

Manual Tarjeta Termopar Hardware & Software

*Instituto Tecnológico Nacional de México Campus Morelia
&
Departamento de Posgrado de Ingeniería Electrónica*

Documento realizado el día 23 de julio del 2021; La versión 1.0 de este documento fue realizado por el alumno Rafael Raya Tapia de la carrera de ingeniería electrónica para obtener la liberación del servicio social con la supervisión del M.c Pascual Neftali Chavez Campos.

Resumen: Esta es una guía para el correcto manejo de la Tarjeta de Multiplexado de Termopar Tipo K donde se abordara los dos modos de alimentación, conexiones Tarjeta-Arduino y como conectar los termopares así como el uso de un programa para la visualización de datos y su guardado.

1. Introduction

Esta placa esta basada en el circuito integrado [MAX31855K](#) que soporta el termopar de punta fría tipo K a esto se le agrega un multiplexor esto para tener la oportunidad de conectar hasta 8 termopares de igual forma se diseño con la intención de que esta sea controlada por una tarjeta de control de Arduino para facilitar su uso y para la visualización de datos se diseño un pequeño programa en el que se puede graficar las temperaturas y guardar los datos estos se hizo en el entorno de desarrollo de [Processing](#). Los usos que se le pueden dar son.

- Monitoreo múltiple de altas temperaturas donde se requiera colocar termopares.
- Automotriz: las temperaturas del motor, el escape y los frenos son más altas de lo que pueden soportar la mayoría de los sensores de temperatura, pero dentro de la gama de termopares es posible.
- Procesos químicos: donde los productos químicos corrosivos pueden dañar las sondas o alcanzar alta temperatura los termopares están más disponibles para tales procesos.

2. Hardware

2.1. Conexiones Tarjeta-Arduino

A continuación se verán las conexiones correspondientes de la tarjeta a la placa Arduino. En la figura 1 y la figura 2. Como se puede apreciar son 7 conexiones mas 2 de alimentación (una para el Voltaje ya sea para 3.3v o 5v y otra para la tierra). Las conexiones A0-A2 sirven para poder cambiar de termopar a termopar esto con ayuda de nuestro circuito integrado multiplexor estas pueden ir conectadas a cualquier puerto digital en nuestra placa de arduino. La conexión En hace referencia a enable que significa permitir esta conexión siempre se tendrá que poner en "1" lógico para que funcione nuestro multiplexor esta conexión puede ir a cualquier puerto digital de nuestra placa arduino. Las conexiones MISO, MOSI y SCLK sirven para tener la comunicación de datos (temperatura) de los termopares por parte de nuestro CI MAX31855 hacia nuestra placa Arduino de igual forma estos pueden ir conectados a cualquier puerto digital. Por ultimo tenemos

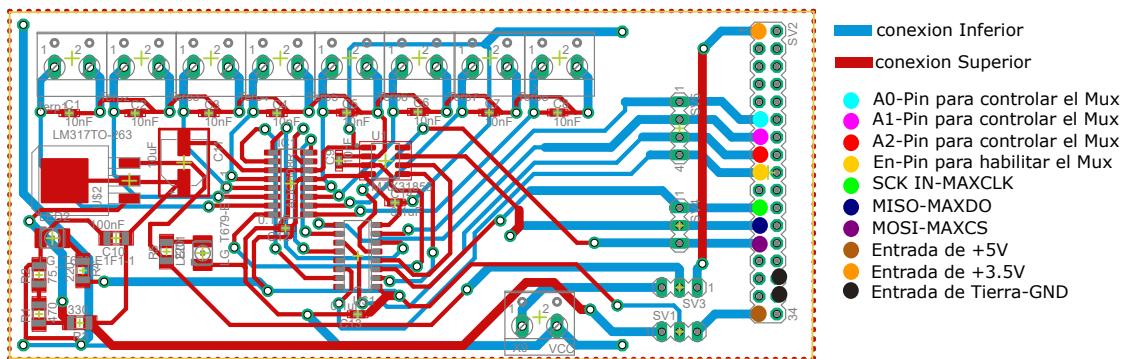


Figura 1. Diagrama Gerber final

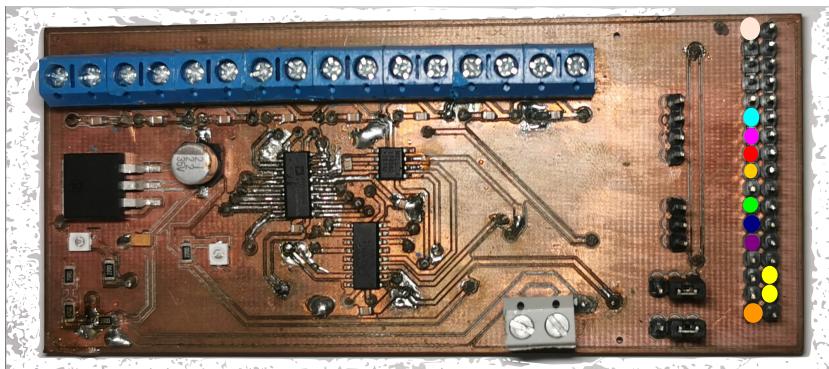


Figura 2. PCB final de la tarjeta

las conexiones de alimentación estas las vamos a ver mas a detalle a continuación pero son dos una a tierra y la otra puede ser ya sea a 5v o 3.3v.

