR:

Para simular el funcionamiento del circuito generador de señales, se utiliza Proteus y la librería *Simulino* que permite utilizar algunas placas de arduino y cargar código sobre ellas. También se utiliza el módulo MCP4725 que incluye el programa, y un módulo de osciloscopio. Se simula utilizando una placa Arduino UNO, de modo que las conexiones entre el DAC y la placa son las siguientes:

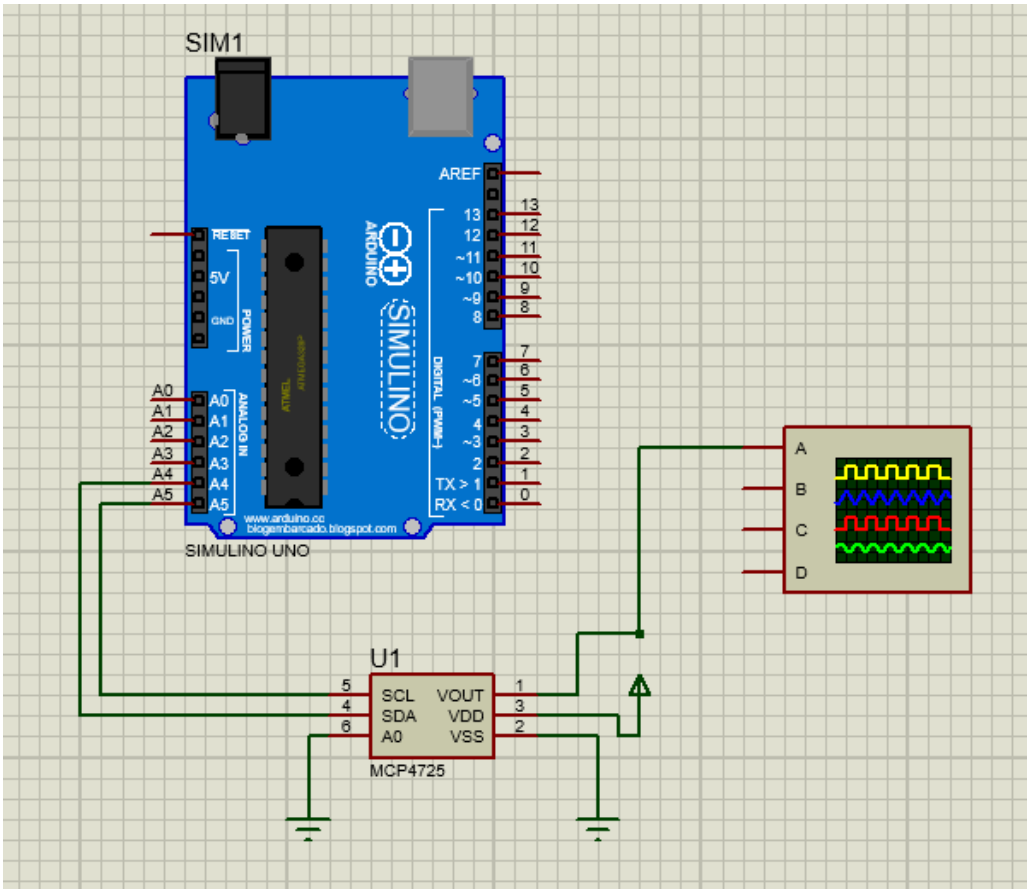
MCP4725	ARDUINO UNO
SCL	A5
SDA	A4
A0	GND
VDD	Vcc
VSS	GND
VOUT	CARGA

Código de arduino:

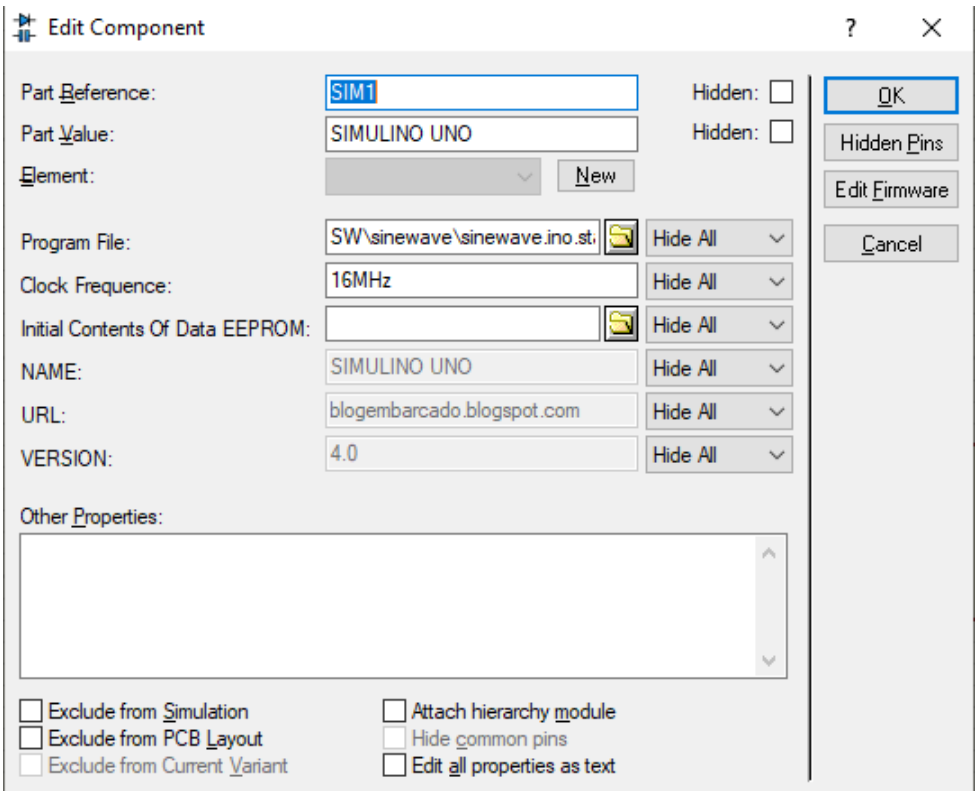
```
#include <Wire.h>
#include <Adafruit_MCP4725.h>

Adafruit_MCP4725 dac;

#define DAC_RESOLUTION    (6)
```

```
const PROGMEM uint16_t DACLookup_FullSine_6Bit[64] =
{
  2048, 2248, 2447, 2642, 2831, 3013, 3185, 3346,
  3495, 3630, 3750, 3853, 3939, 4007, 4056, 4085,
  4095, 4085, 4056, 4007, 3939, 3853, 3750, 3630,
  3495, 3346, 3185, 3013, 2831, 2642, 2447, 2248,
  2048, 1847, 1648, 1453, 1264, 1082,  910,  749,
   600,  465,  345,  242,  156,   88,   39,   10,
     0,   10,   39,   88,  156,  242,  345,  465,
   600,  749,  910, 1082, 1264, 1453, 1648, 1847
};

void setup(void) {
  dac.begin(0x60);
}

void loop(void) {
    uint16_t i;
for (i = 0; i < 64; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullSine_6Bit[i])), false);
      }
}
```

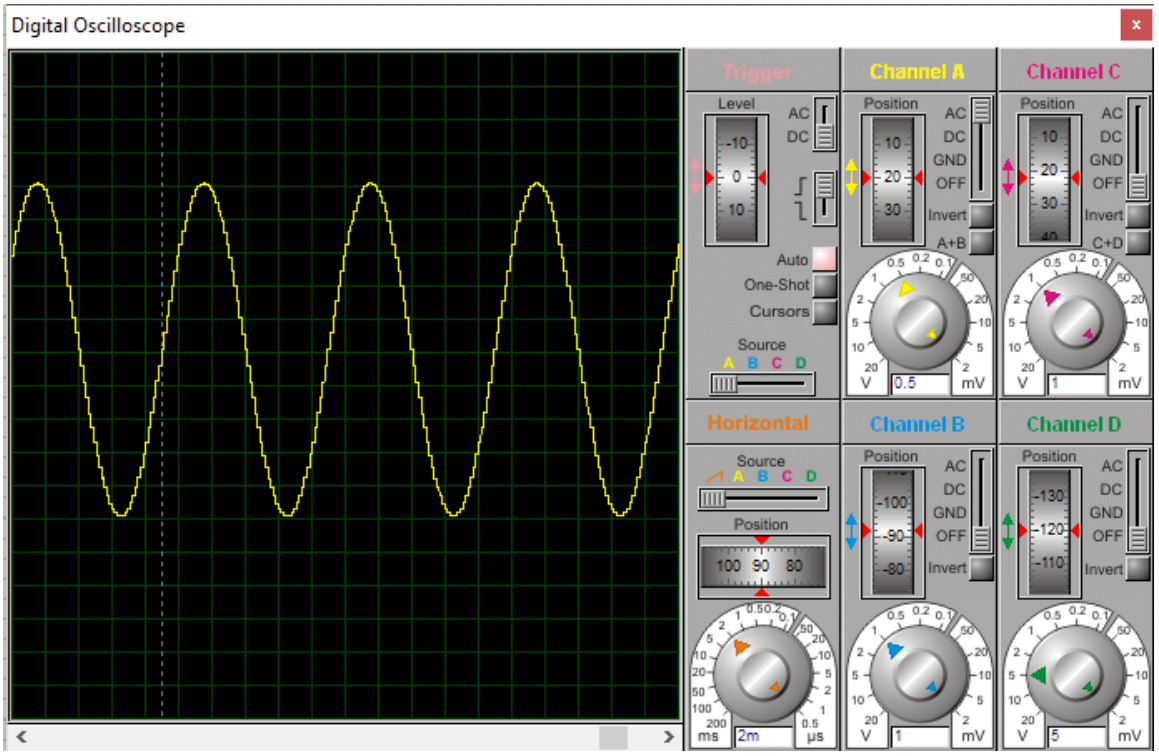
Esquemático de la simulación:



