宠物喂食器

1.介绍:

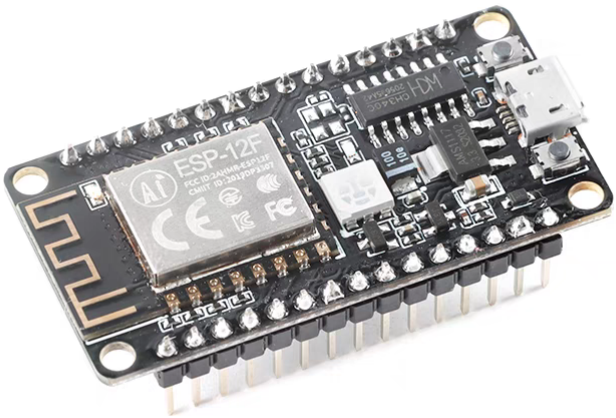
这个项目是我在参加创新创业比赛时参加的一个项目，主要实现的功能有按键控制电机投喂饲料、使用blinker远程控制喂食器投喂饲料、使用离线语言控制投喂饲料、使用小爱语言投喂饲料，另外搭配了ESP32-CAM实现局域网内远程视频。

2.准备环境，需要的软件与硬件

2.1硬件方面

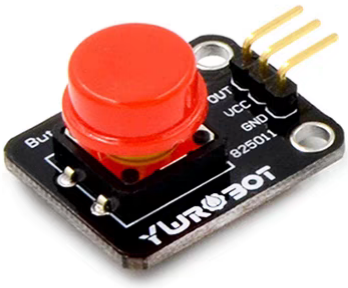
需要准备一块ESP8266开发板、按键、ESP32-CAM、42步进电机、ASR-PRO语音开发板、锂电池、电源分配板、DC公母接头、杜邦线若干。

ESP8266:



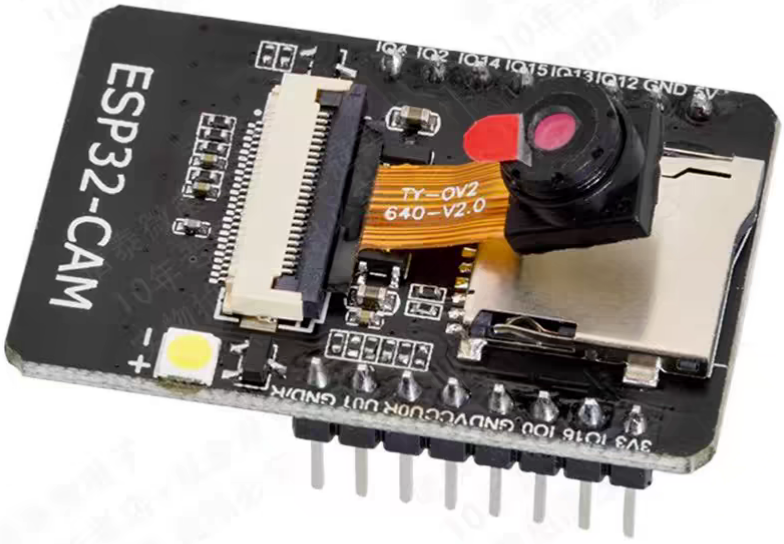
按键(无指定类型，随意找一个合适的自己选即可):

两种款式可选



圆形　　方形

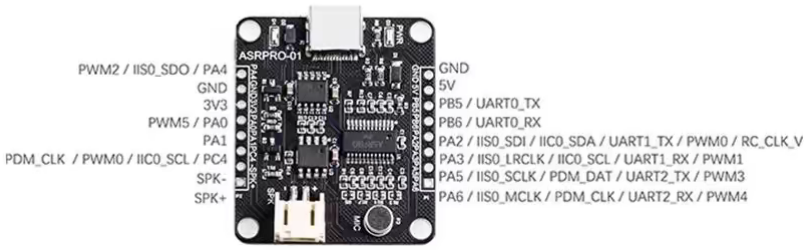
ESP32-CAM：



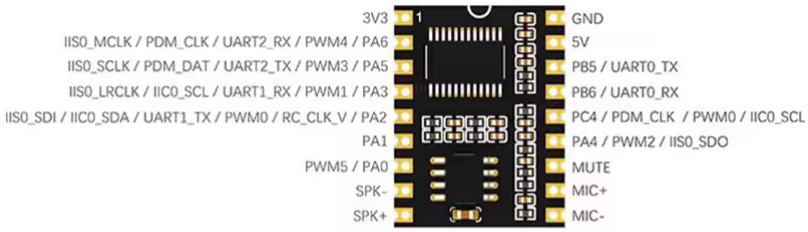
42步进电机(为了方便我选择了套装（包含了电机驱动芯片）):

