Embedded Computing Systems

Digital Control and Mecathronics

STM32 based flight controller

2019 / 2020

Author:
Edoardo Cittadini

Contents

1 Introduction

This project proposes an implementation of a flight controller based on STM32 boards, in particular all the shield based on Cortex M1 M3 M4 M7 following the traces of the most important, popular and used software in the industry in order to guarantee the same standards, settings and ways of acting on the quadcopter controller.

It allows expert user to act in a familiar way with the software and at the same time allows beginners to approach the very basic configuration in the same way as more advanced controllers.

The author during the writing of the software has taken care about beginners, so the firmware is equipped with some very specific and dedicated features to allow a progressive and secure flight experience.

The automatic position correction of the multirotor is based on a PID algorithm that uses data taken from the inertial measurement unit (IMU) as input reference and compares them with an appropriately scaled and dimensioned radio input provided by the operator acting on the radio sticks and with angle rates if the flight mode is settled to STABILIZED.

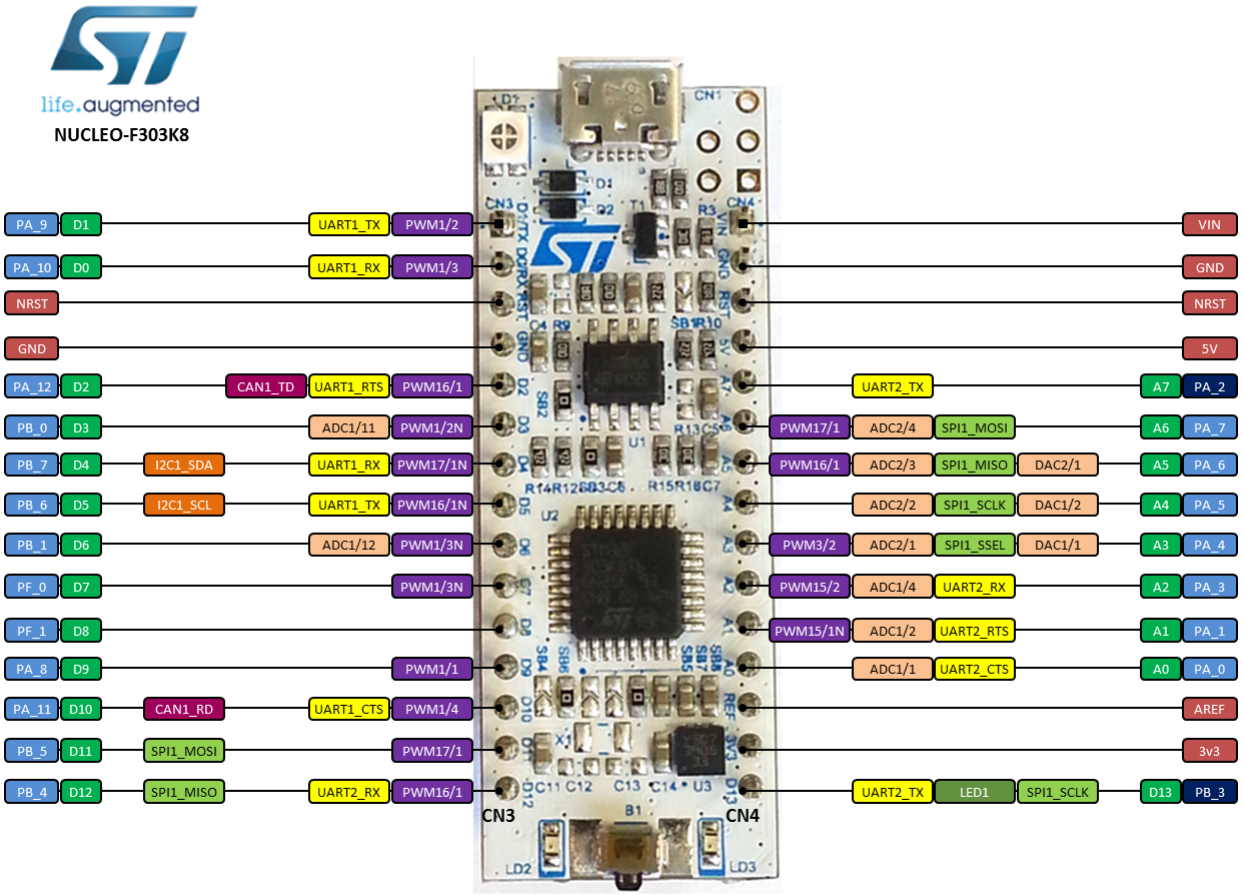
The value processed by the PID controller, that is the controller input given by the operator, is then normalized according to the specific required angular rates and the estimated value compared to the measured ones are processed to produce a reliable output that is then saved in the pulse array to be written on the timer Capture Compare Register (CCR) used to manage motors pulses.

Flight modes available are ACRO and STABILIZED; however ACRO mode is suggested only for expert pilots due to the absence of auto-level feature.

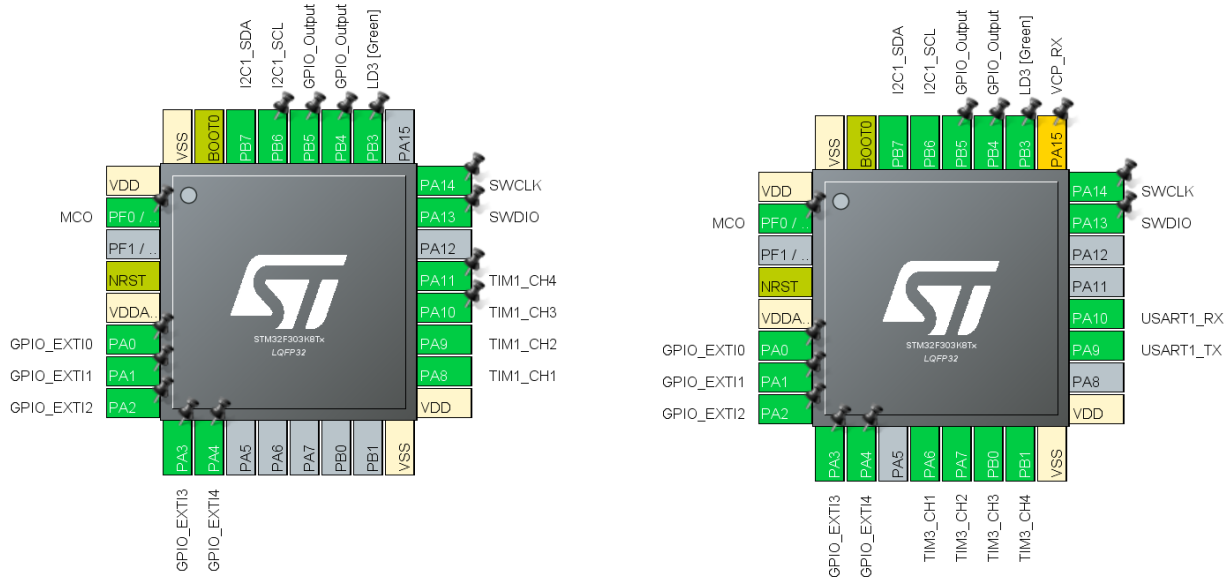
STABILIZED mode is much easier and allows to fly with bounded angles up to a maximum of $\pm$ 45°.

2 STM32F303K8T6 Board

STM32F303K8T6 is a little ST-Microelectronics 32-bit based MCU with an ARM Cortex M4 on board.
This board has an FPU (Floating Point Unit) that helps a lot improving the speedup of the whole system in particular during control logic math calculations such as gyroscope discrete integration and trigonometric functions for the accelerometer allowing a faster loop without any loss in precision.
This is possible because such unit provides a direct hardware support to the microprocessor with dedicated assembly instructions for floating point operations optimization.
This board belongs to the Nucleo-32 family that is the smallest board package of the Nucleo family (other possible configurations are Nucleo-64 or Nucleo-144).



Pin PA9 and PA10 are used for UART (Universal Asynchronous Receiver Transmitter) interface to make possible the communication with external devices such bluetooths or more powerful computational units like SBC (Single Board Computers).



The second version of the project does not use UART interface because its pin are used as PWM output in a more power balanced pinout configuration.

PB6 and PB7 are used for the I^2C (Inter Integrated Circuit) serial communication between the control unit and the IMU (Inertial Measurement Unit) in the middle speed configuration named fast-mode which speed is up to 400KHz.

PA0, PA1, PA2, PA3, PA4 are used for radio external interrupt signals; these pins are multiplexed in order to have different and independent interrupt lines.

PB4 and PB5 are used to drive two led to signal the user the calibration phase of the sensors and possible harmful runtime situations due to corrupted data or unreliable signals.

3 Radio and input processing

The radio receiver used is a 6-channel parallel radio receiver based on the AFHDS 2A protocol (Automatic Frequency Hopping Digital System).

Detecting PWM pulses can be efficiently done using one of the general purpose (GP) timers embedded in STM32 boards; a timer configured to do that has only to count with step increment of 1us and is called Free-running counter.

Using a free-running counter associated to external interrupts is the easiest and the most efficient way to do the job.
External interrupts can be triggered by rising_edge, falling_edge or both edges change and since the purpose is to identify the width of a radio pulse that is a digital square wave the mode used in the firmware is "pin change external interrupt on both edges".

```
void EXTI0_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI0_IRQn 0 */
  global_input_count = __HAL_TIM_GET_COUNTER(&htim2);
    if(CH_1_PREV_STATE == 0){
                CH_1_PREV_STATE = 1; //use the flag to have a trace to know the orientation of future edge
                counter_rising_edge = global_input_count;
        }
        else{
         if(CH_1_PREV_STATE == 1){
           CH_1_PREV_STATE = 0; //use the flag to have a trace to know the orientation of future edge
           counter_falling_edge = global_input_count;

           if( (counter_falling_edge - counter_rising_edge) < 0 ){
                 radio_channel_pulse[PITCH] = (counter_falling_edge - counter_rising_edge) + MAX_TIM2_VALUE;
           }
           else{
                 radio_channel_pulse[PITCH] = counter_falling_edge - counter_rising_edge;
           }
          }
     }
  /* USER CODE END EXTI0_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
  /* USER CODE BEGIN EXTI0_IRQn 1 */

  /* USER CODE END EXTI0_IRQn 1 */
}
```

Interrupt Service Routine (ISR) used is the same for all the 5 input radio channels; notice that CHx_PREV_STATE is used to take trace if the ISR call is going to be called on the rising edge or on the falling one.

On the falling edge, in order to obtain the final [1000us ; 2000us] pulse (Standardized PWM), two possible situation might happen:

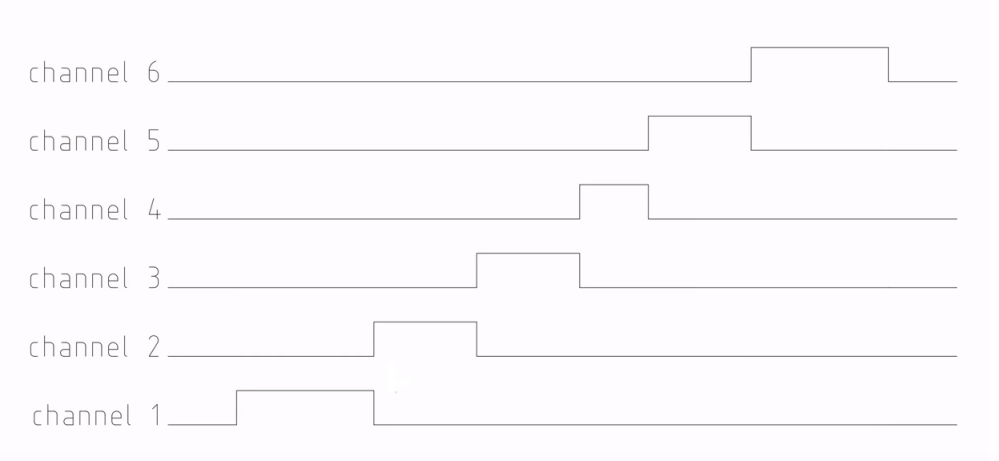
1. - Difference between the counter value on the falling edge and the one on the rising edge is positive, so final pulse can be directly obtained from this subtraction
2. - Difference between the counter value on the falling edge and the one on the rising edge is negative, so it means that the counter reached the value stored in the AutoReloadRegister (ARR) generating an overflow that makes it starts count again from zero.

In the second scenario an additional operation is needed to obtain the final pulse that is an addition of the maximum TIMx counter value, the same contained in the ARR, to the negative result of the difference between the counter value on the falling edge and the one on the rising edge.

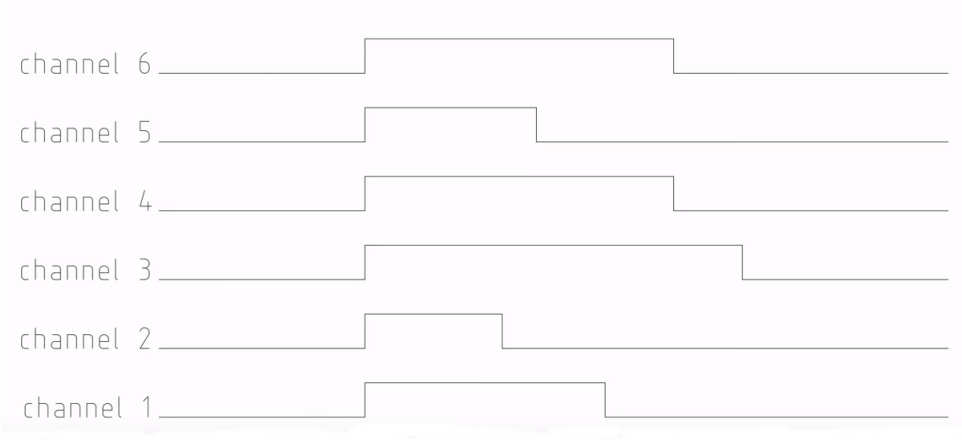
As shown in the ISR in the previous page and formally expained just above it can be noticed that this function is extremely fast and taking the advantage of the use of the timer is really light and computation-free from the MCU perspective, so this is very good because it means less runtime overhead.

Since timer is only used to take a time reference it can also be used to perform other indipendent operation such wave generation or Timer_interrupt generation on compare match using CaptureCompareRegister (CCR) or timer overflow on compare match with AutoReloadRegister (ARR).

Radio input signals depend on the receiver and can be produced either in serial, parallel or both. In this project five input parallel mode is used.
Depending on commercial receivers, PWM on different parallel channels can be generated in two ways as shown below:



The plot above represents the way in which some receiver work (generally AFHDS); it triggers channels in a waterfall fashion and because of that we need to know for each channel if a rising or a falling edge was detected.



Other receivers, such the one we use work in a different way; all the channels start with the same synchronous rising edge interrupt and the length of the pulse is retrieved by detecting the falling edge for each channel.

In order to do that we have to force one channel to be triggered both edges while the others only need to be triggered on the falling edge.

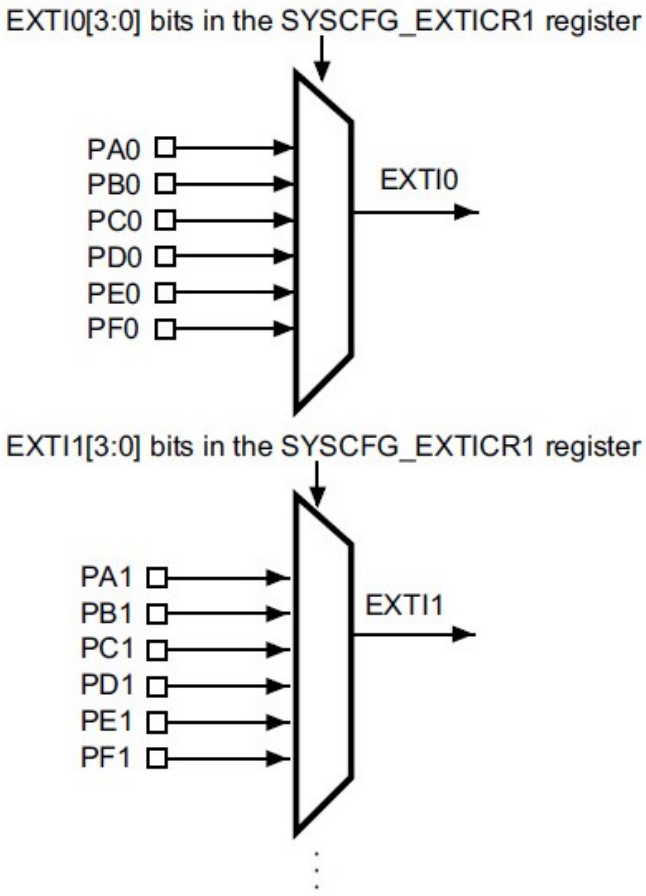
Driving PWM like that leads to another problem; since radio interrupts are settled with the highest priority because they must not have significant delays and they are no maskable, we must handle the situation in which two or more falling edge interrupts occurs in the same moment because one ISR must not be executed with delay otherwise it will result in unreliable data grater than the real one.

This problem is solved by checking if other pins changed state simultaneously with the one whose routine is in execution and if it is true the value calculated within the function is also assigned to the other pin and its logic state reference variable (CH_X_PREV_STATE where X is the channel number) is changed in order to forbid its pending interrupt routine to change its value. These observations lead to writing the following function for channel 1

```
void EXTI0_IRQHandler(void){
  global_input_count = __HAL_TIM_GET_COUNTER(&htim1);

          if(CH_1_PREV_STATE == 0 && HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_0) == SET){
                  CH_1_PREV_STATE = 1; //use the flag to have a trace to know the orientation of future edge
                  CH_2_PREV_STATE = 1; //use the flag to have a trace to know the orientation of future edge
                  CH_3_PREV_STATE = 1; //use the flag to have a trace to know the orientation of future edge
                  CH_4_PREV_STATE = 1; //use the flag to have a trace to know the orientation of future edge
                  CH_5_PREV_STATE = 1; //use the flag to have a trace to know the orientation of future edge
                  counter_rising_edge = global_input_count;
          }
          else{
              if(CH_1_PREV_STATE == 1){
                          CH_1_PREV_STATE = 0; //logic flag to trace edge orientation
                          counter_falling_edge = global_input_count;

                  if( (counter_falling_edge - counter_rising_edge) < 0 ){
                    radio_channel_pulse[ROLL] = (counter_falling_edge - counter_rising_edge) + MAX_TIM1_VALUE;
                  }
                  else{
                    radio_channel_pulse[ROLL] = counter_falling_edge - counter_rising_edge;
                  }
              }
              if( CH_2_PREV_STATE == 1 && HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_1) == RESET){
                      CH_2_PREV_STATE = 0;
                      radio_channel_pulse[PITCH] = radio_channel_pulse[ROLL];
              }
              if( CH_3_PREV_STATE == 1 && HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_2) == RESET){
                      CH_3_PREV_STATE = 0;
                      radio_channel_pulse[THROTTLE] = radio_channel_pulse[ROLL];

              }
              if( CH_4_PREV_STATE == 1 && HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_3) == RESET){
                      CH_4_PREV_STATE = 0;
                      radio_channel_pulse[YAW] = radio_channel_pulse[ROLL];
              }
              if( CH_5_PREV_STATE == 1 && HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_4) == RESET){
                      CH_5_PREV_STATE = 0;
                      radio_channel_pulse[SWITCH_MODE] = radio_channel_pulse[ROLL];
              }
          }
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
}
```

This is the longest and the most articulated since it drives all the rising edges and the falling edge of channel one. All the other ISRs only check for the falling edge and simultaneous activation that is the code inside the first else.

Every GPIO pin is connected to a multiplexer that handles the EXTI (External interrupt controller) as reported in the figure below



Every input unit takes as input all the pins of the GPIO ports with the same numeric index and because of that it is only possible to trigger external interrupts on pins that not share that so cited index.

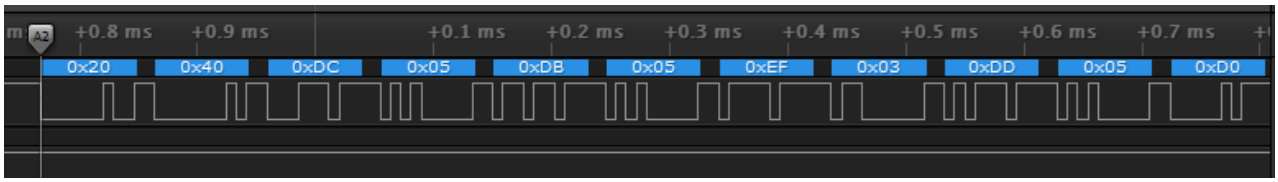
For example referring to the picture it is impossible to enable external interrupts both on PA0 and PB0 because they both refer to the same multiplexer so we are forced to use a pin with different numeric index like [PA0, PA1] but more in general any pin enabled except those belonging to position 0.

Unfortunately not all the EXTI multiplexers work in the same way, in fact if we enable external events/interrupts on lines 5, 6, 7, 8, 9 or 10, 12, 13, 14, 15 on the F303K8T6 we can notice that all these pins are handled by the same software handler function.

