

INSTRUCTABLES

“ The cold Room”

GINNIE LEE ZIQI // G02

Project By: Ginnie Lee, Nicole chen

INSTRUCTABLES

“ The cold Room”

Nowadays, people are getting connected more and more online and even during meal times or time spent with your loves ones, we stare down at our phones and use social media / messaging applications instead of attempting to connect with our loved ones with us physically. The more we are on our phones to stay connected online, the more we neglect those around us, creating a disconnection in real life. This project aims to bring out the loneliness we feel when we are “abandoned” by our loved ones or when we “abandon” our loved ones in exchange for connectivity online. The more we wish to be connected online, the lonelier we feel offline.

DESCRIPTION

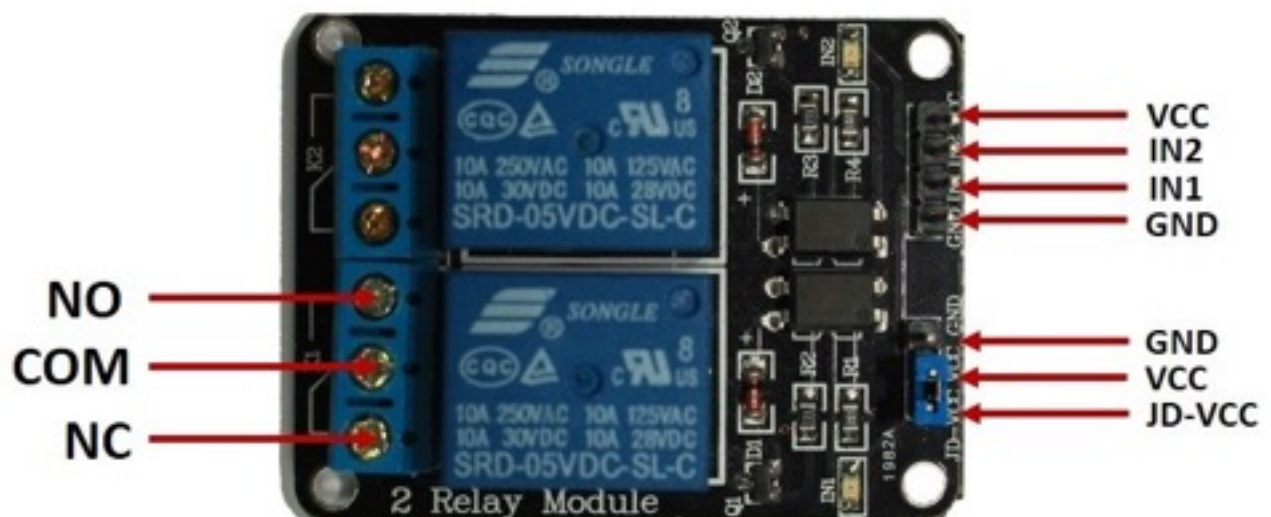
A Peltier module connected to Arduino which is connected to Twitter and hence controllable through Twitter. The Peltier module is inserted into a neck pillow and the neck pillow is situated in a all white room, which has a hidden exit that leads to a secret room.

Step 1: Get materials

Materials needed for this project:

- Arduino
- Breadboard
- 12V Peltier Module
- Relay Module
- Jumper wires
- Male to female jumper wires
- 12V Power adapter
- Heat sink
- Cooling fan
- 10V resistor
- Crocodile / alligator clips
- Button
- Testpen
- Blynk app

- ## Step 2: connecting the Peltier module to Arduino



Based on the above drawing, the positive end of the Peltier module (denoted by the red wire), is connected to the Common of 2nd relay module on our 2 module relay. The negative end of the Peltier module (denoted by the black wire), is connected to the negative end of the 12V Power adapter. The positive end of the 12V power adapter is then connected to NC (Normally Closed) of the 2nd relay module. The IN2 pin of the relay module is then connected to Digital Pin 8 on Arduino with a female to male jumper wire. The GND (Ground) of the Relay module is connected to the GND of Arduino.

Next, the 5V of Arduino is drawn to the Breadboard. This 5V that is drawn to the Breadboard is then connected to VOC on the relay module and to one side of the button. The other side of the button is connected to GND through a 10V resistor and to Digital Pin 8 of Arduino.

This is to connect the Peltier module to Arduino and to the button, which controls the opening and closing of the circuit. When the button is pressed, the circuit is closed and it is then will the Peltier module react with a drop in temperature.

Step 3: Input of the code in Arduino

The code below is keyed in into Arduino on computer and thereafter sent to Arduino.

