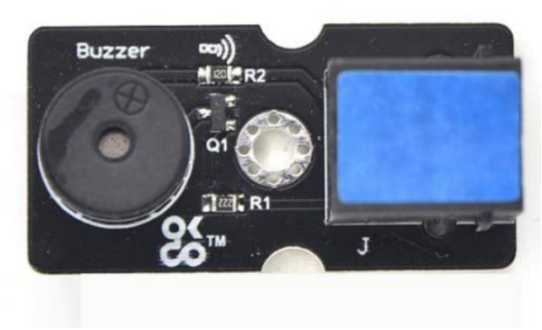
Passive Buzzer Module (000x0000 Article Number) (TS2131)

DESIGN THE WORLD

Product Details

This is the TelePort passive buzzer module which contains a 3.3-5V passive buzzer component. It doesn't carry an oscillation circuit. Therefore, you can change the frequency of PWM to produce different sounds. Additionally, it can serve as an alarm.



Features and Benefits

- Compatible with RJ11 6P6C OKdo TelePort Control boards and expansion shields.
- Variable frequency buzzer based on input signal.

Technical Specifications

Sensor type	Digital output
Working voltage	3.3V-5V
Dimensions	39mm*20mm*18mm
Weight	6g

Applications

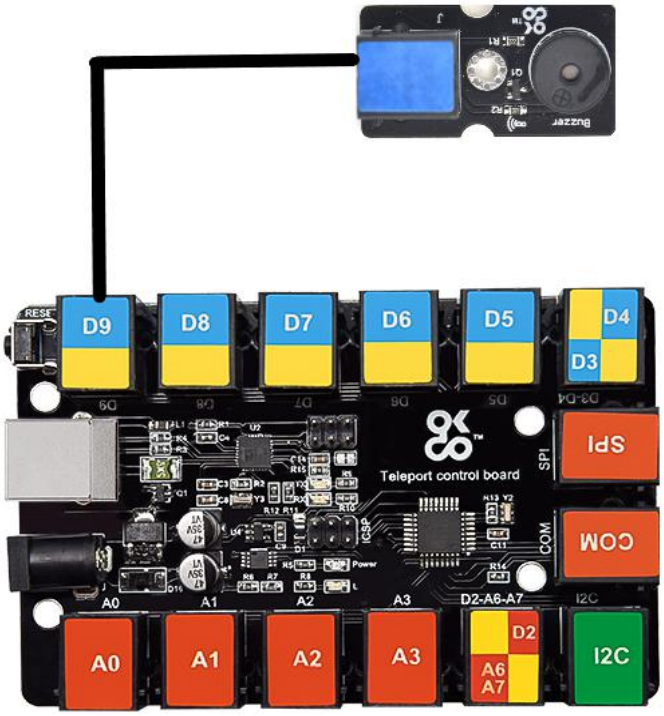
- Alarm
- Music Box

Comparison

Product	Specifications	Features
Active Buzzer	oscillating circuit sound output: >85 dB	require only DC voltage, loud sound
Passive Buzzer	Resonant frequency: 2000Hz sound output: >80 dB	require AC signal, pleasant sound

This module is compatible with the TS2180-Raspberry Pi shield, the TS2179-Micro:bit shield and the TS2178-TelePort main board.

➢ **Arduino Application**



This module is compatible with the TS2178 TelePort control board.

