AS7341 Spectral Color Sensor

With AS7341 visible spectrum sensing IC as the core. this product can sense the visible light component values of different bands in the environment. It is also quite impressive in sensitivity and accuracy. At the same time, its volume is very small. To make a miniature spectrum analyzer, it will be a very good choice.

Feature

- Incorporates AS7341 chip, which integrates 8x visible spectrum channels, 1x near infrared channel, and 1x no filter channel
- Embedded 6x independent 16-bit ADC, allows effectively processing data in parallel
- Dedicated channel to detect ambient light flicker on the specific frequency
- 2x high brightness LEDs, can be used as fill light on the dim environment
- Interrupt pin to output inner ADC real time operating status
- Features spectrum interrupt detection, with programable high/low thresholds
- Provides general purpose input/output GPIO pin
- Onboard voltage translator, compatible with 3.3V/5V operating voltage
- Comes with development resources and manual (examples for Raspberry Pi/Arduino/STM32)

Specification

- Operating voltage: 3.3V/5V
- Operating current: 20mA (without open the LED) 70mA (when open the LED)
- Sensor: AS7341
- Logical voltage: 3.3V/5V
- Interface: I2C
- Dimension: 30.5mm x 23mm

- Mounting hole size: 2.0mm

Pinout

Pin number	PIN	Description
1	VCC	3.3V/5V Power input
2	GND	Ground
3	SDA	I2C data line
4	SCL	I2C clock line
5	INT	Interrupt output pin
6	GPIO	input/output GPIO port

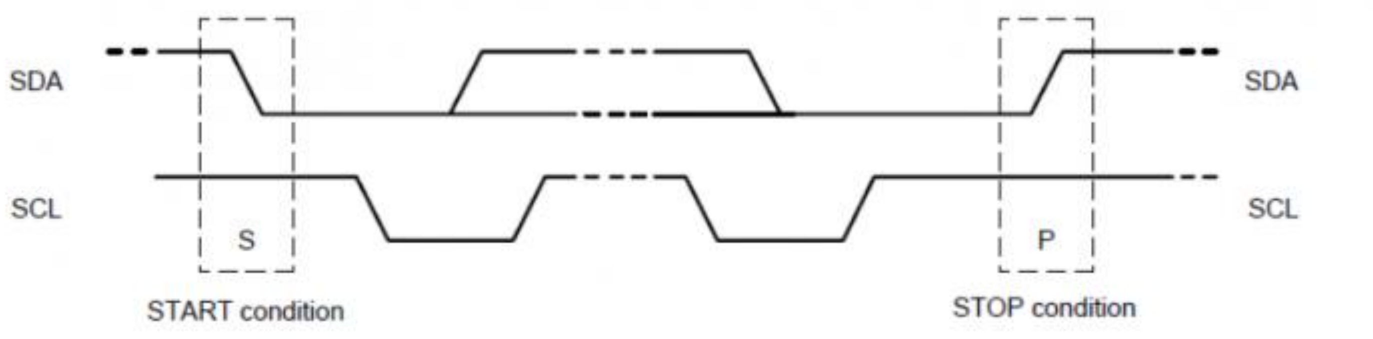
Hardware description

AS7341 Chip

This product uses AS7341-DLGM as the core, which is an 11-channel IC for spectrum recognition and color matching applications. The spectral response is defined at a wavelength of about 350nm to 1000nm. 6 channels can be processed in parallel by independent adcs, while the other channels can be accessed through a multiplexer. AS7341 integrates the filter into the silicon of standard CMOS through a nano-optical deposition interference filter. The technology and its package provide a built-in aperture to control the light entering the sensor array. Its control and spectral data access are realized through the serial I²C interface.

Working Protocol

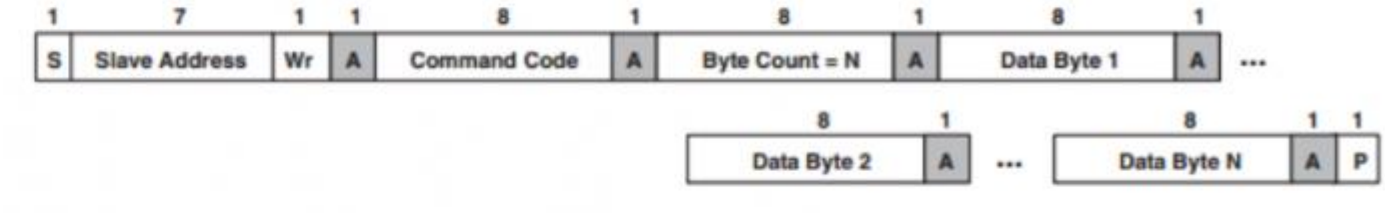
This product uses I2C communication with one data line and one clock line. There are three types of signals in the I2C bus in the process of transmitting data: START signal, STOP signal, and Response signal.



START signal: When SCL is high level, SDA jumps from high to low to start transmitting data.
STOP signal: When SCL is high level, SDA jumps from low level to high level, ending the

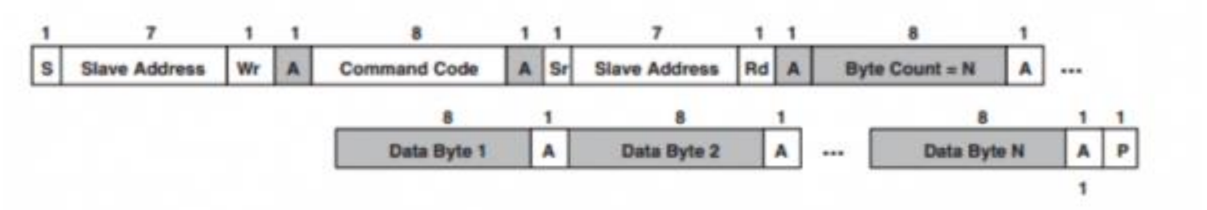
data transmission.
Response signal: After receiving the 8bit data, the data receiving IC sends a specific low-level pulse to the data sending IC to indicate that the data has been received.

- I2C write data timing



First, the host(ie, Raspberry Pi) will send a start signal, and then combine its I2C 7-bit address and write operation bits into 8-bit data and send it to the slave (ie TSL2581 sensor module), the slave will respond with a response signal after receiving it. The host sends the command register address to the slave, and then the slave receives this response signal. At this time, the master sends the value of the command register and the slave responds with a response signal. Until the host sends a STOP signal, the I2C write data operation ends

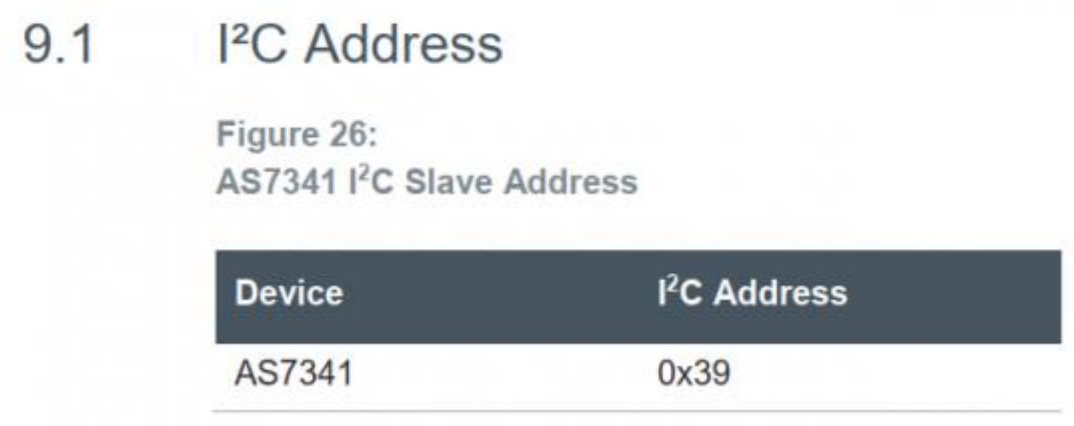
- I2C read data timing



First, the host will send a START signal, and then combine its I2C 7-bit address and write operation bit into 8-bit data and send it to the slave. After receiving it, the slave will respond with a response signal, and the host will send the command register address at this time. After the slave receives the sending response signal, the host will send a START signal again, and combines its 7-bit address and read operation bit into 8-bit data and sends it to the slave. The slave sends a response after receiving the signal, then sends the value in its register to the host. And the host gives a response signal until the host sends a stop signal, and this communication ends.

