LAFVIN

Starter Kit for Raspberry Pi

Content

Company Profile

Established in 2011, lafvin is a manufacturer and trader specialized in research,development and production of 2560 uno,nano boards,and all kinds of accessories or sensors use for arduino,rasperrry.We also complete starter kits designed for interested lovers of any levels to learn Arduino or Raspberry.We are located in Shenzhen,China.All of our products comply with international quality standards and are greatly appreciated in a variety of different markets throughout the world.

Customer Service

We are cooperating with a lot of companies from diffirent countries.Also help them to purchase electronic component products in china,and became the biggest supplier of them. We look forward to build cooperate with more companies in future.

By the way,We also look forward to hearing from you and any of your critical comment or suggestions.Pls email us by lafvin_service@163.com if you have any questions or suggestions.

As a continuous and fast growing company. We keep striving our best to offer you excellent products and quality service.

Our Store

Aliexpress store: https://www.aliexpress.com/store/1942043Brand in Amazon:LAFVIN

Product Catalog

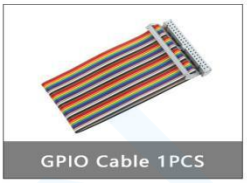
https://drive.google.com/drive/folders/0BwvEeRN9dKllblZING00TkhYbGs?usp=sharing

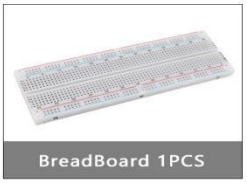
Tutorial

This tutorial include codes,library ,lessons,And related software, system image. It is designed for beginners.This raspberry pie learning starter kit, including code tutorials for C and Python. This kit includes the component modules required by the Raspberry Pi introductory learner, as well as users of both computer languages.

Packing list


GPIO Extension Board 1PCS


GPIO Cable 1PCS


BreadBoard 1PCS

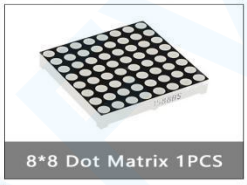

Remote 1PCS


Power Supply Module 1PCS


Resistor 100PCS

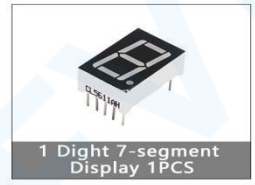

Driver Board 1PCS


Ultrasonic Sensor 1PCS


8*8 Dot Matrix 1PCS

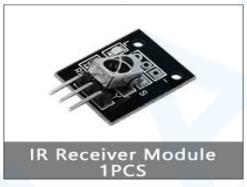

Fan Blade and 3V Motor each 1 PCS


4 Dight 7-segment Display 1PCS


1 Dight 7-segment Display 1PCS


74HC595 2PCS


L293D 1PCS


IR Receiver Module 1PCS


Small Button 5PCS


1N4007 2PCS


Photoresistor 2PCS


Led RGB 1PCS


Led Red 5PCS

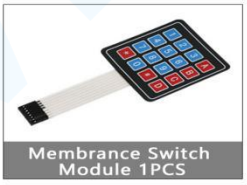

65Jumper Wire 1PCS


Stepper Motor 1PCS


SG90 1PCS


LCD1602 1PCS


HC-SR501 1PCS


Membrance Switch Module 1PCS


5V Relay 1PCS


F-M Dupont Wire 10PCS


DHT11 1PCS


Potentiometer 1PCS


9V Battery Connector with DC 1PCS


GY-521 1PCS


Tilt Switch 1 PCS


Buzzer 2PCS


PN2222 2PCS


Diode Rectifier 1PCS


Led Yellow 5PCS


Led Blue 5PCS


Led Green 5PCS


Led White 5PCS

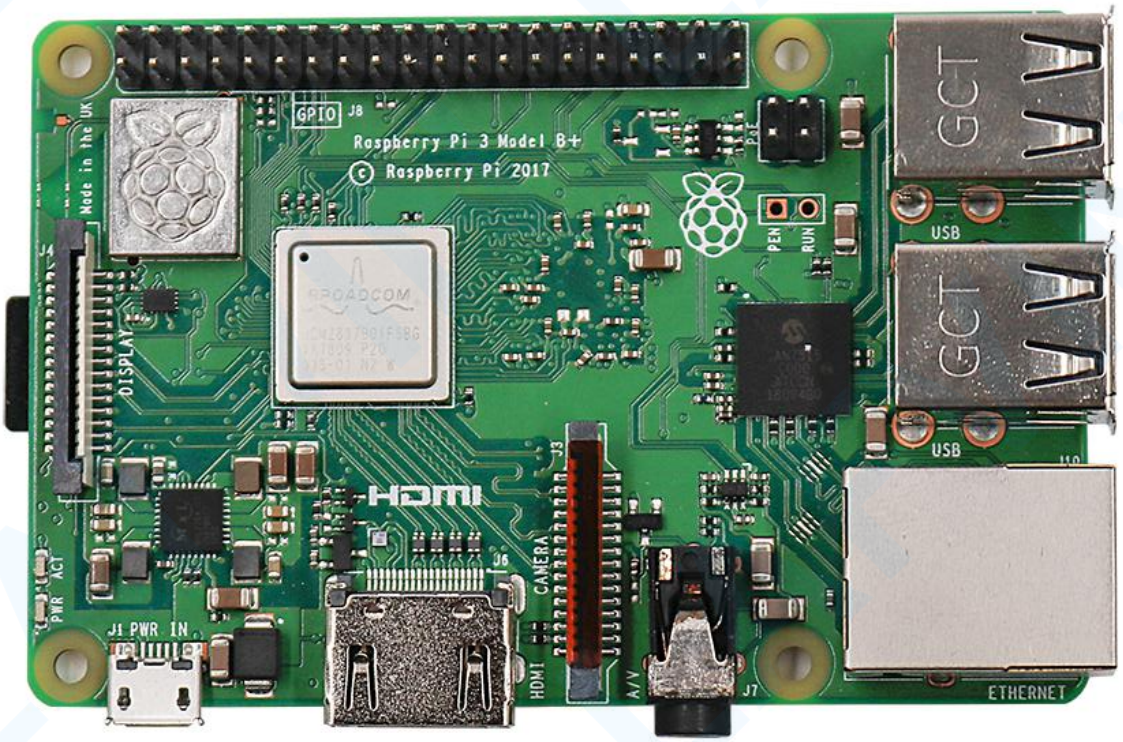
Introduction to Raspberry Pi

The Raspberry Pi Foundation is a small charitable organization in the UK that was founded to promote technology rather than selling technology. The Foundation has never actually published a product in the past, so it chose two global channel partners, e-Community and RS Components, to handle the first Raspberry Pi orders. Faced with amateurs and enthusiastic DIY technology fans, Raspberry Pi sales are very good.

Raspberry Pi is a mini computer for computer amateurs, teachers, elementary school students and small businesses. It is pre-installed with Linux system. It is only a credit card size, equipped with an ARM architecture processor, and its computing performance is similar to that of a smart phone.

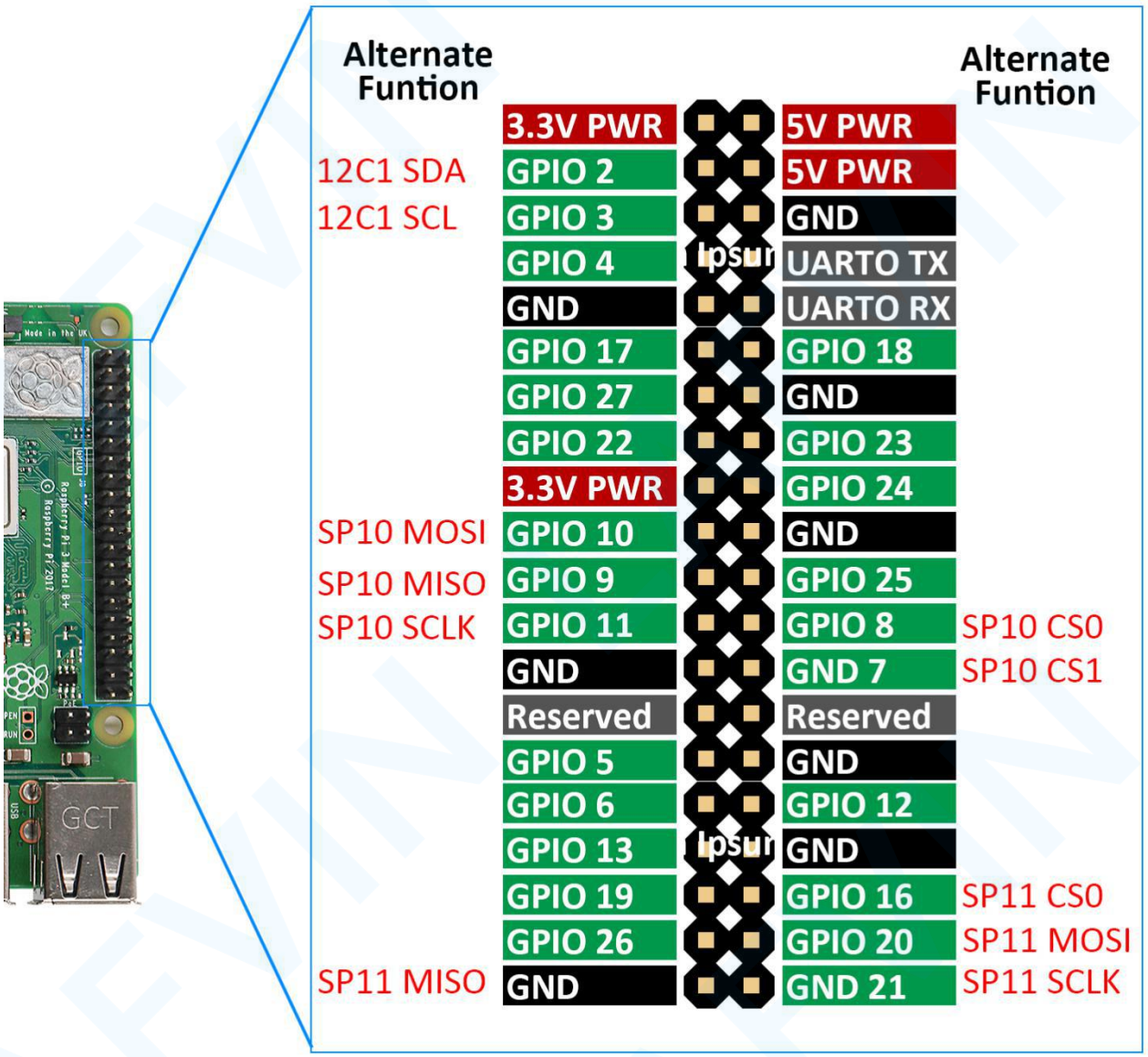
On the interface side, the Raspberry Pi provides a USB interface for the mouse and keyboard, in addition to the Fast Ethernet interface, SD card expansion interface and an HDMI high-definition video output interface, which can be connected to the display or TV.

The Raspberry Pi evolves through many versions including the latest (so far) Raspberry Pi 4 Model B ,3 Model B+ , 3 Model B, 2 model B, 1 Model B+, Zero, and 1 Model A+.This kit is based on Raspberry Pi 3 Model B+ as a development board.



Raspberry Pi Pin Name

There are no pins printed on the latest Raspberry Pi, which may bring various troubles to new users. The following is the actual pins definition and form definition of the Raspberry Pi.



Extension Board

In order to facilitate the experiment, the Raspberry Pi pin will be pulled out through the expansion board, which makes it easier to combine the use of the breadboard.The pin and raspberry pie pins on the expansion

board are the same in number and order.

Install the System

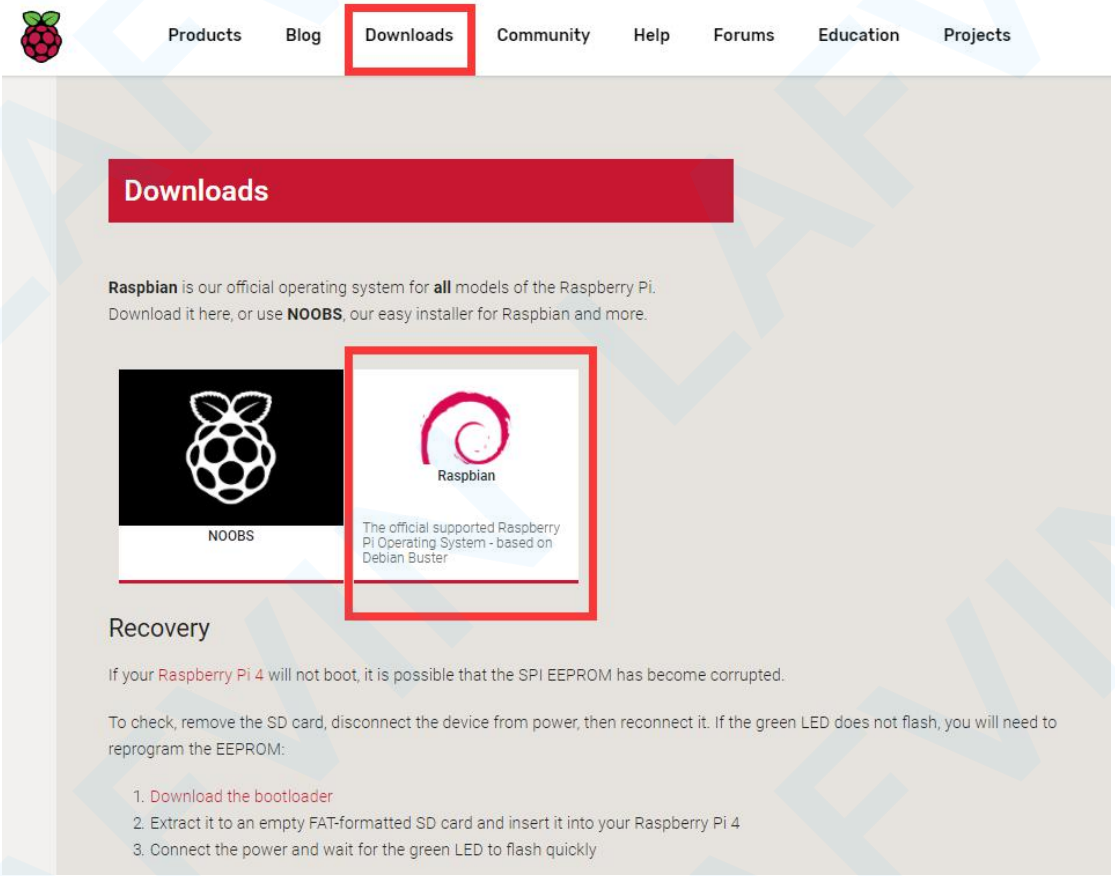
Step1 :download the following two tools

Win32 Disk Imager

A tool Disk Imager Win32 is required to write system. You can download and install it through visiting the web site: https://sourceforge.net/projects/win32diskimager/

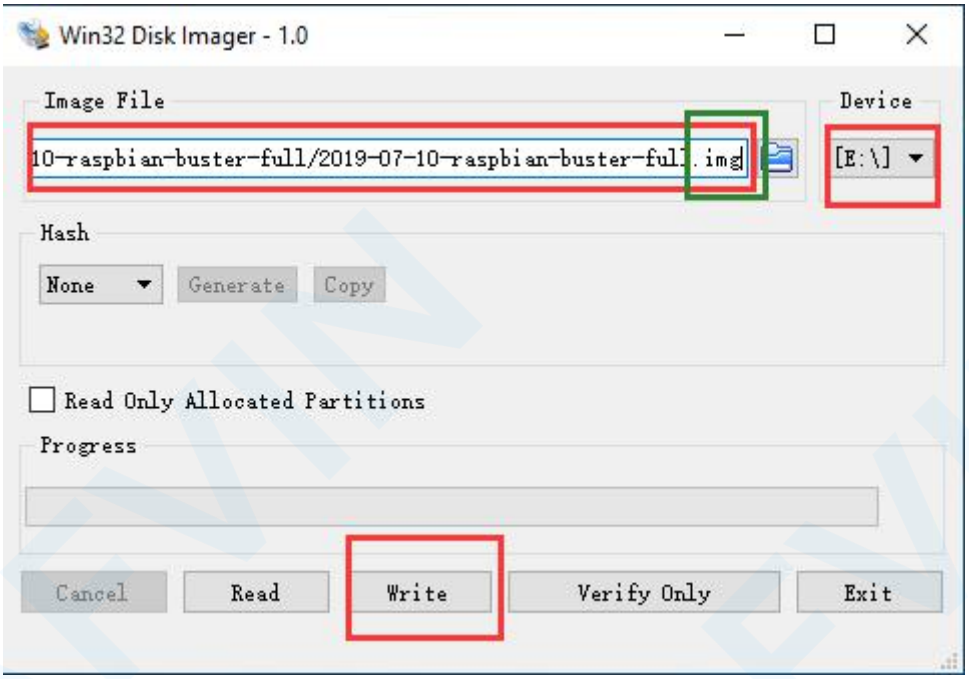
Raspbian operating system

Visit RPi official website (https://www.RaspberryPi.org/), click “Downloads” and choose to download “RASPBIAN”. RASPBIAN supported by RPI is an operating system based on Linux, which contains a number of contents required for RPi. We recommended RASPBIAN system to beginners. All projects in this tutorial are operated under the RASPBIAN system.



Step2 :Write an image file to the SD card

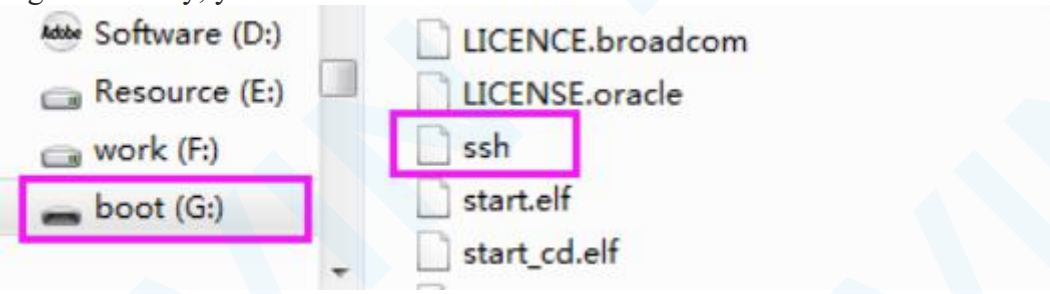
Connect the SD to the USB port of the computer through the card reader.Open the **Win32 Disk Imager** tool, select the system image file you just downloaded, select the disk location where the SD card is located, and finally click Write.Note:To select a system image file, be sure to select the .img suffix file. The files downloaded from the website must be decompressed to get the image file of the .img suffix.



Step3 :Open SSH function

Under previous Raspbian system, SSH is opened by default. Under the latest version of Raspbian system, it is closed by default. So you need to open it first.

Method: For 2016-11-25 release or above, SSH (a protocol securing remote login session and other network service) is Disabled by default. Therefore, when you need to log in remotely, you need to create a file named "ssh" under /boot/ to enable it.



Step4 :Download **putty** software

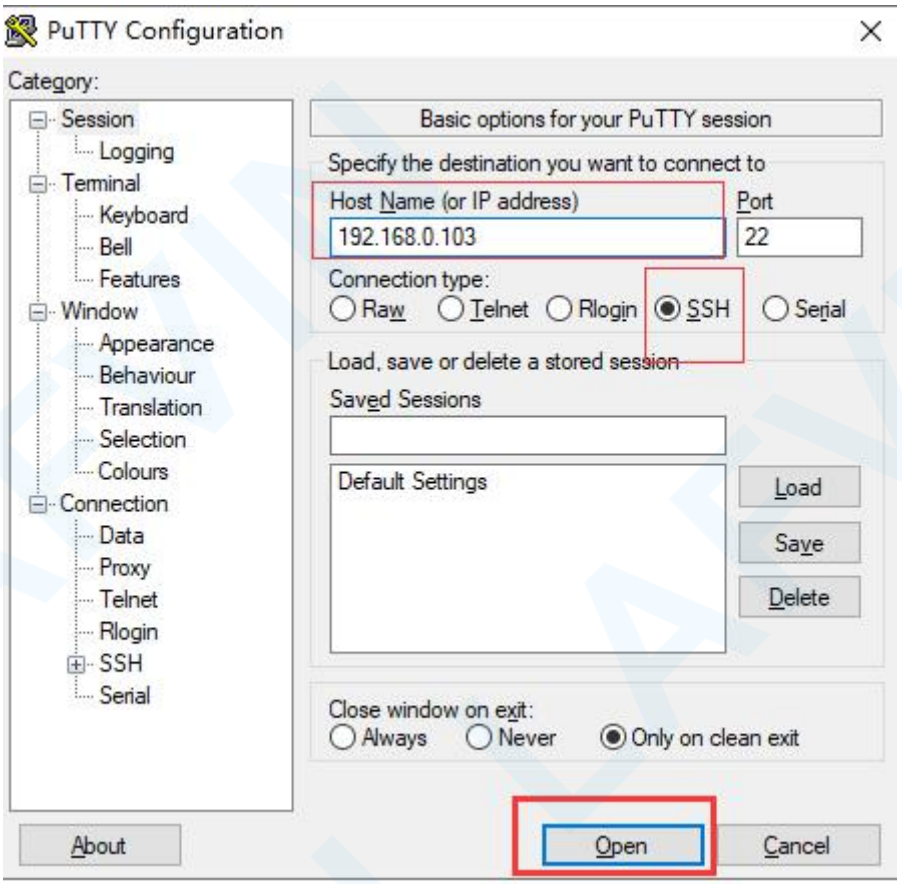
We can use the Raspberry Pi in two ways: **remote desktop** and **physical LCD display.**Regardless of which desktop connection method is used, we need to communicate with the Raspberry Pi through a command window. We send control commands to the Raspberry Pi through the command window of the **putty** software.
download the tool software Putty. Its official address: http://www.putty.org/

Step5 :Establish a command window to communicate with the Raspberry Pi

Establish a command window for communication with the Raspberry Pi through the **SSH** function.

Then use cable to connect your RPi to the routers of your PC LAN, to ensure your PC and your RPi in the same LAN. Then put the system Micro SD Card prepared before into the slot of the RPi and turn on the power supply waiting for starting RPi(May take two minutes). For example, I have inquired to my RPi IP

address, and it is “192.168.0.103". Then open Putty,enter the address, select SSH, and then click "OPEN", as shown below:

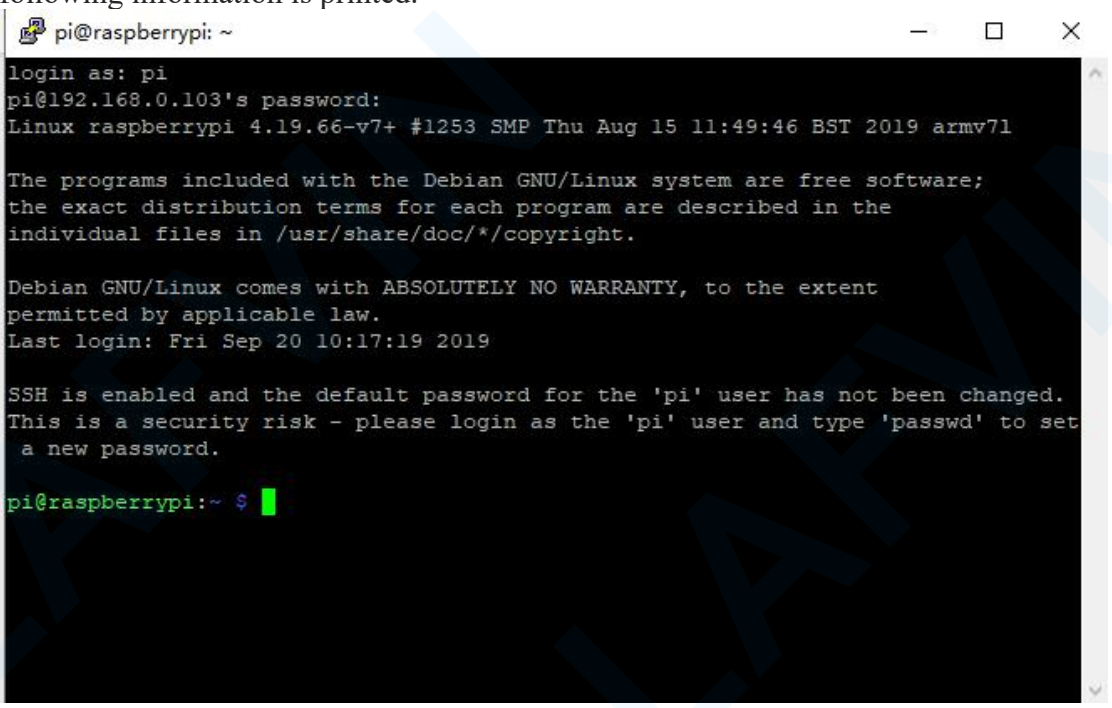


There will appear a security warning at first login. Just click “YES”.If the window prints the following message, the connection is successful.

Then enter the account password on the window to log in.

RPi default user name: pi;
password: raspberry).

When you enter the password, there will be no display on the screen. This is normal. After the correct output, press "Enter" to confirm.The login is successful when the following information is printed.



Step6 :Connection control desktop

We can use the Raspberry Pi in two ways: **remote desktop** and **physical LCD display.**

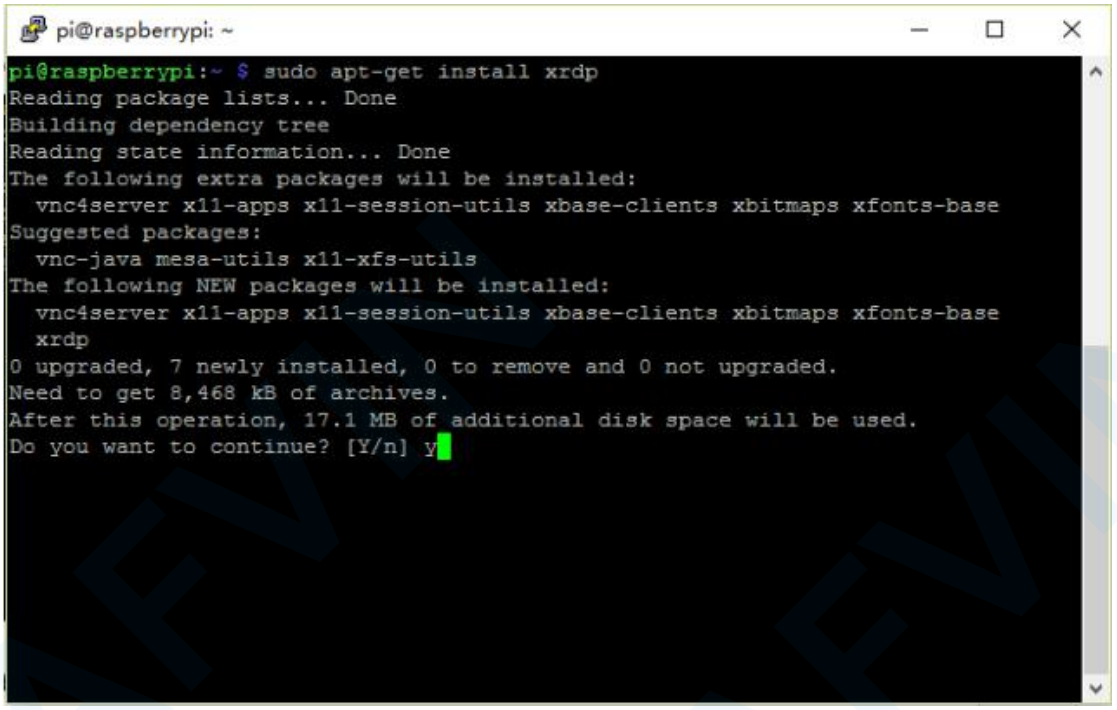
If you don't have an LCD display, you can use the Raspberry Pi by connecting to the remote desktop.There are two ways to connect to the remote desktop. The first is to use the remote desktop of the pc and the xrdp function of the Raspberry Pi. The second is to download the vnc software to connect.

Method 1: PC Remote Desktop<——>xrdp Raspberry Pi

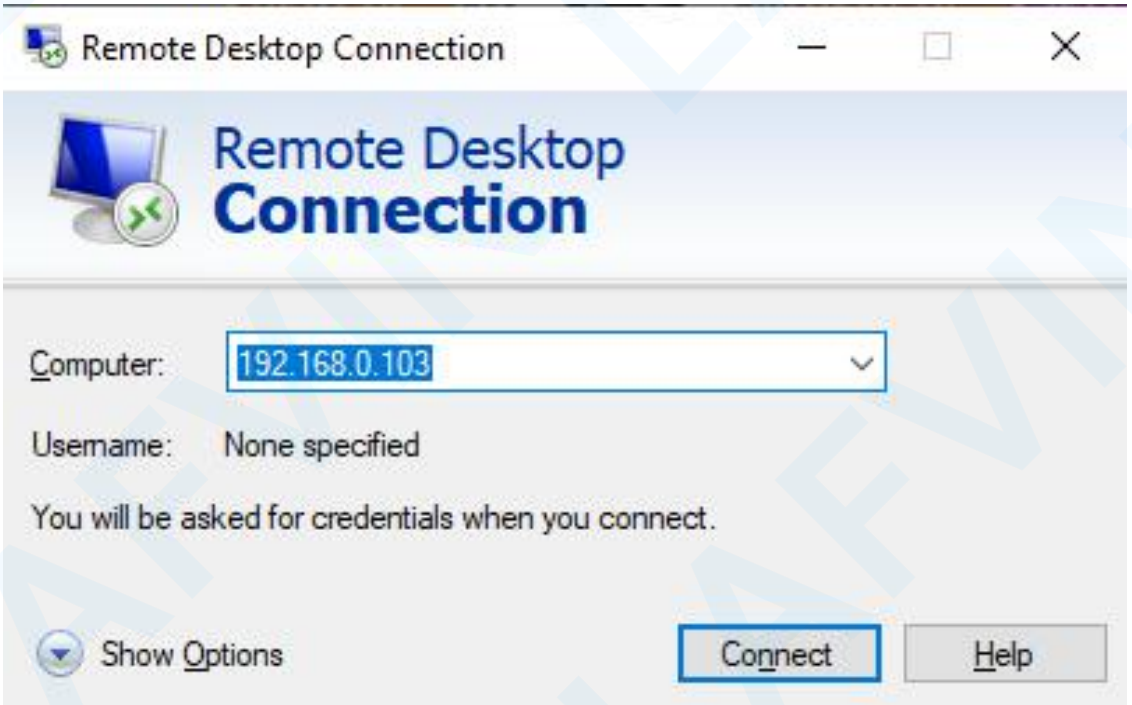
Firstly,install a xrdp service, an open source remote desktop protocol(rdp) server, for RPi. Type the following command, then press enter to confirm:

```
sudo apt-get install xrdp
```

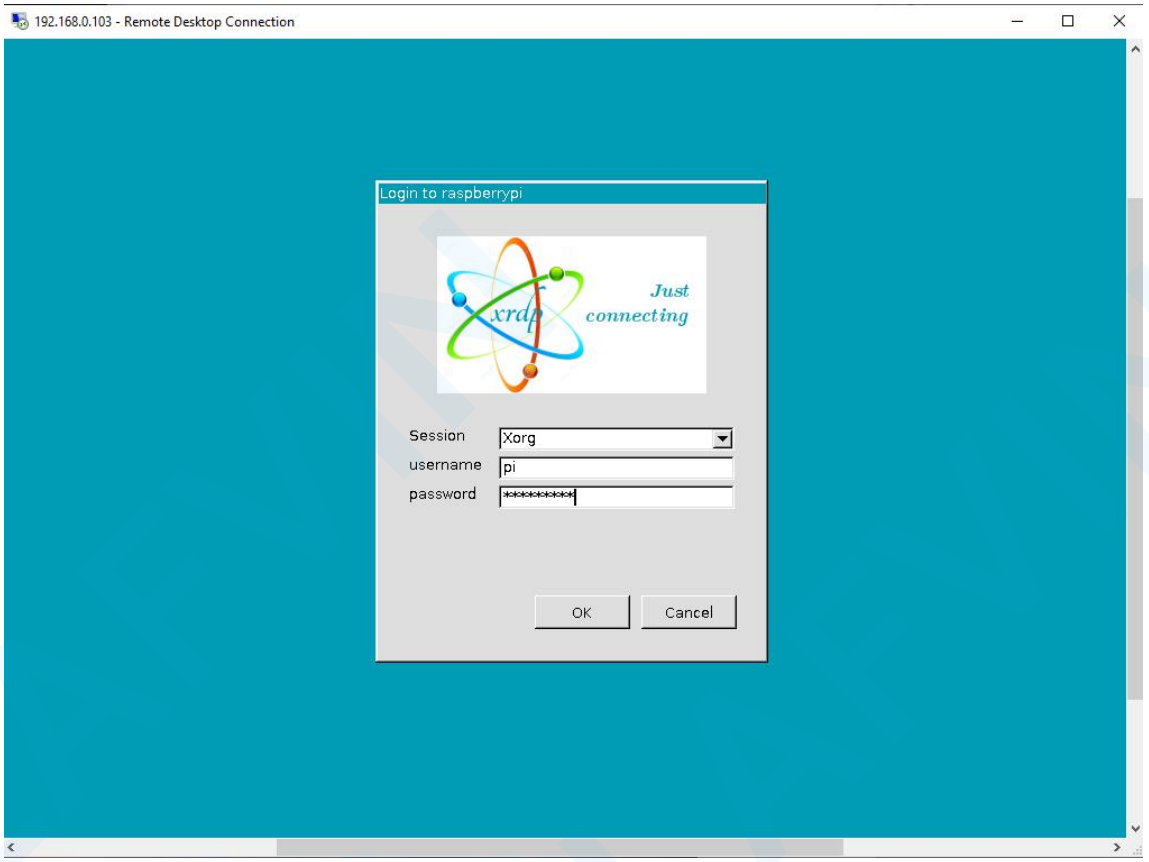
Then type Y and press Enter to agree to the installation.



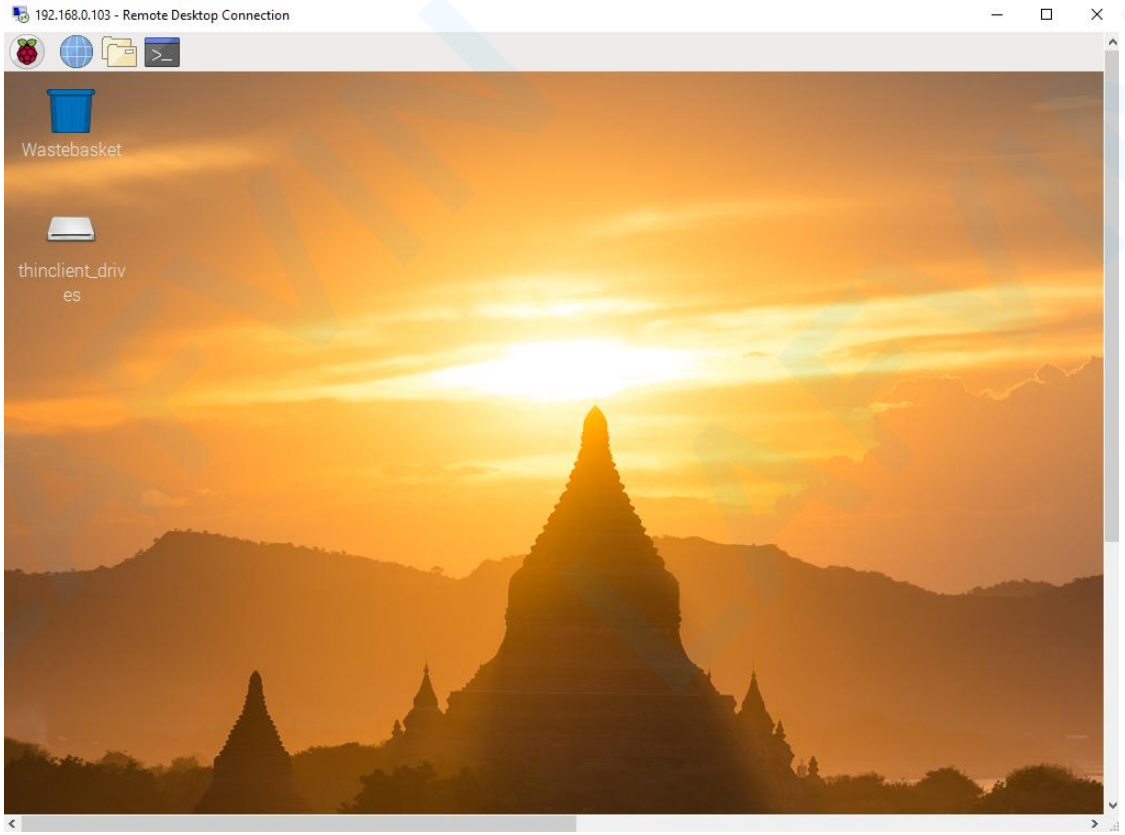
Then press win+R on your pc to search "mstsc.exe".Enter the Raspberry Pi IP address found in the router management background, click “Connect”



Later, there will be xrdp login screen. Enter the user name and password of RPi and click “OK”.
RPi default user name: pi
password: raspberry

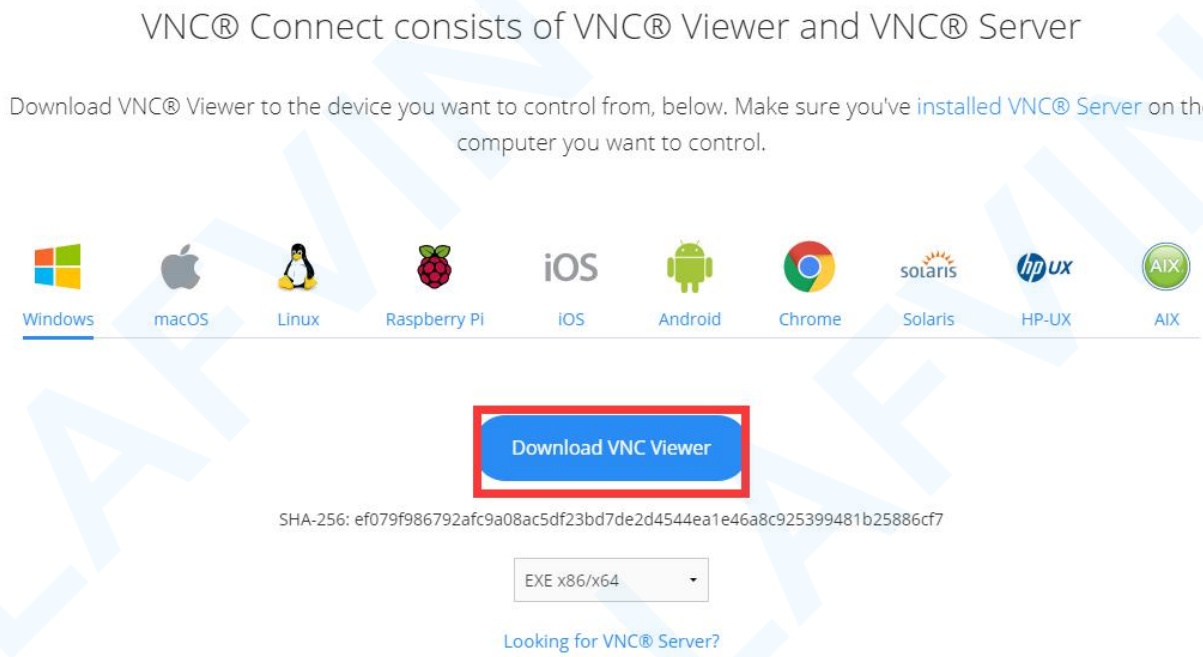


After waiting for the connection to succeed, the Raspberry Pi remote control desktop will appear.



Method 2: PC VNC Viewer<——>VNC Raspberry Pi

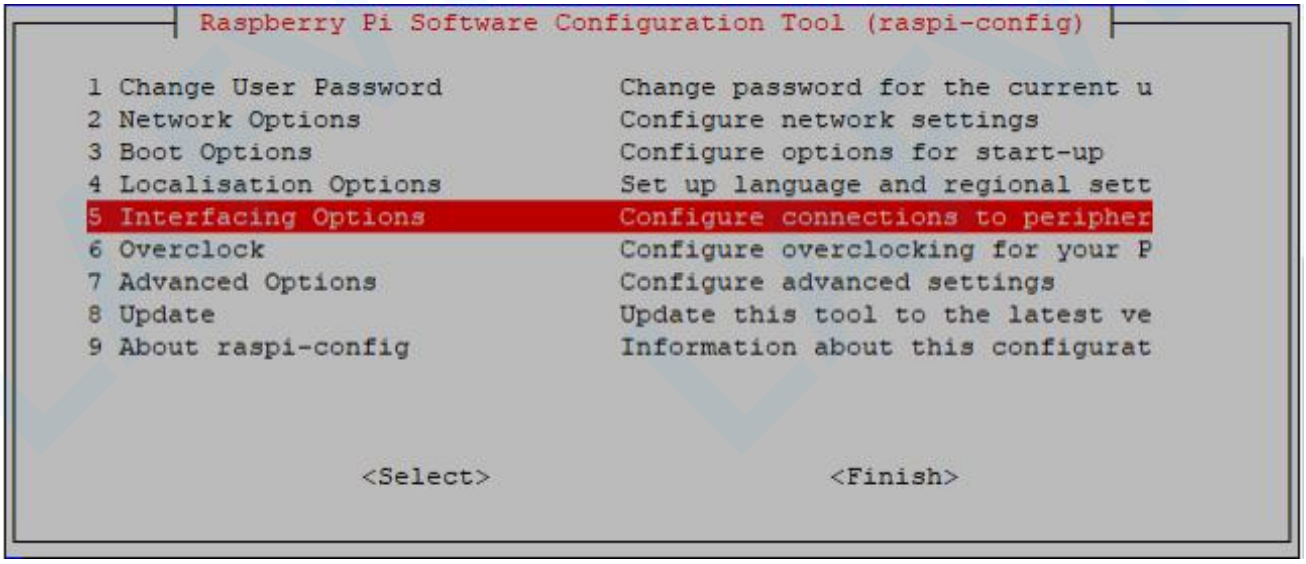
First download and install VNC on the PC.The download address is
https://www.realvnc.com/en/connect/download/viewer/windows/

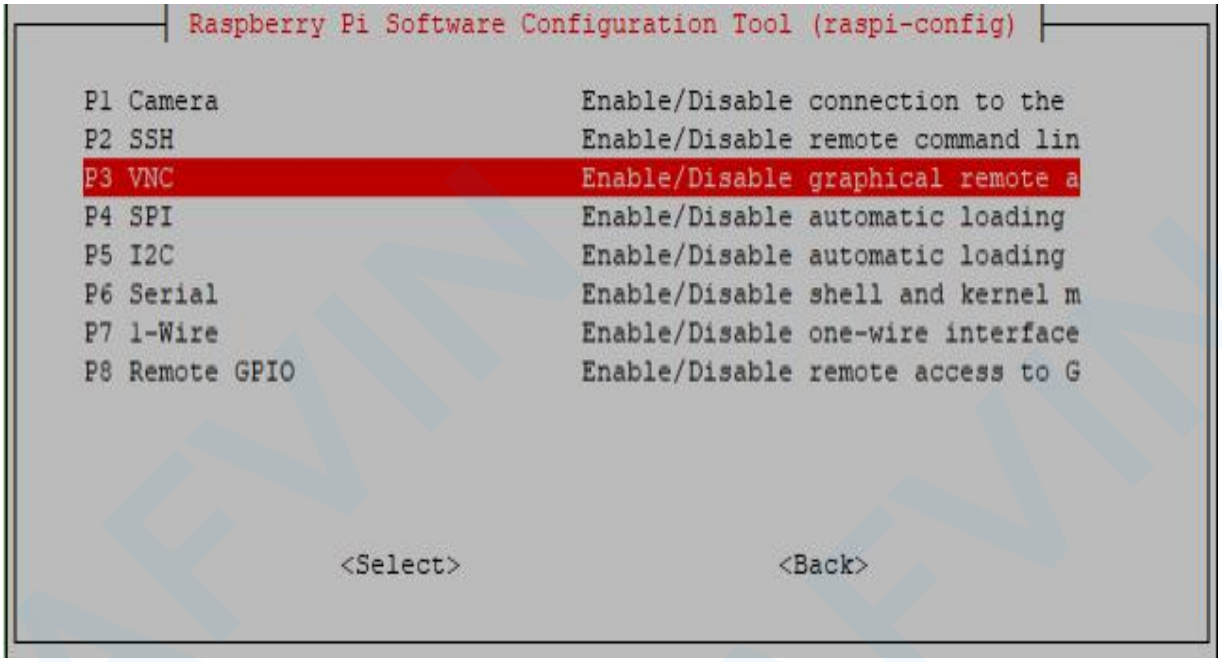


And then,type the following command on the putty.

