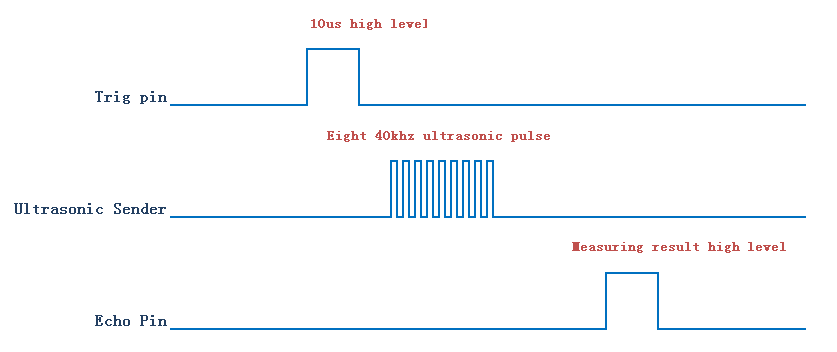
# Ultrasonic Ranging Experiment

## Introduction

The principle of ultrasonic ranging: the ultrasonic transmitter emits ultrasonic wave in a certain direction and we start timing at the same time. While the ultrasonic wave in the air encounters an obstacle, it will immediately return, the ultrasonic receiver receives the reflected wave and then we stop timing. The speed of sound waves in the air is 340 meters per second. According to the recorded time t, we can use the mathematical formula s = 340m / s \* t / 2 to calculate the distance s between the starting point and the obstacle.

The ultrasonic ranging module has four pins, they are **VCC**, Trig, Echo, **GND**. Trig is the trigger pin for distance measurement. As long as it maintains a high level voltage of **10μs**, the ultrasonic module will automatically send **40KHZ** \*8 ultrasonic pulses and detect whether there is a return signal. This step will be automatically completed by the internal module. If it receives any return signal internally and the Echo pin will output a high-level voltage. The duration of the high level voltage is the time from the ultrasonic wave to the return. We can use the pulseIn() function to obtain the result of the distance measurement and calculate the actual distance.



## Experimental Purpose

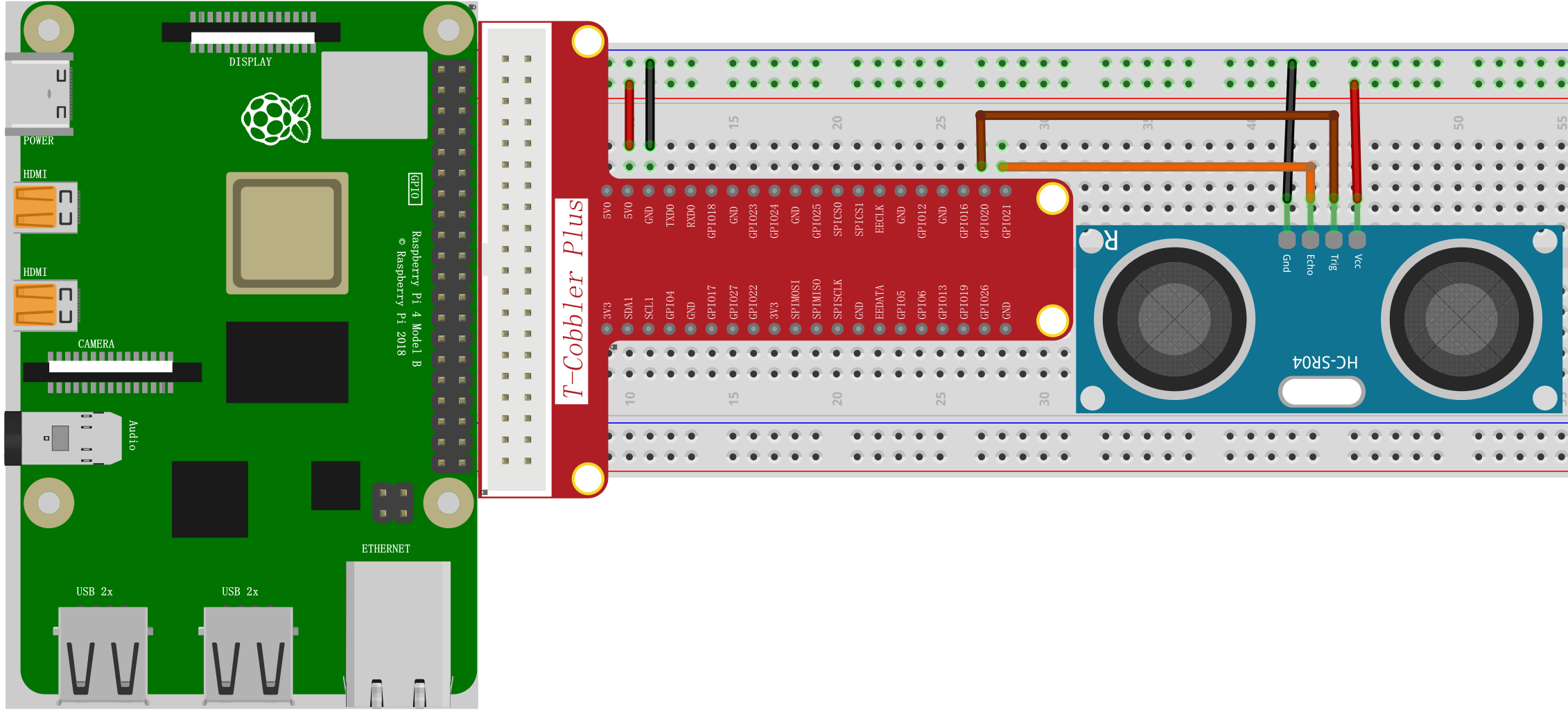
Measure the distance from the object to the destination through the ultrasonic ranging module.。

