四轴无人机姿态调整系统

一、 项目简介

随着信息技术的高速发展，以传感器为应用的物联网（IoT）得到了前所未有的发展， 物联网产品应用广泛，市场规模巨大。主流传感器中，又以位置、流量、气体、湿度等传感器备受市场青睐。

本项目是借助 ZMID5201STKIT 开发套件，利用 ZMID520xMROT36001 Rotary Application Module 模块，将无人机飞行过程中的飞行方向偏移角发送给 MCU，经过 MCU 解析后，发送到 NRF2401。通过 NRF2401 无线数据传输，最终将数据发送到手持设备上，可直观的观察到无人机姿态偏斜方向。

该项目的无人机具有一定的使用价值，可在科教、娱乐等领域应用，在工业、民用安防、智慧城市建设方面具有一定的应用价值，市场前景广阔，在未来商业应用方面潜力巨大。

二、 硬件说明

1、主控芯片：STM32F103C8T6 作为主控 MCU，STM32F103C8T6 采用 Corex-M3 内核，工作频率可达到 72MHz，20KB SRAM，共 3 个定时器和 3 个 USART。48 个 LQFP 引脚足够外围电路使用。如下图所示。

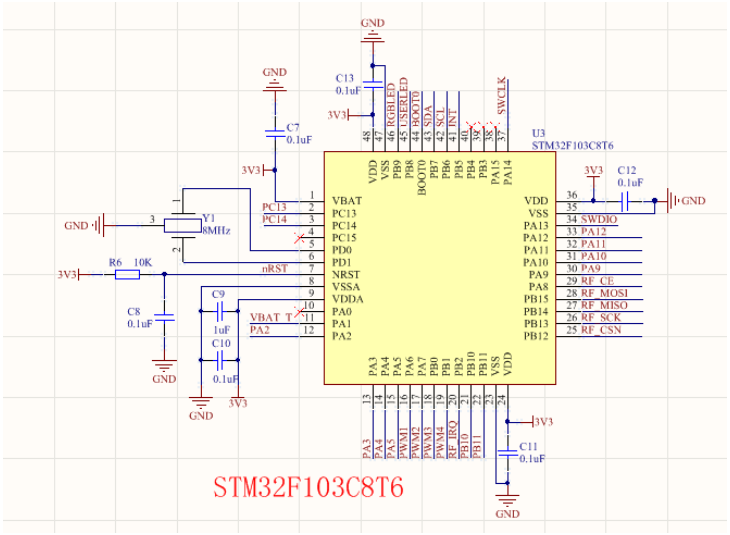

图 1 主控芯片

2、电源部分：供电方案采用先升压再降压的方案，由于第一次使用 1S 的锂离子电池，给四个空心杯进行空载供电的时候，四轴启动是没有问题的。但是如果四个空心杯都带上负载，如果瞬间提速到满速，那么电池输出电压拉低到 3V 以下，如果不采用升压方案，直接用电池给 LDO 供电，那 LDO 就会失效。所以通过升压再降压后给单片机系统供电是一个可行的方案，如下图所示。

