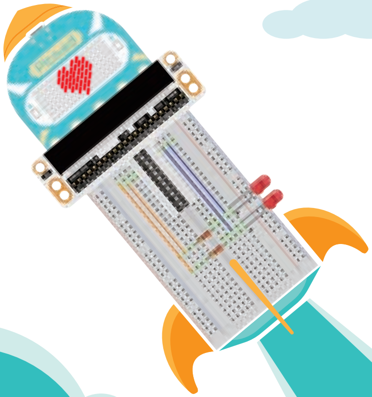
# Pico:ed

Starter Kit Guide

ELECFREAKS

MAKE CODING ACCESSIBLE



10+ YEAR

Your First Pico:ed Kit

Funny Easy Suitable for Your Beginning

MAKER EDUCATION STEM

open hardware

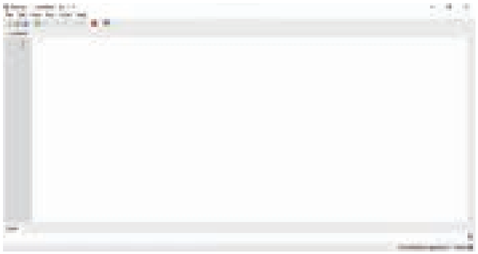
Pico:ed Starter Kit is an electronic starter kit based on RP2040. The kit provides some basic electronic components such as LEDs, buttons, buzzers, temperature sensors, servos and motors, it helps you enter a wonderful electronic world.

# PRODUCT CONTENT

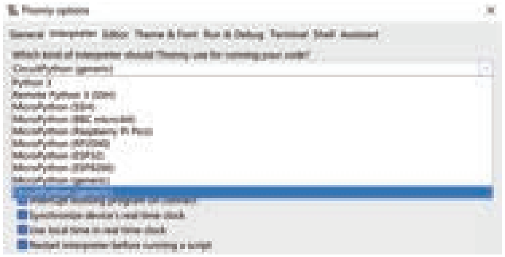
# Preparation for Programming

## Program editor

1. Download and install the software from this link: https://thonny.org.



2. Click the " Options "  from the drawer of the " Tools " , choose " Interpreter " and click " CircuitPython(generic) " , then confirm it.



Thonny options

General Interpreter Editor Theme & Font Run & Debug Terminal Shell Assistant

Which kind of interpreter should Thonny use for running your code?

CircuitPython (generic)

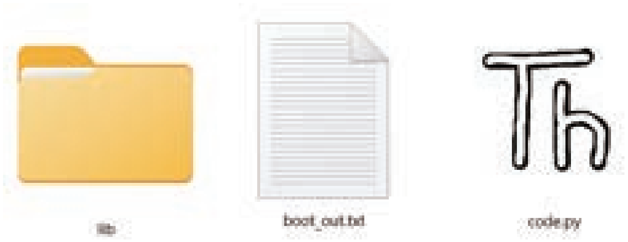
Python 3
Remote Python 3 (SSH)
MicroPython (SSH)
MicroPython (BBC micro:bit)
MicroPython (Raspberry Pi Pico)
MicroPython (RP2040)
MicroPython (ESP32)
MicroPython (ESP8266)
MicroPython (generic)

- ✓ Interrupt working program on connect
- ✓ Synchronize device's real time clock
- ✓ Use local time in real-time clock
- ✓ Restart interpreter before running a script

## Pico:ed firmware

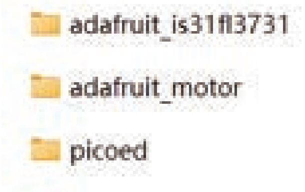
3 Open the link of the official circuitpython and go to the downloading page for Pico:ed with the latest version: https://circuitpython.org/board/elecfreaks_picoed/.

