

ESCUELA DE EDUCACION SECUNDARIA TECNICA N° 5

“2 DE ABRIL” – TEMPERLEY – BUENOS AIRES



Empresa:
Control General (parámetros de impresora 3D)

MATERIA : PROYECTO Y DISEÑO ELECTRÓNICO

GRUPO :

AUTORES:

Alumnos: JOAQUÍN PIMPIGNANO

NOTAS:

PROFESOR: ING. MARTIN LEGUIZAMON

INFORME FINAL DE TALLER

Joaquín Pimpignano

E.E.S.T. N°5 "2 de Abril" Temperley. Lomas de Zamora

7° 6° T.V. Grupo 2

Ing. Leguizamon Martin

14 de Noviembre de 2024

Descripción:	3
Introducción:	3
Diagrama de Gantt:	3
Programación:	4
Diagrama en bloques:	4
Síntesis del código:	4
Código:	4
3D:	5
Plano:	5
Modelo 3D:	5
Resultados:	5
Esquemático:	6
Placas:	7
Electrónica:	8
MSP430G2553-20:	8
LCD:	9
DRIVER STEPPER:	9
TEMPERATURA Y FILAMENTO:	14
CALEFACTOR:	16
VARIOS:	16
Proceso de fabricación:	18
REVISIÓN 1:	18
REVISIÓN 2:	19
REVISIÓN 3:	20
REVISIÓN 4:	20
REVISIÓN 4.1:	21
BOM (Bill of Materials):	22
Conclusión:	24
Bibliografía:	25

Descripción:

El Control de parámetros de impresora 3D es un simple módulo para implementar en cualquier impresora 3D y nos puede ayudar a llevar un mejor control de su funcionamiento y su calibre. Este incluye funciones como el control de temperatura y de motores paso a paso y sensores de filamento y más.

Todo esto al alcance del usuario mediante un display y tres botones ubicados en la cara superior de la carcasa.

Este dispositivo es óptimo para cualquier persona que quiera mejorar la calidad y resultados de sus impresiones de manera simple. Es un proyecto con una complejidad considerable, debido a su extenso diseño y arduo montaje, además de una precisa programación.

Introducción:

Este dispositivo está enteramente programado para el módulo MSP430G2553-20 producido por Texas Instruments implementado directamente en la placa en su formato más pequeño. Mediante el display LCD-016N002L y el potenciómetro para el ajuste del contraste, se le muestra la interfaz al usuario y él mismo la controla con los tres botones. Todos estos elementos se encuentran en la placa superior junto con el sensor de temperatura NTC 100k y los pines correspondientes para la programación y la transmisión de datos a la placa de abajo.

En la placa inferior encontramos el resto de la electrónica como las entradas de alimentación y el regulador LD33V que ajusta la tensión a 3.3v, el driver de motores POLOLU A4988, el buzzer, el sensor del filamento y la salida al calefactor controlado por un RELAY.

Diagrama de Gantt:

Fecha de inicio oficial: 1 de Julio de 2024

Fecha de finalización: 15 de Noviembre de 2024

Tareas:

Entrega del anteproyecto:	Finalizado.
Diseño de placa:	Finalizado.
Diseño y fabricación 3D:	Finalizado.
Fabricación de placa:	Finalizado.
Construcción:	Finalizado.
Prueba de funcionamiento:	Finalizado.

Programación:

Código:

```

/*****
*
*          FILAMENTO DE BOTELLAS
*          *****/
* VERSIÓN 1.3
* 22/9/2024
*
*****/

#include "io430.h"
#include "LCD.h"
#include "DELAY.h"
#include "Tipos.h"
#include "FLASH.h"

/*****

#define Enable_Interrupts    __bis_SR_register(GIE)
#define Disable_Interrupts   __bic_SR_register(GIE)
#define Enable_IrqTimer0     TACTL0|=TAIE
#define Disable_IrqTimer0    TACTL0&=~TAIE
#define Enable_IrqTeclas     P2IE=0x07
#define Disable_IrqTeclas    P2IE=0

***** DEFINICIÓN DE CONEXIONES *****/

#define ENABLE 0x80
#define SENSOR_FIL (P2IN & 0x08)
#define BUZZER 0x40
#define CALEFACTOR 0x01

/***** ACCIONES *****/
#define BUZZER_ON          P2OUT |= BUZZER
#define BUZZER_OFF        P2OUT &= ~BUZZER
#define CALEFACTOR_ON      P1OUT |= CALEFACTOR
#define CALEFACTOR_OFF     P1OUT &= ~CALEFACTOR
#define MOTOR_OFF          P2OUT |= ENABLE
#define MOTOR_ON           P2OUT &= ~ENABLE
#define ACTIVA_TRACCION     TA1CTL |= MC0
#define DESACTIVA_TRACCION  TA1CTL &= ~MC0
#define ACTIVA_CONTROL_TEMP TA0CTL |= (MC1 + TAIE)
#define DESACTIVA_CONTROL_TEMP TA0CTL &= ~(MC1 + TAIE)

#define T_BEEP_CORTO      50
#define T_BEEP_LARGO      300

/***** PROTOTIPOS DE FUNCIONES *****/
void Beep_corto(void);
void Beep_largo(void);
unsigned int Medir(void);    // TOMA MEDICIONES CON EL A/D
int MideTemp();             // CALCULA LA TEMPERATURA MEDIDA
void ActivaMotor (void);
void DesactivaMotor(void);
void Dormir(void);          // MODO BAJO CONSUMO
void pantalla_1(void);
void pantalla_2(void);
void pantalla_3(void);
void pantalla_5(void);
void Read_Status(void);     // LECTURA DE DATOS DE LA FLASH
void Status_Backup(void);   // ESCRITURA DE DATOS EN LA FLASH

/***** VARIABLES GLOBALES *****/

```

```

int *Base_addr = (int*)0xfc00;      // INICIO DEL BLOQUE DE FLASH PARA USAR COMO EEPROM
int *Save_addr;
byte temperatura, velocidad;
byte tecla=0;
int t_out = 0;
const int AD[] = {13,39,93,185,314,465,612,738,826,890};      // LECTURAS DEL A/D CADA 25°C (CALIBRACIÓN
DEL TERMISTOR)
const int velTimer[] = {15360,14260,13160,12060,10970,9870,8770,7680,6580,5480,4390};      // VALORES DEL TIMER PARA
CONTROL DE VELOCIDAD
const int HISTERESIS = 0;      // SEPARACIÓN ENTRE EL APAGADO Y ENCENDIDO DEL
CALEFACTOR [°C]
int tempMedida = 0;

/*****
int main( void )
{

    WDTCTL = WDTPW + WDTHOLD;      // DESHABILITO EL WATCHDOG TIMER PARA PREVENIR RESET
    Disable_Interrupts;

    /***** CONFIGURACIÓN DEL CLK *****/
    /*
    * Basic Clock System Control 2
    *
    * SELM_0 -- DCOCLK
    * DIVM_0 -- Divide by 1
    * ~SELS -- DCOCLK
    * DIVS_0 -- Divide by 1
    * ~DCOR -- DCO uses internal resistor
    *
    * Note: ~<BIT> indicates that <BIT> has value zero
    */

    BCSCCTL2 = 0x00; /* Conecto MCLK y SMCLK al DCO y divisores en 1 en ambos*/
    DCOCTL = 0x00;
    BCSCCTL1 = CALBC1_12MHZ; /* Set DCO to 12MHz */
    DCOCTL = CALDCO_12MHZ;

    /*
    * Basic Clock System Control 1
    *
    * XT2OFF -- Disable XT2CLK
    * ~XTS -- Low Frequency
    * DIVA_0 -- Divide by 1
    *
    * Note: ~XTS indicates that XTS has value zero
    */
    BCSCCTL1 |= XT2OFF | DIVA_0;

    /*
    * Basic Clock System Control 3
    *
    * XT2S_0 -- 0.4 - 1 MHz
    * LFXT1S_2 -- If XTS = 0, XT1 = VLOCLK ; If XTS = 1, XT1 = 3-16-MHz crystal
    * XCAP_1 -- ~6 pF
    */
    BCSCCTL3 = XT2S_0 | LFXT1S_2 | XCAP_1;

    /***** CONFIGURACIÓN DE PUERTOS DE E/S *****/
    /* Port 1 Output Register */
    P1OUT = 0;

    /* Port 1 Direction Register */
    P1DIR = BIT0 | BIT2 | BIT3 | BIT4 | BIT5 | BIT6 | BIT7; /* Salvo el PIN 1, el resto salidas */

    /* Pull up/downn */
    P1REN = 0;      // SIN PULL UP/DOWN

    /* Port 1 Interrupt Flag Register */

```

```

P1IFG = 0;

P1OUT = 0;

/* Port 2 Output Register */
P2OUT = 0;

/* Port 2 Port Select Register */
P2SEL = 0;
P2SEL2 = 0;
P2SEL |= BIT4;    // SELECCIONO LA FUNCIÓN SECUNDARIA P2.4 (TA1.2)

/* Port 2 Direction Register */
P2DIR = BIT4 | BIT5 | BIT6 | BIT7;

P2REN = BIT0 | BIT1 | BIT2 | BIT3;

P2OUT = 0x0f;    //PULL UP EN P2.0:3 (PULSADORES Y SENSOR DE FILAMENTO)

/* Port 2 Interrupt Flag Register */
P2IFG = 0;

/* Port 2 Interrupt Edge Select */
P2IES = 0x07;    // PULSADORES GENERAN IRQ CON FLANCO DESCENDENTE

P2IE = 0x07;    // HABILITO INTERRUPCIONES EN LAS ENTRADAS CORRESPONDIENTES

/***** CONFIGURACIÓN DEL A/D *****/
ADC10CTL0 = 0x1c00;
ADC10CTL1 = 0x10f8;    // input A1
ADC10AE0 |= 0x0002;    // PA.1 ADC option select

/***** CONFIGURACIÓN DE TIMERS *****/
// Configuración del Timer_A0 para generar IRQ y chequear temperatura

TA0CTL = TASSEL1 | ID1 | ID0 | TAIE;
TA0CCR0 = 0xffff;

// Configuración del TimerB o Timer1_A para PWM en P2.4 (TA1.2)

TA1CTL = TASSEL1 | ID1 | ID0; /* SMCLK como fuente dividido por 8 */

/* TA0CCR0, Timer_A Capture/Compare Register 0 */
TA1CCR0 = 0;
TA1CCR2 = 0;
TA1CCTL2 |= OUTMOD2; /* Modo COMPARE, salida en TOGGLE */

MOTOR_OFF;

Save_addr = Base_addr;    // La primera dirección es la dirección base

// Configuración inicial de la pantalla

extern logic Bus;
BUS4;
Inicializa();
Retardo_ms(500);
Inicializa();
Borra();
Modo(CONT_ASCENDENTE, DISP_QUIETO);
Control(DISPLAY_ON,CURSOR_OFF,BLINK_OFF);
Retardo_ms(250);
pantalla_1();
tecla = 0;
Enable_Interrupts;
Retardo_ms(2000);
if (tecla==3){    // Si mantengo presionada la tecla 3 en el encendido restablezco valores por defecto

```

```

    temperatura = 200;
    velocidad = 10;
    Status_Backup();
    Borra();
    Lcd(1,1,"CONFIGURACION");
    Lcd(2,1,"RESTABLECIDA");
    Beep_largo();
    tecla=0;
    Retardo_ms(5000);
}
else {
    Read_Status();
    if (temperatura<180 || temperatura>240)
        temperatura = 200;
    if (velocidad<5 || velocidad>15)
        velocidad = 10;
    Beep_largo();
}

TA1CCR0 = velTimer[velocidad-5];

// Loop principal

Borra();
Enable_Interrupts;
while(1){
    pantalla_2();
}
}
// FUNCIONES

void pantalla_1(void){                                // Pantalla de inicio
    Borra();
    Lcd (1,2,"FILAMENTO POR");
    Lcd (2,0,"PULTRUSION V1.3");
}

void pantalla_2(void){                                // Pantalla principal
    tecla=0;
    Borra();
    Lcd (1,0,"TEMP:  VEL:");
    Lcd (2,0,"INICIO  CONFIG");
    Lcd_num(1,7,temperatura);
    Lcd_num(1,15,velocidad);
    Enable_IrqTeclas;
    Dormir();                                         // Pone al micro en modo bajo consumo a la espera de alguna tecla
    switch (tecla){
    case 1:
        pantalla_5();
        break;
    case 2:
        break;
    case 3:
        pantalla_3();
        break;
    }
    return;
}

void pantalla_3(void){                                // Ajuste de temperatura
    tecla=0;
    byte bucle = 1;
    t_out = 0;
    Borra();
    Lcd (1,0,"TEMPERAT.:  °C");
    Lcd (2,0,"SUBE  BAJA SIG");
    while(bucle == 1 && t_out < 10){
        Lcd_num (1,13,temperatura);
        Enable_IrqTeclas;
        switch (tecla){