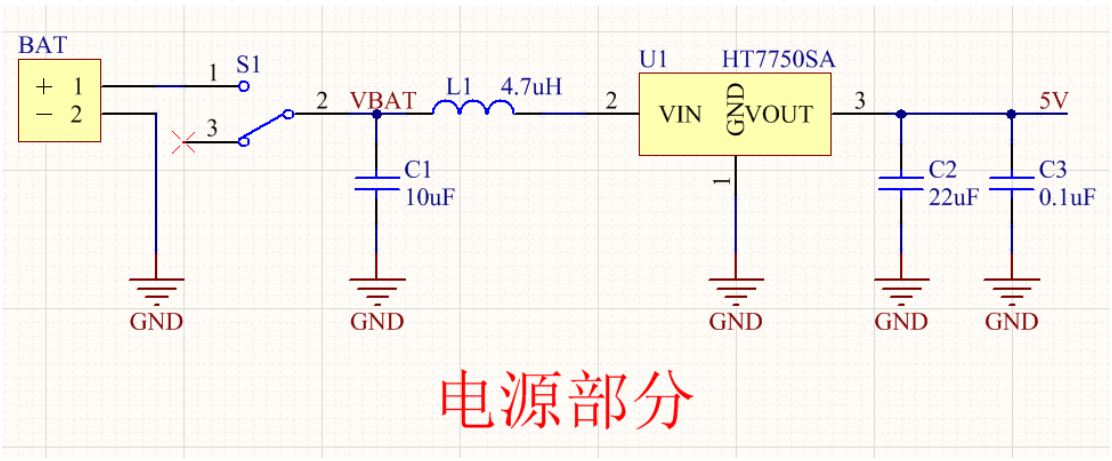

图 2 升压电路部分

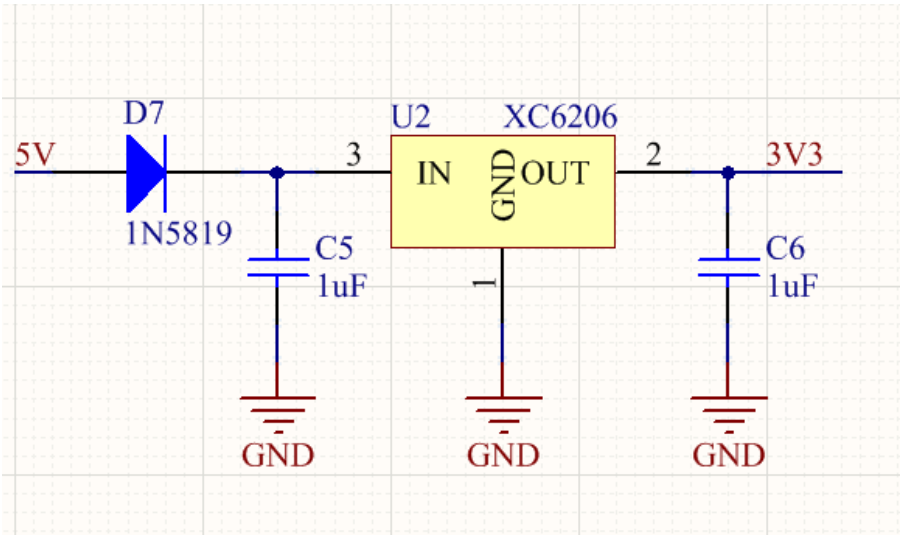

图 3 降压电路部分

3、传感器部分：MPU6050 六轴传感器芯片通过数据滤波处理把三轴陀螺仪数据和三轴加速度数据以及 ZMID520xMROT36001 Rotary Application Module 角度偏移数据引入 MCU 进行 IMU 姿态解算，如下图所示。

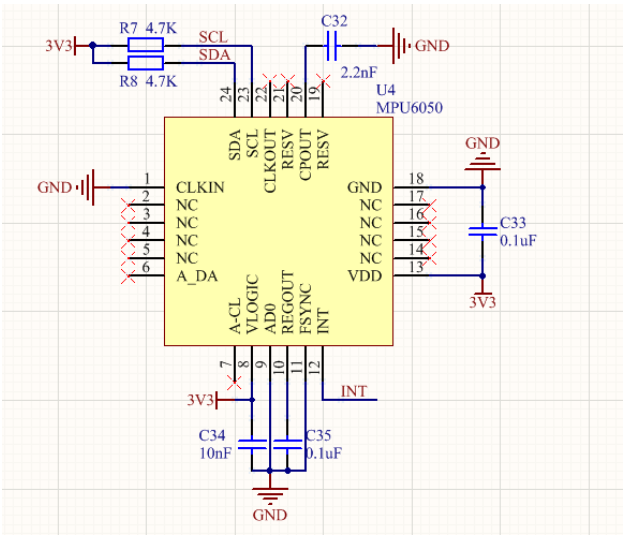

图 4 陀螺仪

4、四轴空心杯：无人机能否平稳操控，主要在四个电机上，空心杯选的好，对应姿态的稳定有较大的帮助。当然了驱动电机的 MOS 管上、陀螺仪上、PCB 对四轴最终的效果也起到关键的作用。电机的频率，会才生