4 After downloading, long pressing the BOOTSEL button of Pico:ed, connect it with the USB cable, release the button until you see a disk named RPI - RP2 on the computer. Open RPI-RP2 and drag into the files that you just downloaded, you will see files in below pictures:



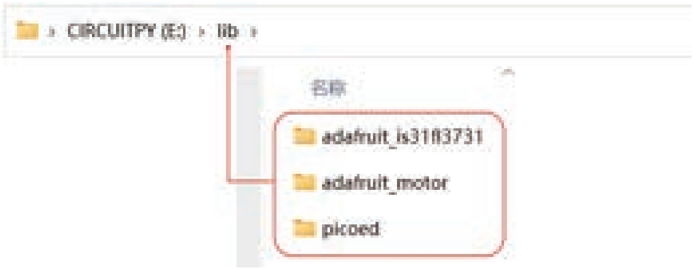
## Relevant libraries

5 Open the below link and find the three libraries of CircuitPython_IS31FL3731, circuitpython_picoed, CircuitPython_Motor and unzip them.
https://www.elecfreaks.com/learn-en/circuitpython-libraries.



adafruit_is31fl3731
adafruit_motor
picoed

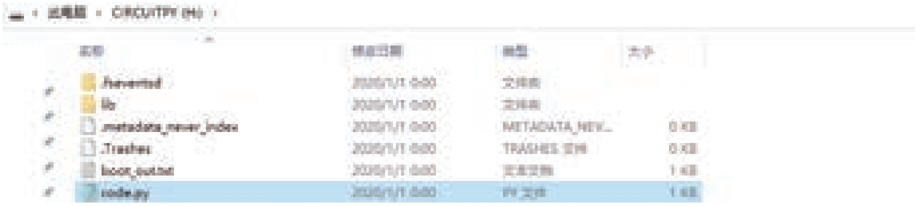
Open the above three files and copy them to the " lib " folder in the " CIRCUITPY " disk.



CIRCUITPY (E:) › lib ›
名称
adafruit_is31fl3731
adafruit_motor
picoed

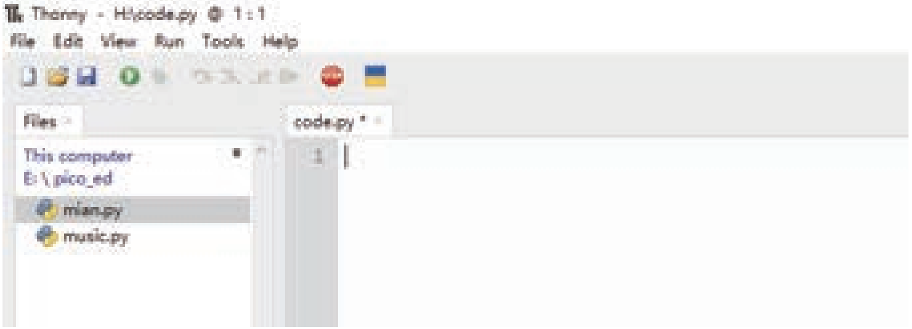
## Programming

6 The following programming is made in the file of code.py which is in CIRCUITPY disk, use Thonny to open code.py and do the program.



此电脑 › CIRCUITPY (H:) ›

| 名称 | 修改日期 | 类型 | 大小 |
|---|---|---|---|
| .fseventsd | 2020/1/1 0:00 | 文件夹 | |
| lib | 2020/1/1 0:00 | 文件夹 | |
| .metadata_never_index | 2020/1/1 0:00 | METADATA_NEV... | 0 KB |
| .Trashes | 2020/1/1 0:00 | TRASHES 文件 | 0 KB |
| boot_out.txt | 2020/1/1 0:00 | 文本文档 | 1 KB |
| code.py | 2020/1/1 0:00 | PY 文件 | 1 KB |



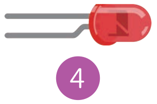
Thonny - H:\code.py @ 1 : 1

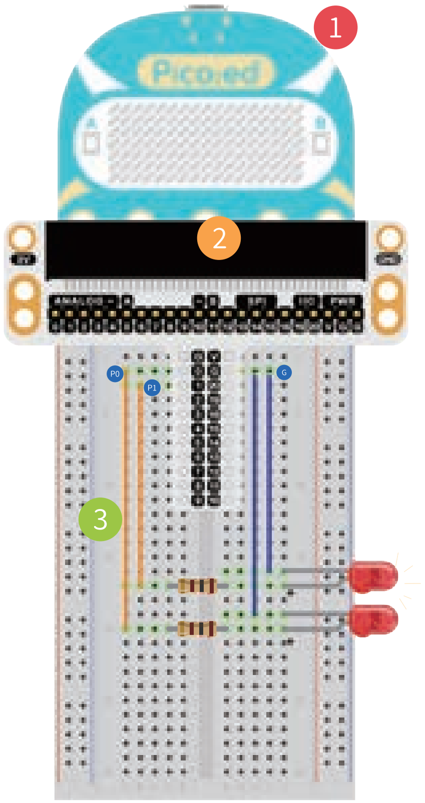
This computer
E: \ pico_ed
mian.py
music.py

# 1. LED

## Component List

1. 1 x Pico:ed
2. 1 x Breadboard Adapter
3. 1 x Breadboard
4. 2 x LEDs
5. 2 x 100Ω Resistors





## Description

The LED lights are used widely, this project is to control the 2 LEDs and make an alternative flashing through Pico:ed.

