8.4-Digital-Segment-Disply

Introduction

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

Hardware Required

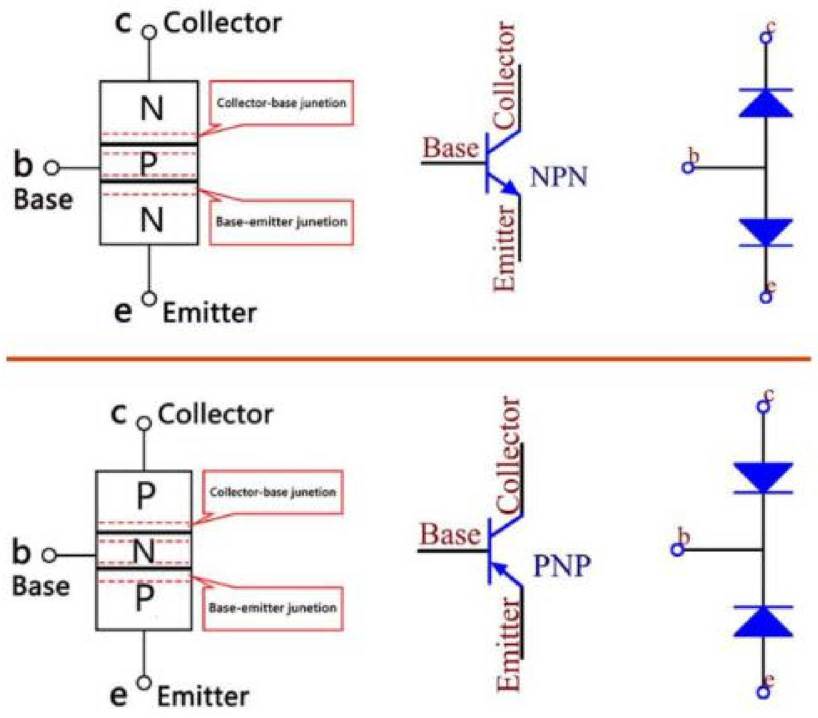
- ✓ 1 * Raspberry Pi
- ✓ 1 * T-Extension Board
- ✓ 1 * 4-Digital 7-Segment Display
- ✓ 1 * 40-pin Cable
- ✓ 4 * S8550 PNP Transistor
- ✓ 1 * 74HC595
- ✓ Several Jumper Wires
- ✓ 1 * Breadboard
- ✓ 8 * Resistor(220Ω)
- ✓ 4 * Resistor(1KΩ)

Principle

Transistor

The transistor is a semiconductor device that controls current by current. It functions by amplifying weak signal to larger amplitude signal and is also used as a non-contact switch. A transistor is a three-layer structure composed of P-type or N-type semiconductors. They form the three regions internally. The thinner in the middle is the base region; the other two are all N-type or P-type ones - the smaller region with intense majority carriers is the emitter region, when the other one is the collector region. This composition enables the transistor to be an amplifier.

From these three regions, three poles are generated respectively, which are base (b), emitter (e), and collector (c). They form two P-N junctions, namely, the emitter junction and collection junction. The arrow in the circuit symbol indicates the direction of emitter junction. Transistors can be divided into two kinds: the NPN and PNP one. The former is made of two N-type semiconductors and one P-type and that the latter is the opposite. See the figure below.

