**Project 34：IR Control Sound and LED**

1. **项目介绍：**

红外遥控是一种低成本、易于使用的无线通信技术。IR光与可见光非常相似，除了它的波长稍长一点。这意味着红外线是人眼无法检测到的，这对于无线通信来说是完美的。例如，当你按下电视遥控器上的一个按钮时，一个红外LED会以每秒38000次的频率反复开关，将信息(如音量或频道控制)传送到电视上的红外感光器。

我们将首先解释常见的红外通信协议是如何工作的。然后我们将从一个遥控器和一个红外接收组件开始这个项目。

1. **项目元件：**

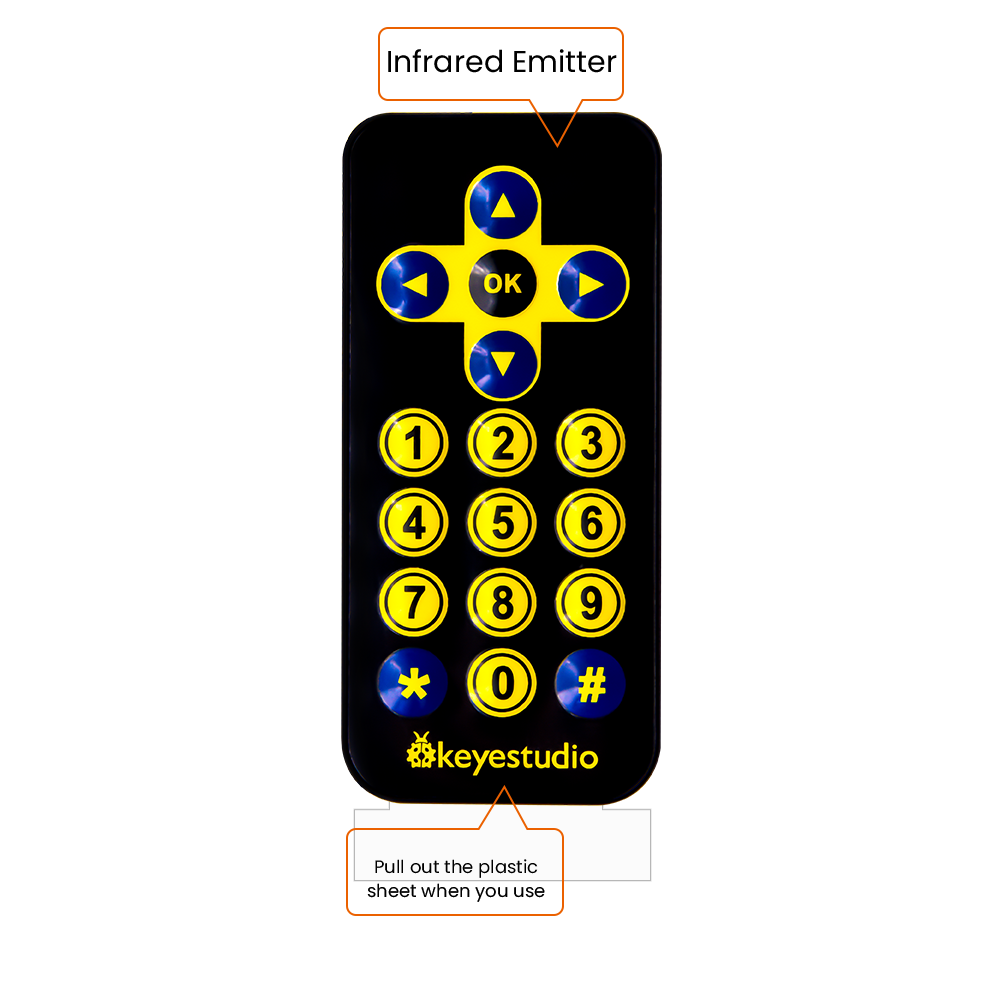
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 17a6d7f241a04d4e932cb06b758197c4 | _DSC2552 | |  | 图层 2 拷贝 2 |  |
| Raspberry Pi Pico\*1 | Raspberry Pi Pico扩展板\*1 | | 红外接收器 \*1 | RGB LED\*1 | 220Ω电阻\*3 |
|  |  | H4QG0GNSDN2S4]TSS)6UP4J | 10k欧 |  |  |
| 红外遥控器\*1 | 面包板\*1 | 无源蜂鸣器\*1 | 10KΩ电阻\*1 | 跳线若干 | USB 线\*1 |