2.2. Modos de Alimentación

Si bien la mayoría de los componentes que conforman nuestra placa se necesita una alimentación de 3.3v como se apodó notar nuestra tarjeta de adquisición de datos puede ser alimentada por +5v o +3.3v esto gracias a que se incorpora un regulador de voltaje interno para reducir de 5v a 3.3v. Sin embargo es requerido conectar con ayuda de "las capuchas" una u otra. Para poder alimentar con +5v mediante la placa de Arduino se tiene que poner la siguiente configuración ver en la figura 3. En caso de que sea necesario alimentar con 5v con una fuente externa se debe hacer las conexiones como se indica en la figura 4 tanto la polarización de esta fuente como la posición de las capuchas. Para alimentar directamente los 3.3v desde la placa Arduino es necesario poner en posición las capuchas como se observa en la figura 5.

Cabe mencionar que se encenderán dos led azules en caso de que se alimente con 5v y solamente se encenderá 1 en caso de que se alimente con 3.3v esto para visualizar que se ha conectado de manera satisfactoria.

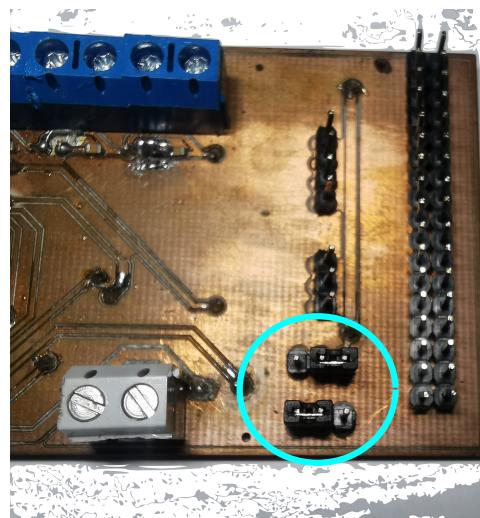


Figura 3. Posición de conectores para alimentación de 5v con la placa Arduino

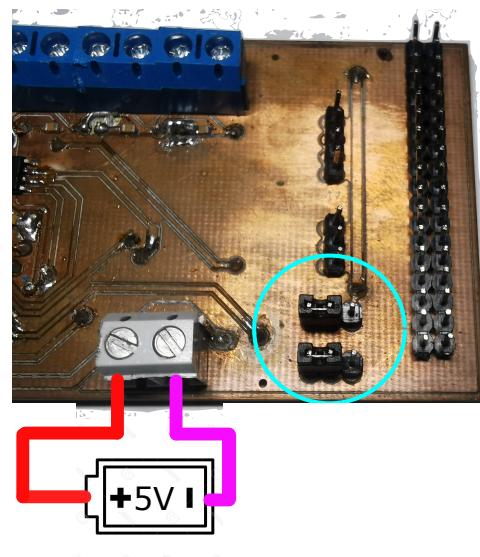


Figura 4. Posición de conectores para alimentación de 5v con una fuente externa

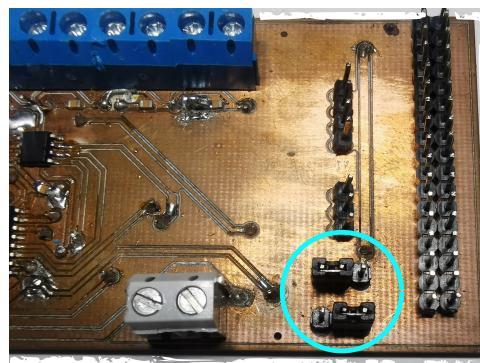


Figura 5. Posición de conectores para alimentación de 3.3v con la placa Arduino

2.3. Conexión de los Termopares

En la figura 6 podemos observar como es la conexión de los termopares, analizando la imagen se aprecia que la punta negativa de los termopares siempre va del lado izquierdo (desde esta vista relativamente) y la positiva del lado derecho (desde esta vista relativamente).

Para saber cual es la polaridad de nuestro termopar es necesario tener la hoja de datos de este puesto que no hay una estandarización de colores ejemplo; Algunos autores utilizan los colores rojo-azul, otros rojo-amarillo, algunos verde-negro. Es por esta variación de colores que se recomienda verificar propiamente la polarización del termopar.