```



```

case 1:
    tecla=0;
    temperatura+=5;
    if (temperatura > 240)
        temperatura = 240;
    Lcd (1,10," ");
    break;
case 2:
    tecla=0;
    temperatura-=5;
    if (temperatura < 180)
        temperatura = 180;
    Lcd (1,10," ");
    break;
case 3:
    tecla=0;
    bucle = 0;
    break;
}
}
if (t_out>=10){
    t_out=0;
    Status_Backup();
    Enable_Interrupts;
    return;
}
bucle = 1;
t_out=0;
Borra();
Lcd (1,0,"VELOCIDAD: ");
Lcd (2,0,"SUBE BAJA FIN");
while(bucle==1 && t_out<10){
    Lcd_num (1,12,velocidad);
    Enable_IrqTeclas;
    switch (tecla){
        case 1:
            tecla=0;
            velocidad +=1;
            if (velocidad > 15)
                velocidad = 15;
            Lcd (1,10," ");
            break;
        case 2:
            tecla=0;
            velocidad-=1;
            if (velocidad < 5)
                velocidad = 5;
            Lcd (1,10," ");
            break;
        case 3:
            tecla=0;
            bucle = 0;
            break;
    }
}
t_out=0;
Status_Backup();
Enable_Interrupts;
return;
}

void pantalla_5(void){
    byte bucle=1;
    tecla=0;
    int medicion=0;
    Borra();
    Lcd (1,3,"CALENTANDO");
    Lcd (2,10,"ABORTA");
    ACTIVA_CONTROL_TEMP;
    // Pantalla calentando

```

```

Enable_IrqTeclas;
while ((medicion=MideTemp())<temperatura){
  Lcd (2,0," °C");
  Lcd_num (2,3,medicion);
  if (tecla==3){
    CALEFACTOR_OFF;
    DESACTIVA_CONTROL_TEMP;
    tecla=0;
    return;
  }
  Retardo_ms(300);
}
Beep_corto();
while (1){
  Borra();
  // Atar filamento al tractor
  Lcd (1,2,"ATE FILAMENTO");
  Lcd (2,0,"CONT. ABORTA");
  bucle = 1;
  tecla=0;
  while (bucle){
    switch (tecla){
      case 1:
        if (!SENSOR_FIL)
          bucle=0;
        tecla=0;
        break;
      case 3:
        tecla=0;
        CALEFACTOR_OFF;
        DESACTIVA_CONTROL_TEMP;
        return;
    }
  }
  Borra();
  ActivaMotor();
  Enable_IrqTeclas;
  while (!SENSOR_FIL){
    if (!(P2OUT&ENABLE)){
      // Si no está detenido muestro pantalla de tracción
      Lcd (1,0,"TRACCION °C");
      Lcd_num (1,13,tempMedida);
      Lcd (2,0,"AJUS. PAUSA FIN");
    }
    switch (tecla){
      case 1:
        tecla=0;
        pantalla_3();
        Borra();
        break;
      case 2:
        tecla=0;
        P2OUT ^= ENABLE;
        if (P2OUT&ENABLE){
          Lcd (1,0,"");
          Lcd(1,4,"DETENIDO");
          Lcd(2,0," CONT.");
        }
        else {
          Borra();
        }
        Retardo_ms(150);
        break;
      case 3:
        tecla=0;
        DesactivaMotor();
        DESACTIVA_CONTROL_TEMP;
        CALEFACTOR_OFF;
        return;
    }
  }
}
tecla=0;

```

```

DesactivaMotor();
Borra();
Lcd (1,0,"** FINALIZADO **");
Lcd (2,0,"-----");
for (byte i = 0; i < 5; i++){
  Beep_largo();
  Retardo_ms(1000);
}
Borra();
Lcd (1,0,"RETIRE FILAMENTO");
Lcd (2,0,"CONTINUA  FIN");
tecla = 0;
t_out = 0;
Enable_IrqTeclas;
while (!tecla && t_out<100){
  switch (tecla){
    case 1:
      break;
    case 3:
      tecla=0;
      DESACTIVA_CONTROL_TEMP;
      CALEFACTOR_OFF;
      return;
  }
}
if (t_out>=100){
  DESACTIVA_CONTROL_TEMP;
  CALEFACTOR_OFF;
  for (byte i = 0; i < 5; i++){
    Beep_corto();
    Retardo_ms(200);
  }
  return;
}
}
}

void Beep_corto(void){
  BUZZER_ON;
  Retardo_ms(T_BEEP_CORTO);
  BUZZER_OFF;
  Retardo_ms(T_BEEP_CORTO);
  return;
}

void Beep_largo(void){
  BUZZER_ON;
  Retardo_ms(T_BEEP_LARGO);
  BUZZER_OFF;
  Retardo_ms(T_BEEP_LARGO);
  return;
}

unsigned int Medir(void){
  ADC10CTL0 |= ADC10ON;           // ENCIENDO EL A/D
  Retardo10us();
  ADC10CTL0 |= (ADC10SC + ENC);   // HABILITO E INICIO LA CONVERSIÓN
  while (ADC10CTL1 & ADC10BUSY);
  ADC10CTL0 &= ~(ADC10ON + ENC);
  ADC10CTL0 &= ~ADC10ON;
  return ADC10MEM;
}

int MideTemp(){
  unsigned int temp =0;
  double result,m,b;
  int n;
  for (byte i = 0; i<5; i++){      // Promedio la medida en 5 lecturas para mejorar la estabilidad
    temp+=Medir();
    Retardo_ms(5);
  }
}

```

```

    }
    temp=temp/5;
    for(n = 0; AD[n]<temp && n<10; n++){ // Hago interpolación lineal con el valor medido y los datos de calibración
        if (n==0)
            return 25;
        m = 25.0/(AD[n]-AD[n-1]);
        b = 25.0*(n+1) - m*AD[n];
        result = m*temp + b;

    return (int)result;
}

void ActivaMotor(void){
    ACTIVA_TRACCION;
    MOTOR_ON;
}

void DesactivaMotor(void){
    DESACTIVA_TRACCION;
    MOTOR_OFF;
}

void Dormir(void){
    Enable_Interrupts;
    LPM0;
    return;
}

//***** INTERRUPTOS *****

#pragma vector = TIMER1_A1_VECTOR
__interrupt void Timer1_A1 (void){
    if (TA1IV == 0x0a)
        return;
}

#pragma vector = TIMER0_A1_VECTOR
__interrupt void Timer0_A1 (void){
    TA1CCR0 = velTimer[velocidad-5]; // Acá aprovecho y actualizo el PWM
    static int irqs = 0;
    irqs++;
    if (irqs == 20){
        irqs = 0;
        tempMedida = MideTemp(); // Actualizo la medición de temperatura
        t_out++;
        if (tempMedida > temperatura + HISTERESIS)
            CALEFACTOR_OFF;
        else if (tempMedida < temperatura)
            CALEFACTOR_ON;
    }
    TA0CTL&= ~TAIFG; // Borro el flag de IRQ
    return;
}

#pragma vector = PORT2_VECTOR
__interrupt void Teclas(void){
    Disable_IrqTeclas;
    Beep_corto();
    Retardo_ms(300); // Para evitar el rebote de contactos
    switch (P2IFG&0x07){
    case 1:
        tecla=1;
        break;
    case 2:
        tecla=2;
        break;
    case 4:
        tecla=3;
        break;
    }
}

```

```

}
P2IFG = 0; // Borro los flags
Enable_IrqTeclas;
LPM0_EXIT; // Despertar
return;
}

//*****

// GRABACIÓN DE PARÁMETROS EN LA FLASH

void Status_Backup (void){
    int j=0;
    Save_addr = Base_addr; // Inicio la búsqueda desde la primera dirección del bloque
    BCSCCTL1 = CALBC1_1MHZ; /* Set DCO to 1MHz */ // Hay que bajar el SMCLK para poder grabar
    DCOCTL = CALDCO_1MHZ;
    for (j=0;j<256;j++){ // Recorro el bloque buscando la primera dirección libre (con 0xffff)
        if(*(Save_addr+j)== 0xffff){ // Tener en cuenta que se usan 2 posiciones para cada entero a grabar
            Save_addr+=j;
            Flash_Write (Save_addr, temperatura);
            Save_addr+=1;
            Flash_Write (Save_addr, velocidad);
            BCSCCTL1 = CALBC1_12MHZ; // Reconfiguro el SMCLK a 12MHz.
            DCOCTL = CALDCO_12MHZ;
            Retardo10us();
            return;
        }
    }
    Flash_Erase (Base_addr); // Si el bloque está lleno lo borro
    Save_addr = Base_addr;
    Flash_Write (Save_addr, temperatura);
    Save_addr+=1;
    Flash_Write (Save_addr, velocidad);
    BCSCCTL1 = CALBC1_12MHZ; /* Set DCO to 12MHz */
    DCOCTL = CALDCO_12MHZ;
    Retardo10us();
    return;
}

// RECUPERACIÓN DE PARÁMETROS DE LA FLASH

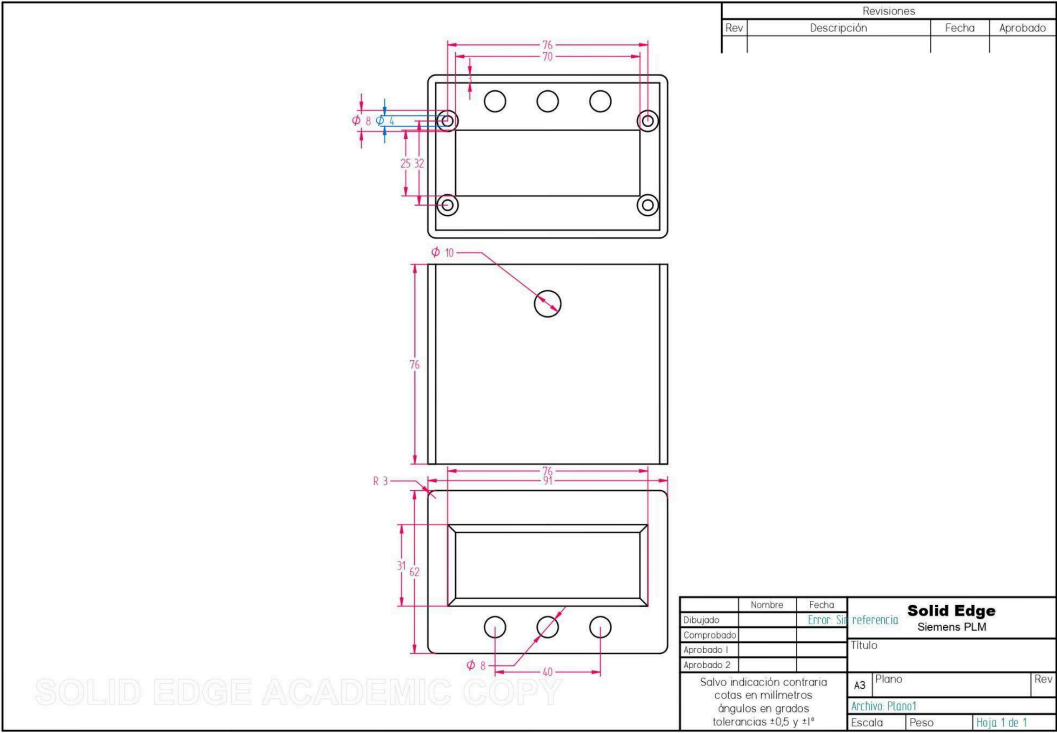
void Read_Status (void){ // Para recuperar el último backup recorro el bloque hasta encontrar la
    while (1){ // primera posición libre (0xffff) y tomo los 2 anteriores
        if(*(Save_addr == 0xffff){
            velocidad = *(Save_addr-1);
            temperatura = *(Save_addr-2);
            return;
        }
        Save_addr++;
    }
}

```

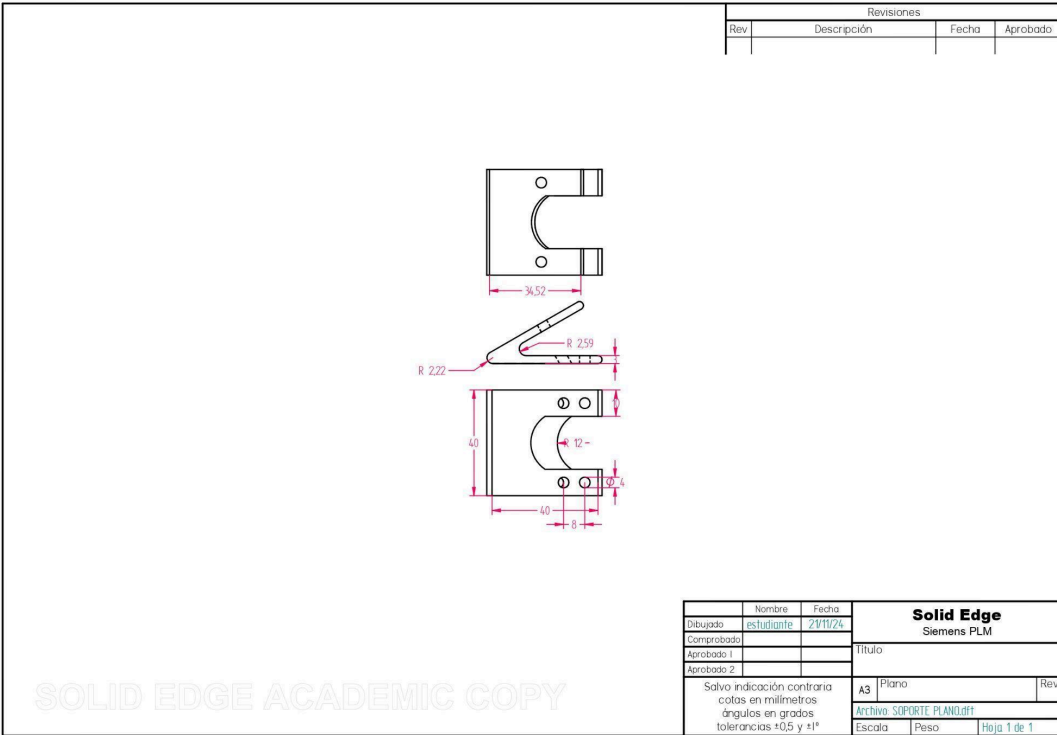
3D:

Plano:

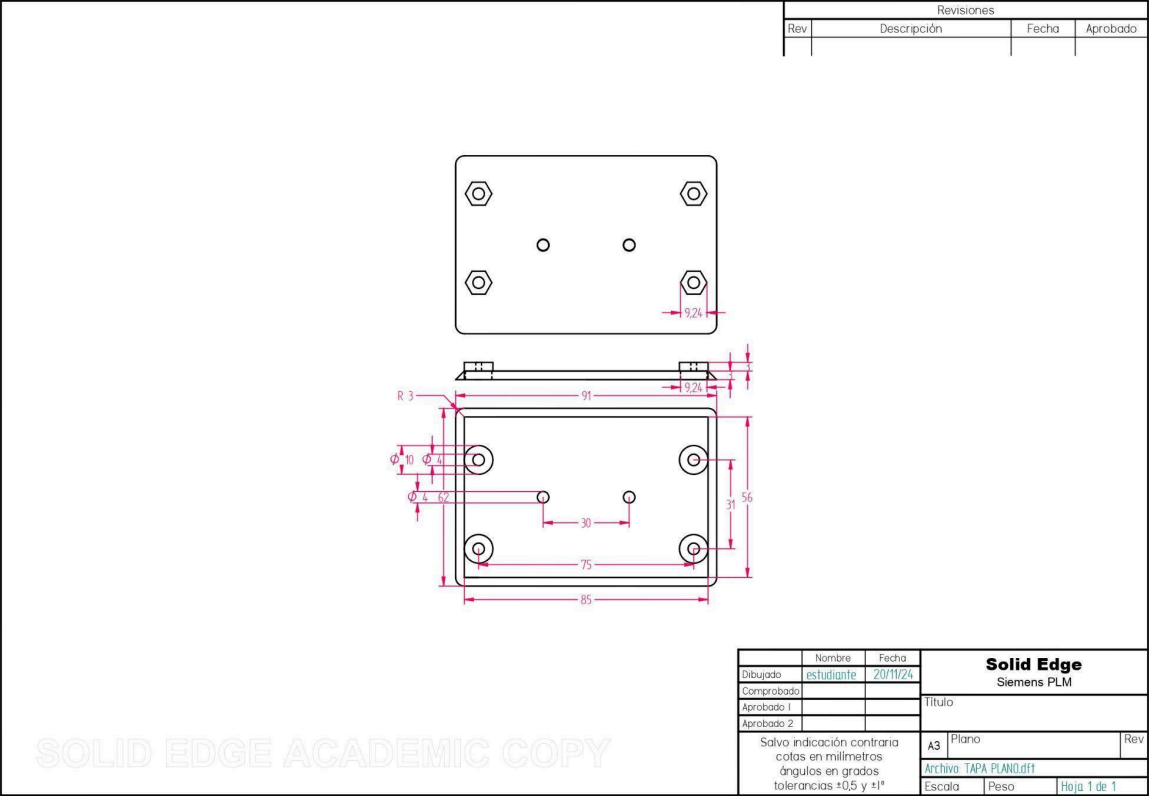
CAJA:



SOPORTE:

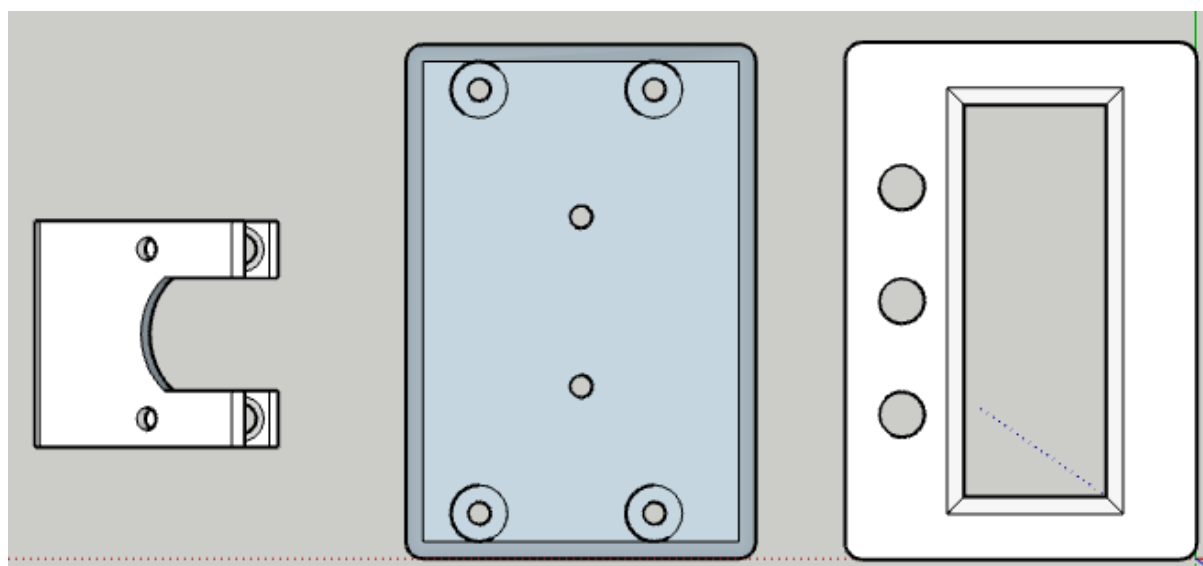


TAPA:

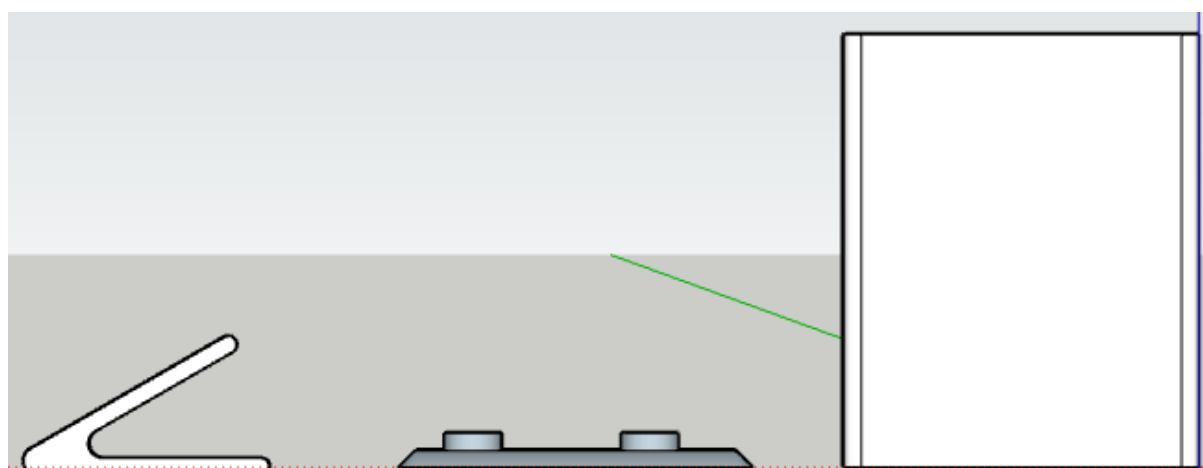


Modelo 3D:

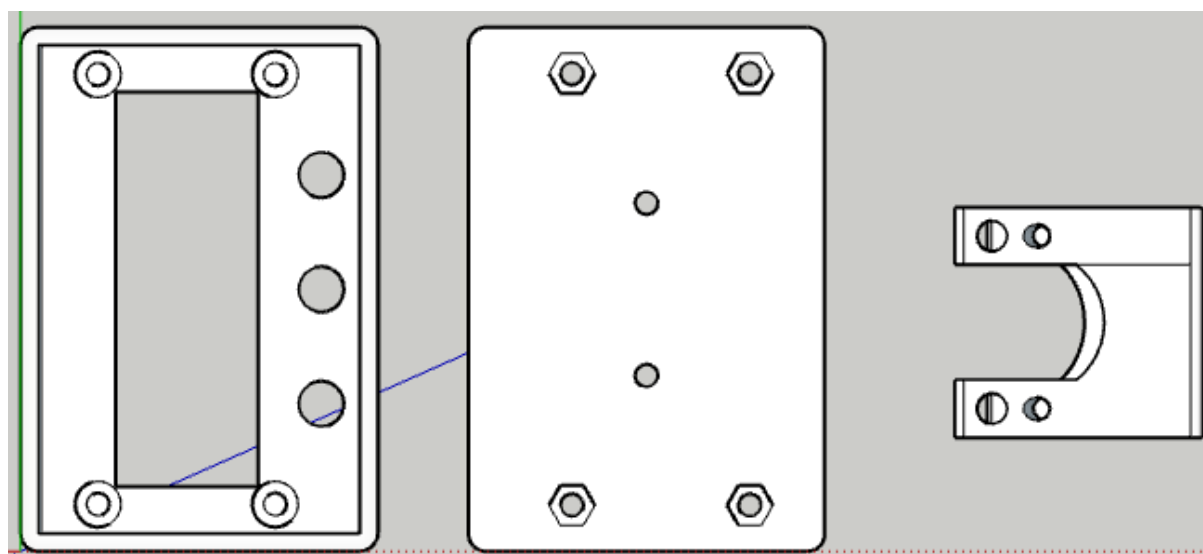
Vista Planta:



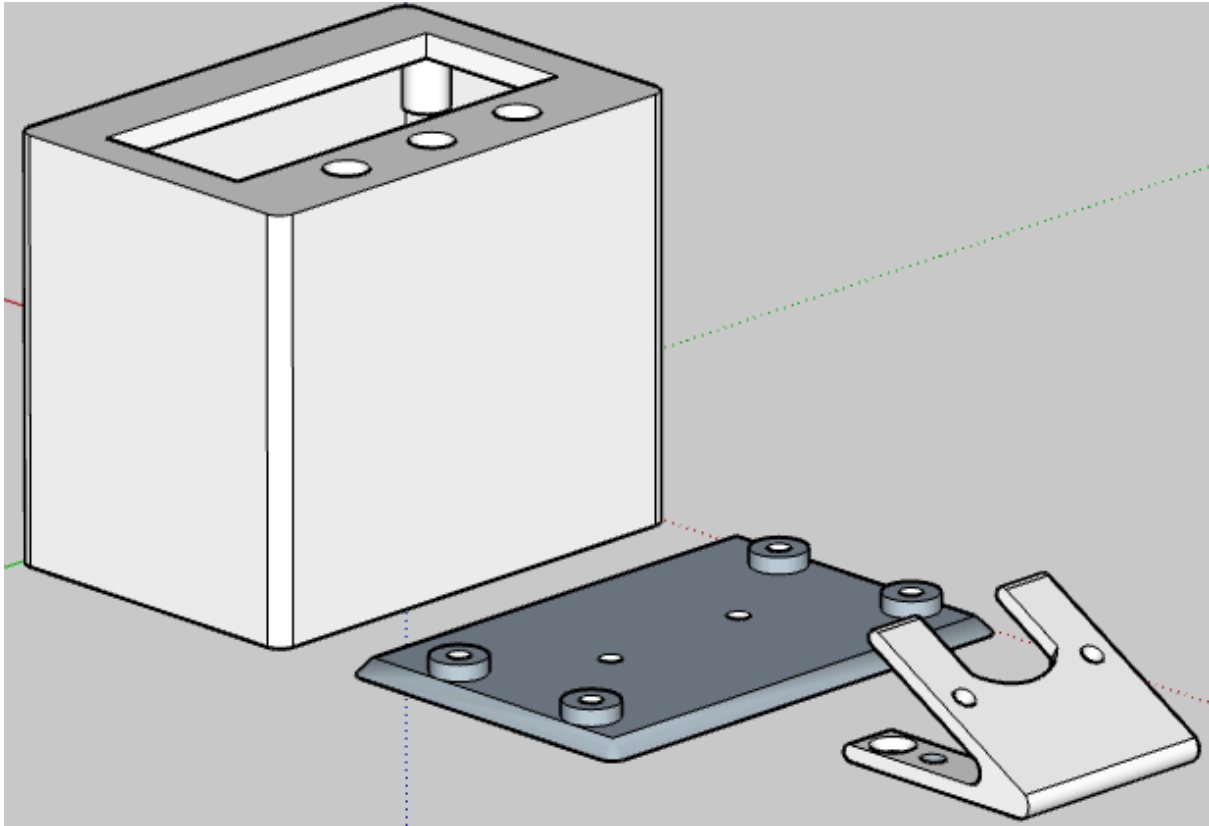
Vista alzada Sur:



Vista Inferior:

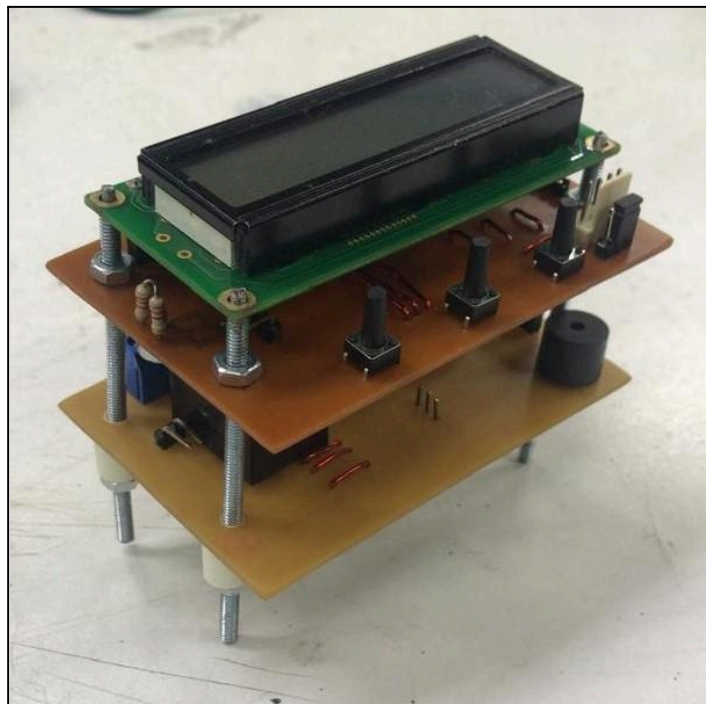


Vista Isométrica:

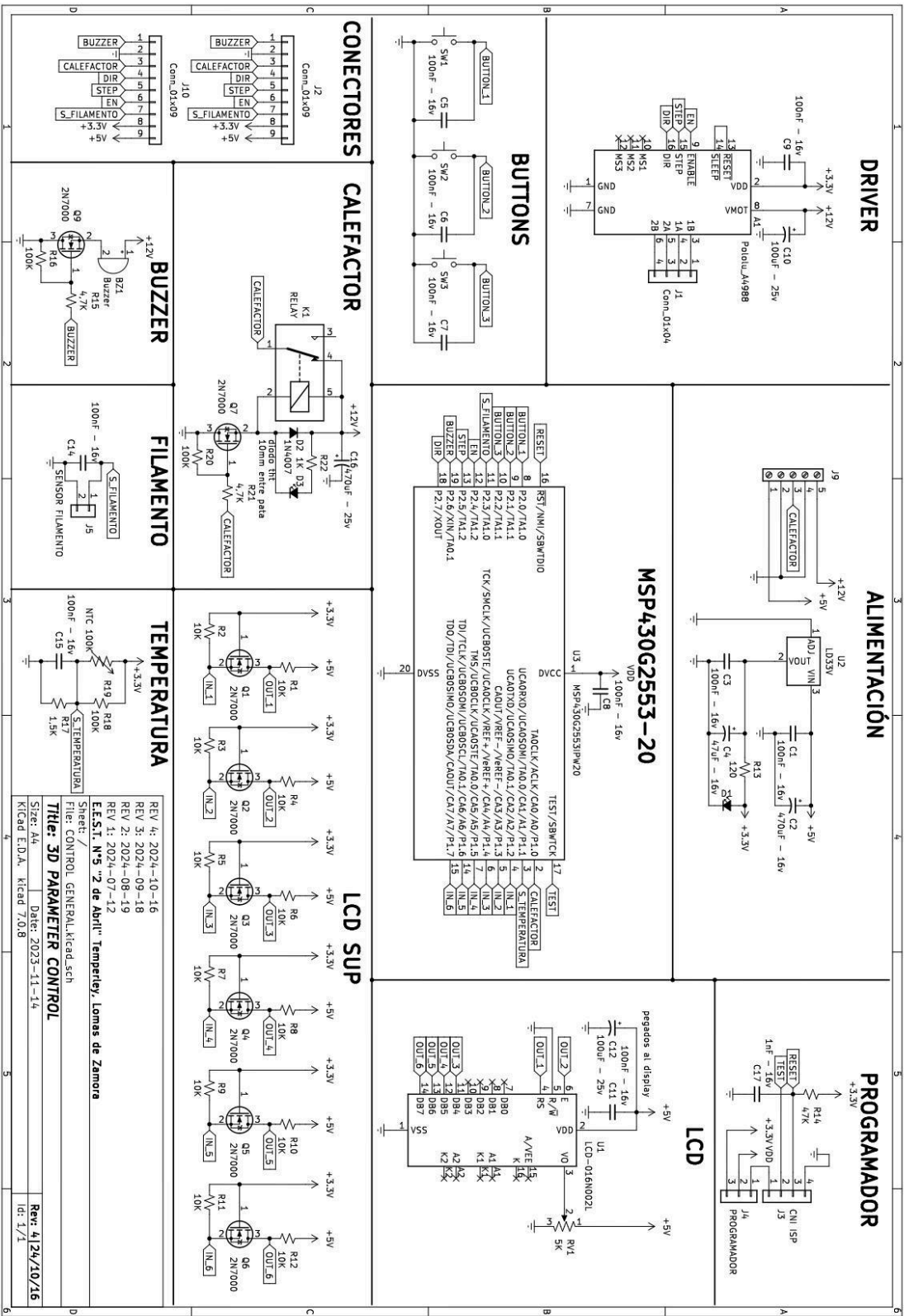


Resultados:

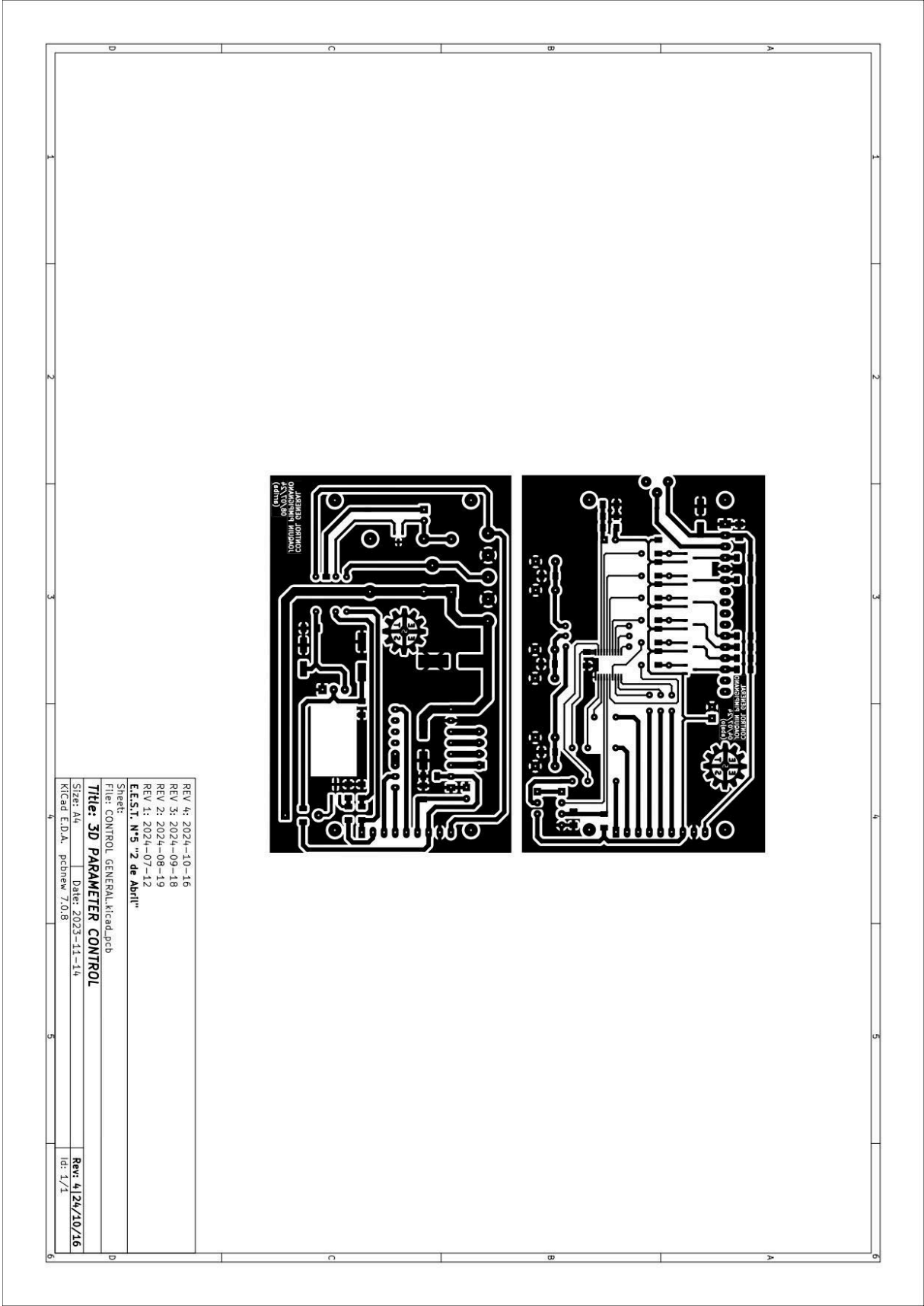
De momento estas piezas no están impresas, y las placas están sujetas mediante varillas roscadas y separadores de placas.



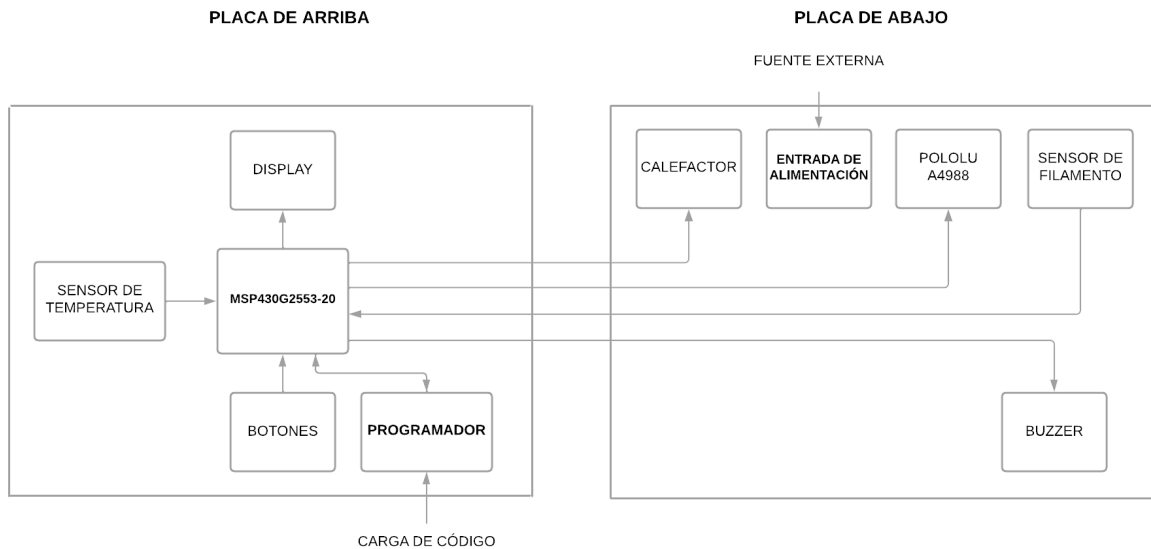
Esquemático:



Placas:



Electrónica:



Como se aprecia en el diagrama la electrónica se encuentra dividida en dos placas, en la superior vemos el microcontrolador la parte del programador, los botones y el display y el sensor de temperatura. Y en la placa de abajo se encuentra el sensor de filamento, el buzzer, el driver del motor y la salida al calefactor, y las entradas de alimentación.

El microcontrolador recibe información de los botones y sensores, envía señales analógicas al calefactor, el buzzer y el driver (a este ultimo pulsos digitales también). Por ultimo se comunica con el programador con un conector y un jumper para alternar en el modo de carga de código y el funcionamiento normal.