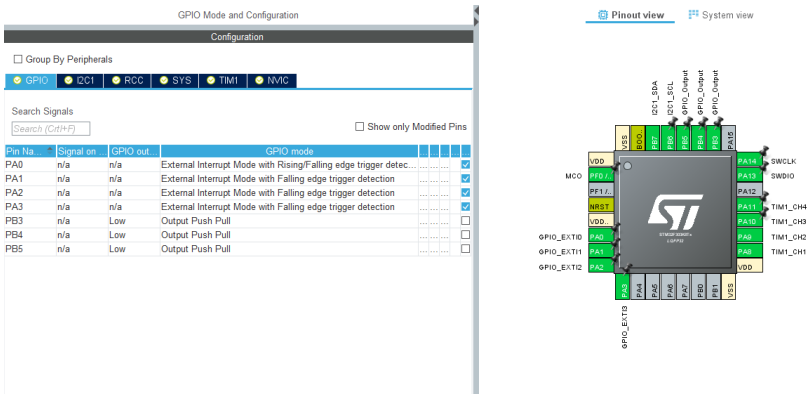
This situation has some drawbacks; in most of the cases the programmer has to perform some additional operations in order to know which pin changed state and consequently decode the event when ISR (Interrupt Service Routine) occurs generating a lot more complex code and a little more runtime overhead.

Considering what just said, for applications that need a wide range of radio channels, is more convenient and suggested the use of radio with native support for serial protocols such IBUS and SBUS that use up to 18 different channels instead of classic PPM (Pulse Position Modulation) which works up to 8 analog channels.

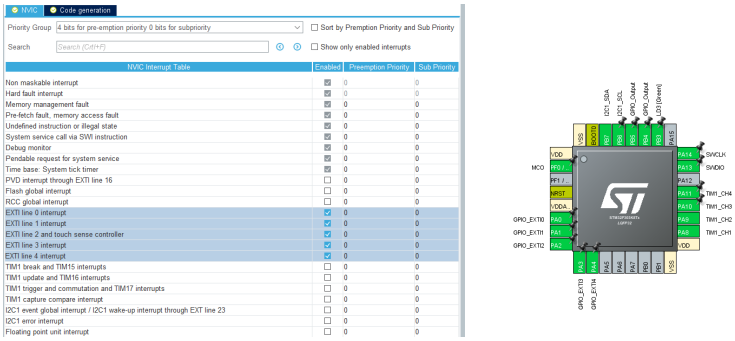
The advantage of PPM is that it can be used straight as it is; IBUS and SBUS require some hardware support that can be given both by the microcontroller or by the receiver in order to invert the UART signal.



In order to make external interrupt detection works we have to change the EXTI mode, that by default is only RISING_EDGE triggered, in the full mode that is RISING/FALLING_EDGE triggered for CH1 and FALLING_EDGE for all other channels.



We also have to enable the interrupt in the NVIC (Nested Vectored interrupt Controller) in order to make them effective at system startup.

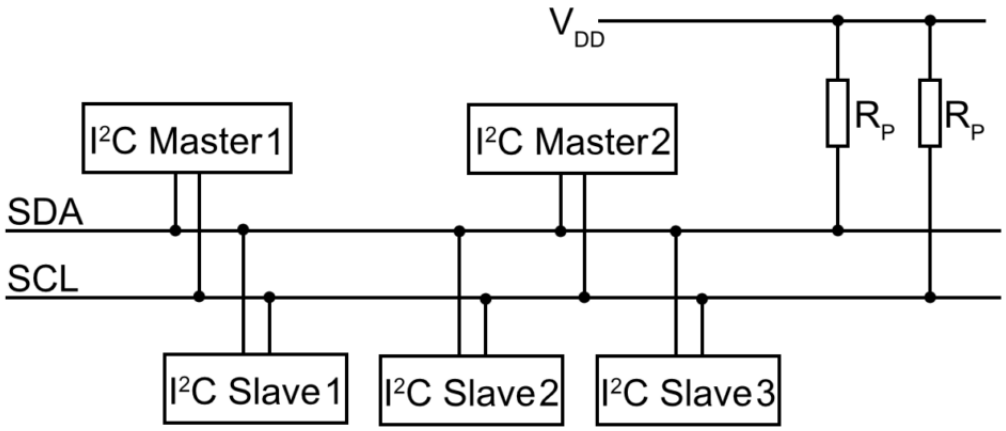


From the NVIC it is also possible to set the priority or the preemption level; this is very important for our project because we need read a counter to get the length of each radio pulse and any delay on that will result in an incorrect or a coarse grained approximation of the original signals. In order to avoid this situation we set for all EXTI pins the priority level 0 that is the highest level possible in the system and we also achieve the best possible accuracy.

4 I²C protocol

I²C (Inter-Integrated Circuit) is a synchronous, multi-master, multi-slave serial computer bus invented in 1982 by Philips Semiconductor nowadays NXP.
It is generally used for attaching lower-speed peripheral (discrete integrated circuits) to processors and microcontrollers within short-distance communication.

Compatible I²C devices become popular and started to appear in the market since the mid-1990s.



I²C uses only two bidirectional open collector or open drain lines, SDA (Serial Data line) and SCL (Serial CLock line), both pulled up with resistors. Typical voltages used are $5V$ or $3.3V$, although systems with other voltages are permitted.

The I²C reference design has a 7-bit address space, with a rarely-used (almost ever) 10-bit extension.

Common I²C bus speeds are 100 kbit/s in the so called standard mode and 400 kbit/s in the so called Fast mode. There is also a 10 kbit/s low-speed mode, but arbitrarily low clock frequencies are also allowed.

Since the protocol is still very widely used, improvements have been made and relatively recent revisions of I²C can host more nodes and run at faster speeds; in almost all nowadays MCUs we find:

1. 100 kbit/s Standard mode
2. 400 kbit/s Fast mode
3. 1 Mbit/s Fast mode plus
4. 3.4 Mbit/s High Speed mode
5. 5 Mbit/s Ultra Fast-mode

However these speeds are more widely used on embedded systems than on PCs.

It is important to notice that these speed are just theoretical measurement units in fact such bit rates are quoted for the transfers between master and slave without any clock stretching or other hardware overhead.
Protocol overheads include a slave address, register address in the slave device and the ACK/NACK (acknowledgment) bits for each byte.

Thus the actual transfer rate of data is much lower than those nominal values. For example, if each interaction with a slave inefficiently allows only 1 byte of data to be transferred, the data rate will be less than the half of the nominal bit rate value.

The number of nodes which can exist on a given I²C bus is limited by the address space and also by the total bus capacitance of 400 pF, which restricts practical communication distances to a few meters. The relatively high impedance and low noise immunity requires a common ground potential, which again restricts practical use to communication within the same PC board or small system of boards.

We just showed in the last picture that I²C, is an open collector/drain serial protocol and it depends on the technology used to make the microcontroller, in the case of F303K8T6 MOSFET are used so we can say that its I²C is composed by two open drain lines.

The IMU we are going to use, MPU_6050, has already on board the two pull-up resistor ($2.22K\Omega$) so is it not really mandatory to use also the resistors embedded in the MCU rather that it is suggested to disable them even if the system should work the same if they are not modified.

Another important information we can directly get from the datasheet of the sensor is the maximum interfacing speed on the bus as reported below

6.7 I^2C Timing Characterization

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

Parameters	Conditions	Min	Typical	Max	Units	Notes
I^2C TIMING	**I^2C FAST-MODE**					
f_{SCL}, SCL Clock Frequency				400	kHz	
$t_{HD.STA}$, (Repeated) START Condition Hold Time		0.6			µs	
t_{LOW}, SCL Low Period		1.3			µs	
t_{HIGH}, SCL High Period		0.6			µs	
$t_{SU.STA}$, Repeated START Condition Setup Time		0.6			µs	
$t_{HD.DAT}$, SDA Data Hold Time		0			µs	
$t_{SU.DAT}$, SDA Data Setup Time		100			ns	
t_r, SDA and SCL Rise Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
t_f, SDA and SCL Fall Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
$t_{SU.STO}$, STOP Condition Setup Time		0.6			µs	
t_{BUF}, Bus Free Time Between STOP and START Condition		1.3			µs	
C_b, Capacitive Load for each Bus Line			< 400		pF	
$t_{VD.DAT}$, Data Valid Time				0.9	µs	
$t_{VD.ACK}$, Data Valid Acknowledge Time				0.9	µs	

Note: Timing Characteristics apply to both Primary and Auxiliary I^2C Bus

We can observe that we can go up to 400 KHz, this means that we can use Fast mode to communicate with such device.

5 IMU MPU-6050

IMU is the short name for Inertial Measurement Unit and it is the hardware that allows, through the use of accelerometers, gyroscopes and other sensors to measure angular velocities, accelerations and other parameters along the three axes and therefore allows to know the position of the quadcopter on which it is installed whenever it is interrogated by transforming the raw data into meaningful values such accelerations and angular velocities.

The most used IMUs are those belonging to the 6 series; MPU-6000 and MPU-6050 . The MPU-6000 interfaces with the MCU via SPI (Serial Peripheral Interface); it is faster than MPU-6050 and it is also more sensitive to external interferences and noise. For this reason it is usually found in the commercial flight controllers already built-in as an integrated circuit in the same board of the microcontroller.

MPU-6050 is used in this project for 3 main reasons:

1. It is undoubtedly the most widely used inertial sensor
2. It costs just a few dollars and this allows to lower the total budget for the project and consequently allows a larger number of users to deal with or carry this project out
3. It communicates via I2C (Inter Integrated Circuits) therefore it requires only the two SCL and SDA lines plus power supply and ground and it is also more resistant to external interference so for a DIY flight controller this is a great advantage

I2C serial protocol is greatly implemented and abstracted by the HAL (Hardware Abstraction Layer) library that allows to interface in a very compact way with devices that use this protocol and in the case of this project with F303K8T6 board that only have one I2C interface we used pin PB6 and PB7.

This type of sensor provides 16-bit signed data, but is composed of 8-bit registers such that in order to obtain a single value, two contiguous registers must be read and combined with the left shift operator provided by the programming language («).

Furthermore, if sensor architecture is analyzed, it can be seen that all the relevant data registers (apart from temperature) are contiguous and this is reasonable because the sensor is able to return all the raw data in one loop starting from a given address with an operation called register burst. We will use burst readings because they minimize sensor accesses and allows us to be more efficient and therefore save CPU time which is essential to improve overall loop speed.

To be more precise we will not care about accelerometer and temperature data that are not involved in the rate/acro mode implementation. Accelerometer is used to correct gyroscope drift in stabilized mode because it is more reliable over long time but is more affected by vibrations and process noise both generated by propellers motion and electromagnetic fields.

When using sensors that have to return reliable data we have to provide and perform a calibration routine during setup; this is mandatory because all the sensors are intrinsically different and come out with small differences that must be smoothed out to make operations the same on all the platforms that will run the software minimizing differences.

What we need to know about this sensor before turning into the code is provided by the datasheet and the register map and in particular we are interested in all registers involving configuration and data interface not related to the DMP (Digital Motion Processor).

3 Register Map

The register map for the MPU-60X0 is listed below.

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0D	13	SELF_TEST_X	R/W	XA_TEST[4-2]			XG_TEST[4-0]				
0E	14	SELF_TEST_Y	R/W	YA_TEST[4-2]			YG_TEST[4-0]				
0F	15	SELF_TEST_Z	R/W	ZA_TEST[4-2]			ZG_TEST[4-0]				
10	16	SELF_TEST_A	R/W	RESERVED		XA_TEST[1-0]		YA_TEST[1-0]		ZA_TEST[1-0]	
19	25	SMPLRT_DIV	R/W	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	-	-	-	FS_SEL [1:0]		-	-	-
1C	28	ACCEL_CONFIG	R/W	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]				

InvenSense | MPU-6000/MPU-6050 Register Map and Descriptions | Document Number: RM-MPU-6000A-00
Revision: 4.2
Release Date: 08/19/2013

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO_RESET	ACCEL_RESET	TEMP_RESET
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET
6B	107	PWR_MGMT_1	R/W	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		
6C	108	PWR_MGMT_2	R/W	LP_WAKE_CTRL[1:0]		STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG
72	114	FIFO_COUNTH	R/W	FIFO_COUNT[15:8]							
73	115	FIFO_COUNTL	R/W	FIFO_COUNT[7:0]							
74	116	FIFO_R_W	R/W	FIFO_DATA[7:0]							
75	117	WHO_AM_I	R	-	WHO_AM_I[6:1]						-

In the documentation we can read that MPU-6050 comes up in sleep mode by the time the system is powered and every register, apart from PWR_MGMT_1 whose reset value is 0x40 and WHO_AM_I which value is 0x68, are set to 0x00 so in order to wake-up the sensor and begin the configuration function we have to write first 0x00 in the PWR_MGMT_1 register.

We can now ask for the content of the WHO_AM_I register, that is only readable, and we have to ensure that this value is 0x68 just to have the guarantee that everything went correct and sensor actually behaves correctly.

4.5 Register 28 – Accelerometer Configuration
ACCEL_CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1C	28	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]		-		

AFS_SEL	Full Scale Range
0	± 2*g*
1	± 4*g*
2	± 8*g*
3	± 16*g*

Next step is to set accelerometer parameters; so as reported in the figure above we have to write bits 3 and 4 in order to set the acceleration full scale sensitivity factor. A good value for the quadcopter is $\pm 8g$, so considering the full byte we will send to register 0x1C, that according to the first table in the previous page is ACCEL_CONFIG, the value 0b00010000 (16 in decimal format).

4.4 Register 27 – Gyroscope Configuration
GYRO_CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1B	27	XG_ST	YG_ST	ZG_ST	FS_SEL[1:0]		-	-	-

FS_SEL	Full Scale Range
0	± 250 °/s
1	± 500 °/s
2	± 1000 °/s
3	± 2000 °/s

The same procedure is applied for the GYRO_CONFIG register addressed at 0x1B in which we will write the value 0b00001000 (8 in decimal format) corresponding to a full scale sensitivity of $\pm$ 500°/s.

Notice that both values chosen for accelerometer and gyroscope are good compromises between speed and accuracy but it is always possible to change them with the other values shown in the respective tables reported above.

```
void initialize_imu() {
        uint8_t Data = 0;
          HAL_I2C_Mem_Read (&hi2c1, MPU_6050,0x75,1, &check, 1, 1000);

                // PWR_MGMT_1 we write all 0's to wake the sensor up
                Data = 0;
                HAL_I2C_Mem_Write(&hi2c1, MPU_6050, WAKE_UP_REG, 1, &Data, 1, 1000);

                // Set DATA RATE of 1KHz by writing SMPLRT_DIV register
                Data = 0x07;
                HAL_I2C_Mem_Write(&hi2c1, MPU_6050, SMPLRT_DIV_REG, 1, &Data, 1, 1000);

                // Set accelerometer configuration in ACCEL_CONFIG Register FS_SEL=2 ->  8g
                Data = 0x10;
                HAL_I2C_Mem_Write(&hi2c1, MPU_6050, ACCEL_CONFIG_REG, 1, &Data, 1, 1000);

                // Set Gyroscopic configuration in GYRO_CONFIG Register FS_SEL=1 -> 500 g/s
                Data = 0x08;
                HAL_I2C_Mem_Write(&hi2c1, MPU_6050, GYRO_CONFIG_REG, 1, &Data, 1, 1000);
}
```

After the end of the registers setup it is possible to proceed with the calibration of the sensor. The easiest way to do this is to take consecutive readings with a fixed update rate. In each iteration the sensor is able to provide a new value when invoked, and calculating the averaging these values we can find the best offset to be subtracted from the raw data before their processing.

6.2 Accelerometer Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	AFS_SEL=0		±2		*g*	
	AFS_SEL=1		±4		*g*	
	AFS_SEL=2		±8		*g*	
	AFS_SEL=3		±16		*g*	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*	
	AFS_SEL=1		8,192		LSB/*g*	
	AFS_SEL=2		4,096		LSB/*g*	
	AFS_SEL=3		2,048		LSB/*g*	

6 Electrical Characteristics

6.1 Gyroscope Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		°/s	
	FS_SEL=1		±500		°/s	
	FS_SEL=2		±1000		°/s	
	FS_SEL=3		±2000		°/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)	
	FS_SEL=1		65.5		LSB/(°/s)	
	FS_SEL=2		32.8		LSB/(°/s)	
	FS_SEL=3		16.4		LSB/(°/s)	

Raw data become concrete data if they are divided by an appropriate constant that we know from the two tables reported above taken from the datasheet.

We see for the gyroscope that when FS_SEL = 1 the scaling factor is 65.5 which means that 65.5 is the raw value for $1°/s$, and for sure if we change the full scale sensitivity we have also to change the scaling factor. In the end every raw data divided by this value corresponds to the real angular velocity along its specific axis.

We reason in the same way for the accelerometer concluding that for a full scale sensitivity of $\pm$ 8g the scaling factor is 4096 that means that a raw acceleration of 4096 corresponds to 1g and so dividing every raw data by this value we obtain acceleration expressed in g for the specific axis.

We now have all the elements to accurately retrieve angular velocities and accelerations among the 3 axes of the quadcopter.

Combining what explained in the previous pages we get all the data needed to calculate all variables to control our quadcopter.

MPU_6050 is a MEMS (Micro Electro-Mechanical Systems) based sensor and so its output is divided into signals of different frequencies.

The practical and concrete idea is that most of the times we typically know informations about the system dynamics and what we are going to measure and so we can perform some kind of optimization.

For example, vibrations of the frame of a vehicle in which IMU is mounted on is noise with respect to the measure of the direction the vehicle is accelerating to. So if we want remove this disturbing noise component from the signal in order to have more precise data we can do this by filtering out signals with high frequencies.

Imagine a scenario in which, for example, a vibration moves the MPU_6050 up and down 250 times a second. This means the frequency of the vibration is 250Hz. Using a DLPF (Digital Low-Pass Filter) it is possible to cut-off everything with a frequency of 250 Hz or higher. In this way noise of the vibration is removed from the signal.

Because of that, it's very useful, that MPU_6050 provides an hardware implemented configurable DLPF on board.

This is especially convenient because it's not trivial at all to implement such kind of filter in software, as the math behind is not that easy and in particular is very computational expensive for the most of microcontrollers and must be taken into account that such kind of processing function requires much more samples as we usually would use (more or less on the order of 1000 measurements per second), which we have to implement in a real-time mode.

Set up the DLPF is easy and according to the datasheet table reported below we can do that writing in the first 3 bits of the 0x1A register.

4.3 Register 26 – Configuration
CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1A	26	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		

Description:

This register configures the external Frame Synchronization (FSYNC) pin sampling and the Digital Low Pass Filter (DLPF) setting for both the gyroscopes and accelerometers.

The DLPF is configured by *DLPF_CFG*. The accelerometer and gyroscope are filtered according to the value of *DLPF_CFG* as shown in the table below.

DLPF_CFG	Accelerometer (F_s = 1kHz)		Gyroscope		
	Bandwidth (Hz)	Delay (ms)	Bandwidth (Hz)	Delay (ms)	Fs (kHz)
0	260	0	256	0.98	8
1	184	2.0	188	1.9	1
2	94	3.0	98	2.8	1
3	44	4.9	42	4.8	1
4	21	8.5	20	8.3	1
5	10	13.8	10	13.4	1
6	5	19.0	5	18.6	1
7	RESERVED		RESERVED		8

Bit 7 and bit 6 are reserved.

Parameters:

EXT_SYNC_SET 3-bit unsigned value. Configures the FSYNC pin sampling.
DLPF_CFG 3-bit unsigned value. Configures the DLPF setting.

It is important to evaluate the delay that DLPF introduces working at a given frequency; we see that for a bandwidth of $5Hz$ we get $19ms$ delay and this is too much for the quadcopter. We also see that gyroscope raw output is $8KHz$ but when we enable the digital low-pass filter at whatever frequency this value drops down to $1KHz$ and this due to oversampling and bottleneck induced by the accelerometer.

For this project we will use the DLPF settled at $42Hz$ referring to the gyroscope so we will add to the initialize_imu() function the line

```
//DLPF_CFG = 0b00000011 --> 3 = 42Hz gyro 44Hz accel
Data = 0x03;
HAL_I2C_Mem_Write(&hi2c1, MPU_6050, DLPF_CONFIG_REG, 1, &Data, 1, 1000);
```

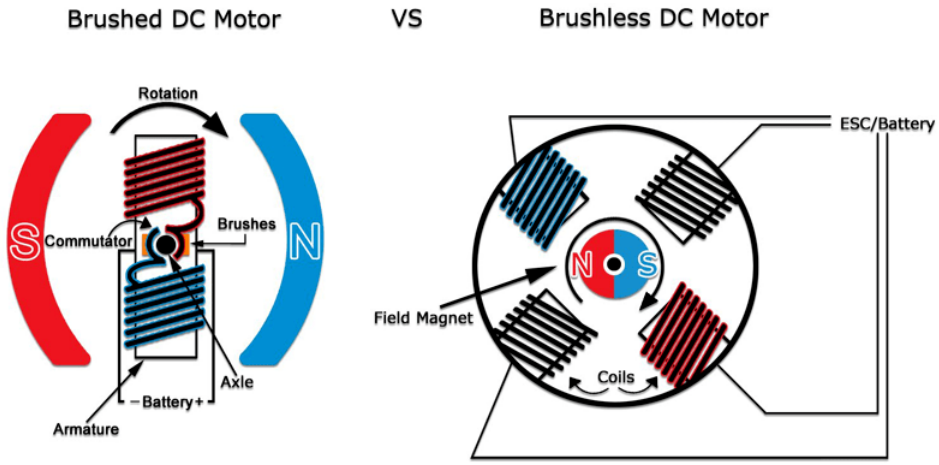
6 Brushless Motors

Brushless motors (BLDC motor for short) are DC motor made by a permanent magnet rotor and a stationary magnetic field stator (BLDCM).

The advantages of a brushless motor over brushed motors are high power-to-weight ratio, high speed, electronic control, and very low maintenance. Moreover these engines abandon the mechanical switch used in traditional engines to replace it with an electronic device that improves the reliability and durability of the appliance called ESC (Electronic Speed Controller).

