

# MATERIA DE SISTEMAS EMBEBIDOS

## PROYECTO FINAL

Paulo C. Garrido-Lechón  
Cristian S. Hernández-Ramírez  
Luis A. Lima-Pambaquishpe

14 de diciembre de 2018

## 1. Introducción

En el siguiente laboratorio se va a realizar un sistema con una determinada estructura que pueda tomar sus propias decisiones sobre el conjunto de datos a trabajar, implementando una base de datos, librerías creadas y los algoritmos aprendidos en clase como los son

CNN y K-NN, probando el rendimiento de cada matriz encontrada con k-nn y la matriz de datos de prueba, analizando la que tiene mejor rendimiento y finalmente descartando el resto de las matrices de datos.

## 2. Diseño del Sistema

### 2.1. Diagrama de Flujo

Diagrama de flujo realizado en DIA.

Figura 1: Diagrama de flujo

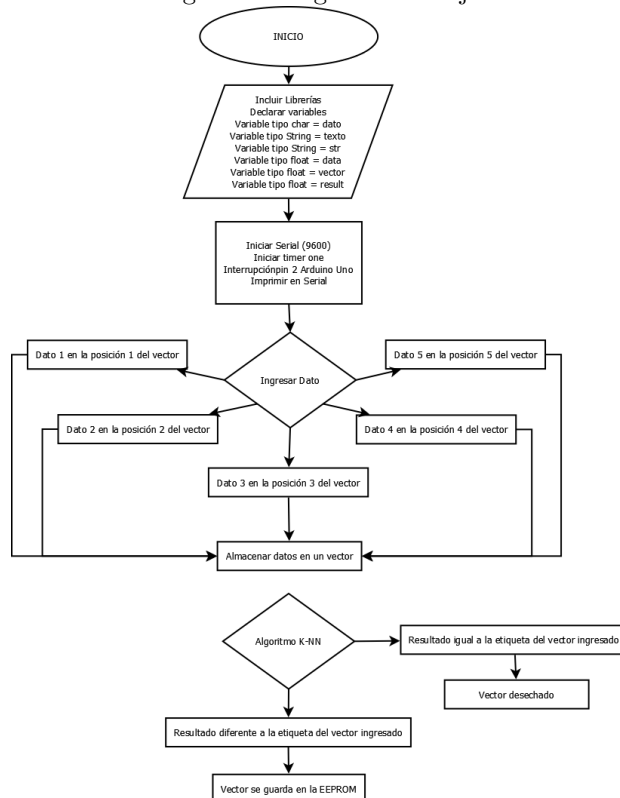
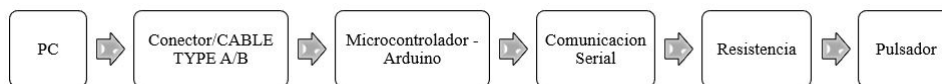


Diagrama de bloques para el funcionamiento del programa.

Figura 2: Diagrama de bloques

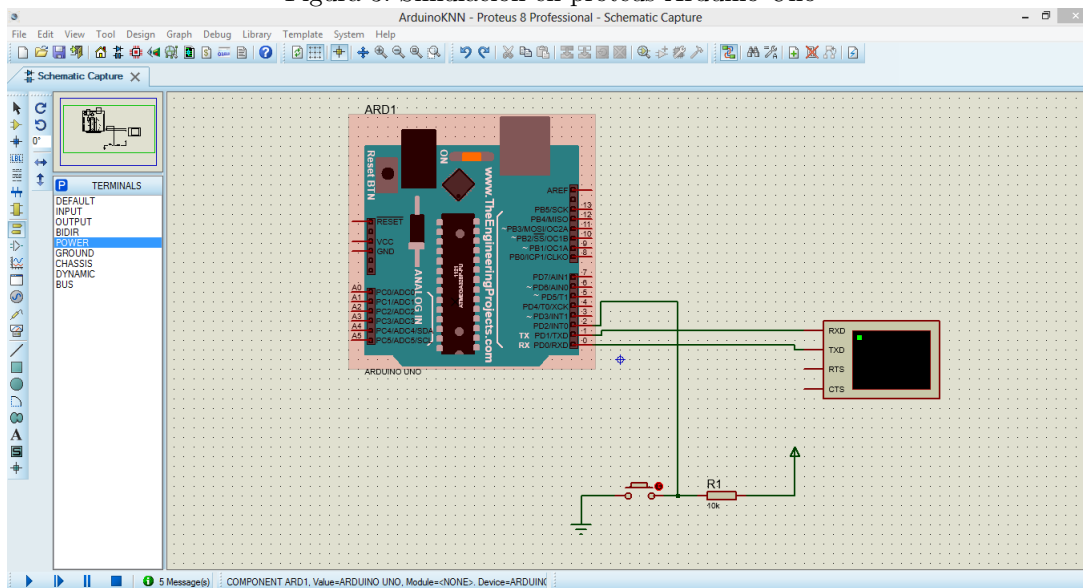


### 3. Desarrollo

#### 3.1. Simulación

Simulación en Proteus

Figura 3: Simulación en proteus Arduino Uno



Código KNN realizado en el software Arduino IDE.