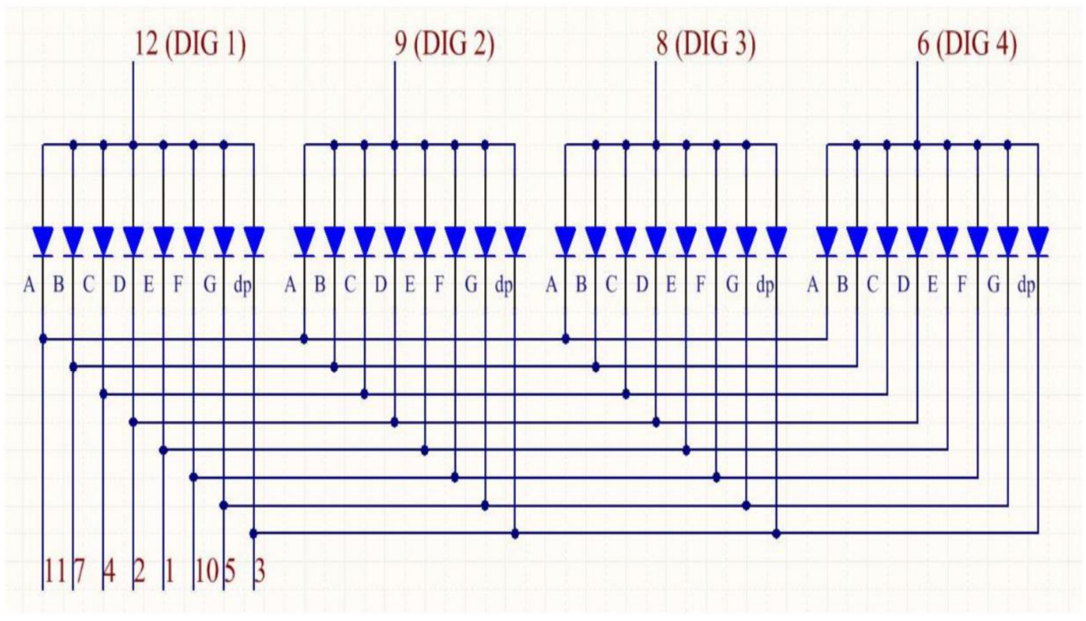


When a High level signal goes through an NPN transistor, it is energized. But a PNP one needs a Low level signal to manage it. Both types of transistor are frequently used for contactless switches, just like in this experiment.

4-Digital -Segment Display

The 4-digtal 7-segment display works independently. It uses the principle of human visual persistence to quickly display the characters of each 7-segment in a loop to form continuous strings.

For example, when "1234" is displayed on the display, "1" is displayed on the first 7-segment, and "234" is not displayed. After a period of time, the second 7-segment shows "2", the 1st 3th 4th of 7-segment does not show, and so on, the four digital display show in turn. This process is very short (typically 5ms), and because of the optical afterglow effect and the principle of visual residue, we can see four characters at the same time.



Display Codes

To help you get to know how 7-segment displays(Common Anode) display Numbers, we have drawn the following table. Numbers are the number 0-F displayed on the 7- segment display; (DP) GFEDCBA refers to the corresponding LED set to 0 or 1, For example, 11000000 means that DP and G are set to 1, while others are set to 0.

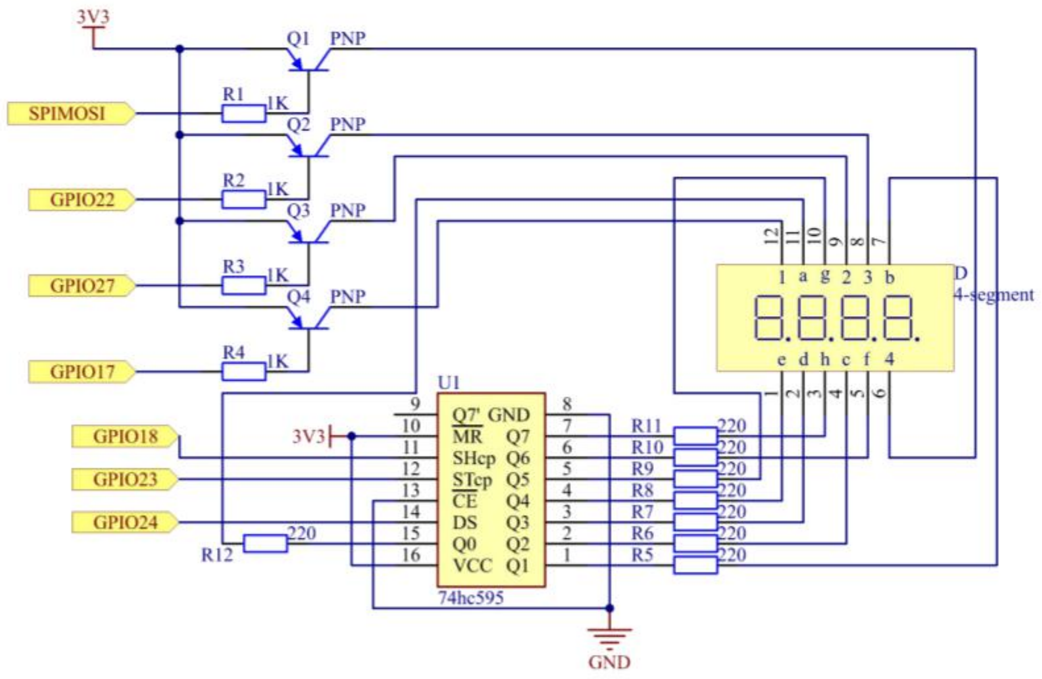
Therefore, the number 0 is displayed on the 7-segment display, while HEX Code corresponds to hexadecimal number.

