Arduino智能小车驱动开发说明文档

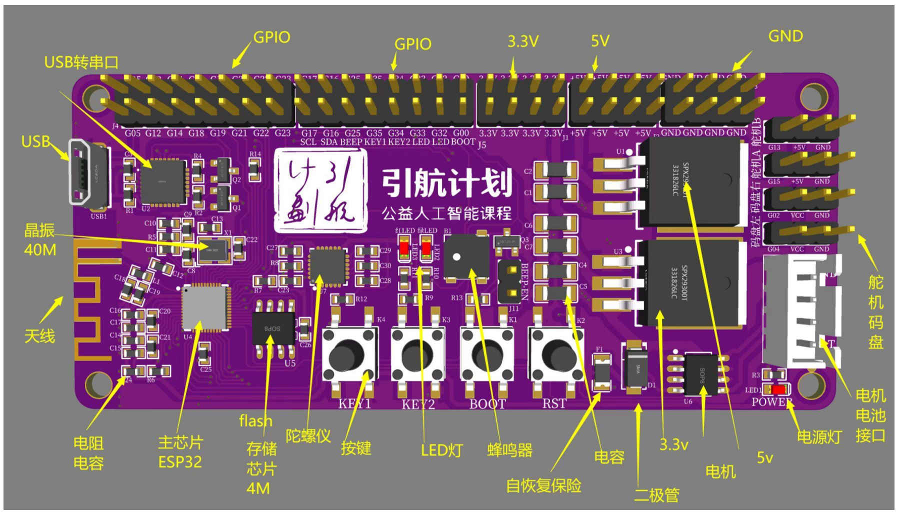
本说明文档主要负责说明在小车上使用arduino完成一些驱动开发，能够使小车完成以下内容：

1.小车控制板可以连接WiFi热点并发出信号

2.实现键盘控制小车自动行走

3.通过获取小车摄像头照片传入电脑

首先展示小车主板如下：

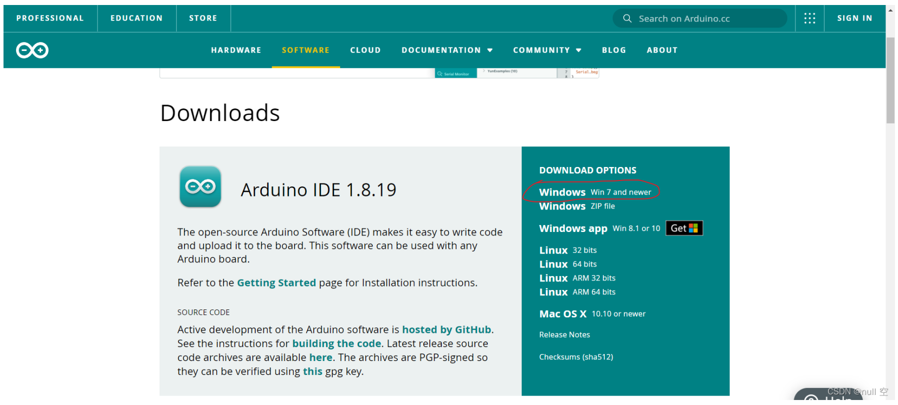


1. USB接口：通过数据线与电脑相连
2. USB转串口模块：将USB信号转换成串口可接收的形式
3. GPIO管脚：负责连接的管脚，上下一致
4. 3.3V管脚：接3.3V电
5. 5V管脚：接5V电
6. GND管脚：接地
7. 舵机码盘：用来连接舵机
8. LED灯：分别由33号管脚和32号管脚控制
9. 按键：最右侧为复位键
10. 蜂鸣器:连接两个端口即可使用

一、配置arduino环境

进入arduino官网下载arduino安装包：

安装Arduino后，为了让Arduino支持编译、上传ESP32的程序，还要安装开发板



Downloads

Arduino IDE 1.8.19

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the **Getting Started** page for Installation instructions.

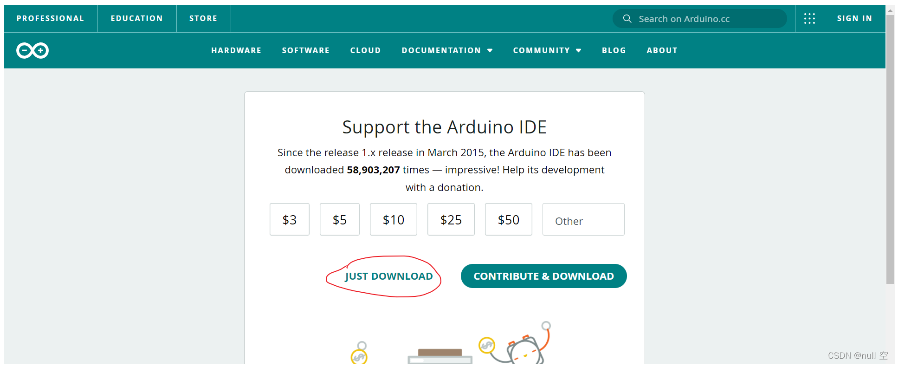
SOURCE CODE

Active development of the Arduino software is **hosted by GitHub**. See the instructions for **building the code**. Latest release source code archives are available **here**. The archives are PGP-signed so they can be verified using **this** gpg key.

