

# 2024GEARS\_Soft\_Robotics Manual

Author: Xizhe Hao, Jianzhi Tang, Zhiqi Guo

---

## 0 Web Reference Links

Manual Link(Include videos)

GitHub Link

## 1. Arduino Nano 33 IoT

### 1.1 Hardware

#### 1.1.1 Circuit

#### 1.1.2 Datasheets

#### 1.1.3 Reference Link

### 1.2 Software

#### 1.2.1 WiFi

#### 1.2.2 Bluetooth

## 2. Tinycircuit

### 2.1 Hardware

### 2.2 Software

## 3. Heater

### 3.1 Test result

### 3.2 Suggestions

## 0 Web Reference Links

### Manual Link(Include videos)

[2024GEARS\\_Soft\\_Robotics Manual](#)

### GitHub Link

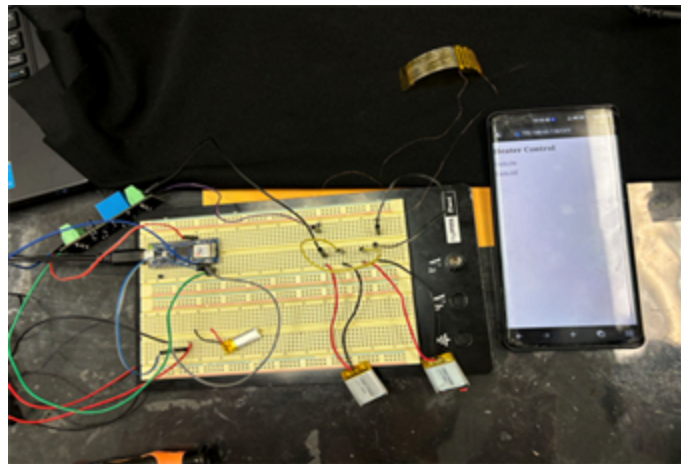
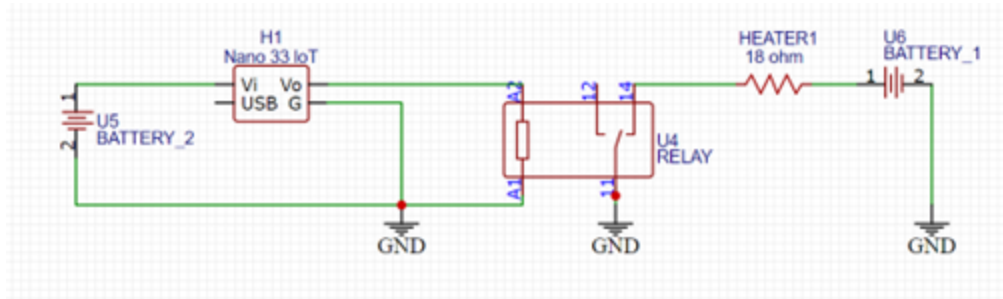
[https://github.com/Xizhe-Hao/Soft-Robotics\\_control](https://github.com/Xizhe-Hao/Soft-Robotics_control)

## 1. Arduino Nano 33 IoT

# 1.1 Hardware

## 1.1.1 Circuit

1- Arduino Nano 33 IoT Circuit (Use Relay As A Switch)

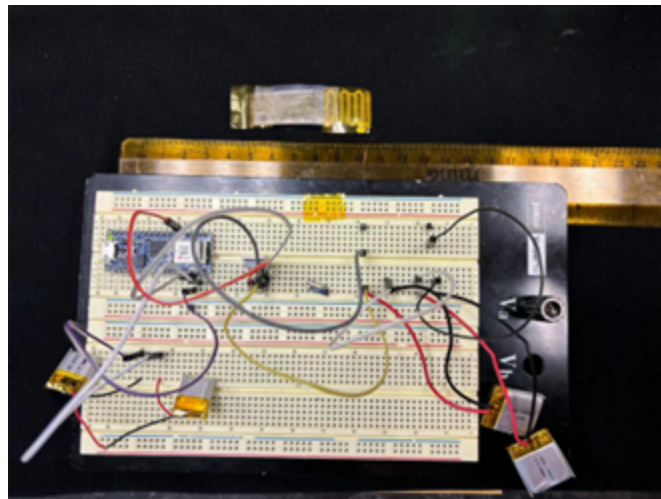
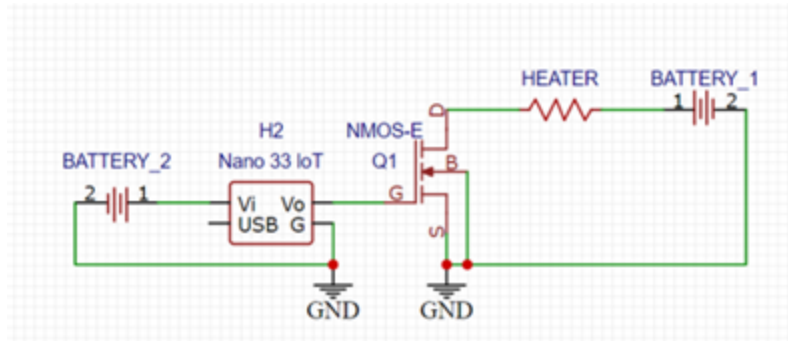