Brushless motors find applications in such places as computer peripherals (disk drives, printers), hand-held power tools, and vehicles ranging from model aircraft, UAV (SAPR in Italy) to automobiles.

Another advantage of BLDC motors is that, for the same output power, they can be more compact and lighter than those with brushes and are therefore more suitable for applications with space constraints.

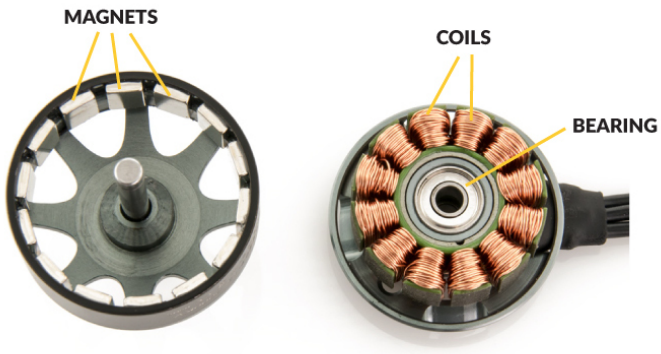


As shown in the previous picture in brushless DC motors the mechanical commutator is replaced by the more advanced three-phase ESC.

It detects the angle of the rotor, and controls semiconductor switches such as transistors which switch current through the windings, either reversing the direction of the current, or in some motors turning it off, at the correct time each 180° shaft rotation so the electromagnets create a torque in one direction. The elimination of the sliding contact allows brushless motors to have less friction and their working life is only limited by the lifetime of their bearings.

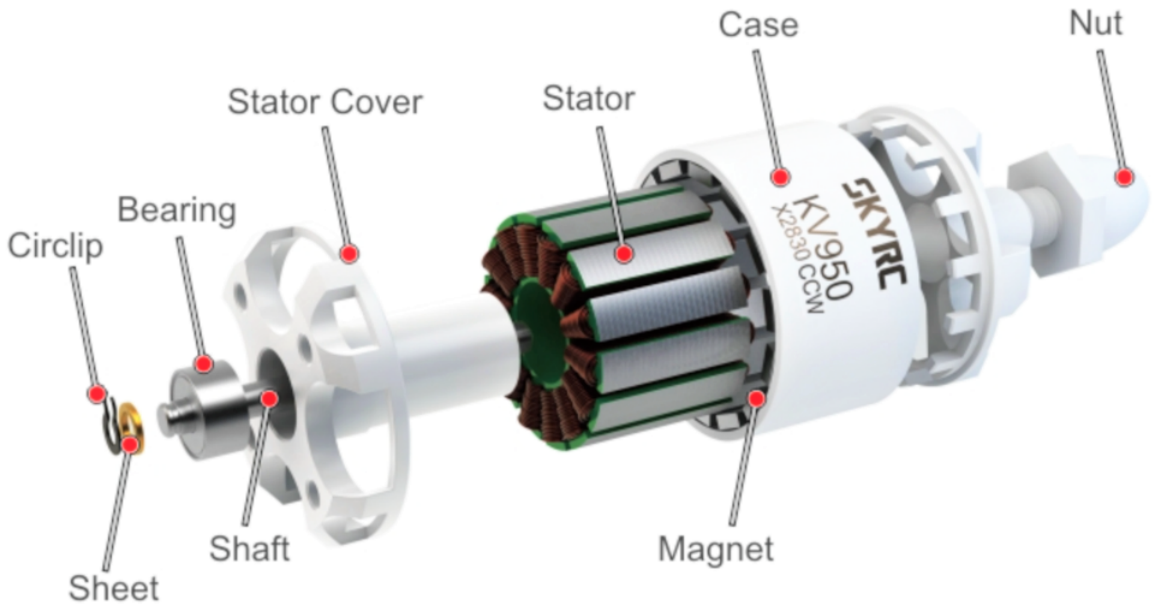
Brushed DC motors develop a maximum torque when stationary, linearly decreasing as velocity increases. Some limitations of brushed motors can be overcome by brushless motors due to higher efficiency and a lower susceptibility to mechanical wear.

A typical brushless motor has permanent magnets which rotate around a fixed armature, eliminating problems associated with connecting current to the moving armature. The ESC totally replaces the brush commutator of the classic brushed motor and continually switches the phase to the windings to keep the motor turning applying timed power distribution by using a solid-state circuit rather than the mechanic commutation system.



It is very important to notice that this kind of motorbrings to an overall reduction of electromagnetic interference (EMI). With no windings on the rotor, they are not subjected to centrifugal forces, and because the windings are supported by the housing, they can be cooled by conduction, requiring no airflow inside the motor for cooling.

Brushless motor commutation can be implemented in software using a microcontroller or microprocessor computer, or may alternatively be implemented in analogue hardware, or in digital firmware using a FPGA (Field Programmable Gate Array). Commutation with electronics instead of brushes also allows for greater capabilities not available with brushed DC motors; one of the most appreciable is micro stepping for slow and fine motion control and torque holding when stationary. Controller software can be customized to the specific motor being used in the application, resulting in greater commutation efficiency but almost all the ESCs are chosen in advance with respect to the application and the motors they are going to drive and hardly ever are customized "on the fly".



The maximum power that can be applied to a brushless motor is limited almost exclusively by heat, in fact too much heat weakens the magnets and will damage the winding's insulation.

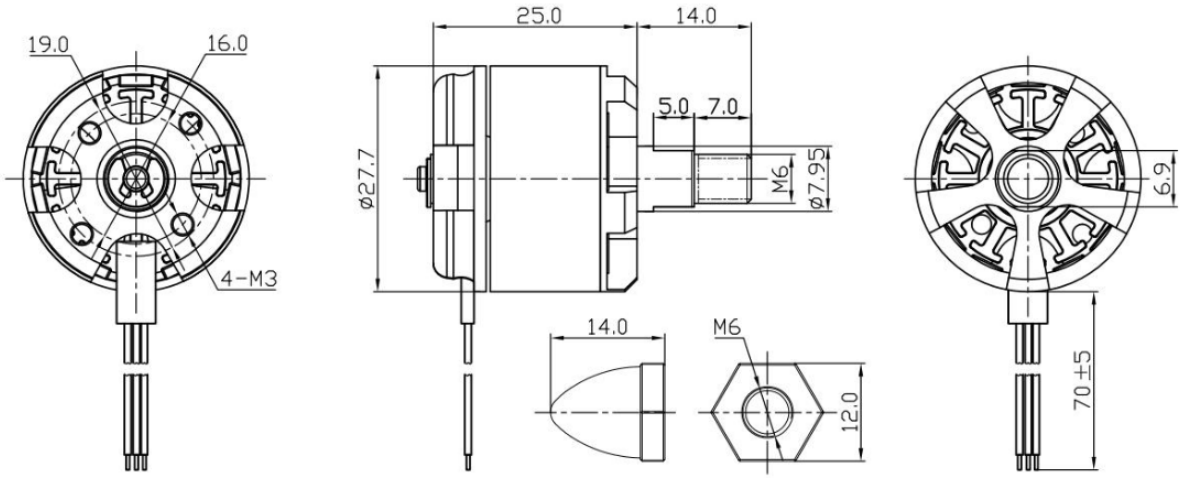
The advantage of BLDC motor in terms of efficiency is mostly due to the frequency at which the electricity is switched that is determined by the position sensor feedback. Additional gains are due to the absence of brushes, which reduces mechanical energy loss and heat peaks due to friction. The enhanced efficiency is greatest in the no-load and low-load region of the motor's performance curve.Under high mechanical loads, brushless motors and high-quality brushed motors are comparable in efficiency but it is not the case of the quadcopter.

The construction of a brushless motor may resemble that of a stepper motor. Unlike a stepper, a brushless motor is usually intended to produce continuous rotation. Stepper motors generally do not include a shaft position sensor for internal feedback of the rotor position. Instead of the stepper controller BLDC will rely on a hardware sensor to detect the position of the driven device.

The motors chosen for the project, whose characteristics are shown in the figure in the next page, have a relatively low KV value and this is because since we have a quite big frame for a racer we can equip it with bigger propellers. Pairing relatively big propellers with low-KV motors improves motor efficiency because of lower RPMs. Choosing low KV motors is a good choice also due to the fact that the ESCs, that we are going to present in the next chapter, allow the use of batteries up to 4S (4 cells in series) while the more powerful motors use either small propellers or bigger LiPos in terms of number of series cells and have much more RPMs; but to do this they usually require higher battery voltages to be fully exploited and they have lower efficiency but more power.

Mathematically we can say that since power is retrieved by multiplying the torque by the RPM for that fixed torque we can easily understand that a motor that is for instance 2300KV with a 4s LiPo battery will theoretically reach its power peak of $P_m = \tau_m * 2300 * 16.8 = \tau_m * 38640$ but due to friction and mechanical limitation such high rotation rate will never be reached. However a slower motor is generally (most of the times) characterized by a higher torque such the previous multiplication gives a lower result due to the significant gap in the RPM multiplier in the power equation but result in a greater efficiency of the whole ESC/motor system.

MOTOR OUTLINE DRAWING



MOTOR PERFORMANCE DATA

MODEL	KV (rpm/V)	Voltage (V)	Prop	Load Current (A)	Pull (g)	Power (W)	Efficiency (g/W)	Lipo Cell	Weight (g) Approx
B2212	920	11.1	8045	7.3	465	81	5.7	2-4S	50
			1045	9.5	642	105	6.1		
	980	11.1	8045	8.1	535	90	5.9		
			1045	10.6	710	118	6.0		

From the characteristics we can see that when powered with a 3S LiPo battery and using 1045 propellers, where 10 equals 10 inches and 45 is the pitch of the blades, in the full throttle position is able to generate a thrust equal to 710 g. Multiplying this value for the 4 motors, we obtain 710 x 4 = 2840 g which is the maximum thrust that the quadcopter is able to deliver at full throttle measured at a voltage of $11.1V$.

Notice now that a good practical rule is to ensure that the weight of the quadcopter has to be no more than half of the maximum thrust that the motors are able to deliver in order to have a reactive aircraft and have good controllers responses both in actuation and stabilization.

Another parameter to consider and to keep under strict observation is the Load Current which, as can be seen in the case of our 920 KV, is 7.3 A which multiplied by 4 results in 10.6 x 4 = 42.4 A.

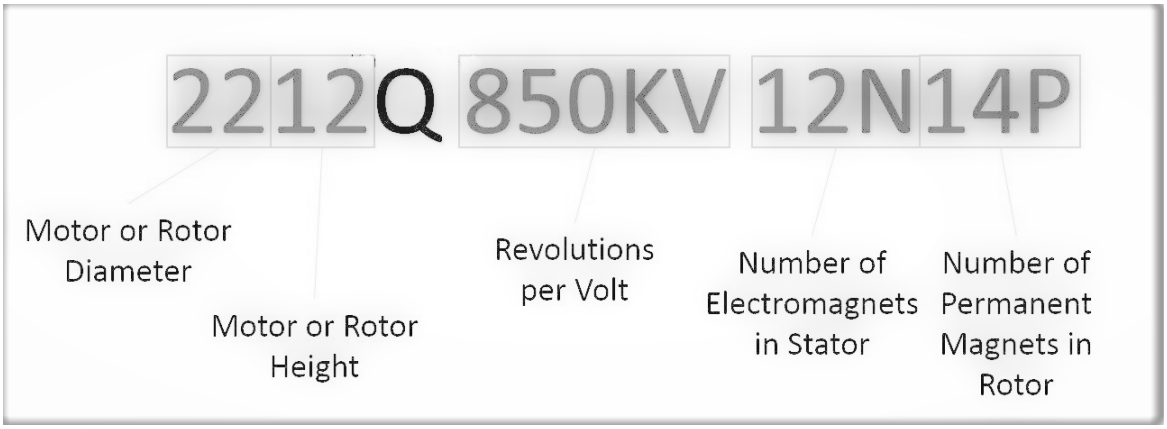
This last calculation gives us two fundamental information on two further constraints to be included in the model:

1. The load current of the single motor at full speed is 7.3 A therefore we deduce that each single ESC must be at least 10 A but personally I advise to oversize the ESCs with respect to this parameter to reduce stress and avoid excessive overheating; anyway 10 A surely will work.

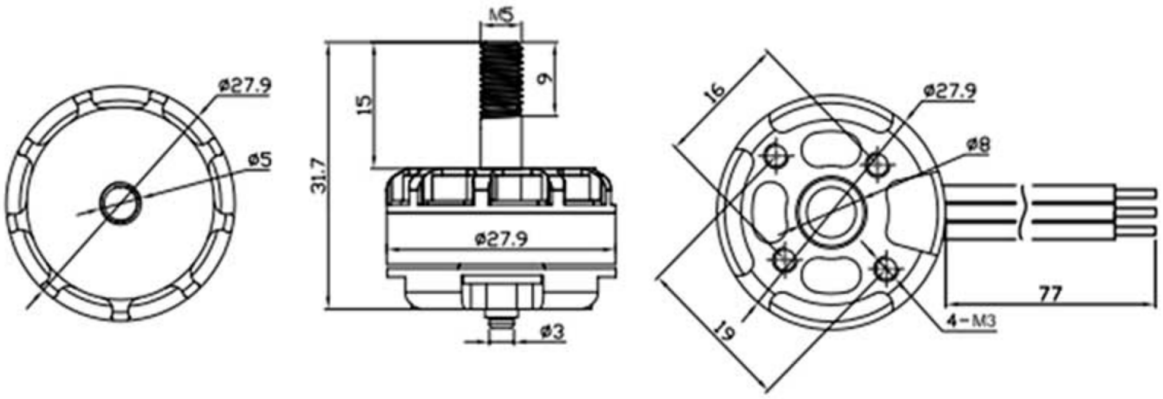
2. The lithium polymer battery (LiPo) with which we are going to feed our system must be able to supply the power simultaneously to 4 motors at full throttle; it means that as calculated before it must be able to provide 29.2 A and this parameter can be easily calculated as the capacity of the battery expressed in Ah multiplied by its C value that can be found written on the battery or is provided by the producer. If two C values are given take the lowest one because the other represents the burst value that is the peak value which can be delivered for a little time interval (usually 10 seconds).

However, the best performing racer drones reach ratios weight : max_thrust even in the order of 1:8 but they have very big values of Load Current for each motor which translates into high power consumption and consequently less autonomy.

This should guide to the correct sizing of the battery which is an important choice that has to be done. LiPo must provide continuously all the Amperes required by the 4 motors without working in its critical area and must a burst current (usually 10 seconds long) that have to be as bigger as possible in to the total max current that can be asked by the 4 motors together. A final piece of advice I would like to give is to evaluate the purchase of graphene LiPo because the weight is no longer so much greater compared to normal LiPos and the higher price is totally compensated by the in flight performance and by the sensibly reduced charging times up to 15 times the capacity of the battery.



Increasing the width or height of an engine will increase both the size of the permanent magnet and the coils of the electromagnetic stator. The main difference is that when the height of the stator is increased, the size of the permanent magnet increases more than the size of the coil and when the width of the stator is increased, the size of the electromagnetic coil increases more than the permanent magnet.



Specifications of our motors report the value B2212. The number 22 indicates the number of electromagnets in the stator, or the poles, and 12 indicates the number

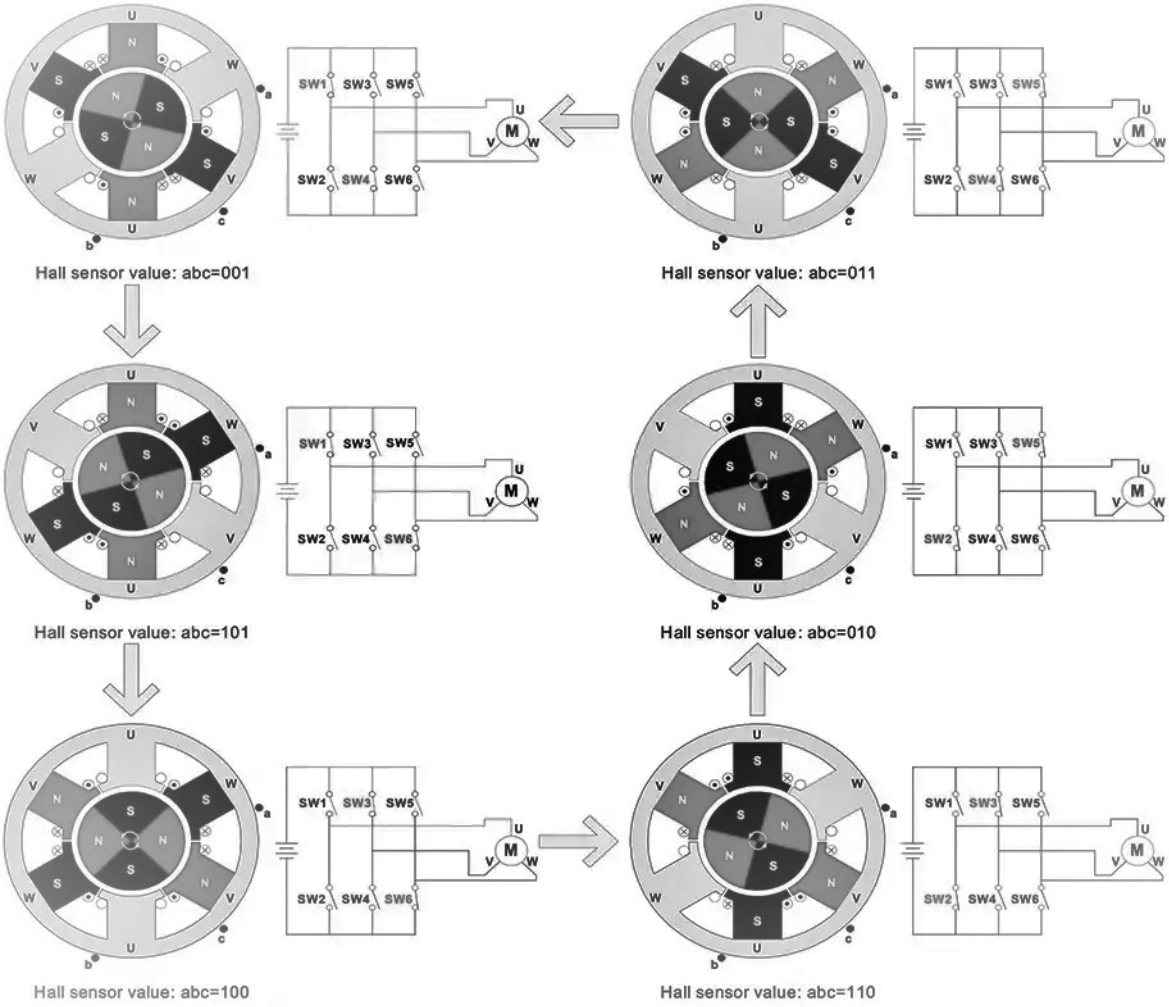
of permanent magnets in the bell (rotor). The motors of different sizes have a different number of poles, the 22XX and 23XX motors generally have 12 poles and 14 magnets and this is our case.

The number of poles determines the spacing between them. If you have fewer poles, you can insert more iron content in the stator, so as to get more power from the motor. But with a greater number of poles, the magnetic field is spreading more uniformly, and therefore there is a more fluid motor because there is more precise control over the rotation of the bell.

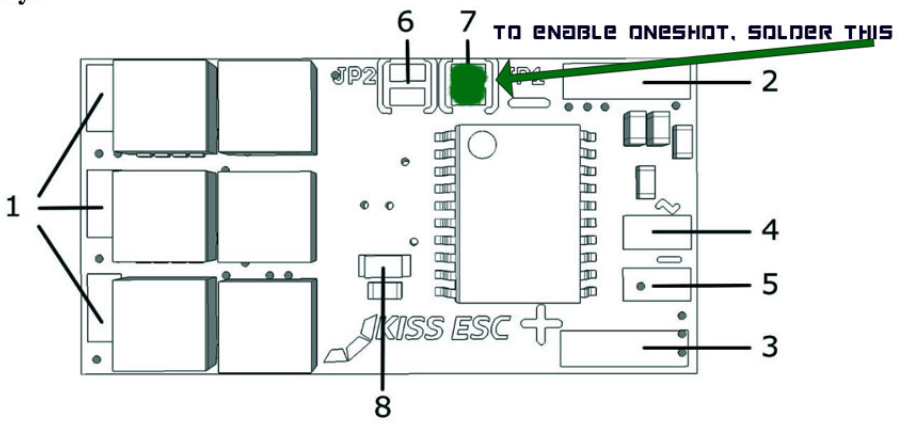
Overall we always have to consider a compromise between these two configurations because more poles implies smoother motor control but less poles implies more powerful motors.

7 ESC - Electronic Speed Controller

The ESC or Electronic Speed Controller is the component that takes care of converting a PWM (Pulse Width Modulated signal) pulse that arrives as input from the flight controller into the corresponding voltage that will be used by the motor to keep itself in motion, throttle-up or slow down. The signal that is transferred from this to the motors is a three-phase signal in which each phase is 120° out of phase, $\phi = 120$, with respect to the others and this is needed to ensure the correct motor rotation direction.



Assembly:



1. Solder pads for the three BLDC motor phases
2. LiPo connection Negative (-)
3. LiPo connection Positive (+)
4. PWM Signal input (Servosignal)
5. PWM Signal ground reference
6. Solder jumper, for changing the rotation direction (CW/CCW)
7. Solder jumper, for changing the PWM input signal type (normal PWM 25-500Hz/OenShot125)
8. State LED

Typical configuration of an Electronic Speed Controller is the one showed above in which the only missing element is the electrolytic capacitor between the input signal and the ESC itself.