DOWNLOAD OPTIONS

Windows Win 7 and newer
Windows ZIP file
Windows app Win 8.1 or 10 Get
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits
Mac OS X 10.10 or newer
Release Notes
Checksums (sha512)

根据系统选择对应版本下载安装：



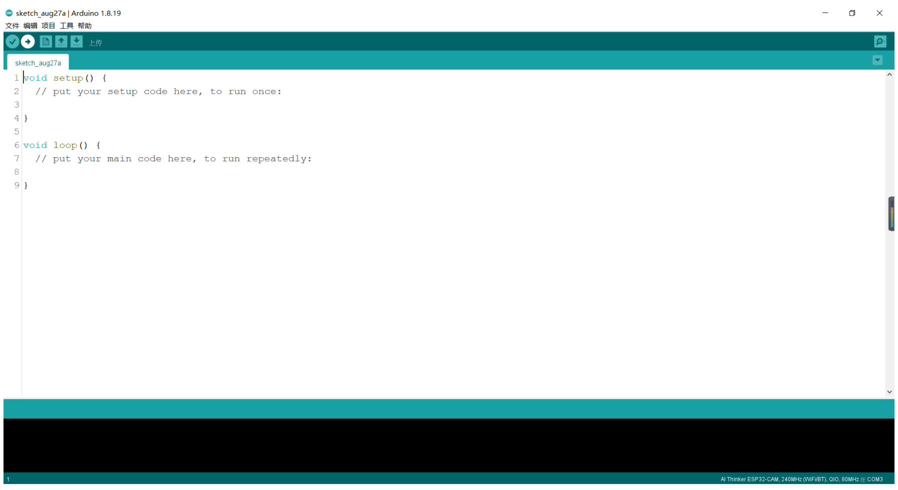
Support the Arduino IDE

Since the release 1.x release in March 2015, the Arduino IDE has been downloaded **58,903,207** times — impressive! Help its development with a donation.

$3 $5 $10 $25 $50 Other

JUST DOWNLOAD　CONTRIBUTE & DOWNLOAD

下载完成后根据提示安装即可，安装完成后页面如下：



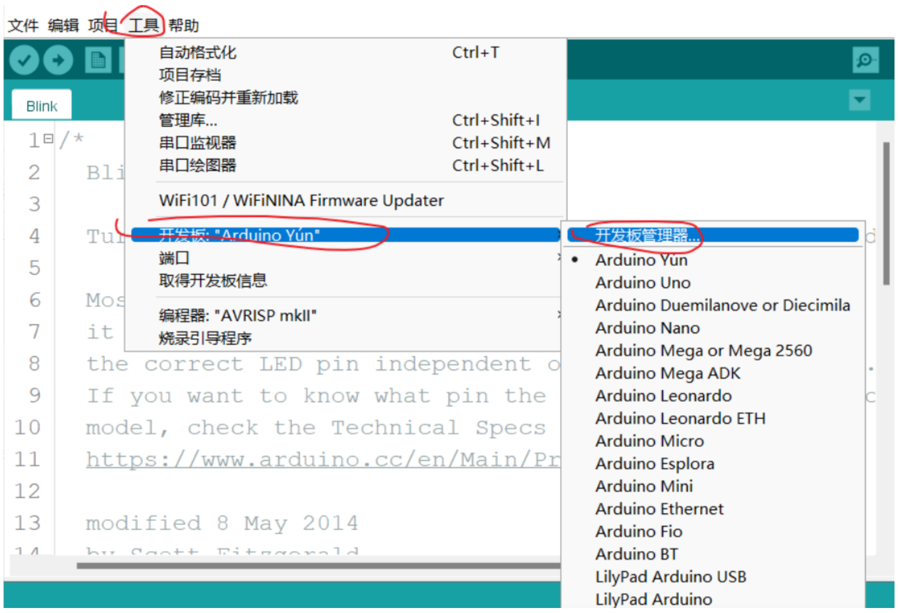
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Al Thinker ESP32-CAM, 240MHz (WiFi/BT), QIO, 80MHz 在 COM3

二、安装ESP开发板

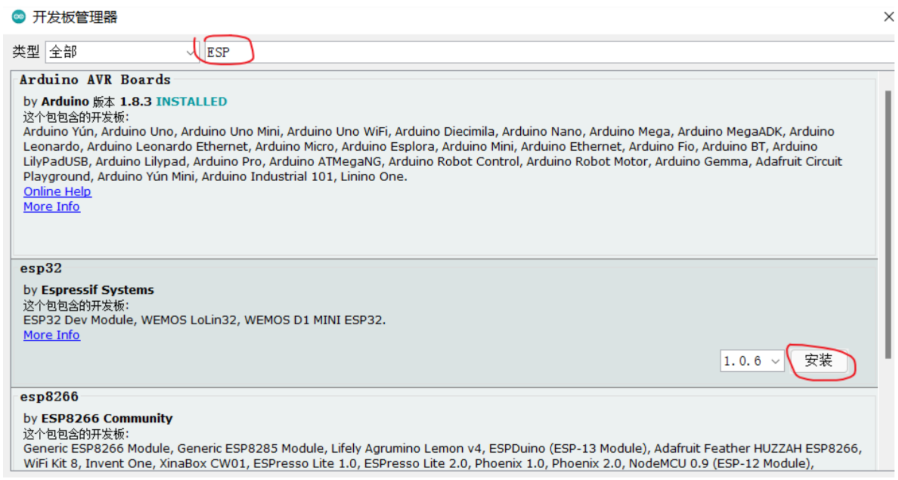
安装Arduino后，为了让Arduino支持编译、上传ESP32的程序，还要安装开发板

点击 工具—开发板—开发板管理器



```
/*
  Bli

  Tu

  Mos
  it
  the correct LED pin independent o
  If you want to know what pin the
  model, check the Technical Specs
  https://www.arduino.cc/en/Main/Pr

  modified 8 May 2014
```

搜索ESP，点击安装esp32，等待下载安装完毕



开发板管理器

类型 全部 ESP

Arduino AVR Boards
by **Arduino** 版本 **1.8.3** INSTALLED
这个包包含的开发板:
Arduino Yún, Arduino Uno, Arduino Uno Mini, Arduino Uno WiFi, Arduino Diecimila, Arduino Nano, Arduino Mega, Arduino MegaADK, Arduino Leonardo, Arduino Leonardo Ethernet, Arduino Micro, Arduino Esplora, Arduino Mini, Arduino Ethernet, Arduino Fio, Arduino BT, Arduino LilyPadUSB, Arduino Lilypad, Arduino Pro, Arduino ATMegaNG, Arduino Robot Control, Arduino Robot Motor, Arduino Gemma, Adafruit Circuit Playground, Arduino Yún Mini, Arduino Industrial 101, Linino One.
Online Help
More Info