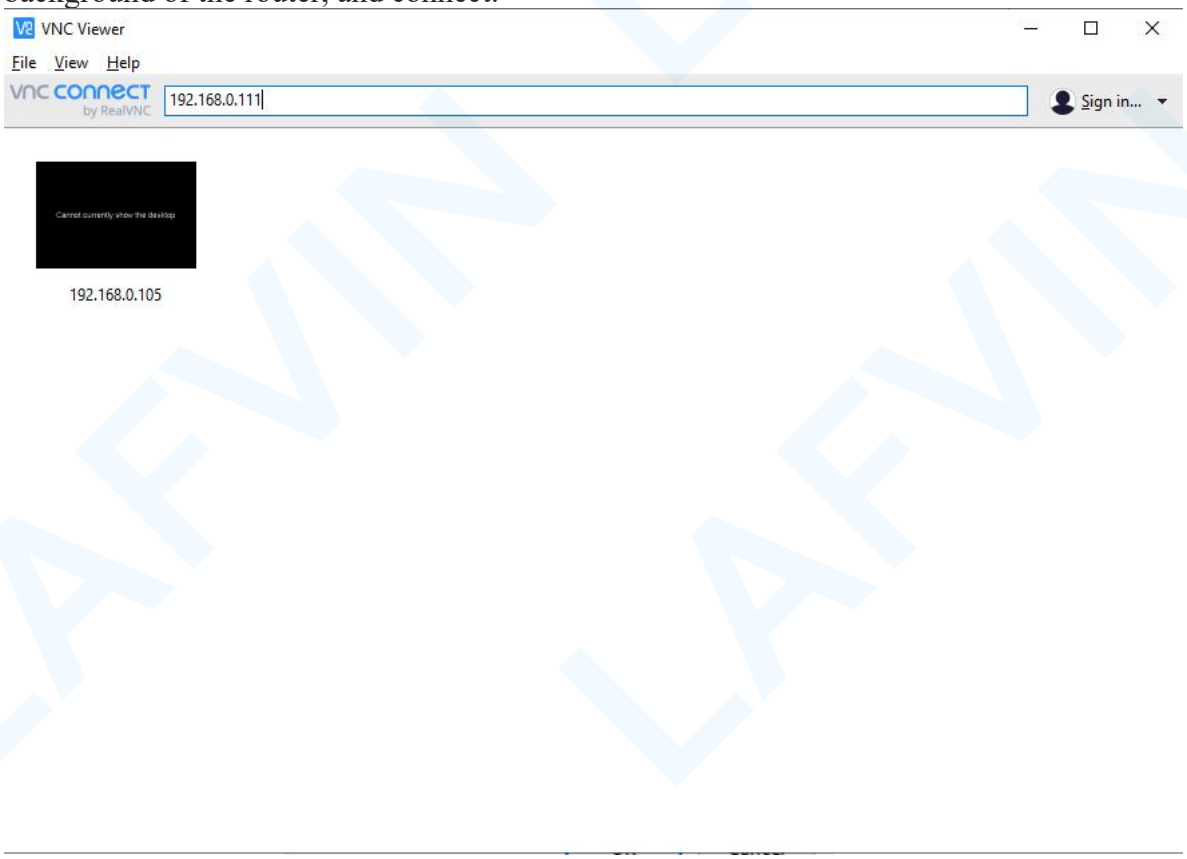
sudo raspi-config

The following interface will appear.
select 5 Interfacing Options-->P3VNC-->Yes-->OK-->Finish. Here Raspberry Pi may need be restarted, and choose ok. Then open VNC interface.



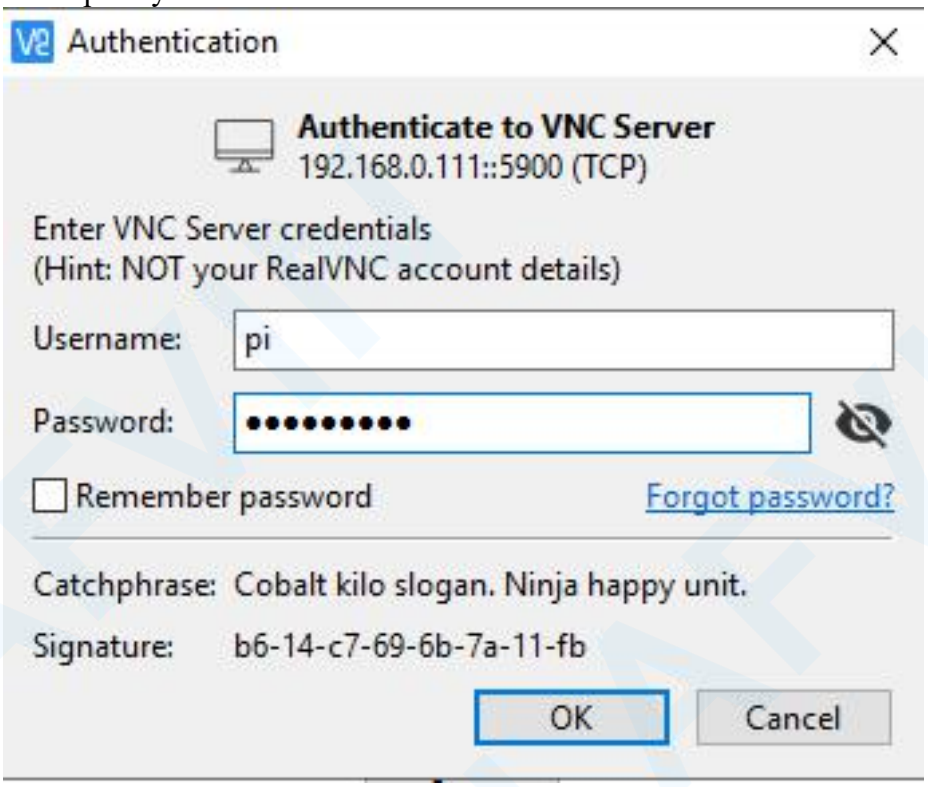


Open the VNC software, enter the IP address of the Raspberry Pi viewed in the background of the router, and connect.

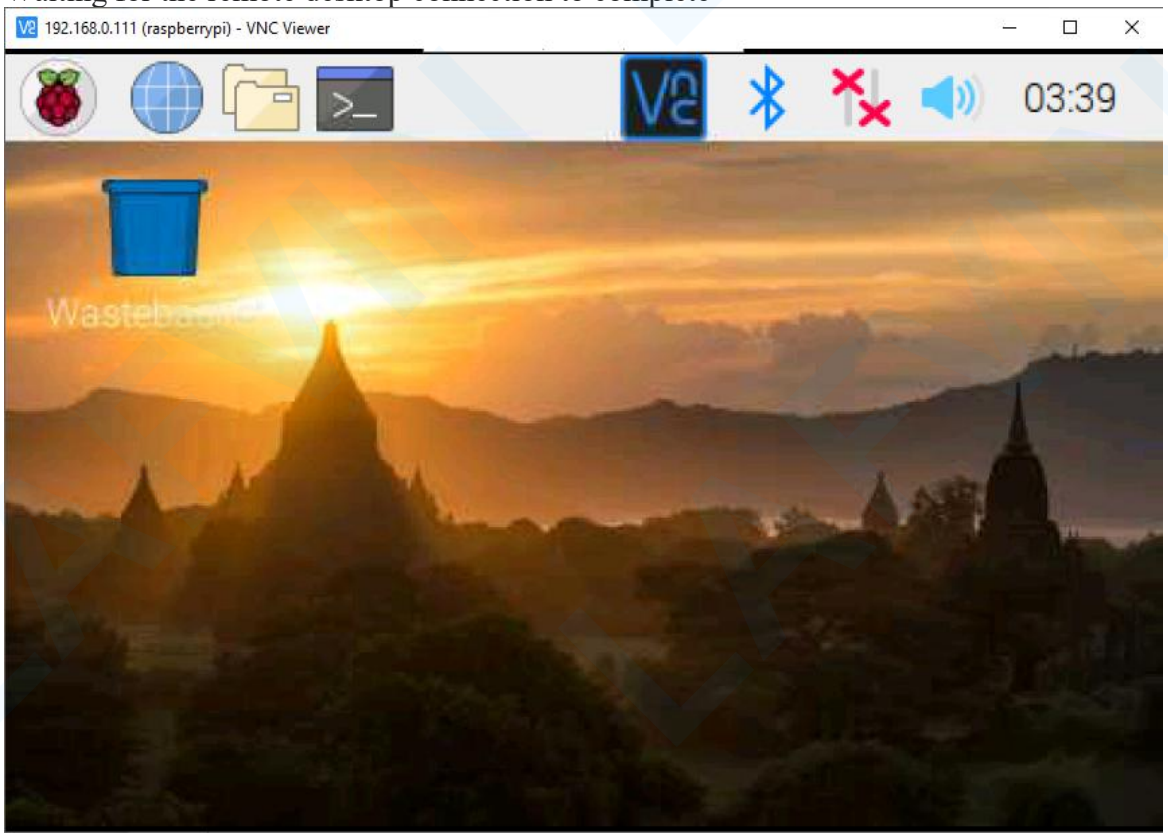


Enter account password to log in

RPi default user name: pi
password: raspberry



Waiting for the remote desktop connection to complete



GPIO Libraries

WiringPi

WiringPi is a GPIO library for C applied to the Raspberry Pi. It complies with GUN Lv3. The functions in wiringPi are similar to those in the wiring system of Arduino. They enable the users familiar with Arduino to use wiringPi more easily.

Now the Raspbian system has wiringPi pre-installed, you can use it directly.Test whether wiringPi is installed or not.WiringPi includes lots of GPIO commands which enable you to control all kinds of interfaces on Raspberry Pi. You can test whether the wiringPi library is installed successfully or not by the following instructions.

gpio –v

```
pi@raspberrypi:~ $ gpio -v
gpio version: 2.46
Copyright (c) 2012-2018 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
  Type: Pi 3+, Revision: 03, Memory: 1024MB, Maker: Sony
  * Device tree is enabled.
  *--> Raspberry Pi 3 Model B Plus Rev 1.3
  * This Raspberry Pi supports user-level GPIO access.
pi@raspberrypi:~ $
```

Then you can use the following instructions to see the distribution of the pins

gpio readall

```
pi@raspberrypi: ~
  * This Raspberry Pi supports user-level GPIO access.
pi@raspberrypi:~ $ gpio readall
 +-----+-----+---------+------+---+---Pi 3+---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 | ALT0 | 1 |  3 || 4  |   |      | 5v      |     |     |
 |   3 |   9 |   SCL.1 | ALT0 | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT5 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT5 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 1 | OUT  | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI | ALT0 | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO | ALT0 | 0 | 21 || 22 | 1 | OUT  | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK | ALT0 | 0 | 23 || 24 | 1 | OUT  | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | OUT  | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3+---+---+------+---------+-----+-----+
pi@raspberrypi:~ $
```

Besides the original name of the pins, there are other three ways of naming including physical,wiringPi and BCM. If you see a pin being defined as 0 in the C language, its original name is GPIO 0. In Python code, it's 17 (BCM) or 11(physical). So you need to know the name,physical, wiringPi and BCM of a pin.

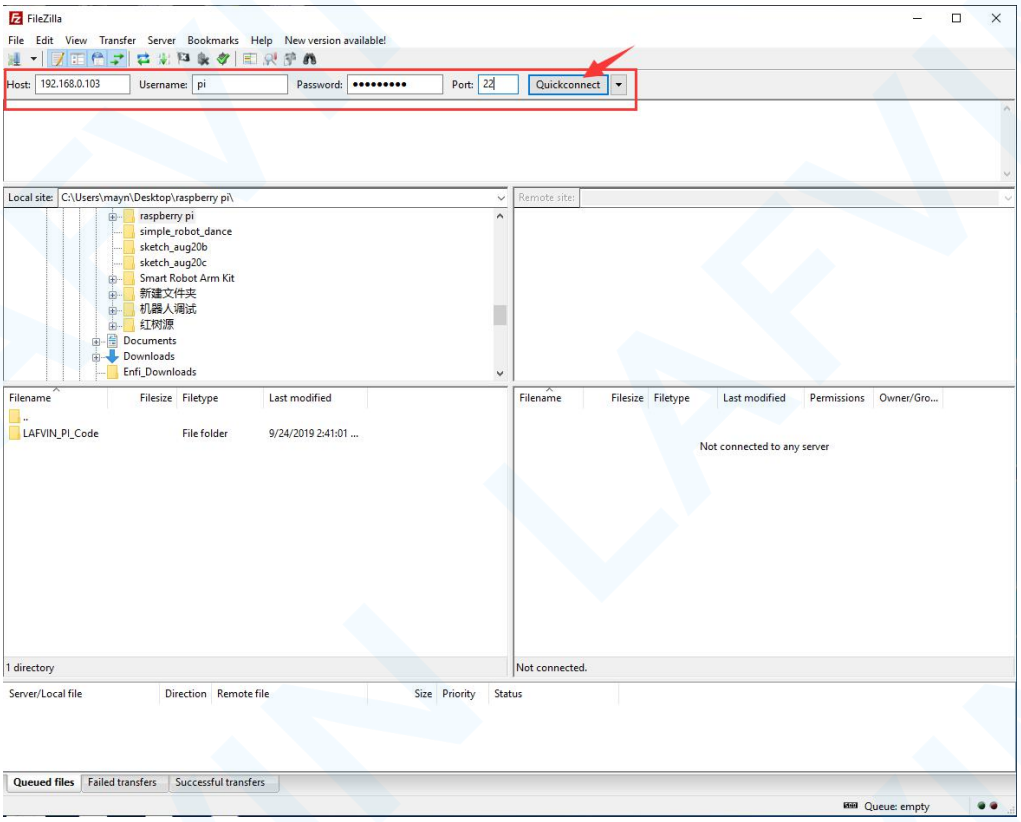
wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin		
—	—	3.3v	1 | 2	5v	—	—	For Pi B	For A+, B+, 2B, 3B, 3B+, Zero
8	R1:0/R2:2	SDA	3 | 4	5v	—	—		
9	R1:1/R2:3	SCL	5 | 6	0v	—	—		
7	4	GPIO7	7 | 8	TxD	14	15		
—	—	0v	9 | 10	RxD	15	16		
0	17	GPIO0	11 | 12	GPIO1	18	1		
2	R1:21/R2:27	GPIO2	13 | 14	0v	—	—		
3	22	GPIO3	15 | 16	GPIO4	23	4		
—	—	3.3v	17 | 18	GPIO5	24	5		
12	10	MOSI	19 | 20	0v	—	—		
13	9	MISO	21 | 22	GPIO6	25	6		
14	11	SCLK	23 | 24	CE0	8	10		
—	—	0v	25 | 26	CE1	7	11		
30	0	SDA.0	27 | 28	SCL.0	1	31		
21	5	GPIO.21	29 | 30	0V				
22	6	GPIO.22	31 | 32	GPIO.26	12	26		
23	13	GPIO.23	33 | 34	0V				
24	19	GPIO.24	35 | 36	GPIO.27	16	27		
25	26	GPIO.25	37 | 38	GPIO.28	20	28		
		0V	39 | 40	GPIO.29	21	29		
wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin		

Code operation

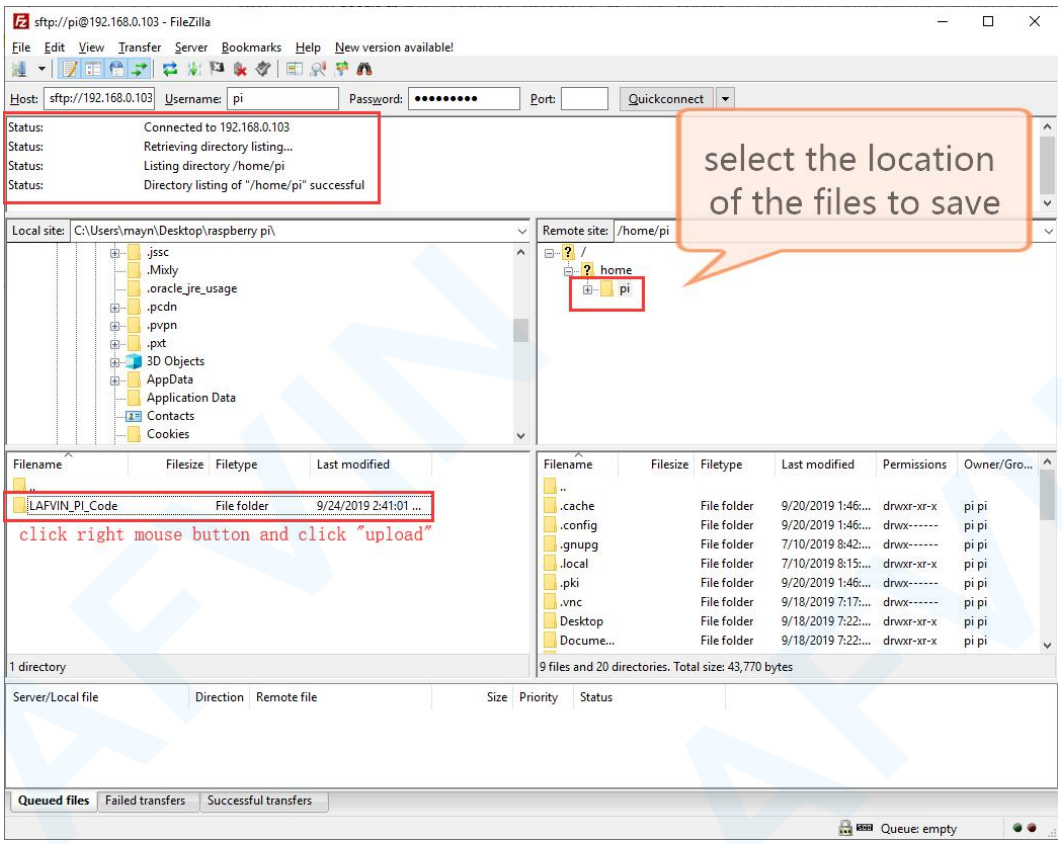
We need to transfer the code from the pc side to the Raspberry Pi operating platform. A common file transfer platform is FileZilla.

This software can be called the software folder in the information we provide.According to your computer's operating system (32-bit or 64-bit?) choose the appropriate version to install.

After downloading and installing, open the software, fill in the ip address of the Raspberry Pi, and the account password of the login, port number 22, click to connect

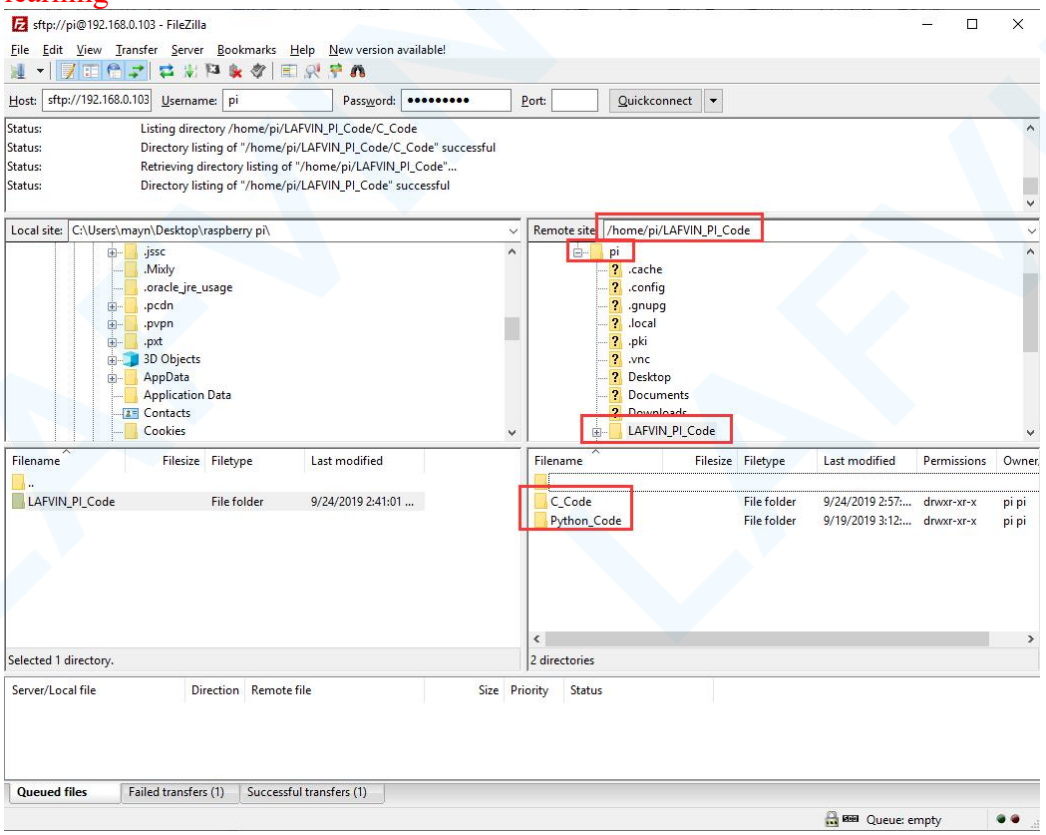


After the connection is successful, select the code folder "LAFVIN_PI_Code" to be uploaded on the left (provided in the information provided by the learning kit), and the path "/home/pi" saved in the Raspberry Pi on the right.



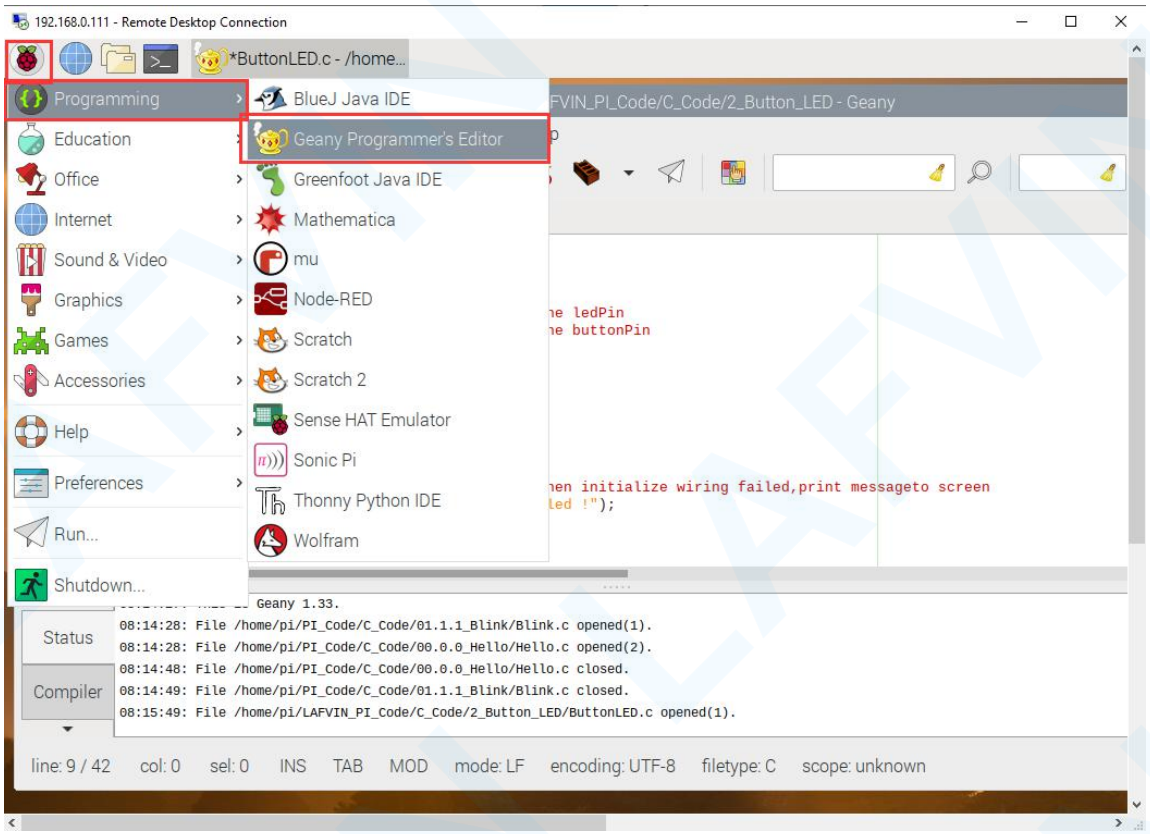
After successful upload, you will see the path of the code transferred from the PC side in the right dialog box is /home/pi/LAFVIN_PI_Code.

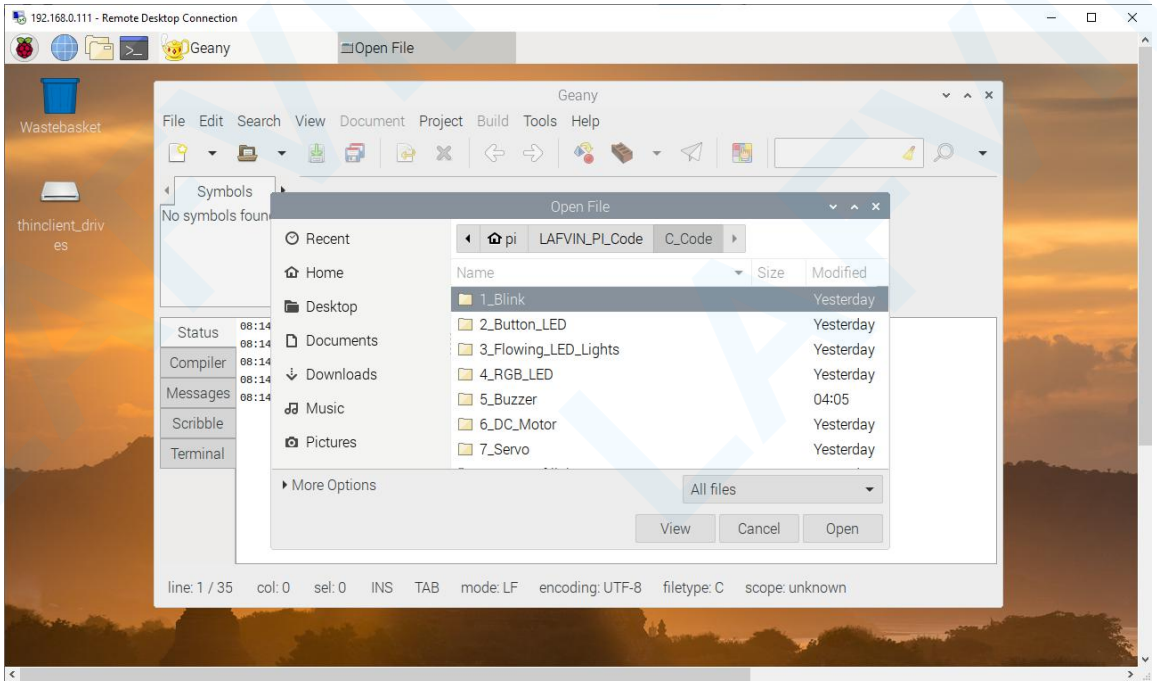
Note: This path is very important, then our program operation is directly from the /home/pi/LAFVIN_PI_Code path, if this path error will affect the subsequent course learning

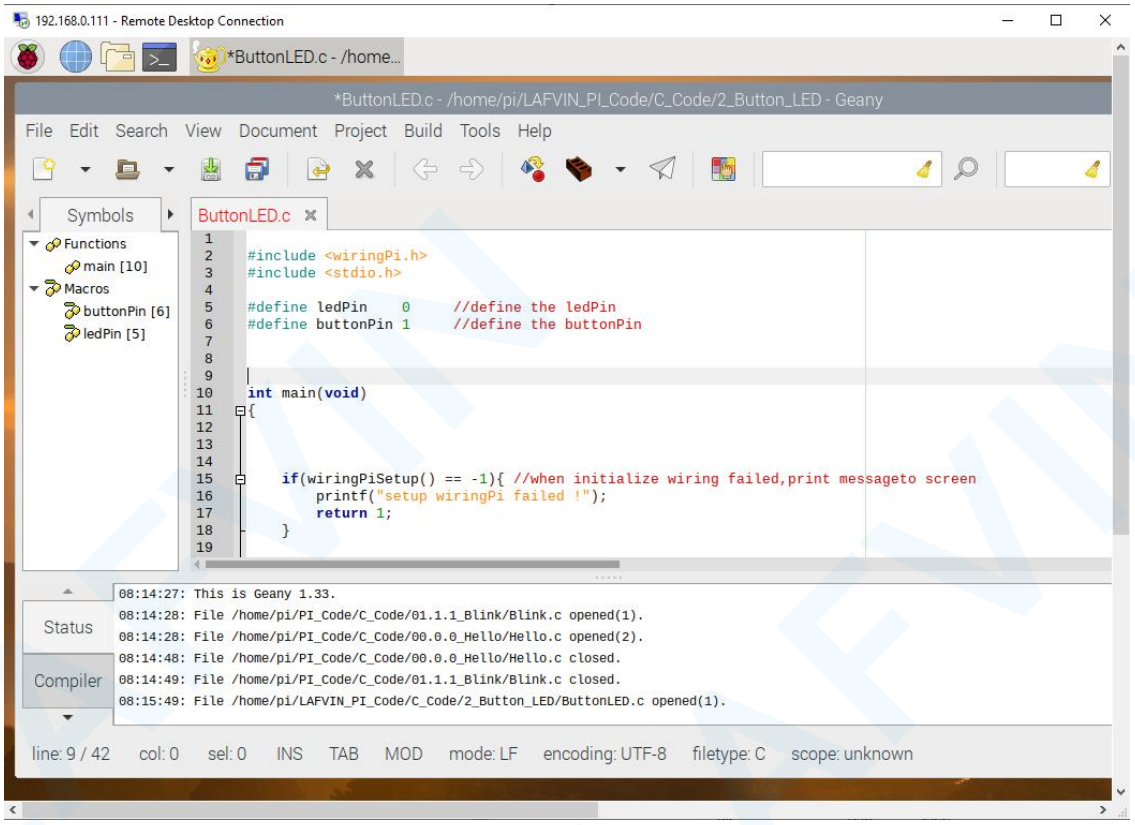


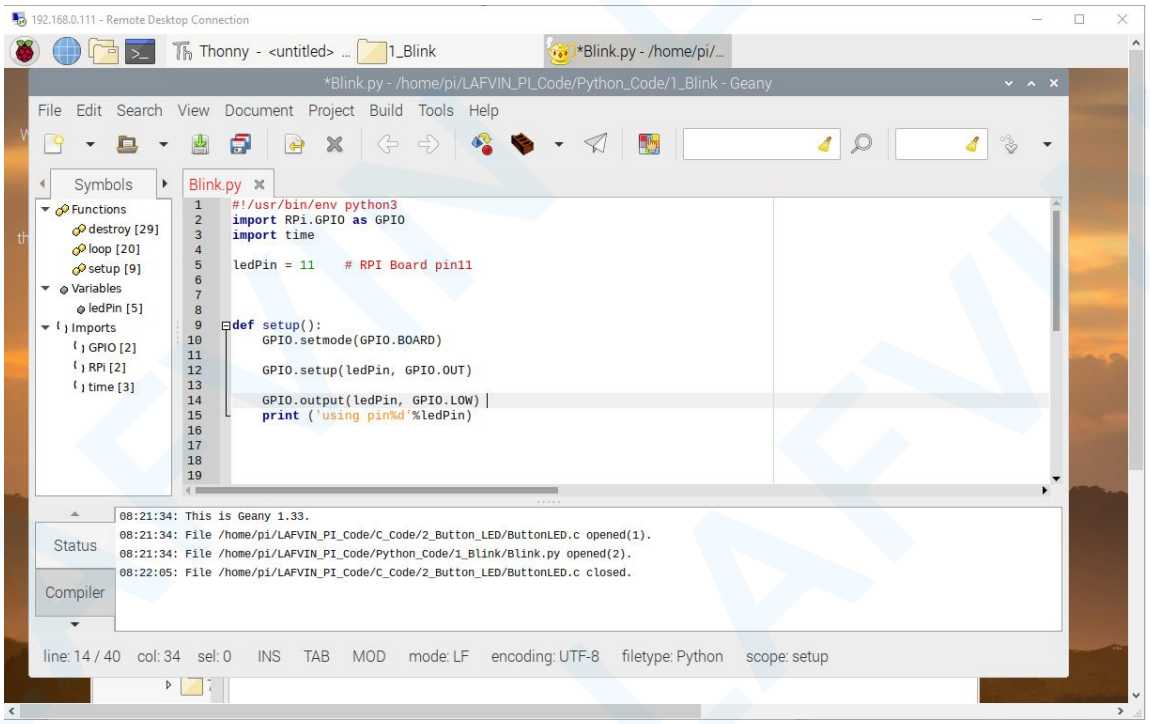
code editor

After successfully connecting to the remote desktop and transferring the code to the Raspberry Pi platform via the FileZilla software, you can edit the code in the program editor that comes with the remote desktop.






192.168.0.111 - Remote Desktop Connection
*ButtonLED.c - /home...
*ButtonLED.c - /home/pi/LAFVIN_PI_Code/C_Code/2_Button_LED - Geany
File Edit Search View Document Project Build Tools Help
Symbols
ButtonLED.c
Functions
main [10]
Macros
buttonPin [6]
ledPin [5]
#include <wiringPi.h>
#include <stdio.h>
#define ledPin 0 //define the ledPin
#define buttonPin 1 //define the buttonPin
int main(void)
{
if(wiringPiSetup() == -1){ //when initialize wiring failed,print messageto screen
printf("setup wiringPi failed !");
return 1;
}
Status
Compiler
08:14:27: This is Geany 1.33.
08:14:28: File /home/pi/PI_Code/C_Code/01.1.1_Blink/Blink.c opened(1).
08:14:28: File /home/pi/PI_Code/C_Code/00.0.0_Hello/Hello.c opened(2).
08:14:48: File /home/pi/PI_Code/C_Code/00.0.0_Hello/Hello.c closed.
08:14:49: File /home/pi/PI_Code/C_Code/01.1.1_Blink/Blink.c closed.
08:15:49: File /home/pi/LAFVIN_PI_Code/C_Code/2_Button_LED/ButtonLED.c opened(1).
line: 9 / 42 col: 0 sel: 0 INS TAB MOD mode: LF encoding: UTF-8 filetype: C scope: unknown


192.168.0.111 - Remote Desktop Connection
Thonny - <untitled> ...
1_Blink
*Blink.py - /home/pi/...
*Blink.py - /home/pi/LAFVIN_PI_Code/Python_Code/1_Blink - Geany
File Edit Search View Document Project Build Tools Help
Symbols
Blink.py
Functions
destroy [29]
loop [20]
setup [9]
Variables
ledPin [5]
Imports
GPIO [2]
RPi [2]
time [3]
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time
ledPin = 11 # RPI Board pin11
def setup():
GPIO.setmode(GPIO.BOARD)
GPIO.setup(ledPin, GPIO.OUT)
GPIO.output(ledPin, GPIO.LOW)
print ('using pin%d'%ledPin)
Status
Compiler
08:21:34: This is Geany 1.33.
08:21:34: File /home/pi/LAFVIN_PI_Code/C_Code/2_Button_LED/ButtonLED.c opened(1).
08:21:34: File /home/pi/LAFVIN_PI_Code/Python_Code/1_Blink/Blink.py opened(2).
08:22:05: File /home/pi/LAFVIN_PI_Code/C_Code/2_Button_LED/ButtonLED.c closed.
line: 14 / 40 col: 34 sel: 0 INS TAB MOD mode: LF encoding: UTF-8 filetype: Python scope: setup

Lesson 1 Blink

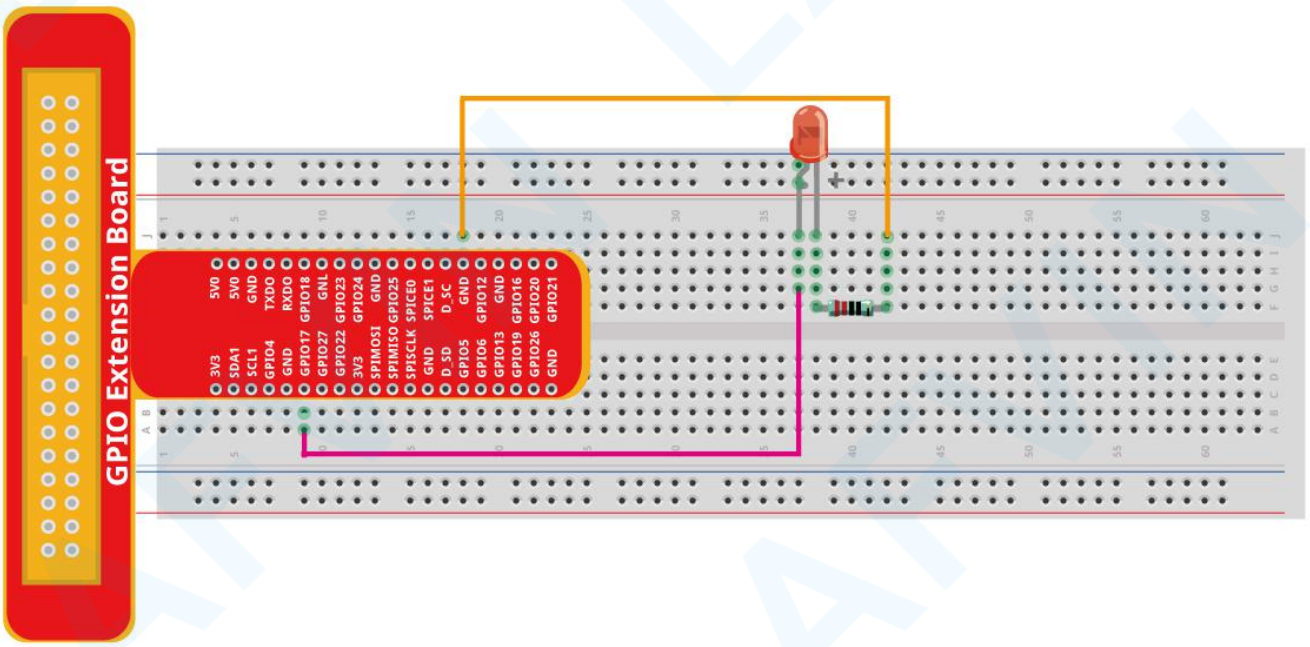
About this lesson:

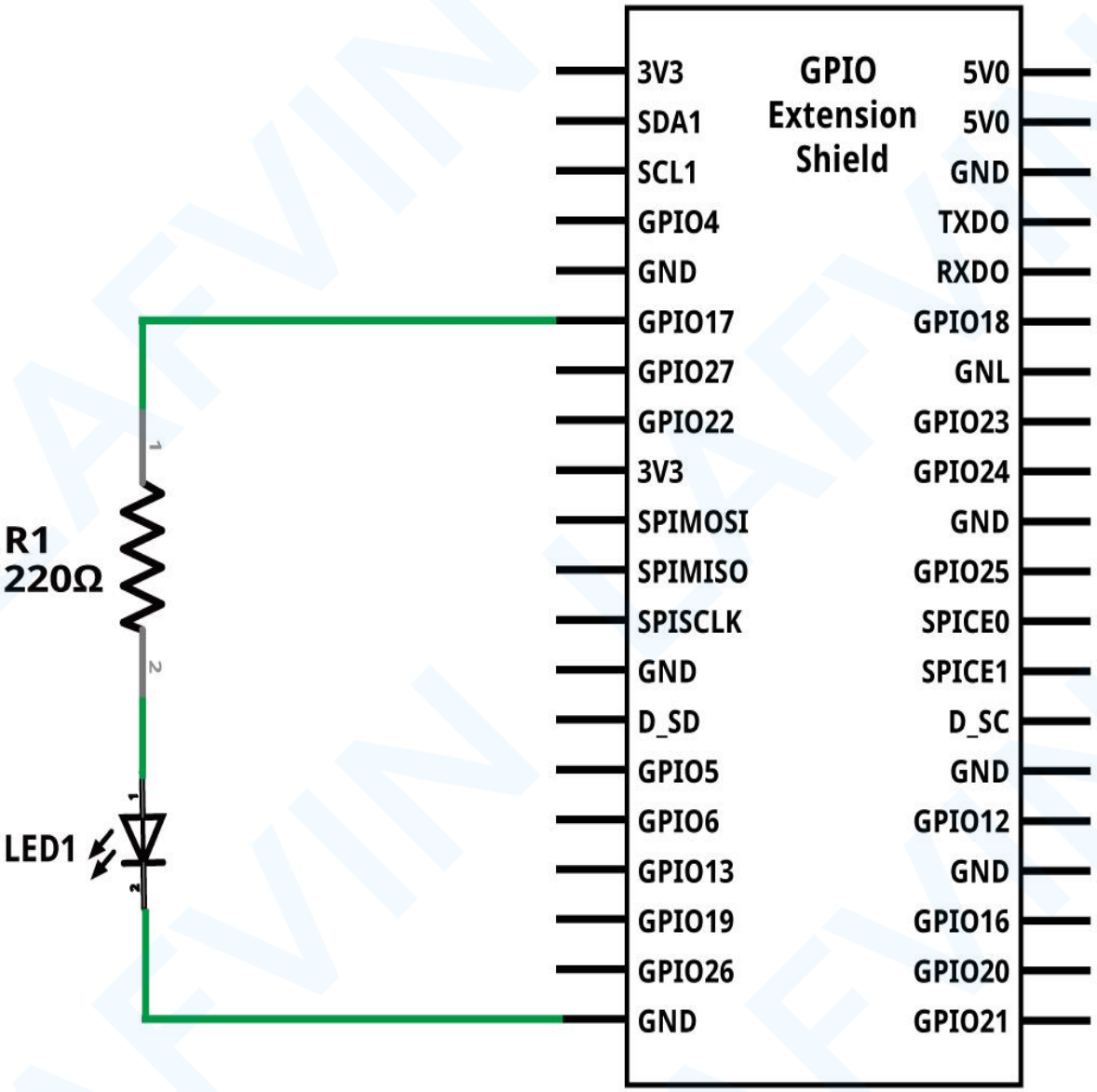
In this lesson, we will learn how to program Raspberry Pi to make an LED blink. You can play
numerous tricks with an LED as you want. Now get to start and you will enjoy the fun of DIY
at once!

