Project 0: T-RECS: Build; Model; System Identification

Grant King

2/21/2021

Abstract

In this report we describe how the T-RECS was built, how it functions, and how experiments were used to identify the system and its parameters. This will be done to look at the open-loop system and understand how the T-RECS works.

1 Introduction

The Transportable Rotorcraft Electronic Control System (T-RECS) is an educational project kit that is designed to help students learn about system dynamics and feedback control. The T-RECS fully assembled can be seen in Figure 1. The T-RECS can be thought about like a quarter model of a quad-copter; this can be modeled as a arm with a pivot point at one end and a perpendicular force on the other, more on this in section 3. The T-RECS is controlled in this project by an Arduino Nano that is mounted to the side; more on this in section 4. Given what we know about the system, basic open-loop system experiments will be applied to validate the model found from section 3; more on this in section 5. But before Modeling and System Identification the T-RECS need to be built; this can be found in section 2.

2 Building the T-RECS

The T-RECS' main body is made up of laser cut wood that has notches for parts to fit into. These part are then glued together. The write up supplied

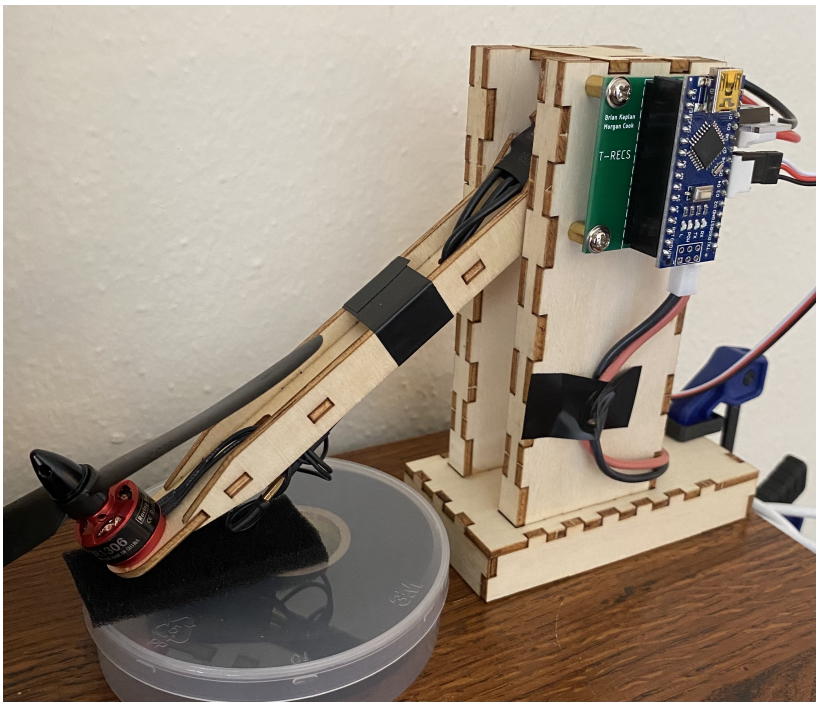


Figure 1: Fully built T-RECS system

with the T-RECS suggests superglue be used as the adhesive to hold the wood parts. As the superglue was not holding well, wood glue was substituted and held properly. The base was assembled first and let dry for 30 minutes.

The arm was assembled next and left to dry. The offsets need to be mounted before the towers are assembled otherwise it will become very difficult to place the nuts inside the towers after drying. The board does not need to be mounted yet. For this project and future projects, there is additional mounting hardware that was purchased separately. This will be used to later mount an encoder. These offsets will need to be placed as well. There is some conflict with the hardware interfering with the next step. There are bearings that are to be placed on the inside on the towers but the mounting offsets' nuts do not allow the bearing to fit. To alleviate this the top two nuts on the of the encoder side will be left off. To ensure they are not loose apply a small amount superglue to the threads of the offsets. The towers are assembled by

placing their bottom side into the base, placing the notching into the holes on the tower. This can be seen in Figure 2. The thin side supports are then glued into place. Make note of the fully assembled T-RECS; if the arm is thought of as the front, then the controller would be on the left side of the assembly and the encoder would be on the right. With the towers in place and glue dried, the bearing can be put into place. The arm is slid into place and the 3D printing shafts are placed into the square pegs holes on the arm. Make note that there is a shaft that has a flat side and this should be placed in the left side to later interface into the controller board. The power wiring and other hardware can be placed per the kit instructions. There are some steps that were not performed in the assembly, such as the top wood parts for the towers and the arm were not glued in place to allow access to the electronics and allow some level of disassembly for troubleshooting purposes.

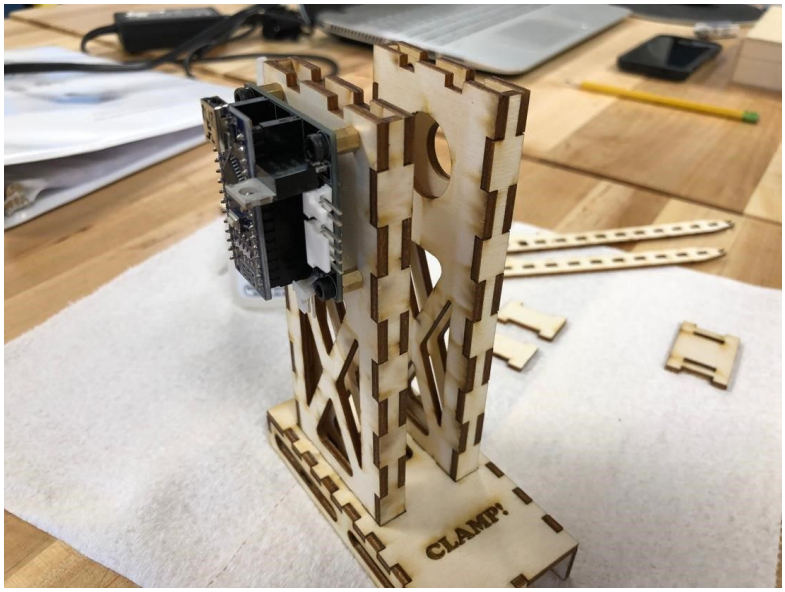


Figure 2: T-RECS towers glued in place [2]

3 Modeling

As said in section 1 the T-RECS can be thought about like a quarter model of a quad-copter. This can further be broken down into a free body diagram.

This diagram will be used to determine diffident components of the system and find equations that will actuary represent the motion of the system. The free body diagram can be seen in Figure 3. In this diagram there is the force F from the thrust of the propeller. There is the force due to gravity $Mg\cos(\theta)$ and a resulting force F_R. This can be reduced to the moments about a point; using the point where the arm rotates the diagram can be seen in Figure 4.