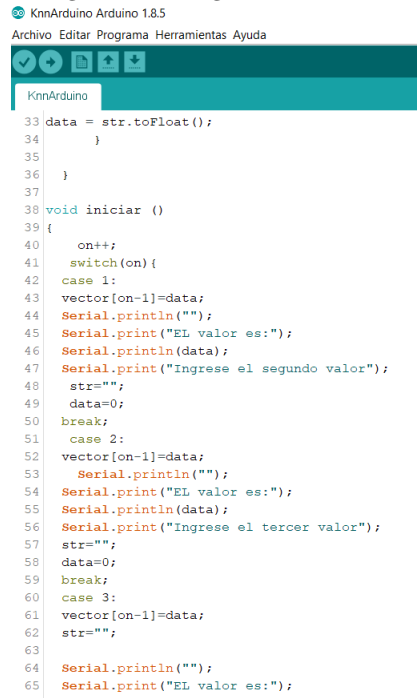
Figura 4: Código en Arduino 1

```
KnnArduino 1.8.5
Archivo Editar Programa Herramientas Ayuda

KnnArduino

1 #include <EEPROM.h>
2 #include <TimerOne.h>
3 #include <knn.h>
4
5 String texto= ""; // palabra original
6 char dato;
7 String str;
8 float data;
9 int i=0;
10 int j=0;
11 int k=0;
12 int l=0;
13 float vector[5];
14 int on=0;
15 float result;
16 int a=0, b=0;
17 int total=0;
18 void setup() {
19   // put your setup code here, to run once:
20   Serial.begin(9600);
21   Timer1.initialize(1000000);
22   Timer1.attachInterrupt(escritura);
23   attachInterrupt(0, iniciar, LOW); // interrupcion 2
24   Serial.println("Ingrese el primer dato");
25 }
26
27 void loop() {
28   if(Serial.available()>0) // obtiene la cantidad de bytes mayor a 0
29   {
30     //dato=Serial.read(); // asignar dato entrante en la variable.
31     //texto +=dato; // suma cada dato en la variable texto
32     str = Serial.readStringUntil('\n');
33     data = str.toFloat();
```

Figura 5: Código en Arduino 2

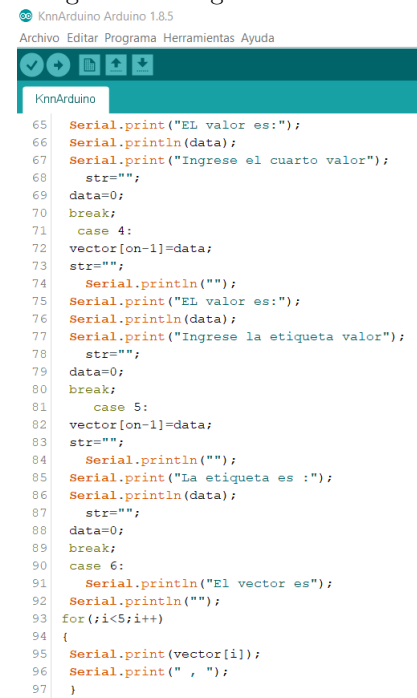


```

33 data = str.toFloat();
34 }
35 }
36 }
37
38 void iniciar ()
39 {
40     on++;
41     switch(on){
42     case 1:
43         vector[on-1]=data;
44         Serial.println("");
45         Serial.print("EL valor es:");
46         Serial.println(data);
47         Serial.print("Ingrese el segundo valor");
48         str="";
49         data=0;
50         break;
51     case 2:
52         vector[on-1]=data;
53         Serial.println("");
54         Serial.print("EL valor es:");
55         Serial.println(data);
56         Serial.print("Ingrese el tercer valor");
57         str="";
58         data=0;
59         break;
60     case 3:
61         vector[on-1]=data;
62         str="";
63
64         Serial.println("");
65         Serial.print("EL valor es:");

```

Figura 6: Código en Arduino 3



```

65     Serial.print("EL valor es:");
66     Serial.println(data);
67     Serial.print("Ingrese el cuarto valor");
68     str="";
69     data=0;
70     break;
71     case 4:
72         vector[on-1]=data;
73         str="";
74         Serial.println("");
75         Serial.print("EL valor es:");
76         Serial.println(data);
77         Serial.print("Ingrese la etiqueta valor");
78         str="";
79         data=0;
80         break;
81     case 5:
82         vector[on-1]=data;
83         str="";
84         Serial.println("");
85         Serial.print("La etiqueta es :");
86         Serial.println(data);
87         str="";
88         data=0;
89         break;
90     case 6:
91         Serial.println("El vector es");
92         Serial.println("");
93         for(,i<5;i++)
94         {
95             Serial.print(vector[i]);
96             Serial.print(" , ");
97         }