NOTA; En caso de que se requiera recomiendo correr únicamente el programa en el IDE de arduino para verificar que este correctamente conectados, en caso de que se muestren datos (temperatura) negativos en el "Monitor serie" significa que algún termopar esta mal conectado.

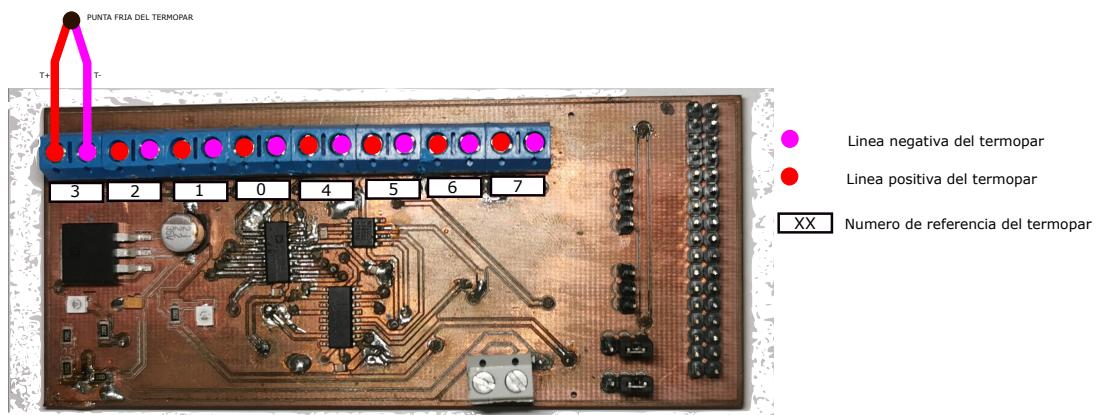


Figura 6. Polarización para la conexión de los termopares

3. Software

3.1. Programa-Arduino

A continuación se va explicar el código arduino por partes y al final de esto se pondrá el código completo.

Parte 1

```
1 #include <SPI.h>
2 #include "Adafruit_MAX31855.h"
3
4 // Definicion de los pines conectados con el Max31855
5 #define MAXDO 3 //miso
6 #define MAXCS 4 // mosi
7 #define MAXCLK 5 //sck in
8 int r0 = 0; //A0
9 int r1 = 0; // A1 los tres pines digitales para los pins
10 int r2 = 0; //A2
```

En las líneas 1-2 se incluyen las librerías necesarias.

En las líneas 5-7 se definen los pines para la comunicación con el circuito integrado MAX31855, en este caso se define que el pin 3 corresponde al miso, el pin 4 al mosi, el pin 5 con el sck in. En las líneas 8-10 se definen las variables r0, r1, r2 que nos van a ayudar mas adelante para poder cambiar de canal en el multiplexor.

Parte 2

```

1 int bin [] = {0,1,2,3,4,5};
2 int cantArray=6;
3 // inizializacion del termoacople
4 Adafruit_MAX31855 thermocouple(MAXCLK, MAXCS, MAXDO) ;
5
6 // creacion de la peticion SPI del thermocouple con el hardware
7 void setup() {
8 #define PINEN 8 //Mux Enable pin
9 // Bits para las entradas de seleccion del multiplexor
10 digitalWrite(PINEN, HIGH); // enable on
11 pinMode(11, OUTPUT); // A0 //Mux Address 0 pin
12 pinMode(10, OUTPUT); // A1 Mux Address 1 pin
13 pinMode(9, OUTPUT); // A2 Mux Address 2 pin}

```

la linea 1 es muy importante puesto que en esta se define los termopares que vamos a estar midiendo en este ejemplo se están analizando los termopares 1-5 como se muestra en la figura 6. En caso de que se quiera ver solo tres termopares seria ejem; 4, 1, 3, como se puede ver en este ejemplo no importa la acomodación de estos se puede utilizar la que se dese. en caso que nomas se dese observar un solo termopar únicamente se pondrá el numero de referencia de este ejem; 5.

En la linea 2 se pondrá en numero de termopares para visualización en el ejemplo se aprecia que hay 6 termopares del 0 al 5 por lo que se pondrá 6 si nomas hubiera uno se pondría 1.

En la linea 4 se inicializa la función de “Adafruit Max”.

En la linea 7 se inicializa la función “void setup”.

En la linea 8 se define como PIEN el pin 8.

En las lineas 10-13 se definen como salidas dejando siempre en uno lógico el PIEN en cambio de la linea 11-13 se definen únicamente como salidas siendo A0 el pin 11, el A1 el pin 10 y el A2 el pin 9.

