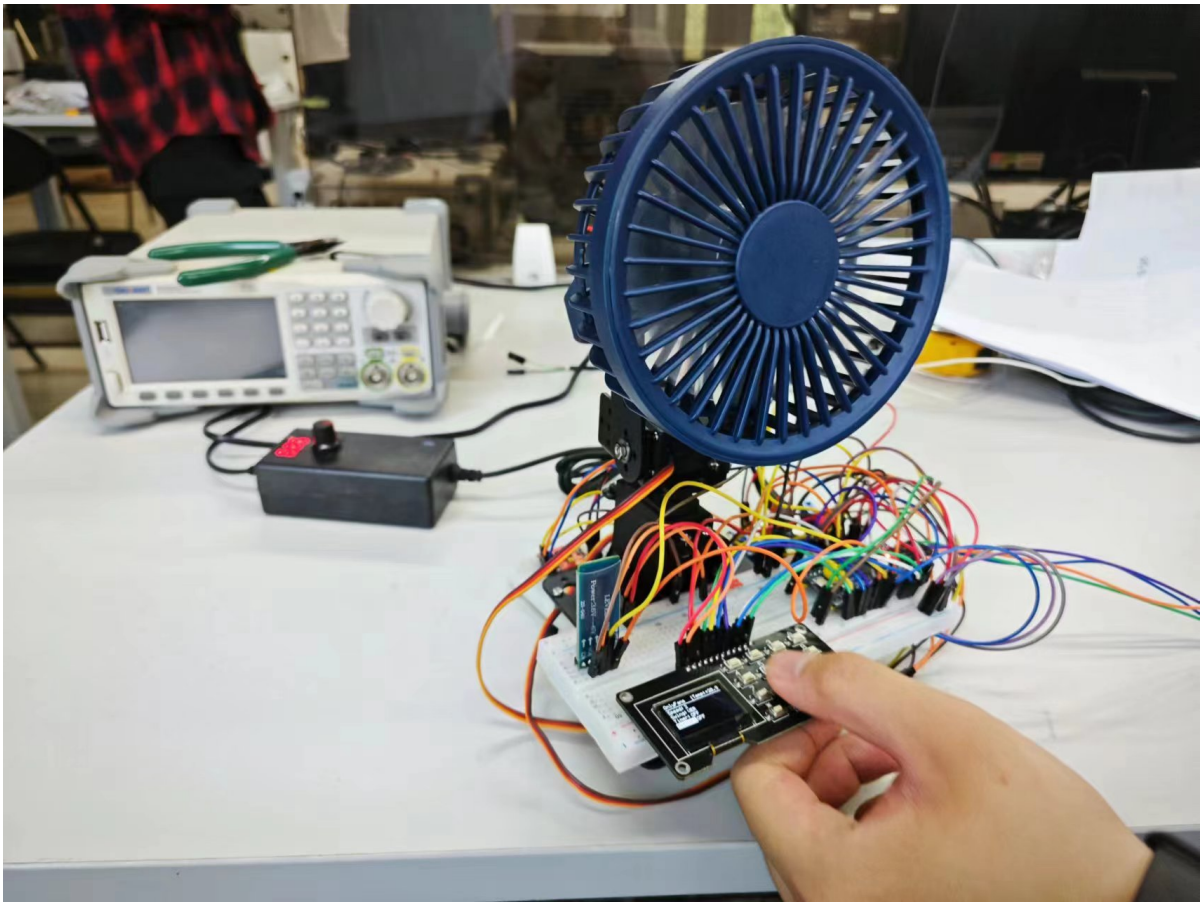


智创大赛智能小风扇项目报告

作者：　谢雨鑫　袁天宇

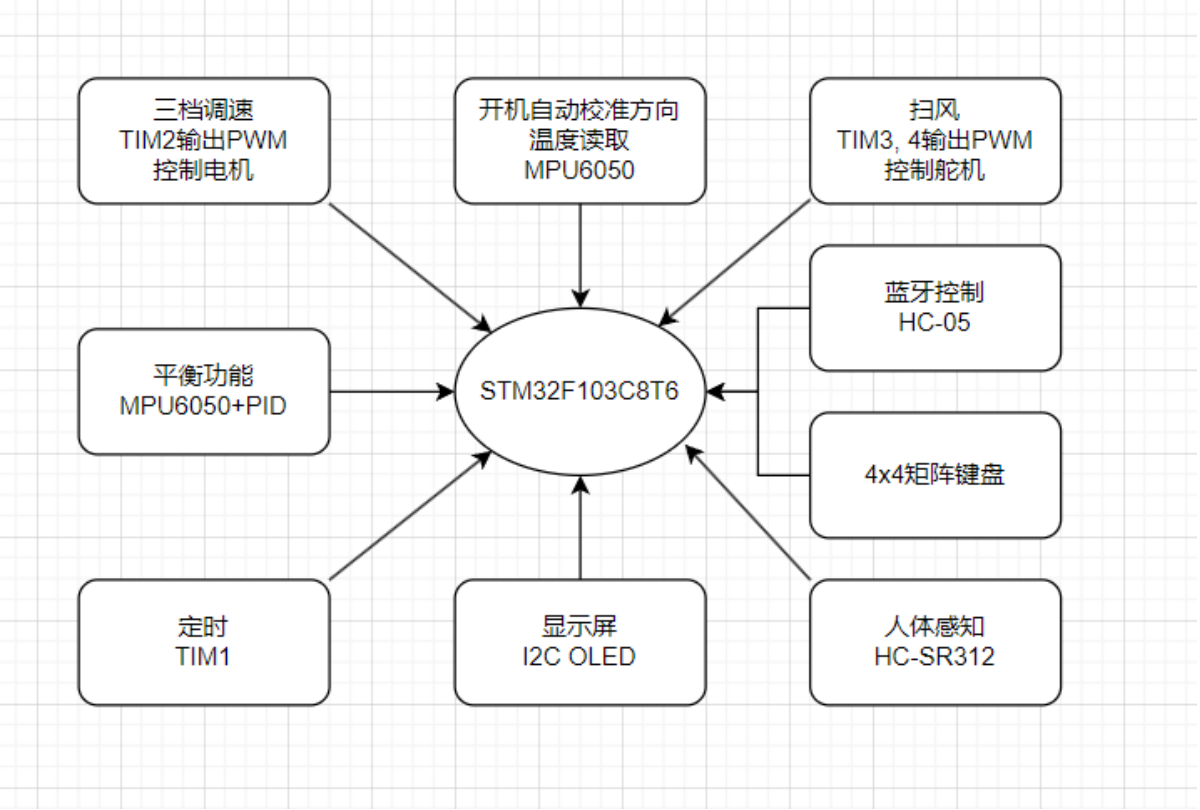
目录

1.硬件选型和作品硬件结构框图

1.硬件清单

- **mcu**：**stm32f103c8t6** 入门款stm32单片机
- **电机驱动**：**TB6612fng** 对扇叶进行驱动
- **陀螺仪**：**MPU6050** 用于角度感知 测量风扇倾角
- **0.96寸OLED显示屏** 状态参数显示界面，人机交互
- **MG996r舵机x2** 风扇左右上下摆头
- **4X4矩阵键盘** 输入设备
- **蓝牙模块 HC-05** 和上位机无线通信
- **人体红外感知模块HC-SR312** 人体感知
- 12V稳压电源/18650电池 Power supply

2.项目总体组织方式



在keilv5工程中，我们采用模块式编程的方法，把不同的外设功能函数放在不同的.c文件中，main.c中通过应用写有对应函数名对应的头文件调用，所有的操作逻辑都写在Menu_Proc( )函数中，通过**轮询、中断和中断嵌套**来完成所有功能。

由于作者水平有限，本项目没有采用RTOS（嵌入式实时操作系统），在项目调试中，随着外设功能的增多，代码的执行逻辑也越发复杂，为避免外设之间发生冲突，我们在调试过程中花费相当之多的时间，因此我们深感进一步**学习嵌入式操作系统**的必要，这也是我们之后学习的目标。