	BATTERY1-HEATER	BATTERY2-Nano33	Vo-Relay
Voltage(V)	7.4V (3.7V 150mAh*2)	9V	3.3V
Current(A)	~0.4A	~0.1A	~0.01A

- The relay needs to consume a lot of energy in the on-off moment, and the small battery will dead in the instant it is switched on, so BATTERY2 chooses a larger capacity battery with 9V voltage for power supply.
- The Bluetooth function requires a large operating voltage (>7V), which is greater than the operating voltage of power light.

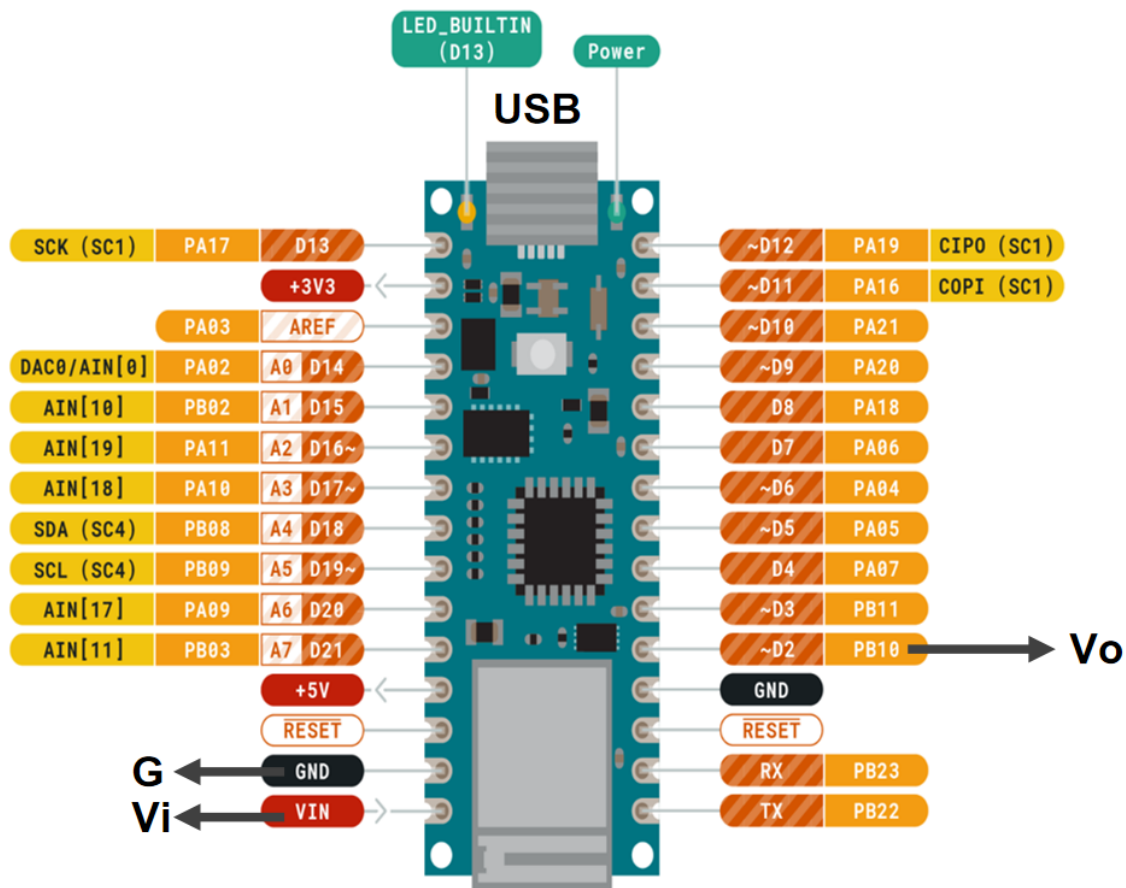
- The power of the HEATER is relatively large, and two 3.7V 150mAh batteries will be exhausted in a few minutes.

