11. Nine-axis attitude sensor to obtain data

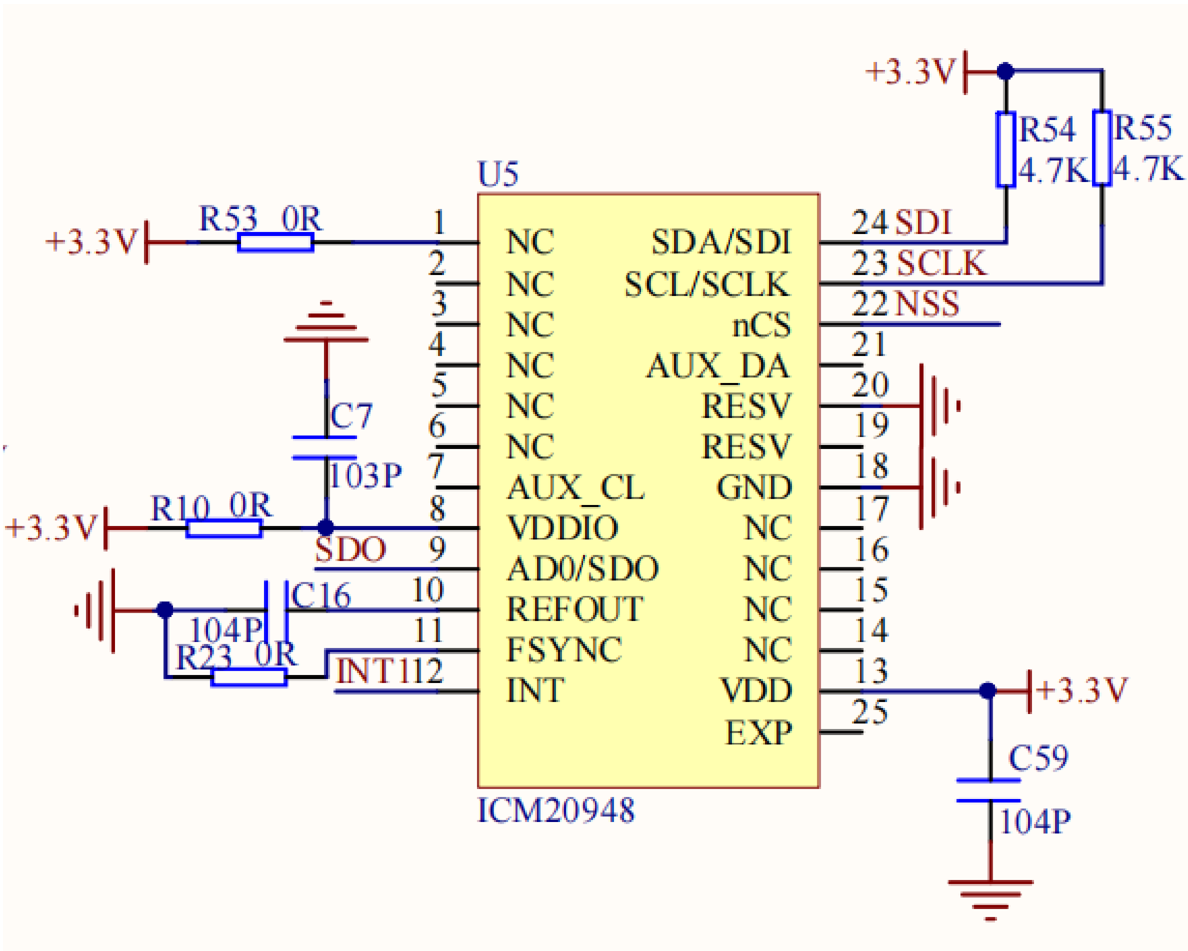
11.1. Experimental purpose

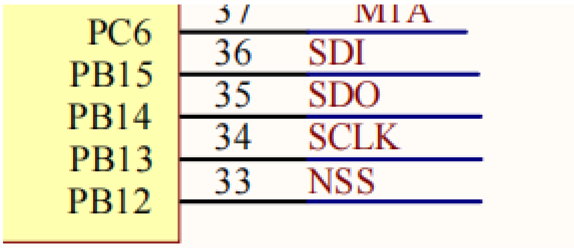
Use the GPIO port of STM32 to simulate IIC communication, read the raw data of the nine-axis attitude sensor MPU9250, and print it out through the serial port assistant.

