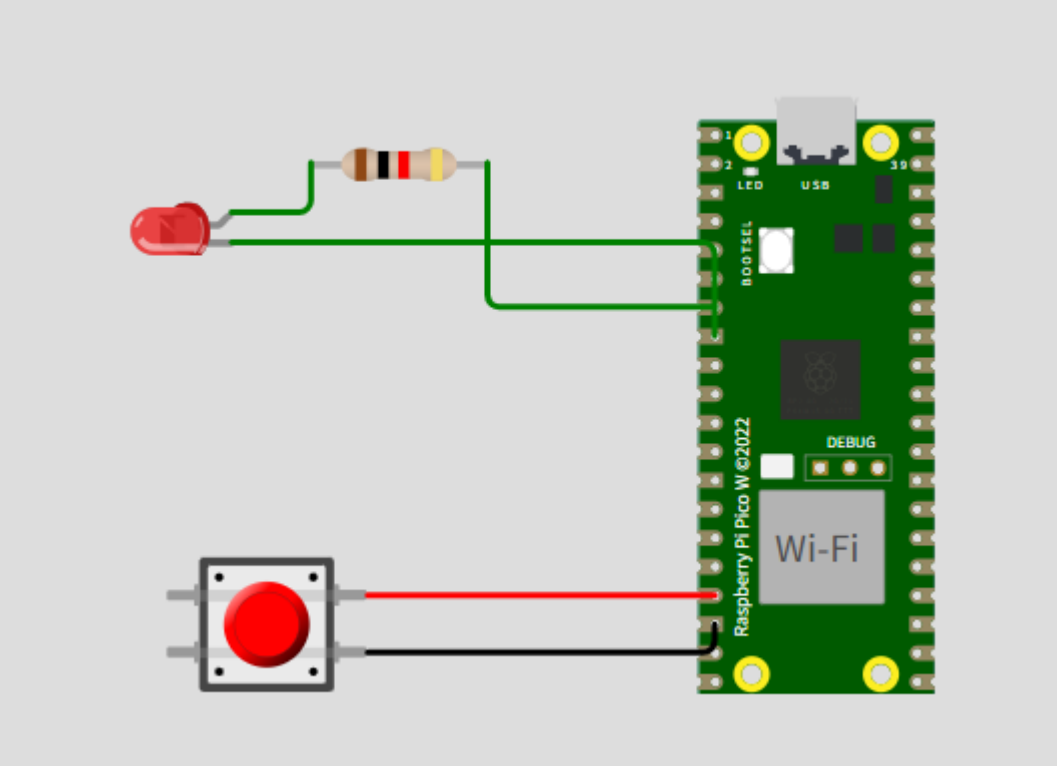
task3

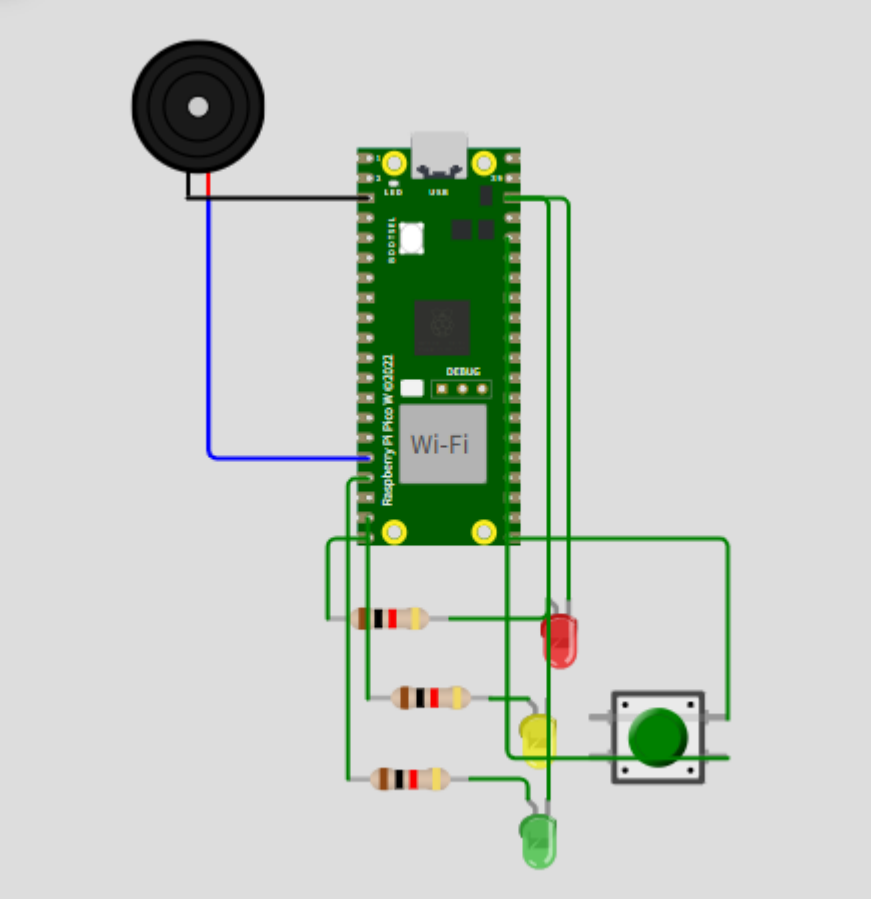
Turn the led on with a button




```
// 定义引脚名称，让代码更容易阅读
const int buttonPin = 13; // 按钮连接到 GP14 这里弄错了 得自己读取
const int ledPin = 5;    // LED 连接到 GP15

void setup() {
  // 设置 LED 引脚为输出模式
  pinMode(ledPin, OUTPUT);

  // 设置按钮引脚为输入模式，并启用内部上拉电阻
  // 内部上拉电阻会将引脚默认拉高 (HIGH)。
  // 当按钮按下时，引脚被连接到 GND (LOW)。
  pinMode(buttonPin, INPUT_PULLUP);
}
```

```
void loop() {
  // 读取按钮的状态
  int buttonState = digitalRead(buttonPin);

  // 因为我们使用了 INPUT_PULLUP，所以:
  // 如果 buttonState 是 LOW (0)，表示按钮被按下
  if (buttonState == LOW) {
    // 按钮被按下: 打开 LED
    digitalWrite(ledPin, HIGH);
  } else {
    // 按钮被释放 (buttonState 是 HIGH): 关闭 LED
    digitalWrite(ledPin, LOW);
  }

  // 小延迟，防止 CPU 过快循环，但对于这种简单的任务，可以省略或保持很小的值。
  // delay(5);
}
```

Traffic lights

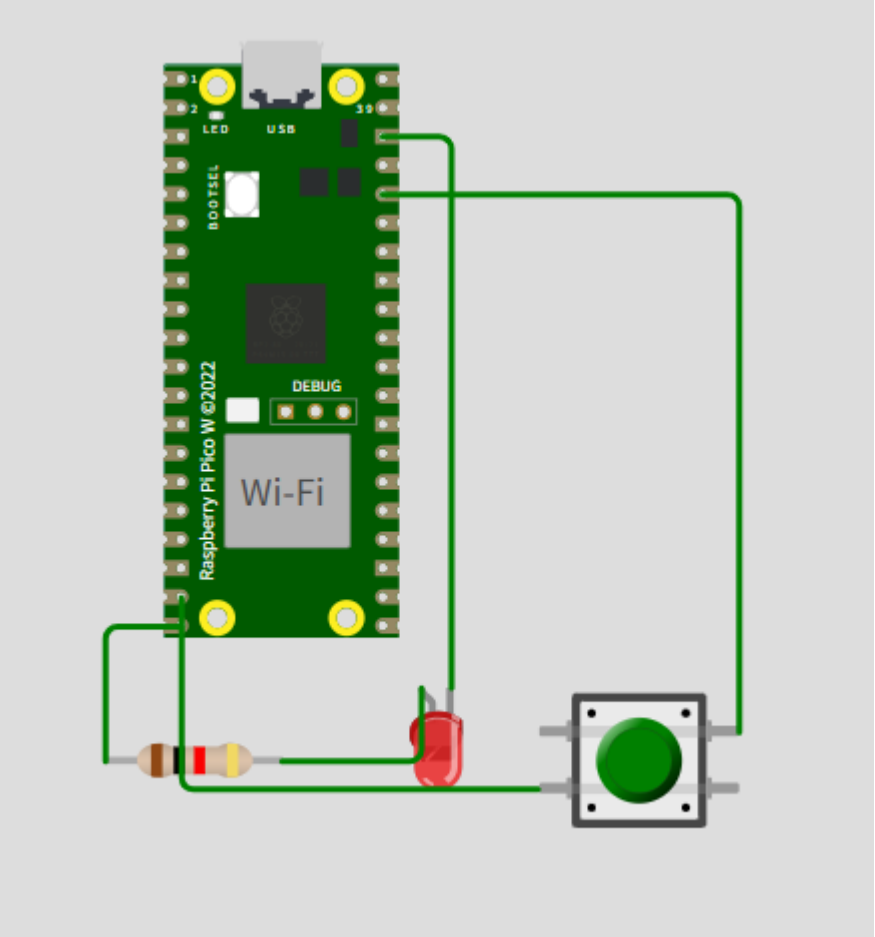



```
// --- Pin 定义 ---
const int redLED = 15;
const int yellowLED = 14;
const int greenLED = 13;

const int buttonPin = 5;     // 按钮接到 GPIO5 ，接 3.3V OUT
const int buzzerPin = 12;     // 蜂鸣器

// 时间设置（可调整）
unsigned long lightDelay = 1500;    // 每个灯亮 1.5 秒
unsigned long alertDuration = 2000; // 按住按钮后蜂鸣器持续 2 秒

void setup() {
  pinMode(redLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
```

```
  // ✅ 按钮接 3.3V —— 使用内部下拉
  pinMode(buttonPin, INPUT_PULLDOWN);

  // 先全部关闭