# 1. LED
step

```python
import board
import digitalio
import time
led_0 = digitalio.DigitalInOut(board.P0_A0)
led_1 = digitalio.DigitalInOut(board.P1_A1)
led_0.direction = digitalio.Direction.OUTPUT
led_1.direction = digitalio.Direction.OUTPUT
while True:
    led_0.value = True
    led_1.value = False
    time.sleep(1)
    led_0.value = False
    led_1.value = True
    time.sleep(1)
```

1. Import the modules of board, digitalio and time that we need.
2. Set the pins and the directions of the resistor, here we use P0_A0 and P1_A1.
3. Set the status of the 2 LEDs alternates from on and off.

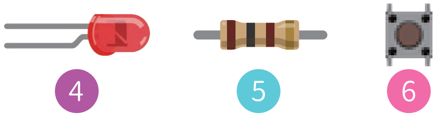
◉ Result: The 2 LEDs flash alternately, please check your settings if not working.

◉ Question: How to simulate the traffic lights with 3 LEDs?

# 2. Button

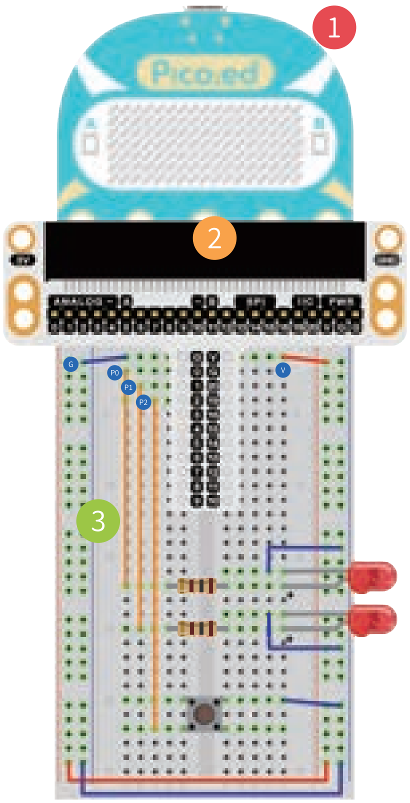
## Component List

1. 1 X Pico:ed
2. 1 X Breadboard Adapter
3. 1 X Breadboard
4. 2 X LEDs
5. 2 X 100Ω Resistors
6. 1 X Momentary Push Button



## Description

We are going to use the button module to control the 2 LEDs. When we press the button, the two LEDs flash alternately and they stop flashing once you release the button.



## 2. Button
step

```
import board
import digitalio
import time
led_0 = digitalio.DigitalInOut(board.P0_A0)
led_1 = digitalio.DigitalInOut(board.P1_A1)
led_0.direction = digitalio.Direction.OUTPUT
led_1.direction = digitalio.Direction.OUTPUT
button = digitalio.DigitalInOut(board.P2_A2)
button.pull = digitalio.Pull.UP
while True:
    if button.value == False:
        led_0.value = True
        led_1.value = False
        time.sleep(1)
        led_0.value = False
        led_1.value = True
        time.sleep(1)
```

1 Import the modules of board, digitalio and time that we need.

2 Set the pins and directions of the LEDs, here we use P0_A0 and P1_A1.

3 Set the pins of the button as P2_A2 and pull up the input status.

4 Judge if the button is pressed, turn on/off the LEDs.

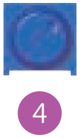
◎ Result: Press the button and the two LEDs flash alternately.