Test Code

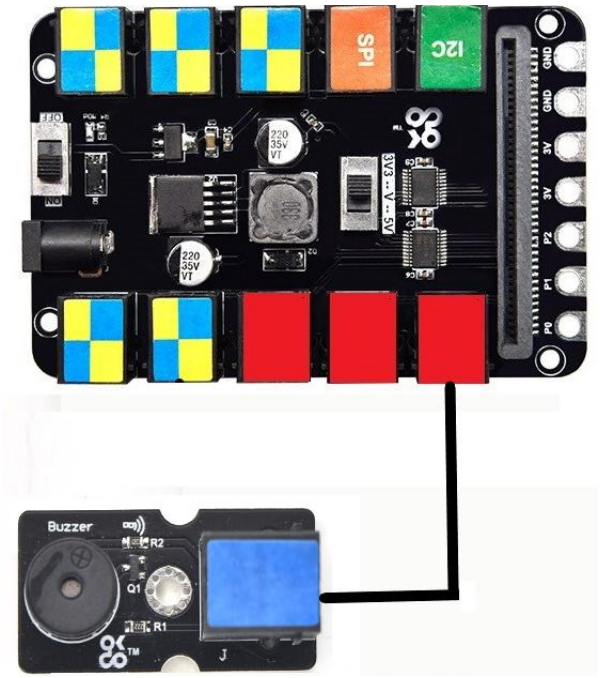
```
int buzzer=9;         //define the digital port 9
void setup()
{
pinMode(buzzer,OUTPUT);//set buzzer to output
}
void loop()
{
unsigned char i,j;//define variable i，  j
while(1)
{
for(i=0;i<80;i++)// output a sound
{
digitalWrite(buzzer,HIGH);
delay(1);//delay in 1ms
digitalWrite(buzzer,LOW);
delay(1);//delay in 1ms
}
for(i=0;i<100;i++)//  output another sound
{
digitalWrite(buzzer,HIGH);
delay(2);//delay in 2ms
digitalWrite(buzzer,LOW);
delay(2);//delay in 2ms
}
}
}
```

Test Result

It doesn't have the oscillating circuit; therefore only square waves can drive it. In the experiment, we input square waves with two frequencies, making the buzzer emit different sounds.

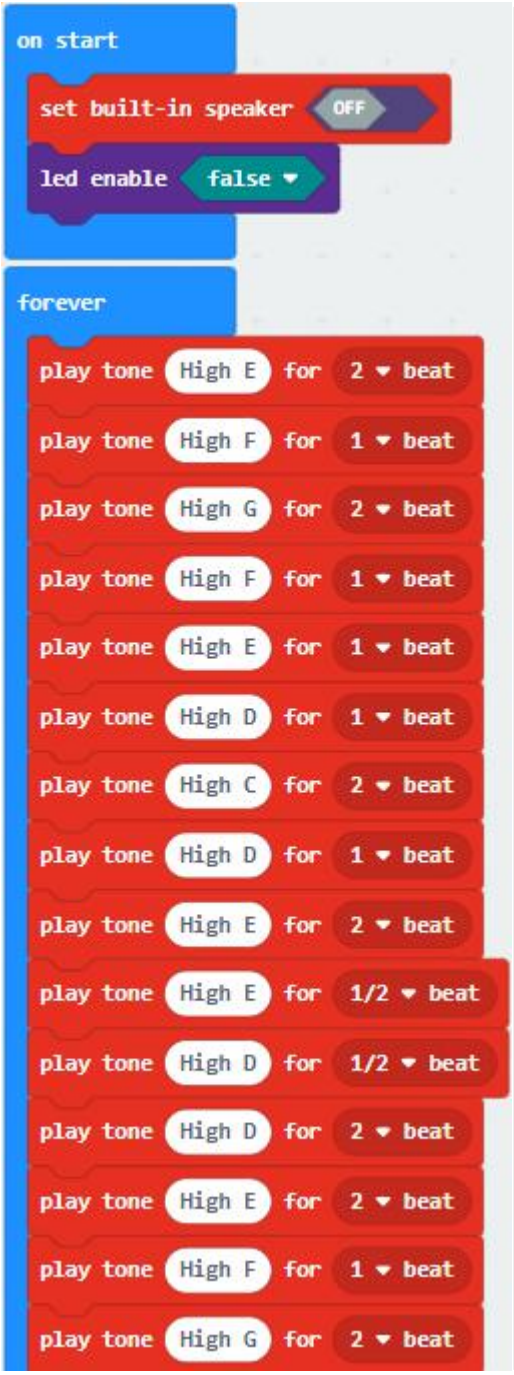
If you want to know more details about Arduino and the TelePort control board, you can refer to TS2178.