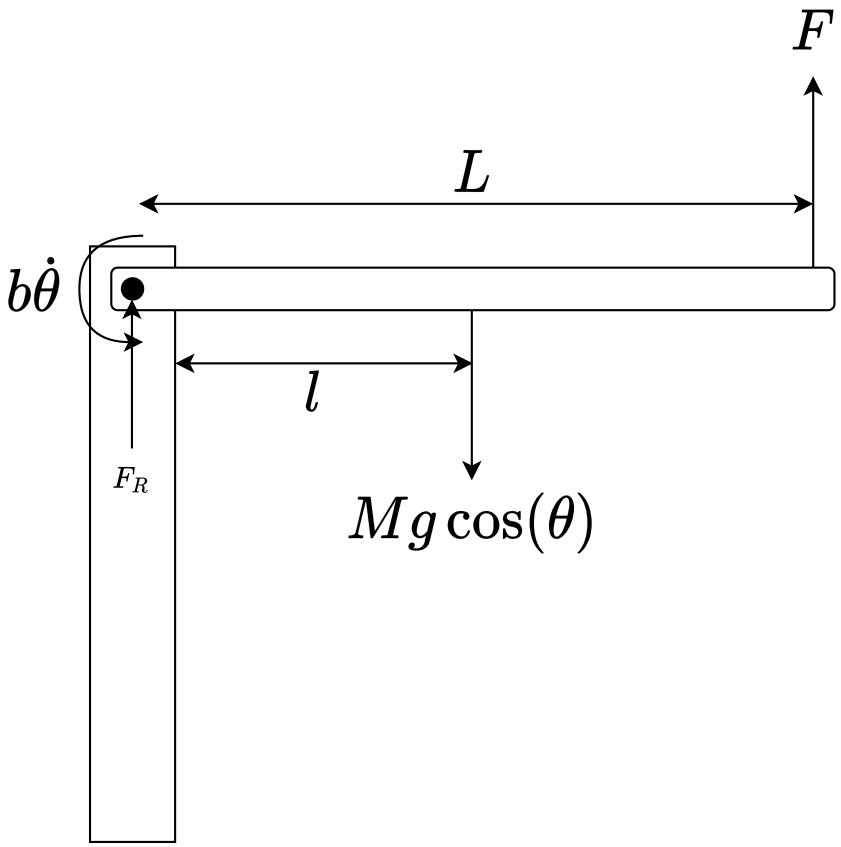


Figure 3: Free body diagram of the T-RECS arm

From the free body diagram the Equation 1 is found. where inertia component of the system is J; The mass of the system is M; this creates a moment due to gravity that changes due to the angle the arm is at (θ) and the length of the arm L; the damping due to drag, air resistance, or friction in the device is b.

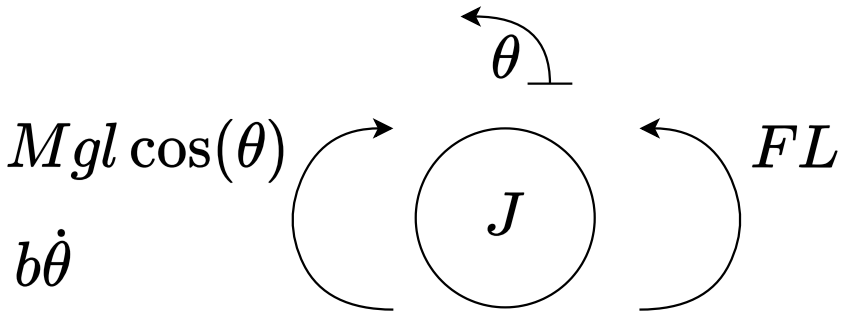


Figure 4: Collapsed Free body Diagram

$$J\ddot{\theta} = -b\dot{\theta} - Mgl\cos\theta + FL \tag{1}$$

Rearranging Equation 1 so that the angle (θ) and the force (F) are on opposite sides. This gives equation 2

$$J\ddot{\theta} + b\dot{\theta} + Mgl\cos\theta = FL \tag{2}$$

Dividing by inertia J gives the Equation 3.

$$\ddot{\theta} + \frac{b}{J}\dot{\theta} + \frac{Mgl}{J}\cos\theta = F\frac{L}{J} \tag{3}$$

F is equal to T. Where T represents the thrust. The Thrust also needs to be related to the input command. The input command (u) is a value between 0 and 1000. This value is proportional to the angular velocity (V_p) of the blade. So the input u times some scalar value B is equal to V_p. The derivation of the relationship between can be found in Appendix A.4. This gives the Equation 4. Where ρ is the is the air density, A is the propeller disk area.

$$Thrust = T = 2\rho A V_p^2 = 2\rho ABu^2 \tag{4}$$

Equation 4 can be substituted into Equation 3 giving Equation 5

$$\ddot{\theta} + \frac{b}{J}\dot{\theta} + \frac{Mgl}{J}\cos\theta = \frac{L}{J}2\rho ABu^2 \tag{5}$$

The constant can be lumped into single variables: $\frac{b}{J} = C_1$, $\frac{Mgl}{J} = C_2$, and $\frac{L}{J}2\rho AB = C_3$. This give Equation 6.

$$\ddot{\theta} + C_1\dot{\theta} + C_2 \cos\theta = C_3 u^2 \tag{6}$$

Equation 6 will be used later in determining the values of the coefficients (C_1, C_2, C_3). This equation can further be used to create the transfer function of the system by taking the Laplace transform of 6 and moving variables to be in the correct form. The $cos\theta$ can be assumed to be θ. This is due to the small angle theorem assuming small angle charges around the angle we are operating at; this gives 7.

$$\frac{\theta(s)}{T(s)} = G(s) = \frac{C_3}{s^2 + C_1 s + C_2} \tag{7}$$

This is the transfer function of the arm with the input of the thrust and an output of the angle of the arm. Equation 7 can be considered the plant of the whole system. There are other considerations that we must keep in mind when thinking about the entire T-RECS.