In recent years, numerous protocols have been developed and this is because these devices no longer interface in the traditional way directly with the radio receiver but the signals are processed and supplied by the flight controller.

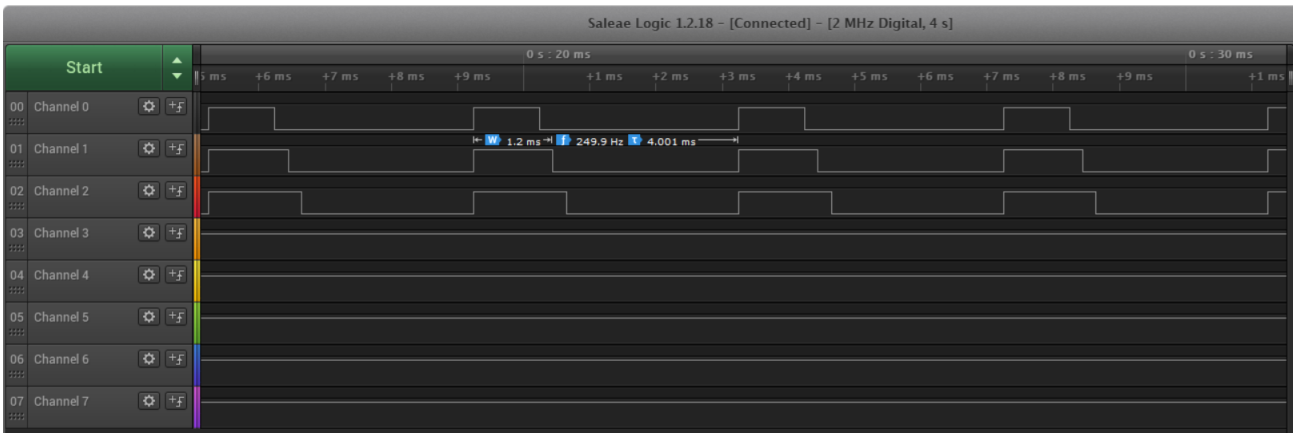
In fact, before the advent of drones, brushless motors were controlled by these devices like normal servos with [1000;2000] microseconds pulses at 50 Hz.

Today (2019/2020) there are some brands that provide configuration suites in order to provide the user a choice of the ESC protocol with respect to the speed that the flight controller car reach.

In our drone we use some BLHeli that with their default configuration allow Standard_PWM, PWM_500Hz and also OneShot_125

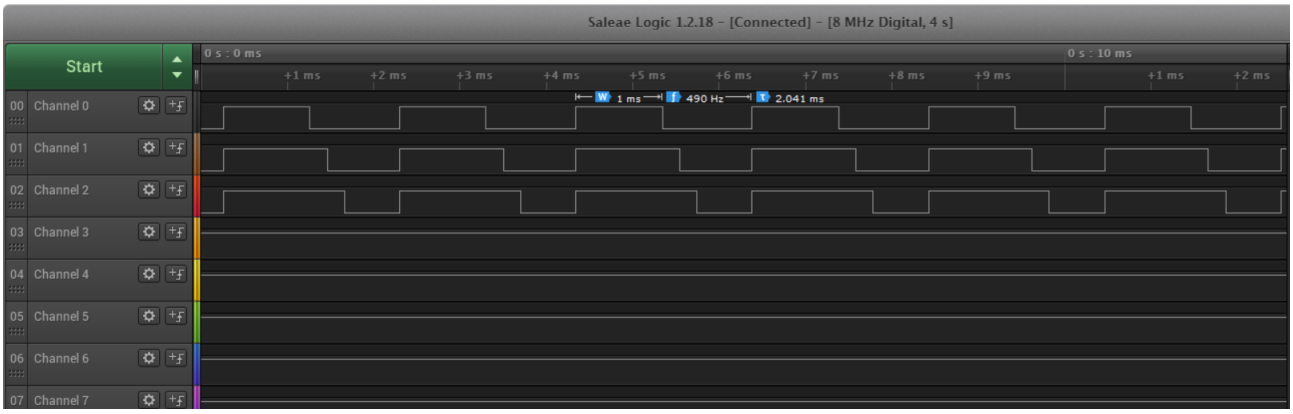
The greater speed that ESCs provide allows to have faster PID cycles able to effectively update signal writes on the motors at the same speed as the microcontroller control cycle which is getting faster and faster with each ARM generation.

Classic PWM, the same one that uses the radio control to send the input signals, is too slow for the management of any modern flight controller therefore all the ESCs support at minimum 250Hz refresh rate which is 5 times faster than the normal one.



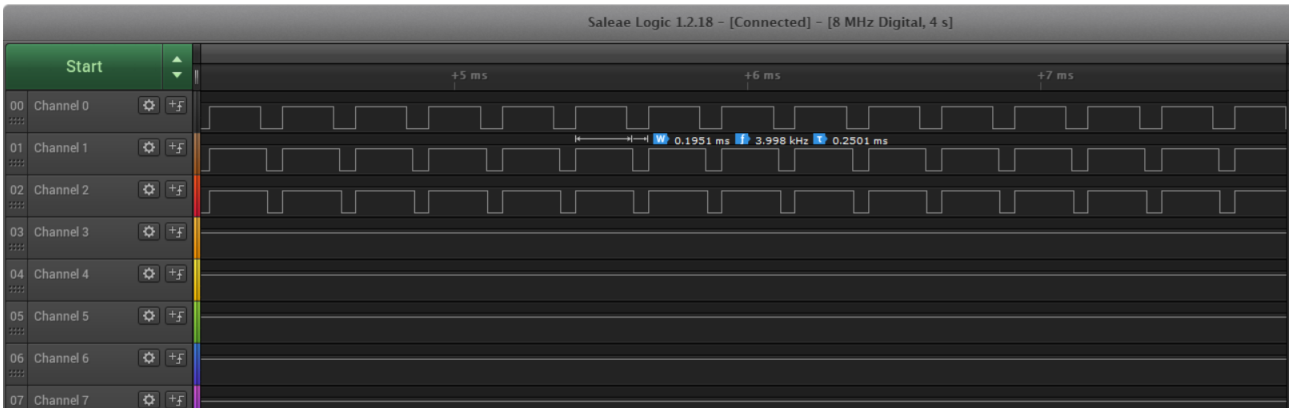
However it makes sense only if the control loop is equal or faster than 4ms and for sure more or less all the microcontrollers starting from Arduino which is an AVR 16MHz 8-bit MCU (ATMega 328p) to more powerful STM32 F1, F3, F4, F7 are able to reach such speed.

STM32F303K8T6 comes with a Cortex M4 (72MHz) with a FPU (Floating Point Unit) that makes angles math faster due its hardware direct support. We can now observe that using such board we can ran the control loop at a much higher rate and so ESC refresh rate can be updated to 500Hz (490Hz in practice) in order to follow the control loop with a ratio between control/actuation of 1:1.



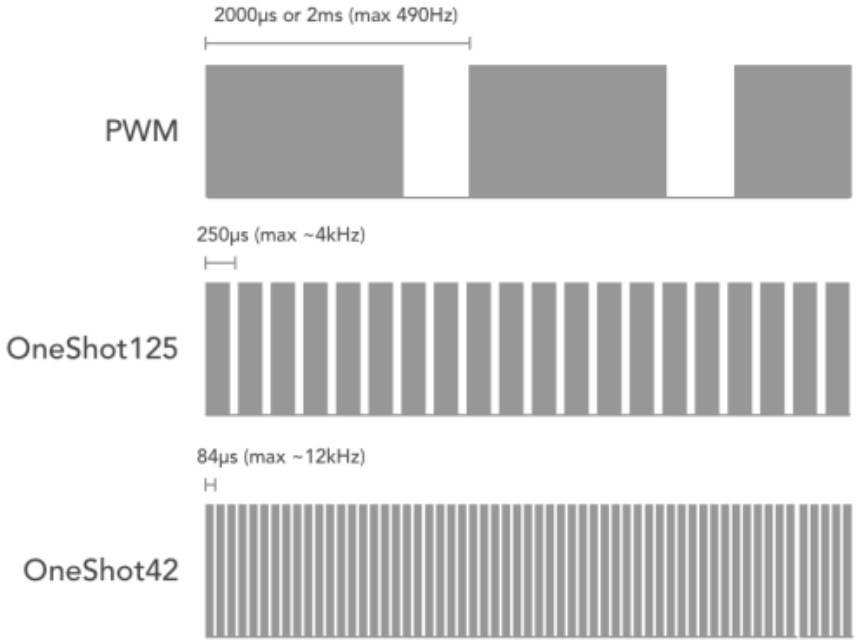
It is also possible to push PWM to very much higher rate but with some limitations. Typically in fast flight, the one of racing quadcopters, pilots and engineers try to have higher rates to minimize latency and optimize input response to radio input. This is very important because such kind of platforms can reach speed in the order of $140Km/h$ or more and with these dynamics is not desirable to have lazy responses.

OneShot_125 was designed when F3 flights controller became popular and used in DRL quadcopters and its name is due to the fact that classic PWM of [1000;2000] microseconds is scaled to [125;250] microseconds and so this protocol allows a ratio 1:1 between control and actuation for a control loop that runs at 4KHz that is 8 time faster than the 250Hz PWM.



Such controller speed is only possible using only the gyroscope, in fact we showed in the IMU chapter that when accelerometer is enabled the update rate of the MPU_6050 drops drastically but 4Khz or higher rates are useful mostly only in acro/rate mode and microcontrollers aimed with hardware to speedup heavy math.

After OneShot_125 was invented OneShot_42 but it haven't had the same popularity of the previous protocol because simultaneously were developed digital protocols that changed the way to drive BLDC motors. However OneShot_42 allow up to $12KHz$ control loop and it is 3 times faster than OneShot_125 A magnitude comparison between the main 3 analog protocols is reported below



The last analog ESC protocol is MultiShot; it allows a refresh rate of $32KHz$ that is more than enough for all the commercial flight controllers in the market. It uses PWM pulses of [5;25] microseconds with a resolution of 240 steps. This protocol is not so much used because it has no native support and most of the times such extreme rate is completely useless for the majority of the applications.

Another modern approach to drive brushless motor is using digital ESC protocols whose name include the speed. DShot comes in different versions; DShot_150, DShot_300, DShot_600, DShot_1200.

They have the following rates :

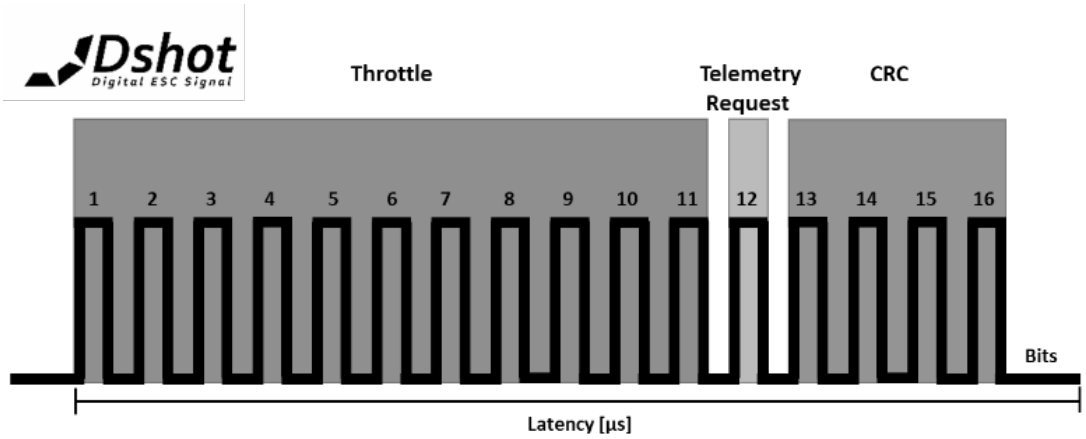
1. DShot_300 150000 bits/Sec
2. DShot_300 300000 bits/Sec
3. DShot_600 600000 bits/Sec
4. DShot_1200 1200000 bits/Sec

While Standard PWM, OneShot_125, OneShot_42 and MultiShot rely on the length of an electrical pulse to determine the value DShot ESCs only use digital signals.

Analog signals can have issues with signal accuracy:

1. Due to the possibly different speed of the oscillators (or clock) in ESCs and FC, the length of the pulse might not be measured accurately, especially when we are talking about microseconds. This is the reason why ESC calibration is required to sync the oscillators
2. Electrical noise (voltage spikes) can corrupt analog data – that's why we sometimes see people suggest running higher motor update rate than PID loops helps flight performance. Because when you repeatedly sending the same value to the motor, it averages out the error and this contributes to increase precision

With digital protocol, there won't be any of these problems, calibration will no longer be necessary. Because of the nature of digital signal, which is composed by one and zero, it will also be much more resistant to electrical noise induced for example by motors EMFs (Electro Magnetic Fields).



A DShot data packet consists of a total of 16 bits: 11 bits for throttle values, so $2^{11} = 2048$ steps, 1 bit for telemetry request and 4 bit for CRC checksum (Cyclic Redundancy Check).

For example DShot600 would have a frequency of

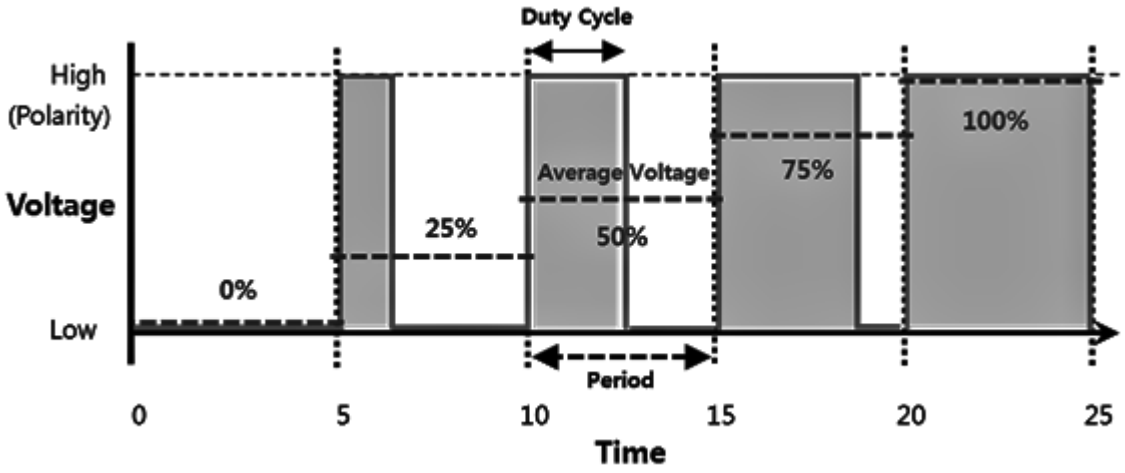
$$600,000/16 = 37500Hz = 37.5KHz$$

which means it will take about 26.7uS to send a single throttle value from the flight controller to the ESC. DShot transmission will theoretically allow up to 37.5KHz FC Looptime but in practice it will be 33KHz because a space is needed between values.

8 PWM - Pulse Width Modulation and motor outputs

PWM, short name for Pulse Width Modulation, is a technique to drive a signal modulating its amplitude. It is characterized by a switching activity that determines the transferred signal.

Square wave is defined by a period T, that is the time after that the signal is repeated and a duty cycle that is the ratio between the time the signal stay high and the signal period as shown in the picture below.



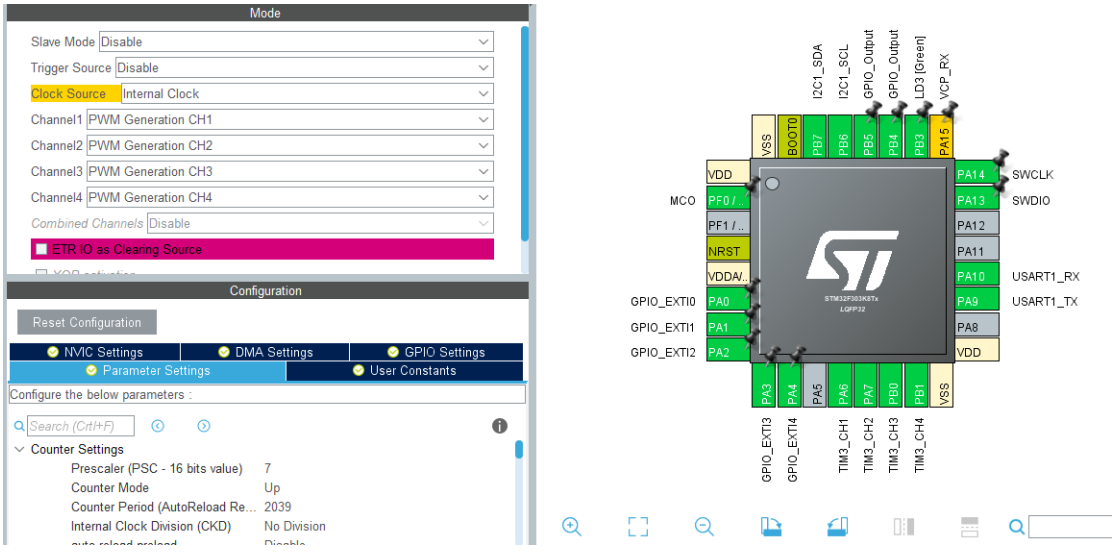
The higher the duty cycle the higher the total power supplied to the load. PWM is particularly suited for running inertial loads such as motors, which are not as easily affected by this discrete switching, because they have inertia to react slow.

Switching frequency has to be high enough not to affect the load, which is to say that the resultant waveform perceived by the load must be as smooth as possible or equivalently the signal must appear as small as possible. The rate (or frequency) at which the power supply must switch can differ a lot depending on the kind of load and application (a servo is much different from a brushless motor).

For example, switching has to be done at 120 Hz in a lamp, between a few kilohertz(kHz) and hundreds of Hz for motor drive. Switching activity is also used to drive computer power supplies. The main advantage of PWM is that power loss in the switching devices is very low. When a switch is off there is no current (or at least negligible); when it is closed so the signal is on and the current is being transferred to the load, there no significant voltage drop.

As we just said power loss, obtained by the product of voltage and current, is thus that in both cases is such close to zero that can be neglected without problems. PWM also works well with digital controls, which, because of their on/off nature, can easily set the needed duty cycle.

For analog PWM protocols we need to periodically send to the ESC a pulse, a signal that represents the information about the speed we want to transfer to the motor or more in general to the load. The switching activity we mentioned before must be handled in hardware peripherals in order to free the CPU from this calculation by limiting the interaction only to the modification of the amplitude of the signal to be written. Hardware dedicated to generate PWM are timers. Typically STM32 have some embedded timers on board of which in almost all cases the Timer1 is the one with more advanced functionalities followed by other general purpose 16 or 8 bits units.



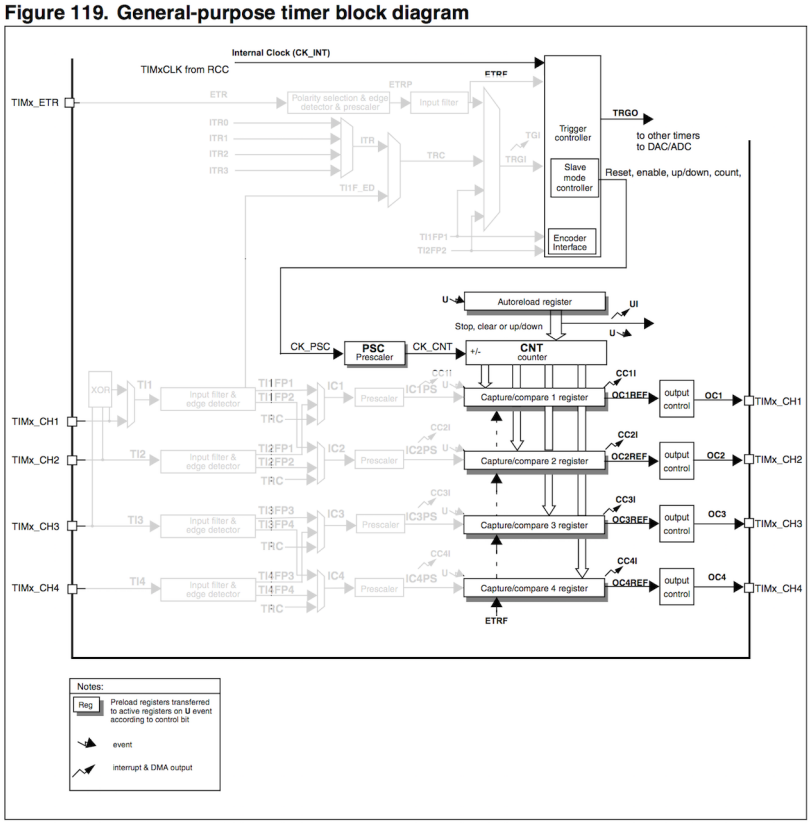
For this project we use Timer3 to generate PWM for the motors and we must configure it as reported in the picture in the previous page.

All the channels must be settled as PWM_GENERATION_CHX where X is the number of the channel, CH Polarity must be settled to "high" for all the channels.

As reported in the datasheet Timer3 is clocked at 8MHz by default so in order to have a counting step of 1 microsecond we have to divide that frequency by 7 obtaining a 1MHz clock for the timer ($\frac{1}{1000000} = 0.000001s = 1\mu s$).