➢ Micro:bit Application



It is compatible with the Micro:bit board and the TS2179 Micro:bit expansion board.

Test Code



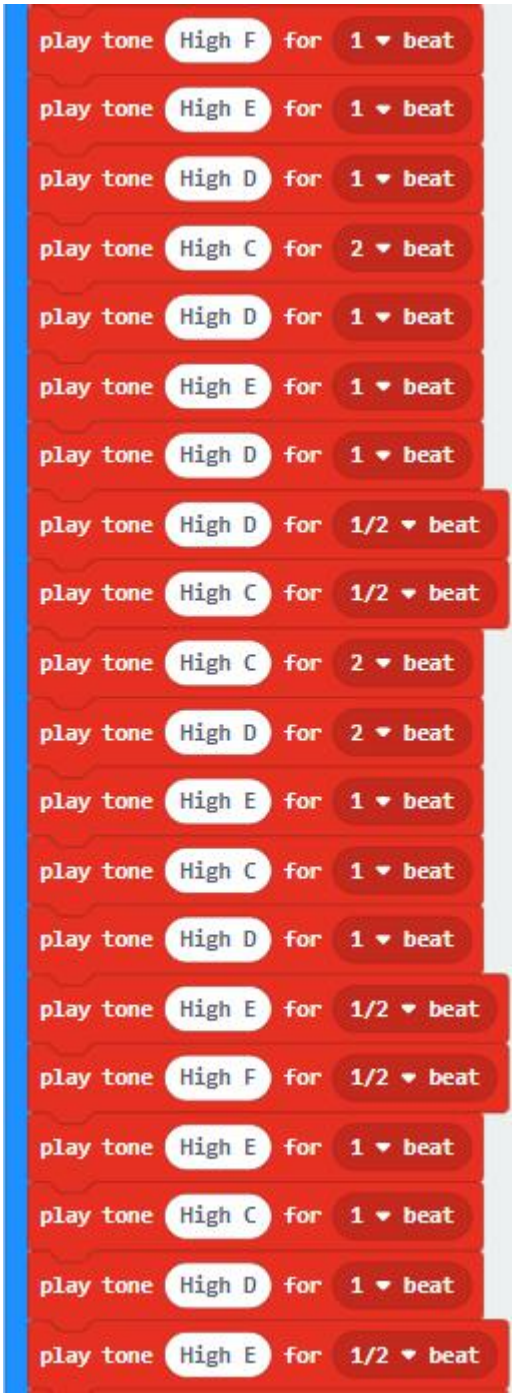
.....①Run the "on start" block to boot the program