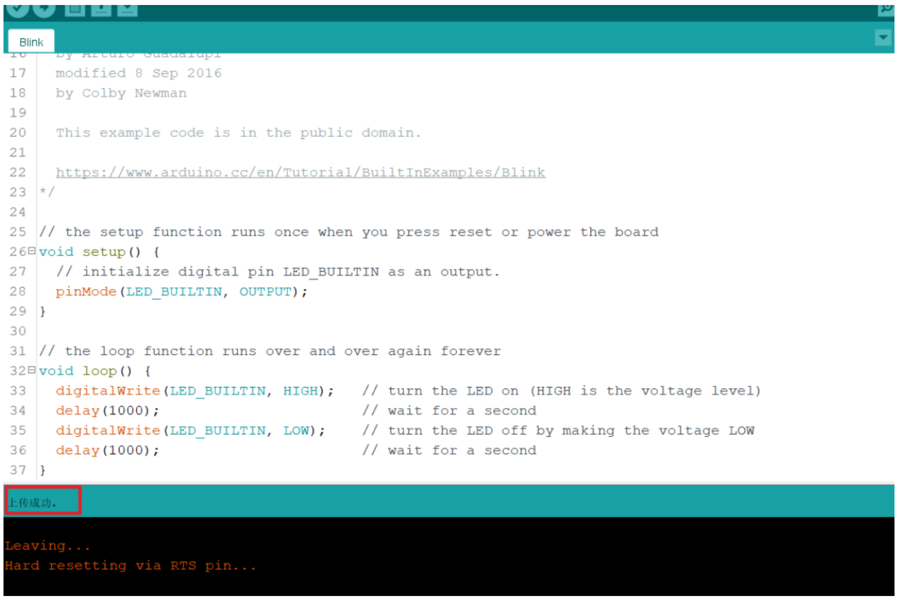
esp32
by **Espressif Systems**
这个包包含的开发板:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info
1.0.6 安装

esp8266
by **ESP8266 Community**
这个包包含的开发板:
Generic ESP8266 Module, Generic ESP8285 Module, Lifely Agrumino Lemon v4, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, WiFi Kit 8, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module),

之后就可在开发板中选择ESP32开发板了

先用USB数据线连接电脑和ESP32开发板，如果一切正常，点击 工具—端口 会出现一个新的COM串行端口，选择它作为Arduino与ESP32开发板传输数据的通道

运行程序时，当看到显示“上传成功”即代表程序已经烧录到开发板上。



```
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

上传成功。

```
Leaving...
Hard resetting via RTS pin...
```

如果开发板想向电脑输出一些数据则先在`setup()`中添加`Serial.begin(9600)`，9600为波特率(即每秒钟传输9600bit的数据)，可设置其他值，开发板和电脑上的波特率要一致才能正常通信。

打开串口监视器，设置波特率和开发板一样：115200。

三、电机

电机连接电池，可以直接供电

更改频率、分辨率，配置ledc通道、调节占空比来控制电机

```
const int a=26;
const int b=27;
//pwm
const int freq=2000;//频率
const int resolution=8;//分辨率
const int channel=0;//通道
const int duty_cycle=255;//占空比

void setup() {
  ledcSetup(channel,freq,resolution);//配置ledc通道
  ledcAttachPin(b,channel);//将PIN管脚与通道相连接
  //将a号管脚值定位low
  pinMode(a,OUTPUT);
  digitalWrite(a,LOW);

}

void loop() {
  ledcWrite(channel,duty_cycle);//将主程序写入
}
```

四、mcpwm单元

两个定时器A和B计算占空比

```
#include  "driver/mcpwm.h"
void setup() {
  // 用选定的MCPWM_UNIT_0来初始化gpio口
  mcpwm_gpio_init(MCPWM_UNIT_0,MCPWM0A,26);
  mcpwm_gpio_init(MCPWM_UNIT_0,MCPWM0A,27);

  //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,
    .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PWMOA和PWMOB
  mcpwm_init(MCPWM_UNIT_0,MCPWM_TIMER_0,&motor_pwm_config);
}

void loop() {
  // pwm
  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,100)
;
  mcpwm_start(MCPWM_UNIT_0,MCPWM_TIMER_0);
  delay(5000);
  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,0);
  mcpwm_start(MCPWM_UNIT_0,MCPWM_TIMER_0);
  delay(5000);
  mcpwm_stop(MCPWM_UNIT_0,MCPWM_TIMER_0);
}
```

五、pwm舵机、蜂鸣器

舵机连接开发板上的舵机管脚

蜂鸣器：直接将开发板蜂鸣器两个端口相连

```
//舵机初始化
const int a=15;

//pwm
const int f=50;
const int r=8;
const int c=0;
const int d=20;//77-32

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

}

void loop() {
  ledcWrite(c,d);



}
```

```
const int buzzer=25;//蜂鸣器

//pwm
const int f=2000;
const int c=0;
const int r=8;
const int d=128;

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

}

void loop() {
  //固定频率（音调）--修改占空比（响度）
  ledcWriteTone(c,f);

  ledcWrite(c,d);
  delay(100);

}
```

六、小车行走

```
const int servo=15;//13 15--舵机
const int motorA=26;//电机A
const int motorB=27;//电机B
//pwm
const int freq=50;//频率
const int resolution=8;//分辨率
//通道
const int servo_channel=0;//舵机
const int motorA_channel=1;//电机A
const int motorB_channel=2;//电机B

void setup() {
 // 舵机
 ledcSetup(servo_channel,freq,resolution);
 ledcAttachPin(servo,servo_channel);
 //电机A
 ledcSetup(motorA_channel,freq,resolution);
 ledcAttachPin(motorA,motorA_channel);
 //电机
 ledcSetup(motorB_channel,freq,resolution);
 ledcAttachPin(motorB,motorB_channel);

 /*初始化*/
 ledcWrite(servo_channel,20);//舵机初始化-->90度-->20,取值范围7--32
 ledcWrite(motorA_channel,0);//A前进
 ledcWrite(motorB_channel,0);//B后退

