Implementación de un sistema genérico de control para motores BLDC

Andrés Alejandro Moreno Sánchez
andres.moreno-s@mail.escuelaing.edu.co
Programa de Ingeniería Electrónica
Escuela Colombiana de Ingeniería Julio Garavito
Bogotá D.C., Colombia

***Resumen* - En** este documento se presenta un controlador versátil para motores BLDC, capaz de operar en un rango amplio de corrientes y con la opción de poderse configurar con facilidad para evaluar el desempeño del mismo. Se describe el procedimiento de diseño realizado para su realización; también se presenta una serie de sentencias que permitirán primeras pruebas. Se desarrolla aplicación para sistema operativo Android, la cual sirve como interfaz con el usuario.

Palabras Clave - Controlador, Motor BLDC, Arduino, Android.

***Abstract* - This document presents a versatile controller for BLDC motors, capable of operating in a wide range of currents and with the option of being able to configure with ease to evaluate its performance. The design procedure performed for its realization is described; a series of sentences that will allow first evidence is also presented. Application for Android operating system is developed, which serves as an interface with the user.**

Key Words - Controller, BLDC Motor, Arduino, Android.

I. Introducción

Los motores DC sin escobillas (BLDC) se han convertido en dispositivos ampliamente utilizados en varios campos debido a que al no traer escobillas tienen una duración mayor. Con estos motores tienen la capacidad de tener alto torque en un espacio reducido [1]. Actualmente, debido a la construcción propia de los motores, se necesita una secuencia de conmutación electrónica con el fin de lograr un correcto funcionamiento y buen desempeño. Para este fin se han desarrollado diferentes métodos de control para lograr un alto rendimiento en todo el rango de velocidad, generando un par completo en velocidad. Uno de los métodos más utilizados es el control de campo orientado, FOC (de sus siglas en inglés *Field Oriented Control*, el cual tiene la bondad de consumir baja energía [2].

Se busca implementar un sistema de control versátil, que adicionalmente dé lineamientos necesarios para que este controlador sea capaz de operar con otro tipo de motores, permitiendo de esta manera la exploración del funcionamiento de los motores BLDC; este trabajo abre las puertas a una serie de trabajos que permiten implementar y evaluar estrategias de control.

II. Descripción Del Problema

Los ESC (de sus siglas en ingles *Electronic Speed Controller*) que se encuentran en el mercado no tienen un buen manejo con los picos de corriente que se generan con los cambios de velocidad en el motor ocasionando que se tenga que suplir una corriente pico demasiado grande y en algunos casos se produce un fallo completo del sistema. Además, los controladores que se consiguen en el mercado ya están configurados para operar en unas condiciones especificas, que no son modificables por el usuario [3]. Este sistema al ser versátil, permite la evaluación de las ya existentes estrategias de control, o bien, la evaluación de nuevas estrategias de control.

III. Objetivos

El desarrollo permite generar la implementación de un sistema genérico de control para motores BLDC con el uso de una tarjeta de adquisición, la programación de diferentes algoritmos de control y el desarrollo de un programa de Arduino, para medir y controlar los picos de corriente en una planta de prueba para estos motores.

IV. Marco Conceptual

IV-A. Generalidades BLDC

Los motores *brushless* BLDC son motores síncronos DC que se controlan por conmutación electrónica a través de inversores o por fuentes conmutadas [4]. Estos son cada día más usados, pues como su nombre lo indica, no tienen escobillas que son las que más se desgastan en los motores tradicionales. Estos motores se remplazan con un control electrónico que da mayor confiabilidad [1]; al ser más pequeño y liviano que un motor de escobilla (operando a la misma potencia), resulta ideal para aplicaciones en donde el espacio es reducido, sin que su torque se vea reducido [2].

La construcción de un sistema de motor sin escobillas es típicamente similar a un motor síncrono de imán permanente [5]. Dado que no hay contacto mecánico o eléctrico entre el estator y el rotor del motor BLDC [7], se necesitan diferentes soluciones para conocer la posición del rotor en relación al estator para facilitar el control del motor [6][7]. Los motores BLDC utilizan dos diferentes métodos para lograrlo, ya sea mediante sensores Hall o midiendo la fuerza contraelectromotriz [2].

IV-B. Tipos de control en motores bldc

Existen varios tipos de sistema de control de motores sin escobillas, entre los que se encuentran el control, trapezoidal, sinusoidal y de campo (FOC) alterando y caracterizando el costo y funcionamiento del motor.

V. Metodología

Para el desarrollo de un sistema de control genérico se inicia con el conocimiento del comportamiento del sistema con el uso de una tarjeta de adquisición de Texas Instruments (MyDaQ) de datos para pruebas. Se procedió a observar pruebas con un ESC y el software Labview. Asi la programación de una tarjeta de pruebas para probar diferentes algoritmos de control del dispositivo que brinda comunicación con la aplicación. Se procede al desarrollo de un programa en Arduino para el control y su comunicación con una aplicación para Android y medir los picos de corriente.

VI. Implementación

La primera prueba que se realizó fue con la tarjeta de adquisición de Texas Instruments (MyDaQ) en Labview en la que se probó un controlador en lazo abierto, con este se controló un motor de la planta "Helicóptero"del programa de ingeniería electrónica de la Escuela Colombiana de Ingeniería. Los módulos de Labview quedan en los anexos.

Figura 1: Estación de pruebas

La siguiente fase es el desarrollo de una planta de pruebas (fig. 1) con más capacidad en corriente y control sobre mas variables, para esto se utilizó como sistema de control un Arduino minipro el cual es una tarjeta de desarrollo que cuenta con una gran comunidad de desarrolladores, por lo que es útil para futuras investigaciones. El driver que va a controlar la corriente que requiere el motor va a ser el L6234, el cual controla controla con el diagrama de tiempos mostrado en la siguiente figura.