干扰的频率点，会使无人机振动较大不好控制。另外就是同一批次的电机性能差异很大，导致 PID 调节的输出差异很大，最终会影响 MOS 管的寿命、电机寿命。空心杯电机使用 SI2302 这款 MOS 管进行驱动，这是非常常见的一款 MOS 管，便宜又好用。如下图所示。

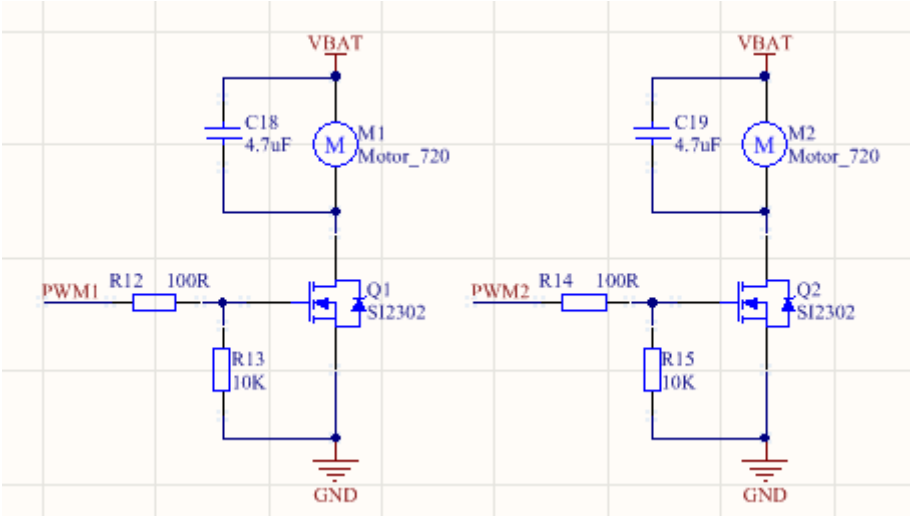

图 5 空心杯（前）

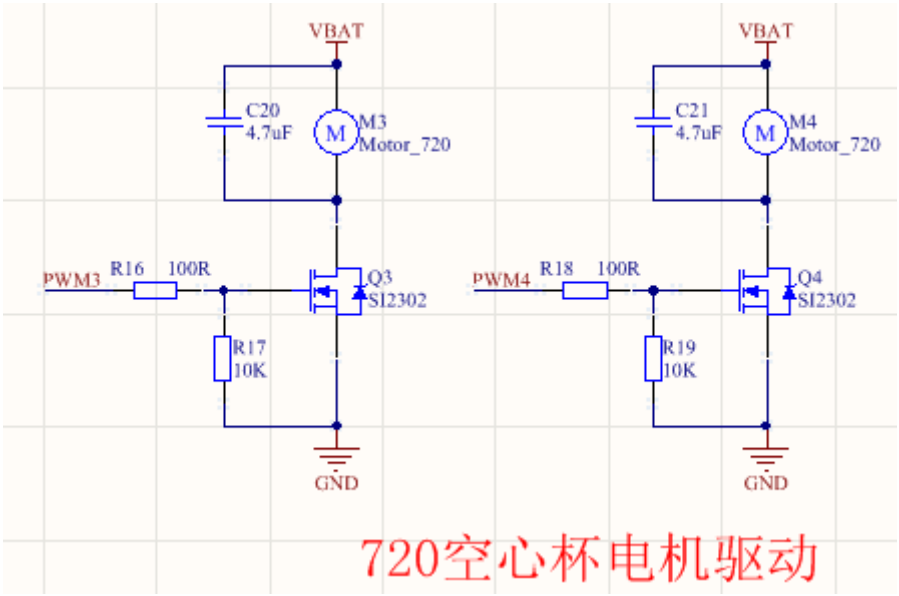

图 6 空心杯（后）

5、射频部分：采用 NRF2401，这是一块常用芯片，性能也是可以的，无线发射可以做到 7dB，加上发射和接收端都采用陶瓷天线，可以达到 50m 的通讯距离。如果加上 AP，那达到 100 米应该没有问题。通过两个低成本

