ON7WP AZ/EL WIFI rotator controller based on Wemos D1 R1

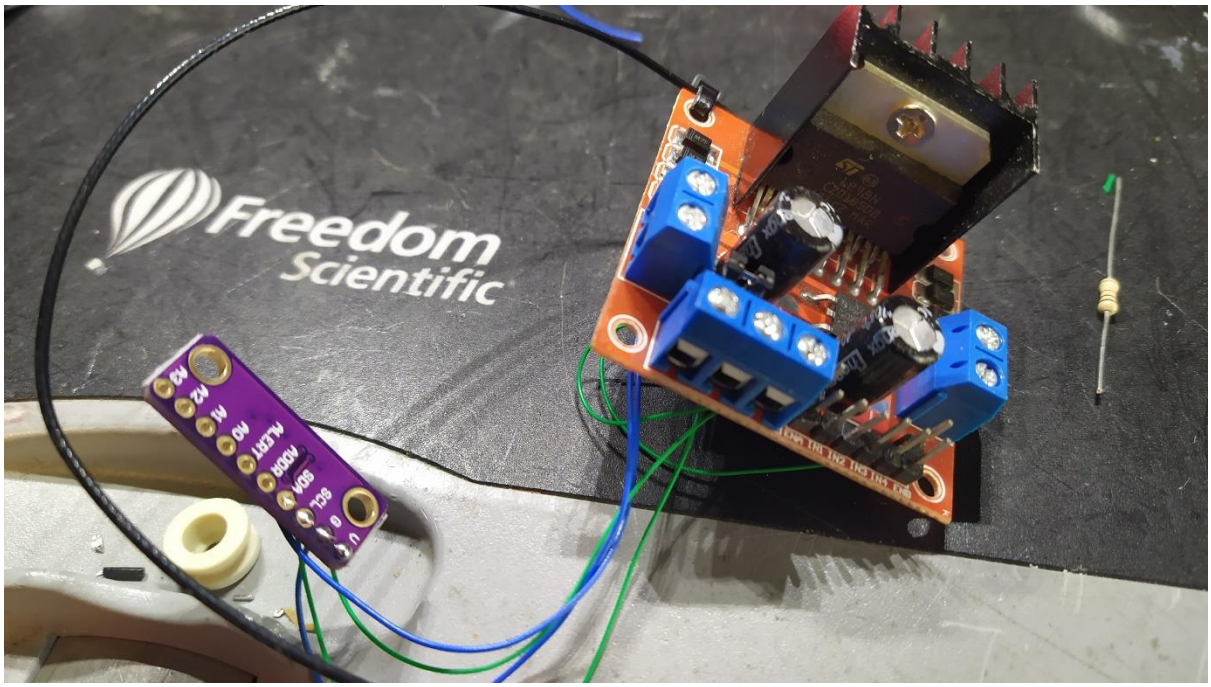
A very cheap Wireless controller that can be used from anywhere in the world using a Blynk app on your phone.

The Hardware are 3 Chinese boards:

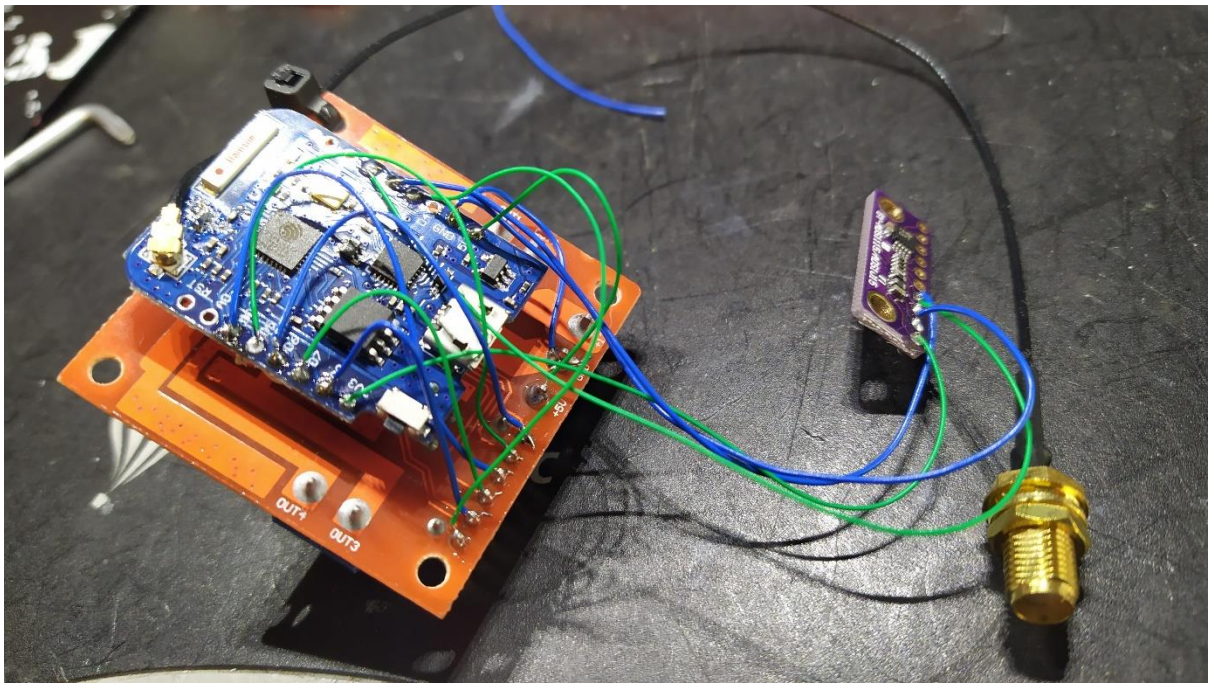
- A Wemos Mini D1 ESP8266 based microcontroller
- An ADS1015 10 bit Precision 4 input ADC
- A L298N Dual motor driver bridge