Configuración de la placa arduino:



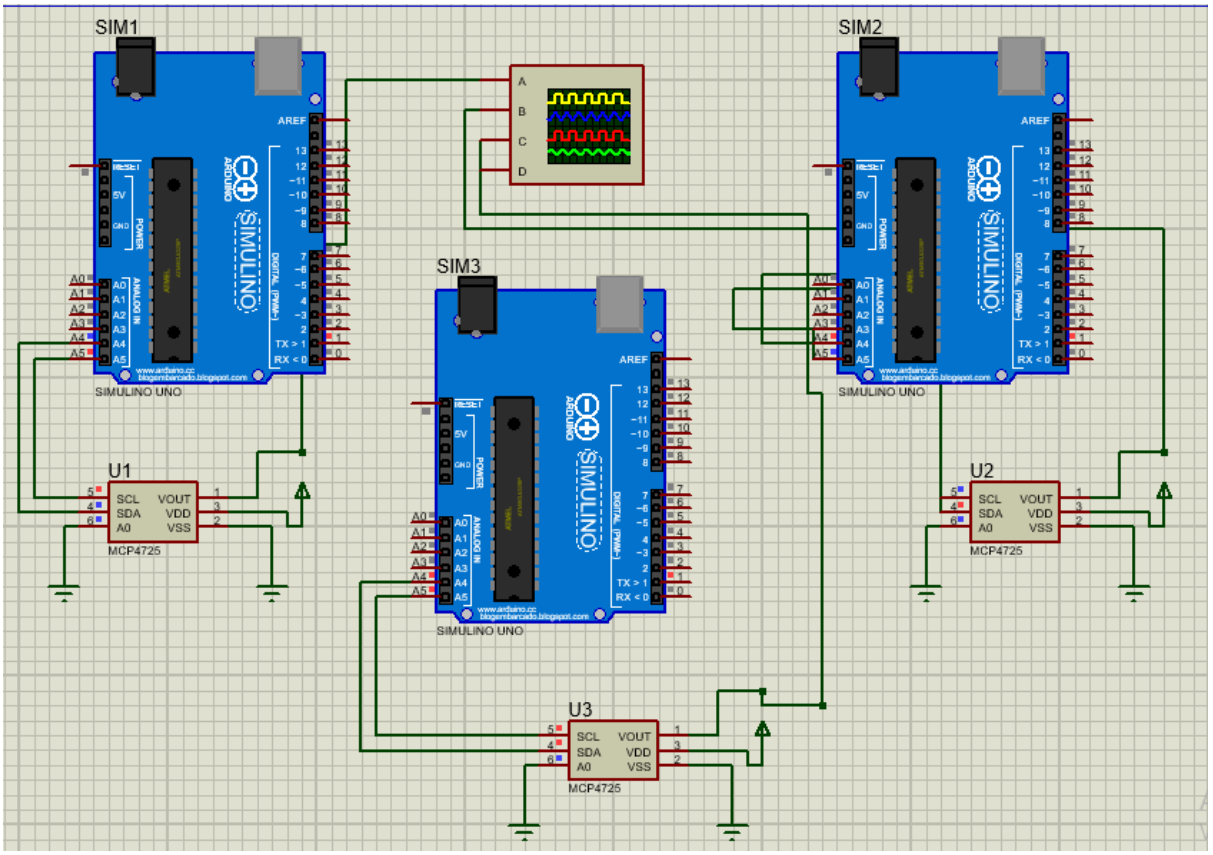
Ejecución de la simulación:



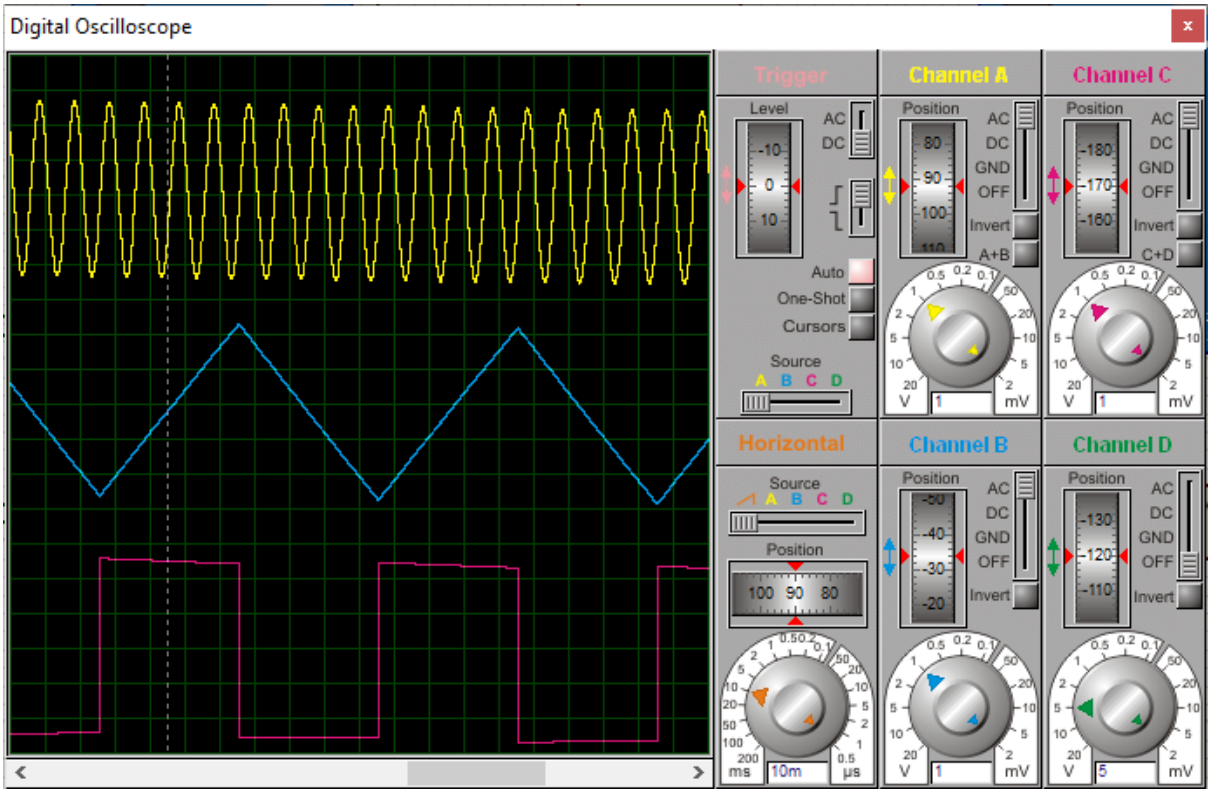
Se observa como a la salida del DAC se obtiene una señal senoidal con un periodo de aproximadamente 10ms y amplitud de 5 voltios.

Luego, se comprueba que es posible generar diferentes señales mediante la lectura de diferentes *Lookup Table* en la ejecución del programa. Para esto se simulan varias placas arduino, cada una cargada con un código diferente donde varía la Lookup Table leída.

Esquemático simulación de distintas señales:



Resultado de la simulación:



Se observa como cada placa simulada genera un tipo de señal diferente en su forma, amplitud y frecuencia.