## 2- Arduino Nano 33 IoT Circuit (Use NMOS As A Switch)



	BATTERY1-HEATER	BATTERY2-Nano33	Vo-NMOS
Voltage(V)	7.4V (3.7V 150mAh*2)	7.4V (3.7V 70mAh*2)	3.3V
Current(A)	~0.4A	~0.1A	0A

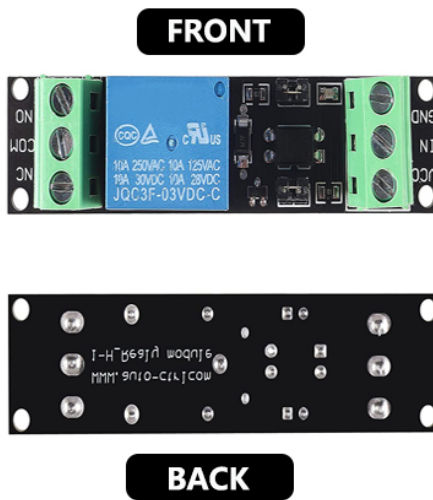
## 1.1.2 Datasheets



**Model number of relay: QC19004**

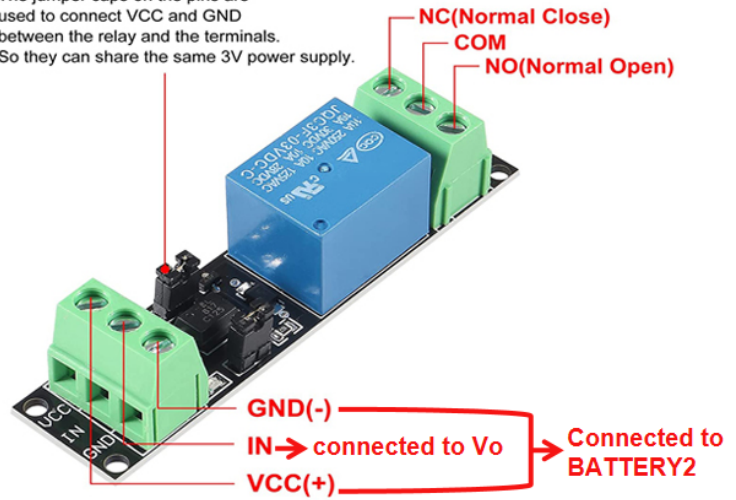
Power supply: 3.3v

Trigger voltage: 1.3V



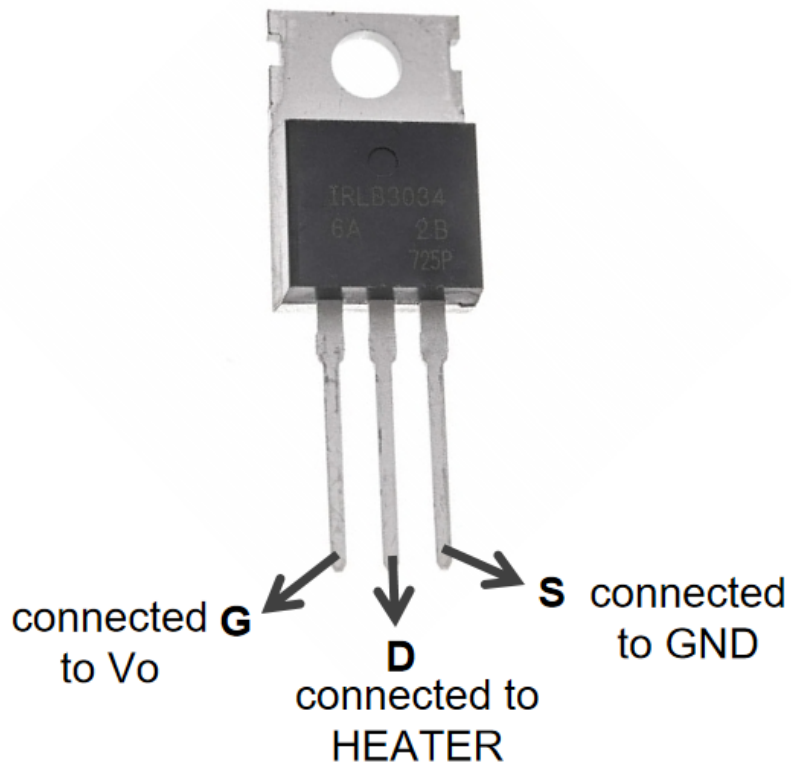
#### Jumper cap on the pins

The jumper caps on the pins are used to connect VCC and GND between the relay and the terminals. So they can share the same 3V power supply.



**Model number of NMOS:**

**IRLB3034 IRLB3034PBF MOSFET Transistor N Channel, 195 A, 40 V, TO-220**



### 1.1.3 Reference Link

Nano33IoT

<https://store-usa.arduino.cc/products/arduino-nano-33-iot>

Relay

<https://a.co/d/9s4Mw9O>

[https://blog.csdn.net/weixin\\_46251230/article/details/125470826](https://blog.csdn.net/weixin_46251230/article/details/125470826)

NMOS

<https://a.co/d/6L95oG6>

## 1.2 Software

## 1.2.1 WiFi

- Connect the Nano33IoT board with your PC by using USB wire and turn on the Arduino IDE
- Upload the Heater Code (**Remember to change the WiFi ssid and Password**) to the Arduino 33 IoT board by using [Arduino IDE](#)

```
#include <WiFiNINA.h>

// Please use our own WiFi network information
char ssid[] = "Mate 40 Pro"; // Network SSID (name)
char pass[] = "0e951e51d75d"; // Network Password

int status = WL_IDLE_STATUS; // Initial WiFi status uses the idle status
WiFiServer server(80); // Create a server that listens on port 80

void setup() {
  Serial.begin(9600); // Start serial communication
  // while (!Serial); // Wait for the serial port to connect. Not needed for Nano33IoT

  pinMode(LED_BUILTIN, OUTPUT); // Set the built-in LED to output mode
  pinMode(2, OUTPUT); // Set the built-in LED to output mode

  // Attempt to connect to the WiFi network
  while (status != WL_CONNECTED) {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    status = WiFi.begin(ssid, pass); // Connect to a WPA/WPA2 network

    // Wait 10 seconds for the connection to establish