- I2C address

The I2C device address of AS7341 is 0X39 which you can check on page 21 of the AS7341 data sheet.



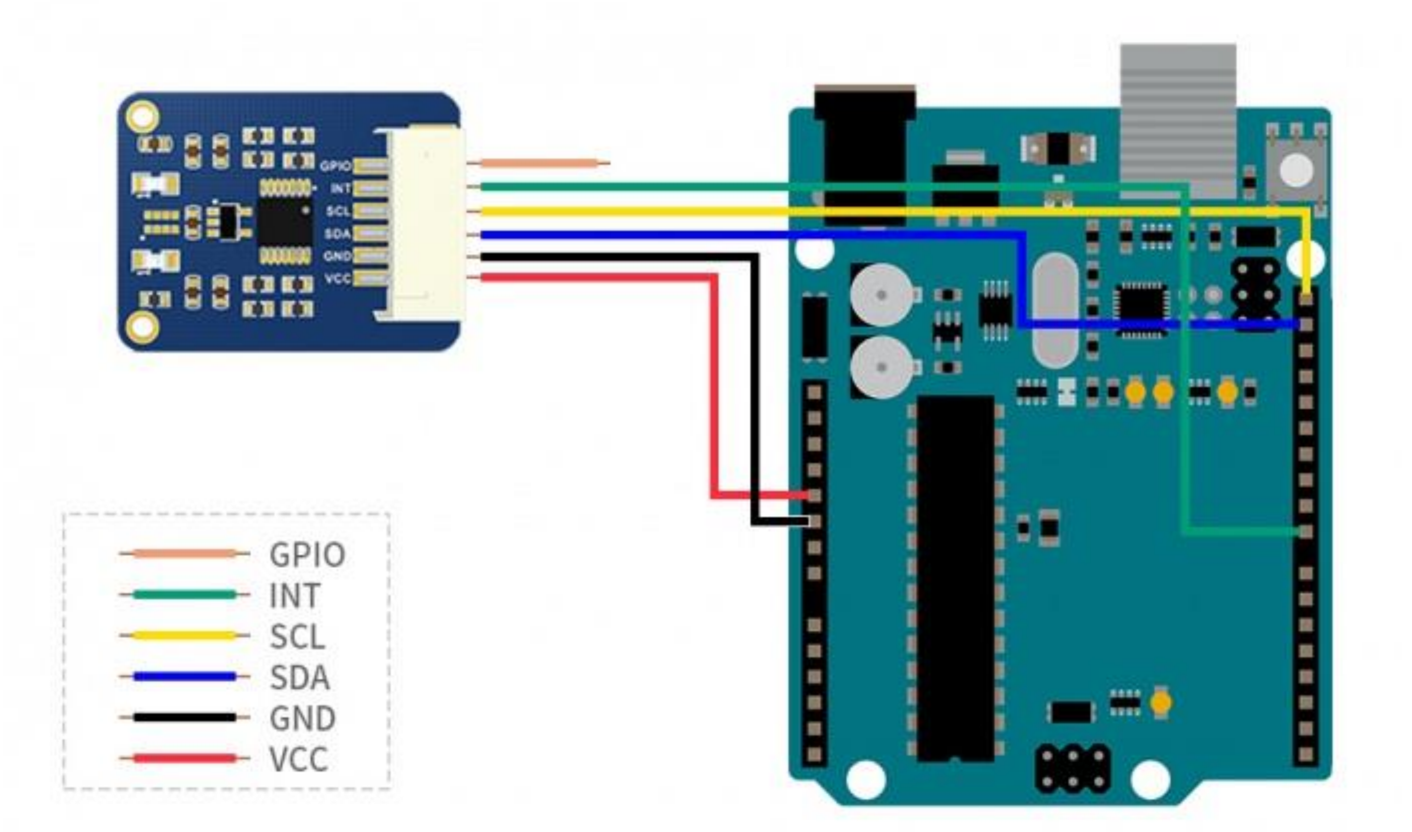
9.1 I²C Address

Figure 26:
AS7341 I²C Slave Address

Device	I²C Address
AS7341	0x39

This example is tested on Arduino UNO. If you use other models of Arduino, please pay attention to whether the related pins are connected correctly.

