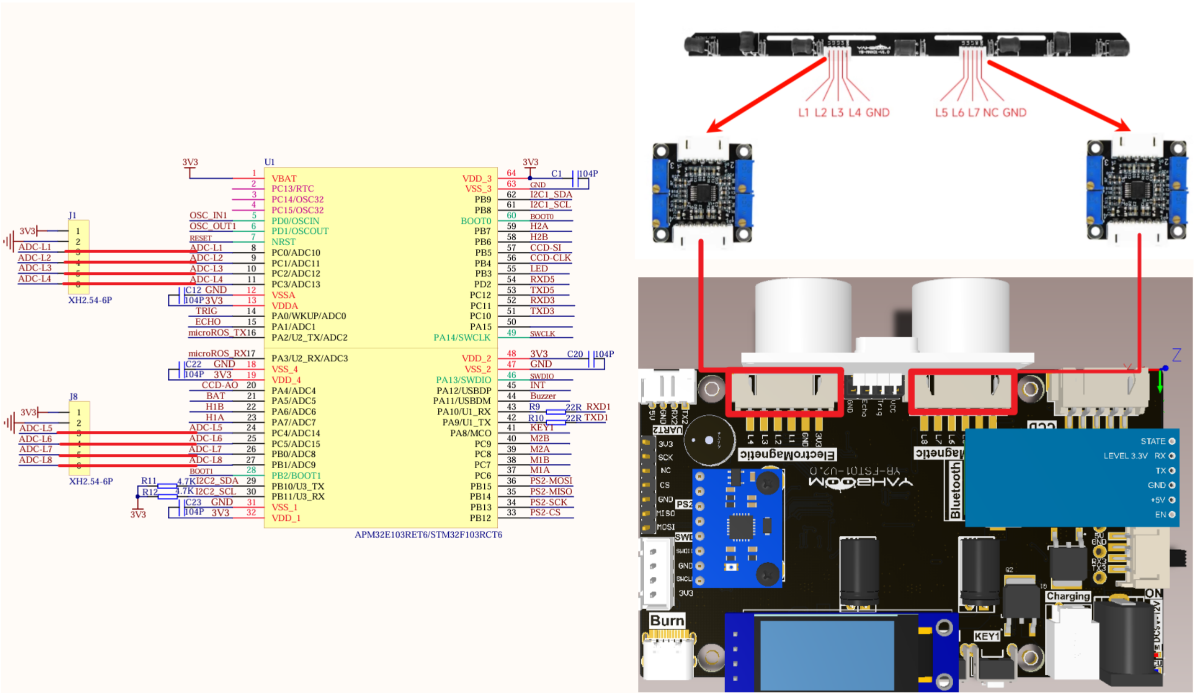
Electromagnetic module-Reading data (ADC)

The tutorial combines OLED and serial port to display the data output by the electromagnetic line patrol module.

```
The tutorial only introduces the standard library project code
```

Hardware connection



Since we have configured a special connection line, we only need to install it to the corresponding interface:

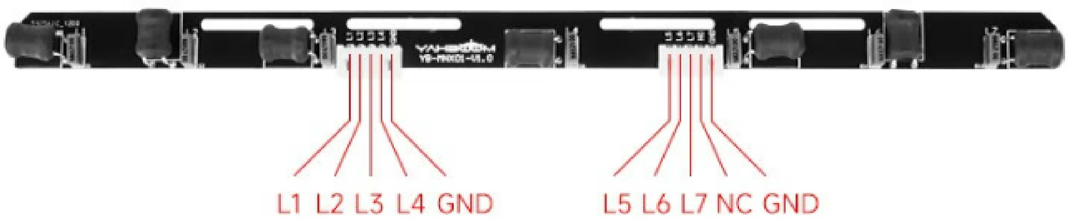
Peripherals	Development board
Electromagnetic line patrol: ADC1-L1	PC0
Electromagnetic line patrol: ADC2-L2	PC1
Electromagnetic line patrol: ADC3-L3	PC2
Electromagnetic line patrol: ADC4-L4	PC3

