Data Reading

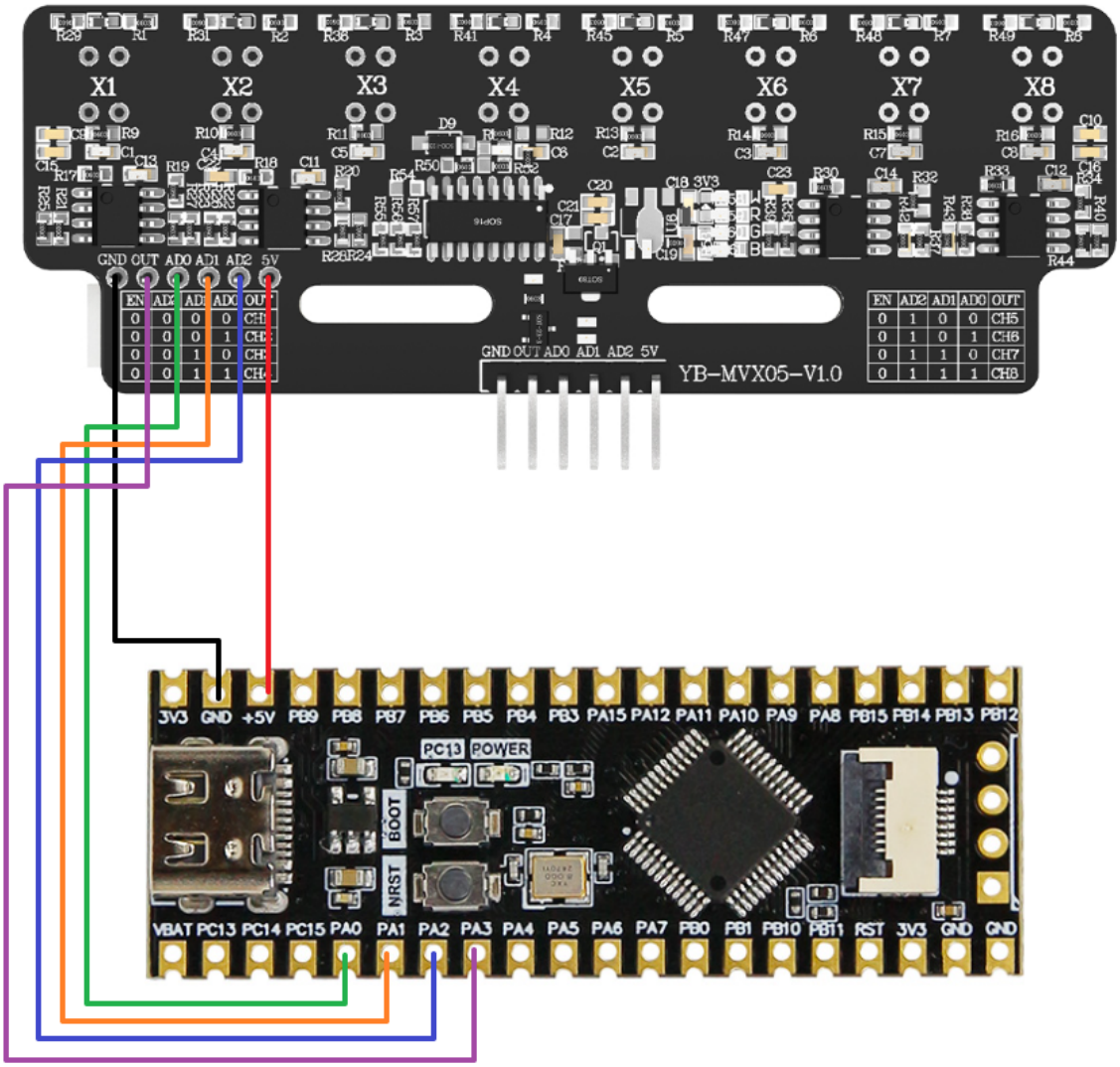
1. Quick Start

This tutorial explains how to use the STM32F103C8T6 motherboard to read the digital information from an eight-channel grayscale line-following module and print the results via a serial port assistant.

The hardware used in this tutorial is the STM32F103C8T6 core board and eight-channel grayscale line-following module sold by Yabo. Refer to the wiring diagram below to connect the wires, then program the motherboard to obtain the data. Modules not from Yabo are for reference only.

2. Hardware Wiring

The STM32 motherboard's Type-C port needs to be connected to the computer's USB port.

