一、硬件

1. ESP32——烧录程序的地方
2. 2804 直流无刷电机
3. 磁编码器 AS5600——反馈位置数据
4. SimpleFOC 板——电机驱动器
5. 12V 直流电源

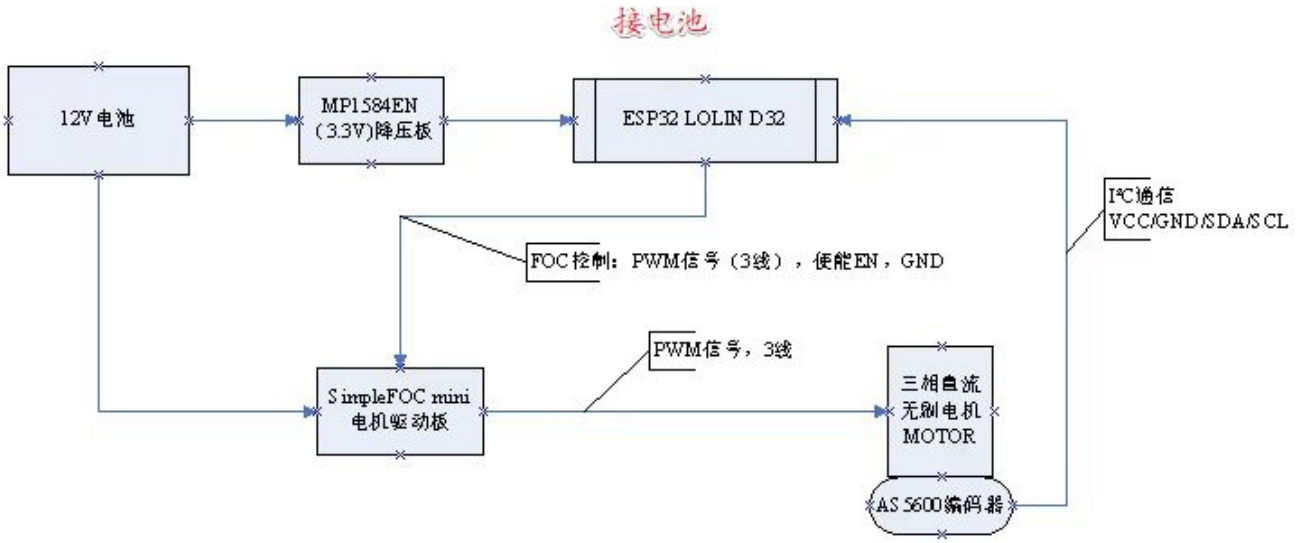
淘宝搜索：M 创动工坊或直接打开：mcdgf.taobao.com，可以找到我们，我们这有整套的硬件，提供免费的技术支持。