Parte 3

```

1 { Serial.begin(9600);
2 while (!Serial) delay(1); // Esperando por el serial encendido Leonardo/Zero ,etc
3 Serial.println("MAX31855 test");
4 // Espera a que el chip Max se estabilice .
5 delay(500);
6 Serial.print("Initializing sensor... ");
7 if (!thermocouple.begin()) {
8   Serial.println("ERROR. ");
9   while (1) delay(10);
10 }
11 Serial.println("Listo .");
12 }

```

En esta parte a grandes rasgos se inicializa la comunicación serial a 9600 y se espera a que se estabilice.

Parte 4

```
1 void loop() {
2     Serial.println("");
3     for (int count=0; count < cantArray; count++) {
4         int row = bin[count];
5         r0 = bitRead(row,0); //
6         r1 = bitRead(row,1); //canal 7 = 111, 1 = 2nd bit
7         r2 = bitRead(row,2); // tercer bit
8         digitalWrite(11, r0);
9         digitalWrite(10, r1);
10        digitalWrite(9, r2);
11        delay (1000); // tiempo para leer y poner los valores
12        double c =thermocouple.readCelsius();
13        if (isnan(c)) {
14            Serial.print(0); // "algo va mal con elthermocouple ");
15        }
16        else {
17            Serial.print(c);
18            Serial.write('\t');
19        }
20    }
21 }
```

En la linea 1 empieza la función void loop.

En la linea 2 se imprime un salto de linea esto para el empaquetado y envío de datos.

En la linea 3 funciona para cambiar de canal en nuestro multiplexor empieza desde 0 y aumenta un contador count hasta la cantidad de números de termopares a revisar(cantArray) inicializada y asignada con el valor anteriormente.

En la linea 4 se podría decir que se cambia de decimal a binario.

De las lineas 5 a 7 se sacan los bits de la variable row poniendo el bit menos significativo en r0, seguido del 2do bit poniéndose en la variable r1 y por ultimo el bit mas significativo en la variable r2. Para tener mas claro esto a continuación se pone un ejemplo.

Ejemplo

```
1 row=5dec
2 5dec=101bin
```

siendo el bit menos significativo el 1 que esta a la derecha este se pondría en r0 siguiendo con r1=0 y por ultimo r2=1.

En las lineas 8 a 10 se manda los bits de r0, r1, r2 hacia los pines 11, 10, 9 respectivamente que a su vez mandan la señal al multiplexor para cambiar de canal y poder analizar otro termopar.

En la linea 11 se hace un delay con la intención de esperar la medición por parte del circuito integrado MAX31855

En la linea 12 se ejecuta la función para la medición de la señal del termopar y se guarda en una variable double c.

En las ultimas lineas se revisa si hay señal por parte del termopar en caso de que no hubiese se imprime un 0 en el serial port en caso de efectuarse una buena medición se imprime el dato en el puerto serie y posteriormente se agrega un espacio esto para el empaquetado de datos.

3.2. Código-Arduino

```

1
2 #include <SPI.h>
3 #include "Adafruit_MAX31855.h"
4 // Definicion de los pines conectados con el Max41855
5 #define MAXDO 3 //miso
6 #define MAXCS 4 // mosi
7 #define MAXCLK 5 //sck in
8 //definicion de las variables para la escritura
9 //de nuestras entradas de seleccion del multiplexor
10 int r0 = 0; //A0
11 int r1 = 0; // A1 the three digital pins for the bits
12 int r2 = 0; //A2
13 //Banco de valores para la la seleccion de nuestro canal pueden ir en desorden.
14 //Sabiendo que se escribiran en binario ejemplo 7DEC=111BIN.
15 int bin [] = {0,1,2,3,4,5}; //list of binary values __lista de valores binarios 1, 2,
   3,5, 6,
16 int cantArray=6;
17 // inizializacion del termoacople
18 Adafruit_MAX31855 thermocouple(MAXCLK, MAXCS, MAXDO);
19 // creacion de la peticion SPI del thermocouple con el hardware
20 void setup() {
21 #define PINEN 8 //Mux Enable pin
22 // Bits para las entradas de seleccion del multiplexor
23 digitalWrite(PINEN, HIGH); // enable on
24 pinMode(11, OUTPUT); // A0 //Mux Address 0 pin
25 pinMode(10, OUTPUT); // A1 Mux Address 1 pi
26 pinMode(9, OUTPUT); // A2 Mux Address 2 pin
27
28 Serial.begin(9600);
29 while (!Serial) delay(1); // espera por Serial en Leonardo/Zero, etc
30
31 Serial.println("MAX31855 test");
32 // wait for MAX chip to stabilize
33 delay(500);
34 Serial.print("Initializing sensor... ");
35 if (!thermocouple.begin()) {
36 Serial.println("ERROR.");
37 while (1) delay(10);
38 }
39 Serial.println("Listo.");
40 }
41 void loop() {
42 Serial.println("");
43 for (int count=0; count < cantArray; count++) {
44 int row = bin[count]; //
45 r0 = bitRead(row,0); //
46 r1 = bitRead(row,1); //
47 r2 = bitRead(row,2); //
48 digitalWrite(11, r0); //
49 digitalWrite(10, r1);
50 digitalWrite(9, r2);
51 delay (100); //
52 double c =thermocouple.readCelsius();
53 if (isnan(c)) {
54 Serial.print(0);
55 }
56 else {
57 Serial.print(c);
58 Serial.write('\t');
59 }
60 }
61 }

```

Listing 1. Código completo

3.3. Programa-processing

A continuación se va explicar el código de processing por partes y al final de esto se pondrá el código completo.