}

void loop() {
 while (1){
  ledcWrite(motorA_channel,128);//0--255,速度从小到大
  delay(3000);
  ledcWrite(servo_channel,7);//向左转行驶
  delay(2000);//延时2S
  ledcWrite(motorA_channel,128);//前进
  delay(4000);//延时4S
  ledcWrite(servo_channel,20);//回正直行
  delay(2000);//延时2S
  ledcWrite(motorA_channel,128);
  delay(4000);
  ledcWrite(servo_channel,32);//向右转行驶
  delay(2000);
  ledcWrite(motorA_channel,128);
  delay(4000);

  ledcWrite(servo_channel,20);//回正直行
```

```
    delay(2000);
  }

}
```

七、连接WiFi并键盘控制小车行走

将使用 ESP32-CAM 通过 Wi-Fi 进行控制。我们将创建一个基于Web的界面来控制小车，可以在本地网络内的任何设备中访问。

```
/*********
  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";//开发板发出的信号名称
const char* password = "1812003xyz";//开发板发出的密码

// 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);
}
```

通过此程序生成一个界面，通过这个界面来控制小车的前进后退、左转右转。以及摄像头拍照功能。

步骤：

连接开发板，确保端口为开发板连接的端口，将程序写入开发板。

电脑网络中出现minicar11信号，连接

打开串口监视器，重启开发板，出现网页地址192.168.4.1

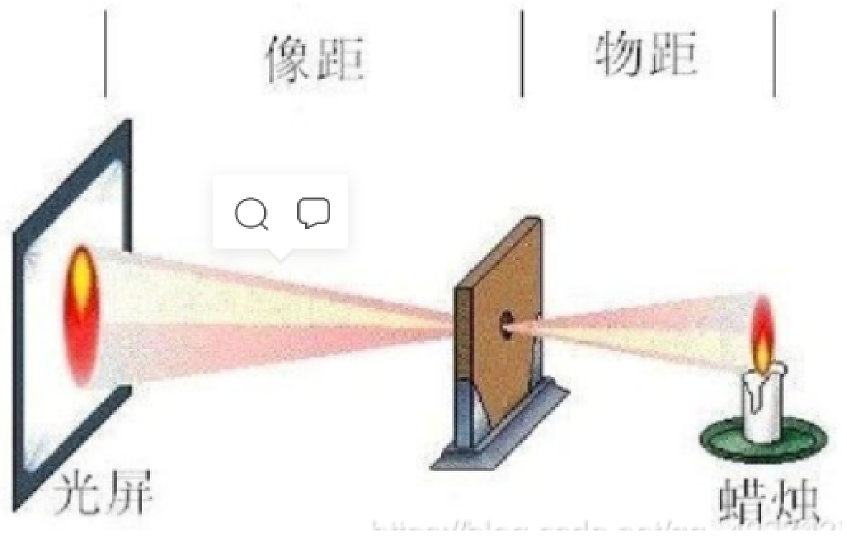
将数据线拔掉，使用电池供电，与小车连接

进入该网站，遥控小车行驶

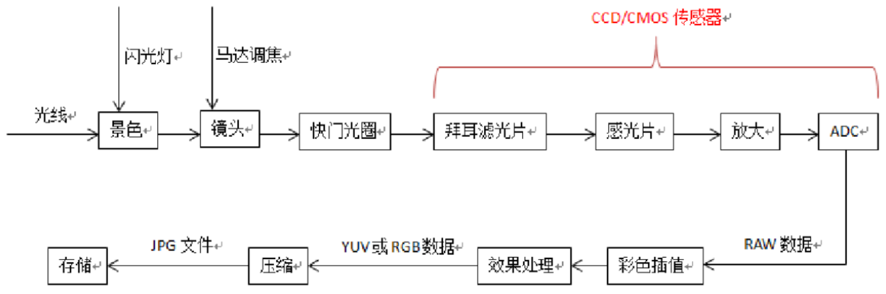
八、摄像头的原理和使用

一、摄像头成像、组成原理

摄像头的设计与人的眼睛成像原理一致



camera的成像框架：



景物通过凸镜头反射聚焦到感光片（CCD/CMOS：图像传感器）中，感光片产生电荷传导自ADC进行数模信号转换，形成RAW颜色数据，经过ISP图像处理算法（变成RGB,YUV常见的像素颜色格式）储存到相应的储存其中，在让CPU读取显示到相应的显示设备中

二、摄像头基本结构

摄像头结构组成：
Lens:镜头,负责成像和对焦
Holder:基座,负责固定镜头
IR:红外滤波片,负责过滤红外光（滤除人不可见的波）
Sensor:图像传感器,负责将图像转换电信号
PCB:印刷电路板,负责供电控制及信号传输
FPC:可绕性印刷电路板,负责接口
马达:用来改变像距

三、摄像头几个核心

图像传感器CCD,CMOS