Introduction

In this experiment, connect a 220Ω resistor to the anode (the long pin of the LED), then the resistor to 3.3 V, and connect the cathode (the short pin) of the LED to B17 of Raspberry Pi. We can see from the schematic diagram that the anode of LED connects to a current limiting resistor and then to 3.3V. **Therefore**, to turn on an LED, we need to make B17 low (0V) level. It can be realized by programming.

Wiring diagram




GPIO
Extension
Shield
3V3
SDA1
SCL1
GPIO4
GND
GPIO17
GPIO27
GPIO22
3V3
SPIMOSI
SPIMISO
SPISCLK
GND
D_SD
GPIO5
GPIO6
GPIO13
GPIO19
GPIO26
GND
5V0
5V0
GND
TXDO
RXDO
GPIO18
GNL
GPIO23
GPIO24
GND
GPIO25
SPICE0
SPICE1
D_SC
GND
GPIO12
GND
GPIO16
GPIO20
GPIO21
R1
220Ω
1
2
LED1
1
2

Test instructions

According to the circuit, when the GPIO17 of RPi output high level, LED is turned on. Conversely, when the
GPIO17 RPi output low level, LED is turned off. Therefore, we can let GPIO17 output high and low level in
cycle to make LED blink. We will use both C code and Python code to achieve the target.

C_Code_1_Blink

First, type command into the terminal, observe the project result, and then analyze the code.

1. Use cd command to enter 1_Blink directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/1_Blink
```

2. Use the following command to compile the code “Blink.c” and generate executable file “Blink”.

```
gcc Blink.c -o Blink -lwiringPi
```

3. Then run the generated file “blink”.

```
sudo ./Blink
```

Now, LED start blink. You can press “Ctrl+C” to end the program.

Code explanation

```
#include <wiringPi.h>
/*The hardware drive library designed for the C language of Raspberry Pi. Adding this library is convenient for hardware initialization, I/O ports,PWM outputs, etc.*/

#include <stdio.h>
/*Standard I/O library. The pintf function used for printing the data displayed on the screen is realized by this library. There are many other performance functions for you to explore.*/

#define LedPin 0 /*Pin B17 of the T_Extension Board is corresponding to the pin0 in wiringPi, namely, GPIO 0 of the raspberry Pi. Assign GPIO 0 to LedPin, LedPin represents GPIO 0 in the code later.*/

pinMode(LedPin, OUTPUT)// Set LedPin as output to write value to it.

digitalWrite(LedPin, LOW)
/*Set GPIO0 as 0V (low level). Since the cathode of LED is connected to GPIO0, thus the LED will light up if GPIO0 is set low. On the contrary, set GPIO0 as high level, digitalWrite (LedPin, HIGH): LED will go out.*/

void pinMode(int pin, int mode);
/*This sets the mode of a pin to either INPUT, OUTPUT, PWM_OUTPUT or GPIO_CLOCK. Note that only wiringPi pin 1 (BCM_GPIO 18) supports PWM output and only wiringPi pin 7 (BCM_GPIO 4) supports CLOCK output modes.This function has no effect when in Sys mode. If you need to change the pin mode, then you can do it with the gpio program in a script before you start your program*/

void digitalWrite (int pin, int value);
```

```
/*Writes the value HIGH or LOW (1 or 0) to the given pin which must have been previously set as an output.*/
```

Python_Code_1_Blink

Net, we will use Python language to make LED blink.

First, observe the project result, and then analyze the code.

1. Use cd command to enter 01.1.1_Blink directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/1_Blink
```

2. Use python command to execute python code blink.py.

```
Python Blink.py
```

Now, LED start blinking.

Code explanation

```
RPi.GPIO
" " "This is a Python module to control the GPIO on a Raspberry Pi. It includes basic output function and input function of GPIO, and function used to generate PWM. " " "


GPIO.setmode(mode)
#Set the mode for pin serial number of GPIO.

mode=GPIO.BOARD
#which represents the GPIO pin serial number is based on physical location of RPi.

mode=GPIO.BCM
#which represents the pin serial number is based on CPU of BCM chip.

GPIO.setup(pin,mode)
" " "Set pin to input mode or output mode. “pin” for the GPIO pin, “mode” for INPUT or OUTPUT." " "

GPIO.output(pin,mode)
" " "Set pin to output mode. “pin” for the GPIO pin, “mode” for HIGH (high level) or LOW (low level)." " "
```

Lesson 2 Button control LED

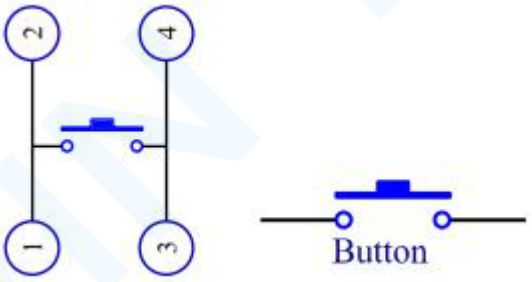
About this lesson:

In this lesson, we will learn how to turn an LED on or off by a button.

Introduction

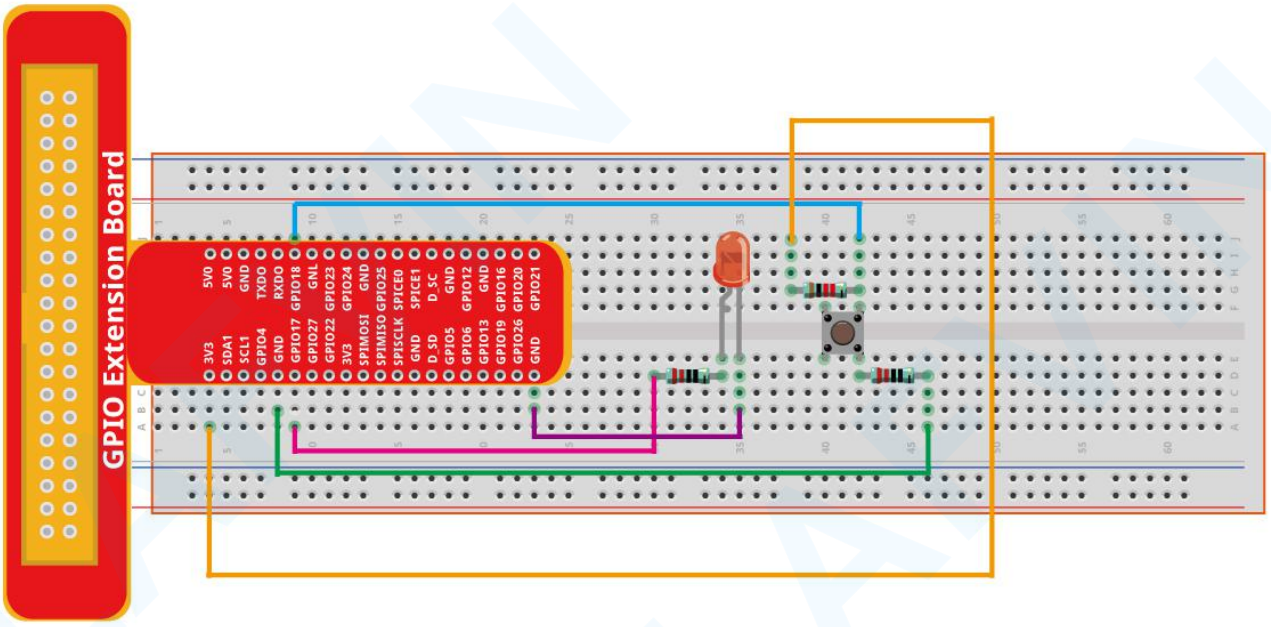
Button
Buttons are a common component used to control electronic devices. They are usually used as switches to connect or disconnect circuits. Although buttons come in a variety of sizes and shapes, the one used here is a 6mm mini-button as shown in the following pictures. Pin 1 is connected to pin 2 and pin 3 to pin 4. So you just need to connect either of pin 1 and pin 2 to pin 3 or pin 4.

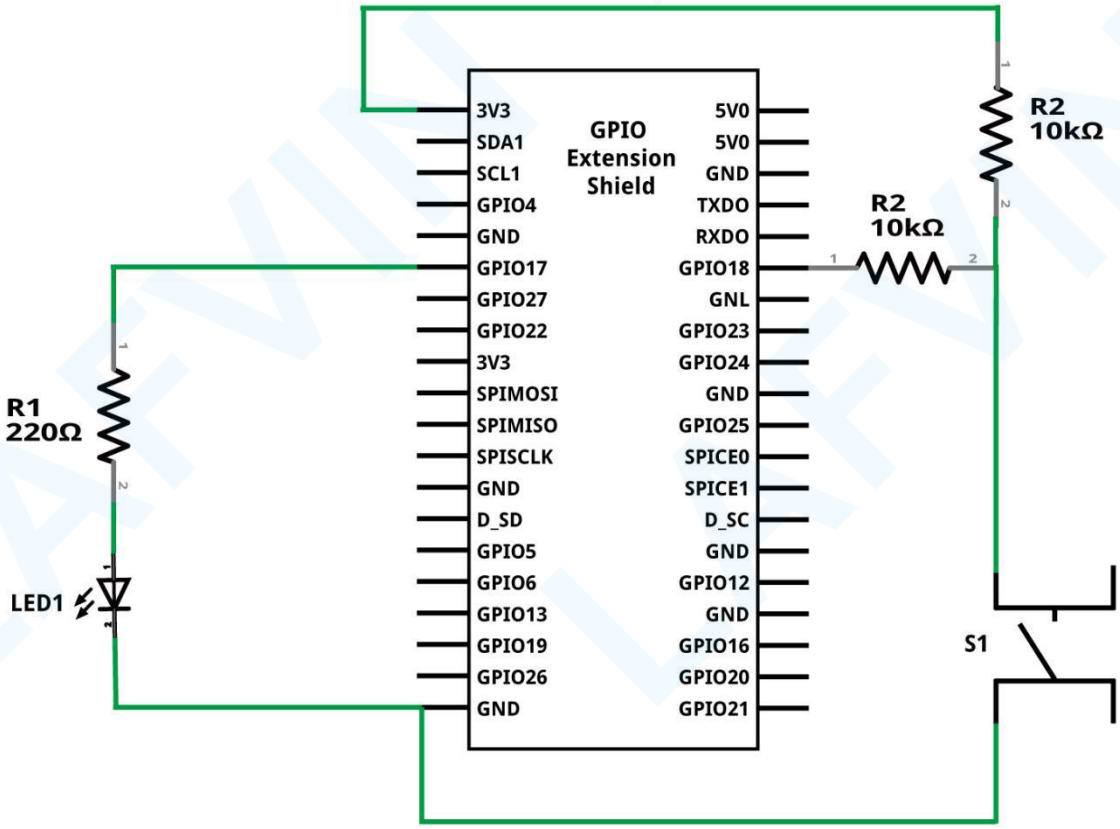
The following is the internal structure of a button. Since the pin 1 is connected to pin 2, and pin 3 to pin 4. The symbol on the right below is usually used to represent a button in circuits.



When the button is pressed, the 4 pins are connected, thus closing the circuit.
Use a normally open button as the input of Raspberry Pi, the detailed connection is as shown in the schematic diagram below. When the button is pressed, the B18 will turn into low level (0V). We can detect the state of the B18 through programming. That is, if the B18 turns into low level, it means the button is pressed. You can run the corresponding code when the button is pressed, and then the LED will light up. Note: The longer pin of the LED is the anode and the shorter one is the cathode.

Wiring diagram





Test instructions

This project is designed for learning how to use button to control LED. We first need to read the state of button, and then determine whether turn on LED according to the state of the button.

C_Code_2_Button_LED

First, observe the project result, then analyze the code.

1. Use cd command to enter 2_Button_LED directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/2_Button_LED
```

2. Use the following command to compile the code “Button_LED.c ” and generate executable file “Button_LED.c ”

```
gcc Button_LED.c -o Button_LED -lwiringPi
```

3. Then run the generated file “Button_LED”.

```
sudo ./Button_LED
```

Later, the terminal window continues to print out the characters “led off…”. Press the button, then LED is urned on and then terminal window prints out the "led on…". Release the button, then LED is turned off and then terminal window prints out the "led off…". You can press "Ctrl+C" to terminate the program.

Code explanation

```
int digitalRead (int pin);
/*This function returns the value read at the given pin. It will be “HIGH” or “LOW”(1 or 0) depending on the logic level at the pin.*/
```

Python_Code_2_Button_LED

First, observe the project result, then analyze the code.

1. Use cd command to enter 2_Button_LED directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/2_Button_LED
```

2. Use Python command to execute Button_LED.py.

```
python Button_LED.py
```

Later, the terminal window continue to print out the characters “led off…”, press the button, then LED is turned on and then terminal window print out the "led on…". Release the button, then LED is turned off and then terminal window print out the "led off…". You can press "Ctrl+C" to terminate the program.

Code explanation

```
GPIO.input()
" " "his function returns the value read at the given pin. It will be “HIGH” or “LOW”(1 or 0) depending on the logic level at the pin." " "
```

Lesson 3 Flowing LED Lights

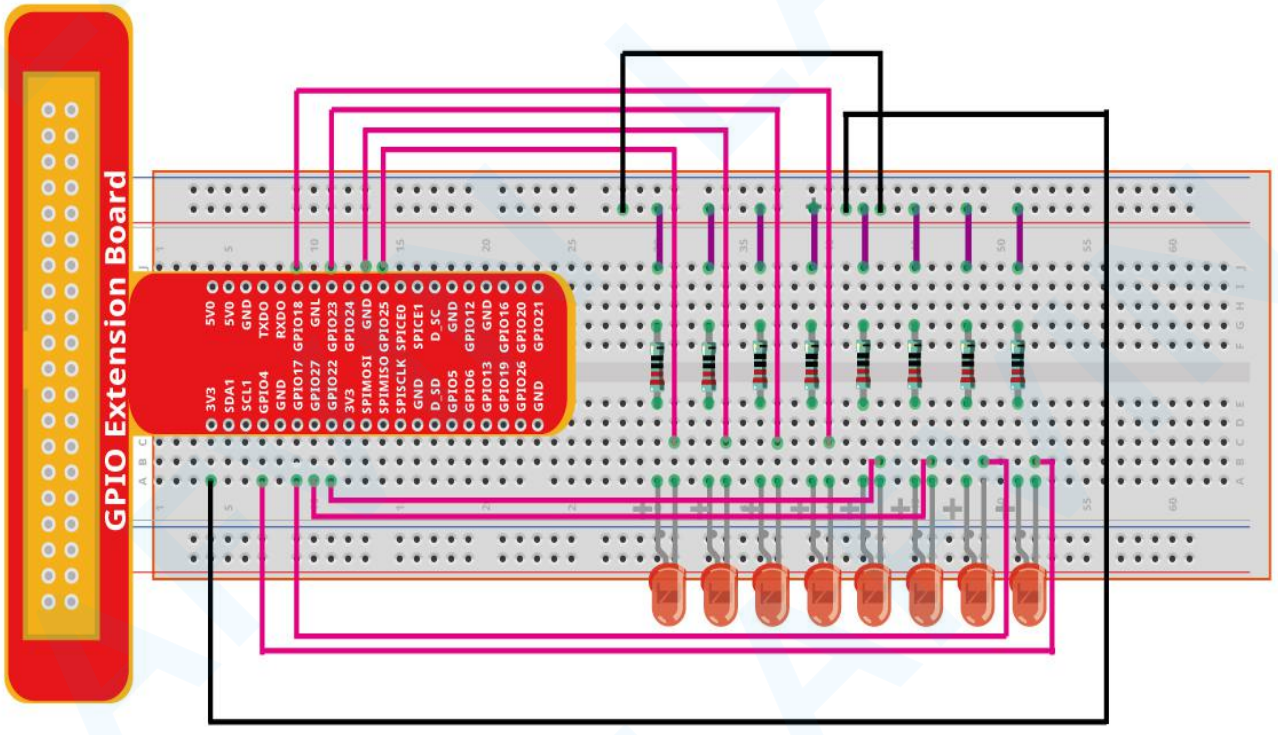
About this lesson:

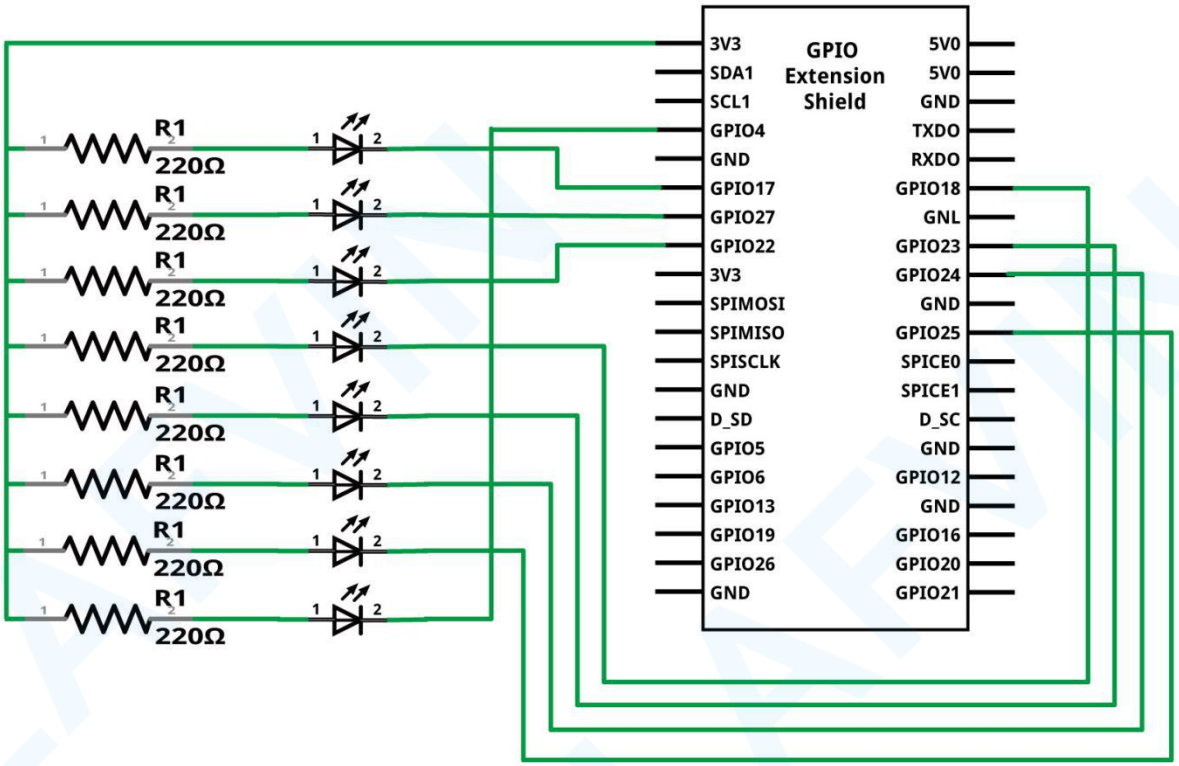
In this lesson, we will learn how to make eight LEDs blink in various effects as you want based on Raspberry Pi.

Introduction

Judging from the schematic diagram, we can know that a LED and a current-limiting resistor have been connected to B17, B18, B27, B22, B23, B24, B25, and B4 respectively.The current-limiting resistor has been connected to the 3.3V power supply on other side.Therefore, if we want to light up one LED, we only need to set the GPIO of the LED as low level. So in this experiment, set B17, B18, B27, B22, B23, B24, B25, and B4 to low level in turn by programming, and then LED0-LED7 will light up in turn. You can make eight LEDs blink in different effects by controlling their delay time and the order of lighting up.anode and the shorter one is the cathode.

Wiring diagram





Test instructions

This project is designed to make a water lamp. First turn on the first LED, then turn off it. Then turn on the second LED, and then turn off it..... Until the last LED is turned on, then is turned off. And repeats the process to achieve the effect of flowing water light.

C_Code_3_Flowing_LED_Lights

First, observe the project result, then analyze the code.

1. Use cd command to enter 3_Flowing_LED_Lights directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/3_Flowing_LED_Lights
```

2. Use following command to compile "Flowing_LED_Lights.c" and generate executable file "Flowing_LED_Lights".

```
gcc Flowing_LED_Lights.c -o Flowing_LED_Lights -lwiringPi
```

3. Then run the generated file "Flowing_LED_Lights".

```
sudo ./Flowing_LED_Lights
```

After the program is executed, you will see that LED starts with the flowing water way to be turned on from left to right, and then from right to left.

Python_Code_3_Flowing_LED_Lights

First, observe the project result, then analyze the code.

1. Use cd command to enter 3_Flowing_LED_Lights directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/3_Flowing_LED_Lights
```

2. Use Python command to execute Python code "Flowing_LED_Lights.py".

```
python Flowing_LED_Lights.py
```

After the program is executed, you will see that LED starts with the flowing water way to be turned on from left to right, and then from right to left.

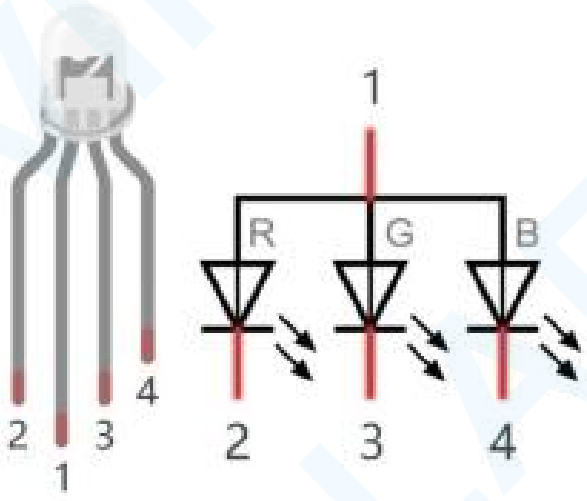
Lesson 4 RGB LED

About this lesson:

In this chapter, we will learn how to control a RGB LED.

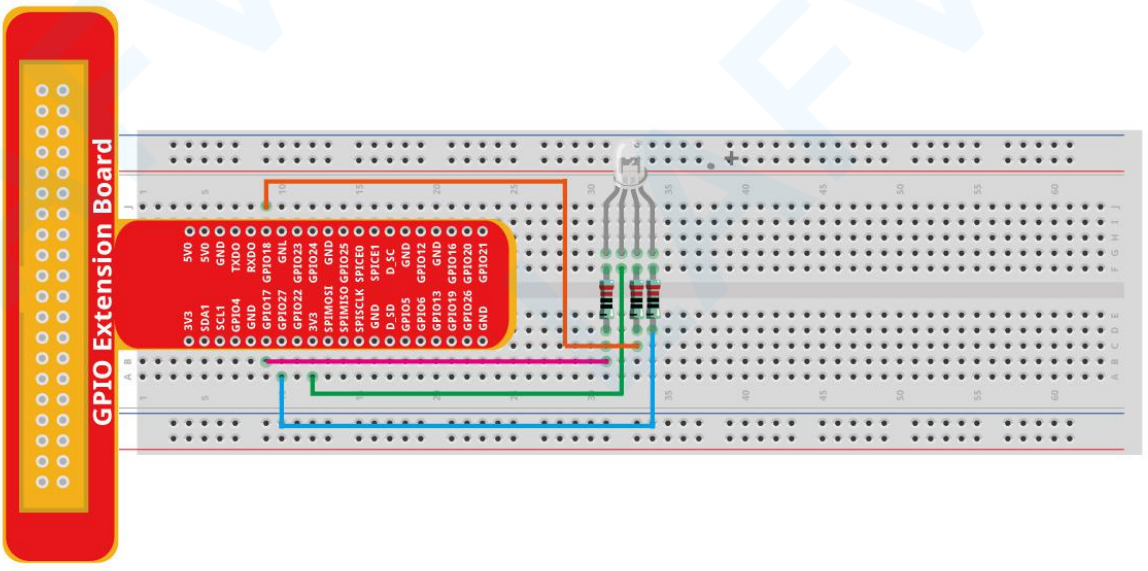
Introduction

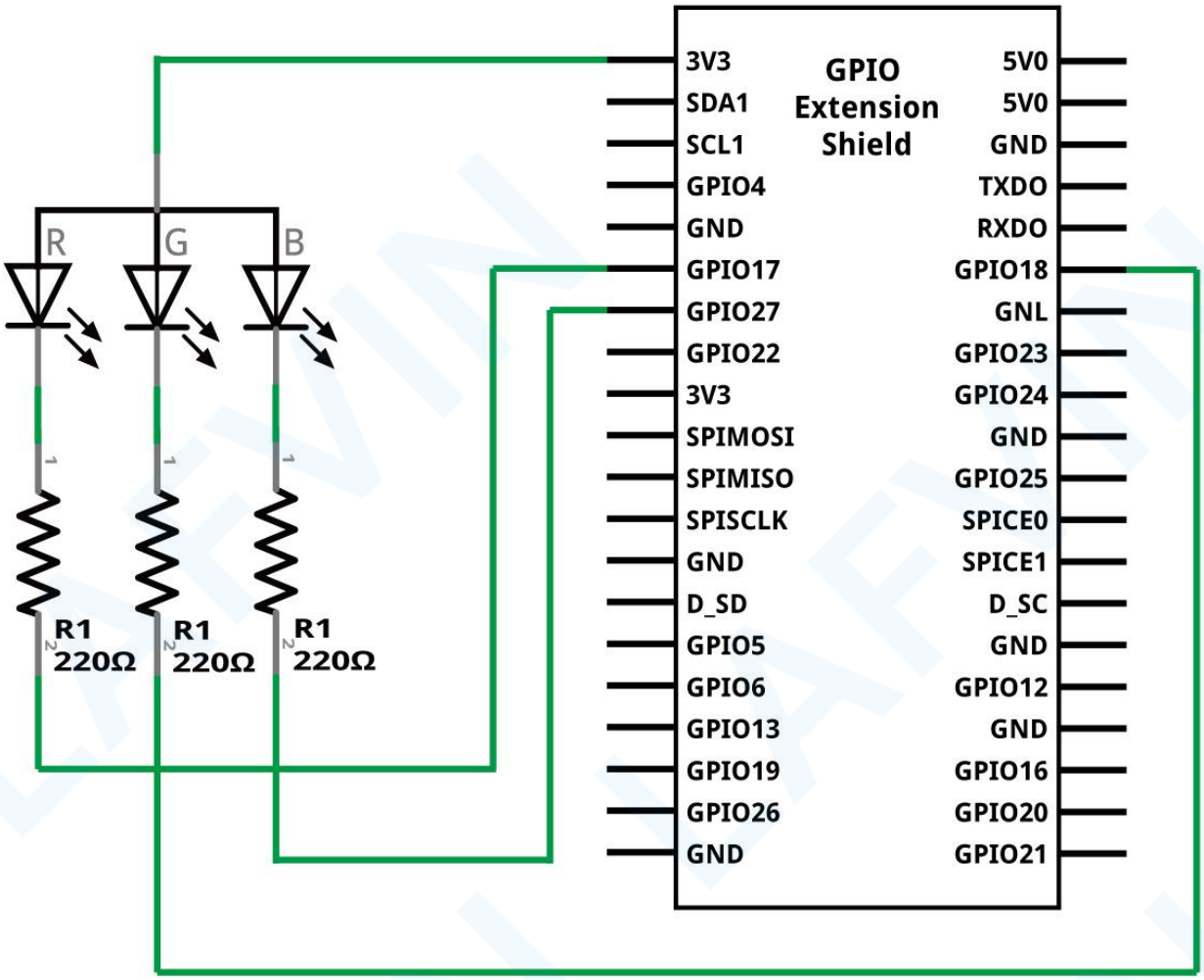
RGB LED has integrated 3 LEDs that can respectively emit red, green and blue light. And it has 4 pins. The long pin (1) is the common port, that is, 3 LED 's positive or negative port. The RGB LED with common positive port and its symbol are shown below.



The three primary colors of the RGB LED can be mixed into various colors by brightness. The brightness of LED can be adjusted with PWM. the softPwm library simulates PWM (softPwm) by programming. You only need to include the header file softPwm.h (for C language users), and then call the API it provides to easily control the RGB LED by multi-channel PWM output, so as to display all kinds of color.

Wiring diagram





Test instructions

Since this project requires 3 PWM, but in RPi, only one GPIO has the hardware capability to output PWM, we need to use the software to make the ordinary GPIO output PWM.

C_Code_4_RGB_LED

First, observe the project result, then analyze the code.

1Use cd command to enter 4_RGB_LED directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/4_RGB_LED
```

2. Use the following command to compile the code “RGB_LED.c ” and generate executable file “RGB_LED.c ”

```
gcc RGB_LED.c –o RGB_LED -lwiringPi
```

3. Then run the generated file “RGB_LED”.

```
sudo ./RGB_LED
```

After the program is executed, you will see that the RGBLED shows light of different color randomly.You can press "Ctrl+C" to terminate the program.

Code explanation

```
int softPwmCreate (int pin, int initialValue, int pwmRange) ;
// This creates a software controlled PWM pin.
void softPwmWrite (int pin, int value) ;
// This updates the PWM value on the given pin.
long random();
//This function will return a random number.
```

Python_Code_4_RGB_LED

First, observe the project result, then analyze the code.

1. Use cd command to enter 4_RGB_LED directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/4_RGB_LED
```

2. Use Python command to execute RGB_LED.py.

```
python RGB_LED.py
```

After the program is executed, you will see that the RGBLED shows light of different color randomly. You can press "Ctrl+C" to terminate the program.

Code explanation

```
def loop ():
    while    True : :
        r= =random. .randint( (0, ,100) )
        g= =random. .randint( (0, ,100) )
        b= =random. .randint( (0, ,100) )
        setColor( (r, ,g, ,b) )
        print ( ('r=%d, g=%d, b=%d '    %(r , ,g, , b ))
        time. .sleep( (0.3) )
" " "The function random.randint(a, b) can returns a random integer within the specified range (a, b).In last chapter, we have learned how to use python language to make a pin output PWM. In this project, we let three pins output PWM, and the usage is exactly the same as last chapter. In the “while” cycle of “loop”function, we first obtain three random numbers, and then specify these three random numbers as the PWM value of the three pins.o that the RGBLED switching of different colors randomly." " "
```

Lesson 5 Buzzer

About this lesson:

n this lesson, we will learn how to drive an active buzzer to beep with a PNP transistor.

Introduction

BUZZER:

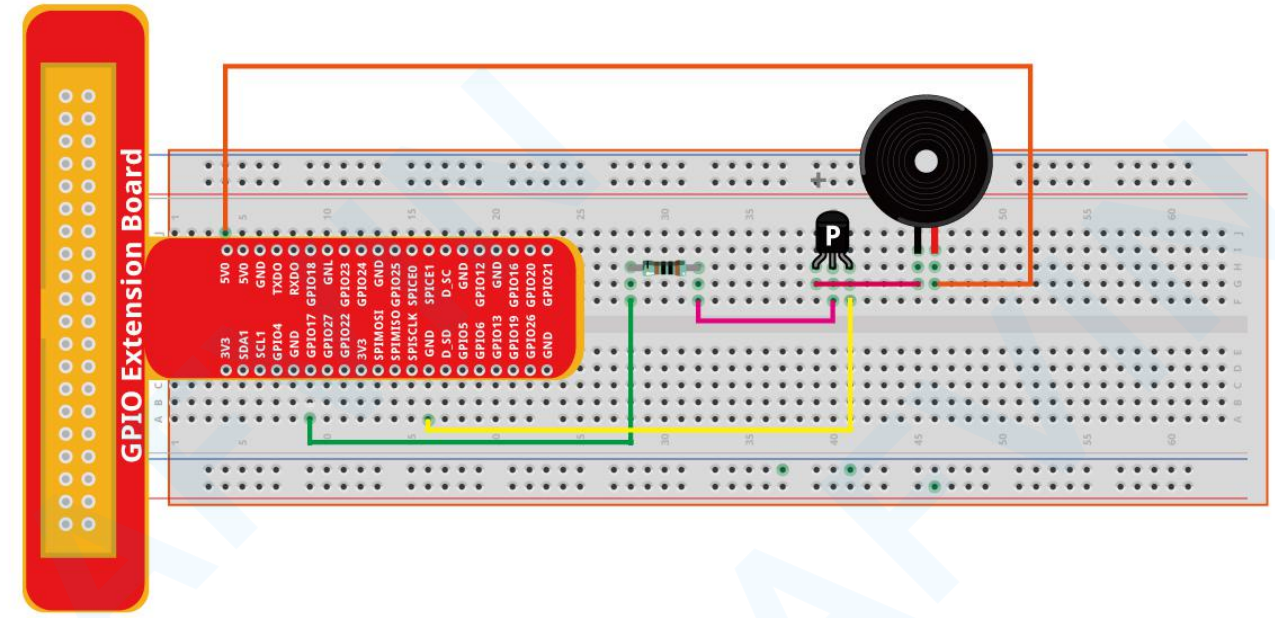
Electronic buzzers are DC-powered and equipped with an integrated circuit. They are widely used in computers, printers, photocopiers, alarms, electronic toys, automotive electronic devices, telephones, timers and other electronic products for voice devices. Buzzers can be categorized as active and passive ones. Turn the pins of two buzzers face up. The one with a green circuit board is a passive buzzer, while the other enclosed with a black tape is an active one.

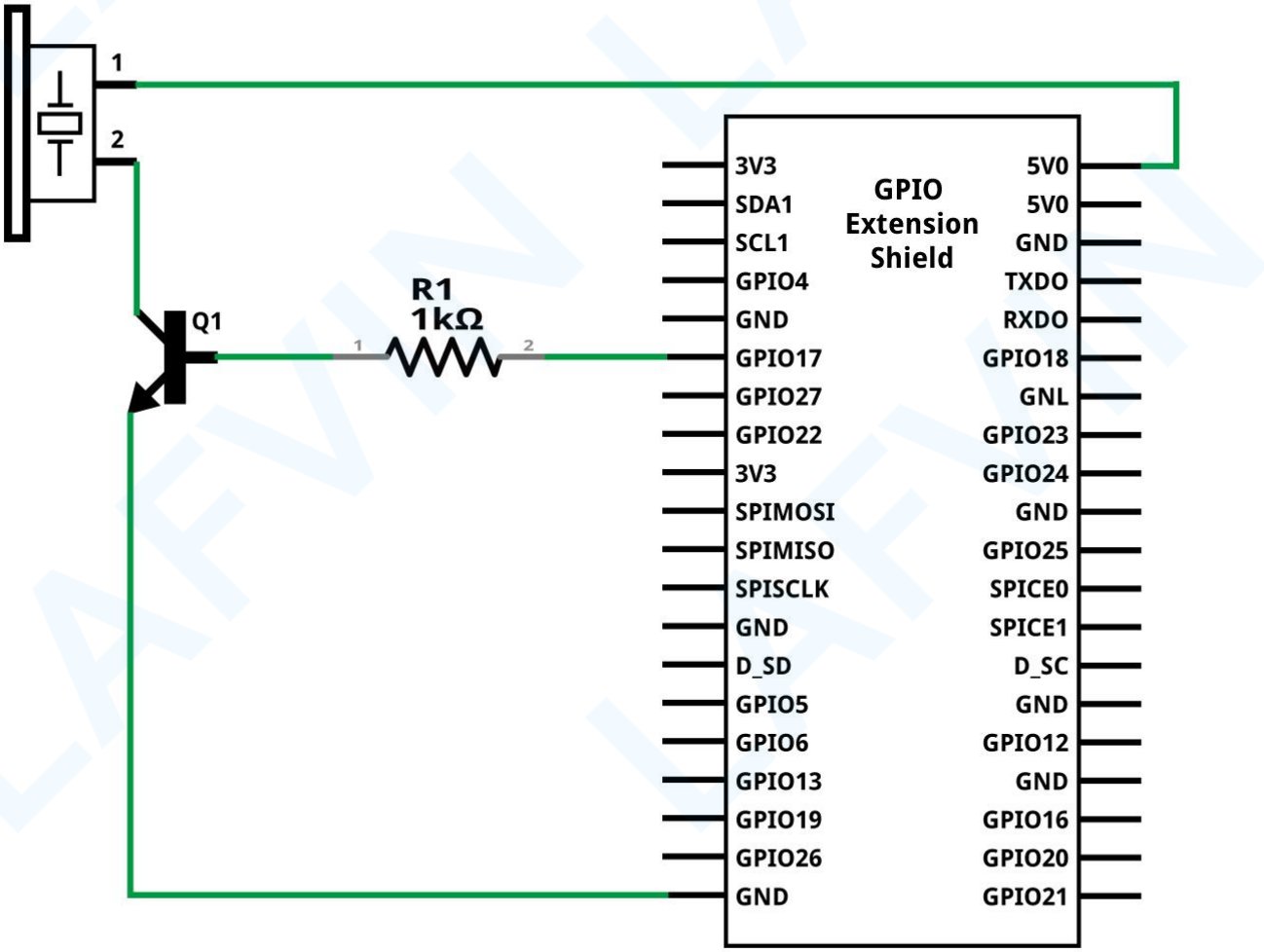
The difference between the two is that an active buzzer has a built-in oscillating source, so it will generate a sound when electrified. A passive buzzer does not have such a source so it will not tweet if DC signals are used; instead, you need to use square waves whose frequency is between 2K and 5K to drive it. The active buzzer is often more expensive than the passive one because of multiple built-in oscillating circuits.

Transistor：

Due to the current operating of buzzer is so large that GPIO of RPi output capability can not meet the requirement. A transistor of NPN type is needed here to amplify the current. Transistor, the full name: semiconductor transistor, is a semiconductor device that controls current. Transistor can be used to amplify weak signal, or works as a switch. It has three electrodes(PINs): base (b), collector (c) and emitter (e). When there is current passing between "be", "ce" will allow several-fold current (transistor magnification) pass, at this point, transistor works in the amplifying area. When current between "be" exceeds a certain value, "ce" will not allow current to increase any longer, at this point, transistor works in the saturation area. Transistor has two types shown below: PNP and NPN,

Wiring diagram





Test instructions

In this project, Raspberry Pi control buzzer sounds for a while, stops and waits for a while, repeats the loop.

C_Code_5_Buzzer

First, observe the project result, then analyze the code.

1Use cd command to enter 5_Buzzer directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/5_Buzzer
```

2. Use the following command to compile the code “Buzzer.c ” and generate executable file “Buzzer.c ”

```
gcc Buzzer.c -o Buzzer -lwiringPi
```

3. Then run the generated file “Buzzer”.

```
sudo ./Buzzer
```

Code explanation

```
digitalWrite(BeepPin, LOW);
/* We use an active buzzer in this experiment, so it will make sound automatically when connecting to the direct current. This sketch is to set the I/O port as low level (0V), thus to manage the transistor and make the buzzer beep.*/
digitalWrite(BeepPin, HIGH);
/* To set the I/O port as high level(5V), thus the transistor is not energized and the buzzer doesn’t beep.*/
```

Python_Code_5_Buzzer

First, observe the project result, then analyze the code.

1. Use cd command to enter 5_Buzzer directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/5_Buzzer
```

2. Use Python command to execute Buzzer.py.

```
python Buzzer.py
```

Code explanation

```
GPIO.output(BeepPin, GPIO.LOW)# Set the buzzer pin as low level.

time.sleep(0.1)
" " "Wait for 0.1 second. Change the switching frequency by changing this parameter. Note: Not the sound frequency. Active Buzzer cannot change sound frequency." " "

GPIO.output(BeepPin, GPIO.HIGH) # close the buzzer

time.sleep(0.1)
" " "Wait for 0.1 second. Change the switching frequency by changing this parameter. Note: Not the sound frequency. Active Buzzer cannot change sound frequency." " "
```

Lesson 6 DC Motor

About this lesson:

In this lesson, we will learn to how to use L293D to drive a DC motor and make it rotate clockwise and counterclockwise. Since the DC Motor needs a larger current, for safety purpose, here we use the Power Supply Module to supply motors.