的 0 欧姆电阻对电源进行了单点接地，防止电机回路的电流波动串进射频回路对射频造成干扰。

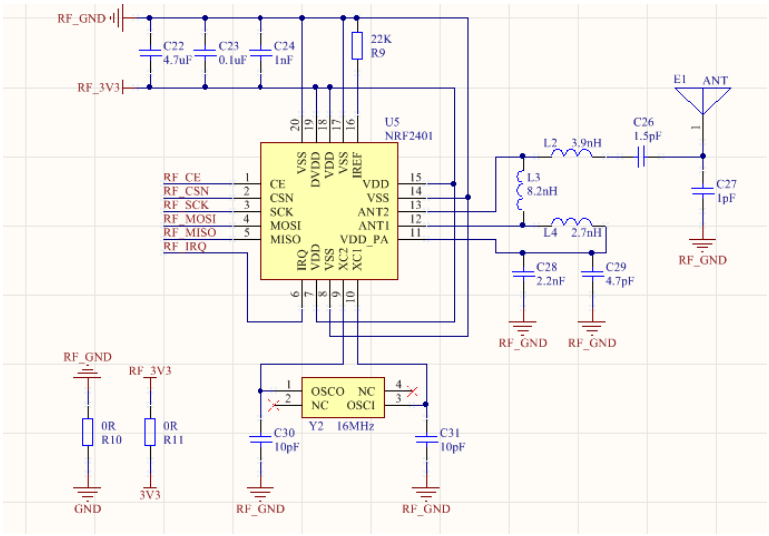

图 7 射频部分

6、灯光模块：采用 3 个 SOT23-3 封装的 MOS 管进行开关控制，与 MCU 隔离电源。

7、遥控器电路：采用传统的 DBUS 遥控器反向电路。

三、 软件说明

1、无线射频传输部分代码：

```
/**********************************************************************
配置 NRF2401 为 RX 模式，准备开始接收数据
**********************************************************************/
void RX_Mode(void)
{
    CE_LOW;                                                    //拉低 CE,进入待机模式，准备开始往 NRF2401 中的寄存器中写入数据

    SPI_Write_Byte(WRITE_REG_CMD + CONFIG, 0x0f);      //配置为接收模式
    SPI_Write_Byte(WRITE_REG_CMD + STATUS, 0x7e);      //写 0111 xxxx 给 STATUS，清除所有中断标志，防止一进入接收模式就触发中断

    CE_HIGH;                                                   //拉高 CE,
```

```
准备接受从外部发送过来的数据
}

/**************************************************************************
从 NRF2401 的 RX 的 FIFO 中读取一组数据包
输入参数 rx_buf:FIFO 中读取到的数据的保存区域首地址
**************************************************************************/
void NRF2401_ReceivePacket(u8* rx_buf)
{
    CE_LOW;
    SPI_Read_Buf(RD_RX_PLOAD,rx_buf,RX_PLOAD_WIDTH);      //从 RX 端的 FIFO 中读取数
据，并存入指定的区域，注意：读取完 FIFO 中的数据后，NRF2401 会自动清除其中的数据
    SPI_Write_Byte(FLUSH_RX,0xff);                        //清除接收 FIFO
    CE_HIGH;                                                          //重新
拉高 CE，让其重新处于接收模式，准备接收下一个数据
}

/**************************************************************************
配置 NRF2401 为 TX 模式，并发送一个数据包
输入参数 tfbuf:即将要发送出去的数据区首地址
**************************************************************************/
void NRF2401_SendPacket(u8* tfbuf)
{
    CE_LOW;                                                                //
拉低 CE，进入待机模式，准备开始往 NRF2401 中的寄存器中写入数据

//    SPI_Write_Buf(WRITE_REG_CMD + RX_ADDR_P0, TX_ADDRESS, TX_ADR_WIDTH);    //
装载接收端地址，由于这里只有一个通道通讯，不用改变接收端的 NRF2401 的接收通道地
址,所以，这句可以注释掉
    SPI_Write_Buf(WR_TX_PLOAD, tfbuf, TX_PLOAD_WIDTH);              //将数据写入 TX
端的 FIFO 中,写入的个数与 TX_PLOAD_WIDTH 设置值相同

    SPI_Write_Byte(WRITE_REG_CMD + CONFIG, 0x0e);                   //将 NRF2401 配
置成发射模式
    SPI_Write_Byte(WRITE_REG_CMD + STATUS, 0x7e);                  // 写 0111 xxxx 给
STATUS，清除所有中断标志，防止一进入发射模式就触发中断

    CE_HIGH;                                                           //拉高 CE, 准备发
射 TX 端 FIFO 中的数据

    delay_ms(1);                                                       //CE 拉高后, 需要延迟
至少 130us
}
```

```
void WaitFlY_Connection(void)
{
    static u8 cnt = 0,preaddr;
    ConnectingDisplay();//断线连接状态显示
    while(1)
    {
        if(FLY_Connect_OK)
        {
            cnt = 0;
            if(preaddr != TX_ADDRESS[TX_ADR_WIDTH-1])
            {