Hardware connection

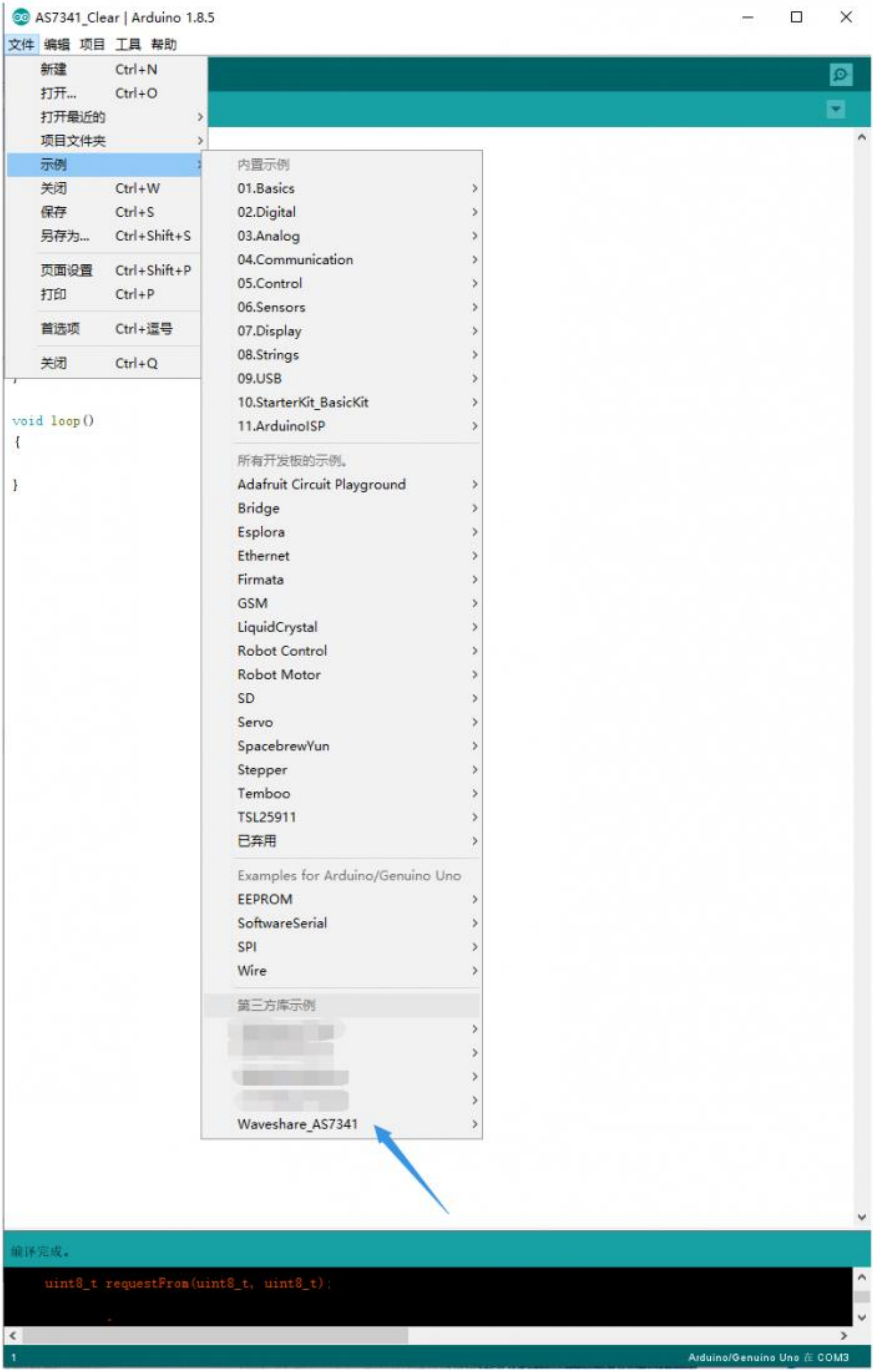


Run the demo

- After downloading the demo, unzip this .7z file on your PC.
- The Arduino program is located in ~/Arduino/... Copy the Waveshare_AS7341 folder from this directory to the libraries in the Arduino installation directory,

usually in C:\Users\XXX\Documents\Arduino\libraries or C:\Program Files (x86)\Arduino \libraries.

- Open Arduino IDE, click File->Example, and check whether there has Waveshare_AS7341 option.

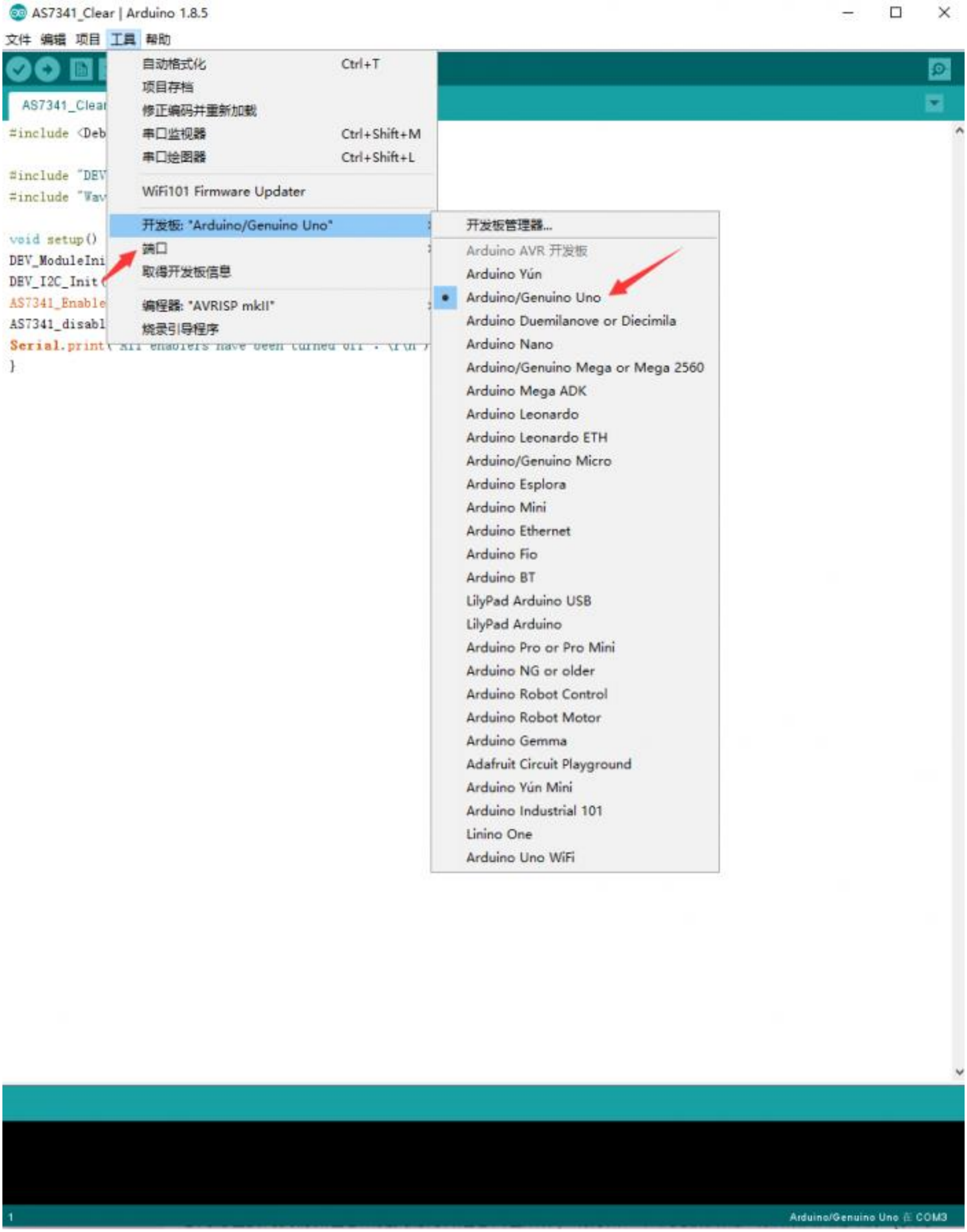


```
void loop()
{

}
```

编译完成。

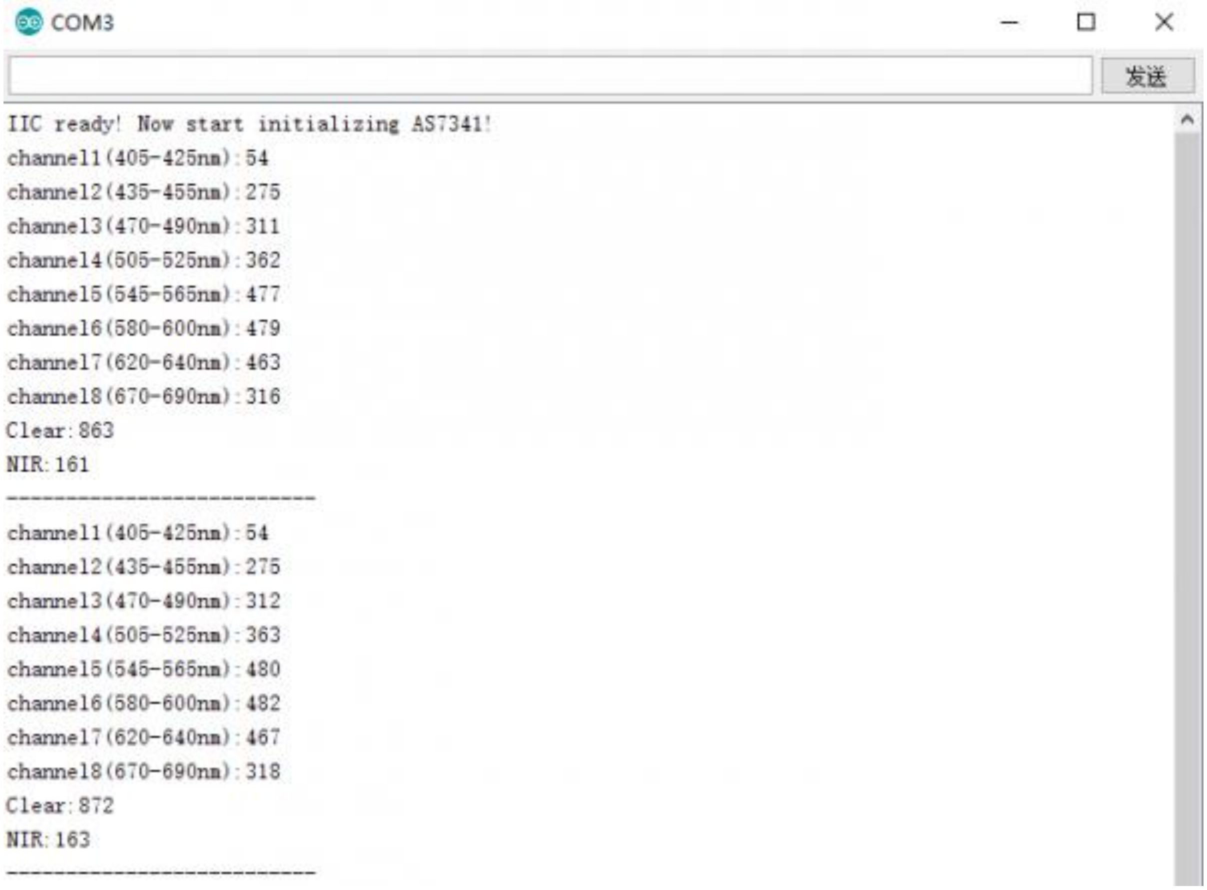
```
    uint8_t requestFrom(uint8_t, uint8_t);
```