IMPLEMENTACIÓN CIRCUITO GENERADOR:

Al emplear el DAC en las pruebas de funcionamiento realizadas en el laboratorio se tienen las siguientes conexiones

MCP4725	ARDUINO UNO
SCL	A5
SDA	A4
GND	GND
VDD	Vcc
GND	GND
VOUT	CARGA

Dado que en este punto se realizaron pruebas con la interfaz gráfica y se dejó implementada la función que permite desde está modificar entre diferentes valores de amplitud preestablecidos, se cargó al Arduino Uno el siguiente código:

```
#include <Wire.h>
#include <Adafruit_MCP4725.h>

Adafruit_MCP4725 dac;
char estadoChar;
int estado = 0;
char amp;

// Cambiar para ajustar la resolucion
//int DAC_RESOLUTION = 9;
int amplitud = 1;//valores de 1-5V

/* Note: If flash space is tight a quarter sine wave is enough
   to generate full sine and cos waves, but some additional
```

```
   calculation will be required at each step after the first
   quarter wave.                                        */
//nota uint16_t es como una medida en bytes
const PROGMEM uint16_t DACLookup_FullSine5V_9Bit[512] =
{
  2048, 2073, 2098, 2123, 2148, 2174, 2199, 2224,
  2249, 2274, 2299, 2324, 2349, 2373, 2398, 2423,
  2448, 2472, 2497, 2521, 2546, 2570, 2594, 2618,
  2643, 2667, 2690, 2714, 2738, 2762, 2785, 2808,
  2832, 2855, 2878, 2901, 2924, 2946, 2969, 2991,
  3013, 3036, 3057, 3079, 3101, 3122, 3144, 3165,
  3186, 3207, 3227, 3248, 3268, 3288, 3308, 3328,
  3347, 3367, 3386, 3405, 3423, 3442, 3460, 3478,
  3496, 3514, 3531, 3548, 3565, 3582, 3599, 3615,
  3631, 3647, 3663, 3678, 3693, 3708, 3722, 3737,
  3751, 3765, 3778, 3792, 3805, 3817, 3830, 3842,
  3854, 3866, 3877, 3888, 3899, 3910, 3920, 3930,
  3940, 3950, 3959, 3968, 3976, 3985, 3993, 4000,
  4008, 4015, 4022, 4028, 4035, 4041, 4046, 4052,
  4057, 4061, 4066, 4070, 4074, 4077, 4081, 4084,
  4086, 4088, 4090, 4092, 4094, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4094, 4092, 4090, 4088,
  4086, 4084, 4081, 4077, 4074, 4070, 4066, 4061,
  4057, 4052, 4046, 4041, 4035, 4028, 4022, 4015,
  4008, 4000, 3993, 3985, 3976, 3968, 3959, 3950,
  3940, 3930, 3920, 3910, 3899, 3888, 3877, 3866,
  3854, 3842, 3830, 3817, 3805, 3792, 3778, 3765,
  3751, 3737, 3722, 3708, 3693, 3678, 3663, 3647,
  3631, 3615, 3599, 3582, 3565, 3548, 3531, 3514,
  3496, 3478, 3460, 3442, 3423, 3405, 3386, 3367,
  3347, 3328, 3308, 3288, 3268, 3248, 3227, 3207,
  3186, 3165, 3144, 3122, 3101, 3079, 3057, 3036,
  3013, 2991, 2969, 2946, 2924, 2901, 2878, 2855,
  2832, 2808, 2785, 2762, 2738, 2714, 2690, 2667,
  2643, 2618, 2594, 2570, 2546, 2521, 2497, 2472,
  2448, 2423, 2398, 2373, 2349, 2324, 2299, 2274,
  2249, 2224, 2199, 2174, 2148, 2123, 2098, 2073,
```

```
  2048, 2023, 1998, 1973, 1948, 1922, 1897, 1872,
  1847, 1822, 1797, 1772, 1747, 1723, 1698, 1673,
  1648, 1624, 1599, 1575, 1550, 1526, 1502, 1478,
  1453, 1429, 1406, 1382, 1358, 1334, 1311, 1288,
  1264, 1241, 1218, 1195, 1172, 1150, 1127, 1105,
  1083, 1060, 1039, 1017,  995,  974,  952,  931,
  910,  889,  869,  848,  828,  808,  788,  768,
  749,  729,  710,  691,  673,  654,  636,  618,
  600,  582,  565,  548,  531,  514,  497,  481,
  465,  449,  433,  418,  403,  388,  374,  359,
  345,  331,  318,  304,  291,  279,  266,  254,
  242,  230,  219,  208,  197,  186,  176,  166,
  156,  146,  137,  128,  120,  111,  103,   96,
  88,   81,   74,   68,   61,   55,   50,   44,
  39,   35,   30,   26,   22,   19,   15,   12,
  10,    8,    6,    4,    2,    1,    1,    0,
  0,    0,    1,    1,    2,    4,    6,    8,
  10,   12,   15,   19,   22,   26,   30,   35,
  39,   44,   50,   55,   61,   68,   74,   81,
  88,   96,  103,  111,  120,  128,  137,  146,
  156,  166,  176,  186,  197,  208,  219,  230,
  242,  254,  266,  279,  291,  304,  318,  331,
  345,  359,  374,  388,  403,  418,  433,  449,
  465,  481,  497,  514,  531,  548,  565,  582,
  600,  618,  636,  654,  673,  691,  710,  729,
  749,  768,  788,  808,  828,  848,  869,  889,
  910,  931,  952,  974,  995, 1017, 1039, 1060,
  1083, 1105, 1127, 1150, 1172, 1195, 1218, 1241,
  1264, 1288, 1311, 1334, 1358, 1382, 1406, 1429,
  1453, 1478, 1502, 1526, 1550, 1575, 1599, 1624,
  1648, 1673, 1698, 1723, 1747, 1772, 1797, 1822,
  1847, 1872, 1897, 1922, 1948, 1973, 1998, 2023
};

const PROGMEM uint16_t DACLookup_FullSine4V_9Bit[512] {
  1638, 1658, 1678, 1698, 1718, 1738, 1758, 1779,
  1799, 1819, 1839, 1858, 1878, 1898, 1918, 1938,
```

1958, 1977, 1997, 2016, 2036, 2055, 2075, 2094,
2113, 2133, 2152, 2171, 2190, 2209, 2228, 2246,
2265, 2283, 2302, 2320, 2338, 2356, 2374, 2392,
2410, 2428, 2445, 2463, 2480, 2497, 2514, 2531,
2548, 2565, 2581, 2598, 2614, 2630, 2646, 2662,
2677, 2693, 2708, 2723, 2738, 2753, 2767, 2782,
2796, 2810, 2824, 2838, 2852, 2865, 2878, 2891,
2904, 2917, 2929, 2942, 2954, 2966, 2977, 2989,
3000, 3011, 3022, 3033, 3043, 3053, 3063, 3073,
3083, 3092, 3101, 3110, 3119, 3127, 3135, 3144,
3151, 3159, 3166, 3173, 3180, 3187, 3193, 3200,
3205, 3211, 3217, 3222, 3227, 3232, 3236, 3240,
3245, 3248, 3252, 3255, 3258, 3261, 3264, 3266,
3268, 3270, 3272, 3273, 3274, 3275, 3276, 3276,
3276, 3276, 3276, 3275, 3274, 3273, 3272, 3270,
3268, 3266, 3264, 3261, 3258, 3255, 3252, 3248,
3245, 3240, 3236, 3232, 3227, 3222, 3217, 3211,
3205, 3200, 3193, 3187, 3180, 3173, 3166, 3159,
3151, 3144, 3135, 3127, 3119, 3110, 3101, 3092,
3083, 3073, 3063, 3053, 3043, 3033, 3022, 3011,
3000, 2989, 2977, 2966, 2954, 2942, 2929, 2917,
2904, 2891, 2878, 2865, 2852, 2838, 2824, 2810,
2796, 2782, 2767, 2753, 2738, 2723, 2708, 2693,
2677, 2662, 2646, 2630, 2614, 2598, 2581, 2565,
2548, 2531, 2514, 2497, 2480, 2463, 2445, 2428,
2410, 2392, 2374, 2356, 2338, 2320, 2302, 2283,
2265, 2246, 2228, 2209, 2190, 2171, 2152, 2133,
2113, 2094, 2075, 2055, 2036, 2016, 1997, 1977,
1958, 1938, 1918, 1898, 1878, 1858, 1839, 1819,
1799, 1779, 1758, 1738, 1718, 1698, 1678, 1658,
1638, 1618, 1598, 1578, 1558, 1538, 1518, 1497,
1477, 1457, 1437, 1418, 1398, 1378, 1358, 1338,
1318, 1299, 1279, 1260, 1240, 1221, 1201, 1182,
1163, 1143, 1124, 1105, 1086, 1067, 1048, 1030,
1011, 993, 974, 956, 938, 920, 902, 884,
866, 848, 831, 813, 796, 779, 762, 745,
728, 711, 695, 678, 662, 646, 630, 614,

```
  599, 583, 568, 553, 538, 523, 509, 494,
  480, 466, 452, 438, 424, 411, 398, 385,
  372, 359, 347, 334, 322, 310, 299, 287,
  276, 265, 254, 243, 233, 223, 213, 203,
  193, 184, 175, 166, 157, 149, 141, 132,
  125, 117, 110, 103, 96, 89, 83, 76,
  71, 65, 59, 54, 49, 44, 40, 36,
  31, 28, 24, 21, 18, 15, 12, 10,
  8, 6, 4, 3, 2, 1, 0, 0,
  0, 0, 0, 1, 2, 3, 4, 6,
  8, 10, 12, 15, 18, 21, 24, 28,
  31, 36, 40, 44, 49, 54, 59, 65,
  71, 76, 83, 89, 96, 103, 110, 117,
  125, 132, 141, 149, 157, 166, 175, 184,
  193, 203, 213, 223, 233, 243, 254, 265,
  276, 287, 299, 310, 322, 334, 347, 359,
  372, 385, 398, 411, 424, 438, 452, 466,
  480, 494, 509, 523, 538, 553, 568, 583,
  599, 614, 630, 646, 662, 678, 695, 711,
  728, 745, 762, 779, 796, 813, 831, 848,
  866, 884, 902, 920, 938, 956, 974, 993,
  1011, 1030, 1048, 1067, 1086, 1105, 1124, 1143,
  1163, 1182, 1201, 1221, 1240, 1260, 1279, 1299,
  1318, 1338, 1358, 1378, 1398, 1418, 1437, 1457,
  1477, 1497, 1518, 1538, 1558, 1578, 1598, 1618
};

const PROGMEM uint16_t DACLookup_FullSine3V_9Bit[512] {
  1229, 1244, 1259, 1274, 1289, 1304, 1319, 1334,
  1349, 1364, 1379, 1394, 1409, 1424, 1439, 1454,
  1469, 1484, 1498, 1513, 1528, 1542, 1557, 1571,
  1586, 1600, 1615, 1629, 1643, 1657, 1671, 1685,
  1699, 1713, 1727, 1741, 1754, 1768, 1782, 1795,
  1808, 1822, 1835, 1848, 1861, 1874, 1887, 1899,
  1912, 1924, 1937, 1949, 1961, 1973, 1985, 1997,
  2009, 2020, 2032, 2043, 2054, 2065, 2076, 2087,
  2098, 2109, 2119, 2129, 2140, 2150, 2160, 2169,
```