..... ② turn off the buzzer on the Micro:bit shield

.....③turn off the LED matrix of the Micro:bit board

.....④The program is run circularly under the command of "forever" block

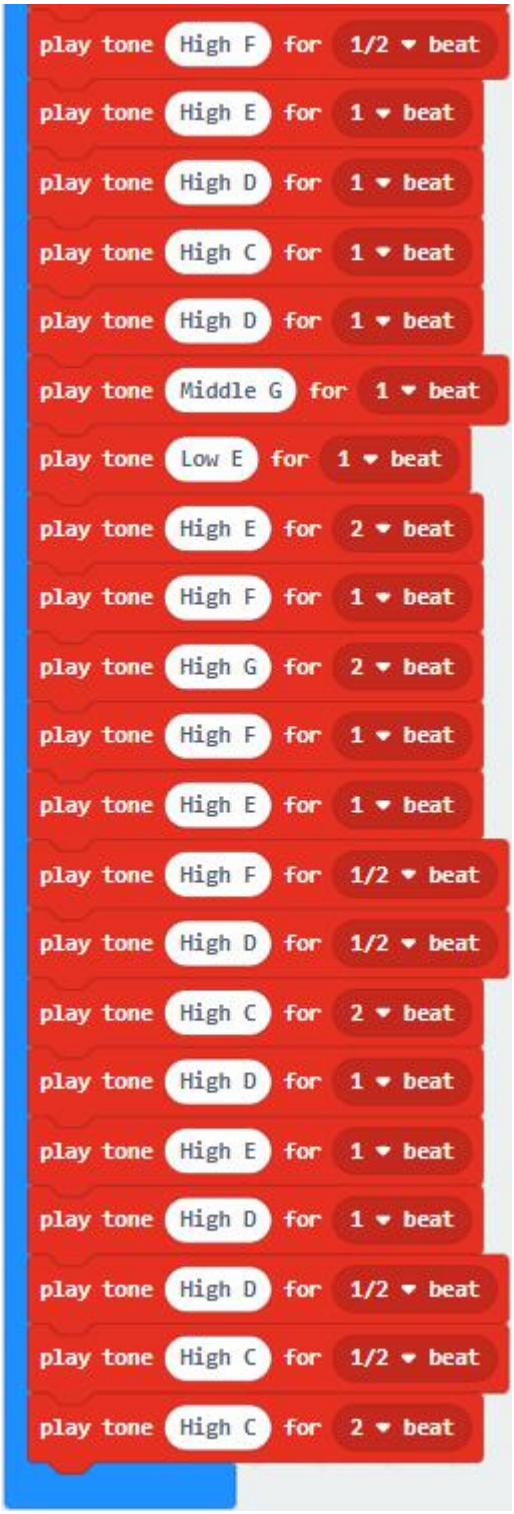
.....⑤ Play tone High E for 2 beats
.....⑥ Play tone High F for 1 beat
.....⑦Play tone High G for 2 beats
.....⑧Play tone High F for 1 beat
.....⑨Play tone High E for 1 beat
.....⑩Play tone High D for 1 beat
.....⑪Play tone High C for 2 beats
.....⑫Play tone High D for 1 beat
.....⑬Play tone High E for 2 beats
.....⑭Play tone High E for 1/2 beat
.....⑮Play tone High D for 1/2 beat
.....⑯Play tone High D for 2 beats
.....⑰Play tone High E for 2 beats
.....⑱Play tone High F for 1 beat
.....⑲Play tone High G for 2 beats



.....⑳Play tone High F for 1 beat
.....㉑Play tone High E for 1 beat
.....㉒Play tone High D for 1 beat
.....㉓Play tone High C for 2 beats
.....㉔Play tone High D for 1 beat
.....㉕Play tone High E for 1 beat
.....㉖Play tone High D for 1 beat
.....㉗Play tone High D for 1/2 beat
.....㉘Play tone High C for 1/2 beat
.....㉙Play tone High C for 2 beats
.....㉚Play tone High D for 2 beats
.....㉛Play tone High E for 1 beat
.....㉜Play tone High C for 1 beat
.....㉝Play tone High D for 1 beat
.....㉞Play tone High E for 1/2 beat
.....㉟Play tone High F for 1/2 beat
.....㊱Play tone High E for 1 beat

.....㊲Play tone High C for 1 beat
.....㊳Play tone High D for 1 beat

.....㊴Play tone High E for 1/2 beat



.....⑩Play tone High F for 1/2 beat
.....⑪Play tone High E for 1 beat
.....⑫Play tone High D for 1 beat
.....⑬Play tone High C for 1 beat
.....⑭Play tone High D for 1 beat
.....⑮Play tone High G for 1 beat
.....⑯Play tone High E for 1 beat
.....⑰Play tone High E for 2 beats