- If the Waveshare_AS7341 library is imported successfully, open the ino project file in the Arduino/Waveshare_AS7341/example
- Select the corresponding model of the development board and COM port, compile the program, and then download it to UNO, and open the serial monitor.



```
#include <Deb

#include "DEV
#include "Wav

void setup()
DEV_ModuleIni
DEV_I2C_Init(
AS7341_Enable
AS7341_disabl
Serial.print( All enablers have been turned off : \r\n )
}
```

- Demo phenomena:



```
IIC ready! Now start initializing AS7341!
channel1(405-425nm):54
channel2(435-455nm):275
channel3(470-490nm):311
channel4(505-525nm):362
channel5(545-565nm):477
channel6(580-600nm):479
channel7(620-640nm):463
channel8(670-690nm):316
Clear:863
NIR:161
--------------------------
channel1(405-425nm):54
channel2(435-455nm):275
channel3(470-490nm):312
channel4(505-525nm):363
channel5(545-565nm):480
channel6(580-600nm):482
channel7(620-640nm):467
channel8(670-690nm):318
Clear:872
NIR:163
--------------------------
```

Demo description

There are several different test projects on the Arduino/Waveshare_AS7341/example directory, here we give some describes and precautions.

- AS7341_Getdata is used to obtain 10 channels of test data. AS7341 has only 6 independent ADCs, but it has 11 channels, which requires multiplexer SMUX. For related configuration, please refer to the reference code of the datasheet manual.
- AS7341_Getdata includes the driver code to turn on the fill light LED and adjust the brightness

```
//AS7341_EnableLED(true);// LED ON or OFF
//AS7341_ControlLed(10);//Adjust the brightness of the LED lamp
```

If you need to use LED fill light, just comment these two lines of code

- AS7341_Getflicker is used to detect 100 or 120Hz ambient light flicker, you need to generate a flickering light of this frequency by yourself.

Adjust the integration time, gain, etc. to detect flicker of different frequencies。

- AS7341_Syns configures the sensor mode as SYNS mode. In this mode, the GPIO port of the sensor needs to receive a falling edge signal to trigger measurement, and each falling edge triggers a measurement.

This module does not directly connect the GPIO port by default. Therefore, during your testing, you can briefly touch the GPIO port with the 3.3v or 5V pin of the development board and then disconnect it to obtain a falling edge signal. And in your actual use, you can connect the GPIO port to the trigger source directly.

```
while(!AS7341_MeasureComplete());    // Jump out of this loop when GPIO receives a valid signal.
```

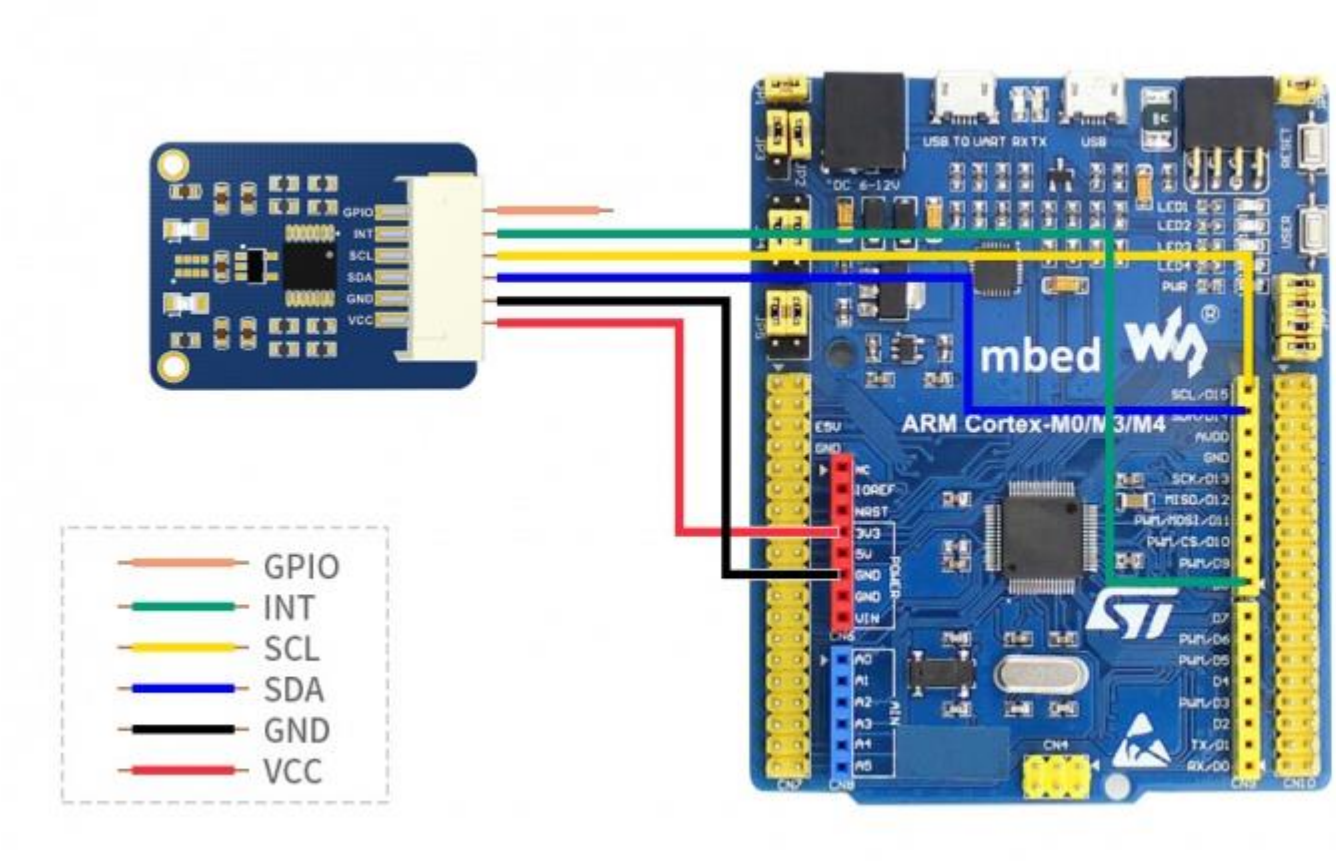
- AS7341_INT is the spectrum interrupt test. It sets the upper and lower thresholds for interrupt generation. At the same time, you can set the channel triggered by the interrupt. The channel selection can be one of CH0-CH4. When the interrupt is triggered by the change of ambient light, read the value of the relevant register to check whether it is triggered.

```
AS7341_SetInterruptPersistence(0);    // Set the sensitivity of spectrum interruption

AS7341_SetSpectralThresholdChannel(4);   //Set the channel to detect interrupts
```

- AS7341_pinINT is an experiment on the INT pin on the module. After each measurement of AS7341, the INT pin will become low level. You can configure the relevant register to set how often the sensor measures the environmental spectrum data, which also determines the INT pin How often does the foot jump. In this demo, the measurement time is set to 1s, and the level status of the INT pin is monitored at the same time.
- AS7341_Clear is to reset all the register enable bits that are turned on in the AS7341.

