Gesture recognition

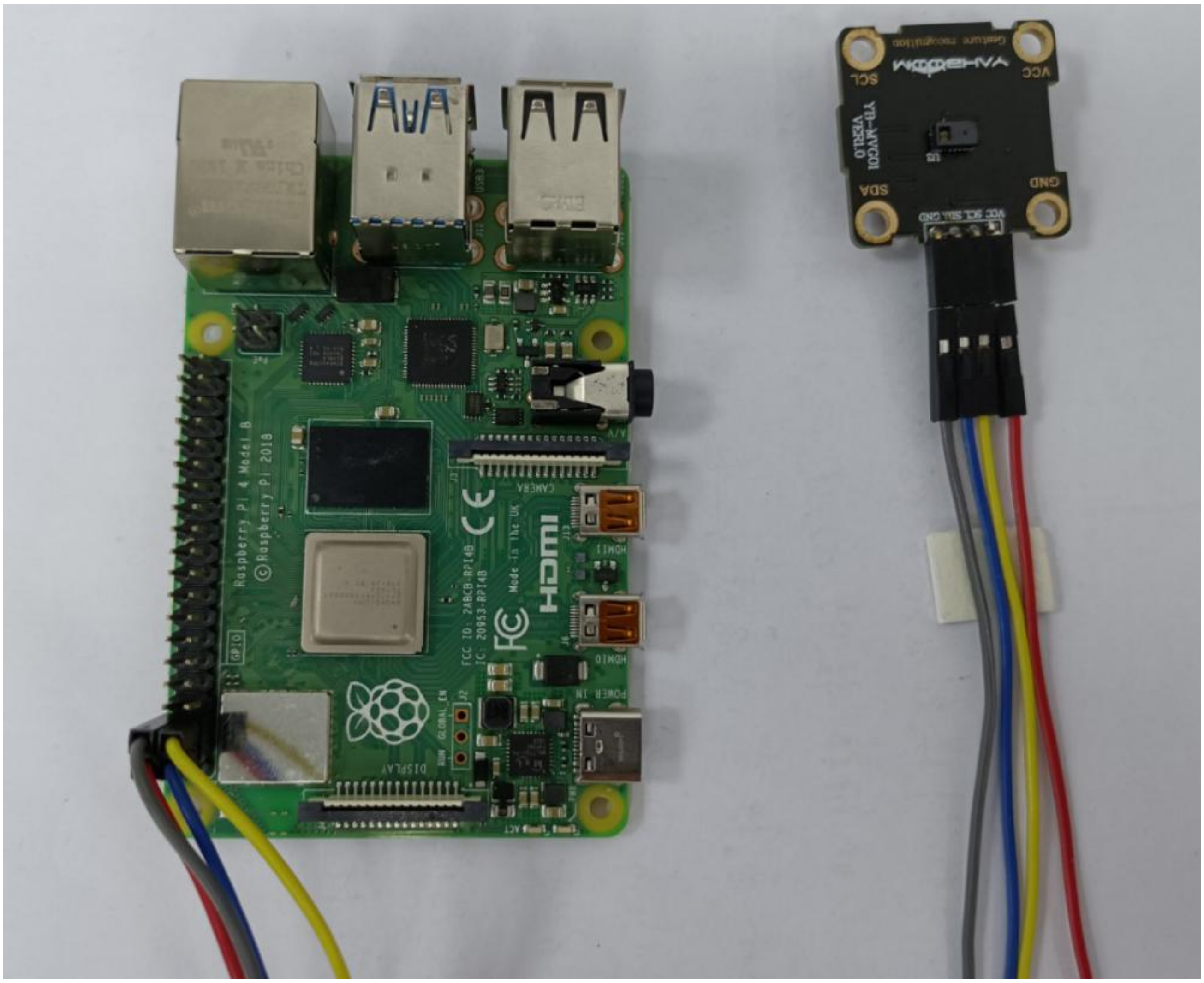
1. Purpose

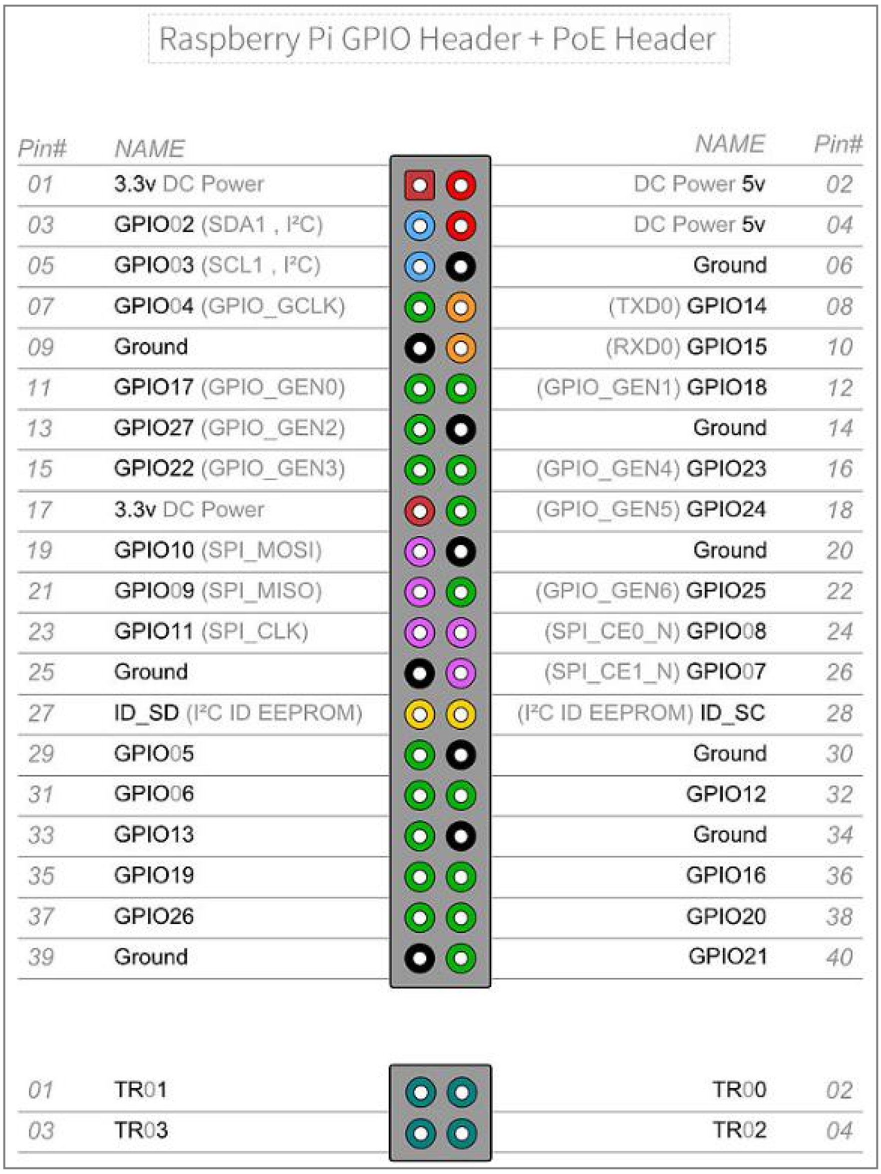
In this course, we mainly learn to use Raspberry Pi and gesture recognition module.

2.Preparation

2.1 About wiring

Speech synthesis module	Raspberry Pi board
SCL	SCL
SDA	SDA
VCC	5V
GND	GND





Raspberry Pi GPIO Header + PoE Header

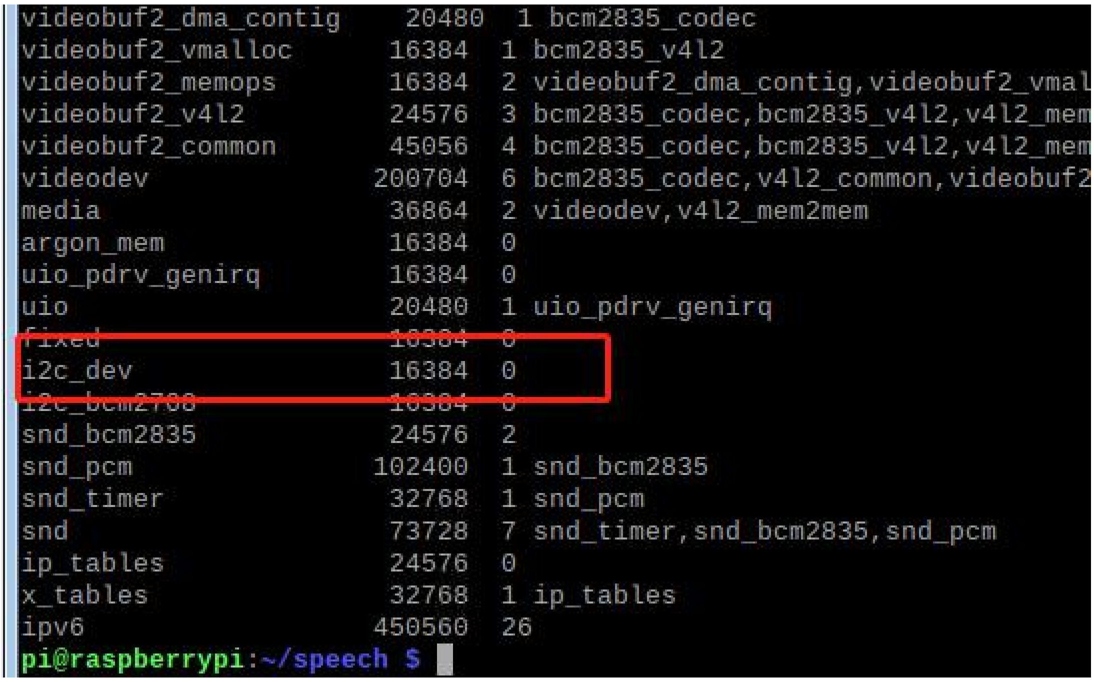
Pin#	NAME			NAME	Pin#
01	3.3v DC Power	▣	●	DC Power 5v	02
03	GPIO02 (SDA1 , I^{2}C)	●	●	DC Power 5v	04
05	GPIO03 (SCL1 , I^{2}C)	●	●	Ground	06
07	GPIO04 (GPIO_GCLK)	●	●	(TXD0) GPIO14	08
09	Ground	●	●	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	●	●	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	●	●	Ground	14
15	GPIO22 (GPIO_GEN3)	●	●	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	●	●	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	●	●	Ground	20
21	GPIO09 (SPI_MISO)	●	●	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	●	●	(SPI_CE0_N) GPIO08	24
25	Ground	●	●	(SPI_CE1_N) GPIO07	26
27	ID_SD (I^{2}C ID EEPROM)	●	●	(I^{2}C ID EEPROM) ID_SC	28
29	GPIO05	●	●	Ground	30
31	GPIO06	●	●	GPIO12	32
33	GPIO13	●	●	Ground	34
35	GPIO19	●	●	GPIO16	36
37	GPIO26	●	●	GPIO20	38
39	Ground	●	●	GPIO21	40