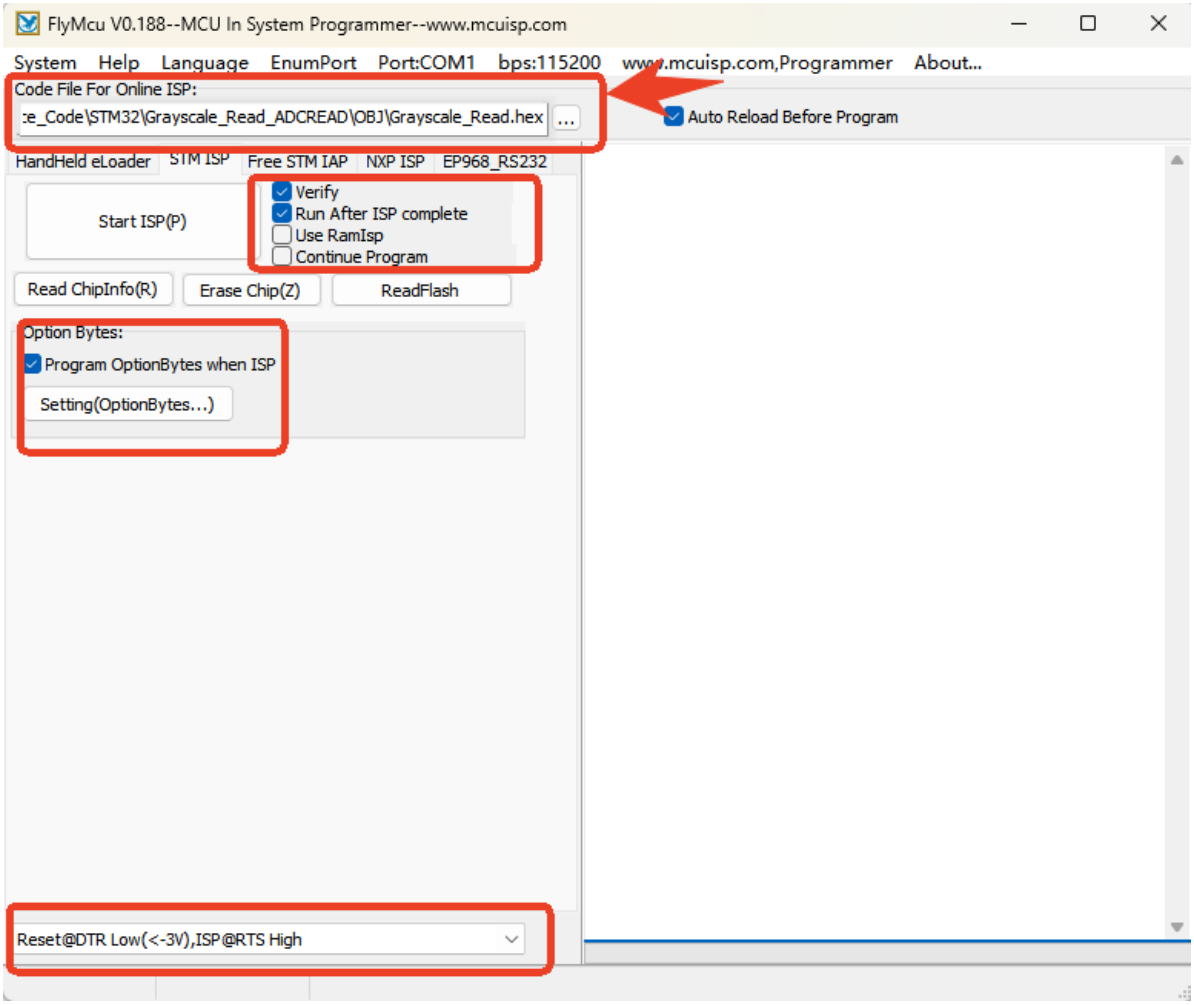


Eight-channel grayscale tracking module	STM32F103C8T6
5V	5V
GND	GND
AD0	PA0
AD1	PA1
AD2	PA2
OUT	PA3