*/**

Button

Turns on and off a peltier module attached at pin 8,

when pressing a pushbutton attached to pin 3.

**/*

// constants won't change. They're used here to set pin numbers:

const int buttonPin = 3; // the number of the pushbutton pin

const int PeltierPin = 8; // the number of the LED pin

// variables will change:

```
int buttonState = 0;    // variable for reading the pushbutton status
```

```
void setup() {
```

```
  // initialize the LED pin as an output:
```

```
  pinMode(PeltierPin, OUTPUT);
```

```
  // initialize the pushbutton pin as an input:
```

```
  pinMode(buttonPin, INPUT);
```

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

```
}
```

```
void loop() {
```

```
  // read the state of the pushbutton value:
```

```
  buttonState = digitalRead(buttonPin);
```

```
  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
```

```
  if (buttonState == HIGH) {
```

```
    // turn LED on:
```

```
    Serial.println("Relay on");
```

```
    digitalWrite(PeltierPin, LOW);
```

```
  } else {
```


```
    // turn LED off:
```

```
    Serial.println("Relay off");
```

```
    digitalWrite(PeltierPin, HIGH);
```

```
  }
```

```
}
```



```
1010ginniecul_cfm_peltier | Arduino 1.8.9

when pressing a pushbutton attached to pin 3.
*/

// constants won't change. They're used here to set pin numbers:
const int buttonPin = 3;    // the number of the pushbutton pin
const int PeltierPin = 8;    // the number of the LED pin

// variables will change:
int buttonState = 0;        // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(PeltierPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);

  Serial.begin(9600);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    Serial.println("Relay on");
    digitalWrite(PeltierPin, HIGH);
  } else {
    // turn LED off:
    Serial.println("Relay off");
    digitalWrite(PeltierPin, LOW);
  }
}

Done uploading.
Sketch uses 1978 bytes (6%) of program storage space. Maximum is 32256 bytes.
Global variables use 206 bytes (10%) of dynamic memory, leaving 1842 bytes for local variables.

9 Arduino/Genuino Uno on /dev/cu.usbmodem1421
```

With this code sent to Arduino, the Peltier module is then able to react with a drop in temperature when the button is pressed.

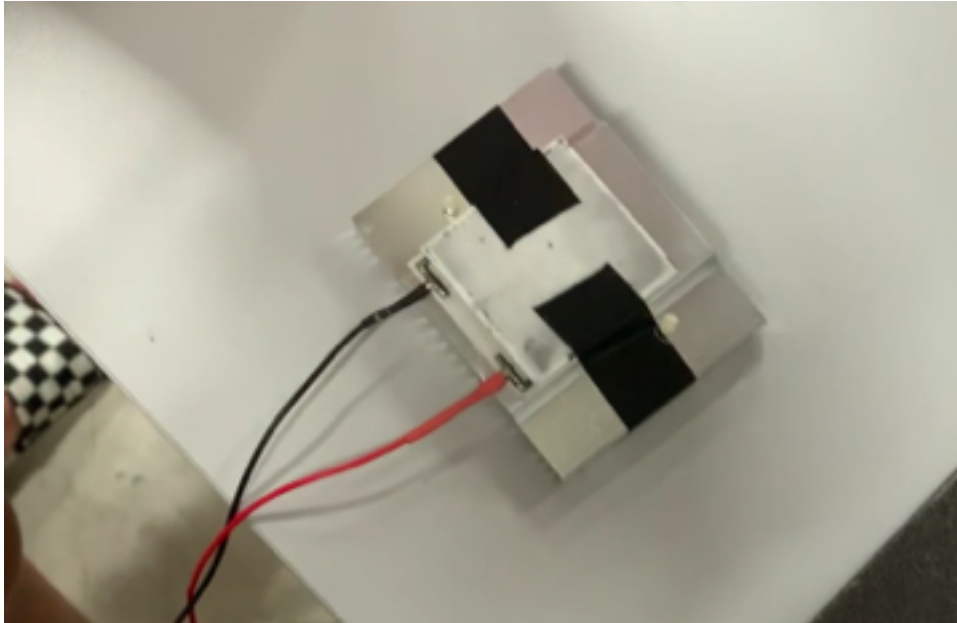
Step 4: connect the cooling fan (12V) to the power source

By connecting the cooling fan to the power adapter, the fan will then work when the power adapter is connected and hence is able to help suck out the heat produced on the other side of the Peltier module to ensure the cold side can stay cold for a longer time.