After that we have to set the ARR (AutoReload Register) value that must match the PWM period minus one, where the minus one is used because also zero counts as counting step, like array indexes. In the picture in particular a PWM with a frequency of $490Hz$ is settled, in fact having a counting step of $1\mu s$ and a period of $2040\mu s$ we can easily calculate $\frac{1}{2039+1} = 490.2Hz$.

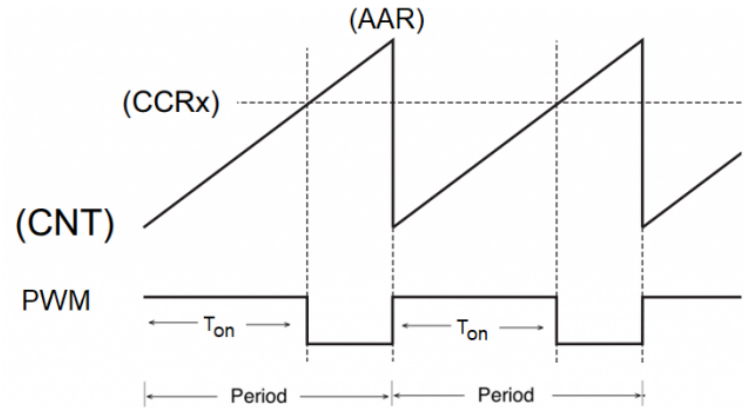
Figure 119. General-purpose timer block diagram



Square wave is created using two registers; the previous cited AutoReload Register and a register called CCR (Capture Compare Register).

There is a CCR register for each channel of the timer called CCRX where X is the number of the channel. By setting the CCR value we assign the width of the pulse and since we settled CH_POL = high for each channel the output value will result in a high signal until CCR value is matched and in a logic zero from CCRX + 1 until counter overflow that is ARR + 1.

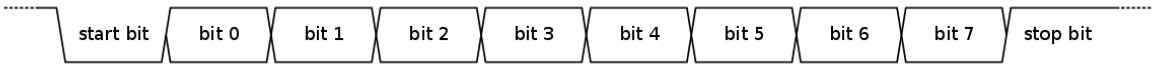
The figure in the previous page shows the typical structure of a general purpose timer such as the Timer3 whose clock signal is latched from an external line, where external means outside the timer unit but inside the board, one internal control logic block, four independent compare units and four alternate function output pins.



$$D = \frac{T_{ON}}{Period} \times 100\%$$

9 UART - Universal Asynchronous Receiver Transmitter

A universal asynchronous receiver-transmitter is a computer hardware device, or in this specific case an alternate function for asynchronous serial communication in which the data format and transmission speeds are configurable. The electric signaling levels and methods are handled by a driver circuit external to the UART. A UART is usually an individual integrated circuit (IC) used for serial communications over a computer or peripheral device serial port. One or more UART peripherals are commonly integrated in microcontroller chips and this is the case of STM32F303K8T6 that has two of these units.



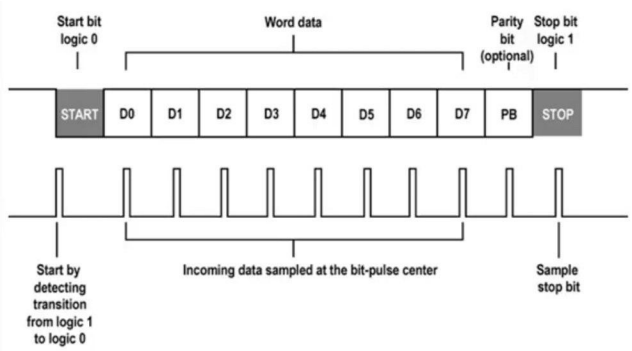
Transmitting UART (TX) takes bytes of data and transmits the individual bits serially. Another UART that acts as receiver (RX) reconstructs the bits into complete meaningful bytes. Each of this unit contains a shift register, which is the fundamental method of conversion between serial and parallel forms.

Serial transmission of digital bits through a single wire or other medium is less costly than parallel transmission through multiple wires and in embedded systems it means less pin to dedicate to interface devices, so the choice of UART but more in general serial communication is preferable rather than the faster parallel mode.

This unit most of the times does not directly generate or receive the external signals. Different devices are used to convert the logic level signals of the UART to and from the external signalling levels, which may be standardized voltage levels, current levels, or other signals.

Communication may be simplex (monodirectional), full duplex (both devices send and receive at the same time) or half duplex (transmitting and receiving but not simultaneously).

When no data is transmitted over the serial port, in the idle state, lines are pulled-up to V_{cc} or high-voltage or logic one. This is due to historical legacy from telegraph whose line were leaved high to make the other part conscious that there were no problem on the line.



The start bit signals the receiver that a new character is coming. The next five to nine bits represent the character. If a parity bit is used, it would be placed after all of the data bits. The next one or two bits are always logic high and called the stop bits. They signal to the receiver that the character is complete. Since the start bit is logic zero and the stop bit is logic one there are always at least two guaranteed signal changes between characters. If the line stays low for longer than a character time, this is a break condition that can be detected by the UART.

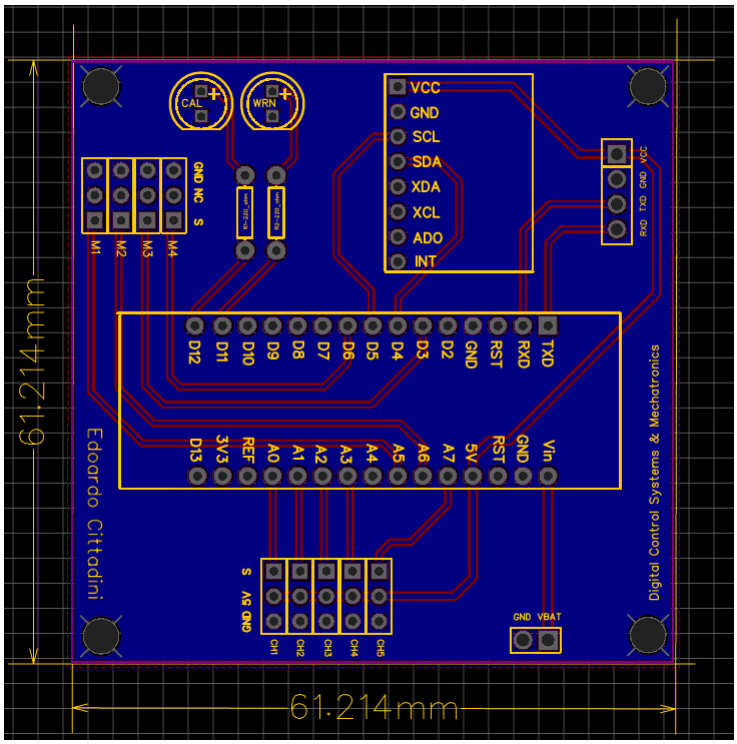
Since such kind of communication is asynchronous there must be some other notion of time that must introduced in order to sample and retrieve data correctly; this measure is baudrate or bitrate that declares how many bits are sent/receiver per second. Every bit of the frame is sampled in the middle of its window; this is the most convenient choice because it minimizes the possibility of error, in fact in that instant the signal hold time is sufficient to eliminate spurious states due to logic level change and have also margin to stabilize an uncertain data.

In the project the basic configuration is used that is 9600/8N1 in which there is one start bit, eight data bits, no parity bit and one stop bit. 8N1 is the most common configuration for PC serial communications today and was chosen because it also allows to communicate with slower devices, however this parameters can be changed whenever it is needed.

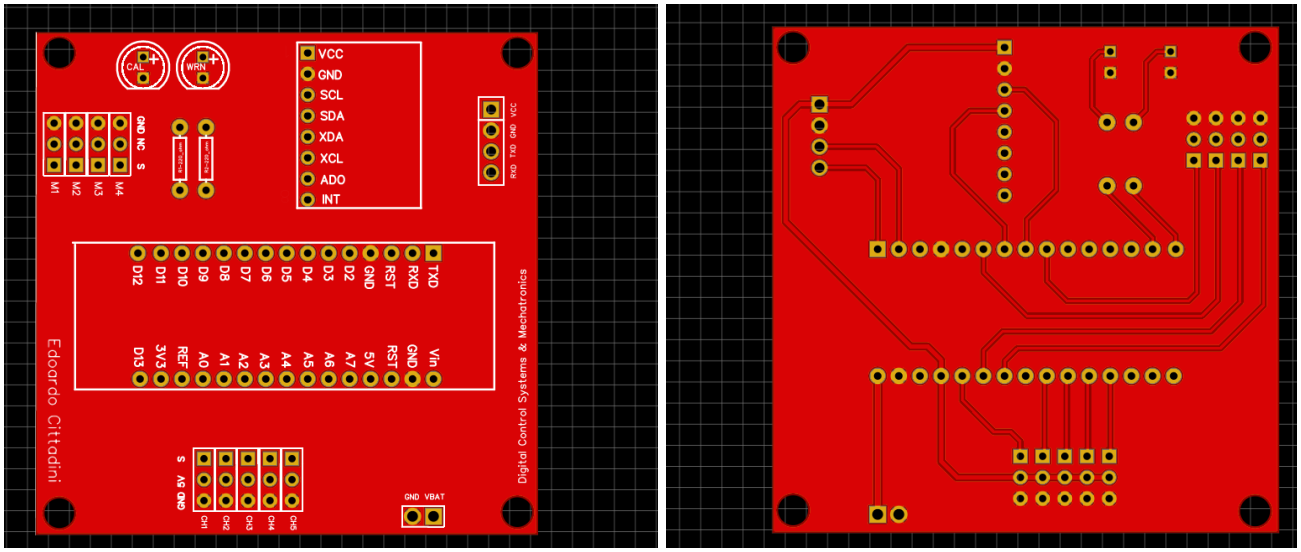
10 PCB Design

It is always better to avoid coarse grained circuit, that are instead the best and fastest way to test simple circuit, so in this section is shown an important step that can make the difference not only in terms of aesthetics but also in terms of performance, noise reduction, features and signals reliability by the mean of cross-talk minimization that is one of the greatest problem of DuPont connections and it is the design of a custom PCB based on the open-source software that has been discussed until now.

In electronic systems implementations in most of the cases hardware is implemented first and software is developed hardware-based; this choice is driven both by the fact that hardware tasks are more reliable over time but also because are more efficient and allows MCU to have a smoother loop which is always desirable given the limited performance of such kind of units.



As shown above it is possible to reach very limited dimensions (6.12cm x 6.12 cm) compared to the STM32F303K8T6 implementation on development board because tracks can be closer one to the other because there is no need to create them by hard and coarse tin soldering because are realized by PCB manufacturer.



Before send the PCB to production I used a rendered view of the circuit in order to see how top and bottom layer will be produced. Tracks are only in the bottom layer because it has been developed and designed as single-layer PCB.

Despite of the fact that tracks are only present in the bottom layer both the external layer, so the bottom and the top one, had been connected on the so called Ground Plane that provides a reference to GND to all ICs on the board.

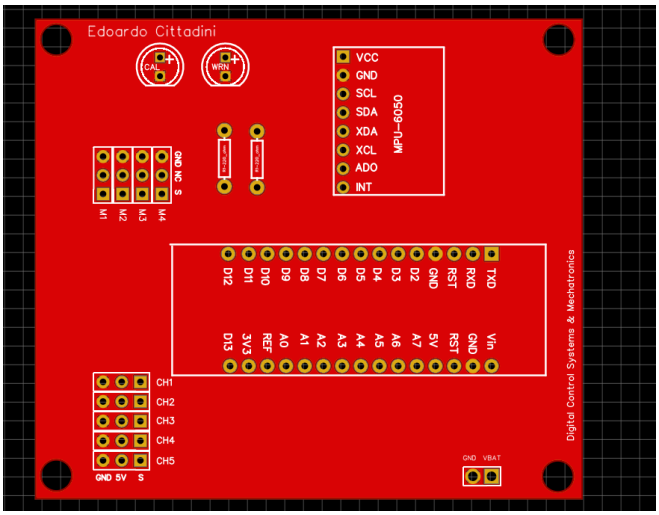
Since ground plane is relatively big with respect to the components of the board its resistance, capacity and so noise production in the circuit is sufficiently small that can be neglected without any problem. That is one of the most important difference from the Dev-board circuit in fact not only tracks are fixed-positioned and well isolated but all the PCB is as well isolated from external noise by the Ground planes.

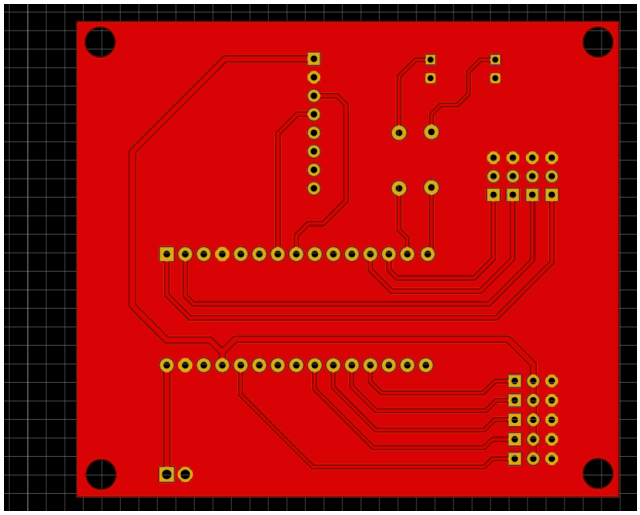
Two more output LEDs are used to give the user a feedback about some critical operations performed by the MCU and IMU.

The blue LED called CAL in the PCB Top Silk Layer is used to make the user know that MCU is performing IMU startup calibration and so the drone must stay spirit-leveled and motionless until it turns off and the MCU starts motor startup and PWM synchronization.

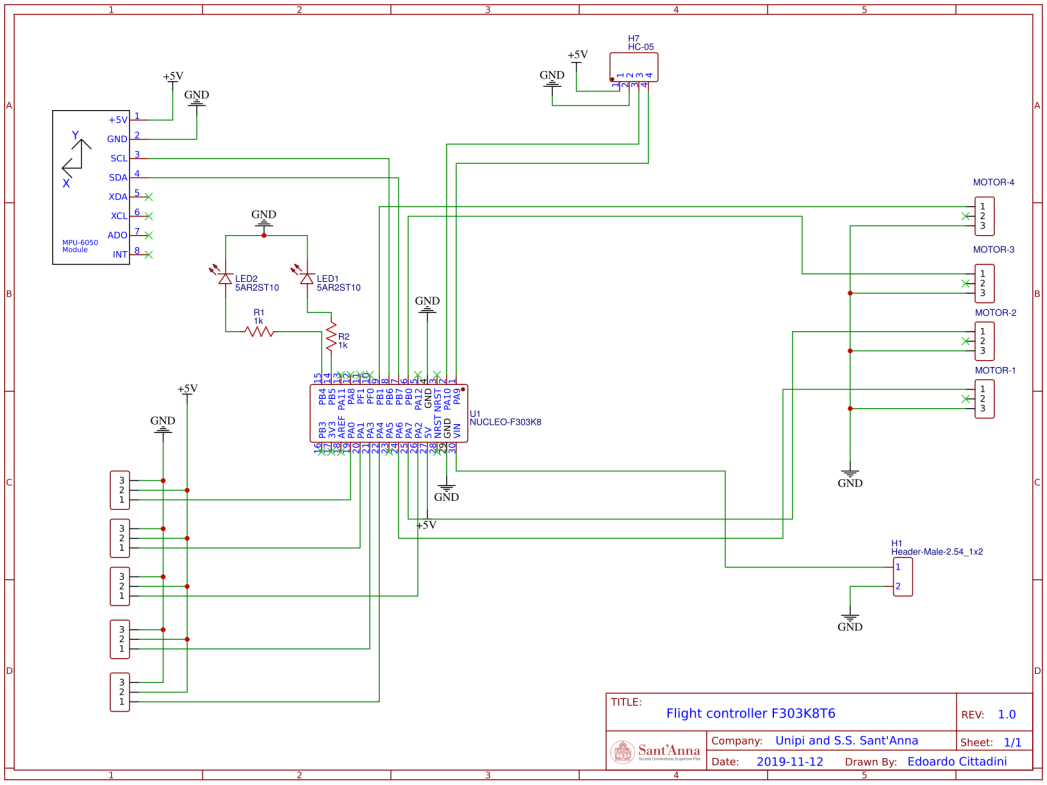
The red LED called WRN in the PCB Top Silk Layer is the general Warning LED and it is used to make the user know about malfunctions or potentially dangerous situations due to the fact that the microcontroller is delaying the only hard deadline present in the system that is the rendez-vous with the timer to generate the PWM pulses for the motors, and because of that a real-time error has occurred and the operator must land the quadcopter to ensure its safety, the safety of other people and the integrity of the vehicle itself.

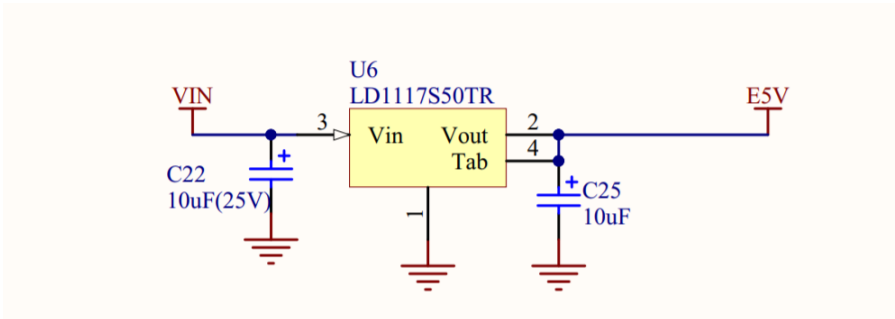
A second version of the PCB was designed in order to use different internal hardware of the STM32F303K8T6 and less peripherals in the configuration reported below





The schematic of the electrical connections of the first board is shown below





Observe that the LiPo is directly connected to the V_{in} pin this is because looking at the datasheet of the board we can see that the board has a LD1117S50TR voltage regulator that can perfectly handle a 12.6V 3S LiPo battery since its upper threshold is 15V.

11 Mathematical model

In order to develop and deploy the system in the correct way, a reliable simulation system must be provided to explore the dynamics of the quadcopter and characterize all the parameters of the control loop.

However, it is necessary to find a compromise between the complexity of the model and the accuracy reached by the latter simulating the behavior of the real system. High detailed mathematical models describe quadcopter mechanics and kinematics more accurately, but they also require more computational resources which leads to longer simulation time or even to inability to successfully complete the simulation.

It is much more usual to fly a quadcopter in the "X" configuration, the one shown above on the right, with respect to the "+" configuration (the left one) so we implemented both the simulation model and the real control in the flight controller to flight in "X" configuration; however, it is possible to implement the other configuration with minimal changes.

In this project we are going to provide a model for the rotational dynamics based on the Inertia since the control loop we implemented uses angular rate both for setpoint and feedback measurements.

To achieve this goal we used MATLAB/SIMULINK simulation environment and we implemented blocks about the radio PWM mapping to angular rate, motors and propellers actuation and quadcopter dynamics motion process.

12 Model and dynamics

The problem of modelling complex objects and their dynamics is not trivial at all, so in engineering design and control theory is common to approximate irregular shapes and volumes as composition of simpler and well known rigidbodies.

In the case of our project we can greatly simplify the problem by considering the quadcopter as a four perpendicular rods binded in the center of mass of the drone and with a equal distributed mass among all the frame.

The quadcopter flies based on the data acquired by the MPU 6050 and the mathematical model is characterized by 6DOF which are present the three rotations around the X,Y,Z axes and the three linear movements along the same axes.

It is known from a theoretical point of view that 6 degrees of freedom should be sufficient to have full control of the vehicle, but in practice this is not true and it is mainly due to the flaw of the sensors (they are not perfect) and the disturbances present in the real system. For example, the angle of yaw on the Z axis in 6DOF can only be controlled by the angular velocity measured by the gyroscope.

Our analysis has been made and is based on data collected at a forward speed of 0 m/s (hovering).

13 Plant data

The first phase consists in collecting data relating to the plant to be controlled.

Data can be:

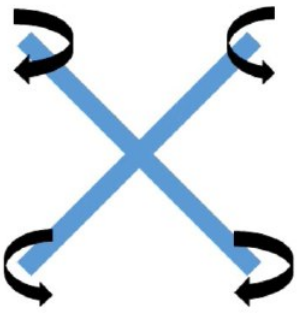
1. Real
2. Estimated
3. Approximated

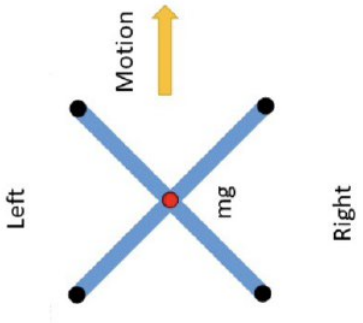
Real data are almost always known and static measurements of the system mostly coming from the components datasheets.

Estimated data are retrieved by a mathematical relationship conducted on the behavior in some instants which can be used to characterize the component.

Approximated data are mostly derived from a mathematical model of an object or from a complex dynamic. In this case, the numbers deduced often derive from simplifications of the model.

In our case the moment of inertia on pitch and roll and yaw is an approximate parameter given by the resolution of the remarkable integral of the homogeneous full bar while for example the motor speed constant (KV) is a real parameter and the thrust coefficient is an estimated parameter.