值得一提的是，由于作者尚未掌握绘制画PCB板子的这项技能，**采用了面包板和杜邦线插接方法连接硬件电路，杜邦线数目之多令人眼花缭乱，即使是拆弹专家看了也会扼腕叹息。**这极大的影响了硬件的鲁棒性，调试过程中产生了相当之多的硬件Bug，时常产生电线虚接的问题，令人叫苦不迭，因此学习PCB制版将会是未来重点的学习内容。

3.硬件连接和引脚分配

外设	MCU引脚分配
电机驱动：TB6612fng(PWMA1,AIN1,AIN2)	PA1,PA4,PA5
陀螺仪：MPU6050(SCL,SDA)	PB10,PB11
OLED显示屏(SCL,SDA)	PB8,PB9
MG996r舵机x2(Signal)	PA7,PA9
4X4矩阵键盘	PB12,PB13,PB14,PB15,PA0,PA5,PA6,PA7
蓝牙模块 HC-05(RXD,TXD)	PA9,PA10
人体红外感知模块HC-SR312	PA11

4.硬件功能函数

- **矩阵键盘**

  函数名：

```
void Key_Init(void);//按键初始化
uint8_t Keyboard_Scan(void);//按键扫描
uint8_t  KeyboardtoNum(uint8_t key);//按键键值对应
```

  实现方式：**状态机法**详见 *附录1*

- **OLED显示屏**

  调用了B站up主江协科技的OLED显示函数 及其底层iic通信协议

  主要调用:

```
void OLED_ShowString(uint8_t X, uint8_t Y, char *String, uint8_t FontSize);//显示字符串
void OLED_ShowNum(uint8_t X, uint8_t Y, uint32_t Number, uint8_t Length, uint8_t FontSize);//显示数字
/*初始化函数*/
void OLED_Init(void);
/*更新函数*/
void OLED_Update(void);
void OLED_UpdateArea(uint8_t X, uint8_t Y, uint8_t Width, uint8_t Height);
/*显存控制函数*/
void OLED_Clear(void);
```

- **舵机扫风**

```
void Fans_Init(void);//初始化
void Fans_Sweep(uint8_t mode);//摆头功能实现
```

  具体实现：

```
void Fans_Sweep(u8 mode)
```

```
{
    mode -= 1;
    static int16_t angle[2] = {40, 30};
    static int8_t flag[2] = {1, 1};

    if (mode == 0)
    {
        Servo_Set1Angle(30);
        if (angle[mode] > 100 | angle[mode] < -20)
            flag[mode] = -flag[mode];
        angle[mode] += flag[mode] * 1;
        Servo_Set2Angle(angle[mode]);
    }
    else if (mode == 1)
    {
        Servo_Set2Angle(40);
        if (angle[mode] > 70 | angle[mode] <= 10)
            flag[mode] = -flag[mode];
        angle[mode] += flag[mode] * 1;
        Servo_Set1Angle(angle[mode]);
    }

}
```

- **电机调速**

```
void Motor_Init(void);
void Motor_SetSpeed(int8_t Speed);
```

底层为TIM通用定时器的输出比较功能调节占空比， PWM输出

```
void PWM_Init(void);
void PWM_SetCompare2(uint16_t Compare);
void PWM_SetCompare1(uint16_t Compare);
void PWM_SetCompare3(uint16_t Compare);
```