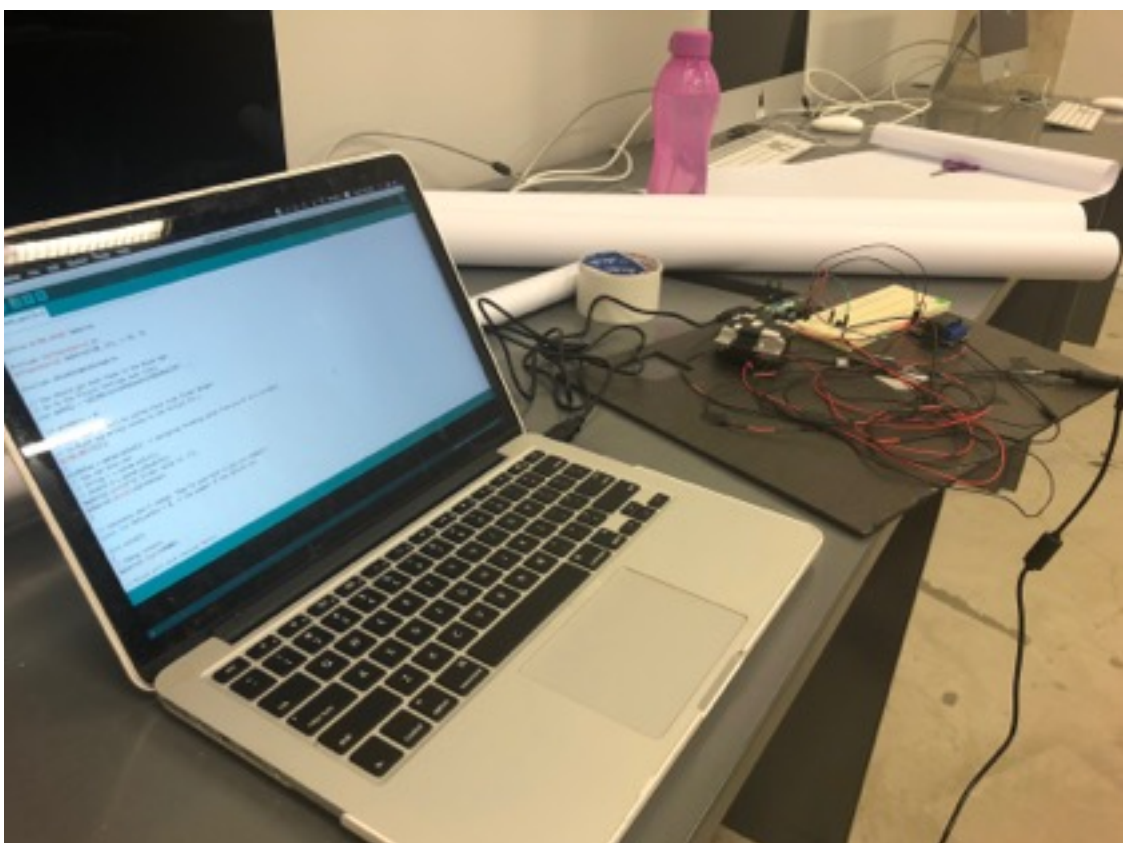
As the fan need not be controlled by Arduino and only needs to work to cool down the Peltier, it can be directly connected to the power adapter that supplies the 12V needed for the fan.

Step 5: Attaching the fan and the heat sink to the Peltier module

We then placed the Peltier module onto the heat sink with the hot side down and attached the cooling fan under the heat sink. The entire thing is bounded together by insulating tape.