Introduction

Breadboard Power Supply

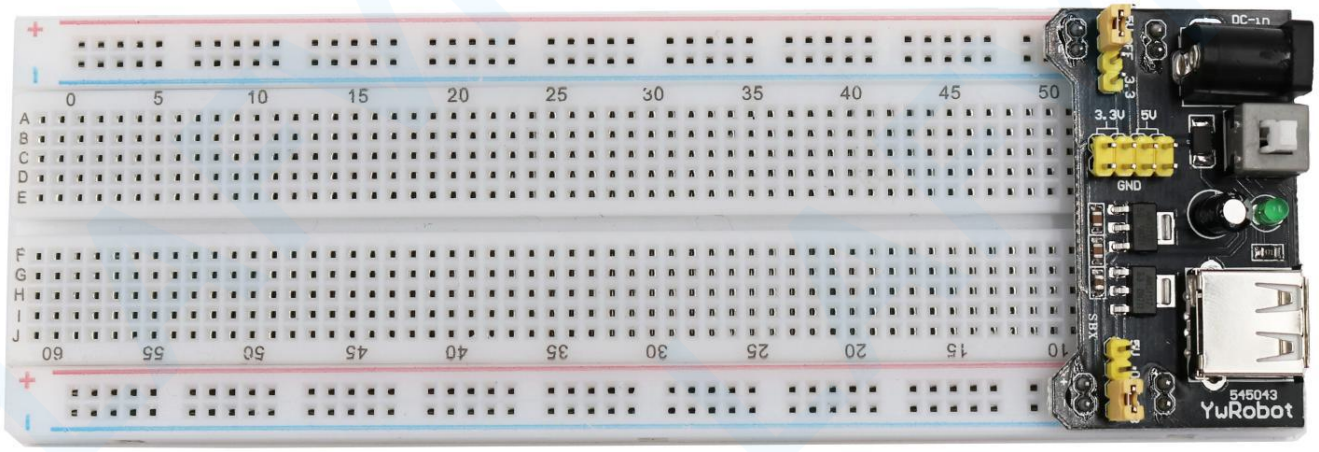
The small DC motor is likely to use more power than an UNO R3 board digital output can handle directly. If we tried to connect the motor straight to an UNO R3 board pin, there is a good chance that it could damage the UNO R3 board. So we use a power supply module provides power supply

Product Specifications:

- Locking On/Off Switch
- LED Power Indicator
- Input voltage: 6.5-9v (DC) via 5.5mm x 2.1mm plug
- Output voltage: 3.3V/5v
- Maximum output current: 700 mA
- Independent control rail output. 0v, 3.3v, 5v to breadboard
- Output header pins for convenient external use
- Size: 2.1 in x 1.4 in
- USB device connector onboard to power external device

Setting up output voltage:

The left and right voltage output can be configured independently. To select the output voltage, move jumper to the corresponding pins. Note: power indicator LED and the breadboard power rails will not power on if both jumpers are in the “OFF” position

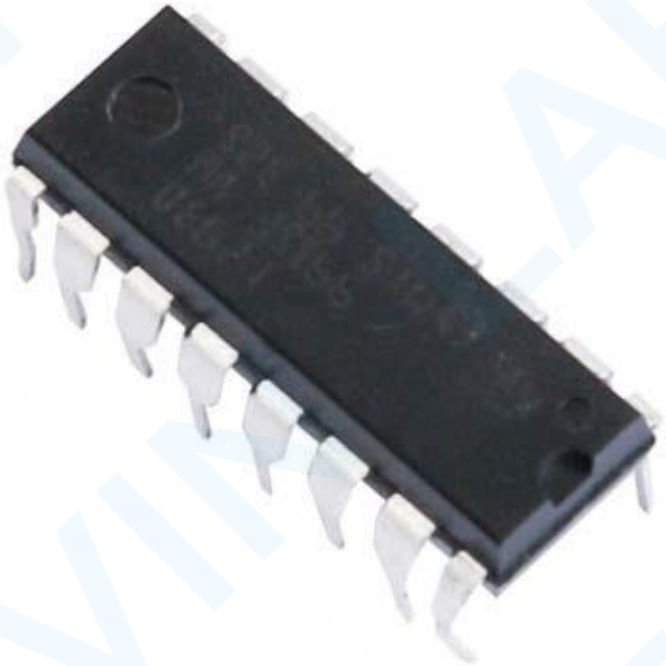


Important note:

Make sure that you align the module correctly on the breadboard. The negative pin(-) on module lines up with the blue line(-) on breadboard and that the positive pin(+) lines up with the red line(+). Failure to do so could result in you accidently reversing the power to your project

L293D

This is a very useful chip. It can actually control two motors independently. We are just using half the chip in this lesson, most of the pins on the right hand side of the chip are for controlling a second motor.



Product Specifications:

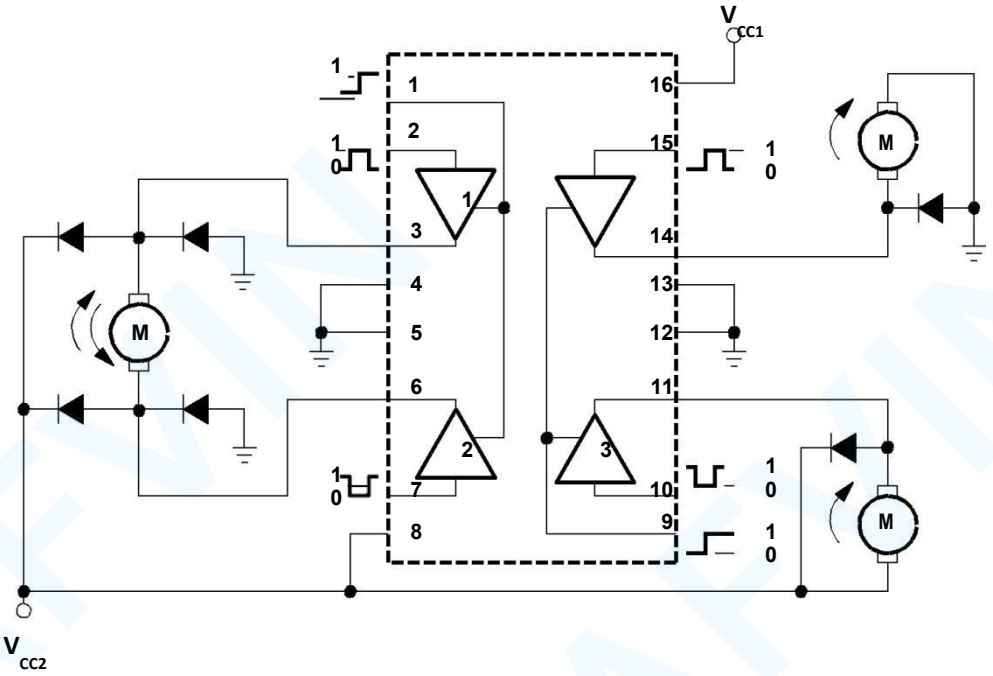
- Featuring Unitrode L293 and L293D Products Now From Texas Instruments
- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- Thermal Shutdown
- High-Noise-Immunity Inputs
- Functionally Similar to SGS L293 and SGS L293D
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive T ransient Suppression (L293D)

Description/ordering information

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

Block diagram



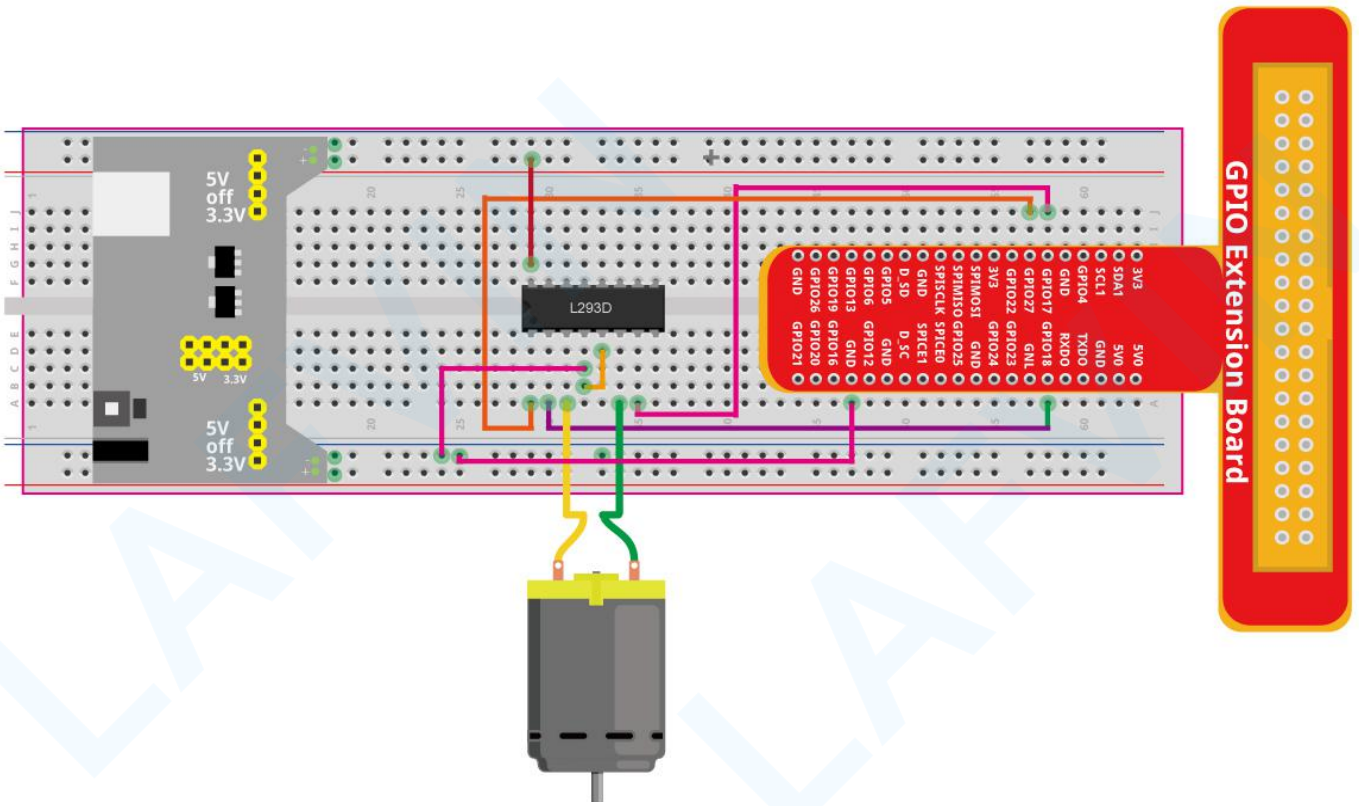
I got fed up with indecipherable pinout diagrams within datasheets, so have designed my own that I think gives more pertinent information.

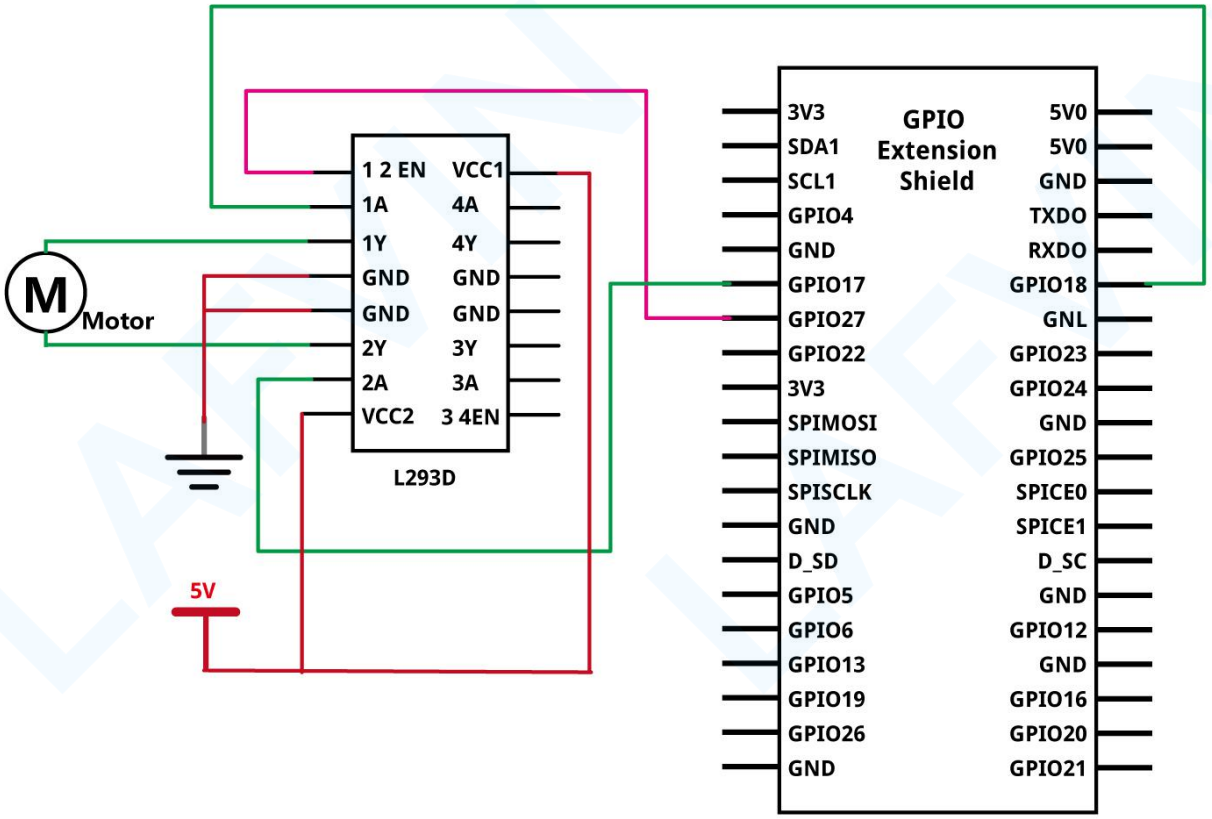
There are 3 wires connected to the Arduino, 2 wires connected to the motor, and 1 wire connected to a battery.

To use this pinout:

The left hand side deals with the first motor, the right hand side deals with a second motor. Yes, you can run it with only one motor connected.

Wiring diagram





Test instructions

This project is designed for learning how to use button to control LED. We first need to read the state of button, and then determine whether turn on LED according to the state of the button.

C_Code_6_DC_Motor

First, observe the project result, then analyze the code.

2. Use cd command to enter 6_DC_Motor directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/6_DC_Motor
```

2. Use the following command to compile the code "DC_Motor.c " and generate executable file "DC_Motor.c "

```
gcc DC_Motor.c –o DC_Motor -lwiringPi
```

3. Then run the generated file "DC_Motor".

```
sudo ./DC_Motor
```

The DC motor will turn for 5 seconds and then stop for 5 seconds, then for another 5 seconds, it will continue the process.

Code explanation

```
igitalWrite(MotorEnable, HIGH); // Enable the L239D

digitalWrite(MotorPin1, HIGH);
/* Set a high level for 2A(pin 7); since 1,2EN(pin 1) is in high level, 2Y will output high level*/

digitalWrite(MotorPin2, LOW);
// Set a low level for 1A, then 1Y will output low level, and the motor will rotate.

for(i=0;i<3;i++){
delay(1000);
} // this loop is to delay for 3*1000ms

digitalWrite(MotorEnable, LOW)
  // If 1,2EN (pin1) is in low level, L293D does not work. Motor stops rotating.

digitalWrite(MotorPin1, LOW)
digitalWrite(MotorPin2, HIGH)
// Reverse the current flow of the motor, then the motor will rotate reversely.
```

Python_Code_6_DC_Motor

First, observe the project result, then analyze the code.

1. Use cd command to enter 6_DC_Motor directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/6_DC_Motor
```

2. Use Python command to execute DC_Motor.py.

```
python DC_Motor.py
```

The DC motor will turn for 5 seconds and then stop for 5 seconds, then for another 5 seconds, it will continue the process.

Code explanation

```
GPIO.setup(MotorPin1, GPIO.OUT) # Set pin1 and pin2 for motor's rotation direction as output pin
GPIO.setup(MotorPin2, GPIO.OUT)
GPIO.setup(MotorEnable, GPIO.OUT)    # Set pins for motor's working condition as output pin
GPIO.output(MotorEnable, GPIO.LOW) # Set the motor low level for initial state
GPIO.output(MotorEnable, GPIO.HIGH) # Set the motor in high level
GPIO.output(MotorPin1, GPIO.HIGH) # Set pin1 in high level and pin2 in low level
GPIO.output(MotorPin2, GPIO.LOW) # Make the motor rotate clockwise
time.sleep(5)    # rotate for 5 seconds
GPIO.output(MotorEnable, GPIO.LOW)    # Stop the motor
time.sleep(5)    #wait for 5 seconds
```

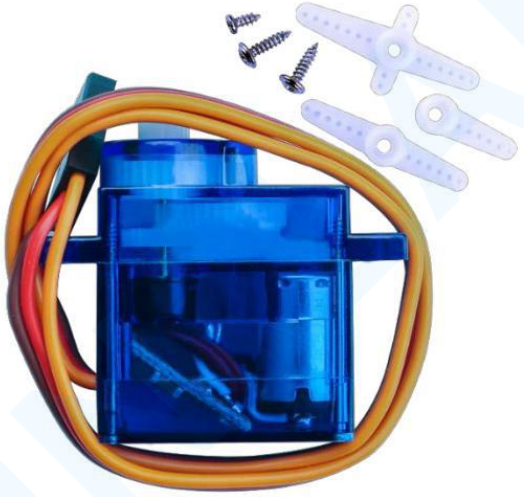
Lesson 7 Servo

About this lesson:

In this lesson, we will learn how to control servo by raspberry pi.

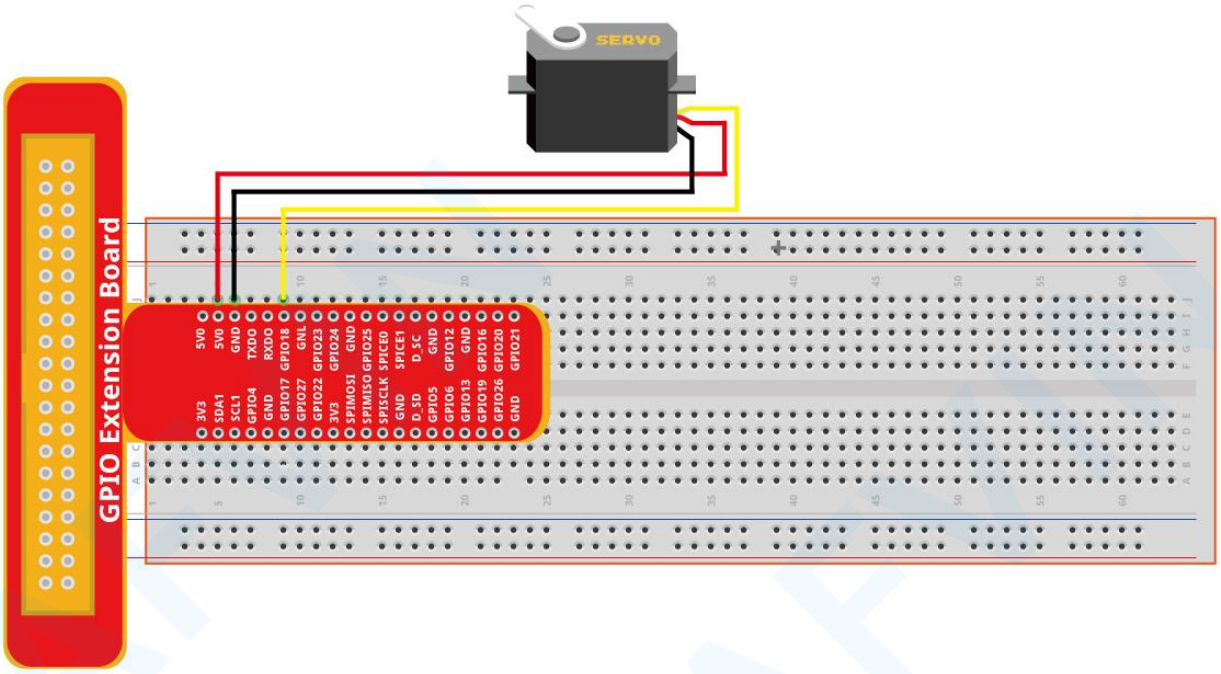
Introduction

Servo is a type of geared motor that can only rotate 180 degrees. It is controlled by sending electrical pulses from your raspberry pi. These pulses tell the servo what position it should move to. The Servo has three wires, of which the brown one is the ground wire and should be connected to the GND port of raspberry pi, the red one is the power wire and should be connected to the 5v port, and the orange one is the signal wire and should be connected to the GPIOx port.

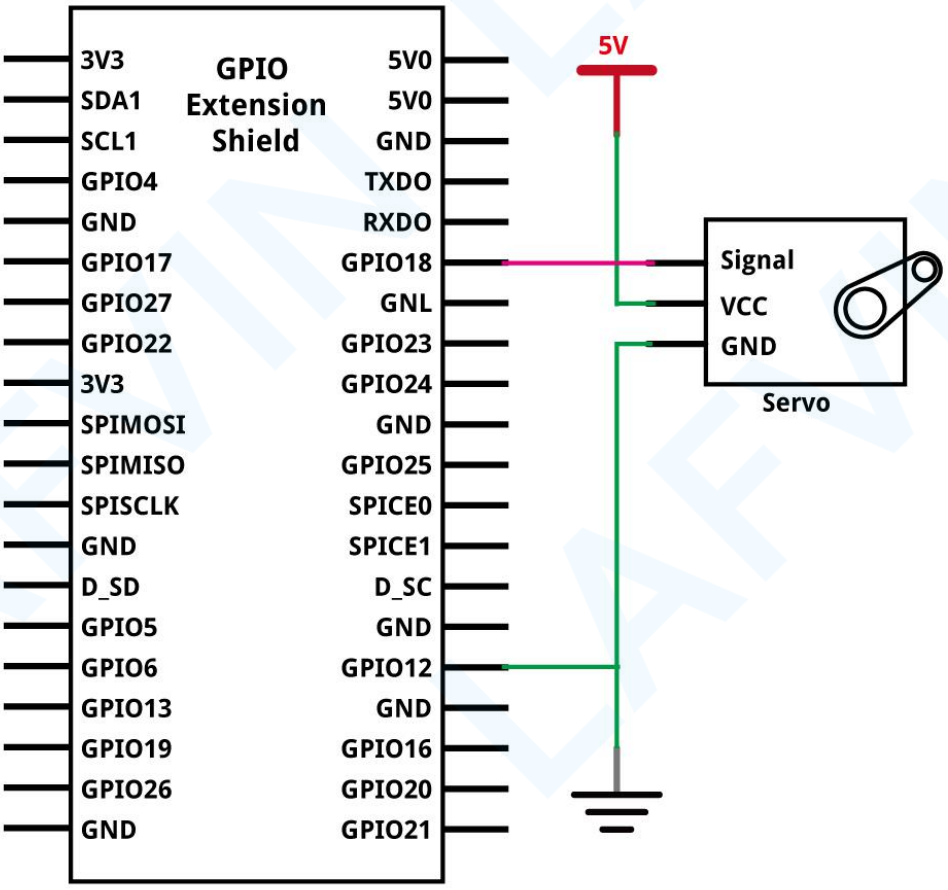


We use 50Hz PWM signal with a duty cycle in a certain range to drive the servo. The lasting time 0.5ms-2.5ms of PWM single cycle high level corresponds to the servo angle 0 degrees - 180 degree linearly. Part of the corresponding values are as follows:

High level time	Servo angle
0.5ms	0 degree
1ms	45 degree
1.5ms	90 degree
2ms	135 degree
2.5ms	180 degree



Wiring diagram



Test instructions

In this experiment,raspberry pi control servo rotate from minimum angle to maximum angle,and then make servo rotate from maximum angle to minimum angle

C_Code_7_Servo

First, observe the project result, then analyze the code.

3. Use cd command to enter 7_Servo directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/7_Servo
```

2. Use the following command to compile the code “Servo.c ” and generate executable file “Servo.c ”

```
gcc Servo.c –o Servo -lwiringPi
```

3. Then run the generated file “Servo”.

```
sudo ./Servo
```

After the program is executed, the servo will rotate from 0 degrees to 180 degrees, and then from 180 degrees to 0 degrees, circularly.

Code explanation

```
void servoInit(int pin)
{
softPwmCreate(pin, 0, 200);
}
 /*initialization function for servo PWM pin,50 Hz pulse, namely cycle for 20ms, is required to control Servo. In function softPwmCreate (int pin, int initialValue, int pwmRange), the unit of third parameter pwmRange is 100US, namely 0.1ms. In order to get the PWM with cycle of 20ms, the pwmRange shoulde be set to 200. So in subfunction of servoInit (), we create a PWM pin with pwmRange 200.*/

void servoWrite(int pin, int angle)
{
if(angle > 180)
angle = 180;
if(angle < 0)
angle = 0;
softPwmWrite(pin,map(angle,0,180,SERVO_MIN_MS,SERVO_MAX_MS));
}
/*Specif a certain rotation angle (0-180) for the servo.In subfunction servoWrite (), input directly angle (0-180 degrees), and map the angle to the pulse width and
then output it.*/
```

Python_Code_7_Servo

First, observe the project result, then analyze the code.

1. Use cd command to enter 7_Servo directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/7_Servo
```

2. Use Python command to execute Servo.py.

```
python Servo.py
```

After the program is executed, the servo will rotate from 0 degrees to 180 degrees, and then from 180 degrees to 0 degrees, circularly.

Code explanation

```
p =GPIO.PWM( (servoPin, , 50) ) # Set Frequency to 50Hz


def loop ():
      while    True: :
         for dc in range( (0, , 181, , 1 ): #make servo rotate from 0°to 180°
               servoWrite( (dc) ) # Write to servo
               time. .sleep( (0.001) )
         time. .sleep( (0.5) )
         for dc    in range( (180, , - -1, , - -1 ): #make servo rotate from 180°to 0°
               servoWrite( (dc) )
               time. .sleep( (0.001) )
         time. .sleep( (0.5) )
         " " "in the "while" cycle of main function, use two "for" cycle to make servo
rotate from 0 degrees to 180 degrees, and then from 180 degrees to 0 degrees." " "
```

Lesson 8 LCD1602

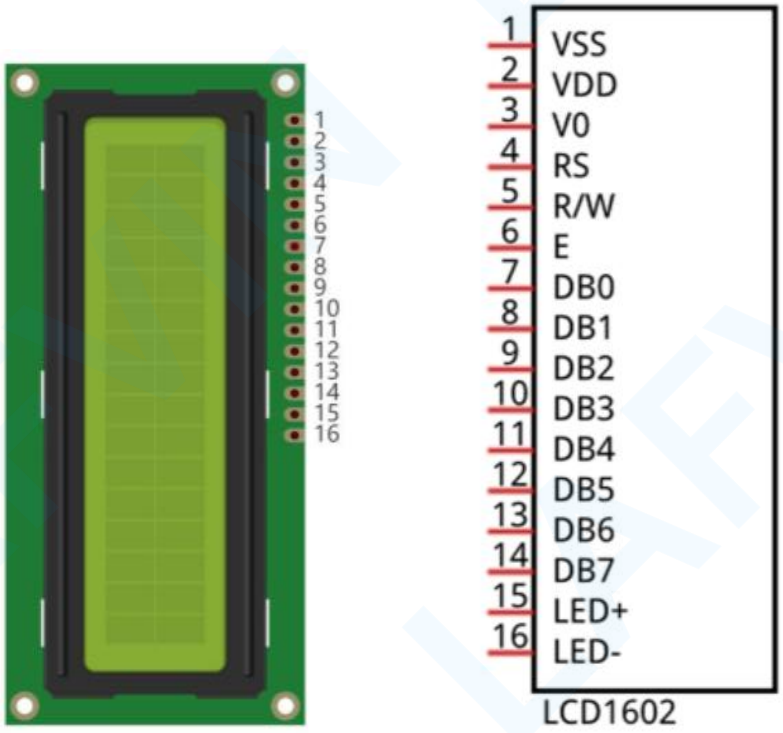
About this lesson:

In this lesson, you will learn how to wire up and use an alphanumeric LCD display. The display has an LED backlight and can display two rows with up to 16 characters on each row. You can see the rectangles for each character on the display and the pixels that make up each character. The display is just white on blue and is intended for showing text.

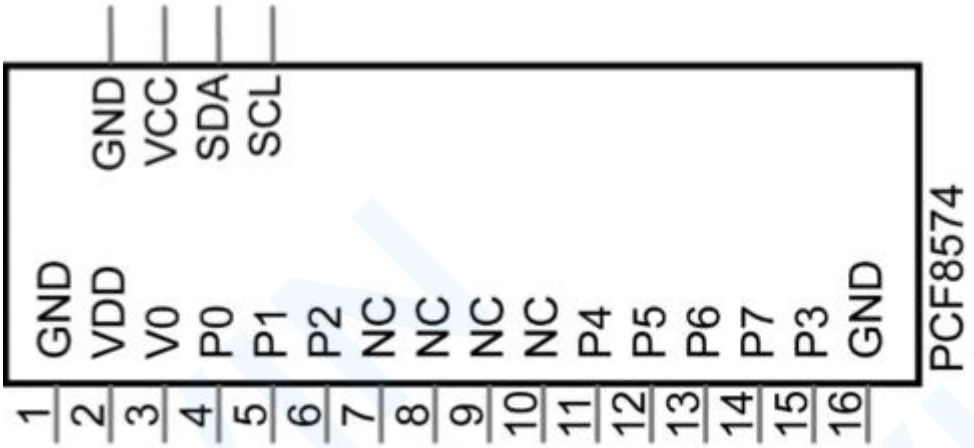
Introduction

LCD1602

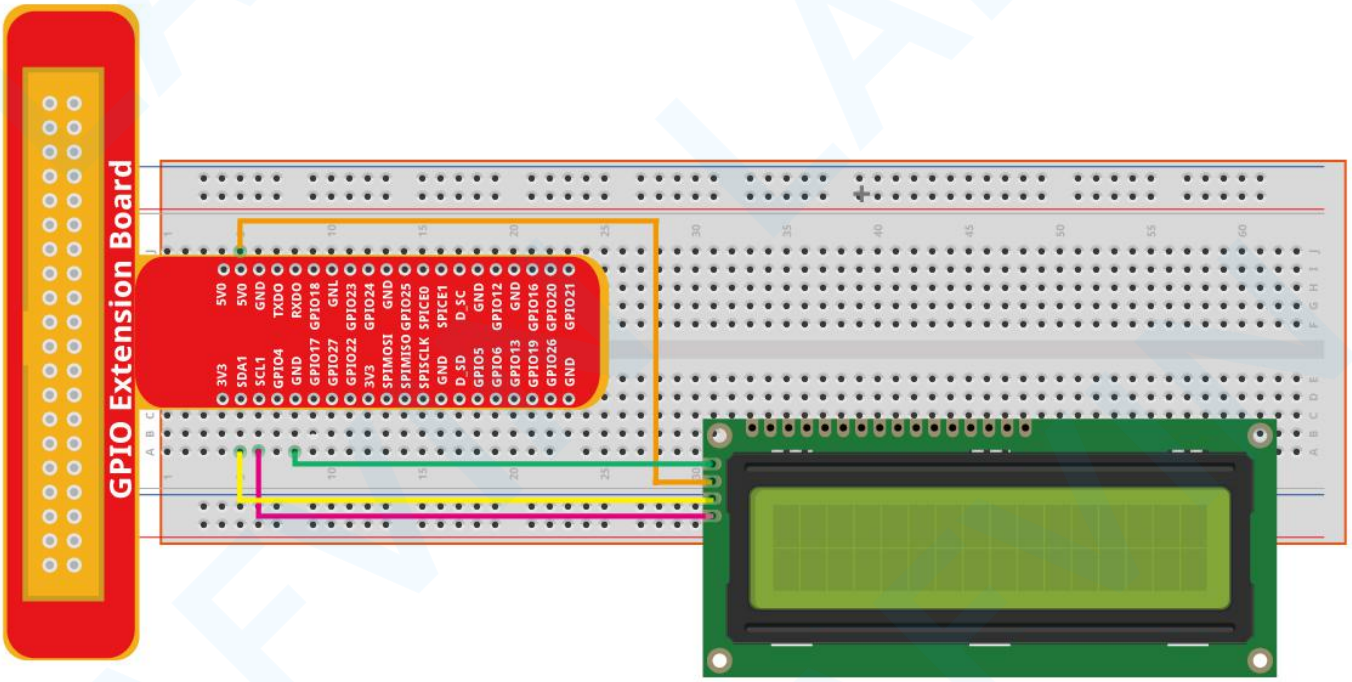
Introduction to the pins of LCD1602:
VSS: A pin that connects to ground
VDD: A pin that connects to a +5V power supply
VO: A pin that adjust the contrast of LCD1602
RS: A register select pin that controls where in the LCD's memory you are writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.
R/W: A Read/Write pin that selects reading mode or writing mode
E: An enabling pin that, when supplied with low-level energy, causes the LDC module to execute relevant instructions.
D0-D7： Pins that read and write data
A and K: Pins that control the LED backlight



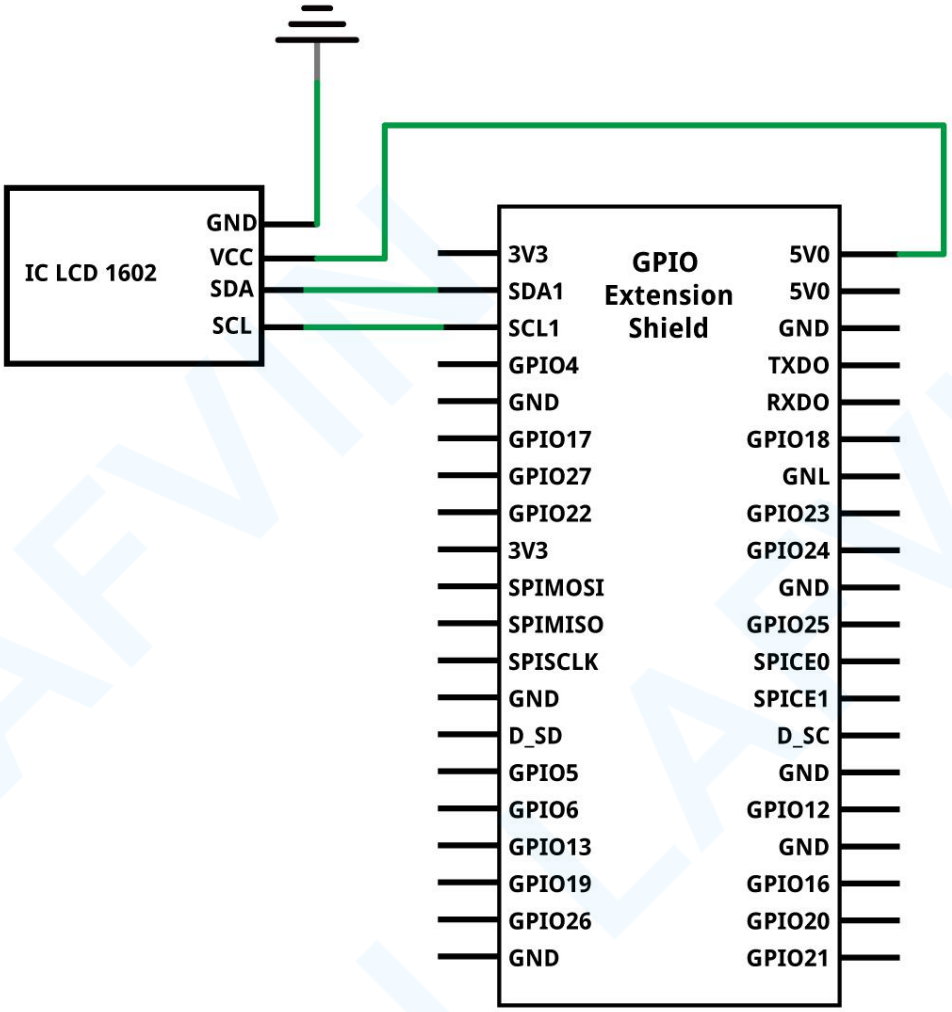
The serial-to-parallel chip used in this module is PCF8574(PCF8574A), and its default I2C address is 0x27(0x3F),and you can view all the RPI bus on your I2C device address through command "i2cdetect –y 1" to. (refer to the "configuration I2C" section below) below is the PCF8574 pin schematic diagram and the block pin diagram:



So, we can use just 4 pins to control LCD1602 with 16 pins easily through I2C interface.In this project, we will use I2CLCD1602 to display some static characters and dynamic variables.



Wiring diagram

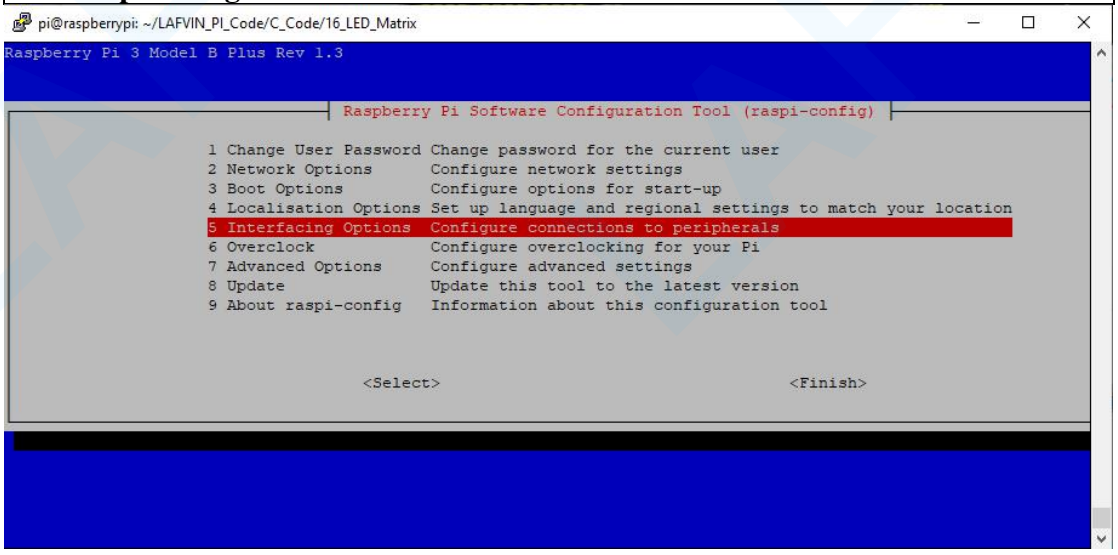


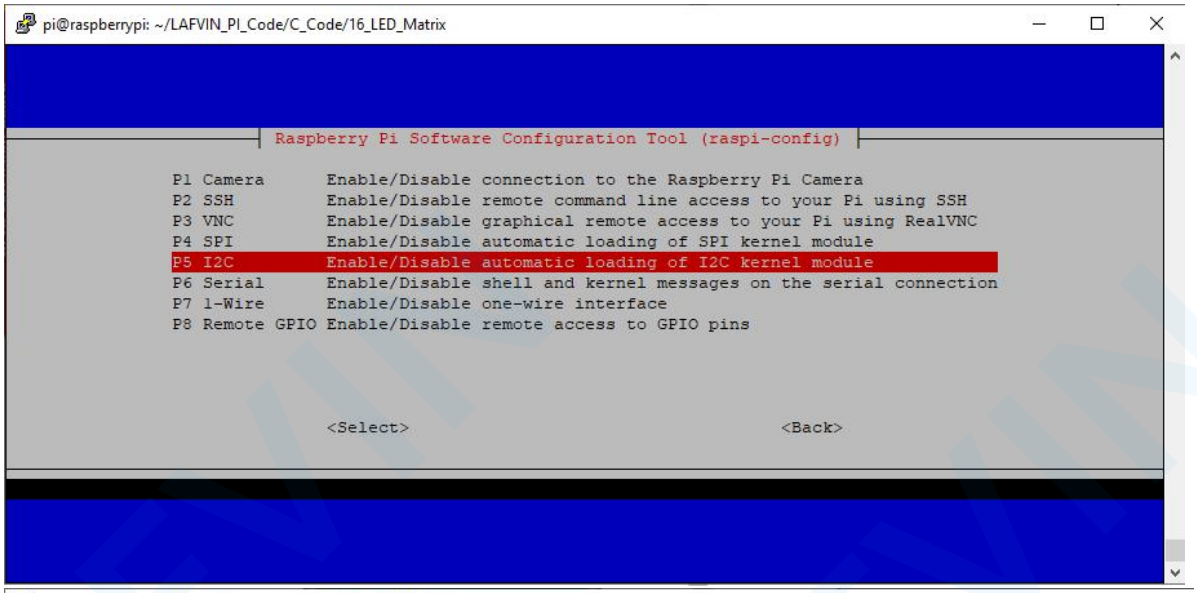
Test instructions

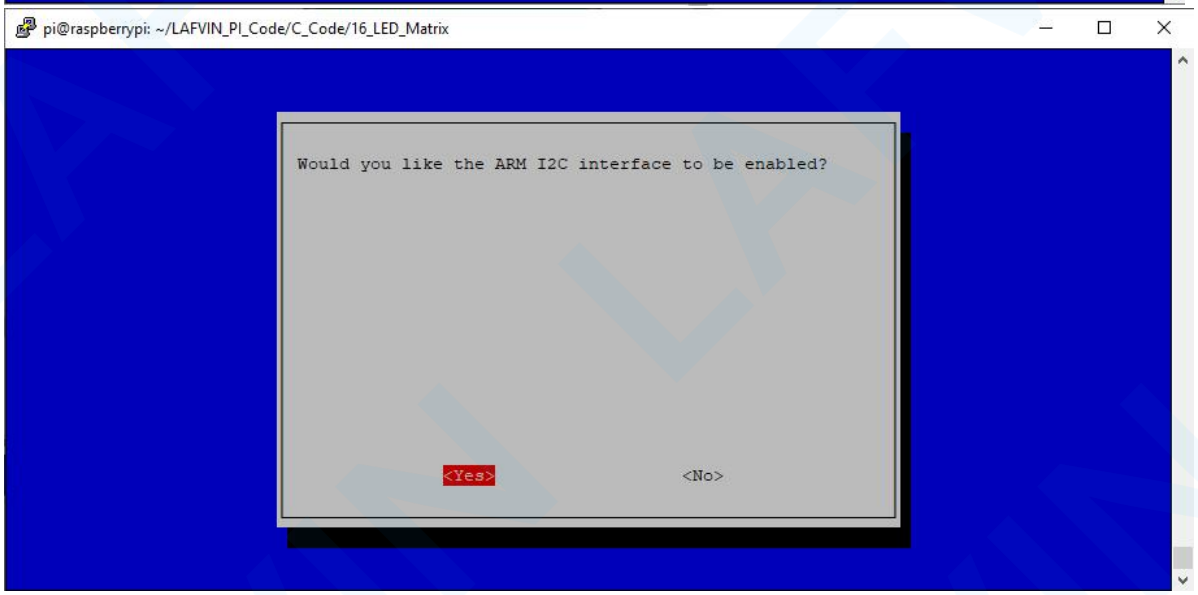
This code will get the CPU temperature and system time of raspberry pi, display them on LCD1602.

Before uploading the code, you need to open the Raspberry Pi I2C interface with the following instructions.

sudo raspi-config







C_Code_8_LCD1602

First, observe the project result, then analyze the code.

