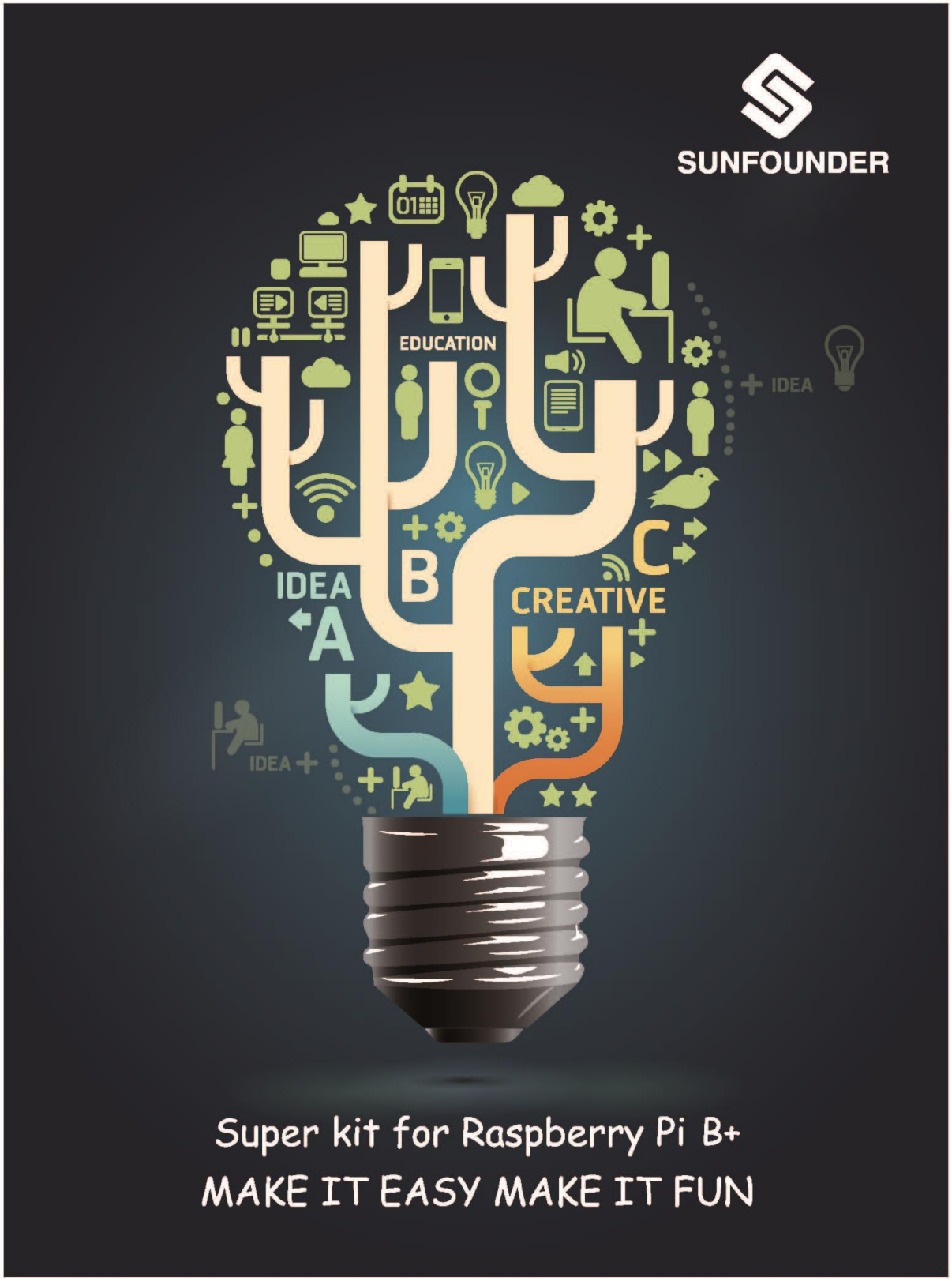

SUNFOUNDER
EDUCATION
IDEA
IDEA
B
C
IDEA
A
CREATIVE
IDEA
Super kit for Raspberry Pi B+
MAKE IT EASY MAKE IT FUN

Components List

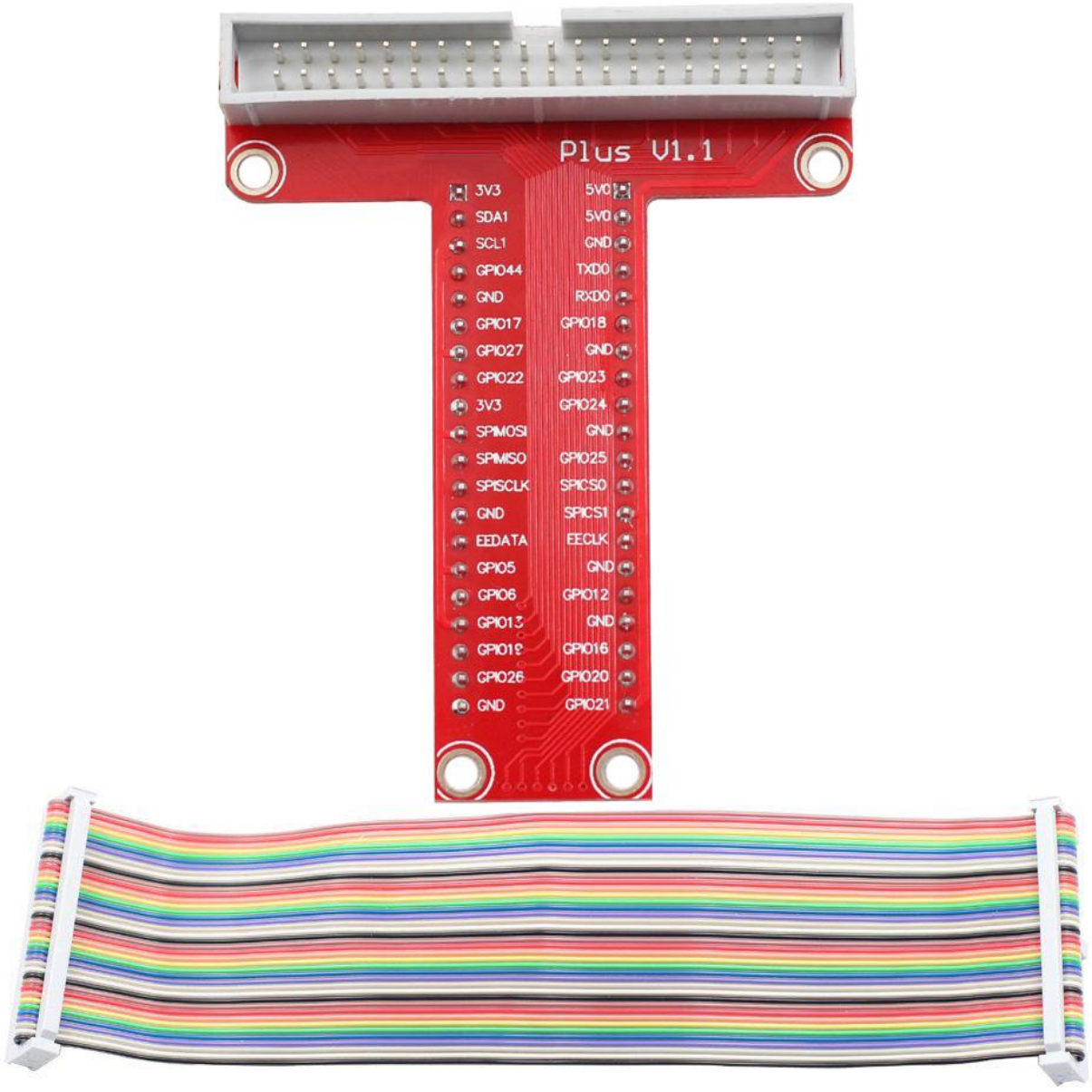
1*Project Box
40*Pin Header
1*Timer555
2*Optocoupler(4N35)
2*74HC595(Shift Register)
1*L293D(H-Bridge L293D)
1*Active Buzzer
1*RGB Led
1*ADXL345 (accelerometer)
1*rotary encoder
5*button
8*resistor(220Ω)
8*resistor(1K)
4*resistor(10K)
4*resistor(100K)
1*resistor(1M)
1*resistor(5.1M)
1*switch
1*trim pot 50K
1*Power Supply Module
1*LCD1602
1*dotMatrix Display(8*8)
2*segment (7-segment display)
1*DC motor
16*Led(red)
2*Led(white)
2*Led(Green)
2*Led(Yellow)
2*Transistor(PNP)
2*Transistor(NPN)
4*Capacitor Ceramic 100nF
4*Capacitor Ceramic 10nF
4*Diode Rectifier

1*Breadboard 830-point
20*dupont wire male to female
65*jumper wire male to male
1*Fan
1*40-Pin GPIO Extension Board

Content

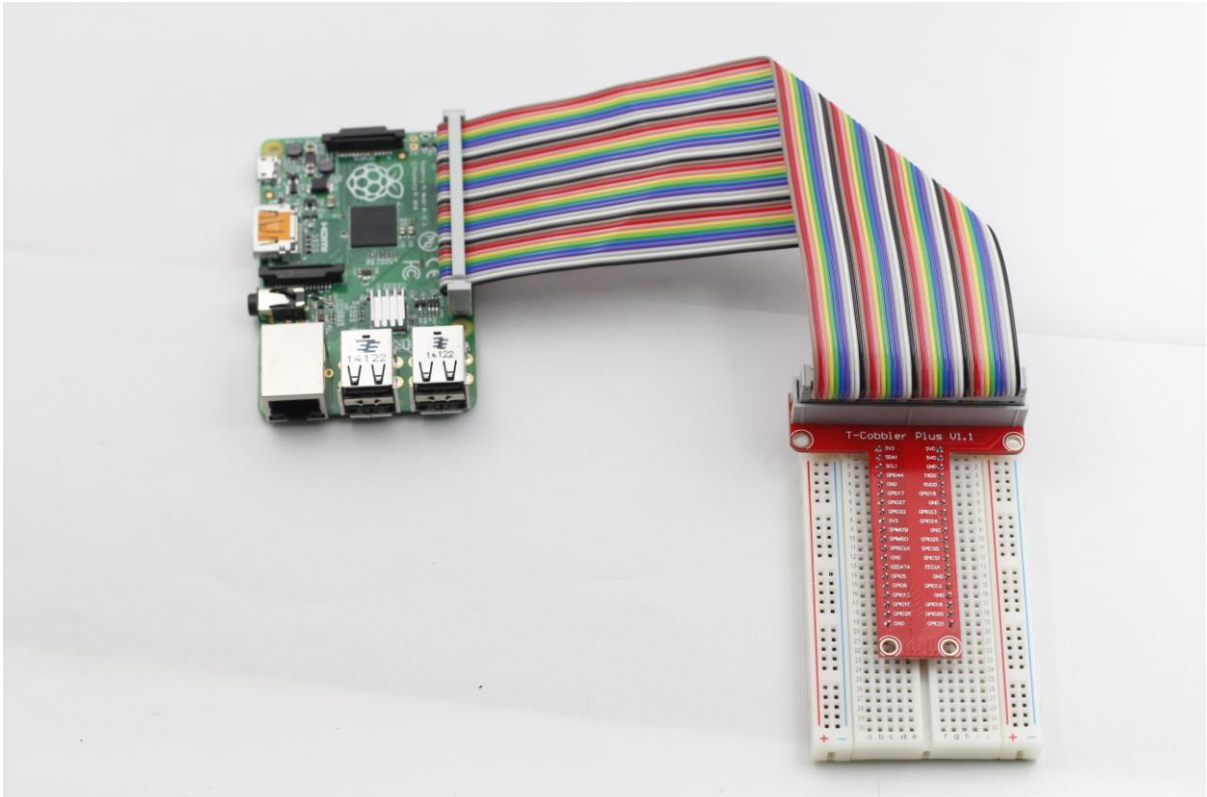
Notice:

This is our GPIO Extension Board and GPIO cable.



You must connect the GPIO cable to the Raspberry Pi B+ like this:

After connection, as shown below:



SUNFOUNDER's Raspberry Pi Lesson 1 — Blinking LED

Introduction

In this lesson, we will learn how to program your Raspberry Pi to make an LED blink. You can play numerous tricks with an LED as long as your imagination is rich enough. Now follow me to learn, and you will enjoy the fun of DIY at once.

Experimental Conditions

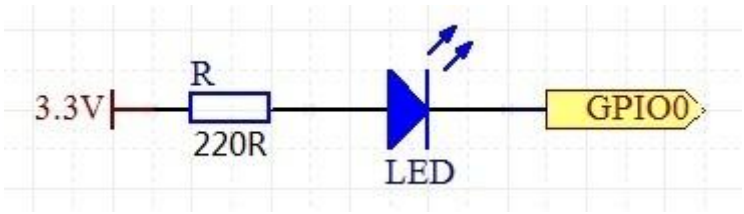
- 1*Raspberry Pi
- 1*Breadboard
- 1*Network cable (or USB wireless network adapter)
- 1*LED
- 1*Resistor (220Ω)
- Jumper wires

Experimental Principle

Semiconductor light-emitting diode is a type of component which can turn electric energy into light energy via PN junctions. According to its wavelength, semiconductor light-emitting diode can be categorized into laser diode, infrared light-emitting diode and visible light-emitting diode which, called light-emitting diode for short, is usually known as LED.

When we supply 2V-3V forward voltage to an LED, it will blink if only forward currents flow through the LED. Usually we have red, yellow, green, blue and color-changing LED which can change its colors with different voltages. LEDs are widely used due to its low operating voltage, low current, luminescent stability and small size.

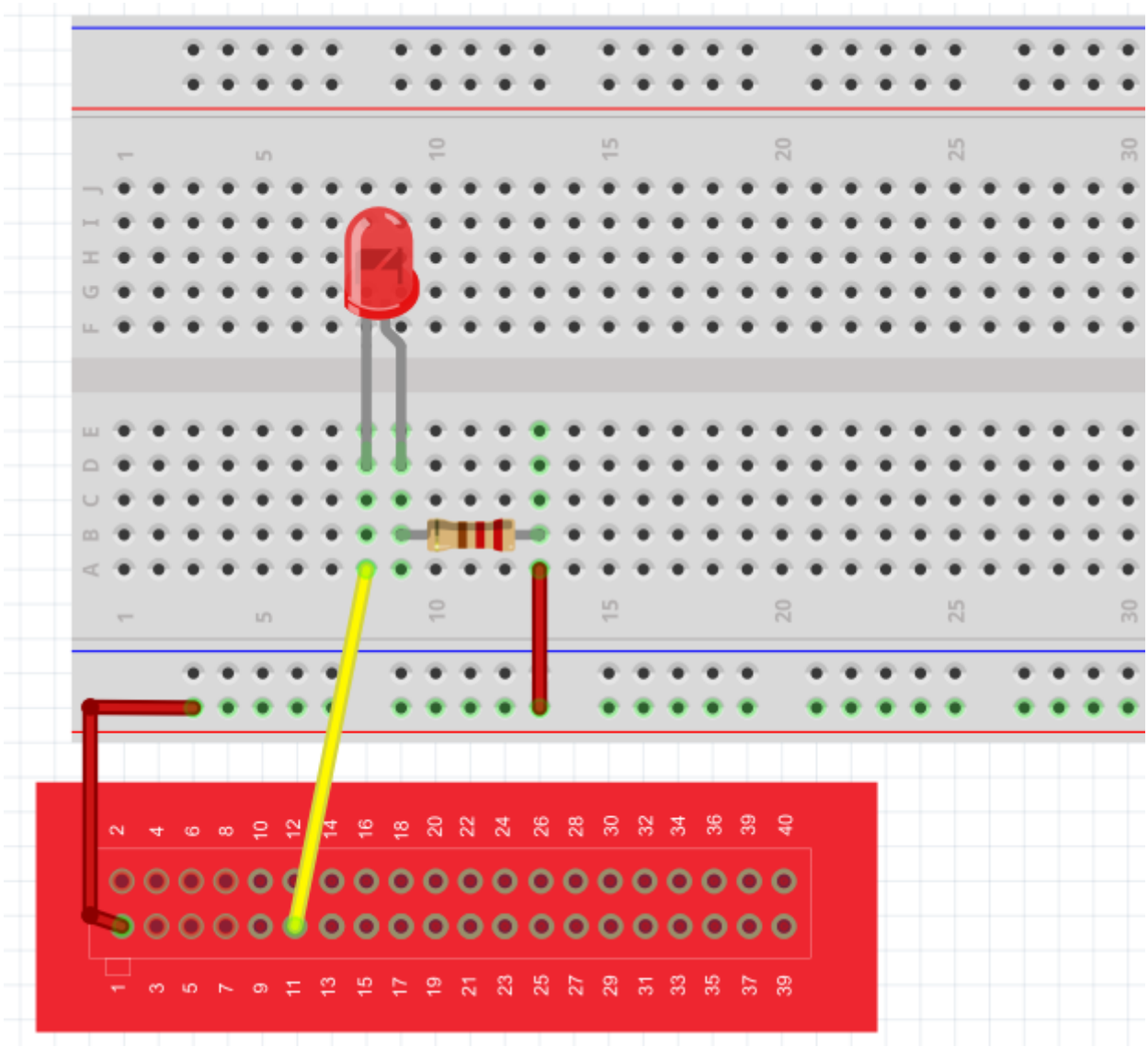
LEDs are diodes too. Hence they have a voltage drop which usually varies between 1V-3V depending on their types. Likewise, LEDs usually can emit light if supplied 5mA-30mA current, and generally we use 10mA-20mA. So when an LED is used, it is necessary to connect a current-limiting resistor to protect the LED from over-burning.



In this experiment, we will connect a 220Ω resistor to the positive pole of the LED and then connect it to 3.3 V power source, and connect the negative pole of LED to GPIO0 (See the Raspberry Pi pins distribution diagram and the above picture). If we write 1 to GPIO0, the voltage of the pin is 3.3V and the LED will not blink; if we write 0 to GIIO0, the output voltage is 0, then the LED will light up according to the above principle.

Experimental Procedure

Step 1: Connect the circuit as shown in the following diagram



Step 2: Edit and save the code with vim (see path/Rpi_SuperKit_Code/01_LED/led.c)

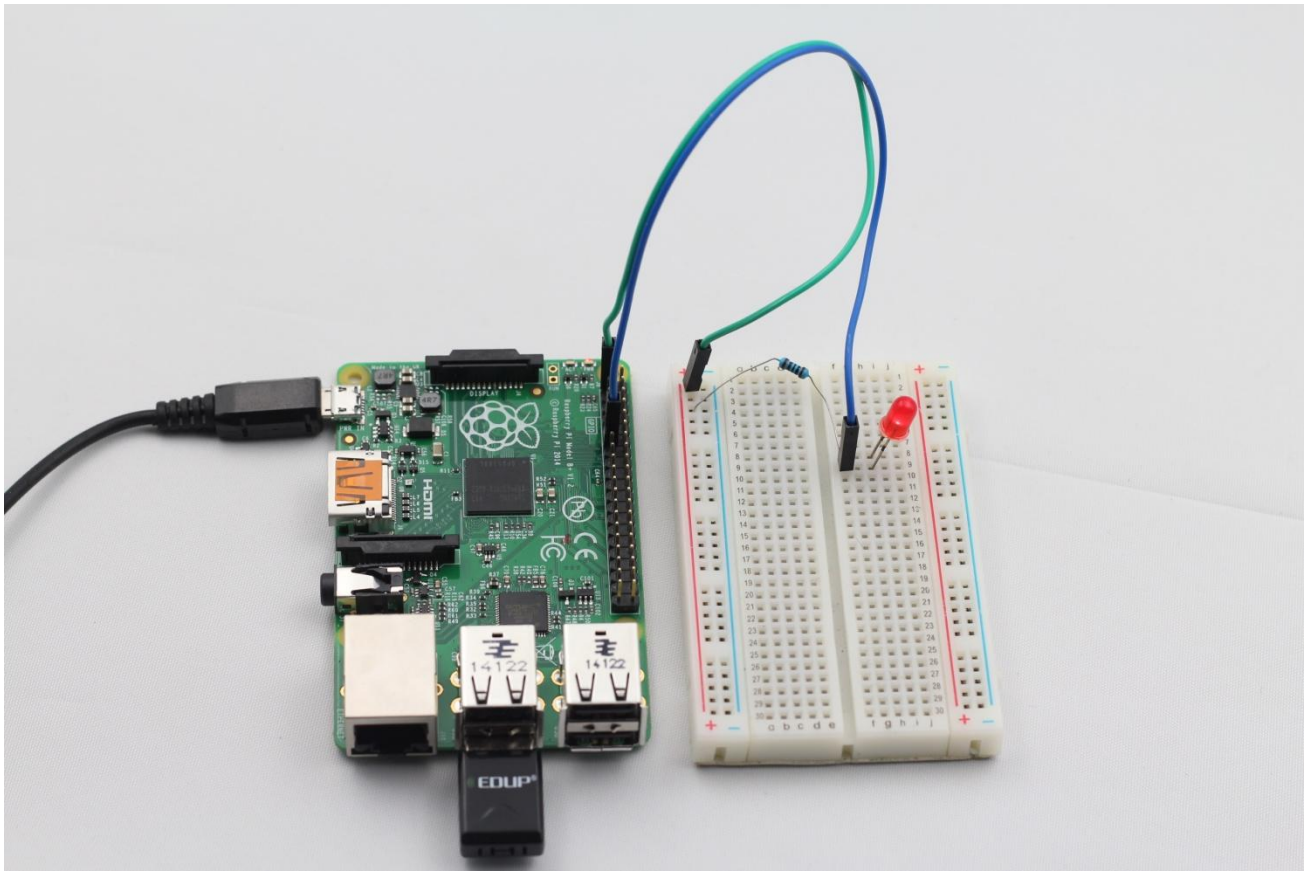
Step 3: Compile the code

```
gcc  led.c  -o  led  -lwiringPi
```

Step 4: Run the program

```
./led
```

Now, you can see your LED blinking.