- **蓝牙收发**

```
void BLT_Init(void);//蓝牙初始化
void BLT_Upd(u16 *val);//数据采集更新
```

通过valuepack 和USART 进行数据包的发送接受

底层协议调用部分github及csdn开源代码

- **人体感知**

```
void IR_Init(void);
void IR_Capture(void);
```

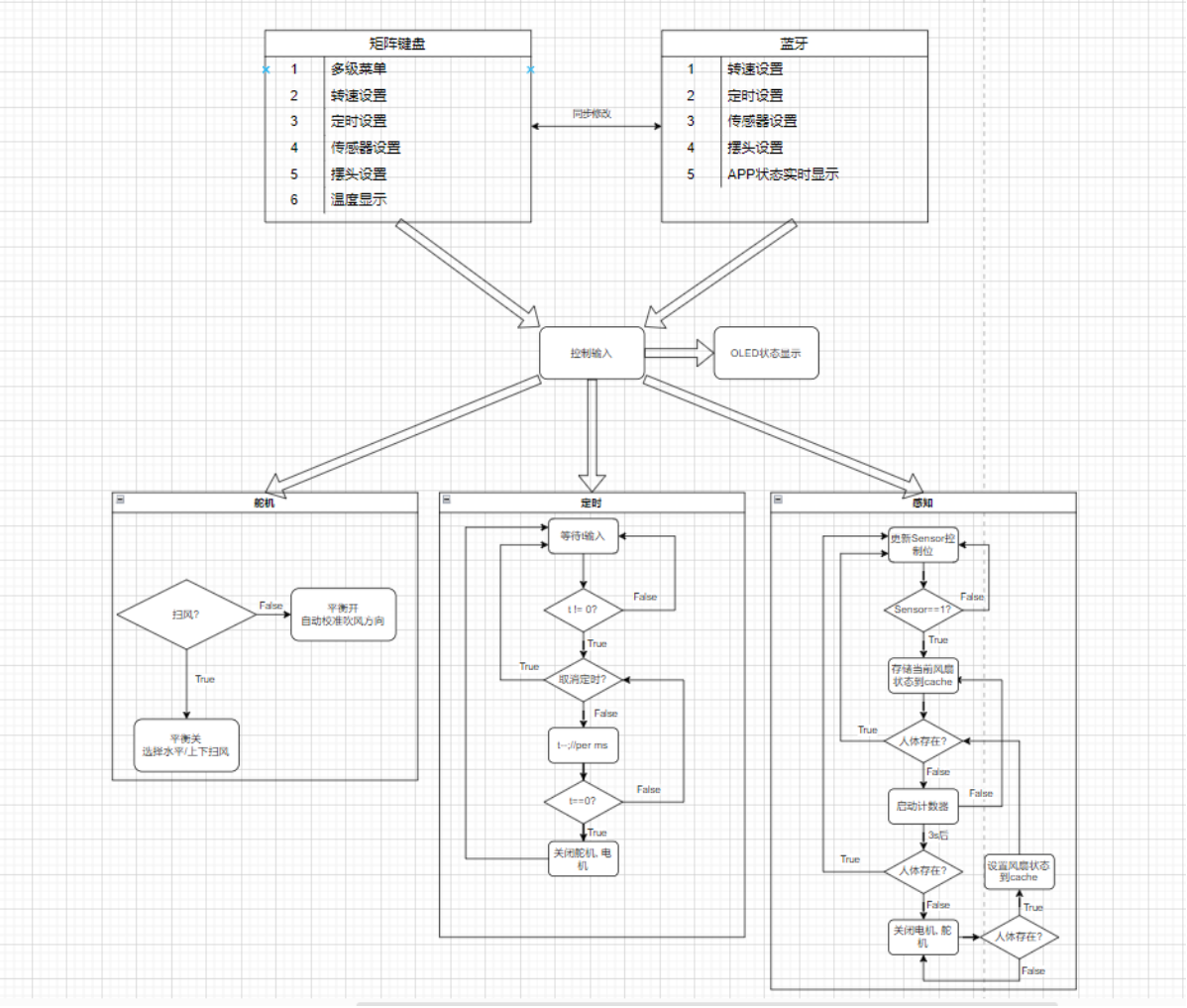
- **MPU6050角度感知（PID）**

```c
void Balance(void);//风扇平衡函数
void MPU6050_WriteReg(uint8_t RegAddress, uint8_t Data);//写寄存器
uint8_t MPU6050_ReadReg(uint8_t RegAddress);//读寄存器

void MPU6050_Init(void);//初始化
void MPU6050_GetData(int16_t *AccX, int16_t *AccY, int16_t *AccZ,
                     int16_t *GyroX, int16_t *GyroY, int16_t
*GyroZ);//获取角度数据
```

在风扇倾角感知和水平调节中采用PID算法控制

2.代码执行流程



代码流程图

代码整体结构如上。`OnlyFan`（以下简称本风扇）主要由控制模块、功能模块组成，控制模块包括`矩阵键盘模块`，`蓝牙模块`，功能模块包括`舵机模块`，`定时模块`及`人体感知模块`。所有模块封装后，在`while`中执行。