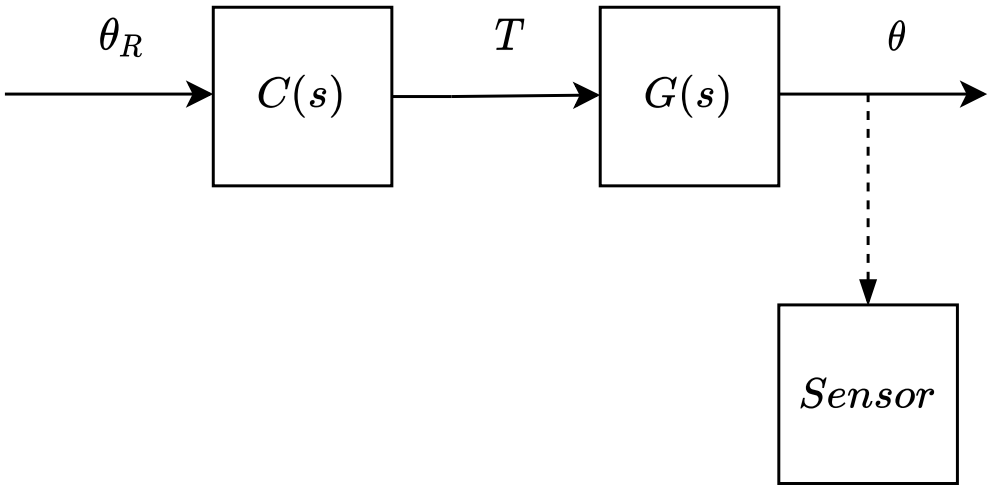


Figure 5: Block Diagram of the system

The input from the computer the would be the the desired angle (θ_R). That would be feed into a controller $(C(s))$ which would then take that reference and turn it into at thrust command (u) or T, this would then be sent to the motor. This and the other physical components make up the plant $(G(s))$. This thrust command is enacted and the arm will raise to an

angle (θ). The angle is then read by a sensor. This can be seen in a block diagram in Figure 5.

4 Hardware

The T-RECS kit comes with a components list that can be found in Appendix A.1. Here we will look at the most important hardware components and their function in the T-RECS. Those components are:

- Arduino Nano
- 1360 Brushless DC Motor
- Electronic Speed Control (ESC)
- Potentiometer and the Printed Circuit Board (PCB)

The Arduino Nano is a very common single-board microcontroller that is programmed using Arduino integrated development environment (IDE). The board can be powered from the micro-USB cable, an unregulated external power supply (6-20 Volts on pin 30), or regulated external power supply (5 Volts on pin 27). For this project the Nano will be powered through USB. The Arduino Nano pin out sheet can be found in the Appendix A.5. It reads the voltage of output pin as integer values between 0 and 1023. This can be use to map input voltages between 0 and the operating voltage. This will be useful to understand for future projects.

The brushless motor that comes with the T-RECS kit is a Racerstar Racing Edition 1306 BR1306 3100KV. This motor is rated for 7.4 Volts and 3100 rpm/V. While the motor's performance data does not have the propeller that comes with the T-RECS kit, the Efficiency ranges from 4.3 to 5 g/W. This and other motors like it are very common in RC quad-copters, helicopters, and airplanes motors. Brushless motors are controlled by pulses of current that come from an Electronic Speed Control (ESC).

The ESC that is used in this project is an ARRIS Swift 20A ESC. The ESC takes in a speed reference signal and converts that to a current to run the motor at a specific speed. The signal that it is taking in comes from the Arduino as a set speed command in the code on the Arduino. The input to the ESC is given in milliseconds from the Arduino. The ESC that we are

using can operate at a max of 20 Amps and at a burst current of 30 Amps for 10 seconds. The ESC supports a frequency signal up to 500Hz. The ESC can be further programmed, but that is outside the scope of this project. The parameters of the ESC can be found in Appendix A.6.

The Arduino is mounted on the side of the tower, and leads for the power and ESC connect to a pre-made printed circuit board (PCB). This PCB also has the potentiometer mounted where it can read the angle of the arm. This potentiometer is a Murata sv03A103AE01R00 Rotary Position Sensor. This sensor is used to relate the change in the output voltage of the sensor to the change in angle of the arm. This is done by the change in internal resistance of the sensor. These resistance changes are from the shaft as it moves a rotating contact in the sensor; this contact forms an adjustable voltage divider. This change in voltage is read by the Arduino and related to the angle. This sensor can be used in feedback control but for the purpose of this project it will only be used to read the angle of the T-RECS's arm.

The electrical schematic for the system can be seen in Figure 6. The T-RECS takes 12 Volts in through a barrel jack located in the base of the tower. This voltage runs into the ESC. The ESC controls the speed of the motor and receives control signals from the Arduino. The Arduino takes in information about the angle from the potentiometer and can adjust the output signal. For this project the Arduino will be sending commands to the ESC. These commands will be inputted from a computer from Arduino IDE's serial monitor by a user.

5 System Identification

Experiments were run to determine the coefficients of Equation 6. This is done to find a equation that can represent the plant of the T-RECS arm.

5.1 Drop Test

The first experiment was a drop test. In this test the arm was running the code seen in appendix A.2. This code reads the angle of the arm and allow for an input thrust command to the ESC. For this first test the thrust command is left at 0. The arm is raised to 0 degrees and then released to fall down to the table. A sponge was placed under the arm so it would not impact the table. This set up can be seen in Figure 7.

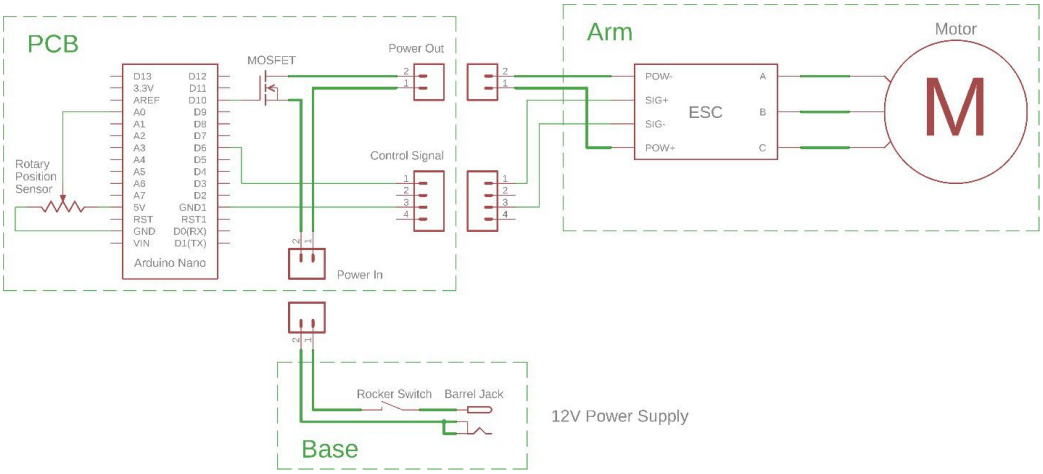


Figure 6: Electrical Schematic of T-RECS system [2]