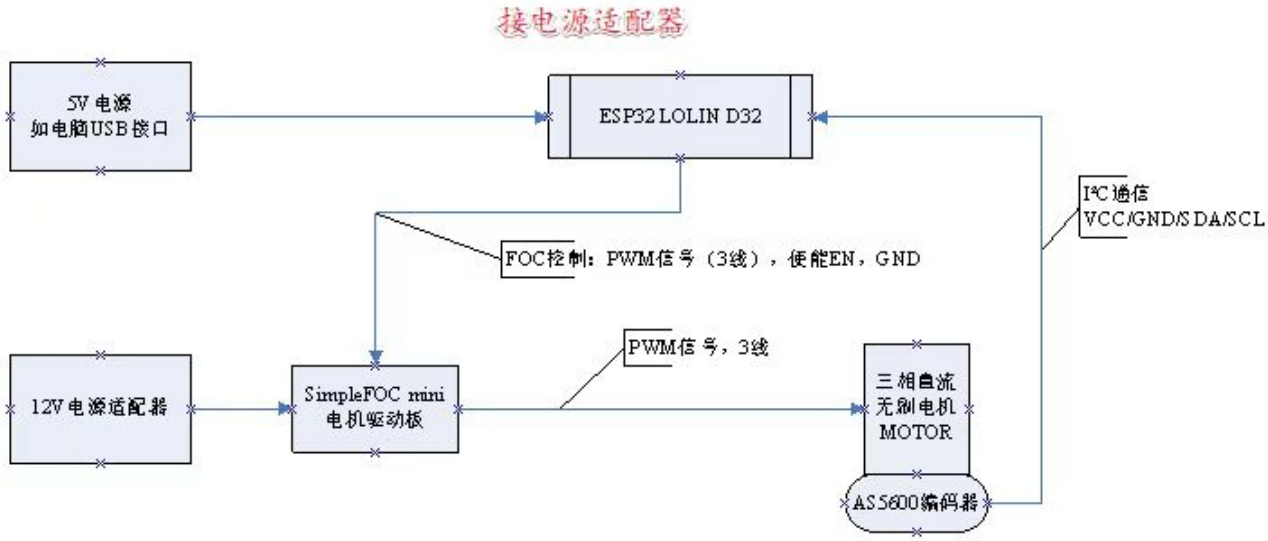
二、软件：

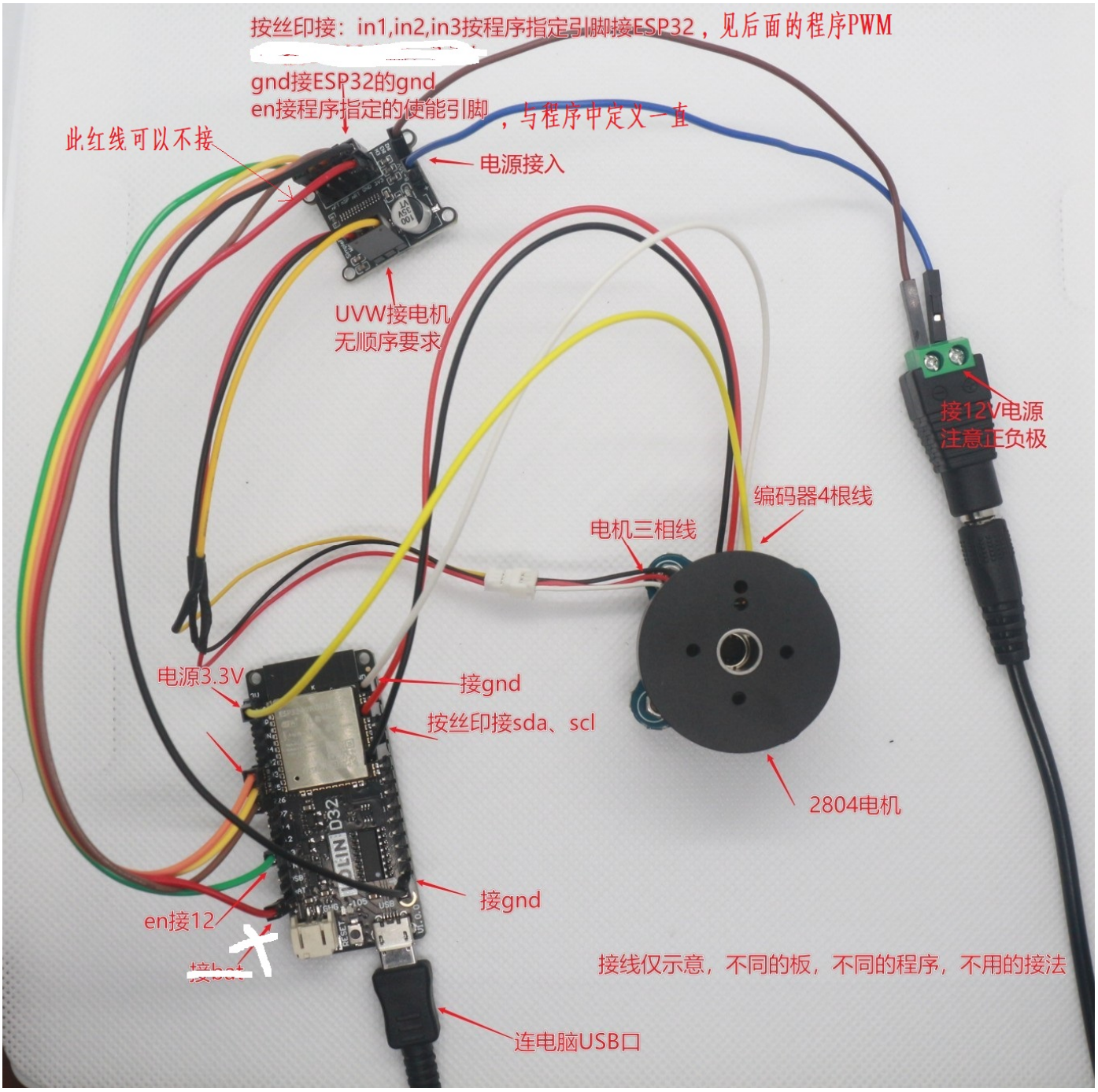
1. Arduino，需要加载 ESP32 板和 Simplefoc 库

2. 也可以用 VSCode+PlatformIO 插件
3. 也可以用 VSCode+Arduino 插件

三、接线


接电池
12V电池
MP1584EN
(3.3V)降压板
ESP32 LOLIN D32
I²C通信
VCC/GND/SDA/SCL
FOC控制：PWM信号（3线），使能EN，GND
PWM信号，3线
SimpleFOC mini
电机驱动板
三相直流
无刷电机
MOTOR
AS5600编码器


接电源适配器
5V电源
如电脑USB接口
ESP32 LOLIN D32
I²C通信
VCC/GND/SDA/SCL
FOC控制：PWM信号（3线），使能EN，GND
12V电源适配器
SimpleFOC mini
电机驱动板
PWM信号，3线
三相直流
无刷电机
MOTOR
AS5600编码器



四、学习网址

http://simplefoc.cn/

https://simplefoc.com/

B 站上也有很多信息值得学习，如：灯哥开源，攀哥

五、

下面是示例程序：闭环位置

```
/**
 *
 * Position/angle motion control example
 * Steps:
 * 1) Configure the motor and magnetic sensor
 * 2) Run the code
 * 3) Set the target angle (in radians) from serial terminal
```

```
 *
 */
#include <SimpleFOC.h>//要先加载库，在工具->管理库，中搜索 simplefoc 下载最新版库

// magnetic sensor instance - MagneticSensorI2C
MagneticSensorI2C sensor = MagneticSensorI2C(AS5600_I2C);//因为编码器是 AS5600，所以选择这个
// BLDC motor & driver instance
BLDCMotor motor = BLDCMotor(7);//2804、2208 电机的极对数都是 7，4015 电机极对数是 11
BLDCDriver3PWM driver = BLDCDriver3PWM(32,33,25,12);//FOC 驱动板与 ESP32 连接的引脚对应起来，PWM 接 32-33-25,板使能接 12;

// angle set point variable
float target_angle = 0;
