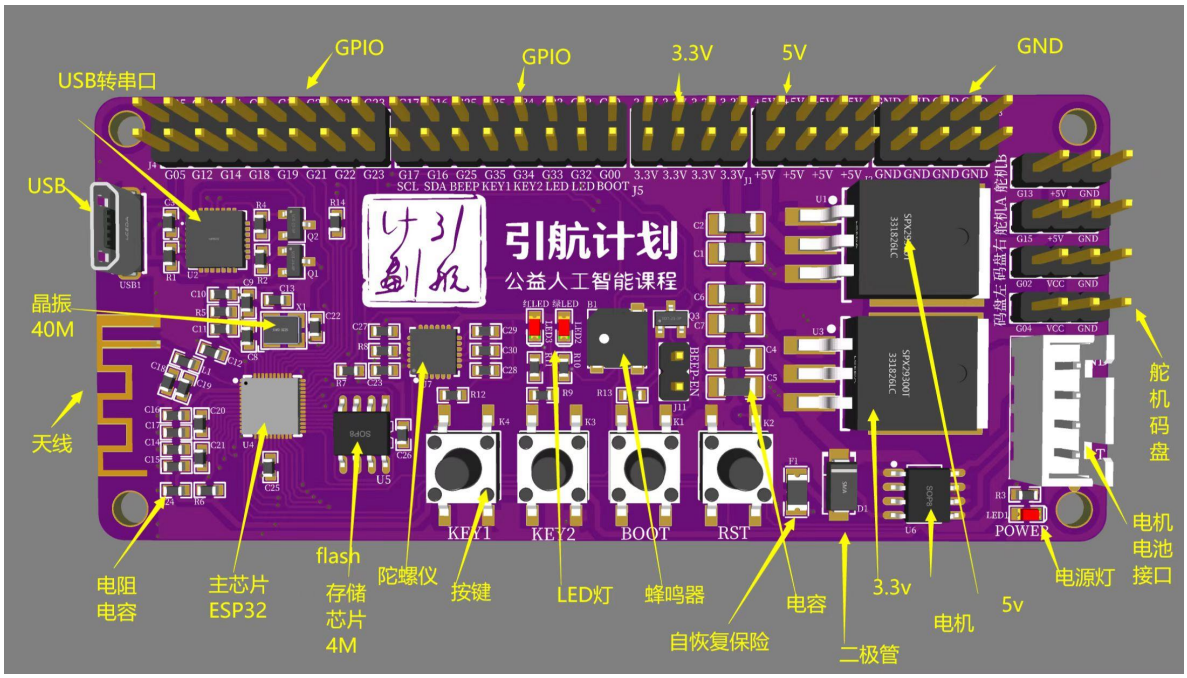
小车驱动开发说明文档

Part1 环境配置

- 在AI Studio的公共数据集中找到 " arduino智能小车 " 数据集，并且下载 " esp32两个目录最新版.zip "文件。（https://aistudio.baidu.com/aistudio/datasetdetail/157740/0）
- 打开文件夹 "免安装环境" 与文件夹 "AppData"，将其中的文件夹"Local"放在C盘文件夹"用户"中的文件夹"AppData"中。
- 打开文件夹 "免安装环境"与文件夹 "Program Files (x86)" ，将其中的文件夹"Arduino"放在C盘文件夹"Program Files (x86)"中。
- 在完成以上操作后可在文件夹"Arduino"中打开应用程序"arduino.exe"，在左上角的工具栏中选择"开发板"选项，选择 "ESP32 Arduino" - "AI Thinker ESP32-CAM"。
- 通过数据线将开发板连接到电脑，在工具栏中选择对应的端口。点击右上角的串口监视器图标打开串口监视器，将波特率改为115200。即可完成开发环境的配置。

注：开发板的构造如图所示



Part2 元器件调试

LED灯

- 由开发板构造图知两个小灯分别连接在GPIO脚的32号和33号。
- 低电平时小灯点亮，高电平时小灯熄灭。

调试程序1：使得两个小灯同时点亮

```
//设置变量
const int led32 = 32; //32号GPIO脚(绿灯)
const int led33 = 33; //33号GPIO脚(红灯)
//基本设置程序(只运行一次)
void setup() {
  pinMode(led32,OUTPUT); //将GPIO脚lede32设置为输出脚
  pinMode(led33,OUTPUT); //将GPIO脚lede33设置为输出脚
}
//主程序(循环运行)
void loop() {
  digitalWrite(led32,LOW); //将led32设置为低电平  由于小灯是一端接电一端接GPIO角，故此时绿灯是亮的(由于绿灯连的是32脚)  (LOW--0 HIGH--1)
  digitalWrite(led33,LOW); //33号管脚控制的红灯此时也亮
}
```

调试程序2：使得32号小灯和33号小灯交替闪烁

```
//目标：使得32号小灯和33号小灯交替闪烁
//设置变量
const int led32 = 32; //32号GPIO脚(绿灯)
const int led33 = 33; //33号GPIO脚(红灯)
//基本程序设置
void setup() {
  pinMode(led32,OUTPUT);
  pinMode(led33,OUTPUT);
}
//主程序
void loop() {
  digitalWrite(led32,LOW);//低电平 点亮
  digitalWrite(led33,HIGH);//高电平 熄灭
  delay(500); //延时500ms
  digitalWrite(led32,HIGH);
  digitalWrite(led33,LOW);
  delay(500);
}
```