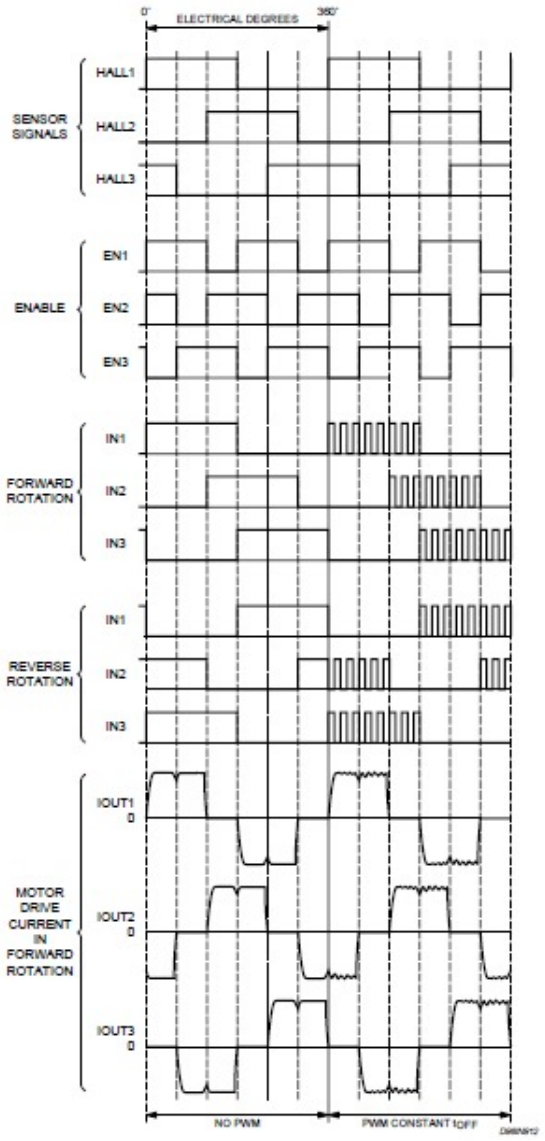

Figura 2: Señales del L6234 [8].

Para incrementar la posibilidad de controlar una mayor variedad de motores, se realiza un control a través de la medición de la fuerza contraelectromotriz, Las señales de control necesarias para este fin se obtienen a través de comparadores de cruce por cero, los cuales no comparan con la tierra del circuito, sino que comparan con una tierra virtual la cual sera la conexión del neutro de la estrella del motor. Dado que no se tiene acceso directo a este punto se conecta resistencias de 10 kΩ en las tres fases del motor conectadas en estrella, con esto se obtiene un neutro virtual que sirve de referencia para el comparador. Esto se hace con el fin de medir el voltaje de fase en la bobina del motor.

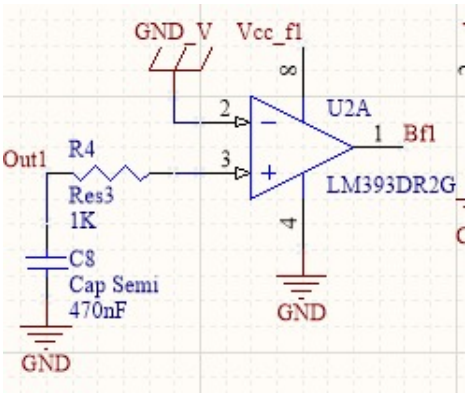

Figura 3: Comparador de cruce por cero

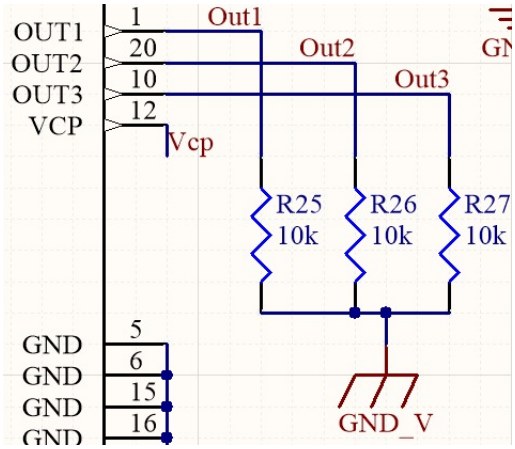

Figura 4: Señal de neutro virtual

Dado que el microcontrolador trabaja a 5 V incluiremos dos fuentes, una de 5 V y una de 9 V, esta última para poder conectar un ventilador como refrigeración del sistema. como el problema de ruidos en este sistema es muy apreciable, estas fuentes tienen filtros EMI para mejorar sus voltajes.

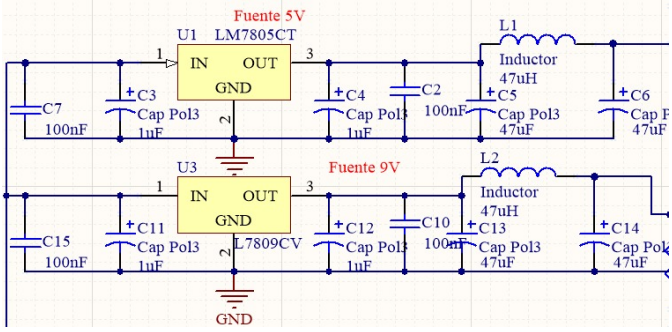

Figura 5: Fuentes de voltaje

La conexión del driver es la siguiente

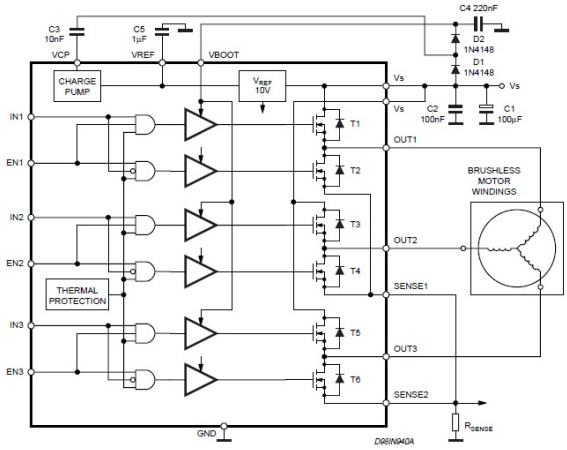

Figura 6: Driver L6234 [8].

Por ultimo se da una revisión a la conexión general de la planta de pruebas en los anexos.

Con el circuito esquemático, se diseña el circuito impreso en una tarjeta de 10 cm x 10 cm, en estas se tuvieron las siguientes recomendaciones para el correcto funcionamiento.