CCD就像传统相机的底片一样的感光系统，是感应光线的电路装置，你可以将它想象成一颗颗微小的感应粒子，铺满在光学镜头后方，当光线与图像从镜头透过、投射到CCD表面产生电流，将感应到的内容转换成数码资料储存起来。
CCD像素数目越多、单一像素尺寸越大，收集到的图像就会越清晰
CCD:好用，图像更清晰，公益复杂，贵
CMOS:便宜，简单，图像质量较差

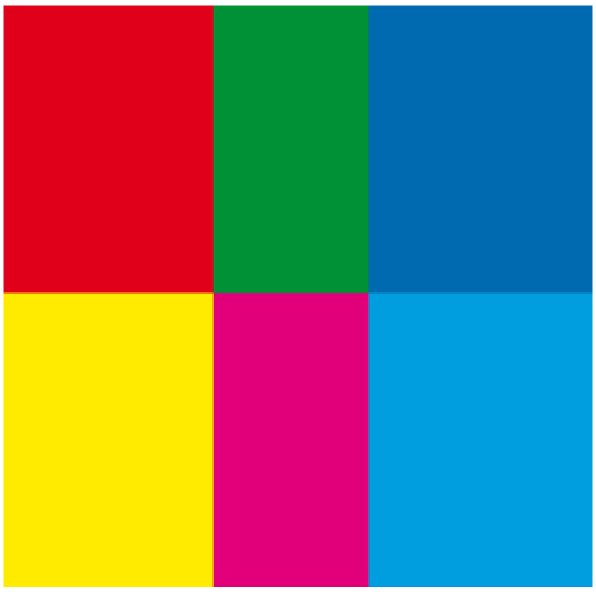
电荷耦合器件（CCD）原理简单。我们可以把它想象成一个没有盖子的记忆芯片。撞击记忆单元的光子在这些单元中产生电子（光电效应），因此光子的数目与电子的数目互成比例（光的明暗）。然而光子的波长（颜色）并没有被转换为电子。换言之，CCD 裸芯片实际上没有把色彩信息转换为任何形式的电信号。拍摄出来的照片是黑白的！（那为啥能得到彩色数据？）

大多数相机的方案：拜尔滤光片（马赛克滤波片）+单CCD+算法插值（ISP运算）。 拜尔滤光片使每个像素只能产生红、绿或蓝三色当中一种颜色的值。但是在输出时，由相机处理单元执行空间色彩插值法，使每个像素均包含三基色的成分。

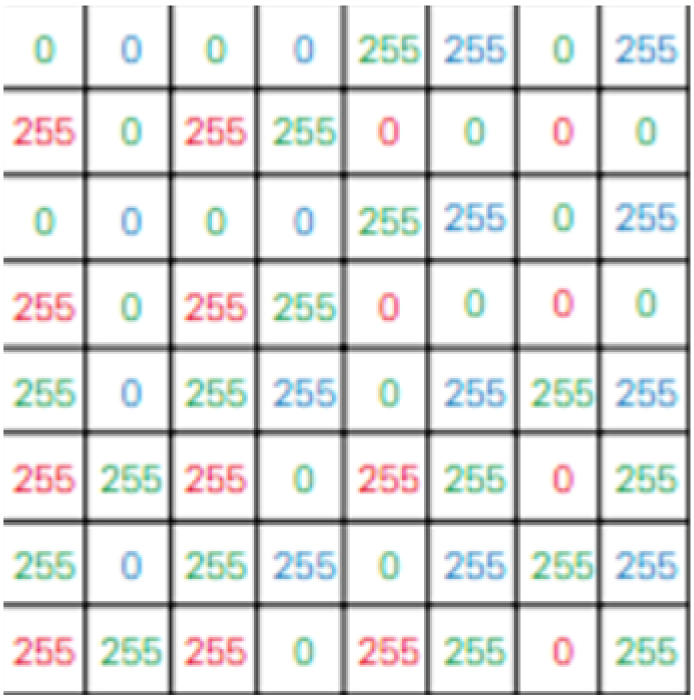
拜尔滤光片原理

拜尔滤光片使每个像素只能产生红、绿或蓝三色当中一种颜色的值

eg:



通过拜尔滤光片的值为

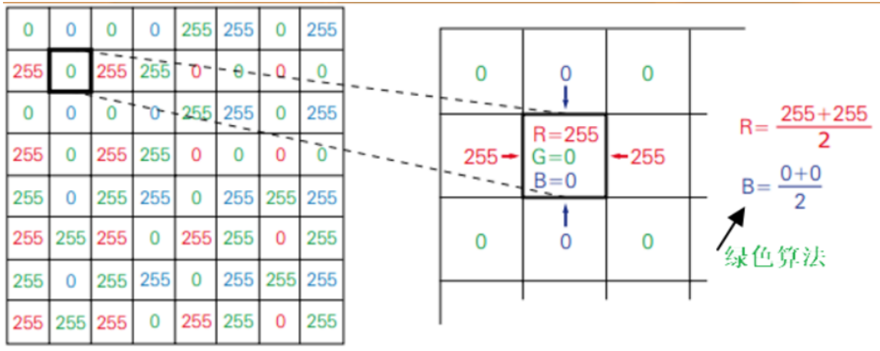


0	0	0	0	255	255	0	255
255	0	255	255	0	0	0	0
0	0	0	0	255	255	0	255
255	0	255	255	0	0	0	0
255	0	255	255	0	255	255	255
255	255	255	0	255	255	0	255
255	0	255	255	0	255	255	255
255	255	255	0	255	255	0	255

数字原始图像

由于人对红色光不敏感，对绿色光敏感，所以拜尔滤光片的色彩比为：红：绿：蓝 = 1：2：1
所以为了得到真实色彩值，红和蓝的算法一样，和绿不一样。

ISP图像处理算法
原始的数据图像经过ISP颜色差算法，算出正确的颜色数据（RGB为一个字，所以ISP中有数据格式的转变，
RAW->RGB,YUV）



数字原始图像

四、摄像头常见的功能模块

摄像头 图像传感器常见的的功能模块，包括：

●感光阵列(Image Array)（共有 656x488 个像素，其中在 YUV 的模式中，有效像素为 640x480 个）（cmos）
●模拟信号处理理（ Analog Processing）
●A/D 转换
●测试图案发生器器（ Test Pattern Generator）
●数字信号处理器(DSP)
●图像缩放（ Image Scaler）
●时序发生器（ Video Timing Generator）内部信号发生器和分布、帧率时序、自动曝光控制、输出外部时序（ VSYNC、HREF/HSYNC 和 PCLK）。
●数字视频端口（ Digital Video Port）
●SCCB 接口
●LED 和闪光灯输出控制
Note1: DSP(镜头校正、去噪声、黑白点补偿、自动白平衡等)

像素输出格式：
VGA，即分辨率为 640480 的输出模式；
**QVGA，即分辨率为 320240 的输出格式，
QQVGA，即分辨率为 160*120 的输出格式；

数据输出时序