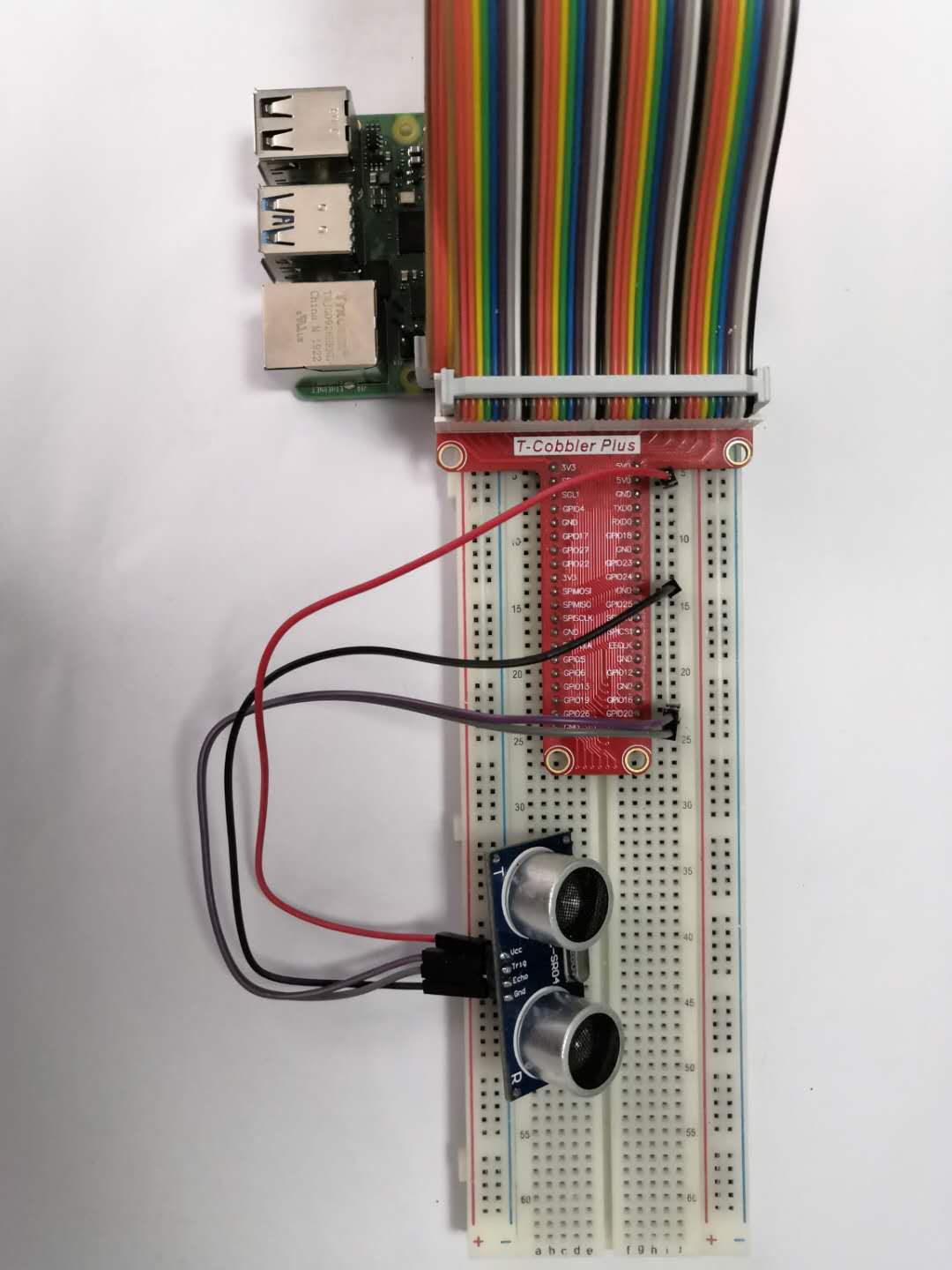
## Component List

* Raspberry Pi main board
* Raspberry Pi T-Cobbler Plus expansion board
* Breadboard
* Cable
* Ultrasonic Module \* 1
* Several jumper wire

## Wiring

|  |  |
| --- | --- |
| Raspberry Pi | Ultrasonic |
| VCC | 1Vcc) |
| 28(wiringPi)/20(BCM) | 2 (Trig) |
| 29(wiringPi)/21(BCM) | 3(Echo) |
| GND | 4 (Gnd) |





## C++ program

|  |
| --- |
| #include <wiringPi.h>  #include <stdio.h>  #include <sys/time.h>    #define Trig 28  #define Echo 29    void ultraInit**(**void**)**  **{**  pinMode**(**Echo**,** INPUT**);**  pinMode**(**Trig**,** OUTPUT**);**  **}**    float disMeasure**(**void**)**  **{**  struct timeval tv1**;**  struct timeval tv2**;**  long start**,** stop**;**  float dis**;**    digitalWrite**(**Trig**,** LOW**);**  delayMicroseconds**(**2**);**    digitalWrite**(**Trig**,** HIGH**);**  delayMicroseconds**(**10**);** // Send out ultrasonic pulses  digitalWrite**(**Trig**,** LOW**);**    **while(!(**digitalRead**(**Echo**)** **==** 1**));**  gettimeofday**(&**tv1**,** **NULL);** // Get current time    **while(!(**digitalRead**(**Echo**)** **==** 0**));**  gettimeofday**(&**tv2**,** **NULL);** // Get current time    start **=** tv1**.**tv\_sec **\*** 1000000 **+** tv1**.**tv\_usec**;** // Microsecond time  stop **=** tv2**.**tv\_sec **\*** 1000000 **+** tv2**.**tv\_usec**;**    dis **=** **(**float**)(**stop **-** start**)** **/** 1000000 **\*** 34000 **/** 2**;** // Calculate the distance    **return** dis**;**  **}**    int main**(**void**)**  **{**  float dis**;**    **if(**wiringPiSetup**()** **==** **-**1**){** //when initialize wiring failed,print messageto screen  printf**(**"setup wiringPi failed !"**);**  **return** 1**;**  **}**    ultraInit**();**    **while(**1**){**  dis **=** disMeasure**();**  printf**(**"distance = %0.2f cm\n"**,**dis**);**  delay**(**1000**);**  **}**    **return** 0**;**  **}** |

