Escuela Superior Politécnica del Litoral

Laboratorio de Diseño de Sistemas Digitales

Guía del proyecto

Machine learning para detección de lenguaje de señas usando datos EMG

Nombre: Carlos Cedeño
Miguel Daquilema

Prof: Ing. Víctor Asanza.

Objetivos

Implementar un Sistema Embebido para usarlo en la detección de lenguaje de señas, utilizando el equipo MYO provisto por el profesor, para medir vía bluetooth señales de electromiografía.

Implementar un proceso de entrenamiento inicial automático para cada persona.

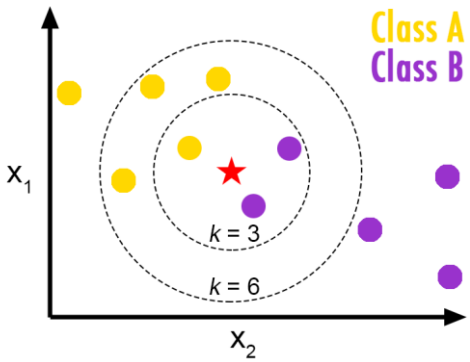
Información

Descripción

La Organización Mundial de la Salud (OMS) especifica que más del 5% de la población mundial, aproximadamente 360 millones de personas, padece pérdida de audición discapacitante. En Ecuador existe a nivel nacional 55.020 personas con discapacidad auditiva. El proyecto propuesto usa el dispositivo MYO de la firma canadiense Thalmic Labs cuyo precio oscila en los $300 y cuenta con giroscopio, acelerómetro, magnetómetro y 8 sensores EMG, los cuales proporcionan datos representativos para entrenar un algoritmo de aprendizaje de máquina y desarrollar una aplicación para detección en tiempo real del lenguaje de señas.

Algoritmo

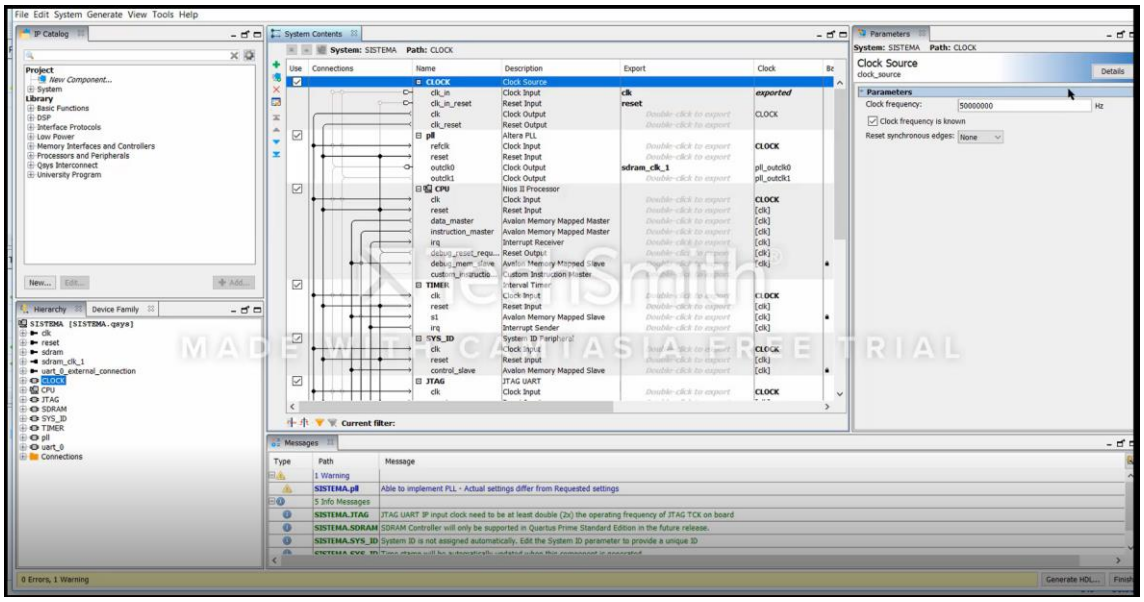
Pertenece a la familia de Aprendizaje supervisado, donde se debe proporcionar un dataset para entrenamiento y posterior clasificación. El algoritmo agrupa puntos del mismo grupo de acuerdo con un identificador de clase.