This demo has been verified on NUCLEO-F103RB (chip model STM32RBT6) and OpenH743I-C (chip model STM32H743IIT6). If you need to migrate to other boards, please pay attention to the relevant configuration and Hardware connection.

Hardware connection



AS7341 Spectral Color Sensor	XNUCLEO-F103RB
VCC	3.3V/5V Power input
GND	Ground
SDA	SDA/D14/PB9
SCL	SCL/D15/PB8
INT	D8/PA9
GPIO	/

Hardware connection with OpenH743I-C

AS7341 Spectral Color Sensor	OpenH743I-C
VCC	3.3V/5V Power input
GND	Ground
SDA	PD13(I2C4 SDA)
SCL	PD12(I2C4 SCL)
INT	PD11
GPIO	/

Demo description

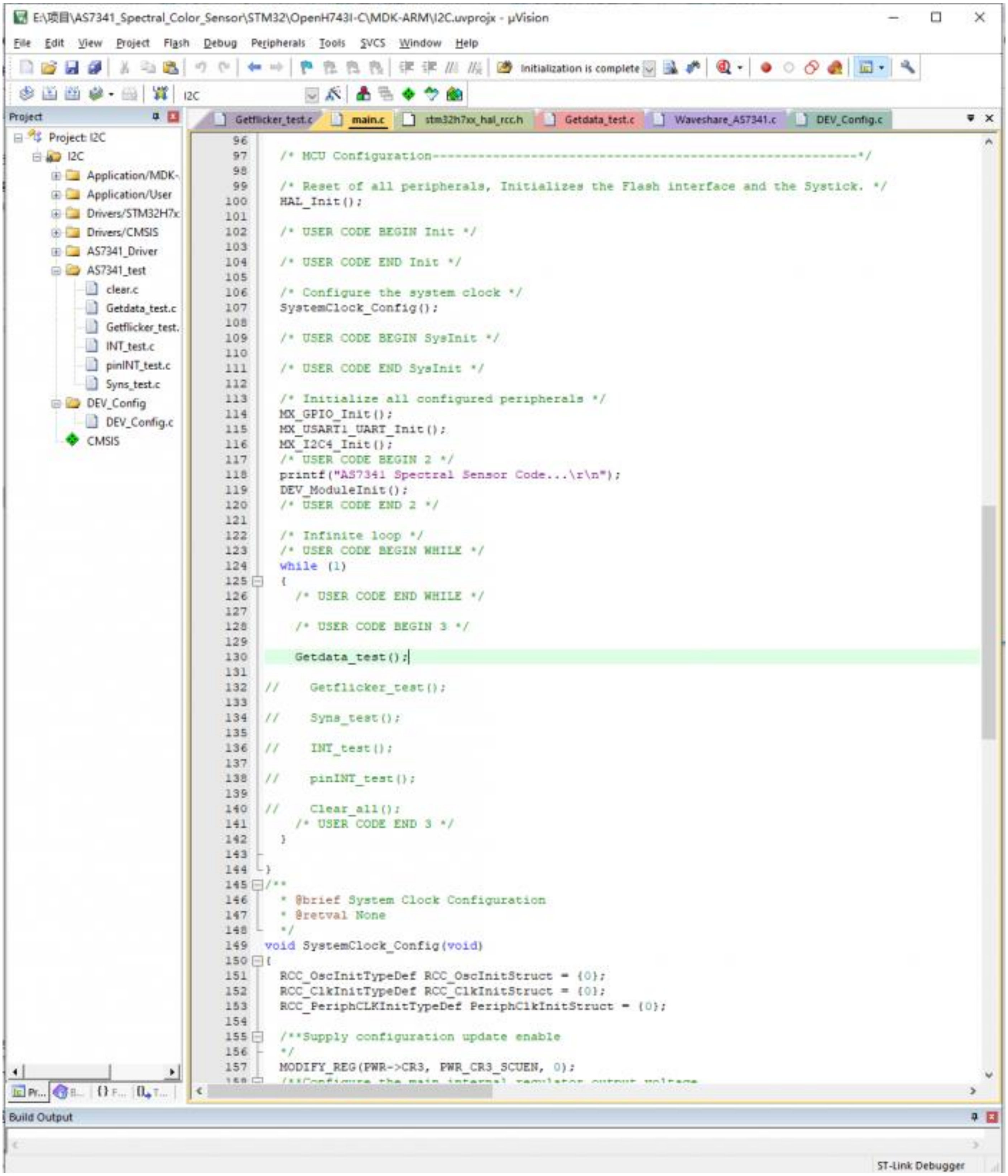
After downloading the demo, unzip this .7z file on your PC. The STM32 demo is located in ~/ STM32/... . You can see the two folders NUCLEO-F103RB and OpenH743I-C.

NUCLEO-F103RB

- Open STM32\XNUCLEO-F103RB\MDK-ARM\demo.uvprojx on the demo folder, this demo uses HAL library.
- If you need to change the chip or want to use the standard library, you just need to change DEV_Config.c and .h to implement the functions and macro definitions inside. The chip can also be configured by using STM32CubeMX. This demo uses serial port 2 (PA2, PA3) to output data.
- The serial port baud rate is 115200, and other options use the default values: 8 data bits, 1 stop bit, and no parity bit. The serial port assistant tool is provided in the folder.