下面略去底层的`I2C`通信、定时中断等内容，介绍每一个模块的顶层逻辑。所有代码请见附件。

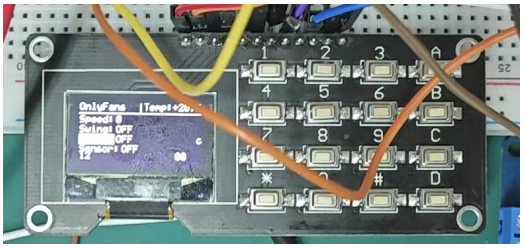
控制部分

本风扇可通过自带矩阵键盘控制与手机蓝牙 APP 控制。所有控制修改后，都会同步显示在 OLED 显示屏及手机 APP 上。

矩阵键盘控制

采用逐行扫描 + 状态机法实现，进行了非阻塞式消抖。利用江协科技提供的 OLED 显示模块绘制了一个控制菜单。菜单具有功能选择、数值修改两种模式。

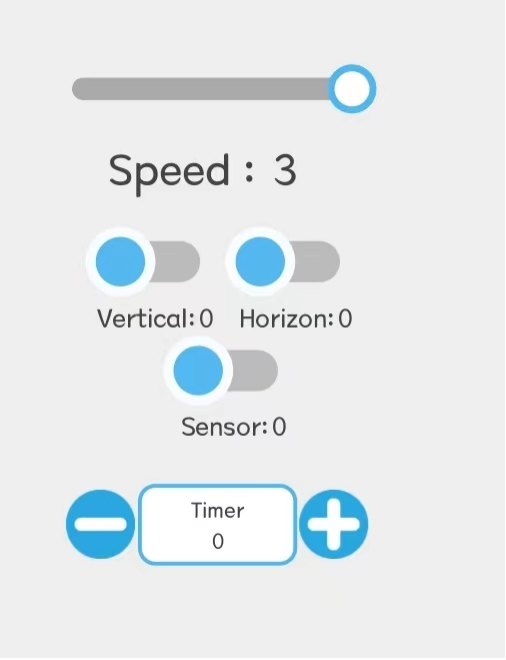
- 功能选择
  用户选择在转速、定时等功能之间滚动选择。菜单支持循环滚动。按下确认键后，对选中的功能进行设置，进入数值修改模式。
- 数值修改
  用户可以直接输入需要修改的数值。再次按下确认后，若将数值修改为 0，则将当前功能显示为 `OFF` 字样。如果按下取消，则恢复修改前的数值。定时功能可以单独设置分钟与秒钟。