MSP430G2553-20:

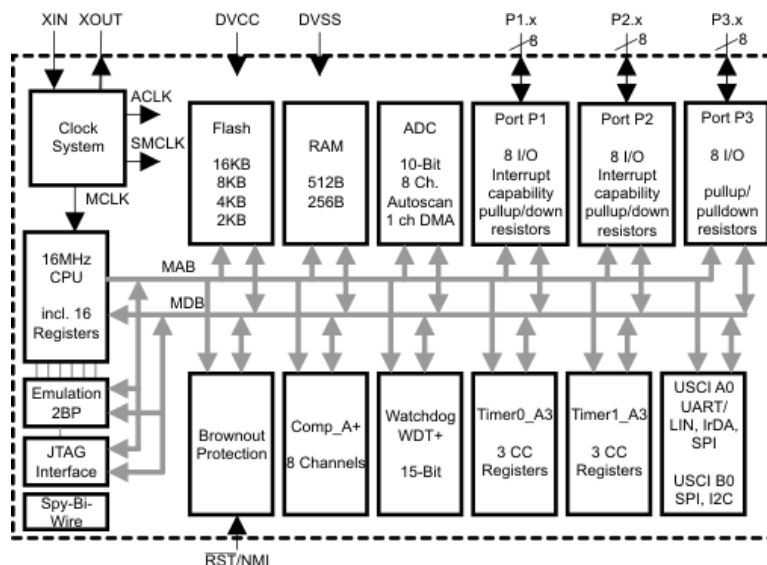
Este es un pequeño microprocesador producido por Texas Instruments muy potente y útil. Los MSP430G2x20 son microcontroladores de señal mixta de consumo ultrabajo con temporizadores de 16 bits integrados, hasta 24 pines de entrada/salida con capacidad táctil capacitiva, un comparador analógico versátil y capacidad de comunicación integrada mediante la interfaz de comunicación serial universal. Las aplicaciones típicas incluyen sistemas de sensores de bajo costo que capturan señales analógicas, las convierten en valores digitales y luego procesan los datos para su visualización o transmisión a un sistema host.

A continuación algunas de sus características:

Frecuencia (MHz)	16
Memoria non volatile (kByte)	16
RAM (kByte)	0.5
tipo ADC	10-bit SAR
Número de canales ADC	8
Número de GPIOs	24

Características	Real-time clock, Spy-bi-wire, Watchdog timer
UART	1
Número de I2Cs	1
SPI	2
Timers - 16-bit	2
Bootloader (BSL)	UART

Un buen punto a destacar es su bajo consumo, y su baja tensión de trabajo de 1,8V - 3,3V contando con un modo de reposo de super bajo consumo, su reloj interno es muy preciso y útil para hacer comparaciones de tiempo y los pines IOS tienen la función de ser activados con alteraciones capacitivas. El sistema del programador es muy confiable, al no necesitar una tensión externa para su activación es mucho más seguro.

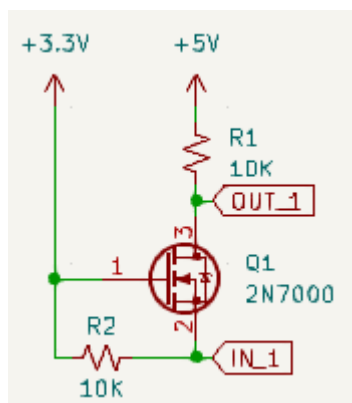


LCD:

Para que cualquier usuario pueda visualizar todas las funciones del equipo, y que pueda variar valores de manera consciente y precisa se decidió utilizar un display LCD de 16x2 dígitos.

Esta pantalla resulta ideal gracias a sus grandes aplicaciones y funciones integradas como el backlight, el contraste ajustable y todas sus entradas de datos. La única desventaja es que las entradas de datos requieren una tensión de 5V y las salidas del microcontrolador llegan como máximo a 3.3V.

Y aunque existen alternativas para solucionar este problema pero en el caso de este proyecto se utilizó un level shifter, este sistema eleva la tensión de un circuito de manera lineal, por lo que replica la señal PWM de manera que el display reciba la tensión apta.



En la fuente (Source 2) llega la señal PWM del microcontrolador y de drenaje (Drain 3) se conecta al pin del display, el cual también tiene una resistencia Pull-up de 5V. La tensión se ajusta con el pin de puerta (Gate 1) que se altera con la señal que llega del microcontrolador.

Todo esto reacciona con la pull-up suministrando correctamente al display

DRIVER STEPPER:

Para cualquier impresora 3D necesitamos un dispositivo para controlar un motor paso a paso, y para esto hay diferentes alternativas. En este caso elegimos utilizar un driver de motores llamado POLOLU A4988, debido a su fiabilidad, grandes capacidades y tecnología incorporada, además de su facilidad de programación.



Primero que nada hay que explicar cómo funciona un motor paso a paso. Este tipo de motores es muy común en la industria por su gran precisión de giro y su torque, todo esto debido a su construcción y funcionamiento.

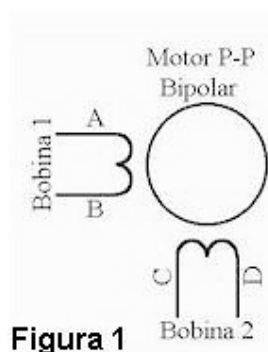


Figura 1

Como se ve en la imagen estos motores cuentan esencialmente con dos bobinas y un Rotor magnético en el centro, siendo esta la pieza que gira. Este rotor tiene polaridad magnética, de manera simple es como un imán con sus dos extremos siendo positivo y negativo, de manera que cuando se polarizan las bobinas formando un campo magnético en estas, los imanes se alinean. De esta manera podemos conseguir que energizando las bobinas el rotor gire de manera precisa. Cada uno de estos movimientos equivale a un paso, y con 200 pasos el eje da una vuelta entera sobre sí mismo, por lo tanto un paso equivale a $1,8^\circ$ ($360^\circ / 200$ pasos).

Por esto es necesario llevar un control rápido y preciso de la polarización de las bobinas y para esto hay un orden concreto.

Paso simple	PASO	A	B	C	D
	1	H	L	L	L
	2	L	H	L	L
	3	L	L	H	L
	4	L	L	L	H

Esta secuencia es la más simple ya que se va polarizando una por una las bobinas. Por esto mismo no posee mucha fuerza.

Paso doble	PASO	A	B	C	D
------------	------	---	---	---	---

	1	H	H	L	L
	2	L	H	H	L
	3	L	L	H	H
	4	H	L	L	H

Con esta configuración conseguimos más fuerza y más torque. activando dos bobinas en simultáneo aunque los pasos son más bruscos resultado de un campo magnético más intenso.

Medio paso	PASO	A	B	C	D
	1	H	L	L	L
	2	H	H	L	L
	3	L	H	L	L
	4	L	H	H	L
	5	L	L	H	L
	6	L	L	H	H
	7	L	L	L	H
	8	H	L	L	H

Se consigue realizar medio paso combinando los dos tipos de pasos, de esta manera conseguimos el doble de pasos, o sea se consigue más precisión.

Tampoco hay que olvidar que la tensión con la que trabaja nuestro motor es de 12V y su corriente alrededor de 2A. Por esto es muy complicado trabajar directamente el motor con el microcontrolador ESP32, no solo por su difícil programación sino también porque sus salidas no son lo suficientemente poderosas para alimentarlo.

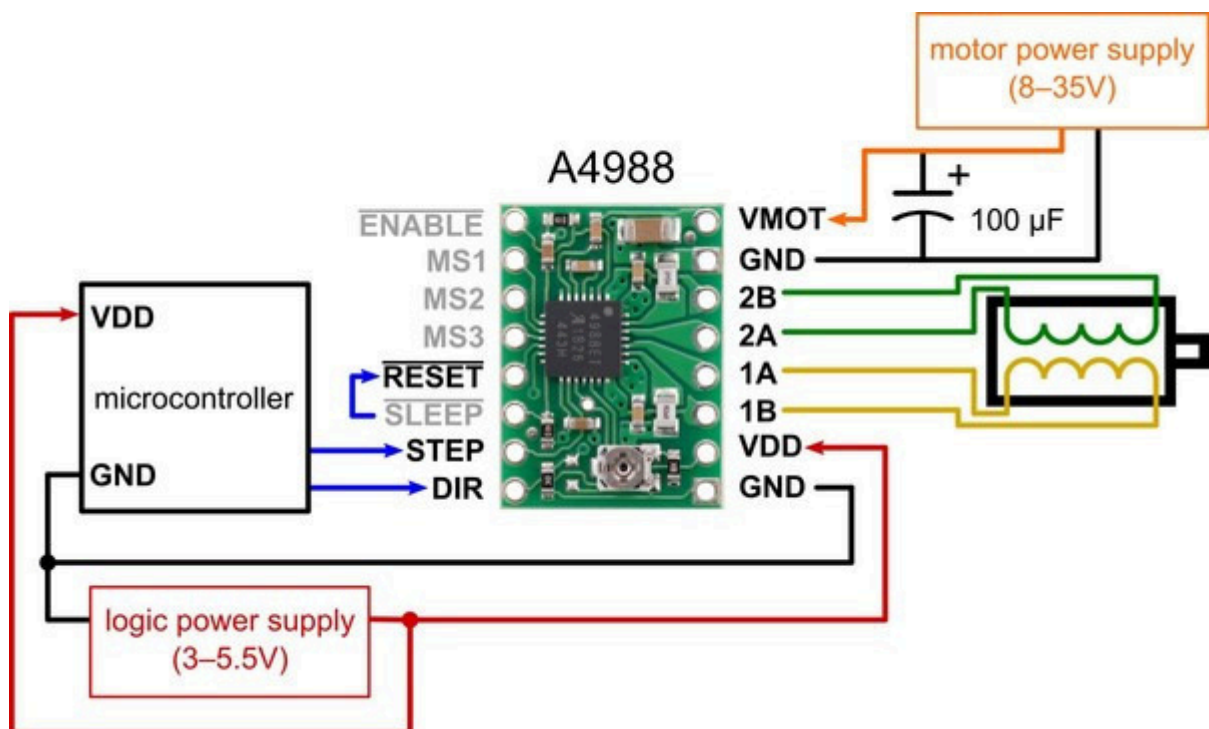
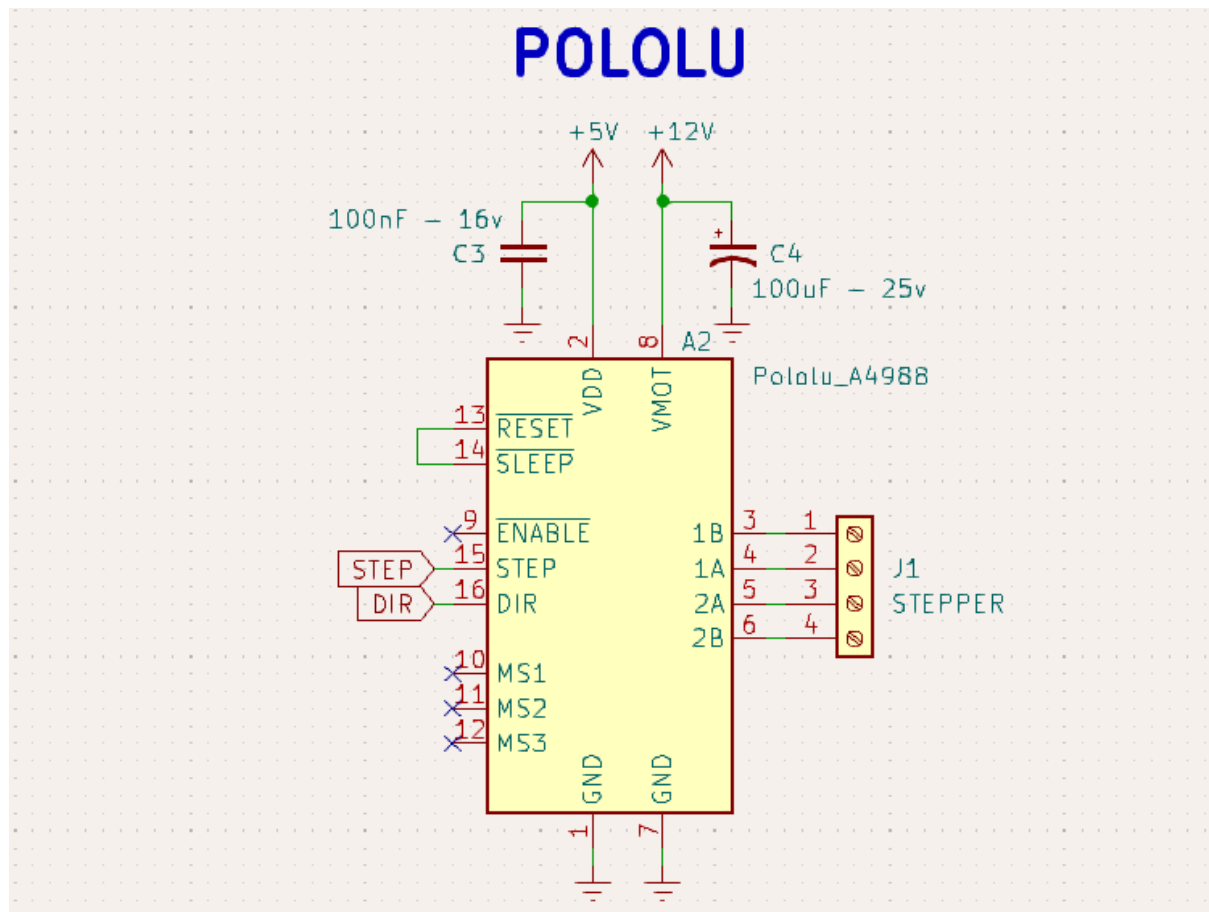
Aquí aparece el Driver POLOLU A4988, esta poderosa herramienta soluciona todos estos problemas y más. Entre sus varias características encontramos las siguientes:

- Tensión de salida de hasta 35V (En nuestro caso solo necesitamos 12V)
- Corriente de salida de 2A. Aunque normalmente no puede disipar la suficiente potencia para tener esta corriente, junto al módulo viene incluido un disipador.
- Fácil programación: Con solo dos pines podemos controlar la cantidad de pasos que da, y su sentido de giro, y con otros tres pines podemos seleccionar el tipo de pasos que da (simple, $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$). Todo esto gracias a la lógica interna del driver que con un pulso en el pin de STEP (paso) mueve un paso el motor.