.....⑱Play tone High F for 1 beat
.....⑲Play tone High G for 2 beats

.....⑳Play tone High F for 1 beat

.....Play tone High E for 1 beat
.....Play tone High F for 1/2 beat

.....Play tone High D for 1/2 beat
.....Play tone High C for 2 beats

.....Play tone High D for 1 beat

.....Play tone High E for 1 beat
.....Play tone High D for 1 beat

.....Play tone High D for 1/2 beat
.....Play tone High C for 1/2 beat

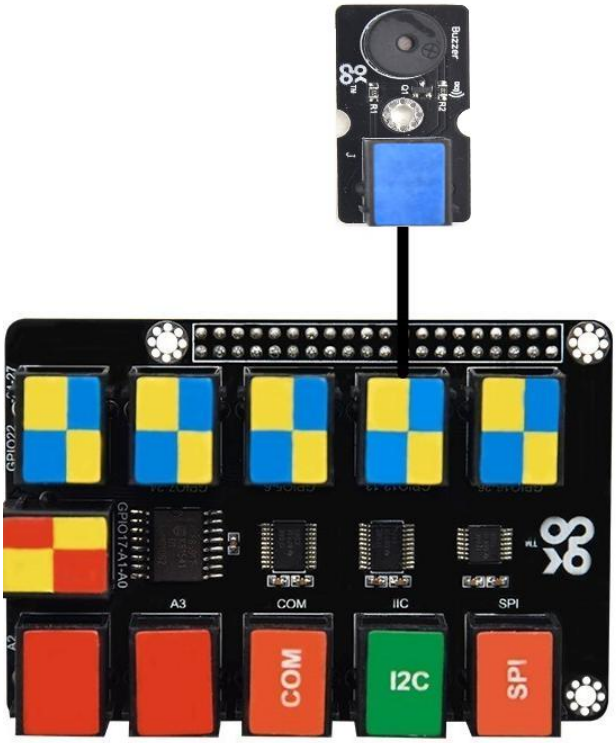
.....Play tone High C for 2 beats

Test Result

Wire up and upload the test code. The passive buzzer will play a song.

If you want to know more details about the Micro:bit board and the Micro:bit shield, you can refer to TS2179.