4. Use cd command to enter 8_LCD1602 directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/8_LCD1602
```

2. Use the following command to compile the code “LCD1602.c ” and generate executable file “LCD1602.c ”

```
gcc LCD1602.c -o LCD1602 -lwiringPi -lwiringPiDev
```

3. Then run the generated file “LCD1602”.

```
sudo ./LCD1602
```

After the program is executed, LCD1602 screen will display current CPU temperature and system time. If there is no display or the display is not clear, adjust potentiometer of PCF8574 module to adjust the contrast of LCD1602 until the screen can display clearly.

Code explanation

```
int lcdInit (int rows, int cols, int bits, int rs, int strb,int d0, int d1, int d2, int d3, int d4, int d5, int d6, int d7) ;
/*This is the main initialization function and must be called before you use any other LCD functions. Rows and cols are the rows and columns on the display (e.g. 2, 16 or
```

```
4,20). Bits is the number of bits wide on the interface (4 or 8). The rs and strb
represent the pin numbers of the displays RS pin and Strobe (E) pin. The parameters
d0 through d7 are the pin numbers of the 8 data pins connected from the Pi to the
display. Only the first 4 are used if you are running the display in 4-bit mode.
The return value is the 'handle' to be used for all subsequent calls to the lcd library
when dealing with that LCD, or -1 to indicate a fault. (Usually incorrect parameters)*/

lcdPosition (int handle, int x, int y);
//Set the position of the cursor for subsequent text entry.

lcdPutchar (int handle, uint8_t data)
lcdPuts (int handle, char *string)
lcdPrintf (int handle, char *message, …)
/*These output a single ASCII character, a string or a formatted string using the usual
printf formatting commands.*/
```

Python_Code_8_LCD1602

First, observe the project result, then analyze the code.

1. Use cd command to enter 8_LCD1602 directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/8_LCD1602
```

2. Use Python command to execute LCD1602.py.

```
python LCD1602.py
```

After the program is executed, LCD1602 screen will display current CPU temperature and system time. If there is no display or the display is not clear, adjust potentiometer of PCF8574 module to adjust the contrast of LCD1602 until the screen can display clearly.

Code explanation

```
from PCF8574    import PCF8574_GPIO
" " "This module provides two classes PCF8574_I2C and PCF8574_GPIO.
Class PCF8574_I2C:    provides reading and writing method for PCF8574.
Class PCF8574_GPIO: provides a standardized set of GPIO functions. More
information can be viewed through opening PCF8574.py." " "

from Adafruit_LCD1602    import Adafruit_CharLCD
" " "This module provides the basic operation method of LCD1602, including class
Adafruit_CharLCD. Some
member functions are described as follows:
def begin(self, cols, lines): set the number of lines and columns of the screen.
def clear(self): clear the screen
def setCursor(self, col, row): set the cursor position
def message(self, text): display contents
More information can be viewed through opening Adafruit_CharLCD.py." " "
```

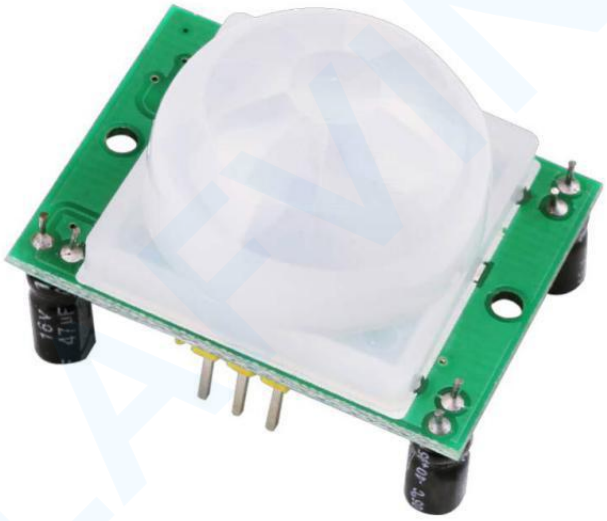
Lesson 9 PIR motion sensor

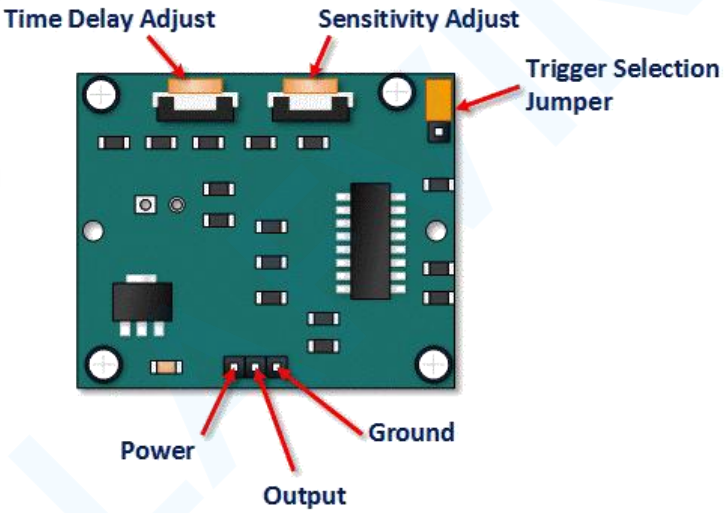
About this lesson:

In this lesson you will learn how to use a PIR movement detector with an raspberry pi. The raspberry pi is the heart of this project. It 'listens' to the PIR sensor and when motion is detected, instructs the LED to light on or shut off.

Introduction

PIR sensors are more complicated than many of the other sensors explained in this tutorial (like photocells, FSRs and tilt switches) because there are multiple variables that affect the sensors input and output. .
The PIR sensor itself has two slots. Each slot is made of a special material that is sensitive to IR. The lens used here is not really doing much and so we see that the two slots can 'see' out past some distance (basically the sensitivity of the sensor). When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors. When a warm body like a human or an animal passes by, it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves. When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.





Pin or Control	Function
Time Delay Adjust	Sets how long the output remains high after detecting motion.... Anywhere from 5 seconds to 5 minutes.
Sensitivity Adjust	Sets the detection range.... from 3 meters to 7 meters
Trigger Selection Jumper	Set for single or repeatable triggers.
Ground pin	Ground input
Output Pin	Low when no motion is detected.. High when motion is detected. High is 3.3V
Power Pin	5 to 20 VDC Supply input

HC SR501 PIR Functional Description

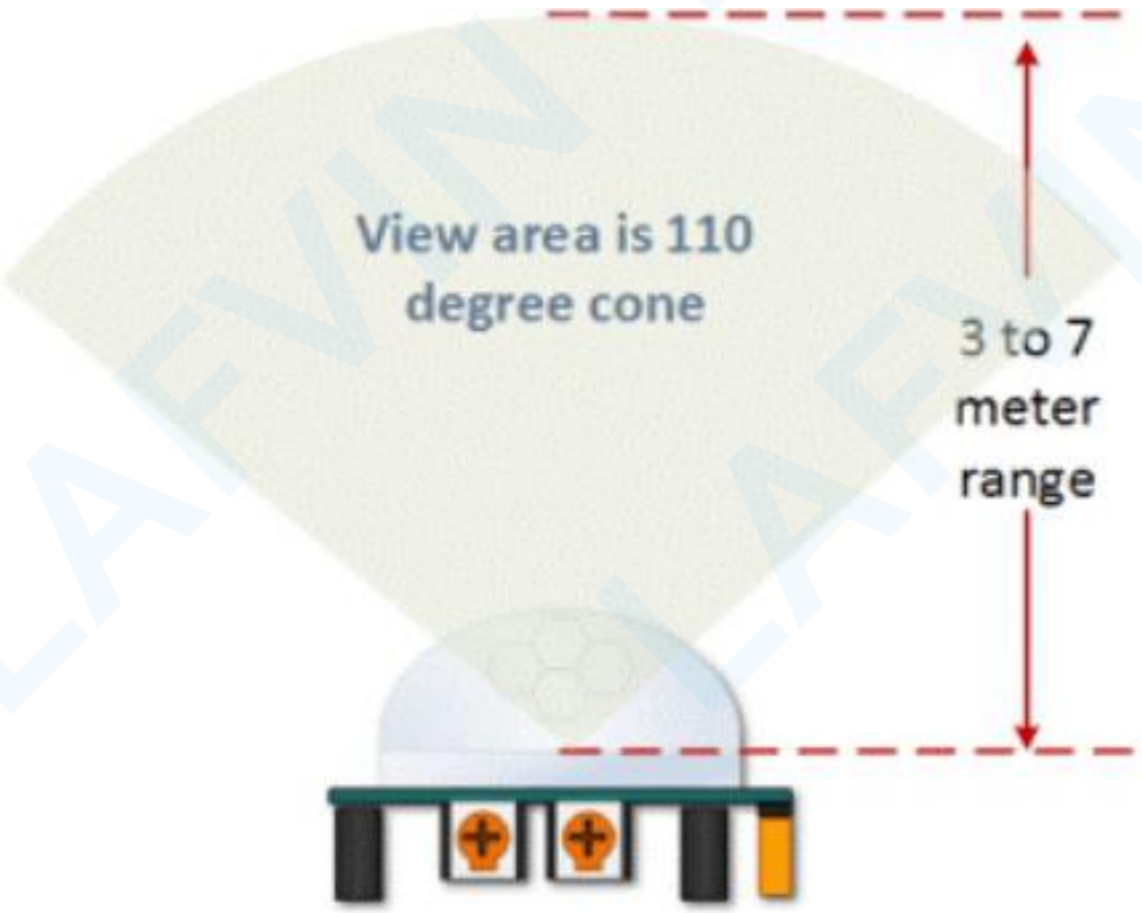
The SR501 will detect infrared changes and if interpreted as motion, will set its output low. What is or is not interpreted as motion is largely dependent on user settings and adjustments.

Device Initialization

The device requires nearly a minute to initialize. During this period, it can and often will output false detection signals. Circuit or controller logic needs to take this initialization period into consideration.

Device Area of Detection

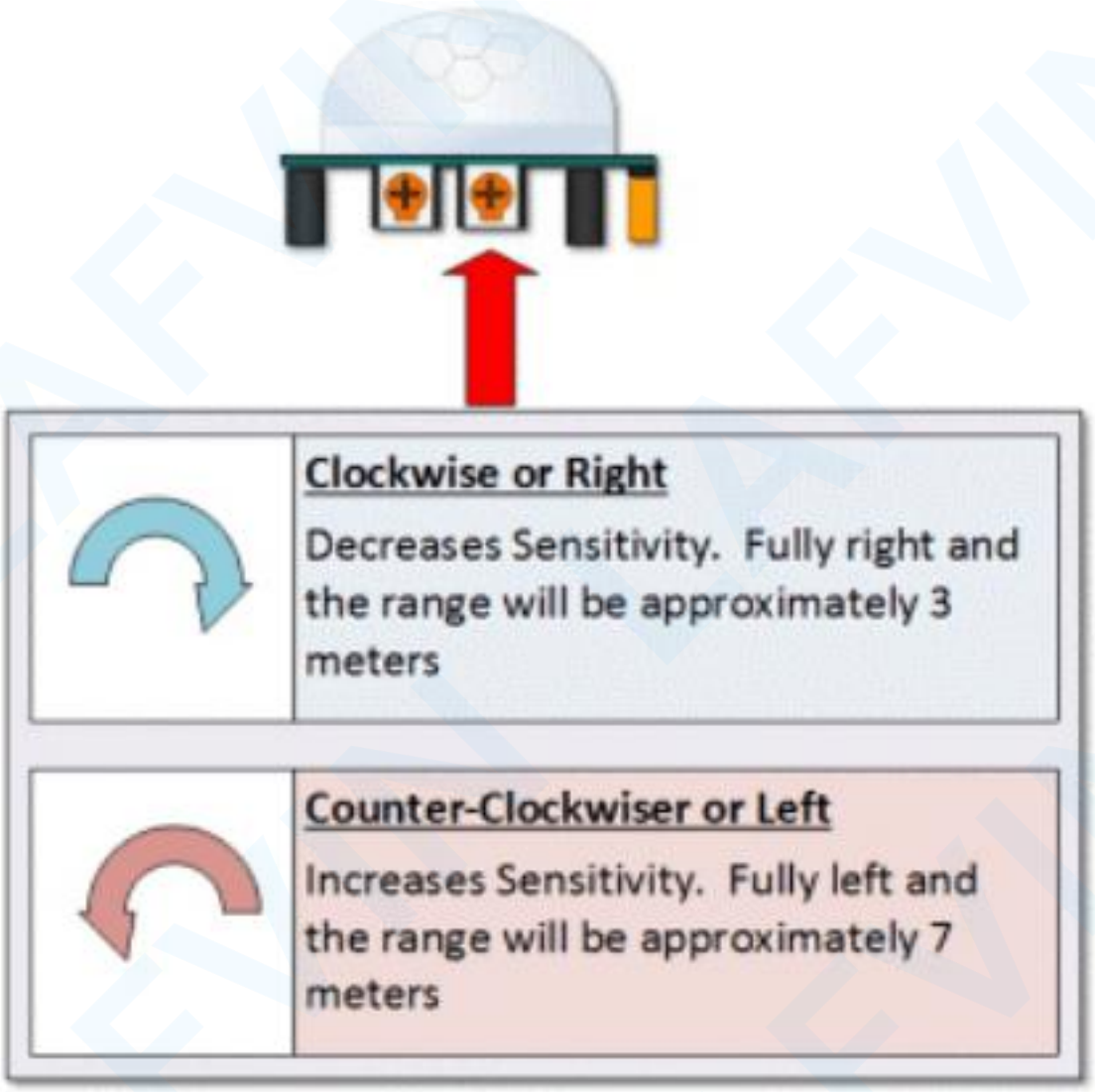
The device will detect motion inside a 110 degree cone with a range of 3 to 7 meters



HC SR501 View Area

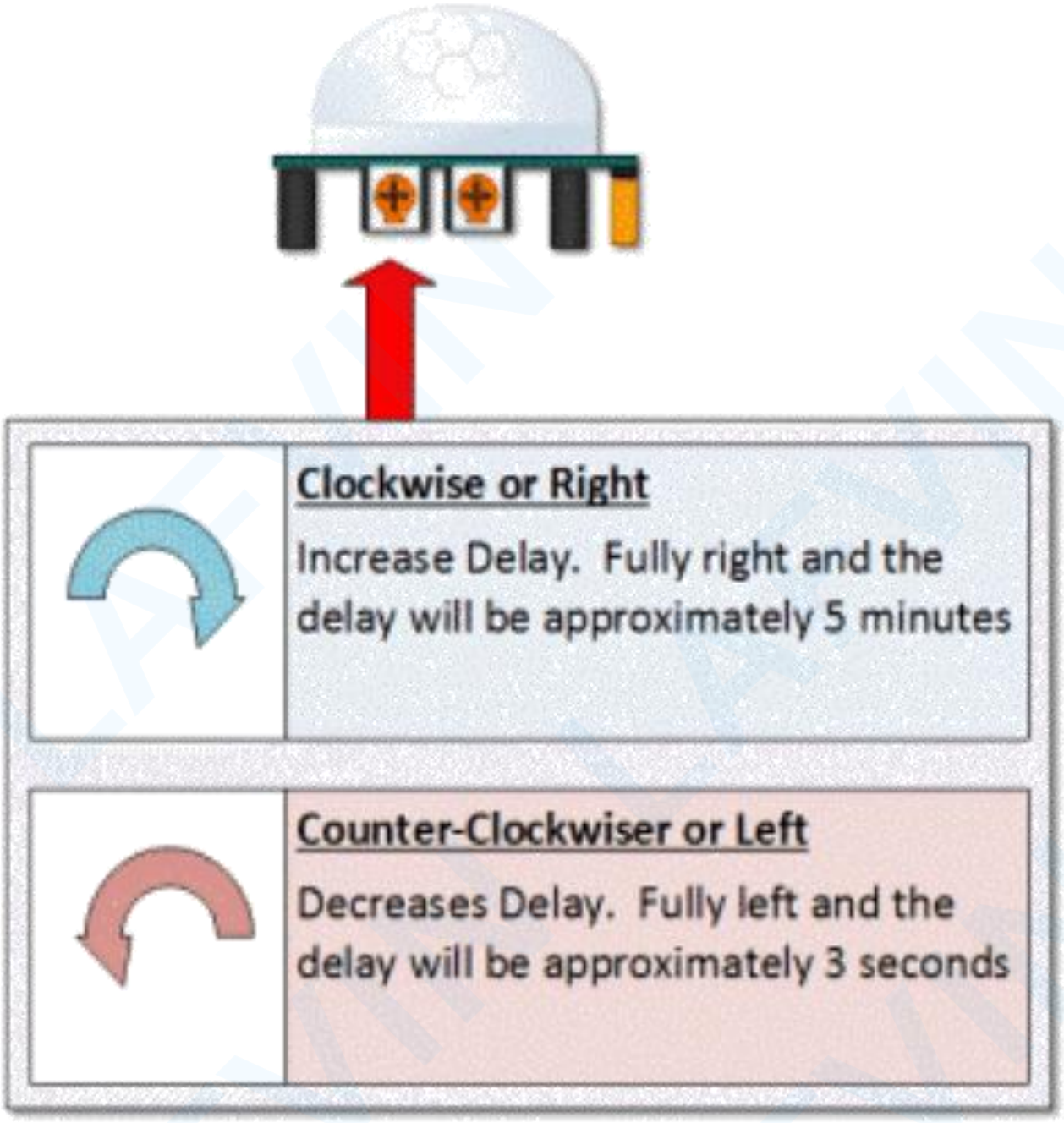
PIR Range (Sensitivity) Adjustment

As mentioned, the adjustable range is from approximately 3 to 7 meters. The illustration below shows this adjustment.



HC SR501 Sensitivity Adjust Time Delay Adjustment

The time delay adjustment determines how long the output of the PIR sensor module will remain high after detection motion. The range is from about 3 seconds to five minutes.

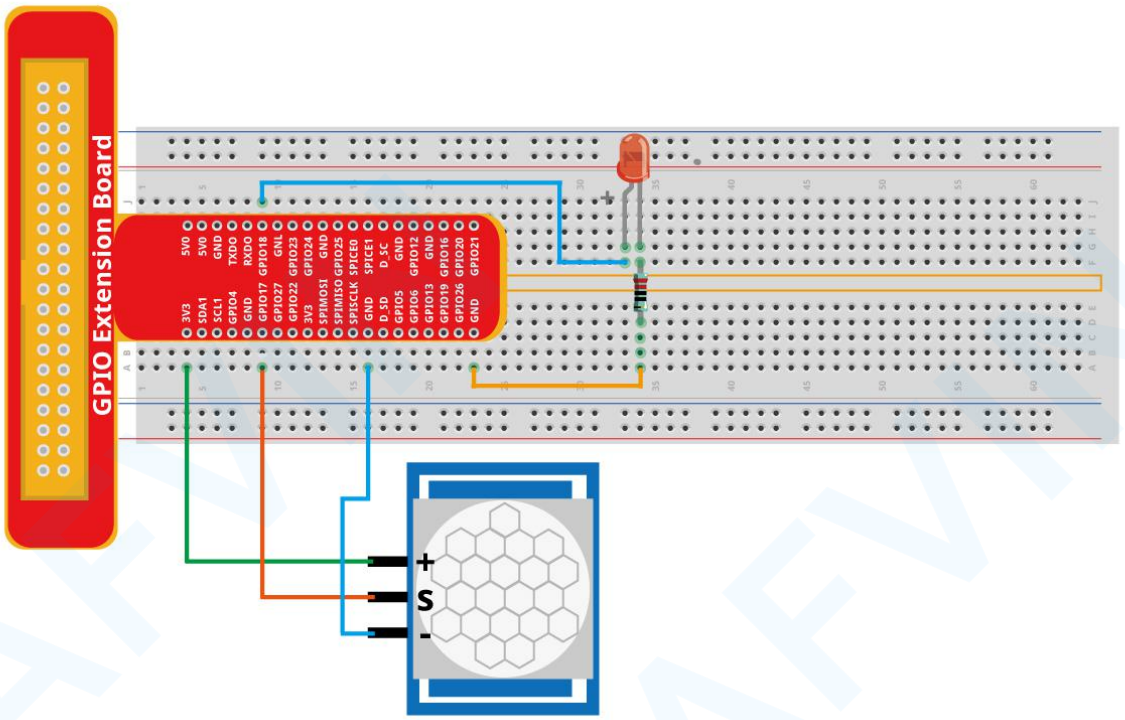


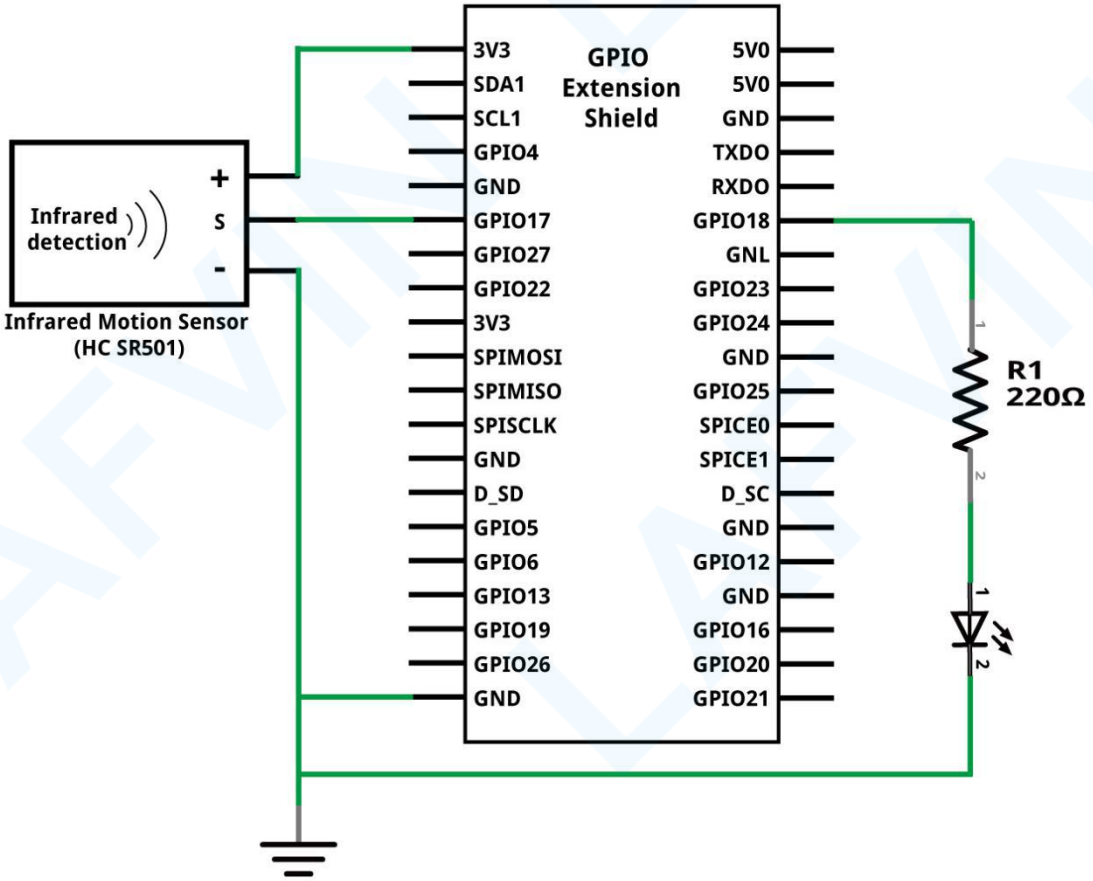
HC SR501 Time Delay Adjustment

3 Seconds Off After Time Delay Completes – IMPORTANT

The output of this device will go LOW (or Off) for approximately 3 seconds AFTER the time delay completes. In other words, ALL motion detection is blocked during this three second period.

Wiring diagram


GPIO Extension Board
+
S
-


Infrared detection
+
S
-
Infrared Motion Sensor
(HC SR501)
GPIO
Extension
Shield
3V3
SDA1
SCL1
GPIO4
GND
GPIO17
GPIO27
GPIO22
3V3
SPIMOSI
SPIMISO
SPISCLK
GND
D_SD
GPIO5
GPIO6
GPIO13
GPIO19
GPIO26
GND
5V0
5V0
GND
TXDO
RXDO
GPIO18
GNL
GPIO23
GPIO24
GND
GPIO25
SPICE0
SPICE1
D_SC
GND
GPIO12
GND
GPIO16
GPIO20
GPIO21
R1
220Ω

Test instructions

In this project, we use infrared motion sensor to control LED, and take infrared motion sensor as a switch, so the code very similar to front project "Button&LED" in logic. The difference is that, when infrared motion sensor detects change, it will output high level; when button is pressed, it will output low level. When the sensor output high level, the LED will be turned on, or it will be turned off.

C_Code_9_PIR_motion_sensor

First, observe the project result, then analyze the code.

5. Use cd command to enter 9_PIR motion sensor directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/9_PIR_motion_sensor
```

2. Use the following command to compile the code “PIR motion sensor.c ” and generate executable file “PIR motion sensor.c ”

```
gcc PIR_motion_sensor.c -o PIR motion sensor -lwiringPi
```

3. Then run the generated file “PIR motion sensor”.

```
sudo ./PIR_motion_sensor
```

After the program is executed, try to leave away from or get closed to the Motion Sensor Infrared and observe whether the LED will be turned on or off. The terminal window will print out the state of LED constantly. As is shown below:

Python_Code_9_PIR_motion_sensor

First, observe the project result, then analyze the code.

1. Use cd command to enter 9_PIR_motion_sensor directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/9_PIR_motion_sensor
```

2. Use Python command to executePIR_motion_sensor.py.

```
python PIR_motion_sensor.py
```

After the program is executed, try to leave away from or get closed to the Motion Sensor Infrared and observe whether the LED will be turned on or off. The terminal window will print out the state of LED constantly. As is shown below:

Lesson 10 Stepper Motor

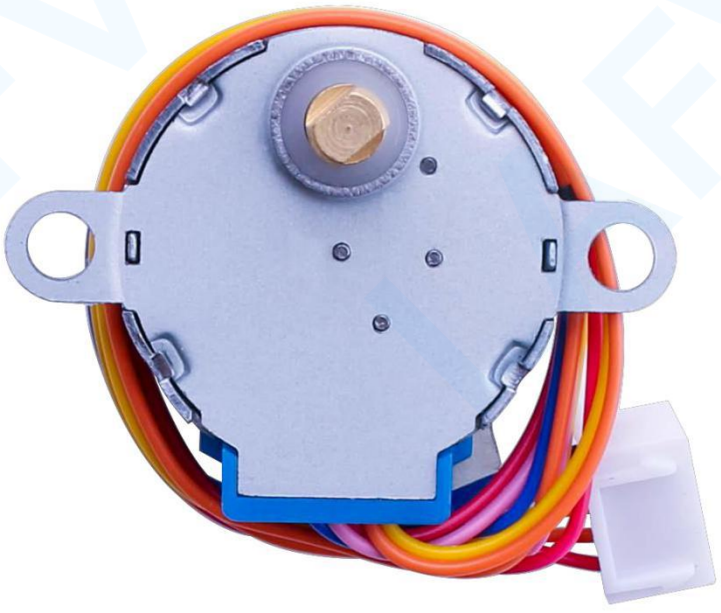
About this lesson:

In this lesson, you will learn a fun and easy way to drive a stepper motor.The stepper we are using comes with its own driver board making it easy to connect to our raspberry pi.

Introduction

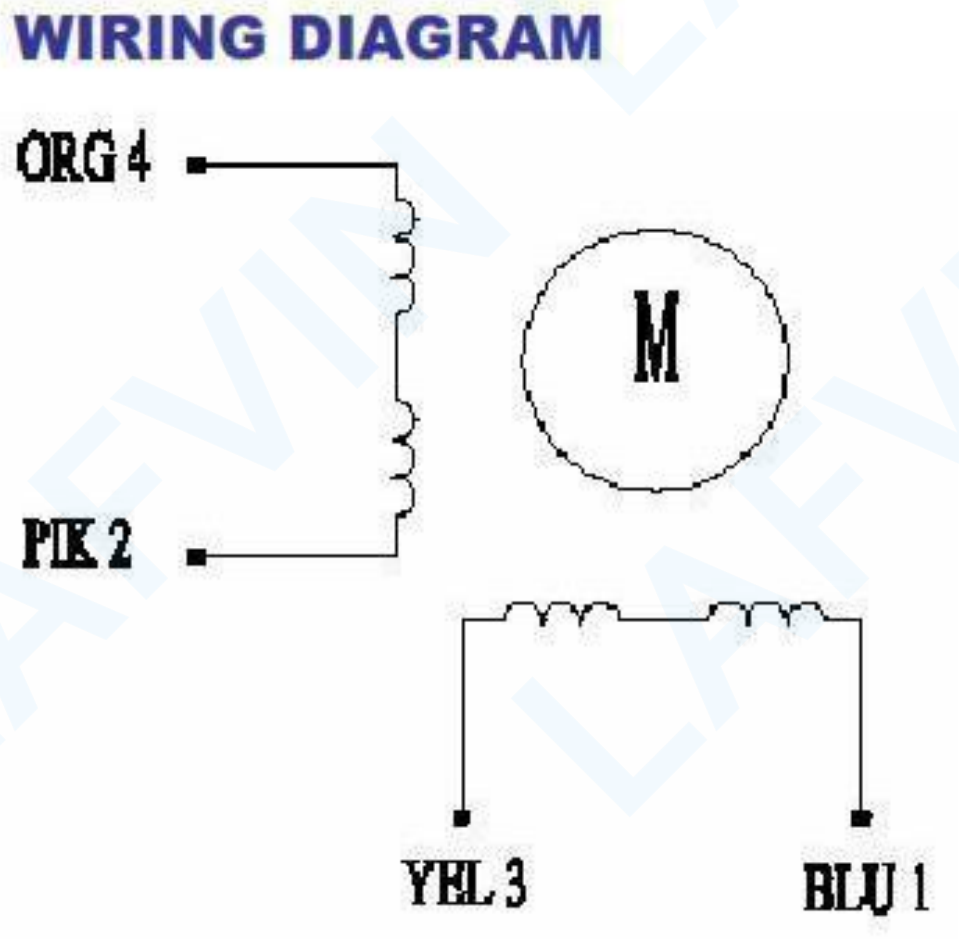
Stepper Motor

A stepper motor is an electromechanical device which converts electrical pulses into discrete mechanical movements. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical command pulses are applied to it in the proper sequence. The motors rotation has several direct relationships to these applied input pulses. The sequence of the applied pulses is directly related to the direction of motor shafts rotation. The speed of the motor shafts rotation is directly related to the frequency of the input pulses and the length of rotation is directly related to the number of input pulses applied. One of the most significant advantages of a stepper motor is its ability to be accurately controlled in an open loop system. Open loop control means no feedback information about position is needed. This type of control eliminates the need for expensive sensing and feedback devices such as optical encoders. Your position is known simply by keeping track of the input step pulses.



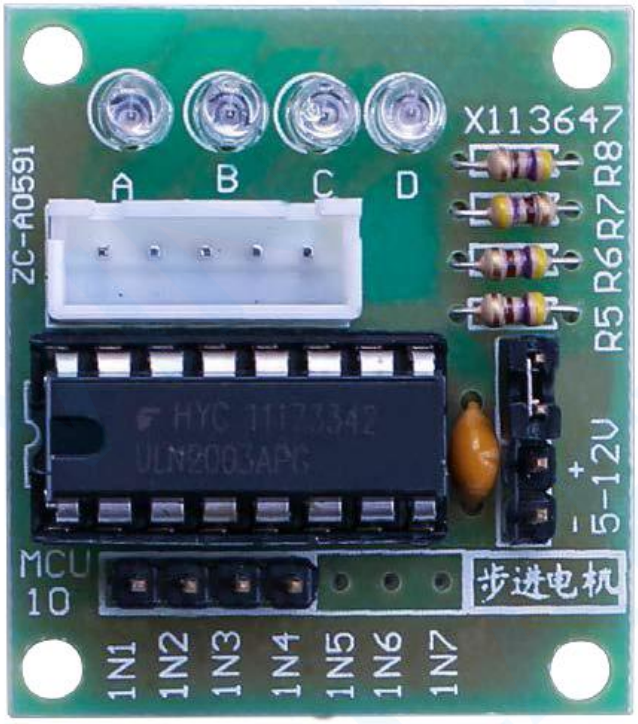
Stepper motor 28BYJ-48 Parameters

- Model: 28BYJ-48
- Rated voltage: 5VDC
- Number of Phase: 4
- Speed Variation Ratio: 1/64
- Stride Angle: 5.625° /64
- Frequency: 100Hz
- DC resistance: 50Ω±7%(25℃)
- Idle In-traction Frequency: > 600Hz
- Idle Out-traction Frequency: > 1000Hz
- In-traction Torque >34.3mN.m(120Hz)
- Self-positioning Torque >34.3mN.m
- Friction torque: 600-1200 gf.cm
- Pull in torque: 300 gf.cm
- Insulated resistance >10MΩ(500V)
- Insulated electricity power：600VAC/1mA/1s
- Insulation grade：A
- Rise in Temperature <40K(120Hz)
- Noise <35dB(120Hz,No load,10cm)



The bipolar stepper motor usually has four wires coming out of it. Unlike unipolar steppers, bipolar steppers have no common center connection. They have two independent sets of coils instead. You can distinguish them from unipolar steppers by measuring the resistance between the wires. You should find two pairs of wires with equal resistance. If you've got the leads of your meter connected to two wires that are not connected (i.e. not attached to the same coil), you should see infinite resistance (or no continuity).

ULN2003 Driver Boar



Product Description

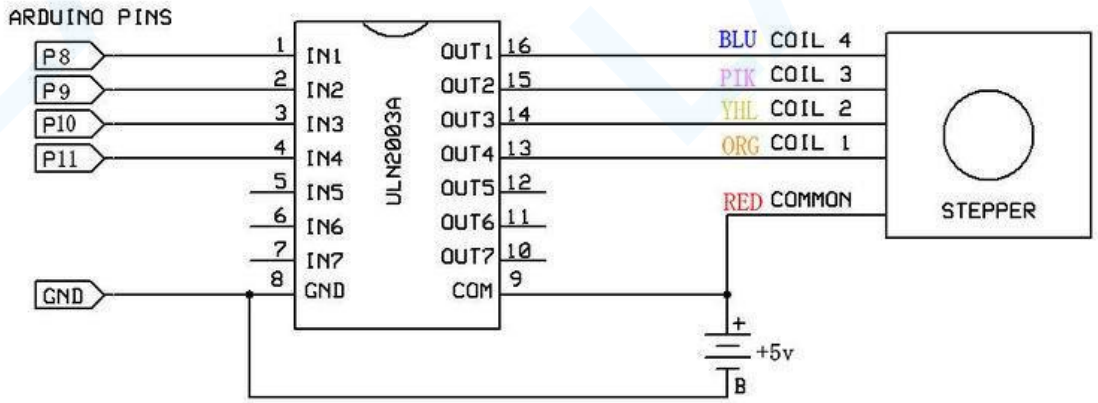
- Size: 42mmx30mm
- Use ULN2003 driver chip, 500mA
- A. B. C. D LED indicating the four phase stepper motor working condition. White jack is the four phase stepper motor standard jack.
- Power pins are separated
- We kept the rest pins of the ULN2003 chip for your further prototyping.

The simplest way of interfacing a unipolar stepper to Arduino is to use a breakout for ULN2003A transistor array chip. The ULN2003A contains seven Darlington transistor drivers and is somewhat like having seven TIP120 transistors all in one package. The ULN2003A can pass up to 500 mA per channel and has an internal voltage drop of about 1V when on. It also contains internal clamp diodes to dissipate voltage spikes when driving inductive loads. To control the stepper, apply voltage to each of the coils in a specific sequence.

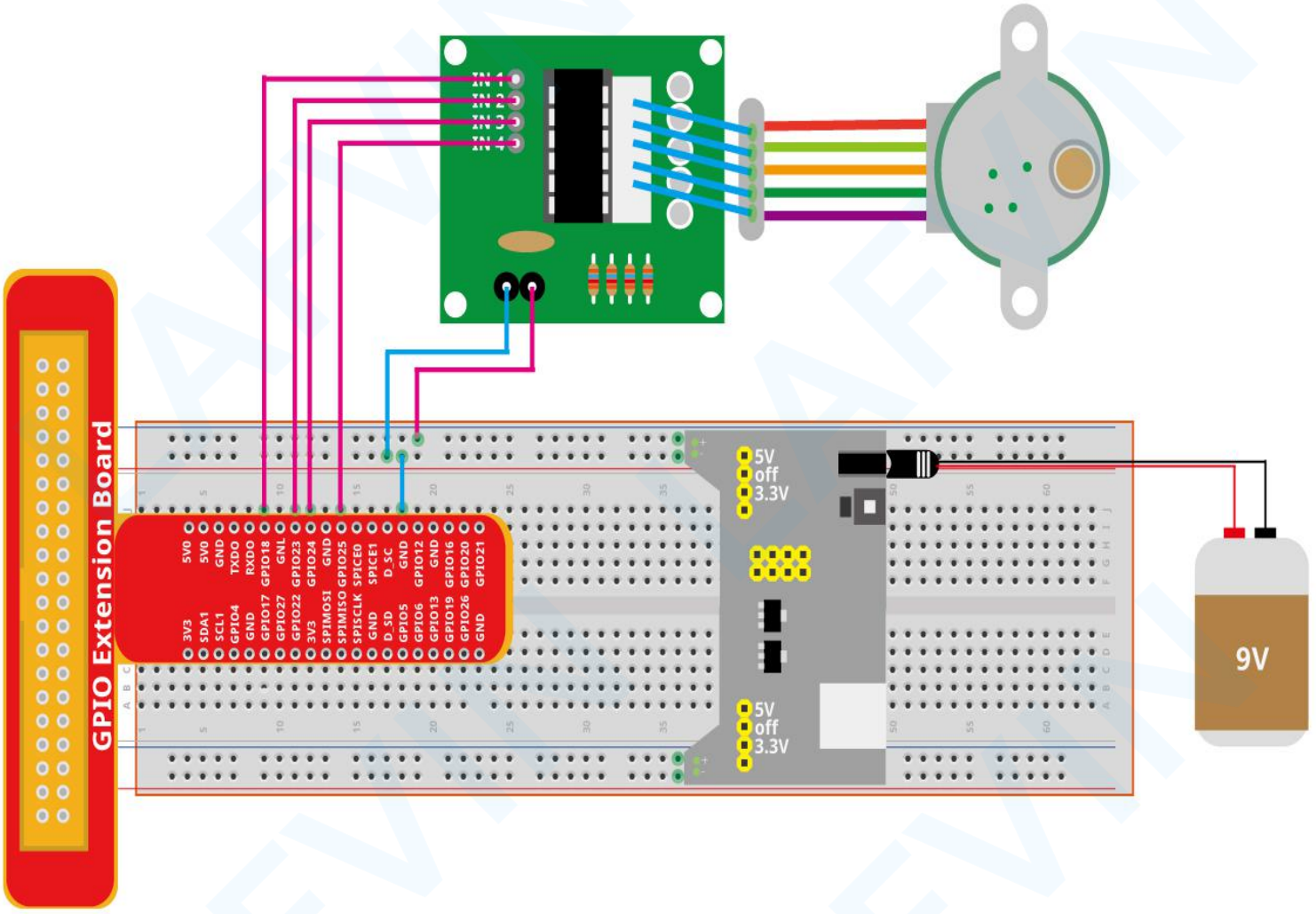
The sequence would go like this:

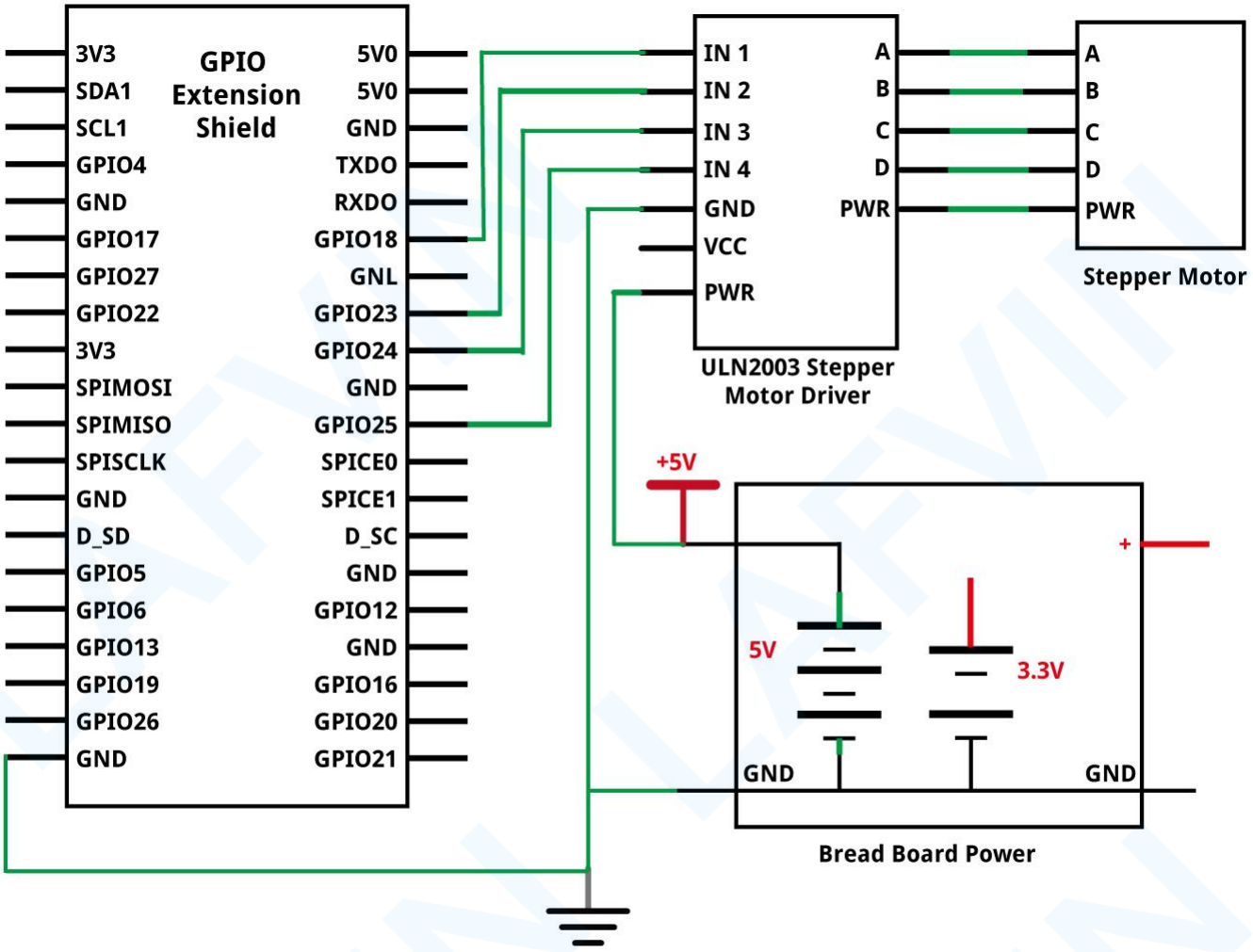
Lead Wire Color	---> CW Direction (1-2 Phase)							
	1	2	3	4	5	6	7	8
4 ORG	-	-						-
3 YEL		-	-	-				
2 PIK				-	-	-		
1 BLU						-	-	-

Here are schematics showing how to interface a unipolar stepper motor to four controller pins using a ULN2003A, and showing how to interface using four com



Wiring diagram





Test instructions

This code use four step four pat mode to drive the stepping motor forward and reverse direction.

