Pulse Width Modulation (PWM) is usually the most applied system to control RC motors (RC= Radio Control). PWM Controller as the PCA9685 which has 16-channel of 12-bit PWM with I2C-bus is the one used by Freenove Smart 4WD Card.

***How FREENOVE software determines the use of each channel*:**

The PCA9685 has an external clock input pin that will accept user-supplied clock

(50 MHz max.) in place of the internal 25 MHz oscillator for the PWM control. The frequency used for PWM control in the PCA9685 is adjustable from about 24 Hz to 1526 Hz. This allows the use of PCA9685 with external power supply controllers. All bits are set at the same frequency.

Freenove fix to 50 Hz (it means that each 20ms will be the period for refreshing the Ton and Toff for every channel).

Two types of motors which use the Ton for different uses:

* Speed Control: “Ton” is proportional to the RPMs
* Position Control (named this type of motor as “Servo”): “Ton” is proportional to a determine angle of the motor’s shaft.

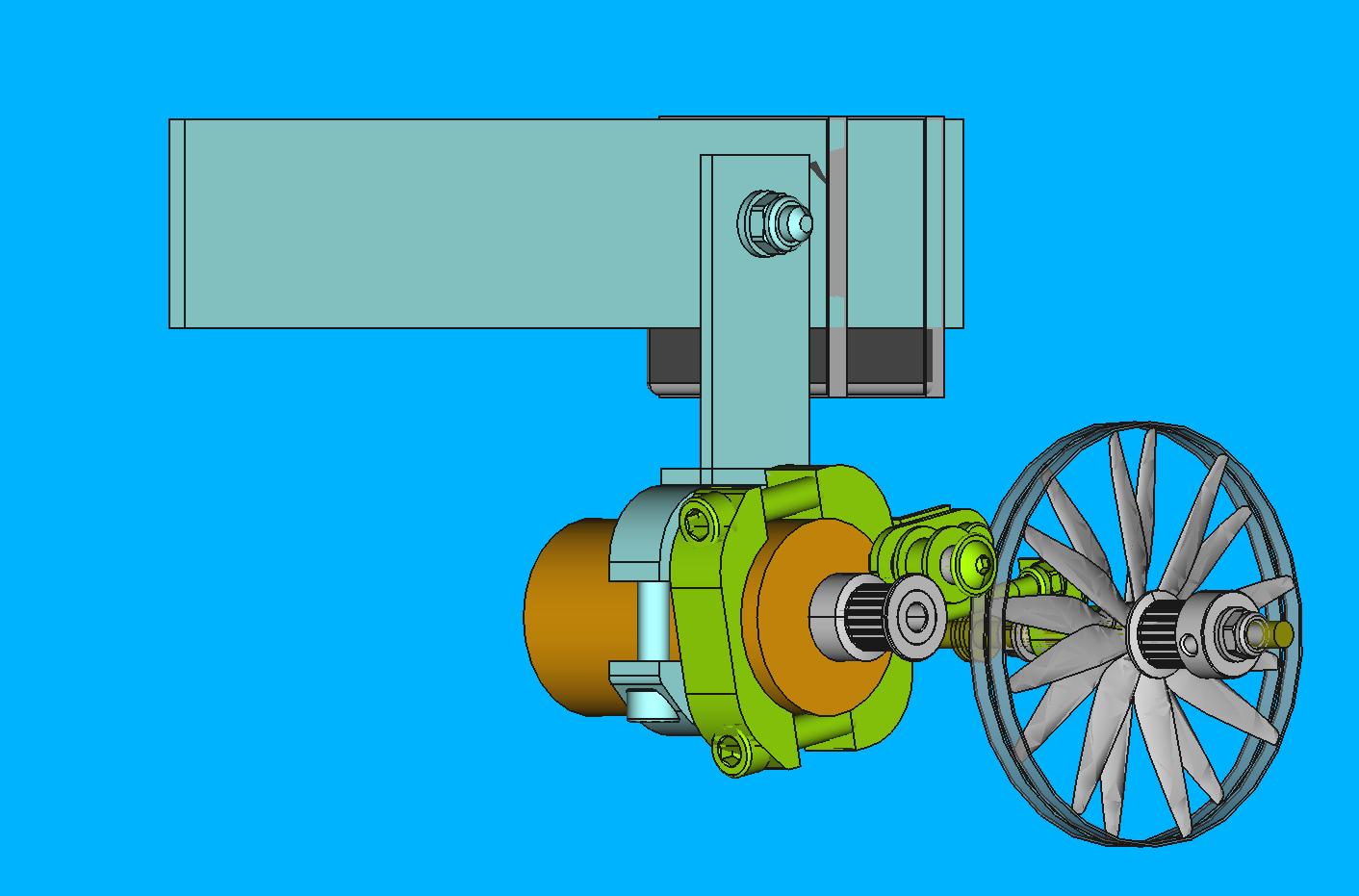
Freenove uses channel 0 to 7 for Speed Control and 8 to 15 for Position (this means it must be installed one type of motor or other regarding which channel is used.

