Report – ESC Calibration

October 2023

Dinh-Son Vu

School of Science, Engineering, and Technology

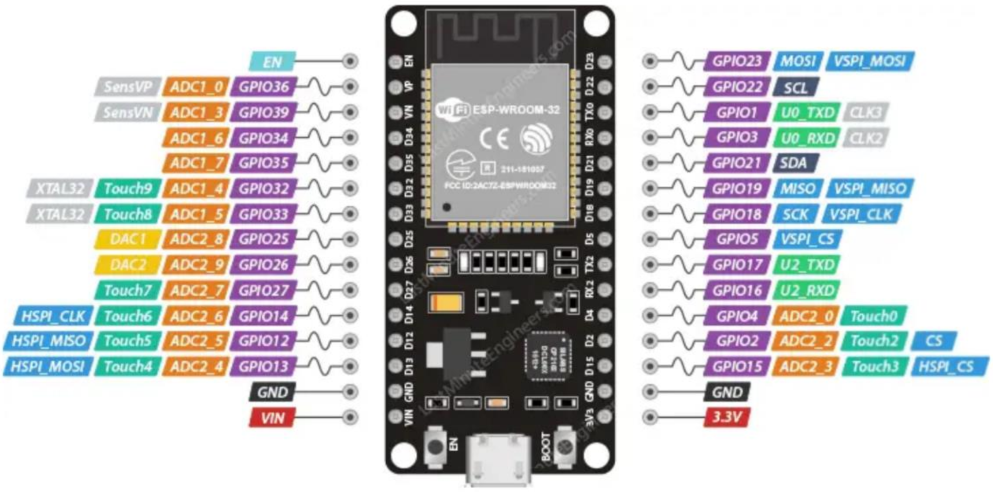
What's next...

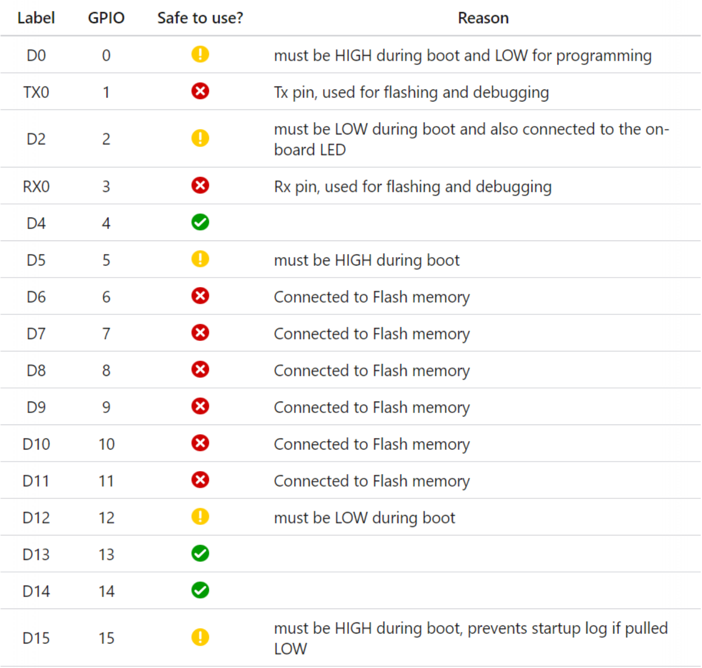
Table of Contents

1. Introduction

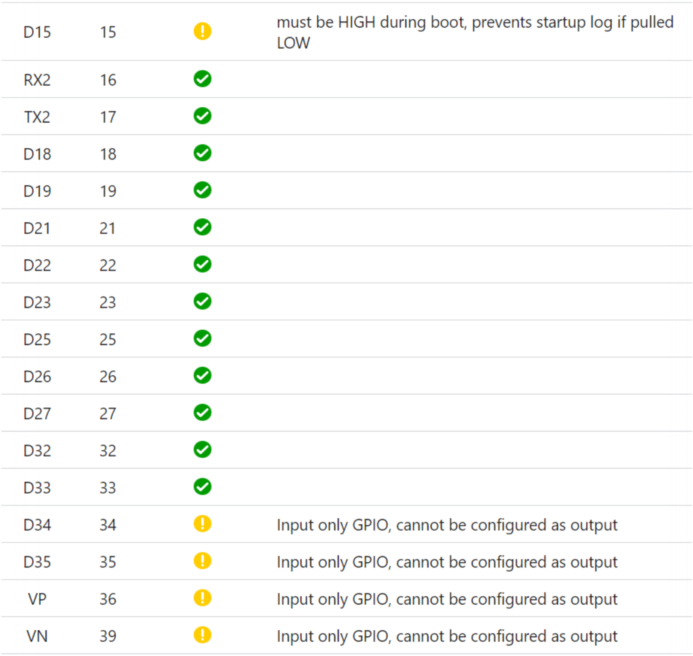
This document introduces how to use an ESC with a BLDC motor and how to calibrate an ESC to use it on its full range. This procedure is essential to work with a balanced drone.