C_Code_10_Stepper_Motor

First, observe the project result, then analyze the code.

1.Use cd command to enter 10_Stepper_Motor directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/10_Stepper_Motor
```

2. Use the following command to compile the code “Stepper_Motor.c ” and generate executable file “Stepper_Motor.c ”

```
gcc Stepper_Motor.c -o Stepper_Motor -lwiringPi
```

3. Then run the generated file “Stepper_Motor”.

```
sudo ./Stepper_Motor
```

After the program is executed, the stepping motor will rotate 360° clockwise and then 360° anticlockwise,circularly

Code explanation

```
void moveOnePeriod(int dir,int ms){}
```

```
/*Subfunction moveOnePeriod ((int dir,int ms) will drive the stepping motor rotating four step clockwise or anticlockwise, four step as a cycle. Where parameter "dir" indicates the rotation direction, if "dir" is 1, the servo will rotate forward, otherwise it rotates to reverse direction. Parameter "ms" indicates the time between each two steps. The "ms" of stepping motor used in this project is 3ms (the shortest time), less than 3ms will exceed the speed limit of stepping motor resulting in that motor can not rotate.*/

void motorStop()
   {
      int i;
      for(i=0;i<4;i++)
      {
      digitalWrite(motorPins[i],LOW);
      }
   }
//function used to stop rotating


void moveSteps(int dir, int ms, int steps){}
/*Subfunction moveSteps (int dir, int ms, int steps) is used to specific cycle number of stepping motor.*/
```

Python_Code_10_Stepper_Motor

First, observe the project result, then analyze the code.

1. Use cd command to enter 10_Stepper_Motor directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/10_Stepper_Motor
```

2. Use Python command to execute Stepper_Motor.py.

```
python Stepper_Motor.py
```

After the program is executed, the stepping motor will rotate 360° clockwise and then 360° anticlockwise,circularly

Code explanation

```
def motorStop (): #Subfunction motorStop () is used to stop the stepping motor

def moveSteps( (direction,ms,steps)
" " "Subfunction moveSteps (direction, ms, steps) is used to specific cycle number of stepping motor.
" " "

def moveOnePeriod( (direction, ,ms):
" " "Subfunction moveOnePeriod (direction, ms) will drive the stepping motor rotating four step clockwise or anticlockwise, four step as a cycle. Where parameter "dir" indicates the rotation direction, if "dir" is 1, the servo will rotate forward, otherwise it rotates to reverse direction. Parameter "ms" indicates the time between each two steps. The "ms" of stepping motor used in this project is 3ms (the shortest time), less than 3ms will exceed the speed limit of stepping motor resulting in that motor can not rotate."
```

Lesson 11 Segment Display

About this lesson:

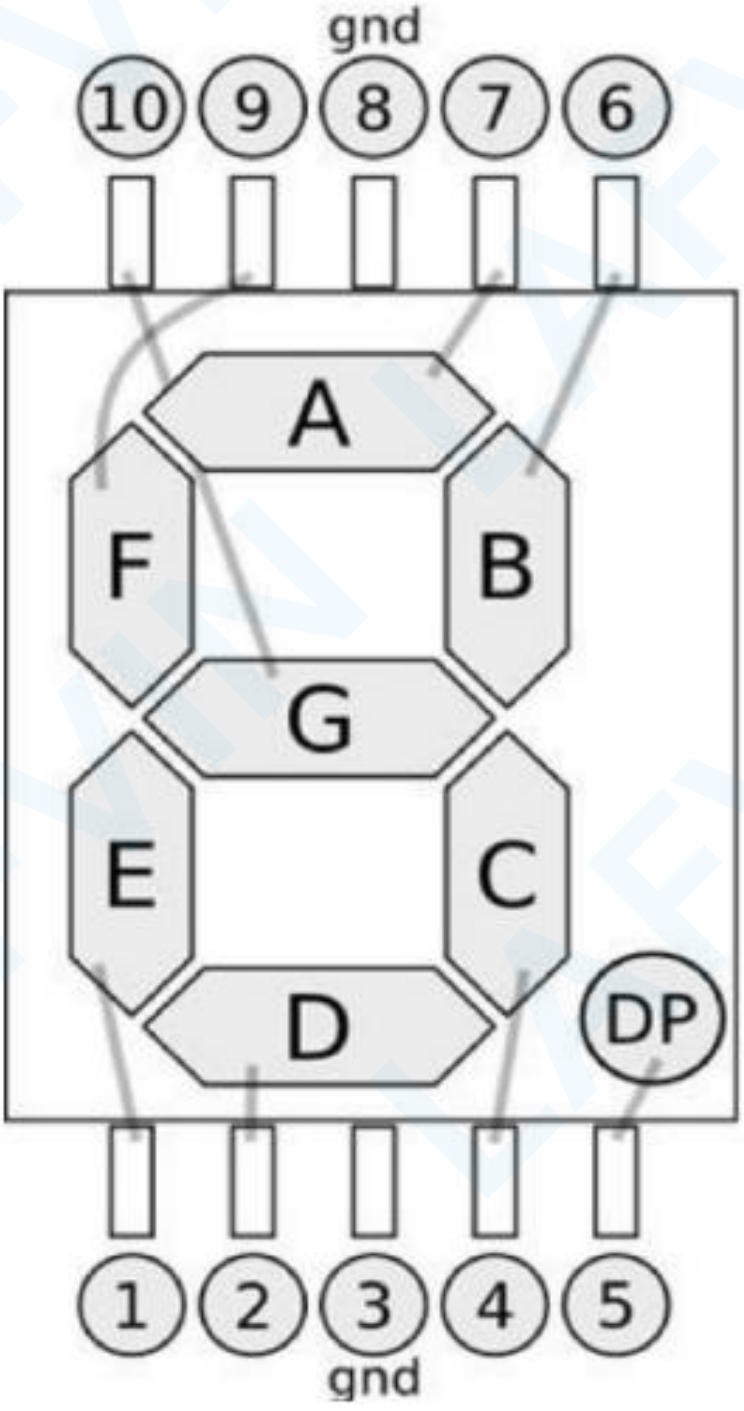
In this lesson, we will use the 74HC595 shift register to control the segment display.

The segment display will show number from 9-0.

Introduction

Seven segment display

Below is the seven-segment pin diagram.

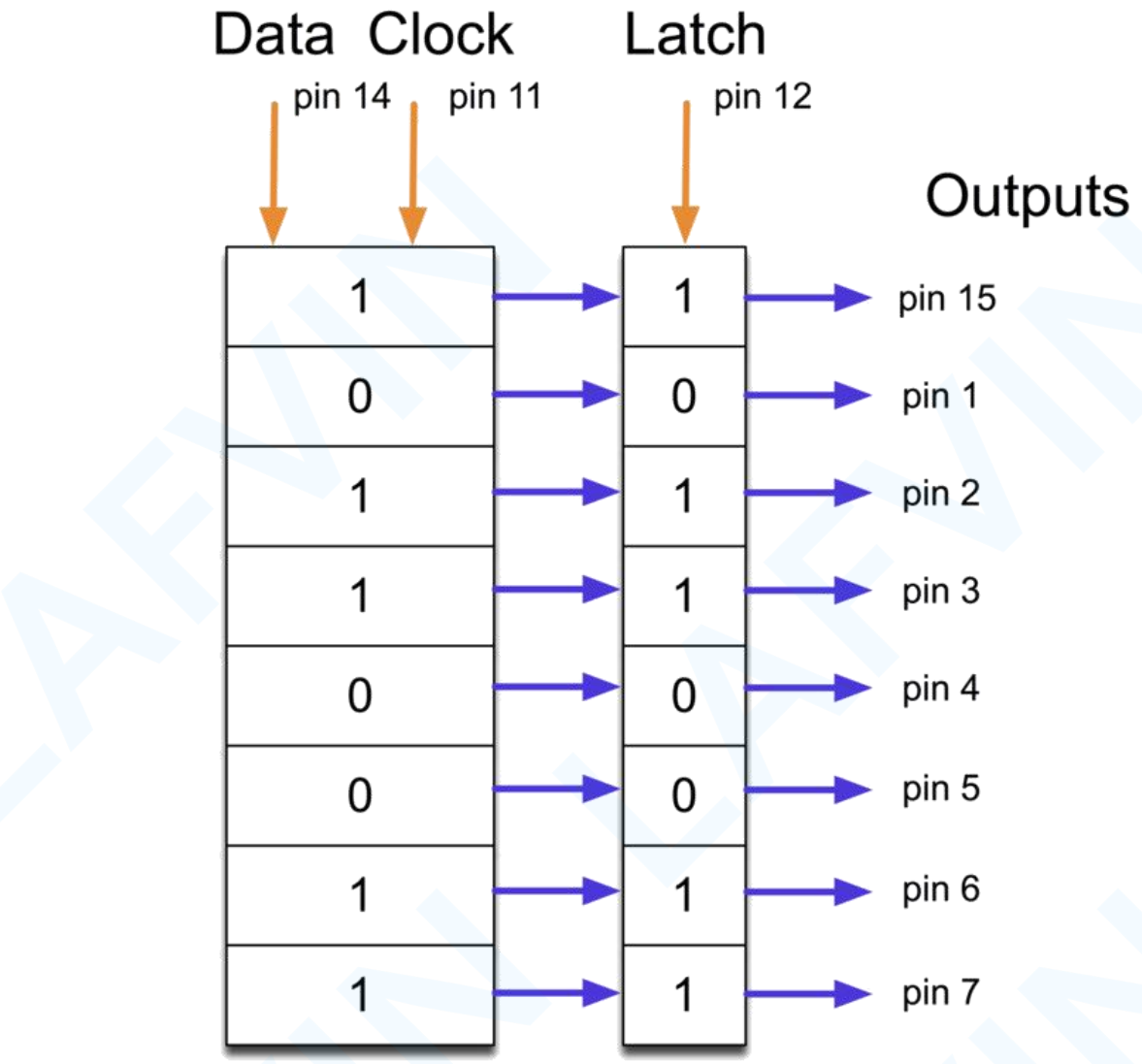


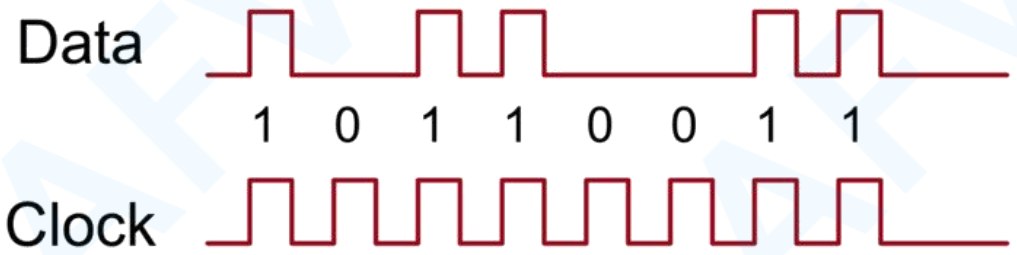
0-9 ten digits correspond with each segment are as follows (the following table applies common cathode seven segment display device, if you are using a common anode, the table should be replaced every 1 0 0 should all replaced by 1):

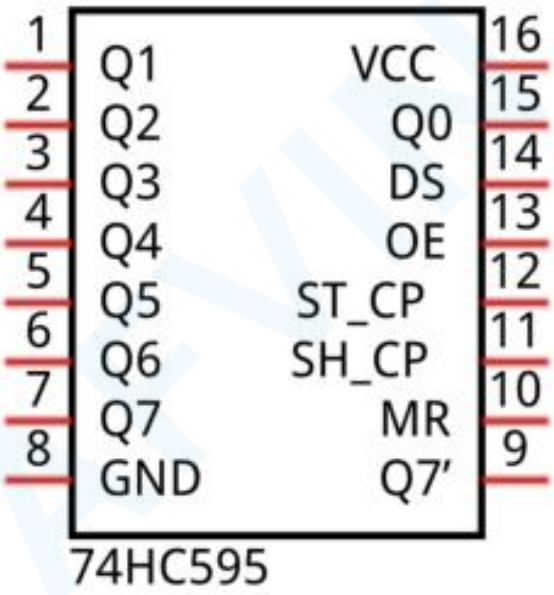
Display digital	dp	a	b	c	d	e	f	g
0	0	1	1	1	1	1	1	0
1	0	0	1	1	0	0	0	0
2	0	1	1	0	1	1	0	1
3	0	1	1	1	1	0	0	1
4	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1
6	0	1	0	1	1	1	1	1
7	0	1	1	1	0	0	0	0
8	0	1	1	1	1	1	1	1
9	0	1	1	1	1	0	1	1

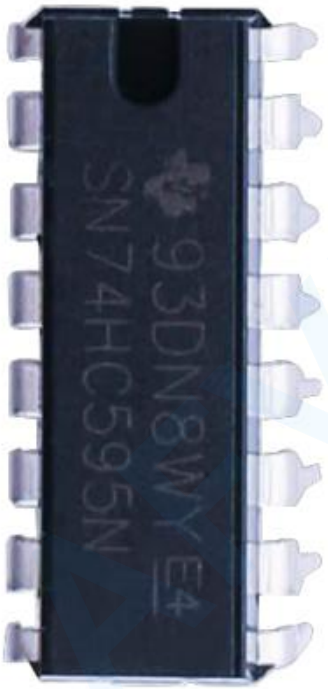
74HC595 Shift Register:

The shift register is a type of chip that holds what can be thought of as eight memory locations, each of which can either be a 1 or a 0. To set each of these values on or off, we feed in the data using the 'Data' and 'Clock' pins of the chip.


Data
Clock
Latch
pin 14
pin 11
pin 12
Outputs
1
0
1
1
0
0
0
1
1
1
0
1
1
0
0
0
1
1
pin 15
pin 1
pin 2
pin 3
pin 4
pin 5
pin 6
pin 7

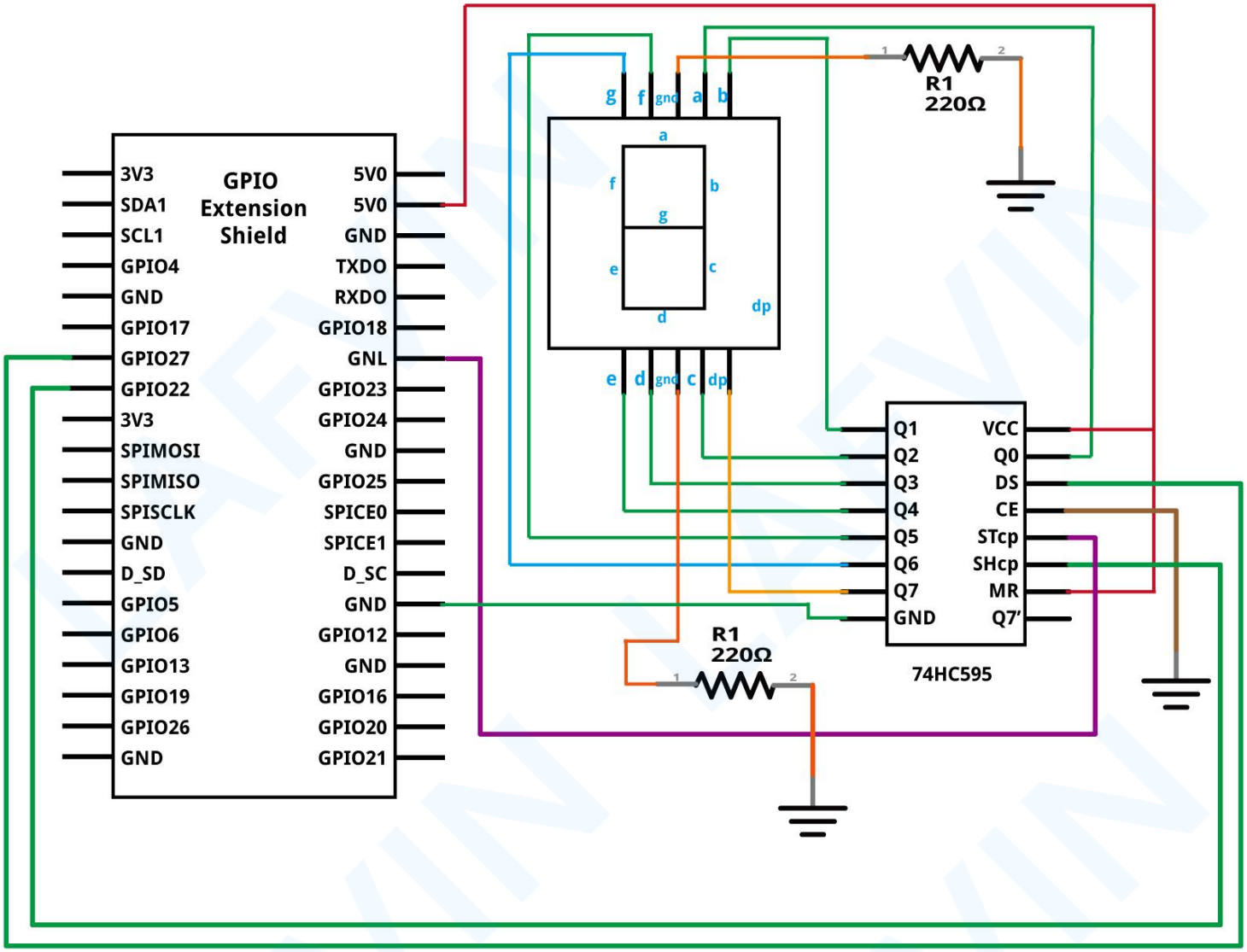

Data
1 0 1 1 0 0 1 1
Clock

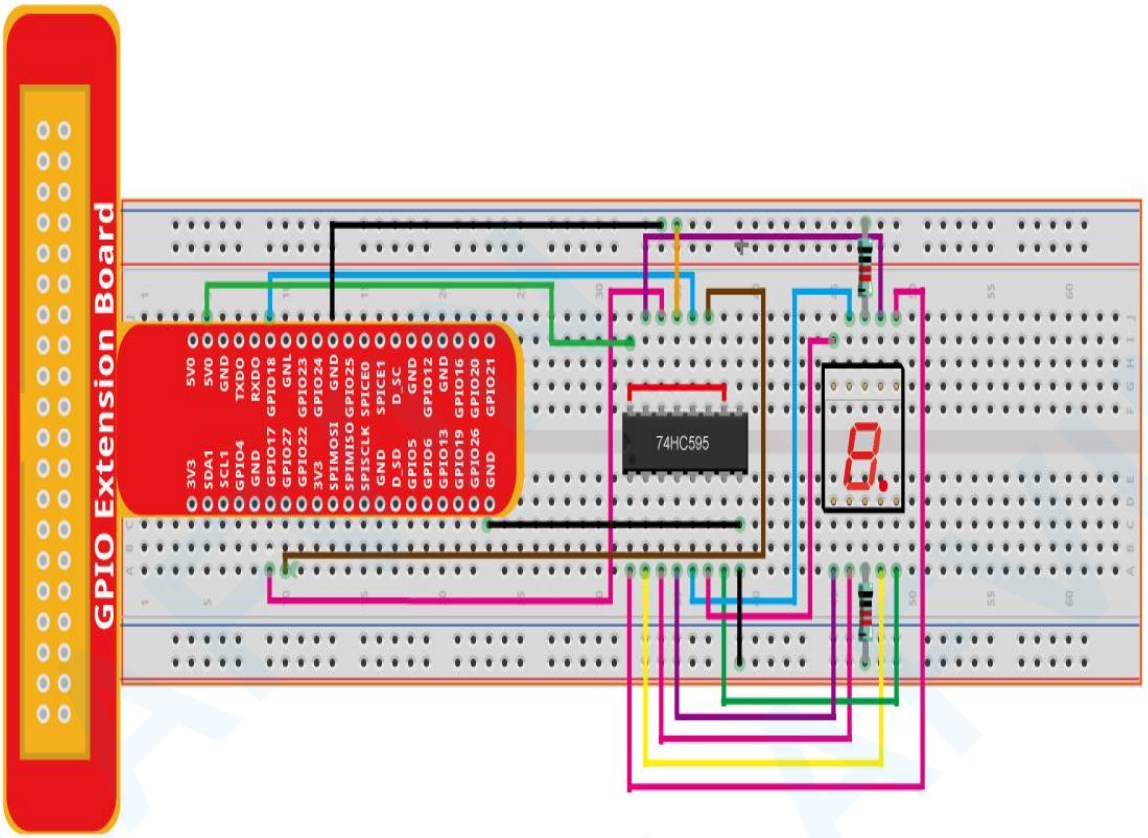




Pins of 74HC595 and their functions:
Q0-Q7: 8-bit parallel data output pins, able to control 8 LEDs or 8 pins of 7-segment display
Q7: Series output pin, connected to DS of another 74HC595 to connect multiple 74HC595s in series
MR: Reset pin, active at low level; here it is directly connected to 5V to keep the chip from resetting.
SH_CP: Time sequence input of shift register. On the rising edge, the data in shift register moves successively one bit, i.e. data in Q1 moves to Q2, and so forth. While on the falling edge, the data in shift register remain unchanged.
ST_CP: Time sequence input of storage register. On the rising edge, data in the shift register moves into memory register.
OE: Output enable pin, active at low level; here connected to GND to keep 74HC595 in output enable state.
DS: Serial data input pin
VCC: Positive supply voltage
GND: Ground

Wiring diagram





Test instructions

In this code ,Raspberry Pi controls 7 digits to display the specified number.

C_Code_11_Segment_Display

First, observe the project result, then analyze the code.

6. Use cd command to enter 11_Segment_Display directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/11_Segment_Display
```

2. Use the following command to compile the code “Segment_Display.c ” and generate executable file “Segment_Display.c ”

```
gcc Segment_Display.c -o Segment_Display -lwiringPi
```

3. Then run the generated file “Segment_Display”.

```
sudo ./Segment_Display
```

The hexadecimal format of number 0~15 are (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F) You should see the 7-segment display from 0 to 9 and A to F

Python_Code_11_Segment_Display

First, observe the project result, then analyze the code.

1. Use cd command to enter 8_LCD1602 directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/11_Segment_Display
```

2. Use Python command to execute Segment_Display.py.

```
python Segment_Display.py
```

After the program is executed, SevenSegmentDisplay starts to display the character “0”- “F” successivel

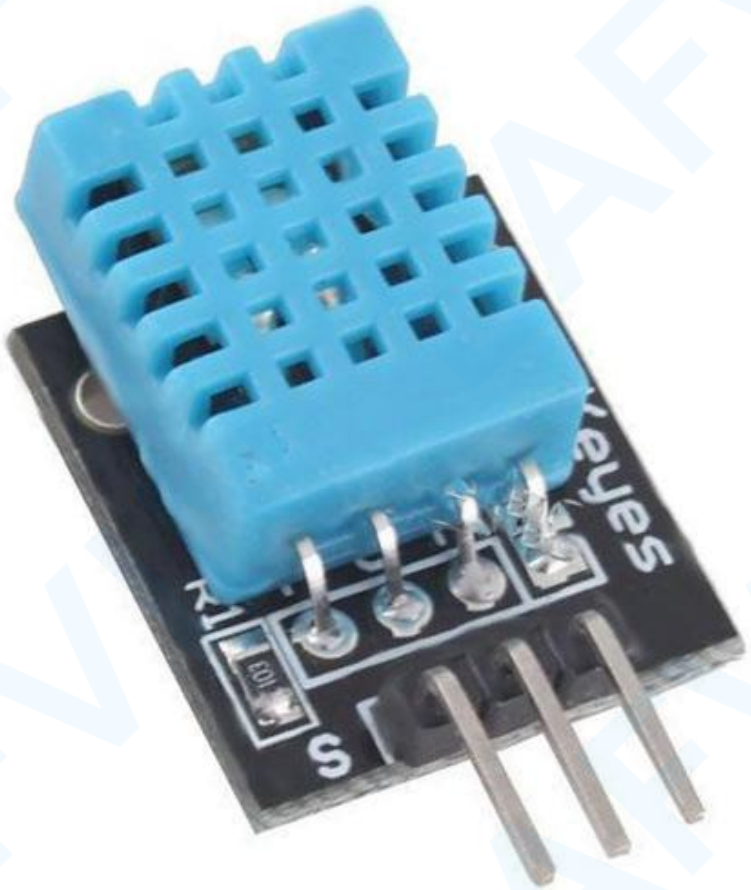
Lesson 12 DHT11

About this lesson:

In this lesson, we will learn how to use a DHT11 Temperature and Humidity Sensor. It's accurate enough for most projects that need to keep track of humidity and temperature readings.

Introduction

Temp and humidity sensor:



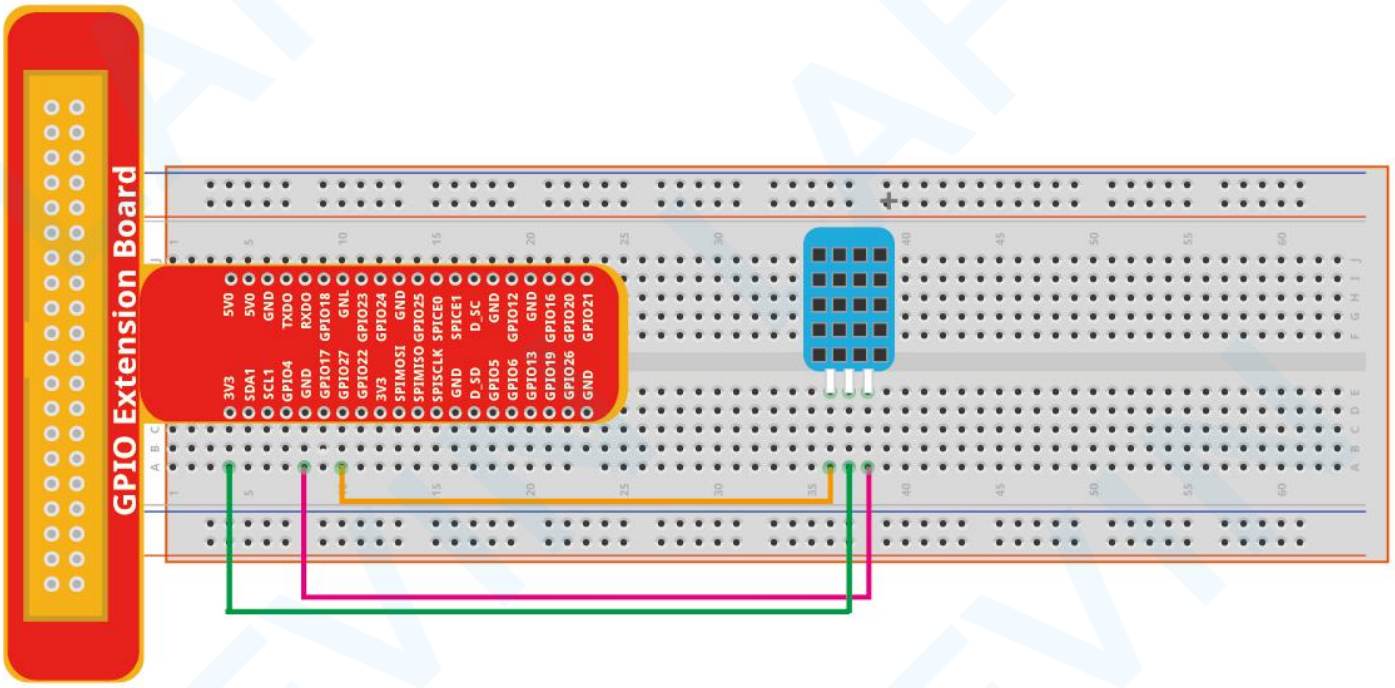
DHT11 digital temperature and humidity sensor is a composite Sensor which contains a calibrated digital signal output of the temperature and humidity. The dedicated digital modules collection technology and the temperature and humidity sensing technology are applied to ensure that the product has high reliability and

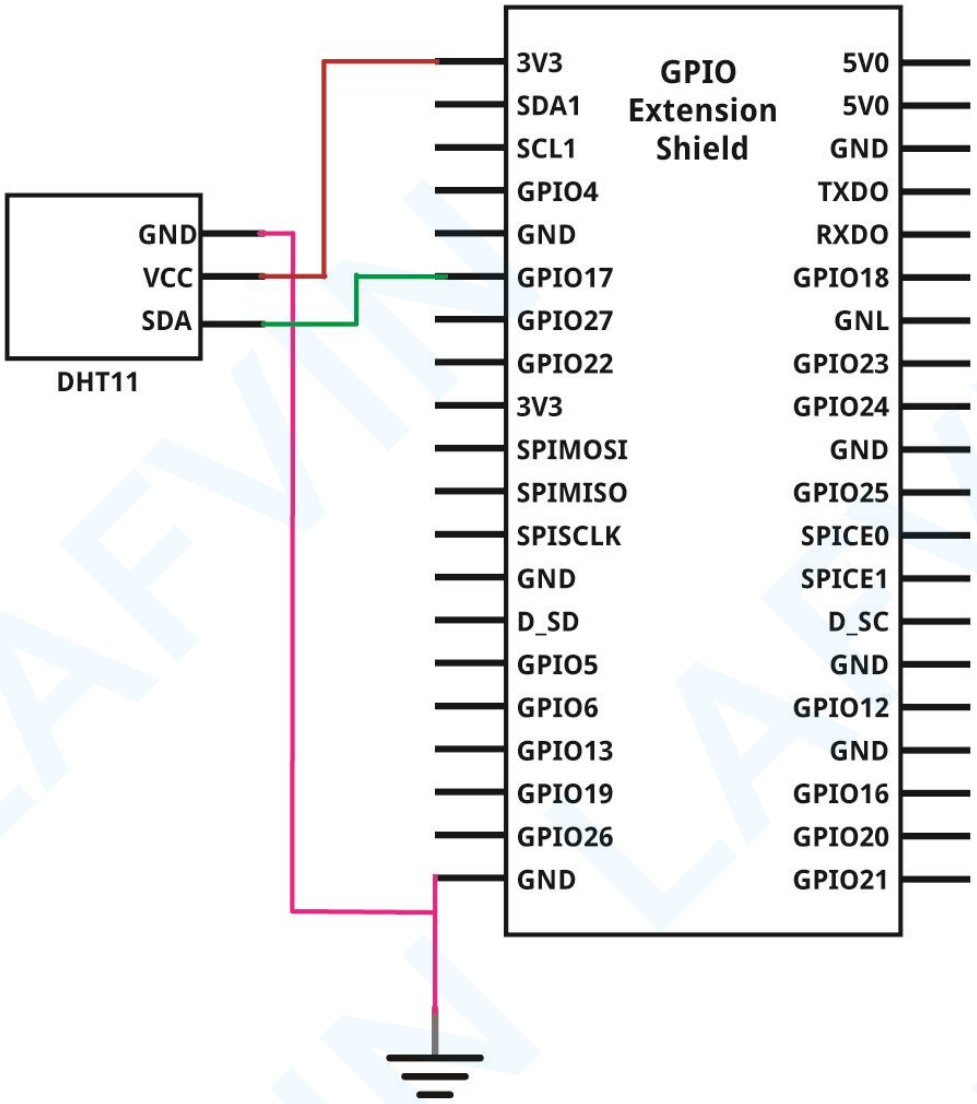
excellent long-term stability. The sensor includes a resistive sense of wet components and a NTC temperature measurement devices, and connects with a high-performance 8-bit microcontroller.

Pin Description:

1. the VDD power supply 3.5～5.5V DC
2. DATA serial data, a single bus
3. NC, empty pin
4. GND ground, the negative power

Wiring diagram





Test instructions

This code will get the CPU temperature and system time of raspberry pi, display them on LCD1602.

C_Code_12_DHT11

First, observe the project result, then analyze the code.

1.Use cd command to enter 12_DHT11 directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/12_DHT11
```

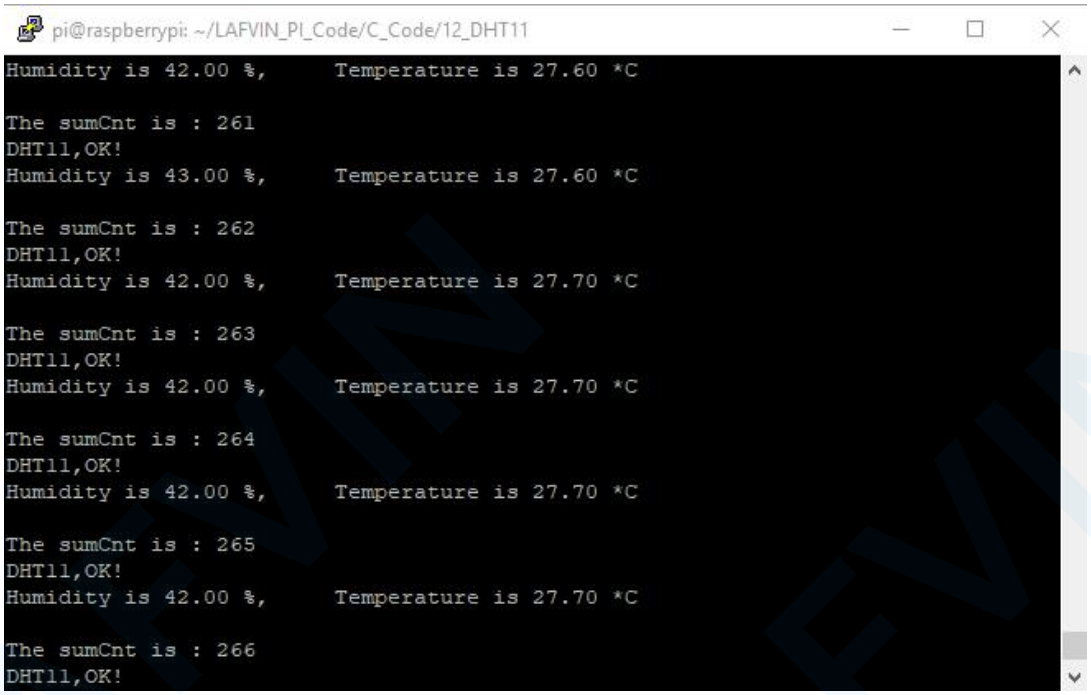
2. Code of this project contains a custom header file. Use the following command to compile the code DHT11.cpp and DHT.cpp and generate executable file DHT11. And the custom header file will be compiled at the same time.

```
gcc DHT.cpp DHT11.cpp -o DHT11 -lwiringPi
```

3. Then run the generated file "DHT11 ".

```
sudo ./DHT11
```

After the program is executed, the terminal window will display the current total number of reading times, the read state, as well as the temperature and humidity value. As is shown below:



Code explanation

```
DHT dht;
/*we use a custom library file "DHT.hpp". It is located in the same directory with program files "DHT11.cpp" and "DHT.cpp", and methods for reading DHT sensor are provided in the library file. By using this library, we can easily read the DHT sensor. First create a DHT class object in the code.*/

chk = dht.readDHT11( (DHT11_Pin );
/*use chk = dht.readDHT11 (DHT11_Pin) to read the DHT11, and determine whether the data read is normal according to the return value "chk".*/


class DHT{ {
public:
double humidity, ,temperature; ; //use to store temperature and humidity data read
int readDHT11( (int pin ); //read DHT11
private:
int bits[ [5 ]; //Buffer to receiver data
int readSensor( (int pin, ,int wakeupDelay );
};
/*Library file "DHT.hpp" contains a DHT class and his public member functions int readDHT11 (int pin) is used to read sensor DHT11 and store the temperature and humidity data read to member variables double humidity and temperature. The implementation method of the function is included in the file "DHT.cpp"*/
```

Python_Code_12_DHT11

First, observe the project result, then analyze the code.

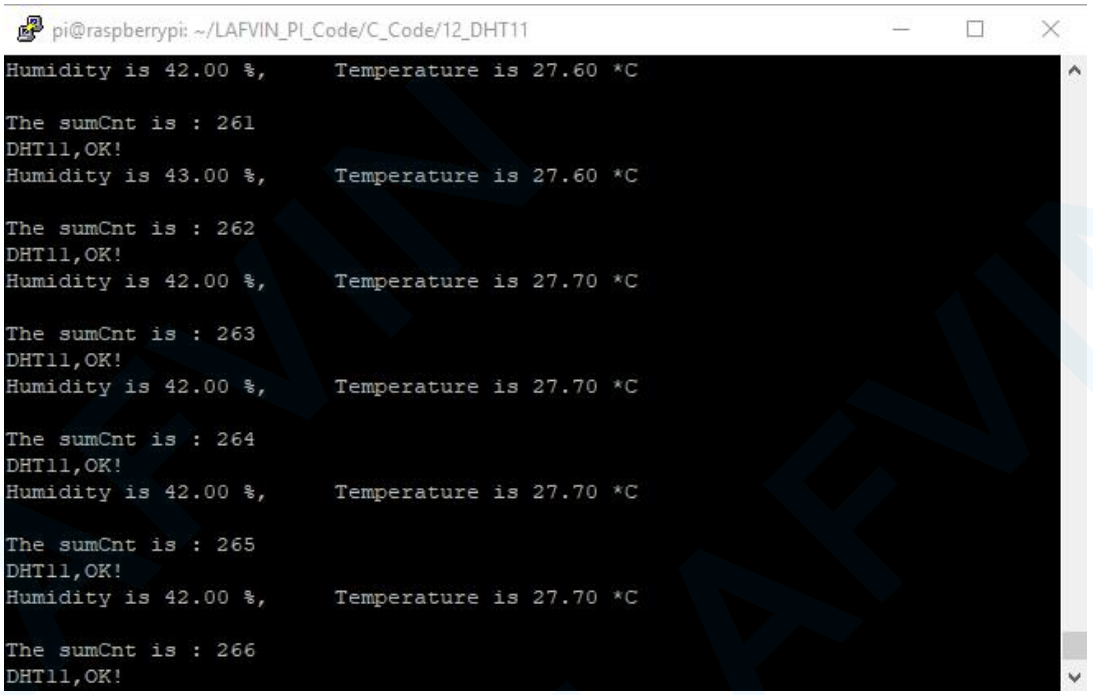
1. Use cd command to enter 12_DHT11 directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/12_DHT11
```

2. Use Python command to execute DHT11.py.

```
python DHT11.py
```

After the program is executed, the terminal window will display the current total number of reading times, the read state, as well as the temperature and humidity value. As is shown below:



Code explanation

```
import RPi. .GPIO    as GPIO
import time
import LAFVIN_DHT    as DHT
" " "
we use a module "LAFVIN_DHT.py", which provide method of reading sensor DHT. It is located in the same directory with program files "DHT11.py". By using this library, we can easily read the DHT sensor. First create a DHT class object in the code..
This is a Python module for reading the temperature and humidity data of the DHT sensor. Partial
functions and variables are described as follows:
Variable humidity: store humidity data read from sensor
Variable temperature: store temperature data read from sensor
def    readDHT11 (pin): read the temperature and humidity of sensor DHT11, and return values used todetermine whether the data is normal.
" " "

dht = DHT. .DHT( (DHTPin) ) #create a DHT class object
```

Lesson 13 Ultrasonic

About this lesson:

Ultrasonic sensor is great for all kind of projects that need distance measurements, avoiding obstacles as examples.

Introduction

Ultrasonic sensor

Ultrasonic sensor module HC-SR04 provides 2cm-400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

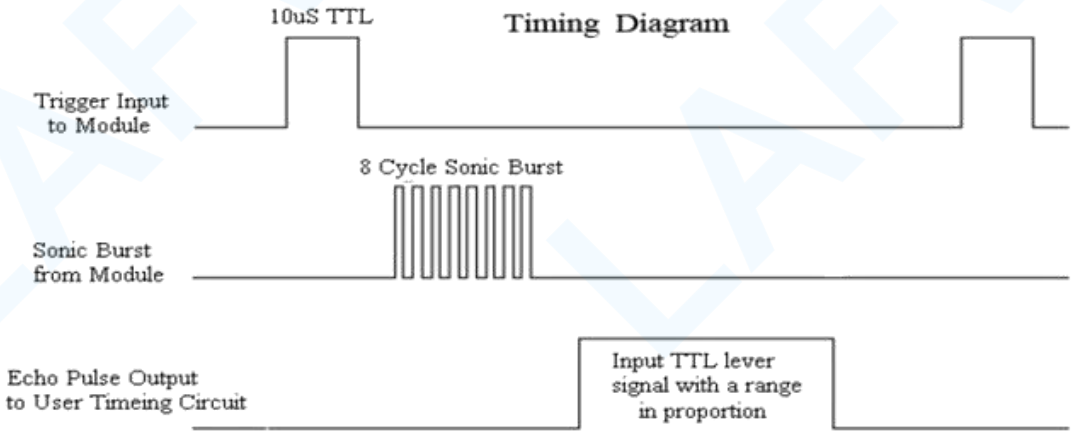
(1) Using IO trigger for at least 10us high level signal,

(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.