  digitalWrite(redLED, LOW);
  digitalWrite(yellowLED, LOW);
  digitalWrite(greenLED, LOW);
  digitalWrite(buzzerPin, LOW);
}

void loop() {

  // ✅ 如果按钮被按住（高电平 = 按下）
  if (digitalRead(buttonPin) == HIGH) {
    handleButtonAlert();
  } else {
    trafficLightLoop();
  }
}

// ------------------------------------------------------
// ✅ 正常交通灯循环
// ------------------------------------------------------
void trafficLightLoop() {
  // 绿灯
  digitalWrite(greenLED, HIGH);
  delay(lightDelay);
  digitalWrite(greenLED, LOW);

  // 黄灯
  digitalWrite(yellowLED, HIGH);
  delay(lightDelay);
  digitalWrite(yellowLED, LOW);

  // 红灯
  digitalWrite(redLED, HIGH);
  delay(lightDelay);
  digitalWrite(redLED, LOW);
```

```
}

// ------------------------------------------------------
// ✅ 按钮按住时：红灯 + 蜂鸣器报警
// ------------------------------------------------------
void handleButtonAlert() {

  // 开启红灯与蜂鸣器
  digitalWrite(redLED, HIGH);
  digitalWrite(buzzerPin, HIGH);

  delay(alertDuration);  // 持续蜂鸣器一段时间

  // 关闭蜂鸣器（红灯继续亮）
  digitalWrite(buzzerPin, LOW);

  // ✅ 等待按钮松开（防止立即恢复）
  while (digitalRead(buttonPin) == HIGH) {
    delay(10);
  }

  // 用户松手后，关闭红灯，恢复正常交通灯循环
  digitalWrite(redLED, LOW);
}
```

Interrup

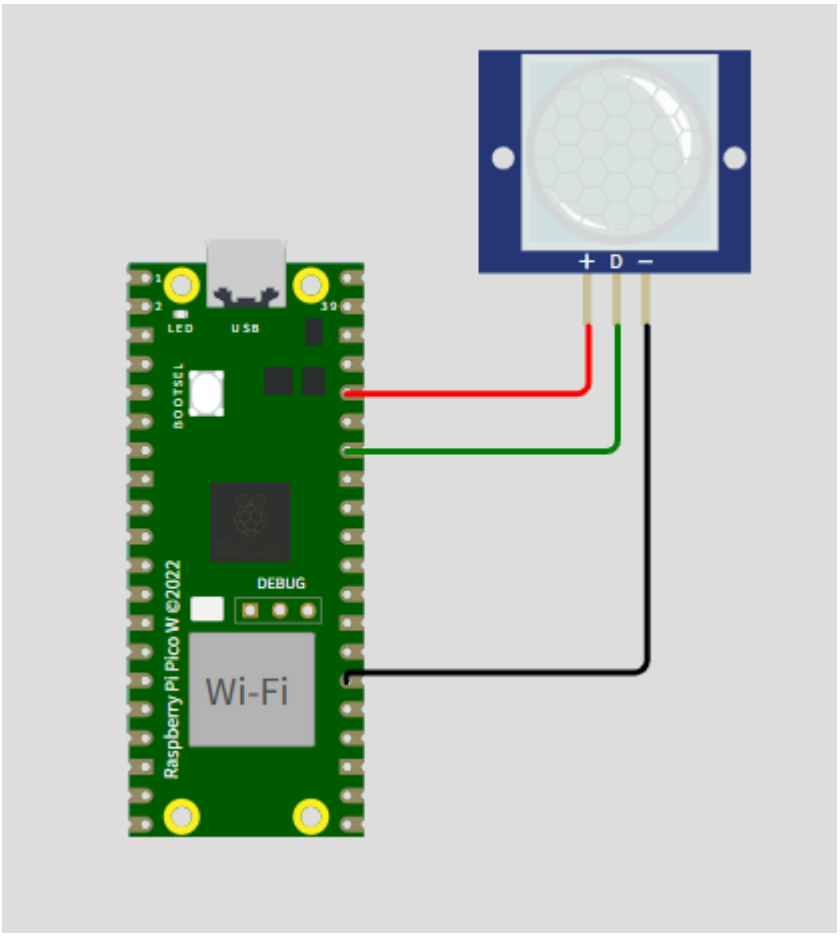



```
// ---------------------------------------------------
// Reaction Time Game for Raspberry Pi Pico