    delay(10000);
  }

  // Once connected, print the IP address
  Serial.print("Successfully connected to ");
  Serial.println(ssid);
}
```

```

Serial.print("IP Address: ");
Serial.println(WiFi.localIP());

server.begin(); // Start the server
}

void loop() {
  WiFiClient client = server.available(); // Listen for client (

  if (client) { // If a client is connected
    Serial.println("New Client."); // Print a message that a new
    String currentLine = ""; // Used to store characters received
    while (client.connected()) { // While the client stays connected
      if (client.available()) { // If there's data available from client
        char c = client.read(); // Read the data
        Serial.write(c); // Write the data to the serial monitor
        if (c == '\n') { // If the character is a newline

          // If the current line is empty, i.e., the end of the line
          if (currentLine.length() == 0) {
            // Send a standard HTTP response header
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();

            // Control the LED
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">");
            client.println("<body><h2>Heater Control</h2>");
            client.println("<p><a href=\"/ON\">Turn On</a></p>");
            client.println("<p><a href=\"/OFF\">Turn Off</a></p>");
            client.println("</body></html>");

            // Disconnect
            break;
          } else { // If a new line is received, clear the current line

```



```

        currentLine = "";
    }
} else if (c != '\r') { // If the character is a carriage return
    currentLine += c; // Add other characters to the current line
}

// Check if the current line contains a command to control the LED
if (currentLine.endsWith("GET /ON")) {
    digitalWrite(LED_BUILTIN, HIGH); // Turn on the LED
    digitalWrite(2, HIGH); // Turn on the LED
}
if (currentLine.endsWith("GET /OFF")) {
    digitalWrite(LED_BUILTIN, LOW); // Turn off the LED
    digitalWrite(2, LOW); // Turn off the LED
}
}
}
// When the client disconnects
client.stop();
Serial.println("Client Disconnected.");
}
}