11.2. Configuration pin information

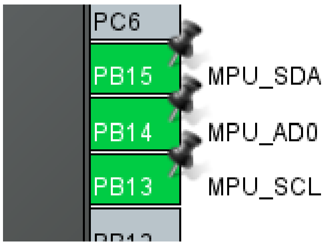
1. Import the ioc file from the Serial project and name it Read_IMU.

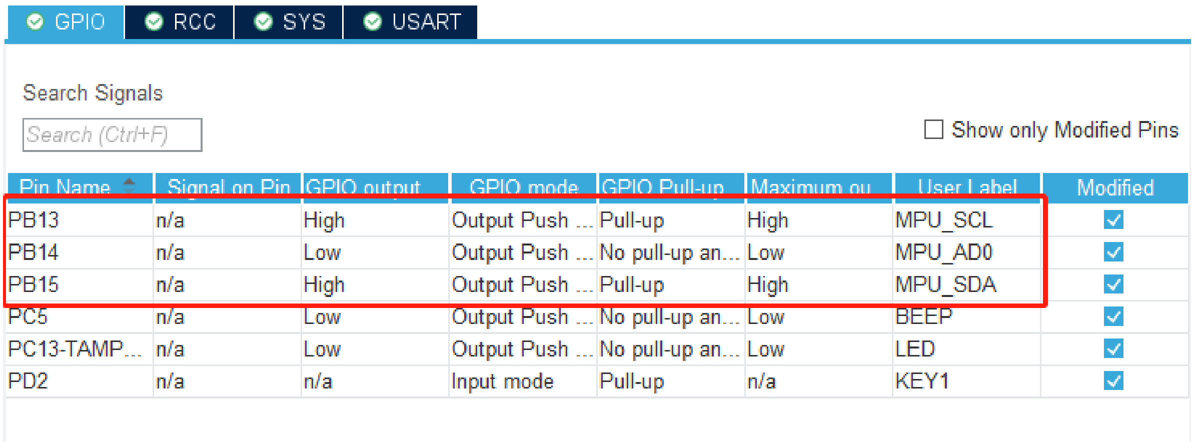
According to the schematic diagram, the SDA/SDI pin of the nine-axis attitude sensor is connected to PB15, the SCL/SCLK pin is connected to PB13, and the AD0/SDO pin is connected to PB14.