// LED on GP15, Button on GP14 (connected to 3.3V)
// ---------------------------------------------------

const int ledPin = 15;
const int buttonPin = 14;

unsigned long startTime;
unsigned long reactionTime;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLDOWN);  // button connected to 3.3V

  Serial.begin(9600);
  delay(2000);  // 给串口一点时间
```

```
  Serial.println("Reaction Game Ready!");
}

void loop() {
  Serial.println("Get Ready...");
  delay(1000);

  // ------------------------------
  // Step 1: LED 亮
  // ------------------------------
  digitalWrite(ledPin, HIGH);
  Serial.println("LED ON");

  // ------------------------------
  // Step 2: 随机等待时间（1000 - 5000 ms）
  // ------------------------------
  unsigned long randomDelayTime = random(1000, 5000);
  delay(randomDelayTime);

  // ------------------------------
  // Step 3: LED 熄灭 → 开始计时
  // ------------------------------
  digitalWrite(ledPin, LOW);
  Serial.println("LED OFF! Click NOW!");

  startTime = millis();

  // ------------------------------
  // Step 4: 等用户按按钮
  // ------------------------------
  while (digitalRead(buttonPin) == LOW) {
    // 等待用户点击
  }

  // 用户点击时间
  reactionTime = millis() - startTime;

  // ------------------------------
```

```
  // Step 5: 输出反应时间
  // ------------------------------
  Serial.print("Your reaction time: ");