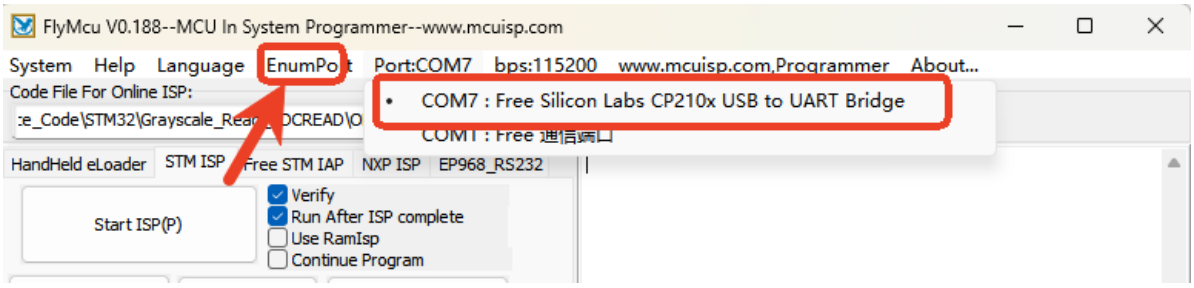
The eight-channel grayscale module sold by Yabo uses an XH2.54 to 6-pin DuPont cable for connection:

3. Usage Method

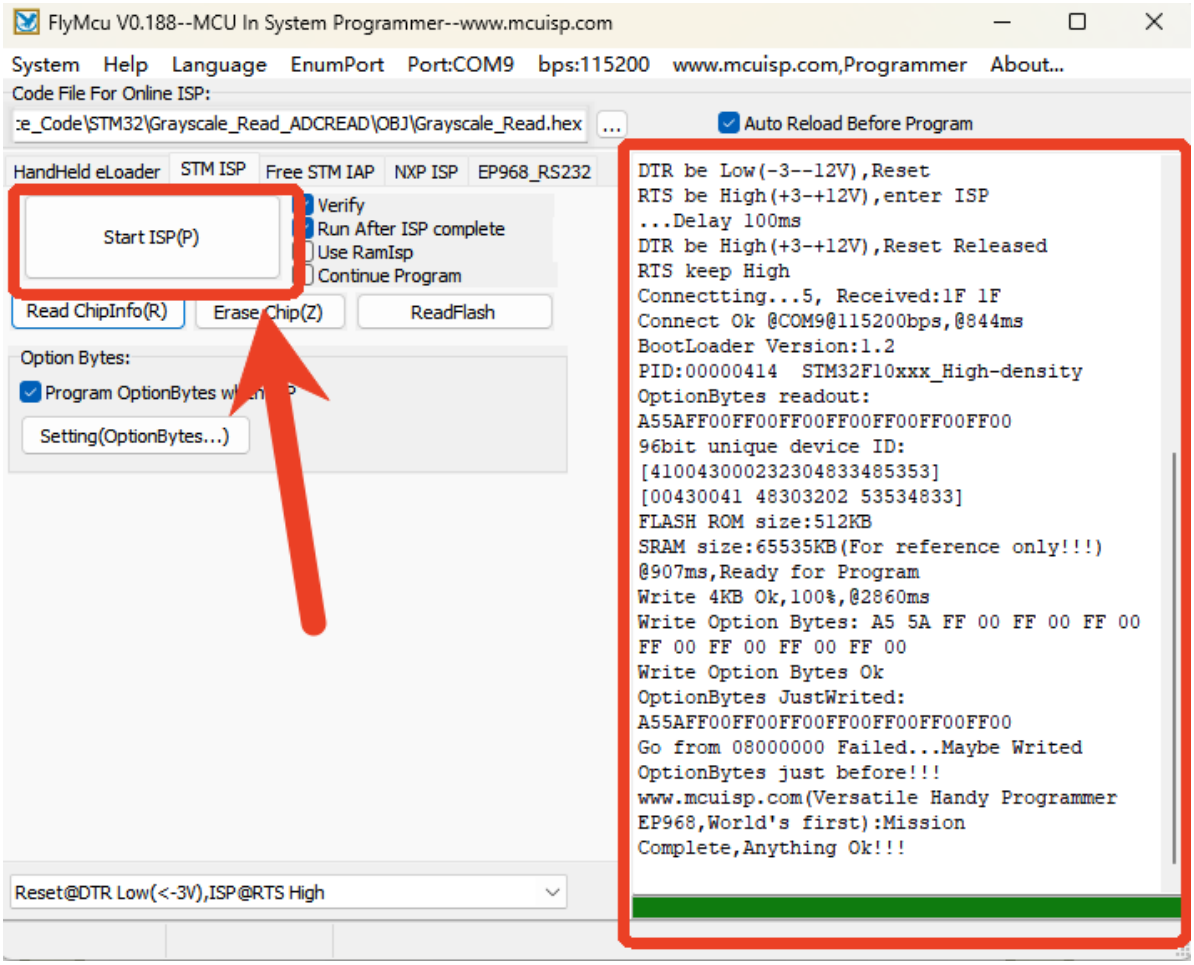
1. Open the FlyMcu programming tool. Select the hex file in the code project. The areas highlighted in red should match the image.