ASR-PRO语音开发板:

ASRPRO引脚说明



ASRPRO开发板



ASRPRO-CORE核心板

锂电池(随意找一个供电的都行，这里使用的是18650锂电池)：

A品电芯

智能保护板

电源分配板：

2.2软件方面

开发软件：Arduino_IDE、天问block

远程控制软件(APP)：Blinker

下载地址：

Arduino_IDE：Arduino下载

天问block：下载地址

blinker:binker下载

2.3环境配置

Arduino_IDE安装参考文章：参考文章

Arduino_IDE安装ESP8266固件支持包：参考文章

安装固件包的在线方法（需要一个梯子）：

1.在Arduino IDE中，打开 **文件(File) -> 首选项(Preferences)。**在 "附加开发板管理器网址" (Additional Board Manager URLs) 中添加以下网址：

`http://arduino.esp8266.com/stable/package_esp8266com_index.json`

2.打开 **工具(Tools) -> 开发板(Boards) -> 开发板管理器(Board Manager)**

3.在搜索栏中输入 "esp8266"，找到 "esp8266 by ESP8266 Community" 并点击安装（等待下载，如果没有梯子大概率会下载失败）

4.安装完成后，关闭开发板管理器

选择ESP8266开发板

1.再次打开 **工具(Tools) -> 开发板(Boards)。**

2.滚动或搜索找到 **ESP8266 Boards** 并选择您使用的ESP8266型号（这个项目用的是Generic ESP8266 Module）

配置端口和上传速度

1.选择正确的 **串口端口**，这取决于您连接ESP8266的USB端口

2.配置 **上传速度**，通常选择 **115200波特率**

3.开始编写制作

须知：本人菜鸡一个，许多代码资料是在网上查找，然后根据自己的需要进行整合在实现想要的功能的，以下代码并不需要太多编程知识，直接搬来套用就行了

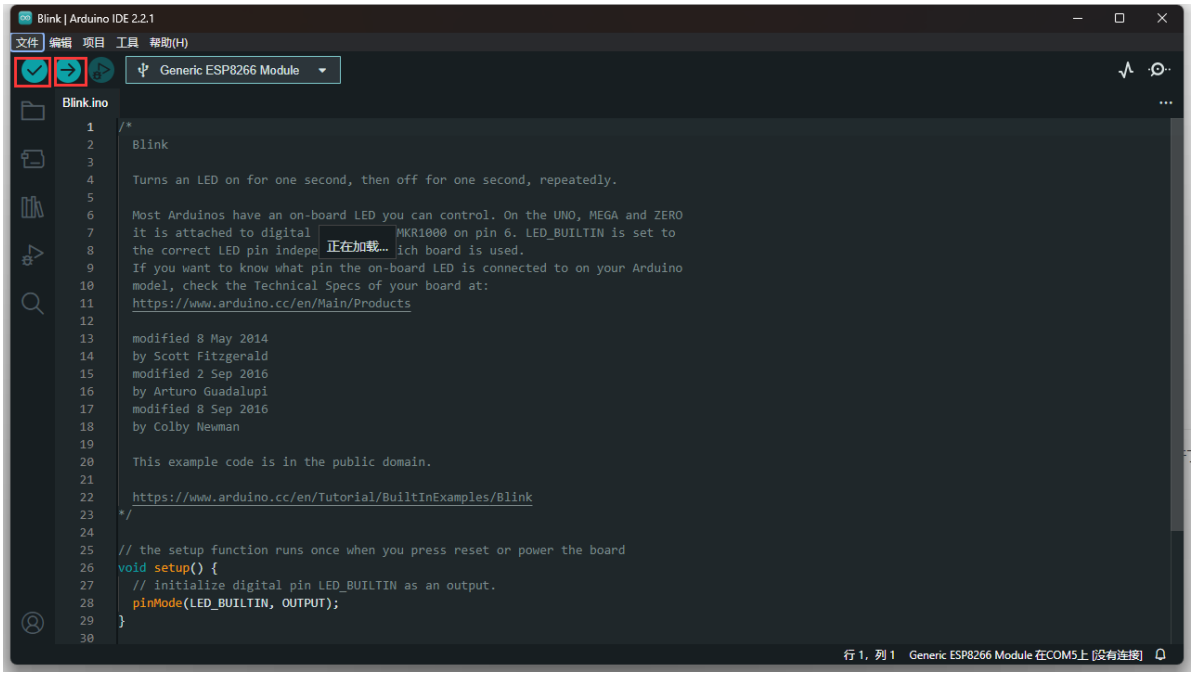
3.1ESP8266代码实现

3.1.1.首先监测软件环境

使用程序对开发板写入一个简单程序来验证你的软件以及配置环境是否正常

点击**文件**，点击**示例**，点击**01.Basics**,点击**Blink。**

选择你的串口，通常是COM1之类的，如果你没有，则要安装CH340驱动。



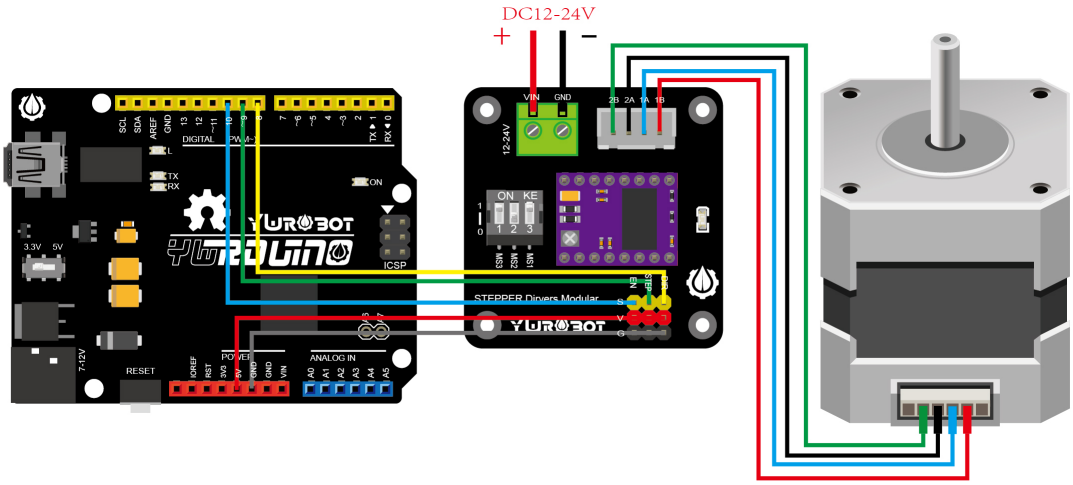
左边红框是编译程序，右边红框是上传（即上传代码到ESP8266）

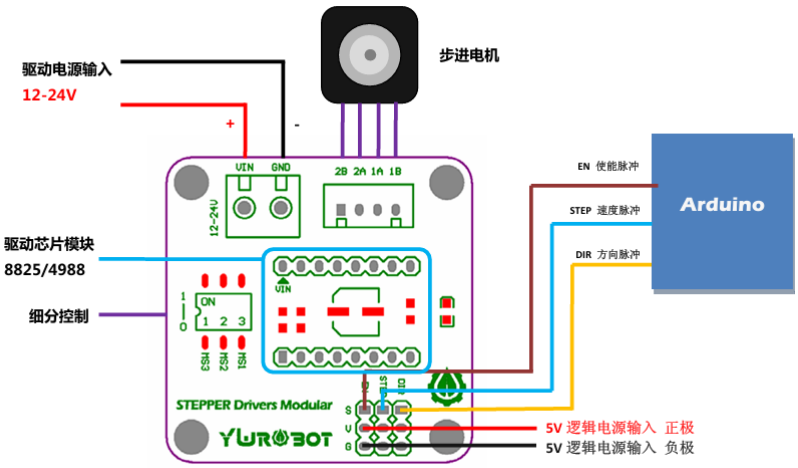
点击上传，上传成功后，若8266开发板的灯一闪一闪的则代表环境是正常的，可以进行下一步操作了