Parte 1

```
1 import processing.serial.Serial;
2 import grafica.*;
3 import java.util.Random;
4
5 public GPlot plot2 ;
6 static final int PORT_INDEX = 0 , BAUDS = 9600;
7 int i=0;
8 float[] vals = {};
9 int[] x = { 0, 1, 2, 3, 4,5,6,7};
```

De la linea 1-3 Se importan todas las librerías necesarias para el correcto funcionamiento del programa, en la linea 5 se crea un plano y se llama plot2, en la linea 6 se ponen los paramentos para la interconexión PORT INDEX la posición del puerto que se va ocupar !Nota!:Para sistemas windows normalmente se localiza el puerto en esa posición 0 sin embargo esta puede variar para sistemas Linux y los se tiene que localizar manualmente esta posición. A continuación se ponen los batíos por segundo en este caso son 9600/seg, en la linea 7 se crea una variable i que nos va a servir mas adelante como contador, en la linea 8 se crea una variable de vector flotante con el nombre de vals esto con la intención de guardar y procesar los valores de lectura del serial y por ultimo una variable de vector entera con el único propósito de colocar el nombre del canal por lo que se necesita poner la misma cadena de números que en la linea 1 de la pagina 5 vista anteriormente.

Parte 2

```
1 Table table;           // crear la tabla de datos
2 //Se inicializan las variables
3 GPointsArray points2 = new GPointsArray();
4 GPointsArray points3 = new GPointsArray();
5 GPointsArray points4 = new GPointsArray();
6 GPointsArray points5 = new GPointsArray();
7 GPointsArray points6 = new GPointsArray();
8 GPointsArray points7 = new GPointsArray();
9 GPointsArray points8 = new GPointsArray();
```

En la linea 1 se crea un objeto Table y lo llamamos table, en la lineas 3-9 se crean arreglos GpointsArray y se llaman Points(2-8) para inicializarlos como New GpointsArray esto para que no haya error por no tener ningún valor antes de iniciar la operación.

Parte 3 Void setup 1

```

1 void setup() {
2   size(850, 660);
3   noLoop();
4   final String[] ports = Serial.list();
5   printArray(ports);
6   new Serial(this, ports[PORT_INDEX], BAUDS).bufferUntil(ENTER);
7   table = new Table(); //se crean las columnas necesarias
8   if (x.length>0) {
9     table.addColumn("canal"+x[0]);
10    if (x.length>1) {
11      table.addColumn("canal"+x[1]);
12      if (x.length>2) {
13        table.addColumn("canal"+x[2]);
14        if (x.length>3) {
15          table.addColumn("canal"+x[3]);
16          if (x.length>4) {
17            table.addColumn("canal"+x[4]);
18            if (x.length>5) {
19              table.addColumn("canal"+x[5]);
20              if (x.length>=6) {
21                table.addColumn("canal"+x[6]);
22                if (x.length>=7) {
23                  table.addColumn("canal"+x[7]);
24                }
25              }
26            }
27          }
28        }
29      }
30    }
31  }
}

```

En la sección de void setup se configura y se cargan los parámetros iniciales, en la linea dos se tiene la configuración del tamaño de la pantalla de la aplicación, seguido de la linea 2 con un noloop o no retorno, en las lineas 4-5 se inspecciona los puertos conectados y se imprime la lista en la barra de comandos esto únicamente se debe analizar en caso de que con la configuración ya propuesta no se encuentra el puerto deseado, en la linea 6 se tiene la configuración de comunicación con el puerto, en la linea 7 se crea una tabla para guardar los datos y en las lineas 8-31 se añaden las columnas necesarias, esto, pues solo sera la cantidad de números que se encuentren en el vector "x" poniendo como nombre canal- "la posición del vector".