1 Las resistencia de medición de la corriente debe estar lo mas cerca al microcontrolador con lineas dedicadas de tierra, buscando reducir los efectos de las inductancias parásitas que aparecen al operar en altas frecuencias [9].
2 Estarán fuera del plano de tierra lo que son el microcontrolador, el driver, y los comparadores.
3 Las pistas del motor serán mínimo de 2.5mm, para poder operar el motor en un rango de corrientes de hasta 6 amperios.

El resultado queda en los anexos: Una recopilación de los resultados de diseño se encuentra disponible en los anexos, y es posible acceder tanto al diagrama esquemático, como al circuito impreso y sus respectivos archivos de fabricación a través del siguiente link https://github.com/andres14077/planta_de_pruebas/ Con la tarjeta desarrollada, se prosigue a realizar el código para el control del motor en Arduino se inicia con la declaración de variables y pines

Después se inicializan los pines como entradas y salidas respectivamente, además se inicia la comunicación serial a 115200 baudios, esto lo hacemos en la sección de *void setup* que es la sección que se ejecuta una vez cuando el microcontrolador se enciende o se reinicia.

Para poder comunicarse con la aplicación se utilizan 2 funciones, una detecta qué tipo de información le envía la aplicación mediante cabeceras, y la otra envía los datos entre 2 cabeceras, esto para que la aplicación pueda determinar en dónde comienza y dónde finalizan los números.

En la sección del *void loop*, se revisa constantemente si existen información a recibir por el puerto serial, después revisa en que estado se encuentra la planta, la cual esta puede estar operando con el motor a una velocidad, o desactivada, motor apagado. Dependiendo del estado se leen los pines digitales y se colocan las salidas según la figura 2, el código final queda en los anexos.

Para finalizar la aplicación para Android se desarrolló en la plataforma de app Inventor, esta se encuentra en la Play Store con el nombre de Driver_BLDC en esta se diseñaran 5 pantallas para el funcionamiento. estas fueron.

1 Inicio
2 Ayuda
3 Esperando conexión de bluetooth
4 Bluetooth apagado
5 Principal

En la pantalla de inicio se muestra el nombre de la aplicación, el numero de la versión y el creador, esta se muestra cuando la aplicación se abre y dura en pantalla 3 segundos.

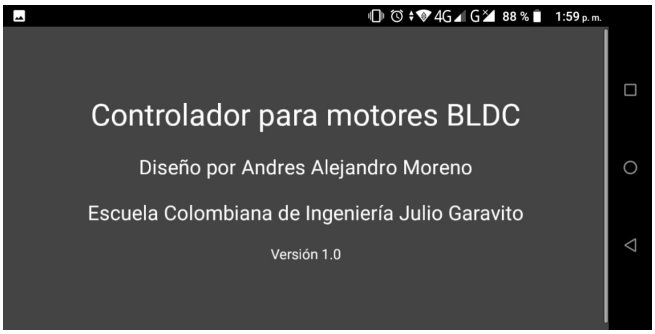

Figura 7: Pantalla de inicio

La siguiente pantalla se da cuando el usuario no a encendido el bluetooth del dispositivo.

Figura 8: Bluetooth apagado

La siguiente pantalla se muestra cuando el dispositivo se encuentra con el bluetoth encendido pero desconectado del dispositivo.

Figura 9: Esperando conexión

Una vez el dispositivo este conectado correctamente pasaremos a la pantalla principal, en esta podremos observar la velocidad actual del motor, la corriente que esta consumiendo, el sentido de giro y si esta apagado el motor. Podremos modificar la velocidad de operación, el sentido de giro y prender y apagar el motor desde la aplicación. Por ultimo contamos con el botón de ayuda, este le dará al usuario la información que necesita para ponerse en contacto con el creador en caso de ser necesario.

Figura 10: Pantalla principal

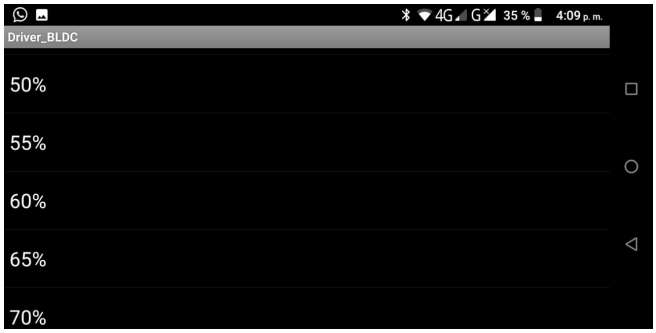

Figura 11: Selección de velocidad

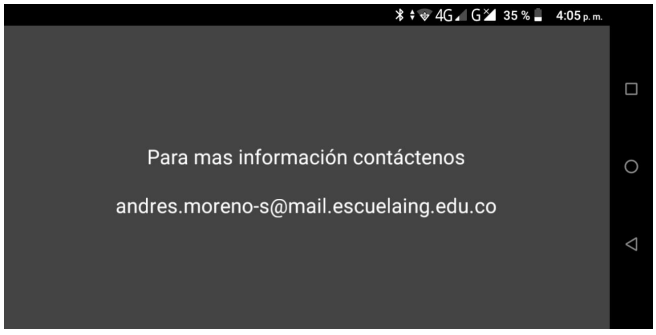

Figura 12: Pantalla de ayuda

VII. Conclusiones

Buscando ofrecer una alternativa versátil para controlar motores, se describe en este documento la etapa de diseño de un sistema de control para la evaluación de estrategias de control para motores BLDC.

Al medir de forma permanente la corriente que se le suministra a los motores, este sistema garantiza que durante

la etapa de evaluación de la estrategia de control del motor BLDC, el motor opere de forma apropiada, siempre y cuando dicha corriente no supere los valores absolutos de operación (máximo 6 amperios).

El desarrollo que se logró permite la modificación de los parámetros de operación de forma rápida y sencilla a través de la aplicación para sistemas operativos Android.

Es importante notar que el sistema permite hacer seguimiento en la aplicación y en la tarjeta para observar su adecuado funcionamiento.