3.1.2分析代码需要实现的功能

1.首先我们需要使用ESP8266来控制42步进电机

将8266与电机控制板和42步进电机进行正确的连接（注意42步进电机供电需要额外的12V电压）





正确连接后在arduinoIDE写入测试程序验证步进电机功能

测试程序：

```
int dirPin = 14;
int stepperPin = 12;
int EN=13;
void setup() {
  pinMode(dirPin, OUTPUT);
  pinMode(stepperPin, OUTPUT);
  pinMode(EN, OUTPUT);
}
void step(boolean dir,int steps){
  digitalWrite(dirPin,dir);
  delay(50);
  for(int i=0;i<steps;i++){
    digitalWrite(stepperPin, HIGH);
    delayMicroseconds(800);
    digitalWrite(stepperPin, LOW);
    delayMicroseconds(800);
  }
}
void loop(){
  digitalWrite(EN, LOW);
  step(true,360);
  delay(500);
  step(false,360);
  delay(500);
```

```
}
```

2.使用按钮控制步进电机

3.添加Blinker，使用blinker远程控制电机

3.接入小爱语音接口，在手机上使用小爱语音来控制电机

4.编译ASR-PRO开发板程序，并让ASR-PRO与ESP8266实现串口通信，作用是使用离线语音控制开发板

5.编译ESP32CAM开发板程序，实现局域网内远程视频功能（可扩展功能，使用内网穿透或服务器部署服务端代码可突破局域网限制）

3.1.3编写代码

1.接下来在以上测试程序的基础上编写一个按键控制电机的程序

```
int dirPin = 14;
int stepperPin = 12;
int EN = 13;
int buttonPin = 2;  // 定义按钮引脚
boolean buttonState = false;

void setup() {
  pinMode(dirPin, OUTPUT);
  pinMode(stepperPin, OUTPUT);
  pinMode(EN, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);  // 设置按钮引脚为输入并启用内部上拉电阻
}

void step(boolean dir, int steps) {
  digitalWrite(dirPin, dir);
  delay(50);
  for (int i = 0; i < steps; i++) {
    digitalWrite(stepperPin, HIGH);
    delayMicroseconds(800);
    digitalWrite(stepperPin, LOW);
    delayMicroseconds(800);
  }
}

void loop() {
  buttonState = digitalRead(buttonPin);  // 读取按钮状态

  if (buttonState == LOW) {  // 检测按钮按下（假设按钮按下时引脚为低电平）
    digitalWrite(EN, LOW);
    step(true, 400);  // 顺时针旋转2圈（假设每圈200步）
    delay(500);
    step(false, 400); // 逆时针旋转2圈（假设每圈200步）
    delay(500);
    while(digitalRead(buttonPin) == LOW); // 等待按钮释放
  }
}
```

2.使用blinker控制电机程序

```
#define BLINKER_PRINT Serial
#include <Blinker.h>

char auth[] = "点灯科技 添加设置的key";
char ssid[] = "WiFi名称";
char pswd[] = "WiFi密码";

int dirPin = 14;
int stepperPin = 12;
int EN = 13;
BlinkerButton Button1("btn-abc");  // 创建一个 Blink 按钮实例

void setup() {
  Serial.begin(115200);

  Blinker.begin(auth);
  pinMode(dirPin, OUTPUT);
  pinMode(stepperPin, OUTPUT);
  pinMode(EN, OUTPUT);

  Button1.attach(button1_callback);  // 绑定按钮回调函数
}

void step(boolean dir, int steps) {
  digitalWrite(dirPin, dir);
  delay(50);
  for (int i = 0; i < steps; i++) {
    digitalWrite(stepperPin, HIGH);
    delayMicroseconds(800);
    digitalWrite(stepperPin, LOW);
    delayMicroseconds(800);
  }
}

void button1_callback(const String & state) {
  BLINKER_LOG("Button state: ", state);

  if (state == "on") {
    digitalWrite(EN, LOW);
    step(true, 400);  // 顺时针旋转2圈（假设每圈200步）
    delay(500);
    step(false, 400); // 逆时针旋转2圈（假设每圈200步）
    delay(500);
  }
}

void loop() {
  Blinker.run();
}
```

要使用 Blink 控制电机，你需要以下步骤：

1. 在手机上安装 Blink App。
2. 创建一个新的项目，选择你使用的开发板（例如 Arduino）和连接方式（WiFi、蓝牙等）。
3. 获取 Blink 提供的 Auth Token。
4. 在代码中包含 Blink 库，并使用该库进行远程控制。

3.给代码加入小爱语音接口（参考文章）

首先在代码的开始接入参数(下面的是小爱插座的参数，使用插座的开关功能来模拟控制电机的旋转功能)

```
#define BLINKER_WIFI
#define BLINKER_MIOT_OUTLET
```