Channel 0 and 7 control the speed of the wheels by combining the PCA9685 with the DRV8837C 1-A Low-Voltage H-Bridge Driver from Texas Instruments which use 2 PWM channels to determine if the motor turns CW or CCW (by activating one or other diagonal of the H-Bridge in a DC motor).

Channel 8 to 15 (which pins are easily accessing by the 8x3-pin connector behind the one used to connect to the RPi3) are used to control al motors named as Servo (Position Control). It uses only Channel 8 and 9 to control both motors move the Cam and Ultrasonic gauges.

***Adaptation of the Freenove Smart Card for the “Electronic Differential”*:**

In the new system adaptation, the prototype installs in each wheel and “Independent” DC-motor with its associated ESC (Electronic Speed Controller).



Motor is a 550 DC-Brushed and the “Ysido-ESC” is a 60A Electronic Speed Controller directly connected to a PWM.

The speed-control “*learn*s” the neutral, full-speed forward and full-speed reverse point of the radio system at the SETUP procedure.

For example, normally for RC cars, it could be (in case is fixed the refresh frequency to 50Hz or period of 20ms):

Neutral: Ton=T0ff=10ms

Forward: Ton>10ms

Reverse: Ton<10ms

Note: it could be as well like that

For the Car mode:

Neutral: Ton=5ms and T0ff=15ms

Forward: Ton>5ms

Reverse: Ton<5ms

So Maximum Forward speed could be more precise (3/4 of the 12-bit codes) and the Reverse would be only 2/3 smaller than the Forward.

For the Drone mode:

CW Neutral: Ton=0ms and T0ff=20ms

CW Forward: Ton>0ms

CW Reverse: Never

And

CCW Neutral: Ton=20ms and T0ff=0ms

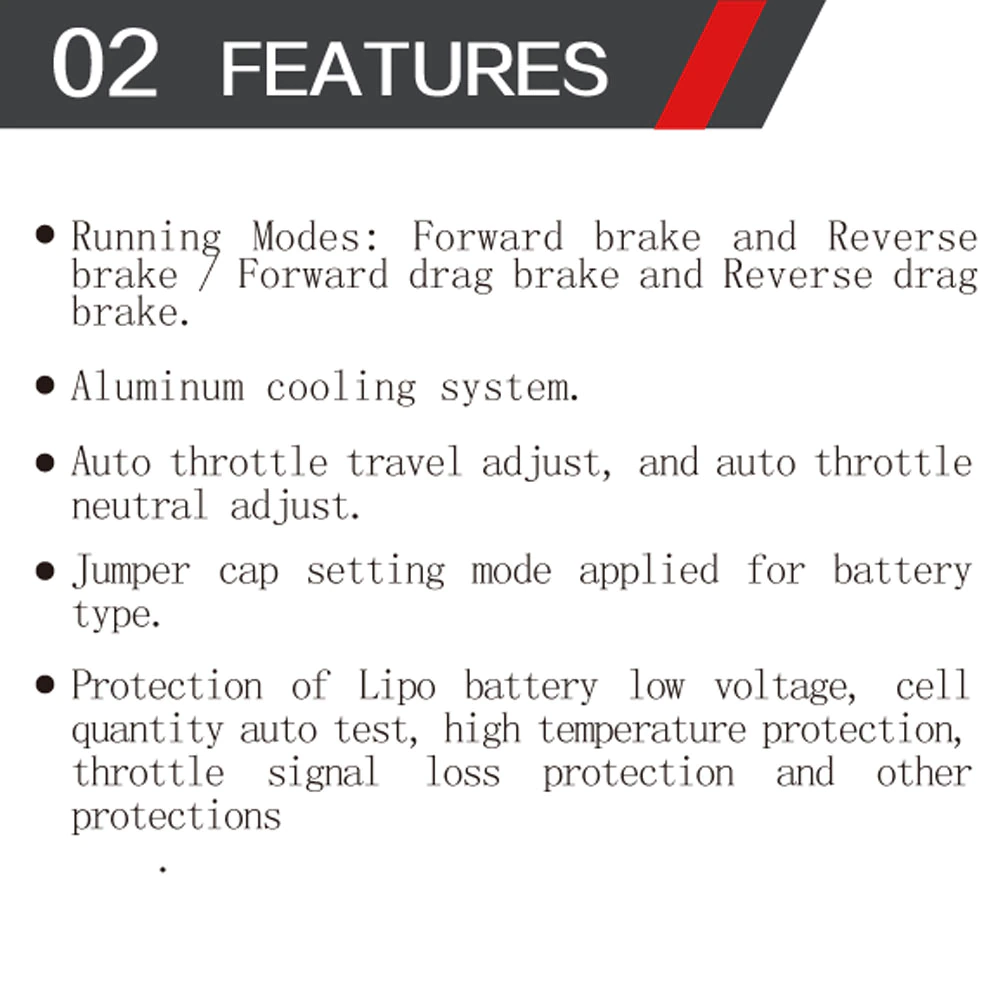
CCW Forward: Never

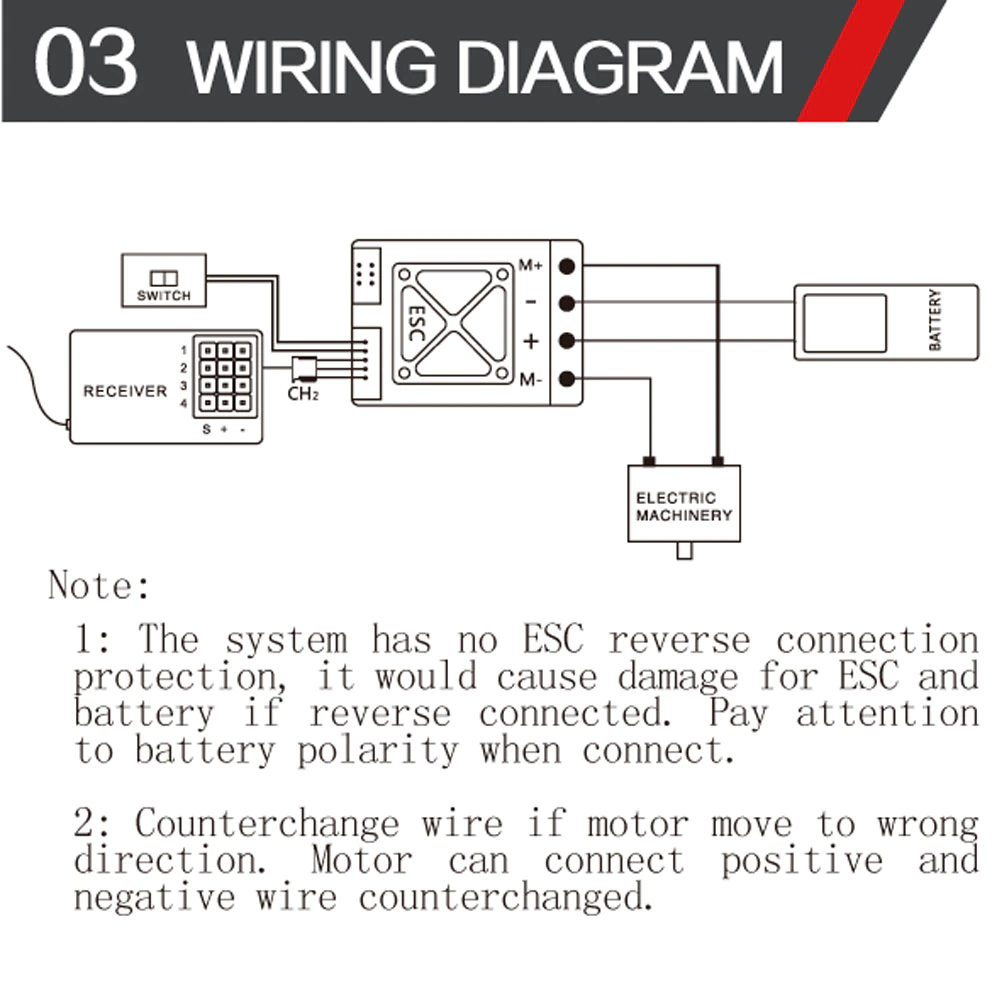