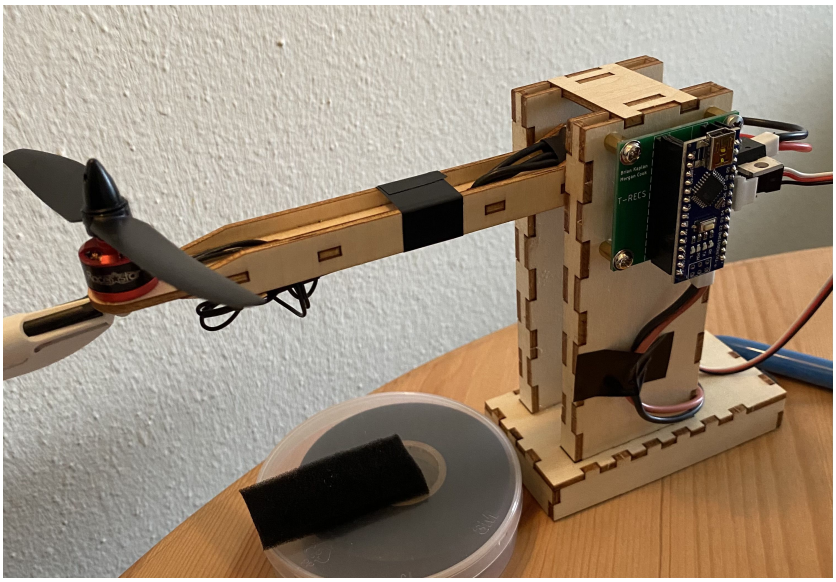


Figure 7: Drop Test set up

The data was recorded from the serial monitor, reduced to time of interest and loaded into MatLab and plotted, see Figure 8. Code used can be found in Appendix A.3. The test was run three times and averaged. This was used

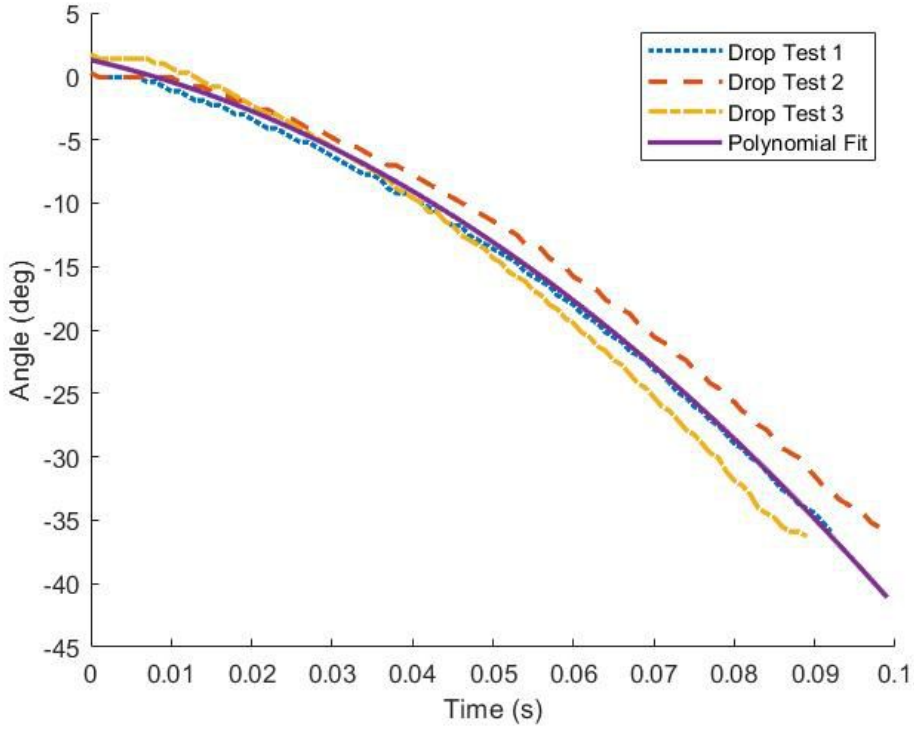


Figure 8: Drop Test: Time v. Angle

to create a polynomial fit, Equation 8.

$$\theta = -2871.95t^2 - 143.58t + 1.27741 \tag{8}$$

The initial condition of the drop test are: $u = 0, \theta(0) = 0, \dot{\theta}(0) = 0$. Using these values in Equation 6 results in Equation 9.

$$\ddot{\theta} + C_2 = 0 \tag{9}$$

The second derivative of Equation 8 is Equation 10.

$$\ddot{\theta} = -5743.89 \tag{10}$$

This is substituted into Equation 9 enables us to solve for C_2. $C_2 = 5743.89$. With this Equation 6 becomes:

$$\ddot{\theta} + C_1\dot{\theta} + 5743.89\cos\theta = C_3u^2 \tag{11}$$

5.2 Pendulum Test

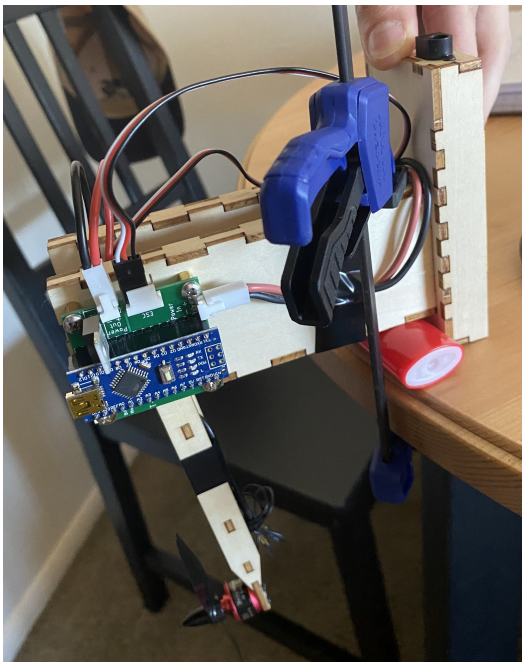


Figure 9: Pendulum Test Setup

For this test the T-RECS arm was placed on its side to allow it to swing like a pendulum. This set up can be seen in Figure 9. The arm was released from 90 degrees and the angle and time was recorded. The data was shifted so the the arm would seem like it was released 0 degrees. This was done so the equation of the system would not have to be changed.

Equation 11 was used with the thrust equal to zero. This equation was used to set up the a function in MatLab to solve the differential equation. From this different C_1 values were test. This can be seen in Figure 10. With this the value of the coefficient was determined to be around 2.5. This value is an estimation and could be further refined by testing more values of C_1.