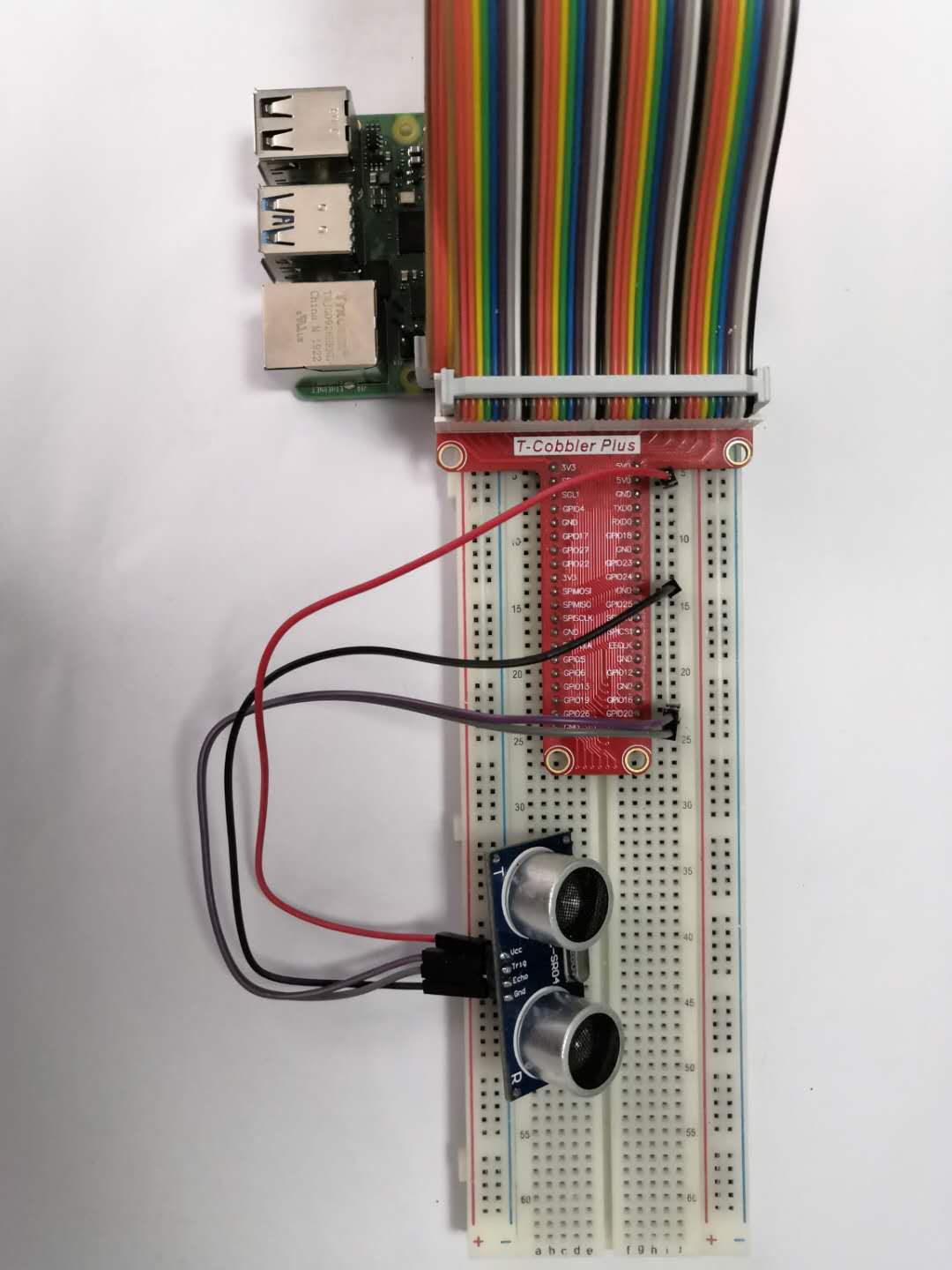
## Python program

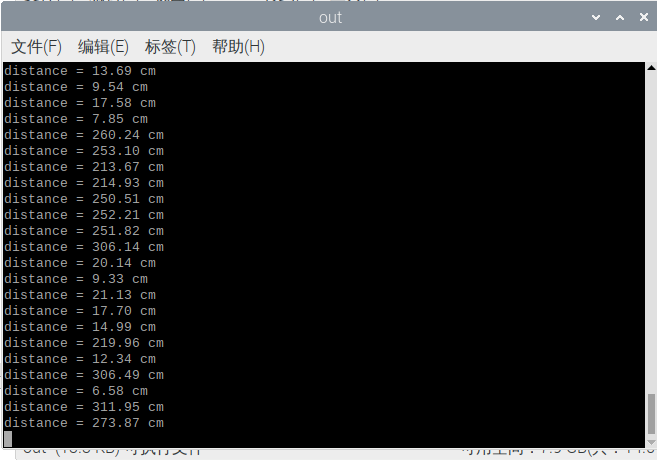
|  |
| --- |
| import RPi**.**GPIO as GPIO  import time  GPIO**.**setmode**(**GPIO**.**BCM**)**  trig**=**20 #send**-**pin  echo**=**21 #receive**-**pin  GPIO**.**setup**(**trig**,** GPIO**.**OUT**,**initial**=**GPIO**.**LOW**)**  GPIO**.**setup**(**echo**,** GPIO**.**IN**)**    def Measure**():**    #send  GPIO**.**output**(**trig**,** True**)**  time**.**sleep**(**0.00001**)** #1us  GPIO**.**output**(**trig**,** False**)**    #start recording  **while** GPIO**.**input**(**echo**)==**0**:**  pass  start**=**time**.**time**()**    #end recording  **while** GPIO**.**input**(**echo**)==**1**:**  pass  end**=**time**.**time**()**    #compute distance  distance**=**round**((**end**-**start**)\***343**/**2**\***100**,**2**)**  print**(**"distance:{0}cm"**.**format**(**distance**))**    **while** True**:**  Measure**()**  time**.**sleep**(**1**)**    GPIO**.**cleanup**();** |