Numbers	Common Cathode		Numbers	Common Cathode	
	(DP)GFEDCBA	Hex Code		(DP)GFEDCBA	Hex Code
0	11000000	0xc0	A	10001000	0x88
1	11111001	0xf9	B	10000011	0x83
2	10100100	0xa4	C	11000110	0xc6
3	10110000	0xb0	D	10100001	0xa1
4	10011001	0x99	E	10000110	0x86
5	10010010	0x92	F	10001110	0x8e
6	10000010	0x82	.	01111111	0x7f
7	11111000	0xf8			
8	10000000	0x80			
9	10010000	0x90			

Schematic Diagram

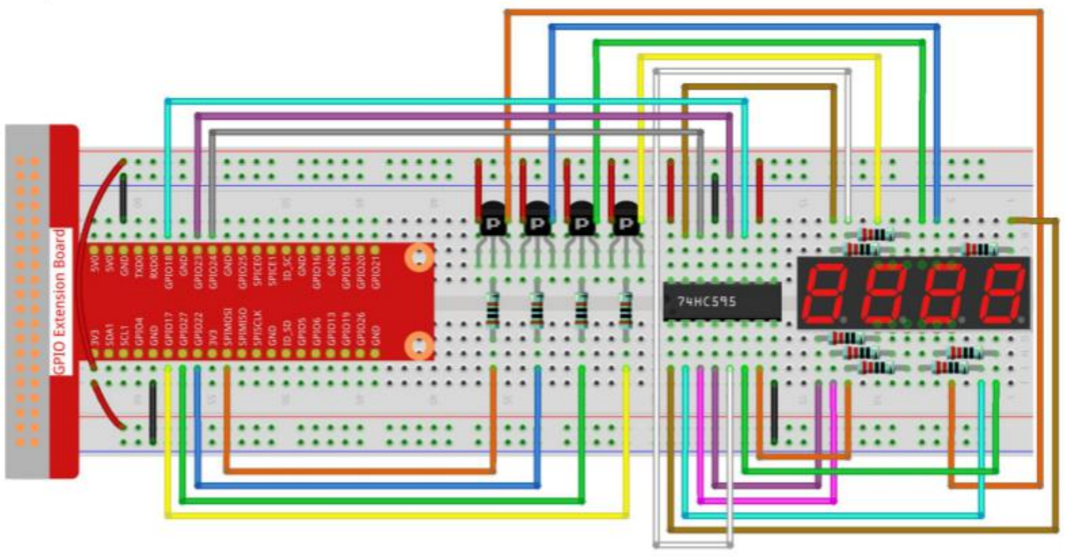
T-Board Name	physical	wiringPi	BCM
GPIO17	Pin 11	0	17

GPIO27	Pin 13	2	27
GPIO22	Pin 15	2	22
SPIMOSI	Pin 19	12	10
GPIO18	Pin 12	1	18
GPIO23	Pin 16	4	23
GPIO24	Pin 18	5	24



Experimental Procedures

Step 1: Build the circuit.