小爱控制程序:

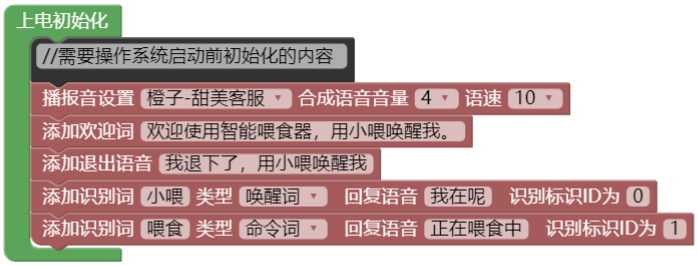
```
#define BLINKER_WIFI // WIFI
#define BLINKER_MIOT_OUTLET // 米家插座类型

#include <Blinker.h>

char auth[] = ""; // 点灯科技设备KEY
char ssid[] = "";  // WIFI名称-只支持2.4G
char pswd[] = ""; // WIFI密码

int dirPin = 14;
int stepperPin = 12;
int EN = 13;
int stepsPerRevolution = 200; // 每圈的步数

// 新建组件对象
BlinkerButton Button1("btn-max");    // 位置1 按钮 数据键名

// 小爱控制
void miotPowerState(const String & state)
{
    BLINKER_LOG("need set power state: ", state);
    // 对小爱同学说“喂食”
    if (state == BLINKER_CMD_ON) {
        digitalWrite(LED_BUILTIN, HIGH);

        BlinkerMIOT.powerState("on");
        BlinkerMIOT.print();

        step(true, stepsPerRevolution * 2); // 顺时针旋转2圈
        delay(500); // 延时0.5s
     }
}

// 位置1 按钮
void button1_callback(const String & state) {
    step(true, stepsPerRevolution * 2); // 顺时针旋转2圈
    delay(500); // 延时0.5s

    step(false, stepsPerRevolution * 2); // 逆时针旋转2圈
    delay(500); // 延时0.5s
}
```

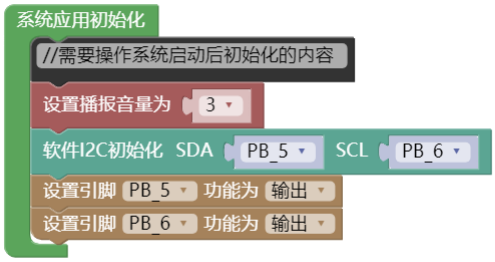
```
void step(boolean dir, int steps) {
    digitalWrite(dirPin, dir);
    delay(50);
    for (int i = 0; i < steps; i++) {
        digitalWrite(stepperPin, HIGH);
        delayMicroseconds(800);
        digitalWrite(stepperPin, LOW);
        delayMicroseconds(800);
    }
}

void setup() {
    // 初始化串口
    Serial.begin(115200);
    BLINKER_DEBUG.stream(Serial);
    Blinker.begin(auth, ssid, pswd);
    Button1.attach(button1_callback);

    // 绑定小爱电源控制
    BlinkerMIOT.attachPowerState(miotPowerState);

    // 初始化步进电机引脚
    pinMode(dirPin, OUTPUT);
    pinMode(stepperPin, OUTPUT);
    pinMode(EN, OUTPUT);
    digitalWrite(EN, LOW); // 启用步进电机驱动器
}

void loop() {
    Blinker.run();
}
```

