!Note:
The maximum continuous input and output voltage of the Raspberry Pi's GPIO pin is 3.3V. Do not connect it directly with other electronic components, otherwise it will damage the Raspberry Pi.

Step 1:Create a file to store the project

```
nano gpio.c
```

Step 2: Writing code

```
#include<stdio.h>
#include<wiringPi.h>            //Import wiringPi.h library

#define GPIO0 0         //Define GPIO0 as pin 0 of wPi, corresponding to GPIO0
int main()
{
        printf("This is an example of controlling the high and low levels of GPIO pin output\n");
        wiringPiSetup();                        //Initialize wiringPi
        pinMode(GPIO0,OUTPUT);                  //Set GPIO0 to output mode
        printf("Set GPIO0 : H\n");
        digitalWrite(GPIO0,HIGH);       //GPIO0 output high level
        //printf("Set GPIO0 : L\n");
        //digitalWrite(GPIO0,LOW);      //GPIO0 output low level
        while(1)
        {

        }
        return 0;
}
```

After writing, press Ctrl + X to exit this file.
The system will prompt you whether you need to save, press Y to save and exit.

Step 3: Compile this .c file.

```
gcc gpio.c -o gpio -lwiringPi
```

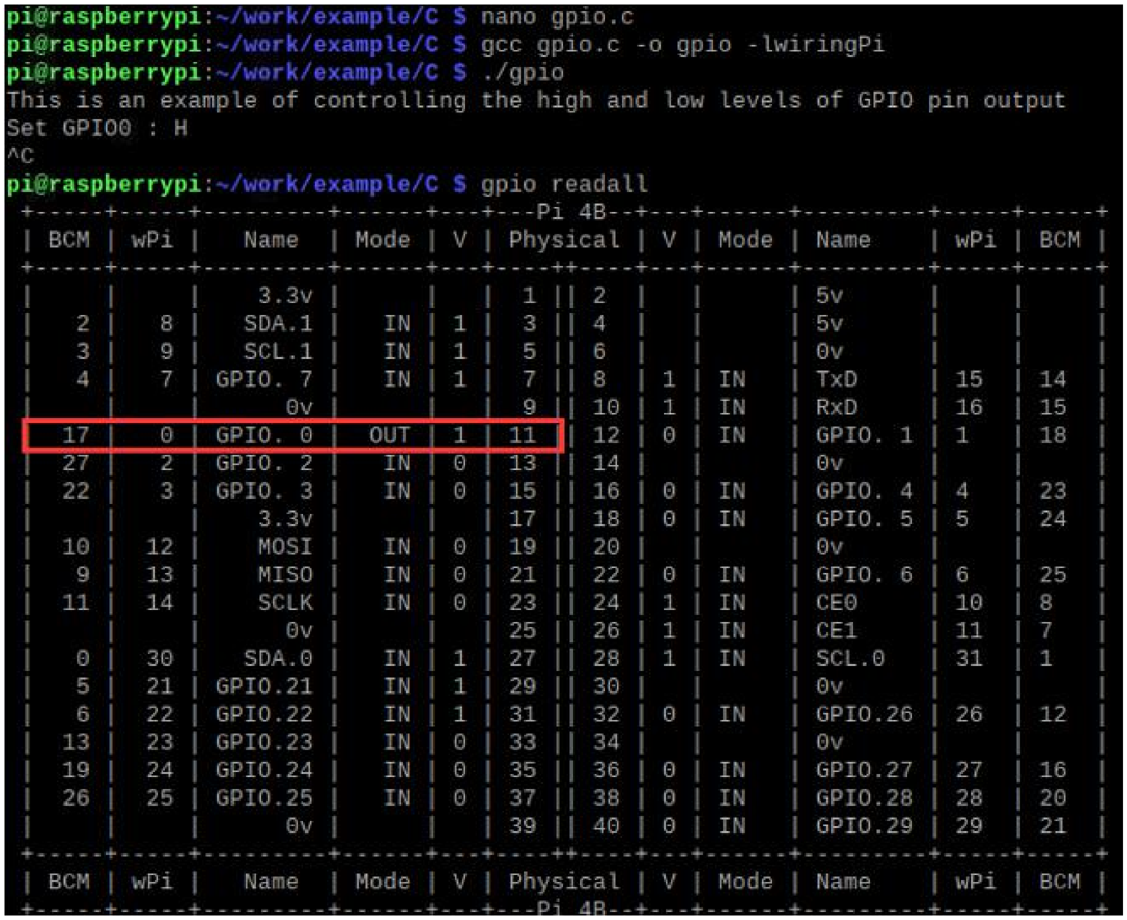
Step 4: Run this code

```
./gpio
```

Step 5: Press Ctrl + C

Step 6: Check the pin status of the Raspberry Pi.

```
gpio readall
```



```
pi@raspberrypi:~/work/example/C $ nano gpio.c
pi@raspberrypi:~/work/example/C $ gcc gpio.c -o gpio -lwiringPi
pi@raspberrypi:~/work/example/C $ ./gpio
This is an example of controlling the high and low levels of GPIO pin output
Set GPIO0 : H
^C
pi@raspberrypi:~/work/example/C $ gpio readall
 +-----+-----+---------+------+---+---Pi 4B--+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5v      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | IN   | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | IN   | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |  OUT | 1 | 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 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | IN   | 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 4B--+---+------+---------+-----+-----+