调试程序3：呼吸灯

```
//目标：利用lec产生利用pwm信号,进而通过调整占空比(高电平占整个周期的比重)来实现呼吸灯

const int led = 33;

//pwm信号  四部分组成 ①频率 即1s内生成几个周期 如50HZ表示1s内生成50个周期 每一个周期20ms ②信号通道 有16个 用来计数 ③分辨率 8 像素的精度 ④占空比 高电平所占比重
const int freq = 2000; //频率
const int resolution = 8; //分辨率
const int channel = 0; //通道
const int duty_cycle = 0; //占空比


void setup() {
  //配置ledc通道
  ledcSetup(channel,freq,resolution);
  //将Pin脚放置通道中
  ledcAttachPin(led,channel);
}

void loop() {
```

```
  //示例程序，用来控制小灯的亮度，利用它来实现呼吸灯(从最亮到最暗再到最亮)
  //ledcWrite(channel,0);  //占空比为0时，应该是最亮的 相当于GPIO脚是低电平 占空比为255时是很暗很暗的,再大就不亮了

  for (int a = 0;a<=255;a++){ //逐渐变暗
    delay(10);
    ledcWrite(channel,a);
    }

  for (int a = 255;a>=0;a--){ //逐渐变亮
    delay(10);
    ledcWrite(channel,a);
    }
}
```

按钮

- 由开发板构造图知两个按钮KEY1 KEY2分别连接在GPIO脚的35号和34号。

调试程序：使得按一下按钮小灯点亮，再按一下按钮小灯熄灭

```
//设置LED与按钮变量，目标是按一下亮再按一下灭
const int led32 = 32;
const int button35 = 35; //按钮一端是0V，另一端是按下那端，按下时接收到0V(低电平)    35号管脚对应的是最左边的按钮KEY1

//设置按钮初始状态
int buttonstate = 1;

//交替的亮灭状态
boolean change = true; //储存亮灭的情况

void setup() {
  Serial.begin(115200); //将程序的波特率与串口设为一致的
  pinMode(led32,OUTPUT);
  pinMode(button35,INPUT);
}

void loop() {
  delay(500); //延时监测，由于按钮灵敏度太高

  while(digitalRead(button35) == HIGH){} //不按按钮时，空的死循环，也就是什么都不发生

  if (change == true){  //假设现在是暗的，需要让他亮
  change = !change; //!change 的值是 false
  digitalWrite(led32,LOW);
    }
    else{
    change = !change;
    digitalWrite(led32,HIGH);}
}
```

电机

- 电机共有4个脚，除去VCC和GND还有A(26号GPIO脚) B(27号GPIO脚) 两个脚，就是通过A B的对抗来实现电机的转动。如A设置为低电平B设置为高电平时电机正转，A设置为高电平B设置为低电平时电机倒转，A B都设置为高电平时电机不转。

调试程序1：利用ledc信号让电机转动，先加速在减速再加速如此反复

```
//利用ledc信号让电机转动,先加速在减速再加速...
const int a = 26; //a由26号脚控制
const int b = 27; //b由27号脚控制

//pwm 给电机的pwm配置一些准备工作
const int freq = 2000; //频率  1s内2000次
const int resolution = 8;//分辨率  占空比位数精度 8的话就是256位 占空比取值就是0-255
const int channel_A = 0;//通道0  设置信号通道 信号通道一共有16个
const int channel_B = 1;//通道1
const int duty_cycle = 255; //占空比 高电平占周期的比重 可以通过调整占空比生成不同的pwm信号(模拟电压信号)

void setup() {
  //配置ledc通道
  ledcSetup(channel_A,freq,resolution);
  ledcSetup(channel_B,freq,resolution);
  //通过pwm设置管脚的转速值
  ledcAttachPin(a,channel_A);
  ledcAttachPin(b,channel_B);
}

void loop() {
  //这段是A控制B，即正转，且缓慢加速
  for(int i = 0;i<=255;i = i+5){
    ledcWrite(channel_A,i);
    ledcWrite(channel_B,0);
    delay(50);
    }
   //这段是保持最大的速度匀速转
   ledcWrite(channel_A,duty_cycle);
   delay(5000);

  //这段是A控制B，即正转，且缓慢减速
  for(int i = 255;i>=0;i = i-10){
    ledcWrite(channel_A,i);
    ledcWrite(channel_B,0);
    delay(50);
  }
}
```