2. ESP32 Pinout





Label	GPIO	Safe to use?	Reason
D0	0	!	must be HIGH during boot and LOW for programming
TX0	1	✗	Tx pin, used for flashing and debugging
D2	2	!	must be LOW during boot and also connected to the on-board LED
RX0	3	✗	Rx pin, used for flashing and debugging
D4	4	✓	
D5	5	!	must be HIGH during boot
D6	6	✗	Connected to Flash memory
D7	7	✗	Connected to Flash memory
D8	8	✗	Connected to Flash memory
D9	9	✗	Connected to Flash memory
D10	10	✗	Connected to Flash memory
D11	11	✗	Connected to Flash memory
D12	12	!	must be LOW during boot
D13	13	✓	
D14	14	✓	
D15	15	!	must be HIGH during boot, prevents startup log if pulled LOW



D15	15	!	must be HIGH during boot, prevents startup log if pulled LOW
RX2	16	✓	
TX2	17	✓	
D18	18	✓	
D19	19	✓	
D21	21	✓	
D22	22	✓	
D23	23	✓	
D25	25	✓	
D26	26	✓	
D27	27	✓	
D32	32	✓	
D33	33	✓	
D34	34	!	Input only GPIO, cannot be configured as output
D35	35	!	Input only GPIO, cannot be configured as output
VP	36	!	Input only GPIO, cannot be configured as output
VN	39	!	Input only GPIO, cannot be configured as output

3. ESC and BLDC motor functioning

An electronic speed controller (ESC) is a component to drive a brushless direct current (BLDC) motor. An ESP32 is a microcontroller that is used to send the PWM command to the ESC, which send the desired voltage to drive the BLDC at the desired speed.

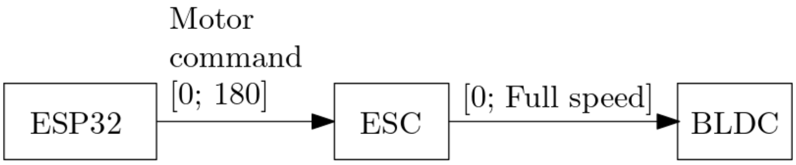


Figure 1: Functioning of an ESP32, ESC, and BLDC.

A sample code is given as follows. The code uses the library *ESP32Servo.h,* which is specific to the microcontroller ESP32. For Arduino UNO, the library *Servo.h* may be used.