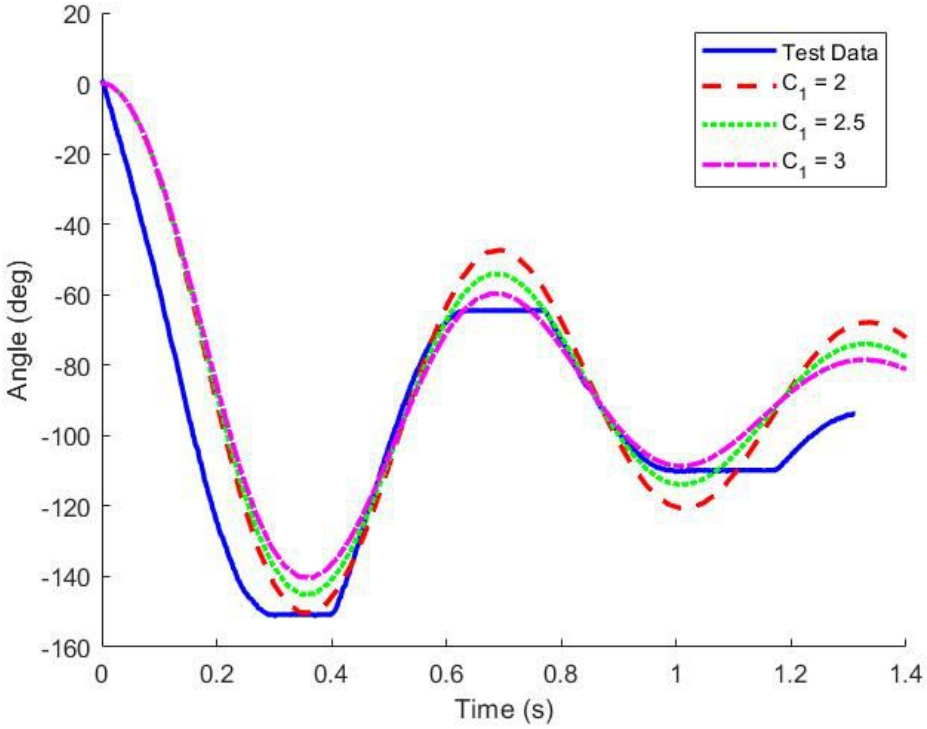


Figure 10: Pendulum Test: Time v. Angle

5.3 Thrust Test

For this experiment the same code as in section 5.1 was used, but this time instead of having a thrust input of 0, specific values were inputted. These inputs resulted in the arm hovering at a specific angle. This data was recorded and Matlab was used to plot the date in Figure 11.

A linear trend line was fitted and the thrust value at angle equal to 0 degrees was used to find C_3. The initial condition of this experiment are: $\ddot{\theta}(0) = 0, \dot{\theta}(0) = 0$. Using these in Equation 11 yields Equation 12.

$$5743.89 \cos\theta = C_3 u^2 \tag{12}$$

This can be rearranged to solve for C_3 in Equation 13.

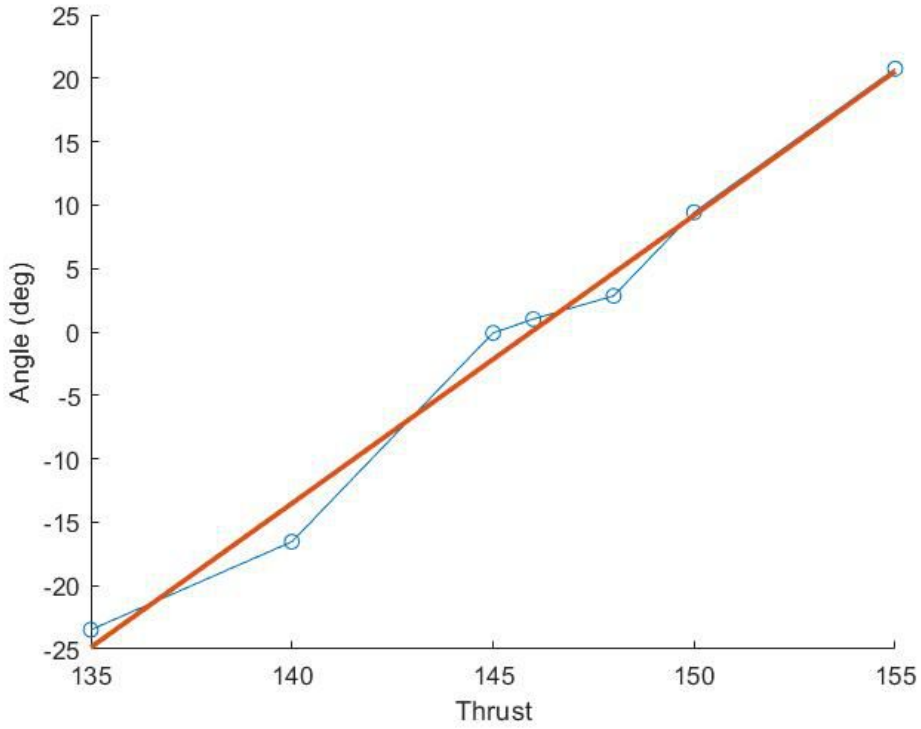


Figure 11: Thrust Test: Thrust v. Angle

$$\frac{5743.89\cos\theta}{u^2} = C_3 \tag{13}$$

From the trend line $2.27u - 331.39 = \theta$ at 0 degrees the thrust is 145.95. Using this in Equation 13 solves for $C_3 = 0.2696$ this gives Equation 14.

$$\ddot{\theta} + 2.5\dot{\theta} + 5743.89\cos\theta = 0.2696u^2 \tag{14}$$

This can be used to get the transfer function of the system as seen in Equation 15.