调试程序2：利用mcpwm模块来控制电机（前面介绍的ledc主要是控制小灯的(虽然也可控制电机)）

```
//mcpwm单元的介绍 控制电机舵机用的(电机驱控：motorcontrol) 前面介绍的ledc主要是控制小灯的(虽然也可控制电机)
//目标:前进后退来回切换

#include "driver/mcpwm.h" //导入模块

void setup() {
  //用选定的MCPWM_UNIT_0来初始化gpio口          ps：在芯片中共有两个电机驱动单元，标号分别为0和1，在每一个单元中又有3个定时器(用来计算占空比)，每个定时器又有两组信号，分别发出信号A和信号B
  mcpwm_gpio_init(MCPWM_UNIT_0,MCPWM0A,26); //0号单元 定时器0 信号A 26号管脚
  mcpwm_gpio_init(MCPWM_UNIT_0,MCPWM0B,27); //0号单元 定时器0 信号B 27号管脚

  //通过mcpwm_config_t结构体为定时器设置频率和初始值
  mcpwm_config_t motor_pwm_config = {
```

```
    .frequency = 1000, //频率
    .cmpr_a = 0, //A的占空比(%)
    .cmpr_b = 0, //B的占空比(%)
    .duty_mode = MCPWM_DUTY_MODE_0, //占空比模式(高电平) --一般不用改
    .counter_mode = MCPWM_UP_COUNTER, //计数器模式(上位计数)  --一般不用改
    };

  //使用以上设置配置PWM0A和PWM0B
  {
    mcpwm_init(MCPWM_UNIT_0,MCPWM_TIMER_0,&motor_pwm_config);
    };
}

void loop() {
  //PWM
  //如果希望禁止可以都设成一样的数,如都设为30
  //全速后退
  mcpwm_set_duty(MCPWM_UNIT_0,MCPWM_TIMER_0,MCPWM_OPR_A,0);//设置占空比
  mcpwm_set_duty(MCPWM_UNIT_0,MCPWM_TIMER_0,MCPWM_OPR_B,100);//这里设100和255无区别 由于是百分比
  mcpwm_start(MCPWM_UNIT_0,MCPWM_TIMER_0);//开始输出信号(mcpwm单元,定时器)
  delay(5000); //持续5s
  mcpwm_stop(MCPWM_UNIT_0,MCPWM_TIMER_0);//停止输出
  //全速前进
  mcpwm_set_duty(MCPWM_UNIT_0,MCPWM_TIMER_0,MCPWM_OPR_A,100);//设置占空比
  mcpwm_set_duty(MCPWM_UNIT_0,MCPWM_TIMER_0,MCPWM_OPR_B,0);
  mcpwm_start(MCPWM_UNIT_0,MCPWM_TIMER_0);//开始输出信号(mcpwm单元,定时器)
  delay(5000);
  mcpwm_stop(MCPWM_UNIT_0,MCPWM_TIMER_0);
}
```

舵机

- 由开发板构造图知预留了两个舵机连接口，舵机A的GPIO脚为15号，舵机B的GPIO脚为13号。舵机需要连三根线分别连接GPIO脚、VCC、GND。
- 舵机用来调整角度，设置舵机pwm信号进而调整其角度。0度是小车左转，180度是小车右转。
- 其频率为固定值(50HZ)，一个周期为20ms，高电平固定在0.5-2.5ms。0.5ms代表0度，2.5ms代表180度，1ms代表45度。因此0.5ms对应的占空比为2.5%，对应的具体数值约为7；2.5ms对应的占空比为12.5%，对应的具体数值约为32。7与32的中间值20便是90度对应的具体数值。

调试程序1：舵机复原90度(舵机的初始化程序)

```
//舵机 通过pwm信号来确定角度,其频率为固定值(50HZ)....一个周期为20ms,高电平固定在0.5-2.5ms,由于0.5ms代表0度,2.5ms代表180度,故1ms代表45度
//0.5ms的占空比为 0.5/20 = 0.025 = 2.5%  即0度对应的占空比为2.5 同理180度对应的占空比为12.5 90度对应的占空比为7.5
//90度是舵机初始化的程序,比较重要
const int a = 15; //舵机A 15号管脚

//pwm
const int f = 50; //固定频率
const int r = 8; //分辨率
const int c = 0; //通道 用0号
const int d = 20; //占空比 实际取值在7-32之间
```

```
void setup() {
  ledcSetup(c,f,r);//将通道c与频率和分辨率相连
  ledcAttachPin(a,c);
}

void loop() {
  ledcWrite(c,d);//将信号写入
}
```

调试程序2：实现舵机三(多)角度旋转

```
//舵机实现不同角度的旋转
const int a = 15; //舵机A 15号管脚

//pwm
const int f = 50; //频率
const int r = 8; //分辨率
const int c = 0; //通道 用0号
const int d = 20; //占空比 实际取值在7-32之间


void setup() {
  ledcSetup(c,f,r);
  ledcAttachPin(a,c);

}
//实现三(多)角度旋转
void loop() {
  //ledcWrite(c,7);
  //delay(1000);
  //ledcWrite(c,20);
  //delay(1000);
  //ledcWrite(c,32);
  //delay(1000);

  //若想实现连续转动则设置for循环
  for (int i=7;i<=32;i++){
  ledcWrite(c,i);
  delay(500);
    }
}
```

调试程序3：利用mcpwm模块来控制舵机

```
//舵机的调试程序

#include "driver/mcpwm.h"

void setup() {
  Serial.begin(115200); //这是查看窗口的方法，让窗口输出一个115200的波特率

  //用选定的MCPWM_UNIT_1来初始化gpio口
  mcpwm_gpio_init(MCPWM_UNIT_1,MCPWM1A,13);  //用的1号PWM单元,该单元中的第一组定时器，配置到13 号管脚

  //通过mcpwm_config_t结构体为定时器设置频率和初始值    结构体部分的另一种写法
  mcpwm_config_t servo_pwm_config;
  servo_pwm_config.frequency = 50;
```

```
  servo_pwm_config.cmpr_a = 0;//初始占空比 0%
  servo_pwm_config.duty_mode = MCPWM_DUTY_MODE_0;//占空比的类型
  servo_pwm_config.counter_mode = MCPWM_UP_COUNTER;//计数器的类型

  //使用以上设置配置PWM1A
  mcpwm_init(MCPWM_UNIT_1,MCPWM_TIMER_1,&servo_pwm_config);
}

void loop() {
  Serial.println("Setting motor pwm success!");//在窗口中输出这样的一段话
  //mcpwm_stop(MCPWM_UNIT_1,MCPWM_TIMER_1);
  mcpwm_set_duty(MCPWM_UNIT_1,MCPWM_TIMER_1,MCPWM_OPR_A,7.5); //占空比取值2.5%-12.5%
7.5%是90度
  delay(1000);
  mcpwm_set_duty(MCPWM_UNIT_1,MCPWM_TIMER_1,MCPWM_OPR_A,2.5); //0度
  delay(1000);
  mcpwm_set_duty(MCPWM_UNIT_1,MCPWM_TIMER_1,MCPWM_OPR_A,7.5);
  delay(1000);
  mcpwm_set_duty(MCPWM_UNIT_1,MCPWM_TIMER_1,MCPWM_OPR_A,12.5); //180度
  delay(1000);
}
```