Further Exploration

If you want the LED to blink faster, just change the delay time. For example, change the time to delay (200), recompile the program, and then run the program, you will see the LED blinking faster.

Summary

Raspberry Pi packages many underlying detail designs, which enable us to explore our own apps more conveniently. Maybe that is the most attractive place of Raspberry Pi.

Now you have already basically mastered the basic operation and application of Raspberry Pi GPIOs (General Purpose Input/Output). I hope you can try harder and continue to learn the next contents.

SUNFOUNDER's Raspberry Pi Lesson 2 — Controlling an LED by a Button

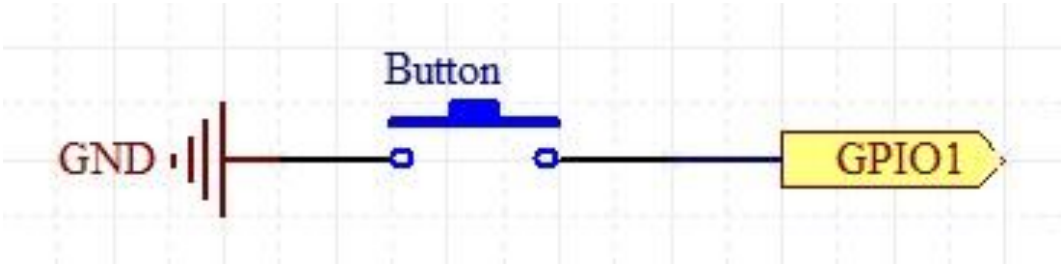
Introduction

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

Experimental Conditions

- 1*Raspberry Pi
- 1*Breadboard
- 1*Network cable (or USB wireless network adapter)
- 1*LED
- 1*Button
- 1*Resistor (220Ω)
- Jumper wires

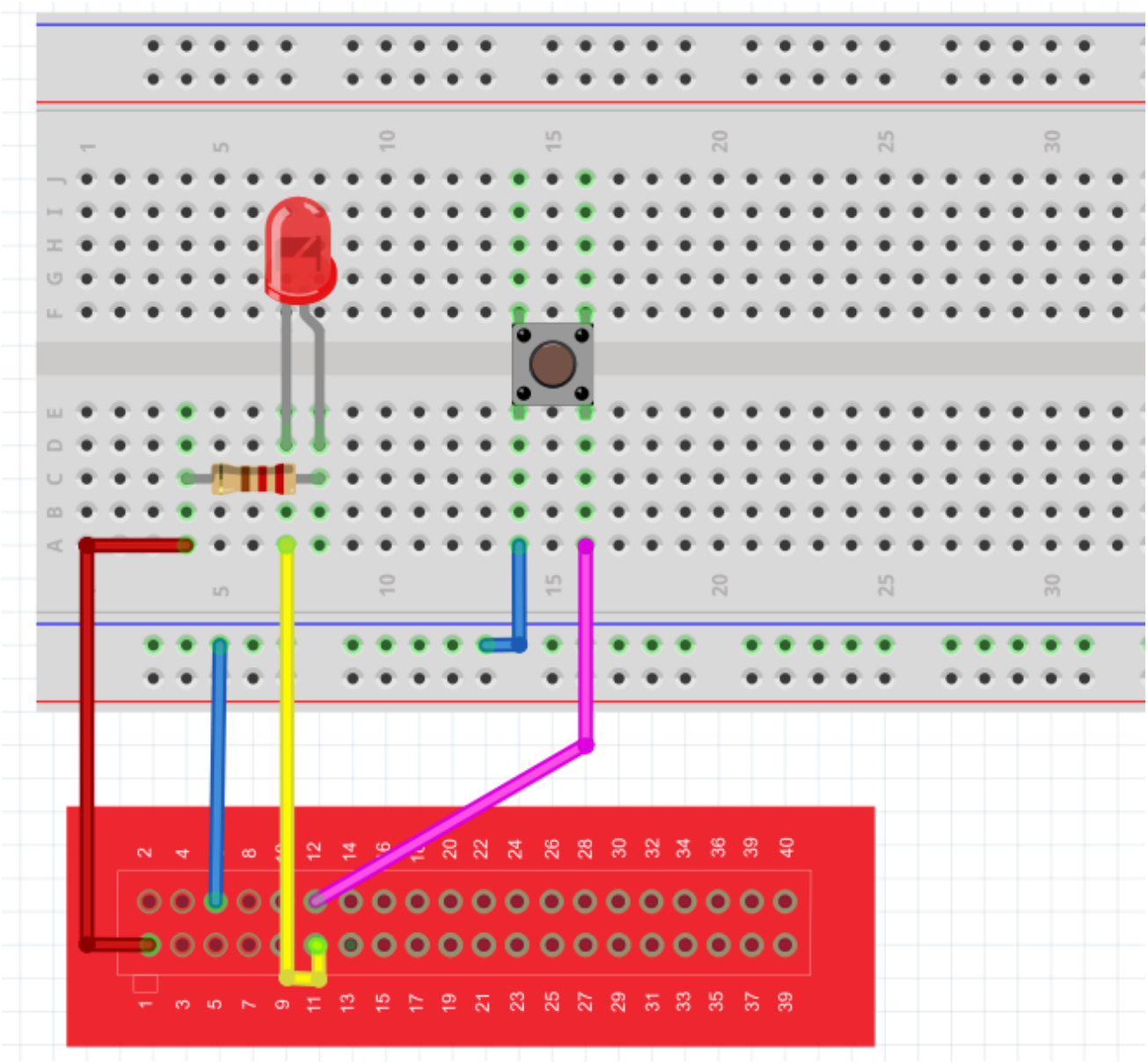
Experimental Principle



Use a normally open button as an input of Raspberry Pi, when the button is pressed, the GPIO (General Purpose Input/Output) connected to the button will turn into low level (0V). We can detect the state of the GPIO connected to the button through programming. That is, if the GPIO turns into low level, it means the button is pressed, you can run the corresponding code according to this condition. In this experiment, we make the LED light up.

Experimental Procedures

Step 1: Connect the circuit as shown in the following diagram



Step 2: Edit and save the code with vim (see path/Rpi_SuperKit_Code/02_BtnAndLed/BtnAndLed.c)

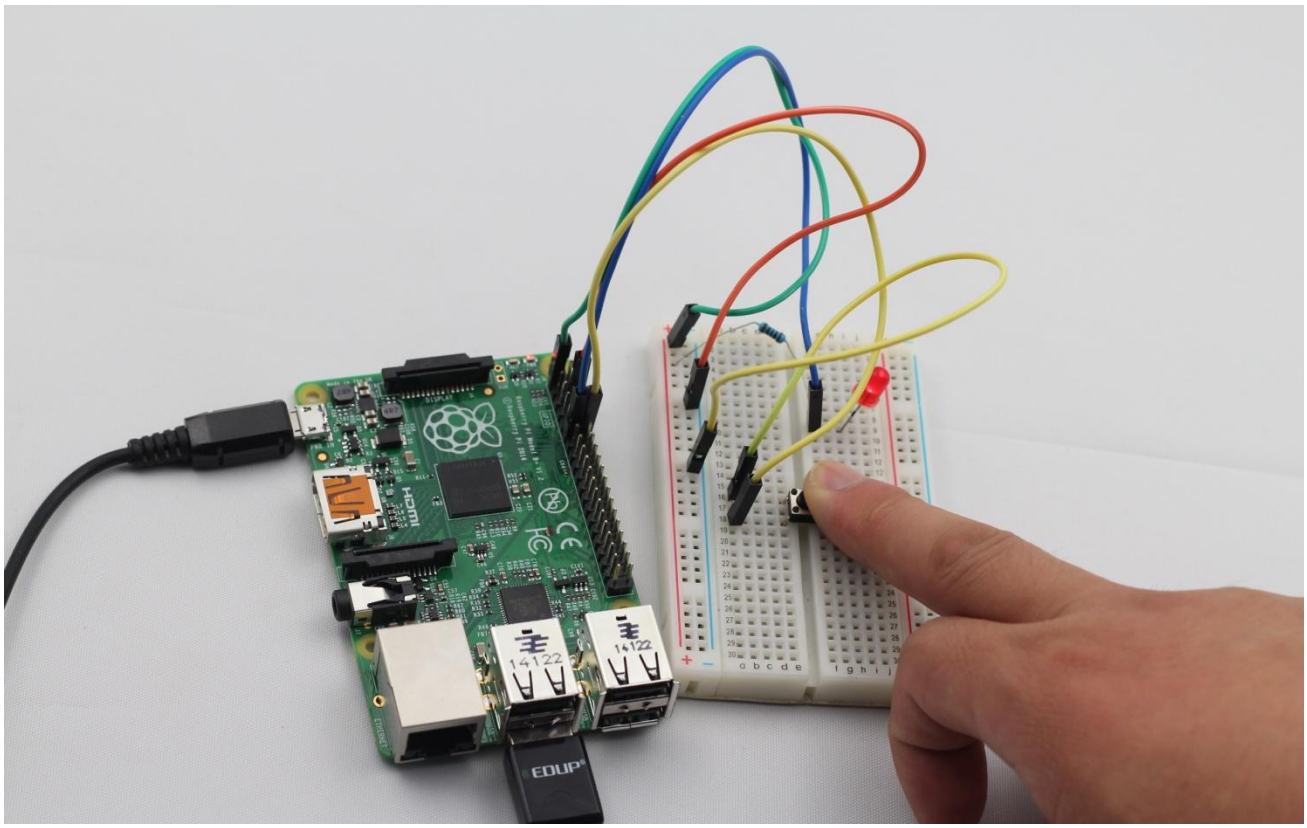
Step 3: Compile the code

gcc BtnAndLed.c -o BtnAndLed -lwiringPi

Step 4: Run the program

./BtnAndLed

Now, when you press Enter, and then press the button, the LED will light up; when you release the button, the LED will go out.



Summary

Through this experiment, you have basically mastered the Input and Output programming operation of Raspberry Pi GPIOs. I hope you can make persistent efforts and continue to learn the next contents.

SUNFOUNDER's Raspberry Pi Lesson 3 — LED Flowing Lights

Introduction

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

Experimental Conditions

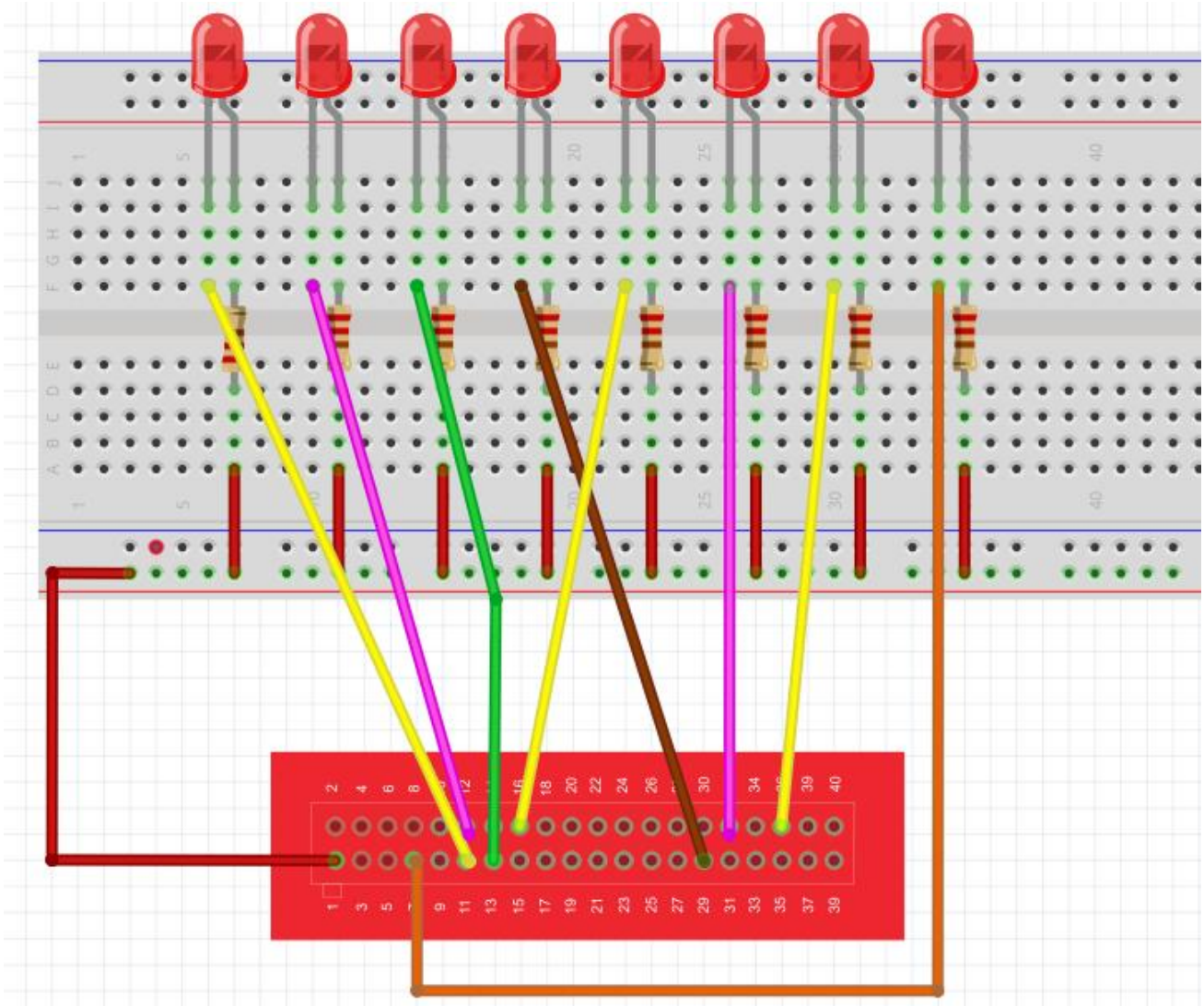
- 1*Raspberry Pi
- 1*Breadboard
- 1*Network cable (or USB wireless network adapter)
- 8*LED
- 8*Resistor (220Ω)
- Jumper wires

Experimental Principle

Set GPIO0 to GPIO7 to low in turn by programming, then LED0 to 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.

Experimental Procedures

Step 1: Connect the circuit as shown in the following diagram



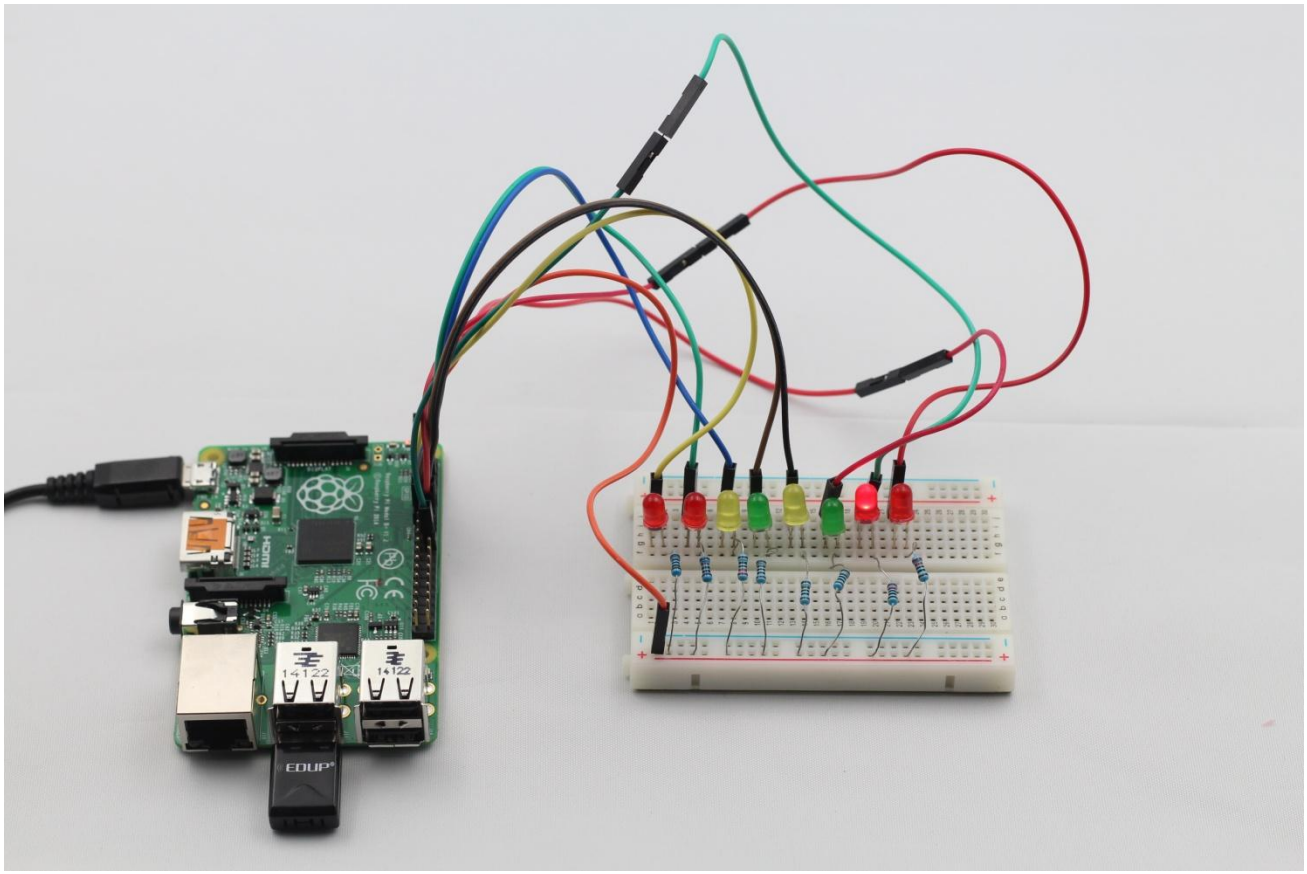
Step 2: Edit and save the code with **vim** (see path/Rpi_SuperKit_Code/03_8Led/8Led.c)
Step 3: Compile the code

```
gcc 8Led.c -o 8Led -lwiringPi
```

Step 4: Run the program

```
./8Led
```

Now, when you press Enter, eight LEDs will light up circularly and render different effects.



Further Exploration

We can write the blinking effects of LEDs in an array. If you want to use one of these effects, you can call it in the **main()** function directly.

SUNFOUNDER's Raspberry Pi Lesson 4 — Controlling LEDs by Main Function Parameters

Introduction

In this lesson, we will learn how to turn LEDs on or off by passing parameters to the **main()** function.

Experimental Conditions

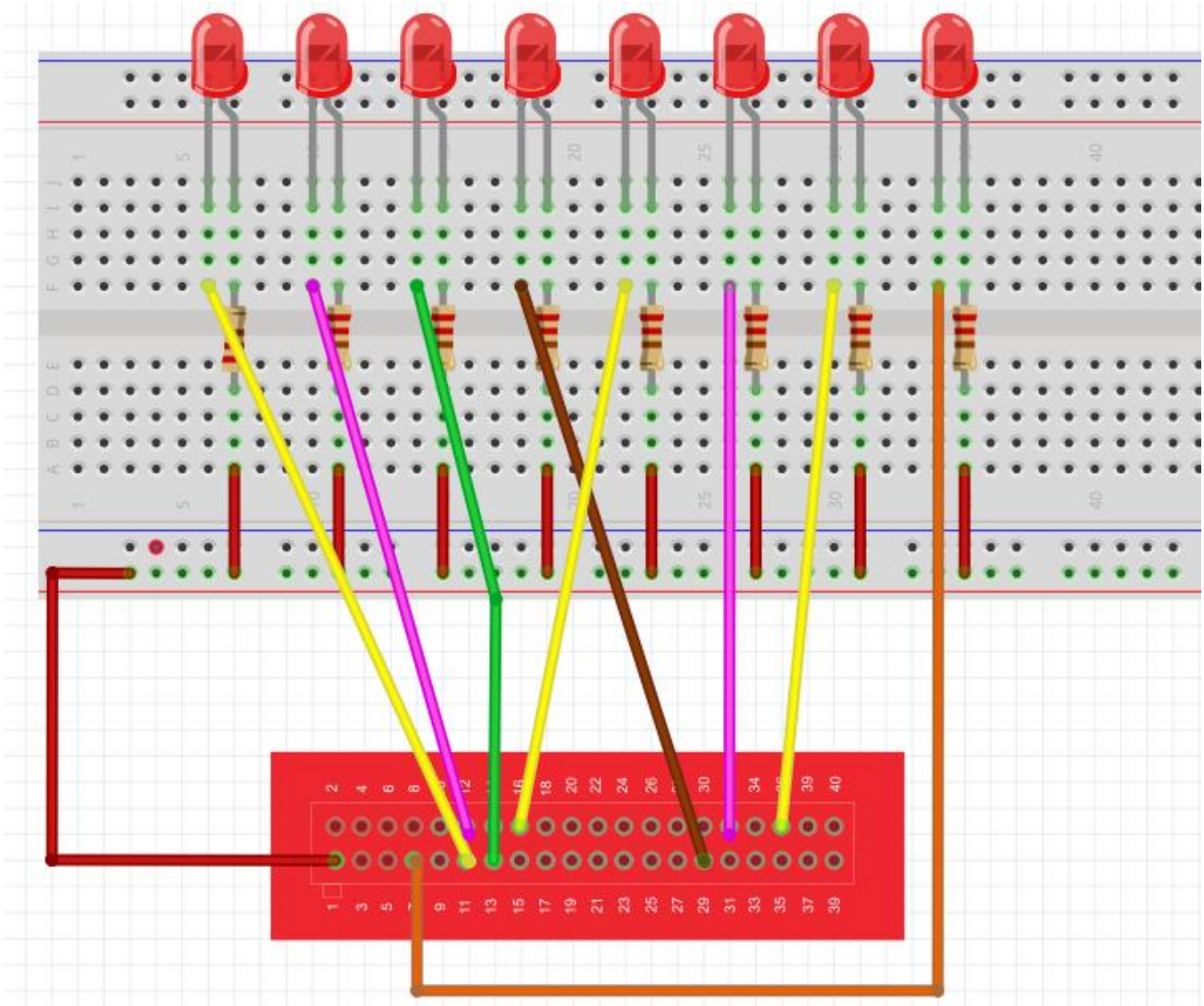
- 1*Raspberry Pi
- 1*Breadboard
- 1*Network cable (or USB wireless network adapter)
- 8*LED
- 8*Resistor (220Ω)
- Jumper wires

Experimental Principal

Control the states of LEDs by passing parameters to the **main()** function.