2. Set PB13, PB14 and PB15 as output mode, the specific parameters are as shown in the figure below:





GPIO RCC SYS USART

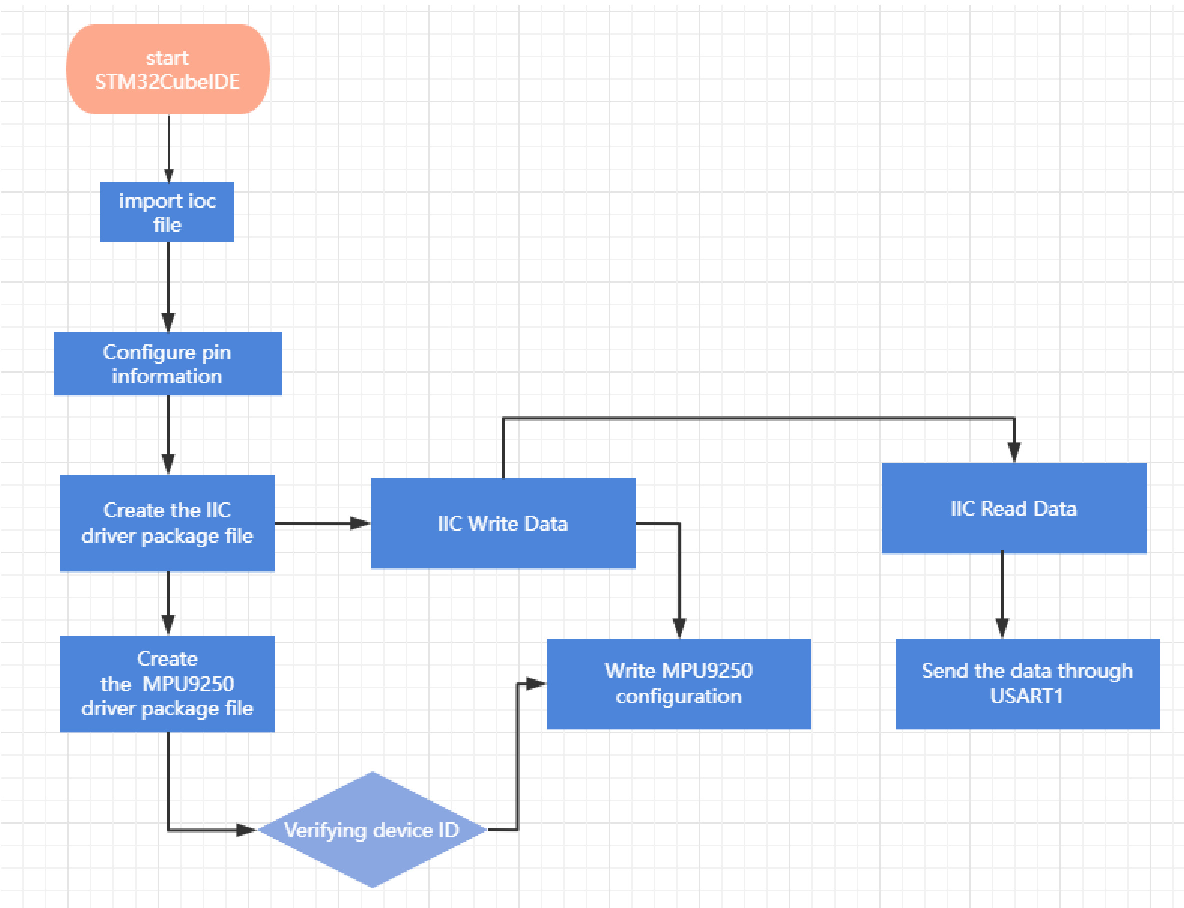
Search Signals

Search (Ctrl+F)

Show only Modified Pins

Pin Name	Signal on Pin	GPIO output	GPIO mode	GPIO Pull-up	Maximum ou	User Label	Modified
PB13	n/a	High	Output Push ...	Pull-up	High	MPU_SCL	☑
PB14	n/a	Low	Output Push ...	No pull-up an...	Low	MPU_AD0	☑
PB15	n/a	High	Output Push ...	Pull-up	High	MPU_SDA	☑
PC5	n/a	Low	Output Push ...	No pull-up an...	Low	BEEP	☑
PC13-TAMP...	n/a	Low	Output Push ...	No pull-up an...	Low	LED	☑
PD2	n/a	n/a	Input mode	Pull-up	n/a	KEY1	☑

11.3. Analysis of the experimental flow chart



11.4. core code explanation

1. Create new bsp_mpuiic.h and bsp_mpuiic.c, and add the following content to bsp_mpuiic.h:

```
// SCL PB13, SDA PB15
#define MPU_SDA_IN()                    \
    {                                   \
        GPIOB->CRH &= 0X0FFFFFFF;       \
        GPIOB->CRH |= (uint32_t)8 << 28; \
    }
#define MPU_SDA_OUT()                   \
    {                                   \
        GPIOB->CRH &= 0X0FFFFFFF;       \
        GPIOB->CRH |= (uint32_t)3 << 28; \
    }

#define MPU_IIC_SCL(a) HAL_GPIO_WritePin(MPU_SCL_GPIO_Port, MPU_SCL_Pin, a)
#define MPU_IIC_SDA(a) HAL_GPIO_WritePin(MPU_SDA_GPIO_Port, MPU_SDA_Pin, a)
#define READ_SDA HAL_GPIO_ReadPin(MPU_SDA_GPIO_Port, MPU_SDA_Pin)


void MPU_IIC_Delay(void);
void MPU_IIC_Init(void);
void MPU_IIC_Start(void);
void MPU_IIC_Stop(void);
void MPU_IIC_Send_Byte(uint8_t txd);
uint8_t MPU_IIC_Read_Byte(unsigned char ack);
uint8_t MPU_IIC_Wait_Ack(void);
void MPU_IIC_Ack(void);
void MPU_IIC_NAck(void);
```