Resolución		PINES		
A4988		M1	M2	M3
paso	simple	L	L	L
$\frac{1}{2}$	paso	H	L	L
$\frac{1}{4}$	paso	L	H	L
$\frac{1}{8}$	paso	H	H	L
$\frac{1}{16}$	paso	L	L	H

- Circuito de apagado térmico en caso de sobrecalentamiento por histéresis.
- Bloqueo por subtensión (UVLO), este sistema protege el motor en caso de que la tensión sea muy baja como para operarlo.
- Protección contra corrientes cruzadas, en caso de que el motor devuelva una corriente al driver.
- Reset durante funcionamiento: Cuenta con un pin especial para poder alterar el tipo de paso durante el uso, tan solo mandando un pulso.
- Regulador de tensiones, para poder activar el motor con una corriente nominal menor permitiendo alimentarlo con una tensión mayor. El limitador interrumpe la señal proporcionando una señal pulsada PWM de forma que el valor promedio de la intensidad que atraviesa la bobina es la intensidad nominal del motor. Terminando nuestro ejemplo, el limitador de tensión aplicaría el pulso durante el 15% del tiempo y mantendrá el motor apagado el 85% restante. Esta corriente puede ser controlada con el potenciómetro incorporado en la placa.

Su conexión es bastante sencilla:



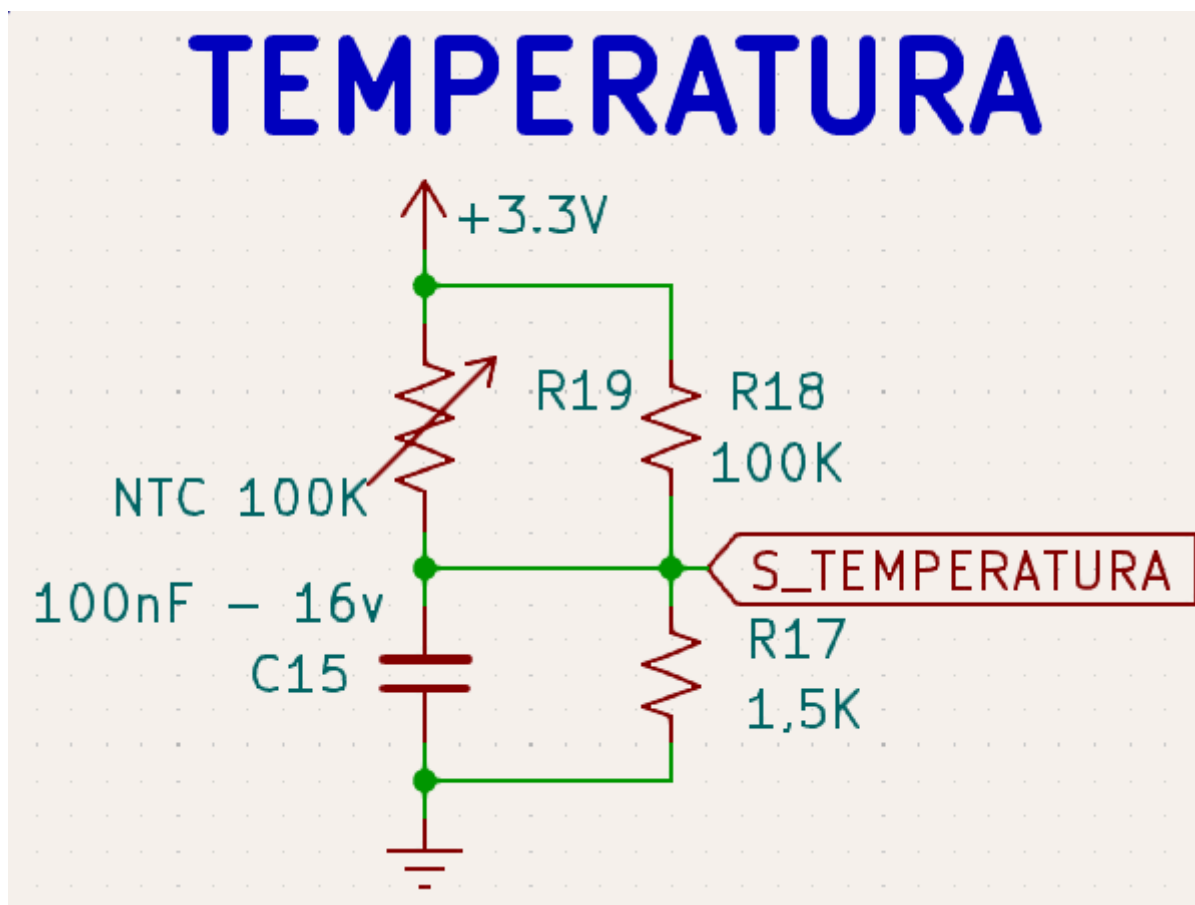
DRIVER POLOLU A4988		
PIN 1	GND	Conexión a tierra de la alimentación de la lógica.
PIN 2	VDD	Entrada de 3,3V - 5V para alimentar la lógica . (El fabricante recomienda colocar un capacitor de 100nF lo más cerca del módulo posible). Nosotros lo alimentamos con 5V.
PIN 3	1B	Salida a la bobina B1.
PIN 4	1A	Salida a la bobina A1.
PIN 5	2B	Salida a la bobina B2.
PIN 6	2A	Salida a la bobina A2.
PIN 7	GND	Conexión a tierra de la alimentación del motor.
PIN 8	VMOT	Entrada de 5V - 35V para alimentar el motor. (El fabricante recomienda colocar un capacitor electrolítico de 100uF lo más cerca del módulo posible). Nosotros lo alimentamos con 12V.
PIN 9	<u>ENABLE</u>	Activa el módulo cuando es energizado, por defecto lo está.
PIN 10	MS1	Control de pasos
PIN 11	MS2	Control de pasos
PIN 12	MS3	Control de pasos
PIN 13	<u>RESET</u>	Pin de reset del módulo. Este pin está energizado por defecto.
PIN 14	<u>SLEEP</u>	Pin de SLEEP, se suele conectar con RESET ya que este pin no está activado por defecto.
PIN 15	STEP	Pin de paso. Va conectado al ESP32
PIN 16	DIR	Cuando recibe un pulso HIGH gira para un lado, y con un LOW para el otro. Va conectado al ESP32

TEMPERATURA Y FILAMENTO:

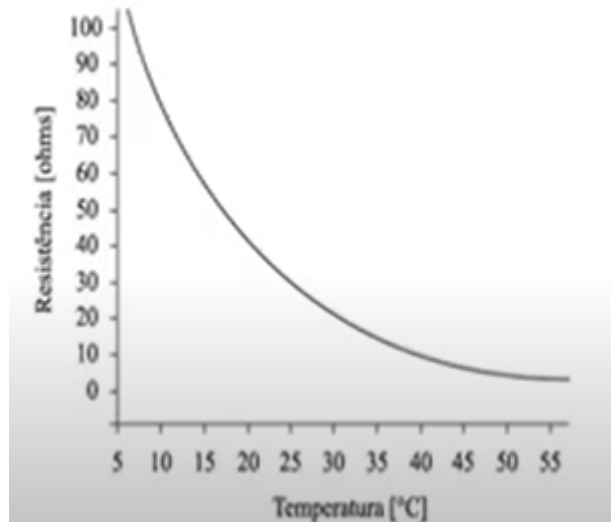
NTC 100k: Estos sensores de temperatura son resistencias semiconductoras sensibles al calor que muestran una disminución de la resistencia a medida que aumenta la temperatura. En esta aplicación, los sensores se utilizan de acuerdo con sus características de potencia cero.

Las unidades de sensor consisten en un elemento de resistencia NTC (coeficiente de temperatura negativo) de 100 K que luego se encapsula en un tubo de cobre sellado.

El termistor NTC es un semiconductor hecho de óxidos metálicos. Presenta una resistencia eléctrica que tiene un cambio muy predecible con la temperatura. La resistencia varía significativamente con la temperatura, más que en las resistencias estándar. Son extremadamente sensibles a los cambios de temperatura, muy precisos. Tienen un amplio rango de temperatura y se pueden sellar herméticamente para su uso en entornos húmedos.



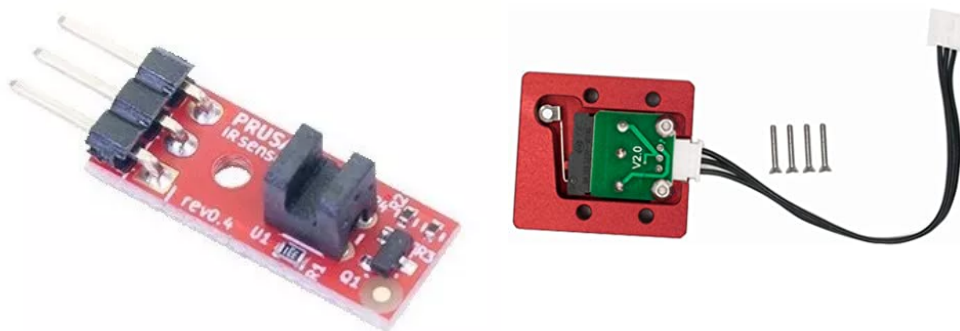
Para que el sensor de temperatura sea preciso y apto para una impresora 3D debido a que la calidad de sensado de los termistores NTC no es muy fiable.



Como podemos ver en la imagen el valor de la resistencia decrece drásticamente con un bajo cambio en bajas temperaturas, y tiene cambios de la resistencia casi imperceptibles en altas temperaturas, algo muy importante en una impresora 3D.

Al funcionar bajando la resistencia a medida que aumenta la temperatura, este tiene una temperatura mínima y máxima bastante clara, que suele ser de entre -40°C y 220°C .

SENSOR DE FILAMENTO: Otro sensor muy importante para este proyecto es el sensor de filamento, el cual se encarga de verificar constantemente que siempre haya material yendo al extrusor.



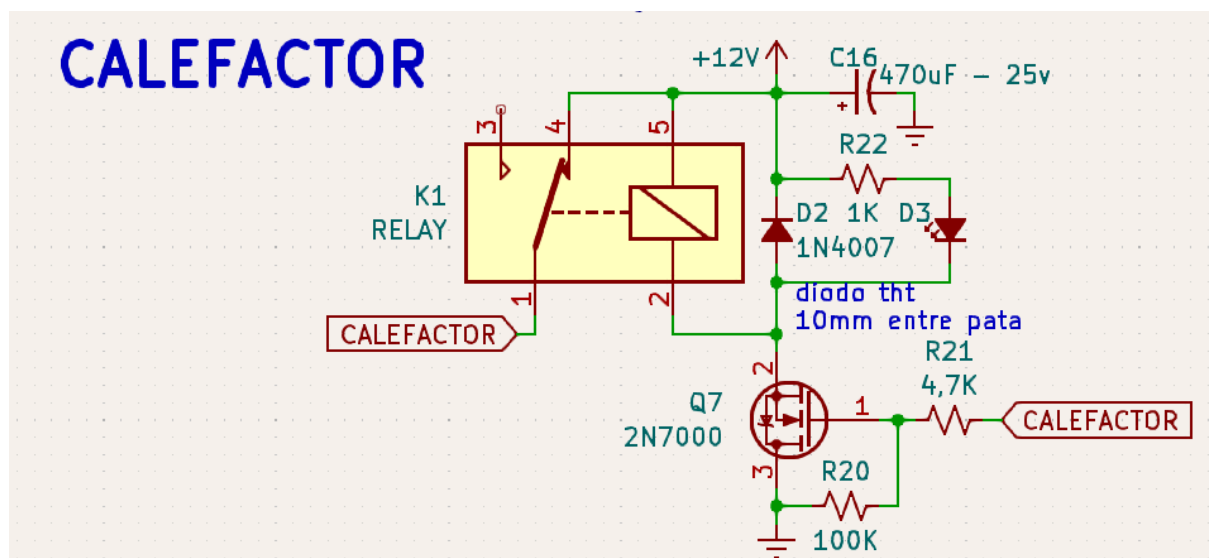
Hay muchos sistemas que cumplen esta función, como el de la primera imagen que no tiene piezas mecánicas solo un sensor capacitivo, y otros más simples como uno con un pulsador, que mientras el filamento pasa a través mantiene el pulsador cerrado.