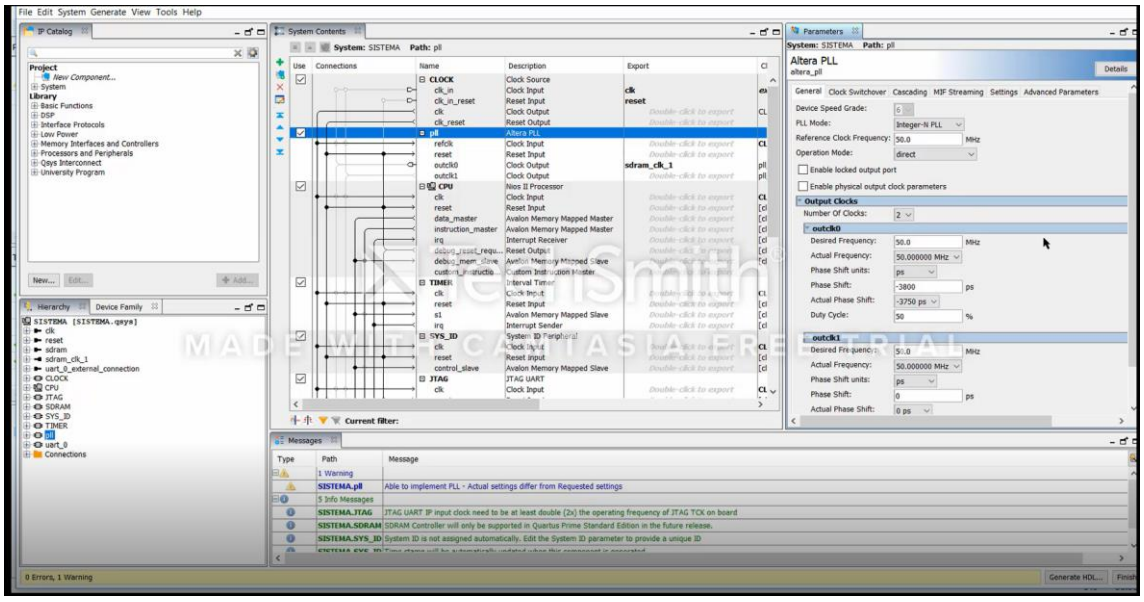
Al introducir un punto para su clasificación se calcula la distancia euclidiana hacia los puntos de entrenamiento y se seleccionan los K puntos más cercanos para estimar su afinidad a un grupo en particular.

Desarrollo

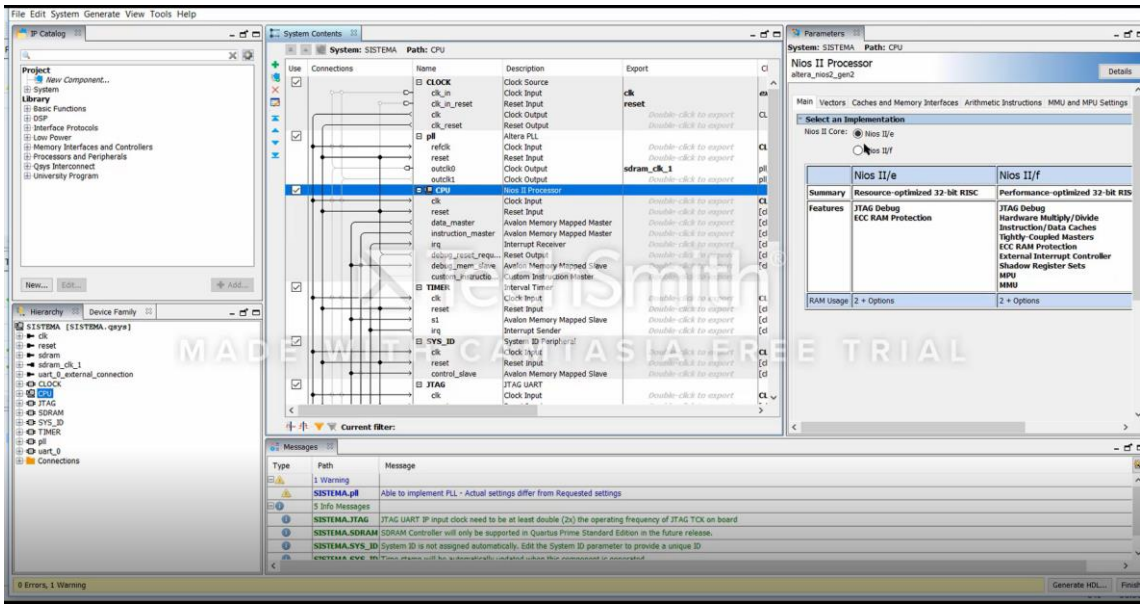
1. En la ventana de CLOCK se define como parámetro una frecuencia de 50MHz.