Parte 3 Void setup 2

```

1 plot2 = new GPlot(this);
2 plot2.setPos(0, 0);
3 plot2.setDim(550, 550);
4 plot2.getTitle().setText("Posicion");
5 plot2.getAxisX().getAxisLabel().setText("X");
6 plot2.getAxisY().getAxisLabel().setText("Y");
7 plot2.setLineWidth(1.5);

```

En la linea 1 se agrega valor de "new GPlot(this)." esto para no generar errores con valores vacíos. en la linea 2 se tiene la posición del plano dentro de la dimensión de la aplicación, en la linea 3 se tiene la dimensión del plano, en las lineas 4-6 se configuran los nombres de los ejes y del plano. Por ultimo se tiene la configuración del grosor de la linea que se grafica

Parte 3 Void setup 3

```

1 plot2.addLayer("layer 2", points2);
2 plot2.getLayer("layer 2").setLineColor(color(255, 150, 150));
3 plot2.getLayer("layer 2").setLineWidth(1.5);
4 //
5 plot2.addLayer("layer 3", points3);
6 plot2.getLayer("layer 3").setLineColor(color(150, 255, 150));
7 plot2.getLayer("layer 3").setLineWidth(1.5);
8 //
9 plot2.addLayer("layer 4", points4);
10 plot2.getLayer("layer 4").setLineColor(color(150, 150, 255));
11 plot2.getLayer("layer 4").setLineWidth(1.5);
12 //
13 plot2.addLayer("layer 5", points5);
14 plot2.getLayer("layer 5").setLineColor(color(250, 15, 15));
15 plot2.getLayer("layer 5").setLineWidth(1.5);
16 //
17 plot2.addLayer("layer 6", points6);
18 plot2.getLayer("layer 6").setLineColor(color(15, 250, 15));
19 plot2.getLayer("layer 6").setLineWidth(1.5);
20 //
21 plot2.addLayer("layer 7", points7);
22 plot2.getLayer("layer 7").setLineColor(color(15, 15, 250));
23 plot2.getLayer("layer 7").setLineWidth(1.5);
24 //
25 plot2.addLayer("layer 8", points8);
26 plot2.getLayer("layer 8").setLineColor(color(0,255,0));
27 plot2.getLayer("layer 8").setLineWidth(1.5);

```

En esta sección se configura los puntos que se grafican. En la linea 1 se añade una etiqueta al arreglo de puntos, en la linea 2 se define los colores con la siguiente definición color(Rojo(0-255) verde(0-255) azul(0-255)), en la linea 4 se configura el grosor de los puntos. se las lineas 5-27 se repiten este mismo proceso para cada uno de los arreglos de puntos.

Parte 4 void draw 1

```

1 void draw() {
2 background(0);
3 // Dibujo del plano
4 plot2.beginDraw();
5 plot2.drawBackground();
6 plot2.drawBox();
7 plot2.drawXAxis();
8 plot2.drawYAxis();
9 plot2.drawTitle();
10 plot2.drawGridLines(GPlot.CTRLMOD );
11 plot2.drawLines();
12 plot2.endDraw();
13 textSize(20);}

```

En la linea 2 se dibuja el fondo de la aplicación, sobre este fondo se dibuja el plano que comprende desde la linea 5 a 12, en la linea 13 se configura un tamaño de letra fuera del plano.

Parte 4 void draw 2

```

1 { if (vals.length >= 1) {
2     i++;
3     TableRow newRow = table.addRow();
4     plot2.addPoint(i, vals[0]);
5     newRow.setFloat(0, vals[0]);
6     text("Chanel" + x[0] + "=" + vals[0] + " c ", 660, 150);
7     if (vals.length >= 2) {
8         plot2.addPoint(Point(i, vals[1]), "layer 2");
9         newRow.setFloat(1, vals[1]);
10        text("Chanel " + x[1] + "=" + vals[1] + " c ", 660, 200);
11        if (vals.length >= 3) {
12            plot2.addPoint(Point(i, vals[2]), "layer 3");
13            newRow.setFloat(2, vals[2]);
14            text("Chanel " + x[2] + "=" + vals[2] + " c ", 660, 250);
15            if (vals.length >= 4) {
16                plot2.addPoint(Point(i, vals[3]), "layer 4");
17                newRow.setFloat(3, vals[3]);
18                text("Chanel " + x[3] + "=" + vals[3] + " c ", 660, 300);
19                if (vals.length >= 5) {
20                    plot2.addPoint(Point(i, vals[4]), "layer 5");
21                    newRow.setFloat(4, vals[4]);
22                    text("Chanel " + x[4] + "=" + vals[4] + " c ", 660, 350);
23                    if (vals.length >= 6) {
24                        plot2.addPoint(Point(i, vals[5]), "layer 6");
25                        newRow.setFloat(5, vals[5]);
26                        text("Chanel " + x[5] + "=" + vals[5] + " c ", 660, 400);
27                        if (vals.length >= 7) {
28                            plot2.addPoint(Point(i, vals[6]), "layer 7");
29                            newRow.setFloat(6, vals[6]);
30                            text("Chanel " + x[6] + "=" + vals[6] + " c ", 660, 450);
31
32                            if (vals.length >= 8) {
33                                plot2.addPoint(Point(i, vals[7]), "layer 8");
34                                newRow.setFloat(7, vals[7]);
35                                text("Chanel " + x[7] + "=" + vals[7] + " c ", 660, 500);
36
37                        }
38                    }
39                }
40            }
41        }
42    }
43}
44 } text("Press S to save", 660, 600);
45 }
46 }
```