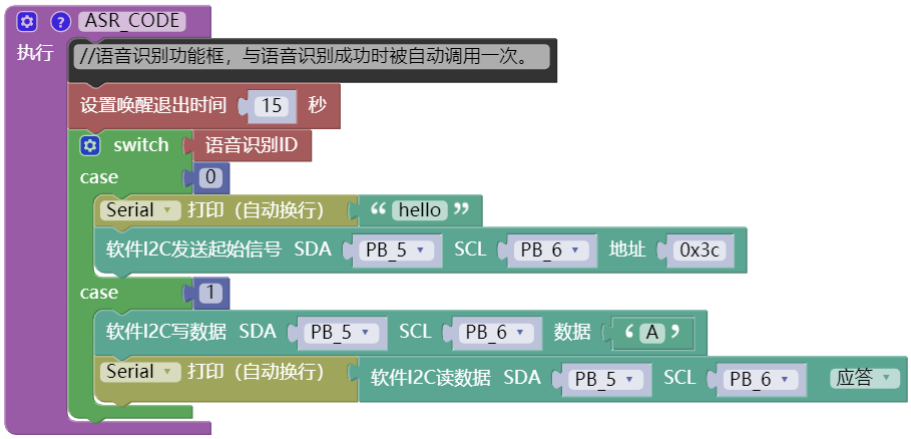
4.编写ASR-PRO语音程序

（我对天问block这个软件也不是很熟悉，经过长久的摸索勉强能实现我想要的效果，有人带就好了TAT）

注意引脚设置以及串口通信就好了，下面是实现拼图：







程序一并给出（图拼好后程序是自动生成的）

```
#include "asr.h"
extern "C"{ void * __dso_handle = 0 ;}
#include "setup.h"
#include "myLib/asr_softiic.h"
#include "HardwareSerial.h"

uint32_t snid;
KSoftIIC KSoftIIC_13_14;
void ASR_CODE();

//{speak:橙子-甜美客服,vol:4,speed:10,platform:haohaodada}
//{playid:10001,voice:欢迎使用智能喂食器，用小喂唤醒我。}
//{playid:10002,voice:我退下了，用小喂唤醒我}

/*描述该功能...
*/
void ASR_CODE(){
  //语音识别功能框，与语音识别成功时被自动调用一次。
  set_state_enter_wakeup(15000);
  switch (snid) {
   case 0:
```

```
    Serial.println("hello");
    KSoftIIC_13_14.start(0x3c);
    break;
   case 1:
    KSoftIIC_13_14.write('A');
    Serial.println((KSoftIIC_13_14.read(0)));
    break;
  }

}

void hardware_init(){
  //需要操作系统启动后初始化的内容
  vol_set(3);
  KSoftIIC_13_14.begin(13,14);
  setPinFun(13,FIRST_FUNCTION);
  pinMode(13,output);
  setPinFun(14,FIRST_FUNCTION);
  pinMode(14,output);
  vTaskDelete(NULL);
}

void setup()
{
  //需要操作系统启动前初始化的内容
  //{ID:0,keyword:"唤醒词",ASR:"小喂",ASRTO:"我在呢"}
  //{ID:1,keyword:"命令词",ASR:"喂食",ASRTO:"正在喂食中"}
}
```

注意：ASR-PRO写好程序后，ESP8266也需要相应的程序与ASR-PRO进行串口通信

ESP8266需添加的程序：

```
mySerial.begin(9600);    // 初始化虚拟串口通信
// 接收虚拟串口数据
    while (mySerial.available()) {
        byte data = mySerial.read();         // 使用 byte 类型来读取十六进制数据
        hexData += String(data, HEX);        // 将十六进制数据添加到积累的字符串中
        if (hexData.length() == 2) {         // 检测是否接收到完整的十六进制数
            char character = char(strtol(hexData.c_str(), NULL, 16)); // 转换为字
符并显示
            Serial.print(character);

            if (character == 'A') {
                digitalWrite(EN, LOW);  // 使能步进电机
                digitalWrite(dirPin, LOW); // 设置旋转方向为顺时针
                for (int i = 0; i < stepsPerRevolution; i++) { // 发送脉冲来驱动步进
电机旋转
                    digitalWrite(stepperPin, HIGH);
                    delayMicroseconds(1600); // 设置脉冲宽度
                    digitalWrite(stepperPin, LOW);
                    delayMicroseconds(1600); // 设置脉冲间隔
                }
            } else if (character == '0') {
```

```
                digitalWrite(stepperPin, HIGH);  // 禁用步进电机
            }
            hexData = ""; // 重置积累的字符串，准备接收下一个十六进制数
        }
    }
```

(我在这里被卡了很久，两个开发板之间串口通信失败，原因是因为天问block编写的程序写入ASR-PRO之后，ASR-PRO使用串口发送是16进制数据，而ESP8266串口通信使用的是10进制，所以ESP8266在接收数据时要先将数据进行进制转换)