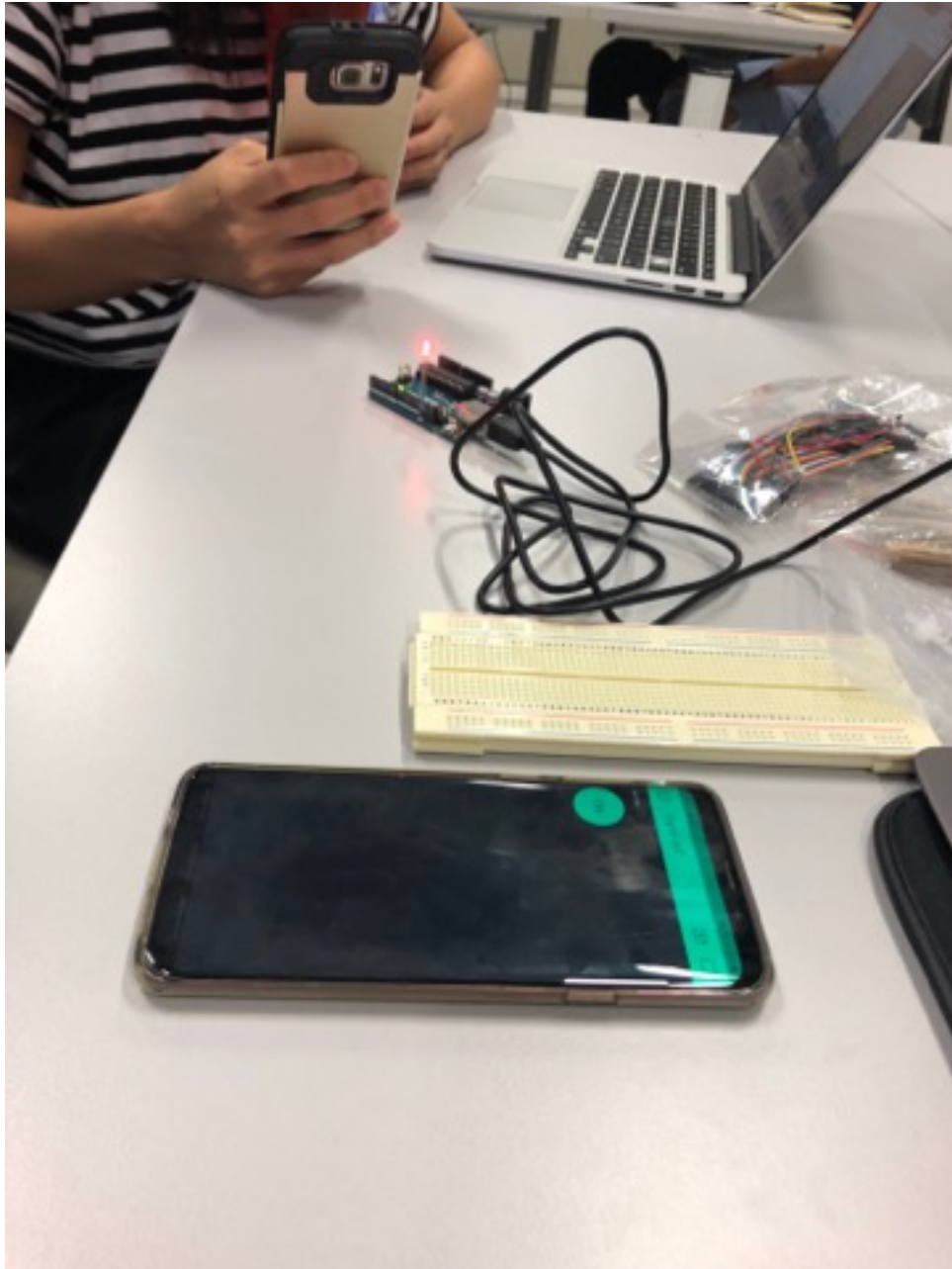
The circuit can now be tested for workability through the use of the button.



Videos - Button tests // (Video Links can be found on OSS post.)

Step 6: connecting Blynk to the computer

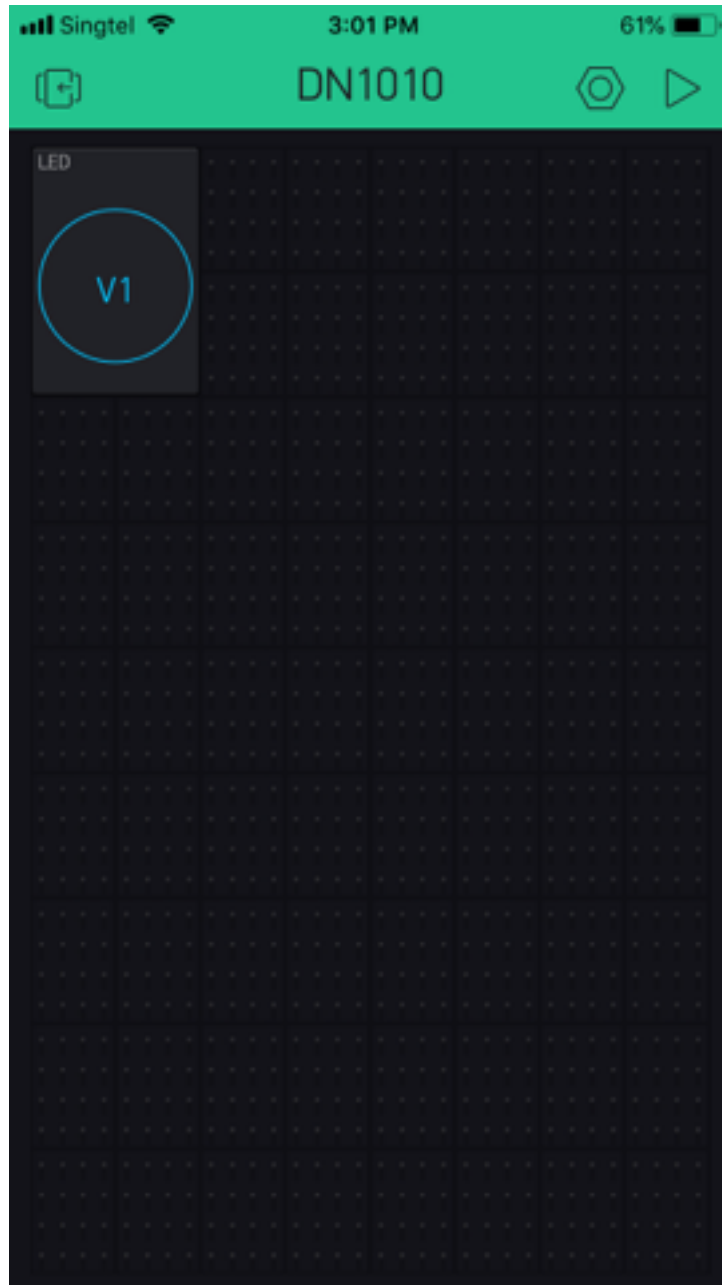
Following instructions on OSS, we then connected the Blynk app to the computer. We tested the workability of the Blynk app afterwards with a LED light.