En la linea 1 se pregunta si el vector de datos recibidos es mayor a 1, en caso de que que si en la linea 2 se aumenta el contador de i"que nos ayudara como valor para el eje x, en la linea 3 se añade una fila nueva a nuestra tabla donde se guardaran los valores, en la linea 4 se añaden los valores del punto 1 en el plano, en la linea 5 se añade los valores a la tabla siendo "newRow.setFloat(el numero de la columna(0-7), valor recibido correspondiente a la posición del vector Vals)", en la linea 6 se crea un texto que refleja el valor de la temperatura del canal, de las lineas 7-44 se repite este mismo proceso para los diferentes canales activos y por ultimo se dibuja un texto para intuir al usuario que presionando S se guarda la tabla en un formato de excel.

Parte 5

```

1 void serialEvent(final Serial s) {
2     vals = float(splitTokens(s.readString()));
3     redraw = true;
4 }
5 GPoint Point(float i, float n) {
6     return new GPoint(i, n);
7 }
8 void keyPressed() {
9     saveTable(table, "data/" + year() + "-" + nf(month(), 2, 0) + "-" +
10    nf(day(), 2, 0) + "--" + nf(hour(), 2, 0) + "-" + nf(minute(), 2, 0) +
11    "-" + nf(second(), 2, 0) + "_" + millis() + ".csv");
12 }

```

En la linea 1 se tiene una función de serial event la cual se activa cada vez que hay actividad en el serial en la linea 2 se tiene la igualación del vector vals con el vector recibido por parte del serial, en la linea 4 se tiene el redraw que hace volver a dibujar todo lo que esta dentro de la función "Void draw", en la linea6-7 se tiene una función necesaria para el acople de los puntos al plano en la linea 9 se tiene la función keypressed la cual se activa cada que se presiona cualquier letra, en la linea 10-12 una vez presionada cualquier letra se guarda la tabla con los valores de temperatura siendo "SaveTable(Nombre de la tabla, Nombre de la carpeta + el nombre y tipo de archivo).en este caso se sugirió poner como nombre del archivo la fecha y hora en la que se presiona la letra.

3.4. Código-Processing

```

1 import processing.serial.Serial;
2 import grafica.*;
3 import java.util.Random;
4 public GPlot plot2;
5 static final int PORT_INDEX = 0, BAUDS = 9600;
6 int i=0;
7 float[] vals = {};
8 int[] x = { 0, 1, 2, 3, 4, 5, 6, 7 };
9 //
10 Table table;           // crear la tabla de datos
11 //Se inicializan las variables
12 GPointsArray points2 = new GPointsArray();
13 GPointsArray points3 = new GPointsArray();
14 GPointsArray points4 = new GPointsArray();
15 GPointsArray points5 = new GPointsArray();
16 GPointsArray points6 = new GPointsArray();
17 GPointsArray points7 = new GPointsArray();
18 GPointsArray points8 = new GPointsArray();
19 void setup() {
20     size(850, 660);
21     noLoop();
22     final String[] ports = Serial.list();
23     printArray(ports);
24     new Serial(this, ports[PORT_INDEX], BAUDS).bufferUntil(ENTER);
25     table = new Table(); //se crean las columnas necesarias
26     if (x.length>0) {
27         table.addColumn("canal"+x[0]);
28         if (x.length>1) {
29             table.addColumn("canal"+x[1]);
30             if (x.length>2) {
31                 table.addColumn("canal"+x[2]);
32                 if (x.length>3) {
33                     table.addColumn("canal"+x[3]);

```

Listing 2. Código Processing Parte 1

[Continua]