La tarjeta de Arduino facilita los sistemas de control y comunicación que puede llevar a lograr mejores desarrollos en la solución de problemáticas aplicadas para el manejo de motores sin escobillas.

REFERENCIAS

[1] L. Prokop, L. Chalupa, y S. Design, «3-Phase BLDC Motor Control with Sensorless Back EMF Zero Crossing Detection Using 56F80x», Freescale Semiconductor, Application Note AN1914, 2005.

[2] M. Gomez y M. Pallones, «Sensored Single-Phase BLDC Motor Driver Using PIC16F1613», p. 22, 2014.

[3] SHIN, Young Tae; TEH, Ying-Khai. Design analysis and considerations of power efficient electronic speed controller for small-scale quadcopter unmanned aerial vehicle. En 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2018. p. 773-776.

[4] P. J. Zhao y Y. Yu, «Brushless DC Motor Fundamentals Application Note», Application Note AN047, 2014.

[5] M. Brejl, M. Princ, F. Semiconductor, y R. Csc, «3-Phase BLDC Motor with Hall Sensors and Speed Closed Loop, driven by eTPU on MCF523x», p. 44.

[6] W. Brown, «Brushless DC Motor Control Made Easy», p. 48, 2002.

[7] H. R. G. Andrés y M. S. Mauricio, «1PROYECTO HELICÓPTERO ESTÁTICO.», p. 10.

[8] V. Marano, «L6235 three phase brushless DC motor driver», STMicroelectronics, Application Note AN1625, 2003.

[9] Dialog Semiconductor, «3-Phase Brushless DC Motor Control with Hall Sensors», Application Note AN-CM-244, 2018.

[10] G. Tomasello, «Advanced BLDC Motor Drive and Control», 2017.

[11] «AN885, Brushless DC (BLDC) Motor Fundamentals», p. 20, 2003.

[12] «BLDC Motor Control with Hall Sensors Driven by DSC», p. 28.

[13] Japan Servo, «bldc-technology-overview.pdf». [En línea]. Disponible en: https://bit.ly/2yxVpwy. [Accedido: 02-ago-2019].

[14] «Driving Three-Phase Stepper Motor With BLDC Motor Driver Reference Design», p. 16, 2017.

[15] D. Arrigo, «L6234 three phase motor driver», p. 14.

[16] «LV8907UW - Sensor-less Three-phase Brushless DC Motor Controller, with Gate Drivers, for Automotive», p. 45.

[17] Semiconductor Components Industries, «Sensor-less Three-phase Brushless DC Motor Controller, with Gate Drivers, for Automotive», Semiconductor Components Industries, Datasheet LV8907UW/D, 2019.