PCLK，即像素时钟，一个 PCLK 时钟，输出一个像素(或半个像素,高字节+低字节)。
VSYNC，即帧同步信号。

（1）如何存储图像数据。
摄像头模块存储图像数据的过程为：等待同步信号→FIFO 写指针复位→FIFO 写使能→等待第二个 OV7670 同步信号→FIFO 写禁止。通过以上 5 个步骤，我们就完成了 1 帧图像数据的存储。
（2）如何读取图像数据。
在存储完一帧图像以后，我们就可以开始读取图像数据了。读取过程为：
FIFO 读指针复位→给 FIFO 读时钟（ FIFO_RCLK）→读取第一个像素高字节

→给 FIFO 读时钟→读取第一个像素低字节→给 FIFO 读时钟→读取第二个像素高字节→循环读取剩余像素→结束。

对外引脚描述：

usb：与数据线相连

sd卡插口：插入sd卡

摄像头5V连接开发板5V

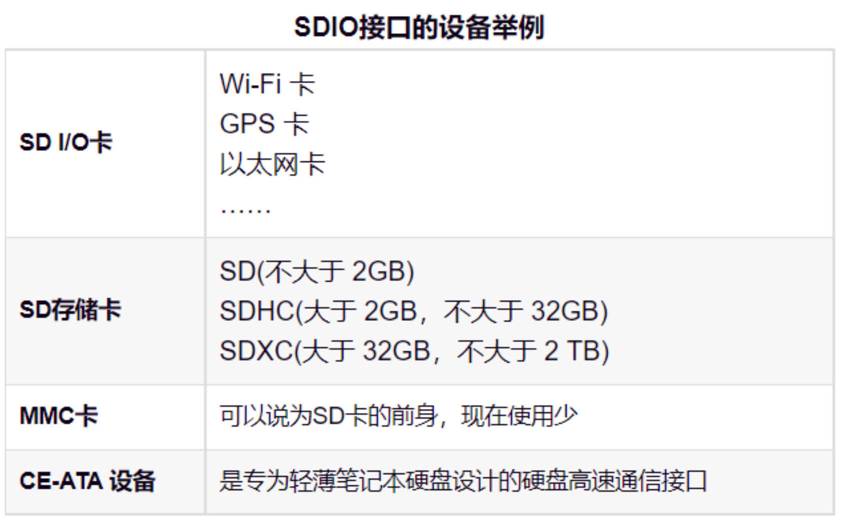
摄像头GND连接开发板GND

五、sd卡介绍

SD卡(Secure Digital Memory Card)

SDIO全称是安全数字输入/输出接口。多媒体卡(MMC)、SD卡、SD I/O卡都有SDIO接口。

SD I/O 卡本身不是用于存储的卡，它是指利用SDIO传输协议的外设。


SDIO接口的设备举例

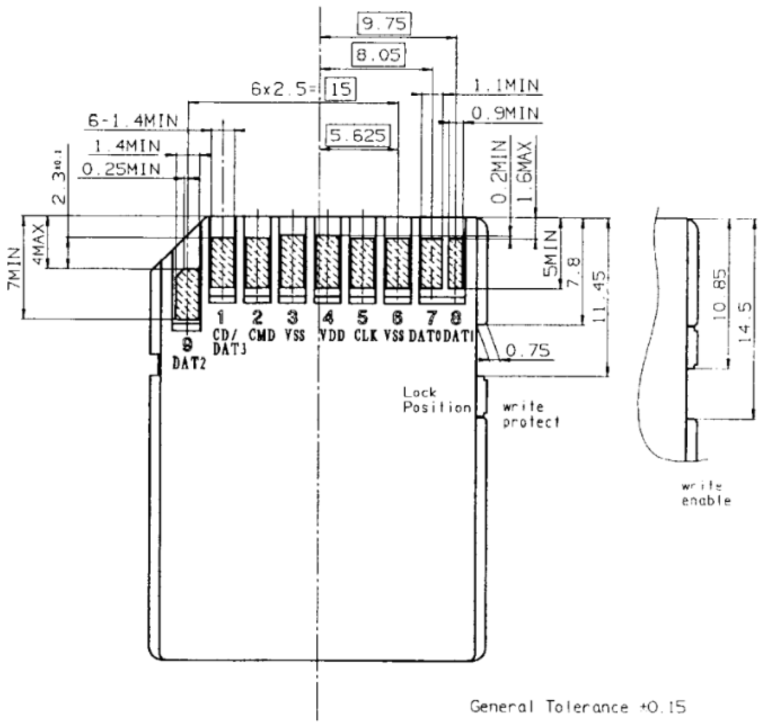
SD I/O卡	Wi-Fi 卡 GPS 卡 以太网卡 ……

SD存储卡	SD(不大于 2GB)
	SDHC(大于 2GB，不大于 32GB)
	SDXC(大于 32GB，不大于 2 TB)

MMC卡	可以说为SD卡的前身，现在使用少
CE-ATA 设备	是专为轻薄笔记本硬盘设计的硬盘高速通信接口

SD卡图片及其接口定义
SD卡（Secure Digital Memory Card）即：安全数码卡，它是在MMC的基础上发展而来，是一种基于半导体快闪记忆器的新一代记忆设备，它被广泛地于便携式装置上使用，例如数码相机、个人数码助理(PDA)和多媒体播放器等。SD卡由日本松下、东芝及美国SanDisk公司于1999年8月共同开发研制。SD卡图片及其接口定义

sd卡外部接口：



(图片来源： **SanDisk** Secure Digital (SD) Card Product Manual, Rev. 2.2)

SD卡由9个引脚进行通讯，支持SPI、SDIO两种模式，但是在两种模式下引脚定义略有不同：

SDIO和SPI模式下的SD卡引脚功能

引脚	1	2	3	4	5	6	7	8	9
SD卡模式	**CD/DAT3**	**CMD**	**VSS**	**VCC**	**CLK**	**VSS**	**DATO**	**DAT1**	**DAT2**
SPI模式	**CS**	**MOSI**	**VSS**	**VCC**	**CLK**	**VSS**	**MISO**	NC	NC

六、代码及操作步骤

将使用您的 ESP32-CAM 通过 Wi-Fi 进行控制。我们将创建一个基于Web的界面来控制小车摄像头，可以在本地网络内的任何设备中访问。

摄像头管脚设置：

将数据线拔掉之后，使用摄像头5V和GND管脚

摄像头5V连接开发板5V

摄像头GND连接开发板GND

使用电池给开发板供电，开发板给摄像头供电，将其连接在小车上