Video - LED light controlled with Blynk // (Video link can be found on OSS post)

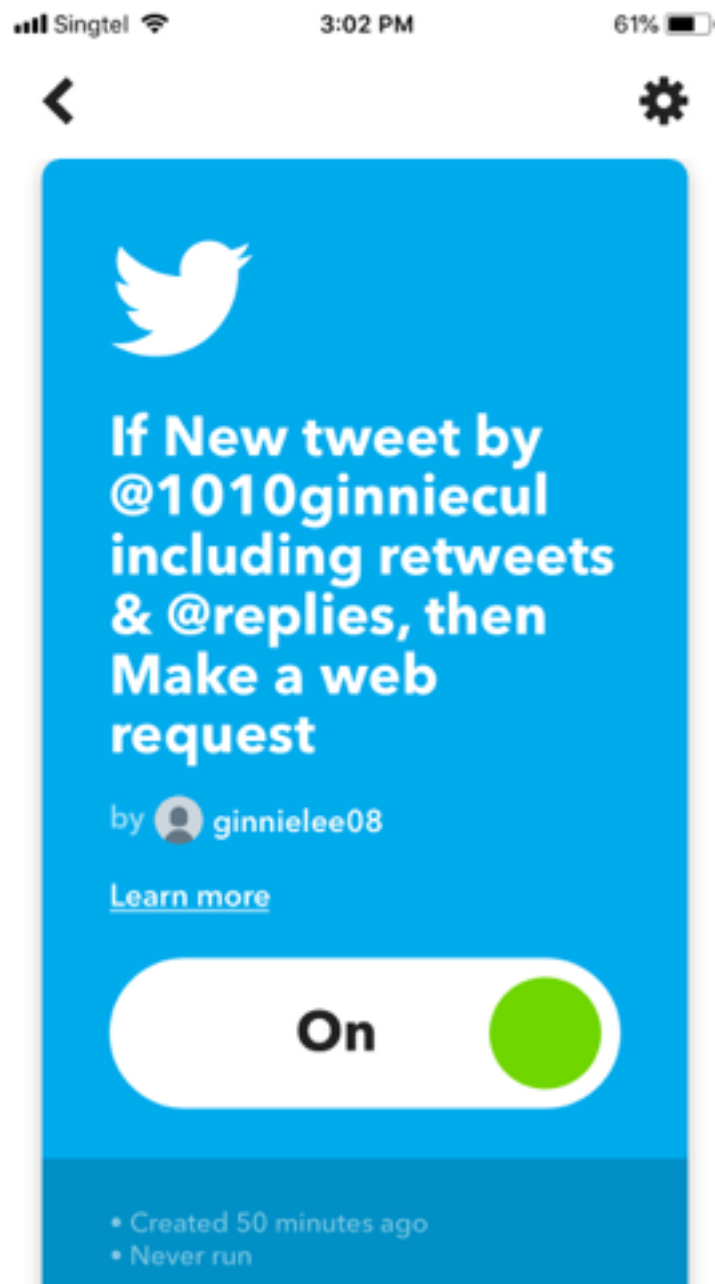
Step 7: connecting Twitter to Arduino via the IFTTT app and Blynk

Thereafter, we created a separate project for the Peltier and connected the button to V1 (Virtual 1). This allows Blynk to control the Peltier module via Arduino.