VIII. ANEXOS

VIII-A. Código del controlador

```
#define vel_max 3600
//pines
#define  rx 0
#define  tx 1
#define  IsenceD 4
#define  In1 3
#define  In2 5
#define  In3 6
#define  En1 7
#define  En2 8
#define  En3 9
#define  Bf1 10
#define  Bf2 11
#define  Bf3 12
#define  LED1 A0
#define  LED2 A1
#define  LED3 A2
#define  LED4 A3
#define  LED5 13
#define  IsenceA A6
//////Variables glovales /////////
int RPM=1000,vr1;
float corr=0;
char giro=2,ess=2;

void setup() {
  //configuracion de pines
  Serial.begin(115200);
  pinMode(IsenceD, INPUT_PULLUP);
  pinMode(In1, OUTPUT);
  pinMode(In2, OUTPUT);
  pinMode(In3, OUTPUT);
  pinMode(En1, OUTPUT);
  pinMode(En2, OUTPUT);
  pinMode(En3, OUTPUT);
  pinMode(Bf1, INPUT_PULLUP);
  pinMode(Bf2, INPUT_PULLUP);
  pinMode(Bf3, INPUT_PULLUP);
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
  pinMode(LED5, OUTPUT);

  vr1=map(RPM,0,vel_max,0,255);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (Serial.available()>0){
    comunicacion();
  }
  enviardatos();
  if(ess==2){
    digitalWrite(LED1,LOW);
    digitalWrite(LED2,LOW);
    digitalWrite(LED3,LOW);
    digitalWrite(LED4,LOW);
    digitalWrite(LED5,LOW);
    digitalWrite(En1,LOW);
    digitalWrite(En2,LOW);
    digitalWrite(En3,LOW);
    analogWrite(In1,0);
    analogWrite(In2,0);
    analogWrite(In3,0);
  }
  else if(giro==2){
    int estado[4]={digitalRead(Bf1),digitalRead(
        Bf2),digitalRead(Bf3),digitalRead(IsenceD)
        };
    if(estado[3]==LOW){
```

```
      if(estado[0]==LOW&&estado[1]==LOW&&estado
          [2]==LOW){
        digitalWrite(En1,LOW);
        digitalWrite(En2,LOW);
        digitalWrite(En3,LOW);
        analogWrite(In1,0);
        analogWrite(In2,0);
        analogWrite(In3,0);
        digitalWrite(LED1,LOW);
        digitalWrite(LED2,LOW);
        digitalWrite(LED3,LOW);
        digitalWrite(LED4,LOW);
        digitalWrite(LED5,HIGH);
      }
      //1
      else if(estado[0]==HIGH&&estado[1]==LOW&&
          estado[2]==LOW){
        digitalWrite(En1,HIGH);
        digitalWrite(En2,LOW);
        digitalWrite(En3,HIGH);
        analogWrite(In1,vr1);
        analogWrite(In2,0);
        analogWrite(In3,0);
        digitalWrite(LED1,HIGH);
        digitalWrite(LED2,LOW);
        digitalWrite(LED3,LOW);
        digitalWrite(LED4,LOW);
        digitalWrite(LED5,HIGH);
      }
      //2
      else if(estado[0]==LOW&&estado[1]==HIGH&&
          estado[2]==LOW){
        digitalWrite(En1,HIGH);
        digitalWrite(En2,HIGH);
        digitalWrite(En3,LOW);
        analogWrite(In1,0);
        analogWrite(In2,vr1);
        analogWrite(In3,vr1);
        digitalWrite(LED1,LOW);
        digitalWrite(LED2,HIGH);
        digitalWrite(LED3,LOW);
        digitalWrite(LED4,LOW);
        digitalWrite(LED5,HIGH);
      }
      //3
      else if(estado[0]==HIGH&&estado[1]==HIGH&&
          estado[2]==LOW){
        digitalWrite(En1,LOW);
        digitalWrite(En2,HIGH);
        digitalWrite(En3,HIGH);
        analogWrite(In1,vr1);
        analogWrite(In2,vr1);
        analogWrite(In3,0);
        digitalWrite(LED1,HIGH);
        digitalWrite(LED2,HIGH);
        digitalWrite(LED3,LOW);
        digitalWrite(LED4,LOW);
        digitalWrite(LED5,HIGH);
      }
      //4
      else if(estado[0]==LOW&&estado[1]==LOW&&
          estado[2]==HIGH){
        digitalWrite(En1,LOW);
        digitalWrite(En2,HIGH);
        digitalWrite(En3,HIGH);
        analogWrite(In1,0);
        analogWrite(In2,0);
        analogWrite(In3,vr1);
        digitalWrite(LED1,LOW);
        digitalWrite(LED2,LOW);
        digitalWrite(LED3,HIGH);
        digitalWrite(LED4,LOW);
        digitalWrite(LED5,HIGH);
      }
      //5
      else if(estado[0]==HIGH&&estado[1]==LOW&&
          estado[2]==HIGH){
        digitalWrite(En1,HIGH);
        digitalWrite(En2,HIGH);
        digitalWrite(En3,LOW);
        analogWrite(In1,vr1);
        analogWrite(In2,0);
        analogWrite(In3,0);
        digitalWrite(LED1,HIGH);
        digitalWrite(LED2,LOW);
        digitalWrite(LED3,HIGH);
        digitalWrite(LED4,LOW);
        digitalWrite(LED5,HIGH);
      }
      //6
      else if(estado[0]==LOW&&estado[1]==HIGH&&
          estado[2]==HIGH){
        digitalWrite(En1,HIGH);
        digitalWrite(En2,LOW);
        digitalWrite(En3,HIGH);
        analogWrite(In1,0);
        analogWrite(In2,vr1);
        analogWrite(In3,vr1);
        digitalWrite(LED1,LOW);
        digitalWrite(LED2,HIGH);
        digitalWrite(LED3,HIGH);
        digitalWrite(LED4,LOW);
        digitalWrite(LED5,HIGH);
      }
      else if(estado[0]==HIGH&&estado[1]==HIGH&&
          estado[2]==HIGH){
        digitalWrite(En1,LOW);
        digitalWrite(En2,LOW);
        digitalWrite(En3,LOW);
        analogWrite(In1,0);
        analogWrite(In2,0);
        analogWrite(In3,0);
        digitalWrite(LED1,LOW);
        digitalWrite(LED2,LOW);
        digitalWrite(LED3,LOW);
        digitalWrite(LED4,LOW);
        digitalWrite(LED5,HIGH);
      }
    }
    else{