2179, 2189, 2198, 2207, 2216, 2225, 2234, 2242,
2251, 2259, 2267, 2275, 2283, 2291, 2298, 2306,
2313, 2320, 2327, 2333, 2340, 2346, 2353, 2359,
2364, 2370, 2376, 2381, 2386, 2391, 2396, 2401,
2405, 2409, 2413, 2417, 2421, 2425, 2428, 2431,
2434, 2437, 2440, 2442, 2445, 2447, 2449, 2451,
2452, 2453, 2455, 2456, 2457, 2457, 2458, 2458,
2458, 2458, 2458, 2457, 2457, 2456, 2455, 2453,
2452, 2451, 2449, 2447, 2445, 2442, 2440, 2437,
2434, 2431, 2428, 2425, 2421, 2417, 2413, 2409,
2405, 2401, 2396, 2391, 2386, 2381, 2376, 2370,
2364, 2359, 2353, 2346, 2340, 2333, 2327, 2320,
2313, 2306, 2298, 2291, 2283, 2275, 2267, 2259,
2251, 2242, 2234, 2225, 2216, 2207, 2198, 2189,
2179, 2169, 2160, 2150, 2140, 2129, 2119, 2109,
2098, 2087, 2076, 2065, 2054, 2043, 2032, 2020,
2009, 1997, 1985, 1973, 1961, 1949, 1937, 1924,
1912, 1899, 1887, 1874, 1861, 1848, 1835, 1822,
1808, 1795, 1782, 1768, 1754, 1741, 1727, 1713,
1699, 1685, 1671, 1657, 1643, 1629, 1615, 1600,
1586, 1571, 1557, 1542, 1528, 1513, 1498, 1484,
1469, 1454, 1439, 1424, 1409, 1394, 1379, 1364,
1349, 1334, 1319, 1304, 1289, 1274, 1259, 1244,
1229, 1214, 1199, 1184, 1169, 1154, 1139, 1124,
1109, 1094, 1079, 1064, 1049, 1034, 1019, 1004,
989, 974, 960, 945, 930, 916, 901, 887,
872, 858, 843, 829, 815, 801, 787, 773,
759, 745, 731, 717, 704, 690, 676, 663,
650, 636, 623, 610, 597, 584, 571, 559,
546, 534, 521, 509, 497, 485, 473, 461,
449, 438, 426, 415, 404, 393, 382, 371,
360, 349, 339, 329, 318, 308, 298, 289,
279, 269, 260, 251, 242, 233, 224, 216,
207, 199, 191, 183, 175, 167, 160, 152,
145, 138, 131, 125, 118, 112, 105, 99,
94, 88, 82, 77, 72, 67, 62, 57,
53, 49, 45, 41, 37, 33, 30, 27,

```
  24, 21, 18, 16, 13, 11, 9, 7,
  6, 5, 3, 2, 1, 1, 0, 0,
  0, 0, 0, 1, 1, 2, 3, 5,
  6, 7, 9, 11, 13, 16, 18, 21,
  24, 27, 30, 33, 37, 41, 45, 49,
  53, 57, 62, 67, 72, 77, 82, 88,
  94, 99, 105, 112, 118, 125, 131, 138,
  145, 152, 160, 167, 175, 183, 191, 199,
  207, 216, 224, 233, 242, 251, 260, 269,
  279, 289, 298, 308, 318, 329, 339, 349,
  360, 371, 382, 393, 404, 415, 426, 438,
  449, 461, 473, 485, 497, 509, 521, 534,
  546, 559, 571, 584, 597, 610, 623, 636,
  650, 663, 676, 690, 704, 717, 731, 745,
  759, 773, 787, 801, 815, 829, 843, 858,
  872, 887, 901, 916, 930, 945, 960, 974,
  989, 1004, 1019, 1034, 1049, 1064, 1079, 1094,
  1109, 1124, 1139, 1154, 1169, 1184, 1199, 1214
};

const PROGMEM uint16_t DACLookup_FullSine2V_9Bit[512] {
  819, 829, 839, 849, 859, 869, 879, 889,
  899, 909, 919, 929, 939, 949, 959, 969,
  979, 989, 998, 1008, 1018, 1028, 1037, 1047,
  1057, 1066, 1076, 1085, 1095, 1104, 1114, 1123,
  1132, 1142, 1151, 1160, 1169, 1178, 1187, 1196,
  1205, 1214, 1223, 1231, 1240, 1249, 1257, 1266,
  1274, 1282, 1291, 1299, 1307, 1315, 1323, 1331,
  1339, 1346, 1354, 1362, 1369, 1376, 1384, 1391,
  1398, 1405, 1412, 1419, 1426, 1433, 1439, 1446,
  1452, 1458, 1465, 1471, 1477, 1483, 1489, 1494,
  1500, 1506, 1511, 1516, 1521, 1527, 1532, 1537,
  1541, 1546, 1551, 1555, 1559, 1564, 1568, 1572,
  1576, 1579, 1583, 1587, 1590, 1593, 1597, 1600,
  1603, 1606, 1608, 1611, 1613, 1616, 1618, 1620,
  1622, 1624, 1626, 1628, 1629, 1631, 1632, 1633,
  1634, 1635, 1636, 1636, 1637, 1637, 1638, 1638,
```