```

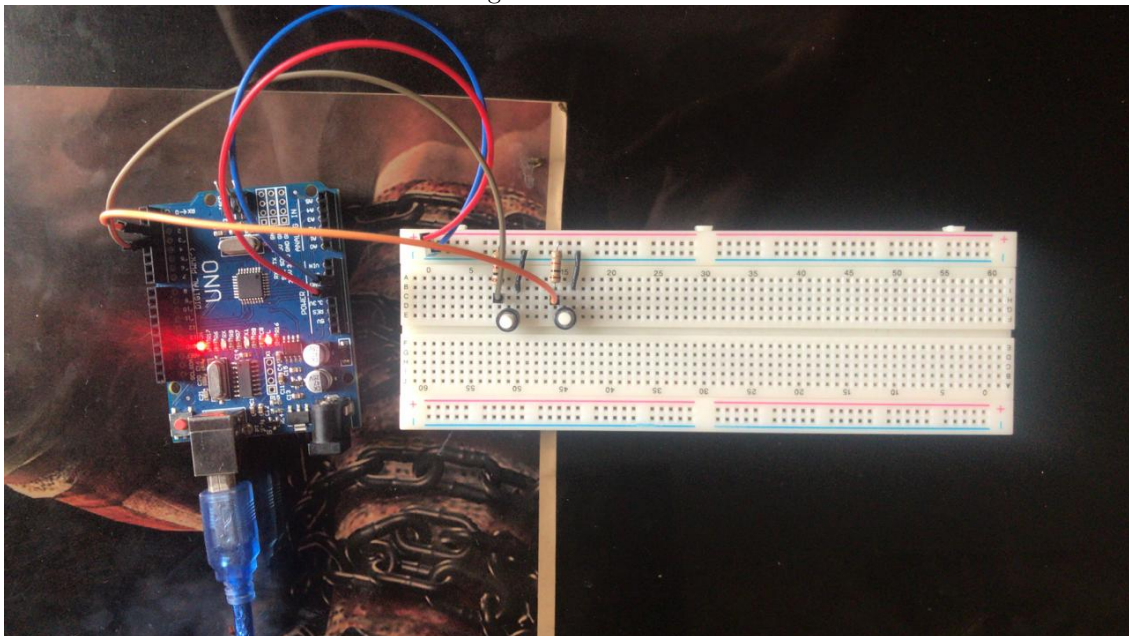
Figura 7: Código en Arduino 4

```
KnnArduino Arduino 1.8.5
Archivo Editar Programa Herramientas Ayuda
KnnArduino $
--
98     break;
99     case 7:
100         result=knn(8,5,3,3,vector);
101         if(result==vector[4])
102         {
103             Serial.println("Correcto");
104         }
105         else
106         {
107             Serial.println("Incorrecto");
108             l=1;
109         }
110     }
111 }
112 void escritura()
113 {
114     if(l==1)
115     {
116         k++;
117         if(k<=5)
118         {
119             EEPROM.write(40+k,vector[k-1]);
120         }
121         else
122         {
123             k=0;
124             l=0;
125         }
126     }
127     if(k==5)
128     {
129         Serial.println ("listo");
130     }
131 }
```

## 4. Análisis de Resultados

Armado del circuito para el programa realizado.

Figura 8: Circuito



Simulación realizada con el Arduino Mega.