      digitalWrite(En1,LOW);
      digitalWrite(En2,LOW);
      digitalWrite(En3,LOW);
      analogWrite(In1,0);
      analogWrite(In2,0);
      analogWrite(In3,0);
      digitalWrite(LED1,HIGH);
      digitalWrite(LED2,HIGH);
      digitalWrite(LED3,HIGH);
      digitalWrite(LED4,HIGH);
      digitalWrite(LED5,HIGH);
    }
  }
  else{
    int estado[4]={digitalRead(Bf1),digitalRead(
        Bf2),digitalRead(Bf3),digitalRead(IsenceD)
        };
    if(estado[3]==LOW){
      if(estado[0]==LOW&&estado[1]==LOW&&estado
          [2]==LOW){
        digitalWrite(En1,LOW);
        digitalWrite(En2,LOW);
        digitalWrite(En3,LOW);
        analogWrite(In1,0);
        analogWrite(In2,0);
        analogWrite(In3,0);
        digitalWrite(LED1,LOW);
        digitalWrite(LED2,LOW);
        digitalWrite(LED3,LOW);
        digitalWrite(LED4,HIGH);
        digitalWrite(LED5,LOW);
      }
      //1
```

```
      else if(estado[0]==HIGH&&estado[1]==LOW&&
          estado[2]==LOW){
        digitalWrite(En1,HIGH);
        digitalWrite(En2,LOW);
        digitalWrite(En3,HIGH);
        analogWrite(In1,0);
        analogWrite(In2,vr1);
        analogWrite(In3,vr1);
        digitalWrite(LED1,HIGH);
        digitalWrite(LED2,LOW);
        digitalWrite(LED3,LOW);
        digitalWrite(LED4,HIGH);
        digitalWrite(LED5,LOW);
      }
      //2
      else if(estado[0]==LOW&&estado[1]==HIGH&&
          estado[2]==LOW){
        digitalWrite(En1,HIGH);
        digitalWrite(En2,HIGH);
        digitalWrite(En3,LOW);
        analogWrite(In1,vr1);
        analogWrite(In2,0);
        analogWrite(In3,0);
        digitalWrite(LED1,LOW);
        digitalWrite(LED2,HIGH);
        digitalWrite(LED3,LOW);
        digitalWrite(LED4,HIGH);
        digitalWrite(LED5,LOW);
      }
      //3
      else if(estado[0]==HIGH&&estado[1]==HIGH&&
          estado[2]==LOW){
        digitalWrite(En1,LOW);
        digitalWrite(En2,HIGH);
        digitalWrite(En3,HIGH);
        analogWrite(In1,0);
        analogWrite(In2,0);
        analogWrite(In3,vr1);
        digitalWrite(LED1,HIGH);
        digitalWrite(LED2,HIGH);
        digitalWrite(LED3,LOW);
        digitalWrite(LED4,HIGH);
        digitalWrite(LED5,LOW);
      }
      //4
      else if(estado[0]==LOW&&estado[1]==LOW&&
          estado[2]==HIGH){
        digitalWrite(En1,LOW);
        digitalWrite(En2,HIGH);
        digitalWrite(En3,HIGH);
        analogWrite(In1,vr1);
        analogWrite(In2,vr1);
        analogWrite(In3,0);
        digitalWrite(LED1,LOW);
        digitalWrite(LED2,LOW);
        digitalWrite(LED3,HIGH);
        digitalWrite(LED4,HIGH);
        digitalWrite(LED5,LOW);
      }
      //5
      else if(estado[0]==HIGH&&estado[1]==LOW&&
          estado[2]==HIGH){
        digitalWrite(En1,HIGH);
        digitalWrite(En2,HIGH);
        digitalWrite(En3,LOW);
        analogWrite(In1,0);
        analogWrite(In2,vr1);
        analogWrite(In3,vr1);
        digitalWrite(LED1,HIGH);
        digitalWrite(LED2,LOW);
        digitalWrite(LED3,HIGH);
        digitalWrite(LED4,HIGH);
        digitalWrite(LED5,LOW);
      }
      //6
      else if(estado[0]==LOW&&estado[1]==HIGH&&
          estado[2]==HIGH){
        digitalWrite(En1,HIGH);
        digitalWrite(En2,LOW);
        digitalWrite(En3,HIGH);
        analogWrite(In1,vr1);
        analogWrite(In2,0);
        analogWrite(In3,0);
        digitalWrite(LED1,LOW);
        digitalWrite(LED2,HIGH);
        digitalWrite(LED3,HIGH);
        digitalWrite(LED4,HIGH);
        digitalWrite(LED5,LOW);
      }
      else if(estado[0]==HIGH&&estado[1]==HIGH&&
          estado[2]==HIGH){
        digitalWrite(En1,LOW);
        digitalWrite(En2,LOW);
        digitalWrite(En3,LOW);
        analogWrite(In1,0);
        analogWrite(In2,0);
        analogWrite(In3,0);
        digitalWrite(LED1,LOW);
        digitalWrite(LED2,LOW);
        digitalWrite(LED3,LOW);
        digitalWrite(LED4,HIGH);
        digitalWrite(LED5,LOW);
      }
    }
    else{
      digitalWrite(En1,LOW);
      digitalWrite(En2,LOW);
      digitalWrite(En3,LOW);
      analogWrite(In1,0);
      analogWrite(In2,0);
      analogWrite(In3,0);
      digitalWrite(LED1,HIGH);
      digitalWrite(LED2,HIGH);
      digitalWrite(LED3,HIGH);
      digitalWrite(LED4,HIGH);
      digitalWrite(LED5,HIGH);
    }
  }
}
void comunicacion(void){
    char gar =Serial.read();
    //configuracion
    if (gar=='v'){
      vr1=map(Serial.readString().toInt()
          ,0,100,0,255);
    }
    else if(gar=='e'){
      ess=Serial.readString().toInt();
    }
    else if(gar=='g'){
      giro=Serial.readString().toInt();
    }
}

void enviardatos(void){
  Serial.print("v");
  RPM=map(vr1,0,255,0,3600);
  Serial.print(RPM);
  Serial.print("V");
  int adccorr=map(analogRead(IsenceA),0,1024,0,6);
  Serial.print("i");
  Serial.print(adccorr);
  Serial.print("I");
  Serial.print("e");
  Serial.print(ess);
  Serial.print("E");
  Serial.print("g");
  Serial.print(giro);
  Serial.print("G\n");
}
```