Video - Connecting Blynk app to Arduino // (Video link can be found on OSS post)

We then created a link via webhooks on IFTTT to connect Twitter to Arduino.



Video - Connecting Twitter to Arduino // (Video link can be found on OSS post)

The code below is used to allow Twitter to control the Peltier module through Arduino.

```
#define BLYNK_PRINT SwSerial

#include <SoftwareSerial.h>

SoftwareSerial SwSerial(10, 11); // RX, TX

#include <BlynkSimpleStream.h>

// You should get Auth Token in the Blynk App.

// Go to the Project Settings (nut icon).

char auth[] = "dd2304c3e7cb4382be621395648e6270";

int pinValue = 0;

// This function will be called every time Slider Widget

// in Blynk app writes values to the Virtual Pin 1

BLYNK_WRITE(V1)

{ pinValue = param.asInt(); // assigning incoming value from pin V1 to a variable

// You can also use:

// String i = param.asStr();

// double d = param.asDouble();

SwSerial.print("V1 Slider value is: ");

SwSerial.println(pinValue); }

// constants won't change. They're used here to set pin numbers:

const int PeltierPin = 8; // the number of the Peltier pin

void setup()

{

// Debug console
```

```
SwSerial.begin(9600);

// Blynk will work through Serial

// Do not read or write this serial manually in your sketch

Serial.begin(9600);

Blynk.begin(Serial, auth);


pinMode(PeltierPin, OUTPUT);

}

void loop()

{

  Blynk.run();

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:

  if (pinValue == 1) {

    // turn LED on:

    Serial.println("Relay on");

    digitalWrite(PeltierPin, LOW);

  } else {

    // turn LED off:

    Serial.println("Relay off");

    digitalWrite(PeltierPin, HIGH);

  }

}
```

```
twitter_to_arduino_ | Arduino 1.8.9

twitter_to_arduino_ $

// constants won't change. They're used here to set pin numbers:
const int PeltierPin = 8; // the number of the Peltier pin

void setup()
{
  // Debug console
  Serial.begin(9600);

  // Blynk will work through Serial
  // Do not read or write this serial manually in your sketch
  Serial.begin(9600);
  Blynk.begin(Serial, auth);

  pinMode(PeltierPin, OUTPUT);
}

void loop()
{
  Blynk.run();

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (pinValue == 1) {
    // turn LED on:
    Serial.println("Relay on");
    digitalWrite(PeltierPin, LOW);
  } else {
    // turn LED off:
    Serial.println("Relay off");
    digitalWrite(PeltierPin, HIGH);
  }
}

Done uploading.
Sketch uses 9554 bytes (29%) of program storage space. Maximum is 32256 bytes.
Global variables use 437 bytes (21%) of dynamic memory, leaving 1611 bytes for local variables.
Arduino/Genuino IDE on /dev/tty.usbmodem1421
```

```
twitter_to_arduino_ | Arduino 1.8.9

twitter_to_arduino_ $

#define BLYNK_PRINT Serial

#include <SoftwareSerial.h>
SoftwareSerial SsSerial(10, 11); // RX, TX

#include <BlynkSimpleStream.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "dd2304c3e7cb4382be621395648e6270";

int pinValue = 0;

// This function will be called every time Slider Widget
// in Blynk app writes values to the Virtual Pin 1
BLYNK_WRITE(V1)
{
  pinValue = param.asInt(); // assigning incoming value from pin V1 to a variable
  // You can also use:
  // String s = param.asStr();
  // double d = param.asDouble();
  SsSerial.print("V1 Slider value is: ");
  SsSerial.println(pinValue);
}

// constants won't change. They're used here to set pin numbers:
const int PeltierPin = 8; // the number of the Peltier pin

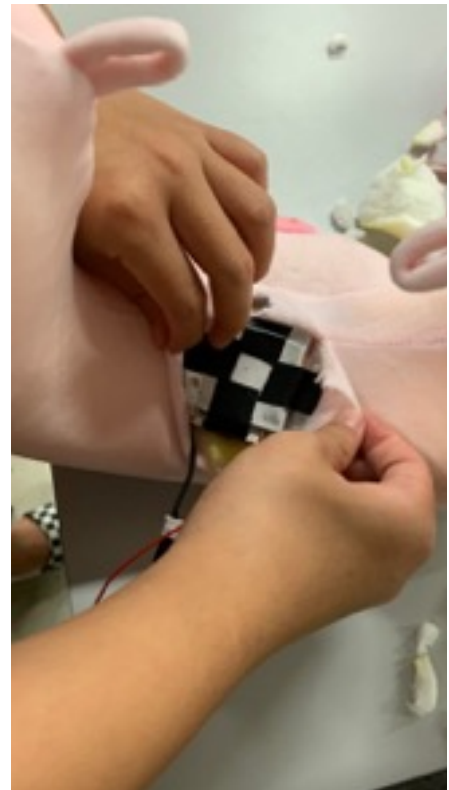
void setup()
{
  // Debug console
  Serial.begin(9600);

  // Blynk will work through Serial
}

Done uploading.
Sketch uses 9554 bytes (29%) of program storage space. Maximum is 32256 bytes.
Global variables use 437 bytes (21%) of dynamic memory, leaving 1611 bytes for local variables.
Arduino/Genuino IDE on /dev/tty.usbmodem1421
```