1638, 1638, 1638, 1637, 1637, 1636, 1636, 1635,
1634, 1633, 1632, 1631, 1629, 1628, 1626, 1624,
1622, 1620, 1618, 1616, 1613, 1611, 1608, 1606,
1603, 1600, 1597, 1593, 1590, 1587, 1583, 1579,
1576, 1572, 1568, 1564, 1559, 1555, 1551, 1546,
1541, 1537, 1532, 1527, 1521, 1516, 1511, 1506,
1500, 1494, 1489, 1483, 1477, 1471, 1465, 1458,
1452, 1446, 1439, 1433, 1426, 1419, 1412, 1405,
1398, 1391, 1384, 1376, 1369, 1362, 1354, 1346,
1339, 1331, 1323, 1315, 1307, 1299, 1291, 1282,
1274, 1266, 1257, 1249, 1240, 1231, 1223, 1214,
1205, 1196, 1187, 1178, 1169, 1160, 1151, 1142,
1132, 1123, 1114, 1104, 1095, 1085, 1076, 1066,
1057, 1047, 1037, 1028, 1018, 1008, 998, 989,
979, 969, 959, 949, 939, 929, 919, 909,
899, 889, 879, 869, 859, 849, 839, 829,
819, 809, 799, 789, 779, 769, 759, 749,
739, 729, 719, 709, 699, 689, 679, 669,
659, 649, 640, 630, 620, 610, 601, 591,
581, 572, 562, 553, 543, 534, 524, 515,
506, 496, 487, 478, 469, 460, 451, 442,
433, 424, 415, 407, 398, 389, 381, 372,
364, 356, 347, 339, 331, 323, 315, 307,
299, 292, 284, 276, 269, 262, 254, 247,
240, 233, 226, 219, 212, 205, 199, 192,
186, 180, 173, 167, 161, 155, 149, 144,
138, 132, 127, 122, 117, 111, 106, 101,
97, 92, 87, 83, 79, 74, 70, 66,
62, 59, 55, 51, 48, 45, 41, 38,
35, 32, 30, 27, 25, 22, 20, 18,
16, 14, 12, 10, 9, 7, 6, 5,
4, 3, 2, 2, 1, 1, 0, 0,
0, 0, 0, 1, 1, 2, 2, 3,
4, 5, 6, 7, 9, 10, 12, 14,
16, 18, 20, 22, 25, 27, 30, 32,
35, 38, 41, 45, 48, 51, 55, 59,
62, 66, 70, 74, 79, 83, 87, 92,

```
  97, 101, 106, 111, 117, 122, 127, 132,
  138, 144, 149, 155, 161, 167, 173, 180,
  186, 192, 199, 205, 212, 219, 226, 233,
  240, 247, 254, 262, 269, 276, 284, 292,
  299, 307, 315, 323, 331, 339, 347, 356,
  364, 372, 381, 389, 398, 407, 415, 424,
  433, 442, 451, 460, 469, 478, 487, 496,
  506, 515, 524, 534, 543, 553, 562, 572,
  581, 591, 601, 610, 620, 630, 640, 649,
  659, 669, 679, 689, 699, 709, 719, 729,
  739, 749, 759, 769, 779, 789, 799, 809

};

const PROGMEM uint16_t DACLookup_FullSine1V_9Bit[512] {
  410, 415, 420, 425, 430, 435, 440, 445,
  450, 455, 460, 465, 470, 475, 480, 485,
  490, 495, 500, 505, 510, 514, 519, 524,
  529, 534, 539, 543, 548, 553, 558, 562,
  567, 572, 576, 581, 585, 590, 594, 599,
  603, 608, 612, 616, 621, 625, 629, 634,
  638, 642, 646, 650, 654, 658, 662, 666,
  670, 674, 678, 682, 685, 689, 693, 696,
  700, 703, 707, 710, 714, 717, 720, 724,
  727, 730, 733, 736, 739, 742, 745, 748,
  751, 754, 756, 759, 762, 764, 767, 769,
  772, 774, 776, 778, 781, 783, 785, 787,
  789, 791, 793, 794, 796, 798, 799, 801,
  802, 804, 805, 806, 808, 809, 810, 811,
  812, 813, 814, 815, 816, 816, 817, 818,
  818, 818, 819, 819, 820, 820, 820, 820,
  820, 820, 820, 820, 820, 819, 819, 818,
  818, 818, 817, 816, 816, 815, 814, 813,
  812, 811, 810, 809, 808, 806, 805, 804,
  802, 801, 799, 798, 796, 794, 793, 791,
  789, 787, 785, 783, 781, 778, 776, 774,
  772, 769, 767, 764, 762, 759, 756, 754,
```

751, 748, 745, 742, 739, 736, 733, 730,
727, 724, 720, 717, 714, 710, 707, 703,
700, 696, 693, 689, 685, 682, 678, 674,
670, 666, 662, 658, 654, 650, 646, 642,
638, 634, 629, 625, 621, 616, 612, 608,
603, 599, 594, 590, 585, 581, 576, 572,
567, 562, 558, 553, 548, 543, 539, 534,
529, 524, 519, 514, 510, 505, 500, 495,
490, 485, 480, 475, 470, 465, 460, 455,
450, 445, 440, 435, 430, 425, 420, 415,
410, 405, 400, 395, 390, 385, 380, 375,
370, 365, 360, 355, 350, 345, 340, 335,
330, 325, 320, 315, 310, 306, 301, 296,
291, 286, 281, 277, 272, 267, 262, 258,
253, 248, 244, 239, 235, 230, 226, 221,
217, 212, 208, 204, 199, 195, 191, 186,
182, 178, 174, 170, 166, 162, 158, 154,
150, 146, 142, 138, 135, 131, 127, 124,
120, 117, 113, 110, 106, 103, 100, 96,
93, 90, 87, 84, 81, 78, 75, 72,
69, 66, 64, 61, 58, 56, 53, 51,
48, 46, 44, 42, 39, 37, 35, 33,
31, 29, 27, 26, 24, 22, 21, 19,
18, 16, 15, 14, 12, 11, 10, 9,
8, 7, 6, 5, 4, 4, 3, 2,
2, 2, 1, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 1, 2,
2, 2, 3, 4, 4, 5, 6, 7,
8, 9, 10, 11, 12, 14, 15, 16,
18, 19, 21, 22, 24, 26, 27, 29,
31, 33, 35, 37, 39, 42, 44, 46,
48, 51, 53, 56, 58, 61, 64, 66,
69, 72, 75, 78, 81, 84, 87, 90,
93, 96, 100, 103, 106, 110, 113, 117,
120, 124, 127, 131, 135, 138, 142, 146,
150, 154, 158, 162, 166, 170, 174, 178,
182, 186, 191, 195, 199, 204, 208, 212,

```
  217, 221, 226, 230, 235, 239, 244, 248,
  253, 258, 262, 267, 272, 277, 281, 286,
  291, 296, 301, 306, 310, 315, 320, 325,
  330, 335, 340, 345, 350, 355, 360, 365,
  370, 375, 380, 385, 390, 395, 400, 405
};



/*---------Look Up Table Onda Triangular----------*/
const PROGMEM uint16_t DACLookup_FullTriangle5V_9Bit[512] =
{
  16, 32, 48, 64, 80, 96, 112, 128,
  144, 160, 176, 192, 208, 224, 240, 256,
  272, 288, 304, 320, 336, 352, 368, 384,
  400, 416, 432, 448, 464, 480, 496, 512,
  528, 544, 560, 576, 592, 608, 624, 640,
  656, 672, 688, 704, 720, 736, 752, 768,
  784, 800, 816, 832, 848, 864, 880, 896,
  912, 928, 944, 960, 976, 992, 1008, 1024,
  1040, 1056, 1072, 1088, 1104, 1120, 1136, 1152,
  1168, 1184, 1200, 1216, 1232, 1248, 1264, 1280,
  1296, 1312, 1328, 1344, 1360, 1376, 1392, 1408,
  1424, 1440, 1456, 1472, 1488, 1504, 1520, 1536,
  1552, 1568, 1584, 1600, 1616, 1632, 1648, 1664,
  1680, 1696, 1712, 1728, 1744, 1760, 1776, 1792,
  1808, 1824, 1840, 1856, 1872, 1888, 1904, 1920,
  1936, 1952, 1968, 1984, 2000, 2016, 2032, 2048,
  2063, 2079, 2095, 2111, 2127, 2143, 2159, 2175,
  2191, 2207, 2223, 2239, 2255, 2271, 2287, 2303,
  2319, 2335, 2351, 2367, 2383, 2399, 2415, 2431,
  2447, 2463, 2479, 2495, 2511, 2527, 2543, 2559,
  2575, 2591, 2607, 2623, 2639, 2655, 2671, 2687,
  2703, 2719, 2735, 2751, 2767, 2783, 2799, 2815,
  2831, 2847, 2863, 2879, 2895, 2911, 2927, 2943,
  2959, 2975, 2991, 3007, 3023, 3039, 3055, 3071,
  3087, 3103, 3119, 3135, 3151, 3167, 3183, 3199,
```