Experimental Procedures

Step 1: Connect the circuit as shown in the following diagram (same as lesson 3)



Step 2: Edit and save the code with **vim** (see path/Rpi_SuperKit_Code/04_CmdCtrlLed/CmdCtrlLed.c)

Step 3: Compile the code

```
gcc CmdCtrLed.c -o CmdCtrLed -lwiringPi
```

Step 4: Run the program

```
./CmdCtrLed 2 1
```

Now, when you press Enter, you will see the second LED light up (The first parameter represents the number of an LED; the second represents the states of an LED, thereinto, 1 stands for ON, while 0 stands for OFF).

Further Exploration

We can also pass three parameters to **main()** function, thereinto, the first parameter indicates which LED is used, the second indicates the states of the LED, the third indicates the time when the LED is turned on. The LED will go out automatically if this time is surpassed. Please program to realize this effect by yourself.

Summary

This lesson explains how to pass parameters to main function. Take main(int argc, char *argv[]) for example, argc represents the number of parameters, *argv[] represents parameter list.

SUNFOUNDER's Raspberry Pi Lesson 5 — LED Breathing Light

Introduction

In this lesson, we will gradually increase and decrease the luminance of an LED with PWM (Pulse-width modulation) technology, which looks like breathing. So we give it a magical name - Breathing Light.

Experimental Conditions

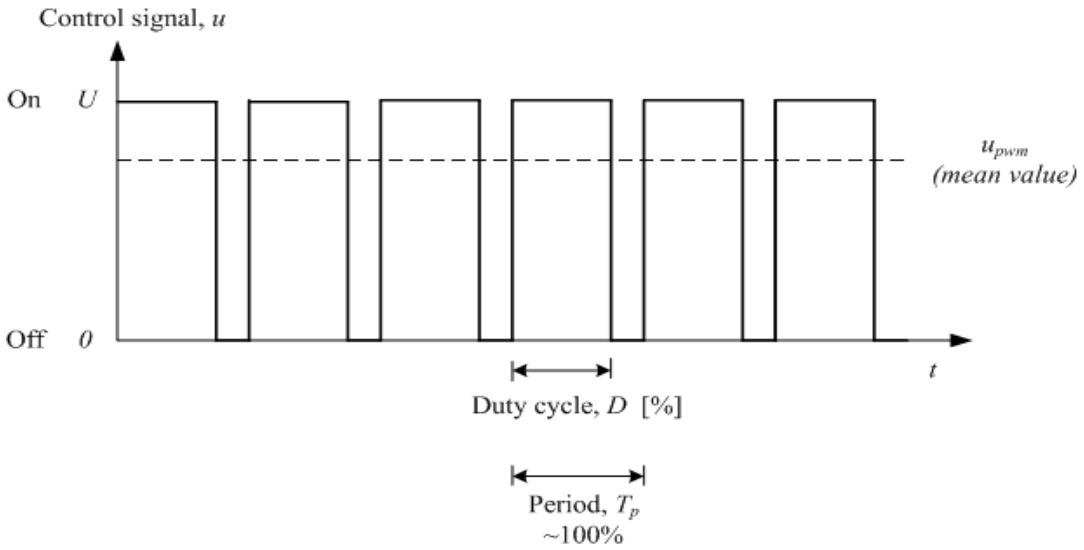
- 1*Raspberry Pi
- 1*Breadboard
- 1*Network cable (or USB wireless network adapter)
- 1*LED
- 1*Resistor (220Ω)
- Jumper wires

Experimental Principle

Before we talk about PWM, let's have a look at the applications of PWM first. PWM has been successfully applied in motor speed regulation, steering angle control, light intensity control and signal output. For example, when PWM is applied to a horn, it will make a sound. After we know about its special functions, let's find out what PWM really is.

Pulse Width Modulation commonly refers to PWM. Pulse Width Modulation (PWM) is a digital coding method for analog signal levels. Since a computer cannot output an analog voltage but digital voltage value 0V or 3.3V, we modulate the duty cycle of square waves to encode a specific level of analog signal by using a high-resolution counter. PWM signals are essentially digital signals, for the full amplitude DC power supply is either 3.3V (ON) or 0V (OFF) at any given time. Voltage or current source is applied to an analog load in the form of ON or OFF repetitive pulse sequence. When it is on, DC power supply will be applied to the load; when it is off, DC power supply will be disconnected. If only the bandwidth is wide enough, any analog value can be encoded by PWM. The output voltage value is calculated by the on and off time, $V_{out} = (T_{on} / T)*V_{max}$.

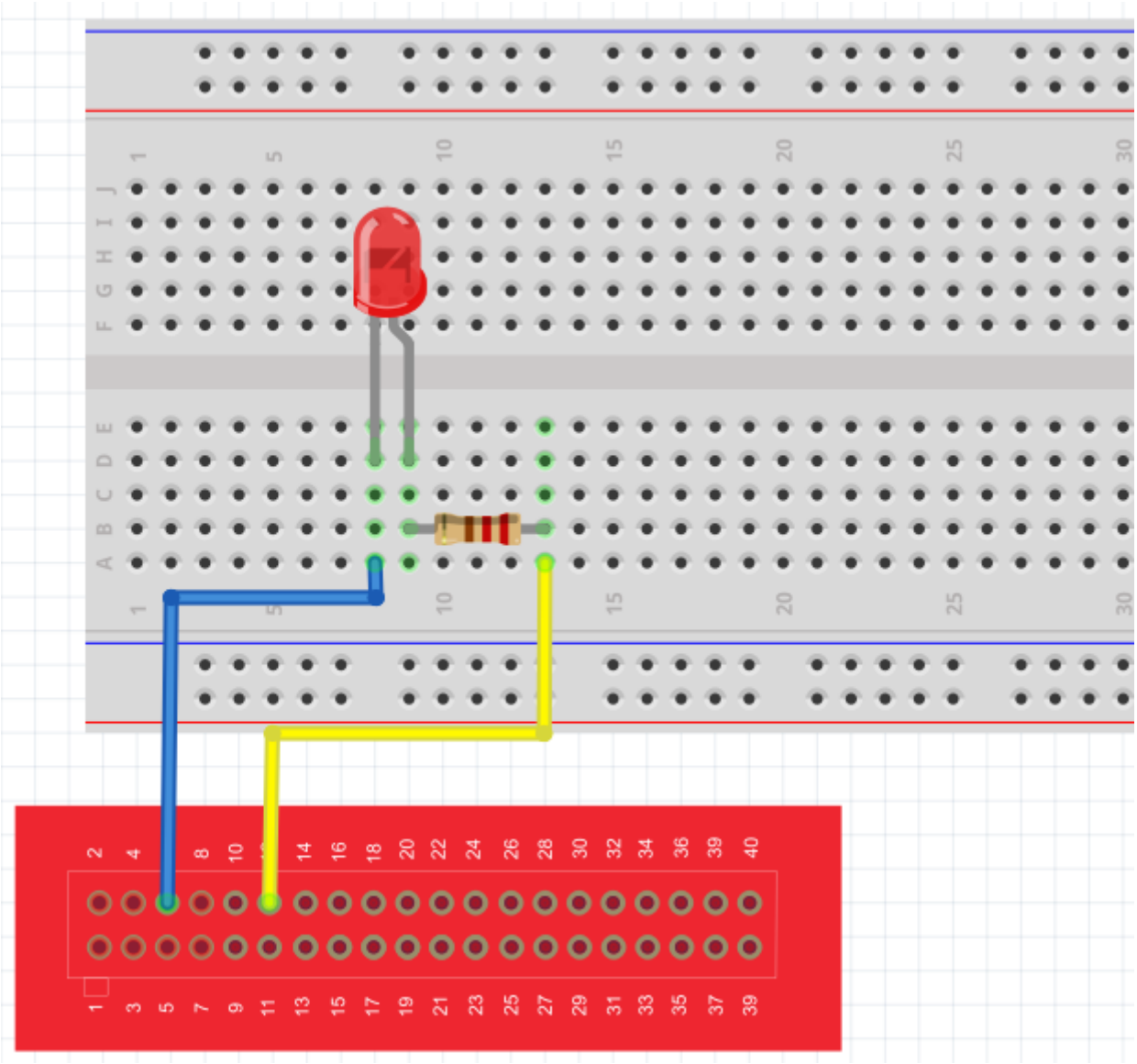
Here is the introduction to three basic parameters of PWM:



1. The term *duty cycle* describes the proportion of 'on' time to the regular interval or 'period' of time
2. The term *period* describes the reciprocal of pulses in one second
3. Voltage amplitude (e.g. 0V-3.3V)

Experimental Procedures

Step 1: Connect the circuit as shown in the following diagram



Step 2: Edit and save the code with vim (see path/Rpi_SuperKit_Code/05_PwmLed/PwmLed.c)

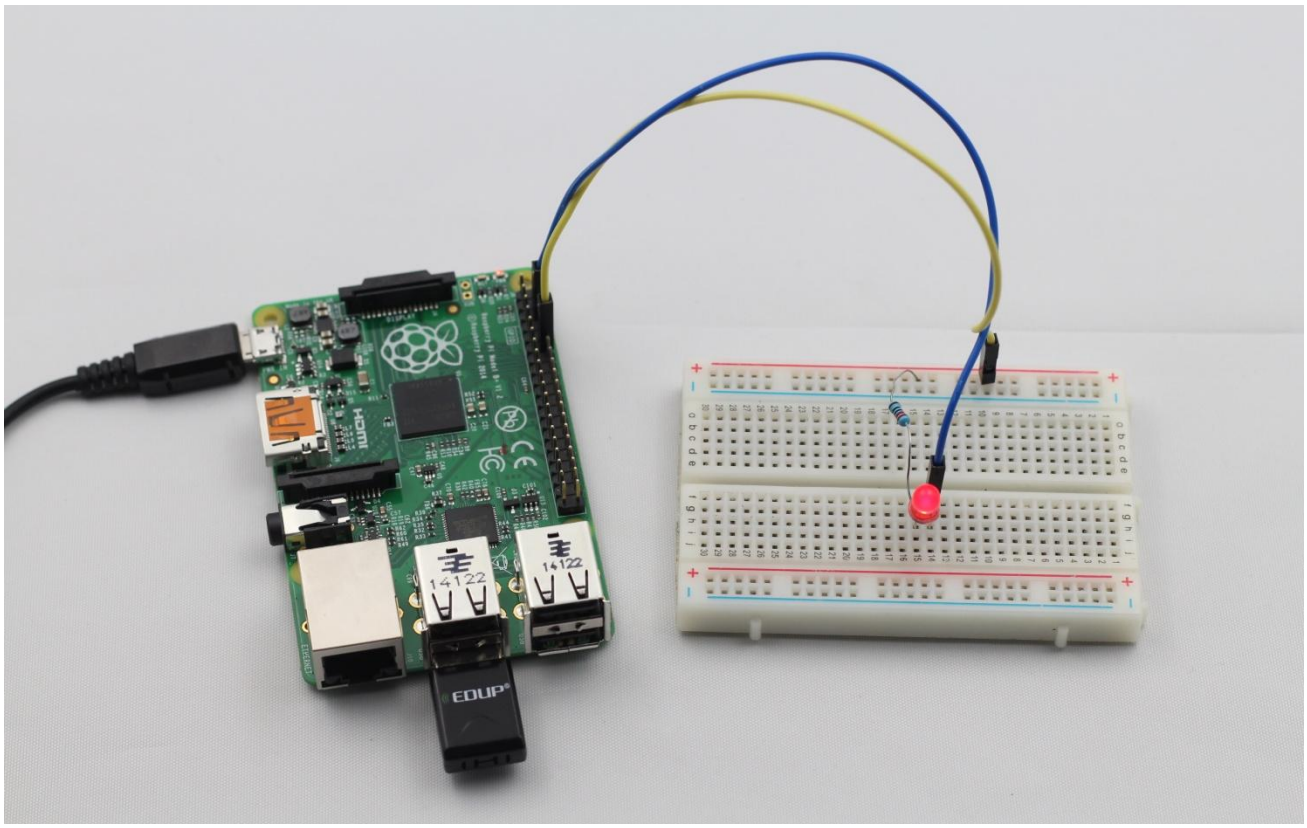
Step 3: Compile the code

```
gcc PwmLed.c -o PwmLed -lwiringPi
```

Step 4: Run the program

```
./PwmLed
```

Press Enter and you will see a gradual change of the LED luminance, rendering the effect of breathing.



Summary

Through this experiment, you should have mastered the technical principle of PWM and how to program Raspberry Pi with PWM. You can apply this technology to DC motor speed regulation in the future.

SUNFOUNDER's Raspberry Pi Lesson 6 — RGB LED

Introduction
In this lesson, we will learn how to use RGB LEDs.

RGB LEDs can emit various colors of light. They are manufactured by packaging three LEDs of red, green, and blue into a transparent or semitransparent plastic shell and lead out four pins. The three primary colors of red, green, and blue can be mixed to all kinds of colors by brightness, so you can make RGB LEDs emit light with all kinds of colors by controlling the circuit.

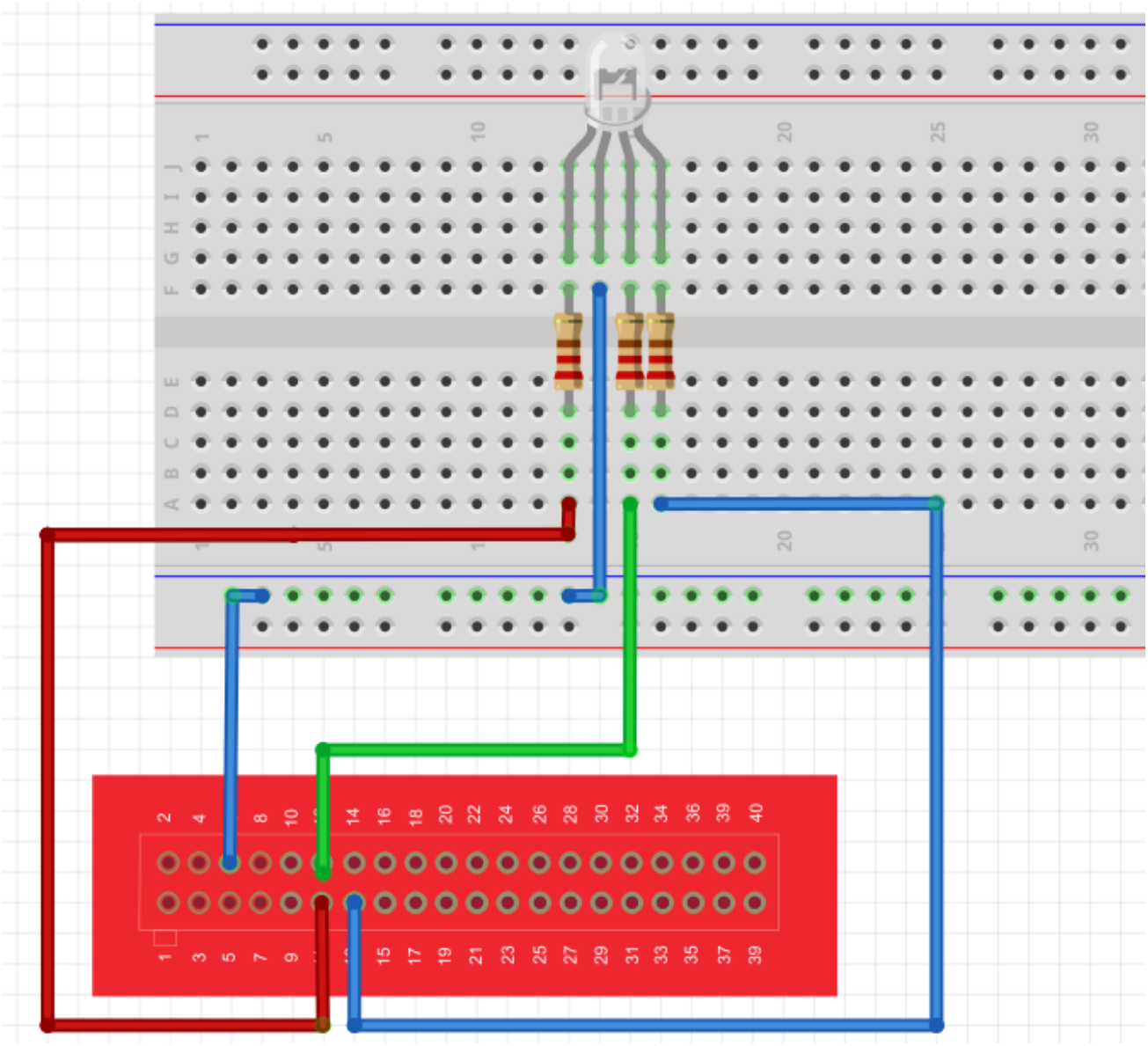
Components
- 1* Raspberry Pi
- 1* Network cable (or USB wireless network adapter)
- 1* RGB LED
- Several jumper wires

Experimental Principle
The three primary color red, green and blue of a RGB LED can compose various colors by brightness. We can adjust the brightness of LED with PWM technology. Raspberry Pi has only one channel hardware PWM output, but we need three channels to control the RGB LED. As a result, it is difficult to realize with the hardware PWM of Raspberry Pi. Do not worry! Fortunately the softPwm library simulates PWM (softPwm) for us with software method. Based on this, we only need to include the header file softPwm.h, then call the API it provided to easily achieve multi-channel PWM output to control the RGB LED to display all kinds of colors.

There are two types of package for RGB LEDs. One is patch type, and the other is dual-in-line type. The difference between them is color resolution. The former has better color resolution.

RGB LEDs can be categorized into common anode type and common cathode type. In this experiment, we use common cathode RGB LED.

Experimental Procedures
Step 1: Connect the circuit as shown in the following diagram



Step 2: Edit and save the code with **vim** (see path/Rpi_SuperKit_Code/06_RGB/rgb.c)

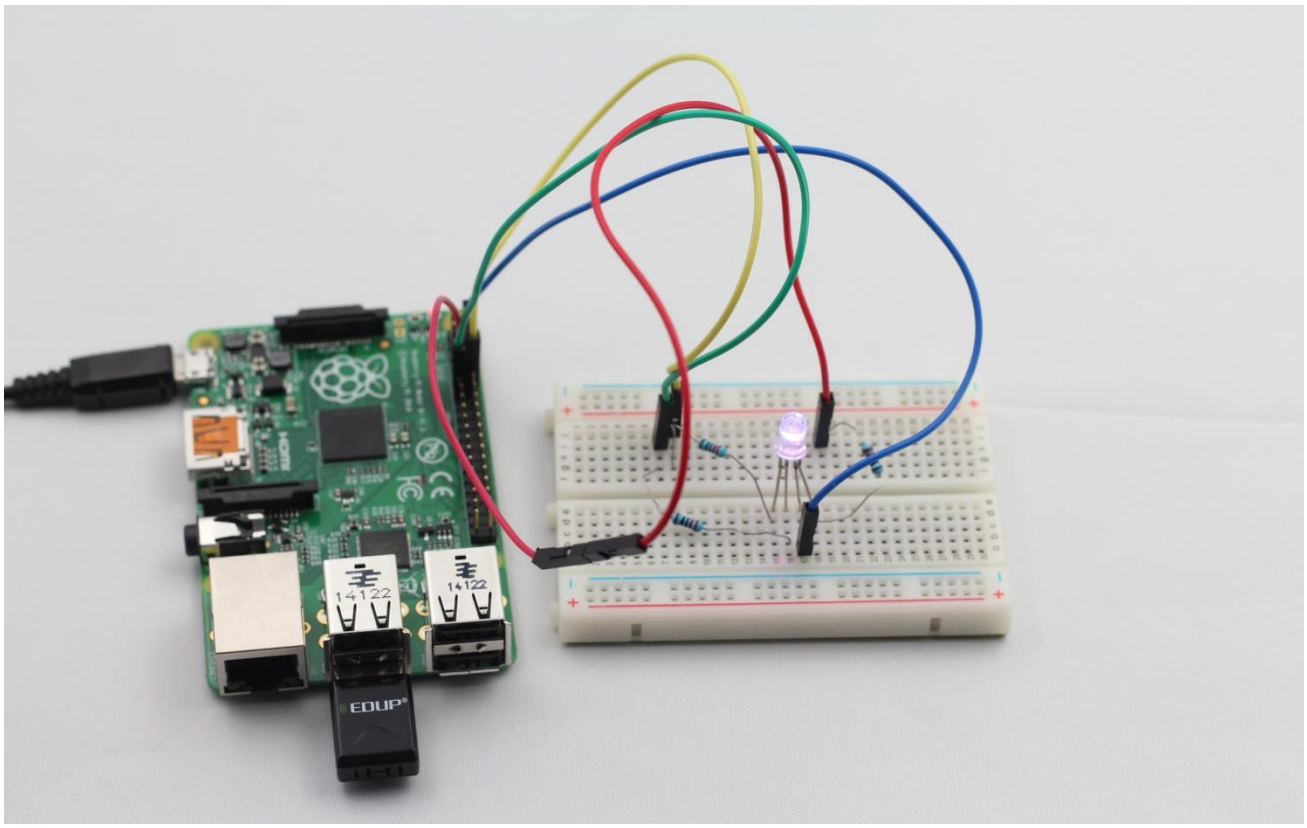
Step 3: Compile the code

```
gcc rgb.c -o rgb -lwiringPi -lpthread
```

Step 4: Run the program

```
./rgb
```

Press Enter, you will see the RGB LED light up, and display different colors in turn.