Peripherals	Development board
Electromagnetic line patrol: ADC5-L5	PC4
Electromagnetic line patrol: ADC6-L6	PC5
Electromagnetic line patrol: ADC7-L7	PB0
Electromagnetic line patrol: NC	Not connected
Electromagnetic line patrol: GND	GND
Electromagnetic line patrol: GND	GND

Control principle

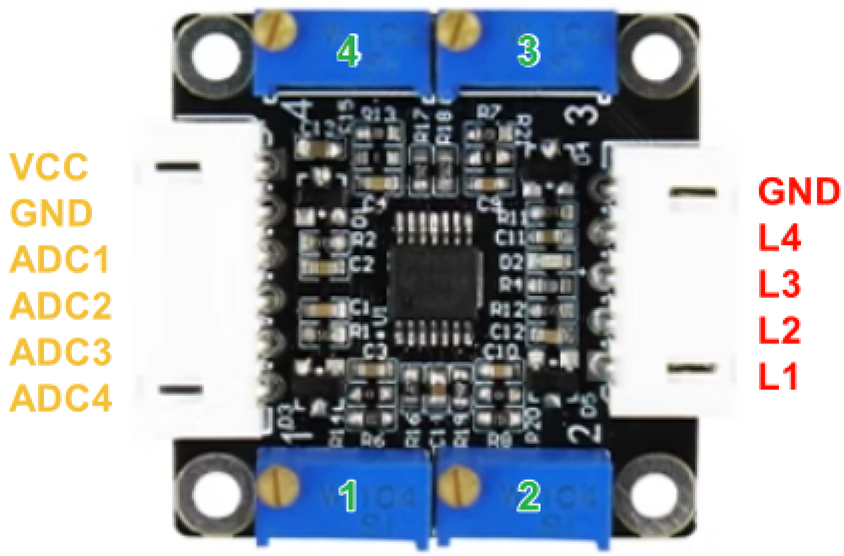
The electromagnetic line patrol module consists of an electromagnetic line patrol probe module and an operational amplifier.

- Electromagnetic line patrol probe module

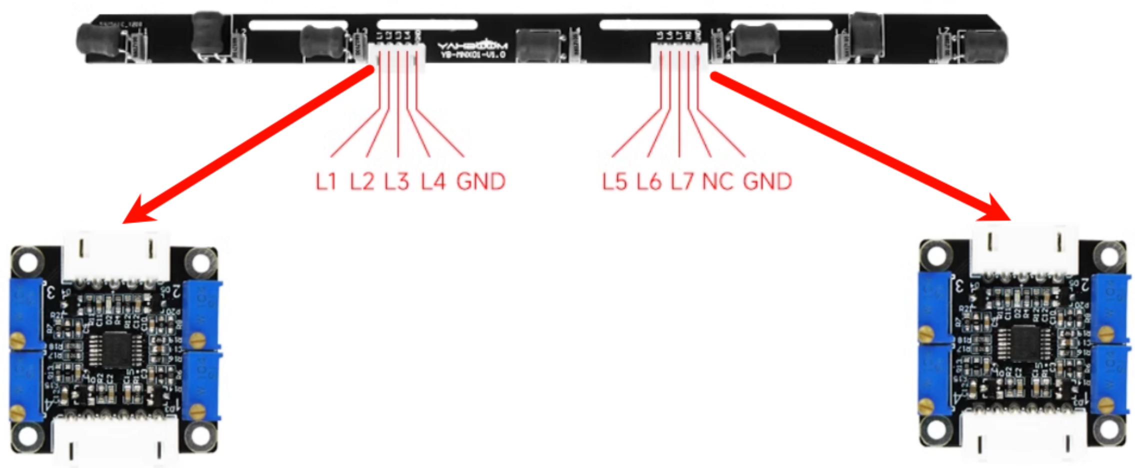


Peripherals	Description
Electromagnetic line patrol: L1-L7	Induction signal output pin: need to connect to operational amplifier
Electromagnetic line patrol: GND	Power supply pin: GND
Electromagnetic line patrol: NC	Do not connect