VIII-B. Diagrama esquemático

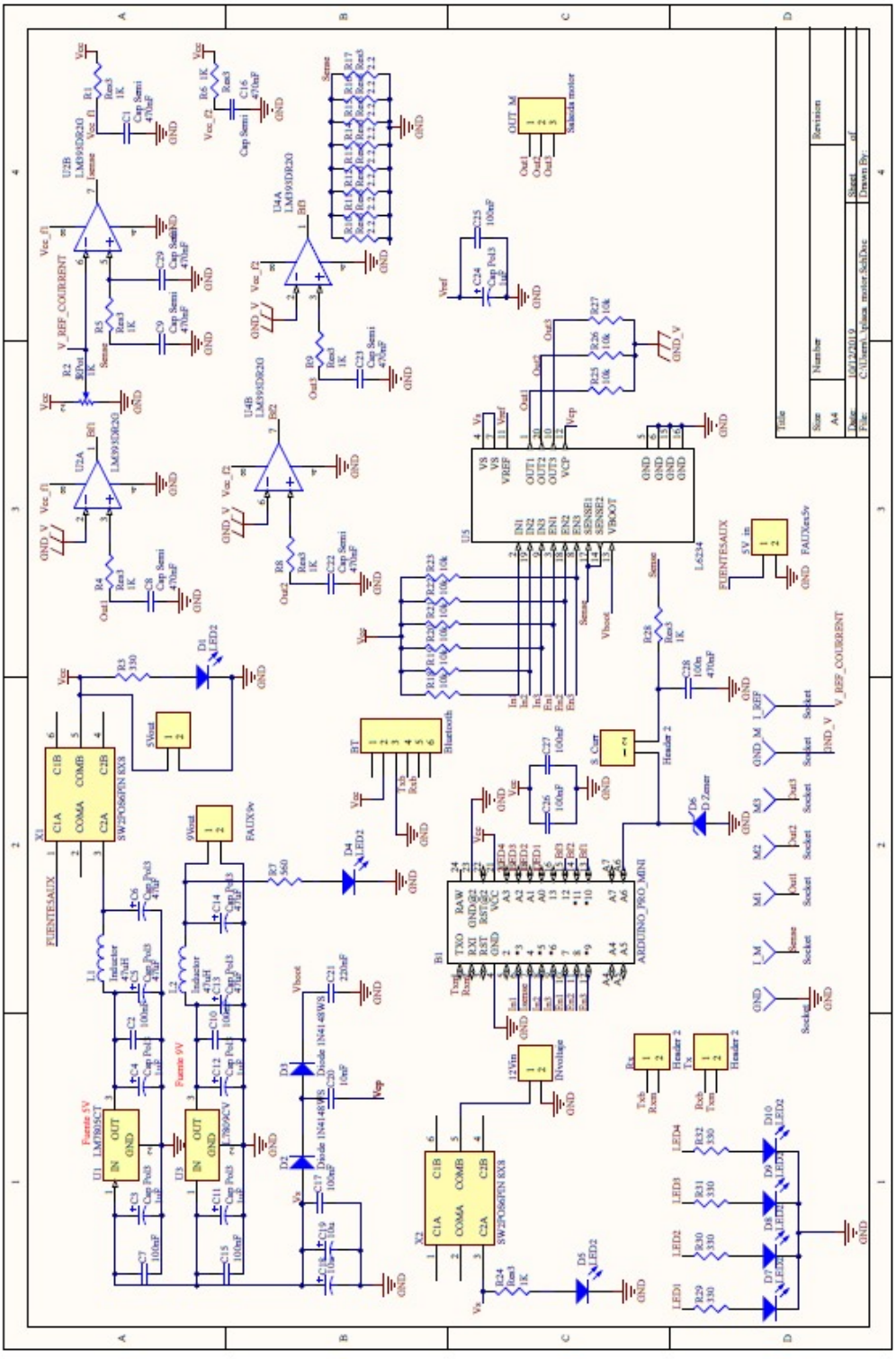

Figura 13: Circuito esquemático

VIII-C. Modulos de Labview

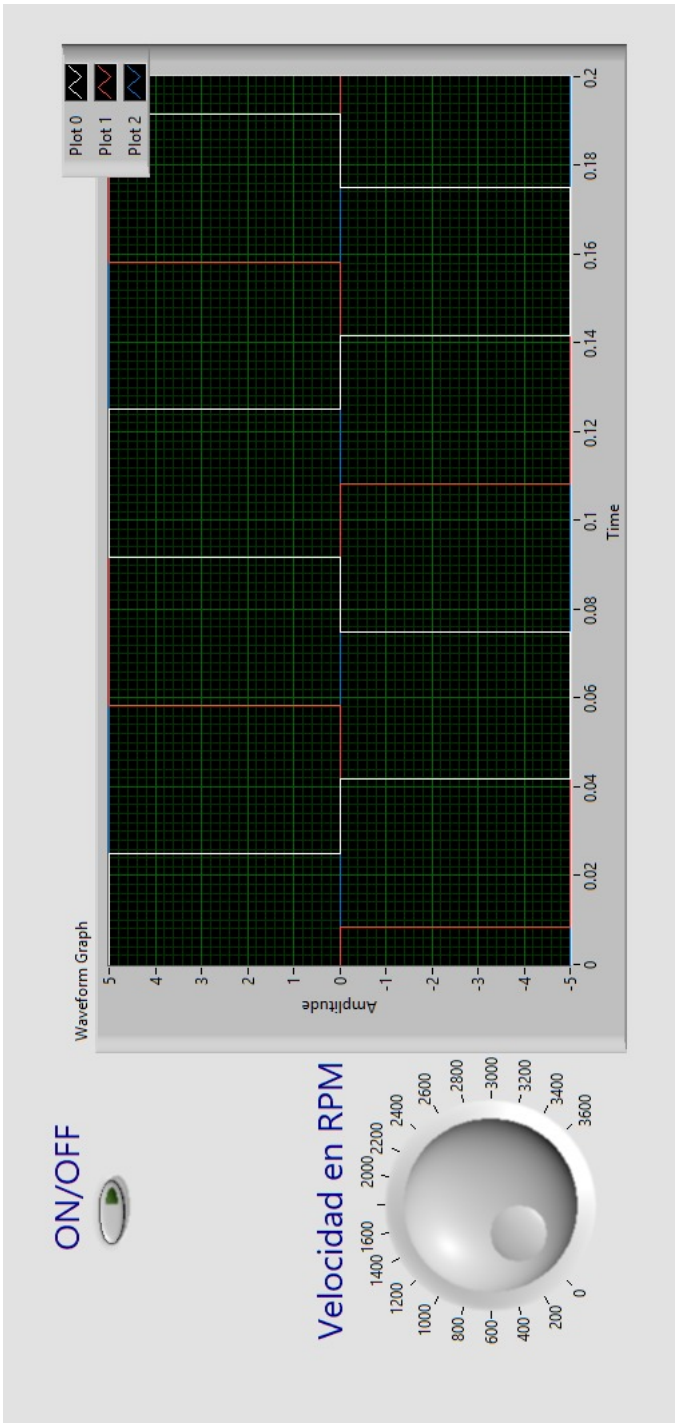

Figura 14: Interfaz gráfica en Labview

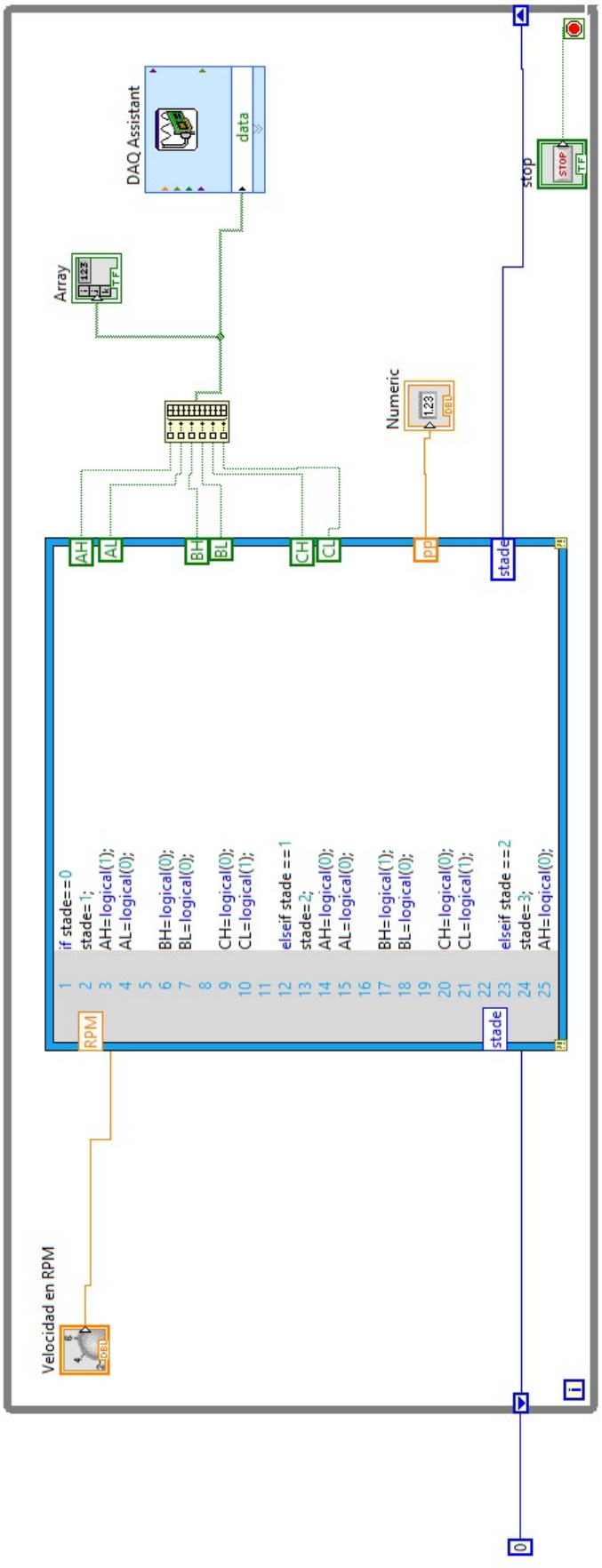

Figura 15: Módulos en Labview

VIII-D. Circuito impreso

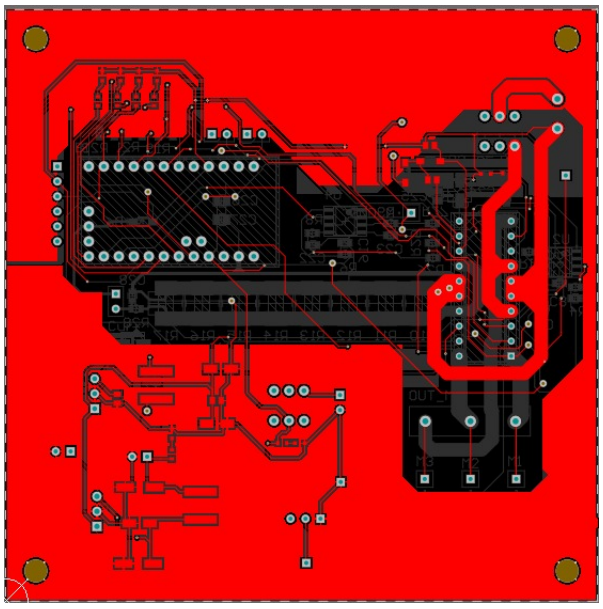