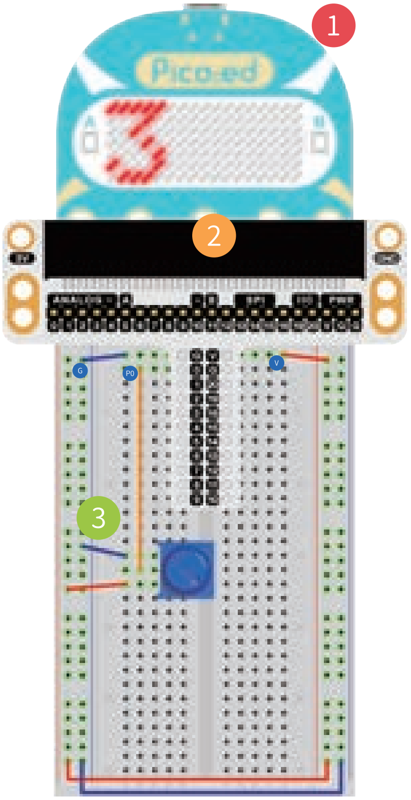
◎ Question: How to light on in red while pressing the button and light on in green after releasing it?

# 3. Trimpot

## Component List

1. 1 x Pico:ed
2. 1 x Breadboard Adapter
3. 1 x Breadboard
4. 1 x 10kΩ Trimpot





## Description

We will read the output voltage of the trimpot, and display it on the 7*17 LED screen of Pico:ed with the mapping value.

## 3. Trimpot
step

```python
import board
import picoed
import analogio
potentiometer = analogio.AnalogIn(board.P0_A0)
while True:
    picoed.display.scroll(int(potentiometer.value / 655))
```

1. Import the modules of board, picoed and analogio that we need.
2. Set the pins of the trimpot.
3. Map the analog value (0~65535) reading from pin P0_A0 to the range (0~100), and display them on the Pico:ed.

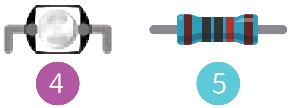
◉ Result: Rotate the trimpot, the mapping value of the voltage shall display on the 7 * 17 LED screen.

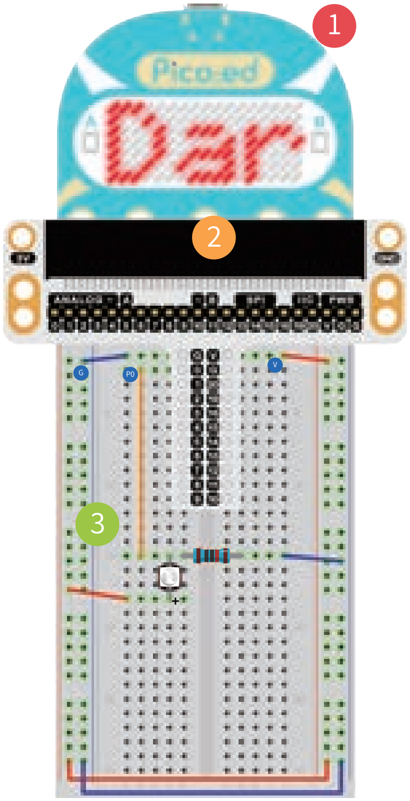
◉ Question: How to change the light level of the LEDs with the trimpot?

# 4. Photodiode

## Component List

1. 1 x Pico:ed
2. 1 x Breadboard Adapter
3. 1 x Breadboard
4. 1 x Photodiode
5. 1 x 10kΩ Resistor





## Description

We are going to display different information on Pico:ed according to the light intensity detected from the photoresistor.

## 4. Photodiode

step

```python
import board
import picoed
import analogio
import time
light = analogio.AnalogIn(board.P0_A0)
light_value = light.value
while True:
    light_new = light.value
    if light_new < light_value:
        picoed.display.scroll("Dark")
    else:
        picoed.display.scroll("Bright")
```

1. Import the modules of board, picoed, analogio and time that we need.
2. Set the pin connected to the photoresistor and read the analog voltage as the reference value of the brightness.
3. Determine whether the analog voltage value of the real-time photoresistor is less than the reference value, and display "Bright" or "Dark" according to the judgment result.

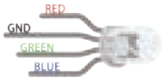
◉ Result: Show " Bright " on Pico:ed when the light is on, or show " Dark " when the light is off.

◉ Question: How to use a photoresistor to control the on and off of an LED?