For C Language Users

Step 2: Go to the folder of the code.

```
cd /home/pi/REXQualis_Raspberry_Pi_Complete_Starter_Kit/C/8.4-Digital-Segment
```

Step 3: Compile the code.

```
gcc 8.4-Digital-Segment.c -o 4-Digital-Segment.out -lwiringPi
```

Step 4: Run the executable file.

```
sudo ./4-Digital-Segment.out
```

After the code runs, the program takes a count, increasing by 1 per second, and the 4-digit 7-segment display

Code

```
#include <wiringPi.h>
#include <stdio.h>
#include <wiringShift.h>
#include <signal.h>
#include <unistd.h>
#define SDI        5     //serial data input(DS)
#define RCLK       4     //memory clock input(STCP)
#define SRCLK      1      //shift register clock input(SHCP)
const int placePin[]={0,2,3,12};            // Define 4 digit's common pin.
```

```
// A segment code array from 0 to 9 in Hexadecimal (Common anode)
unsigned char number[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
int counter = 0;//the number will be displayed.
//select the place of the value, this is only one place that will be enable each time.
//the parameter digit is optional for 0001,0010,0100,1000
void selectPlace(int digit){
    digitalWrite(placePin[0],!(digit&0x08));
    digitalWrite(placePin[1],!(digit&0x04));
    digitalWrite(placePin[2],!(digit&0x02));
    digitalWrite(placePin[3],!(digit&0x01));
}
void hc595_shift(int8_t data){//To assign 8 bit value to 74HC595's shift register
    int i;
    for(i = 0; i < 8; i++){
        digitalWrite(SDI, 0x80 & (data << i));
        digitalWrite(SRCLK,1);
        delay(1);
        digitalWrite(SRCLK,0);
    }
        digitalWrite(RCLK,1);
        delay(1);
        digitalWrite(RCLK,0);
}
void display(int counter){   //display the number
    hc595_shift(0xff);
    selectPlace(0x01);
    hc595_shift(number[counter%10]);
//display the number on the single digit of the value
    delay(1);
    hc595_shift(0xff);
```

```
        selectPlace(0x02);
        hc595_shift(number[counter%100/10]);
        delay(1);
        hc595_shift(0xff);
        selectPlace(0x04);
        hc595_shift(number[counter%1000/100]);
        delay(1);
        hc595_shift(0xff);
        selectPlace(0x08);
        hc595_shift(number[counter%10000/1000]);
        delay(1);
}
void timer(int sig){            //Timer function
        if(sig == SIGALRM){    //If the signal is SIGALRM, the value of counter plus 1, and
update the number displayed by 7-segment display
                counter ++;
                alarm(1);                   //set the next timer time
                printf("counter : %d \n",counter);
        }
}
int main(void)
{
        if(wiringPiSetup() == -1){ //when initialize wiring failed,print messageto screen
                printf("setup wiringPi failed !");
                return 1;
        }
        pinMode(SDI,OUTPUT);         //set the pin connected to74HC595 for output mode
        pinMode(RCLK,OUTPUT);
        pinMode(SRCLK,OUTPUT);
//set the pin connected to 7-segment display common end to output mode
```

```
    int i;
    for(i=0;i<4;i++){
        pinMode(placePin[i],OUTPUT);
        digitalWrite(placePin[i],HIGH);
    }
    signal(SIGALRM,timer);   //configure the timer
    alarm(1);                //set the time of timer to 1s
    while(1){
        display(counter);    //display the number counter
    }
    return 0;
}
```

Code Explanation

```
const int placePin[]={0,2,3,12};
```

These four pins control the common anode pins of the four-digit 7-segment displays.

```
unsigned char number[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
```

A segment code array from 0 to 9 in Hexadecimal (Common anode).

```
void selectPlace(int digit){
    digitalWrite(placePin[0],!(digit&0x08));
    digitalWrite(placePin[1],!(digit&0x04));
    digitalWrite(placePin[2],!(digit&0x02));
    digitalWrite(placePin[3],!(digit&0x01));
}
```

Select the place of the value. there is only one place that should be enable each time.