1. **元件知识：**

**红外(IR)遥控器：**是一种具有一定数量按钮的设备。按下不同的按钮会使位于遥控器前端的红外发射管以不同的编码发送红外信号。红外遥控技术应用广泛，如电视、空调等。因此，在当今科技发达社会，红外遥控技术使你切换电视节目和调节空调温度都很方便。

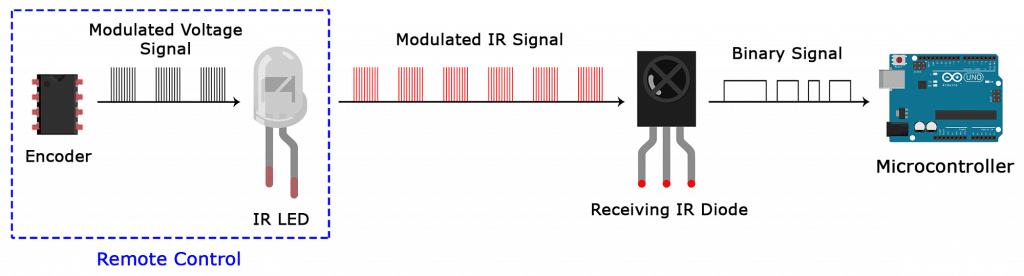
我们使用的遥控器如下所示：

该红外遥控器采用NEC编码，信号周期为110ms。



**红外(IR)接收器：**它是一种元件，可以接收红外光，所以可以用它来检测红外遥控器发出的红外光信号。红外接收器解调接收到的红外光信号，并将其转换回二进制，然后将信息传递给微控制器。

**红外信号调制过程图：**



**NEC Infrared communication protocol：**

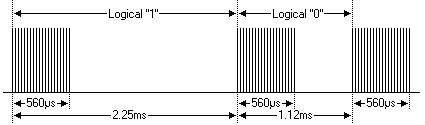
**NEC Protocol**

To my knowledge the protocol I describe here was developed by NEC (Now Renesas). I've seen very similar protocol descriptions on the internet, and there the protocol is called Japanese Format.   
I do admit that I don't know exactly who developed it. What I do know is that it was used in my late VCR produced by Sanyo and was marketed under the name of Fisher. NEC manufactured the remote control IC.   
This description was taken from my VCR's service manual. Those were the days, when service manuals were filled with useful information!

**Features**

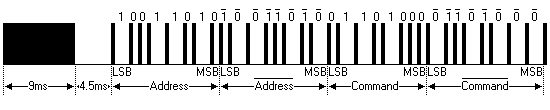
* 8 bit address and 8 bit command length.
* Extended mode available, doubling the address size.
* Address and command are transmitted twice for reliability.
* Pulse distance modulation.
* Carrier frequency of 38kHz.
* Bit time of 1.125ms or 2.25ms.

**Modulation**

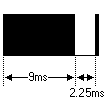


The NEC protocol uses pulse distance encoding of the bits. Each pulse is a 560µs long 38kHz carrier burst (about 21 cycles). A logical "1" takes 2.25ms to transmit, while a logical "0" is only half of that, being 1.125ms. The recommended carrier duty-cycle is 1/4 or 1/3

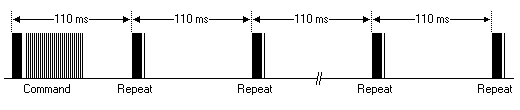
**Protocol**



The picture above shows a typical pulse train of the NEC protocol. With this protocol the LSB is transmitted first. In this case Address $59 and Command $16 is transmitted. A message is started by a 9ms AGC burst, which was used to set the gain of the earlier IR receivers. This AGC burst is then followed by a 4.5ms space, which is then followed by the Address and Command. Address and Command are transmitted twice. The second time all bits are inverted and can be used for verification of the received message. The total transmission time is constant because every bit is repeated with its inverted length. If you're not interested in this reliability you can ignore the inverted values, or you can expand the Address and Command to 16 bits each!   
Keep in mind that one extra 560µs burst has to follow at the end of the message in order to be able to determine the value of the last bit.