01	TR01	●	●	TR00	02
03	TR03	●	●	TR02	04

2.2 You need to open the IIC service of Raspberry Pi board.

We can input following command to check whether I2C is successfully started.

```
lmusb
```



```
videobuf2_dma_contig    20480  1 bcm2835_codec
videobuf2_vmalloc       16384  1 bcm2835_v4l2
videobuf2_memops        16384  2 videobuf2_dma_contig,videobuf2_vmal
videobuf2_v4l2          24576  3 bcm2835_codec,bcm2835_v4l2,v4l2_mem
videobuf2_common        45056  4 bcm2835_codec,bcm2835_v4l2,v4l2_mem
videodev               200704  6 bcm2835_codec,v4l2_common,videobuf2
media                   36864  2 videodev,v4l2_mem2mem
argon_mem               16384  0
uio_pdrv_genirq         16384  0
uio                     20480  1 uio_pdrv_genirq
fixed                   16384  0
i2c_dev                 16384  0
i2c_bcm2708             16384  0
snd_bcm2835             24576  2
snd_pcm                102400  1 snd_bcm2835
snd_timer               32768  1 snd_pcm
snd                     73728  7 snd_timer,snd_bcm2835,snd_pcm
ip_tables               24576  0
x_tables                32768  1 ip_tables
ipv6                   450560  26
pi@raspberrypi:~/speech $
```

2.2 Install I2Ctool

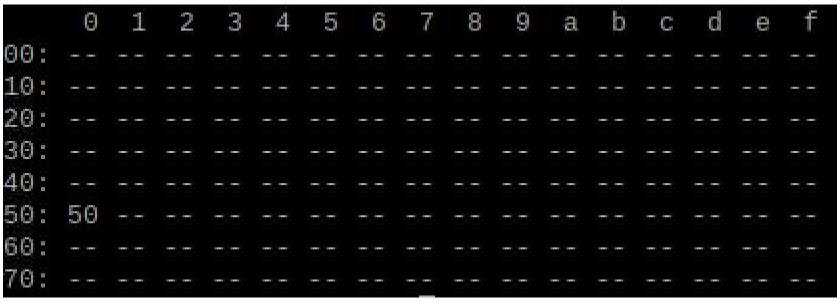
Input following command in command terminal,