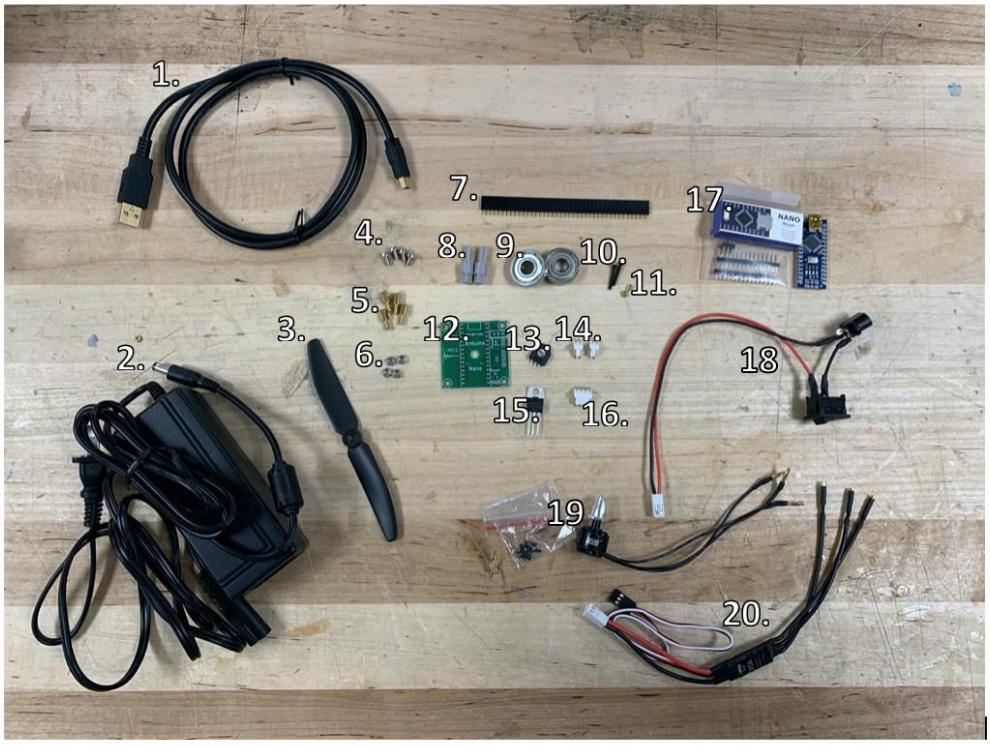
$$\frac{\theta(s)}{T(s)} = G(s) = \frac{0.2696}{s^2 + 2.5s + 5743.89} \tag{15}$$

6 Conclusion

This project gave the opportunity to learn about the Transportable Rotorcraft Electronic Control System. The system was built from the kit, then modeled, and from this model the parameters were identified. The hardware was also explored to gain a better understanding of the systems and to give insight in discerning what makes this system run. The transfer function was found and this will be of great use in later projects and in designing a feedback controller for the system in the future.

A Appendix

A.1 T-RECS: Components List



Components Guide:

1. 1x Programming USB cable
2. 1x Wall adapter power supply
3. 1x Propeller
4. 4x M3 screws
5. 4x M3 standoffs
6. 4x M3 nuts
7. 1x 40 pin female connectors
8. 2x shaft bearing inserts
9. 2x ball bearings
10. 2x M2 screws
11. 2x M2 nuts
12. 1x T-RECS PCB
13. 1x Rotary position sensor
14. 2x 2-pin receptacle
15. 1x MOSFET
16. 1x 4-pin receptacle
17. 1x Arduino Nano
18. 1x Power harness
19. 1x 1306 brushless motor
20. 1x ESC

Figure 12: Components list from [2].

A.2 Arduino Code

```
// Simple program to test the Arduino's
// ability to control ESCs/Brushless Motors

#include "Servo.h"

#define ESC_PIN         (6)   // PWM pin for signaling ESC
#define DRIVE_PIN       (10)  // Drive pin for power MOSFET
#define SENSOR_PIN      (A0)

int sensorVal;

Servo ESC;
int speed = 0;

void arm(){
  Serial.print("Arming ESC... ");
  digitalWrite(DRIVE_PIN, LOW);   // Disconnect ESC from power
  delay(500);                     // Wait 500ms for ESC to power down
  setSpeed(0);                    // Set speed to 0
  digitalWrite(DRIVE_PIN, HIGH);  // Reconnect ESC to power
  delay(2500);                    // 2.5 second delay for ESC to respond
  Serial.println("Arming complete");
}

/* Callibrate ESC's PWM range for first use */
void callibrate() {
  Serial.print("Calibrating ESC... ");
  digitalWrite(DRIVE_PIN, LOW);   // Disconnect ESC from power
  delay(500);                     // Wait 500ms
  setSpeed(1000);                 // Request full speed
  digitalWrite(DRIVE_PIN, HIGH);  // Reconnect ESC to power
  delay(5000);                    // Wait 5 seconds
  setSpeed(0);                    // Request 0 speed
  delay(8000);                    // Wait 8 seconds
  Serial.println("Calibration complete");
}
```

```
void setSpeed(int input_speed){
  //Sets servo positions to different speed
  int us = map(input_speed, 0, 1000, 1000, 2000);
  ESC.writeMicroseconds(us);
}

unsigned long myTime;

void setup() {
  // Initialize serial communication
  Serial.begin(115200);

  // Configure MOSFET drive pin
  pinMode(DRIVE_PIN, OUTPUT);
  digitalWrite(DRIVE_PIN, LOW);

  // Attach ESC to designated pin.
  ESC.attach(ESC_PIN);

  // Arm ESC
  arm();
}

void loop() {

  myTime = millis();

  Serial.print(myTime); //prints time since program started
  Serial.print(",");

  sensorVal = analogRead(SENSOR_PIN);
  Serial.println(-0.3656*sensorVal+185.64);

  if (Serial.available()) {
    // Check for calibration request
    if (Serial.peek() == 'c') {
      Serial.println("Callibrating ESC");
```

```
      callibrate();
      Serial.read();
    }
    // Otherwise, interpret as new throttle value
    else {
      speed = Serial.parseInt();
      Serial.println(speed);
      setSpeed(speed);
    }
  }
}
```

A.3 MatLab Code

```
%%%%%%%Drop Test%%%%%%%
clc
clear
close all

A = csvread('Drop_Data_1.csv');
B = csvread('Drop_Data_2.csv');
C = csvread('Drop_Data_3.csv');

Angle1 = A(:,2).';
Angle2 = B(:,2).';
Angle3 = C(:,2).';

n1 = length(Angle1)/1000;
n2 = length(Angle2)/1000;
n3 = length(Angle3)/1000;

Time1 = 0:.001:(n1-.001);
Time2 = 0:.001:(n2-.001);
Time3 = 0:.001:(n3-.001);

p1 = polyfit(Time1,Angle1,2);
p2 = polyfit(Time2,Angle2,2);
```

```
p3 = polyfit(Time3,Angle3,2);

p = [(mean([p1(1) p2(1) p3(1)]))
(mean([p1(2) p2(2) p3(2)])) (mean([p1(3) p2(3) p3(3)]))];


figure('Name','Drop Test Results')
hold on
plot(Time1,Angle1,':','LineWidth',2)
plot(Time2,Angle2,'--','LineWidth',2)
plot(Time3,Angle3,'-.','LineWidth',2)
fplot(@(x) polyval(p,x), [0 .099],'LineWidth',2)
ylabel('Angle (deg)')
xlabel('Time (s)')
legend('Drop Test 1','Drop Test 2','Drop Test 3','Polynomial Fit')

%%%%%%%Thrust Test%%%%%%%
clc
clear
close all

A = csvread('T_theta.csv');

Thrust = A(:,1).';
Angle = A(:,2).';

p = polyfit(Thrust,Angle,1);

p

figure('Name','Thrust Test Results')
hold on
plot(Thrust,Angle,'-o')
fplot(@(x) polyval(p,x), [135 155],'LineWidth',2)
ylabel('Angle (deg)')
xlabel('Thrust')
%legend('Drop Test 1','Drop Test 2','Drop Test 3',
'Polynomial Fit')
```

```
%%%%%%%Diff-equation%%%%%%%
function sdot = pentest(t,s)
C_1 = getGlobalx;
sdot = [s(2) ;
    -C_1*s(2) - 5743.89*cosd(s(1))];

%%%%%%%Pendulum Test%%%%%%%
clc
clear

A = csvread('penTest.csv');
t = A(:,1)/1000;
deg = A(:,2);

hold on
figure(1)
plot(t,deg,'b','LineWidth',2)

tspan = [0 1.4];
IC = [0 0];

setGlobalx(2);
[t2,stateV] = ode45('pentest',tspan,IC);
theta = stateV(:,1);
plot(t2,theta,'r--','LineWidth',2)

setGlobalx(2.5);
[t2,stateV] = ode45('pentest',tspan,IC);
theta = stateV(:,1);
plot(t2,theta,'g:','LineWidth',2)

setGlobalx(3);
[t2,stateV] = ode45('pentest',tspan,IC);
theta = stateV(:,1);
plot(t2,theta,'m-.','LineWidth',2)

ylabel('Angle (deg)')
```

```
xlabel('Time (s)')
legend('Test Data','C_1 = 2','C_1 = 2.5','C_1 = 3')
```