  Serial.print(reactionTime);
  Serial.println(" ms");

  delay(2000); // 休息后开始下一轮
}
```

Burglary alarm



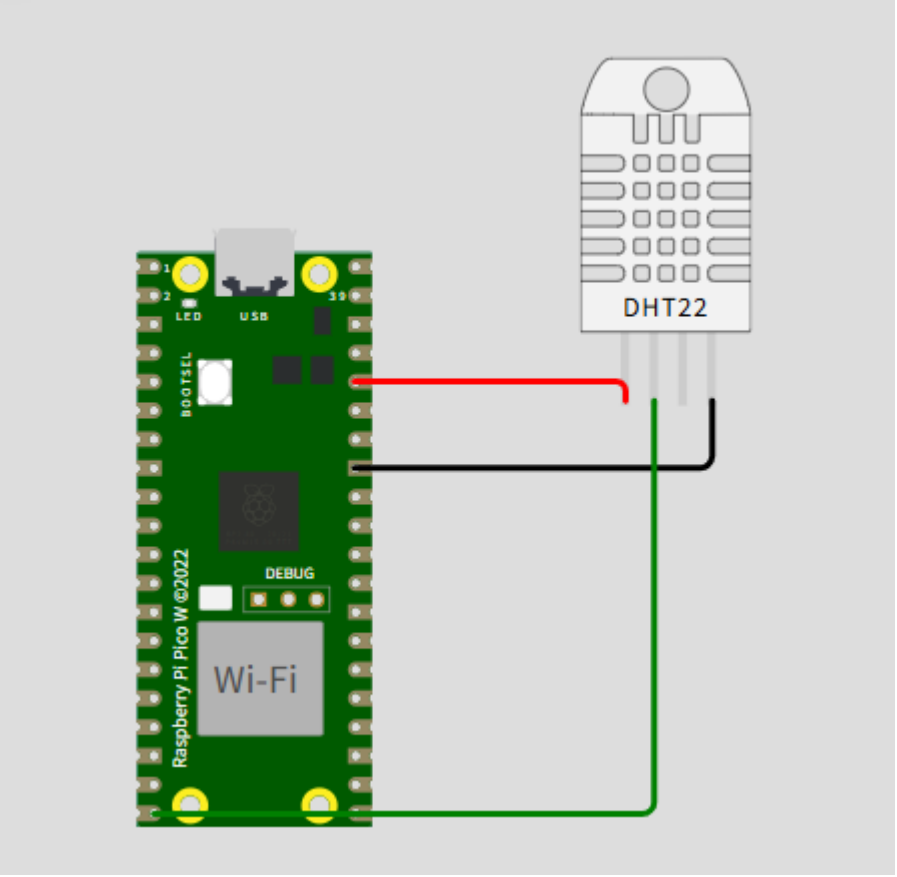

```
// Burglary alarm with PIR (Raspberry Pi Pico / Pico W - Arduino Core)
// Wiring:
// PIR VCC → 3.3V
// PIR GND → GND
// PIR OUT → GP14  (pirPin)
// Optional: Buzzer → GP15 (buzzerPin)
```

```
// Onboard LED → GP25 (ledPin)

const int pirPin = 14;      // PIR output pin
const int ledPin = 25;      // Onboard LED on Pico
const int buzzerPin = 15;   // Optional buzzer (set USE_BUZZER to true to e
nable)

const bool USE_BUZZER = false; // 改为 true 如果你接了蜂鸣器到 buzzerPin

// Alert behaviour
const unsigned long alertBlinkCount = 6;   // 闪烁次数
const unsigned long alertBlinkInterval = 200; // 每次闪烁间隔（ms）
const unsigned long buzzerOnMillis = 1000; // 蜂鸣器持续时长（ms）
const unsigned long triggerCooldown = 5000; // 触发后冷却时间（ms），防
止重复通知

// internal state
int lastPirState = LOW;
unsigned long lastTriggerTime = 0;

void setup() {
  pinMode(pirPin, INPUT); // PIR 有驱动输出，不使用上/下拉
  pinMode(ledPin, OUTPUT);
  if (USE_BUZZER) pinMode(buzzerPin, OUTPUT);

  digitalWrite(ledPin, LOW);
  if (USE_BUZZER) digitalWrite(buzzerPin, LOW);

  Serial.begin(115200);
  delay(50);
  Serial.println();
  Serial.println("PIR Alarm starting...");
  Serial.println("Warming up PIR... please wait a moment.");