CALEFACTOR:

Para cualquier impresora 3D es muy importante el control del extrusor y la cama, y para ambas partes necesitamos una pieza que caliente y un método eficaz que lo controle y lo resista.

En este proyecto se decidió utilizar un RELAY para controlar todo. Este método es bastante sencillo ya que con un relé y un transistor se puede controlar mediante pulsos la temperatura del cartucho calefactor.

El relé al ser activado deja pasar la corriente a la carga, el cartucho calefactor, el cual es sensado por el sensor de temperatura, cuando este llega a la temperatura deseada corta la alimentación y cuando baja de la misma vuelve a activar, de esta manera oscilando para mantenerlo en un punto exacto.



VARIOS:

Buzzer: El buzzer es un electroimán que al energizarse mueve una pequeña pieza de metal produciendo las vibraciones. Este dispositivo es controlado con una señal PWM y un transistor.

Botones: Los botones para esta placa a diferencia de los otros microprocesadores que usan pull-ups o pull-down son con capacitores, gracias a una función integrada del MSP430.

Conectores: Para transmitir la información de los sensores y los controladores de una placa a otra, y la alimentación desde la placa de abajo a la de arriba utilizamos dos pineras macho conectadas con cables arduino.

Alimentación: Tanto 5V cómo 12V vienen de una fuente externa pero para conseguir los 3,3V utilizamos el regulador LD33V con el circuito de regulación indicado por el fabricante y un led para indicar su correcto funcionamiento.

Programador: Para cargarle el código al microprocesador se utiliza un programador externo USB con tecnología UART, con un pequeño jumper se alterna en el modo para cargar el código y la alimentación normal para el funcionamiento. Además de esto no se necesita nada más, el MSP430 tiene un método de programación muy sencilla.

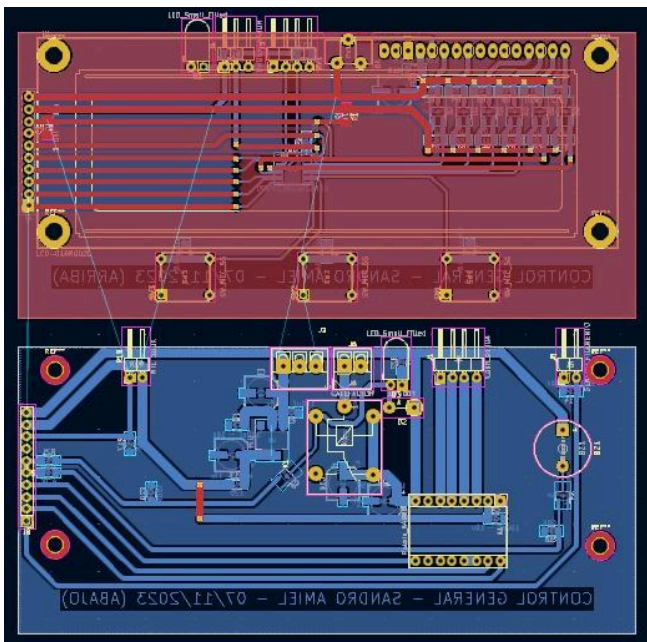
Proceso de fabricación:

En este apartado se comentan algunos puntos que fueron tenidos en cuenta al momento de la fabricación, tanto en lo físico como en el diseño.

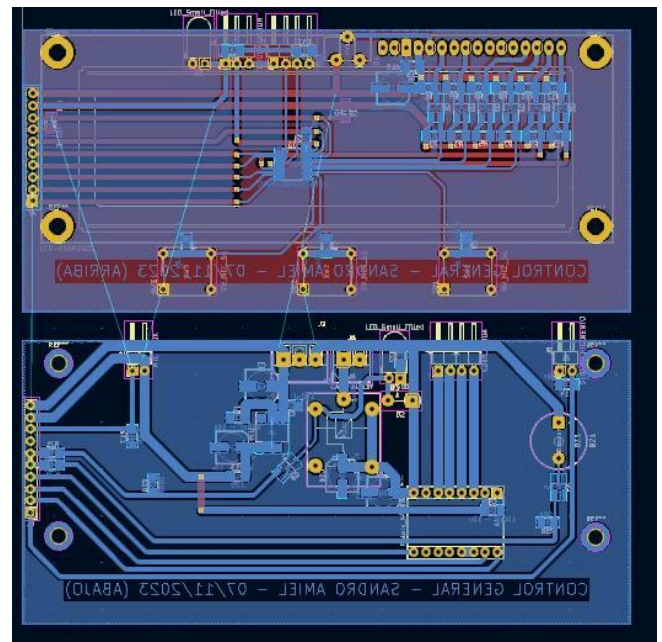
El diseño pasó varias versiones antes de llegar al primer diseño exitoso, fueron en total 4 grandes revisiones, que se detallan a continuación.

REVISIÓN 1:

Una de las primeras cuestiones por las que se realizaron cambios a la primera versión fueron las distribuciones, la mayoría de los fabricantes recomiendan que los capacitores y todos los componentes anexos estén lo más cerca posible del propio módulo. Además los tamaños de las pistas, el espacio entre las mismas y su largo era innecesario o muy poco.



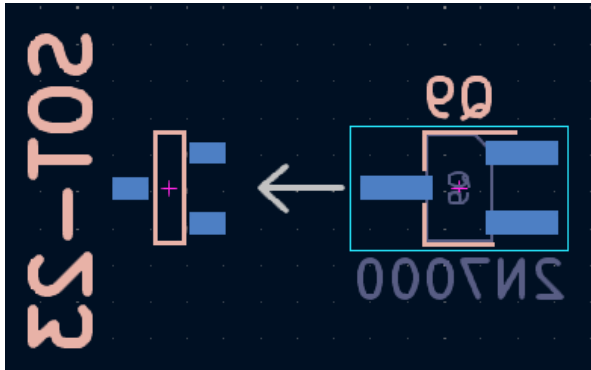
FRONT FACE



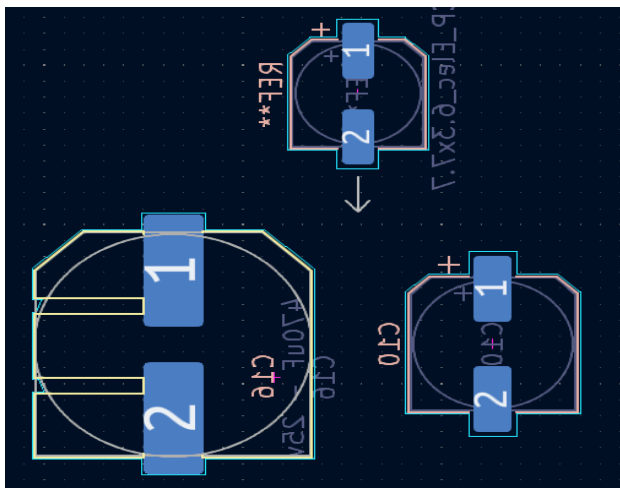
BACK FACE

REVISIÓN 2:

A esta altura ya se acercaba la fabricación por lo cual fue necesario ajustar la huella de la mayoría de los componentes, ya que al momento de la primera versión no contaba con los mismos.



En este caso fue necesario reemplazar la huella de todos los transistores por una más pequeña.

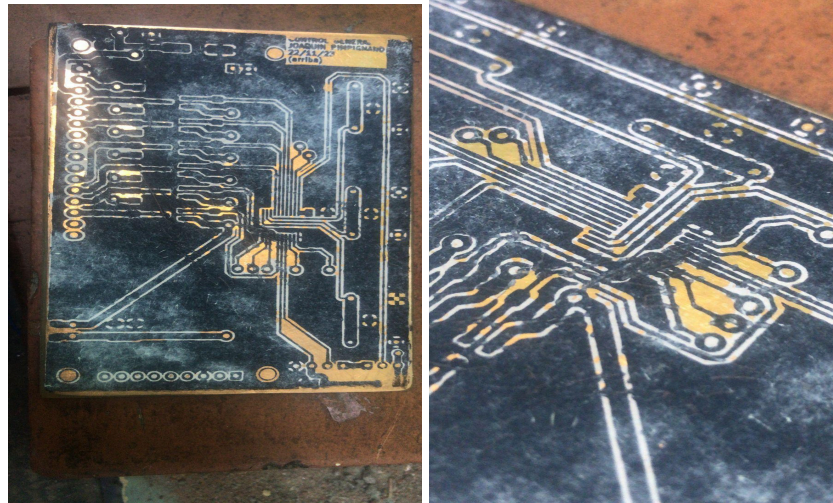


Algunos capacitores electrolíticos también cambiaron, incluso tuve que diseñar la huella del capacitor más grande, ya que no se encontraba en los archivos del programa.

Aunque sin duda el cambio más significativo fue el del DISPLAY que cambió por uno más pequeño y con otra distribución de pines lo que achicó significativamente la placa.

REVISIÓN 3:

En este punto se fabricó la primera placa física, la cual salió mal



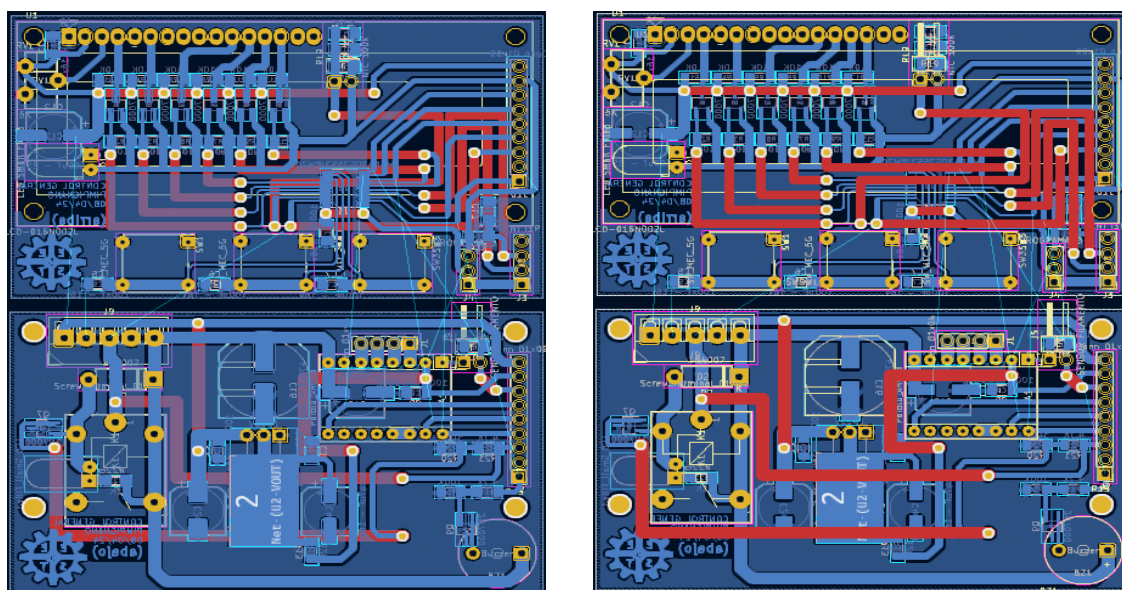
Como se ve en la imagen las pistas más pequeñas que se conectan al microcontrolador salían mal, esto debido al método salvaje de fabricación mediante transferencia por calor. Por eso se decidió que la placa sea simple para poder realizar el método de transferencia fotográfica.