sudo apt-get install i2c-tools

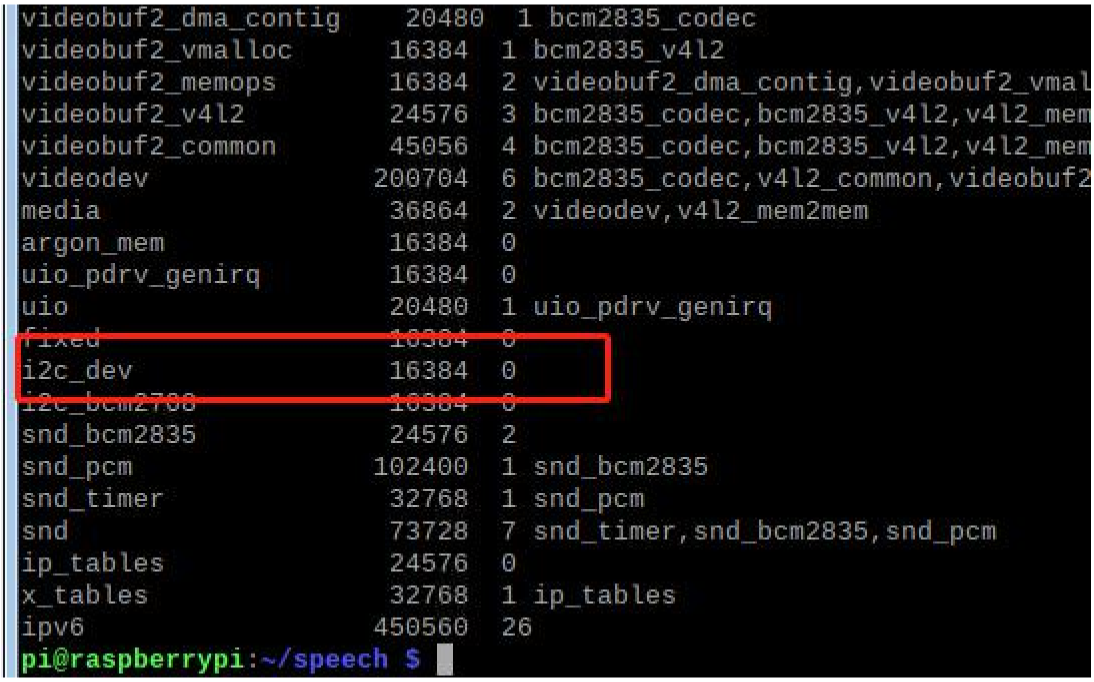
2.4 Scan all i2c devices on a certain bus, and print out the device i2c bus address.

i2cdetect -y -a 1

For gesture recognition module, IIC address is 0x73.



```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```