CCW Reverse: Ton < 10ms

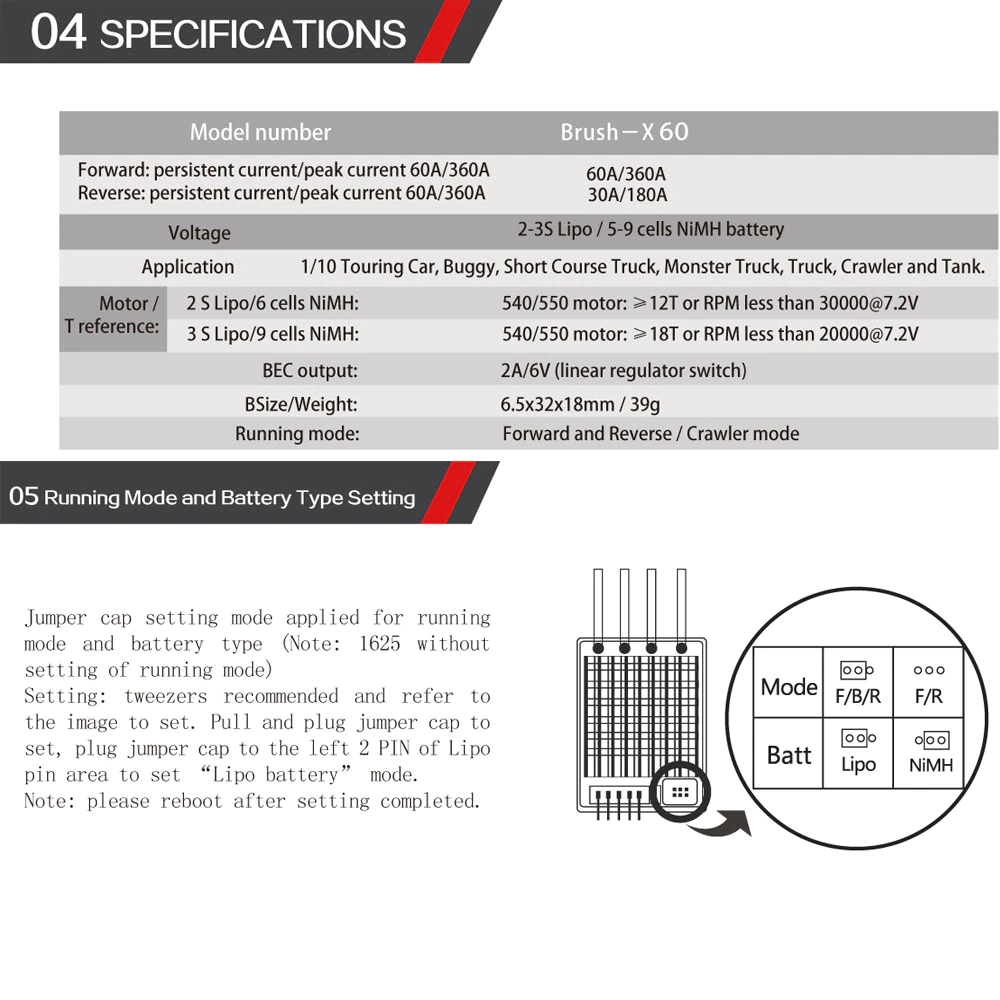
**In case the ESC has and Automatic-SETUP, it is adjusted automatically each time at power up the system while the car is running.**