// instantiate the commander
Commander command = Commander(Serial);//从串口输入命令
void doTarget(char* cmd) { command.scalar(&target_angle, cmd); }

void setup() {

  // initialise magnetic sensor hardware
  sensor.init();//传感器初始化
  // link the motor to the sensor
  motor.linkSensor(&sensor);//电机连接传感器

  // driver config
  // power supply voltage [V]
  driver.voltage_power_supply = 12;//供电电压为 12V
  driver.init();//驱动器初始化
  // link the motor and the driver
  motor.linkDriver(&driver);//定义电机与驱动器连接

  // choose FOC modulation (optional)
  motor.foc_modulation = FOCModulationType::SpaceVectorPWM;//采用 PWM 方式驱动

  // set motion control loop to be used
  motor.controller = MotionControlType::angle;//位置模式，且输入值为弧度

  // contoller configuration
  // default parameters in defaults.h

  // velocity PI controller parameters
  motor.PID_velocity.P = 0.05f;//定义 PID 参数值，值不要定义太大，容易抖动
  motor.PID_velocity.I = 0.06;
  motor.PID_velocity.D = 0;
  // maximal voltage to be set to the motor
  motor.voltage_limit = 6;//限制最大电压，电源电压是 12V，而电机要正反转，所以从-6 到 6，幅度也是 12V

  // velocity low pass filtering time constant
  // the lower the less filtered
```

```
  motor.LPF_velocity.Tf = 0.01f;//这可以滤除电机的噪声和高频振动，从而使速度控制更加稳定。

  // angle P controller