```
videobuf2_dma_contig    20480  1 bcm2835_codec
videobuf2_vmalloc      16384  1 bcm2835_v4l2
videobuf2_memops       16384  2 videobuf2_dma_contig,videobuf2_vmal
videobuf2_v4l2         24576  3 bcm2835_codec,bcm2835_v4l2,v4l2_mem
videobuf2_common       45056  4 bcm2835_codec,bcm2835_v4l2,v4l2_mem
videodev              200704  6 bcm2835_codec,v4l2_common,videobuf2
media                  36864  2 videodev,v4l2_mem2mem
argon_mem              16384  0
uio_pdrv_genirq        16384  0
uio                    20480  1 uio_pdrv_genirq
fixed                  16384  0
i2c_dev                16384  0
i2c_bcm2708            16384  0
snd_bcm2835            24576  2
snd_pcm               102400  1 snd_bcm2835
snd_timer              32768  1 snd_pcm
snd                    73728  7 snd_timer,snd_bcm2835,snd_pcm
ip_tables              24576  0
x_tables               32768  1 ip_tables
ipv6                  450560  26
pi@raspberrypi:~/speech $
```

3. Code

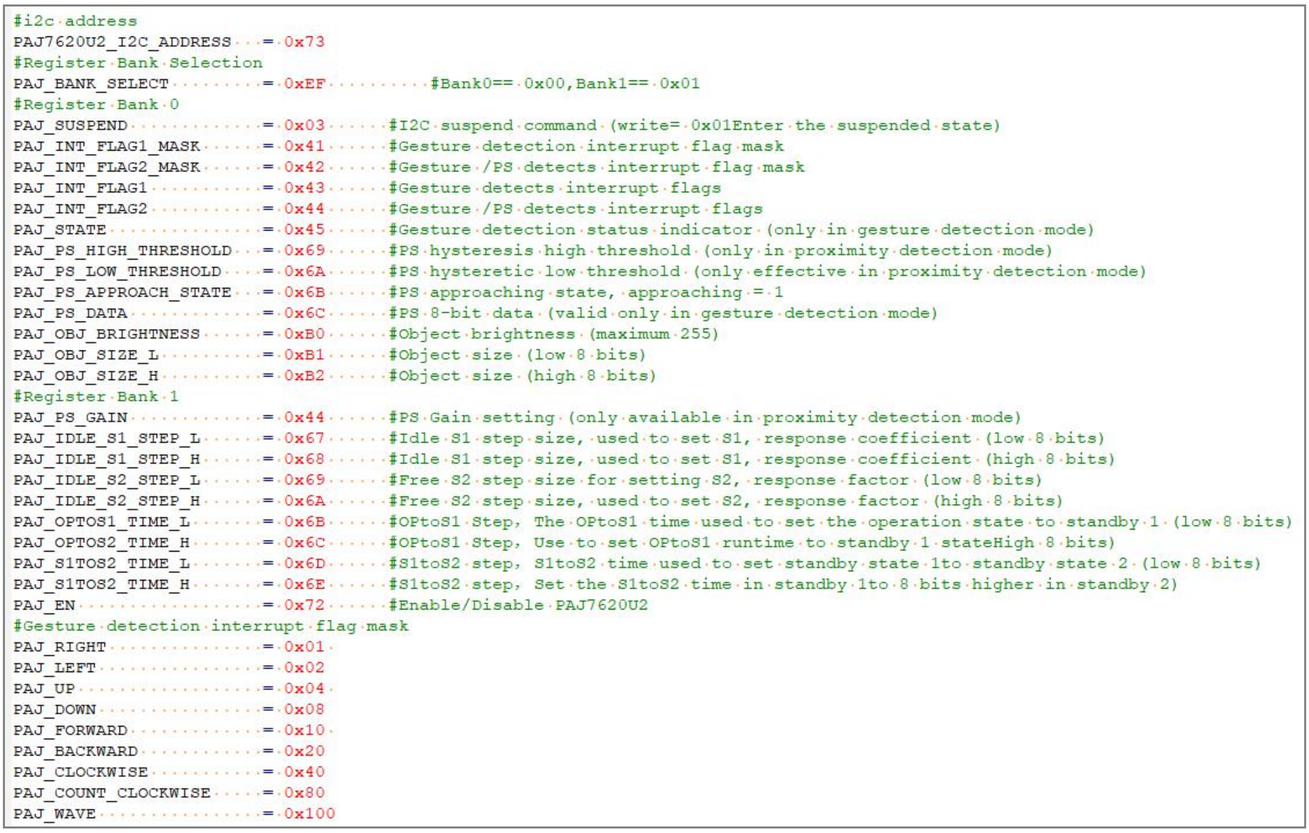
About code, please view **PAJ7620U2.py** file.

4. Running code

Input following command in command terminal of Raspberry Pi.

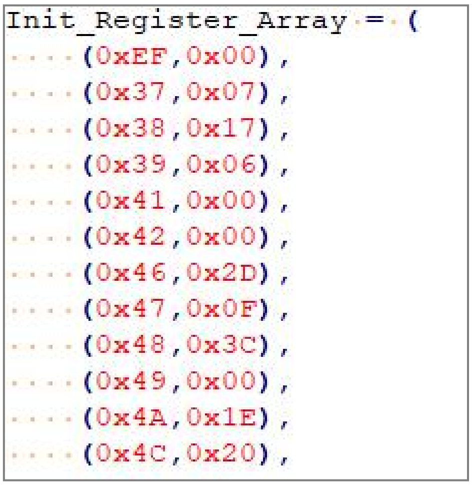
python3 PAJ7620U2.py

3.1 Define the device address and register address of the module.

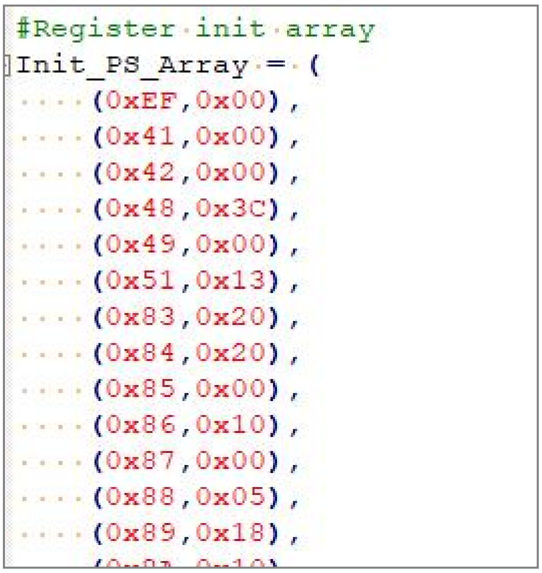