```
/*********
  https://techtutorialsx.com/2017/10/07/esp32-arduino-timer-interrupts/
  https://github.com/espressif/arduino-esp32/issues/1313
  https://github.com/espressif/arduino-
esp32/blob/master/cores/esp32/esp32-hal-timer.c
*********/


#include "WiFi.h"
#include "esp_camera.h"

#include "esp_timer.h"
```

```
#include "img_converters.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 <ESPAsyncWebServer.h>
#include <StringArray.h>
#include "FS.h"               // SD Card ESP32
#include "SD_MMC.h"           // SD Card ESP32
#include "time.h"
#include <WiFiUdp.h>
#include "driver/timer.h"

// Replace with your network credentials
const char* ssid = "minicar11";
const char* password = "1812003xyz";
// Set your Static IP address
IPAddress local_IP(192, 168, 4, 5);
// Set your Gateway IP address
IPAddress gateway(192, 168, 4, 1);

IPAddress subnet(255, 255, 0, 0);
IPAddress primaryDNS(8, 8, 8, 8);   //optional
IPAddress secondaryDNS(8, 8, 4, 4); //optional

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

boolean takeNewPhoto = false;
String lastPhoto = "";
String list = "";
hw_timer_t * timer = NULL;
hw_timer_t * timer_1 = NULL;
boolean isRecording = false;

// HTTP GET parameter
const char* PARAM_INPUT_1 = "photo";
const char* PARAM_INPUT_2 = "record_time";
const char* PARAM_INPUT_3 = "record_interval";

// OV2640 camera module pins (CAMERA_MODEL_AI_THINKER)
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27


#define Y9_GPIO_NUM       35
```

```
#define Y8_GPIO_NUM      34
#define Y7_GPIO_NUM      39
#define Y6_GPIO_NUM      36
#define Y5_GPIO_NUM      21
#define Y4_GPIO_NUM      19
#define Y3_GPIO_NUM      18
#define Y2_GPIO_NUM       5
#define VSYNC_GPIO_NUM   25
#define HREF_GPIO_NUM    23
#define PCLK_GPIO_NUM    22

// Stores the camera configuration parameters
camera_config_t config;

File root;

void setup() {
  // Turn-off the brownout detector
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);

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

  // Configures static IP address
  if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {
    Serial.println("STA Failed to configure");
  }
  // Wi-Fi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.print("Camera Stream Ready! Go to: http://");
  Serial.println(WiFi.localIP());

  Serial.println("Initializing the camera module...");
  configInitCamera();

  Serial.println("Initializing the MicroSD card module... ");
  initMicroSDCard();

  server.on("/capture", HTTP_GET, [](AsyncWebServerRequest * request) {
    if (takeNewPhoto) {
      request->send_P(200, "text/plain", "");

    } else {
```

```
    camera_fb_t * frame = esp_camera_fb_get();
    request->send_P(200, "image/jpeg", (const uint8_t *)frame->buf, frame-
>len);
    esp_camera_fb_return(frame);
    frame = NULL;
  }
 });

 server.on("/list", HTTP_GET, [](AsyncWebServerRequest * request) {
   listDirectory(SD_MMC);
   request->send_P(200, "text/html", list.c_str());
 });

 server.on("/view", HTTP_GET, [](AsyncWebServerRequest * request) {
   String inputMessage;
   String inputParam;
   // GET input1 value on <ESP_IP>/view?photo=<inputMessage>
   if (request->hasParam(PARAM_INPUT_1)) {
     inputMessage = "/" + request->getParam(PARAM_INPUT_1)->value();
     Serial.print("Trying to open ");
     Serial.println(inputMessage);
     inputParam = PARAM_INPUT_1;
   }
   else {
     inputMessage = "No message sent";
     inputParam = "none";
   }
   Serial.println(inputMessage);
   request->send(SD_MMC, inputMessage, "image/jpg", false);
 });

 // Send a GET request to <ESP_IP>/delete?photo=<inputMessage>
 server.on("/delete", HTTP_GET, [] (AsyncWebServerRequest *request) {
   String inputMessage;
   String inputParam;
   // GET input1 value on <ESP_IP>/delete?photo=<inputMessage>
   if (request->hasParam(PARAM_INPUT_1)) {
     inputMessage = "/" + request->getParam(PARAM_INPUT_1)->value();
     inputParam = PARAM_INPUT_1;
   }
   else {
     inputMessage = "No message sent";
     inputParam = "none";
   }
   Serial.println(inputMessage);
   deleteFile(SD_MMC, inputMessage.c_str());


   request->send(200, "text/html", "Done. Your photo named " +
inputMessage + " was removed." +
                                  "<br><a href=\"/list\">view/delete other photos</a>.");
 });

 server.on("/record", HTTP_GET, [](AsyncWebServerRequest * request) {
   String inputMessage_2;
   String inputMessage_3;
   // GET input1 value on <ESP_IP>/view?photo=<inputMessage>
   if (request->hasParam(PARAM_INPUT_2)) {
     inputMessage_2 = request->getParam(PARAM_INPUT_2)->value();
     if (request->hasParam(PARAM_INPUT_3)) {
       Serial.println("Start record... Can't get ip camera stream now.");
       inputMessage_3 = request->getParam(PARAM_INPUT_3)->value();
       take_save_record(inputMessage_2.toInt(), inputMessage_3.toInt());
     }
     else {
       inputMessage_3 = "Request on /record lacks Param 1!";
       Serial.println(inputMessage_3);
     }
   }
   else {
     inputMessage_2 = "Request on /record lacks Param 2!";
     Serial.println(inputMessage_2);
   }
 });

 server.on("/stop_record", HTTP_GET, [](AsyncWebServerRequest * request) {
   Serial.println("Stop record. You can get ip camera stream now.");
   stop_record();
 });

 // Start server
 server.begin();

 root = SD_MMC.open("/");
 listDirectory(SD_MMC);
}

void loop() {
 if (takeNewPhoto) {
   takeSavePhoto();
   takeNewPhoto = false;
 }
 delay(1);
}

void configInitCamera(){

 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; //
YUV422,GRAYSCALE,RGB565,JPEG

 config.frame_size = FRAMESIZE_SVGA; // FRAMESIZE_ +
QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA
 config.jpeg_quality = 8; //0-63 lower number means higher quality
 config.fb_count = 2;