A.4 Thrust Equation

The base equations seen here were found in Mechanics and Thermodynamics of Propulsion [1]. While this relation is not needed if one just relates the impulse from the ESC in milliseconds to the angular velocity, these equations were used first in the the derivation of other equations.

A is area of the blades. ρ is air density. V_e is velocity of the exit stream. V_o is velocity of the incoming stream. V_p is velocity of the propeller blade. $\dot{m}_e$ in mass flow of the exit stream. $\dot{m}_o$ in mass flow of the incoming stream.

$$Thrust = F = \dot{m}_e V_e - \dot{m}_o V_o \tag{16}$$

$$V_p = .5(V_e - V_o) \tag{17}$$

If V_o is assumed to be far away then $V_o = 0$

$$V_p = .5(V_e) \tag{18}$$

$$F = \dot{m}_e V_e \tag{19}$$

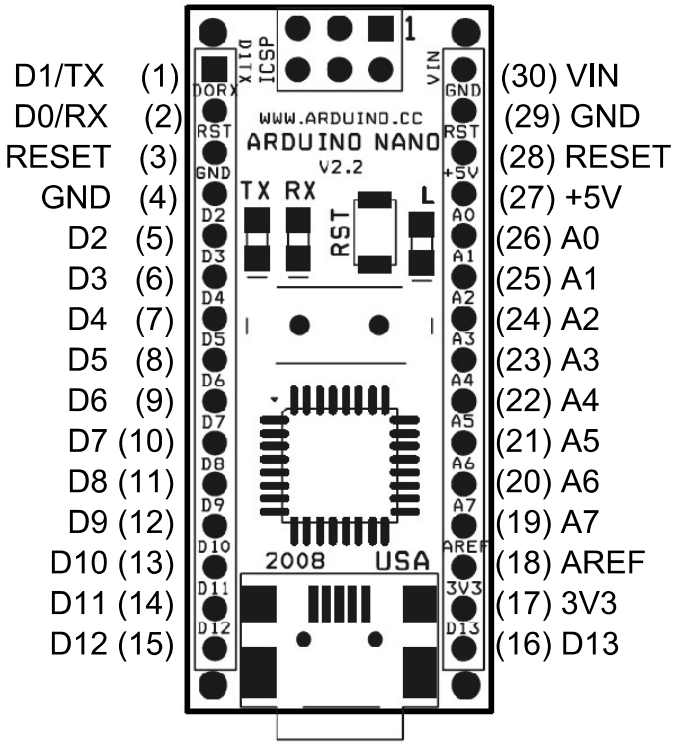
$$\dot{m}_e = \rho V_p A \tag{20}$$

$$F = \rho V_p A(2V_p) \tag{21}$$

$$F = 2\rho A V_p^2 \tag{22}$$

A.5 Arduino Nano pinout

Arduino Nano Pin Layout



Pin No.	Name	Type	Description
1-2, 5-16	D0-D13	I/O	Digital input/output port 0 to 13
3, 28	RESET	Input	Reset (active low)
4, 29	GND	PWR	Supply ground
17	3V3	Output	+3.3V output (from FTDI)
18	AREF	Input	ADC reference
19-26	A7-A0	Input	Analog input channel 0 to 7
27	+5V	Output or Input	+5V output (from on-board regulator) or +5V (input from external power supply)
30	VIN	PWR	Supply voltage

A.6 ESC Datasheet

User Manual
Multi-Rotor Brushless ESC

¸Swift ¢±Ï¢±¢ °ηk¸

Thank you for purchasing our brushless electronic speed controller (ESC) . Any Improper operation may cause personal injury damage to the product and related equipments. This high power system for RC model can be dangerous ,we strongly recommend reading the user manual carefully and completely. We will not assume any responsibility for any losses caused by unauthorized modifications to our product.

01 Main features

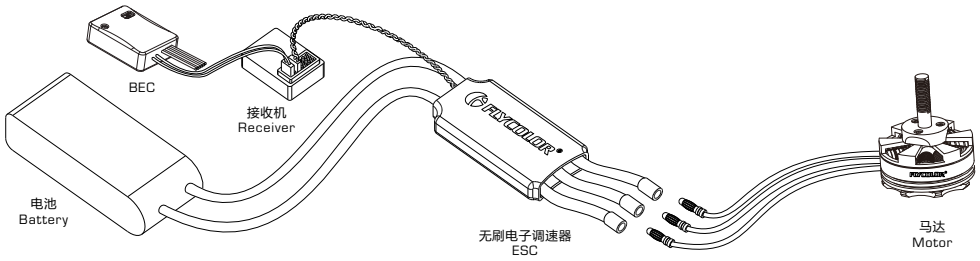
- High performance MCU.
- Mini size, lighter in weight.
- Optimized firmware is specialized for disc motor,excellent compatibility.
- The firmware is specialized for multi-rotor , fast throttle response during flying.
- Strong self-adaptable firmware, 8 timing options.
- Support frequency of throttle signal to 500Hz max , compatible with various kinds of flight control. (≥500Hz throttle signal is nonstandard signal)

