Parse Data Control Servo

Parse the data in the specified data format through the serial port, and use the data to control the rotation angle of the servo: Here we demonstrate the serial port control of the servo of the S1 interface. The servos of other interfaces need to modify the code by themselves.

> The format of the serial port parsed data refers to the communication protocol of the WiFi camera module. The serial port control tutorial below also refers to the communication protocol of the WiFi camera module!
>
> Do not install the WiFi camera module here, otherwise it will occupy the serial port and abnormal information will appear.

Device connection

Hardware connection

Use Type-B data cable to connect Arduino Uno and computer.

Software connection

Open the "Arduino IDE" software and select the model and serial port number corresponding to the development board.

Serial port parsing data

Control principle

Receive data through the serial port and determine whether the data is in the specified format, parse and store the specified data, and use the stored data to control the servo to rotate the specified angle.

Control pin

Use the serial port on the Arduino IDE to send and parse data.

Serial port	Arduino Uno
RX	0
TX	1

Data format (communication protocol)

We only parse data in two data formats.

Data format	Description	Example
$	Data header	
#	Data tail	
Data,Data,Data	Data type 1	Data represents only numbers: 1,0,0,0
Alphabet+Data	Data type 2	Alphabet+Data represents a letter plus the following three digits: A100

Example:

Data format 1: $1,0,0,0#, $1,2,0,0#, $1,2,0,1#

Data format 2: $A100#, $A090#, $B120#

Note:

- Data format 1 is mainly used to control the car status, RGB, buzzer, motor
- Data format 2 is mainly used to control the servo angle

The data needs to be sent strictly in the above format.

Control the servo

We use the three digits in data format 2 to control the servo rotation angle.

Data packet	Header	Servo number	Data: Control servo angle	Tail
Control servo rotation 45 degrees	$	A	045	#
Control servo rotation 90 degrees	$	A	090	#
Control servo rotation 135 degrees	$	A	135	#

Example:

Serial port controls the car to move forward, RGB displays blue, and buzzer sounds: $1,3,1,0#

Serial port controls the servo to rotate 45 degrees: $A045#

> The code can control the car, RGB lights, and buzzer status through the serial port, but this tutorial only introduces the angle of controlling the servo. By learning this tutorial, the functions of serial port control of peripherals have been basically demonstrated!

Note: Due to the limited rotation angle of the camera, the control servo rotation range is 35°~145°. It will not rotate beyond this range and maintain the boundary value.

Code Analysis

Here we only briefly introduce the code content. For detailed code, please refer to the corresponding code file, which is provided in the download area!

- Include `Wire`、`Adafruit_NeoPixel`、`Adafruit_PWMServoDriver` library

```
#include <Wire.h>                       // 包含Wire(I2C)通讯库 Include Wire library
#include <Adafruit_NeoPixel.h>          // 包含Adafruit NeoPixel库 Include Adafruit NeoPixel library
#include <Adafruit_PWMServoDriver.h>    // 包含Adafruit PWMServoDriver库 Include Adafruit PWMServoDriver library
```

- Define data such as arrays, flag bits, etc

```
// 定义数据，如数组，标志位等 Define data such as arrays, flag bits, etc
#define BUFFER_SIZE 20
volatile uint8_t Rx_Index = 0;
volatile uint8_t Data_Flag = 0;
char* Data_Array[BUFFER_SIZE];
char Data_Servo[BUFFER_SIZE];

int DataArraySize = 0;

// 定义电机控制引脚 Define motor control pins
#define Motor_L1_F_PIN 11  // 控制小车左前方电机前进 Control the motor on the left front of the car
#define Motor_L1_B_PIN 10  // 控制小车左前方电机后退 Control the motor back on the left front of the car
#define Motor_L2_F_PIN 8   // 控制小车左后方电机前进 Control car left rear motor forward
#define Motor_L2_B_PIN 9   // 控制小车左后方电机后退 Control the car left rear motor back
#define Motor_R1_F_PIN 13  // 控制小车右前方电机前进 Control the right front motor of the car to move forward
#define Motor_R1_B_PIN 12  // 控制小车右前方电机后退 Control the motor back on the right front of the car
#define Motor_R2_F_PIN 14  // 控制小车右后方电机前进 Control car right rear motor forward
#define Motor_R2_B_PIN 15  // 控制小车右后方电机后退 Control car right rear motor back

// 定义底层驱动芯片参数 Bottom-layer driver chip related parameters
#define Bottom_Layer_Driver_ADDR 0x40
```

```
// 定义PWM频率 Define PWM frequency
#define PWM_FREQUENCY 50

// 定义RGB控制引脚和数量 Define RGB control pins and quantity
#define RGB_PIN 6
#define RGB_NUM 1

// 定义蜂鸣器控制引脚 Define BUZZER control pins
#define BUZZER_PIN 10

// 定义电机速度和小车状态 Define motor speed and trolley status
unsigned int uMotorSpeed = 50;
unsigned int uCurCarState = 0;
unsigned int uPreCarState = 0;
unsigned int uCurRgbState = 0;
unsigned int uPreRgbState = 0;
unsigned int uCurBuzState = 0;
unsigned int uPreBuzState = 0;
```