OpenH743I-C

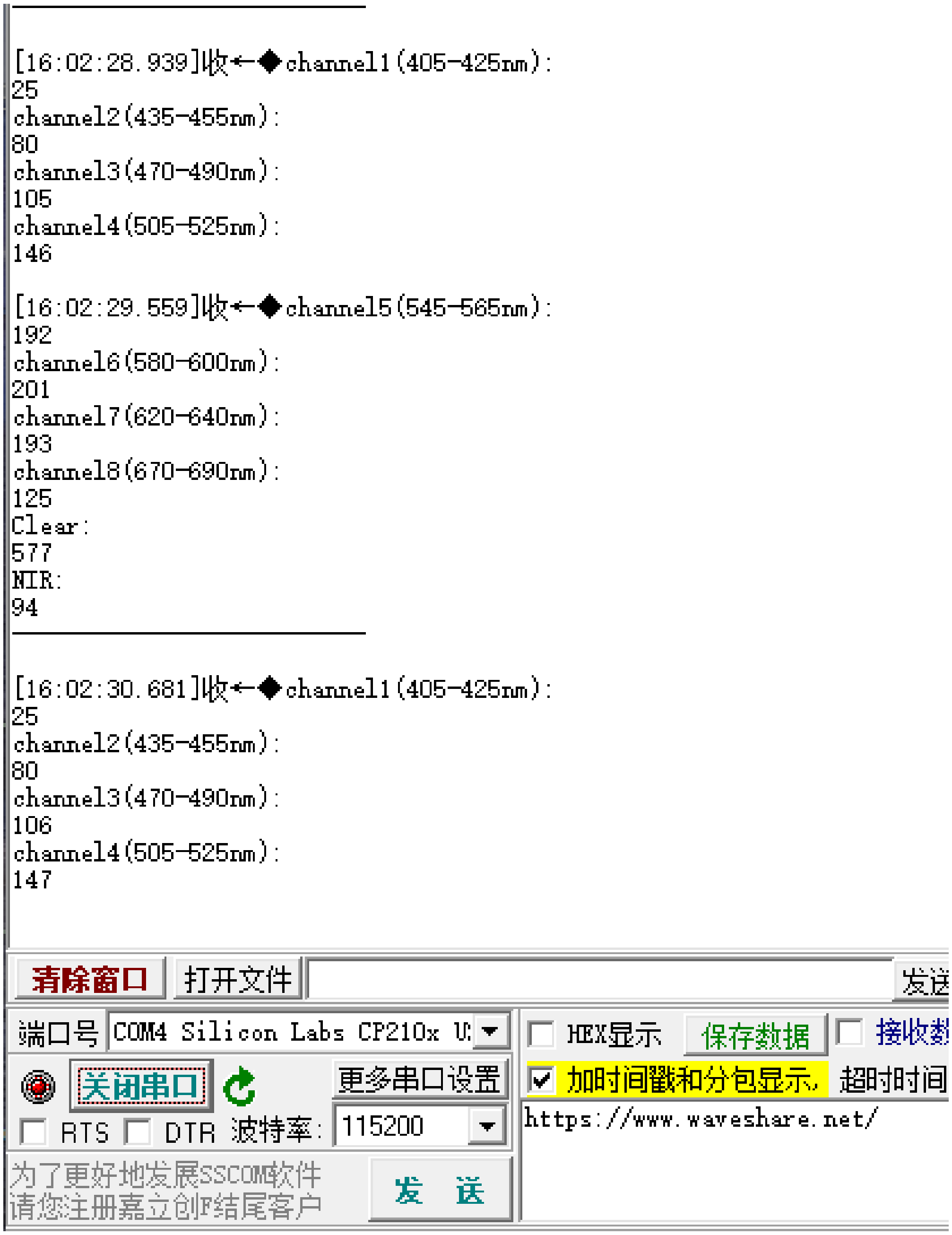
- Open STM32\OpenH743I-C\MDK-ARM\I2C.uvprojx, this demo also uses HAL library.The two are different in terms of chip signals and peripheral configuration, but the test demo used are exactly the same. Here we take OpenH743I-C as an example.
- Open main.c in the project, uncomment the program that needs to be tested. Shown as the picture below:



```

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART1_UART_Init();
  MX_I2C4_Init();
  /* USER CODE BEGIN 2 */
  printf("AS7341 Spectral Sensor Code...\r\n");
  DEV_ModuleInit();
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

    Getdata_test();

//    Getflicker_test();

//    Syns_test();

//    INT_test();

//    pinINT_test();

//    Clear_all();
    /* USER CODE END 3 */
  }

}
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};

  /**Supply configuration update enable
  */
  MODIFY_REG(PWR->CR3, PWR_CR3_SCUEN, 0);
  /**Configure the main internal regulator output voltage
```

- Connect the downloader, connect the serial data line to USART1, click compile to download and verify.
- The related program usage and instructions have been explained in the Arduino tutorial, you can check it on the Arduino chapter page.

- Test result:



```
[16:02:28.939]收←◆channel1(405-425nm):
25
channel2(435-455nm):
80
channel3(470-490nm):
105
channel4(505-525nm):
146

[16:02:29.559]收←◆channel5(545-565nm):
192
channel6(580-600nm):
201
channel7(620-640nm):
193
channel8(670-690nm):
125
Clear:
577
NIR:
94

[16:02:30.681]收←◆channel1(405-425nm):
25
channel2(435-455nm):
80
channel3(470-490nm):
106
channel4(505-525nm):
147
```

端口号 COM4 Silicon Labs CP210x U

□ RTS □ DTR 波特率: 115200

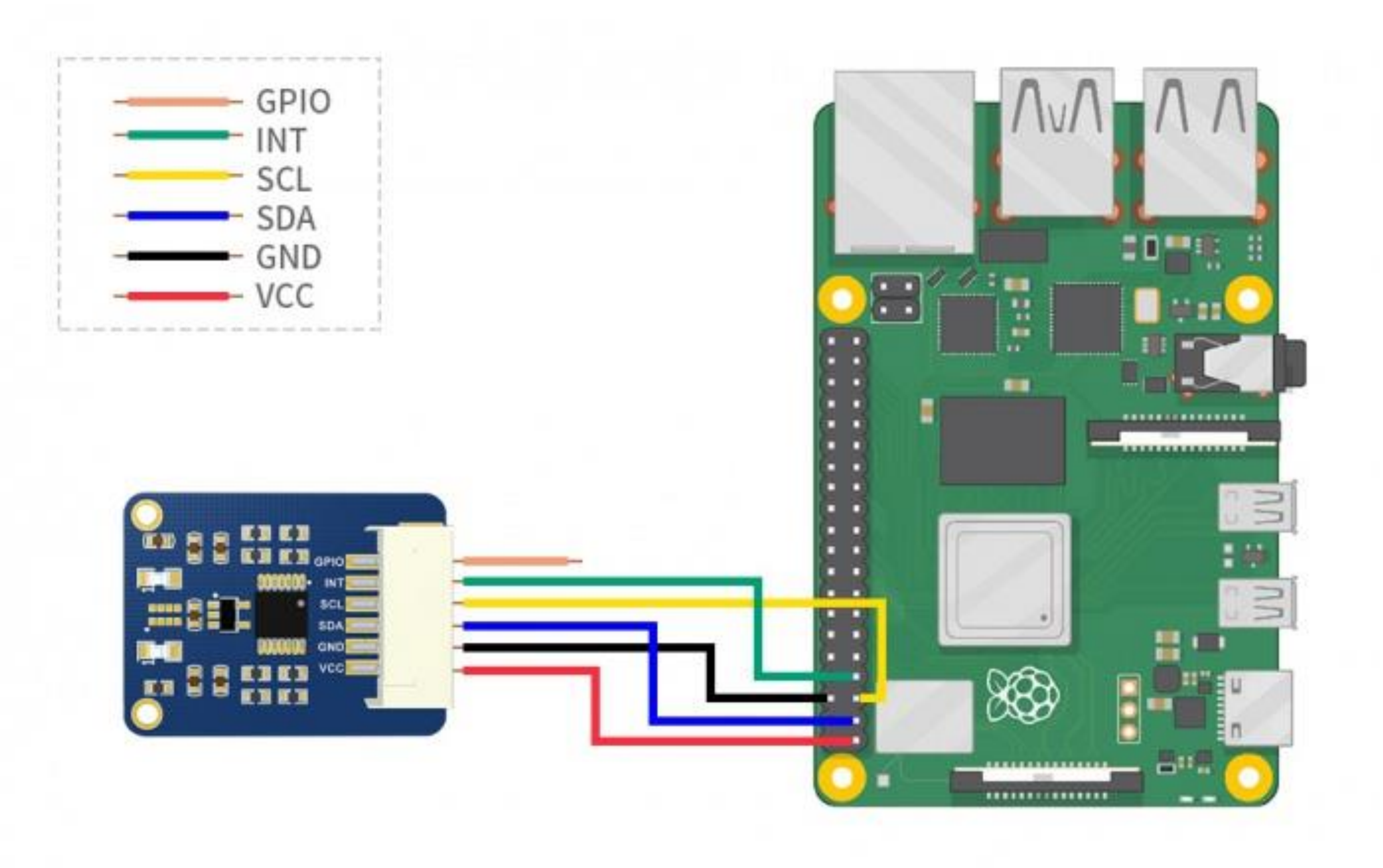
□ HEX显示 □ 接收数

☑ 加时间戳和分包显示

https://www.waveshare.net/

This example uses Raspberry Pi 3 Model B, provides BCM2835, WiringPi, file IO, RPI (Python) library demos.