矩阵键盘模块

蓝牙控制

利用 `HC-05` 与 手机APP 进行蓝牙连接后，可以对风扇直接进行设置。通过矩阵键盘进行的修改也会同步显示在 APP 上。



蓝牙控制模块

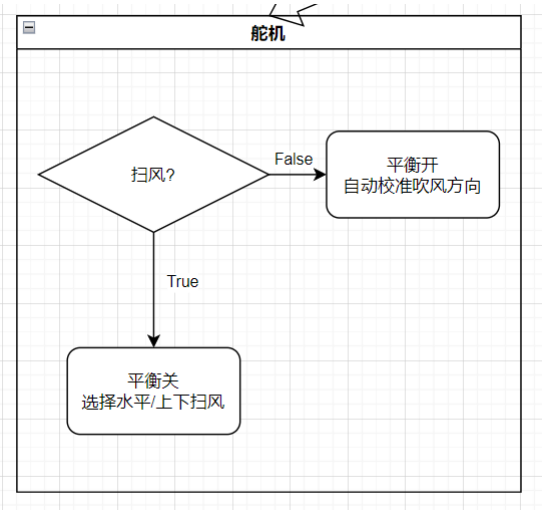
功能部分

本风扇主要可控制模块包括 `舵机模块`，`定时模块` 及 `人体感知模块`，全面实现了题目所需的要求。下面详细介绍每一个模块的实现。

初始化

在上电之后，本风扇默认关闭所有功能，同时通过 `MPU6050` 将风扇朝向校准至与地面平行。

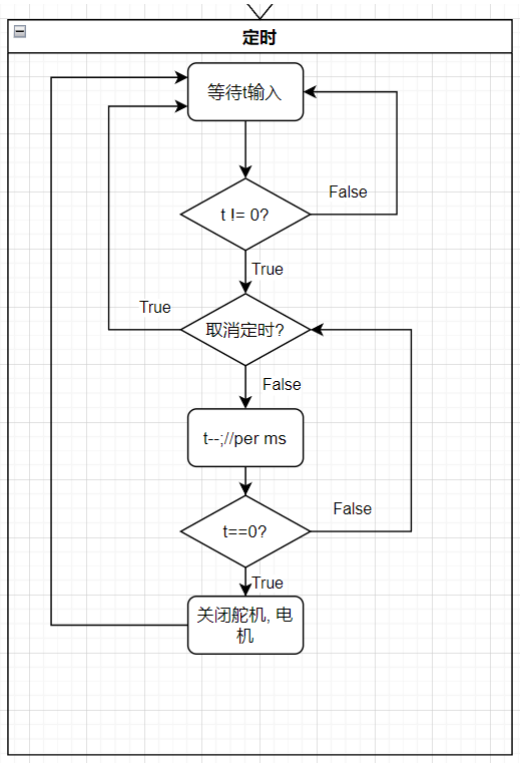
舵机模块



舵机模块控制流程图

当扫风功能开启时，平衡功能自动关闭，根据需求分别控制风扇上下、左右摆头。
扫风功能关闭后，平衡功能自动开启，此时风扇吹风方向校准至与地面平行。

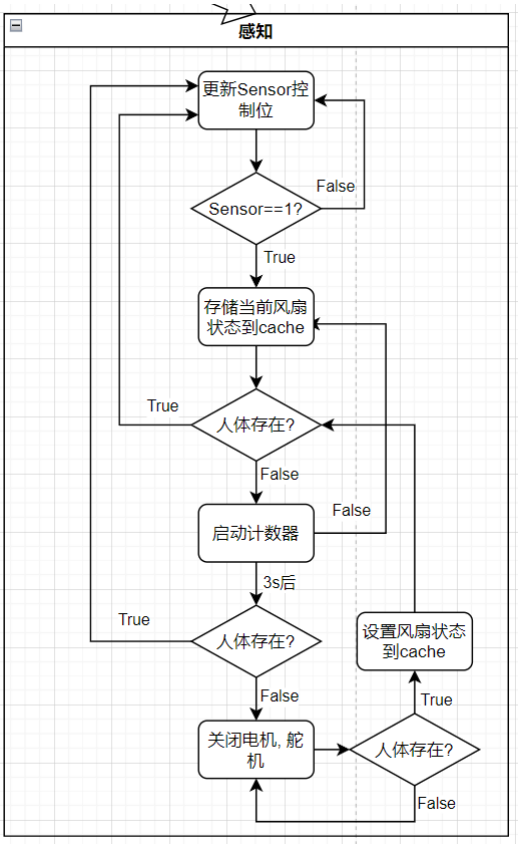
定时模块



定时模块控制流程图

在检测到定时时间 `t` 不为0后，通过 `TIM1` 实现的定时中断进行计数。 `t` 每秒自减1。在t为0后，将舵机、电机关闭。在 `t` 不为0时，如果用户取消定时（将 `t` 设置为0），则保持风扇原状态不变，同时继续检测 `t` 输入。