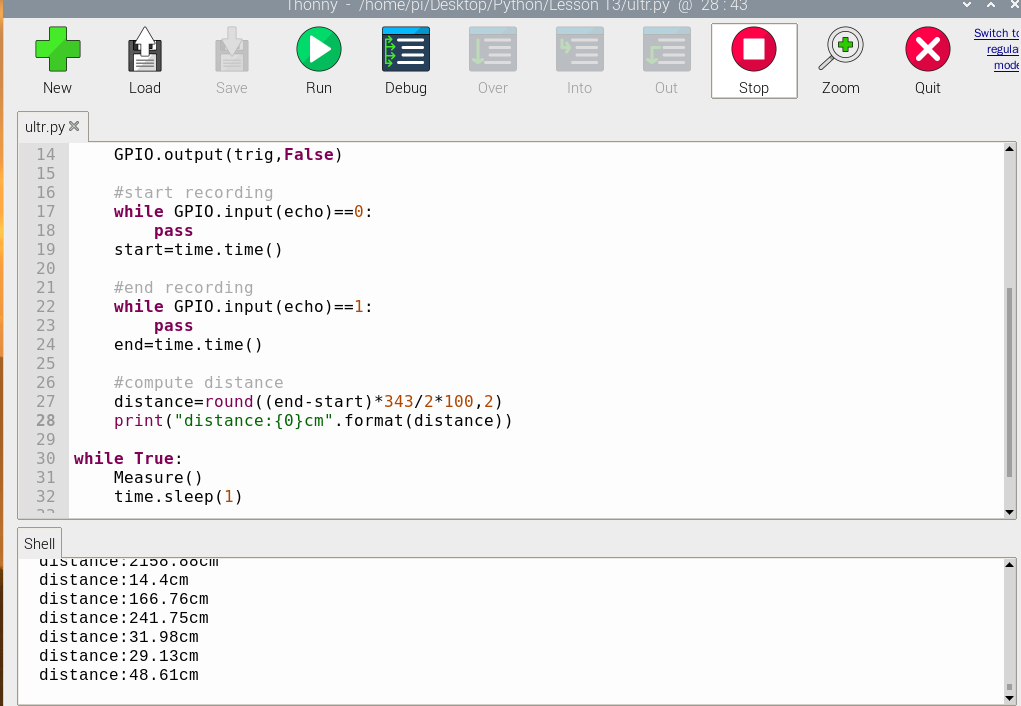
## Java program

|  |
| --- |
| **import** com**.**pi4j**.**wiringpi**.**Gpio**;**  **import** com**.**pi4j**.**io**.**gpio**.**PinEdge**;**  **import** com**.**pi4j**.**wiringpi**.**GpioInterrupt**;**  **import** com**.**pi4j**.**wiringpi**.**GpioInterruptListener**;**  **import** com**.**pi4j**.**wiringpi**.**GpioInterruptEvent**;**  **import** com**.**pi4j**.**wiringpi**.**GpioUtil**;**  public class test **{**  static double distance **=** 0.0**,** time\_1 **=** 0**,** time\_2 **=** 0**;**  public static Long getmicTime**()** **{**  Long cutime **=** System**.**currentTimeMillis**()** **\*** 1000**;** // 微秒  Long nanoTime **=** System**.**nanoTime**();** // 纳秒  **return** cutime **+** **(**nanoTime **-** nanoTime **/** 1000000 **\*** 1000000**)** **/** 1000**;**  **}**  public static void clean\_date**()** **{**  Ultr**.**time\_1 **=** 0**;**  Ultr**.**time\_2 **=** 0**;**  **}**    public static void main**(**String args**[])** **throws** InterruptedException **{**    // setup wiring pi  **if** **(**Gpio**.**wiringPiSetup**()** **==** **-**1**)** **{**  System**.**out**.**println**(**" ==>> GPIO SETUP FAILED"**);**  **return;**  **}**    // configure GPIO 29 as an INPUT pin; GPIO\_028 is set to output enable it for callbacks  Gpio**.**pinMode**(**28**,** Gpio**.**OUTPUT**);**  Gpio**.**pinMode**(**29**,** Gpio**.**INPUT**);**  //Gpio.pullUpDnControl(29, Gpio.PUD\_UP);  GpioInterrupt**.**enablePinStateChangeCallback**(**29**);**    // continuously loop to prevent program from exiting  **for** **(;;)** **{**  // GPIO\_29 is set to high level  Gpio**.**pinMode**(**28**,** Gpio**.**OUTPUT**);**  Gpio**.**digitalWrite**(**28**,** 0**);**  Gpio**.**delayMicroseconds**(**2**);**  Gpio**.**digitalWrite**(**28**,** 1**);**  Gpio**.**delayMicroseconds**(**10**);**  Gpio**.**digitalWrite**(**28**,** 0**);**  **while** **(!(**Gpio**.**digitalRead**(**29**)** **==** 1**));**  Ultr**.**time\_1 **=** Ultr**.**getmicTime**();**  **while** **(!(**Gpio**.**digitalRead**(**29**)** **==** 0**));**  Ultr**.**time\_2 **=** Ultr**.**getmicTime**();**  Ultr**.**distance **=** **(**Ultr**.**time\_2 **-** Ultr**.**time\_1**)** **/** 1000000 **/** 2 **\*** 340 **\*** 100**;**  System**.**out**.**println**(**Ultr**.**distance**);**  Ultr**.**clean\_date**();**  Thread**.**sleep**(**500**);**  **}**  **}**  **}** |

## Experimental Effect







We use Raspberry Pi to control ultrasonic to realize this aim of the object distance measurement. The experiment requires us to understand the principle of ultrasonic distance measurement and learn how to use the Raspberry Pi IO port.