  // PIR module warm-up time (可根据模块需要调整)
  unsigned long warmup = 30000; // 30 seconds
  unsigned long t0 = millis();
  while (millis() - t0 < warmup) {
```

```
    // 在暖机期间简单闪烁一次板载灯提示进度（非阻塞）
    digitalWrite(ledPin, (millis() / 500) % 2);
    delay(50);
  }
  digitalWrite(ledPin, LOW);
  Serial.println("PIR ready. Monitoring for movement...");
}

void loop() {
  int pirState = digitalRead(pirPin);
  unsigned long now = millis();

  // 检测从 LOW → HIGH 的上升沿（新检测到移动）
  if (pirState == HIGH && lastPirState == LOW) {
    // 检查冷却时间，避免在短时间内重复警告
    if (now - lastTriggerTime >= triggerCooldown) {
      lastTriggerTime = now;
      onMotionDetected();
    }
  }

  lastPirState = pirState;

  // 其余循环可做其他事情（目前仅检测）
  delay(10); // 小延时，避免 CPU 全占用
}

void onMotionDetected() {
  unsigned long t = millis();
  Serial.print("[ALERT] Motion detected! Time (ms): ");
  Serial.println(t);

  // LED 闪烁提示
  for (unsigned long i = 0; i < alertBlinkCount; ++i) {
    digitalWrite(ledPin, HIGH);
    delay(alertBlinkInterval);
    digitalWrite(ledPin, LOW);
    delay(alertBlinkInterval);
```

```
  }

  // 蜂鸣器提示（如果启用）
  if (USE_BUZZER) {
    digitalWrite(buzzerPin, HIGH);
    delay(buzzerOnMillis);
    digitalWrite(buzzerPin, LOW);
  }

  Serial.println("Alert sequence finished. Waiting for next trigger...");
}
```

Weather station




```