- Enumerate the common movement modes of omnidirectional cars

```
// 枚举全向小车的常见运动方式 Enumerate the common movement modes of omnidirectional cars
enum OmniDirectionalCar {
  STOP,
  FORWARD,
  BACKWARD,
  LEFT,
  RIGHT,
  LEFT_ROTATE,
  RIGHT_ROTATE,
};
```

- Enumerate common colors

```
// 枚举常见颜色 Enumerate common colors
enum ColorType {
  BLACK,
  RED,
  GREEN,
  BLUE,
  YELLOW,
  MAGENTA,
  CYAN,
  WHITE,
};
```

- Enumerate buzzer state

```
// 枚举蜂鸣器状态 Enumerate buzzer state
enum BuzzerState {
  BuzzerOff,
  BuzzerOn,
};
```

- Create an instance of the `Adafruit_NeoPixel` `Adafruit_PWMServoDriver` class

```
// 创建Adafruit_PWMServoDriver类的实例 Create an instance of the Adafruit_PWMServoDriver class
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(Bottom_Layer_Driver_ADDR);
// 创建Adafruit_NeoPixel类的实例 Create an instance of the Adafruit_NeoPixel class
Adafruit_NeoPixel RGB = Adafruit_NeoPixel(RGB_NUM, RGB_PIN, NEO_GRB + NEO_KHZ800);
```

- Setting the Motor Speed

```
/**
 * @brief 设置单个电机速度 Setting the Motor Speed
 * @param motor_forward_pin: 控制电机前进引脚 Control the motor forward pin
 * @param motor_backward_pin: 控制电机后退引脚 Control the motor backward pin
 * @param motor_speed: 设置电机速度 Setting the Motor Speed
```

```
 * @retval 无 None
 */
void setMotorSpeed(uint16_t motor_forward_pin, uint16_t motor_backward_pin, int motor_speed) {
  motor_speed = map(motor_speed, -255, 255, -4095, 4095);
  if (motor_speed >= 0) {
    pwm.setPWM(motor_forward_pin, 0, motor_speed);
    pwm.setPWM(motor_backward_pin, 0, 0);
  } else if (motor_speed < 0) {
    pwm.setPWM(motor_forward_pin, 0, 0);
    pwm.setPWM(motor_backward_pin, 0, -(motor_speed));
  }
}
```

- Set the car movement mode and speed

```
/**
 * @brief 设置小车运动方式和速度 Set the car movement mode and speed
 * @param Movement: 小车运动方式 Car movement
 * @param Speed: 小车运动速度 Car speed
 * @retval 无 None
 */
void setCarMove(uint8_t Movement, int Speed) {
  switch (Movement) {
    case STOP:
      Serial.println("Car State:STOP");
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, 0);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, 0);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, 0);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, 0);
      break;
    case FORWARD:
      Serial.println("Car State:FORWARD");
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, Speed);
      break;
    case BACKWARD:
      Serial.println("Car State:BACKWARD");
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, -Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, -Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, -Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, -Speed);
      break;
    case LEFT:
      Serial.println("Car State:LEFT");
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, -Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, -Speed);
      break;
    case RIGHT:
      Serial.println("Car State:RIGHT");
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, -Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, -Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, Speed);
      break;
    case LEFT_ROTATE:
      Serial.println("Car State:LEFT_ROTATE");
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, -Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, -Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, Speed);
      break;
    case RIGHT_ROTATE:
      Serial.println("Car State:RIGHT_ROTATE");
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, -Speed);
```

```
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, -Speed);
      break;
    default:
      Serial.println("Car State:STOP");
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, 0);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, 0);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, 0);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, 0);
      break;
  }
}
```