2. Create the following content in the bsp_mpuiic.c file:

According to the content of the IIC protocol, MPU_IIC_Start() generates the IIC start signal, and MPU_IIC_Stop() generates the IIC stop signal.

```
// Generates the IIC initiation signal  产生IIC起始信号
void MPU_IIC_Start(void)
{
    MPU_SDA_OUT();
    MPU_IIC_SDA(1);
    MPU_IIC_SCL(1);
    delay_us(4);
    MPU_IIC_SDA(0);
    delay_us(4);
    MPU_IIC_SCL(0);
}

// Generates an IIC stop signal  产生IIC停止信号
void MPU_IIC_Stop(void)
{
    MPU_SDA_OUT();
    MPU_IIC_SCL(0);
    MPU_IIC_SDA(0);
    delay_us(4);
    MPU_IIC_SCL(1);
    MPU_IIC_SDA(1);
    delay_us(4);
}
```

3.IIC response correlation function.

```
// 等待应答信号到来
// 返回值：1，接收应答失败．0，接收应答成功
// Wait for the answer signal to arrive.
// Return value: 1, receive and reply failed 0, receive and reply succeeded
uint8_t MPU_IIC_Wait_Ack(void)
{
    uint8_t ucErrTime=0;
    MPU_SDA_IN();
    MPU_IIC_SDA(1);delay_us(1);
    MPU_IIC_SCL(1);delay_us(1);
    while(READ_SDA)
    {
        ucErrTime++;
        if(ucErrTime>250)
        {
            MPU_IIC_Stop();
            return 1;
        }
    }
    MPU_IIC_SCL(0);
    return 0;
}

// Generate AN ACK reply  产生ACK应答
void MPU_IIC_Ack(void)
{
    MPU_IIC_SCL(0);
    MPU_SDA_OUT();
    MPU_IIC_SDA(0);
    delay_us(2);
    MPU_IIC_SCL(1);
    delay_us(2);
    MPU_IIC_SCL(0);
}
// No ACK response is generated  不产生ACK应答
void MPU_IIC_NAck(void)
{
    MPU_IIC_SCL(0);
    MPU_SDA_OUT();
    MPU_IIC_SDA(1);
    delay_us(2);
    MPU_IIC_SCL(1);
    delay_us(2);
    MPU_IIC_SCL(0);
}
```

4.IIC send and read data related functions.

```c
// IIC发送一个字节，返回从机有无应答，1，有应答，0，无应答
// The IIC sends a byte that returns whether the slave machine answered, 1, yes, 0, no
void MPU_IIC_Send_Byte(uint8_t txd)
{
    uint8_t t;
    MPU_SDA_OUT();
    MPU_IIC_SCL(0);
    for(t=0;t<8;t++)
    {
        MPU_IIC_SDA((txd&0x80)>>7);
        txd<<=1;
        delay_us(2);
        MPU_IIC_SCL(1);
        delay_us(2);
        MPU_IIC_SCL(0);
        delay_us(2);
    }
}
// 读1个字节，ack=1时，发送ACK，ack=0，发送nACK
// Read 1 byte, ack=1, send ACK, ack=0, send nACK
uint8_t MPU_IIC_Read_Byte(unsigned char ack)
{
    unsigned char i,receive=0;
    MPU_SDA_IN();
    for(i=0;i<8;i++ )
    {
        MPU_IIC_SCL(0);
        delay_us(2);
        MPU_IIC_SCL(1);
        receive<<=1;
        if(READ_SDA)receive++;
        delay_us(1);
    }
    if (!ack)
        MPU_IIC_NAck();
    else
        MPU_IIC_Ack();
    return receive;
}
```