调试程序4：将电机与舵机程序结合

```
//舵机与电机结合到一起

#include "driver/mcpwm.h"

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

  //初始化gpio口
  mcpwm_gpio_init(MCPWM_UNIT_1,MCPWM1A,13);//设置舵机脚
  mcpwm_gpio_init(MCPWM_UNIT_0,MCPWM0A,26);//设置电机
  mcpwm_gpio_init(MCPWM_UNIT_0,MCPWM0B,27);//设置电机

  //通过mcpwm_config_t结构体为定时器设置频率和初始值  舵机
  mcpwm_config_t servo_pwm_config;
  servo_pwm_config.frequency = 50;
  servo_pwm_config.cmpr_a = 0;
  servo_pwm_config.duty_mode = MCPWM_DUTY_MODE_0;
  servo_pwm_config.counter_mode = MCPWM_UP_COUNTER;

  //通过mcpwm_config_t结构体为定时器设置频率和初始值  电机
  mcpwm_config_t motor_pwm_config = {
    .frequency = 1000,
    .cmpr_a = 0,
    .cmpr_b = 0,
    .duty_mode = MCPWM_DUTY_MODE_0,
    .counter_mode = MCPWM_UP_COUNTER,
    };

  //使用以上设置配置PWM1A
  mcpwm_init(MCPWM_UNIT_1,MCPWM_TIMER_1,&servo_pwm_config);
  //使用以上设置配置PWM0A和PWM0B
  mcpwm_init(MCPWM_UNIT_0,MCPWM_TIMER_0,&servo_pwm_config);
}
```

```
void loop() {
  Serial.println("Setting motor pwm success!");
  mcpwm_set_duty(MCPWM_UNIT_0,MCPWM_TIMER_0,MCPWM_OPR_A,100);//全速前进
  mcpwm_set_duty(MCPWM_UNIT_0,MCPWM_TIMER_0,MCPWM_OPR_B,0);

  mcpwm_set_duty(MCPWM_UNIT_1,MCPWM_TIMER_1,MCPWM_OPR_A,7.5);//直行
  delay(5000);
  mcpwm_set_duty(MCPWM_UNIT_1,MCPWM_TIMER_1,MCPWM_OPR_A,2.5);//左转
  delay(5000);
  mcpwm_set_duty(MCPWM_UNIT_1,MCPWM_TIMER_1,MCPWM_OPR_A,7.5);
  delay(5000);
  mcpwm_set_duty(MCPWM_UNIT_1,MCPWM_TIMER_1,MCPWM_OPR_A,12.5);//右转
  delay(5000);
}
```

蜂鸣器

- 蜂鸣器使用25号GPIO脚控制，已经分别接了VCC与GND，因此只需将二者连接起来即可工作。
- 开发板上是无源蜂鸣器，因此可以通过调控频率发出不同的音调，通过调整占空比去发出不同的响度。
- 依然是使用pwm信号进行控制。

调试程序：蜂鸣器实现不同的音调和响度

```
//蜂鸣器 25号管脚
const int buzzer = 25;

//pwm
const int f = 1000;
const int c = 0;
const int r = 8;
const int d = 128 ;//占空比 取值在0-255 控制响度

void setup() {
  ledcSetup(c,f,r);
  ledcAttachPin(buzzer,c);
}

void loop() {
  //固定频率修改占空比查看响度变化
  ledcWriteTone(c,f);
  for(int d = 0;d<=255;d = d+10){
      ledcWrite(c,d);
      delay(1000);
  }

  //固定占空比修改频率查看音调的变化
  ledcWrite(c,d);
  for(int f = 200;f<=2000;f = f+10){
      ledcWriteTone(c,f);
      delay(1000);
  }
}
```

摄像头

- 在用USB线连接好摄像头后，选择右上角的"文件"——"示例"——"ESP32"——"Camera"——"CameraWebServer"，会出现示例程序，对文件"CameraWebServer.ino"做一些修改后得到以下程序：(特别注意要将代码22行改为电脑现在正在连接的WiFi的名称和密码)

```
#include "esp_camera.h"
#include <WiFi.h>

//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//            Ensure ESP32 Wrover Module or other board with PSRAM is selected
//            Partial images will be transmitted if image exceeds buffer size
//

// Select camera model
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM

#include "camera_pins.h"

//这里需要修改为电脑现在正在连接的WiFi的名称和密码!
const char* ssid = "Nova7";
const char* password = "301301301";

void startCameraServer();

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
```

```
  config.pixel_format = PIXFORMAT_JPEG;

  // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
  //                      for larger pre-allocated frame buffer.
  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
  }
  // drop down frame size for higher initial frame rate
  s->set_framesize(s, FRAMESIZE_QVGA);

#if defined(CAMERA_MODEL_M5STACK_WIDE) || defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {
```

```
  // put your main code here, to run repeatedly:
  delay(10000);
}
```