//  // Select lower framesize if the camera doesn't support PSRAM
//  if(psramFound()){
//    config.frame_size = FRAMESIZE_UXGA; // FRAMESIZE_ +
QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA
//    config.jpeg_quality = 10; //0-63 lower number means higher quality
//    config.fb_count = 2;
//  }
//  else {
//    config.frame_size = FRAMESIZE_SVGA;
//    config.jpeg_quality = 12;
//    config.fb_count = 1;
//  }

 // Initialize the Camera
 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();
 s->set_brightness(s, 0);     // -2 to 2

 s->set_contrast(s, 2);       // -2 to 2
 s->set_saturation(s, 0);     // -2 to 2
}

void initMicroSDCard(){
 // Start Micro SD card
 Serial.println("Starting SD Card");
 if(!SD_MMC.begin()){
   Serial.println("SD Card Mount Failed");
   return;
 }
 uint8_t cardType = SD_MMC.cardType();
 if(cardType == CARD_NONE){
   Serial.println("No SD Card attached");
   return;
 }
}

void takeSavePhoto(){
 struct tm timeinfo;
 char now[20];
 // Take Picture with Camera
 camera_fb_t  * fb = esp_camera_fb_get();
 if(!fb) {
   Serial.println("Camera capture failed");
   return;
 }
 // Path where new picture will be saved in SD Card
 getLocalTime(&timeinfo);
 strftime(now, 20, "%Y%m%d_%H%M%S", &timeinfo); // Format Date &
Time
 String path = "/photo_" + String(now) +".jpg";
 lastPhoto = path;
 Serial.printf("Picture file name: %s\n", path.c_str());
 // Save picture to microSD card
 fs::FS &fs = SD_MMC;
 File file = fs.open(path.c_str(),FILE_WRITE);
 if(!file){
   Serial.printf("Failed to open file in writing mode");
 }
 else {
   file.write(fb->buf, fb->len); // payload (image), payload length
   Serial.printf(" Saved: %s\n", path.c_str());
 }
 file.close();
 esp_camera_fb_return(fb);
}


void IRAM_ATTR onTimer(){
 takeNewPhoto = true;
}

void IRAM_ATTR onTimer1() {
 Serial.println("Recording is complete. You can get ip camera stream now.");
 if (isRecording) {
   isRecording = false;
   //timer_disable_intr(TIMER_GROUP_0, TIMER_0);
   if (timer != NULL) {
     timerAlarmDisable(timer);
     timerDetachInterrupt(timer);
     timerEnd(timer);
     timer = NULL;
   }
   if (timer_1 != NULL) {
     timerAlarmDisable(timer_1);
     timerDetachInterrupt(timer_1);
     timerEnd(timer_1);
     timer_1 = NULL;
   }
 }
}

void take_save_record(long duration, long interval) {
 Serial.print("Duration: ");
 Serial.print(duration);
 Serial.print(" minute, Interval: ");
 Serial.print(interval);
 Serial.println(" s.");
 isRecording = true;
 if (timer == NULL) {
   timer = timerBegin(0, 40, true);
   // Attach onTimer function to our timer
   timerAttachInterrupt(timer, &onTimer, true);
 }
 /* Set alarm to call onTimer function every second 1 tick is 1us
 => 1 second is 1000000us
 Repeat the alarm (third parameter) */
 timerAlarmWrite(timer, interval * 1000000, true);
 /* Start an alarm */
 yield();
 timerAlarmEnable(timer);

 timer_1 = timerBegin(1, 80, true);
 timerAttachInterrupt(timer_1, &onTimer1, true);
 timerAlarmWrite(timer_1, duration * 60 * 1000000, false);
 yield();

 timerAlarmEnable(timer_1);
```

```
}

void stop_record() {
  if (isRecording) {
    isRecording = false;
    //timer_disable_intr(TIMER_GROUP_0, TIMER_0);
    if (timer != NULL) {
      timerAlarmDisable(timer);
      timerDetachInterrupt(timer);
      timerEnd(timer);
      timer = NULL;
    }
    if (timer_1 != NULL) {
      timerAlarmDisable(timer_1);
      timerDetachInterrupt(timer_1);
      timerEnd(timer_1);
      timer_1 = NULL;
    }
  }
}

void listDirectory(fs::FS &fs) {
  File root = fs.open("/");
  list = "";
  if(!root){
    Serial.println("Failed to open directory");
    return;
  }
  if(!root.isDirectory()){
    Serial.println("Not a directory");
    return;
  }

  File file = root.openNextFile();
  while(file){
    if(!file.isDirectory()){
      String filename=String(file.name());
      filename.toLowerCase();
      if (filename.indexOf(".jpg")!=-1){
        list = "<tr><td><button onclick=\"window.open('/view?
photo="+String(file.name())+"','_blank')\">View</button></td><td><button
onclick=\"window.location.href='/delete?
photo="+String(file.name())+"'\">Delete</button></td>
<td>"+String(file.name())+"</td><td></td></tr>"+list;
      }
    }
    lastPhoto = file.name();

    file = root.openNextFile();
  }

  if (list=="") {
    list="<tr>No photos Stored</tr>";
  }
  else {
    list="<h1>ESP32-CAM View and Delete Photos</h1><table><th
colspan=\"2\">Actions</th><th>Filename</th>"+list+"</table>";
  }
}

void deleteFile(fs::FS &fs, const char * path){
  Serial.printf("Deleting file: %s\n", path);
  if(fs.remove(path)){
    Serial.println("File deleted");
    listDirectory(SD_MMC);
  }
  else {
    Serial.println("Delete failed");
  }
}
```

步骤:

使用数据线连接摄像头，

将程序写入摄像头中（此时sd卡不能连接摄像头）

打开串口监视器，重启摄像头，出现网址192.168.4.5

将sd卡安回摄像头，重启摄像头，

连接开发板发出信号的名称和密码

打开网址192.168.4.5/list查看拍摄到的图片