- Set RGB display color

```
/**
 * @brief 设置RGB显示的颜色 Set RGB display color
 * @param color: 显示的颜色 Set the color
 * @retval 无 None
 */
void setRGBColor(ColorType color) {
  switch (color) {
    case BLACK:
      Serial.println("RGB State:BLACK");
      RGB.setPixelColor(0, RGB.Color(0, 0, 0));
      RGB.show();
      break;
    case RED:
      Serial.println("RGB State:RED");
      RGB.setPixelColor(0, RGB.Color(255, 0, 0));
      RGB.show();
      break;
    case GREEN:
      Serial.println("RGB State:GREEN");
      RGB.setPixelColor(0, RGB.Color(0, 255, 0));
      RGB.show();
      break;
    case BLUE:
      Serial.println("RGB State:BLUE");
      RGB.setPixelColor(0, RGB.Color(0, 0, 255));
      RGB.show();
      break;
    case YELLOW:
      Serial.println("RGB State:YELLOW");
      RGB.setPixelColor(0, RGB.Color(255, 255, 0));
      RGB.show();
      break;
    case MAGENTA:
      Serial.println("RGB State:MAGENTA");
      RGB.setPixelColor(0, RGB.Color(255, 0, 255));
      RGB.show();
      break;
    case CYAN:
      Serial.println("RGB State:CYAN");
      RGB.setPixelColor(0, RGB.Color(0, 255, 255));
      RGB.show();
      break;
    case WHITE:
      Serial.println("RGB State:WHITE");
      RGB.setPixelColor(0, RGB.Color(255, 255, 255));
      RGB.show();
      break;
    default:
      Serial.println("RGB State:BLACK");
      RGB.setPixelColor(0, RGB.Color(0, 0, 0));
      RGB.show();
      break;
  }
}
```

- Set Buzzer sound

```c
/**
 * @brief 设置蜂鸣器声音 Set Buzzer sound
 * @param State: 发声开关 Whether to make a sound
 * @retval 无 None
 */
void setBuzzerSound(BuzzerState State) {
  switch (State) {
    case BuzzerOn:
      Serial.println("Buzzer State:On");
      tone(BUZZER_PIN, 1000);
      delay(5);
      break;
    default:
      Serial.println("Buzzer State:Off");
      noTone(BUZZER_PIN);
      break;
  }
}
```

- Set the servo rotation angle

```c
/**
 * @brief 设置舵机旋转角度 Set the servo rotation angle
 * @param servoPin: 舵机控制引脚 Servo control pin
 * @param angle: 舵机旋转角度 Servo rotation angle
 * @retval 无 None
 */
void setServoAngle(uint8_t servoPin, int angle) {
  if (angle > 145) {
    angle = 145;
    Serial.println("Servo rotation limit, change the servo angle 145 degrees.");
  }
  if (angle < 35) {
    angle = 35;
    Serial.println("Servo rotation limit, change the servo angle 35 degrees.");
  }
  Serial.print("Servo Angle:");
  Serial.println(angle);
  long pulseWidth = map(angle, SERVO_MIN_ANGLE, SERVO_MAX_ANGLE, SERVO_MIN_PULSE_WIDTH,
SERVO_MAX_PULSE_WIDTH);
  long PWM_Value = pulseWidth * 4096 / SERVO_PULSE_WIDTH_CYCLE;
  pwm.setPWM(servoPin, 0, PWM_Value);
}
```

