ARDUINO : PLOTTER

Este código se ejecuta en la placa que controla el plotter su misión es el manejo del plotter para que recibiendo unos comandos mueva los motores de la forma que estos le indican.

Utilizamos la librería Servo.h, estándar de Arduino y que facilita el movimiento de servomotores, para los paso a paso se decidió no utilizar librería porque producía un movimiento errático y poco fiable.

Se declaran las numerosas variables que se usan, de las que cabría destacar :

- Motor y motor2, que son los pines que utiliza el motor
- StepsLookup, que marca los estados de los bobinados de los motores para su control

```
8   //definicion de pins de motores paso a paso
9
10  int motor[] = {4, 5, 6, 7};
11  int motor2[] = {8, 9, 10, 11};
12
13  //definicion variables
14  int stepCounter = 0;     // contador para los pasos
15  int stepsPerRev = 2048;  // pasos para una vuelta completa
16  //secuencia 2-fases
17  const int numSteps = 4;
18  const int stepsLookup[4] = { B1100, B0110, B0011, B1001 };
19  const int off[1] = { B0000 };
20
21  const int StopButtonPin = 12;
22  int buttonsPin = 1;// button state, when pressed = 0
23  bool isHomed = false;
24  int steps = 0;
25  // definicion servo Actuador
26  Servo lapiz;
```

Llegados al *void setup* en este se comienza el puerto serial, que recibirá comandos, y muy importante se ejerce el home, que mueve el cabezal hasta que activa un botón fijo en un lateral, esto permite a la máquina situarse en un origen desde donde contar los pasos del motor y comenzar siempre desde la misma posición.

```
27  void setup()
28  {
29    Serial.begin(9600);
30    lapiz.attach(3);
31    //declarar pines como salida
32    pinMode(motor[0], OUTPUT);
33    pinMode(motor[1], OUTPUT);
34    pinMode(motor[2], OUTPUT);
35    pinMode(motor[3], OUTPUT);
36    pinMode(motor2[0], OUTPUT);
37    pinMode(motor2[1], OUTPUT);
38    pinMode(motor2[2], OUTPUT);
39    pinMode(motor2[3], OUTPUT);
40
41    // ir a origen
42    void homing();
43  }
102 void homing()
103 {
104   lapiz.write(45);
105   for (int i = 0; i < 500; i++)
106   {
107     anticlockwise(motor2);
108     delay(3);
109   }
110   apagar();
111   while (digitalRead(StopButtonPin) == HIGH) // que se hacerque al home
112   {
113     anticlockwise(motor);
114     delay(3);
115     isHomed = true;
116     lapiz.write(0);
117   }
118   while (digitalRead(StopButtonPin) == LOW) //  que se aleje lo justo para dejar de pulsar el boton
119   {
120     clockwise(motor);
121     delay(3);
122     lapiz.write(45);
123   }
124   steps = 0; // inicializamos los steps a 0
125   apagar();
126 }
127
```

Por último se apaga el motor para evitar el sobrecalentamiento dado que tampoco necesitar ejercer una fuerza de bloqueo.

```
140   void apagar()
141   {
142     //apagar para evitar sobre calentar porque no hace falta fuerza para mantener pos
143     digitalWrite(motor[0], bitRead(off[1], 0));
144     digitalWrite(motor[1], bitRead(off[1], 0));
145     digitalWrite(motor[2], bitRead(off[1], 0));
146     digitalWrite(motor[3], bitRead(off[1], 0));
147   }
```

Llegados al *void loop,* función que se repite, encontramos las funcionalidades relacionadas con la lectura de comandos y movimiento de los motores acorde a estos.

```
45  void loop()
46  {
47    if (Serial.available() && isHomed) {
48      // leer serial
49      int *st1, *st2, *servo;
50      readSerial( st1,  st2,  servo);
51
52      //Servo
53      moveServo(*servo);
54      //segundo motor
55      if (*st2 != 0)
56      {
57        moveMotor(motor2, *st2);
58        apagar();
59      }
60      //primer motor
61      if (*st1 != 0)
62      {
63        moveMotor(motor, *st1);
64        apagar();
65      }
66    }
67  }
```

Tras comprobar si ha hecho el proceso de home y ver si hay datos en el serial, se lee el mismo. Se recibirán comandos de la forma A,B,C, tres números separados entre comas que representas los movimientos de los 2 ejes y el servo, respectivamente.

Se separan cada valor en una variable para ejecutar el movimiento de cada eje correspondientemente.(función read serial)

```
128  void readSerial(int *st1, int *st2, int *servo)
129  {
130    String first  = Serial.readStringUntil(',');
131    Serial.read(); //next character is comma, so skip it using this
132    String second = Serial.readStringUntil(',');
133    Serial.read();
134    String third  = Serial.readStringUntil(',');
135    * st1 = first.toInt();
136    * st2 = second.toInt();
137    * servo = third.toInt();
138
139  }
```

- A la hora del movimiento ejecutaremos una función que avanza un paso el determinado número de veces, para el servo será entre 2 posiciones, de pintar y no pintar. Dependiendo de si el valor es positivo se moverá en un sentido o el otro.

```
//funcion que controla el bucle de movimiento de los motores de los ejes
void moveMotor(int motor[], int st)
{
  if (st > 0)
  {
    for (int i = 0; i < st; i++)
    {
      steps++;
      clockwise(motor);
      delay(3);
    }
  } else
  {
    for (int i = 0; i > st; i--)
    {
      steps--;
      anticlockwise(motor);
      delay(3);
    }
  }
}
```

- Esto llama a unas funciones que controlan el encendido y la secuencia del mismo para poder mover el motor correctamente.

```
void clockwise(int motorvc[])//  secuencia en sentido reloj
{
  stepCounter++;
  if (stepCounter >= numSteps) stepCounter = 0;
  setOutput(stepCounter, motorvc);
}

void anticlockwise(int motorva[])//  secuencia en sentido anti-horario
{
  stepCounter--;
  if (stepCounter < 0) stepCounter = numSteps - 1;
  setOutput(stepCounter, motorva);
}

void setOutput(int step, int motorVar[])// manda la señal para encender las bobinas que corresponden al motor
{
  digitalWrite(motorVar[0], bitRead(stepsLookup[step], 0));
  digitalWrite(motorVar[1], bitRead(stepsLookup[step], 1));
  digitalWrite(motorVar[2], bitRead(stepsLookup[step], 2));
  digitalWrite(motorVar[3], bitRead(stepsLookup[step], 3));
}
```