                PID_WriteFlash();

//                printf("Address save :%d preaddr:%d\r\n",TX_ADDRESS[4],preaddr);
            }
//            printf("Fly connect OK!!!\r\n");
            return;
        }else if(cnt++ < 10)
        {
            PID_ReadFlash();                        //读取上一次保存的飞机的 NRF2401
地址
            preaddr = TX_ADDRESS[TX_ADR_WIDTH-1];
            NRF2401_Config();
            delay_ms(50);
//            printf("Flash read NRF2401addr:%d\r\n",TX_ADDRESS[4]);
        }
        else
        {
            TX_ADDRESS[TX_ADR_WIDTH-1]++ ;
            RX_ADDRESS[TX_ADR_WIDTH-1]++ ;
            if(TX_ADDRESS[TX_ADR_WIDTH-1]>AddrMax                    &&
RX_ADDRESS[TX_ADR_WIDTH-1]>AddrMax)
            {
                TX_ADDRESS[TX_ADR_WIDTH-1] = 0x00;
                RX_ADDRESS[TX_ADR_WIDTH-1] = 0x00;
            }
            SPI_Write_Buf(WRITE_REG_CMD + TX_ADDR, TX_ADDRESS, TX_ADR_WIDTH);

            SPI_Write_Buf(WRITE_REG_CMD      +      RX_ADDR_P0,      RX_ADDRESS,
RX_ADR_WIDTH);
            delay_ms(100);
        }
```

```
            if(ADC_CALIBRATOR_OK)
            {
                OLED_ShowString(byte(2),line4,"Calibration",8);
            }
            else
            {
                OLED_ShowString(byte(2),line4,"Connecting...",8);
            }
        }
}

//断线重连函数
void ReconnectionFly(void)
{
    if(Reconnection_flag)
    {
        ConnectingDisplay();
        if(FLY_Connect_OK)          //遥控已重新连接
        {
            Display_init();
            Reconnection_flag = 0;   //断线重连标志复位
            return;
        }
        NRF2401_Config();
    }
}
```

```
162 配置NRF2401为RX模式，准备开始接收数据
163 *********************************************************************/
164 void RX_Mode(void)
165 {
166   CE_LOW;                                  //拉低CE，进入待机模式，准备开始往SI24R1中的寄存器中写入数据
167
168   SPI_Write_Byte(WRITE_REG_CMD + CONFIG, 0x0f);   //配置为接收模式
169   SPI_Write_Byte(WRITE_REG_CMD + STATUS, 0x7e);   //写0111 xxxx 给STATUS，清除所有中断标志，防止一进入接收模式就
170
171   CE_HIGH;                                 //拉高CE，准备接受从外部发送过来的数据
172 }
173
174 /*********************************************************************
175 从NRF2401的RX的FIFO中读取一组数据包
176 输入参数rx_buf:FIFO中读取到的数据的保存区域首地址
177 *********************************************************************/
178 void SI24R1_ReceivePacket(u8* rx_buf)
179 {
180   CE_LOW;
181   SPI_Read_Buf(RD_RX_PLOAD,rx_buf,RX_PLOAD_WIDTH);  //从RX端的FIFO中读取数据，并存入指定的区域，注意：读取完FIFO中
182   SPI_Write_Byte(FLUSH_RX,0xff);                    //清除接收FIFO
183   CE_HIGH;                                 //重新拉高CE，让其重新处于接收模式，准备接收下一个数据
184 }
```

