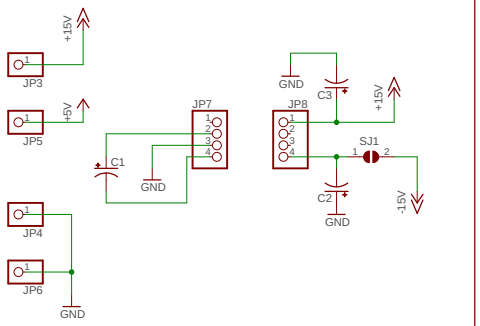
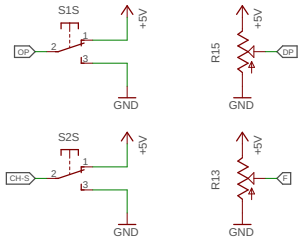
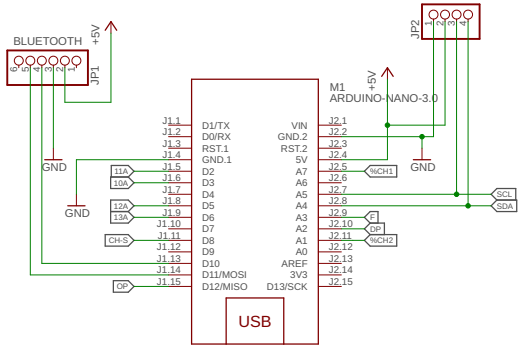
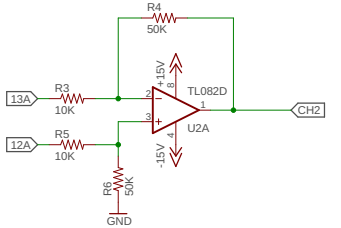
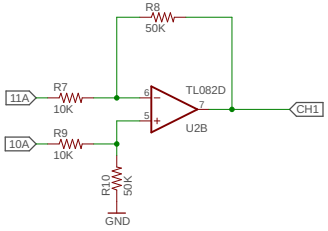
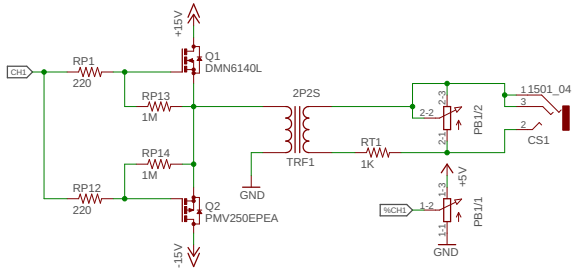


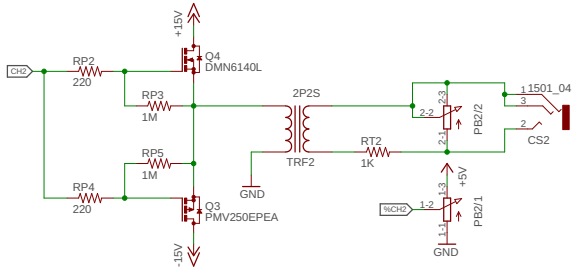




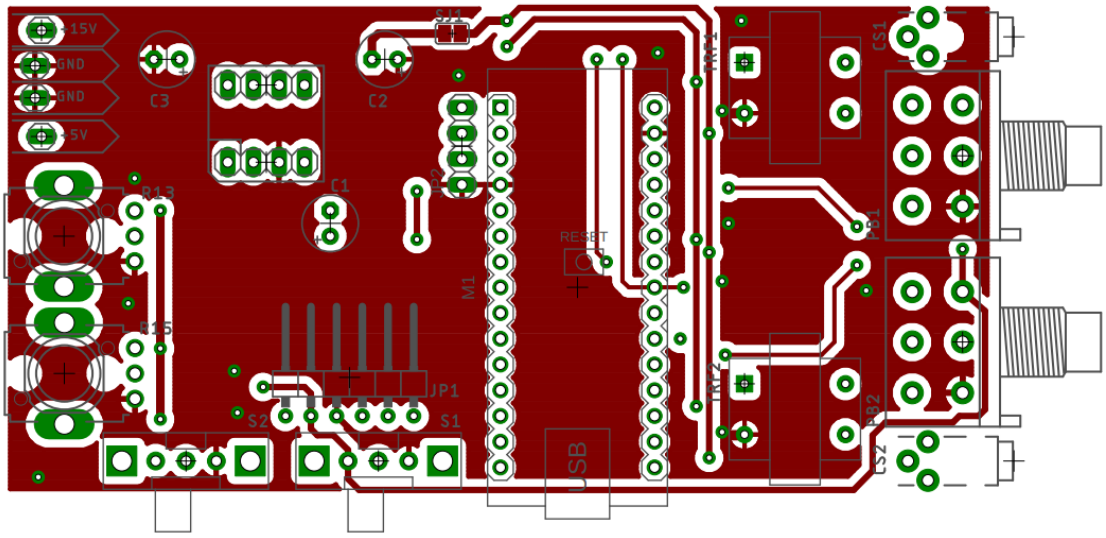


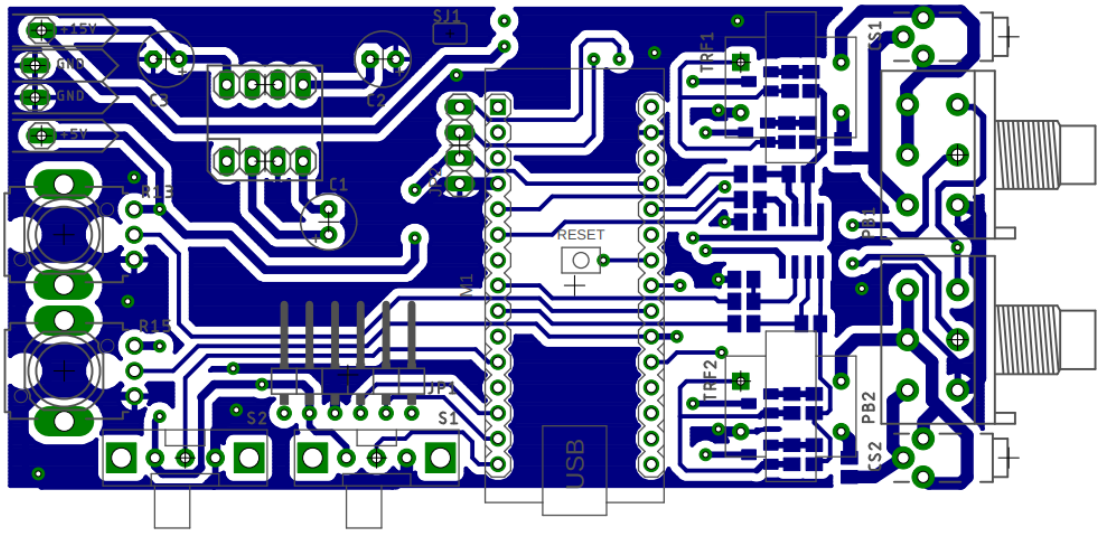






TITLE: Tarjeta_Control_v1.2	
Document Number:	REV:
Date: 17/06/2024 8:04 p. m.	Sheet: 1/1


15V
GND
GND
5V
C3
C2
C1
R13
R15
JP1
S2
S1
RESET
M1
USB
TRF1
TRF2
PB1
PB2
CS1
CS2


15V
GND
GND
5V
C3
C2
C1
R13
R15
JP1
S2
S1
RESET
M1
USB
TRF1
TRF2
PB1
PB2
CS1
CS2

```
//Librerías
#include <Wire.h>
#include <SoftwareSerial.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define DEBUGOUTPUT 0

//Variables Globales

const int F = A3;             // Pines Analógicos
const int DP = A2;
const int OLED_SDA = A4;
const int OLED_SCK = A5;
const int VOLTAJE_CH1 = A1;
const int VOLTAJE_CH2 = A7;

const int MODO = 12;           // Pines Digitales
const int CANAL = 8;
const int CH1_POS = 3;
const int CH1_NEG = 2;
const int CH2_POS = 5;
const int CH2_NEG = 6;
const int TX_BLUETOOTH = 11;
const int RX_BLUETOOTH = 10;

const int ANCHO_OLED = 128;   // Dimensiones de la pantalla
const int ALTO_OLED = 64;

//Variables
int FTon2[2];

int F2;
int TON2;

float voltajeCh1 = 0;
float voltajeCh2 = 0;

int  generatedChecksum = 0;
byte checksum = 0;
byte payloadLength = 0;
byte payloadData[32] = {0};
byte signalquality = 0;
byte ATENCION = 0;
byte MEDITACION = 0;

//Objetos
Adafruit_SSD1306 display(ANCHO_OLED, ALTO_OLED, &Wire, 4);
SoftwareSerial SerialBT(RX_BLUETOOTH, TX_BLUETOOTH);

void setup() {
```

```
  //Pantalla
  Wire.begin();
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);

  //Pines
  pinMode(CH1_POS, OUTPUT);
  pinMode(CH1_NEG, OUTPUT);