# 5. RGB LED

## Component List

1. 1 x Pico:ed
2. 1 x Breadboard Adapter
3. 1 x Breadboard
4. 1 x RGB LED
5. 3 x 100Ω Resistors

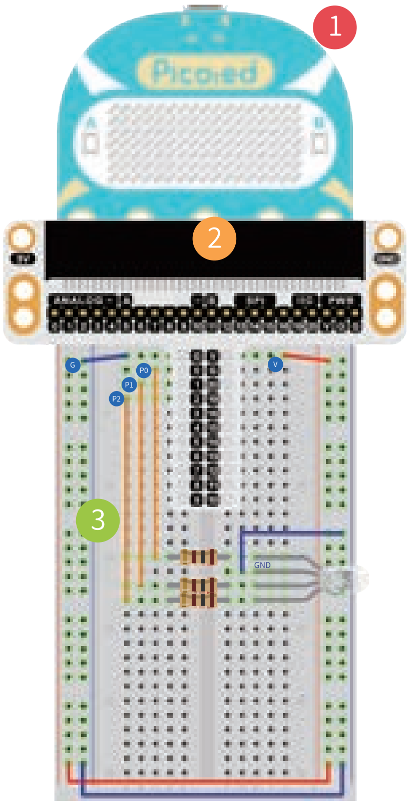


4

5

## Description

We are going to program the RGB LED to change among red, green and blue gradually.



# 5. RGB LED

step

```
import board
import digitalio
import time
from picoed import button_a, button_b
led_0 = digitalio.DigitalInOut(board.P0)
led_1 = digitalio.DigitalInOut(board.P1)
led_2 = digitalio.DigitalInOut(board.P2)
led_0.direction = digitalio.Direction.OUTPUT
led_1.direction = digitalio.Direction.OUTPUT
led_2.direction = digitalio.Direction.OUTPUT
while True:
    if button_a.is_pressed() and button_b.is_pressed():
        led_0.value = False
        led_1.value = False
        led_2.value = True
    elif button_a.is_pressed():
        led_0.value = True
        led_1.value = False
        led_2.value = False
    elif button_b.is_pressed():
        led_0.value = False
        led_1.value = True
        led_2.value = False
    else:
        led_0.value = True
        led_1.value = True
        led_2.value = True
    time.sleep(0.1)
```

1 Import the modules of board, digitalio, time and picoed that we need.

2 Set the pins and directions of the breadboard adapter connecting to the LEDs.

3 While pressing button A, set the value of led_0 as true, the led_1 and led_2 as false. In the same way, program when button B being pressed and when button A+B being pressed.

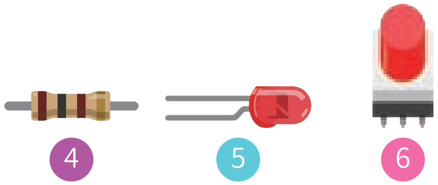
◉ Result: Press button A, the LED turns red; B for green; A+B for blue.

◉ Question: How to program to make it light on in cyan, magenta, and yellow?

# 6. Self-lock Switch

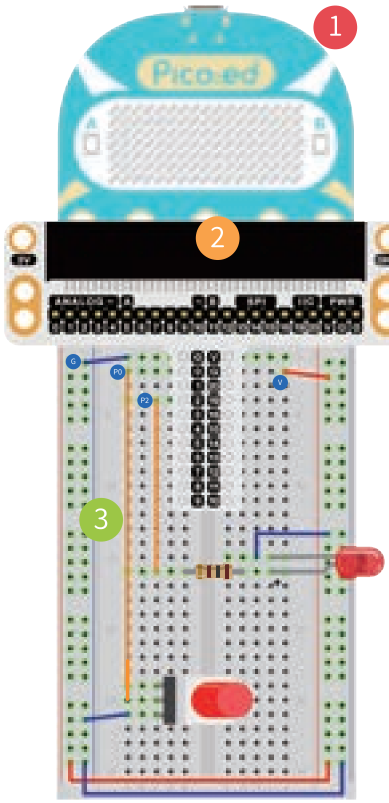
## Component List

1 1 x Pico:ed
2 1 x Breadboard Adapter
3 1 x Breadboard
4 1 x 100Ω Resistor
5 1 x LED
6 1 x Self-lock Switch



## Description

We are going to use the self-lock switch to control the on and off of the LEDs.



# 6. Self-lock Switch

step

```
import board
import digitalio
led = digitalio.DigitalInOut(board.P2_A2)
led.direction = digitalio.Direction.OUTPUT
locking = digitalio.DigitalInOut(board.P0_A0)
locking.direction = digitalio.Direction.INPUT
while True:
    led.value = locking.value
```

1. Import the modules of board and digitalio that we need.
2. Set the pins and directions of the breadboard adapter connecting to the LEDs.
3. Set the pins and directions of the adapter connecting to self-lock switch.
4. While true, set the status of the LED as the counterpart of the self-lock switch.