REVISIÓN 4:

Esta es simplemente una primera versión de la placa simple faz.¹

FRONT FACE

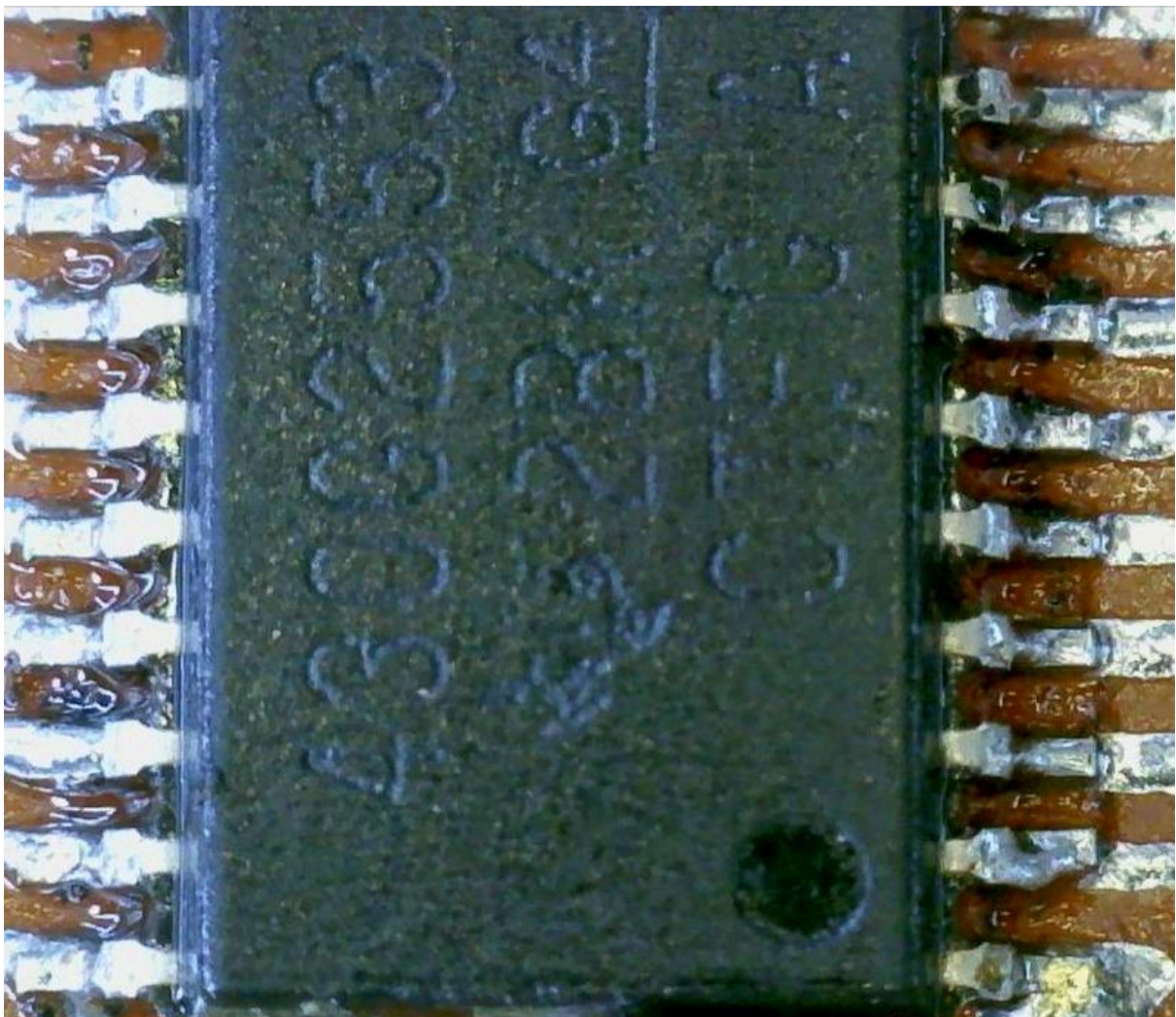
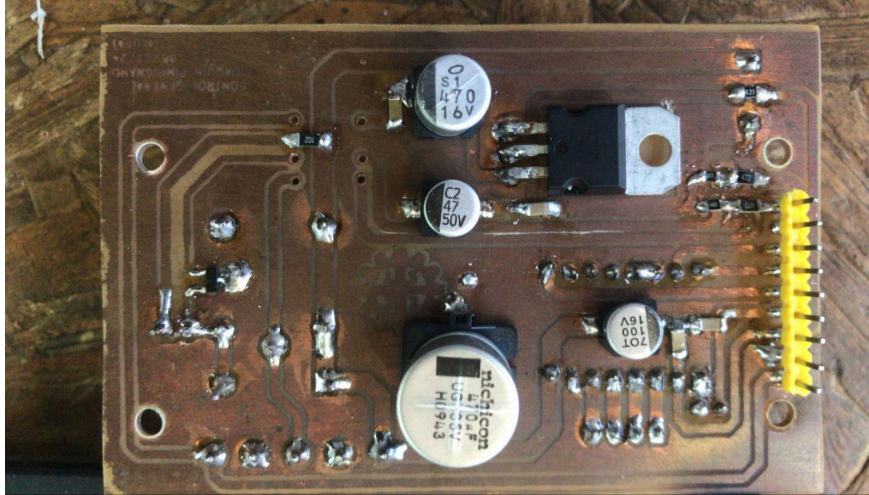
BACK FACE



¹ Las pistas en rojo representan los puentes, como se ven eran muchos, muy largos y de forma irregular, algo bastante malo.

REVISIÓN 4.1:

Luego de pasar por todas las anteriores versiones se llegó a la actual, que se puede consultar en el apartado de ESQUEMÁTICO y PLACAS. La placa de abajo que cuenta con la mayoría de controles y procesos si se fabricó con método de transferencia por calor al ser considerablemente más sencilla.



BOM (Bill of Materials):

COMPONENTE	CANTIDAD	PRECIO DE COMPRA	PRECIO X UNIDAD	PÁGINA	LINK	DIFERENCIA
MSP430G2553IPW20	1	2166,55	2166,55	OCTOPART	CLICK	784,27
LCD-016N002L	1	1176,41	1176,41	ALIBABA	CLICK	686,24
SOT23	8	19,61	156,88	ALIBABA	CLICK	10,57
LD33 TO 220	1	19,61	19,61	ALIBABA	CLICK	9,8
Resistencia 4,7K smd	2	49,02	98,06	ALIBABA	CLICK	40,46
Resistencia 47K smd	1	49,02	49,02	ALIBABA	CLICK	40,46
Resistencia 100K smd	3	49,02	147,06	ALIBABA	CLICK	40,46
Resistencia 10K smd	11	49,02	539,22	ALIBABA	CLICK	40,46
Botón	3	39,21	117,63	ALIBABA	CLICK	22,05
Capacitor 100nF smd	11	88,23	882,3	ALIBABA	CLICK	49,02
Pinera Hembra	32	98,03	78,42	ALIBABA	CLICK	9,8
Pinera Macho	33	98,03	86,98	ALIBABA	CLICK	9,8
Potenciómetro ACP	1	68,62	68,62	ALIBABA	CLICK	58,8
LED 5mm	1	2930,99	586,19	ALIBABA	CLICK	980,27
Relay	1	137,24	137,24	ALIBABA	CLICK	39,22
Terminal Block	5	88,23	441,15	ALIBABA	CLICK	58,8
POLOLU A4988	1	970,46	970,46	ALIBABA	CLICK	872,43
Diodo 1N4007	1	98,03	98,03	ALIBABA	CLICK	88,23
Capacitor 47uF - 16v	1	294,08	294,08	ALIBABA	CLICK	284,28
Capacitor 470uF - 16v	1	294,08	294,08	ALIBABA	CLICK	284,28
Capacitor 100uF - 16v	2	294,08	588,16	ALIBABA	CLICK	284,28
TOTAL	X	X	8996,15	X	X	4007,74

- El largo total de los puentes es equivalente a 205,8 mm que es un punto muy importante a tener en cuenta debido al alto valor del cobre.
- La cantidad de pines se contó uno por uno a pesar de que comúnmente se venden en tiras de 40, esto para hacer el precio final más exacto.
- Al costado derecho se aprecia una columna color verde la cual indica la diferencia a favor que obtenemos al realizar una compra de más de 1000 piezas al vendedor. Esta diferencia es lo que le tenemos que descontar al PRECIO DE COMPRA para obtener el nuevo PRECIO POR UNIDAD. Con el TOTAL podemos ver la gran diferencia a favor que obtenemos comprando como mayorista.

- Al costado izquierdo del LINK podemos ver la página de donde se realiza la compra.

A continuación voy a detallar el precio total que nos costaría fabricar mil placas:

- Para calcular el coste de envío tendré como referencia el precio de ALIBABA ya que casi la totalidad de mis componentes provienen de allí. Este tiene un coste de 10\$(USD) x 1kg. Pesé los componentes totales de las placas para saber el coste de envío, lo que nos da:

$$92g \cdot 1.000 = 92.000g = 92kg$$

$$92kg \cdot 10USD = 920USD$$

- Este es el coste de envío para la fabricación de 1000 placas, al que hay que sumarle el costo en sí de los materiales más los impuestos y la fabricación de la placa
- Para calcular el coste de los materiales es tan simple como tomar el TOTAL, restarle el TOTAL de la DIFERENCIA y multiplicarlo por mil:

$$(8.996,15 - 4.007,74) \cdot 1.000 = 4.988,41 \cdot 1.000 = 4.988.410$$


El coste de mandar a fabricar las placas más su envío es el siguiente:

PCB Build time



[Price Comparison Matrix](#)

Build Time	Total
✓ 7-8 days	\$1157.00
✓ New User Discount	-\$25

Shipping costs



UNITED STATES OF AMERICA

3-5 days , Weight: 35.820 kg



✓ New User Discount

\$392.00

CHN Time Zone(GMT+8): 2024/10/18 05:38:41

Payment before 2024/10/18 17:00
(GMT+8 Only PCB)

Delivery time
2024/10/25

 --> 

Estimation
2024/10/30

PCB Cost	\$ 1157.00
Shipping	\$392.00
Total Amount	\$1549.00
	Currency

Este coste es el sugerido por PCBWAY una reconocida página que fabrica placas a pedido, ellos además se encargan de montarla por precio extra. El coste es para 2000 placas, ya que necesitamos la placa de arriba y de abajo.

- Ahora esto hay que pasarlo a dólares para calcular los impuestos de importación

$$4.988.410 \div 1.183,85 = 4.213,71$$

$$4.213,71 + 1.549 + 920 = 6.682,71$$

Que si lo pasamos por una calculadora de impuestos nos da:

$$AR\$11.846.914,49$$

Que nos da un coste por la electrónica de AR\$11.846,91.

Conclusión:

Creo que es un proyecto con mucho alcance y escalabilidad, su funcionalidad es bastante amplia, y da pie al diseño y fabricación de diferentes sistemas acoplados a impresoras 3D y similares como las CNC, me acercó a estos sistemas, y abre las puertas a una futura tecnología que todavía está en desarrollo y probablemente revolucione la industria como ya se ve. Tiene todo lo que un buen proyecto de electrónica debe tener con una complejidad alta.

Entre las posibles mejoras estaría cambiar de lugar la pinera que conecta las dos placas, y reemplazarla directamente por una tira de cables más pequeña, esto podría ayudar a reducir el tamaño y costes. En el primer diseño se decidió dejar así para poder hacer las debidas pruebas y maniobrar más fácil.

Otra mejora podría ser cambiar el driver de motores paso a paso POLOLU A4988 por su hermano mayor el DRV8825 que tiene un coste ligeramente mayor pero lo compensa con una capacidad mayor manejando tensiones más altas y mayores corrientes.

El último punto que cambiaría sería el sensor de temperatura por una termocupla tipo K que mide en un rango mayor de temperatura, hasta los 1000°C por lo que operamos lejos de sus límites con una mayor precisión inclusive.

Aunque funciona el método actual por una diferencia de precio muy pequeña se le puede agregar un módulo del LCD para suplementar la tensión que requiere, con esto simplificamos la programación, reducimos la cantidad de pines que necesitamos y también achicamos el tamaño de la placa.

Además al mandar a fabricar se podría pensar en una placa doble faz mucho más compacta y sencilla con una distribución más óptima de los componentes evitando también el uso de puentes, para comunicar las dos placas se podría utilizar un cable plana más compacto y barato. También cambiar las pineras y borneras por conectores estándar para volverlo más comercial.

Bibliografía:

MSP430G2553-20 REFERENCIA 1:

https://www.ti.com/lit/ds/symlink/msp430g2553.pdf?ts=1730240572351&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252Fes-mx%252FMSP430G2553

LCD-016N002L REFERENCIA 1:

<https://www.es.co.th/schematic/pdf/LCD-016N002L.pdf>

MSP430G2553-20 REFERENCIA 2:

<https://www.ti.com/product/es-mx/MSP430G2553>

POLOLU A4988 REFERENCIA 1:

[A4988 datasheet\(1/19 Pages\) ALLEGRO | DMOS Microstepping Driver with Translator and Overcurrent Protection](#)

SENSOR DE TEMPERATURA REFERENCIA 1:

<https://www.alldatasheet.com/html-pdf/1950238/DANFOSS/NTC100K/691/2/NTC100K.html>

SENSOR DE TEMPERATURA REFERENCIA 2:

<https://www.alldatasheet.com/html-pdf/1132216/ETC2/NTC100D10/113/1/NTC100D10.html>

BUZZER REFERENCIA 1:

<https://osakaelectronicsltda.com/blog/biblioteca/que-es-un-buzzer#:~:text=Funcionan%20de%20manera%20similar%20a.requieren%20sonidos%20con%20baja%20frecuencia.>

LD33V REFERENCIA 1:

<https://www.alldatasheet.com/html-pdf/94497/STMICROELECTRONICS/LD33CV/1620/1/LD33CV.html>

CALCULADORA DE COSTE DE ENVIO:

<https://es.sino-shipping.com/flete-china-argentina/#:~:text=Actualizaci%C3%B3n%20de%20Env%C3%ADos%20Julio%202024%3A%20De%20China%20a%20Argentina&text=El%20env%C3%ADo%20regular%20cuesta%20%2410.tr%C3%A1nsito%20de%206%2D8%20d%C3%ADas.>

FABRICADOR DE PCB:

<https://www.pcbgogo.com/pcb-fabrication-quote.html>

CALCULADORA DE IMPUESTOS DE IMPORTACIÓN:

<https://www.impuestito.org/>