  pinMode(CH2_POS, OUTPUT);
  pinMode(CH2_NEG, OUTPUT);
  pinMode(MODO, INPUT);
  pinMode(CANAL, INPUT);

  //Inicializar pines en estado bajo
  digitalWrite(CH1_POS, LOW);
  digitalWrite(CH1_NEG, LOW);
  digitalWrite(CH2_POS, LOW);
  digitalWrite(CH2_NEG, LOW);

  //
  Serial.begin(57600);
  //Serial.println("listo");
  SerialBT.begin(57600);

}

void loop() {
  if (digitalRead(MODO) == HIGH) {
    calibracion(FTon2);}
  else {
    if(FTon2[0] && FTon2[1] != 0){
      while(digitalRead(MODO) == LOW){
        operacion(FTon2[0],FTon2[1]);}
      }
    else{
      mensaje();
      }
  }
}

// FUNCIONES
void calibracion(int FTon[2]){

  int F1 = map(analogRead(F), 0, 1023, 20, 45);
  int TON1 = map(analogRead(DP), 0, 1023, 100, 300);

  FTon[0] = (F1 % 5 == 0) ? F1 : FTon[0];
  FTon[1] = (TON1 % 5 == 0) ? TON1 : FTon[1];

  voltajeCh1 = (analogRead(VOLTAJE_CH1) / 1023.0) * 100;
  voltajeCh2 = (analogRead(VOLTAJE_CH2) / 1023.0) * 100;
```

```
  actualizar_pantalla(int(voltajeCh1), int(voltajeCh2), FTon[0], FTon[1]);

  if (digitalRead(CANAL) == HIGH) {
    generar_pulso(CH1_POS, CH1_NEG, FTon[0], FTon[1]);
  } else {
    generar_pulso(CH2_POS, CH2_NEG, FTon[0], FTon[1]);
  }
}

void operacion(int Fr, int Ton){
  //buscar los 2 bytes de sincronización del paquete de datos
  if (ReadOneByte() == 0xAA) //leer el primero
  {
    if (ReadOneByte() == 0xAA) //leer el segundo
    {
      payloadLength = ReadOneByte();//leer el tercer byte, que indica la longitud del paquete
de datos
      if (payloadLength == 0x20) //continuar leyendo solo si se recibe un paquete grande,
descartar paquetes pequeños
      {
        generatedChecksum = 0; //poner a cero las variables de verificación
        for (int i = 0; i < payloadLength; i++) //leer 32 bytes de forma continua
        {
          payloadData[i] = ReadOneByte();//leer los datos del paquete de datos de longitud