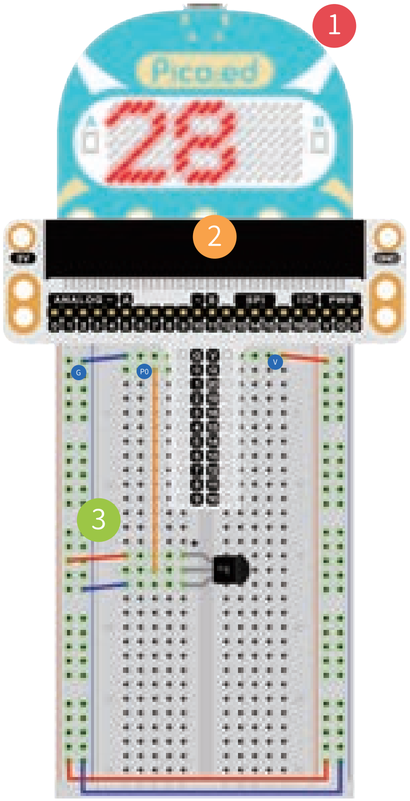
◉ Result: Press the button once to light up the LED and press again to turn it off.

◉ Question: How to use two self-lock switches to realize the function of the stair light?

# 7. Temperature Sensor

## Component List

1. 1 x Pico:ed
2. 1 x Breadboard Adapter
3. 1 x Breadboard
4. 1 x TMP36 Temperature Sensor



## Description

We are going to learn the analog TMP36 sensor to detect the temperature and display the value on Pico:ed.

# 7. Temperature Sensor

step

```
import board
import digitalio
import analogio
import picoed
temperature = analogio.AnalogIn(board.P0_A0)
while True:
    voltage = temperature.value * (3300 / 65535)
    temperature_value = (voltage - 500) / 10
    picoed.display.scroll(int(temperature_value))
```

1. Import the modules of board, digitalio, analogio and picoed that we need.
2. Set the pins of the TMP36 sensor.
3. While true, display the value detected by the TMP36 sensor on the Pico:ed.

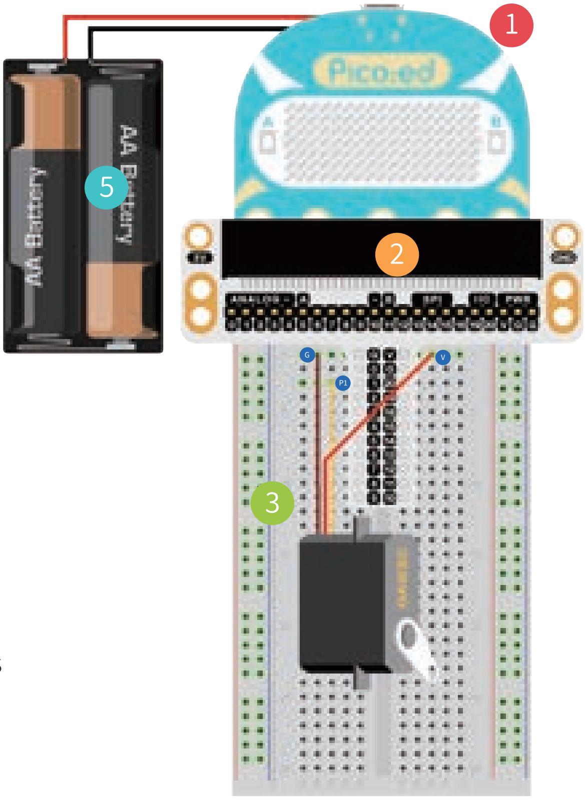
◉ Result: The current temperature value displays on the Pico:ed.

◉ Question: How to display the value of the temperature in Fahrenheit degree?

## 8. Servo

### Component List

1. 1 x Pico:ed
2. 1 x Breadboard Adapter
3. 1 x Breadboard
4. 1 x Mini Servo
5. 1 X Battery Holder



### Description

We are going to drive the servo within its rotation scopes with Pico:ed.