The software offers remote feedback of the actual rotor position and uses smart motor drive control, slowing down the speed in three steps when the target is almost reached.

The unit can drive any DC motor (Yaesu, Create,….) An AC version is in development.



Motor driver L289N Azimuth and potmeter feedback connections



Azimuth motor connections to L298 Motor driver board:

int enA = 12 = D6

int in1 = 14 = D5

int in2 = 16 = D0

Elevation motor connections to L298 Motor driver board:

int enB = 0 = D3

int in3 = 13 = D7

int in4 = 15 = D8

ADS1015 12bits ADC board connections:

SCL = 5 = D1

SDA = 4 = D2

V = 3,3 V

G = GND

analog in value coming from potentiometers: **(max 3,3 V !!!)**

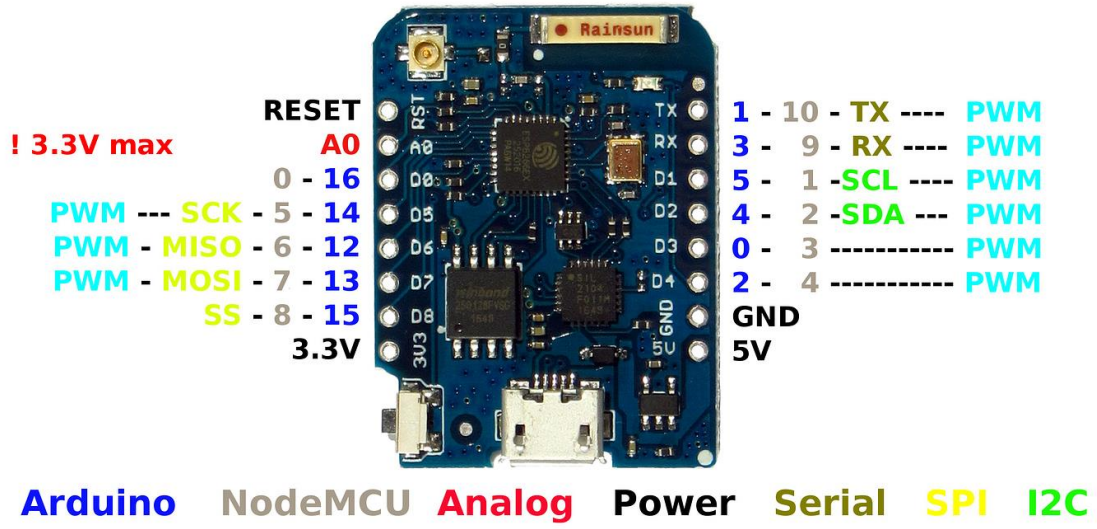
Becareful with the pinning used in the Arduino program, they differ from the naming on the Wemos D1 board !!

As example if you would drive PIN5 in the Arduino software…

> Strangely this activates pin D1 instead of D5.

This might be a clue 😉 … pin #5 in Arduino format is referring to D1 on the board … which is what the board actually responds to, except in limited cases where the Arduino IDE is set for that actual board.

In other words, ignore the silk screened references and follow the Arduino pinouts for that board.



PS, this “issue” of mistakenly using the silkscreened labeling instead of Arduino referenced numbers **IS** mentioned repeatedly in this forum 😜

Don not forget to resolder the jumper next to the external antenna connector if you want to use the external antenna input !