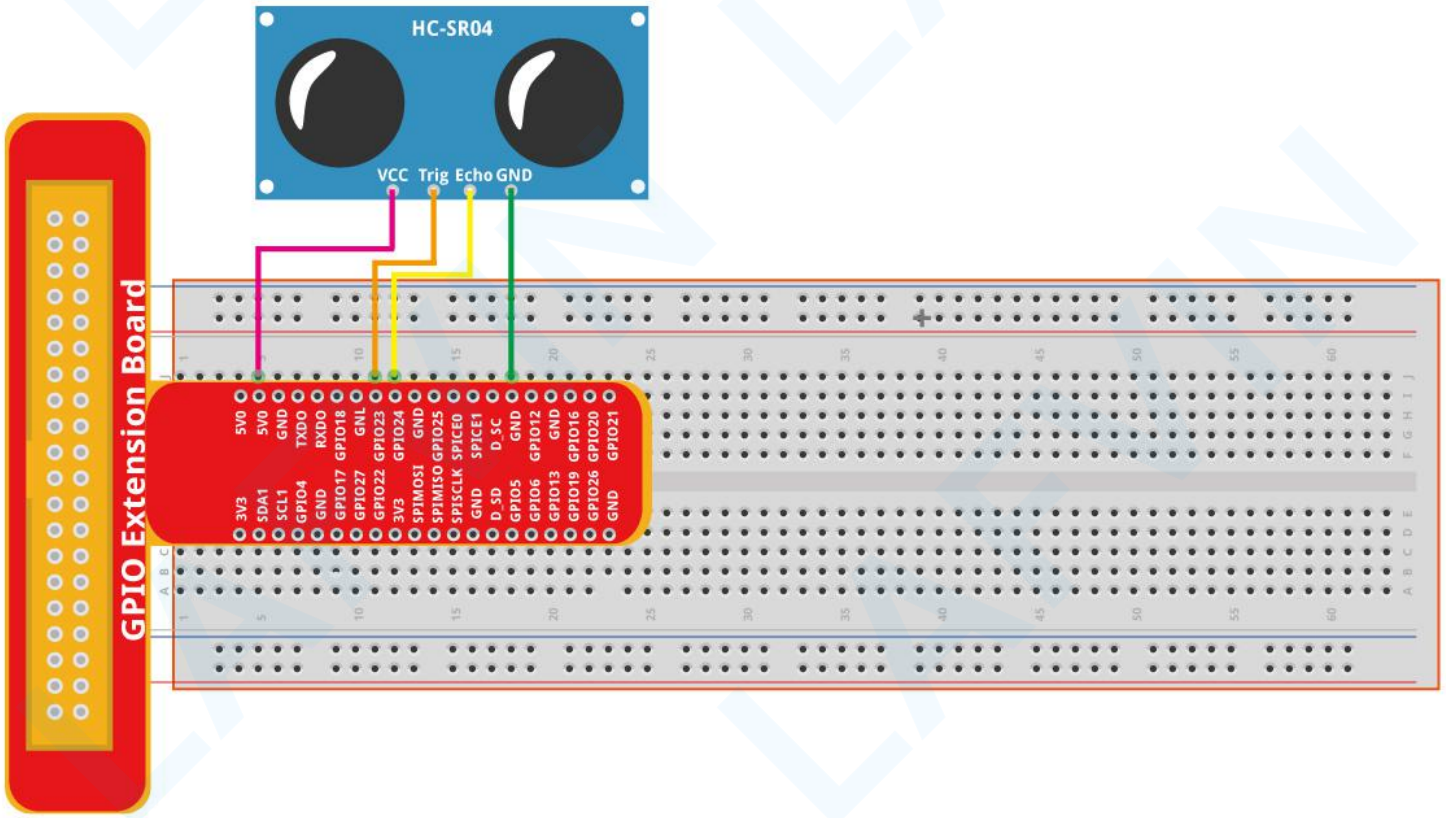
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic tore turning.

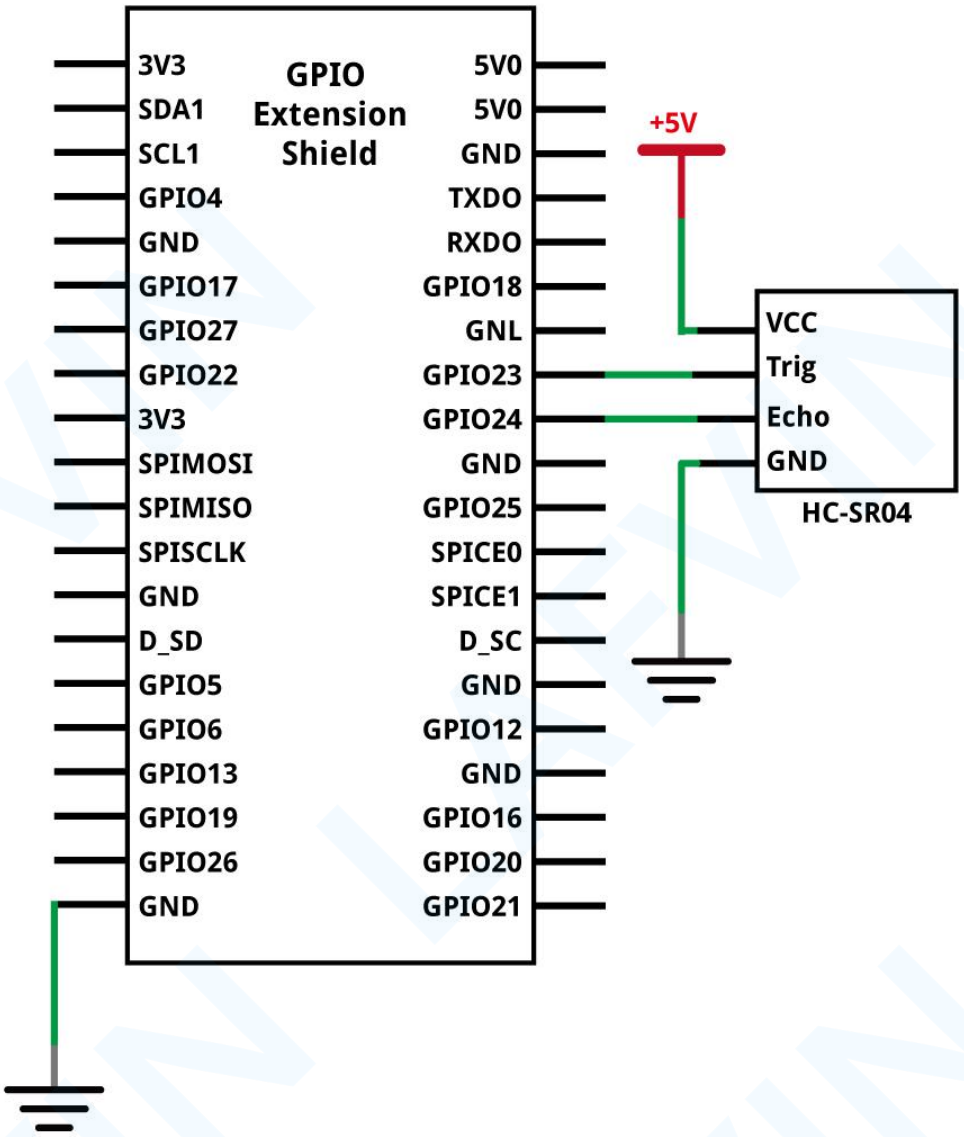
Test distance = (high level time × velocity of sound (340m/s) /2

The Timing diagram is shown below. You only need to supply a short 10us pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: us / 58 = centimeters or us / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



Wiring diagram





Test instructions

C_Code_13_Ultrasonic

First, observe the project result, then analyze the code.

7. Use cd command to enter 13_Ultrasonic directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/13_Ultrasonic
```

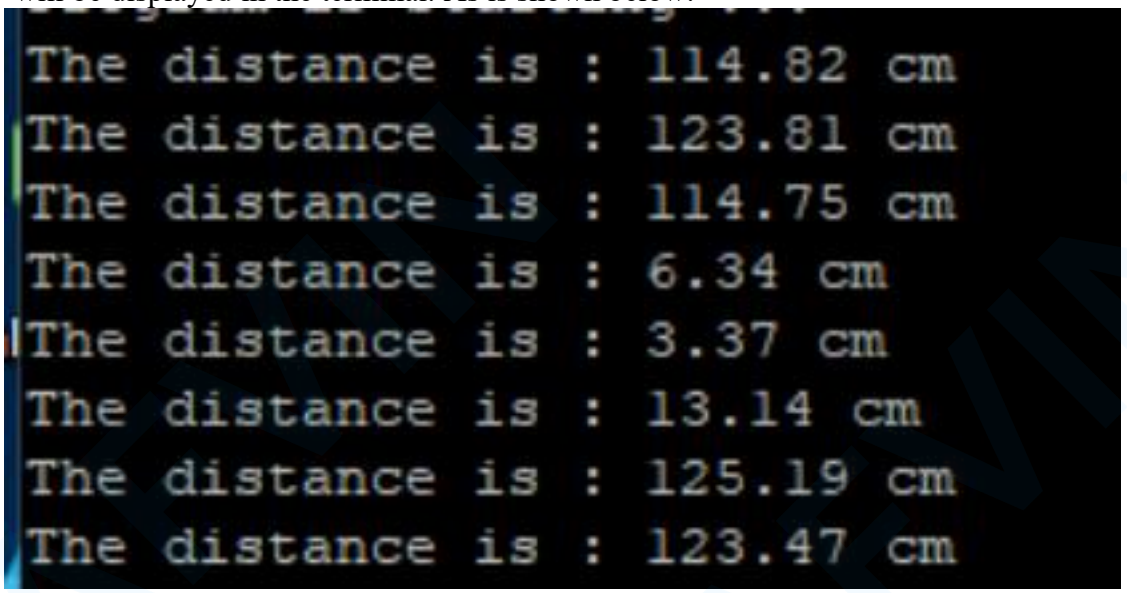
2. Use the following command to compile the code “Ultrasonic.c ” and generate executable file “Ultrasonic.c ”

```
gcc Ultrasonic.c -o Ultrasonic -lwiringPi
```

3. Then run the generated file “Ultrasonic”.

```
sudo ./Ultrasonic
```

After the program is executed, make the detector of ultrasonic ranging module aim at the plane of an object,then the distance between the ultrasonic module and the object will be displayed in the terminal. As is shown below:



Code explanation

```
int pulseIn(int pin, int level, int timeout);
/*Return the length of the pulse (in microseconds) or 0 if no pulse is completed before the timeout (unsigned long).*/

#define timeOut MAX_DISTANCE*60
/*If the module does not return high level, we can not wait forever. So we need to calculate the lasting time over maximum distance, that is,
time Out. timOut= 2*MAX_DISTANCE/100/340*1000000.
  The constant part behind is approximately equal to 58.8.*/
```

Python_Code_13_Ultrasonic

First, observe the project result, then analyze the code.

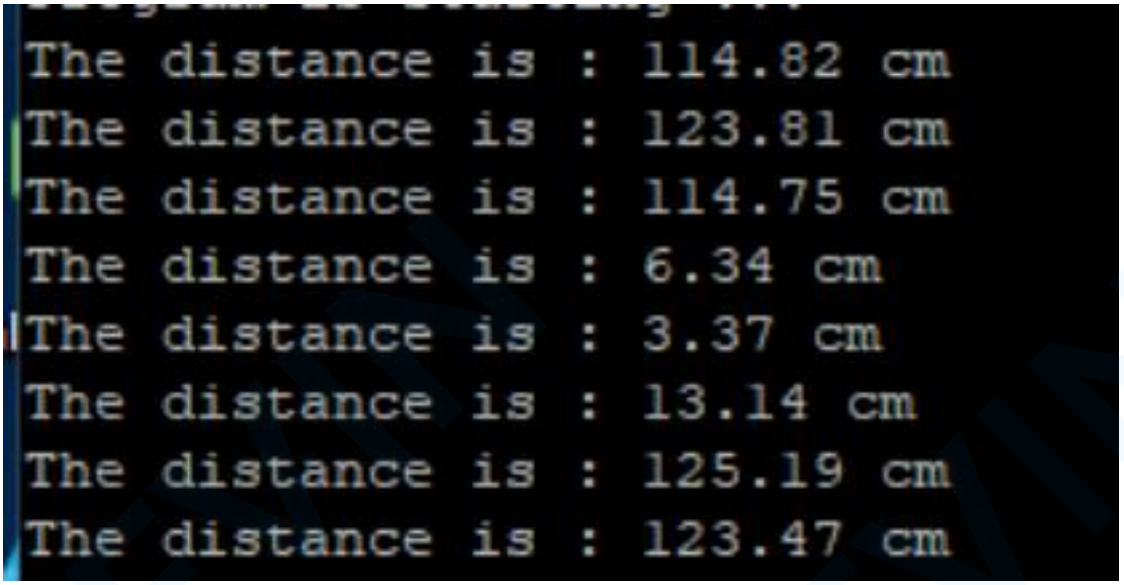
1. Use cd command to enter 13_Ultrasonic directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/13_Ultrasonic
```

2. Use Python command to execute Ultrasonic.py.

```
python Ultrasonic.py
```

After the program is executed, make the detector of ultrasonic ranging module aim at the plane of an object,then the distance between the ultrasonic module and the object will be displayed in the terminal. As is shown below:



Code explanation

```
timeOut = =    MAX_DISTANCE* *60
" " "If the module does not return high level, we can not wait forever. So we need to calculate the lasting time over maximum distance, that is,
time Out. timOut= 2*MAX_DISTANCE/100/340*1000000.
  The constant part behind is approximately equal to 58.8." " "

def pulseIn(pin,level,timeOut):
" " "Return the length of the pulse (in microseconds) or 0 if no pulse is completed before the timeout (unsigned long)."
```

Lesson 14 Membrane Switch Module

About this lesson:

In this lesson,In this project, we will go over how to integrate a keyboard with a raspberry pi so that the raspberry pi can read the keys being pressed by a user.

Introduction

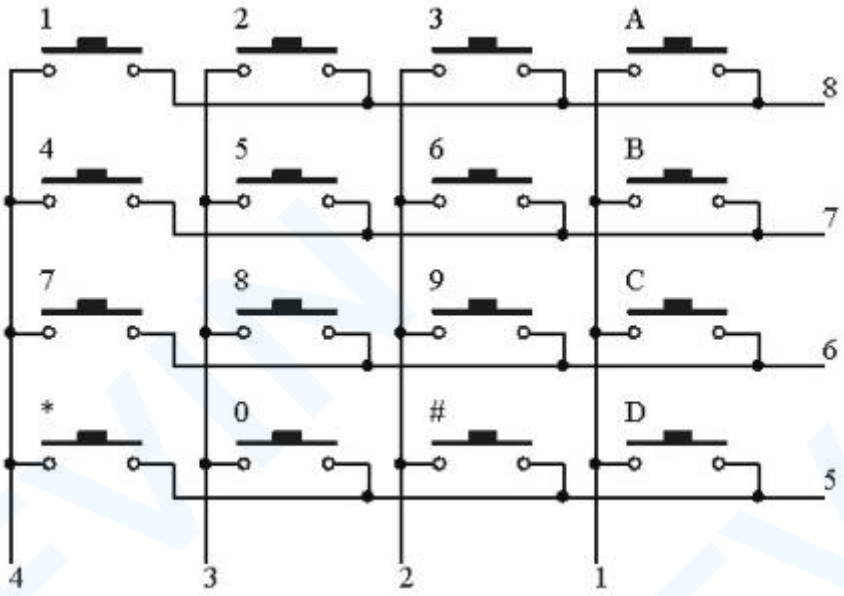
Membrane Switch Module

Keypads are used in all types of devices, including cell phones, fax machines, microwaves, ovens, door locks, etc. They're practically everywhere. Tons of electronic devices use them for user input.

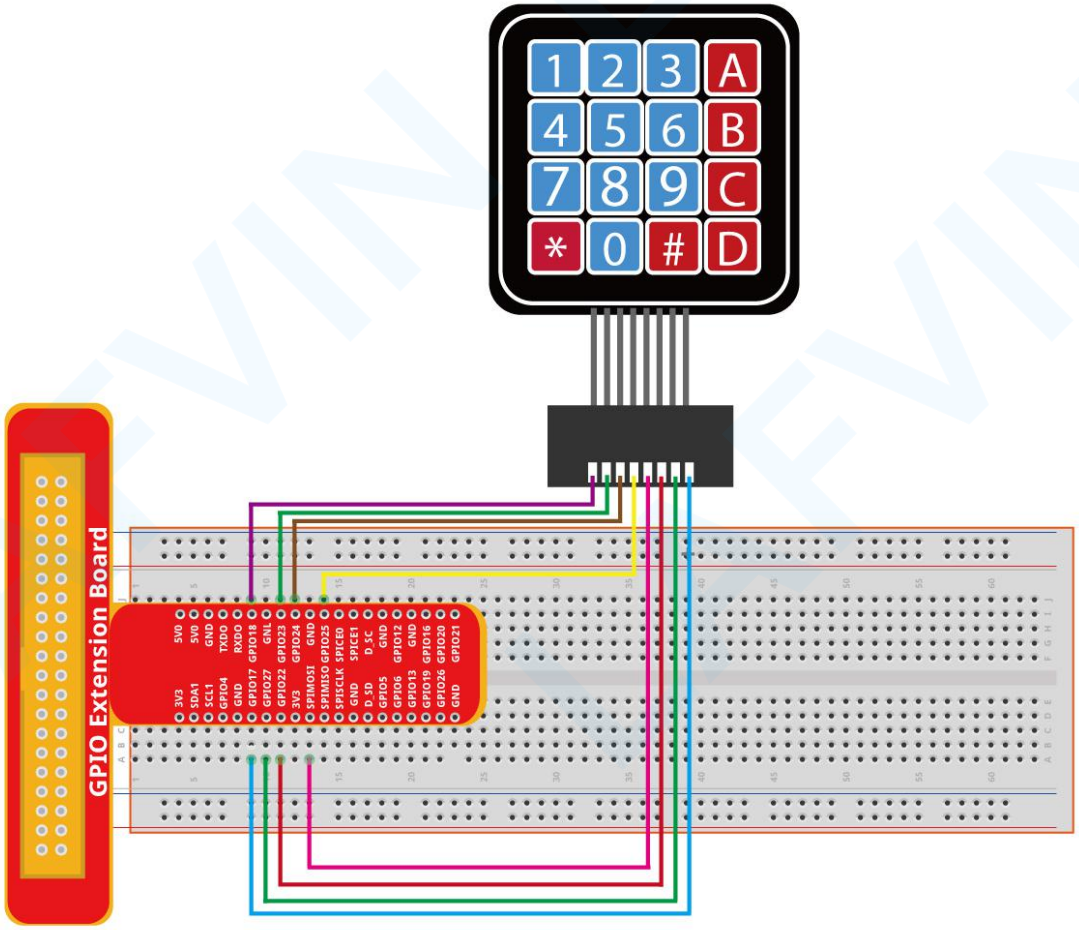
So knowing how to connect a keypad to a microcontroller such as an UNO R3 board is very valuable for building many different types of commercial products.

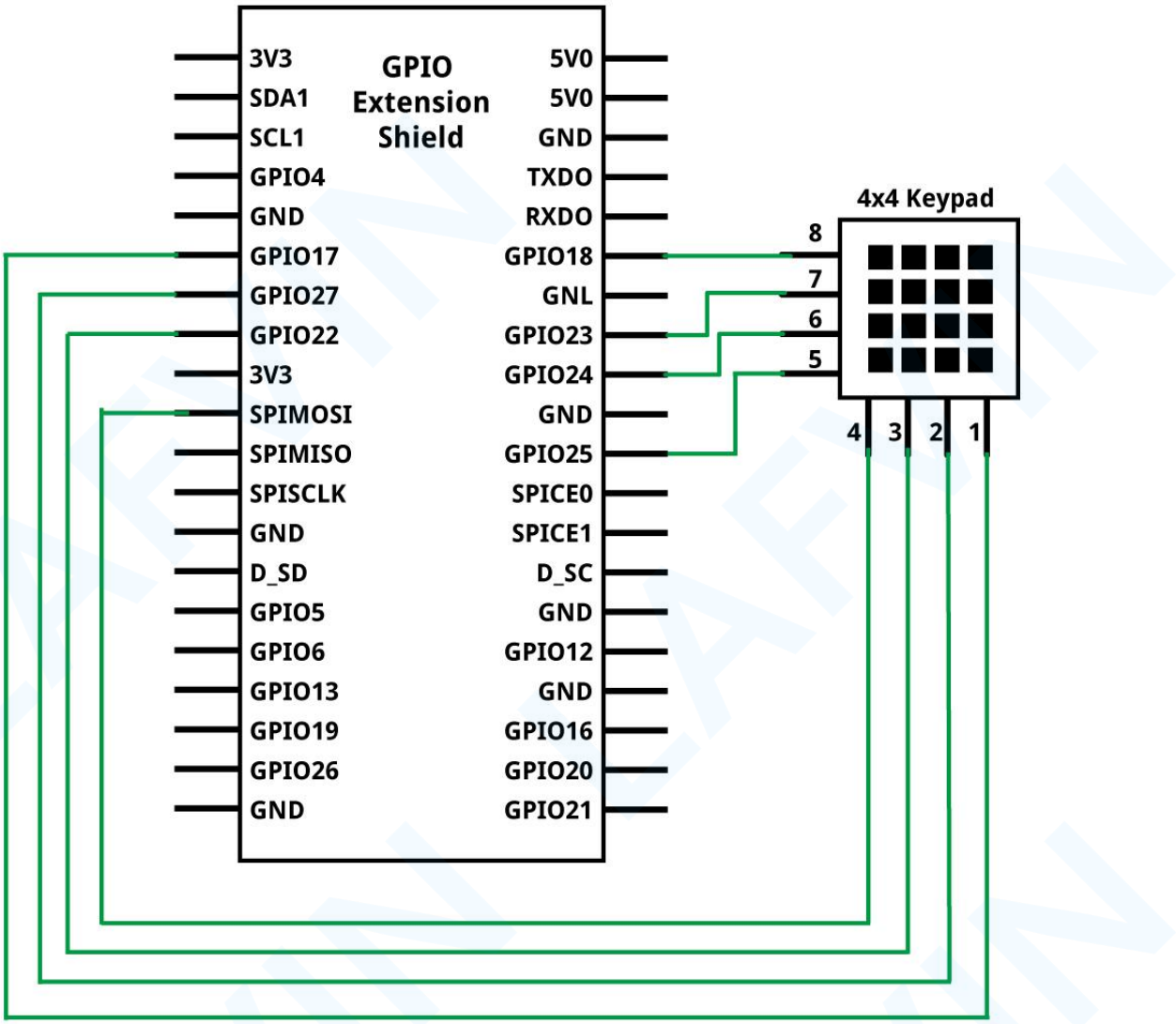
At the end when all is connected properly and programmed, when a key is pressed, it shows up at the Serial Monitor on your computer. Whenever you press a key, it shows up on the Serial Monitor. For simplicity purposes, we start at simply showing the key pressed on the computer.

For this project, the type of keypad we will use is a matrix keypad. This is a keypad that follows an encoding scheme that allows it to have much less output pins than there are keys. For example, the matrix keypad we are using has 16 keys (0-9, A-D, *, #), yet only 8 output pins. With a linear keypad, there would have to be 17 output pins (one for each key and a ground pin) in order to work. The matrix encoding scheme allows for less output pins and thus much less connections that have to make for the keypad to work. In this way, they are more efficient than linear keypads, being that they have less wiring.



Wiring diagram




GPIO
Extension
Shield
3V3
SDA1
SCL1
GPIO4
GND
GPIO17
GPIO27
GPIO22
3V3
SPIMOSI
SPIMISO
SPISCLK
GND
D_SD
GPIO5
GPIO6
GPIO13
GPIO19
GPIO26
GND
5V0
5V0
GND
TXDO
RXDO
GPIO18
GNL
GPIO23
GPIO24
GND
GPIO25
SPICE0
SPICE1
D_SC
GND
GPIO12
GND
GPIO16
GPIO20
GPIO21
4x4 Keypad
8
7
6
5
4
3
2
1

Test instructions

This code is used to obtain all key code of 4x4 Matrix Keypad, when one of keys is pressed, the key code will
be printed out in the terminal window.

C_Code_14_Membrane_Switch_Module

First, observe the project result, then analyze the code.

8. Use cd command to enter 14_Membrane Switch Module directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/14_Membrane_Switch_Module
```

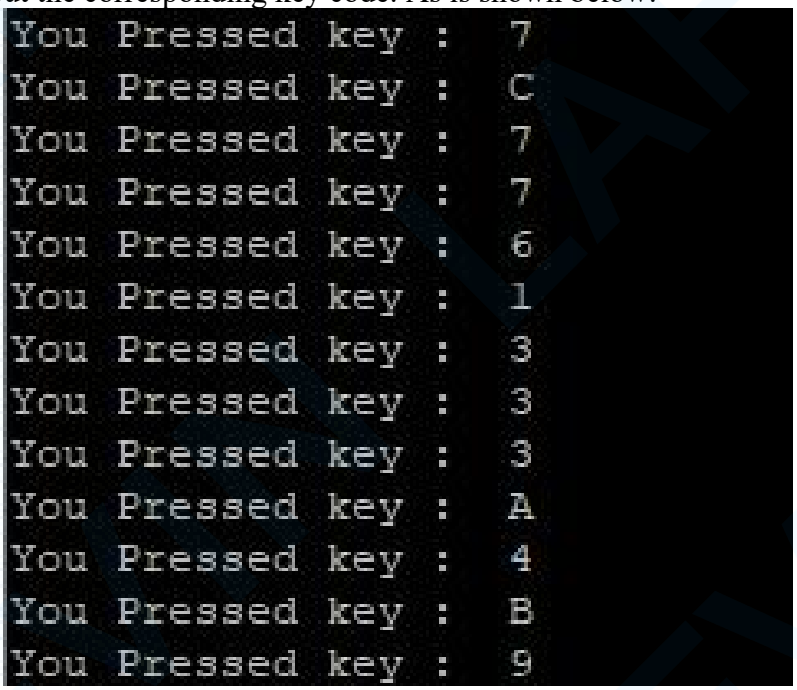
2. Use the following command to compile the code
"Membrane_Switch_Module.c " and generate executable file
"Membrane_Switch_Module.c "

```
gcc Membrane_Switch_Module.cpp Keypad.cpp Key.cpp -o Membrane_Switch_Module -lwiringPi
```

3. Then run the generated file "Membrane_Switch_Module".

```
sudo ./Membrane_Switch_Module
```

After the program is executed, press any key on the MatrixKeypad, the terminal will print out the corresponding key code. As is shown below:



Code explanation

```
#include "Keypad.hpp"
/*In this project code, we use two custom library file "Keypad.hpp" and "Key.hpp". They are located in the same directory with program files "MatrixKeypad.cpp", "Keypad.cpp" and "Key.cpp". Library Keypad is transplanted from the Arduino library Keypad. And this library file provides a method to read the keyboard. By using this library, we can easily read the matrix keyboard.*/

Keypad keypad = = Keypad( ( makeKeymap( (keys ), rowPins, , colPins, , ROWS, , COLS );/*based on the above information, instantiate a Keypad class object to operate the matrix keyboard.*/
```

class Keypad

Keypad((char * * userKeymap, , byte * * row, , byte * * col, , byte numRows, , byte numCols);

```
/*Constructor, the parameters are: key code of keyboard, row pin, column pin, the number of rows, the number of columns.*/

char getKey ();
//Get the key code of the pressed key. If no key is pressed, the return value is NULL.

void setDebounceTime( ( uint );
/*Set the debounce time. And the default time is 10ms.*/

void setHoldTime( ( uint );
/*Set the time when the key holds stable state after pressed.*/

bool isPressed( ( char keyChar );
/*Judge wether the key with code "keyChar" is pressed.*/

char waitForKey ();
/*Wait for a key to be pressed, and return key code of the pressed key.*/

KeyState getState ();
/*Get state of the keys.*/

bool keyStateChanged ();
/*Judge whether there is a change of key state, then return True or False.*/
```

Python_Code_14_Membrane_Switch_Module

First, observe the project result, then analyze the code.

1. Use cd command to enter 8_LCD1602 directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/14_Membrane_Switch_Module
```

2. Use Python command to execute Membrane_Switch_Module.py.

```
python Membrane_Switch_Module.py
```

After the program is executed, press any key on the MatrixKeypad, the terminal will print out the corresponding key code. As is shown below:

Code explanation

```
import Keypad
" " ""Keypad.py" is located in the same directory with program file "MatrixKeypad.py". And this library file, which is transplanted from Arduino function library Keypad, provides a method to read the keyboard. By using this library, we can easily read the matrix keyboard." " "
```

class Keypad

```
def __init__( ( self, , usrKeyMap, , row_Pins, , col_Pins, , num_Rows, , num_Cols ):
" " "Constructed function, the parameters are: key code of keyboard, row pin, column pin, the number of rows, the number of columns." " "

def getKey( ( self ):
#Get a pressed key. If no key is pressed, the return value is keypad NULL.

def setDebounceTime( ( self, , ms ):
#Set the debounce time. And the default time is 10ms.

def setHoldTime( ( self, , ms ):
#Set the time when the key holds stable state after pressed.

def isPressed( ( keyChar ):
#Judge wether the key with code "keyChar" is pressed.

def waitForKey ():
#Wait for a key to be pressed, and return key code of the pressed key.

def getState ():
#Get state of the keys.

def keyStateChanged ():
#Judge whether there is a change of key state, then return True or False.
```

Lesson 15 GY-521 SENSOR

About this lesson:

In this lesson, we will learn how to use GY-521 module which is one of the best IMU (Inertia Measurement Unit) sensors, compatible with Arduino. IMU sensors like the GY-521 are used in self balancing robots, UAVs, smart phones, etc.

Introduction

GY-521 SENSOR

The InvenSense GY-521 sensor contains a MEMS accelerometer and a MEMS gyro in a single chip. It is very accurate, as it contains 16-bits analog to digital conversion hardware for each channel. Therefore it captures the x, y, and z channel at the same time. The sensor uses the I2C-bus to interface with the Arduino.

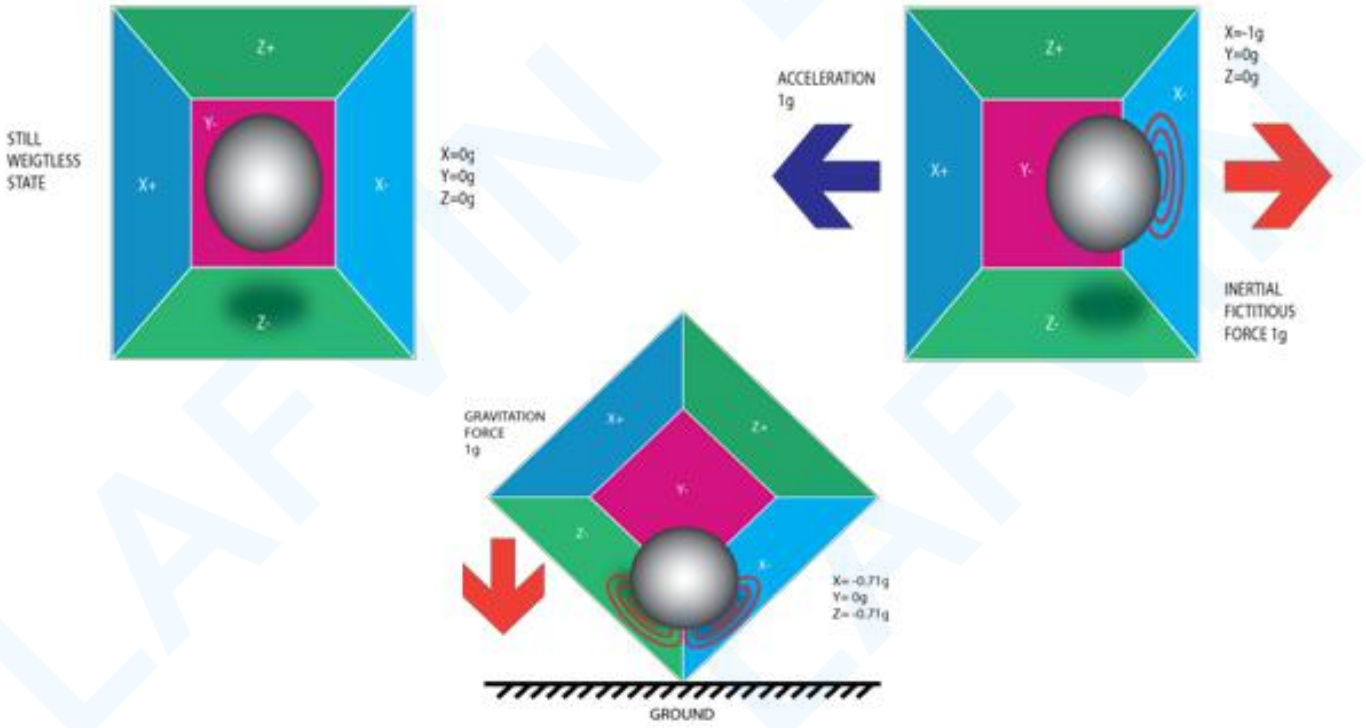
The GY-521 is not expensive, especially given the fact that it combines both an accelerometer and a gyro.

IMU sensors are one of the most inevitable type of sensors used today in all kinds of electronic gadgets. They are seen in smart phones, wearables, game controllers, etc. IMU sensors help us in getting the attitude of an object, attached to the sensor in three dimensional space. These values usually in angles, thus help us to determine its attitude. Thus, they are used in smart phones to detect its orientation. And also in wearable gadgets like the nike fuel band or fit bit, which use IMU sensors to track movement.

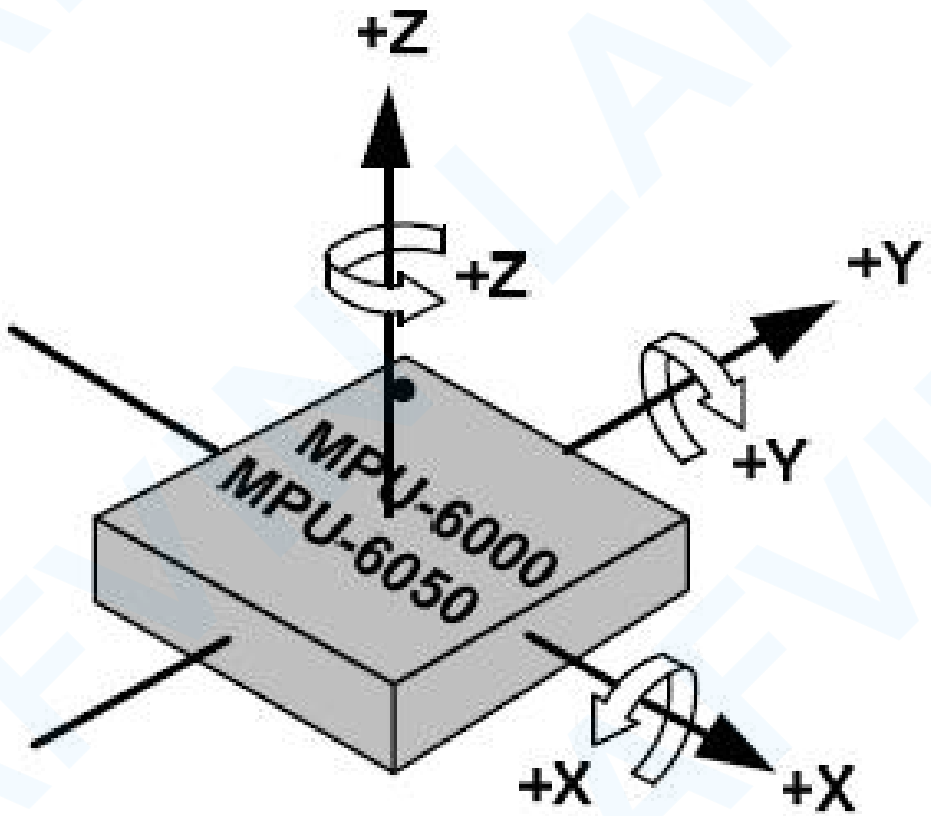
How does it work?

IMU sensors usually consists of two or more parts. Listing them by priority, they are : accelerometer, gyroscope, magnetometer and altimeter. The GY-521 is a 6 DOF (Degrees of Freedom) or a six axis IMU sensor, which means that it gives six values as output. Three values from the accelerometer and three from the gyroscope. The GY-521 is a sensor based on MEMS (Micro Electro Mechanical Systems) technology. Both the accelerometer and the gyroscope is embedded inside a single chip. This chip uses I2C (Inter Integrated Circuit) protocol for communication

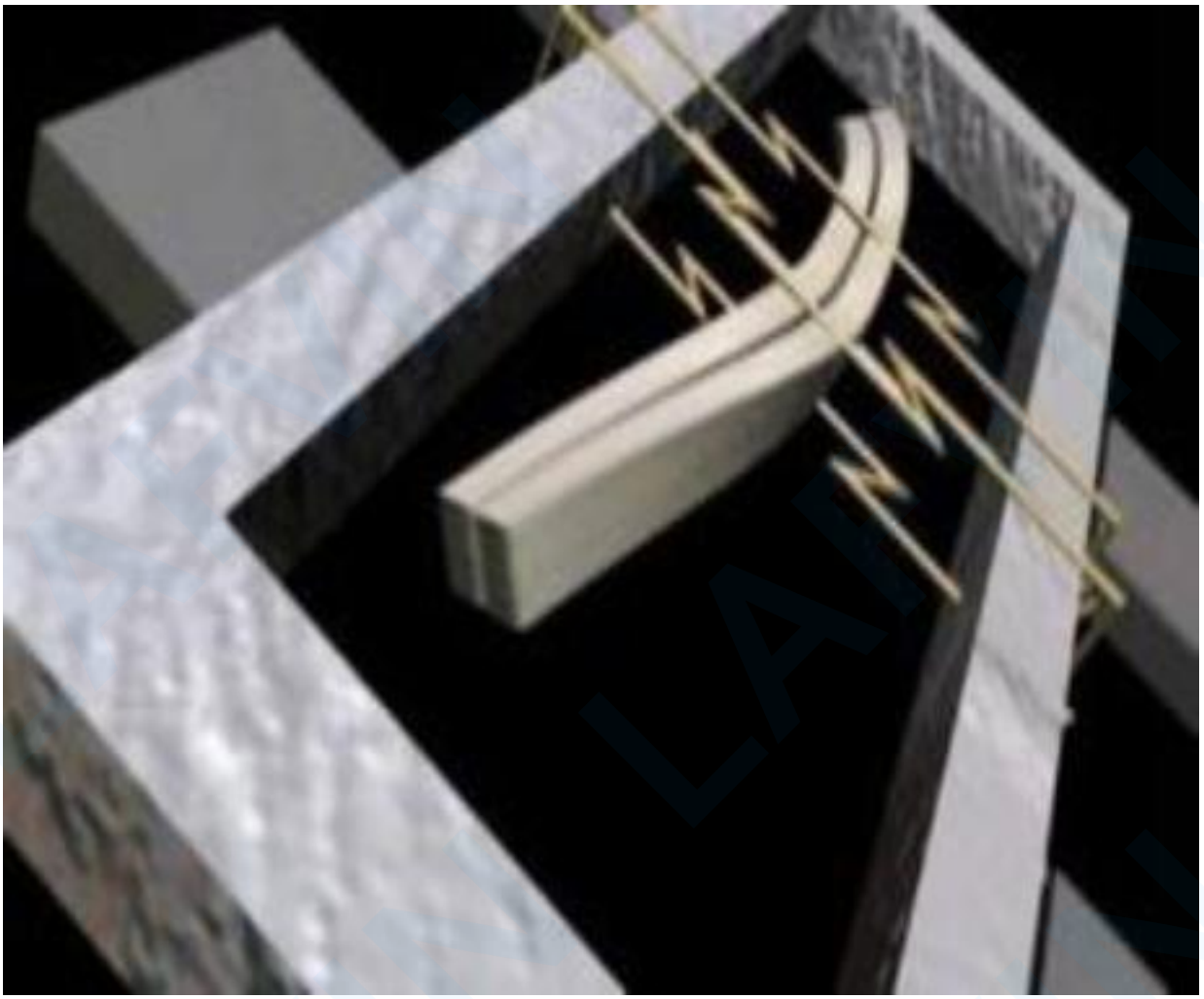
How does an accelerometer work?



An accelerometer works on the principle of piezo electric effect. Here, imagine a cuboidal box, having a small ball inside it, like in the picture above. The walls of this box are made with piezo electric crystals. Whenever you tilt the box, the ball is forced to move in the direction of the inclination, due to gravity. The wall with which the ball collides, creates tiny piezo electric currents. There are totally, three pairs of opposite walls in a cuboid. Each pair corresponds to an axis in 3D space: X, Y and Z axes. Depending on the current produced from the piezo electric walls, we can determine the direction of inclination and its magnitude. For more information check this.