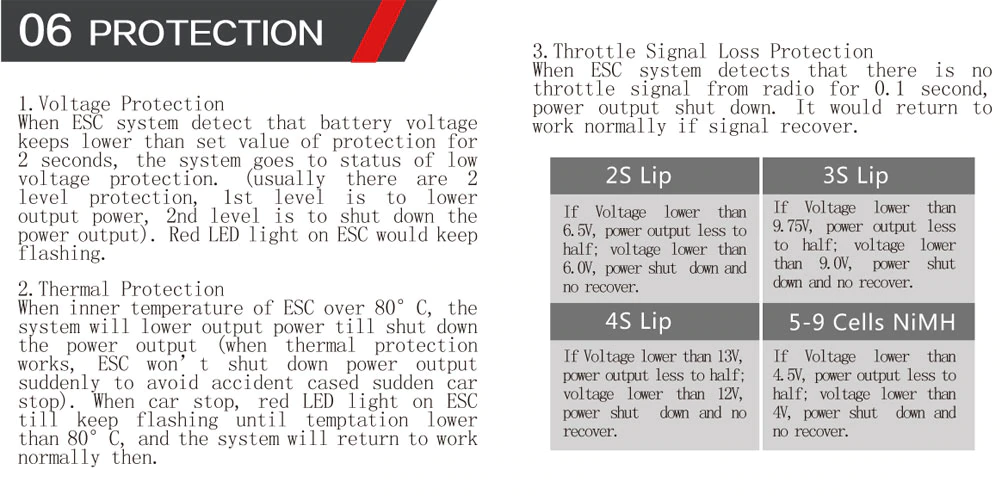
To do so, at MY\_PRGs directory, the “**test.py**” file has been changed to add the file “**My\_Motor.py**” that makes the same as the “motor.py” but with the PWM-channel 12 to 15 (and not from 0 to 7 as the “**Motor.py**” does).

**The *YSIDO-ESC* characteristics (it was bought to Surpass Hobby)**:









Description:

Protection of Lipo battery low voltage, cell quantity auto test, high temperature protection, throttle signal loss protection and other protections.

Features:

1. Running Modes: forward brake and reverse brake (F/R mode) / forward drag brake and reverse drag brake (Crawler or F/B/R mode).

 2. Auto throttle travel adjust, and auto throttle neutral adjust.

 3. Jumper cap setting mode applied for battery type.  
 Specifications:

Model number: Brush-X60

**Forward: 60A/360A**

**Reverse: 30A/180A**

 Voltage: 2-3s lipo/5-9 Cells NiMH Battery

 Application： 1/10 Touring Car, Buggy, Short-Course Truck, Monster Truck, Crawler & Tank

Motor Reference:

2s Lipo/6 cells NiMH: 540/550 motor: > 12T or RPM less than 30000@7.2V

3S Lipo/9 cells NiMH: 540/550 motor: >18T or RPM less than 20000@7.2V

BEC output: 2A/6V (linear regulator switch)

 Size/Weight: 6.5\*32\*18mm/39g

 Running mode: Forward and Reverse / Crawler mode

How our 550-motor is 25T (RPM 13,000@7.2V when motor is not loaded), battery used could be 2S (2\*3,6V) or 3S (3\*3,6V). We’ll apply 2S as the Freenove 4WD uses!