Further Exploration

I hope you can modify the parameters of the function ledColorSet( ) by yourself, then compile and run the program to see the color changes of the RGB LED.

Experimental Summary

You have learnt how to control RGB LEDs with softPwm of the Raspberry Pi through this lesson. I hope you can continue to explore softPwm application in DC motor speed regulation.

SUNFOUNDER's Raspberry Pi Lesson 7 — Buzzer

Introduction

In this lesson, we will learn how to drive an active buzzer with a PNP transistor to make sounds.

Experimental Conditions

- 1*Raspberry Pi
- 1*Breadboard
- 1*Network cable (or USB wireless network adapter)
- 1*Buzzer (Active)
- 1*PNP transistor (8550)
- 1*Resistor (1KΩ)
- Jumper wires

Experimental Principle

As a type of electronic buzzer with integrated structure, buzzers, which use DC power supply, are widely used in computers, printers, photocopiers, alarms, electronic toys, automotive electronic equipments, telephones, timers and other electronic products for voice devices. Buzzers can be categorized as piezoelectric and magnetic buzzers. A piezoelectric buzzer is mainly composed of multivibrator, piezoelectric buzzer slice, impedance matcher, resonance chamber, shell, etc. A magnetic buzzer is mainly composed of oscillator, electromagnetic coil, magnet, vibrating diaphragm, shell, etc. Buzzers can also be categorized as active and passive buzzers (See the following pictures). When we place the pins of two buzzers upwards, we can see the one with green circuit board is a passive buzzer, while the one without circuit board instead of enclosing with black tape is an active buzzer.

The difference between an active buzzer and a passive buzzer is:

The active buzzer has built-in oscillating source, so it will make sounds as long as it is electrified. While the passive buzzer does not have oscillating source, so it will not tweet if

you use DC signals, instead you must use square waves whose frequencies are between 2K and 5K to drive it. The active buzzer is often more expensive than the passive because multiple built-in oscillating circuits exist.

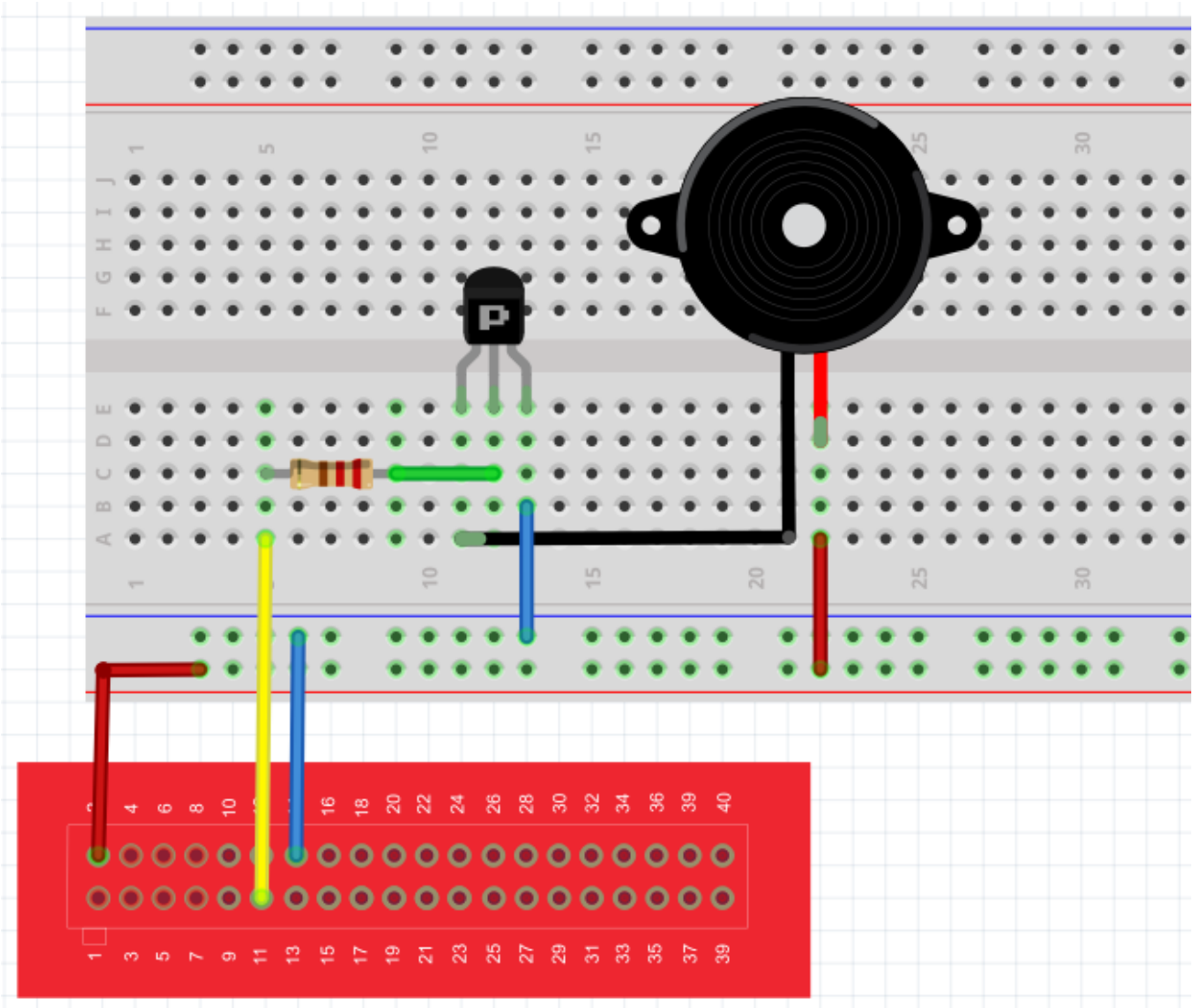
The advantage of a passive buzzer is:
1. Cheap
2. Voice frequency controllable, able to make the sounds ”do re mi fa so la si do”
3. Able to share a control port with an LED in some special cases

The advantage of an active buzzer is:
Easy to control by the program

In this experiment, we use an active buzzer. When we make the GPIO of Raspberry Pi output low level (0V) by programming, the transistor will conduct because of current saturation and the buzzer will make sounds. But when we supply high level to the IO of Raspberry Pi, the transistor will cut-off and the buzzer will not make sounds.

Experimental Procedures

Step 1: Connect the circuit as shown in the following diagram (Pay attention to the positive and negative poles of the buzzer)



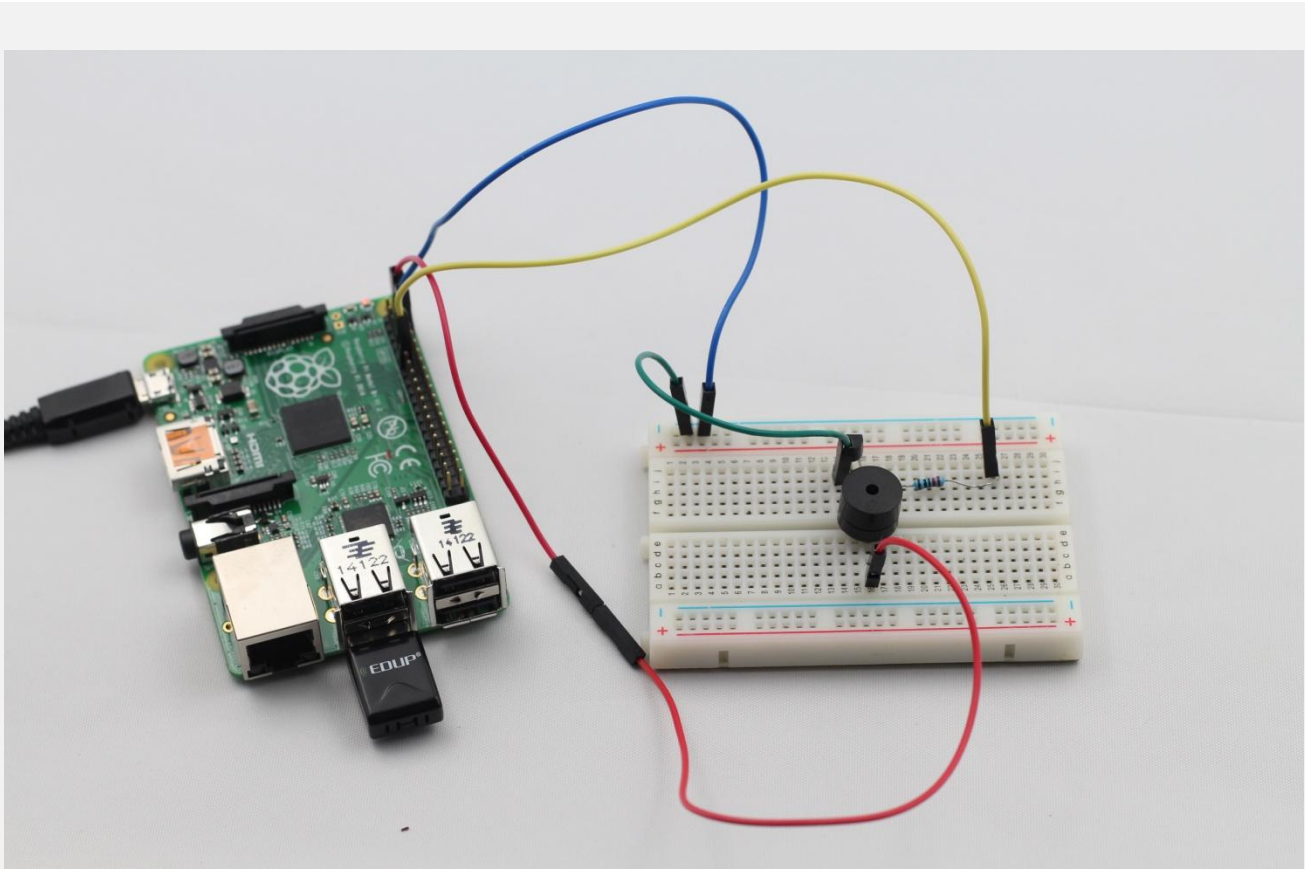
Step 2: Edit and save the code with **vim** (see path/Rpi_SuperKit_Code/07_Beep/beep.c)
Step 3: Compile the code

```
gcc beep.c -o beep -lwiringPi
```

Step 4: Run the program

```
./beep
```

Now, you should hear the buzzer make sounds.



Further Exploration

If you have a passive buzzer in hand, you can replace the active buzzer with it. So you can make it sound “do re mi fa so la si do” through programming as long as you have some programming foundation and enough patience.

Summary

Through this lesson, you have mastered the classification and principle of buzzers. I hope you can try harder, and there will be more wonderful and interesting contents waiting for you in later parts.

SUNFOUNDER's Raspberry Pi Lesson 8 — How to Drive a DC Motor

Introduction

In this lesson, we will learn to how to use L293D to drive a DC motor and make it rotate clockwise and counterclockwise.

Experimental Conditions

- 1*Raspberry Pi
- 1*Breadboard
- 1*Network cable (or USB wireless network adapter)
- 1*L293D
- 1*DC motor
- 2*Optocoupler (4N35)
- 4*Diode (1N4007)
- 4*Resistor (2*1KΩ, 2*10K)
- Jumper wires

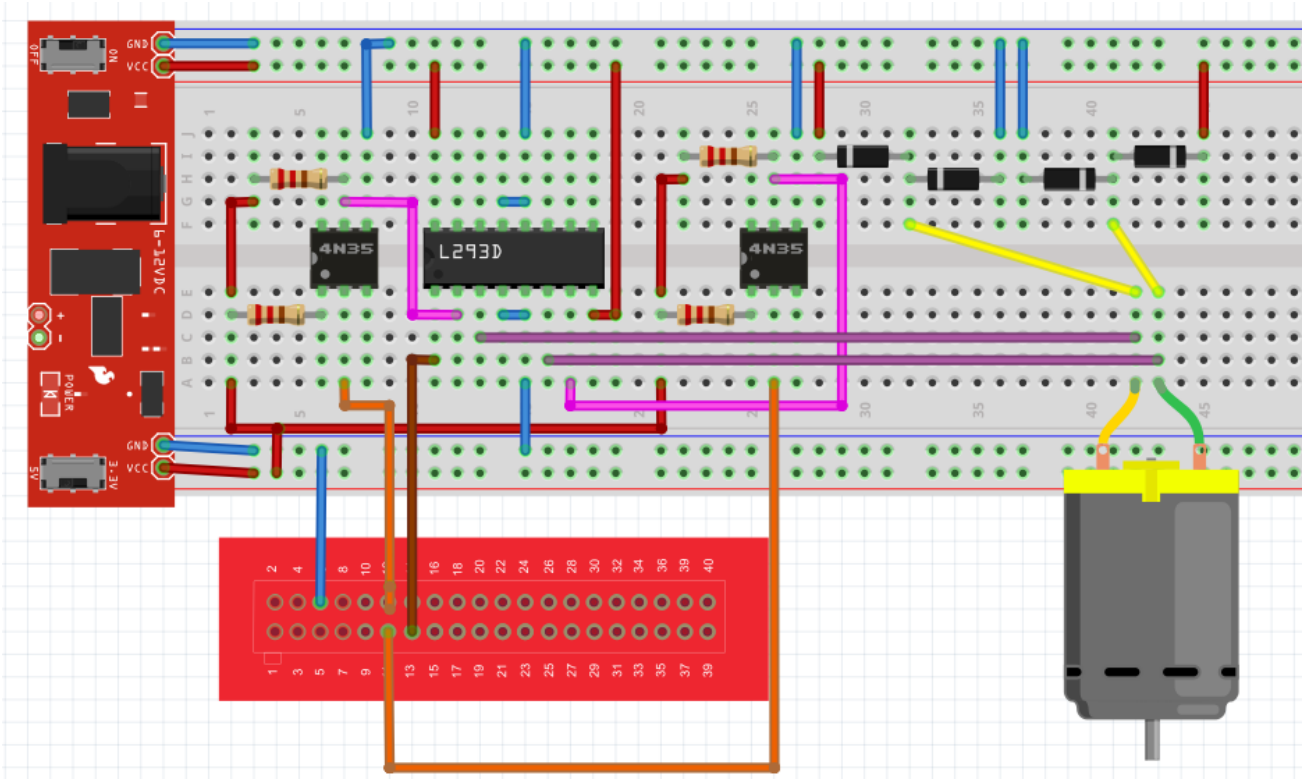
Experimental Principle

DC Motors are devices that turn DC electrical energy into mechanical energy. They are widely used in electric drive for their superior speed regulation performance. Based on the exciting mode, DC Motors can be categorized into three types: permanent magnet motor, separately excited motor and self-excited motor. Thereinto, separately excited motor also can be categorized into three types: shunt excited motor, series excited motor and compounded excited motor. DC motors are ubiquitous in our daily life from electric shavers and toy cars to the traction device of subway.

Connect the two IOs of your Raspberry Pi to two optocoupler which isolates the power supply to prevent your Raspberry Pi from interference and crashing when the motor works. Connect the output port of optocoupler to the driver IC (L293D) of motor. When one electrical level of the two Raspberry Pi IOs is high, but the other is low, you can realize the positive and reversal control of motor. By regulating the pulse width, you can realize DC motor speed regulation as well. Motors belong to inductive equipment, the four diodes of which play an important role in freewheeling and suppress the surge voltage of motor coils to prevent irreversible damages to driver ICs.

Experimental Procedures

Step 1: Connect the circuit as shown in the following diagram



Step 2: Edit and save the code with vim (see path/Rpi_SuperKit_Code/08_Motor/motor.c)

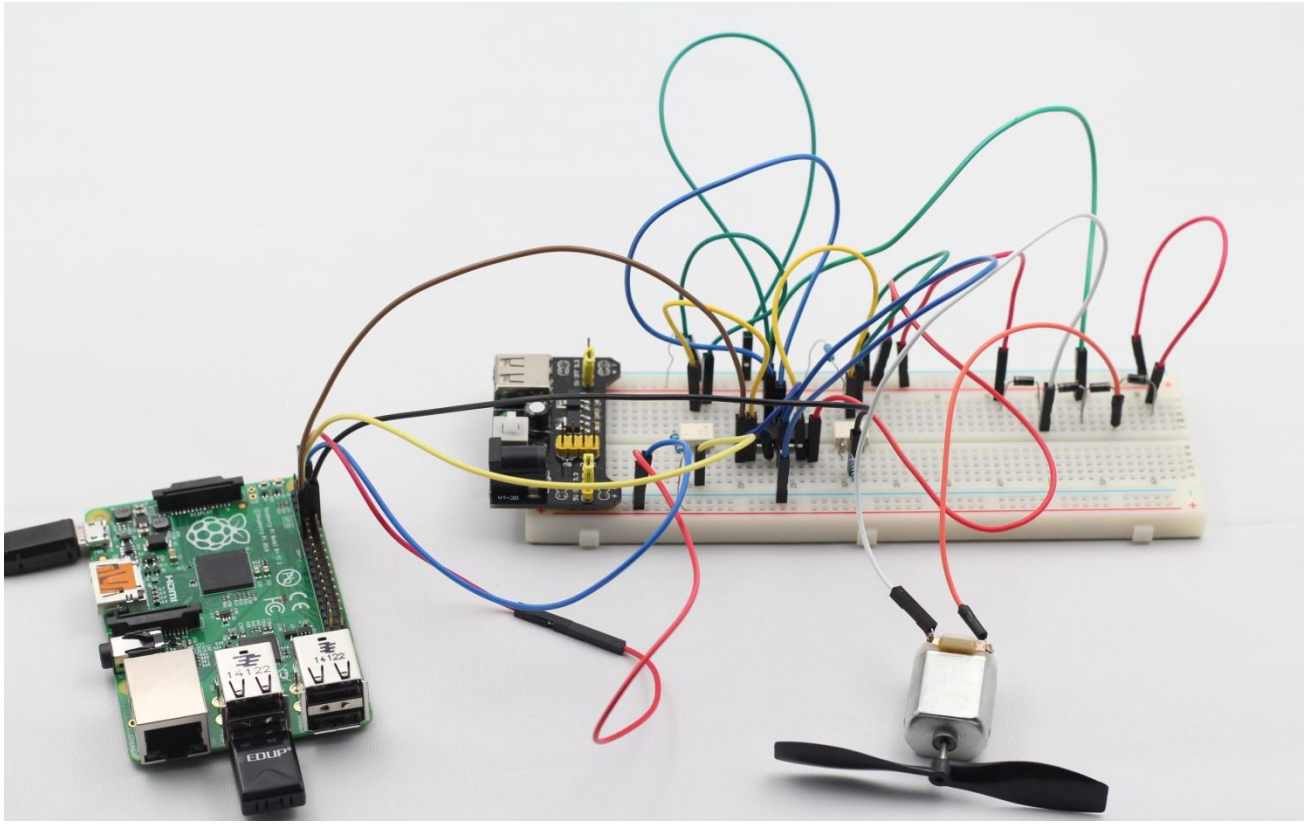
Step 3: Compile the code

```
gcc motor.c -o motor -lwiringPi
```

Step 4: Run the program

```
./motor
```

Now, you should see the motor rotating.



Further Exploration

We can use buttons to control the positive and reversal rotation of motor with previous learned knowledge, and we can use PWM principle to control the rotation of motor too.

Summary

Through this lesson, we have learnt the principle of operation and driving mode of DC motors. In addition, we also have learnt how to drive a motor based on the Raspberry Pi. What you must pay special attention to is that a DC motor will greatly interfere with the whole circuit when it works, so we must adopt photoelectric isolation and separate power supply. A freewheeling diode is also necessary, otherwise the whole system will not work reliably and steadily.

SUNFOUNDER's Raspberry Pi Lesson 9 — Rotary Encoder

Introduction

In this lesson, we will learn how to use rotary encoders.

Experimental Conditions

- 1*Raspberry Pi
- 1*Breadboard
- 1*Network cable (or USB wireless network adapter)
- 1*Rotary Encoder
- 2*Capacitor (100nF)
- 2*Resistor (10KΩ)
- Jumper wires

Experimental Principle

Rotary Encoder is a switch electronic component with a set of regular and strict timing sequence pulses. When using with IC, it can achieve increment, decrement, page turning and other operations. For example, mouse scroll, menu selection, acoustic sound regulation, frequency regulation, toaster temperature regulation, and so on.

Most rotary encoders have 5 pins with three functions of turning left, turning right and pressing down. Pin 4 and Pin 5 are switch wiring terminals used to press. They have no difference with buttons previously used, so we will no longer discuss them in this experiment. Pin 2 is generally connected to ground. Pin 1 and Pin 3 are first connected to pull-up resistor and then to microprocessor. In this experiment, they are connected to

GPIO0 and GPIO1 of Raspberry Pi. When we rotate clockwise and anti-clockwise, there will be pulse outputs in pin 1 and pin 3.

The figure shows, if GPIO0 is at high level and GPIO1 is also at high level, it indicates the switch rotates clockwise; if GPIO0 is at high level but GPIO1 is at low level, it indicates the switch rotates anti-clockwise. Therefore, during programming, you only need to judge the state of pin 3 when pin 1 is at high level, then you can judge whether the switch is rotates clockwise or anti-clockwise.