感知模块



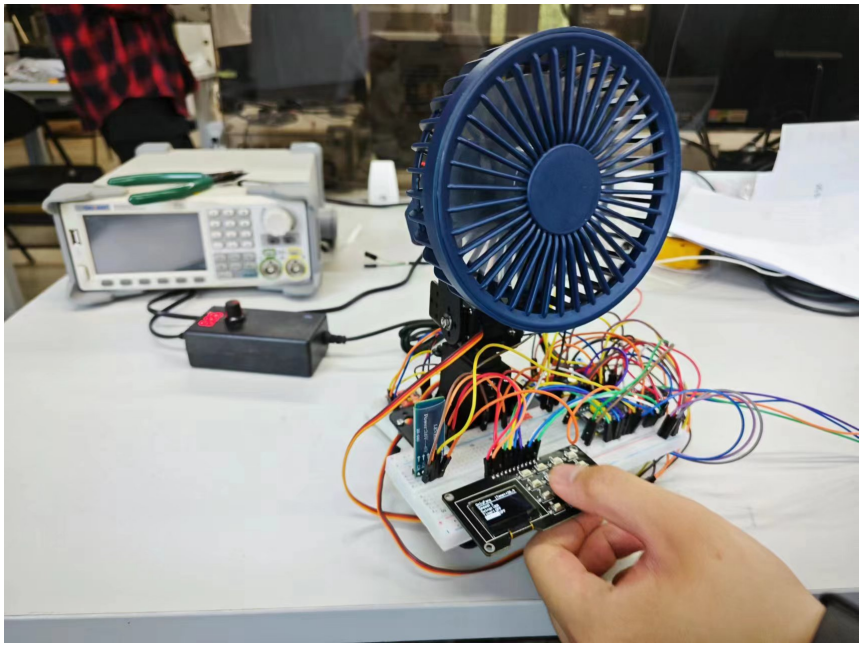
感知模块控制流程图

在感知功能开启后，存储风扇当前状态到`cache`，同时通过`HC-SR602`进行存在感知。如果人体存在，保持风扇当前状态不变。如果人体不存在，等待3s。3s后，人体仍不存在，则关闭舵机、电机；否则无动作。在电机关闭后，如果重新检测到人体存在，风扇状态回到`cache`。

3.功能描述

基础部分：

1.全装置由移动电源供电，至少具有扇叶、支架和必要的控制面板。



2.可设置三档风扇转速，每档转速无定量标准，但应区别明显。

3.每次开启风扇，风扇可校准吹风方向为与地面平行，且相对自身水平朝向固定。

4.具有左右和上下扫风功能，可在左右共 120°和上下共 60°范围内扫风，扫风可 随时开启和关闭。扫风在任意时刻关闭后，风扇回到校准位。

5.具有定时功能，可在自定义时间后停转；每次定时后，到时前可取消定时。

6.配有显示工作状态的显示屏，可显示当前转速档位、扫风状态、是否处于定时状态。

7.具有感知功能，当使用者离开风扇较远距离时，风扇自动停转，使用者回到较 近距离后风扇恢复工作，与停转前工作状态保持一致。

提高部分：

1.具有平衡功能，开启平衡功能后，在俯角或仰角不大于 30°的范围内整体倾斜 风扇，风扇吹风方向可恢复至与地面平行。

2.具有无线控制功能，显示屏上需显示的所有内容可显示在手机上，且可通过手 机设置转速档位、扫风、定时。

补充功能：温度显示

4.总结和反思

这是我们首个基于STM32 CortexM3内核MCU的落地项目，在这个过程中做中学，学习完STM32CortexM3内核单片机的基本内容，从基础的GPIO，到了解TIM 和PWM ，再到学习USART 、IIC 、SPI 、蓝牙等常见的通信协议，尝试应用PID控制算法。从零开始搭建电路，照猫画虎建立工程，从一到多学习外设，反复摸索和调试，硬件Debug的能力和查阅资料解决问题的能力显著提升，为之后的嵌入式开发打下良好基础。

提升空间：

该项目是采用stm32f103完成的，和上位机采用蓝牙通信，如果把智能风扇真正做成LOT物联网项目，那还应该具备WIFI 语音控制等功能，作者尝试接入米家，小爱同学语音控制，这时采用集成了WIFI和蓝牙模块的 ESP32开发更加合适；此外本项目硬件接线复杂，应当绘制PCB板提高电路集成度；受MCU性能限制，OLED显示效果一般，不妨试试调用写好的UI ，提高动画流畅度和交互度。从工业设计角度:产品不够美观大方，应该采用solidworks或者fushion设计打样一个好的亚克力外壳。