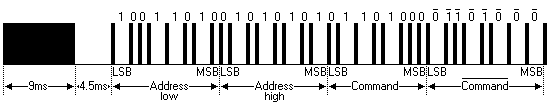
A command is transmitted only once, even when the key on the remote control remains pressed. Every 110ms a repeat code is transmitted for as long as the key remains down. This repeat code is simply a 9ms AGC pulse followed by a 2.25ms space and a 560µs burst.



**Extended NEC protocol**

The NEC protocol is so widely used that soon all possible addresses were used up. By sacrificing the address redundancy the address range was extended from 256 possible values to approximately 65000 different values. This way the address range was extended from 8 bits to 16 bits without changing any other property of the protocol.   
By extending the address range this way the total message time is no longer constant. It now depends on the total number of 1's and 0's in the message. If you want to keep the total message time constant you'll have to make sure the number 1's in the address field is 8 (it automatically means that the number of 0's is also 8). This will reduce the maximum number of different addresses to just about 13000.

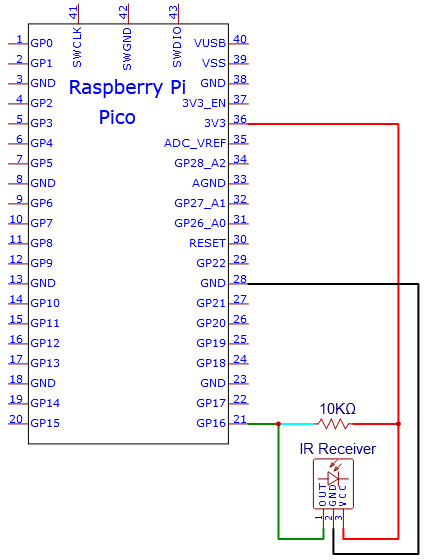
The command redundancy is still preserved. Therefore each address can still handle 256 different commands.

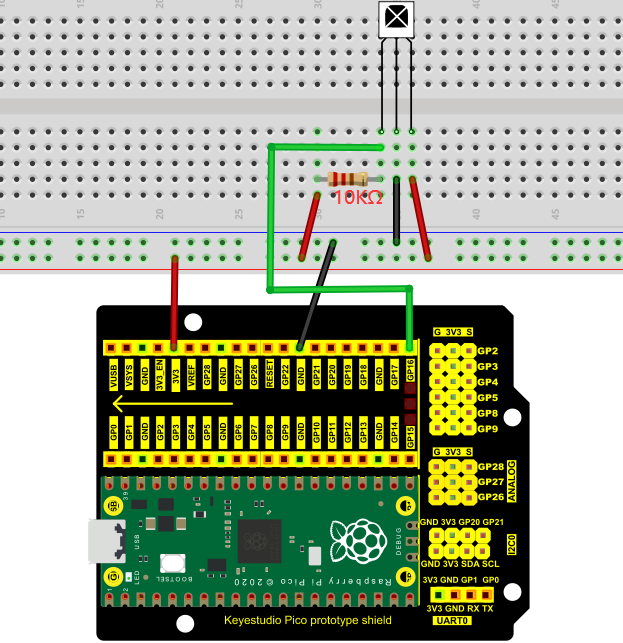


Keep in mind that 256 address values of the extended protocol are invalid because they are in fact normal NEC protocol addresses. Whenever the low byte is the exact inverse of the high byte it is not a valid extended address.