```
#i2c address
PAJ7620U2_I2C_ADDRESS   = 0x73
#Register Bank Selection
PAJ_BANK_SELECT         = 0xEF          #Bank0== 0x00,Bank1== 0x01
#Register Bank 0
PAJ_SUSPEND             = 0x03     #I2C suspend command (write= 0x01Enter the suspended state)
PAJ_INT_FLAG1_MASK      = 0x41     #Gesture detection interrupt flag mask
PAJ_INT_FLAG2_MASK      = 0x42     #Gesture /PS detects interrupt flag mask
PAJ_INT_FLAG1           = 0x43     #Gesture detects interrupt flags
PAJ_INT_FLAG2           = 0x44     #Gesture /PS detects interrupt flags
PAJ_STATE               = 0x45     #Gesture detection status indicator (only in gesture detection mode)
PAJ_PS_HIGH_THRESHOLD   = 0x69     #PS hysteresis high threshold (only in proximity detection mode)
PAJ_PS_LOW_THRESHOLD    = 0x6A     #PS hysteretic low threshold (only effective in proximity detection mode)
PAJ_PS_APPROACH_STATE   = 0x6B     #PS approaching state, approaching = 1
PAJ_PS_DATA             = 0x6C     #PS 8-bit data (valid only in gesture detection mode)
PAJ_OBJ_BRIGHTNESS      = 0xB0     #Object brightness (maximum 255)
PAJ_OBJ_SIZE_L          = 0xB1     #Object size (low 8 bits)
PAJ_OBJ_SIZE_H          = 0xB2     #Object size (high 8 bits)
#Register Bank 1
PAJ_PS_GAIN             = 0x44     #PS Gain setting (only available in proximity detection mode)
PAJ_IDLE_S1_STEP_L      = 0x67     #Idle S1 step size, used to set S1, response coefficient (low 8 bits)
PAJ_IDLE_S1_STEP_H      = 0x68     #Idle S1 step size, used to set S1, response coefficient (high 8 bits)
PAJ_IDLE_S2_STEP_L      = 0x69     #Free S2 step size for setting S2, response factor (low 8 bits)
PAJ_IDLE_S2_STEP_H      = 0x6A     #Free S2 step size, used to set S2, response factor (high 8 bits)
PAJ_OPTOS1_TIME_L       = 0x6B     #OPtoS1 Step, The OPtoS1 time used to set the operation state to standby 1 (low 8 bits)
PAJ_OPTOS2_TIME_H       = 0x6C     #OPtoS1 Step, Use to set OPtoS1 runtime to standby 1 stateHigh 8 bits)
PAJ_S1TOS2_TIME_L       = 0x6D     #S1toS2 step, S1toS2 time used to set standby state 1to standby state 2 (low 8 bits)
PAJ_S1TOS2_TIME_H       = 0x6E     #S1toS2 step, Set the S1toS2 time in standby 1to 8 bits higher in standby 2)
PAJ_EN                  = 0x72     #Enable/Disable PAJ7620U2
#Gesture detection interrupt flag mask
PAJ_RIGHT               = 0x01 
PAJ_LEFT                = 0x02
PAJ_UP                  = 0x04 
PAJ_DOWN                = 0x08
PAJ_FORWARD             = 0x10 
PAJ_BACKWARD            = 0x20
PAJ_CLOCKWISE           = 0x40
PAJ_COUNT_CLOCKWISE     = 0x80
PAJ_WAVE                = 0x100
```

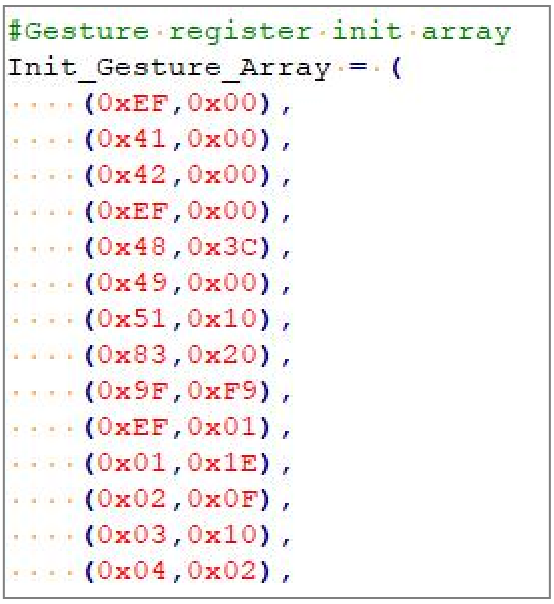
3.2 Define initialization array, register array, gesture register address.