图 8 无线射频传输部分代码

2、手持终端显示部分代码：

```
void ButtonCMDDisplay(void)
{
```

```
    if(GET_FLAG(FLY_ENABLE))
    {
        OLED_ShowString(byte(11),line4,"Nolink",6);
    }else
    {
        OLED_ShowString(byte(11),line4,"Link    ",6);
    }
    SenserOffsetDisplay();
    WiFiOnOffDisplay();
    ModeSelectDisplay();
}

void ConnectingDisplay(void)
{
    OLED_Clear();
    OLED_ShowCHinese(1,line1,WiFi,0); //信号强度
    OLED_ShowString(byte(3)+4,line1," IDT_Fly",8); //大赛名
    OLED_ShowCHinese7x7(121,line1,BATT,0); //电池图标
    OLED_ShowString(111,line1,"%",6); //
    OLED_ShowNum(108,line2,0,2,6); //电池电压比例

}
```

```
252     SignalIntensityDisplay(); //信号强度更新显示
253     ButtonCMDDisplay(); //
254     percent = (FLY.BattV-300.0f)/120.f*100; //电池百分比
255
256     OLED_ShowNum(byte(3),line4,FLY.Thr,4,6);
257
258     OLED_ShowNum(byte(4),line6,FLY.Rol,2,6);  //横滚角
259     OLED_ShowNum(byte(12),line6,FLY.Pit,2,6); //俯仰角
260     OLED_ShowNum(byte(12),line8,FLY.Yaw,2,6); //航向角
261     OLED_ShowNum(byte(4),line8,FLY.Alt,2,6);  //高度
262     OLED_ShowNum(108,line2,percent,2,6); //电池电压比例
263     }
264 }
265
266 //传感器校准显示
267 void SenserOffsetDisplay(void)
268 {
269   //陀螺仪校准
270   if(GET_FLAG(GYRO_OFFSET))
271   {
272     OLED_ClearBlue();
273     while(GET_FLAG(GYRO_OFFSET))
274     {
275       ReceiveDataAnalysis();//继续接收遥控数据
276       OLED_ShowCHineseString(byte(5),line4,TuoLuoYi,3); //陀螺仪
277       OLED_ShowCHineseString(byte(4),line6,JZCG,4); //校准成功
278     }
279     Display_init();
```

图 9 无线射频传输部分代码

四、 效果演示

四轴无人机下方为自制的指南针，用来定位起飞前的方向。南针放在四轴无人机下方，为了防止电机风力而引起的摆动。

无人机起飞、平衡性并不理想，主要还是在数据传输部分，不能及时将传感器发送的数据，进行其实的处理去调整无人机的姿态。但是是 MPU6050 还是 ZMID520xMROT 传输的数据，都能显示在 RGB LED 屏上。