```


- Turn on your phone mobile hotspot (or Computers Hotspot) and set the correct hotspot name and password (same as your code)
- Find the correct IP of Nano 33 IoT by entering the hotspot "Connected devices"

## ← Connected devices

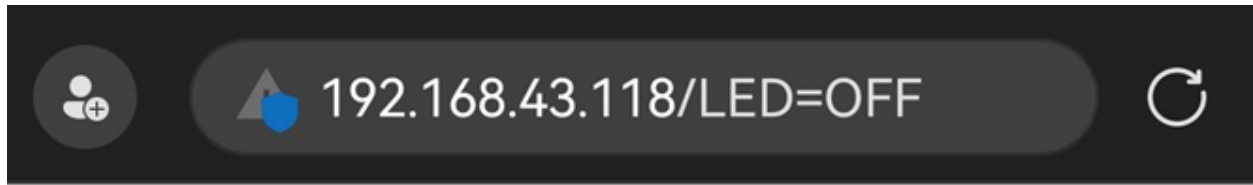
### Blocklist >

Block certain devices from connecting to your hotspot.

### CONNECTED DEVICES

 **LAPTOP-L4GEH55G**  
IP: [REDACTED]  
MAC [REDACTED]

- Type the IP into your Phone or any devices in the same hotspot
- You can see the similar image like this



# LED Control

[Turn On LED](#)

[Turn Off LED](#)

- Click the “Turn on” or “Turn off” button to control the heating process

## 1.2.2 Bluetooth

- Connect the Nano33IoT board with your PC by using USB wire and turn on the Arduino IDE
- Upload the Heater Code to the Arduino 33 IoT board by using [Arduino IDE](#)

```
#include <ArduinoBLE.h>
```

```
BLEService ledService("180A"); // BLE LED Service
```

```
BLEByteCharacteristic switchCharacteristic("2A57", BLERead | BLI
```

```
void setup() {
```

```
  Serial.begin(9600);
```

```
  // while (!Serial);
```

```
  pinMode(LED_BUILTIN, OUTPUT); // Set the built-in LED pin to o
```

```
  pinMode(2, OUTPUT); // Set the built-in LED pin to output mode
```

```

if (!BLE.begin()) {
    Serial.println("Starting Bluetooth® Low Energy failed!");
    while (1);
}

BLE.setLocalName("Nano 33 IoT");
BLE.setAdvertisedService(ledService);
ledService.addCharacteristic(switchCharacteristic);
BLE.addService(ledService);
switchCharacteristic.writeValue(0);
BLE.advertise(); // Start advertising
Serial.println("BLE LED Peripheral mode");
}

void loop() {
    BLEDevice central = BLE.central(); // Listen for BLE central

    if (central) {
        Serial.print("Connected to central device: ");
        Serial.println(central.address());

        while (central.connected()) {
            if (switchCharacteristic.written()) {
                switch (switchCharacteristic.value()) {
                    case 0x01:
                        controlledLED(HIGH, 0); // LED on
                        break;
                    default:
                        controlledLED(LOW, 0); // LED off
                        break;
                }
            }
        }

        Serial.print("Disconnected from central device: ");
        Serial.println(central.address());
    }
}

```

```

    }
}

void controlLED(int state, unsigned long blinkInterval) {
    static unsigned long previousMillis = 0;
    unsigned long currentMillis = millis();

    if (state == LOW) {
        digitalWrite(LED_BUILTIN, LOW);
        digitalWrite(2, LOW);
    } else if (blinkInterval > 0) {
        if (currentMillis - previousMillis >= blinkInterval) {
            previousMillis = currentMillis;
            if (digitalRead(LED_BUILTIN) == LOW) {
                digitalWrite(LED_BUILTIN, HIGH);
                digitalWrite(2, HIGH);
            } else {
                digitalWrite(LED_BUILTIN, LOW);
                digitalWrite(2, LOW);
            }
        }
    } else {
        digitalWrite(LED_BUILTIN, state);
        digitalWrite(2, state);
    }
}
}