2. Connect the STM32 to the computer. Click "Search Serial Ports" and select the COM port corresponding to the STM32.



3. Then click "Start Programming." The program will begin to be burned into the motherboard. Wait for the progress bar on the right to complete, and you will see a message indicating successful burning.



4. Press the NRST reset button on the motherboard to restart the motherboard and run the program. Then open the serial port assistant to view the printed information.

4. Phenomenon and Results

In the serial port assistant, you can see the continuously printed digital values of the eight-channel grayscale tracking module. X1 corresponds to X1 on the module. When the X1 LED is lit, the corresponding value is 1.

```
[2025-11-08 17:18:48.215]# RECV ASCII>
[ X1:1 ][ X2:1 ][ X3:1 ][ X4:1 ][ X5:1 ][ X6:1 ][ X7:1 ][ X8:1 ]


[2025-11-08 17:18:48.312]# RECV ASCII>
[ X1:1 ][ X2:1 ][ X3:1 ][ X4:1 ][ X5:1 ][ X6:1 ][ X7:1 ][ X8:1 ]


[2025-11-08 17:18:48.407]# RECV ASCII>
[ X1:1 ][ X2:1 ][ X3:1 ][ X4:1 ][ X5:1 ][ X6:1 ][ X7:1 ][ X8:1 ]


[2025-11-08 17:18:48.502]# RECV ASCII>
[ X1:1 ][ X2:1 ][ X3:1 ][ X4:1 ][ X5:1 ][ X6:1 ][ X7:1 ][ X8:1 ]


[2025-11-08 17:18:48.613]# RECV ASCII>
[ X1:1 ][ X2:1 ][ X3:1 ][ X4:1 ][ X5:1 ][ X6:1 ][ X7:1 ][ X8:1 ]


[2025-11-08 17:18:48.709]# RECV ASCII>
[ X1:1 ][ X2:1 ][ X3:1 ][ X4:1 ][ X5:1 ][ X6:1 ][ X7:1 ][ X8:1 ]
```

5. Code Explanation

```c
// main.c
int main(void)
{
    bsp_init(); // Initialize onboard peripherals, including the serial port and
delay timer.
    Grayscale_Sensor_Init(); // Initialize grayscale sensor GPIO

    while(1)
    {
        Grayscale_Sensor_Read_All(g_sensor_data);

        for (i = 0; i < GRAYSCALE_SENSOR_CHANNELS; i++)
        {
            printf("[ %s:%d ]", sensor_labels[i], g_sensor_data[i]);
        }
        printf("\n");
        delay_ms(60);
    }
}
```

- `main`: The main entry point of the program. First, `bsp_init()` is called to initialize all Board Support Package (BSP) hardware, including the serial port and Systick delay. Then, `Grayscale_Sensor_Init()` is called to initialize the sensor I/O ports. In an infinite loop, the program continuously calls `Grayscale_Sensor_Read_All()` to acquire sensor data and prints it out using `printf`.

```c
// grayscale_sensor.c
void Grayscale_Sensor_Init(void)
{
```

```
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}

void Grayscale_Sensor_Read_All(uint16_t* sensor_values)
{
    uint8_t i;
    for (i = 0; i < GRAYSCALE_SENSOR_CHANNELS; i++)
    {
        _select_channel(i);
        delay_us(50);
        sensor_values[i] = Read_OUT_value();
    }
}

static void _select_channel(uint8_t channel)
{
    SENSOR_AD0_WRITE((channel >> 0) & 0x01);  // bit0 -> AD0
    SENSOR_AD1_WRITE((channel >> 1) & 0x01);  // bit1 -> AD1
    SENSOR_AD2_WRITE((channel >> 2) & 0x01);  // bit2 -> AD2
}
```

- `Grayscale_Sensor_Init`: Initializes GPIO using STM32 standard peripheral library functions. First, it enables the clock on `GPIOA`, then configures `PA0-PA2` as push-pull outputs for channel selection, and configures `PA3` as a floating input for data reading.
- `Grayscale_Sensor_Read_All`: Iterates through all 8 sensor channels, selects a channel using `_select_channel`, reads its value, and stores it in an array.
- `_select_channel`: Controls the GPIO level using the `SENSOR_ADx_WRITE` macro to select the corresponding sensor.