- 之后将程序写入到摄像头中，打开串口监视器，进入网址"192.168.94.132"即可进入摄像头控制界面。

Part3 小车驱动

总程序

- 通过以下程序生成一个小车的控制界面，通过该界面控制小车的前进后退左右转即摄像头功能，在电脑端或手机端均可以使用。
- 电脑和摄像头同时连接开发板发出的Wifi信号即可使用。

```
/*********
  https://randomnerdtutorials.com/esp32-esp8266-input-data-html-form/
*********/

#include "WiFi.h"
#include "esp_timer.h"
#include "Arduino.h"
#include "soc/soc.h"           // Disable brownout problems
#include "soc/rtc_cntl_reg.h"  // Disable brownout problems
#include "driver/rtc_io.h"
#include "driver/mcpwm.h"
#include <ESPAsyncWebServer.h>
#include <StringArray.h>
#include <FS.h>

esp_err_t esp_err;

// LED pin
const int front_led_pin = 32;
const int back_led_pin = 33;
const int left_turn_led_pin = 21;
const int right_turn_led_pin = 22;
const int brake_led_pin = 23;

// encoder pin
const int encoder_pin = 2;
int count = 0;
float encoder_speed = 0.0;
int encoder_interval_ms = 500;

// motor pwm pin
const int motor_pwm_pin_A = 27;
const int motor_pwm_pin_B = 26;
// servo pwm pin
const int servo_pwm_pin = 13;

// Set your access point network credentials
// const char* ssid = "ESP32-Access-Point";
const char* ssid = "minicar11"; //这个信号是由开发板发出的,可以自己改
const char* password = "1234123411";
```

```
// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

// motor parameters
int motor_duty_cycle = 30;
int servo_turn_angle = 45;
float servo_duty_cycle_center = 7.5;
float servo_duty_cycle_differ = 5;

void toggle_light(int color);
void control_all_light(bool);
void move_forward();
void move_backward();
void motor_stop();
void turn_left();
void turn_right();
void straight();

void IRAM_ATTR count_add() {
  count += 1;
}

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML>
<html>
<head>
  <title>Mini-Car Controller</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
<style>
  { font-family: sans-serif; background: #eee; padding: 1rem; }
  body { max-width: 1200px; margin: 0 auto; background: white; }
  nav {
    background: rgb(50, 70, 99);
    display: flex;
    align-items: center;
    padding: 0 0.5rem;
    min-height: 4em;
  }
  nav h1 {
      flex: auto; margin: 0;
      color: #ffffff;
      font: 1em lucida-grande;
      font-size: 32px;
      font-weight: 1000;
      margin-left: 0.3em;
  }
  .content { padding: 0 1rem 1rem; }
  .content > header {
      /* border-bottom: 2px solid rgba(115, 133, 159, 0.5); */
      display: flex; align-items: flex-end;
      /* background-color: #9fb2bb; */
  }
  .content > header h1 {
      font: 1em lucida-grande;
      font-size: 24px;
      font-weight: 1000;
      color: #ff0000;
```

```
        flex: auto;
        margin: 1rem 0 0.3rem 0;
        margin-left: 0.3em;
    }
    .content p {
        margin: 5px;
        font-family: 'Courier New', Courier, monospace;
        font-size: 16px;
        font-weight: bold;
        line-height: 30px;
    }
    .content input[type=button] {
        align-self: start; min-width: 8em; min-height: 2em;
        font: 1em lucida-grande;
        font-size: 16px;
        font-weight: 1000;
        border: 0px;
        border-radius: 0.4em;
        background: rgba(115, 133, 159, 0.25);
    }
    .content input[type=button]:active {
        background: rgba(115, 133, 159, 0.507);
    }
</style>
</head>
<body>
    <nav>
      <h1 align="center">Mini car controller</h1>
    </nav>
    <section class="content">
      <header>
        <h1 align="center">Camera</h1>
      </header>
      <p align="center">
        <img src="http://192.168.4.5/capture" id="photo" width=300em>
        <br>
        <input type="button" id="start_stream" name="Start Stream" value="Start Stream">
        <input type="button" id="stop_stream" name="Stop Stream" value="Stop Stream">
      </p>
      <script>
        var isStreaming = true;
        var isRecording = false;
        document.getElementById('start_stream').onclick = function() {
          isStreaming = true;
        }
        document.getElementById('stop_stream').onclick = function() {
          isStreaming = false;
        }
        function refresh_img() {
          if (isStreaming && (!isRecording)) {
            document.getElementById("photo").src = "http://192.168.4.5/capture"
                                                  + '?_=' + (new Date()).getTime();
          }
        }
        setInterval(refresh_img, 2000);
      </script>
      <br>
      <p align="center">
```

```
      Set Record Time (in minute):
      <input class="slider" type="range" min="1" max="10" value="2" step="1"
id="record_time">
      <span id="record_time_span"></span>
      <br>
      Set Record Interval (in second):
      <input class="slider" type="range" min="5" max="20" value="5" step="1"
id="record_interval">
      <span id="record_interval_span"></span>
      <style>
        input[type=range] {
            /*滑动条背景*/
            -webkit-appearance: none;
            background-color: rgba(115, 133, 159, 0.5);
            height: 8px;
            width: 100px;
        }
        input[type=range]::-webkit-slider-thumb {
            /*滑动条操作按钮样式*/
            -webkit-appearance: none;
            border-radius: 5px;
            background: rgb(255, 0, 0);
            width: 15px;
            height: 15px;
        }
      </style>
      <script>
        document.getElementById('record_time_span').innerHTML = 2;
        document.getElementById('record_interval_span').innerHTML = 5;
        var record_time = document.getElementById('record_time');
        var record_interval = document.getElementById('record_interval');
        var current;
        record_time.oninput = function() {
          current = this.value;
          document.getElementById('record_time_span').innerHTML = current;
        }
        record_interval.oninput = function() {
          current = this.value;
          document.getElementById('record_interval_span').innerHTML = current;
        }
      </script>
    </p>

    <p align="center">
      <input type="button" name="Start Record" value="Start Record" id="start_record">
      <input type="button" name="Stop Record" value="Stop Record" id="stop_record">
      <script>
        // XMLHttpRequest 在不刷新页面的情况下请求特定 URL，获取数据
        var xhttp = new XMLHttpRequest();
        document.getElementById('start_record').onclick = function() {
          xhttp.open("GET", "http://192.168.4.5/record?record_time="