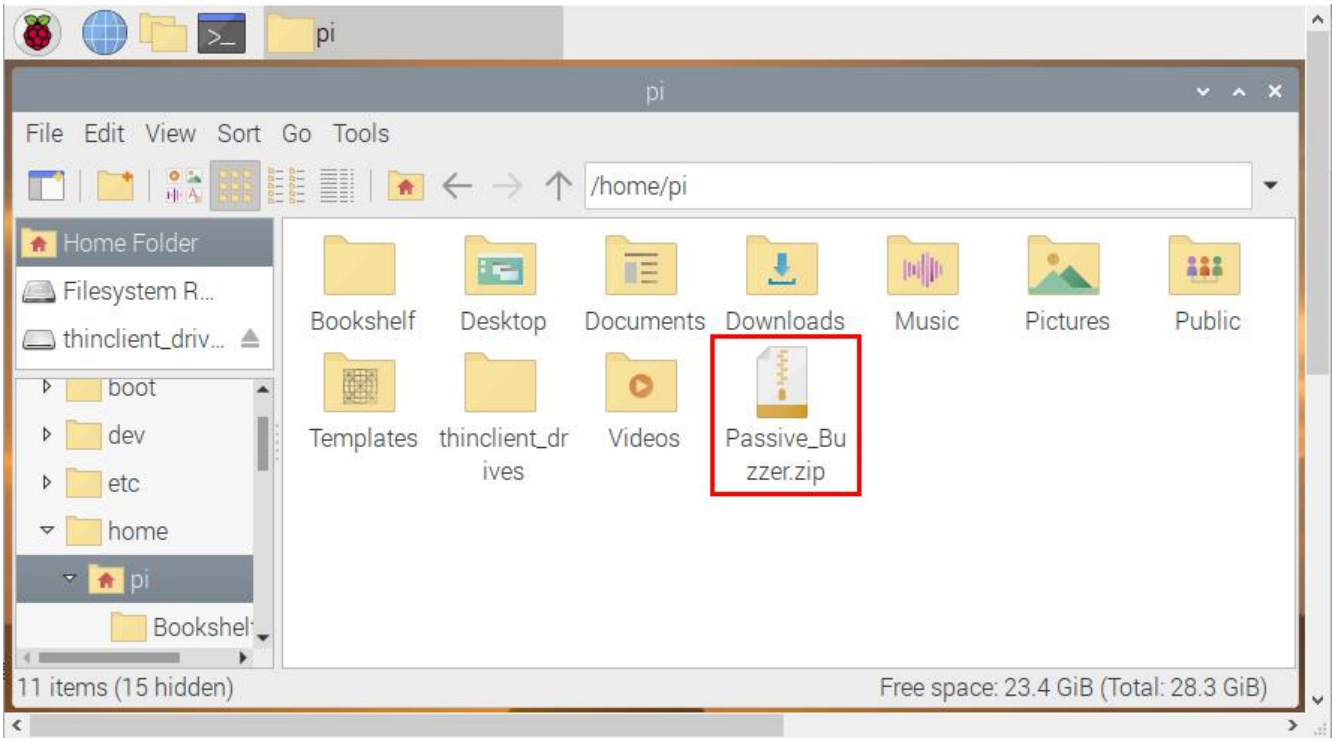
➢ **Raspberry Pi Application**

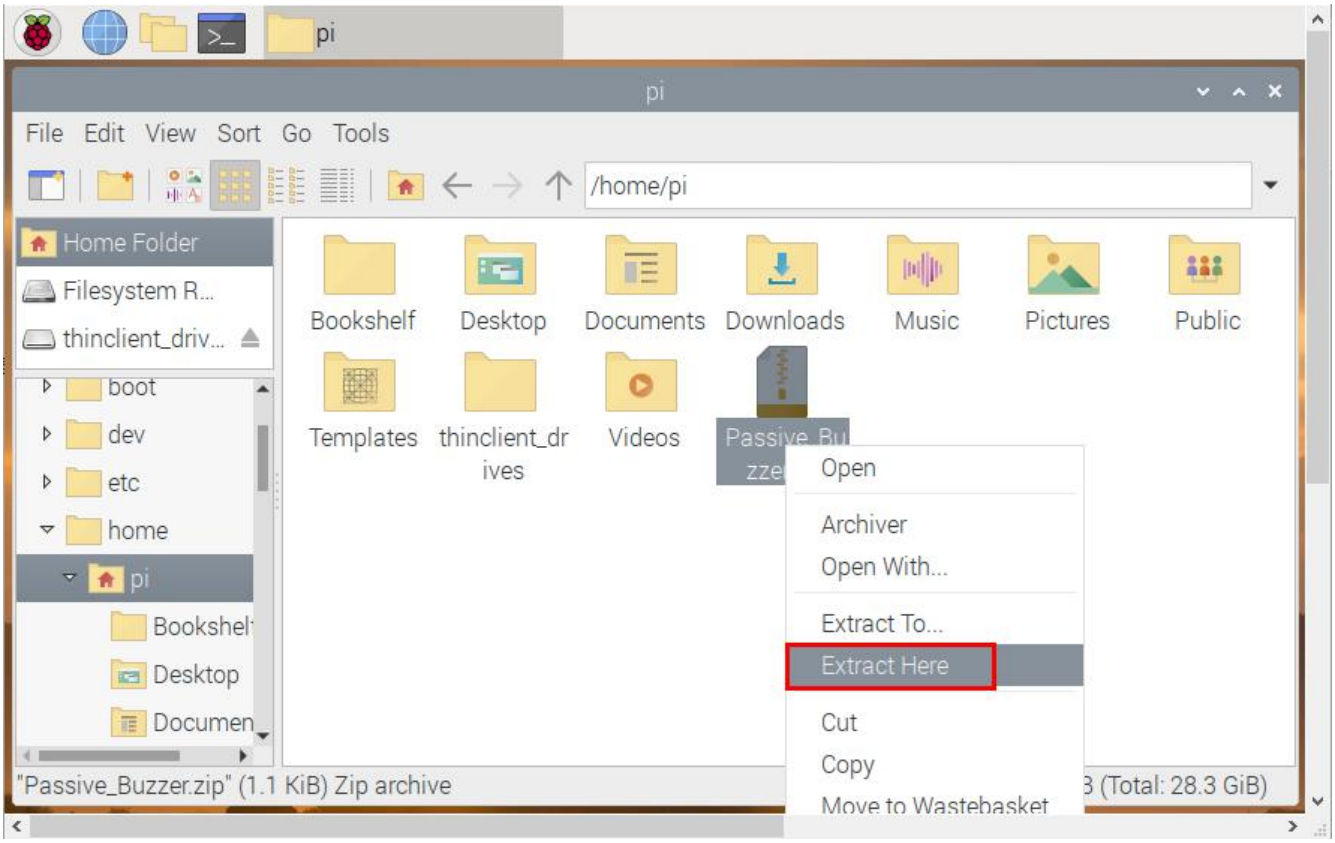