  motor.P_angle.P = 20;//位置的 P 环
  // maximal velocity of the position control
  motor.velocity_limit = 20;//限制最大速度

  // use monitoring with serial
  Serial.begin(115200);//串口频率
  // comment out if not needed
  motor.useMonitoring(Serial);//使用串口输入


  // initialize motor
  motor.init();//电机初始化
  // align sensor and start FOC
  motor.initFOC();//对齐编码器

  // add target command T
  command.add('T', doTarget, "target angle");//使用 T+数字输入位置

  Serial.println(F("Motor ready."));//串口输出到屏幕
  Serial.println(F("Set the target angle using serial terminal:"));
  _delay(1000);
}


void loop() {

  // main FOC algorithm function
  // the faster you run this function the better
  // Arduino UNO loop  ~1kHz
  // Bluepill loop ~10kHz
  motor.loopFOC();//电机起劲

  // Motion control function
  // velocity, position or voltage (defined in motor.controller)
  // this function can be run at much lower frequency than loopFOC() function
  // You can also use motor.move() and set the motor.target in the code
  motor.move(target_angle);//转到输入的角度


  // function intended to be used with serial plotter to monitor motor variables
  // significantly slowing the execution down!!!!
  // motor.monitor();

  // user communication
  command.run();//接收串口的命令和位置
}
```