- Receive data

Note: Receive data between `$` and `#`

```c
/**
 * @brief 接收指定格式的数据 Receives data in the specified format
 * @param 无 None
 * @retval 接收指定数据 Received specified data
 */
char* recData() {
  static char Data_Buffer[BUFFER_SIZE];
  static char Rx_Buffer[BUFFER_SIZE];

  if (Serial.available() > 0) {
    char SerialData = Serial.read();
    if (SerialData == '$') {
      memset(Rx_Buffer, 0, sizeof(Rx_Buffer));
      memset(Data_Buffer, 0, sizeof(Data_Buffer));
      Data_Flag = 1;
      Rx_Index = 0;
    } else if (Data_Flag == 1 && SerialData == '#') {
      if (Rx_Index < BUFFER_SIZE) {
        memcpy(Data_Buffer, Rx_Buffer, Rx_Index);
        memset(Rx_Buffer, 0, sizeof(Rx_Buffer));
        Rx_Index = 0;
        Data_Flag = 0;
```

```
        // Serial.println(Data_Buffer);
        return Data_Buffer;
      } else {
        // Serial.println("Rx_Buffer is full!");
        memset(Rx_Buffer, 0, sizeof(Rx_Buffer));
        Rx_Index = 0;
        Data_Flag = 0;
      }
    } else {
      if (Rx_Index < BUFFER_SIZE) {
        Rx_Buffer[Rx_Index++] = SerialData;
      } else {
        // Serial.println("Rx_Buffer is full!");
        memset(Rx_Buffer, 0, sizeof(Rx_Buffer));
        Rx_Index = 0;
        Data_Flag = 0;
      }
    }
  }
  return nullptr;
}
```

- Separate the data into a new array

```
/**
 * @brief 将数据分离到一个新的数组中 Separate the data into a new array
 * @param dataBuffer: 要分离的数据 The data to be separated
 * @param dataArray: 分离后的数据 The data after separation
 * @param dataArraySize: 分离数据的大小 The size of the separated data
 * @retval 无 None
 */
void splitDataAndSave(char* dataBuffer, char** dataArray, int& dataArraySize) {
  char* token = strtok(dataBuffer, ",");
  int index = 0;
  while (token != NULL) {
    dataArray[index] = token;
    token = strtok(NULL, ",");
    index++;
  }
  dataArraySize = index;
}
```

- Parse the serial port data after processing

```
Dump format 2 data and print format 1/2 data
```

```
/**
 * @brief 处理后解析串口数据 Parse the serial port data after processing
 * @param dataBuffer: 分离后的数据 The data after separation
 * @retval 无 None
 */
void parseSerialData(char* dataBuffer) {
  if (dataBuffer != nullptr) {
    // Serial.println(dataBuffer);
    if (isalpha(dataBuffer[0])) {
      memcpy(Data_Servo, dataBuffer + 1, 3);
      uCurServoAngle = strtol(Data_Servo, NULL, 10);
      // Serial.print("Servo angle:");
      // Serial.println(strtol(Data_Servo, NULL, 10));
    } else {
      splitDataAndSave(dataBuffer, Data_Array, DataArraySize);
      // Serial.print("Car state:");
      // Serial.println(strtol(Data_Array[0], NULL, 10));
      uCurCarState = strtol(Data_Array[0], NULL, 10);  // Control the car status
      // Serial.print("RGB state:");
      // Serial.println(strtol(Data_Array[1], NULL, 10));
      uCurRgbState = strtol(Data_Array[1], NULL, 10);  // Control the RGB status
      // Serial.print("Buzzer state:");
      // Serial.println(strtol(Data_Array[2], NULL, 10));
```

```
        uCurBuzState = strtol(Data_Array[2], NULL, 10);  // Control the Buzzer status
        // Serial.print("No define:");
        // Serial.println(strtol(Data_Array[3], NULL, 10));
      }
    memset(Data_Servo, 0, sizeof(Data_Servo));
    memset(Data_Array, 0, sizeof(Data_Array));
  }
}
```

- Initialization Code

```
void setup() {
  Serial.begin(115200);                // 初始化串口波特率115200 Initialize serial communication at 115200 bps
  Wire.begin();                        // 初始化I2C通讯 Initialize I2C communication
  delay(1000);                         // 如果小车功能异常，可以增加这个延时 If the function is abnormal, you can
increase the delay
  pwm.begin();                         // PWM初始化 Initialize the Pulse Width Modulation (PWM) library
  pwm.setPWMFreq(PWM_FREQUENCY);       // 设置PWM频率 Set the PWM frequency
  setCarMove(STOP, 0);                 // 设置小车停止状态 Set the car to stop state
  RGB.begin();                         // 初始化RGB Initialize RGB
  RGB.show();                          // 刷新RGB显示 Refresh RGB display
  pinMode(BUZZER_PIN, OUTPUT);         // 设置蜂鸣器引脚输出模式 Set the buzzer pin output mode
}
```