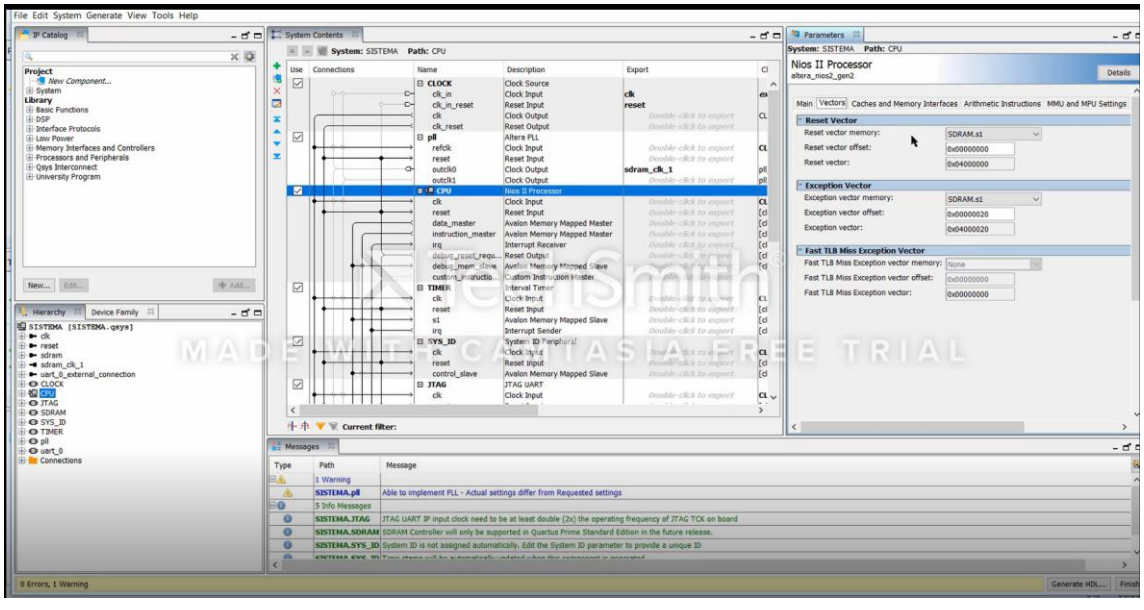
2. En la ventana de PLL, se configura la misma frecuencia del reloj de la tarjeta de desarrollo y se configuran los parámetros de lo Output Clocks tal como se muentra.



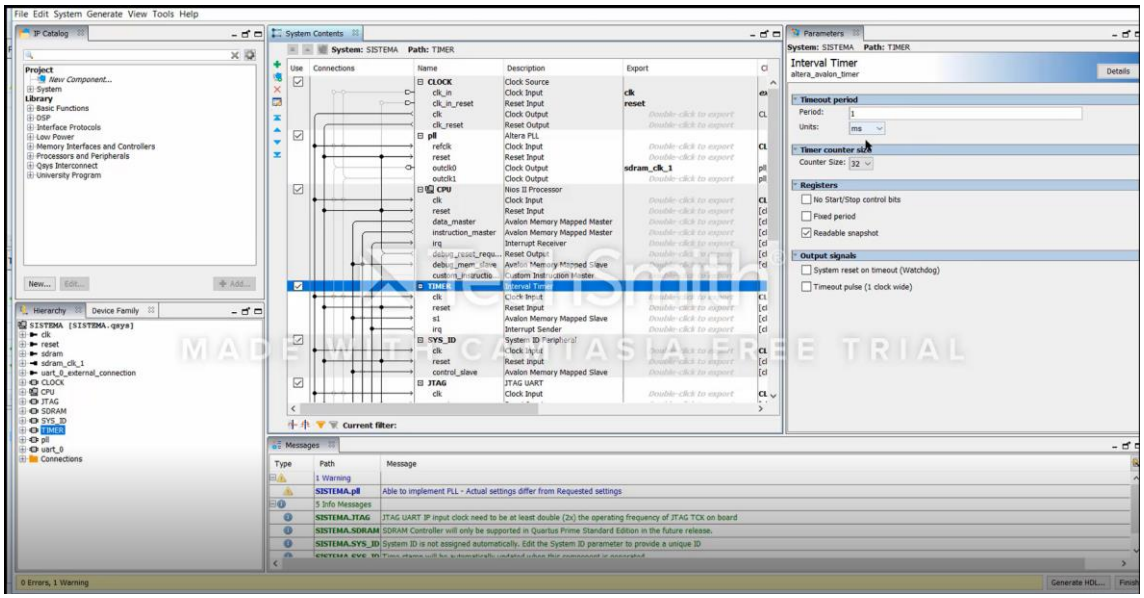
3. En la ventana de CPU, en el apartado de Main se selecciona como core Nios II/e.