Step 8: Packaging the Peltier module into the neck pillow

Thereafter, we cut a hole in the neck pillow where the neck would be and pulled out the stuffing in that area to fit the Peltier module in as well as allow space for the fan to spin inside the neck pillow.



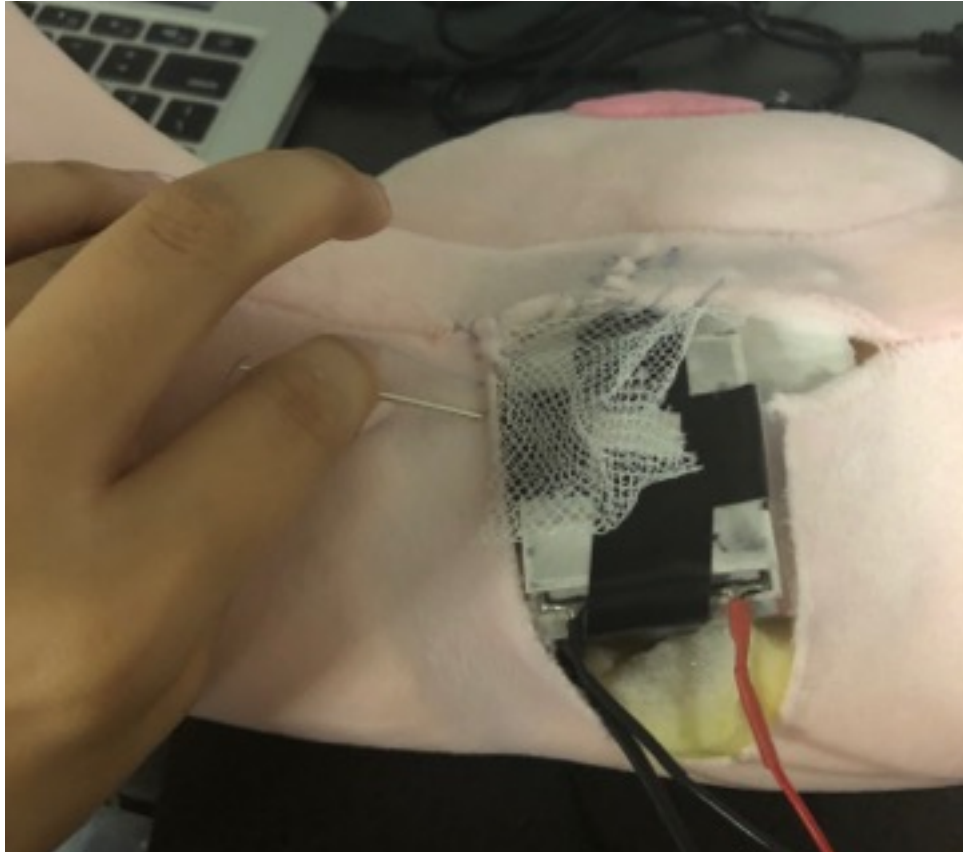
Step 9: cutting a hole through the back of the neck pillow

By cutting a hole through the back of the neck pillow, the heat generated by the other side of the Peltier module is then allowed to escape.



Step 10: Stitching the neck pillow back together

Thereafter, the neck pillow is sewed back together, with a mesh covering the Peltier to ensure that the temperature can be felt but at the same time ensure that the user does not get burnt if the heat is not dissipated fast enough and affects the cold side of the Peltier too.