- Looping code

```
void loop() {
  parseSerialData(recData());  // 处理后解析串口数据 Parse the serial port data after processing
  // 打印小车状态和控制小车 Print the car status and control the car
  if (uCurCarState != uPreCarState) {
    setCarMove(uCurCarState, uMotorSpeed);
    uPreCarState = uCurCarState;
  }
  // 打印RGB状态并控制RGB Print the RGB status and control the RGB
  if (uCurRgbState != uPreRgbState) {
    setRGBColor(uCurRgbState);
    uPreRgbState = uCurRgbState;
  }
  // 打印蜂鸣器状态并控制蜂鸣器 Print the buzzer status and control the buzzer
  if (uCurBuzState != uPreBuzState) {
    setBuzzerSound(uCurBuzState);
    uPreBuzState = uCurBuzState;
  }
  // 打印并控制舵机旋转角度 Print and control the Servo roate angle
  if (uCurServoAngle != uPreServoAngle) {
    setServoAngle(Servo1_PIN, uCurServoAngle);
    uPreServoAngle = uCurServoAngle;
  }
}
```

Experimental results

After compiling the program successfully, upload the code to the Arduino Uno development board.

After the program is started, the servo of the S1 interface will reset to 90°, and then we send data in the specified format, and the program will set the rotation angle of the servo according to the last three digits in the data.

If there is no display content, you can check whether the serial port baud rate is consistent with the code setting, and then press the RESET button on the development board.

```
The drive motor needs an external battery pack and the expansion board switch is turned on to drive normally.
The burning program cannot use other programs to occupy the serial port or an external serial communication
module (for example: WiFi camera module), otherwise the program cannot be burned or an error message will be
prompted!
```

File   Edit   Sketch   Tools   Help

Arduino Uno

09.Parse_Data_Servo.ino

```
1   #include <Wire.h>                      // 包含Wire(I2C)通讯库 Include Wire library
2   #include <Adafruit_NeoPixel.h>         // 包含Adafruit NeoPixel库 Include Adafruit NeoPixel library
3   #include <Adafruit_PWMServoDriver.h>   // 包含Adafruit PWMServoDriver库 Include Adafruit PWMServoDriver library
4
5   // 定义数据，如数组，标志位等 Define data such as arrays, flag bits, etc
6   #define BUFFER_SIZE 20
7   volatile uint8_t Rx_Index = 0;
8   volatile uint8_t Data_Flag = 0;
9   char* Data_Array[BUFFER_SIZE];
10  char Data_Servo[BUFFER_SIZE];
11
12  int DataArraySize = 0;
13
14  // 定义电机控制引脚 Define motor control pins
15  #define Motor_L1_F_PIN 11  // 控制小车左前方电机前进 Control the motor on the left front of the car
16  #define Motor_L1_B_PIN 10  // 控制小车左前方电机后退 Control the motor back on the left front of the car
17  #define Motor_L2_F_PIN 8   // 控制小车左后方电机前进 Control car left rear motor forward
18  #define Motor_L2_B_PIN 9   // 控制小车左后方电机后退 Control the car left rear motor back
19  #define Motor_R1_F_PIN 13  // 控制小车右前方电机前进 Control the right front motor of the car to move forward
20  #define Motor_R1_B_PIN 12  // 控制小车右前方电机后退 Control the motor back on the right front of the car
21  #define Motor_R2_F_PIN 14  // 控制小车右后方电机前进 Control car right rear motor forward
22  #define Motor_R2_B_PIN 15  // 控制小车右后方电机后退 Control car right rear motor back
23
```

Output   Serial Monitor ×

$1,3,1,0# $A045#

New Line   115200 baud

```
16:38:50.762 -> Car State:FORWARD
16:38:50.762 -> RGB State:BLUE
16:38:50.762 -> Buzzer State:On
16:38:50.762 -> Servo Angle:45
```

Ln 138, Col 2   Arduino Uno on COM8