**MEASUREMENTS done with OSCILLOSCOPE and RPM meter**:

When “ReadyTosky” shows TURN Ton (ms) RPMs RPMs 1 Motor BATT’s Voltage

1260 (start moving) CCW 1,25 7,30

1300 CCW 1,3 300 534 7,3

1400 CCW 1,4 600 1047 6,96

1500 CCW 1,5 824 1225 6,87

1600 CCW 1,6 1010 1412 6,53

1700 CCW 1,7 1569

1300 (stop moving) 1,3

1150 (start moving) CW 1,15

800 CW 0,8 -990

1180 (stop moving) 1,3

As it was supposed RPMs are depending on how the Battery is used due to the supplied voltage for it depends of the consumption (Amps of the load). So, exposed “RPMs” are when the 4 motors are running instead of “RPMs with 1 Motor” that is when only one motor is activated (remaining 3 are not activated). Furthermore, the CHARGER is connected (8,5V || 6A) and is applied only 1 motor as well, the speed also increased. Measurement was done at the wheel side, that means the reduction of the pulley of 120 teethes to 16 teethes represent a reduction of 7.5:1!!! (So it means that with one motor only at maximum the RPMs at the motor’s side is: 1569\*7.5= 11,767.5 rpms. The minimum will be 300\*7.5=2,250 rpms.)

To understand better what happens with the supplied voltage by the battery (measured voltage at the “200A watt meter” side) :

1. All disconnected (neither ESCs nor Freenove 4WD) 7,66 V
2. When only the Freenove $WD is On 7,63 V (7,61-7,62 at start up)
3. Freenove 4WD + 1 ESC on (not running) 7,63 V
4. Freenove 4WD + 2 ESC on (not running) 7,63 V
5. Freenove 4WD + 3 ESC on (not running) 7,63-7,62 V
6. Freenove 4WD + 4 ESC on (not running) 7,62 V
7. Not Freenove 4WD off + 4 ESC on (not running) 7,64 V
8. PWM=1260 (start moving) Ton=1,25ms 7,30 V
9. PWM=1300 Ton=1,30 ms 7,3 V
10. PWM=1400 Ton=1,40ms 6,96 V
11. PWM=1500 Ton=1,50ms 6,87 V
12. PWM=1600 Ton=1,60ms 6,53 V

**NOTES about data and measurements with the YSISO-ESC**:

**By using the “ReadyTosky PWM”** (it generates the PWM’s wave from 800 to 2200 or 0,8ms to 2,2ms):

PWM=1260 (start moving) CCW Ton=1,25ms

1300 (stop moving) Ton=1,30ms

1150 (start moving) CW Ton=1,15ms

1180 (stop moving) Ton=1,18ms

The “Neutral window” could be consider to be: 1150 (1,15ms) < neutral < 1300 (1,30m)

When PWM> 1700 (1,7ms) the MAXIMUM SPEED is already achieved (and from 1700 to 2200 apparently neither happens to be! Nor Speed nor consumption increase.)

To change the turning direction something weird must be done:

* Anytime turning at CCW (PWM>1300), by moving the potentiometer of the “ReadyTosky PWM” to reduce the PWM’s value, it was detected that, achieved PWM=1300 it STOPs but when achieved PWM<=1150 the ESC shows the RED LED signals it should be moving CW but it doesn’t! To star moving in CW direction, It is necessary to make PWM=1150 and afterward to increase to PWM 1200 or 1250 and achieved this point, to reduce again the PWM value and, then, the motor will always start moving in CW!!!!