especificada
          generatedChecksum += payloadData[i];//calcular la suma acumulativa de los datos
        }
        checksum = ReadOneByte();//leer el byte de verificación
        //verificación
        generatedChecksum = (~generatedChecksum) & 0xff;
        //comparar el byte de verificación
        if (checksum == generatedChecksum) //la recepción de datos es correcta, continuar con
el procesamiento
        {
          //asignar datos
          ATENCION = payloadData[29];//valor de la atención
          MEDITACION = payloadData[31];//valor de la meditación
          #if !DEBUGOUTPUT
          //imprimir el valor de la atención
          //Serial.print("Attation: ");
          Serial.println(ATENCION, DEC);
          //valor de la meditación
          //Serial.print("Meditation: ");
          Serial.println(MEDITACION, DEC);

          if(Fr && Ton != 0){
            voltajeCh1 = (analogRead(VOLTAJE_CH1) / 1023.0) * 100;
            voltajeCh2 = (analogRead(VOLTAJE_CH2) / 1023.0) * 100;
            actualizar_pantalla(int(voltajeCh1), int(voltajeCh2), Fr, Ton);
```

```
            int ciclos = 2;
            if (ATENCION > MEDITACION and ATENCION > 50) {//LED ROJO
              //int cont=0;
              //while(cont<ciclos*Fr){
                generar_pulso(CH1_POS, CH1_NEG, Fr, Ton);
                //cont=cont+1;
              //}
            }
            if (MEDITACION > ATENCION and MEDITACION > 50) {////LED VERDE
              //int cont=0;
              //while(cont<ciclos*Fr){
              generar_pulso(CH2_POS, CH2_NEG, Fr, Ton);
              //cont=cont+1;
              //}
            }
            if (MEDITACION < 50 and ATENCION < 50) {
              apagar_canales();
              }
            }
          else{
            mensaje();
            }

          #endif