Arduino IDE Program as a text file:

```
#define BLYNK_PRINT Serial

#include <WiFiClient.h>

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

#include <Wire.h>

#include <Adafruit_ADS1X15.h>


Adafruit_ADS1015 ads1015;


char auth[] = "Put here your Blynk Authentication token";

char ssid[] = "Put here your WiFi SSID";

char pass[] = "Put here your WiFi Password";



// Motor Azimuth connections


int enA = 12;


int in1 = 14;


int in2 = 16;


// Motor Elevation connections


int enB = 0;


int in3 = 13;


int in4 = 15;
```

```
int rotorfeedback_AZ; // variable to store the value coming from the analog pin

int act_AZ;// variable to store the position value coming from the conversion formula

int error_AZ;// variable to store the pointing error

int heading_AZ = 180; // variable to store the position value coming from the remote control

int set_AZ;// variable to store the position value coming from the conversion formula

int rotorfeedback_EL; // variable to store the value coming from the analog pin

int act_EL;// variable to store the position value coming from the conversion formula

int error_EL;// variable to store the pointing error

int heading_EL = 0; // variable to store the position value coming from the remote control

int set_EL;// variable to store the position value coming from the conversion formula




void setup() {

ads1015.setGain(GAIN_ONE);     // 1x gain   +/- 4.096V  1 bit = 2mV 0 to 1650 bits
ads1015.begin();

// Set all the motor control pins to outputs
```

```
pinMode(enA, OUTPUT); // PWM Azimuth speed control ESP8266 uses 10 BIT PWM !!

pinMode(in1, OUTPUT);

pinMode(in2, OUTPUT);

pinMode(enB, OUTPUT); // PWM Elevation speed control ESP8266 uses 10 BIT PWM !!

pinMode(in3, OUTPUT);

pinMode(in4, OUTPUT);


// Turn off motors - Initial state

digitalWrite(in1, LOW);

digitalWrite(in2, LOW);

digitalWrite(in3, LOW);

digitalWrite(in4, LOW);

// Display wifi connect status on serial monitor
Serial.begin(115200);

  delay(10);
  Serial.print("Connecting to ");
  Serial.println(ssid);
```

```
  WiFi.begin(ssid, pass);
  int wifi_ctr = 0;
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("WiFi connected");

Blynk.begin(auth, ssid, pass, "server.wyns.it", 8081);
}



#define HEADING_AZ V0
#define ACT_AZ V1
#define HEADING_EL V2
#define ACT_EL V3
BLYNK_WRITE(HEADING_AZ)
{
set_AZ = param.asInt(); // writes parameter from the app to the controller
}

BLYNK_WRITE(HEADING_EL)
{
set_EL = param.asInt(); // writes parameter from the app to the controller
}
// the loop function runs over and over again forever
void loop() {
Blynk.run();
```

```
Blynk.virtualWrite(ACT_AZ, act_AZ); // writes parameter from the rotor to the app

Blynk.virtualWrite(ACT_EL, act_EL); // writes parameter from the rotor to the app


int16_t adc0, adc1, adc2, adc3;


rotorfeedback_AZ= ads1015.readADC_SingleEnded(0); // read the analog azimuth potmeter value:


rotorfeedback_EL= ads1015.readADC_SingleEnded(1); // read the analog elevation potmeter value:

//  adc2 = ads1015.readADC_SingleEnded(2);

//  adc3 = ads1015.readADC_SingleEnded(3);



act_AZ = 1.125 * (rotorfeedback_AZ * (359/1650.0)-39); // position conversion formula, 10 bits for ESP but 12 for external ADS1015

act_EL = 2.24 * (rotorfeedback_EL * (183/1650.0)-52.5); // position conversion formula, 10 bits for ESP but 12 for external ADS1015

//first number is potmeter calibration factor, last term potmeter offset



if(set_AZ < act_AZ)


{


TurnCCW_AZ();


}


else {


TurnCW_AZ();


}
```

```
if(set_EL < act_EL)

{

TurnCCW_EL();

}

else {

TurnCW_EL();

}

//speedcontrol while reaching azimuth target

if ((abs(set_AZ-act_AZ))<20)

{

analogWrite(enA, 900);

}

if ((abs(set_AZ-act_AZ))<5)

{

analogWrite(enA, 800);
```

```
}

if ((abs(set_AZ-act_AZ))<1)

{

analogWrite(enA, 0);
Stop_AZ();

}

//speedcontrol while reaching elevation target

if ((abs(set_EL-act_EL))<20)

{

analogWrite(enB, 900);

}

if ((abs(set_EL-act_EL))<5)

{

analogWrite(enB, 800);

}

if ((abs(set_EL-act_EL))<1)
```

```
{

analogWrite(enB, 0);
Stop_EL();

}

}



// This function lets you control the azimuth motor;

void TurnCW_AZ(){

analogWrite(enA, 1023);

digitalWrite(in1, LOW);

digitalWrite(in2, HIGH);

}

void TurnCCW_AZ(){

analogWrite(enA, 1023);

digitalWrite(in1, HIGH);

digitalWrite(in2, LOW);
```

```
}

void Stop_AZ(){

// Turn off motors

digitalWrite(in1, LOW);

digitalWrite(in2, LOW);

}

// This function lets you control the elevation motor

void TurnCW_EL(){

analogWrite(enB, 1023);

digitalWrite(in3, LOW);

digitalWrite(in4, HIGH);

}

void TurnCCW_EL(){

analogWrite(enB, 1023);

digitalWrite(in3, HIGH);

digitalWrite(in4, LOW);
```

```
}

void Stop_EL(){

// Turn off motors

digitalWrite(in3, LOW);

digitalWrite(in4, LOW);

}
```