Figura 9: Simulación en Proteus en la placa Arduino Mega

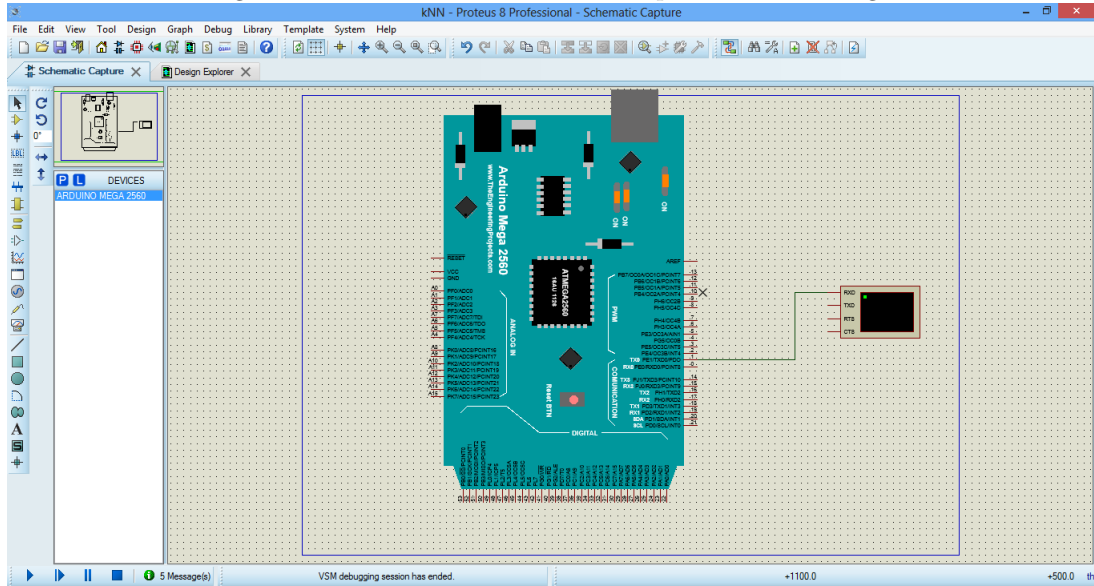


Figura 10: Verificación del código

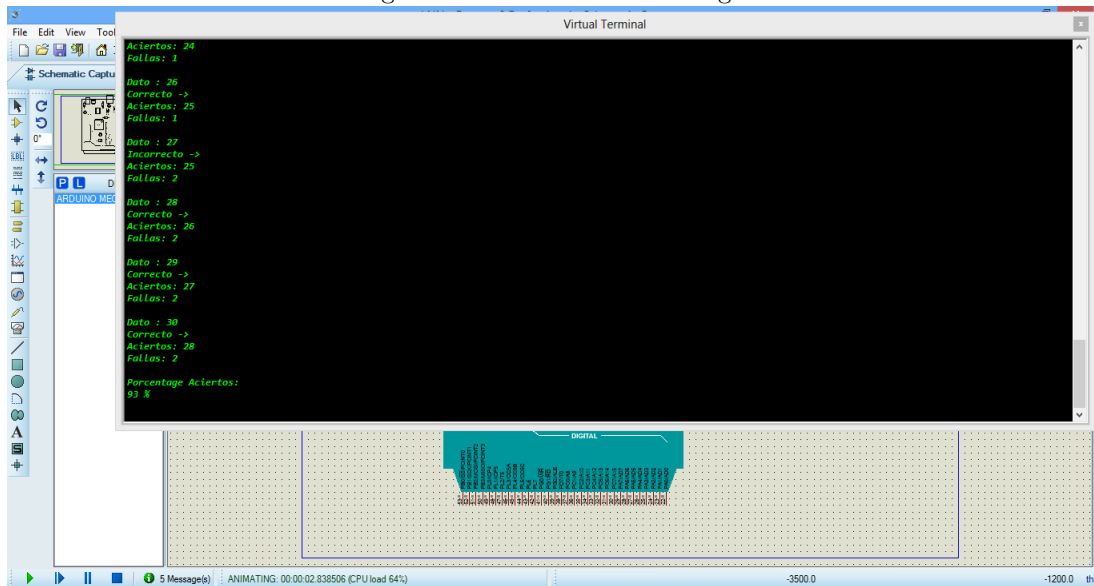
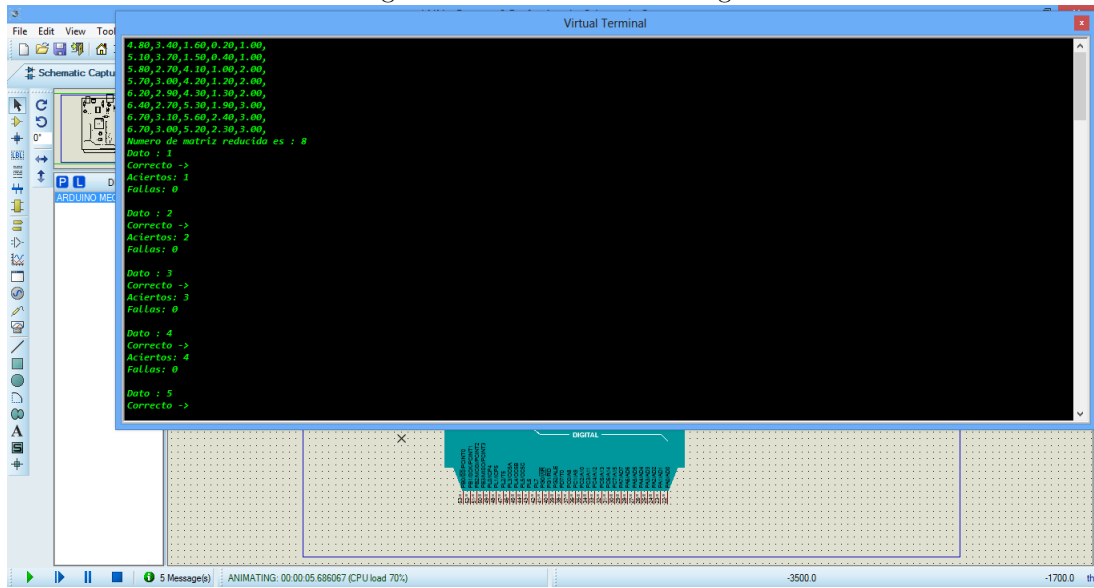


Figura 11: Verificación del código



Simulación en Proteus en la placa Arduino Uno.