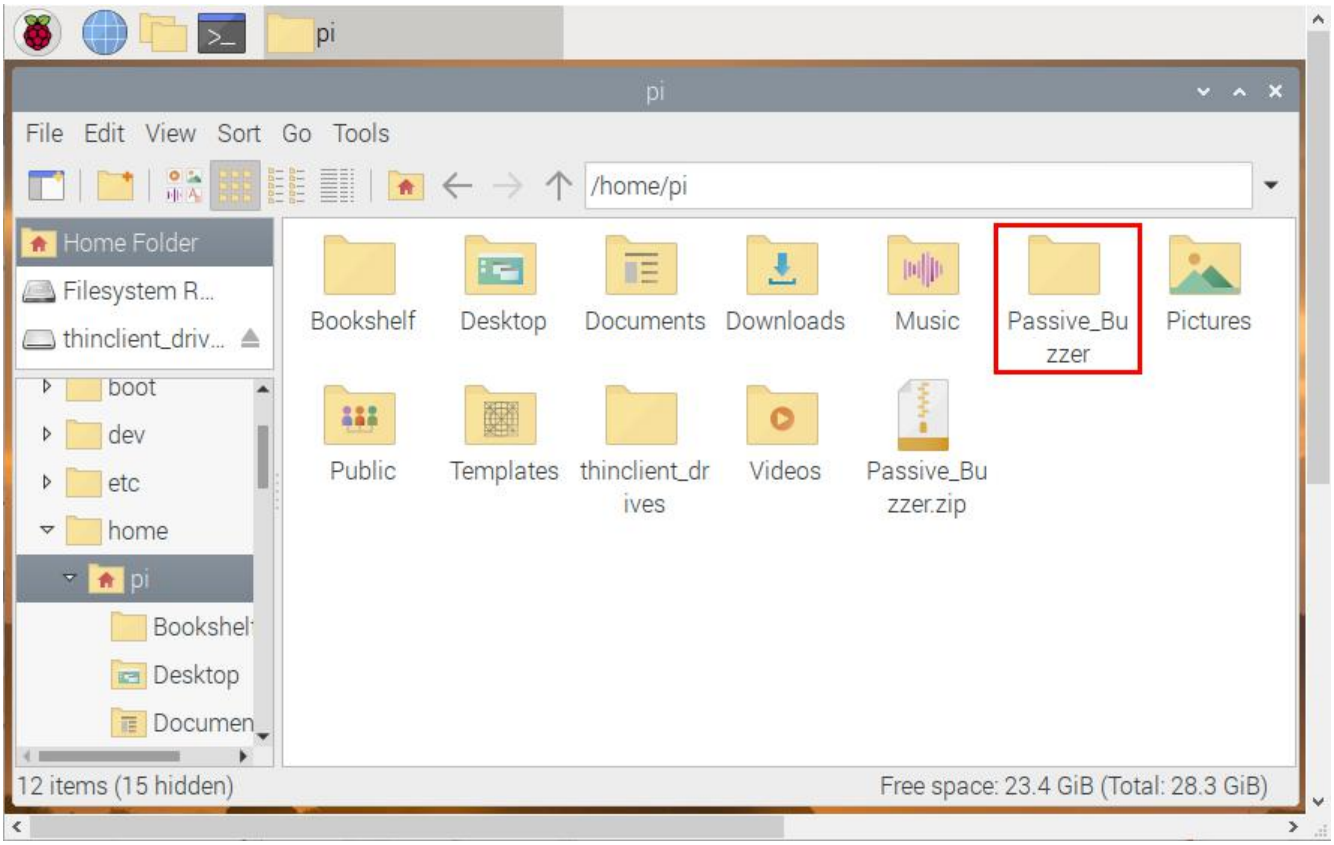
This module is compatible with the Raspberry Pi board and the TS2180 Raspberry Pi shield.