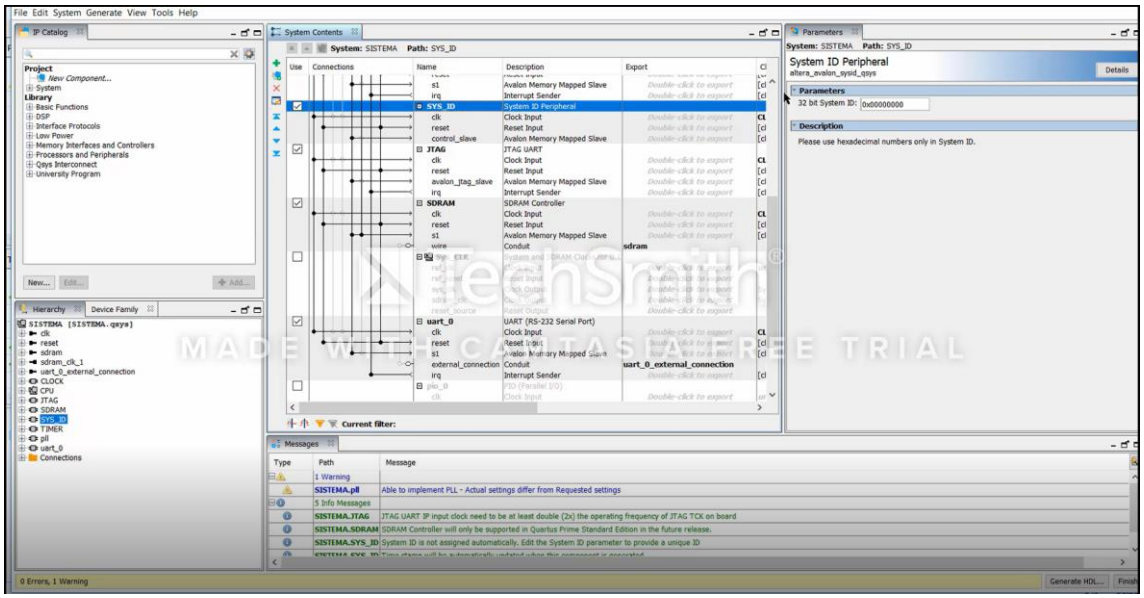
4. En la ventana de CPU, en el apartado de Vectors se asigna la dirección del vector de reset de la SDRAM