Figura 12: Simulación en proteus Arduino Uno

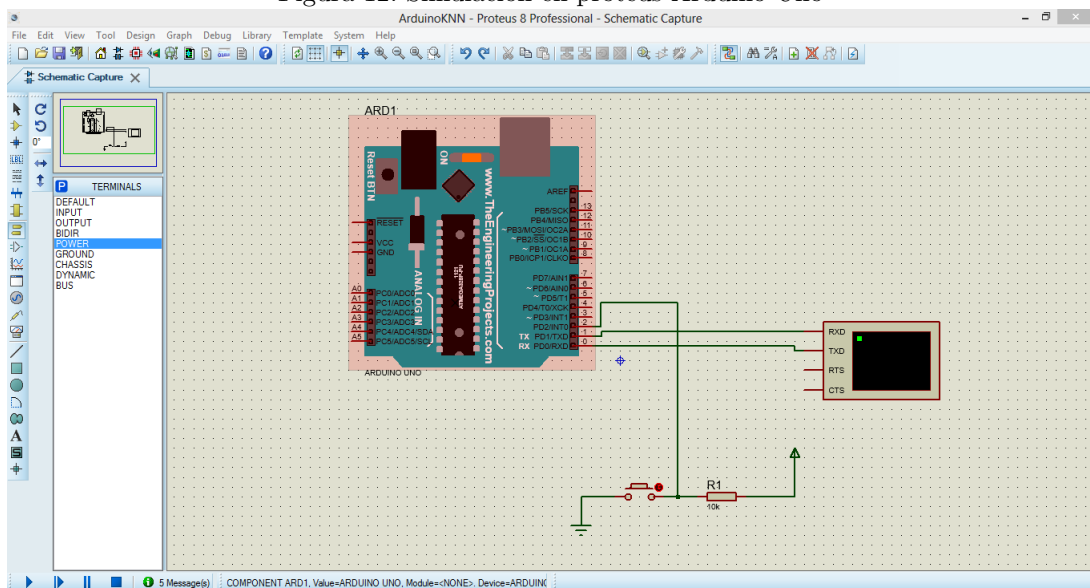


Figura 13: Verificación del código

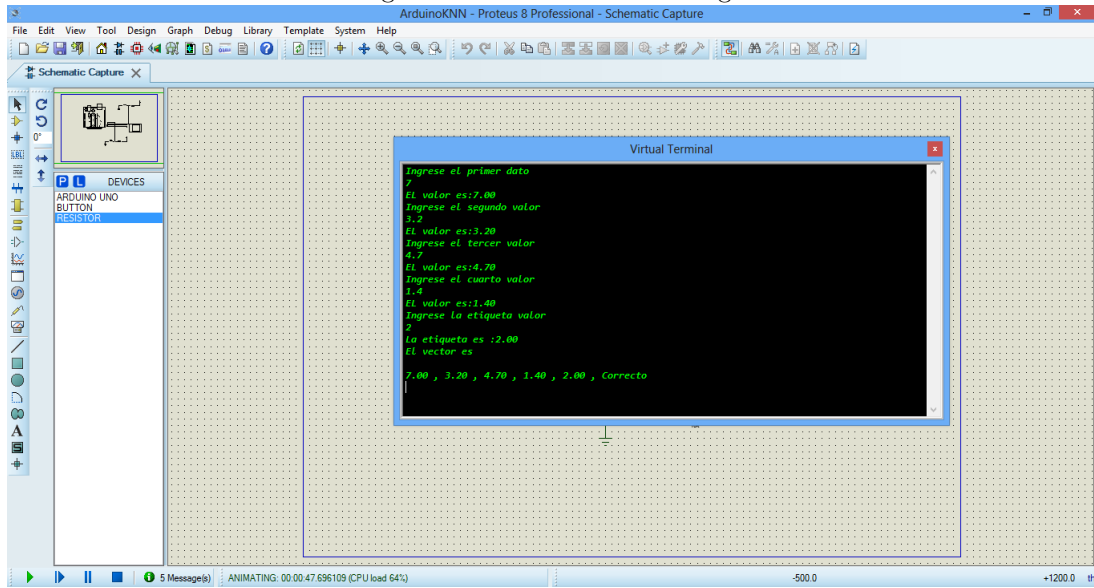


Figura 14: Nuevo dato

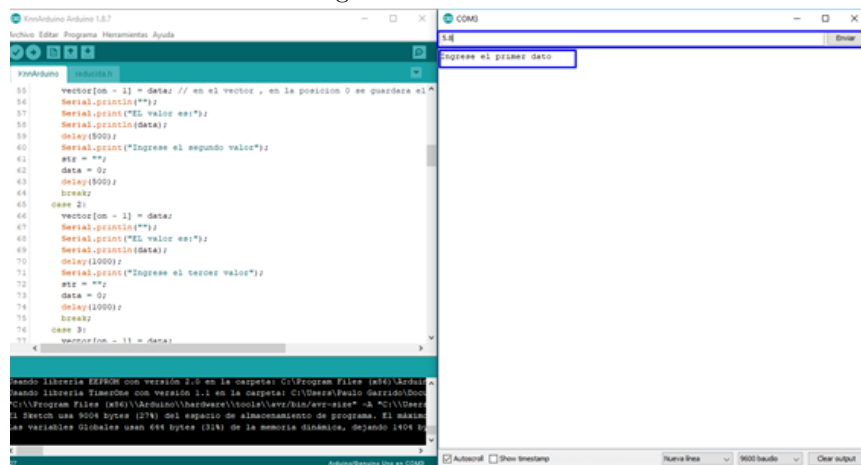


Figura 15: Nuevo dato

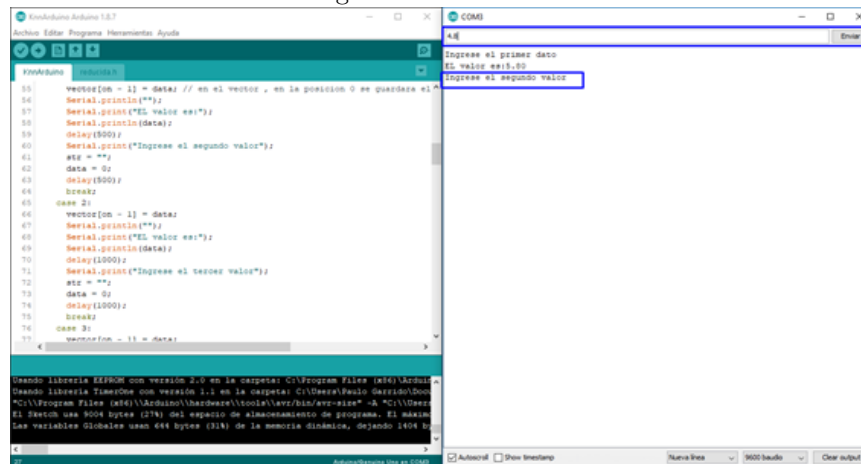


Figura 16: Nuevo dato

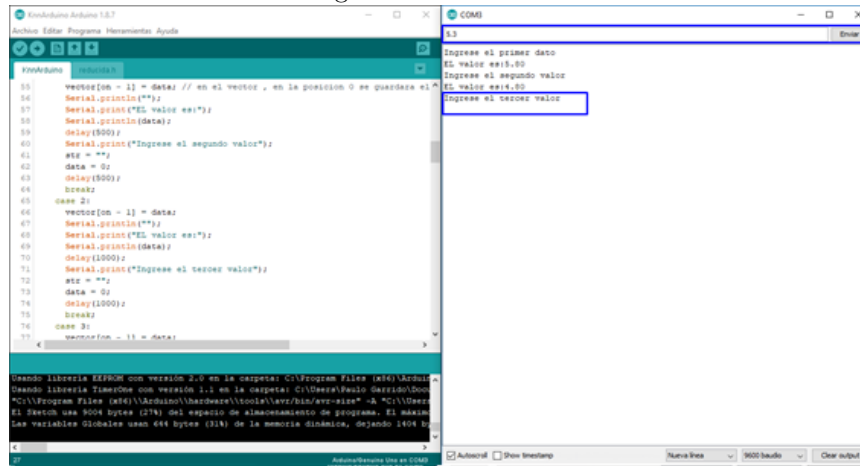


Figura 17: Nuevo dato

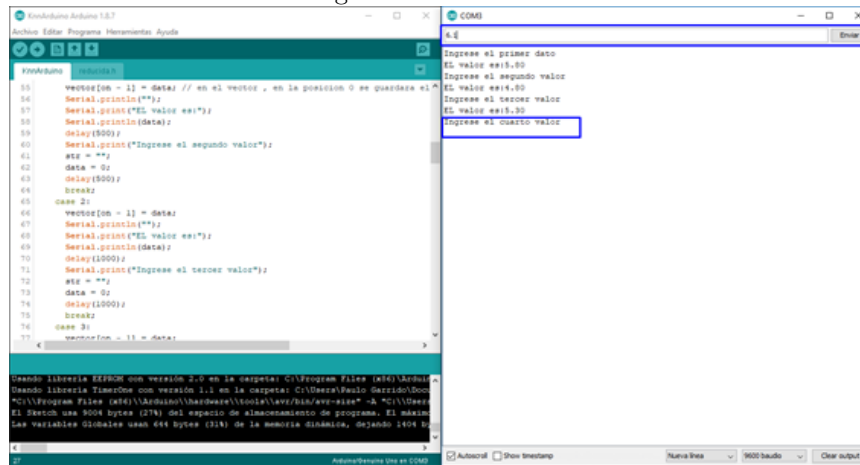


Figura 18: Nuevo dato

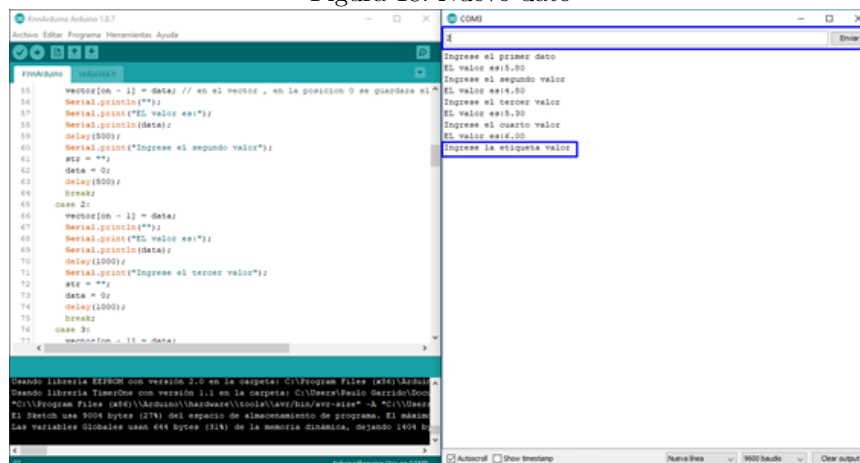




Figura 19: Nuevo dato

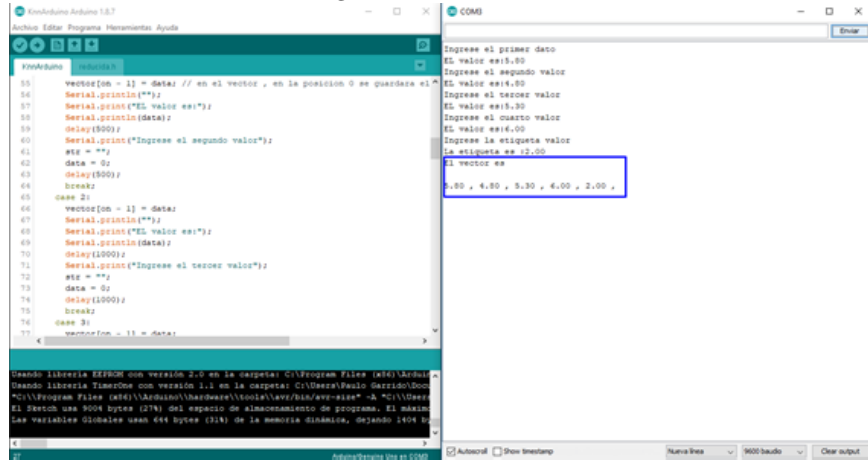


Figura 20: Nuevo dato

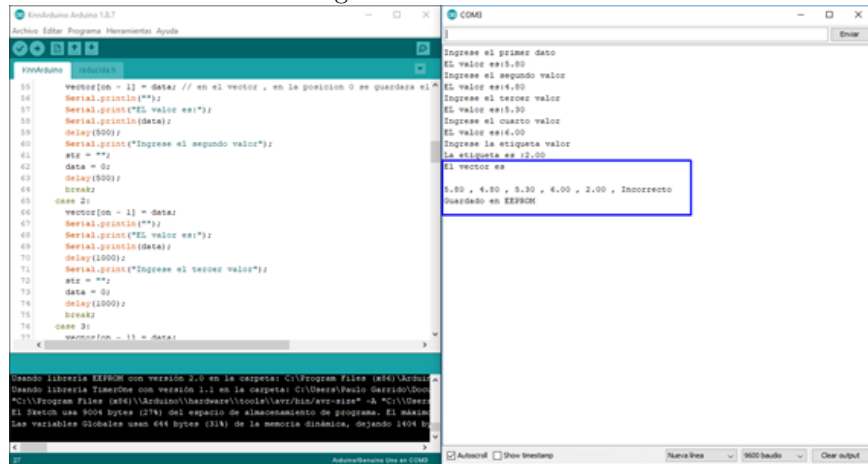


Figura 21: Dato existente

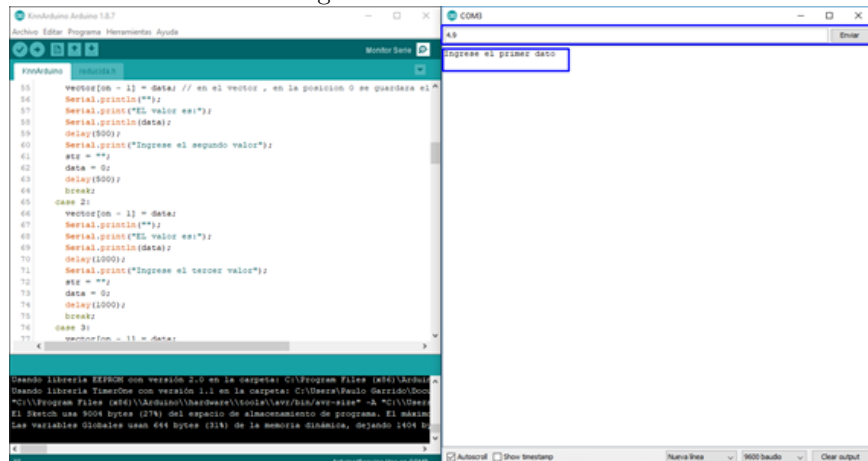


Figura 22: Dato existente

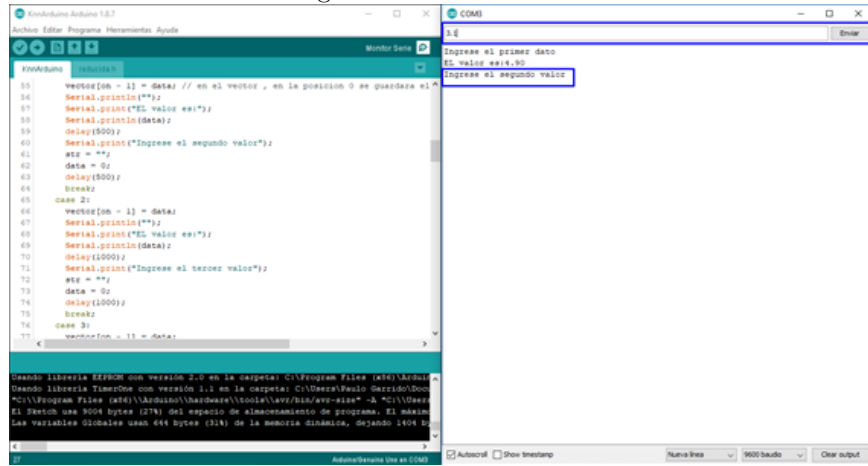


Figura 23: Dato existente

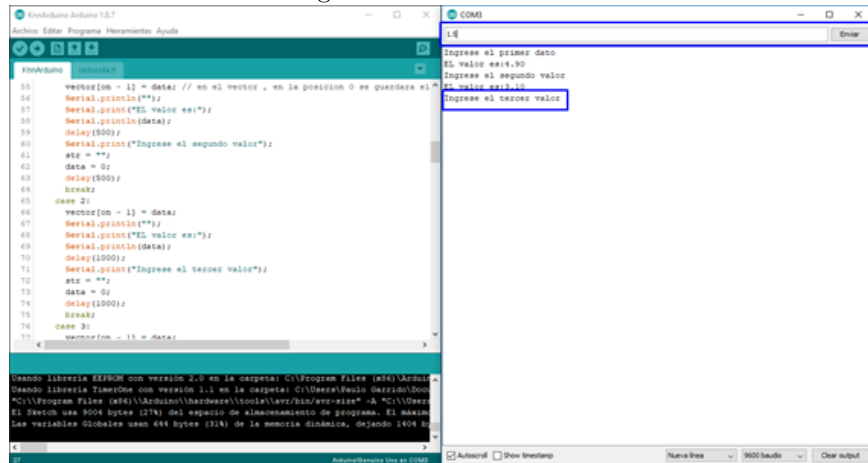


Figura 24: Dato existente

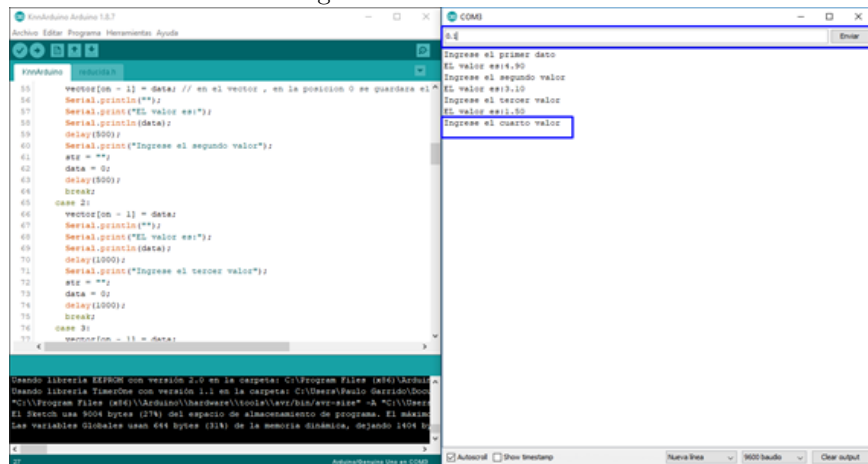


Figura 25: Dato existente

```

// Arduino IDE
// Archivo: Editor, Programa, Herramientas, Ayuda

// Programa: vector.h

// Definición de constantes
const int TAM = 10;

// Definición de tipos de datos
typedef float Vector[TAM];

// Definición de funciones
void inicializar(Vector v);
void imprimir(Vector v);
void agregar(Vector v, float d);
void eliminar(Vector v, float d);
void buscar(Vector v, float d);
void actualizar(Vector v, float d);

// Función principal