        }
      }
    }
  }
}

void actualizar_pantalla(int r1, int r2, int F2, int TON2 ){
  display.clearDisplay();
  display.setTextColor(WHITE);
  //
  display.setCursor(10,1);
  display.setTextSize(1);
  display.print("MODO:");
  //
  display.setCursor(41,1);
  display.setTextSize(1);
  if(digitalRead(MODO) == HIGH){
    display.print("Calibracion");
    }
  else{
    display.print("Operacion");
    }
  //
  display.setCursor(1,25);
  display.setTextSize(1);
  display.print("Duracion fase:");
```

```
  display.setCursor(90,25);
  display.setTextSize(1);
  display.print(TON2);
  //
  display.setCursor(1,35);
  display.setTextSize(1);
  display.print("Frecuencia:");
  display.setCursor(70,35);
  display.setTextSize(1);
  display.print(F2);
  //
  display.setCursor(1,45);
  display.setTextSize(1);
  display.print("Voltaje Ch1:     %");
  display.setCursor(80,45);
  display.setTextSize(1);
  display.print(r1);
  //
  display.setCursor(1,55);
  display.setTextSize(1);
  display.print("Voltaje Ch2:     %");
  display.setCursor(80,55);
  display.setTextSize(1);
  display.print(r2);
  //
  display.display();
  }

byte ReadOneByte(){
  int ByteRead;
  while (!SerialBT.available());
  ByteRead = SerialBT.read();
  return ByteRead;
}

void apagar_canales(){
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
}

void mensaje(){
  display.clearDisplay();
  display.setTextColor(WHITE);
  //
  display.setCursor(10,1);
  display.setTextSize(1);
  display.print("Ingresar a modo de calibracion primero");
  display.display();
```

```
  }

void generar_pulso(int pos_pin, int neg_pin, int F2, int TON2) {
  delay((1.0 / F2 * 1000) - 2 * TON2 / 1000);
  digitalWrite(pos_pin, HIGH);
  delayMicroseconds(TON2);
  digitalWrite(pos_pin, LOW);
  digitalWrite(neg_pin, HIGH);
  delayMicroseconds(TON2);
  digitalWrite(neg_pin, LOW);
}
```