Experimental Procedures

Step1: Connect the circuit according to the following method

Raspberry Pi	Rotary Encoder Module
3.3V	'+'
GND	GND
GPIO0	DT
GPIO1	CLK

Step 2: Edit and save the code with vim (see path/Rpi_SuperKit_Code/09_RotaryEncoder /rotaryEncoder.c)

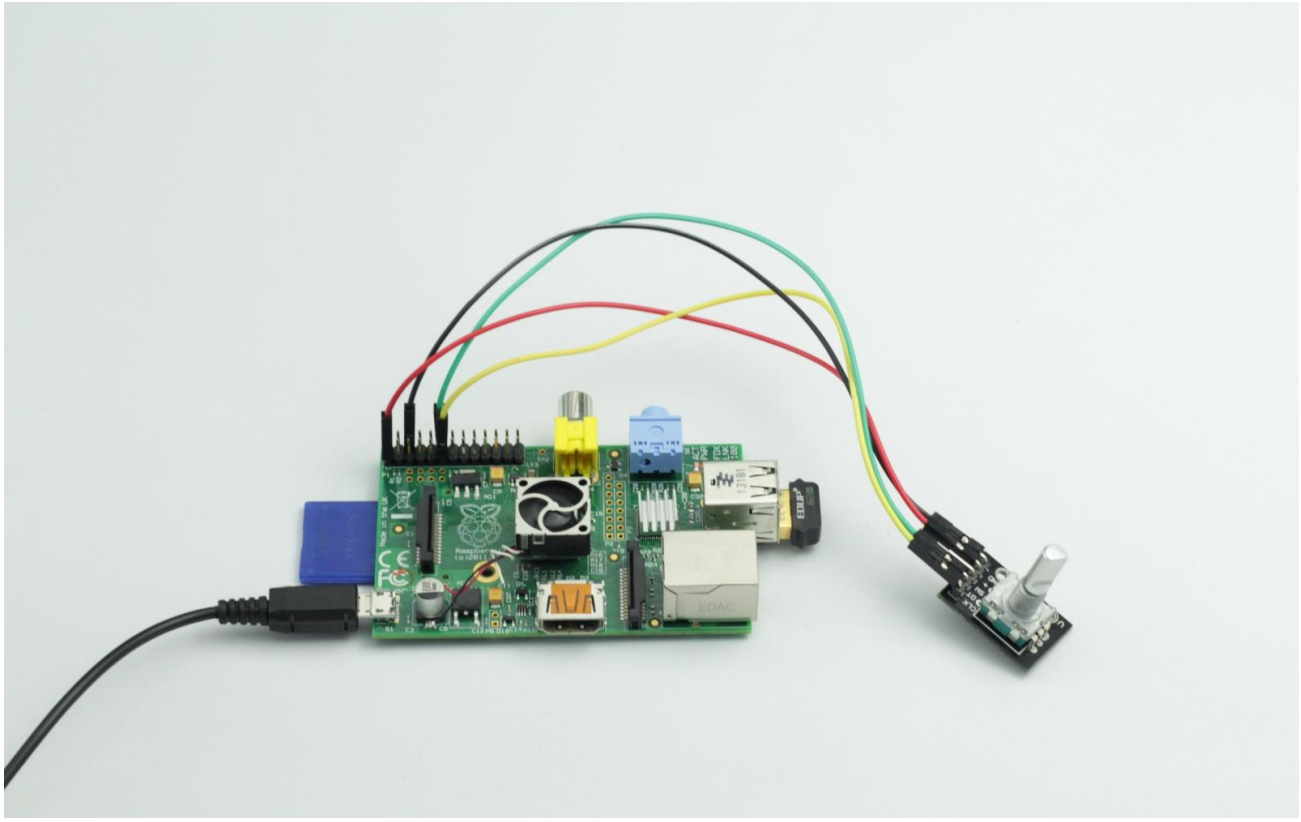
Step 3: Compile the code

```
gcc rotaryEncoder.c -o rotaryEncoder -lwiringPi
```

Step 4: Run the program

```
./rotaryEncoder
```

Now, press Enter, gently rotate the rotary encoder to change the value of variable in the above program, you will see the value of variable printed on the screen. When you rotate the rotary encoder clockwise, the value increase; when you rotate it counterclockwise, the value decrease.



Further Exploration

In this experiment, we do not use the button function of rotary encoder switch. I hope you can realize this function by yourselves. That is, when you press the switch, the value of the variable will be reset.

Summary

Through this lesson, you have been familiar with the principle of operation and programming realization for rotary encoder. I believe that you can use it to make more fun works.

SUNFOUNDER's Raspberry Pi Lesson 10 — 555 Timer

Introduction

In this lesson, we will learn how to use the 555 timer.

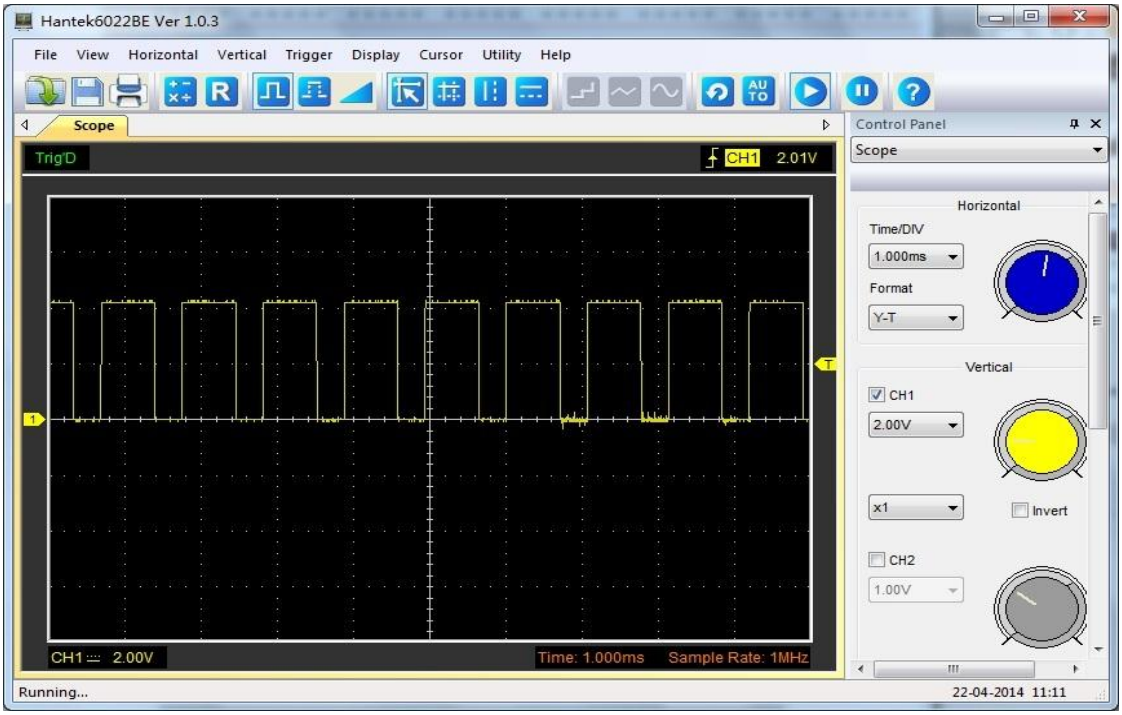
Experimental Conditions

- 1*Raspberry Pi
- 1*Breadboard
- 1*Network cable (or USB wireless network adapter)
- 1*NE555
- 3*Resistor (1*1KΩ, 2*10KΩ)
- 2*Capacitor (100nF)
- Jumper wires

Experimental Principle

The 555 timer is a medium-scale IC device which combines analog and digital functions. With low cost and reliable performance, the 555 timer just attaches external resistors and capacitors to achieve multivibrator, monostable trigger, schmitt trigger and other circuits which can generate and transform pulses. It is also often used as a timer and is widely used in instruments, household appliances, electronic measurement, automatic control and other fields.

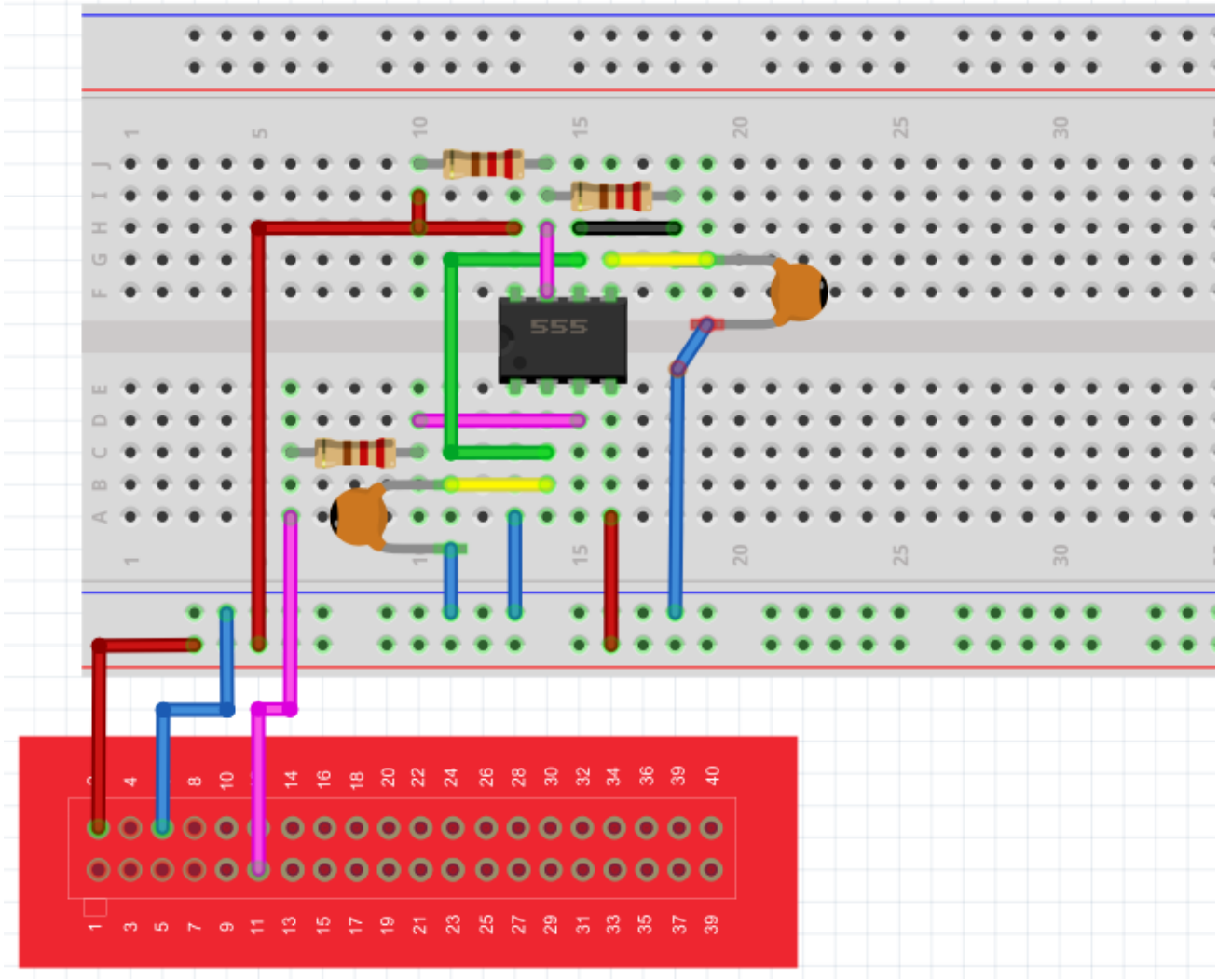
In this experiment, we use the 555 timer as a circuit generating square waves. After connecting the circuit according to schematic diagram, we will use an oscilloscope to observe its output waveform as shown below:



Attach the output pin (e.g. pin 3) of 555 timer to GPIO0 of Raspberry Pi, configure GPIO0 as the mode of rising edge interrupt by programming, then detect square wave pulses generated by the 555 timer with interrupt. The work of Interrupt Service Routine (ISR) is to add 1 to a variable.

Experimental Procedures

Step1: Connect the circuit as shown in the following diagram



Step 2: Edit and save the code with vim (see path/Rpi_SuperKit_Code/10_Timer555/timer555.c)

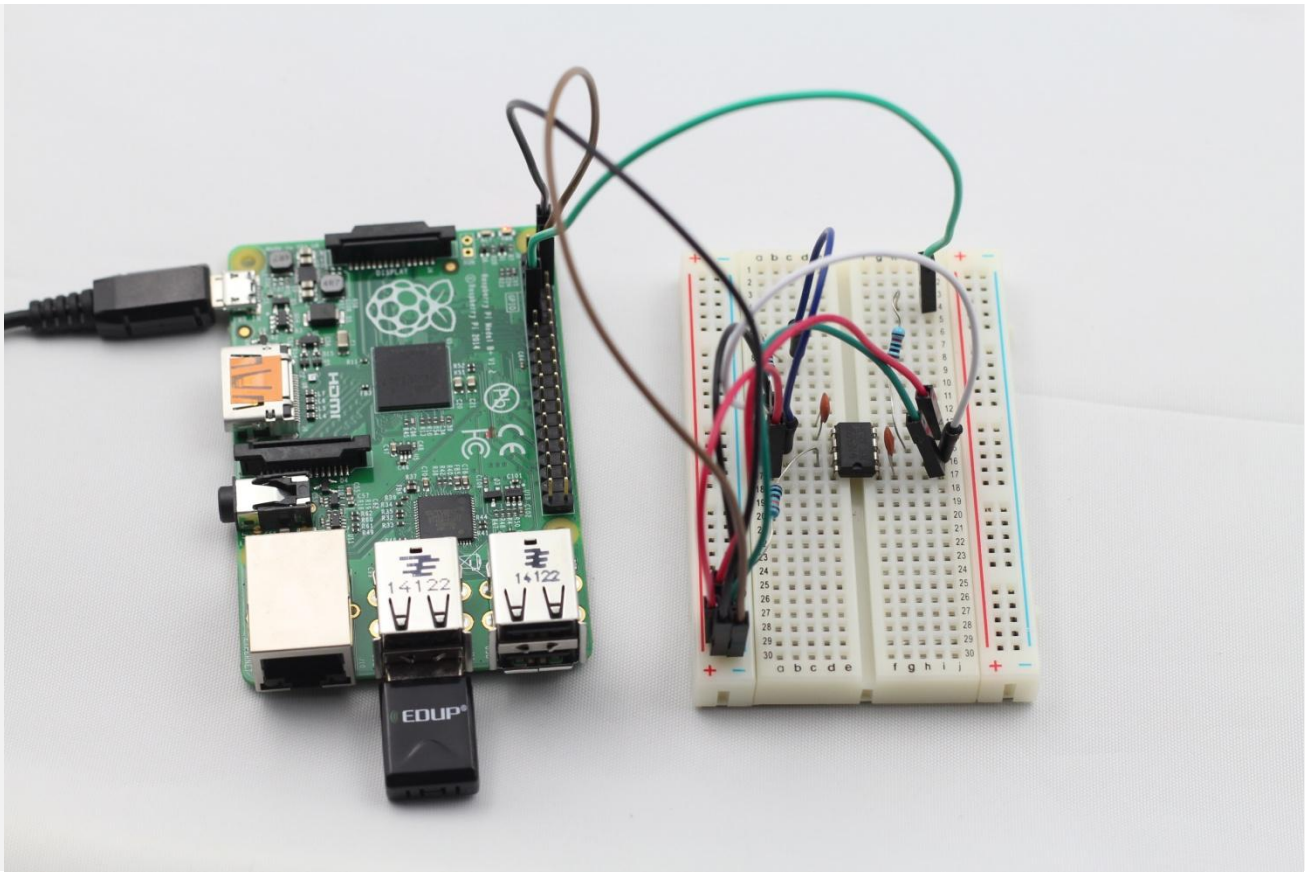
Step 3: Compile the code

```
gcc timer555.c -o timer555 -lwiringPi
```

Step 4: Run the program

```
./timer555
```

Now, you should see data being printed on the display. The data displayed is square waves generated by the 555 timer, for the program will count pulses by interrupt which we have learned previously.



Further Exploration

The above 555 timer circuit outputs square waves with constant frequency and duty cycle. We can simply improve this circuit to realize adjustable frequency and duty cycle. So we can realize LED breathing light explained in previous lesson without Raspberry Pi. Let us get hands-on experience at once.

Summary

Through this lesson, I believe you have basically mastered the usage of the 555 timer and have further learnt the interrupt programming of Raspberry Pi. In fact, the 555 timer has many other interesting functions. Just explore it if you are interested in the 555 timer.

SUNFOUNDER's Raspberry Pi Lesson 11 — Driving LEDs by 74HC595

Introduction

In this lesson, we will learn how to use 74HC595 to make eight LEDs blink regularly.

Experimental Conditions

- 1*Raspberry Pi
- 1*Breadboard
- 1*Network cable (or USB wireless network adapter)
- 1*74HC595
- 8*LED
- 8*Resistor (1KΩ)
- Jumper wires

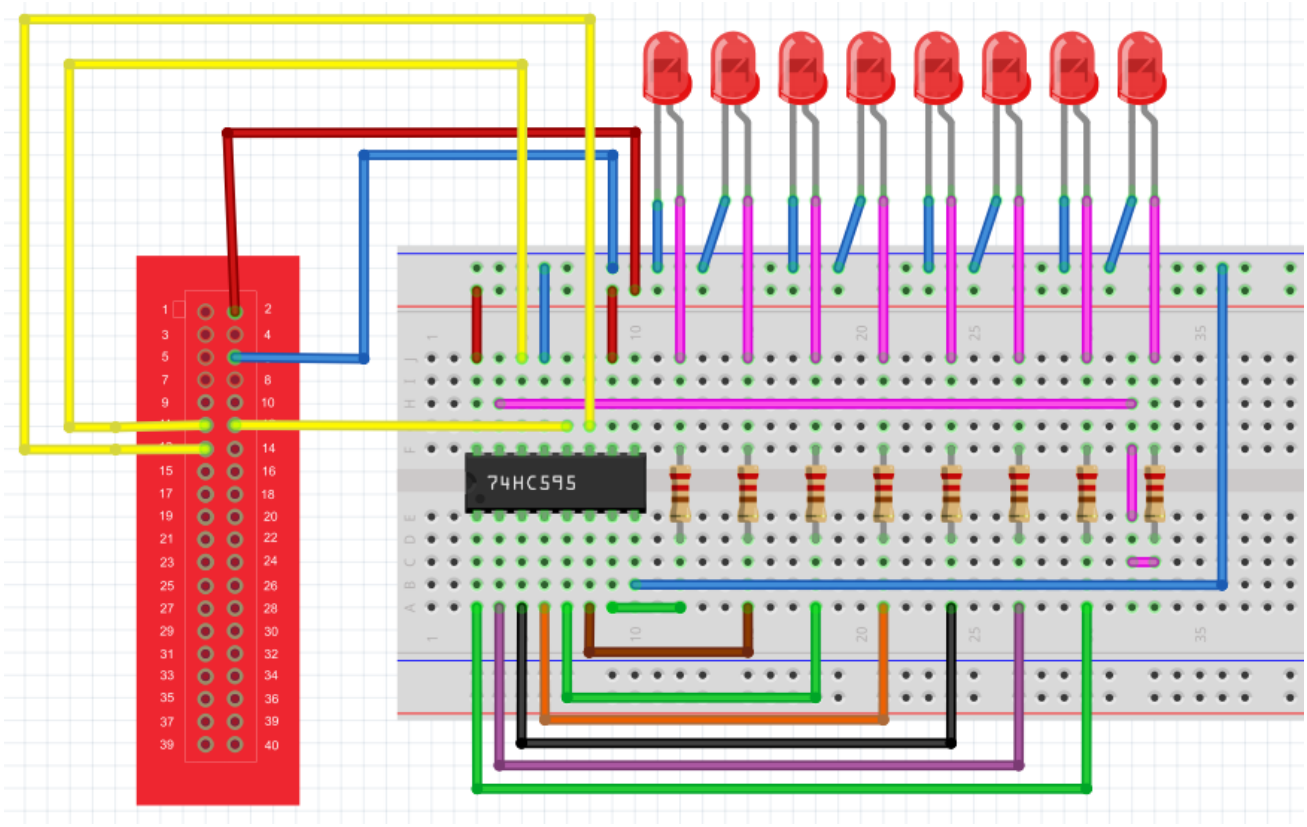
Experimental Principle

74HC595 is a silicon CMOS device, which has an 8-bit shift register and a memory with three-state output function. Compatible with low voltage TTL circuit, 74HC595 can transform serial input of 8-bit data into parallel output of 8-bit data. So it is often used to extend GPIO for embedded system and drive low power devices.

In this experiment, we attach ST_CP to Raspberry Pi GPIO1, SH_CP to GPIO2, and DS to GPIO0. Input data in DS pin to shift register when SH_CP (the clock input of shift register) is at the rising edge and to memory register when ST_CP (the clock input of memory) is at the rising edge. As a result, we can control the states of SH_CP and ST_CP via Raspberry Pi GPIO to transform serial input data into parallel output data so as to save Raspberry Pi GPIOs.