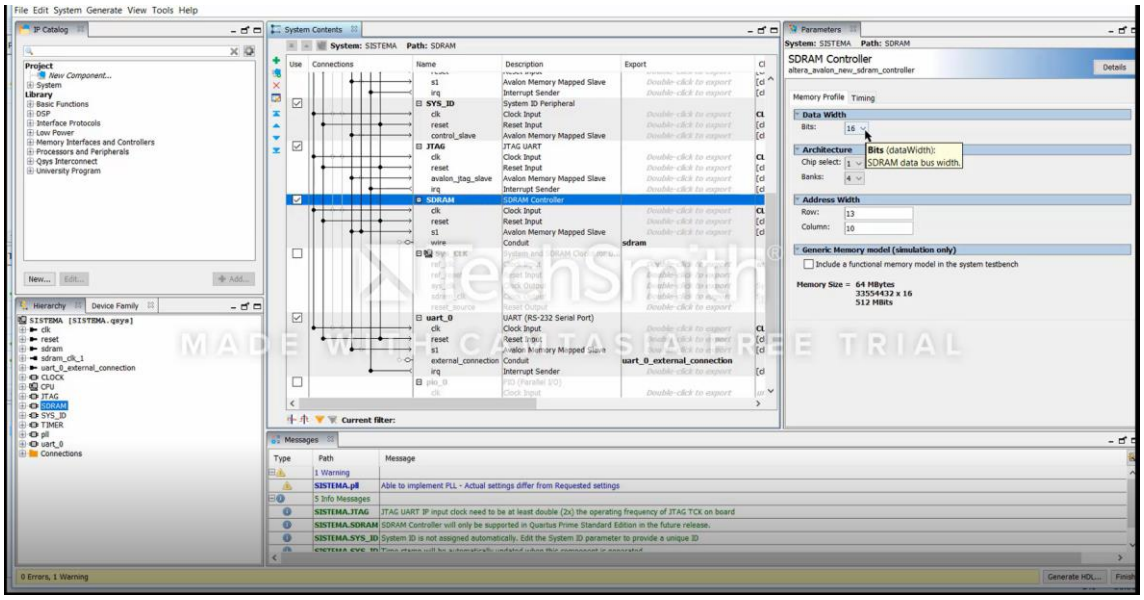
5. En la ventana de TIMER, se define un periodo de 1 ms.



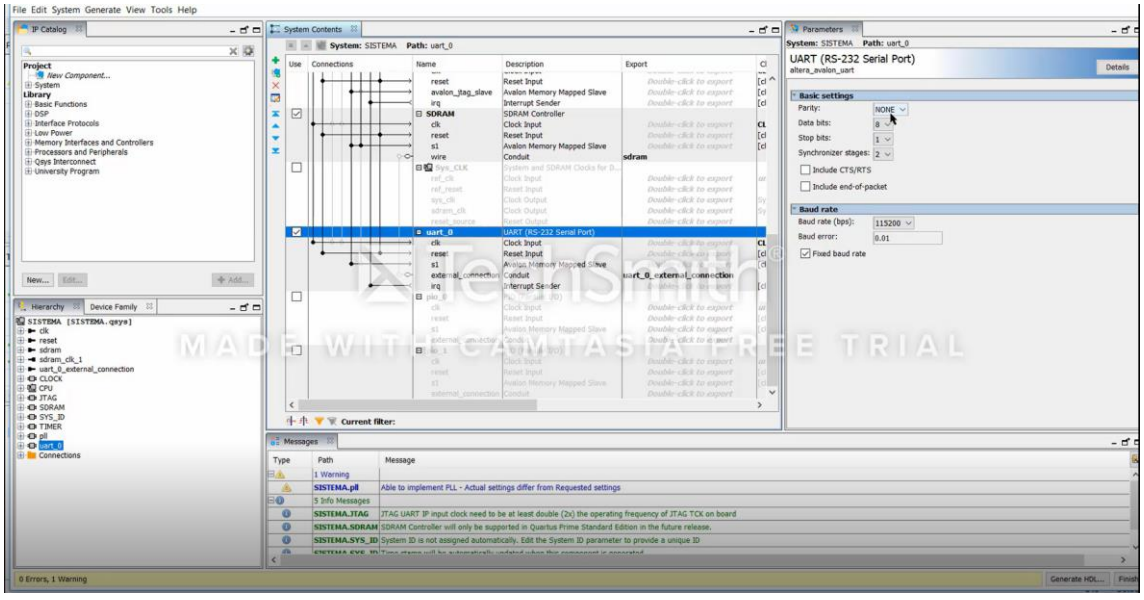
6. En la ventana de SYS_ID, se define la dirección del periférico.