                            +
document.getElementById('record_time_span').innerHTML.toString()
                            + "&record_interval="
                            +
document.getElementById('record_interval_span').innerHTML.toString()
          );
          xhttp.send();
```

```
          isRecording = true;
          function set_isRecording_false() {
            isRecording = false;
            console.log("Stop record. You can get ip camera stream now.");
          }
          setTimeout(set_isRecording_false,
document.getElementById('record_time_span').innerHTML * 60 * 1000);
          console.log("Start record... Can't get ip camera stream now.");
        }
        document.getElementById('stop_record').onclick = function() {
          isRecording = false;
          xhttp.open("GET", "http://192.168.4.5/stop_record");
          xhttp.send();
          console.log("Stop record. You can get ip camera stream now.");
        }
      </script>
    </p>
  </section>

  <section class="content">
    <header>
        <h1 align="center">Light</h1>
    </header>
    <p align="center">
      <input type="button" name="Front Light" value="Front Light" id="front_light">
      <input type="button" name="Brake Light" value="Brake Light" id="brake_light">
    </p>
    <script>
      var xhttp = new XMLHttpRequest();
      document.getElementById('front_light').onclick = function() {
        xhttp.open("POST", "/front_light");
        xhttp.send();
        console.log('toggle front light');
      }
      document.getElementById('brake_light').onclick = function() {
        xhttp.open("POST", "/back_light");
        xhttp.send();
        console.log('toggle back light');
      }
    </script>
    <br>

    <header>
        <h1 align="center">Move</h1>
    </header>
    <p align="center">
      Real-Time Speed From Encoder: <span id="encoder_span">0.0</span>
    </p>
    <script>
      var xhttp_recorder = new XMLHttpRequest();
      xhttp_recorder.onreadystatechange = function() {
        if (xhttp_recorder.status === 200) {
          document.getElementById('encoder_span').innerHTML = this.responseText;
        }
      }
      function refresh_speed() {
        xhttp_recorder.open("GET", "/get_encoder");
        xhttp_recorder.send();
```

```
      }
      setInterval(refresh_speed, 200);
    </script>

    <p align="center">
      Set Speed:
      <input class="slider" type="range" min="30" max="100" value="60" step="10"
id="speed">
      <span id="speed_span"></span>
      <br>
      Set Turning Angle:
      <input class="slider" type="range" min="15" max="45" value="15" step="30"
id="angle">
      <span id="angle_span"></span>
      <style>
        input[type=range] {
            /*滑动条背景*/
            -webkit-appearance: none;
            background-color: rgba(115, 133, 159, 0.5);
            height: 8px;
            width: 100px;
        }
        input[type=range]::-webkit-slider-thumb {
            /*滑动条操作按钮样式*/
            -webkit-appearance: none;
            border-radius: 5px;
            background: rgb(255, 0, 0);
            width: 15px;
            height: 15px;
        }
      </style>
      <script>
        var xhttp = new XMLHttpRequest();
        document.getElementById('speed_span').innerHTML = 60;
        document.getElementById('angle_span').innerHTML = 15;
        var motor_speed = document.getElementById('speed');
        var servo_angle = document.getElementById('angle');
        var current;
        motor_speed.oninput = function() {
          current = this.value;
          document.getElementById('speed_span').innerHTML = current;
        }
        servo_angle.oninput = function() {
          current = this.value;
          document.getElementById('angle_span').innerHTML = current;
        }
        motor_speed.onchange = function() {
          current = this.value;
          xhttp.open("POST", "/change_speed?speed=" + current.toString());
          xhttp.send();
          console.log('change speed');
        }
        servo_angle.onchange = function() {
          current = this.value;
          xhttp.open("POST", "/change_turn_angle?angle=" + current.toString());
          xhttp.send();
          console.log('change turn angle');
        }
```

```
      </script>
    </p>
    <br>

    <p align="center">
      <input type="button" name="Forward" value="Forward" id="forward">
      <input type="button" name="Stop" value="Stop" id="stop">
      <input type="button" name="Backward" value="Backward" id="backward">
    </p>
    <p align="center">
      <input type="button" name="Left" value="Left" id="left">
      <input type="button" name="Straight" value="Straight" id="straight">
      <input type="button" name="Right" value="Right" id="right">
    </p>
    <script>
      // XMLHttpRequest 在不刷新页面的情况下请求特定 URL，获取数据
      var xhttp = new XMLHttpRequest();
      // button elements
      var forward_button = document.getElementById('forward');
      var backward_button = document.getElementById('backward');
      var stop_button = document.getElementById('stop');
      var left_button = document.getElementById('left');
      var right_button = document.getElementById('right');
      var straight_button = document.getElementById('straight');

      forward_button.onclick = function() {
        xhttp.open("POST", "/forward");
        xhttp.send();
        console.log('move forward');
      }
      backward_button.onclick = function() {
        xhttp.open("POST", "/backward");
        xhttp.send();
        console.log('move backward');
      }
      stop_button.onclick = function() {
        xhttp.open("POST", "/stop");
        xhttp.send();
        console.log('stop');
      }
      left_button.onclick = function() {
        xhttp.open("POST", "/left");
        xhttp.send();
        console.log('left');
      }
      right_button.onclick = function() {
        xhttp.open("POST", "/right");
        xhttp.send();
        console.log('right');
      }
      straight_button.onclick = function() {
        xhttp.open("POST", "/straight");
        xhttp.send();
        console.log('straight');
      }
    </script>
  </section>
</body>
```

```
</html>)rawliteral";

void setup() {
  // Serial port for debugging purposes