```
Init_Register_Array = (
    (0xEF,0x00),
    (0x37,0x07),
    (0x38,0x17),
    (0x39,0x06),
    (0x41,0x00),
    (0x42,0x00),
    (0x46,0x2D),
    (0x47,0x0F),
    (0x48,0x3C),
    (0x49,0x00),
    (0x4A,0x1E),
    (0x4C,0x20),
```



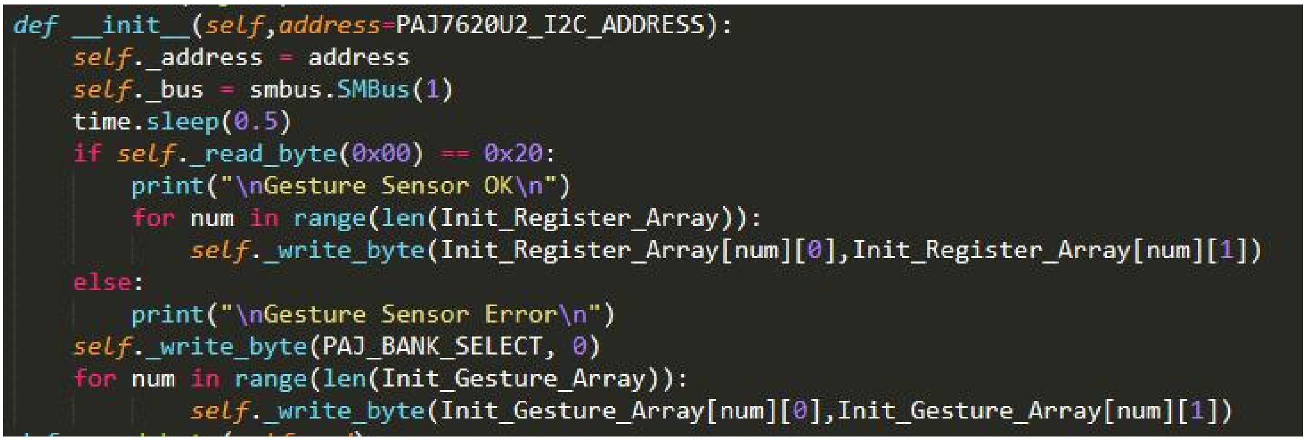
```
#Register init array
Init_PS_Array = (
    (0xEF,0x00),
    (0x41,0x00),
    (0x42,0x00),
    (0x48,0x3C),
    (0x49,0x00),
    (0x51,0x13),
    (0x83,0x20),
    (0x84,0x20),
    (0x85,0x00),
    (0x86,0x10),
    (0x87,0x00),
    (0x88,0x05),
    (0x89,0x18),
```



```
#Gesture register init array
Init_Gesture_Array = (
    (0xEF,0x00),
    (0x41,0x00),
    (0x42,0x00),
    (0xEF,0x00),
    (0x48,0x3C),
    (0x49,0x00),
    (0x51,0x10),
    (0x83,0x20),
    (0x9F,0xF9),
    (0xEF,0x01),
    (0x01,0x1E),
    (0x02,0x0F),
    (0x03,0x10),
    (0x04,0x02),
```

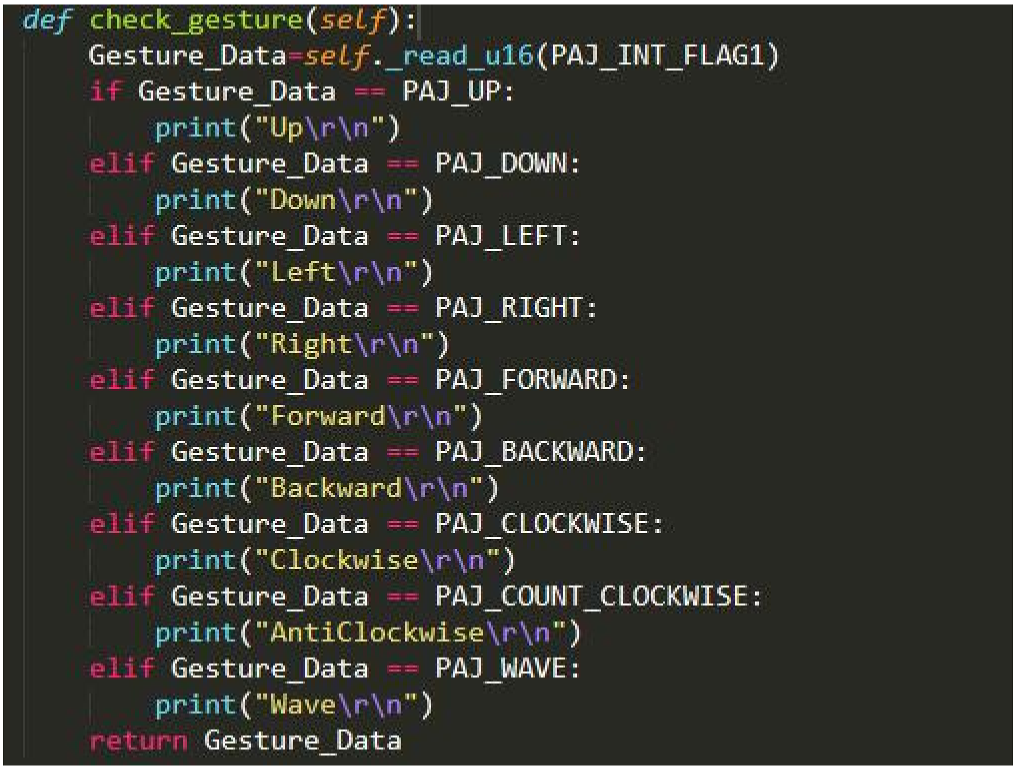
....

3.3 Through I2C, the value of the initializing array and the initializing array of gesture register are written into the corresponding registers to start and initialize the gesture recognition module.



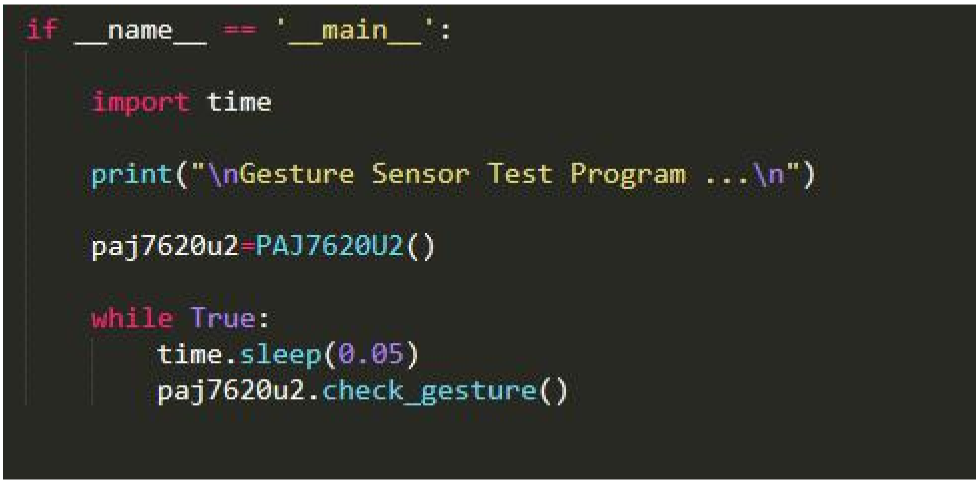
```
def __init__(self,address=PAJ7620U2_I2C_ADDRESS):
    self._address = address
    self._bus = smbus.SMBus(1)
    time.sleep(0.5)
    if self._read_byte(0x00) == 0x20:
        print("\nGesture Sensor OK\n")
        for num in range(len(Init_Register_Array)):
            self._write_byte(Init_Register_Array[num][0],Init_Register_Array[num][1])
    else:
        print("\nGesture Sensor Error\n")
    self._write_byte(PAJ_BANK_SELECT, 0)
    for num in range(len(Init_Gesture_Array)):
            self._write_byte(Init_Gesture_Array[num][0],Init_Gesture_Array[num][1])
```

3.4 Gesture recognition function: judge the currently recognized gesture by reading the value of the gesture recognition storage register and print out the corresponding gesture name.