Figura 16: Capa superior del circuito impreso

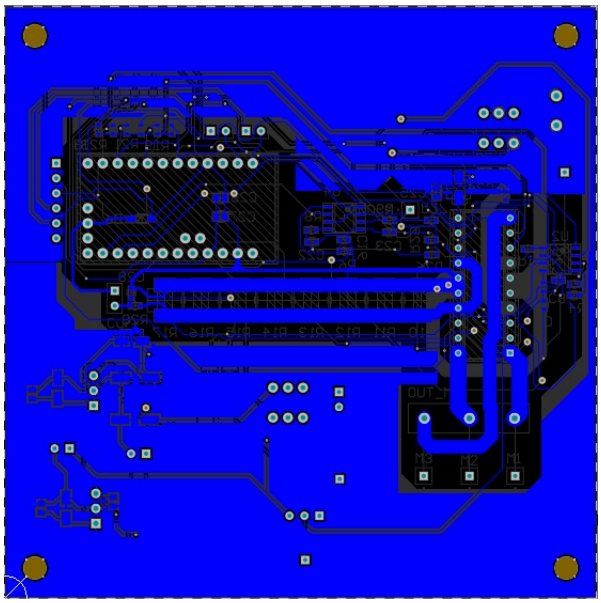

Figura 17: Capa inferior del circuito impreso

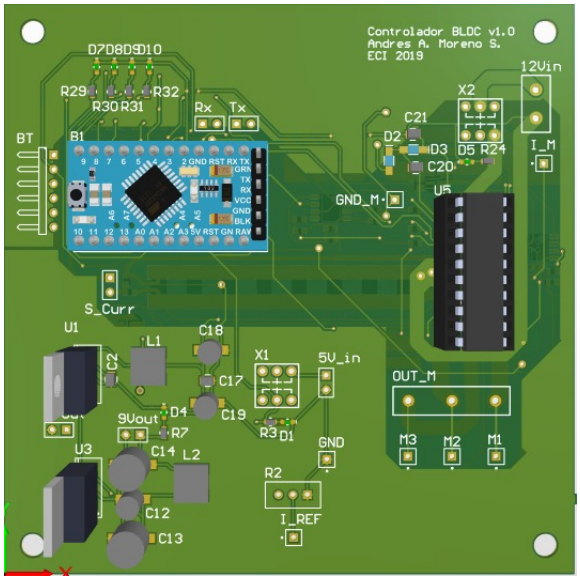

Figura 18: Capa superior 3D del circuito impreso

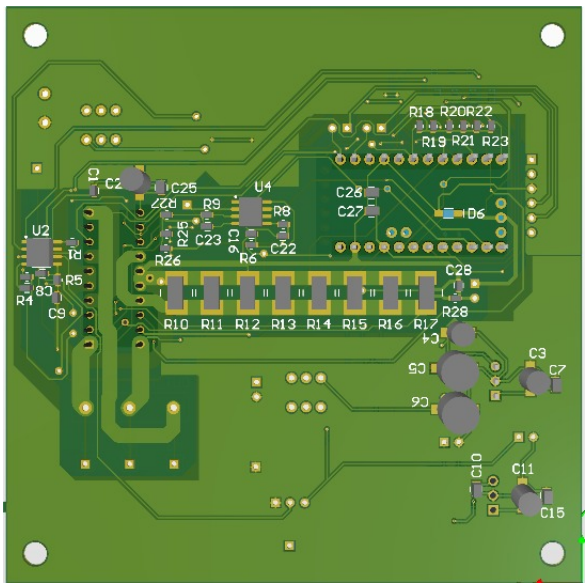

Figura 19: Capa inferior 3D del circuito impreso