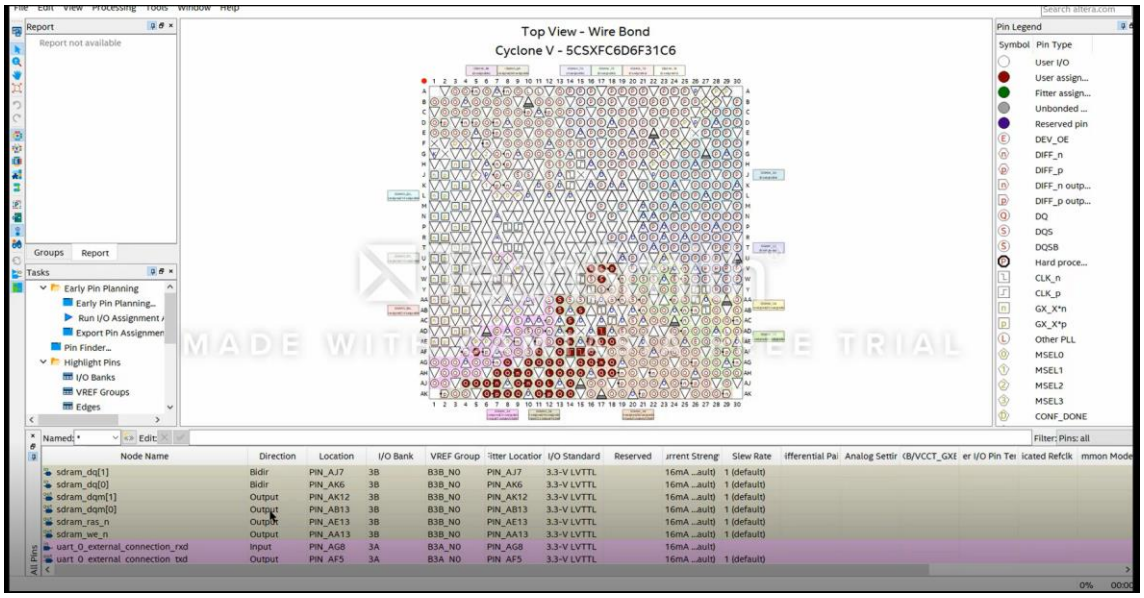
7. En la ventana de SDRAM, se deben configurar los parámetros tal como se muestran.



8. En la ventana de uart_0, se deben configurar los parámetros tal como se muestran.



9. En la ventana de PIN PLANNER se debe direccionar los pines de uart_0 tal como se muestran para entrada y salida.



10. Compilar en VHDL.
11. Abrir la herramienta de NIOS II tool ECLIPSE
12. Parte 1: Se definen librerias y la estructura de PUNTO (Point que se va a usar) junto con las variables que representan este objeto.

```
#include <bits/stdc++.h>

#include <string>
#include <vector>
#include <sstream> //istringstream
#include <iostream> // cout
#include <fstream> // ifstream
#include "altera_avalon_uart_regs.h"
#include "system.h"
#include <stdio.h>
using namespace std;

struct Point
{
    int val;     // Group of point
    //double s1_s, s2_s,s3_s,s4_s,s5_s,s6_s,s7_s,s8_s;
    //double s1_p, s2_p,s3_p,s4_p,s5_p,s6_p,s7_p,s8_p;
    //double s1_m, s2_m,s3_m,s4_m,s5_m,s6_m,s7_m,s8_m;
    double s1_r, s2_r,s3_r,s4_r,s5_r,s6_r,s7_r,s8_r;// Co-ordinate of point
    double distance; // Distance from test point
};
```

13. Parte 2: Proceso de parseo entre string proveniente de dataset de entrenamiento a variable tipo vector<vector<double>>.

```
stringstream BASE(dataset);
    string item;
    vector<string> s;
    while (getline(BASE, item, '\n'))
    {
       s.push_back(item);
    }
    for (auto ct : s){
        vector<double> record;
        stringstream ss(ct);
        vector<string> line;
        string fields;

                        while (getline(ss, fields, ','))
                                {
                                   line.push_back(fields);
                                }
                        for (string field : line)
                            record.push_back(strtof(field.c_str(),0));

                        data.push_back(record);
                        }
```