附录1 状态机法实现矩阵键盘

状态机法实现矩阵键盘按键检测

```
void Key_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
    // hang
    GPIO_InitStruct.GPIO_Pin = GPIO_Pin_12 | GPIO_Pin_13 | GPIO_Pin_14
|GPIO_Pin_15;
    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStruct.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_Init(GPIOB, &GPIO_InitStruct);
    // lie
    GPIO_InitStruct.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_5 | GPIO_Pin_6 |
GPIO_Pin_7;
    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_InitStruct.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_Init(GPIOB, &GPIO_InitStruct);
```

```
}
unsigned char Key_Scan()//定时器或者延时 几号秒检测一次
{
    GPIO_WriteBit(GPIOB, GPIO_Pin_12,Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_13,Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_14,Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_15,Bit_SET);
    unsigned char Key_temp = 0;
    static unsigned char Key_state;
    unsigned char Key_Value = 0;
    GPIO_WriteBit(GPIOB, GPIO_Pin_12,Bit_RESET);
    if(GPIO_ReadInputDataBit( GPIOB,GPIO_Pin_0)==0) Key_temp = 1;
    if(GPIO_ReadInputDataBit( GPIOB,GPIO_Pin_5)==0) Key_temp = 2;
    if(GPIO_ReadInputDataBit( GPIOB,GPIO_Pin_6)==0) Key_temp = 3;
    if(GPIO_ReadInputDataBit( GPIOB,GPIO_Pin_7)==0) Key_temp = 4;
    if(Key_temp ==0){
    GPIO_WriteBit(GPIOB, GPIO_Pin_12,Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_13,Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_14,Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_15,Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_13,Bit_RESET);
    if(GPIO_ReadInputDataBit( GPIOB,GPIO_Pin_0)==0) Key_temp = 5;
    if(GPIO_ReadInputDataBit( GPIOB,GPIO_Pin_5)==0) Key_temp = 6;
    if(GPIO_ReadInputDataBit( GPIOB,GPIO_Pin_6)==0) Key_temp = 7;
    if(GPIO_ReadInputDataBit( GPIOB,GPIO_Pin_7)==0) Key_temp = 8;
    }
    if(Key_temp ==0){
    GPIO_WriteBit(GPIOB, GPIO_Pin_12,Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_13,Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_14,Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_15,Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_14,Bit_RESET);
    if(GPIO_ReadInputDataBit( GPIOB,GPIO_Pin_0)==0) Key_temp = 9;
    if(GPIO_ReadInputDataBit( GPIOB,GPIO_Pin_5)==0) Key_temp = 10;
    if(GPIO_ReadInputDataBit( GPIOB,GPIO_Pin_6)==0) Key_temp = 11;
    if(GPIO_ReadInputDataBit( GPIOB,GPIO_Pin_7)==0) Key_temp = 12;
    }
    if(Key_temp ==0){
    GPIO_WriteBit(GPIOB, GPIO_Pin_12,Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_13,Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_14,Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_15,Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_15,Bit_RESET);
    if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_0)==0) Key_temp = 13;
    if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_5)==0) Key_temp = 14;
    if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_6)==0) Key_temp = 15;
    if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_7)==0) Key_temp = 16;
    }

    switch (Key_state){
        case 0:
            if( Key_temp !=0){
            Key_state = 1;
            }
            break;
        case 1:
```

```
            if(Key_temp == 0)
                Key_state = 0;
            else
            {
                Key_state = 2;
                switch (Key_temp )
                {
                    case 1:
                        Key_Value = 1;
                        break;
                    case 2:
                        Key_Value = 2;
                        break;
                    case 3:
                        Key_Value = 3;
                        break;
                    case 4:
                        Key_Value = 4;
                        break;
                    case 5:
                        Key_Value = 5;
                        break;
                    case 6:
                        Key_Value = 6;
                        break;
                    case 7:
                        Key_Value = 7;
                        break;
                    case 8:
                        Key_Value = 8;
                        break;
                    case 9:
                        Key_Value = 9;
                        break;
                    case 10:
                        Key_Value = 10;
                        break;
                    case 11:
                        Key_Value = 11;
                        break;
                    case 12:
                        Key_Value = 12;
                        break;
                    case 13:
                        Key_Value = 13;
                        break;
                    case 14:
                        Key_Value = 14;
                        break;
                    case 15:
                        Key_Value = 15;
                        break;
                    case 16:
                        Key_Value = 16;
                        break;
                }
```

```
            }
            break;
        case 2:
                if(Key_temp ==0)
                {
                    Key_state = 0;
                }
            break;
    }
    return Key_Value;
}
```