  Serial.begin(115200);

  WiFi.mode(WIFI_AP);
  if(!WiFi.softAPConfig(IPAddress(192, 168, 4, 1), IPAddress(192, 168, 4, 1),
IPAddress(255, 255, 0, 0))){
      Serial.println("AP Config Failed");
  }
  WiFi.softAP(ssid, password, 1, 0, 10);

  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);

  // Turn-off the 'brownout detector'
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);

  // set led pinmode
  pinMode(front_led_pin, OUTPUT);
  pinMode(back_led_pin, OUTPUT);
  pinMode(left_turn_led_pin, OUTPUT);
  pinMode(right_turn_led_pin, OUTPUT);
  pinMode(brake_led_pin, OUTPUT);

  // set encoder interrupt
  pinMode(encoder_pin, INPUT);
  attachInterrupt(encoder_pin, count_add, RISING);

  // motor pwm config
  mcpwm_gpio_init(MCPWM_UNIT_0, MCPWM0A, motor_pwm_pin_A);
  mcpwm_gpio_init(MCPWM_UNIT_0, MCPWM0B, motor_pwm_pin_B);
  mcpwm_config_t motor_pwm_config = {
    .frequency = 1000,
    .cmpr_a = 0,
    .cmpr_b = 0,
    .duty_mode = MCPWM_DUTY_MODE_0,
    .counter_mode = MCPWM_UP_COUNTER,
  };
  esp_err = mcpwm_init(MCPWM_UNIT_0, MCPWM_TIMER_0, &motor_pwm_config);
  if (esp_err == 0)
    Serial.println("Setting motor pwm success!");
  else {
    Serial.print("Setting motor pwm fail, error code: ");
    Serial.println(esp_err);
  }

  // servo pwm config
  mcpwm_gpio_init(MCPWM_UNIT_1, MCPWM1A, servo_pwm_pin);
  mcpwm_config_t servo_pwm_config;
  servo_pwm_config.frequency = 50;
  servo_pwm_config.cmpr_a = 0;
  servo_pwm_config.duty_mode = MCPWM_DUTY_MODE_0;
  servo_pwm_config.counter_mode = MCPWM_UP_COUNTER;
  esp_err = mcpwm_init(MCPWM_UNIT_1, MCPWM_TIMER_1, &servo_pwm_config);
  if (esp_err == 0)
```

```
    Serial.println("Setting servo pwm success!");
  else {
    Serial.print("Setting servo pwm fail, error code: ");
    Serial.println(esp_err);
  }
  mcpwm_start(MCPWM_UNIT_1, MCPWM_TIMER_1);

  // Route for web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send_P(200, "text/html", index_html);
  });

  server.on("/front_light", HTTP_POST, [](AsyncWebServerRequest * request) {
    toggle_light(1);
    request->send(200);
  });
  server.on("/back_light", HTTP_POST, [](AsyncWebServerRequest * request) {
    toggle_light(2);
    request->send(200);
  });
  server.on("/get_encoder", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send(200, "text/plain", String(encoder_speed));
//    request->send_P(200, "text/plain", "123");
  });
  server.on("/change_speed", HTTP_POST, [](AsyncWebServerRequest * request) {
    motor_duty_cycle = request->getParam("speed")->value().toInt();
    request->send(200);
  });
  server.on("/change_turn_angle", HTTP_POST, [](AsyncWebServerRequest * request) {
    servo_turn_angle = request->getParam("angle")->value().toInt();
    if (servo_turn_angle == 45) servo_duty_cycle_differ = 5;
    else servo_duty_cycle_differ = 1.5;
    request->send(200);
  });
  server.on("/forward", HTTP_POST, [](AsyncWebServerRequest * request) {
//    digitalWrite(back_led_pin, LOW);
    move_forward();
    request->send(200);
  });
  server.on("/backward", HTTP_POST, [](AsyncWebServerRequest * request) {
//    digitalWrite(back_led_pin, HIGH);
    move_backward();
    request->send(200);
  });
  server.on("/stop", HTTP_POST, [](AsyncWebServerRequest * request) {
//    digitalWrite(back_led_pin, LOW);
    motor_stop();
    request->send(200);
  });
  server.on("/left", HTTP_POST, [](AsyncWebServerRequest * request) {
//    digitalWrite(left_turn_led_pin, LOW);
//    digitalWrite(right_turn_led_pin, HIGH);
    turn_left();
    request->send(200);
  });
  server.on("/right", HTTP_POST, [](AsyncWebServerRequest * request) {
//    digitalWrite(left_turn_led_pin, HIGH);
//    digitalWrite(right_turn_led_pin, LOW);
```

```
    turn_right();
    request->send(200);
  });
  server.on("/straight", HTTP_POST, [](AsyncWebServerRequest * request) {
//    digitalWrite(left_turn_led_pin, HIGH);
//    digitalWrite(right_turn_led_pin, HIGH);
    straight();
    request->send(200);
  });
  // Start server
  server.begin();

  control_all_light(true);
  delay(500);
  control_all_light(false);
  delay(500);
  control_all_light(true);
  delay(500);
  control_all_light(false);
}

void loop() {
  count = 0;
  delay(encoder_interval_ms);
  encoder_speed = count / 18.0 / 21 * 6.2 * 3.14 * 1000 / encoder_interval_ms;
//  Serial.print("Speed: ");
//  Serial.println(encoder_speed);
}

/*
const int front_led_pin = 21;
const int back_led_pin = 22;
const int left_turn_led_pin = 32;
const int right_turn_led_pin = 33;
const int brake_led_pin = 23;
*/

// some functions
void toggle_light(int color) {
  if (color == 1) {
    bool state = digitalRead(front_led_pin);
    digitalWrite(front_led_pin, !state);
  }
  else if (color == 2) {
    bool state = digitalRead(back_led_pin);
    digitalWrite(back_led_pin, !state);
  }
}
void control_all_light(bool flag) {
  digitalWrite(front_led_pin, !flag);
  digitalWrite(back_led_pin, !flag);
  digitalWrite(left_turn_led_pin, flag);
  digitalWrite(right_turn_led_pin, flag);
  digitalWrite(brake_led_pin, flag);
}

void move_forward() {
  Serial.println("--- move forward...");
```

```
  mcpwm_stop(MCPWM_UNIT_0, MCPWM_TIMER_0);
  mcpwm_set_duty(MCPWM_UNIT_0, MCPWM_TIMER_0, MCPWM_OPR_A, 0);
  mcpwm_set_duty(MCPWM_UNIT_0, MCPWM_TIMER_0, MCPWM_OPR_B, motor_duty_cycle);
  mcpwm_start(MCPWM_UNIT_0, MCPWM_TIMER_0);
}
void move_backward() {
  mcpwm_stop(MCPWM_UNIT_0, MCPWM_TIMER_0);
  mcpwm_set_duty(MCPWM_UNIT_0, MCPWM_TIMER_0, MCPWM_OPR_A, motor_duty_cycle);
  mcpwm_set_duty(MCPWM_UNIT_0, MCPWM_TIMER_0, MCPWM_OPR_B, 0);
  mcpwm_start(MCPWM_UNIT_0, MCPWM_TIMER_0);
  Serial.println("--- move backward...");
}
void motor_stop() {
  Serial.println("--- motor stop...");
  mcpwm_stop(MCPWM_UNIT_0, MCPWM_TIMER_0);
  mcpwm_set_duty(MCPWM_UNIT_0, MCPWM_TIMER_0, MCPWM_OPR_A, 100);
  mcpwm_set_duty(MCPWM_UNIT_0, MCPWM_TIMER_0, MCPWM_OPR_B, 100);
  mcpwm_start(MCPWM_UNIT_0, MCPWM_TIMER_0);
}
void turn_left() {
  mcpwm_set_duty(MCPWM_UNIT_1, MCPWM_TIMER_1, MCPWM_OPR_A, servo_duty_cycle_center -
servo_duty_cycle_differ);
}
void turn_right() {
  mcpwm_set_duty(MCPWM_UNIT_1, MCPWM_TIMER_1, MCPWM_OPR_A, servo_duty_cycle_center +
servo_duty_cycle_differ);
}
void straight() {
  mcpwm_set_duty(MCPWM_UNIT_1, MCPWM_TIMER_1, MCPWM_OPR_A, servo_duty_cycle_center);
}
```