Plant data		
Component	**Value**	**SI unit**
Motor KV	920	KV
Motor Kt (Torque constant)	0,001086956522	Nm/A
Motor pull	0,465	Kg
Battery voltage (from motor DS)	11.1	V
Air density constant (sea level)	1,225	Kg/m³
Arm tickness	0,04	m
Propellers diameter	0,2032	m
Propellers pitch	0,1143	m
Max % drop at full speed	25,00	%
Frame weight	0,160	Kg
Propeller weight	0,005	kg
Motor weight	0,050	Kg
ESC weight	0,0250	Kg
Battery weight	0,105	Kg
Receiver weight	0,0149	Kg
Board weight	0,073	Kg
Mass	0,673	Kg
Distance CoM motor (arm)	0,165	m
Distance adiacent motors	0,233345	m
Moment of Inertia X axis	0,00305	Kg*m²
Moment of Inertia Y axis	0,00305	Kg*m²
Moment of Inertia Z axis	0,00611	Kg*m²
Pull ratio / weight	2,764	-
Ideal max weight slack	0,93	Kg

The moments of inertia relating to the bar at its ends and given the mass M and the length L of the bar itself is the result of the integral of the rod and so:

$$I_{xx} = \frac{1}{3}ML^2 \;\rightarrow\; \frac{1}{3}\,\frac{0.673}{4}\left(\frac{0.2333345}{2}\right)^2 4 = 0.00305\ Kg\ m^2$$

$$I_{yy} = \frac{1}{3}ML^2 \;\rightarrow\; \frac{1}{3}\,\frac{0.673}{4}\left(\frac{0.2333345}{2}\right)^2 4 = 0.00305\ Kg\ m^2$$

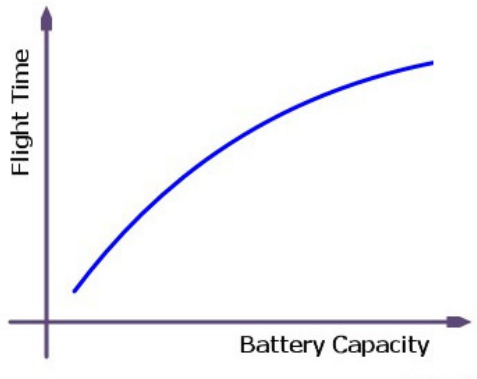
$$I_{zz} = \frac{1}{3}ML^2 \;\rightarrow\; \frac{1}{3}\,\frac{0.673}{4}\left(\frac{0.33}{2}\right)^2 4 = 0.00611\ Kg\ m^2$$

The thrust-weight ratio is given by the maximum thrust that an engine can deliver multiplied by the number of engines and divided by the mass of the aircraft. So according with the data reported in the motor datasheet (page 28) supposing a constant Voltage of 11.1V, a LiPo battery whose weight is the one reported in the Plant data table and 8045 propellers we can calculate

$$Ratio = \frac{Pull}{Weight} = \frac{0.465 * 4}{0.673} = 2.764$$

Basically it is always a good rule of thumb to have a thrust ratio of at least 2 : 1, i.e. the load should never exceed half of the maximum thrust that the motors are able to provide overall.

From the calculations we have just made we can see that if necessary it could still increase the weight of the quadcopter without significantly affecting the performance, for example mount a larger battery to increase its autonomy. We also remember as a clarification that increasing the capacity of the battery does not always lead to greater autonomy; in fact a larger battery also means a greater weight for the motors, and since the discharge of the battery strongly depends on the current consumed by the motors during the activity the correct approach would be to solve a multi-objective optimization problem in which you want to maximize the autonomy variable by minimizing the weight of the vehicle. All valid (or efficient) solutions will be found on the Pareto frontier defined for the specific problem. Roughly (and not ideally) you always prefer to keep a little wider than the ideal ratio.



The ideal max weight slack parameter provides the complementary information to the empirical rule just given, and therefore tells us how many grams we can increase the weight of the quadcopter to reach the full pull ratio of 2 : 1.

14 MATLAB / SIMULINK model

Having a good mathematical model or a simulation system is a fundamental part of the development of a digitally controlled dynamic system. There are many systems for which it is possible in a more or less secure way to develop control systems in an empirical way based on the observation and correction of parameters based on the response that the system provides with respect to an input.

Systems such as quadcopters or small-sized drones can be calibrated in this way, but nevertheless this will never lead to the optimal setting of the parameters that is possible only through the use of tools that allow you to follow the evolution of the dynamics of the system over time.

Therefore a simple mathematical model of the F330 used for this project was created; this model will help us to understand which values can be set as good flight parameters and vice versa which values should be avoided.

However, there are systems, such as airplanes and rockets, that cannot be adjusted based on observing the system's response to inputs and correcting parameters, but must be produced with the certainty that the settled control system is fully functional.

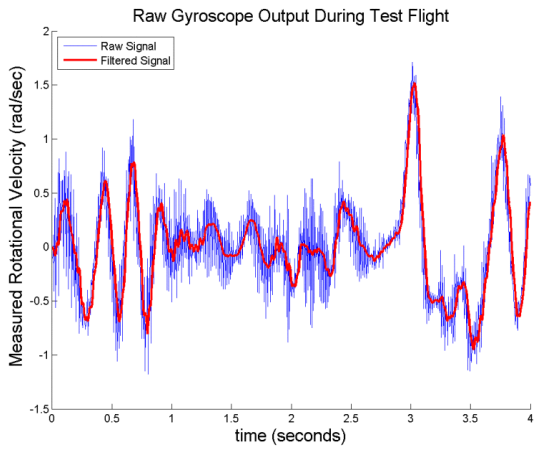
Creating mathematical models with a high degree of reliability is very difficult and often the model is programmed on the specific data collected on the components of the vehicle. Mechanical components such as motors are never perfectly the same, which means that each of them has slightly different characteristics compared to the others. The differences between nominally equal components are minimal but however they can heavily influence the dynamics of the system.

For example, in the quadcopter, if the motors, ESCs and propellers are perfectly the same then no control would be necessary to take-off and hover; in reality without an efficient stabilization system the quadcopter would not even be able to leave the ground.

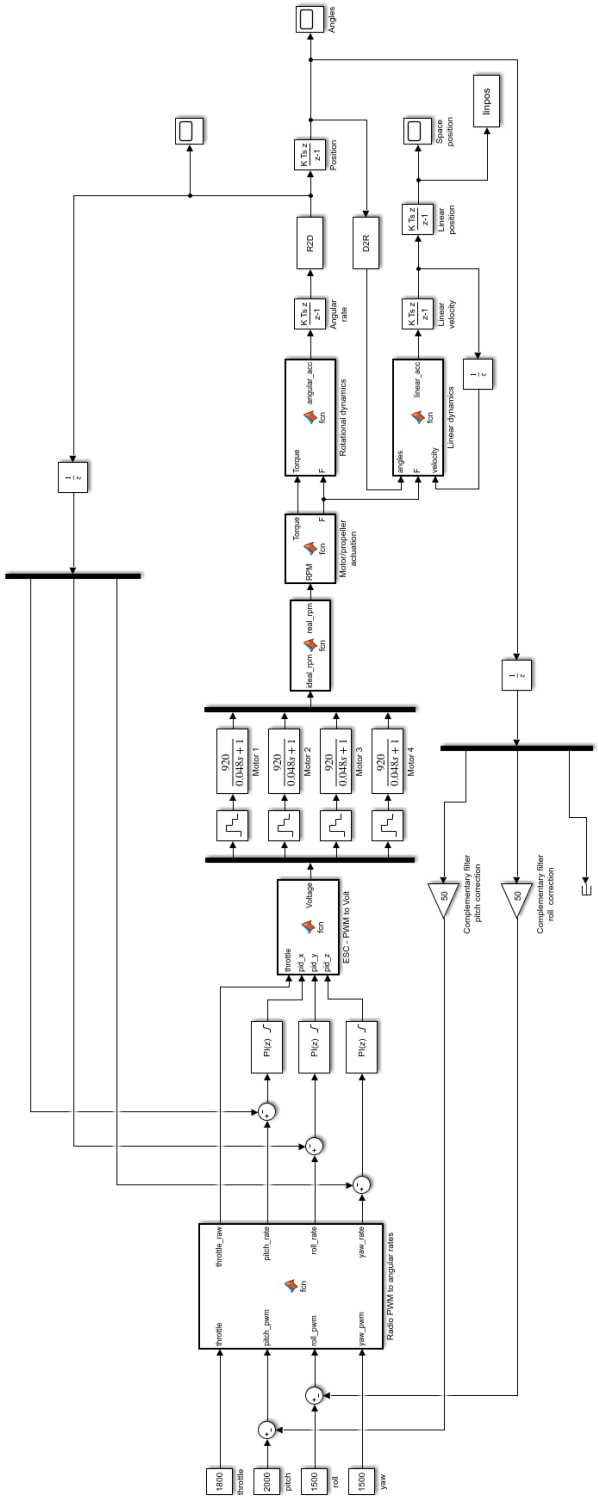
The model we created, shown on the following page, shows the complete dynamics of the system and the control loop.

The quadcopter can be controlled in the two flight modes ACRO and STABILIZED and to do this a control system with double feedback has been implemented. The first system stabilizes the quadcopter based on the angular speeds (or angular rates or simply rates) on the 3 axes X, Y, Z and this is exactly what happens in the ACRO (or RACE) mode. In STABILIZED mode, instead, angles are used to limit the inclination of the quadcopter on the pitch and roll axes which are X, Y acting directly on the radio command given by the operator.

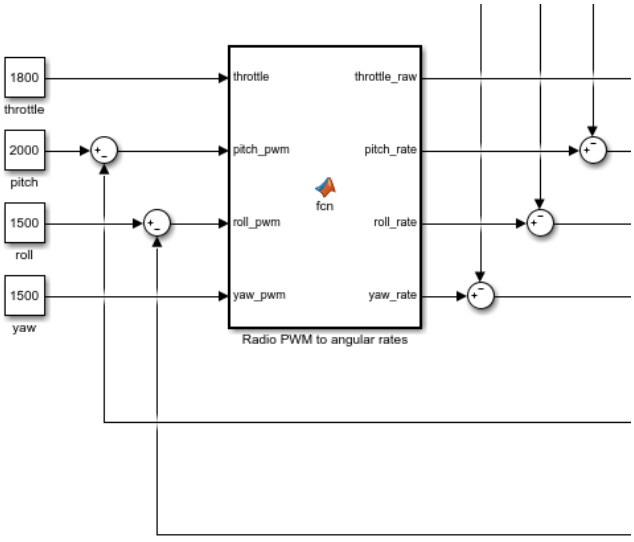
Ideally, since the gyroscope returns the value read in deg / s, it would be sufficient to simply integrate the gyroscope to obtain the position to be input to the second feedback loop, but this is not the case; as already mentioned, the sensors are not ideal machines, therefore a further step will be necessary in the practical implementation which will allow to efficiently estimate the position. We are going to explain this step later on in this text.



The same consideration can be made on angular speeds, in fact in the model we have exactly what is expected from the mathematical formulas that describe the drone's dynamics but nevertheless being able to be the output of these effective formulas it happens that the sensor detects a data different from the real one, more or less similar but not exact. The figure on the left shows why in the real system we can't rely on the raw data arriving from the sensors which must be appropriately interpreted in order to be used by the control system as closely as possible to how they are used in the MATLAB/SIMULINK model.


Angles
linpos
Space position
K.Ts.z z-1 Position
R2D
D2R
K.Ts.z z-1 Linear position
K.Ts.z z-1 Angular rate
K.Ts.z z-1 Linear velocity
angular_acc fcn
Torque
F
Rotational dynamics
linear_acc fcn
angles
velocity
F
Linear dynamics
1 z
Torque
RPM
F
fcn
Motor/propeller actuation
ideal_rpm
real_rpm
fcn
920 0.048s + 1 Motor 1
920 0.048s + 1 Motor 2
920 0.048s + 1 Motor 3
920 0.048s + 1 Motor 4
Voltage
fcn
throttle
pid_x
pid_y
pid_z
ESC - PWM to Volt
PI(z)
PI(z)
PI(z)
50
Complementary filter pitch correction
50
Complementary filter roll correction
throttle_raw
pitch_rate
roll_rate
yaw_rate
fcn
throttle
pitch_pwm
roll_pwm
yaw_pwm
Radio PWM to angular rates
1800 throttle
2000 pitch
1500 roll
1500 yaw

15 Radio PWM and voltage simulation



The control function maps inputs provided by the operator in the corresponding angular speeds on the three axes in deg/s to then be used directly for the calculation of the error that will be forwarded to the controllers at the input.

It is in this block that both the sensitivity of the vehicle with respect to the movement of the radio sticks and the maximum rotational rate that can be achieved are determined.

Particular attention should be paid to the following limitation: The maximum angular speed that can be reached by the quadcopter in absolute value on the three axes must never be greater than the absolute value of the maximum full scale value set for the gyroscope.

If this constraint is violated there would be undesirable and non-deterministic behaviors of the medium therefore protection and saturation mechanisms of the outputs have been included in the software within the limits allowed by the sensor settings.

Even the signal sent by the PWM (Pulse Width Modulation) remote control is not perfect, in fact this too is subject to noise and interference. This can create problems and become visible when no command is given by the radio operator, or better when the command to maintain the position or 0 deg/s is given on all axes; it doesn't mean that the problem vanishes when the system is moving but the effect is much more significant and the signal ripple is well dampen by the system major dynamics.

Ripple at rest is corrected by introducing a small deadband around the central signal, 1500µs, within which the desired value continues to be 0 deg/s. This value can be freely addressed but it is not useful to have more than ±8 µs in that band both because it is not needed by the system and because for the way in which the input subsystem was developed the greater the bandwidth the smaller the setpoint range of the system when coming out from the 0 deg/s command area. Of course a greater bandwidth means that we have to move the stick wider to get the same result with respect to a lower bandwidth, therefore it is suggested and preferable to have a lower bandwidth with a greater setpoint divider than the opposite.

For each radio channel, apart from throttle which is the only raw data that remains in raw format for the whole loop, we can calculate the setpoint dividing the PWM range [1000µs;2000µs] in two specular parts respectively for positive and negative rotation rates with the following math:

```
desired_angle_rate_x = 0;  //default value for value included in the deadband

if( radio_channel_pulse[PITCH] > DEADBAND_UPPER_BOUND ){
    desired_angle_rate_x = ( radio_channel_pulse[PITCH] - DEADBAND_UPPER_BOUND ) * SW_PITCH_REVERSE;
}
if( radio_channel_pulse[PITCH] < DEADBAND_LOWER_BOUND ){
    desired_angle_rate_x = ( radio_channel_pulse[PITCH] - DEADBAND_LOWER_BOUND ) * SW_PITCH_REVERSE;
}

desired_angle_rate_x = desired_angle_rate_x / SETPOINT_PITCH_SPEED_RATE;
```

The example just reported above is specific about the pitch command but it works exactly the same for also roll and yaw. The SW_XXXX_REVERSE is just a setup parameter used in some radio to swap between the positive and negative setpoint halfs because for efficiency reason some protocols transmit the channel reversed; so if the channel is correct its value will be 1 meanwhile in all the case there will be the need of swap the channel it will be settled to -1.

SETPOINT_XXXX_SPEED_RATE is the value that will map the stick range; the lower the value the higher the reachable setpoint. It is exactly this parameter we told before in the previous page about the gyroscope rate constraint; in fact this float value can be settled by the pilot based on his preferences but it has to be choiced in the way that the following value must be less or equal to GYRO_DPS_FULL_SCALE (generally 500 deg/s in MPU-6050).

$$| (2000 - DEADBAND\_UPPER\_BOUND)/SETPOINT\_XXXX\_SPEED\_RATE |$$

16 Motor ideal model and approximation

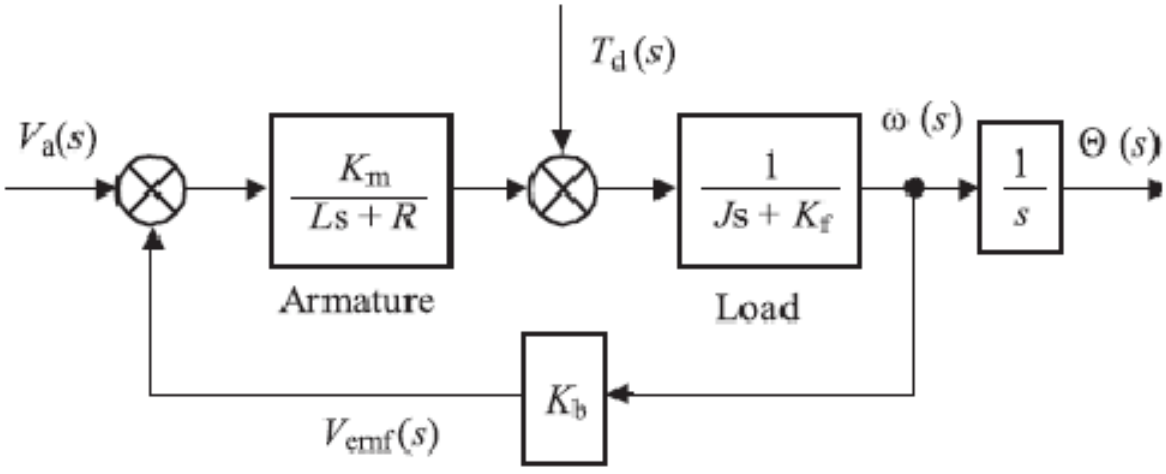
Any system that conveys a mechanical, or in any case not exclusively mathematical, actuation system has its own dynamics which must be characterized for a correct response of the system over time.

Each actuation system does not act in immediately but reaches the desired set-point in a not negligible time interval and generally with non-linear behaviour.

Elements such motors have different poles in its transfer function; generally we have mechanical poles and electric poles.

Mechanical poles are much slower than electric ones and so they are the ones that characterize more the motor response. They are related to physical properties and components coupling effect, like bearings around the shaft, that generates heat and drag dispersions that make the real component different and less performing than the real one.

The mathematical model of a DC motor can approximate the Brushless Motor one, so we can describe it by the following block diagram



Th main issue of such model is that the large majority of motors manufacturers, in particular small and cheap ones used in small models, does not provide enough information to describe the motor behaviour in such way.

Analyzing the diagram in the previous page we see that the motor is characterized by two poles (electric and mechanical) and some other specific parameters like

1. K_m is motor's static gain
2. L is motor's inductance
3. R is motor's equivalent resistance
4. J is rotor moment of inertia
5. K_f is the damping coefficient (due to drag)
6. K_b is the back-EMF constant

Brushless motors (BLDC) because of their nature, have very little friction due to the fact that few moving parts are in contact inside them so most of the time the damping coefficient b is in practice so small that can be neglected without significant losses.

The fraction on the left in the open-loop function describes the electric pole while the one on the right the mechanical pole which is the dominant one as it is much slower than the other. Being slower, it affects more the response of the engine.

The motor constant K_m corresponds in our case to the motor's speed constant expressed in KV (RPM per Volt) which in our case is KV = 920 or 920 RMP/V. This value represents the ideal no-load gain of the motor for each Volt applied to the input, without considering any kind of loss.

Electric motors have a particularity compared to all other electric actuators; they are the only devices that in their operating cycle also operate as generators. This means that when activated they generate an inverse magnetic field that opposes

the one that generates the movement.

Back-EMF is characterized by the constant K_b expressed as

$$K_b = \frac{V_{peak}}{\omega_{no-load}} = \frac{V}{\frac{rad}{s}}$$

By the last formula we can understand why angular velocity of the rotor is reported as feedback variable, in fact

$$V_{feedback} = K_b \omega = \frac{V}{\frac{rad}{s}} \frac{rad}{s} = V$$

This means that the motor continues to accelerate until the difference between the applied and generated magnetic field is different from zero or equivalently until the voltage of the applied magnetic field is greater than the back-EMF one.

When reference voltage and feedback voltage equal each other the motor stops increasing its speed and starts to keep the speed as constant as possible.

If we integrate the output of the system ω we can retrieve angular position, so mathematically for the rotor position Θ we have

$$\Theta = \int_0^t \omega \, dt$$

But for our project, most of this data are not provided by the manufacturer, so we need to do it differently.

Manufacturers have been explicitly asked for data relating to the real behavior of the engines, and in particular the drop in efficiency (or losses with respect to the ideal behavior) due to the increase in RPM and we have been informed that at full speed (100% throttle) it has a loss from 20% to 25% compared to the ideal value with an approximate loss trend with a good precision margin to the quadratic one.

Because of that, for our model we wanted to consider, reasoning with respect to the worst case, a loss of 25 % with a quadratic trend that corresponds to the greatest loss found on the engine model we used (Racestar BR2212 920KV). We believe it is always safer to think about the worst case as it represents the lower bound under which mechanical problems could be encountered, while as long as you are above this the system will surely respond correctly with respect to the model.