The enabled place will be written low. For example, if digit=0x01:

!(0x01&0x08)=1,

!(0x01&0x04)=1,

!(0x01&0x02)=1,

!(0x01&0x01)=0.

Write 0 into placePin[3](GPIO12), while the corresponding pins of placePin[] are connected to a PNP transistor,

Then the fourth common anode pin DIG4 of the 4-digit segment display corresponding to placePin[3](GPIO12) will get a high level to light up the 4th 7-segment display.

Similarly, placePin[0]~[2] are written to 1. After the signal passes through the PNP transistor, these three corresponding common anode pins will get a low level and then the three 7-segment displays will not light up.

```
void display(int counter){   //display the number
    hc595_shift(0xff);
    selectPlace(0x01);
    hc595_shift(number[counter%10]);
    delay(1);
```

The function display() is used to set the number displayed on the 4-digit 7-segment display

hc595_shift(0xff) ： write in 11111111 to turn off these eight LEDs on 7-segment display so as to clear the displayed content.

selectPlace(0x01)：Light up the fourth 7-segment display.

hc595_shift(number[counter%10]) ：the number in the single digit of counter will display on the forth segment.

```
signal(SIGALRM,timer);
```

This is a system-provided function, the prototype of code is:

```
sig_t signal(int signum,sig_t handler);
```

After executing the signal(), once the process receives the corresponding signum (in this case SIGALRM), it immediately pauses the existing task and processes the set function (in this case timer(sig)).

```
alarm(1);
```

This is also a system-provided function. The code prototype is

```
unsigned int alarm (unsigned int seconds);
```

It generates a SIGALRM signal after a certain number of seconds.

```
void timer(int sig){
    if(sig == SIGALRM){
        counter ++;
        alarm(1);
    }
}
```

We use the functions above to implement the timer function.

After the alarm() generates the SIGALRM signal, the timer function is called. Add 1 to counter, and the function, alarm(1) will be repeatedly called after 1 second.

For Python Language Users

Step 2: Go to the folder of the code.

```
cd /home/pi/REXQualis_Raspberry_Pi_Complete_Starter_Kit/Python
```

Step 3: Run the executable file.

```
sudo python3 8.4-Digital-Segment.py
```

After the code runs, the program takes a count, increasing by 1 per second, and the 4 digit display displays the count.

Code

```
import RPi.GPIO as GPIO
import time
import threading
#define the pins connect to 74HC595
SDI    = 18        #serial data input(DS)
RCLK   = 16        #memory clock input(STCP)
SRCLK = 12         #shift register clock input(SHCP)
number = (0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90)
```

```
placePin = (11,13,15,19)

counter = 0               # the number will be displayed

t = 0                     # define the Timer object

def setup():

    GPIO.setmode(GPIO.BOARD)      # Number GPIOs by its physical location

    GPIO.setup(SDI, GPIO.OUT)        # Set pin mode to output

    GPIO.setup(RCLK, GPIO.OUT)

    GPIO.setup(SRCLK, GPIO.OUT)

    for pin in placePin:

        GPIO.setup(pin,GPIO.OUT)

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

    for bit in range(0, 8):

        GPIO.output(SDI, 0x80 & (dat << bit))

        GPIO.output(SRCLK, GPIO.HIGH)

        time.sleep(0.001)

        GPIO.output(SRCLK, GPIO.LOW)

GPIO.output(RCLK, GPIO.HIGH)

    time.sleep(0.001)

    GPIO.output(RCLK, GPIO.LOW)

def selectPlace(digit):   # select the place of the value

    GPIO.output(placePin[0],GPIO.LOW if (digit&0x08) else GPIO.HIGH)

    GPIO.output(placePin[1],GPIO.LOW if (digit&0x04) else GPIO.HIGH)

    GPIO.output(placePin[2],GPIO.LOW if (digit&0x02) else GPIO.HIGH)

    GPIO.output(placePin[3],GPIO.LOW if (digit&0x01) else GPIO.HIGH)

def display(counter):     #display the number

    hc595_shift(0xff)     #clean up the display

    selectPlace(0x01)     #Select place

    hc595_shift(number[counter%10]) #display the number on the single digit of the
value

    time.sleep(0.003)
    hc595_shift(0xff)
    selectPlace(0x02)
    hc595_shift(number[counter%100//10])
    time.sleep(0.003)
    hc595_shift(0xff)
    selectPlace(0x04)
    hc595_shift(number[counter%1000//100])
    time.sleep(0.003)
    hc595_shift(0xff)
    selectPlace(0x08)
    hc595_shift(number[counter%10000//1000])
    time.sleep(0.003)
def timer():            #timer function
    global counter
    global t
    t = threading.Timer(1.0,timer)       #set the time again
    t.start()                              #Start timing
    counter+=1
    print ("counter : %d"%counter)
def loop():
    global t
    global counter
    t = threading.Timer(1.0,timer)       #set the timer
    t.start()                              # Start timing
    while True:
        display(counter)                   # display the number counter
def destroy():    # When "Ctrl+C" is pressed, the function is executed.
    global t
    GPIO.cleanup()
    t.cancel()        #cancel the timer
```

```
if __name__ == '__main__': # Program starting from here
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

Code Explanation