1. **解码红外信号：**

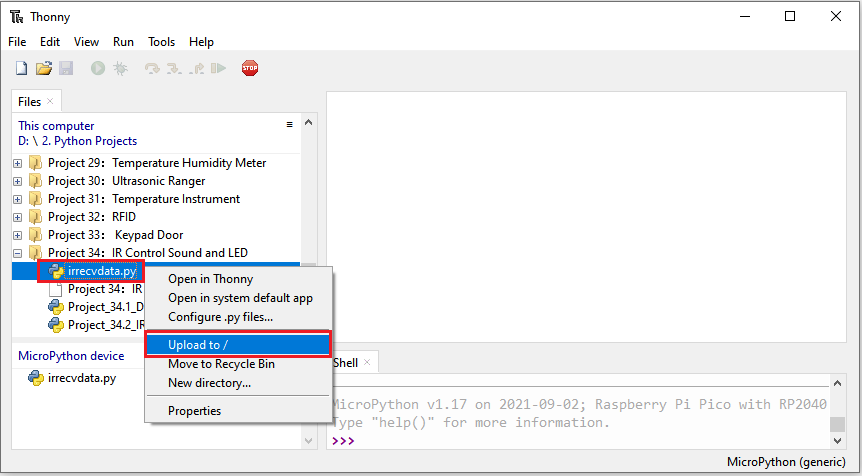
我们按照下面接线图将红外接收元件连接到Raspberry Pi Pico。

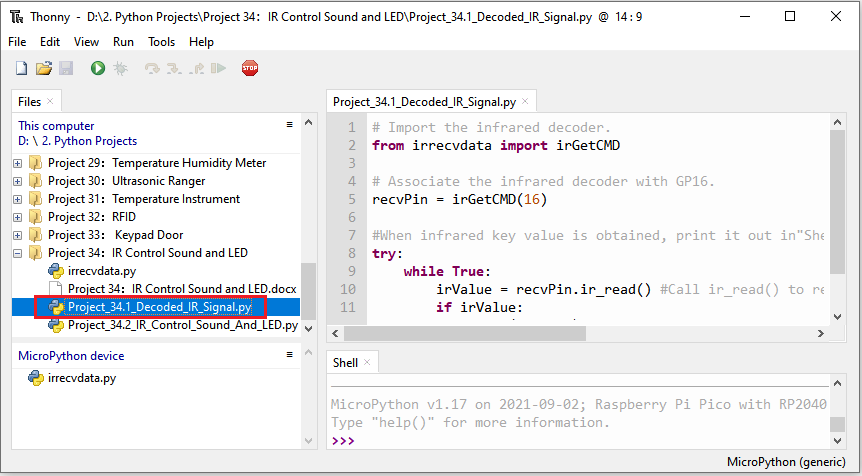




本项目中使用的代码保存在文件夹KS3020 Keyestudio Raspberry Pi Pico Learning Kit Ultimate Edition\2. Windows System\1. Python\_Tutorial\2. Python Projects\Project 34：IR Control Sound and LED。你可以把代码移到任何地方。例如，我们将代码保存在Disk(D)中，路径为D:\2. Python Projects。

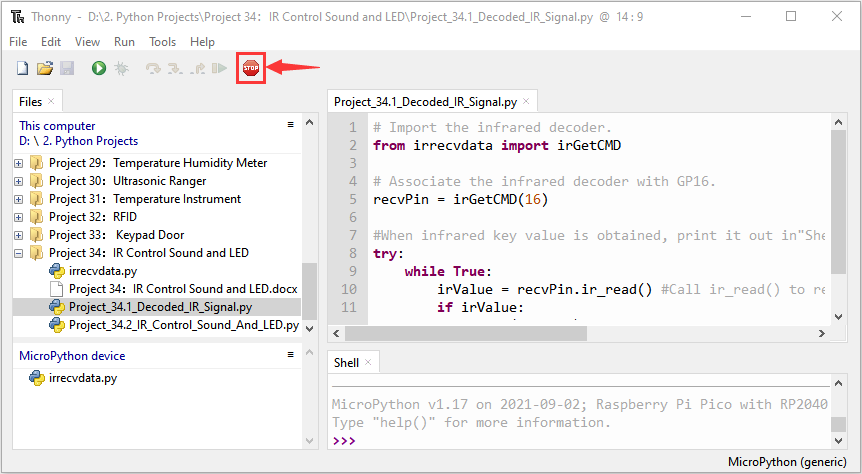
打开“Thonny”软件，点击“This computer”→“D:”→“2. Python Projects”→“Project 34：IR Control Sound and LED”。选择“irrecvdata.py”，右键单击选择 “Upload to /”,等待“irrecvdata.py”被上传到Raspberry Pi Pico，然后鼠标左键双击“Project\_34.1\_Decoded\_IR\_Signal.py”。