5. Create new bsp_mpu9250.h and bsp_mpu9250.c, and add the following content to bsp_mpu9250.h:

```c
uint8_t MPU9250_Init(void);
uint8_t MPU_Write_Byte(uint8_t devaddr,uint8_t reg,uint8_t data);
uint8_t MPU_Read_Byte(uint8_t devaddr,uint8_t reg);
uint8_t MPU_Set_Gyro_Fsr(uint8_t fsr);
uint8_t MPU_Set_Accel_Fsr(uint8_t fsr);
uint8_t MPU_Set_Rate(uint16_t rate);
uint8_t MPU_Write_Len(uint8_t addr,uint8_t reg,uint8_t len,uint8_t *buf);
uint8_t MPU_Read_Len(uint8_t addr,uint8_t reg,uint8_t len,uint8_t *buf);

uint8_t MPU_Get_Gyroscope(int16_t *gx, int16_t *gy, int16_t *gz);
uint8_t MPU_Get_Accelerometer(int16_t *ax, int16_t *ay, int16_t *az);
uint8_t MPU_Get_Magnetometer(int16_t *mx, int16_t *my, int16_t *mz);

void MPU9250_Read_Data_Handle(void);
void MPU_Delay_ms(uint16_t time);
```

6. Add the following related functions in bsp_mpu9250.c.

Pull the AD0 pin low to make the ID of the MPU6500 0x68.

```
// 拉低AD0引脚，让MPU6500的ID为0x68
// Lower the AD0 pin so that the ID of the MPU6500 is 0x68
void MPU_ADDR_CTRL(void)
{
    HAL_GPIO_WritePin(MPU_AD0_GPIO_Port, MPU_AD0_Pin, GPIO_PIN_RESET);
}
```

Initialize MPU9250, return value: 0, success, other, error code

```
uint8_t MPU9250_Init(void)
{
    MPU_ADDR_CTRL();
    MPU_IIC_Init();
    MPU_Delay_ms(10);

    uint8_t res = 0;
    // Reset MPU9250 //复位MPU9250
    MPU_Write_Byte(MPU9250_ADDR, MPU_PWR_MGMT1_REG, 0X80);
    // Delay 100 ms //延时100ms
    MPU_Delay_ms(100);
    // Wake mpu9250 //唤醒MPU9250
    MPU_Write_Byte(MPU9250_ADDR, MPU_PWR_MGMT1_REG, 0X00);

    // Gyroscope sensor  陀螺仪传感器,±500dps=±500°/s ±32768 (gyro/32768*500)*PI/180(rad/s)=gyro/3754.9(rad/s)
    MPU_Set_Gyro_Fsr(1);
    // Acceleration sensor 加速度传感器,±2g=±2*9.8m/s^2 ±32768 accel/32768*19.6=accel/1671.84
    MPU_Set_Accel_Fsr(0);
    // Set the sampling rate to 50Hz //设置采样率50Hz
    MPU_Set_Rate(50);

    // Turn off all interrupts //关闭所有中断
    MPU_Write_Byte(MPU9250_ADDR, MPU_INT_EN_REG, 0X00);
    // The I2C main mode is off //I2C主模式关闭
    MPU_Write_Byte(MPU9250_ADDR, MPU_USER_CTRL_REG, 0X00);
    // Close the FIFO //关闭FIFO
    MPU_Write_Byte(MPU9250_ADDR, MPU_FIFO_EN_REG, 0X00);
    // The INT pin is low, enabling bypass mode to read the magnetometer directly
    // INT引脚低电平有效，开启bypass模式，可以直接读取磁力计
    MPU_Write_Byte(MPU9250_ADDR, MPU_INTBP_CFG_REG, 0X82);
    // Read the ID of MPU9250  读取MPU9250的ID
    res = MPU_Read_Byte(MPU9250_ADDR, MPU_DEVICE_ID_REG);
    printf("MPU6500 Read ID=0x%02X\n", res);
    // Check whether the device ID is correct 判断器件ID是否正确
    if (res == MPU6500_ID1 || res == MPU6500_ID2)
    {
        // Set CLKSEL,PLL X axis as reference //设置CLKSEL,PLL X轴为参考
        MPU_Write_Byte(MPU9250_ADDR, MPU_PWR_MGMT1_REG, 0X01);
        // Acceleration and gyroscope both work //加速度与陀螺仪都工作
        MPU_Write_Byte(MPU9250_ADDR, MPU_PWR_MGMT2_REG, 0X00);
        // Set the sampling rate to 50Hz //设置采样率为50Hz
        MPU_Set_Rate(50);
    }
    else
        return 1;
    // Read AK8963ID 读取AK8963ID
    res = MPU_Read_Byte(AK8963_ADDR, MAG_WIA);
    printf("AK8963 Read ID=0x%02X\n", res);
    if (res == AK8963_ID)
    {
        // Set AK8963 to single measurement mode 设置AK8963为单次测量模式
        MPU_Write_Byte(AK8963_ADDR, MAG_CNTL1, 0X11);
    }
```