Experimental Procedures

Step 1: Connect the circuit as shown in the following diagram



Step 2: Edit and save the code with vim (see path/Rpi_SuperKit_Code/11_74HC595_LED /74HC595_LED.c)

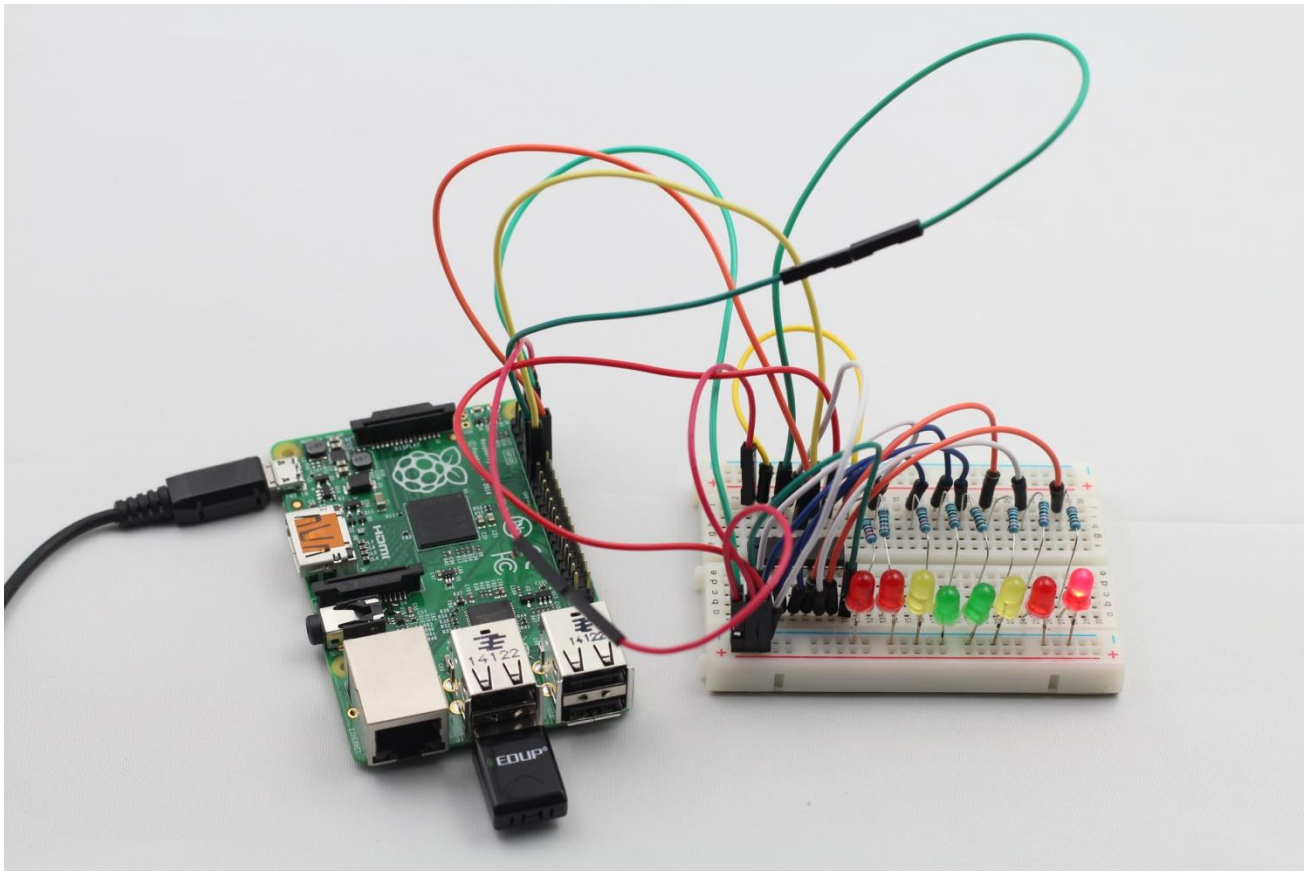
Step 3: Compile the code

gcc 74HC595_LED.c -o 74HC595_LED -lwiringPi

Step 4: Run the program

./74HC595_LED

Press Enter, and you will see eight LEDs blinking regularly, and actual effect is shown as follow:



Further Exploration

In this experiment, we use three Raspberry Pi GPIOs to separately control 8 LEDs based on 74HC595. In fact, 74HC595 has another powerful function – cascade. With cascade, we can use microprocessor, such as 3 Raspberry Pi IOs, to control more peripherals. This function will be explained in later lessons. I hope you can prepare in advance.

Summary

Through this lesson, I believe you have had a basic understanding about 74HC595 and have learnt how to drive 74HC595 based on Raspberry Pi GPIOs. As an enduring integrated circuit (IC), 74HC595 also has many other applications worth us to explore.

SUNFOUNDER's Raspberry Pi Lesson 12 — Driving LED Segment Display by 74HC595

Introduction

In this lesson, we will learn how to use 74HC595 to drive an LED Segment Display to display a figure from 0 to 9.

Experimental Conditions

- 1*Raspberry Pi
- 1*Breadboard
- 1*Network cable (or USB wireless network adapter)
- 1*74HC595
- 1*LED segment display
- 1*Resistor (1KΩ)
- Jumper wires

Experimental Principle

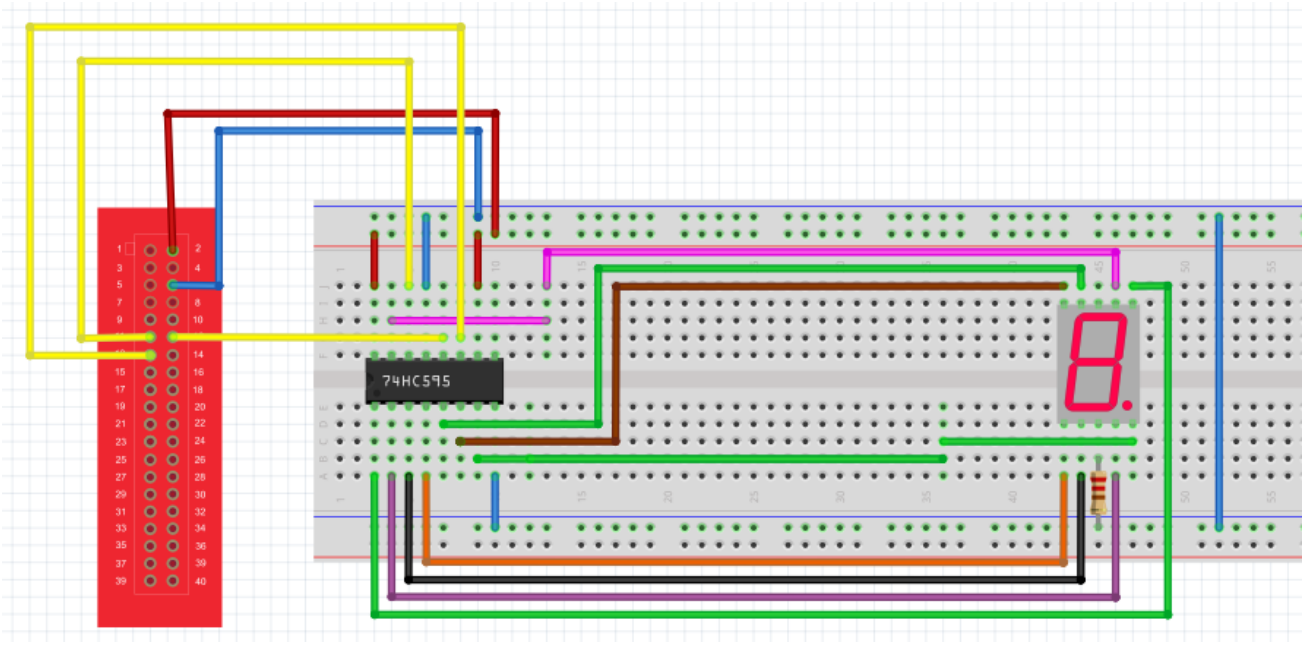
An LED Segment Display is an 8-shaped component which is made up of LEDs packaged together. The leads of an LED Segment Display have been connected internally. External pins are its segments and common electrodes. These segments are respectively represented by letter a, b, c, d, e, f, g, and dp. LED Segment Displays can be categorized into common cathode type and common anode type depending on light-emitting diode connections. LED Segment Displays are widely used in electronic appliances, especially in home appliances such as air conditionings, water heaters, refrigerators and so on because of its cheap price, ease of use, high brightness and long life. Most temperature display units of water heaters are LED segment displays.

In this experiment, we use a common cathode LED segment display, which refers to connecting all cathodes of LEDs together to form a common cathode (COM) segment display. A common cathode LED segment display should connect its common electrode COM to ground. When the anode of an LED in a certain segment is high, the corresponding segment will light up; when it is low, it will not light up.

We connect pin ST_CP of 74HC595 to Raspberry Pi GPIO1, SH_CP to GPIO2, DS to GPIO0, parallel output ports to 8 segments of the LED segment display. Input data in DS pin to shift register when SH_CP (the clock input of shift register) is at the rising edge and to memory register when ST_CP (the clock input of memory) is at the rising edge. As a result, we can control the states of SH_CP and ST_CP via the Raspberry Pi GPIOs to transform serial data input into parallel data output so as to save Raspberry Pi GPIOs and to drive the LED segment display.

Experimental Procedures

Step 1: Connect the circuit as shown in the following diagram



Step 2: Edit and save the code with vim (see path/Rpi_SuperKit_Code/12_Segment/segment.c)

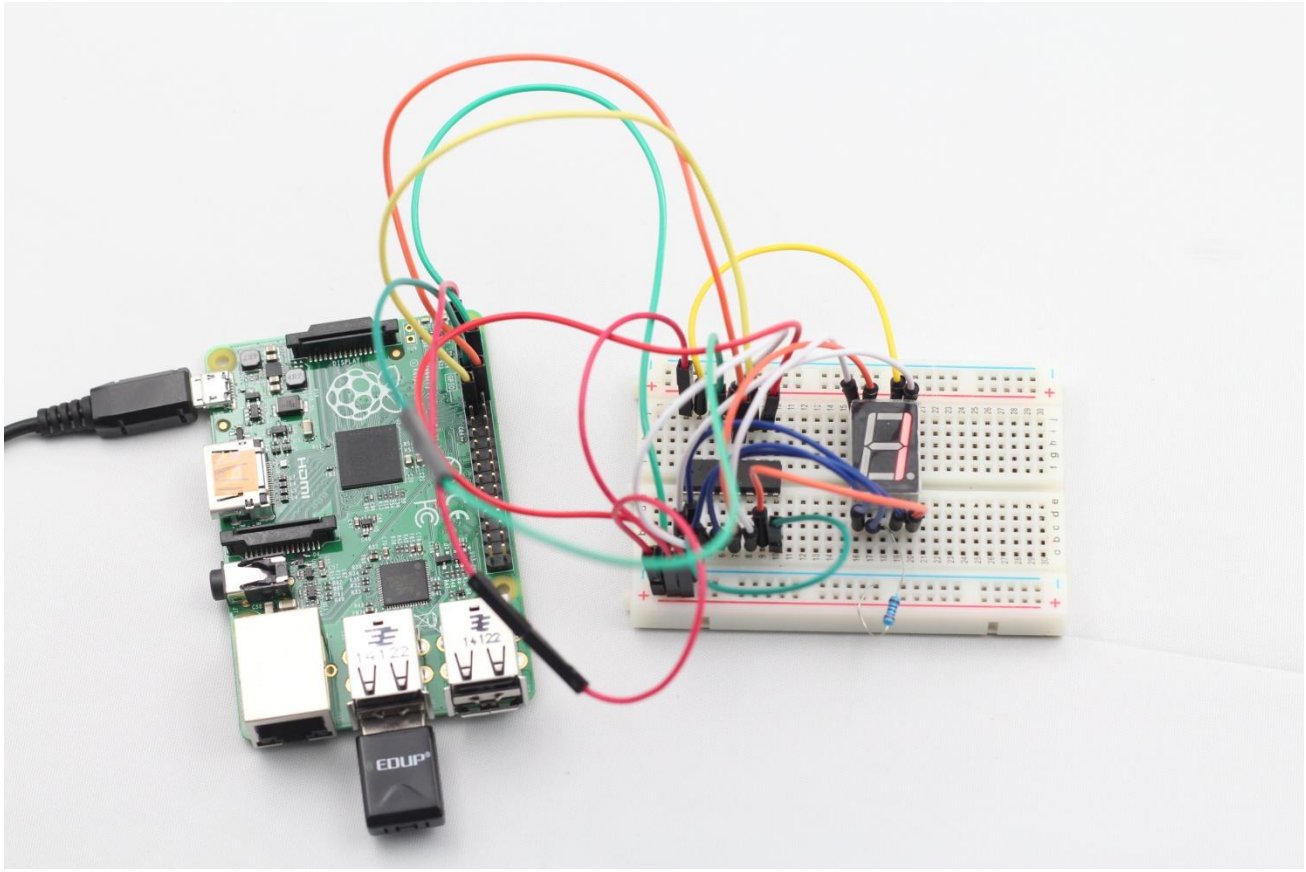
Step 3: Compile the code

```
gcc segment.c -o segment -lwiringPi
```

Step 4: Run the program

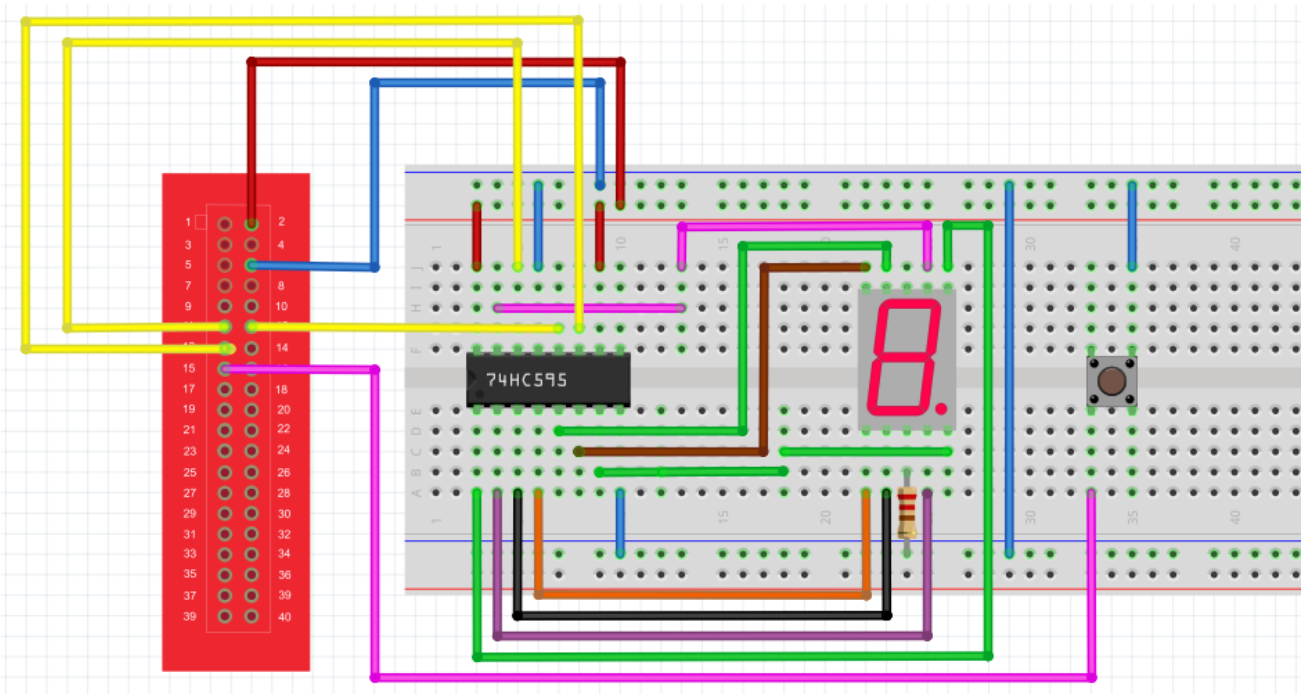
```
./segment
```

Press Enter, then you will see the LED segment display circularly display a figure from 0 to 9. Real effect is shown as follow:



Further Exploration

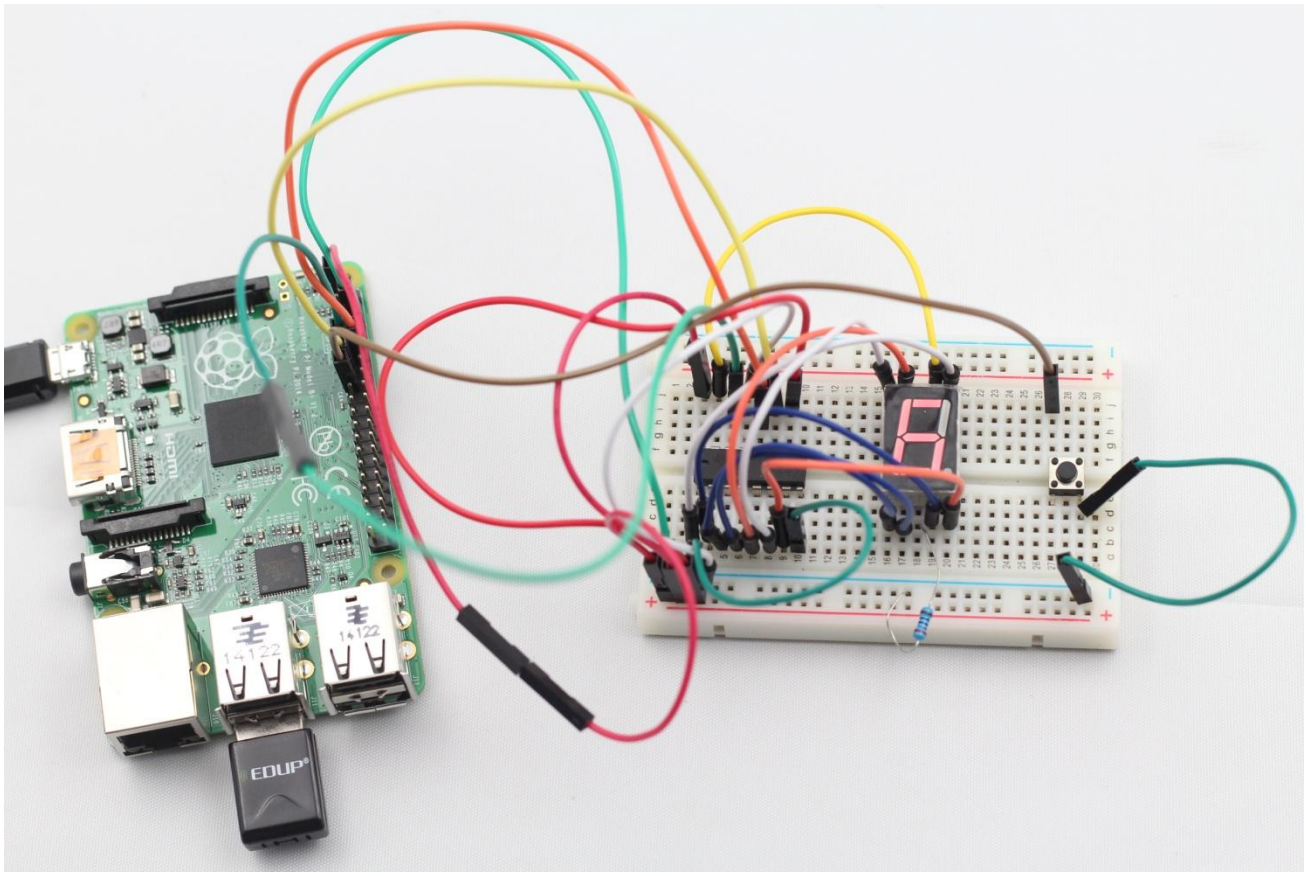
We can slightly modify the hardware and software based on this experiment to make a dice. For hardware, we can add a button on the basis of original board and the modified circuit is shown as follow:



Next we enter the directory Dice for modified program and run the program

./dice

Press Enter, then you will see numbers between 0 and 6 flashing quickly on the segment display. When we press the button on the breadboard, the segment display will statically display a random number between 0 and 6 for 2 seconds and then circularly display random numbers between 0 and 6 quickly again.



Summary

Through this section, you have basically mastered the principle of operation and programming implementation for an LED segment display based on Raspberry Pi. You have more deeply learnt the usage of 74HC595 as well. I hope you can apply what you have learnt and experience its usage in depth by practice.

SUNFOUNDER's Raspberry Pi Lesson 13 — Driving Dot-Matrix by 74HC595

Introduction

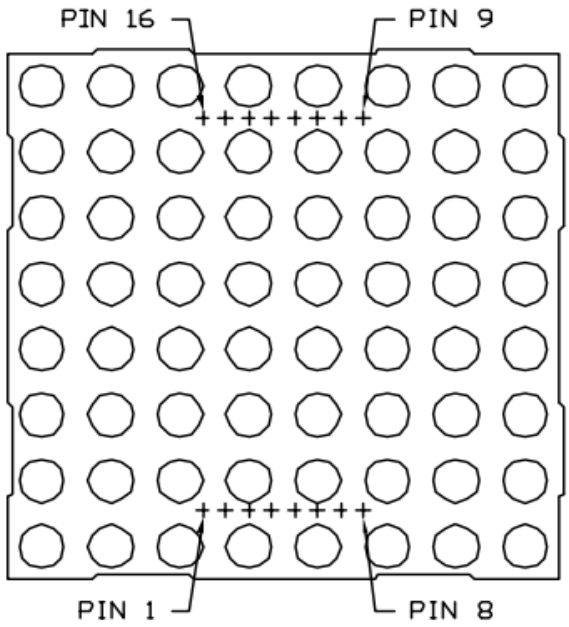
In this lesson, we will learn how to use 74HC595 to drive an LED dot-matrix.