```

- Download the "Lightblue" APP in your phone from the app store
- Connect your phone with the Nano33IoT PCB board, write "1"(turn on the heater circuit)or "0"(default state, turn off the heater circuit) to control the heating process

How to Use LightBlue®: The Go-To BLE Development Tool

## 2. Tinycircuit

## 2.1 Hardware

To make sure that the weight of the PCB board and batteries is as light as possible, we first bought the TinyCircuit boards to achieve the basic function of control. We purchased three TinyCircuit boards which are a TinyZero Processor board, a Servo TinyShield, and a Bluetooth TinyShield. The TinyZero Processor board is used only to upload codes to the Servo board or the Bluetooth board. By sandwiching the TinyZero Processor with the Servo board or the Bluetooth board, you can then start writing and uploading your code. The Servo board is specially designed to power servos, there are four output channels, and each can be used separately. Every channel has three outputs, one for a voltage of 4.3, one for the ground, and one for a PWM wave whose voltage is around 0.001V. The Bluetooth board can be used for Bluetooth communication just as the name indicates.

## 2.2 Software

The Bluetooth board was tested first to make sure that it could be successfully used for Bluetooth communication. The codes are shown below. This code can also be found on the website of the Bluetooth board.

```
//-----  
//  TinyCircuits ST BLE TinyShield UART Example Sketch  
//  Last Updated 2 March 2016  
//  
//  This demo sets up the BlueNRG-MS chipset of the ST BLE module  
//  with Nordic's virtual UART connection, and can pass data between  
//  serial monitor and Nordic nRF UART V2.0 app or another computer  
//  terminal. This example is written specifically to be fairly  
//  with the Nordic NRF8001 example, with a replacement UART.ino  
//  'aci_loop' and 'BLEsetup' functions to allow easy replacement  
//  
//  Written by Ben Rose, TinyCircuits http://tinycircuits.com  
//  
//-----  
  
#include <SPI.h>
```

```

#include <STBLE.h>

//Debug output adds extra flash and memory requirements!
#ifndef BLE_DEBUG
#define BLE_DEBUG true
#endif

#if defined (ARDUINO_ARCH_AVR)
#define SerialMonitorInterface Serial
#elif defined(ARDUINO_ARCH_SAMD)
#define SerialMonitorInterface SerialUSB
#endif

uint8_t ble_rx_buffer[21];
uint8_t ble_rx_buffer_len = 0;
uint8_t ble_connection_state = false;
#define PIPE_UART_OVER_BTLE_UART_TX_TX 0

const int ledPin = LED_BUILTIN; // Pin connected to the onboard

void setup() {
  SerialMonitorInterface.begin(9600);
  while (!SerialMonitorInterface); //This line will block until
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, HIGH);
  BLEsetup();
}

void loop() {
  aci_loop();//Process any ACI commands or events from the NRF80
  //SerialMonitorInterface.print(ble_rx_buffer_len);

  if (ble_rx_buffer_len) {//Check if data is available
    SerialMonitorInterface.print(ble_rx_buffer_len);
    SerialMonitorInterface.print(" : ");
  }
}

```

```

SerialMonitorInterface.println((char*)ble_rx_buffer);

if (strcmp((char*)ble_rx_buffer, "on") == 0) {
    digitalWrite(ledPin, HIGH);
    SerialMonitorInterface.print(ble_rx_buffer_len);
}

if (strcmp((char*)ble_rx_buffer, "off") == 0) {
    digitalWrite(ledPin, LOW);
    SerialMonitorInterface.print(ble_rx_buffer_len);
}

ble_rx_buffer_len = 0; //clear after reading
}

if (SerialMonitorInterface.available()) { //Check if serial input
    delay(10); //should catch input
    uint8_t sendBuffer[21];
    uint8_t sendLength = 0;
    while (SerialMonitorInterface.available() && sendLength < 20) {
        sendBuffer[sendLength] = SerialMonitorInterface.read();
        sendLength++;
    }
    if (SerialMonitorInterface.available()) {
        SerialMonitorInterface.print(F("Input truncated, dropped: "));
        if (SerialMonitorInterface.available()) {
            SerialMonitorInterface.write(SerialMonitorInterface.read());
        }
    }
    sendBuffer[sendLength] = '\0'; //Terminate string
    sendLength++;
    if (!lib_aci_send_data(PIPE_UART_OVER_BTLE_UART_TX_TX, (uint8_t*)sendBuffer, sendLength)) {
        SerialMonitorInterface.println(F("TX dropped!"));
    }
}