[Sigue]

```

1      if (x.length>4) {
2          table.addColumn("canal"+x[4]);
3          if (x.length>5) {
4              table.addColumn("canal"+x[5]);
5              if (x.length>=6) {
6                  table.addColumn("canal"+x[6]);
7                  if (x.length>=7) {
8                      table.addColumn("canal"+x[7]);
9                  }
10             }
11         }
12     }
13 }
14 }
15 }
16 }
17 plot2 = new GPlot(this);
18 plot2.setPos(0, 0);
19 plot2.setDim(550, 550);
20 plot2.getTitle().setText("Posicion");
21 plot2.getXAxis().getAxisLabel().setText("X");
22 plot2.getYAxis().getAxisLabel().setText("Y");
23 plot2.setLineWidth(1.5);
24 //
25 plot2.addLayer("layer 2", points2);
26 plot2.getLayer("layer 2").setLineColor(color(255, 150, 150));
27 plot2.getLayer("layer 2").setLineWidth(1.5);
28 //
29 plot2.addLayer("layer 3", points3);
30 plot2.getLayer("layer 3").setLineColor(color(150, 255, 150));
31 plot2.getLayer("layer 3").setLineWidth(1.5);
32 //
33 plot2.addLayer("layer 4", points4);
34 plot2.getLayer("layer 4").setLineColor(color(150, 150, 255));
35 plot2.getLayer("layer 4").setLineWidth(1.5);
36 //
37 plot2.addLayer("layer 5", points5);
38 plot2.getLayer("layer 5").setLineColor(color(250, 15, 15));
39 plot2.getLayer("layer 5").setLineWidth(1.5);
40 //
41 plot2.addLayer("layer 6", points6);
42 plot2.getLayer("layer 6").setLineColor(color(15, 250, 15));
43 plot2.getLayer("layer 6").setLineWidth(1.5);
44 //
45 plot2.addLayer("layer 7", points7);
46 plot2.getLayer("layer 7").setLineColor(color(15, 15, 250));
47 plot2.getLayer("layer 7").setLineWidth(1.5);
48 //
49 plot2.addLayer("layer 8", points8);
50 plot2.getLayer("layer 8").setLineColor(color(0,255,0));
51 plot2.getLayer("layer 8").setLineWidth(1.5);
52 //
53 plot2.activatePanning();
54 plot2.activateZooming(1.2, CENTER, CENTER);
55 plot2.activateReset();
56 void draw() {
57 background(0);
58 // dibujar el plano
59 plot2.beginDraw();

```

Listing 3. Código Processing Parte 2

[Continua]

[Sigue]

```

1  plot2.drawBackground();
2  plot2.drawBox();
3  plot2.drawXAxis();
4  plot2.drawYAxis();
5  plot2.drawTitle();
6  plot2.drawGridLines(GPlot.CTRLMOD );
7  plot2.drawLines();
8  plot2.endDraw();
9  textSize(20);
10 if (vals.length >= 1) {
11   i++;
12   TableRow newRow = table.addRow();
13   plot2.addPoint(i, vals[0]);
14   newRow.setFloat(0, vals[0]);
15   text("Chanel " + x[0] + "=" + vals[0] + " c ", 660, 150);
16   if (vals.length >= 2) {
17     plot2.addPoint(Point(i, vals[1]), "layer 2");
18     newRow.setFloat(1, vals[1]);
19     text("Chanel " + x[1] + "=" + vals[1] + " c ", 660, 200);
20   if (vals.length >= 3) {
21     plot2.addPoint(Point(i, vals[2]), "layer 3");
22     newRow.setFloat(2, vals[2]);
23     text("Chanel " + x[2] + "=" + vals[2] + " c ", 660, 250);
24   if (vals.length >= 4) {
25     plot2.addPoint(Point(i, vals[3]), "layer 4");
26     newRow.setFloat(3, vals[3]);
27     text("Chanel " + x[3] + "=" + vals[3] + " c ", 660, 300);
28   if (vals.length >= 5) {
29     plot2.addPoint(Point(i, vals[4]), "layer 5");
30     newRow.setFloat(4, vals[4]);
31     text("Chanel " + x[4] + "=" + vals[4] + " c ", 660, 350);
32   if (vals.length >= 6) {
33     plot2.addPoint(Point(i, vals[5]), "layer 6");
34     newRow.setFloat(5, vals[5]);
35     text("Chanel " + x[5] + "=" + vals[5] + " c ", 660, 400);
36   if (vals.length >= 7) {
37     plot2.addPoint(Point(i, vals[6]), "layer 7");
38     newRow.setFloat(6, vals[6]);
39     text("Chanel " + x[6] + "=" + vals[6] + " c ", 660, 450);
40
41   if (vals.length >= 8) {
42     plot2.addPoint(Point(i, vals[7]), "layer 8");
43     newRow.setFloat(7, vals[7]);
44     text("Chanel " + x[7] + "=" + vals[7] + " c ", 660, 500);
45
46   }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 text("Press S to save", 660, 600);
55 }
56
57 void serialEvent(final Serial s) {
58   // vals = int(splitTokens(s.readString()));
59   vals = float(splitTokens(s.readString()));
60   redraw = true;
61 }
```

Listing 4. Código Processing Parte 3

[Continua]

[Sigue]

```
1 GPoint Point(float i, float n) {  
2     return new GPoint(i, n);  
3 }  
4 void keyPressed() {  
5     saveTable(table, "data/" + year() + "-" + nf(month(), 2, 0) + "-" + nf(day(), 2, 0) + "--"  
+ nf(hour(), 2, 0) + "-" + nf(minute(), 2, 0) + "-" + nf(second(), 2, 0) + "_" + millis  
() + ".csv");  
6 }
```

Listing 5. Código Processing Parte 4