int main() {
    Vector v;
    inicializar(v);
    imprimir(v);
    agregar(v, 1.0);
    imprimir(v);
    eliminar(v, 1.0);
    imprimir(v);
    buscar(v, 1.0);
    imprimir(v);
    actualizar(v, 1.0);
    imprimir(v);
    return 0;
}

// Definición de funciones
void inicializar(Vector v) {
    for (int i = 0; i < TAM; i++) {
        v[i] = 0.0;
    }
}

void imprimir(Vector v) {
    for (int i = 0; i < TAM; i++) {
        Serial.print(v[i]);
        Serial.print(" ");
    }
    Serial.println();
}

void agregar(Vector v, float d) {
    v[TAM - 1] = d;
    imprimir(v);
}

void eliminar(Vector v, float d) {
    for (int i = 0; i < TAM; i++) {
        if (v[i] == d) {
            v[i] = 0.0;
        }
    }
    imprimir(v);
}

void buscar(Vector v, float d) {
    for (int i = 0; i < TAM; i++) {
        if (v[i] == d) {
            Serial.print("El valor ");
            Serial.print(d);
            Serial.print(" se encuentra en la posición ");
            Serial.print(i);
            Serial.println();
        }
    }
}

void actualizar(Vector v, float d) {
    for (int i = 0; i < TAM; i++) {
        v[i] = d;
    }
    imprimir(v);
}
    
```

Figura 26: Dato existente

```

// Arduino IDE
// Archivo: Editor, Programa, Herramientas, Ayuda

// Programa: vector.h

// Definición de constantes
const int TAM = 10;

// Definición de tipos de datos
typedef float Vector[TAM];