3215, 3231, 3247, 3263, 3279, 3295, 3311, 3327,
3343, 3359, 3375, 3391, 3407, 3423, 3439, 3455,
3471, 3487, 3503, 3519, 3535, 3551, 3567, 3583,
3599, 3615, 3631, 3647, 3663, 3679, 3695, 3711,
3727, 3743, 3759, 3775, 3791, 3807, 3823, 3839,
3855, 3871, 3887, 3903, 3919, 3935, 3951, 3967,
3983, 3999, 4015, 4031, 4047, 4063, 4079, 4095,
4079, 4063, 4047, 4031, 4015, 3999, 3983, 3967,
3951, 3935, 3919, 3903, 3887, 3871, 3855, 3839,
3823, 3807, 3791, 3775, 3759, 3743, 3727, 3711,
3695, 3679, 3663, 3647, 3631, 3615, 3599, 3583,
3567, 3551, 3535, 3519, 3503, 3487, 3471, 3455,
3439, 3423, 3407, 3391, 3375, 3359, 3343, 3327,
3311, 3295, 3279, 3263, 3247, 3231, 3215, 3199,
3183, 3167, 3151, 3135, 3119, 3103, 3087, 3071,
3055, 3039, 3023, 3007, 2991, 2975, 2959, 2943,
2927, 2911, 2895, 2879, 2863, 2847, 2831, 2815,
2799, 2783, 2767, 2751, 2735, 2719, 2703, 2687,
2671, 2655, 2639, 2623, 2607, 2591, 2575, 2559,
2543, 2527, 2511, 2495, 2479, 2463, 2447, 2431,
2415, 2399, 2383, 2367, 2351, 2335, 2319, 2303,
2287, 2271, 2255, 2239, 2223, 2207, 2191, 2175,
2159, 2143, 2127, 2111, 2095, 2079, 2063, 2048,
2032, 2016, 2000, 1984, 1968, 1952, 1936, 1920,
1904, 1888, 1872, 1856, 1840, 1824, 1808, 1792,
1776, 1760, 1744, 1728, 1712, 1696, 1680, 1664,
1648, 1632, 1616, 1600, 1584, 1568, 1552, 1536,
1520, 1504, 1488, 1472, 1456, 1440, 1424, 1408,
1392, 1376, 1360, 1344, 1328, 1312, 1296, 1280,
1264, 1248, 1232, 1216, 1200, 1184, 1168, 1152,
1136, 1120, 1104, 1088, 1072, 1056, 1040, 1024,
1008, 992, 976, 960, 944, 928, 912, 896,
880, 864, 848, 832, 816, 800, 784, 768,
752, 736, 720, 704, 688, 672, 656, 640,
624, 608, 592, 576, 560, 544, 528, 512,
496, 480, 464, 448, 432, 416, 400, 384,
368, 352, 336, 320, 304, 288, 272, 256,

```
  240, 224, 208, 192, 176, 160, 144, 128,
  112, 96, 80, 64, 48, 32, 16, 0
};

const PROGMEM uint16_t DACLookup_FullTriangle4V_9Bit[512] =
{
  13, 26, 38, 51, 64, 77, 90, 102,
  115, 128, 141, 154, 166, 179, 192, 205,
  218, 230, 243, 256, 269, 282, 294, 307,
  320, 333, 346, 358, 371, 384, 397, 410,
  422, 435, 448, 461, 473, 486, 499, 512,
  525, 537, 550, 563, 576, 589, 601, 614,
  627, 640, 653, 665, 678, 691, 704, 717,
  729, 742, 755, 768, 781, 793, 806, 819,
  832, 845, 857, 870, 883, 896, 909, 921,
  934, 947, 960, 973, 985, 998, 1011, 1024,
  1037, 1049, 1062, 1075, 1088, 1101, 1113, 1126,
  1139, 1152, 1165, 1177, 1190, 1203, 1216, 1229,
  1241, 1254, 1267, 1280, 1292, 1305, 1318, 1331,
  1344, 1356, 1369, 1382, 1395, 1408, 1420, 1433,
  1446, 1459, 1472, 1484, 1497, 1510, 1523, 1536,
  1548, 1561, 1574, 1587, 1600, 1612, 1625, 1638,
  1651, 1664, 1676, 1689, 1702, 1715, 1728, 1740,
  1753, 1766, 1779, 1792, 1804, 1817, 1830, 1843,
  1856, 1868, 1881, 1894, 1907, 1920, 1932, 1945,
  1958, 1971, 1984, 1996, 2009, 2022, 2035, 2048,
  2060, 2073, 2086, 2099, 2111, 2124, 2137, 2150,
  2163, 2175, 2188, 2201, 2214, 2227, 2239, 2252,
  2265, 2278, 2291, 2303, 2316, 2329, 2342, 2355,
  2367, 2380, 2393, 2406, 2419, 2431, 2444, 2457,
  2470, 2483, 2495, 2508, 2521, 2534, 2547, 2559,
  2572, 2585, 2598, 2611, 2623, 2636, 2649, 2662,
  2675, 2687, 2700, 2713, 2726, 2739, 2751, 2764,
  2777, 2790, 2803, 2815, 2828, 2841, 2854, 2867,
  2879, 2892, 2905, 2918, 2930, 2943, 2956, 2969,
  2982, 2994, 3007, 3020, 3033, 3046, 3058, 3071,
  3084, 3097, 3110, 3122, 3135, 3148, 3161, 3174,
```

```
  3186, 3199, 3212, 3225, 3238, 3250, 3263, 3276,
  3263, 3250, 3238, 3225, 3212, 3199, 3186, 3174,
  3161, 3148, 3135, 3122, 3110, 3097, 3084, 3071,
  3058, 3046, 3033, 3020, 3007, 2994, 2982, 2969,
  2956, 2943, 2930, 2918, 2905, 2892, 2879, 2867,
  2854, 2841, 2828, 2815, 2803, 2790, 2777, 2764,
  2751, 2739, 2726, 2713, 2700, 2687, 2675, 2662,
  2649, 2636, 2623, 2611, 2598, 2585, 2572, 2559,
  2547, 2534, 2521, 2508, 2495, 2483, 2470, 2457,
  2444, 2431, 2419, 2406, 2393, 2380, 2367, 2355,
  2342, 2329, 2316, 2303, 2291, 2278, 2265, 2252,
  2239, 2227, 2214, 2201, 2188, 2175, 2163, 2150,
  2137, 2124, 2111, 2099, 2086, 2073, 2060, 2048,
  2035, 2022, 2009, 1996, 1984, 1971, 1958, 1945,
  1932, 1920, 1907, 1894, 1881, 1868, 1856, 1843,
  1830, 1817, 1804, 1792, 1779, 1766, 1753, 1740,
  1728, 1715, 1702, 1689, 1676, 1664, 1651, 1638,
  1625, 1612, 1600, 1587, 1574, 1561, 1548, 1536,
  1523, 1510, 1497, 1484, 1472, 1459, 1446, 1433,
  1420, 1408, 1395, 1382, 1369, 1356, 1344, 1331,
  1318, 1305, 1292, 1280, 1267, 1254, 1241, 1229,
  1216, 1203, 1190, 1177, 1165, 1152, 1139, 1126,
  1113, 1101, 1088, 1075, 1062, 1049, 1037, 1024,
  1011, 998, 985, 973, 960, 947, 934, 921,
  909, 896, 883, 870, 857, 845, 832, 819,
  806, 793, 781, 768, 755, 742, 729, 717,
  704, 691, 678, 665, 653, 640, 627, 614,
  601, 589, 576, 563, 550, 537, 525, 512,
  499, 486, 473, 461, 448, 435, 422, 410,
  397, 384, 371, 358, 346, 333, 320, 307,
  294, 282, 269, 256, 243, 230, 218, 205,
  192, 179, 166, 154, 141, 128, 115, 102,
  90, 77, 64, 51, 38, 26, 13, 0
};

const PROGMEM uint16_t DACLookup_FullTriangle3V_9Bit[512] =
{
```