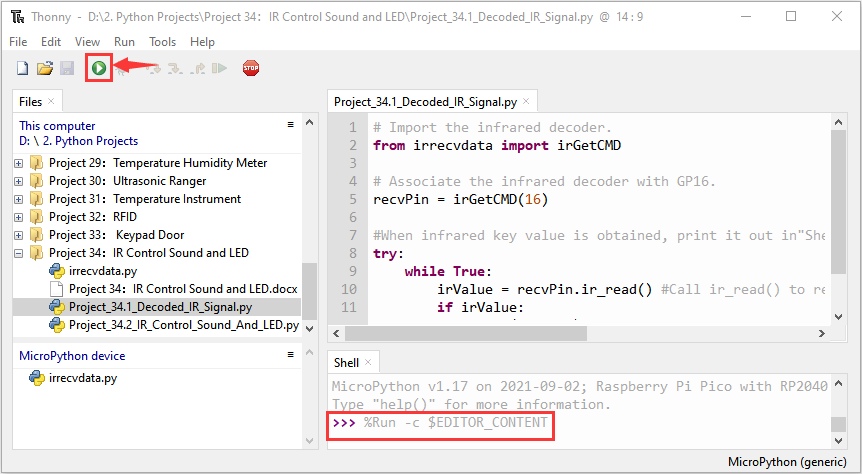


|  |
| --- |
| # Import the infrared decoder.  from irrecvdata import irGetCMD  # Associate the infrared decoder with GP16.  recvPin = irGetCMD(16)  #When infrared key value is obtained, print it out in"Shell".  try:  while True:  irValue = recvPin.ir\_read() #Call ir\_read() to read the value of the pressed key and assign it to IRValue.  if irValue:  print(irValue)  except:  pass |

确保Raspberry Pi Pico已经连接到电脑上，单击“Stop/Restart backend”。

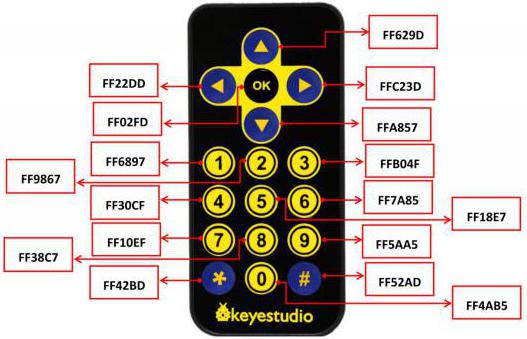


单击“Run current script”，代码开始执行，你会看到的现象是：将红外遥控器发射器对准红外接收头，按下红外控制器上的按键，Thonny IDE下的”Shell”窗口将打印当前接收到的按键编码值。按“Ctrl+C”或单击“Stop/Restart backend”退出程序。

