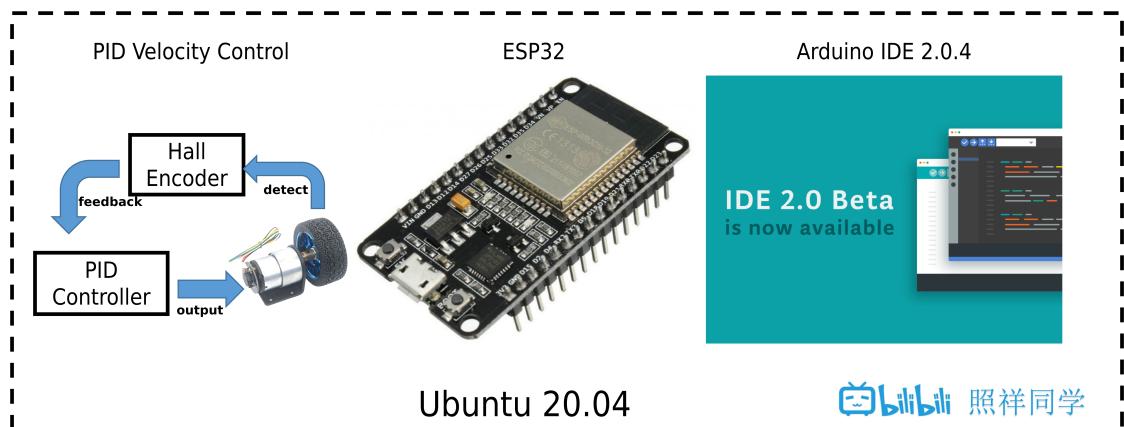


Chapter 7 ESP32_GPS_BDS_Module

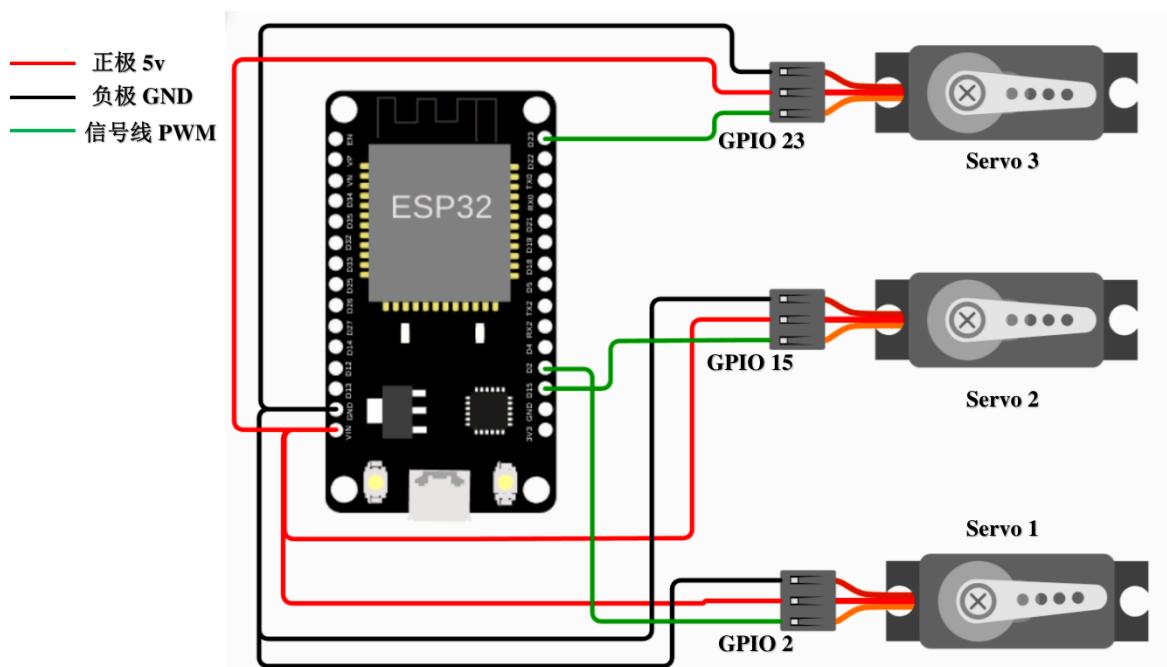


07-ESP32_Motor_Velocity_PID_Control

URL: https://github.com/ZhaoXiangBox/esp32_ros2_robot

Videos from Bilibili 照祥同学: [第七节：ESP32通过PID实现霍尔编码电机的速度控制](#)