14. Parte 3.1: Algoritmo de clasificación, en el método se recibe como parámetro un array con todos los puntos de entrenamiento, el numero de puntos, el parámetro K y el punto actual. Se ordena el arreglo dependiendo de las distancias encontradas.

```
int classifyAPoint(Point arr[], int n, int k, Point p)
{
    for (int i = 0; i < n; i++)
        arr[i].distance =
            sqrt(//(arr[i].s1_s - p.s1_s) * (arr[i].s1_s - p.s1_s) +
                 //(arr[i].s1_p - p.s1_p) * (arr[i].s1_p - p.s1_p) + (arr[i].s1_m - p.s1_m) * (arr[i].s1_m - p.s1_m) +
                 (arr[i].s1_r - p.s1_r) * (arr[i].s1_r - p.s1_r) + //(arr[i].s2_s - p.s2_s) * (arr[i].s2_s - p.s2_s) +
                 //(arr[i].s2_p - p.s2_p) * (arr[i].s2_p - p.s2_p) + (arr[i].s2_m - p.s2_m) * (arr[i].s2_m - p.s2_m) +
                 (arr[i].s2_r - p.s2_r) * (arr[i].s2_r - p.s2_r) + //(arr[i].s3_s - p.s3_s) * (arr[i].s3_s - p.s3_s) +
                 //(arr[i].s3_p - p.s3_p) * (arr[i].s3_p - p.s3_p) + (arr[i].s3_m - p.s3_m) * (arr[i].s3_m - p.s3_m) +
                 (arr[i].s3_r - p.s3_r) * (arr[i].s3_r - p.s3_r) + //(arr[i].s4_s - p.s4_s) * (arr[i].s4_s - p.s4_s) +
                 //(arr[i].s4_p - p.s4_p) * (arr[i].s4_p - p.s4_p) + (arr[i].s4_m - p.s4_m) * (arr[i].s4_m - p.s4_m) +
                 (arr[i].s4_r - p.s4_r) * (arr[i].s4_r - p.s4_r) + //(arr[i].s5_s - p.s5_s) * (arr[i].s5_s - p.s5_s) +
                 //(arr[i].s5_p - p.s5_p) * (arr[i].s5_p - p.s5_p) + (arr[i].s5_m - p.s5_m) * (arr[i].s5_m - p.s5_m) +
                 (arr[i].s5_r - p.s5_r) * (arr[i].s5_r - p.s5_r) + //(arr[i].s6_s - p.s6_s) * (arr[i].s6_s - p.s6_s) +
                 //(arr[i].s6_p - p.s6_p) * (arr[i].s6_p - p.s6_p) + (arr[i].s6_m - p.s6_m) * (arr[i].s6_m - p.s6_m) +
                 (arr[i].s6_r - p.s6_r) * (arr[i].s6_r - p.s6_r) + //(arr[i].s7_s - p.s7_s) * (arr[i].s7_s - p.s7_s) +
                 //(arr[i].s7_p - p.s7_p) * (arr[i].s7_p - p.s7_p) + (arr[i].s7_m - p.s7_m) * (arr[i].s7_m - p.s7_m) +
                 (arr[i].s7_r - p.s7_r) * (arr[i].s7_r - p.s7_r) + //(arr[i].s8_s - p.s8_s) * (arr[i].s8_s - p.s8_s) +
                 //(arr[i].s8_p - p.s8_p) * (arr[i].s8_p - p.s8_p) + (arr[i].s8_m - p.s8_m) * (arr[i].s8_m - p.s8_m) +
                 (arr[i].s8_r - p.s8_r) * (arr[i].s8_r - p.s8_r));

    // Ordenar puntos
    sort(arr, arr+n, comparison);
```

15. Parte 3.2: Una vez ordenado los puntos por las distancias se asocian estas distancias a una frecuencia identificadora de grupo. Se recorre el arreglo de puntos hasta el *k* elemento y se contabilizan la frecuencia de ocurrencia.

```
// Considerar la frecuencia de los puntos
int freq1 = 0;          // Frecuencia del grupo 0   A
    int freq2 = 0;      // Frecuencia del grupo 1   B
    int freq3= 0;       // Frecuencia del grupo 2   I
    int freq4= 0;       // Frecuencia del grupo 3   L
    int freq5= 0;       // Frecuencia del grupo 4   N
    int freq6= 0;       // Frecuencia del grupo 5   O
    int freq7= 0;       // Frecuencia del grupo 6   P
    int freq8= 0;       // Frecuencia del grupo 7    Q
    int freq9= 0;       // Frecuencia del grupo 8    U
    for (int i = 0; i < k; i++)
    {
        if (arr[i].val == 0)
            freq1++;
        else if (arr[i].val == 1)
            freq2++;
        else if (arr[i].val == 2)
            freq3++;
        else if (arr[i].val == 3)
                freq4++;
        else if (arr[i].val == 4)
                freq5++;
        else if (arr[i].val == 5)
                freq6++;
        else if (arr[i].val == 6)
                freq7++;
        else if (arr[i].val == 7)
                freq8++;
        else if (arr[i].val == 8)
                        freq9++;
    }
```