# 8. Servo
step

```python
import time
import board
import pwmio
from adafruit_motor import servo
pwm = pwmio.PWMOut(board.P1_A1, duty_cycle=2 ** 15, frequency=50)
my_servo = servo.Servo(pwm)
while True:
    for angle in range(0, 180, 5):
        my_servo.angle = angle
        time.sleep(0.05)
    for angle in range(180, 0, -5):
        my_servo.angle = angle
        time.sleep(0.05)
```

1 Import the modules of time, board, pwmio, and servo that we need.

2 Set the pins of the servo connecting with the breadboard and create the object for servo.

3 While true, set the servo rotate among (0,180) degrees back and forth.

◎ Result: The servo rotates among (0,180) degrees back and forth.

◎ Question: How to make a pointer thermometer with a TMP36 sensor and a servo?

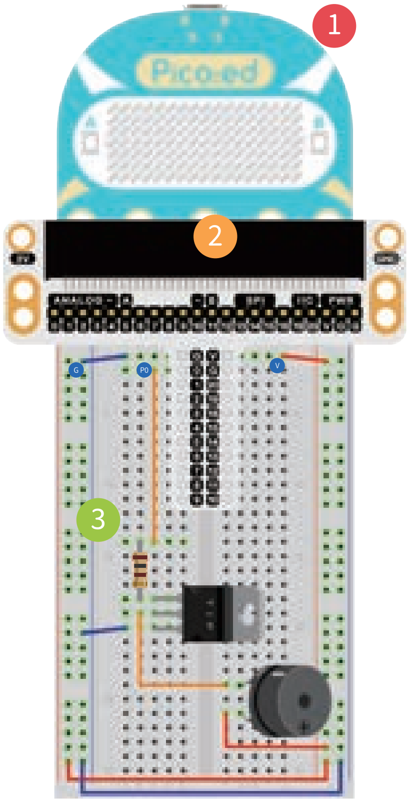
# 9. Buzzer

## Component List

1. 1 x Pico:ed
2. 1 x Breadboard Adapter
3. 1 x Breadboard
4. 1 x Buzzer
5. 1 x NPN Transistor
6. 1 x 100Ω Resistor

## Description

We are going to drive the buzzer through the Pico:ed as an alarm.



## 9. Buzzer
step

```python
import board
import pwmio
import time
piezo = pwmio.PWMOut(board.P0_A0, duty_cycle=0,
                     frequency=440, variable_frequency=True)
def play_note(note):
    piezo.frequency = note
    piezo.duty_cycle = 65535 // 2
    time.sleep(0.25)
    piezo.duty_cycle = 0
    time.sleep(0.05)
while True:
    play_note(494)
    time.sleep(0.2)
    play_note(262)
    time.sleep(0.2)
    play_note(294)
    time.sleep(0.2)
```

1 Import the modules of board, pwmio and time that we need.

2 Set the pins used for buzzer and the playing tones of the buzzer.

3 While true, play the tones.

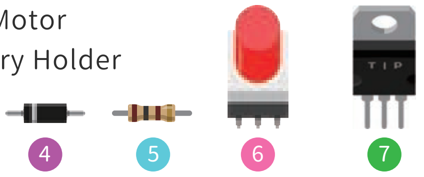
◎ Result: The buzzer alarms and repeats in the specific tones.

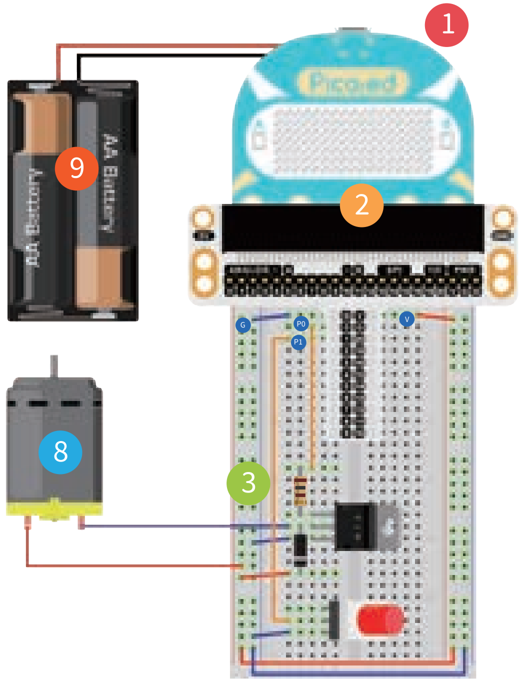
◎ Question: How to alarm for high temperature with the TMP36 sensor and the buzzer?