First : ESP32 Arduino PWM servo test



·Code Tips:

```

// 定义PWM输出引脚为 GPIO2
#define Servo_Pin1 2

int freq = 50; // 频率为50Hz, 即周期为20ms
int channel = 8; // ESP32一共有16个PWM通道[0, 15], 其中前8个为高速PWM(80MHz), 后8个为低速(1MHz)
int resolution = 8; // 引脚输出的分辨率为 8bit 即范围为 [0, 255]

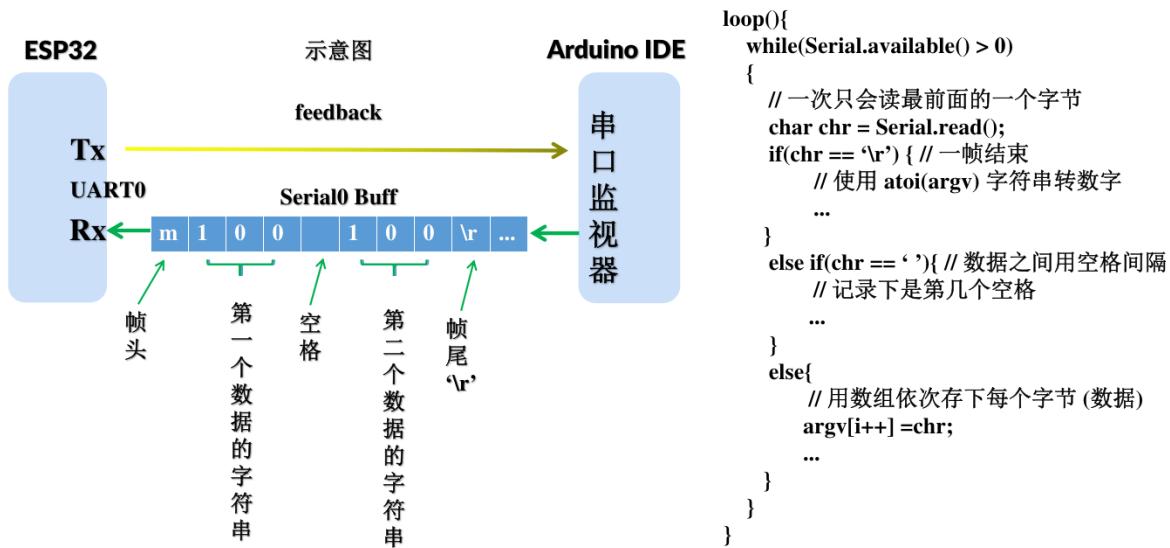
// 分别设置 每路 PWM 输出的通道、频率、引脚输出时的分辨率
ledcSetup(channel, freq, resolution);

// 给指定的GPIO引脚绑定 PWM通道
ledcAttachPin(Servo_Pin1, channel);

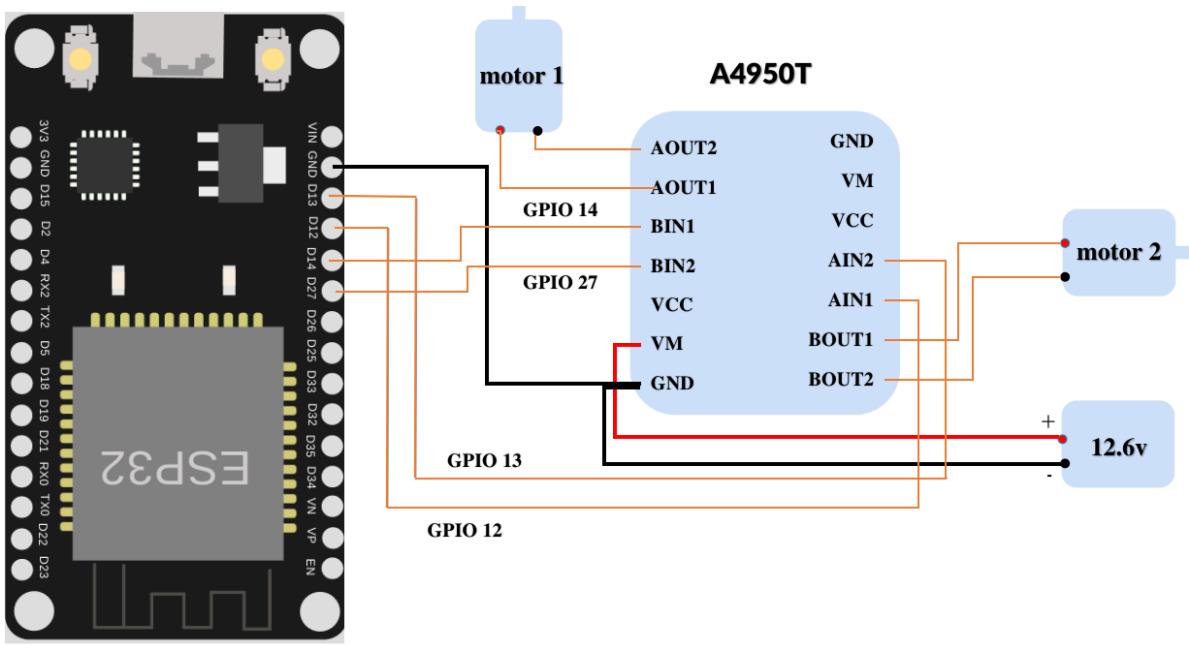
```