Experimental Conditions

- 1*Raspberry Pi
- 1*Breadboard
- 1*Network cable (or USB wireless network adapter)
- 2*74HC595
- 1*Dot-Matrix
- Jumper wires

Experimental Principle

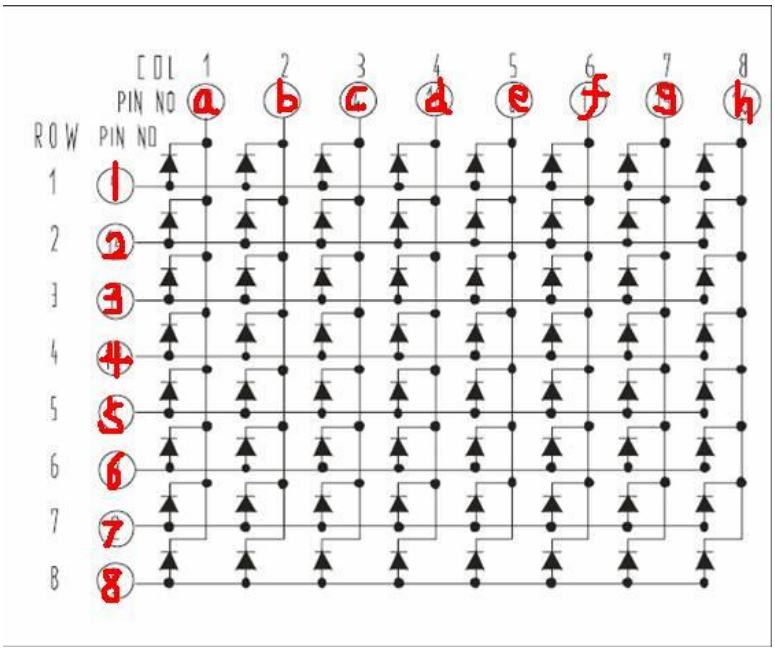
The external view of a dot-matrix is shown as follow:



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, then the corresponding LED will light up; if you want to light the LED on the first dot, you should set PIN 1 to high level and PIN a to low level, then the LED on the first dot will light up; if you want to light the LEDs on the first row, you should set PIN 1 to high level and PIN (a, b, c, d, e, f, g, h) to low level, then all the LEDs on the first row will light up; if you want to light the LEDs on the first column,

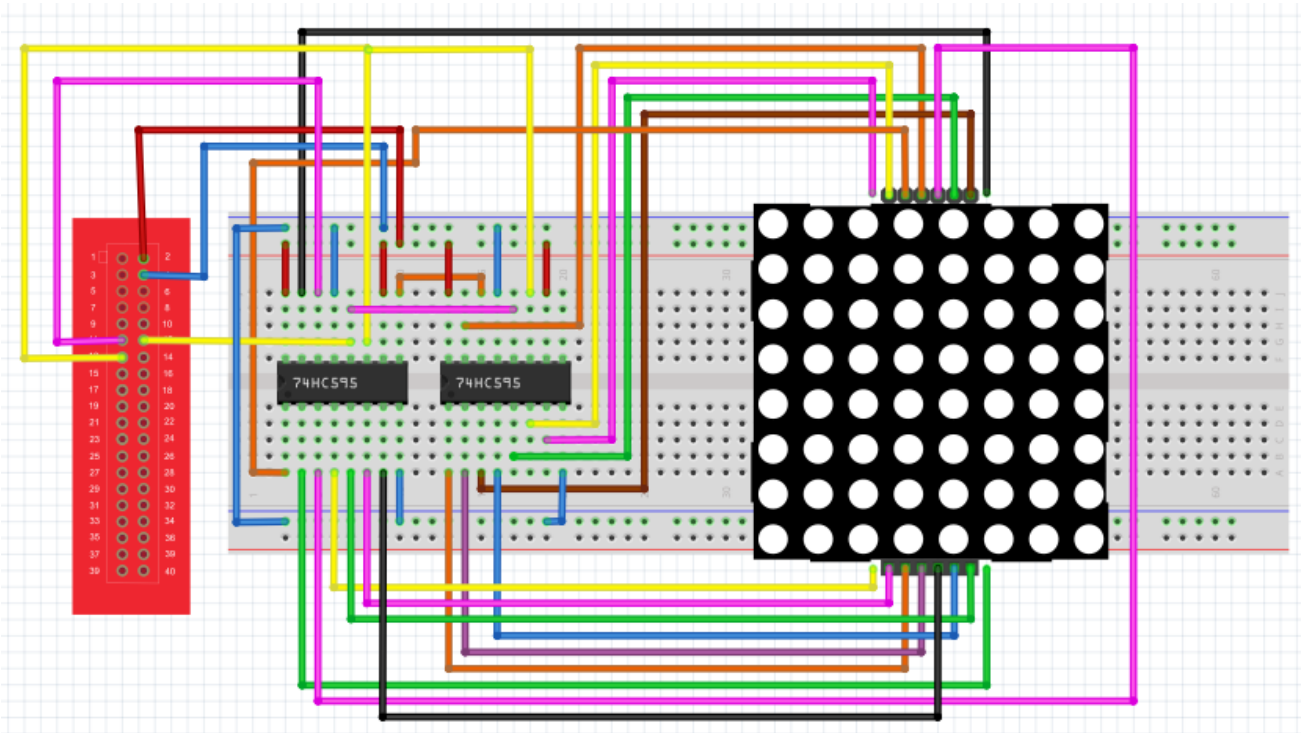
you should set PIN a to low level and PIN (1, 2, 3, 4, 5, 6, 7, 8) to high level, then all the LEDs on the first column will light up.



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

Experimental Procedures

Step 1: Connect the circuit as shown in the following diagram



Step 2: Edit and save the code with vim (see path/Rpi_SuperKit_Code/13_DotMatrix/dot Matrix.c)

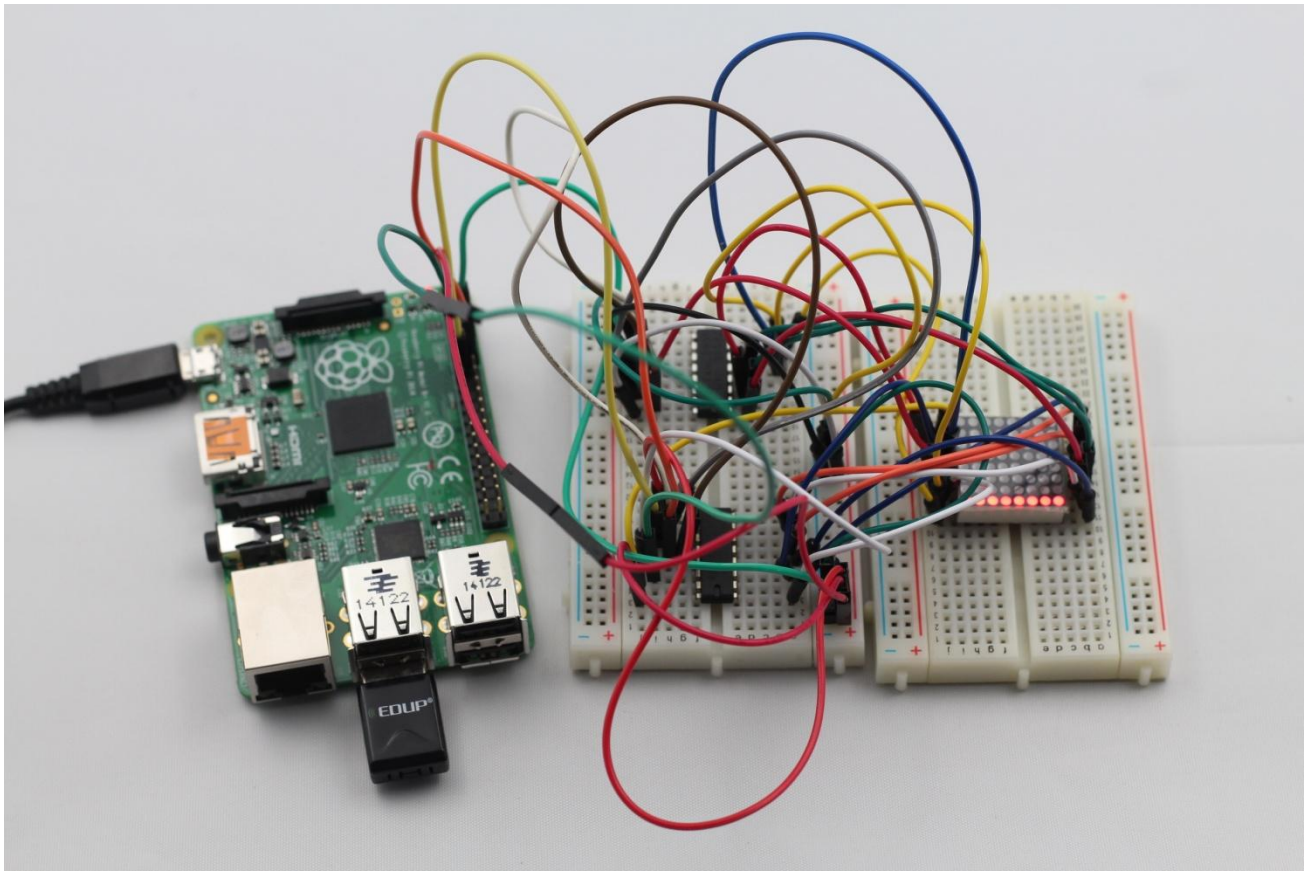
Step 3: Compile the code

```
gcc dotMatrix.c -o dotMatrix -lwiringPi
```

Step 4: Run the program

```
./dotMatrix
```

Press Enter, then you will see LEDs light up as you want.



Summary

Through this section, you have basically mastered the principle of LED dot-matrix and how to program your Raspberry Pi to drive an LED dot-matrix based on 74HC595 cascade.

SUNFOUNDER's Raspberry Pi Lesson 14 — LCD1602

Introduction

In this lesson, we will learn how to use LCD1602 to display character strings.

Experimental Conditions

- 1*Raspberry Pi
- 1*Breadboard
- 1*Network cable (or USB wireless network adapter)
- 1*LCD1602
- 1*Potentiometer
- Jumper wires

Experimental Principle

LCD1602, also called character type LCD1602, is a dot matrix LCD module that specially used to display letters, figures, symbols, and so on. It consists of 16X2 dot matrixes, and each dot matrix is composed of 5X7 or 5X11 character bit. Each character bit can display a character. There is a dot space between each adjacent character bit. And there is a dot space between each row too. The dot space functions as a character space or a line space, and because of this, LCD1602 cannot display graphics very well. It is widely used in pocket instruments and low power application systems due to its micro power consumption, small size, rich contents display, ultra-thin and lightness.

LCD1602 use the standard 16-pin port, among which:

Pin 1(GND): connected to Ground;

Pin 2(Vcc): connected to 5V positive power supply;

Pin 3(Vo): used to adjust the contrast of LCD1602, lowest when connected to positive power supply, highest when connected to ground (you can connect a 10K potentiometer to adjust its contrast when using LCD1602);

Pin 4 (RS): A Register Selection pin, select data register when supplied high level (1), select instruction register when supplied low level (0);

Pin 5 (R/W): A Read/Write signal pin, read signals when supplied high level (1), write signals when supplied low level (0). Since we only need to write data to LCD1602, we directly connect this pin to ground;

Pin 6 (E): An Enable pin, when supplied low level, the LCD module will execute relevant instructions

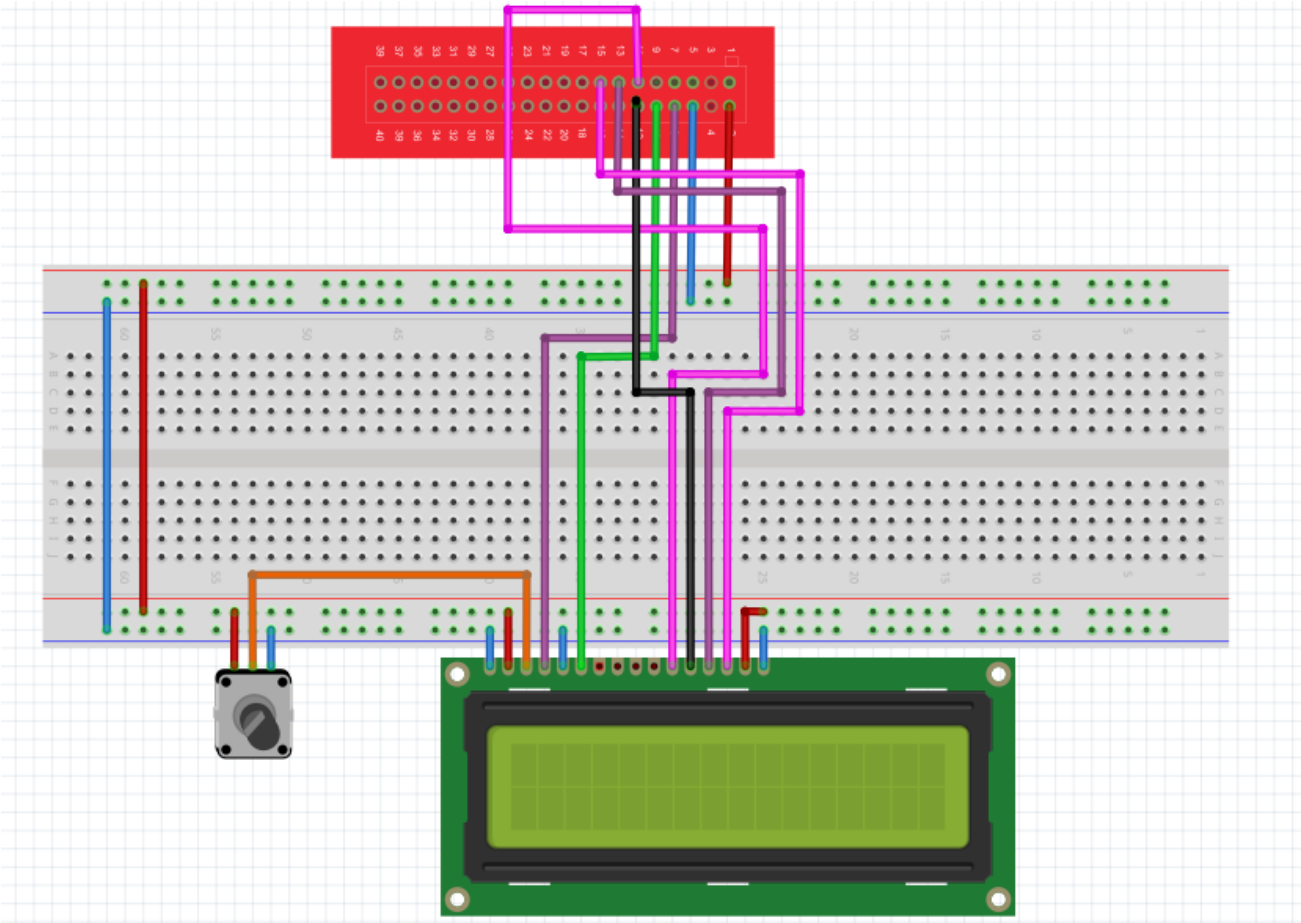
Pin 7 (D0-D7) : Pins that read and write data;

A and K: LCD backlight power source;

LCD1602 has two operation modes: 4-bit mode and 8-bit mode. When the IOs of microprocessor (MCU) is nervous, you can choose 4-bit mode, which only use D4~D7 pins. After connecting the circuit, you can operate LCD1602 by Raspberry Pi.

Experimental Procedures

Step1: Connect the circuit as shown in the following diagram

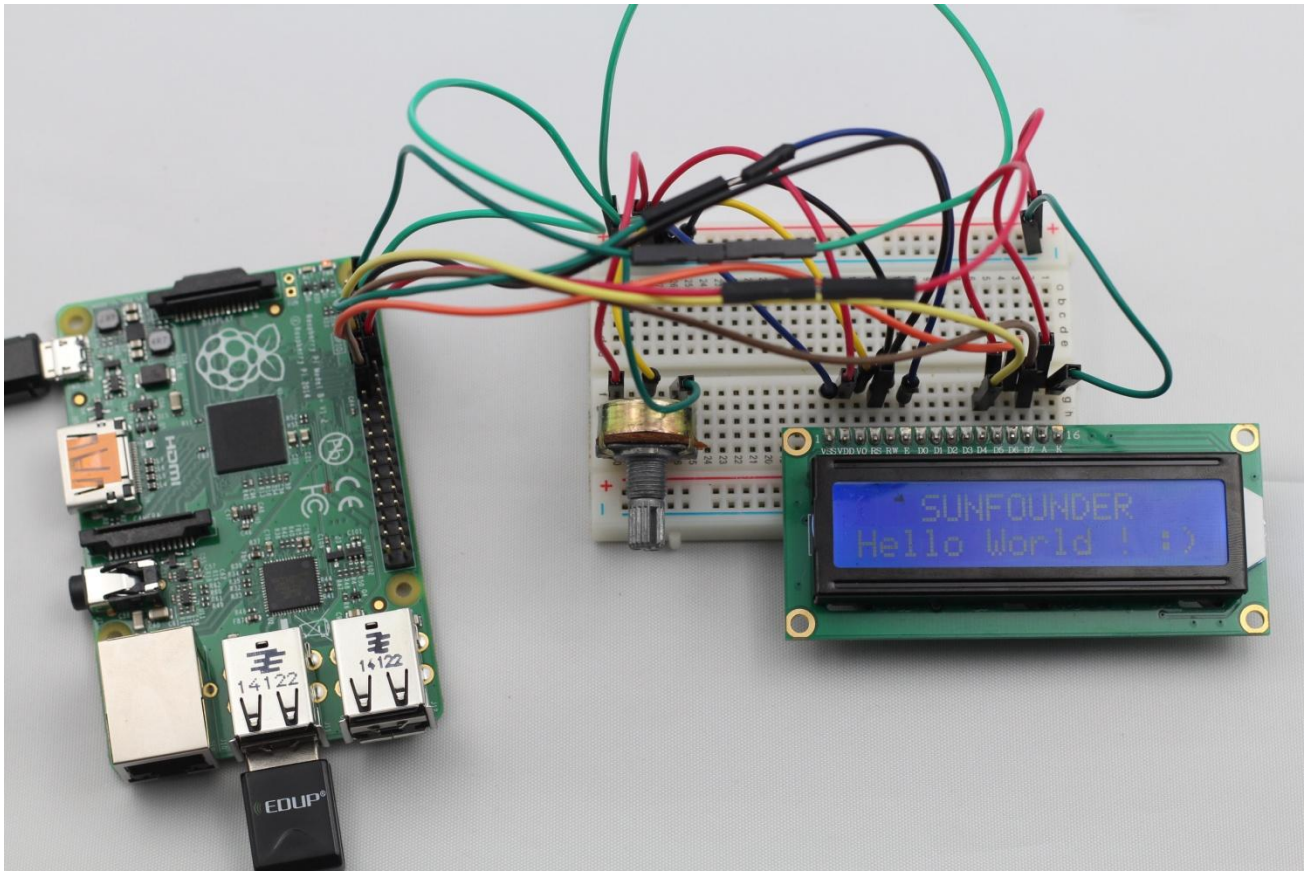


Step 2: Edit and save the code (see path/Rpi_SuperKit_Code/14_LCD1602/Adafruit_CharLCD.py)

Step 3: Run the program

./Adafruit_CharLCD.py

Press Enter, and you will see two lines of information being displayed on the LCD1602. The first line display our logo "**SUNFOUNDER**" and the second line display "**Hello World ！：)** ". The real effect is shown as follow:



Further Exploration

In this experiment, we drive LCD1602 in 4-bit mode. I hope you can program by yourselves to drive LCD1602 in 8-bit mode.

Summary

Through this lesson, you have basically mastered the principle and programming implementation for LCD1602 based on Raspberry Pi. I hope you can create many more fun works on the basis of Raspberry Pi.

SUNFOUNDER's Raspberry Pi Lesson 15 — Acceleration Sensor

Introduction
In this lesson, we will learn how to use acceleration sensor ADXL345.

Experimental Conditions
- 1*Raspberry Pi
- 1*Breadboard
- 1*ADXL345 module
- Jumper wires

Experimental Principles
The ADXL345 is a small, thin, low power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 g. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I2C digital interface.

The ADXL345 is well suited to measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (4 mg/LSB) enables measurement of inclination changes less than 1.0°.

The excellent sensitivity (3.9mg/LSB @2g) provides high-precision output up to ±16g.

In this experiment, we use I2C digital interface.

Experimental Procedures
Step 1: Connect the circuit according to the following method

Raspberry Pi		ADXL345 Module
GND	----------	GND
3.3V	----------	3.3V
SCL0	----------	SCL
SDA0	----------	SDA
CS	----------	3.3V
SDO	----------	GND

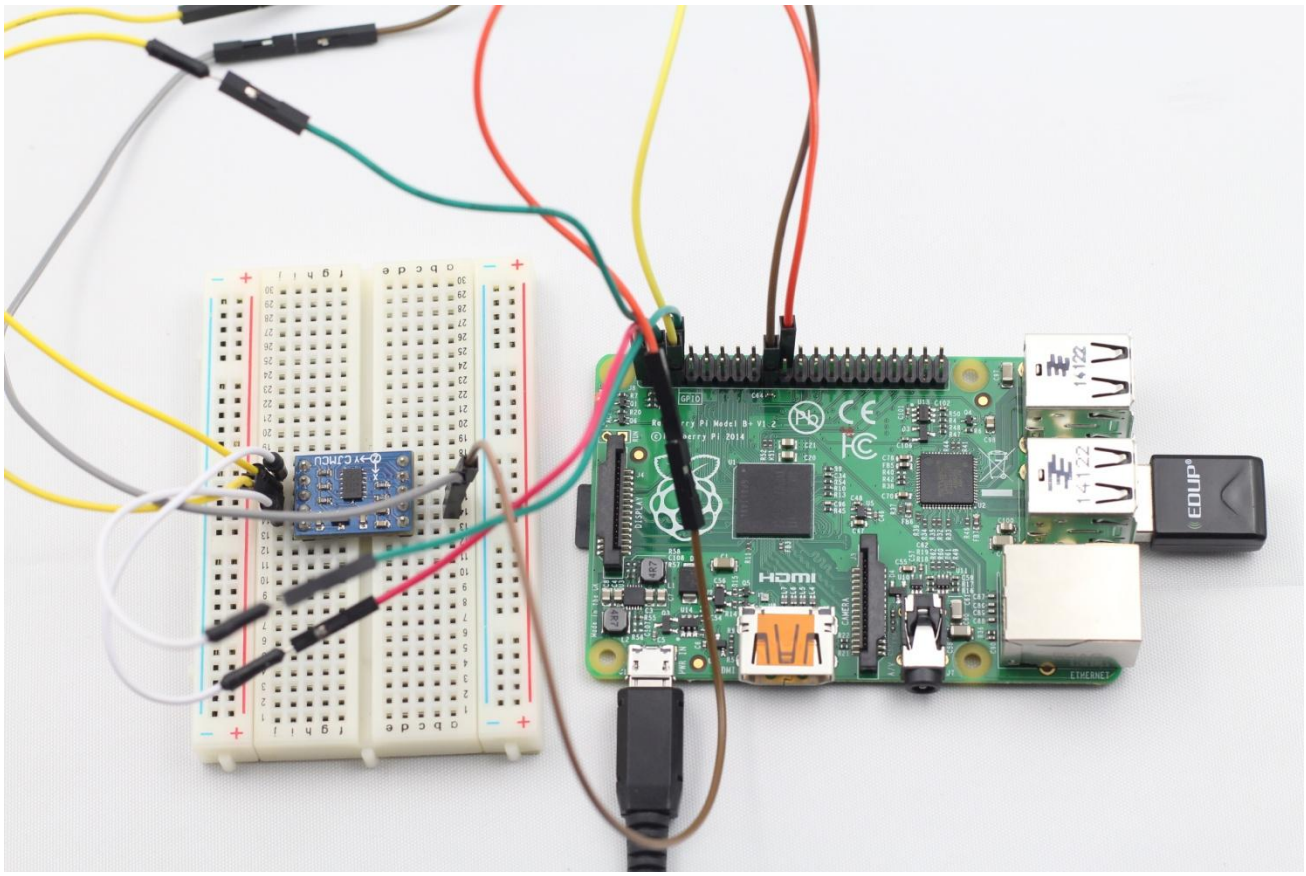
Step 2: Edit and save the code with vim (see path/Rpi_SuperKit_Code/15_ADXL345/adxl345.c)

Step 3: Compile the code

```
gcc adxl345.c -o adxl345 -lwiringPi
```

Step 4: Run the program

```
./adxl345
```



Summary

Trough this experiment, we have learnt I2C interface programming realization for Raspberry Pi.