Read the gyroscope value (original value), return value: 0, success, other, error code

```
// 读取陀螺仪值(原始值), 返回值:0,成功, 其他,错误代码
// Read gyroscope value (original value), return value :0, success, other, error code
uint8_t MPU_Get_Gyroscope(int16_t *gx, int16_t *gy, int16_t *gz)
{
    uint8_t buf[6], res;
    res = MPU_Read_Len(MPU9250_ADDR, MPU_GYRO_XOUTH_REG, 6, buf);
    if (res == 0)
    {
        *gx = ((uint16_t)buf[0] << 8) | buf[1];
        *gy = ((uint16_t)buf[2] << 8) | buf[3];
        *gz = ((uint16_t)buf[4] << 8) | buf[5];
    }
    return res;
}
```

Read acceleration value (original value), return value: 0, success, other, error code

```
// 读取加速度值(原始值), 返回值:0,成功, 其他,错误代码
// Read acceleration value (original value), return value :0, success, other, error code
uint8_t MPU_Get_Accelerometer(int16_t *ax, int16_t *ay, int16_t *az)
{
    uint8_t buf[6], res;
    res = MPU_Read_Len(MPU9250_ADDR, MPU_ACCEL_XOUTH_REG, 6, buf);
    if (res == 0)
    {
        *ax = ((uint16_t)buf[0] << 8) | buf[1];
        *ay = ((uint16_t)buf[2] << 8) | buf[3];
        *az = ((uint16_t)buf[4] << 8) | buf[5];
    }
    return res;
}
```

Read magnetometer value (raw value), return value: 0, success, other, error code

```
// 读取磁力计值(原始值)，返回值:0,成功，其他,错误代码
// Read magnetometer value (original value), return value :0, success, other, error code
uint8_t MPU_Get_Magnetometer(int16_t *mx, int16_t *my, int16_t *mz)
{
    uint8_t buf[6], res;
    res = MPU_Read_Len(AK8963_ADDR, MAG_XOUT_L, 6, buf);
    if (res == 0)
    {
        *mx = ((uint16_t)buf[1] << 8) | buf[0];
        *my = ((uint16_t)buf[3] << 8) | buf[2];
        *mz = ((uint16_t)buf[5] << 8) | buf[4];
    }
    // AK8963每次读完以后都需要重新设置为单次测量模式
    // AK8963 needs to be reset to single measurement mode after each reading
    MPU_Write_Byte(AK8963_ADDR, MAG_CNTL1, 0X11);
    return res;
}
```

Read and print data, called every 10ms.

```
// Read and print the data  读取并打印数据
void MPU9250_Read_Data_Handle(void)
{
    // Get accelerometer data  得到加速度传感器数据
    MPU_Get_Accelerometer(&aacx, &aacy, &aacz);
    // Get the gyroscope data  得到陀螺仪数据
    MPU_Get_Gyroscope(&gyrox, &gyroy, &gyroz);
    // Get magnetometer data  得到磁力计数据
    MPU_Get_Magnetometer(&magx, &magy, &magz);

    // 为了打印不太快，每10个数据打印一次。
    // In order not to print too fast, print every 10 pieces of data
    static uint8_t show = 0;
    show++;
    if (show > 10)
    {
        show = 0;
        printf("accel:%d, %d, %d\n", aacx, aacy, aacz);
        printf("gyro:%d, %d, %d\n", gyrox, gyroy, gyroz);
        printf("mag:%d, %d, %d\n", magx, magy, magz);
    }
}
```