- Operational amplifier



Peripherals	Description
Operational amplifier: VCC	Power supply pin: 3.3-5V
Operational amplifier: GND	Power supply pin: GND
Operational amplifier: L1-L4	Sensing signal input pin: electromagnetic line patrol probe module
Operational amplifier: ADC1-ADC4	Gained signal: connected to MCU's IO port
Operational amplifier: 1-4	Gain potentiometer: gain adjustment



The gain potentiometer needs to be adjusted before use: Place each inductor of the electromagnetic line patrol probe module vertically in the same position directly above the enameled wire of the electromagnetic signal generator, and adjust the gain potentiometer so that the output voltage after the operational amplifier is basically consistent (the voltage is determined by the ADC data converted by the MCU).

Software configuration

Pin definition

Main control chip	Pin	Main function (after reset)	Default multiplexing function	Redefine function
STM32F103RCT6	PC0	PC0	ADC123_IN10	
STM32F103RCT6	PC1	PC1	ADC123_IN11	
STM32F103RCT6	PC2	PC2	ADC123_IN12	
STM32F103RCT6	PC3	PC3	ADC123_IN13	
STM32F103RCT6	PC4	PC4	ADC12_IN14	
STM32F103RCT6	PC5	PC5	ADC12_IN15	
STM32F103RCT6	PB0	PB0	ADC12_IN8/TIM3_CH3/TIM8_CH2N	TIM1_CH2N

Software code

Since the default function of the pin is the normal IO pin function, we need to use the default multiplexing function.

```
Product supporting materials source code path: Attachment → Source code summary →
2.Extended_Course → 10.ElE
```

Control function

The tutorial only briefly introduces the code, you can open the project source code to read it in detail.

> **ele_Init**

```
void  ele_Init(void)
{
    ADC_InitTypeDef ADC_InitStructure;
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC | RCC_APB2Periph_GPIOB |
ELE_ADC_CLK , ENABLE );
    RCC_ADCCLKConfig(RCC_PCLK2_Div6); //Set ADC division factor 6 72M/6=12, ADC
maximum time cannot exceed 14M
    //Set analog channel input pins

    //Left ADC 10,11,12
    GPIO_InitStructure.GPIO_Pin = ELE_L1_Pin | ELE_L2_Pin |ELE_L3_Pin;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD; //Analog input pins If it is
not 0 when it is floating, it can be changed to pull-down
    GPIO_Init(ELE_L1_Port, &GPIO_InitStructure); //All are GPIOC, just choose one

    //Middle ADC 13
    GPIO_InitStructure.GPIO_Pin = ELE_MID_Pin;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD; //Analog input pin If it is not
0 when it is floating, it can be changed to pull-down
```

```
    GPIO_Init(ELE_MID_Port, &GPIO_InitStructure);

    //Right ADC 14,15
    GPIO_InitStructure.GPIO_Pin = ELE_R1_Pin |ELE_R2_Pin;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD; //Analog input pin If it is not
0 when it is floating, it can be changed to pull-down
    GPIO_Init(ELE_R1_Port, &GPIO_InitStructure); //All are GPIOC, just choose one

    //ADC 8 on the right
    GPIO_InitStructure.GPIO_Pin = ELE_R3_Pin;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD; //Analog input pin is not 0
when it is floating, it can be changed to pull-down
    GPIO_Init(ELE_R3_Port, &GPIO_InitStructure);

    ADC_DeInit(ELE_ADC); //Reset ADC and reset all registers of peripheral ADC to
default values
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent; //ADC working mode: ADC1
and ADC2 work in independent mode
    ADC_InitStructure.ADC_ScanConvMode = DISABLE; //Analog-to-digital conversion
works in single-channel mode
    ADC_InitStructure.ADC_ContinuousConvMode = DISABLE; //Analog-to-digital
conversion works in single conversion mode
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
//Conversion is started by software instead of external trigger
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right; //ADC data right-
aligned
    ADC_InitStructure.ADC_NbrOfChannel = 1; //Number of ADC channels for regular
conversion in sequence
    ADC_Init(ELE_ADC, &ADC_InitStructure); //Initialize the registers of the
peripheral ADCx according to the parameters specified in ADC_InitStruct
    ADC_Cmd(ELE_ADC, ENABLE);
    ADC_ResetCalibration(ELE_ADC); //Enable reset calibration
    while(ADC_GetResetCalibrationStatus(ELE_ADC)); //Wait for reset calibration
to end
    ADC_StartCalibration(ELE_ADC); //Start AD calibration
    while(ADC_GetCalibrationStatus(ELE_ADC)); //Wait for calibration to end
}
```