Technology Production — Fingertip Touch Switch

Statement: The two experiments in this lesson have nothing to do with the Raspberry Pi, and we just take power from the Raspberry Pi.

Introduction

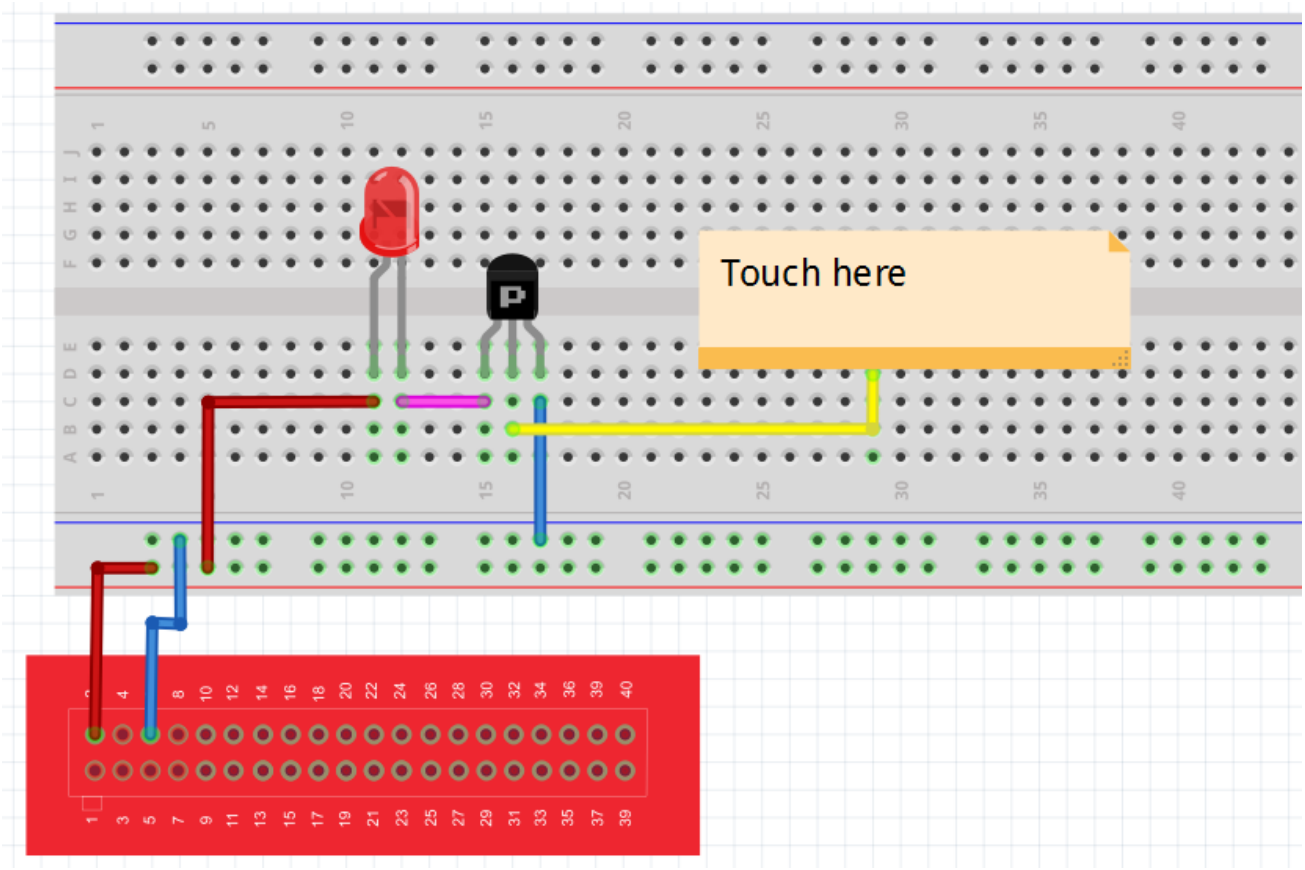
A touch switch is a kind of new product with the development of science and technology. It generally refers to a wall switch that is designed by the principle of touch sensitive chip and can replace traditional mechanical button type wall switch. Compared with traditional switches, touch switches have unparalleled advantages, such as more intelligent and easy to use. It is a very popular decorative switch in current home product. In this experiment, we will use commonly used components to assemble two types of touch switches with low cost.

Experimental Conditions

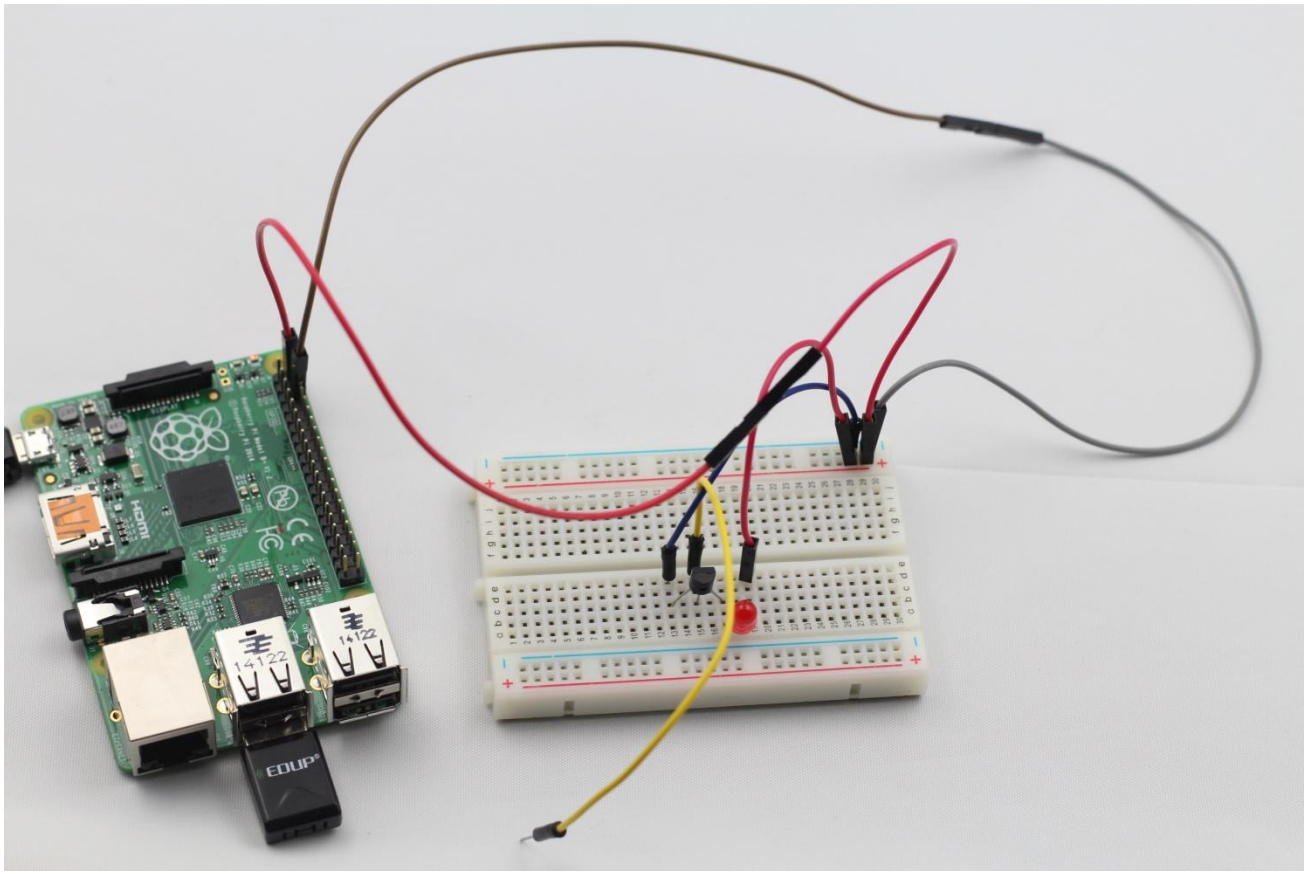
- 1*Raspberry Pi
- 1*Breadboard
- 1*Transistor(NPN)
- 1*NE555
- 2*Resistor(1*10K, 1*1K)
- 2*Capacitor(100nF)
- 1*LED
- Jumper wires

Experimental Principles and Procedures

Experimental 1: A touch switch based on a transistor

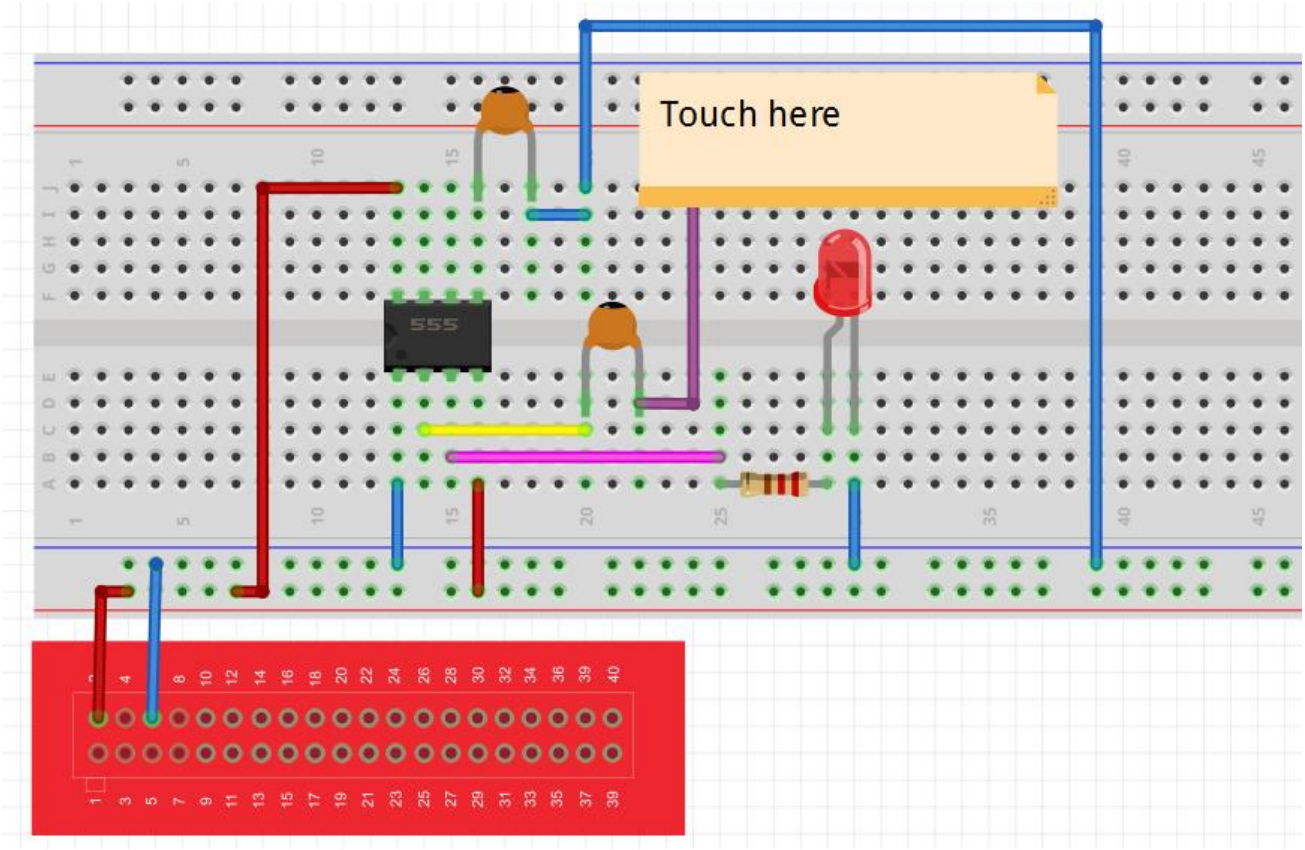


After connecting the circuit, when you touch the base of the transistor, you will see the LED light up. When you release your hand, you will see the LED go out.



Principle: when you touch the base of the transistor, the transistor gradually transfer from cut-off state to saturation conducting state. As a result, the LED will light up.

Experiment 2: A touch switch based on NE555



After connecting the circuit according to the above diagram, when you touch the other electrode of the ceramic capacitor connected to the second pin of NE555, you will see the LED light up. When you release your hand, the LED will go out.

Principle: In this experiment, connect NE555 in monostable circuit. Since usually there is no voltage on touch terminal, after the capacitor connected to pin 6 discharges by pin 7 and pin 3 outputs low level, the LED will be off. When you touch the other electrode of the capacitor connected to pin 2, human clutter signals will transfer to the trigger terminal of NE555 and make the output (pin 3) high, then the LED will light up. At the same time, pin 7 of NE555 will cut off internally, and the power source will supply power for the capacitor connected to pin 6 via a resistor, and that is the beginning of the timing. For the capacitance of the capacitor we choose is small, we cannot see obvious delay effects.

Advanced application of Raspberry Pi — Controlling an LED Based on Web

Introduction

The reason why the Raspberry Pi is so powerful is its networking function and numerous open source web application support. We can easily use it to realize our own Internet of Things system. In this lesson, we will control an LED based on web with the Raspberry Pi. You can use your mobile phone, tablet pc or computer to control the LED too, as long as it is in the same Local Area Network (LAN) with your Raspberry Pi.

Experimental Conditions

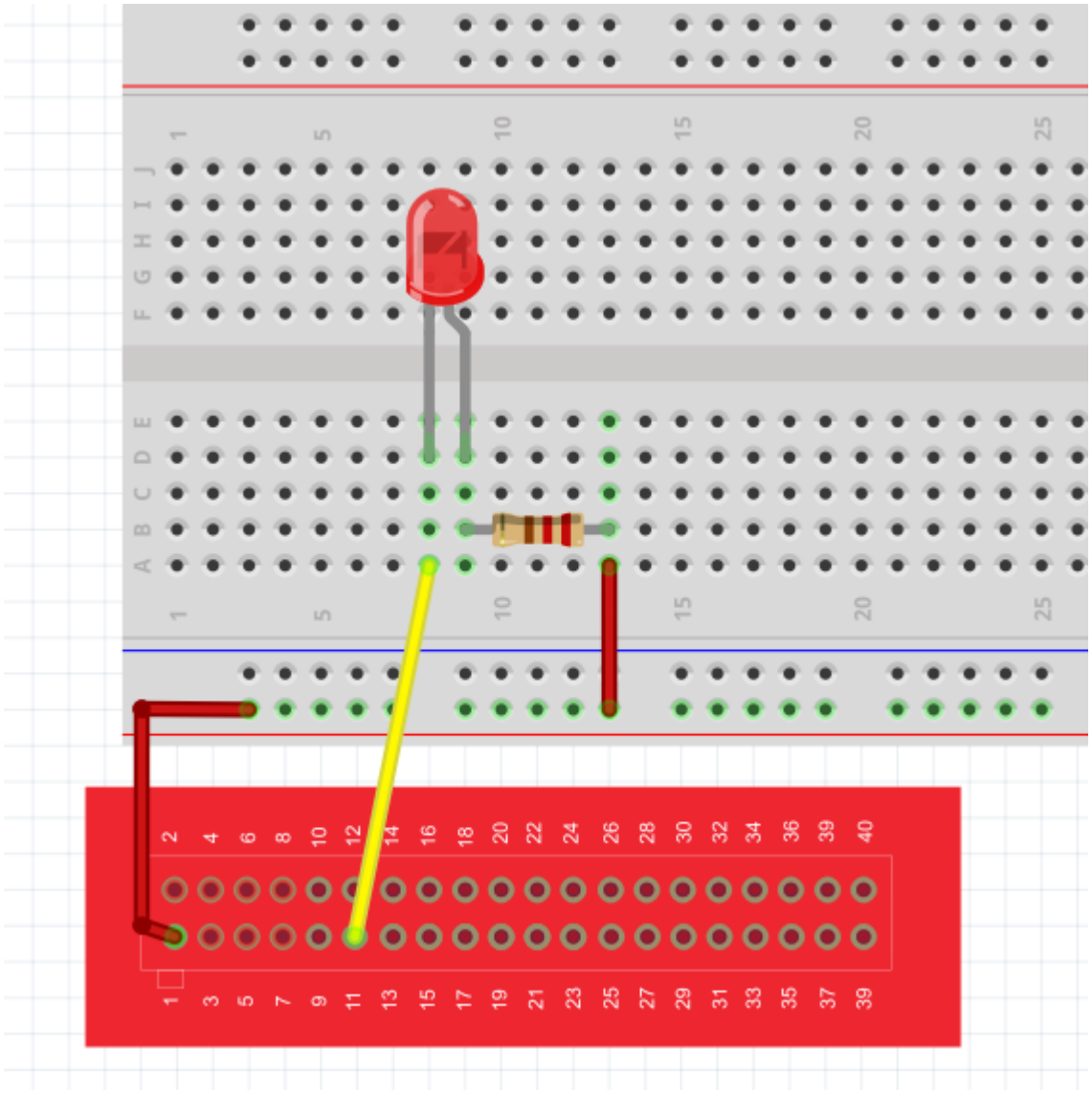
- 1*Raspberry Pi
- 1*Breadboard
- 1*Network cable (or USB wireless network adapter)
- 1*LED
- 1*Resistor (220Ω)
- Jumper wires

Experimental Principle

WebIOPi is a fully integrated Internet of Things framework for the Raspberry Pi. Webiopi is a lightweight web server program on Raspberry Pi platform. It provides users a solution that can access Raspberry Pi remotely. Based on this, you can remotely operate your Raspberry Pi and control the hardware connected to it by a browser or an APP.

Experimental Procedures

1. Connect the circuit as shown in the following diagram



2. Install WebIOPi

wget http://webiopi.googlecode.com/files/WebIOPi-0.5.3.tar.gz

tar xvzf WebIOPi-0.5.3.tar.gz

cd WebIOPi-0.5.3

sudo ./setup.sh

3. Set or modify the password of WebIOPi

sudo webiopi-passwd

Username: webiopi

If you do not modify the password, the default password is: raspberry

4. Start WebIOPi

sudo python -m webiopi 8000

You can change the port by yourself, the default port is 8000

5. Run WebIOPi in background, or it will be killed when you press ctrl+c

sudo /etc/init.d/webiopi start

and

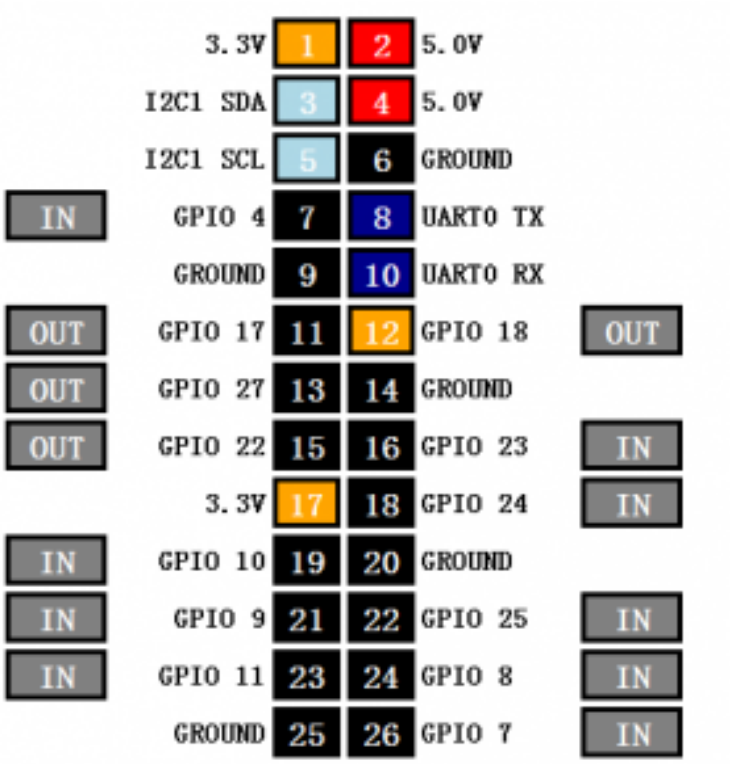
sudo /etc/init.d/webiopi stop

6. Set WebIOPi to start with the system

sudo update-rc.d webiopi defaults

7. Open IP address access management interface with a browser

http://192.168.1.106:8000/webiopi/



Control method:

1) Click outside IN/OUT to switch between the input and output mode of GPIOs
2) In output mode, click inside figures to switch output between high and low. After connecting the LED to GPIO17, when you click block 11, you will see the states of the LED changed
3) In input mode, inside figures represent the input states of GPIO

Summary

In this lesson, we have learnt how to build Internet of Things for WebIOPi platform based on Raspberry Pi and learnt how to control our device remotely based on this platform. I hope you can learn WebIOPi in depth and use your brain to make many more fun things.

If you have any questions, please send email to support@sunfounder.com. We will reply you asap!

For more tutorials, please visit our website www.sunfounder.com.

If you have great works, welcome to our website to publish them to share with others.

Thanks!

Sunfounder technical support team

SUNFOUNDER

OPEN SOURCE

ELECTRONICS

THE BEST DIY

COMPONENTS

FOR ARDUINO

WWW.SUNFOUNDER.COM