7. Add the content of initializing MPU9250 in the Bsp_Init() function, if the initialization fails, stop the program.

```c
// The peripheral device is initialized
void Bsp_Init(void)
{
    uint8_t res = 0;
    USART1_Init();
    res = MPU9250_Init();
    if (res != 0)
    {
        printf("MPU9250 INIT ERROR\n");
        while(1);
    }
    Beep_On_Time(50);
}
```

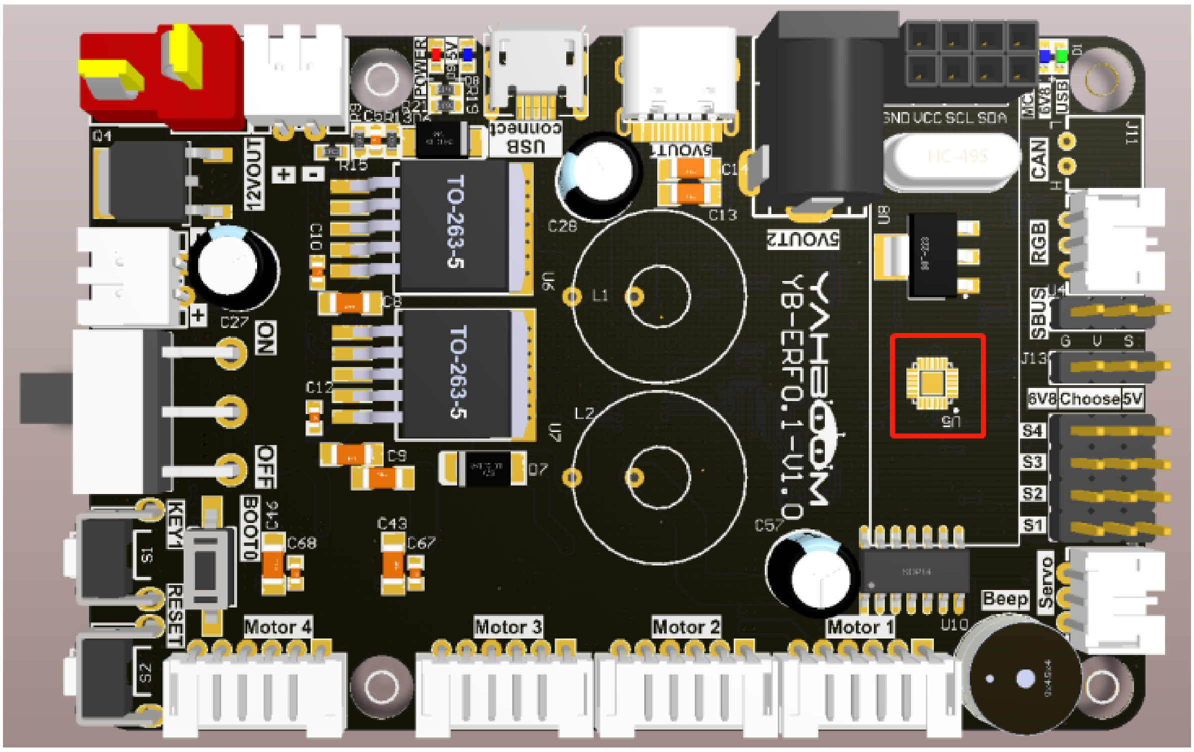
8. Add the function of reading MPU9250 data in the Bsp_Loop() function.

```c
void Bsp_Loop(void)
{
    // Detect button down events   检测按键按下事件
    if (Key1_State(KEY_MODE_ONE_TIME))
    {
        Beep_On_Time(50);
        static int press = 0;
        press++;
        printf("press:%d\n", press);
    }
    MPU9250_Read_Data_Handle();

    Bsp_Led_Show_State_Handle();
    // The buzzer automatically shuts down when times out
    Beep_Timeout_Close_Handle();
    HAL_Delay(10);
}
```

11.5. Hardware connection

The MPU9250 nine-axis attitude sensor has been soldered on the expansion board, so there is no need to manually connect the device.



11.6. Experimental effect

After the program is programmed, the LED light flashes every 200 milliseconds. Open the serial port assistant (the parameters are as shown in the figure below), you can see that the serial port assistant has been printing the data of the MPU9250's accelerometer accel, gyroscope gyro, and magnetometer mag.