```
number = (0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90)
```

A segment code array from 0 to 9 in Hexadecimal (common anode).

```
placePin = (11,13,15,19)      # Define the pin of 7-segment display common end
```

These four pins control the common anode of the 4-digit 7-segment display.

```
def selectPlace(digit):    # select the place of the value
    GPIO.output(placePin[0],GPIO.LOW if (digit&0x08) else GPIO.HIGH)
    GPIO.output(placePin[1],GPIO.LOW if (digit&0x04) else GPIO.HIGH)
    GPIO.output(placePin[2],GPIO.LOW if (digit&0x02) else GPIO.HIGH)
    GPIO.output(placePin[3],GPIO.LOW if (digit&0x01) else GPIO.HIGH)
```

Select the place of the value. there is only one place that should be enable each time. The enabled place will be written low.

GPIO.LOW if (digit&0x08) else GPIO.HIGH): If digit&0x08 is equal to 1, it is GPIO.LOW; otherwise, it's GPIO.HIGH.

For instance, if digit=0x01 and the selectPlace(digital) function is run, only placePin[3](physical pin19) can get a low level that will let the PNP transistor activated, and then the fourth 7-segment display will get a high level which is to light up the fourth digit 7-segment display.

Similarly, placePin[0]~[2] are written to 1. After the signal passes through the PNP transistor, these three corresponding common anode pins will get a low level and then the three 7-segment displays will not light up.

```
def display(counter):     #display the number
    hc595_shift(0xff)    #clean up the display
    selectPlace(0x01)    #Select place
```

hc595_shift(number[counter%10]) #display the number on the single digit of the value

time.sleep(0.003)

The display() function is used to set the number displayed on the 4-digit 7-segment Dispaly.

hc595_shift(0xff)： write in 0xff (binary system：1111111) so that the eight LEDs on the 7-segment Dispaly will turn off so as to clear the displayed content.

selectPlace(0x01)：Light up the fourth 7-segment display .

hc595_shift(number[counter%10]) ：the number in the single digit of counter will display on the forth segment.

```
    t = threading.Timer(1.0,timer)
     t.start()
```

The module, threading is the common threading module in Python， and Timer is the subclass of it.

The prototype of code is:

```
class threading.Timer(interval, function, args=[], kwargs={})
```

After the interval, the function will be run. Here, the interval is 1.0， and the function is timer().

t.start () means the Timer will start at this point.

```
def timer():            #timer function
    global counter
    global t
    t = threading.Timer(1.0,timer)        #set the time again
    t.start()                               #Start timing
    counter+=1
```

After Timer reaches 1.0s, the Timer function is called; add 1 to counter, and the Timer is used again to execute itself repeatedly every second.

Phenomenon Picture