10, 19, 29, 38, 48, 58, 67, 77,
86, 96, 106, 115, 125, 134, 144, 154,
163, 173, 182, 192, 202, 211, 221, 230,
240, 250, 259, 269, 278, 288, 298, 307,
317, 326, 336, 346, 355, 365, 374, 384,
394, 403, 413, 422, 432, 442, 451, 461,
470, 480, 490, 499, 509, 518, 528, 538,
547, 557, 566, 576, 586, 595, 605, 615,
624, 634, 643, 653, 663, 672, 682, 691,
701, 711, 720, 730, 739, 749, 759, 768,
778, 787, 797, 807, 816, 826, 835, 845,
855, 864, 874, 883, 893, 903, 912, 922,
931, 941, 951, 960, 970, 979, 989, 999,
1008, 1018, 1027, 1037, 1047, 1056, 1066, 1075,
1085, 1095, 1104, 1114, 1123, 1133, 1143, 1152,
1162, 1171, 1181, 1191, 1200, 1210, 1219, 1229,
1239, 1248, 1258, 1267, 1277, 1287, 1296, 1306,
1315, 1325, 1335, 1344, 1354, 1363, 1373, 1383,
1392, 1402, 1411, 1421, 1431, 1440, 1450, 1459,
1469, 1479, 1488, 1498, 1507, 1517, 1527, 1536,
1546, 1555, 1565, 1575, 1584, 1594, 1603, 1613,
1623, 1632, 1642, 1651, 1661, 1671, 1680, 1690,
1699, 1709, 1719, 1728, 1738, 1747, 1757, 1767,
1776, 1786, 1795, 1805, 1815, 1824, 1834, 1844,
1853, 1863, 1872, 1882, 1892, 1901, 1911, 1920,
1930, 1940, 1949, 1959, 1968, 1978, 1988, 1997,
2007, 2016, 2026, 2036, 2045, 2055, 2064, 2074,
2084, 2093, 2103, 2112, 2122, 2132, 2141, 2151,
2160, 2170, 2180, 2189, 2199, 2208, 2218, 2228,
2237, 2247, 2256, 2266, 2276, 2285, 2295, 2304,
2314, 2324, 2333, 2343, 2352, 2362, 2372, 2381,
2391, 2400, 2410, 2420, 2429, 2439, 2448, 2458,
2448, 2439, 2429, 2420, 2410, 2400, 2391, 2381,
2372, 2362, 2352, 2343, 2333, 2324, 2314, 2304,
2295, 2285, 2276, 2266, 2256, 2247, 2237, 2228,
2218, 2208, 2199, 2189, 2180, 2170, 2160, 2151,
2141, 2132, 2122, 2112, 2103, 2093, 2084, 2074,

```
  2064, 2055, 2045, 2036, 2026, 2016, 2007, 1997,
  1988, 1978, 1968, 1959, 1949, 1940, 1930, 1920,
  1911, 1901, 1892, 1882, 1872, 1863, 1853, 1844,
  1834, 1824, 1815, 1805, 1795, 1786, 1776, 1767,
  1757, 1747, 1738, 1728, 1719, 1709, 1699, 1690,
  1680, 1671, 1661, 1651, 1642, 1632, 1623, 1613,
  1603, 1594, 1584, 1575, 1565, 1555, 1546, 1536,
  1527, 1517, 1507, 1498, 1488, 1479, 1469, 1459,
  1450, 1440, 1431, 1421, 1411, 1402, 1392, 1383,
  1373, 1363, 1354, 1344, 1335, 1325, 1315, 1306,
  1296, 1287, 1277, 1267, 1258, 1248, 1239, 1229,
  1219, 1210, 1200, 1191, 1181, 1171, 1162, 1152,
  1143, 1133, 1123, 1114, 1104, 1095, 1085, 1075,
  1066, 1056, 1047, 1037, 1027, 1018, 1008, 999,
  989, 979, 970, 960, 951, 941, 931, 922,
  912, 903, 893, 883, 874, 864, 855, 845,
  835, 826, 816, 807, 797, 787, 778, 768,
  759, 749, 739, 730, 720, 711, 701, 691,
  682, 672, 663, 653, 643, 634, 624, 615,
  605, 595, 586, 576, 566, 557, 547, 538,
  528, 518, 509, 499, 490, 480, 470, 461,
  451, 442, 432, 422, 413, 403, 394, 384,
  374, 365, 355, 346, 336, 326, 317, 307,
  298, 288, 278, 269, 259, 250, 240, 230,
  221, 211, 202, 192, 182, 173, 163, 154,
  144, 134, 125, 115, 106, 96, 86, 77,
  67, 58, 48, 38, 29, 19, 10, 0
};

const PROGMEM uint16_t DACLookup_FullTriangle2V_9Bit[512] =
{
  6, 13, 19, 26, 32, 38, 45, 51,
  58, 64, 70, 77, 83, 90, 96, 102,
  109, 115, 122, 128, 134, 141, 147, 154,
  160, 166, 173, 179, 186, 192, 198, 205,
  211, 218, 224, 230, 237, 243, 250, 256,
  262, 269, 275, 282, 288, 294, 301, 307,
```

314, 320, 326, 333, 339, 346, 352, 358,
365, 371, 378, 384, 390, 397, 403, 410,
416, 422, 429, 435, 441, 448, 454, 461,
467, 473, 480, 486, 493, 499, 505, 512,
518, 525, 531, 537, 544, 550, 557, 563,
569, 576, 582, 589, 595, 601, 608, 614,
621, 627, 633, 640, 646, 653, 659, 665,
672, 678, 685, 691, 697, 704, 710, 717,
723, 729, 736, 742, 749, 755, 761, 768,
774, 781, 787, 793, 800, 806, 813, 819,
825, 832, 838, 845, 851, 857, 864, 870,
877, 883, 889, 896, 902, 909, 915, 921,
928, 934, 941, 947, 953, 960, 966, 973,
979, 985, 992, 998, 1005, 1011, 1017, 1024,
1030, 1037, 1043, 1049, 1056, 1062, 1069, 1075,
1081, 1088, 1094, 1101, 1107, 1113, 1120, 1126,
1133, 1139, 1145, 1152, 1158, 1165, 1171, 1177,
1184, 1190, 1197, 1203, 1209, 1216, 1222, 1229,
1235, 1241, 1248, 1254, 1260, 1267, 1273, 1280,
1286, 1292, 1299, 1305, 1312, 1318, 1324, 1331,
1337, 1344, 1350, 1356, 1363, 1369, 1376, 1382,
1388, 1395, 1401, 1408, 1414, 1420, 1427, 1433,
1440, 1446, 1452, 1459, 1465, 1472, 1478, 1484,
1491, 1497, 1504, 1510, 1516, 1523, 1529, 1536,
1542, 1548, 1555, 1561, 1568, 1574, 1580, 1587,
1593, 1600, 1606, 1612, 1619, 1625, 1632, 1638,
1632, 1625, 1619, 1612, 1606, 1600, 1593, 1587,
1580, 1574, 1568, 1561, 1555, 1548, 1542, 1536,
1529, 1523, 1516, 1510, 1504, 1497, 1491, 1484,
1478, 1472, 1465, 1459, 1452, 1446, 1440, 1433,
1427, 1420, 1414, 1408, 1401, 1395, 1388, 1382,
1376, 1369, 1363, 1356, 1350, 1344, 1337, 1331,
1324, 1318, 1312, 1305, 1299, 1292, 1286, 1280,
1273, 1267, 1260, 1254, 1248, 1241, 1235, 1229,
1222, 1216, 1209, 1203, 1197, 1190, 1184, 1177,
1171, 1165, 1158, 1152, 1145, 1139, 1133, 1126,
1120, 1113, 1107, 1101, 1094, 1088, 1081, 1075,

```
  1069, 1062, 1056, 1049, 1043, 1037, 1030, 1024,
  1017, 1011, 1005, 998, 992, 985, 979, 973,
  966, 960, 953, 947, 941, 934, 928, 921,
  915, 909, 902, 896, 889, 883, 877, 870,
  864, 857, 851, 845, 838, 832, 825, 819,
  813, 806, 800, 793, 787, 781, 774, 768,
  761, 755, 749, 742, 736, 729, 723, 717,
  710, 704, 697, 691, 685, 678, 672, 665,
  659, 653, 646, 640, 633, 627, 621, 614,
  608, 601, 595, 589, 582, 576, 569, 563,
  557, 550, 544, 537, 531, 525, 518, 512,
  505, 499, 493, 486, 480, 473, 467, 461,
  454, 448, 441, 435, 429, 422, 416, 410,
  403, 397, 390, 384, 378, 371, 365, 358,
  352, 346, 339, 333, 326, 320, 314, 307,
  301, 294, 288, 282, 275, 269, 262, 256,
  250, 243, 237, 230, 224, 218, 211, 205,
  198, 192, 186, 179, 173, 166, 160, 154,
  147, 141, 134, 128, 122, 115, 109, 102,
  96, 90, 83, 77, 70, 64, 58, 51,
  45, 38, 32, 26, 19, 13, 6, 0
};

const PROGMEM uint16_t DACLookup_FullTriangle1V_9Bit[512] =
{
  3, 6, 10, 13, 16, 19, 22, 26,
  29, 32, 35, 38, 42, 45, 48, 51,
  54, 58, 61, 64, 67, 70, 74, 77,
  80, 83, 86, 90, 93, 96, 99, 103,
  106, 109, 112, 115, 119, 122, 125, 128,
  131, 135, 138, 141, 144, 147, 151, 154,
  157, 160, 163, 167, 170, 173, 176, 179,
  183, 186, 189, 192, 195, 199, 202, 205,
  208, 211, 215, 218, 221, 224, 227, 231,
  234, 237, 240, 243, 247, 250, 253, 256,
  259, 263, 266, 269, 272, 275, 279, 282,
  285, 288, 291, 295, 298, 301, 304, 308,
```