Hardware connection



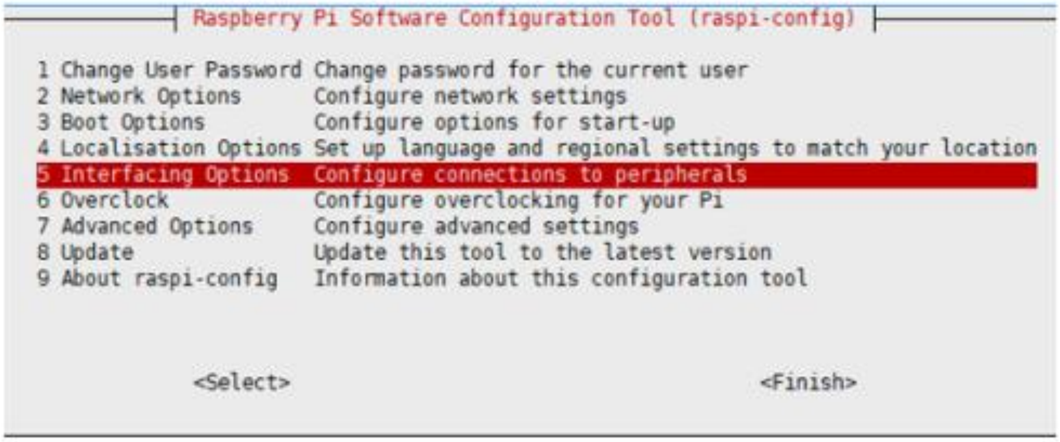
AS7341 Spectral Color Sensor	Raspberry Pi (BCM)
VCC	3.3V/5V Power input
GND	Ground
SDA	SDA(2)
SCL	SCL(3)
INT	4
GPIO	/

Working with Raspberry Pi

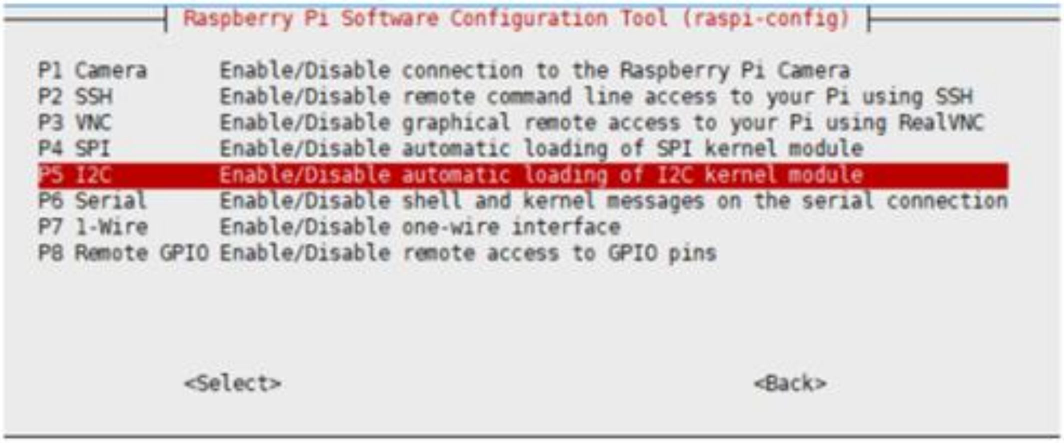
Enable I2C interface

- Open terminal, use command to enter the configuration page

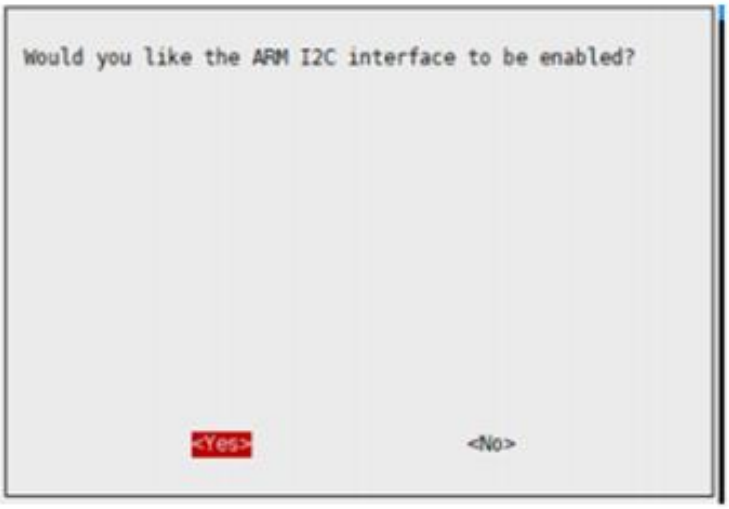
```
sudo raspi-config
Choose Interfacing Options -> I2C -> Yes  to enable I2C interface
```



```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password Change password for the current user
2 Network Options      Configure network settings
3 Boot Options         Configure options for start-up
4 Localisation Options Set up language and regional settings to match your location
5 Interfacing Options  Configure connections to peripherals
6 Overclock            Configure overclocking for your Pi
7 Advanced Options     Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

            <Select>                                <Finish>
```



```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
P2 SSH         Enable/Disable remote command line access to your Pi using SSH
P3 VNC         Enable/Disable graphical remote access to your Pi using RealVNC
P4 SPI         Enable/Disable automatic loading of SPI kernel module
P5 I2C         Enable/Disable automatic loading of I2C kernel module
P6 Serial      Enable/Disable shell and kernel messages on the serial connection
P7 1-Wire      Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to GPIO pins

            <Select>                                <Back>
```



```
Would you like the ARM I2C interface to be enabled?

            <Yes>                <No>
```

And then reboot the system:

```
sudo reboot
```

Libraries Installation

- Install BCM2835 libraries

```
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.68.tar.gz
tar zxvf bcm2835-1.68.tar.gz
```

```
cd bcm2835-1.68/
sudo ./configure && sudo make && sudo make check && sudo make install
```

For more details, please refer to http://www.airspayce.com/mikem/bcm2835/

- Install wiringPi libraries

```
sudo apt-get install wiringpi

#For Pi 4, you need to update it：
wget https://project-downloads.drogon.net/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
gpio -v
#You will get 2.52 information if you install it correctly
```

Download the demo

Open the terminal of the Raspberry Pi, execute command to download demo codes:

```
sudo apt-get install p7zip-full
wget https://www.waveshare.com/w/upload/b/b3/AS7341_Spectral_Color_Sensor_code.7z
7z x AS7341_Spectral_Color_Sensor_code.7z -r -o./AS7341_Spectral_Color_Sensor_code
sudo chmod 777 -R  AS7341_Spectral_Color_Sensor_code
```