This assumption led us to find, identify and calculate the following data shown in the table below

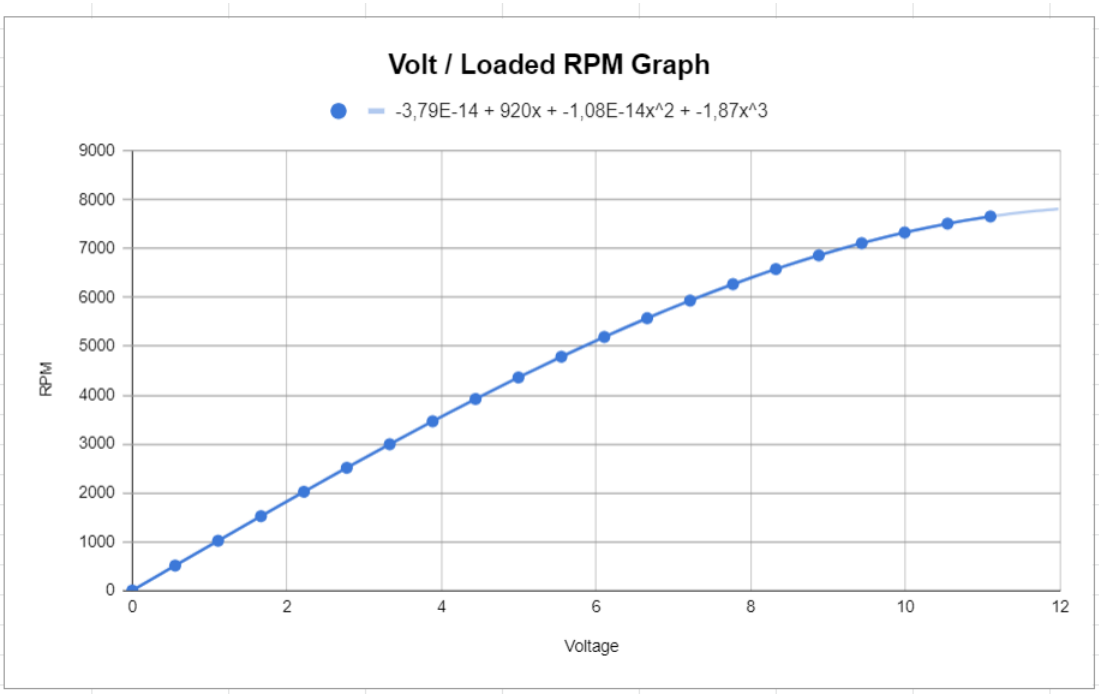
RPM equation with load assumption				
Measurement nr.	**Voltage**	**Drop %**	**No load RPM**	**Loaded RPM**
0	0	0,00%	0	0
1	0,555	0,06%	510,6	510,29364
2	1,11	0,25%	1021,2	1018,647
3	1,665	0,56%	1531,8	1523,22192
4	2,22	1,00%	2042,4	2021,976
5	2,775	1,56%	2553	2513,1732
6	3,33	2,25%	3063,6	2994,669
7	3,885	3,06%	3574,2	3464,82948
8	4,44	4,00%	4084,8	3921,408
9	4,995	5,06%	4595,4	4362,87276
10	5,55	6,25%	5106	4786,875
11	6,105	7,56%	5616,6	5191,98504
12	6,66	9,00%	6127,2	5575,752
13	7,215	10,56%	6637,8	5936,84832
14	7,77	12,25%	7148,4	6272,721
15	8,325	14,06%	7659	6582,1446
16	8,88	16,00%	8169,6	6862,464
17	9,435	18,06%	8680,2	7112,55588
18	9,99	20,25%	9190,8	7329,663
19	10,545	22,56%	9701,4	7512,76416
20	11,1	25,00%	10212	7659

Referring to the motor datasheet, reported on page 28, we see that the reference voltage is 11.1V that is the sum of the nominal voltages of the cells of a 3S LiPo, in fact 3.7V x 3 = 11.1V. Now we divided the motor operating range in 20 slices excluding step 0.

Dividing 11.1 / 20 we obtain 0.555 that is the step increment of each data reported in the voltage column.

The third column represents the square drop value we said before for our motor; note that as RPMs increase the drop becomes more and more significant. This because of the air resistance and the heat generated inside the motor. In fact if the gap between ideal RPMs and real RPMs is 319,125 at middle throttle (5.55V) at full throttle we have 2553 RPM of gap that is 8 magnitude orders bigger than the middle voltage one.

Placing the calculated data in the table in a graph, the function shown in the figure below is obtained

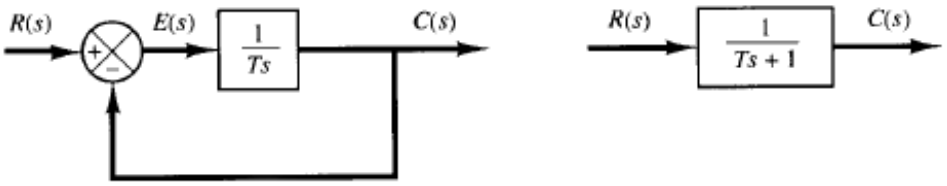


The blue points are exactly the one calculated in the table while the darker blue is the function calculated from excel. Light blue function is best fitting polynomial we found in order to be as close as possible to the function, and as show we achieved our goal with a third order one and in particular

$$RPM = -1.87V^3 - 1,08 * 10^{-14}V^2 + 920V - 3.79 * 10^{-14}$$

This function, as we will see in detail shortly, is not directly applicable to the model because it is not mapped in the time domain, and therefore consequently not even in the frequency domain, and it would lead to instantaneous actuation for each control loop which would make the result found unrealistic. We introduced this chapter about motors underlining the fact that these are characterized by their own dynamics which require a certain not negligible time to react to an input; this behavior can be greatly approximated by a first-order system.

Recalling what is a 1^{st} order system we find that: "A system whose input-output equation is a first order differential equation is called first order system". The order of the differential equation is the highest degree of derivative present in the equation itself.



Its transfer function (see page 85 for more details) according to the block algebra of a feedback closed-loop is given by

$$C(s) = \frac{G(s)}{1 + G(s)} = \frac{\frac{1}{Ts}}{1 + \frac{1}{Ts}} = \frac{1}{Ts + 1}$$

The parameter T is most of the times called τ and it is the parameter characterizing the response to a step input of a first-order Linear Time-Invariant (LTI) system. Time constant is the main characteristic unit of a first-order LTI system.

Physically, the time constant represents the elapsed time required for the system response to decay to zero if the system had continued to decay at the initial rate, because of the progressive change in the rate of decay the response will have actually decreased in value to $1/\mathrm{e} \approx 36.8\%$ in this time (from a step decrease). In an increasing system, the time constant is the time for the system's step response to reach $1 - 1/e \approx 63.2\%$ of its final asymptotic value (from a step increase).

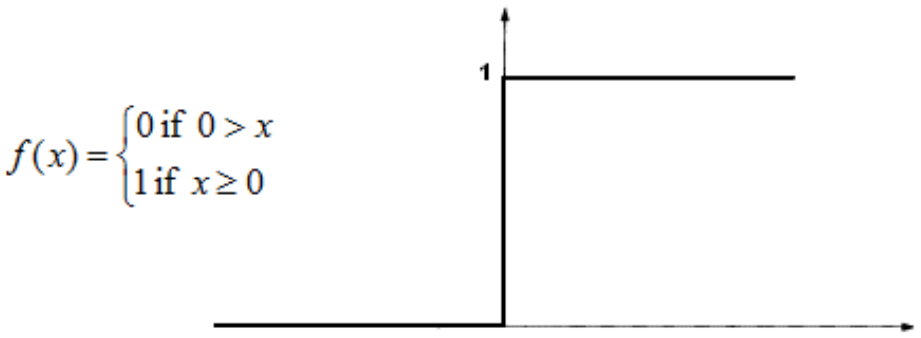
The step function, also called Heaviside step function after Oliver Heaviside (1850–1925), usually used to test the behaviour of such systems can be defined in its simplest definition as the time derivative of a ramp so

$$H(x) = \frac{d}{dx} max\{x, 0\} \qquad \forall x \neq 0$$

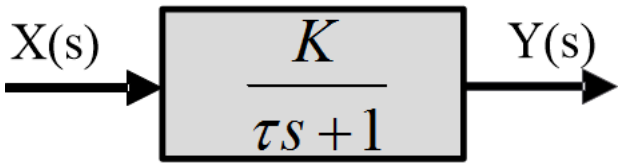
In the discrete time domain it is defined as

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

and its graph is

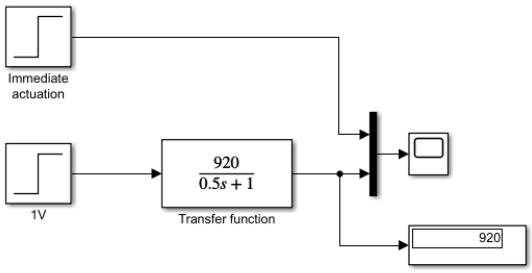


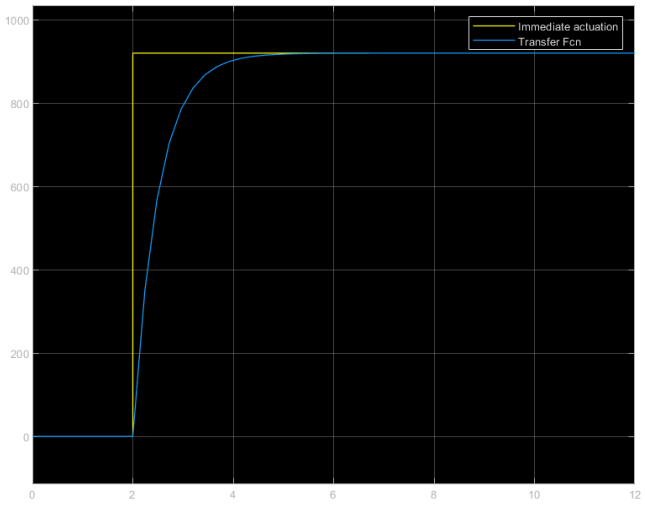
Coming back to the first order transfer function, this allows and describes a characteristic delay, i.e. the transient that the motor takes to reach the setpoint which is characterized more by the time constant τ previously discussed.



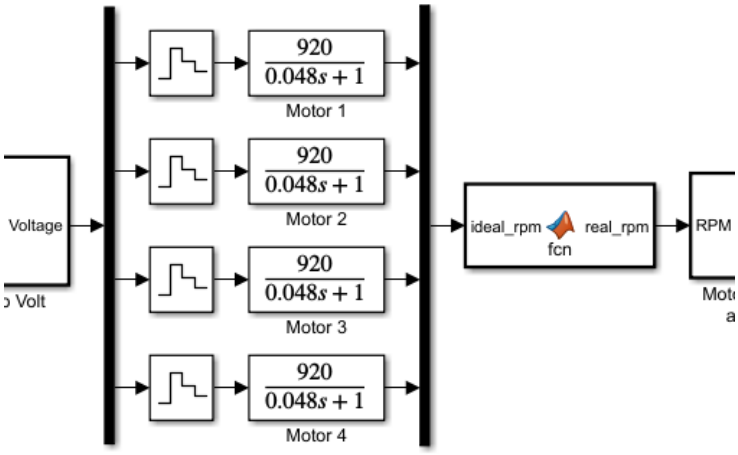
K is the static gain of the system which is often called DC-gain and it is the increase of system output for each unit presented in the input. In our case since the input is the Voltage supplied to the motor and we know the motor KV constant, the latter is exactly the DC-gain of our system which corresponds to 920.

This means that if the input to the motor t.f. is a 1V step function we expect to see the motor reach 920 in a certain not negligible time defined by the τ value. Below is reported what just told in SIMULINK for $\tau = 0.5$ and K = 920 and simulation time of 12 seconds.





It is now clear that if the input is not a unit step, but for example a 2 step the setpoint will be 920x2 = 1840 and so on until the maximum possible voltage for the plant. This brings us back to the drop relation described at page 63, in fact t.f. is a linear mapping between inputs and outputs while our system is not linear at all. We need to filter the output of the transfer function using a matlab_function to map the ideal RPM which is the output of the t.f. with respect to the RPM drop explained before.



This is the part of the model where motors transfer functions take place, in particular Voltages coming from the ESCs dynamics are demuxed in order to be sent as input to the relative motor. After transfer functions RPMs are muxed again and presented as input to the function that will introduce the non-linearity of the motor before calculating the plant dynamics vectors, torques and moments.

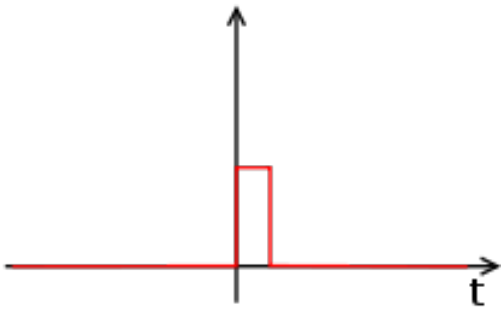
Before each t.f. a Zero Order Hold (ZOH) had been placed, this because we used a continuous time transfer function in a discrete time plant.

The Zero Order Hold (ZOH) is a mathematical model of the practical signal reconstruction done by a conventional Digital to Analog Converter (DAC). It describes the effect of converting a discrete-time signal to a continuous-time signal by holding each sample value for one sample interval.

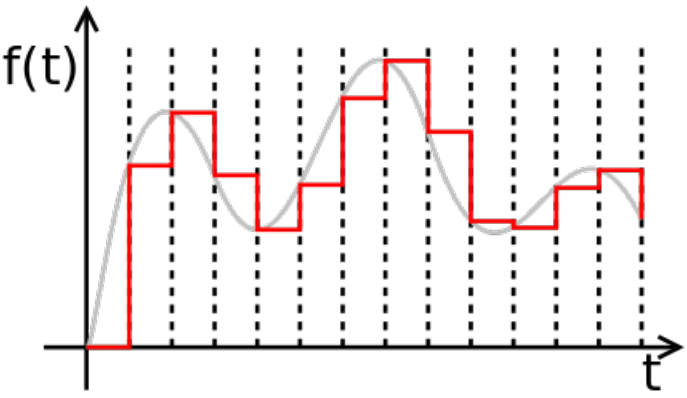
A Zero Order Hold reconstructs a continuous time waveform from a sequence $x[n]$ of samples, assuming one sample per time interval T

$$x_{\mathrm{ZOH}}(t) = \sum_{n=0}^{\infty} x[n] \cdot \mathrm{rect}\left(\frac{t - T/2 - nT}{T}\right) \qquad rect\left(\frac{t - T/2}{T}\right)$$

The function $rect()$ is a rectangular function whose shape is reported in the figure below

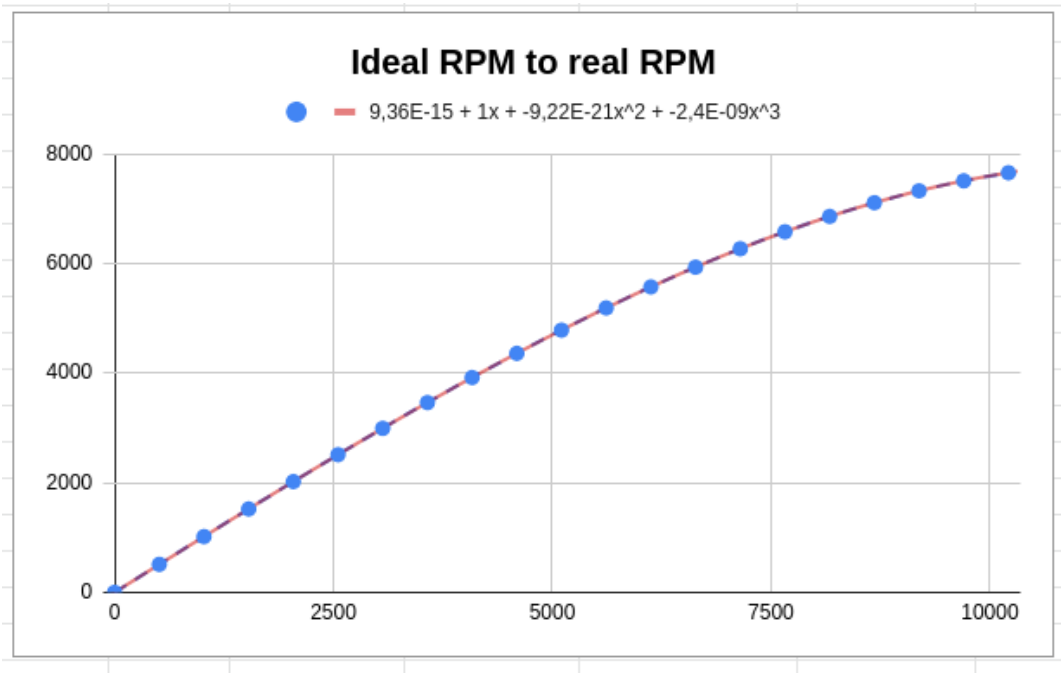


ZOH reconstructs the signal using the rect functions as its basic unit in the following way



The last piece of this controller section is the matlab_function that maps ideal RPM to real RPM. As we told in page 64, we need another relation that binds voltage and real RPM to ideal RPM and real RPM.

This is possible using the last two columns of the table reported in page 62, in fact the no-load RPM matches exactly the ideal RPM. Therefore, as we did before, it is possible to set up a graph in order to find the relation between the two measures.



On the X axis ideal RPM are represented while on the Y axis we have real RPM. As in the previous graph, the points represent the known values from the table while the blue dotted line is the function obtained by excel starting from the known points.

The red line is the best fitting polynomial, that after some tests we fixed to the third order because polynomial of higher degree would not entail any benefit, rather they would be harmful because they would contribute in a completely useless way to make the whole simulation runtime environment more complex.

The function found is the following where $x = RPM_{ideal}$

$$RPM_{real} = -2.4 * 10^{-9}x^3 - 9.22 * 10^{-21} * x^2 + x + 9.36 * 10^{-15}$$

17 Propellers model

Modeling the implementation system of a quadcopter is not only based on the engine but a fundamental part of the whole system is given by the propellers, in fact these characterize factors such as the thrust at a given speed and depending on their conformation they also determine how many RPM are lost compared to to the ideal ones described by the motor speed constant expressed in KV.

In this case the chosen engines are 920KV Racestars which means that nominally for each Volt applied to the engine this should increase its rotation speed by 920 rpm. Due to the friction of the air and the internal dispersions of the engine due to the friction and the poor coupling of the mechanical components, the value will never be a function of the volts but will suffer a loss as previously said, which assumes a trend very close to the quadratic one with respect to the increasing of the RPMs.

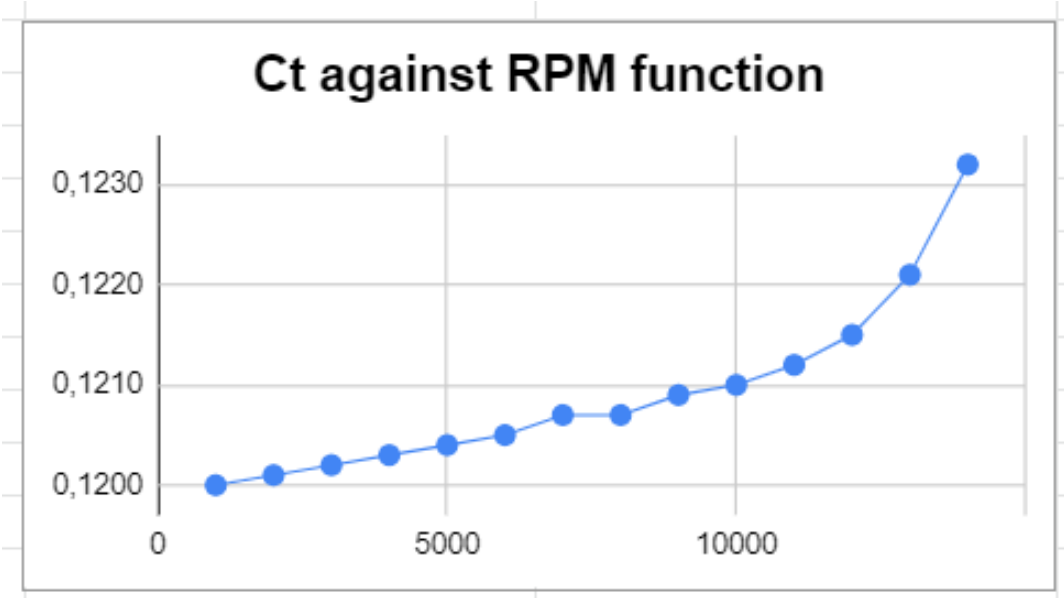
To know the thrust characteristics of a propeller, it is first necessary to calculate the thrust coefficient called C_t.

C_t is not dependent from the specific motor but depends on the RPM. For our purpose we used 8045 propeller with two blades whose C_t can be obtained by interpolating experimental data reported in the APC file attached at the end of the report (page 122).