# 10. Motor

## Component List

1. 1 x Pico:ed
2. 1 x Breadboard Adapter
3. 1 x Breadboard
4. 1 x Diode
5. 1 x 100Ω Resistor
6. 1 x Self-lock Switch
7. 1 x NPN Transistor
8. 1 x Mini Motor
9. 1 X Battery Holder





## Description

We are going to use the self-lock switch to turn on/off the motor.

# 10. Motor

step

```python
import board
import digitalio
motor = digitalio.DigitalInOut(board.P0_A0)
locking = digitalio.DigitalInOut(board.P1_A1)
motor.direction = digitalio.Direction.OUTPUT
locking.direction = digitalio.Direction.INPUT
motor.value = True
locking.pull = digitalio.Pull.UP
while True:
    if locking.value == False:
        motor.value = True
    else:
        motor.value = False
```

1 Import the modules of board and digitalio that we need.

2 Set pins and directions of the motor and the self-lock switch, initialize the motor as True and pull up the switch.

3 While true, judge the status of the self-lock switch to control the motors.

◉ Result: Push once to start the motor and push again to stop it.

◉ Question: How to use the trimpot to control the speed of the motor?

# 11. Rainbow LED

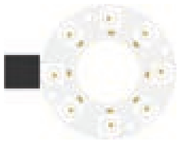
## Component List

1 1 x Pico:ed

2 1 x Breadboard Adapter

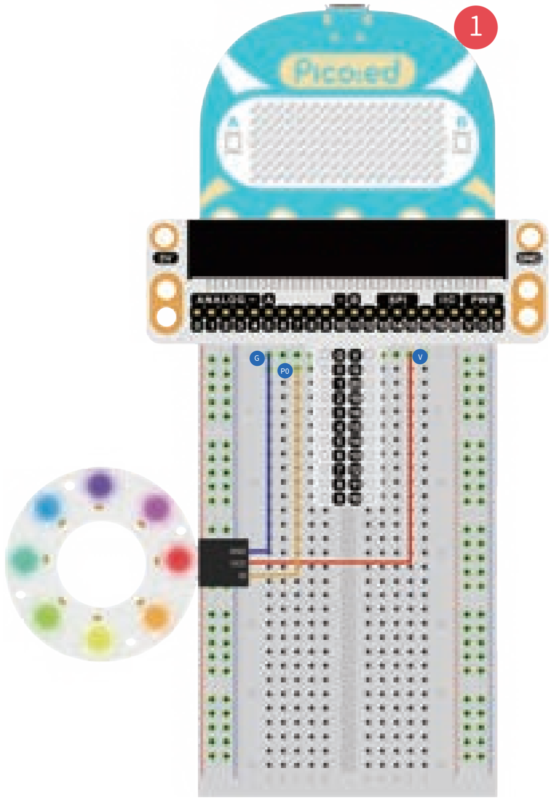
3 1 x Breadboard

4 1 x Rainbow LED(8 beads)



## Description

We are going to light on the Rainbow LED in a colorful way.



# 11. Rainbow LED

step

```python
import board
import random
import neopixel_write
import digitalio
import time
pin = digitalio.DigitalInOut(board.P0_A0)
pin.direction = digitalio.Direction.OUTPUT
list1 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
while True:
    for i in range(24):
        list1[i] = random.randint(0,255)
    pixel_off = bytearray(list1)
    neopixel_write.neopixel_write(pin, pixel_off)
    time.sleep(0.1)
```

1. Import the modules of board, random, neopixel_write, digitalio and time that we need.
2. Set the pins and directions of the Rainbow LED.
3. Initialize the list to save the value of the RGB.
4. While true, loop to change the RGB value for each bead.

◉ Result: The Rainbow LED lights on in a colorful way.

◉ Question: How to make a blink rainbow LED just like a blinking eye?

## For More Information

Please visit

https://www.elecfreaks.com/learn-en/picoed-starter

## About us

ELECFREAKS is an official Chinese Partner of micro:bit Educational Foundation focusing on developing educational and creative micro:bit accessories for the world. We devote to providing the most complete and excellent products and services to our customers. We have created tutorial blogs, learning materials, videos and fun case studies as part of building global micro:bit communities in education.

ELECFREAKS
MAKE CODING ACCESSIBLE

www.elecfreaks.com

V1.0