16. Parte 3.3: De acuerdo con las frecuencias encontradas devuelve el valor de la clasificación de acuerdo al identificador del grupo.

```
}
//Returning highest freq
if (freq1>freq2 and freq1 > freq3 and freq1 > freq4 and freq1 > freq5  and freq1 > freq6 and freq1 > freq7 and freq1 > fre
    return (0);
if (freq2>freq1 and freq2 > freq3 and freq2 > freq4  and freq2 > freq5 and freq2 > freq6  and freq2 > freq7 and freq2 > fr
    return (1);
if (freq3>freq1 and freq3 > freq2 and freq3 > freq4 and freq3 > freq5 and freq3 > freq6 and freq3 > freq7 and freq3 > freq
    return (2);
if (freq4>freq1 and freq4 > freq2 and freq4 > freq3  and freq4 > freq5 and freq4 > freq6 and freq4 > freq7 and freq4 > fre
        return (3);
if (freq5>freq1 and freq5 > freq2 and freq5 > freq3  and freq5 > freq4 and freq5 > freq6 and freq5 > freq7 and freq5 > fre
            return (4);
if (freq6>freq1 and freq6 > freq2 and freq6 > freq3  and freq6 > freq4 and freq6 > freq5 and freq6 > freq7 and freq6 > fre
                return (5);
if (freq7>freq1 and freq7 > freq2 and freq7 > freq3  and freq7 > freq4 and freq7 > freq5 and freq7 > freq6 and freq7 > fre
                    return (6);
if (freq8>freq1 and freq8 > freq2 and freq8 > freq3  and freq8 > freq4 and freq8 > freq5 and freq8 > freq6 and freq8 > fre
                        return (7);
if (freq9>freq1 and freq9 > freq2 and freq9 > freq3  and freq9 > freq4 and freq9 > freq5 and freq9 > freq6 and freq9 > fre
                            return (8);

return (100);
```

17. Bloque de entrenamiento. Se define una variable de nombre línea y longitud 88. Se define como fin de línea de datos al carácter ASCII 122. Se define un rango de datos en una línea entre el carácter 2 y el carácter 88.
    Cada vez que llegue un dato el registro IORD_ALTERA_AVALON_UART_STATUS(UART_0_BASE) se actualiza como un flag positivo. Mientras este flag no se encuentre ready a la par con ALTERA_AVALON_UART_STATUS_RRDY_MSK entonces no se recibirán datos. Una vez que se reciban los datos estos serán almacenados en el registro IORD_ALTERA_UART_RXDATA(UART_0_BASE).
    Esta estructura se puede seguir en general para leer las líneas bajo el core UART_RS232

```
int cont;
int m = 0; //numero de veces
char linea[88]="";
while(m<10){
    cont = 0;
        while(a!=122){

            if ((cont>1)&(cont<89)){
                status = IORD_ALTERA_AVALON_UART_STATUS(UART_0_BASE);
                 while (!(status & ALTERA_AVALON_UART_STATUS_RRDY_MSK))
                    status=IORD_ALTERA_AVALON_UART_STATUS(UART_0_BASE);
                a= IORD_ALTERA_AVALON_UART_RXDATA(UART_0_BASE);
                status=ALTERA_AVALON_UART_STATUS_TRDY_MSK;
                linea[cont-2]=((int)a);

            }else{
                status = IORD_ALTERA_AVALON_UART_STATUS(UART_0_BASE);
                 while (!(status & ALTERA_AVALON_UART_STATUS_RRDY_MSK))
                status=IORD_ALTERA_AVALON_UART_STATUS(UART_0_BASE);
                a= IORD_ALTERA_AVALON_UART_RXDATA(UART_0_BASE);
                status=ALTERA_AVALON_UART_STATUS_TRDY_MSK;
            }
            cont++;
        }
        a=121;
        m++;
        if (m<2){

        }else{
            dataset.append(linea);
            dataset.append("0\n");
            numeroPuntos++;
        }
    }
```