```



```

    }
  }
}

```

We then tested the Servo board, but unluckily we found that this board built especially for servo motivation did not suite our need. That's because the relay or NMOS we used as a switch requires a voltage greater than at least 1.3V to trigger, but this Servo board could only generate a PWM wave whose voltage could only reach 0.001V, way to small to trigger the switch of our circuit, no matter which component we use as the switch. At last we had to give up on the TinyCircuit method. This TinyCircuit Board might find its proper usage in the lab someday and somehow. The following code is used to activate the Servo board's output. This code can also be found and downloaded from the TinyCircuit website.

```

#include <Wire.h>
#include <ServoDriver.h>

ServoDriver servo(NO_R_REMOVED); //this value affects the I2C address
//removing resistors R1-R3. Then
//R2_REMOVED, R1_R2_REMOVED, R1_
//Default is NO_R_REMOVED

#ifdef (ARDUINO_ARCH_AVR)
#define SerialMonitorInterface Serial
#elif defined(ARDUINO_ARCH_SAMD)
#define SerialMonitorInterface SerialUSB
#endif

void setRelay(uint8_t relay, uint16_t val){
  if (val == 0) {
    servo.setServo(relay, 2000);
  } else {
    servo.setServo(relay, 10000);
  }
}

```

```

}

void setup(){
  SerialMonitorInterface.begin(9600);
  Wire.begin();

  servo.useResetPin();          //This tells the library to use 1
  if(servo.begin(20000)){        //Set the period to 20000us or 20
    SerialMonitorInterface.println("Relay not detected!");
    while(1);
  }
  //The failsafe turns off the PWM output if a command is not se
  //Failsafe is set in milliseconds- comment or set to 0 to disa
  servo.setFailsafe(1000);
  setRelay(1, 1);
  setRelay(2, 1);
  setRelay(3, 1);
  setRelay(4, 1);
}

void loop()
{

}

#include <Wire.h>
#include <ServoDriver.h>

ServoDriver servo(NO_R_REMOVED); //this value affects the I2C add
                                //removing resistors R1-R3. Ther
                                //R2_REMOVED, R1_R2_REMOVED, R1_
                                //Default is NO_R_REMOVED

#if defined (ARDUINO_ARCH_AVR)
#define SerialMonitorInterface Serial
#elif defined(ARDUINO_ARCH_SAMD)
#define SerialMonitorInterface SerialUSB

```

```

#endif

void setRelay(uint8_t relay, uint16_t val){
  if (val == 0) {
    servo.setServo(relay, 2000);
  } else {
    servo.setServo(relay, 10000);
  }
}

void setup(){
  SerialMonitorInterface.begin(9600);
  Wire.begin();

  servo.useResetPin();          //This tells the library to use 1
  if(servo.begin(20000)){        //Set the period to 20000us or 20
    SerialMonitorInterface.println("Relay not detected!");
    while(1);
  }
  //The failsafe turns off the PWM output if a command is not set
  //Failsafe is set in milliseconds- comment or set to 0 to disable
  servo.setFailsafe(1000);
  setRelay(1, 1);
  setRelay(2, 1);
  setRelay(3, 1);
  setRelay(4, 1);
}

void loop()
{

}

```

### 3. Heater

## 3.1 Test result

**Test with 8V DC voltage:**

<https://prod-files-secure.s3.us-west-2.amazonaws.com/ea472edd-ed30-44a6-a264-ee7c35e0df01/c619634a-4aa5-492a-88af-52a6cf92e7cc/716c9488827328cf6cccb164dfbc5720.mp4>

**Test in circuit:** Using Arduino Nano 33 IoT Circuit; Using NMOS As A Switch; Controlled by bluetooth; BETTERY1 is two 3.7V 150mAh batteries in series; BETTERY2 is two 3.7V 70mAh batteries in series;

<https://prod-files-secure.s3.us-west-2.amazonaws.com/ea472edd-ed30-44a6-a264-ee7c35e0df01/ec870e11-f83f-44db-9987-724a64711802/dbc1734fc20b6acf1720ffc6ff39925f.mp4>

## 3.2 Suggestions

- Putting a piece of tape over the LCE will make it stiffer and go a little further with each heat.