Copy the test code to Raspberry Pi system to run it

(1) Save the test code in the **pi** folder of Raspberry Pi system. Then place the Passive_Buzzer.zip file we provide in the **pi** folder, right-click and click **Extract Here.** As shown below:







(2) Compile and run the test code：

Input the following code and press“Enter”

```
cd /home/pi/Passive_Buzzer
gcc Passive_Buzzer.c -o Passive_Buzzer -lwiringPi
sudo ./Passive_Buzzer
```

(3) Test Result：

Insert the shield into the Raspberry Pi board. After programming finishes, the passive buzzer will play a birthday song.

Note: press Ctrl + C to exit code running

Test Code

File Name: Passive_Buzzer.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <wiringPi.h>
#define Do 262
#define Re 294
#define Mi 330
#define Fa 349
#define Sol 392
#define La 440
#define Si 494
#define Do_h 532
#define Re_h 587
#define Mi_h 659
#define Fa_h 698
#define Sol_h 784
#define La_h 880
#define Si_h 988

#define buzPin 23   //buzzer pin BCM GPIO 13

//The tones
int song_1[]=
{
   Sol,Sol,La,Sol,Do_h,Si,
   Sol,Sol,La,Sol,Re_h,Do_h,
   Sol,Sol,Sol_h,Mi_h,Do_h,Si,La,
   Fa_h,Fa_h,Mi_h,Do_h,Re_h,Do_h
};

//To the beat
float beat_1[]=
{
   0.5,0.5,1,1,1,1+1,
   0.5,0.5,1,1,1,1+1,
   0.5,0.5,1,1,1,1,1,
```

```
   0.5,0.5,1,1,1,1+1
};

int length;
int x;

void init()
{
  if (wiringPiSetup () == -1)
     exit (1) ;
  pinMode(buzPin, PWM_OUTPUT); //Set the pin to PWM output mode
  pwmSetMode(PWM_MODE_MS);  // Set PWM signal mode to MS mode
  pwmSetClock(32);  // Set the clock base frequency to 19.2m /32=600KHZ
}

void beep(int freq,int t_ms)
{
  int range;
  if(freq<100||freq>1000)
  {
    printf("invalid freq");
    return;
  }
  // Set the range to 600KHZ/ Freq. That is,
  //the freQ frequency period is composed of the range of 1/600khz.
  range=600000/freq;
  pwmSetRange(range);
  pwmWrite(buzPin,range/2);  // Set the duty cycle to 50%.
  if(t_ms>0)
  {
    delay(t_ms);
  }
}

int main()
{
 wiringPiSetup();
 init();
 length=sizeof(song_1)/sizeof(song_1[0]); //Number of tones

 while(1)
 {
  for(x=0;x<length;x++)  //play
```

```
    {
      beep(song_1[x],500*beat_1[x]);
    }
    pwmWrite(buzPin,0);   //turn off buzzer
    delay(2000);
  }
}
```

If you want to know how to utilize Raspberry Pi and the Raspberry Pi shield, you can refer to TS2180.

END