// Definición de funciones
void inicializar(Vector v);
void imprimir(Vector v);
void agregar(Vector v, float d);
void eliminar(Vector v, float d);
void buscar(Vector v, float d);
void actualizar(Vector v, float d);

// Función principal
int main() {
    Vector v;
    inicializar(v);
    imprimir(v);
    agregar(v, 1.0);
    imprimir(v);
    eliminar(v, 1.0);
    imprimir(v);
    buscar(v, 1.0);
    imprimir(v);
    actualizar(v, 1.0);
    imprimir(v);
    return 0;
}

// Definición de funciones
void inicializar(Vector v) {
    for (int i = 0; i < TAM; i++) {
        v[i] = 0.0;
    }
}

void imprimir(Vector v) {
    for (int i = 0; i < TAM; i++) {
        Serial.print(v[i]);
        Serial.print(" ");
    }
    Serial.println();
}

void agregar(Vector v, float d) {
    v[TAM - 1] = d;
    imprimir(v);
}

void eliminar(Vector v, float d) {
    for (int i = 0; i < TAM; i++) {
        if (v[i] == d) {
            v[i] = 0.0;
        }
    }
    imprimir(v);
}

void buscar(Vector v, float d) {
    for (int i = 0; i < TAM; i++) {
        if (v[i] == d) {
            Serial.print("El valor ");
            Serial.print(d);
            Serial.print(" se encuentra en la posición ");
            Serial.print(i);
            Serial.println();
        }
    }
}

void actualizar(Vector v, float d) {
    for (int i = 0; i < TAM; i++) {
        v[i] = d;
    }