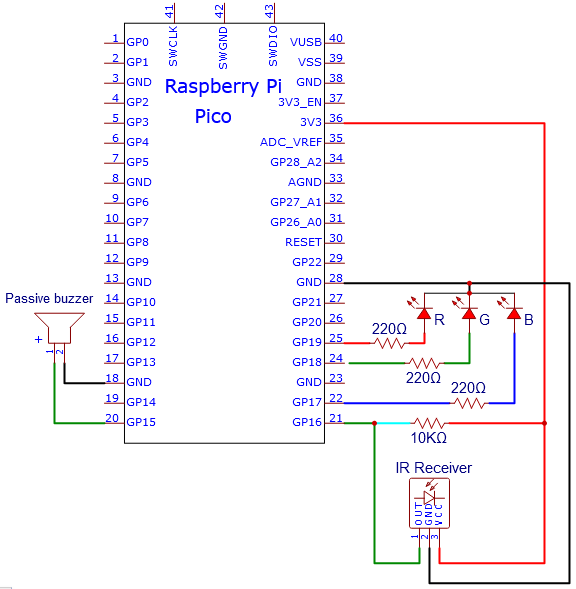


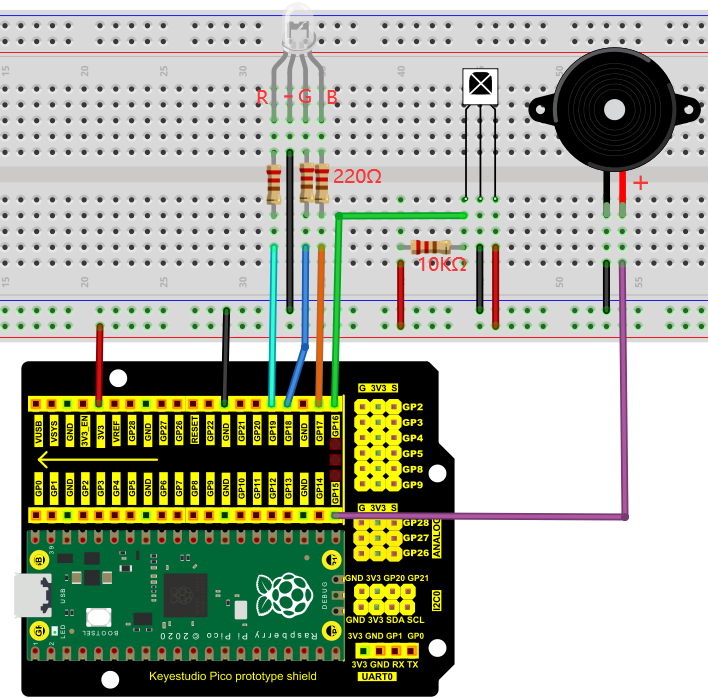


写下红外遥控器与每个按键相关联的按键编码值，因为你稍后将需要这些信息。



**5. 红外遥控的电路图和接线图：**

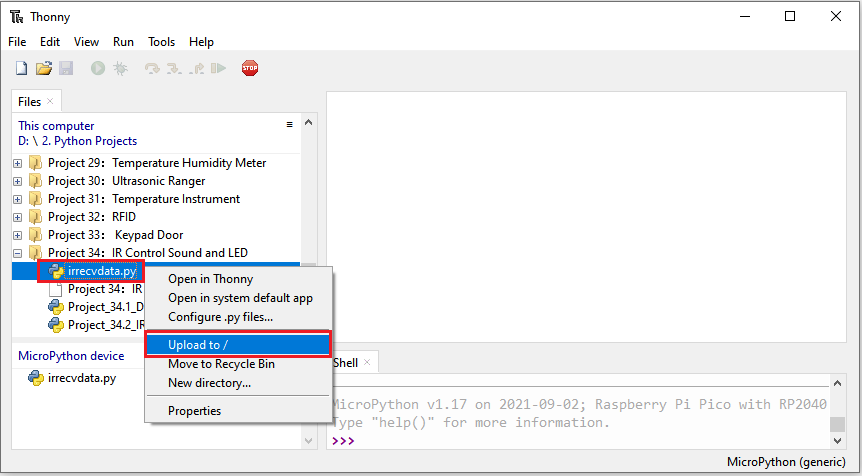


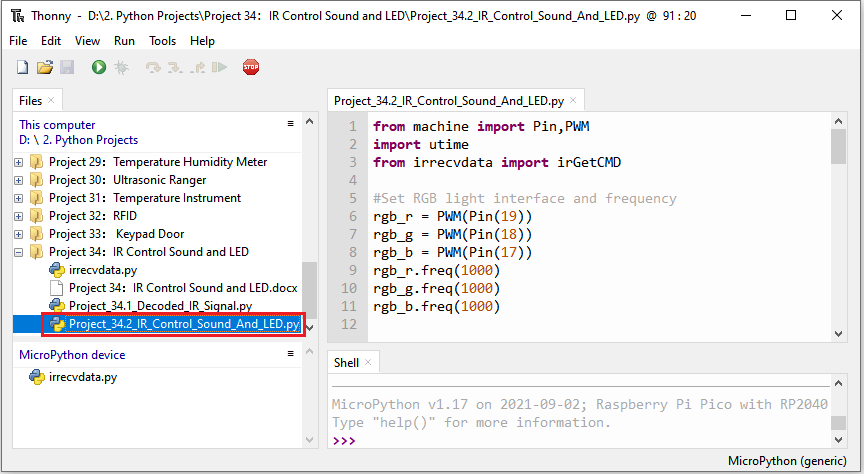


**6.项目代码：**

本项目中使用的代码保存在文件夹KS3020 Keyestudio Raspberry Pi Pico Learning Kit Ultimate Edition\2. Windows System\1. Python\_Tutorial\2. Python Projects\Project 34：IR Control Sound and LED。你可以把代码移到任何地方。例如，我们将代码保存在Disk(D)中，路径为D:\2. Python Projects。

打开“Thonny”软件，点击“This computer”→“D:”→“2. Python Projects”→“Project 34：IR Control Sound and LED”。选择“irrecvdata.py”，右键单击选择 “Upload to /”,等待“irrecvdata.py”被上传到Raspberry Pi Pico，然后鼠标左键双击“Project\_34.2\_IR\_Control\_Sound\_And\_LED.py”。