02 Specification

Model	Con. Current	Burst Current (10S)	BEC	LiPo cells	Weight	Size (Excluding Plugs)	Typical Applications (For reference)
Swift -6A	6A	8A	5V/1A	2-4S	6.7g	28x13x5mm	200-250 Multi-Rotor
Swift -10A	10A	15A	5V/1A	2-4S	8.6g	28x15x6mm	200-280 Multi-Rotor
Swift -12A	12A	18A	5V/1A	2-4S	9.5g	28x15x6mm	200-330 Multi-Rotor
Swift -15A	15A	20A	5V/1A	2-4S	9.5g	28x15x6mm	250-450 Multi-Rotor
Swift -20A	20A	30A	5V/1A	2-4S	10g	28x15x6mm	330-550 Multi-Rotor
Swift -30A	30A	40A	5V/1A	2-4S	11g	28x15x6mm	330-650 Multi-Rotor
Swift -40A	40A	50A	No	2-6S	15g	40x21x7mm	450-850 Multi-Rotor
Swift -50A	50A	60A	No	2-6S	16g	40x21x7mm	650-1000 Multi-Rotor

03 Wiring diagram

Please ensure all solder joints are insulated with heat shrink where necessary.



*All pictures are for reference only

04 Programming parameter value

Programming parameters below in table that can be accessed from the remote control or configuration software (BLHeliSuite):

Function	1	2	3	4	5	6	7	8	9	10	11	12	13
1 - Closed loop P gain	0.13	0.17	0.25	0.38	0.50	0.75	1.00	1.5	2.0	3.0	4.0	6.0	8.0
2 - Closed loop I gain	0.13	0.17	0.25	0.38	0.50	0.75	1.00	1.5	2.0	3.0	4.0	6.0	8.0
3 - Closed loop mode	HiRange	MidRange	LoRange	Off	/	/	/	/	/	/	/	/	/
4 - Multi gain	0.75	0.88	1.00	1.12	1.25	/	/	/	/	/	/	/	/
5 - Startup power* *	0.031	0.047	0.063	0.094	0.125	0.188	0.25	0.38	0.50	0.75	1.00	1.25	1.50
6 - Commutation timing	Low	MediumLow	Medium	MediumHigh	High	/	/	/	/	/	/	/	/
7 - Pwm frequency	High	Low	*DampedLight	/	/	/	/	/	/	/	/	/	/
8 - Pwm dither	Off	7	15	31	63	/	/	/	/	/	/	/	/
9 - Demag compensation	Off	Low	High	/	/	/	/	/	/	/	/	/	/
10 - Rotation direction	Normal	Reversed	Bidirectional	/	/	/	/	/	/	/	/	/	/
11 - Input pwm polarity	Positive	Negative	/	/	/	/	/	/	/	/	/	/	/

Default values are marked in dark gray.
*:Only enabled for some ESCs. From code rev 14.4, damped light is default on the ESCs that support it. For prior code revisions, high is default.
* *: Default startup power varies by ESC. Generally the default power is lower for larger ESCs

1. Closed loop P gain sets the proportional gain for the rpm control loop. This setting controls the gain from speed error to motor power.
2. Closed loop I gain sets the integral gain for the rpm control loop. This setting controls the gain from integrated speed error (summed over time) to motor power.
3. Closed loop mode sets the range of speeds that the control loop can operate on.
   - For the high range, throttle values from 0% to 100% linearly correspond to rpm targets from 0 to 200000 electrical rpm
   - For the middle range, throttle values from 0% to 100% linearly correspond to rpm targets from 0 to 100000 electrical rpm
   - For the low range, throttle values from 0% to 100% linearly correspond to rpm targets from 0 to 50000 electrical rpm
   - When closed loop mode is set to off, the control loop is disabled.
4. Multi gain scales the power applied to the motor for a given input. Note that this is only for PWM input, for PPM input it has no effect. Beware that a low multi gain will also limit the maximum power to the motor.
5. Startup is always done with the direct startup method, which runs the motor using back emf detection from the very start. In this mode power is given by the throttle used, but limited to a maximum level. This maximum level can be controlled with the startup power parameter. Beware that setting startup power too high can cause excessive loading on ESC or motor!
6. Commutation timing can be adjusted in three steps. Low is about 00, mediumlow 80, medium 150, mediumhigh 230 and high 300. Typically a medium setting will work fine, but if the motor stutters it can be beneficial to change timing.
7. Pwm frequency:
   -High: High pwm frequency is around 20kHz.
   -Low: Low pwm frequency is around 8kHz.
   -Damped light : This mode adds loss to the motor for faster retardation. Damped light mode always uses high pwm frequency. This mode is only supported on some ESCs (where fet switching is sufficiently fast).
8. Pwm dither is a parameter that adds some variation to the motor pwm off cycle length. This can reduce problems (like throttle steps or vibration) in rpm regions where the pwm frequency is equal to harmonics of the motor commutation frequency, and it can reduce the step to full throttle. It is primarily beneficial when running damped light mode. Dither is not applied in closed loop mode.
9. Demag compensation is a feature to protect from motor stalls caused by long winding demagnetization time after commutation. The typical symptom is motor stop or stutter upon quick throttle increase, particularly when running at a low rpm. As described earlier, setting high commutation timing normally helps, but at the cost of efficiency. Generally, a higher value of the compensation parameter gives better protection. If demag compensation is set too high, maximum power can be somewhat reduced.
10. The rotation direction setting can be used to reverse motor rotation.
11. The input pwm polarity setting can be used to inverse the throttle behaviour. This is intended to be used with receivers that provide negative pwm. When using PPM input it must be set to positive.

Programming parameters that can only be accessed from configuration software (BLHeliSuite):
- Throttle minimum and maximum values for PPM input (will also be changed by doing a throttle calibration).
- Throttle center value for bidirectional operation with PPM.
- Beep strength, beacon strength and beacon delay.
- Programming by TX. If disabled, the TX can not be used to change parameter values (default is enabled).
- Thermal protection can be enabled or disabled (default is enabled).
  Temperature is above 140℃, motor power is limited to 75%;Above 145℃, motor power is limited to 50%;Above 150℃, motor power is limited to 25%.Above 155℃, motor power is limited to 0%.
- PWM input can be enabled or disabled (default is disabled). If disabled, only 1-2ms PPM and 125-250us OneShot125 are accepted as valid input signals.
- Power limiting for low RPMs can be enabled or disabled (default is enabled). Disabling it can be necessary in order to achieve full power on some low kV motors running on a low supply voltage. However, disabling it increases the risk of toasting motor or ESC.

References

[1] Philip G Hill and Carl R Peterson. Mechanics and thermodynamics of propulsion. 1992.

[2] TANGIBLES THAT TEACH. *Part 1 Instructions (Device Assembly and Testing)*, Accessed Feb 2021.