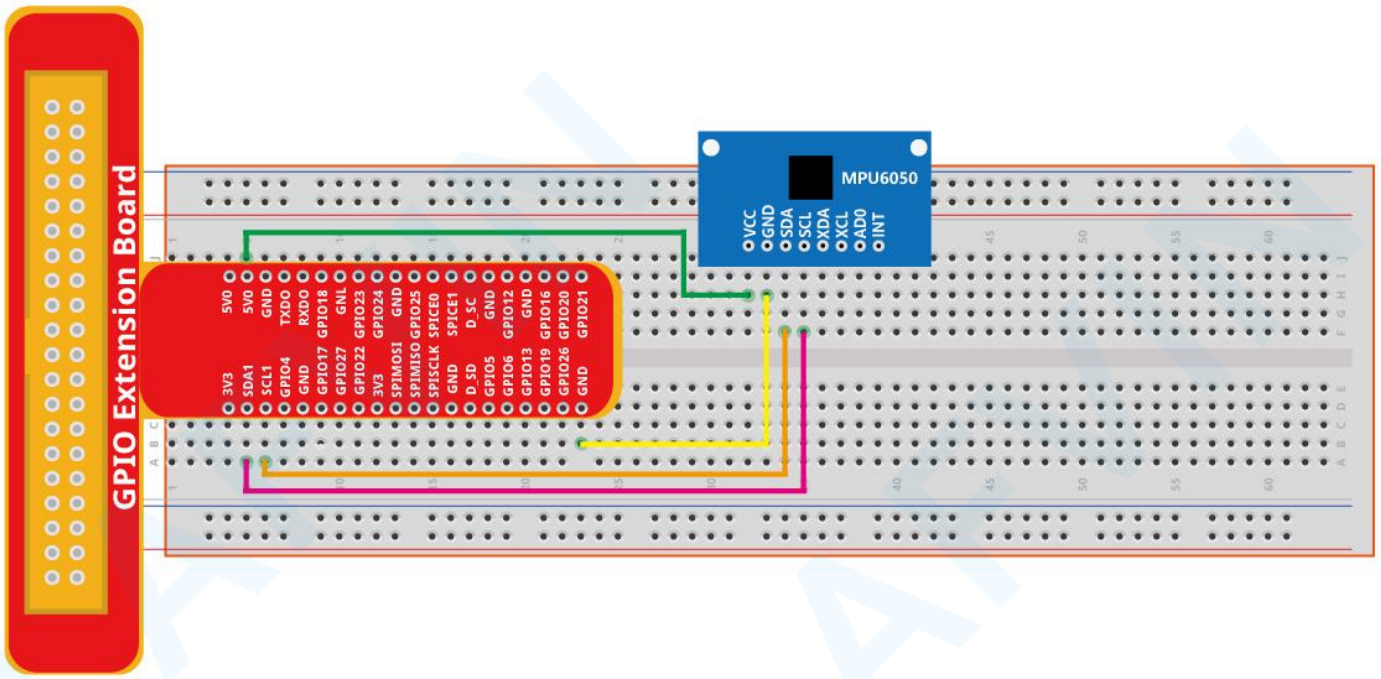


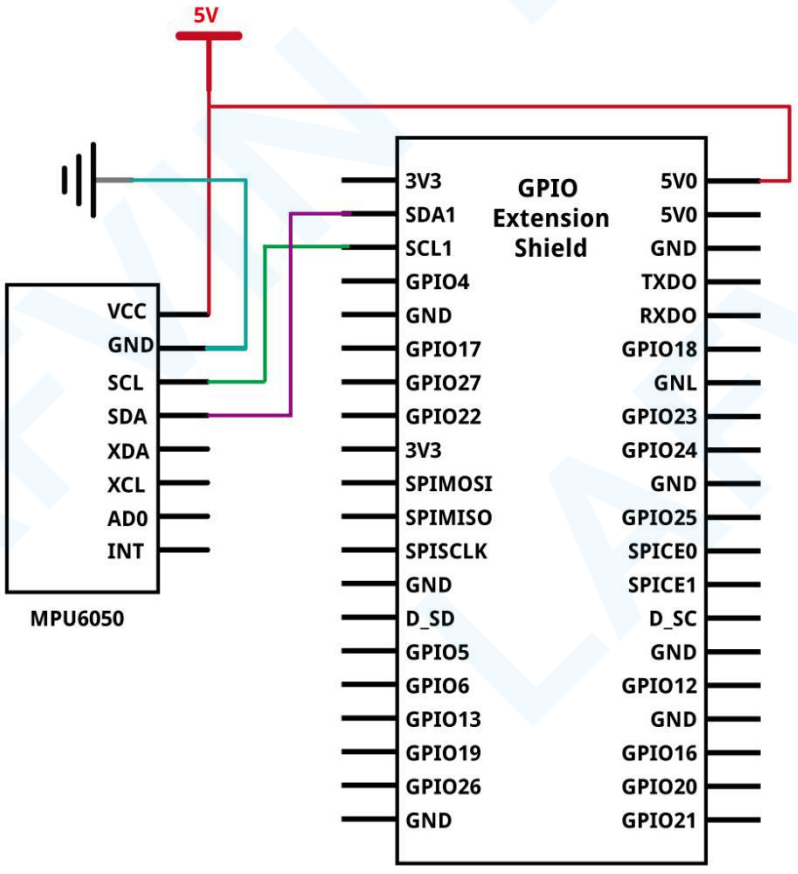
How does a gyroscope work?



Gyroscopes work on the principle of Coriolis acceleration. Imagine that there is a fork like structure, which is in constant back and forth motion. It is held in place using piezo electric crystals. Whenever, you try to tilt this arrangement, the crystals experience a force in the direction of inclination. This is caused as a result of the inertia of the moving fork. The crystals thus produce a current in consensus with the piezo electric effect, and this current is amplified. The values are then refined by the host microcontroller.

Wiring diagram





Test instructions

C_Code_15_MPU6050

First, observe the project result, then analyze the code.

9. Use cd command to enter 15_MPU6050 directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/15_MPU6050
```

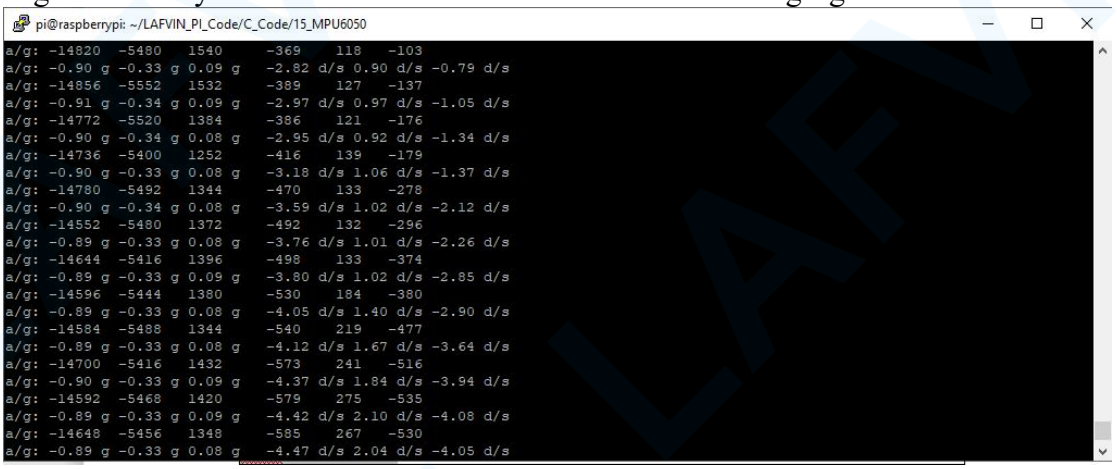
2. Use the following command to compile the code "MPU6050.c " and generate executable file "15_MPU6050.c "

```
gcc 15_MPU6050.cpp MPU6050.cpp I2Cdev.cpp  –o MPU6050
```

3. Then run the generated file "**MPU6050**".

```
sudo ./MPU6050
```

After the program is executed, the terminal will display the original acceleration and gyroscope data of MPU6050, as well as the conversion to gravity acceleration and angular velocity as the unit of data. As shown in the following figure:



Code explanation

Class MPU6050

This is a class library used to operate MPU6050, which can directly read and set MPU6050. Here are some
member functions:
MPU6050 ()
//Constructor. The parameter is I2C address, and the default I2C address is 0x68.
void initialize ();
/*Initialization function, used to wake up MPU6050. Range of accelerometer is ±2g and range of gyroscope is ±250 degrees/sec.*/
void getMotion6((int16_t* * ax, , int16_t* * ay, , int16_t* * az, , int16_t* * gx, , int16_t* * g gy y, , int16_t* * gz);
//Get the original data of accelerometer and gyroscope.
int16_t getTemperature ();
Get the original temperature data of MPU6050.

Python_Code_15_MPU6050

First, observe the project result, then analyze the code.

1. Use cd command to enter 15_MPU6050 directory of Python code.

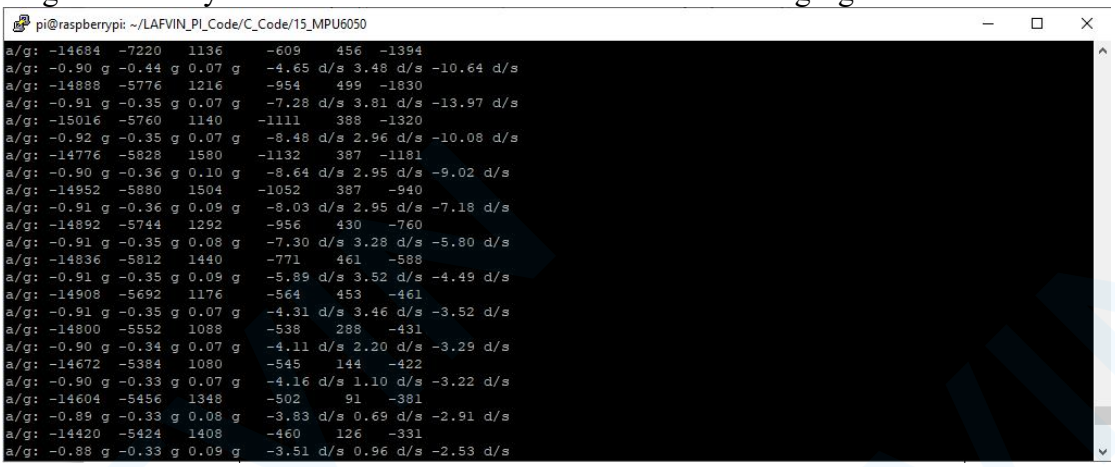
```
cd ~/LAFVIN_PI_Code/Python_Code/15_MPU6050
```

2. Use Python command to execute 15_MPU6050.py.

```
python 15_MPU6050.py
```

After the program is executed, the terminal will display the original acceleration and

gyroscope data of MPU6050, as well as the conversion to gravity acceleration and angular velocity as the unit of data. As shown in the following figure:



Code explanation

mpu = = MPU6050. .MPU6050

" " "A module "MPU6050.py" is used in the code. The module include a class used to operate MPU6050. When used, first instantiate an object." " "

Class MPU6050

This is a class library used to operate MPU6050, which can directly read and set MPU6050. Here are some member functions:

```
def _init_( ( self, a_bus= =1, a_ad dress= =C.MPU6050_DEFAULT_ADDRESS, ,
                a_xAOff== None, a_yAOff== None, a_zAOff== None,a_xGOff==None,
                a_yGOff= = None,a_zGOff= = None, a_debug= = False ):
#Constructor
def dmp_initialize( ( self ):
" " "Initialization function, used to wake up MPU6050. Range of accelerometer is ±2g and range of gyroscope is ±250 degrees/sec." " "

def get_acceleration( ( self ): & def get_rotation( ( self ):
#Get the original data of accelerometer and gyroscope.
```

Lesson 16 LED Matrix

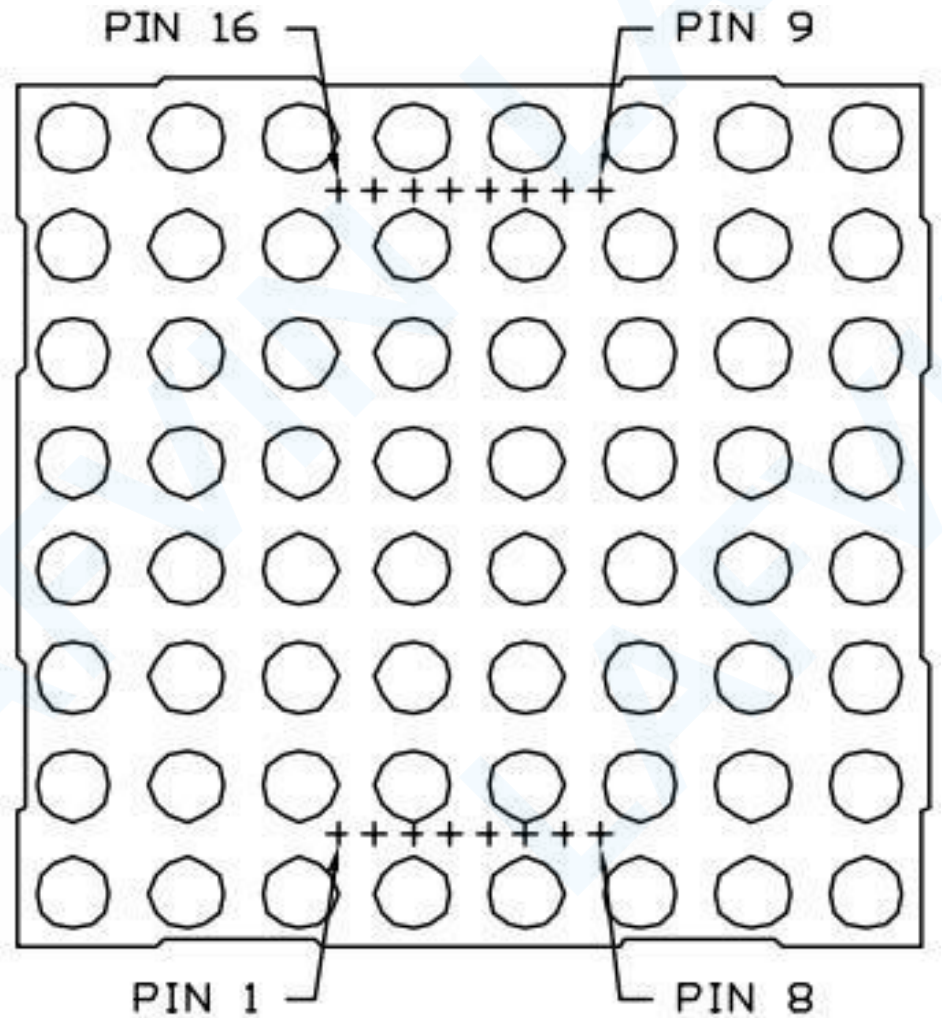
About this lesson:

In this lesson, you will learn how to wire up and use an alphanumeric LCD display. The display has an LED backlight and can display two rows with up to 16 characters on each row. You can see the rectangles for each character on the display and the pixels that make up each character. The display is just white on blue and is intended for showing text.

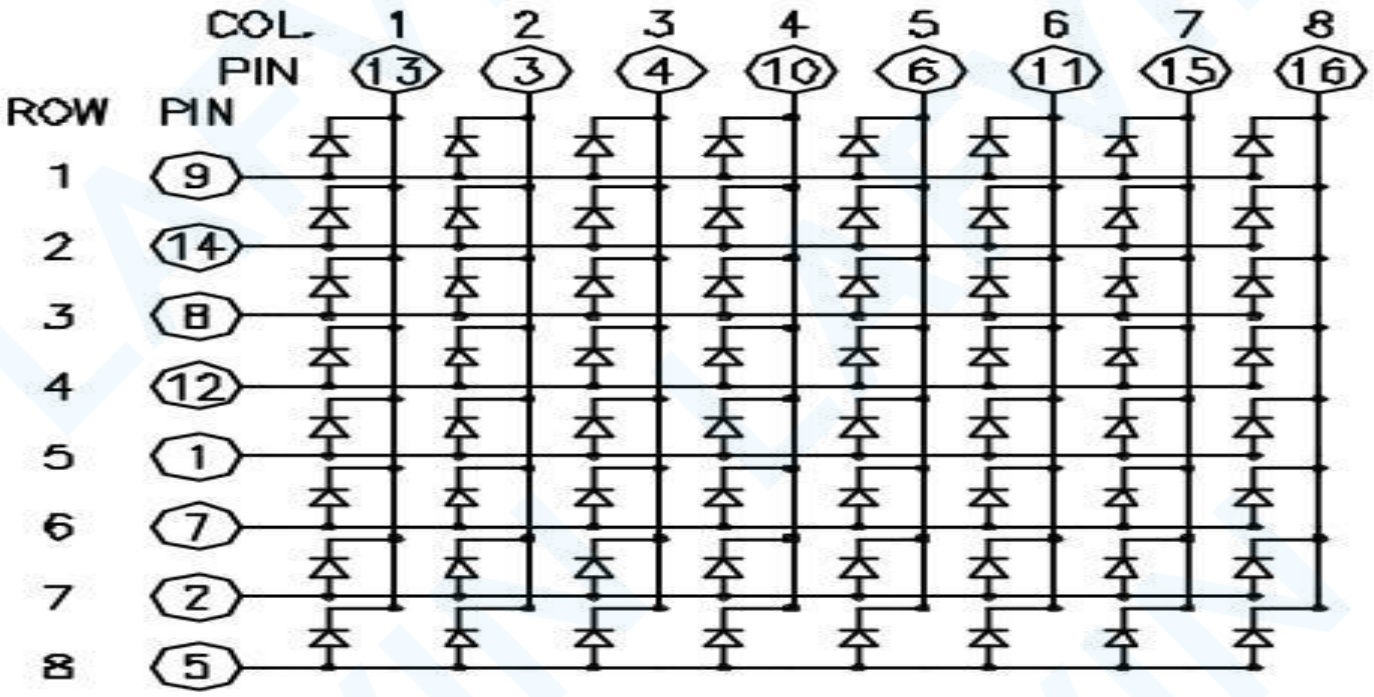
Introduction

With low-voltage scanning, LED dot-matrix displays have advantages such as power saving, long service life, low cost, high brightness, wide angle of view, long visual range, waterproof, and numerous specifications. LED dot-matrix displays can meet the needs of different applications and thus have a broad development prospect. This time, we will conduct an LED dot-matrix experiment to experience its charm firsthand.

The external view of a dot-matrix is shown as follows:

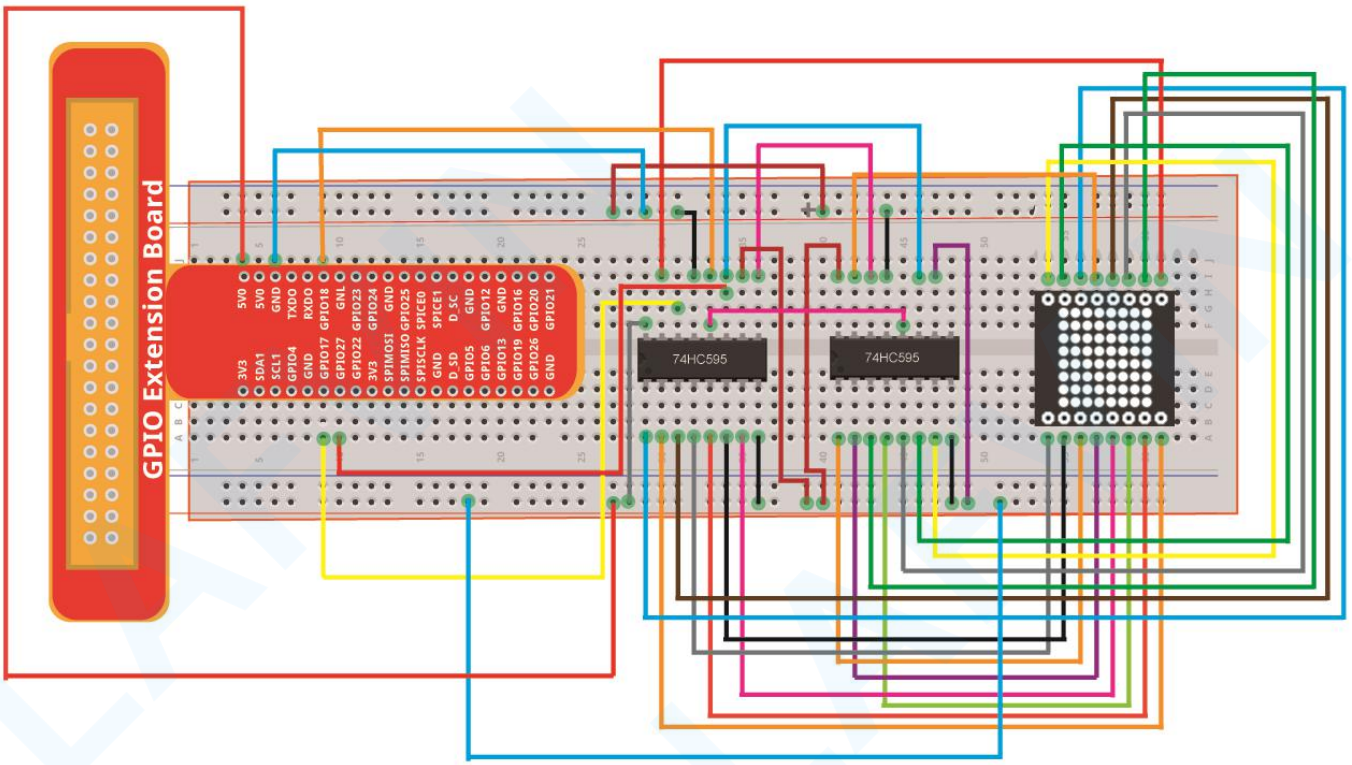


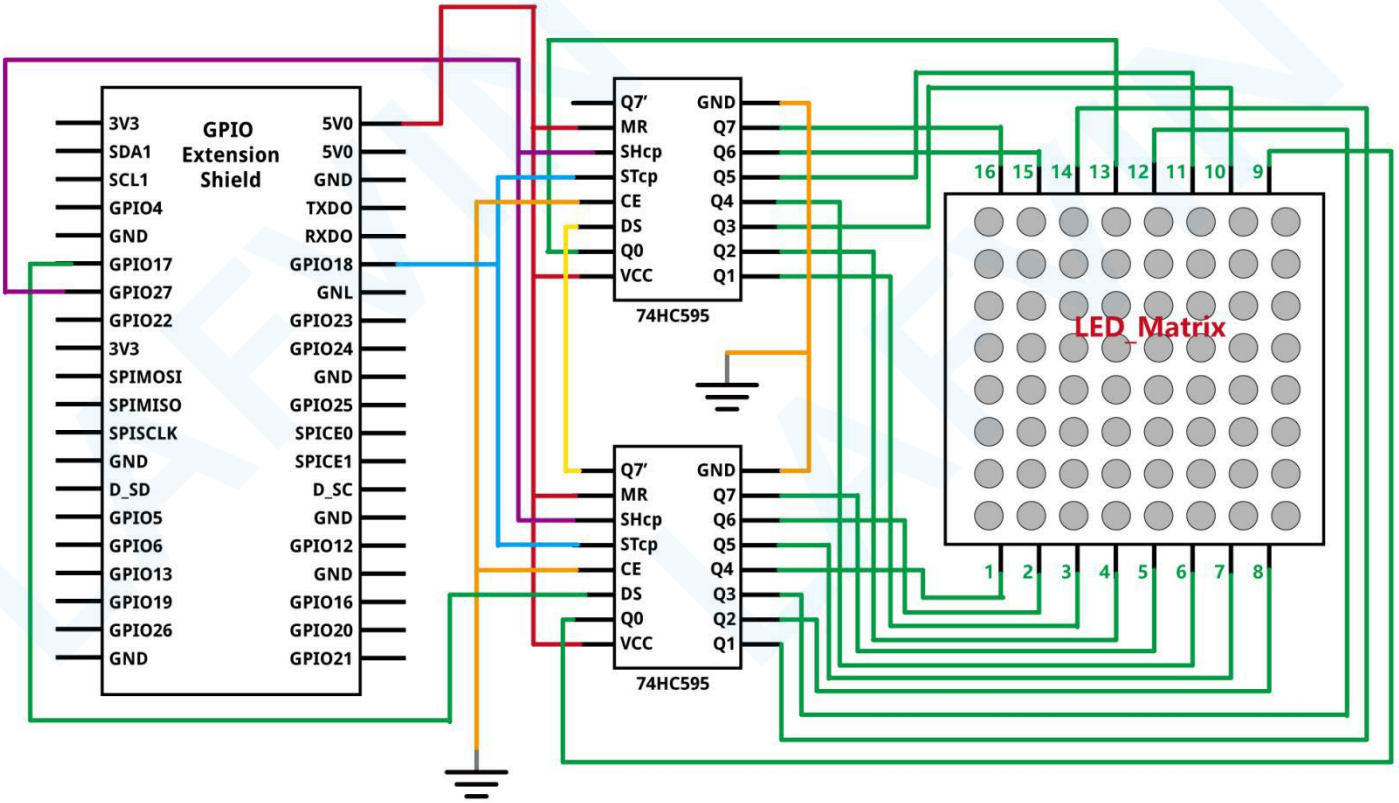
The display principle of the 8*8 dot-matrix:
The 8*8 dot-matrix is made up of sixty-four LEDs, and each LED is placed at the cross point of a row and a column. When the electrical level of a certain row is 1 and the electrical level of a certain column is 0, the corresponding LED will light up. If you want to light the LED on the first dot, you should set pin 9 to high level and pin 13 to low level. If you want to light LEDs on the first row, you should set pin 9 to high level and pins 13, 3, 4, 10, 6, 11, 15 and 16 to low level. If
you want to light the LEDs on the first column, set pin 13 to low level and pins 9, 14, 8, 12, 1, 7, 2 and 5 to high level.
The internal view of a dot-matrix is shown as follows:



The principle of 74HC595 has been previously illustrated. One chip is used to control the rows of the dot-matrix while the other chip is used to control the columns.

Wiring diagram





Test instructions

C_Code_16_LED_Matrix

First, observe the project result, then analyze the code.

10. Use cd command to enter 8_LCD1602 directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/16_LED_Matrix
```

2. Use the following command to compile the code “LED_Matrix.c ” and generate executable file “LED_Matrix.c ”

```
gcc LED_Matrix.c -o LED_Matrix -lwiringPi
```

3. Then run the generated file “LED_Matrix”.

```
sudo ./LED_Matrix
```

After completing the upload of the program, you will see the shape of the LED Matrix display in the code.

Code explanation

```
void hc595_out(){}
/*hc595_out(); // Update the output of the 74HC595; output the data controlled by both two HC595, and the dot matrix will show the pattern.*/

void hc595_in(unsigned char dat){}
// Write an 8-bit data to the shift register of the 74HC595
```

Python_Code_16_LED_Matrix

First, observe the project result, then analyze the code.

1. Use cd command to enter 16_LED_Matrix directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/16_LED_Matrix
```

2. Use Python command to execute LED_Matrix.py.

```
python LED_Matrix.py
```

After completing the upload of the program, you will see the shape of the LED Matrix display in the code.

Code explanation

```
def get_matrix(row_buffer, col_buffer, max_row=8, max_col=8):
" " "The functions is to
print the pattern on the matrix by the 2D array on the command line interface (CLI)." " "

def hc595_shift(dat):
# Shift the data to 74HC595
```

Lesson 17 IR Control

About this lesson:

In this lesson, we learn how to read the key value of the infrared remote control through the Raspberry Pi.

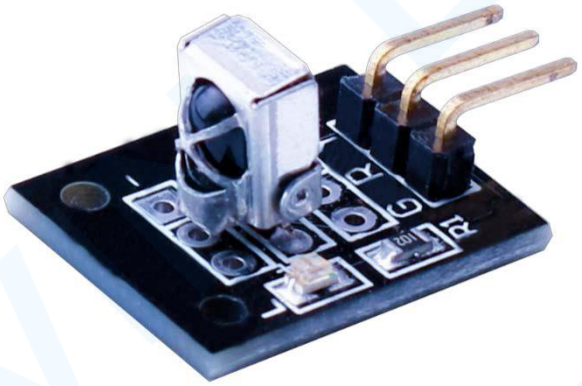
Introduction

IR RECEIVER SENSOR:

IR detectors are little microchips with a photocell that are tuned to listen to infrared light. They are almost always used for remote control detection - every TV and DVD player has one of these in the front to listen for the IR signal from the clicker. Inside the remote control is a matching IR LED, which emits IR pulses to tell the TV to turn on, off or change channels. IR light is not visible to the human eye, which means it takes a little more work to test a setup.

There are a few difference between these and say a CdS Photocells:

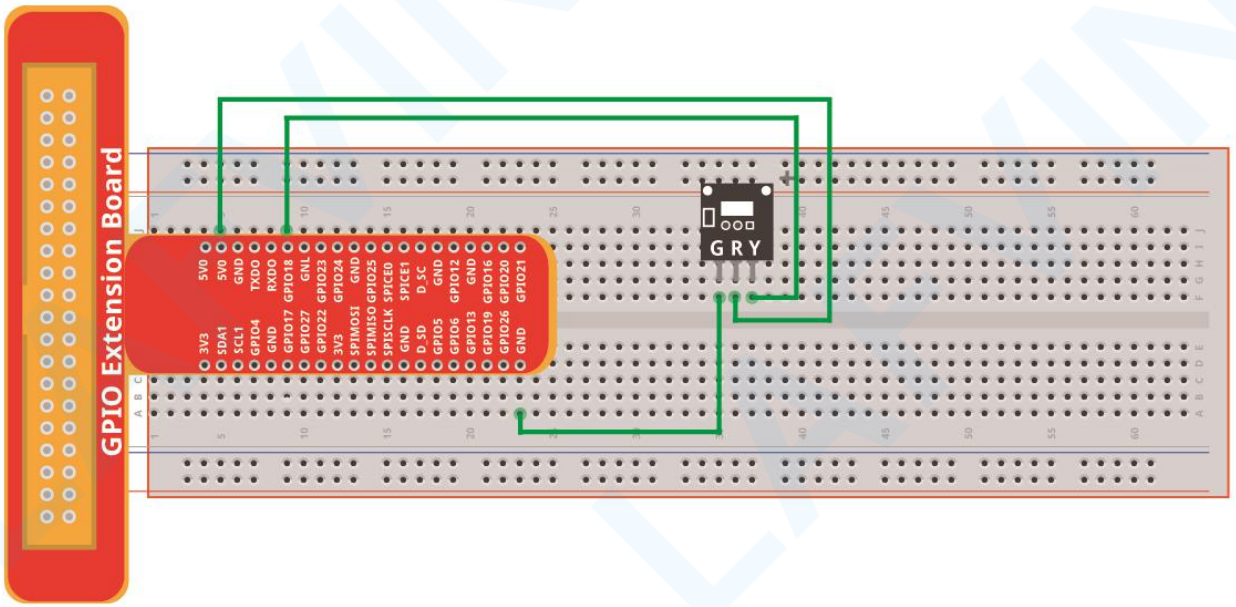
IR detectors are specially filtered for IR light, they are not good at detecting visible light. On the other hand, photocells are good at detecting yellow/green visible light, and are not good at IR light.

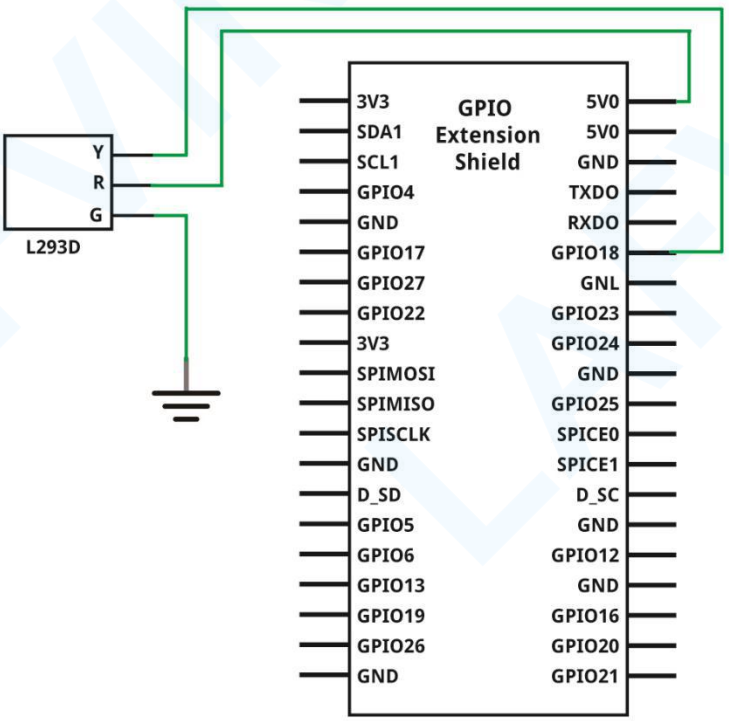


IR detectors have a demodulator inside that looks for modulated IR at 38 KHz. Just shining an IR LED won't be detected, it has to be PWM blinking at 38KHz. Photocells do not have any sort of demodulator and can detect any frequency (including DC) within the response speed of the photocell (which is about 1KHz)

IR detectors are digital out - either they detect 38KHz IR signal and output low (0V) or they do not detect any and output high (5V). Photocells act like resistors, the resistance changes depending on how much light they are exposed to.

Wiring diagram





Test instructions

C_Code_17_IR_Control_Matrix

First, observe the project result, then analyze the code.

11. Use cd command to enter 17_IR_Control directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/17_IR_Control
```

2. Use the following command to compile the code “LED_Matrix.c ” and generate executable file “IR_Control.c ”

```
gcc IR_Control.c -o IR_Control -lwiringPi
```

3. Then run the generated file “LED_Matrix”.

```
sudo ./IR_Control
```

After completing the upload of the program, you will see the shape of the LED Matrix display in the code.

```
IRM Test Program ...
Get the key: 0x19
Get the key: 0x19
Get the key: 0x16
Get the key: 0x16
Get the key: 0x16
Get the key: 0x5e
Get the key: 0x0c
Get the key: 0x5a
Get the key: 0x08
Get the key: 0x1c
Get the key: 0x16
Get the key: 0x19
Get the key: 0x16
Get the key: 0x0d
```

Code explanation

```
#define IO digitalRead(PIN)
/*Define the pin that reads the infrared signal*/

if (count > 25)data[idx] |= (1<<cnt);
// Calculate the value of the button by counting the time of the pulse
```

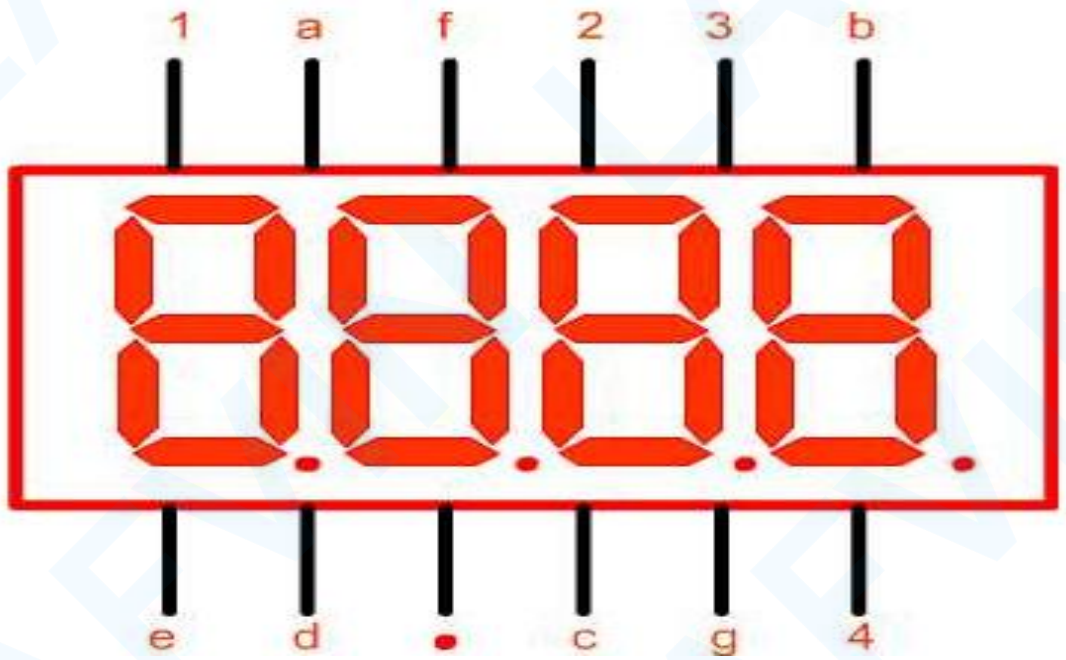
Lesson 18 4_digit_LED_Segment

About this lesson:

In this lesson, you will learn how to use a 4-digit 7-segment display.

Introduction

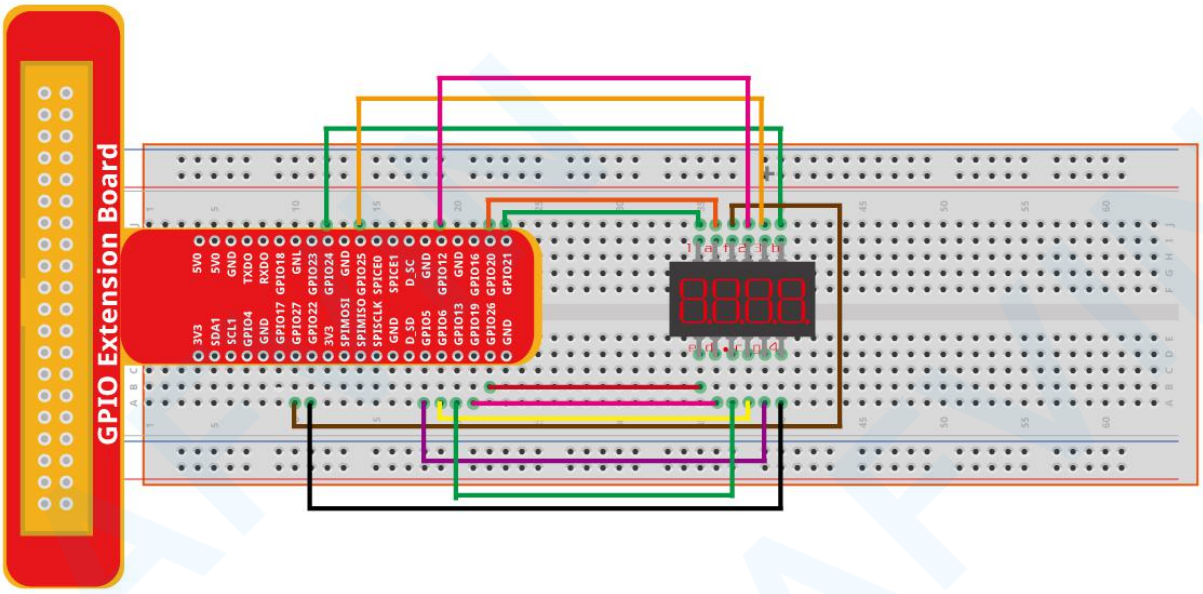
According to the connection mode of the LED unit, it can be divided into a common anode digital tube and a common cathode digital tube. The common anode digital tube refers to the digital tube that connects all the anodes of the light-emitting diodes together to form a common anode (COM). The common anode digital tube should be connected to the common pole COM to +5V when applied, when the cathode of a field LED When it is low, the corresponding field is lit. When the cathode of a field is high, the corresponding field is not lit. The common cathode digital tube refers to a digital tube that connects all the cathodes of the light-emitting diodes together to form a common cathode (COM). When the common cathode digital tube is applied, the common pole COM should be connected to the ground line GND, when a field LED is used. When the anode is high, the corresponding field is lit. When the anode of a field is low, the corresponding field is not lit.

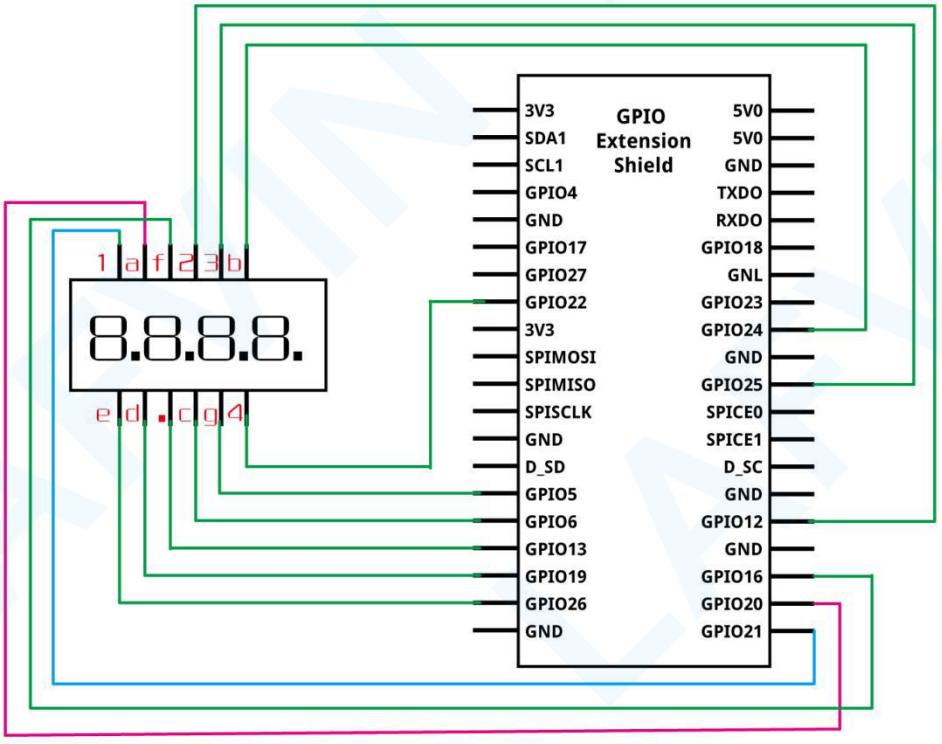


Bit selection: 1, 2, 3, and 4 are selected in the pin diagram of the above digital tube. Segment selection is to choose which digital tube to drive, such as the first and second digits.

Segment selection: In the above digital tube pin diagram, a, b, c...f are segment selection. The segment selection is to choose which segment of the digital tube to drive, such as a, b, c segment digital tube.

Wiring diagram





Test instructions

C_Code_18_4_digit_LED_Segment

First, observe the project result, then analyze the code.

12. Use cd command to enter 18_4_digit_LED_Segment directory of C code.

```
cd ~/LAFVIN_PI_Code/C_Code/18_4_digit_LED_Segment
```

2. Use the following command to compile the code "4_digit_LED_Segment.c " and generate executable file "4_digit_LED_Segment.c "

```
gcc 4_digit_LED_Segment.c -o 4_digit_LED_Segment -lwiringPi
```

3. Then run the generated file "LED_Matrix".

```
sudo ./4_digit_LED_Segment
```

Connect the circuit correctly, after downloading the program, you will see that the number displayed by the four digital tubes is increasing, similar to a chronograph

Code explanation

```
Display(unsigned char x, unsigned char Number)
// take x as coordinate and display number

void WeiXuan(unsigned char n)
/*Bit selection: 1, 2, 3, and 4 are selected in the pin diagram of the above digital tube. Segment selection is to choose which digital tube to drive, such as the first and second digits.*/

void Num_0(){}
void Num_9(){}
//Define a subfunction that displays each digit
```

Python_Code_18_4_digit_LED_Segment

First, observe the project result, then analyze the code.

1. Use cd command to enter 18_4_digit_LED_Segment directory of Python code.

```
cd ~/LAFVIN_PI_Code/Python_Code/18_4_digit_LED_Segment
```

2. Use Python command to execute IR_Control.py.

```
python 4_digit_LED_Segment.py
```

After completing the circuit connection, download the program, the number displayed by the four digits is increased from 1 to 9, from 1111 to 9999.

Code explanation

```
def digitalWriteByte(val):
    GPIO.output(38, val & (0x01 << 0))
    GPIO.output(18, val & (0x01 << 1))
```

```
    GPIO.output(31, val & (0x01 << 2))
    GPIO.output(35, val & (0x01 << 3))
    GPIO.output(37, val & (0x01 << 4))
    GPIO.output(36, val & (0x01 << 5))
    GPIO.output(29, val & (0x01 << 6))
    GPIO.output(33,    val & (0x01 << 7))
" " "Define a function that displays a single digit, and display the value of the parameter val. In fact, it is based on the binary of val to determine which segment of the digital tube is on or off." " "

def display_3(num):
    b0 = num % 10
    b1 = num % 100 / 10
    b2 = num % 1000 / 100
    b3 = num / 1000
" " "The four digits to be displayed are decomposed in decimal, and the digits displayed by ones, tens, hundreds, and thousands are displayed separately." " "
```