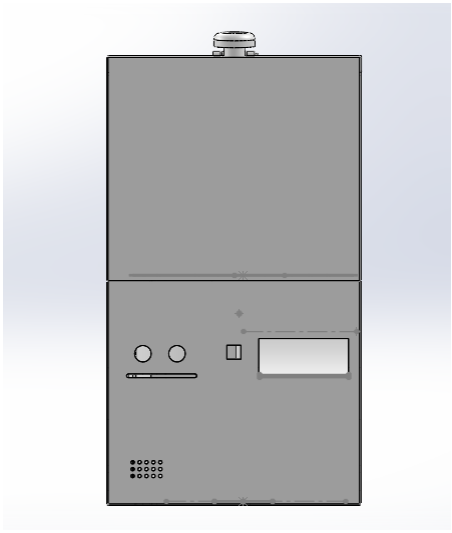
3.2ESP32CAM远程视频实现

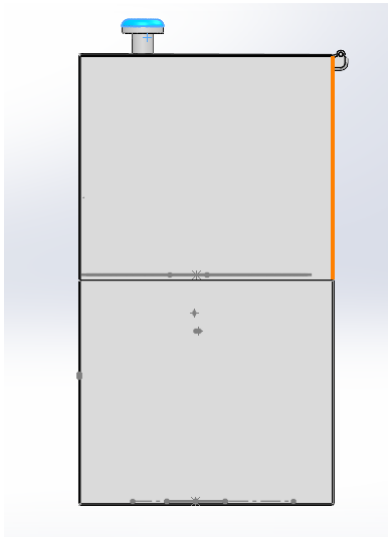
使用的(参考文章)的程序

（我尝试过将ESP8266的程序与ESP32CAM的程序汇总在一起，使用一个开发板来实现控制电机与远程视频的功能，我太菜了，不是很清楚其中的原理，失败了）

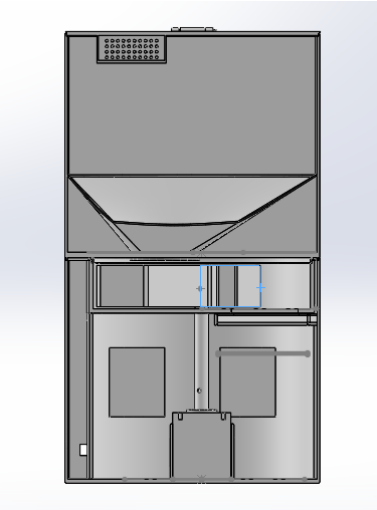
3.3外观设计

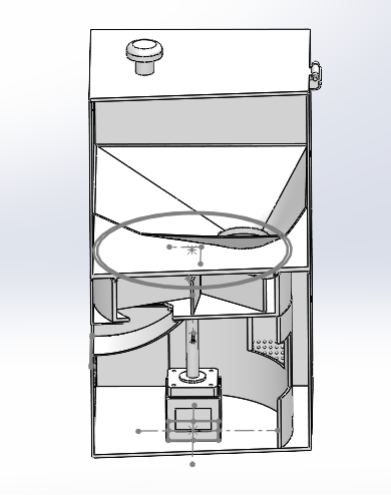
观看视频浅学习了一下solidworks（觉得好的好可以投币支持up主），我自己画了一两个模型，勉强可以把功能给展现出来，但出料口的设计仍然有问题，另外按键孔的布局也有问题，我后面使用了砂纸修改了一下（设计出料口的建议，出料口最好是扁球体，另外在出料口的上方需设计一个缓冲结构，出料口的大小设计为一角钱的三分之二大小差不多）





内部结构：





如果你是第一次使用solidworks进行建模的话，请注意，在建模时如果要用到结构体组装的话，请在为零件建模时留足空隙（2~3mm），不然的话，3D打印完后拼接零件时会因为太紧凑了拼接不上