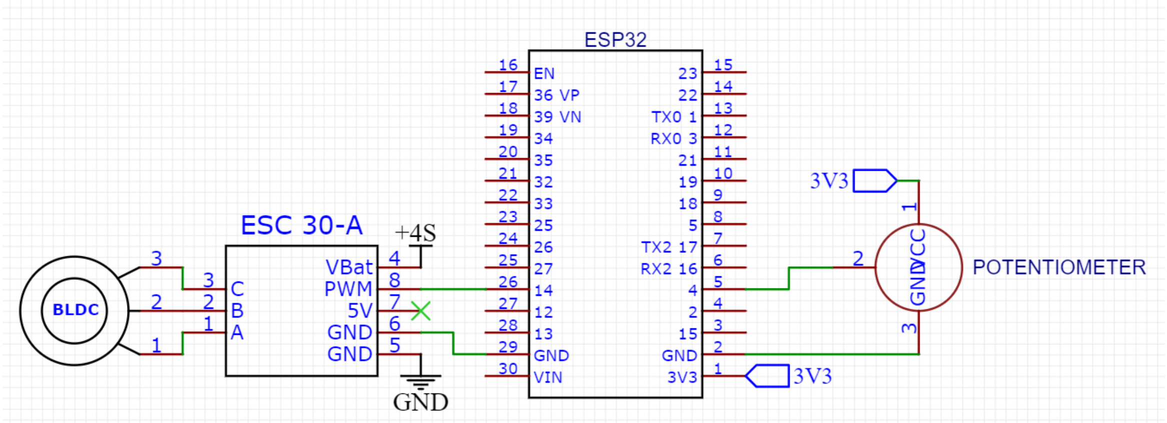


Figure 2: Wiring diagram between a BLDC, ESC, ESP32, and a potentiometer.

```
#include <Arduino.h>
#include <ESP32Servo.h>
// ================================================================
// Variable declaration
// ================================================================
#define MAX_SIGNAL 2000 // Parameter required for the ESC definition
#define MIN_SIGNAL 1000 // Parameter required for the ESC definition
#define MOTOR_PIN 14    // Pin 14 attached to the ESC signal pin
#define POT_PIN 4       // Pin 4 attached to the potentiometer

Servo ESC;                   // Define the ESC
int CtrlPWM;                 // Control Signal. Varies between [0 - 180]
unsigned long time_prev = 0; // Variable used for serial monitoring
// ================================================================
// Function declaration
// ================================================================
void SerialDataPrint(); // Function to print data on the serial monitor
void Init_Serial();     // Function to init the serial monitor
void Init_ESC();        // Function to init the ESC
```

```
// ================================================================
// Setup
// ================================================================
void setup()
{
  Init_Serial();
  Init_ESC();
}
// ================================================================
// Loop
// ================================================================
void loop()
{
  CtrlPWM = map(analogRead(POT_PIN), 0, 4095, 0, 180); // Read the pot, map the reading from [0, 4095] to [0, 180]
  ESC.write(CtrlPWM);                                  // Send the command to the ESC
  SerialDataPrint();
}

// ================================================================
// Function Definition
// ================================================================
void Init_Serial()
{
  Serial.begin(115200);
  while (!Serial)
    ;
}
// ================================================================
void Init_ESC()
{
  ESC.attach(MOTOR_PIN, MIN_SIGNAL, MAX_SIGNAL);
  ESC.writeMicroseconds(MIN_SIGNAL);
}
// ================================================================
void SerialDataPrint()
{
  if (micros() - time_prev >= 20000)
  {
    time_prev = micros();
    Serial.print(millis());
    Serial.print("\t");
    Serial.println(CtrlPWM);
  }
}
```

4. ESC Calibration

WARNING:

Be sure to remove the propeller while doing the calibration. Normally, the motor should not rotate, however the ESC can enter other modes, which can rotate the motor at its maximum speed.

Quite often, the BLDC does not rotate until the potentiometer has reached a certain value: this is the dead zone of the ESC, which can be reduce with calibration. This procedure should be done once for each new ESC. It is possible (and recommended) to calibrate the four ESC of a drone at the same time. Follow the sample code to calibrate one ESC.

The wiring diagram is identical as in Figure 2.

```
#include <Arduino.h>
#include <ESP32Servo.h>
// ================================================================
// Variable declaration
// ================================================================
#define MAX_SIGNAL 2000 // Parameter required for the ESC definition
#define MIN_SIGNAL 1000 // Parameter required for the ESC definition
#define MOTOR_PIN 13    // Pin 13 attached to the ESC signal pin
#define POT_PIN 4       // Pin 4 attached to the potentiometer

Servo ESC;                   // Define the ESC
int CtrlPWM;                 // Control Signal. Varies between [0 - 180]
unsigned long time_prev = 0; // Variable used for serial monitoring
// ================================================================
// Function declaration
// ================================================================
void SerialDataPrint();  // Function to print data on the serial monitor
void Init_Serial();      // Function to init the serial monitor
void WaitForKeyStroke(); // Function to interact with the serial monitor

// ================================================================
// Setup
// ================================================================
void setup()
{
  Init_Serial();                                 // Initialize the serial
communication
  ESC.attach(MOTOR_PIN, MIN_SIGNAL, MAX_SIGNAL); // Initialize the ESC

  Serial.println();
  Serial.println("Calibration step 1. Disconnect the battery.");
  Serial.println("Press any key to continue.");
  WaitForKeyStroke();
  ESC.writeMicroseconds(MAX_SIGNAL); // Sending MAX_SIGNAL tells the ESC to enter
calibration mode

  Serial.println();
  Serial.println("Calibration step 2. Connect the battery.");
  Serial.println("Wait for two short bips.");
  Serial.println("Press any key to continue.");
  WaitForKeyStroke();

  ESC.writeMicroseconds(MIN_SIGNAL); // Sending MIN_SIGNAL tells the ESC the
calibration value
  Serial.println();
  Serial.println("Wait for 4 short bips, and one long bip.");
  Serial.println("Press any key to finish.");
  WaitForKeyStroke();
}

// ================================================================
// Loop
// ================================================================
void loop()
{
  CtrlPWM = map(analogRead(POT_PIN), 0, 4095, 0, 180); // Read the pot, map the reading from [0, 4095] to [0, 180]
  ESC.write(CtrlPWM);                                  // Send the command to the ESC
  SerialDataPrint();                                   // Print data on the serial monitor for debugging
}
// ================================================================
// Function Definition
// ================================================================
void Init_Serial()
{
  Serial.begin(115200);
  while (!Serial)
    ;
}
// ================================================================
void SerialDataPrint()
{
  if (micros() - time_prev >= 20000)
  {
    time_prev = micros();
    Serial.print(millis());
    Serial.print("\t");
    Serial.println(CtrlPWM);
  }
}
// ================================================================
void WaitForKeyStroke()
{
  while (!Serial.available())
    ;
  while (Serial.available())
    Serial.read();
}
```