> **Get_Adc_ele**

```
u16 Get_Adc_ele(u8 ch)
{
    //Set the specified ADC rule group channel, a sequence, sampling time
    ADC_RegularChannelConfig(ELE_ADC, ch, 1, ADC_SampleTime_239Cycles5 ); //ADC1,
ADC channel, sampling time is 239.5 cycles
    ADC_SoftwareStartConvCmd(ELE_ADC, ENABLE); //Enable the software conversion
start function of the specified ADC1
    while(!ADC_GetFlagStatus(ELE_ADC, ADC_FLAG_EOC ));//Wait for the conversion
to end
    return ADC_GetConversionValue(ELE_ADC); //Return the most recent conversion
result of the ADC1 rule group
}
```

> **guiyi_way**

```
int guiyi_way(void)
{
    int sum , Sensor;
    int Sensor_Left,Sensor_Right;

    // Normalized processing
    sum=(Sensor_Left_1*1+Sensor_Left_3*100)
    + Sensor_Middle *200
    +(Sensor_Right_1*300+Sensor_Right_3*399);
    Sensor_Left =  Sensor_Left_1+Sensor_Left_3;
    Sensor_Right = Sensor_Right_1+Sensor_Right_3;

    //  sum=(Sensor_Left_3*1)
    //          + Sensor_Middle *100
    //          +(Sensor_Right_1*199);

    Sensor_Left =  Sensor_Left_3+ Sensor_Left_1 + Sensor_Left_2;
    Sensor_Right = Sensor_Right_1+ Sensor_Right_3 + Sensor_Right_2;

    Sensor=sum/(Sensor_Left+Sensor_Middle+Sensor_Right);    // Find the deviation
    return Sensor; // Return the current position in the magnetic field
}
```

> **getEleData**

```
void getEleData(void)
{
    // With the rear of the car facing you, count from left to right
    Sensor_Left_1=Get_Adc_ele(ELE_L1_CH)>>4;                 // Collect the data
of the left inductor
    Sensor_Left_2=Get_Adc_ele(ELE_L2_CH)>>4;
    Sensor_Left_3=Get_Adc_ele(ELE_L3_CH)>>4;                 // Collect the data
of the left inductor

    Sensor_Middle=Get_Adc_ele(ELE_M1_CH)>>4;               // Collect
intermediate inductance data

    Sensor_Right_1=Get_Adc_ele(ELE_R1_CH)>>4;                // Collect the data
of the right inductor
    Sensor_Right_2=Get_Adc_ele(ELE_R2_CH)>>4;
    Sensor_Right_3=Get_Adc_ele(ELE_R3_CH)>>4;                // Collect the data
of the right inductor

    ele_seat = guiyi_way();
}
```

Experimental phenomenon

The ELE.hex file generated by the project compilation is located in the OBJ folder of the ELE project. Find the ELE.hex file corresponding to the project and use the FlyMcu software to download the program to the development board.

After the program is successfully downloaded: OLED and serial port display the data output by the electromagnetic patrol module.

```
OLED displays L3 L2 L1 R1 R2 R3 sensor data
Serial port displays threshold and intermediate sensor data
```