4.完整的代码与建模文件

ESP8266代码：

```
#define BLINKER_WIFI
#define BLINKER_MIOT_OUTLET

#include <Arduino.h>
#include <Blinker.h>
#include <Stepper.h>
#include <SoftwareSerial.h>



#define MAX_BUFFER_SIZE 50 // 定义缓冲区大小

char auth[] = ""; // 点灯科技设备KEY
char ssid[] = "";  // WIFI名称-只支持2.4G
char pswd[] = ""; // WIFI密码

bool oState = false;
```

```
const int dirPin = 13;
const int stepperPin = 12;
const int EN = 14;
const int buttonPin = 2;
const int stepsPerRevolution = 360;

byte buffer[MAX_BUFFER_SIZE]; // 创建一个存储接收数据的缓冲区
int bufferSize = 0;           // 缓冲区中已接收字节数
String hexData = "";

SoftwareSerial mySerial(4, 5); // RX, TX
Stepper myStepper(stepsPerRevolution, stepperPin, dirPin);

BlinkerButton Button1("btn-abd");

// 函数声明
void rotate360();
void step(bool dir, int steps);
void miotPowerState(const String &state);
void dataRead(const String &data);

// 按钮回调函数
void button1_callback(const String &state) {
    if (Serial.available() > 0)
        BLINKER_LOG("get button state: ", state);
    digitalWrite(EN, LOW);
    step(false, 360); // 设置步数
}

// 步进电机控制函数
void step(bool dir, int steps) {
    digitalWrite(dirPin, dir);
    delay(50);

    for (int i = 0; i < steps; i++) {
        digitalWrite(stepperPin, HIGH);
        delayMicroseconds(1600);
        digitalWrite(stepperPin, LOW);
        delayMicroseconds(1600);
    }
}

// MIOT电源状态处理函数
void miotPowerState(const String &state) {
    BLINKER_LOG("Power state: ", state);

    if (state == BLINKER_CMD_ON) {
        digitalWrite(LED_BUILTIN, LOW);
        digitalWrite(EN, LOW);
        step(false, 360);
        BlinkerMIOT.powerState("on");
        BlinkerMIOT.print();

        oState = true;
```

```
    }
}

// Blinker数据读取处理函数
void dataRead(const String &data) {
    BLINKER_LOG("Blinker readString: ", data);

    Blinker.vibrate();

    uint32_t BlinkerTime = millis();

    Blinker.print("millis", BlinkerTime);
}

void setup() {
    Serial.begin(9600);       // 初始化串口通信
    mySerial.begin(9600);     // 初始化虚拟串口通信
    BLINKER_DEBUG.stream(Serial);
    BLINKER_DEBUG.debugAll();

    Button1.attach(button1_callback);

    Blinker.begin(auth, ssid, pswd);

    pinMode(dirPin, OUTPUT);
    pinMode(stepperPin, OUTPUT);
    pinMode(EN, OUTPUT);
    Blinker.attachData(dataRead);
    BlinkerMIOT.attachPowerState(miotPowerState);

    pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
    Blinker.run();

    // 按钮触发旋转
    if (digitalRead(buttonPin) == LOW) {
        rotate360();
        delay(1000); // 按钮防抖延迟
    }

    // 接收虚拟串口数据
    while (mySerial.available()) {
        byte data = mySerial.read();         // 使用 byte 类型来读取十六进制数据
        hexData += String(data, HEX);        // 将十六进制数据添加到积累的字符串中
        if (hexData.length() == 2) {         // 检测是否接收到完整的十六进制数
            char character = char(strtol(hexData.c_str(), NULL, 16)); // 转换为字
符并显示
            Serial.print(character);

            if (character == 'A') {
                digitalWrite(EN, LOW);  // 使能步进电机
                digitalWrite(dirPin, LOW); // 设置旋转方向为顺时针
```

```
                for (int i = 0; i < stepsPerRevolution; i++) { // 发送脉冲来驱动步进电机旋转
                    digitalWrite(stepperPin, HIGH);
                    delayMicroseconds(1600); // 设置脉冲宽度
                    digitalWrite(stepperPin, LOW);
                    delayMicroseconds(1600); // 设置脉冲间隔
                }
            } else if (character == '0') {
                digitalWrite(stepperPin, HIGH);  // 禁用步进电机
            }
            hexData = ""; // 重置积累的字符串，准备接收下一个十六进制数
        }
    }
}

// 顺时针旋转360度
void rotate360() {
    const int steps = 360; // 根据你的步进电机调整
    for (int i = 0; i < steps; i++) {
        digitalWrite(stepperPin, HIGH);
        delayMicroseconds(1600);
        digitalWrite(stepperPin, LOW);
        delayMicroseconds(1600);
    }
}
```

ESP32代码（请看参考文章）

模型（在文件夹里）

4.1内容补充

blinker有定时功能，具体来说就是设置定时后，时间到了会执行预先设定的功能

操作方法：

点击右上角的编辑功能、在页面下方有定时图标、点击它后会添加到页面、点击确定保存

然后点击页面右上方编辑的左边，点击动作配置，在点击页面下方的自动生成，然后保存

配置代码类似这样

```
[{"cmd":{"btn-abc":"tap"},"text":"点我开关灯"},{"cmd":{"btn-123":"tap"},"text":"点我计数"},{"cmd":{"btn-abd":"tap"},"text":"电机"}]
```

现在点击定时按钮，点击加号就可以实现定时功能啦。