From the fact that:

CCW start at PWM= 1300 and when PWM > 1700 speed doesn’t increase

And

CW start at PWM 1150 and when PWM < 800 Speed is greater than 1000 RPM

Make us suppose that

**CCW 1300 (1,30ms ) < PWM < 1700 (1,70ms)**

**Neutral 1150 (1,15ms) < PWM < 1300 (1,30ms)**

**CCW 800 (0,80ms ) < PWM < 1150 (1,15ms)**

It makes to change a lot how the system runs in this case in front of the original Freenove 4WD behavior.

NOTE: The original Freenove 4WD behavior is a as follows

\* The PWM chip PCA9685 is fixed to 50 hz (period of 20 ms)

\* It uses PWM channels for each motor. For example, for one single axis, PWM0 to move CCW and PWM1 to move CW (it switches the transistor’s diagonal of the motor by using one or the other PWM) and Motor Speed come from Ton=0 ms (motor is stopped) to 20 ms (maximum speed) in both PWMs.

**For the YSIDO ESC we need to use only ONE single PWM for each motor.**

**Frequency could be fixed to 2 ms (500 Hz) because it will be more than enough! (Anyhow, it will be fixed to 250Hz, that is a period of 4 ms).**

**The behavior of the “Speed Control” looks like that:**

**CW Neutral CCW**

**0 0,80 1,15 1,30 1,70 2,00ms**

**\_\_\_\_\_\_\_\_|--------|===|---------------|\_\_\_\_**

**Which represents to have a useful zone of only 0,9ms where 0,15ms of them are Neutral and 0,4ms for CCW and 0,35ms for CW (???!!!).**

**Measurements by using the “Freenove 4WD” and the RPi3**:

**By fixing the Frequency to 250 hz (or 4 ms Period**), it generates a PWM’s wave from 0 to 4095 or 0ms to 4ms.

It was clear that only the Pulse’s Width of the PWM signal supplied toward the Ysido ESC is:

CW Neutral CCW

0 0,75 1,25 1,30 2,10 4,00ms

\_\_\_\_\_\_\_\_|**--------**|===|**---------------**|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* When Ton < 0,75ms is stop moving BACKWARD!
* When Ton < 2,10ms is stop moving FORWARD!

(Note: if Frequency is fixed to **2ms, it doesn’t work the ESC!**)

So, the only ***profitable window is from 0,75ms to 2,10ms***, that is: only 1,35ms of the available 4ms (!!!!????)

Furthermore, the behavior of the ESC is quite “weird”. Despite always is working fine when moving forward, no so to move backward… The “continuous” movement when you come from 2,10ms to 0,75ms, the ESC insides in the NEUTRAL window (it is stopped). By reducing “continuously” the ESC doesn’t MOVE backward and it is needed to make a quite odd “tricky” that is (take a look in the “test.py” and the “test\_My\_Motor()” function by executing “sudo pyton test.py My\_Motor”):

from My\_Motor import \*

my\_pwm=My\_Motor()

def **test\_My\_Motor()**:

try:

my\_pwm.setMy\_MotorFreq(250) # Set **f=250hz (T=4ms**)

for i in range(0,100,1):

my\_pwm.setMy\_MotorModel(1300,1300,1300,1300) #Neutral + (stand still)

time.sleep(0.005)

while True:

for i in range(1300,2100,1):

# for i in range(1300,1350,1): #también funciona con este valor tan pequeño

my\_pwm.setMy\_MotorModel(i,i,i,i) #Forward ++ (CW)

time.sleep(0.005)

for i in range(2099,**1250**,-1):

# for i in range(1349,**1250**,-1): #también funciona con este valor tan pequeño

my\_pwm.setMy\_MotorModel(i,i,i,i) #Forward -- (CW) & Neutral (stand still)

time.sleep(0.005)

for i in range(**1300**,750,-1): #SI NO SE PONE VAR MAYOR A 1250 DEL ccw, NO FUNCIONA EL cw.