Second : Define a protocol of Serial communication

串口协议的设计与实现



Third : Serial A4950T PWM Velocity test



· Code Tips

```
// 左边电机转动方向控制位 引脚
#define Back_Left_D1 12
#define Back_Left_D1_B 13

// 右边电机转动方向控制位 引脚
#define Back_Right_D1 14
#define Back_Right_D1_B 27

void setSpeeds(int m1Speed, int m2Speed) {
    if(m1Speed > 0){      // 控制左侧电机
        analogWrite(Back_Left_D1, m1Speed);
        analogWrite(Back_Left_D1_B, LOW);
    }
    else{
        analogWrite(Back_Left_D1, LOW);
        analogWrite(Back_Left_D1_B, -m1Speed);
    }

    if(m2Speed > 0){      // 控制右侧电机
        analogWrite(Back_Right_D1, m2Speed);
        analogWrite(Back_Right_D1_B, LOW);
    }
    else{
        analogWrite(Back_Right_D1, LOW);
        analogWrite(Back_Right_D1_B, -m2Speed);
    }
}
```

Fourth : PID controller for Motor Velocity

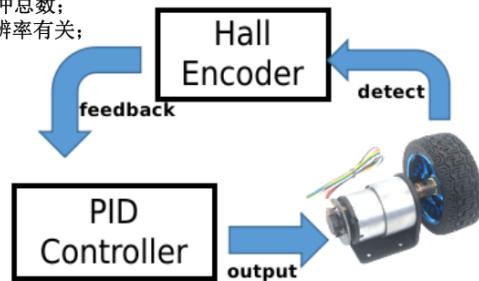
PID-电机速度控制的设计与实现

一、PID 速度控制

- 1、相邻两次PID计算的时间周期（Interval）是10ms，即 $f = 100\text{Hz}$ ；
- 2、PID控制器的输入目标形式为：脉冲/Interval；
- 3、PID的参考（霍尔编码值）为：时间周期（Interval）内的脉冲总数；
- 4、PID的输出为PWM值，其范围为[-255,255]，范围与PWM分辨率有关；

二、电机信息

- 1、编码器：霍尔AB两相编码器，11ppr，电机减速比：1: 30；
- 2、ESP32脉冲采集方式：外部中断、且边沿触发方式；
- 3、电机转一圈的脉冲数： $\text{total} = 2 * 2 * 11 * 30 = 1320 \text{ pulse/圈}$ ；
- 4、电机输出轴轮胎外径： $d = 0.065 \text{ m}$ ；



三、转换公式

1、速度target_velocity (m/s) ---> 脉冲数/pulse
$$\frac{\text{pulses}}{\text{Interval}} = \frac{1}{f} * \frac{\text{total}}{d * PI} * \text{target_velocity}$$

2、脉冲数/pulse ---> 速度current_velocity (m/s)
$$\text{current_velocity} = \frac{d * PI}{\text{total}} * \frac{\text{pulses}}{\text{interval}} * f$$

· Code Tips

```
// 需要留意的是：本次只对一个电机进行了PID调速，所以只用到了一组编码器；  
// "encoder_driver.h" 中  
void Init_Encoder() {  
    resetEncoders();  
    pinMode(LEFT_ENCODER_A, INPUT);  
    pinMode(LEFT_ENCODER_B, INPUT);  
  
    // Attaching the ISR to encoder Left A B  
    attachInterrupt(digitalPinToInterrupt(LEFT_ENCODER_A), Left_encoder_isr_A,  
    CHANGE);  
    attachInterrupt(digitalPinToInterrupt(LEFT_ENCODER_B), Left_encoder_isr_B,  
    CHANGE);  
  
    // 若同时对两个电机调速，需要解除如下程序的注释  
    // pinMode(RIGHT_ENCODER_A, INPUT);  
    // pinMode(RIGHT_ENCODER_B, INPUT);  
  
    // Attaching the ISR to encoder Rigth A B  
    // attachInterrupt(digitalPinToInterrupt(RIGHT_ENCODER_A),  
    Right_encoder_isr_A, CHANGE);  
    // attachInterrupt(digitalPinToInterrupt(RIGHT_ENCODER_B),  
    Right_encoder_isr_B, CHANGE);  
}
```

Tips:

- 1、整个速度控制对于目标脉冲为0时，采用的是自由停车的方式，即：停止PID运算，电机自由停。程序中使用了一个标志位 moving 来判断目标脉冲值是否为 0。
- 2、对于新手如何使用本程序，后续会出一期增刊视频，专门讲解如何调整PID参数和如何判断电机接线正确。
- 3、速度控制的方式很多，我只是用了其中的一种，非常欢迎大佬能提出宝贵的建议和更高效简洁的办法。

4、由于ESP32没有专门的编码器解码器，这里采用的是外部中断的方式记录脉冲值，所以建议使用霍尔编码电机。如果使用分辨率非常高的光电编码器，由于电机转动起来进入中断的次数太频繁了，会影响主程序运行。

第七节：补充说明部分

update by zhaoxiangli 2023.04.21