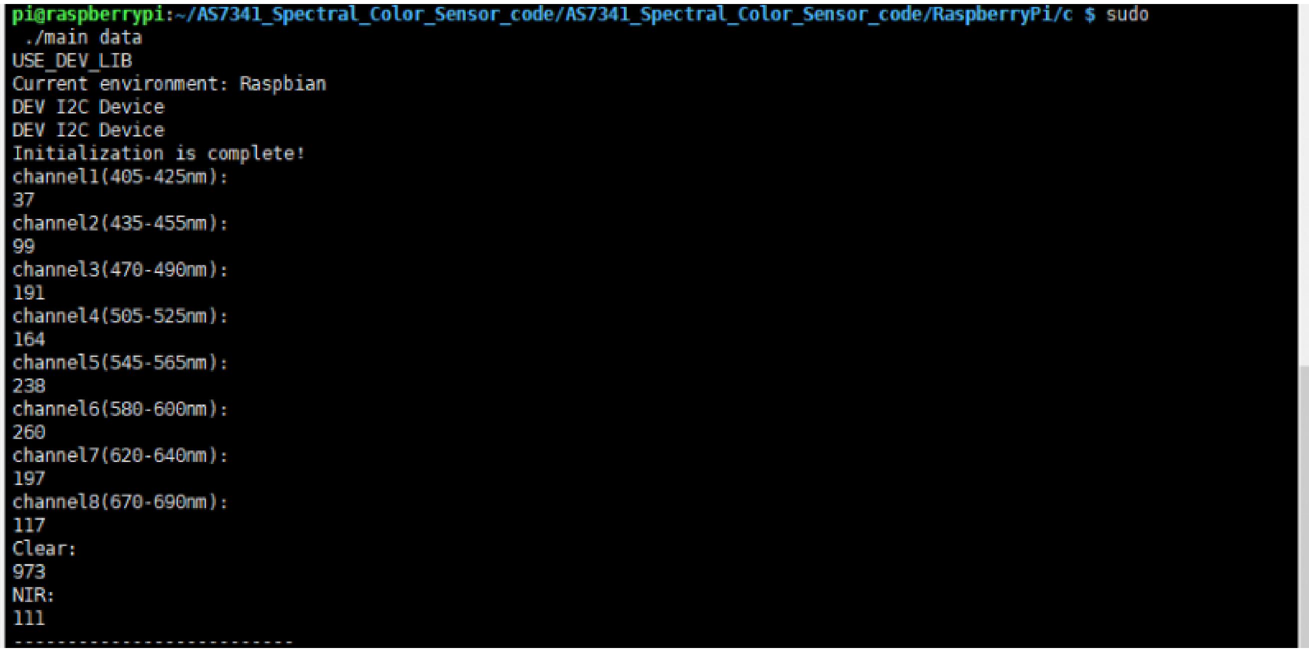
C

```
cd AS7341_Spectral_Color_Sensor_code/AS7341_Spectral_Color_Sensor_code/RaspberryPi/c
make clean
make
```

Enter the following command to execute the demo:

```
sudo ./main data
```

- 【Note】The 'data' here can be changed to flicker, syns, int, pinint, clear to verify different test demos, and its meaning is explained in the code.
- Take the execution of sudo ./main data as an example, the test result is:



```
pi@raspberrypi:~/AS7341_Spectral_Color_Sensor_code/AS7341_Spectral_Color_Sensor_code/RaspberryPi/c $ sudo
 ./main data
USE_DEV_LIB
Current environment: Raspbian
DEV I2C Device
DEV I2C Device
Initialization is complete!
channel1(405-425nm):
37
channel2(435-455nm):
99
channel3(470-490nm):
191
channel4(505-525nm):
164
channel5(545-565nm):
238
channel6(580-600nm):
260
channel7(620-640nm):
197
channel8(670-690nm):
117
Clear:
973
NIR:
111
--------------------------
```

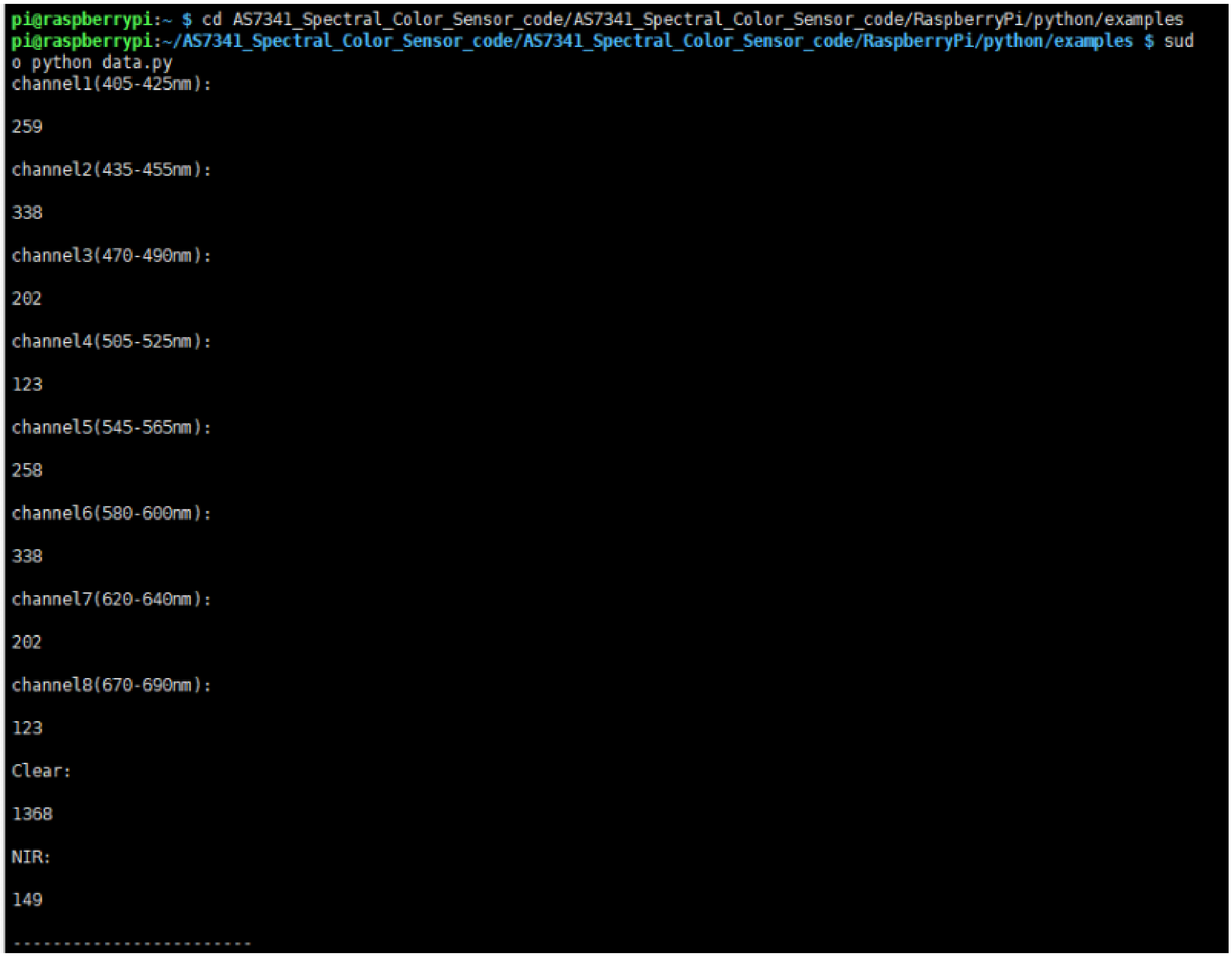
python

```
cd

cd AS7341_Spectral_Color_Sensor_code/AS7341_Spectral_Color_Senso
r_code/RaspberryPi/python/examples
```

Enter the following command to execute the demo:

```
sudo python data.py
```

- 【Note】The 'data' here can be changed to flicker, syns, int, pinint, clear to verify different test demos, and its meaning is explained in the code.
- Take the execution of data.py as an example, the test result is:



```
pi@raspberrypi:~ $ cd AS7341_Spectral_Color_Sensor_code/AS7341_Spectral_Color_Sensor_code/RaspberryPi/python/examples
pi@raspberrypi:~/AS7341_Spectral_Color_Sensor_code/AS7341_Spectral_Color_Sensor_code/RaspberryPi/python/examples $ sud
o python data.py
channel1(405-425nm):

259

channel2(435-455nm):

338

channel3(470-490nm):

202

channel4(505-525nm):

123

channel5(545-565nm):

258

channel6(580-600nm):

338

channel7(620-640nm):

202

channel8(670-690nm):

123

Clear:

1368

NIR:

149

-----------------------
```

Demo description

The functions of all test demos and the points that need attention have been introduced in the Arduino tutorial. When executing sudo ./main syns or python syns.py, you need to pull up the GPIO port and then pull it down to generate a falling edge signal. You can connect the GPIO pin to the high-level pin for a short time and then released to generate a falling edge signal.

Resources

Document

- Schematic

Demo

- demo

Datasheet

- AS7341 datasheet

Software

- Arduino IDE
- Sscom