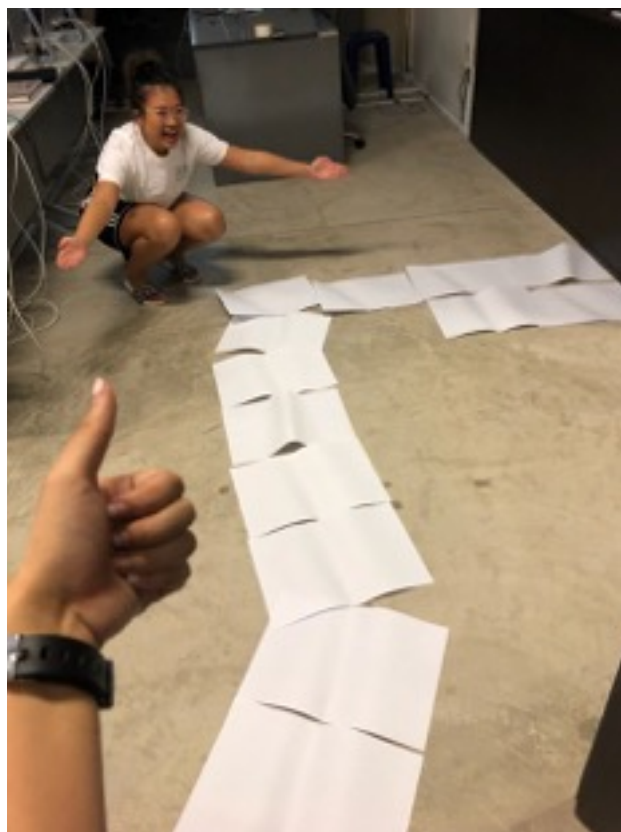


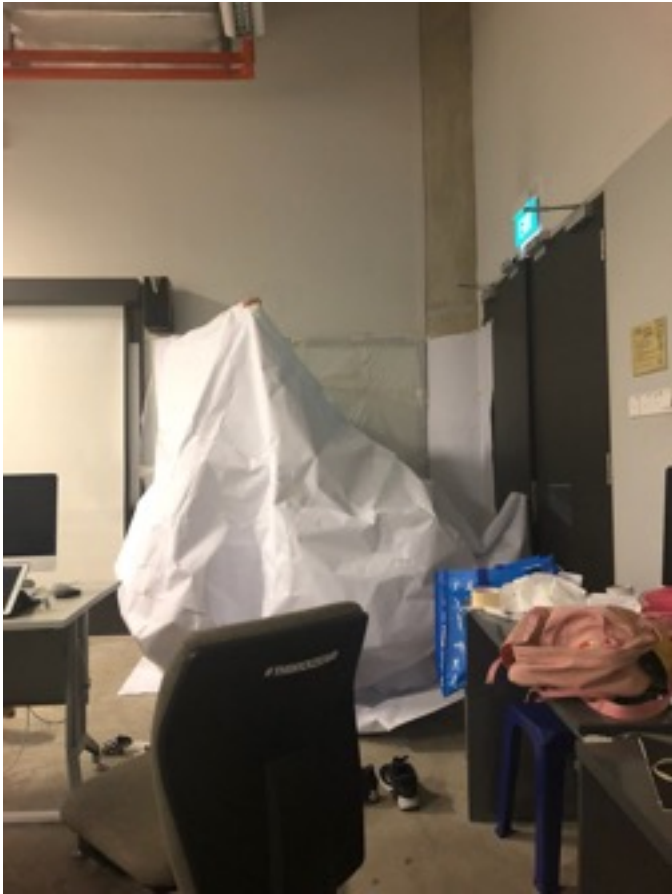
Step 11: Setting up of the room

Thereafter, we started setting up the room. As we wanted the entire room to be white, we bought mahjong paper and sourced for white trash bags to cover / create the walls and floor for the room. We selected a room to build our room in and since there was a mirror, covered the mirror with white trash bags so that the entire wall is covered.



We then used mahjong paper to cover the floor of our room, and similarly used mahjong paper to build the remaining walls of the room.





For the hidden exit, we cut a hole in the mahjong paper wall and replaced the opening with cascading white trash bags.

Step 12: Setting up within the room

The table was then brought into the white room and covered with white mahjong paper. Simple instructions was then written on the table surface.



Instructions include:

1. Pick up the phone. Start tweeting.
2. Reply all responses to your tweet. Interact online.
3. Leave whenever you want.

We then placed the neck pillow on the table, taping visible wires with white cloth tape before hiding the entire rest of the circuit behind the white wall.



Step 13: Set up of the hidden room

Chairs were arranged in a semi circle behind the hidden door and some seats were filled with people to welcome the participants who walk through the hidden door.

Step 14: Placing of the neck pillow

The neck pillow was then placed on the table with the rest of the circuit placed behind one of the white walls. The installation is now ready for its participants!