#include "DHT.h" // 包含 DHT 传感器库
#include <Adafruit_Sensor.h> // 包含 Adafruit Unified Sensor 库

// 定义 DHT 传感器的连接引脚
```

```
#define DHTPIN 0

// 定义您使用的传感器类型（此处是 DHT22）
#define DHTTYPE DHT22

// 初始化 DHT 传感器对象
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  // 启动串行通信，用于输出读数
  Serial.begin(115200);
  Serial.println("--- Raspberry Pi Pico W 气象站 ---");

  // 启动 DHT 传感器
  dht.begin();
}

void loop() {
  // DHT22 传感器在两次读数之间需要至少 2 秒的间隔。
  // 我们使用 3 秒的延迟来确保稳定性。
  delay(3000);

  // --- 读取湿度 ---
  // Read humidity (in percent)
  float h = dht.readHumidity();

  // --- 读取温度 ---
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();

  // Optional: Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);

  // 检查读数是否成功
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("❌ 错误：无法从 DHT 传感器读取数据！");
    return;
  }
```

```
  // --- 计算热指数（Heat Index） ---
  // 计算热指数，单位为摄氏度
  float hic = dht.computeHeatIndex(t, h);

  // 计算热指数，单位为华氏度
  float hif = dht.computeHeatIndex(f, h);

  // --- 打印结果 ---
  Serial.print("湿度: ");
  Serial.print(h);
  Serial.println(" %");

  Serial.print("温度: ");
  Serial.print(t);
  Serial.print(" *C (");
  Serial.print(f);
  Serial.println(" *F)");

  Serial.print("体感温度 (热指数): ");
  Serial.print(hic);
  Serial.print(" *C (");
  Serial.print(hif);
  Serial.println(" *F)");

  Serial.println("--------------------");
}
```