```

We can know that the GPIO0 is OUT mode, and V (state) is 1 (high level).

Step 7: Modify code

```
#include<stdio.h>
#include<wiringPi.h>            //Import wiringPi.h library

#define GPIO0 0         //Define GPIO0 as pin 0 of wPi, corresponding to GPIO0
int main()
{
        printf("This is an example of controlling the high and low levels of GPIO pin outp
ut\n");
        wiringPiSetup();                        //Initialize wiringPi
        pinMode(GPIO0,OUTPUT);                  //Set GPIO0 to output mode
        //printf("Set GPIO0 : H\n");
        //digitalWrite(GPIO0,HIGH);     //GPIO0 output high level
        printf("Set GPIO0 : L\n");
        digitalWrite(GPIO0,LOW);        //GPIO0 output low level
        while(1)
        {
        }
        return 0;
}
```

After writing, press Ctrl + X to exit this file.
The system will prompt you whether you need to save, press Y to save and exi

Step 3: Compile this .c file.

```
gcc gpio.c -o gpio -lwiringPi
```

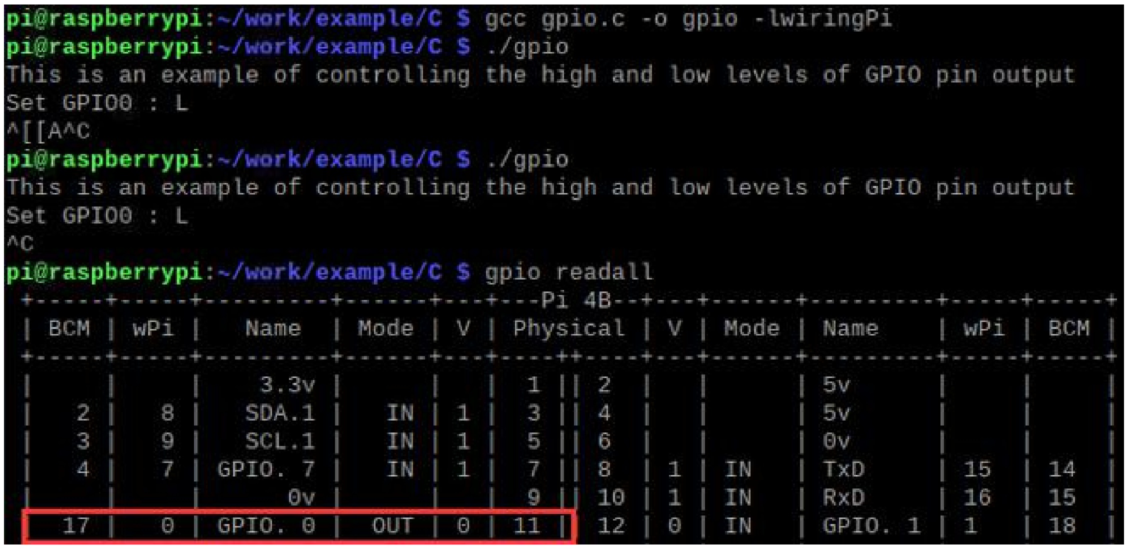
Step 4: Run this code

```
./gpio
```

Step 5: Press Ctrl + C

Step 6: Check the pin status of the Raspberry Pi.

```
gpio readall
```



```
pi@raspberrypi:~/work/example/C $ gcc gpio.c -o gpio -lwiringPi
pi@raspberrypi:~/work/example/C $ ./gpio
This is an example of controlling the high and low levels of GPIO pin output
Set GPIO0 : L
^[[A^C
pi@raspberrypi:~/work/example/C $ ./gpio
This is an example of controlling the high and low levels of GPIO pin output
Set GPIO0 : L
^C
pi@raspberrypi:~/work/example/C $ gpio readall
 +-----+-----+---------+------+---+---Pi 4B--+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5v      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | IN   | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | IN   | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |  OUT | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
```

We can know that the GPIO0 is OUT mode, and V (state) is 0 (low level).