再来一个程序：灯哥开源的，开环速度控制

```
#include <Arduino.h>

// put function declarations here:
//灯哥开源，转载请著名出处
//仅在 DengFOC 上测试过
//PWM 输出引脚定义
int pwmA = 9;
int pwmB = 10;
int pwmC = 11;
int enBle = 8;

//初始变量及函数定义
#define _constrain(amt,low,high) ((amt)<(low)?(low):((amt)>(high)?(high):(amt)))
//宏定义实现的一个约束函数,用于限制一个值的范围。
//具体来说，该宏定义的名称为 _constrain，接受三个参数 amt、low 和 high，分别表示要限制的值、最小值和最大值。该宏定义的实现使用了三元运算符，根据 amt 是否小于 low 或大于 high，返回其中的最大或最小值，或者返回原值。
//换句话说，如果 amt 小于 low，则返回 low；如果 amt 大于 high，则返回 high；否则返回 amt。这样，_constrain(amt, low, high) 就会将 amt 约束在 [low, high] 的范围内。
float voltage_power_supply=12.6;
float shaft_angle=0,open_loop_timestamp=0;
float zero_electric_angle=0,Ualpha,Ubeta=0,Ua=0,Ub=0,Uc=0,dc_a=0,dc_b=0,dc_c=0;


void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  //PWM 设置
  pinMode(pwmA, OUTPUT);
  pinMode(pwmB, OUTPUT);
  pinMode(pwmC, OUTPUT);
  pinMode(enBle,OUTPUT);
//    ledcAttachPin(pwmA, 0);
//    ledcAttachPin(pwmB, 1);
//    ledcAttachPin(pwmC, 2);
//    ledcSetup(0, 30000, 8);   //pwm 频道，频率，精度
//    ledcSetup(1, 30000, 8);   //pwm 频道，频率，精度
//    ledcSetup(2, 30000, 8);   //pwm 频道，频率，精度
  Serial.println("完成 PWM 初始化设置");
  delay(3000);
  digitalWrite(enBle,HIGH);
}

// 电角度求解
```

```
float _electricalAngle(float shaft_angle, int pole_pairs) {
  return (shaft_angle * pole_pairs);
}

// 归一化角度到 [0,2PI]
float _normalizeAngle(float angle){
  float a = fmod(angle, 2*PI);   //取余运算可以用于归一化，列出特殊值例子算便知
  return a >= 0 ? a : (a + 2*PI);
  //三目运算符。格式：condition ? expr1 : expr2
  //其中，condition 是要求值的条件表达式，如果条件成立，则返回 expr1 的值，否则返回 expr2 的值。可以将三目运算符视为 if-else 语句的简化形式。
  //fmod 函数的余数的符号与除数相同。因此，当 angle 的值为负数时，余数的符号将与 _2PI 的符号相反。也就是说，如果 angle 的值小于 0 且 _2PI 的值为正数，则 fmod(angle, _2PI) 的余数将为负数。
  //例如，当 angle 的值为 -PI/2，_2PI 的值为 2PI 时，fmod(angle, _2PI) 将返回一个负数。在这种情况下，可以通过将负数的余数加上 _2PI 来将角度归一化到 [0, 2PI] 的范围内，以确保角度的值始终为正数。
}


// 设置 PWM 到控制器输出
void setPwm(float Ua, float Ub, float Uc) {

  // 计算占空比
  // 限制占空比从 0 到 1
  dc_a = _constrain(Ua / voltage_power_supply, 0.0f , 1.0f );
  dc_b = _constrain(Ub / voltage_power_supply, 0.0f , 1.0f );
  dc_c = _constrain(Uc / voltage_power_supply, 0.0f , 1.0f );

  //写入 PWM 到 PWM 0 1 2 通道
//    ledcWrite(0, dc_a*255);
//    ledcWrite(1, dc_b*255);
//    ledcWrite(2, dc_c*255);
  analogWrite(pwmA, dc_a * 255);
  analogWrite(pwmB, dc_b * 255);
  analogWrite(pwmC, dc_c * 255);
}

void setPhaseVoltage(float Uq,float Ud, float angle_el) {
  angle_el = _normalizeAngle(angle_el + zero_electric_angle);
  // 帕克逆变换
  Ualpha =  -Uq*sin(angle_el);
  Ubeta =   Uq*cos(angle_el);

  // 克拉克逆变换
  Ua = Ualpha + voltage_power_supply/2;
  Ub = (sqrt(3)*Ubeta-Ualpha)/2 + voltage_power_supply/2;
  Uc = (-Ualpha-sqrt(3)*Ubeta)/2 + voltage_power_supply/2;
  setPwm(Ua,Ub,Uc);
}
```

```
//开环速度函数
float velocityOpenloop(float target_velocity){
  unsigned long now_us = micros();  //获取从开启芯片以来的微秒数，它的精度是 4 微秒。 micros() 返回的是一个无符号长整型（unsigned long）的值

  //计算当前每个 Loop 的运行时间间隔
  float Ts = (now_us - open_loop_timestamp) * 1e-6f;

  //由于 micros() 函数返回的时间戳会在大约 70 分钟之后重新开始计数，在由 70 分钟跳变到 0 时，TS 会出现异常，因此需要进行修正。如果时间间隔小于等于零或大于 0.5 秒，则将其设置为一个较小的默认值，即 1e-3f
  if(Ts <= 0 || Ts > 0.5f) Ts = 1e-3f;


  // 通过乘以时间间隔和目标速度来计算需要转动的机械角度，存储在 shaft_angle 变量中。在此之前，还需要对轴角度进行归一化，以确保其值在 0 到 2π 之间。
  shaft_angle = _normalizeAngle(shaft_angle + target_velocity*Ts);
  //以目标速度为 10 rad/s 为例，如果时间间隔是 1 秒，则在每个循环中需要增加 10 * 1 = 10 弧度的角度变化量，才能使电机转动到目标速度。
  //如果时间间隔是 0.1 秒，那么在每个循环中需要增加的角度变化量就是 10 * 0.1 = 1 弧度，才能实现相同的目标速度。因此，电机轴的转动角度取决于目标速度和时间间隔的乘积。

  // 使用早前设置的 voltage_power_supply 的 1/3 作为 Uq 值，这个值会直接影响输出力矩
  // 最大只能设置为 Uq = voltage_power_supply/2，否则 ua,ub,uc 会超出供电电压限幅
  float Uq = voltage_power_supply/3;

  setPhaseVoltage(Uq,  0, _electricalAngle(shaft_angle, 7));

  open_loop_timestamp = now_us;  //用于计算下一个时间间隔

  return Uq;
}


void loop() {
  // put your main code here, to run repeatedly:
    velocityOpenloop(1);
}
```

祝您成功！