    imprimir(v);
}
    
```

## 5. Conclusiones y Recomendaciones

- Las librerías .h realizadas facilitó el desarrollo de la programación logrando comprender de mejor forma el código ya que se vuelve comprensible, eficiente y no se satura de líneas de código en un solo archivo.
- El mejor algoritmo de selección de datos en este caso es K-NN por la razón de que es más efectivo en el rendimiento al escoger datos que el algoritmo CNN.
- El algoritmo de selección de datos es muy dinámico para la obtención y filtrado de datos ya que se encuentra en constante aprendizaje, por lo que si se ingresa un dato nuevo por comunicación serial, el algoritmo establece si el dato le sirve o no para el mejoramiento de datos.
- \* Verificar si los componentes a utilizar están en buen estado sin fallas para realizar el armado.
- \* Conectar ordenadamente los pines para no perderse al momento de armar y/o probar.
- \* Realizar simulaciones previas al armado del circuito para verificar el correcto funcionamiento del programa.
- \* Tener en cuenta las especificaciones de cada elemento para realizar el armado del circuito y no cometer errores.
- \* Se considera esencial comentar cada línea de programación realizado debido a que el código maneja varias variables lo que podría causar equivocaciones y si fuera el caso necesario de efectuar una mejora en la programación sería más fácil hacerlo.