# for i in range(1300,1250,-1): #también funciona con este valor tan pequeño

my\_pwm.setMy\_MotorModel(i,i,i,i) #Backward ++ (CCW)

time.sleep(0.005)

for i in range(751,1249,1):

# for i in range(1250,1249,1): #también funciona con este valor tan pequeño

my\_pwm.setMy\_MotorModel(i,i,i,i) #Backward -- (CCW) & Neutral

time.sleep(0.005)

except KeyboardInterrupt:

my\_pwm.setMy\_MotorModel(0,0,0,0)

print ("\nEnd of program")

Where it can be seen the following in order the movement looks like continuous:

* Should be applied some “steps”
  + Move forward 1300 to 2100 accelerating. (Never more than 2100 for avoiding to stop moving).
  + Move forward 2099 to 1250 deaccelerating. (See than the move continues up to 1250).
  + Move Backward 1300 to 750 accelerating. (Starts from 1300 and not from 1250 where was stopped)
  + Move Backward 749 to 1299 deaccelerating. (And Loop!)
* As it can be seen it stops moving Forward at 1250 but… in order to move Backward, it is not possible to start from 1249 (that would be the appropriate but in this case the motor never never never start moving, though the ESC Led is flashing like if it is moving [!!!!????]), the Ton MUST BE HIGHER THAN the value where it have achieved the Forward’s stop!!!!
* Only 800 (=0,8ms) are used to move forward. From 1300 to 2100 (from Ton 1,3ms to 2,1ms)
* Only 550 (=0,55ms) are used to move Backward. From 1300 to 750 (from Ton 0,75 to 1,30ms).
* Even still much weirder! It is important in the Initial Switch ON what is the value of the PWM (Ton):
  + If before execute the program “sudo python test.py My\_Motor” (or “sh Str\_DSW.sh”), it remains the PWM at 1300 (or 1250) that is the NEUTRAL (Stopped), when it is tried to move Backward, it doesn’t work!!!
  + Is very important for the “sudo python test.py My\_Motor” the ESC detect no PWM signal is detected (Ton=0) where its Red LED flash slowly signaling something is wrong. When the “test.py My\_Motor” starts, it sends Ton=1,3ms and the ESC recognizes PWM is OK and when the cycle of moving Forward and Backward begins, the ESC moves the motor Forward and Backward perfectly (!!!!!!!?????).

**MEASUREMENTS with RPM meter and 3.5” propeller**:

Without the charger and 2 motors on: RPMs|max=10,900/2= 5,450 RPMs

With the charger and 2 motors on: RPMs|max=12,065/2= 6.032,5 RPMs

With the charger and only 1 motor on: RPMs|max=19,307/2= **9.653,5 RPMs**

With charger and 4 motor (ESCs limited by voltage protection): RPMs|max=10,855/2= **5,425.5 RPMs**

**PCA9686.py modification for period equal to 4 ms (250Hz)**:

In order the Servo-Motors achieve the correct functioning, the definition of the function

**def setServoPulse(self, channel, pulse)**

must be changed in order to modularize to 4000 us (4 ms) instead of 20000us (20 ms) by doing the following simple change and the end of the file PCA9685.py:

pulse = pulse\*4096/~~20000~~ to **pulse = pulse\*4096/4000**

That supposed to change as well all things related with the SERVER and so

1. the files still remain at the directory *~~…/Code/Server~~* (**main.py, serve.py, server\_ui.py and Thread.py**) must be located at the directory ***…/Code/MY\_PRGs*** in order to execute the SERVER, by executing the main.py, from the appropriate directory to apply all the modification has been introduced by the ***BiSCOOTER project***.
2. To fix the frequency to 250 Hz (instead the original 50Hz) when the SERVER starts (and because it doesn’t starts the My\_DSW.py), at the files Motor.py, servo.py and My\_Motor.py, the

self.pwm-selfPWMFreq(~~50~~) must be changed to self.pwm-selfPWMFreq(**250**)

That obey to modify as well the file Str\_STR.sh to access “MY\_PRGs directory” instead of the “Server directory”.