```
def check_gesture(self):
    Gesture_Data=self._read_u16(PAJ_INT_FLAG1)
    if Gesture_Data == PAJ_UP:
        print("Up\r\n")
    elif Gesture_Data == PAJ_DOWN:
        print("Down\r\n")
    elif Gesture_Data == PAJ_LEFT:
        print("Left\r\n")
    elif Gesture_Data == PAJ_RIGHT:
        print("Right\r\n")
    elif Gesture_Data == PAJ_FORWARD:
        print("Forward\r\n")
    elif Gesture_Data == PAJ_BACKWARD:
        print("Backward\r\n")
    elif Gesture_Data == PAJ_CLOCKWISE:
        print("Clockwise\r\n")
    elif Gesture_Data == PAJ_COUNT_CLOCKWISE:
        print("AntiClockwise\r\n")
    elif Gesture_Data == PAJ_WAVE:
        print("Wave\r\n")
    return Gesture_Data
```

3.5 After successful initialization, the gesture recognition function is cycled to judge the current gesture.



```
if __name__ == '__main__':

    import time

    print("\nGesture Sensor Test Program ...\n")

    paj7620u2=PAJ7620U2()

    while True:
        time.sleep(0.05)
        paj7620u2.check_gesture()
```

4. Running code

Input following command in command terminal of jetson nano.

python3 PAJ7620U2.py

After the program running, if the module is initialized successfully, Jetson NANO system will print "Gesture Sensor OK", otherwise it will print "Gesture Sensor Error". If the initialization fails, we need to run code again.

After the initialization is successful, the module will start judge the value of gesture recognition, and different gestures will print out different action names through the serial port.

Put the gesture recognition module in the vertical direction, open your palm to face the module,
Swing over your palm from left to right in front of the module, Raspberry Pi system will print "Left".
Swing over your palm from left to right in front of the module, Raspberry Pi system will print "Right".
Swing over your palm from bottom to top in front of the module, Raspberry Pi system will print "Up".
Swing over your palm from top to buttom in front of the module, Raspberry Pi system will print "Down".
Approach from back to front directly in front of the module, Raspberry Pi system will print "Forward".
Approach from front to back directly in front of the module, Raspberry Pi system will print "Backward".
Make a fist and stretch out two or three fingers to point to the front of the module, then circle it clockwise for a while, Raspberry Pi system will print "Clockwise".
Make a fist and stretch out two or three fingers to point to the front of the module, then circle it counterclockwise for a while, Raspberry Pi system will print "AntiClockwise".
Wave your hand in front of the module for a while, Raspberry Pi system will print "Wave".