311, 314, 317, 320, 324, 327, 330, 333,
336, 340, 343, 346, 349, 352, 356, 359,
362, 365, 368, 372, 375, 378, 381, 384,
388, 391, 394, 397, 400, 404, 407, 410,
413, 416, 420, 423, 426, 429, 432, 436,
439, 442, 445, 448, 452, 455, 458, 461,
464, 468, 471, 474, 477, 480, 484, 487,
490, 493, 496, 500, 503, 506, 509, 513,
516, 519, 522, 525, 529, 532, 535, 538,
541, 545, 548, 551, 554, 557, 561, 564,
567, 570, 573, 577, 580, 583, 586, 589,
593, 596, 599, 602, 605, 609, 612, 615,
618, 621, 625, 628, 631, 634, 637, 641,
644, 647, 650, 653, 657, 660, 663, 666,
669, 673, 676, 679, 682, 685, 689, 692,
695, 698, 701, 705, 708, 711, 714, 718,
721, 724, 727, 730, 734, 737, 740, 743,
746, 750, 753, 756, 759, 762, 766, 769,
772, 775, 778, 782, 785, 788, 791, 794,
798, 801, 804, 807, 810, 814, 817, 820,
817, 814, 810, 807, 804, 801, 798, 794,
791, 788, 785, 782, 778, 775, 772, 769,
766, 762, 759, 756, 753, 750, 746, 743,
740, 737, 734, 730, 727, 724, 721, 718,
714, 711, 708, 705, 701, 698, 695, 692,
689, 685, 682, 679, 676, 673, 669, 666,
663, 660, 657, 653, 650, 647, 644, 641,
637, 634, 631, 628, 625, 621, 618, 615,
612, 609, 605, 602, 599, 596, 593, 589,
586, 583, 580, 577, 573, 570, 567, 564,
561, 557, 554, 551, 548, 545, 541, 538,
535, 532, 529, 525, 522, 519, 516, 513,
509, 506, 503, 500, 496, 493, 490, 487,
484, 480, 477, 474, 471, 468, 464, 461,
458, 455, 452, 448, 445, 442, 439, 436,
432, 429, 426, 423, 420, 416, 413, 410,
407, 404, 400, 397, 394, 391, 388, 384,

```
  381, 378, 375, 372, 368, 365, 362, 359,
  356, 352, 349, 346, 343, 340, 336, 333,
  330, 327, 324, 320, 317, 314, 311, 308,
  304, 301, 298, 295, 291, 288, 285, 282,
  279, 275, 272, 269, 266, 263, 259, 256,
  253, 250, 247, 243, 240, 237, 234, 231,
  227, 224, 221, 218, 215, 211, 208, 205,
  202, 199, 195, 192, 189, 186, 183, 179,
  176, 173, 170, 167, 163, 160, 157, 154,
  151, 147, 144, 141, 138, 135, 131, 128,
  125, 122, 119, 115, 112, 109, 106, 103,
  99, 96, 93, 90, 86, 83, 80, 77,
  74, 70, 67, 64, 61, 58, 54, 51,
  48, 45, 42, 38, 35, 32, 29, 26,
  22, 19, 16, 13, 10, 6, 3, 0
};
/*---------Look Up Table Onda Rectangular-------------*/
const PROGMEM uint16_t DACLookup_FullRectangle5V_9Bit[512] =
{
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
  4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
```

4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
4095, 4095, 4095, 4095, 4095, 4095, 4095, 4095,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,

```
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0
};

const PROGMEM uint16_t DACLookup_FullRectangle4V_9Bit[512] =
{
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
  3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
```

3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
3276, 3276, 3276, 3276, 3276, 3276, 3276, 3276,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,

```
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0
};

const PROGMEM uint16_t DACLookup_FullRectangle3V_9Bit[512] =
{
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
```

```
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  2458, 2458, 2458, 2458, 2458, 2458, 2458, 2458,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0
};

const PROGMEM uint16_t DACLookup_FullRectangle2V_9Bit[512] =
```

```
{
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  1638, 1638, 1628, 1638, 1638, 1638, 1638, 1638,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
```

```
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0
};

const PROGMEM uint16_t DACLookup_FullRectangle1V_9Bit[512] =
{
  820, 820, 820, 820, 820, 820, 820, 820,
  820, 820, 820, 820, 820, 820, 820, 820,
  820, 820, 820, 820, 820, 820, 820, 820,
  820, 820, 820, 820, 820, 820, 820, 820,
  820, 820, 820, 820, 820, 820, 820, 820,
```

820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820, 820,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,

```
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0
};

void setup(void) {

  Serial.begin(9600);

  // For Adafruit MCP4725A1 the address is 0x62 (default) or 0x63 (ADDR pin tied to VCC)
  // For MCP4725A0(nuestro MCP4725) the address is 0x60 or 0x61
  // For MCP4725A2 the address is 0x64 or 0x65
  //dac.begin(0x60);
  //dac.begin(0x64);
  dac.begin(0x60); //Modelo del DAC MCP4725A0
  Serial.println(estado);
}
```

```
void loop(void) {

  if (Serial.available() > 0) {
    estadoChar = Serial.read();

    if (estadoChar == 's') {
      estado = 1;
    }
    if (estadoChar == 'r') {
      estado = 2;
    }
    if (estadoChar == 't') {
      estado = 3;
    }

    if (estadoChar != 's' or estadoChar != 'r' or estadoChar != 't') {
      if (estadoChar == 'a') {
        amplitud = 1;
      }
      if (estadoChar == 'b') {
        amplitud = 2;
      }
      if (estadoChar == 'c') {
        amplitud = 3;
      }
      if (estadoChar == 'd') {
        amplitud = 4;
      }
      if (estadoChar == 'e') {
        amplitud = 5;
      }
    }
    if (estadoChar == 'o') {
      estado=0;
    }
  }
```

```
if (estado == 0) {
  dac.setVoltage(0,false);
}

if (estado == 1) {
  if (amplitud == 1) {
    generate_sin(1);
  }
  if (amplitud == 2) {
    generate_sin(2);
  }
  if (amplitud == 3) {
    generate_sin(3);
  }
  if (amplitud == 4) {
    generate_sin(4);
  }
  if (amplitud == 5) {
    generate_sin(5);
  }
}
if (estado == 2) {
  if (amplitud == 1) {
    generate_rect(1);
  }
  if (amplitud == 2) {
    generate_rect(2);
  }
  if (amplitud == 3) {
    generate_rect(3);
  }
  if (amplitud == 4) {
    generate_rect(4);
  }
  if (amplitud == 5) {
    generate_rect(5);
  }
```

```
  }
  if (estado == 3) {
    if (amplitud == 1) {
      generate_triangle(1);
    }
    if (amplitud == 2) {
      generate_triangle(2);
    }
    if (amplitud == 3) {
      generate_triangle(3);
    }
    if (amplitud == 4) {
      generate_triangle(4);
    }
    if (amplitud == 5) {
      generate_triangle(5);
    }
  }
}


void generate_rect(int Resolution) {
  uint16_t i;
  switch (Resolution) {
    case 1:
      for (i = 0; i < 512; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullRectangle1V_9Bit[i])), false);
      }
      break;
    case 2:
      for (i = 0; i < 512; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullRectangle2V_9Bit[i])), false);
      }
      break;
    case 3:
```

```
      for (i = 0; i < 512; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullRectangle3V_9Bit[i])), false);
      }
      break;
    case 4:
      for (i = 0; i < 512; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullRectangle4V_9Bit[i])), false);
      }
      break;
    case 5:
      for (i = 0; i < 512; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullRectangle5V_9Bit[i])), false);
      }
      break;
    default:
      for (i = 0; i < 512; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullRectangle5V_9Bit[i])), false);
      }
      break;
  }

}

void generate_sin(int Resolution) {
  uint16_t i;
  switch (Resolution) {
    case 1:
      for (i = 0; i < 512; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullSine1V_9Bit[i])), false);
        //int d = pow(2,Resolution);//no se para
        //delayMicroseconds(d);//que son estas lineas
      }
```

```
      break;
    case 2:
      for (i = 0; i < 512; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullSine2V_9Bit[i])), false);
      }
      break;
    case 3:
      for (i = 0; i < 512; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullSine3V_9Bit[i])), false);
      }
      break;
    case 4:
      for (i = 0; i < 512; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullSine4V_9Bit[i])), false);
      }
      break;
    case 5:
      for (i = 0; i < 512; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullSine5V_9Bit[i])), false);
      }
      break;
    default:
      for (i = 0; i < 512; i++)
      {
        dac.setVoltage(pgm_read_word(&(DACLookup_FullSine5V_9Bit[i])), false);
      }
      break;
  }

}

void generate_triangle(int Resolution) {
  uint16_t i;
```

```
switch (Resolution) {
  case 1:
    for (i = 0; i < 16; i++)
    {
      dac.setVoltage(pgm_read_word(&(DACLookup_FullTriangle1V_9Bit[i])), false);
    }
    break;
  case 2:
    for (i = 0; i < 32; i++)
    {
      dac.setVoltage(pgm_read_word(&(DACLookup_FullTriangle2V_9Bit[i])), false);
    }
    break;
  case 3:
    for (i = 0; i < 64; i++)
    {
      dac.setVoltage(pgm_read_word(&(DACLookup_FullTriangle3V_9Bit[i])), false);
    }
    break;
  case 4:
    for (i = 0; i < 128; i++)
    {
      dac.setVoltage(pgm_read_word(&(DACLookup_FullTriangle4V_9Bit[i])), false);
    }
    break;
  case 5:
    for (i = 0; i < 256; i++)
    {
      dac.setVoltage(pgm_read_word(&(DACLookup_FullTriangle5V_9Bit[i])), false);
    }
    break;
  default:
    for (i = 0; i < 512; i++)
    {
      dac.setVoltage(pgm_read_word(&(DACLookup_FullTriangle5V_9Bit[i])), false);
    }
    break;
```

```
  }

}
```