动作的封装函数

前进

```
void move_forward() {
  Serial.println("--- move forward...");
  mcpwm_stop(MCPWM_UNIT_0, MCPWM_TIMER_0);
  mcpwm_set_duty(MCPWM_UNIT_0, MCPWM_TIMER_0, MCPWM_OPR_A, 0);
  mcpwm_set_duty(MCPWM_UNIT_0, MCPWM_TIMER_0, MCPWM_OPR_B, motor_duty_cycle);
  mcpwm_start(MCPWM_UNIT_0, MCPWM_TIMER_0);
}
```

后退

```
void move_backward() {
  mcpwm_stop(MCPWM_UNIT_0, MCPWM_TIMER_0);
  mcpwm_set_duty(MCPWM_UNIT_0, MCPWM_TIMER_0, MCPWM_OPR_A, motor_duty_cycle);
  mcpwm_set_duty(MCPWM_UNIT_0, MCPWM_TIMER_0, MCPWM_OPR_B, 0);
  mcpwm_start(MCPWM_UNIT_0, MCPWM_TIMER_0);
  Serial.println("--- move backward...");
}
```

停止

```
void motor_stop() {
  Serial.println("--- motor stop...");
  mcpwm_stop(MCPWM_UNIT_0, MCPWM_TIMER_0);
  mcpwm_set_duty(MCPWM_UNIT_0, MCPWM_TIMER_0, MCPWM_OPR_A, 100);
  mcpwm_set_duty(MCPWM_UNIT_0, MCPWM_TIMER_0, MCPWM_OPR_B, 100);
  mcpwm_start(MCPWM_UNIT_0, MCPWM_TIMER_0);
}
```

左转

```
void turn_left() {
  mcpwm_set_duty(MCPWM_UNIT_1, MCPWM_TIMER_1, MCPWM_OPR_A, servo_duty_cycle_center -
servo_duty_cycle_differ);
}
```

右转

```
void turn_right() {
  mcpwm_set_duty(MCPWM_UNIT_1, MCPWM_TIMER_1, MCPWM_OPR_A, servo_duty_cycle_center +
servo_duty_cycle_differ);
}
```

直行

```
void straight() {
  mcpwm_set_duty(MCPWM_UNIT_1, MCPWM_TIMER_1, MCPWM_OPR_A, servo_duty_cycle_center);
}
```