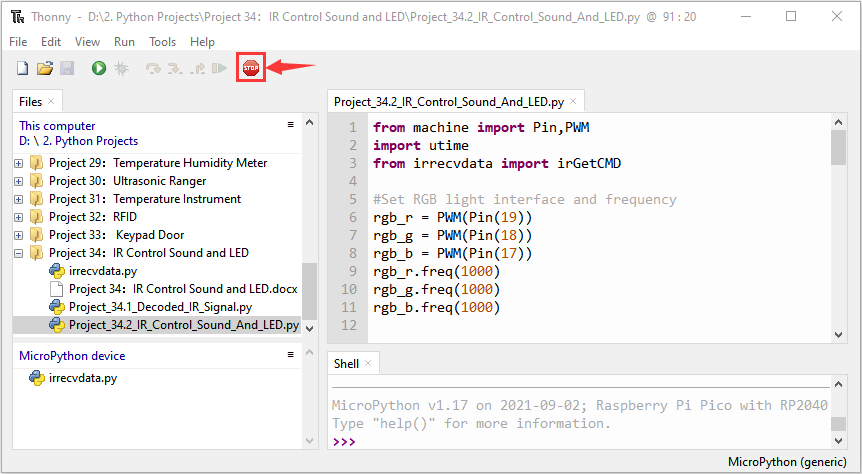


****

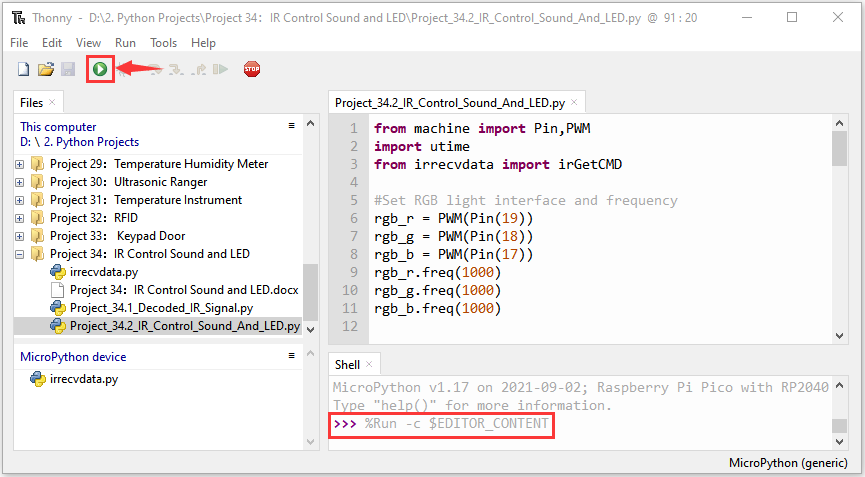
|  |
| --- |
| from machine import Pin,PWM  import utime  from irrecvdata import irGetCMD  #Set RGB light interface and frequency  rgb\_r = PWM(Pin(19))  rgb\_g = PWM(Pin(18))  rgb\_b = PWM(Pin(17))  rgb\_r.freq(1000)  rgb\_g.freq(1000)  rgb\_b.freq(1000)  # Initialize the buzzer pin to PWM function  buzzer=PWM(Pin(15, Pin.OUT))  buzzer.freq(262)  buzzer.duty\_u16(0)  # Play the frequency of midrange tones 1-7  freq = [262, 294, 330, 350, 393, 441, 495]  #Configure infrared receiving pin and library  recvPin = irGetCMD(16)  # Set the buzzer to emit different tones.  # index=[0-7], where 0 is closed, and 1-7 respectively represent middle C, middle D, middle E, middle F, middle G, middle A, middle B.  # time represents the function delay time (a positive integer), in milliseconds.  # auto\_off indicates whether the buzzer will be turned off automatically after the delay time.  def tone(index, time=0, auto\_off=False):  if index == 0:  buzzer.duty\_u16(0)  utime.sleep\_ms(time)  elif index >= 1 and index <= 7:  tone\_freq = freq[int(index - 1)]  buzzer.freq(tone\_freq)  buzzer.duty\_u16(32768)  utime.sleep\_ms(time)  if auto\_off == True:  buzzer.duty\_u16(0)  # print("----freq:", index, tone\_freq)  else:  print("Tones must be 0-7")    delay = 0    tone(1, 100, True)  while True:  irValue = recvPin.ir\_read() # Read remote control