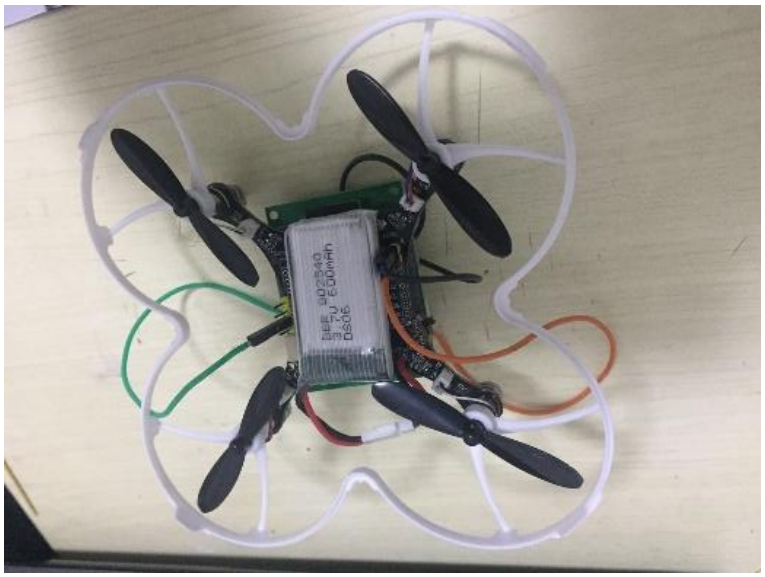

图 9 正面图

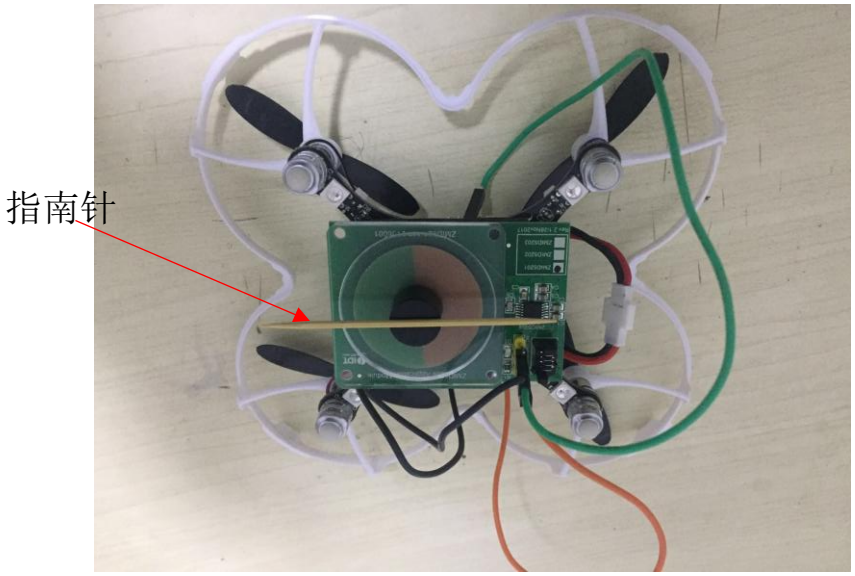

图 10 背面图

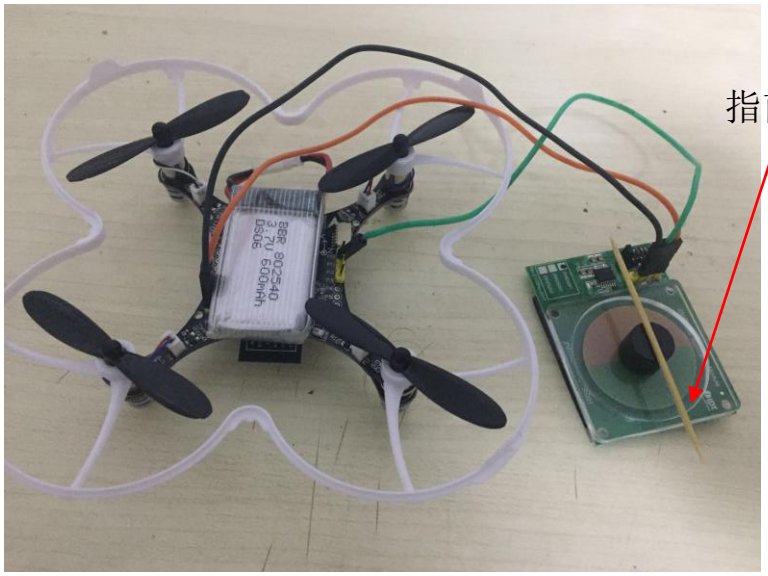

图 11 连接图