data  # Determine whether there is a button that meets the needs  if irValue:  print(irValue)  if irValue == '0xff6897': #1  rgb\_r.duty\_u16(65535)  rgb\_g.duty\_u16(0)  rgb\_b.duty\_u16(0)  tone(1, delay)  elif irValue == '0xff9867': #2  rgb\_r.duty\_u16(0)  rgb\_g.duty\_u16(65535)  rgb\_b.duty\_u16(0)  tone(2, delay)  elif irValue == '0xffb04f': #3  rgb\_r.duty\_u16(0)  rgb\_g.duty\_u16(0)  rgb\_b.duty\_u16(65535)  tone(3, delay)  elif irValue == '0xff30cf': #4  rgb\_r.duty\_u16(65535)  rgb\_g.duty\_u16(65535)  rgb\_b.duty\_u16(0)  tone(4, delay)  elif irValue == '0xff18e7': #5  rgb\_r.duty\_u16(65535)  rgb\_g.duty\_u16(0)  rgb\_b.duty\_u16(65535)  tone(5, delay)  elif irValue == '0xff7a85': #6  rgb\_r.duty\_u16(0)  rgb\_g.duty\_u16(65535)  rgb\_b.duty\_u16(65535)  tone(6, delay)  elif irValue == '0xff10ef': #7  rgb\_r.duty\_u16(65535)  rgb\_g.duty\_u16(65535)  rgb\_b.duty\_u16(65535)  tone(7, delay)  else:  rgb\_r.duty\_u16(0)  rgb\_g.duty\_u16(0)  rgb\_b.duty\_u16(0)  tone(0) |

1. **项目现象：**

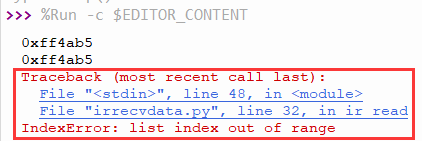
确保Raspberry Pi Pico已经连接到电脑上，单击“Stop/Restart backend”。



单击“Run current script”，代码开始执行，你会看到的现象是：按红外遥控器的1~7键，可以听到do、re、mi、fa、sol、la、si 等蜂鸣器的声音，同时RGB分别亮红灯，绿灯，蓝灯，黄灯，洋红灯，蓝绿灯，白灯。按其他另一按键（除1-7键以外），蜂鸣器就停止播放，RGB熄灭。按“Ctrl+C”或单击“Stop/Restart backend”退出程序。



**特别注意**：当代码在运行时有时候会出现以下类似提示语，只要鼠标先单击“Stop/Restart backend”，然后再单击“Run current script”就可以使代码重新运行。



(注意:在使用前，我们需要将红外遥控器底部的塑料片去掉**)**