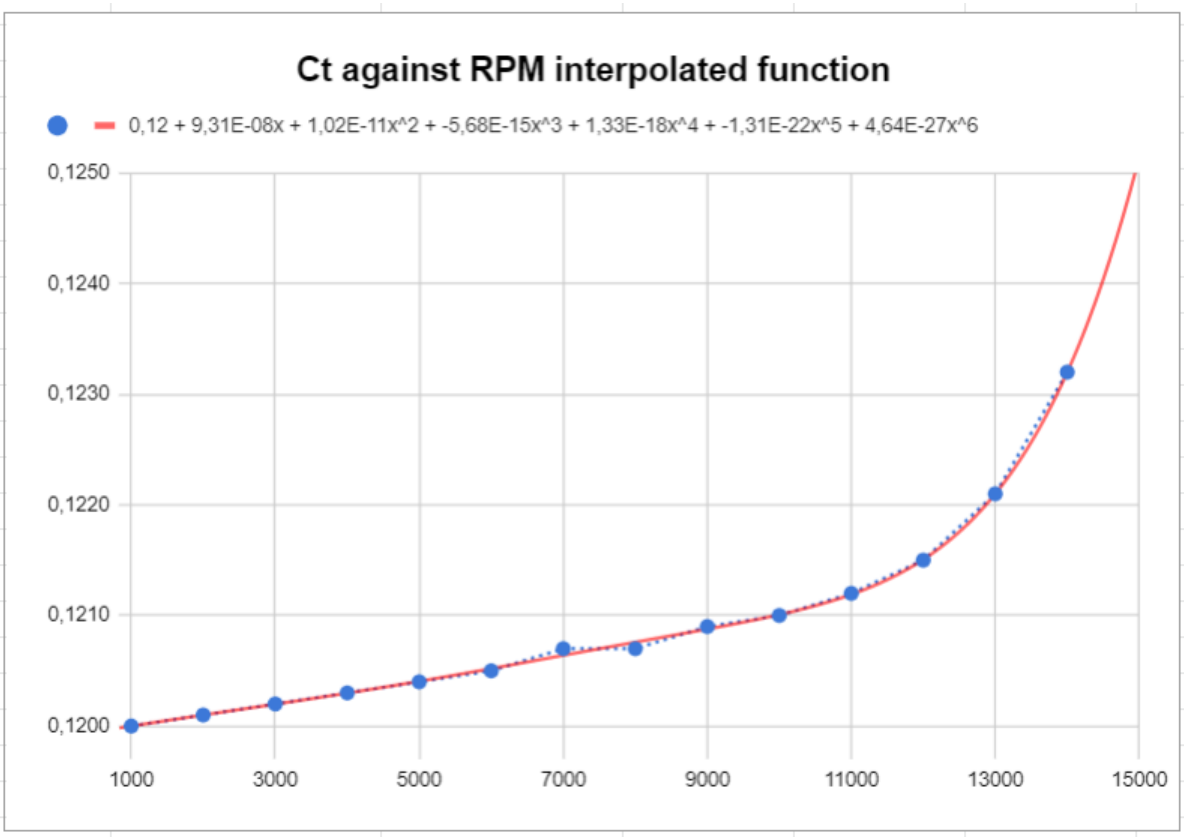
Since we did our analysis in the hovering point we have to consider data in the line of 0 forward speed and all the data must be converted in S.I. values. Converting data as just told leads us to create the following table

Thrust Coefficient equation from exp. data	
RPM	Ct for 0 forward speed
0	0
1000	0,1200
2000	0,1201
3000	0,1202
4000	0,1203
5000	0,1204
6000	0,1205
7000	0,1207
8000	0,1207
9000	0,1209
10000	0,121
11000	0,1212
12000	0,1215
13000	0,1221
14000	0,1232

We reported this table in Google sheets to get the tween function with the following result



The blue points on the function above are the points described in the table, now we need to get an interpolating function as close as possible to the blue one in order to obtain the precise C_t for a given RPM value.

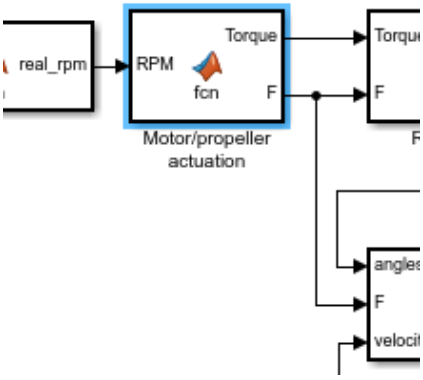


The best fitting polynomial that solves the problem is of degree six and it is

$$4.64 * 10^{-27} * x^6 - 1.31 * 10^{-22} * x^5 + 1.33 * 10^{-18} * x^4 - 5.68 * 10^{-15} * x^3 + 1.02 * 10^{-11} * x^2 + 9.31 * 10^{-8} * x + 0.12$$

We can observe that the effective range of use is around [1100;10000] RPM, so theoretically we can find a function that fits only such interval.

18 Torque and Force



The force created by each motor can be calculated, once the thrust coefficient C_t is known, in direct proportion to the square of the rotations per second and the propeller diameter to the fourth with also taking into account the air density constant.

The formula is

$$F = C_t * \rho * \left(\frac{RPM}{60}\right)^2 * D^4$$

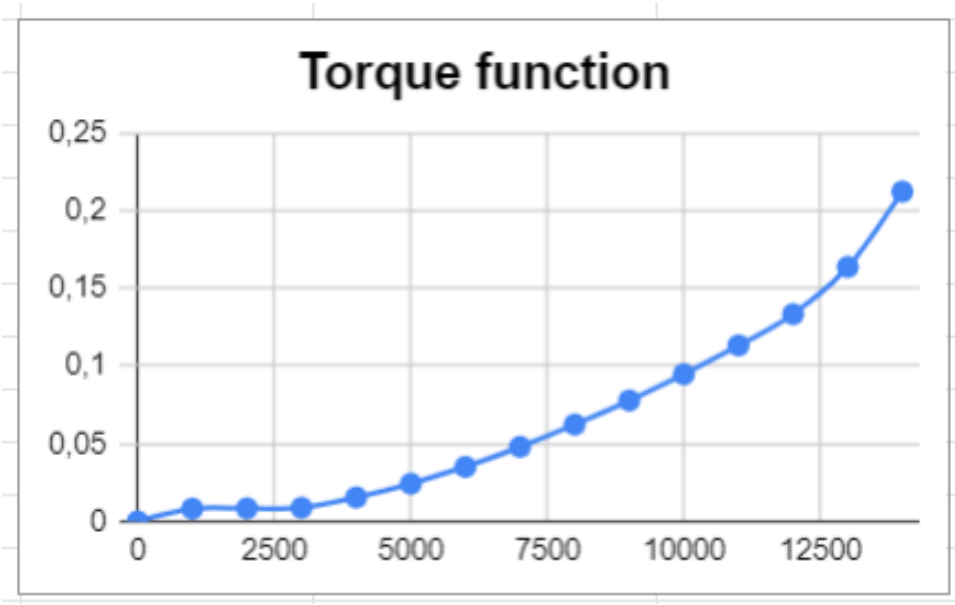
The parameters ρ and D are respectively chosen in our project as $\rho = 1.225\ \frac{kg}{m^3}$, that is air density constant at sea level and D, which is 8 inches, was converted to meters so we have D = 0.2032 m.

Torque is again calculated as a function of the RPMs and it is also obtained from APC data.

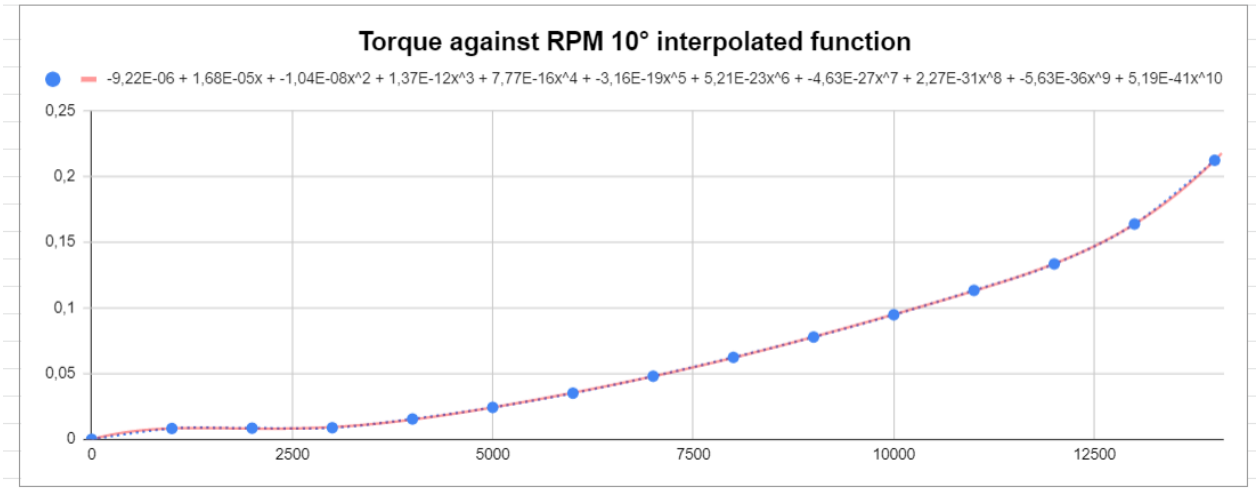
For 8045 propellers for 0 m/s forward speed and converting in standardized Nm unit we obtained the table below

Torque Coefficient equation from exp. data		
RPM	Exp. torque (In-Lbf)	Exp. torque (Nm)
0	0	0
1000	0,073	0,008247905
2000	0,075	0,008473875
3000	0,078	0,00881283
4000	0,137	0,015478945
5000	0,215	0,024291775
6000	0,311	0,035138335
7000	0,425	0,048018625
8000	0,552	0,06236772
9000	0,689	0,077846665
10000	0,839	0,094794415
11000	1,003	0,113323955
12000	1,181	0,133435285
13000	1,450	0,16382825
14000	1,879	0,212298815

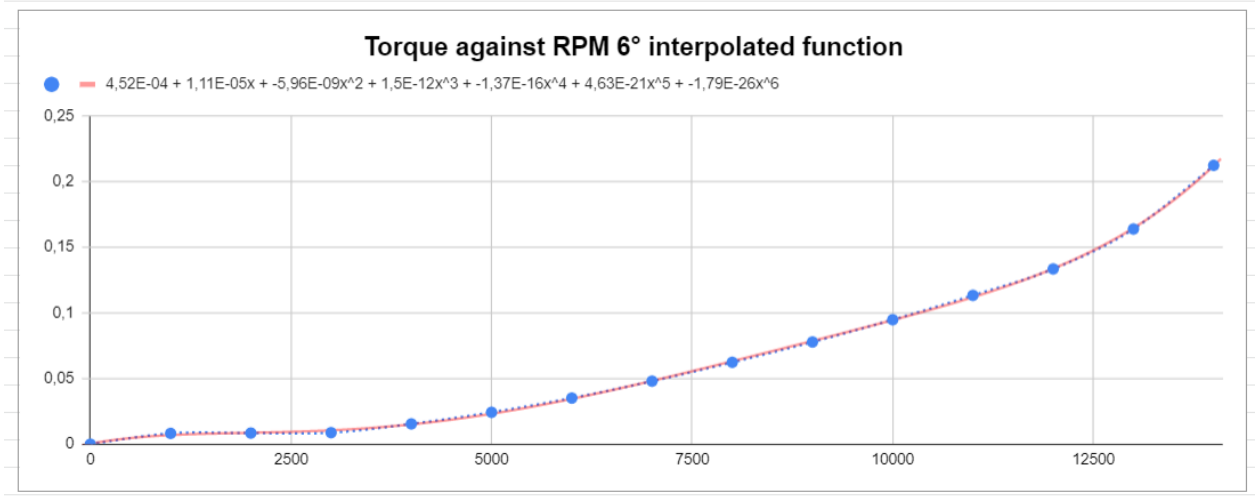
As we did before, for the thrust coefficient we can use google sheets to plot the function describing the Toque against RPM curve obtaining



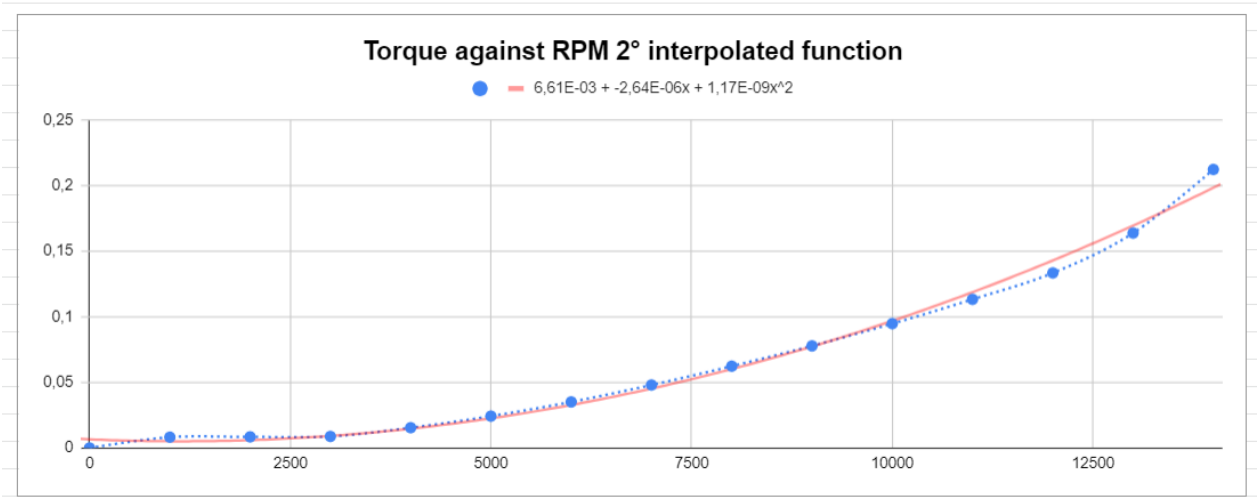
Again the area of interest of that function is more or less between [1000 RPM;10000 RPM] so we can find different interpolating polynomials as long as the interest range is close enough to the blue function.



The tenth degree polynomial is the overall best one but it is a complex expression and it is a good practice trying to simplify the MATLAB/SIMULINK runtime environment every time it is possible without affecting too much the problem.

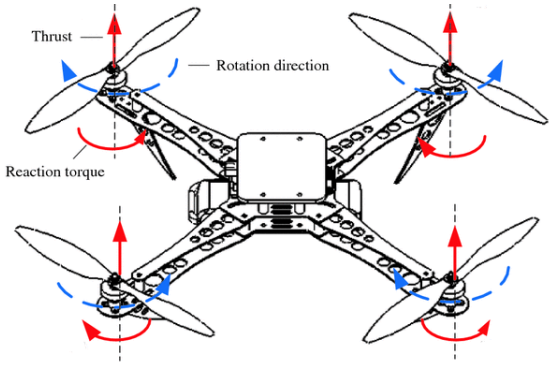


Lowering the fitting degree we still obtain a good approximation of the original function.

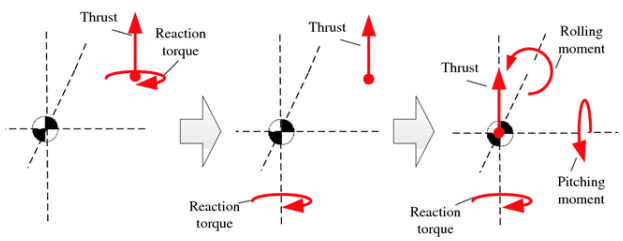


Considering only the effective operative range of the motors we can conclude that the best compromise for our model is given by the function of degree two

$$T(x) = 1.17 * 10^{-9} * x^2 - 2.64 * 10^{-6} * x + 6.61 * 10^{-3}$$

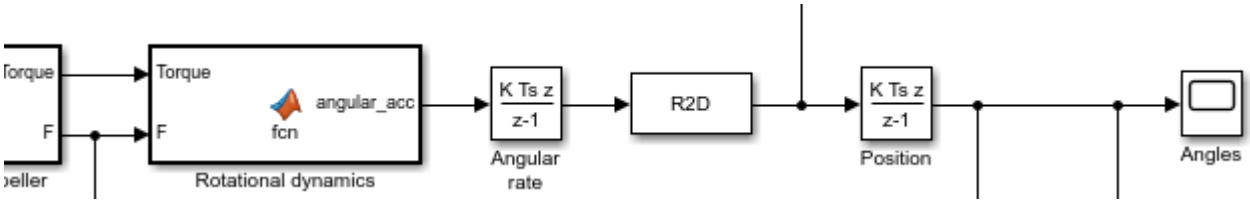


(a) A quadcopter in hovering flight



Torques and Forces are the passed to the rotational dynamics block while only Forces are also passed to the linear dynamics one.

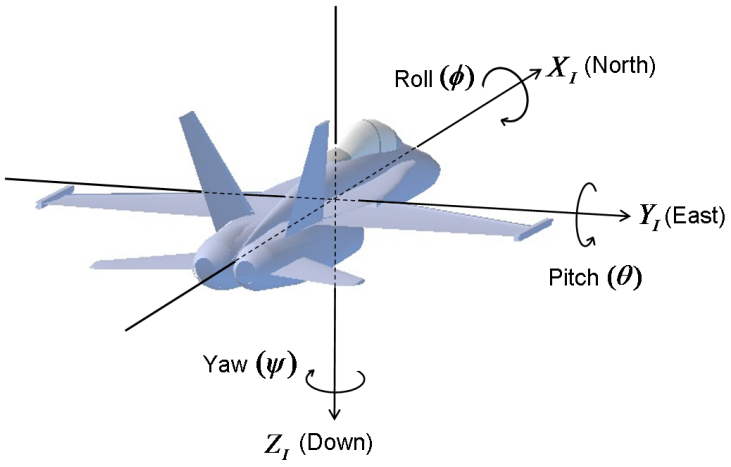
19 Plant process - Rotational dynamics



If the reference frame is well choiced the Inertia tensor, defined as reported in the left simply become as in the right because all the mixed cross product simplify each other

$$\begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \qquad \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

The external angular momentum is described by the following formulas referring to the axis as reported in the figure below



$$M_x = I_{xx}\ddot{\Theta} \qquad M_y = I_{yy}\ddot{\phi} \qquad M_z = I_{zz}\ddot{\Psi}$$

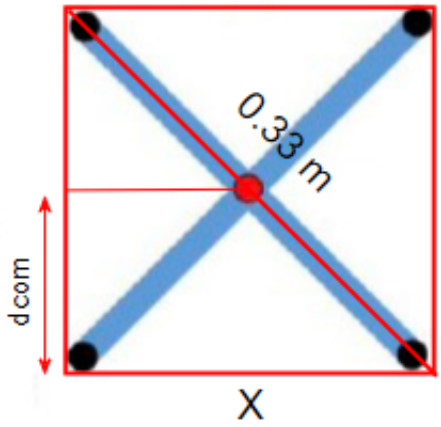
Taking this into consideration we write the equations that describe the specific moments on the three axes of rotation X, Y, Z for our quadcopter

$$M_x = (Force_{motor2} + Force_{motor4})d_{com} - (Force_{motor1} + Force_{motor3})d_{com}$$

$$M_y = (Force_{motor3} + Force_{motor4})d_{com} - (Force_{motor1} + Force_{motor2})d_{com}$$

$$M_z = \quad -Torque_{motor1} + Torque_{motor2} + Torque_{motor3} - Torque_{motor4}$$

where the variable d_{com} is the distance from the CoM (Center of Mass)



that in the case of our frame that is a perfect symmetric F330 is equal to

$$d_{com} = \frac{x}{2} = \frac{\sqrt{\frac{0.33^2}{2}}}{2} = 0.1167m$$

We can now get the rotational acceleration as output of the rotational dynamics block by applying inverse formulas so

$$\ddot{\Theta} = \frac{M_x}{I_{xx}} \qquad \ddot{\phi} = \frac{M_y}{I_{yy}} \qquad \ddot{\Psi} = \frac{M_z}{I_{zz}}$$

and we get the final angular acceleration vector $[\,\ddot{\Theta} \quad \ddot{\phi} \quad \ddot{\Psi}\,]^T$

By integrating the angular acceleration we obtain the angular rate vector that is used as feedback, after the radians to degrees conversion, for the closed loop controller and so mathematically we express this as

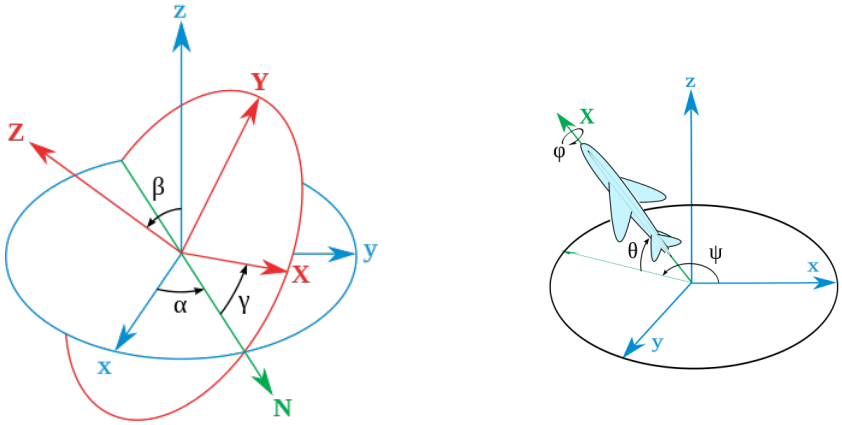
$$angular\_rate = \begin{bmatrix} \dot{\Theta} \\ \dot{\phi} \\ \dot{\Psi} \end{bmatrix} = \begin{bmatrix} \int_0^t \ddot{\Theta} dt \\ \int_0^t \ddot{\phi} dt \\ \int_0^t \ddot{\Psi} dt \end{bmatrix} \Rightarrow \begin{bmatrix} \sum_{k=0}^t \ddot{\Theta} T \\ \sum_{k=0}^t \ddot{\phi} T \\ \sum_{k=0}^t \ddot{\Psi} T \end{bmatrix}$$

If we integrate again we obtain position that is needed for the linear dynamics block we are going to analyze in the next page

$$position = \begin{bmatrix} \Theta \\ \phi \\ \Psi \end{bmatrix} = \begin{bmatrix} \int_0^t \dot{\Theta} dt \\ \int_0^t \dot{\phi} dt \\ \int_0^t \dot{\Psi} dt \end{bmatrix} \Rightarrow \begin{bmatrix} \sum_{k=0}^t \dot{\Theta} T \\ \sum_{k=0}^t \dot{\phi} T \\ \sum_{k=0}^t \dot{\Psi} T \end{bmatrix}$$

20 Euler's angles

Euler angles can be defined by elemental geometry or by composition of rotations. The geometrical definition demonstrates that three composed elemental rotations (rotations about the axes of a coordinate system) are always sufficient to reach any target frame.



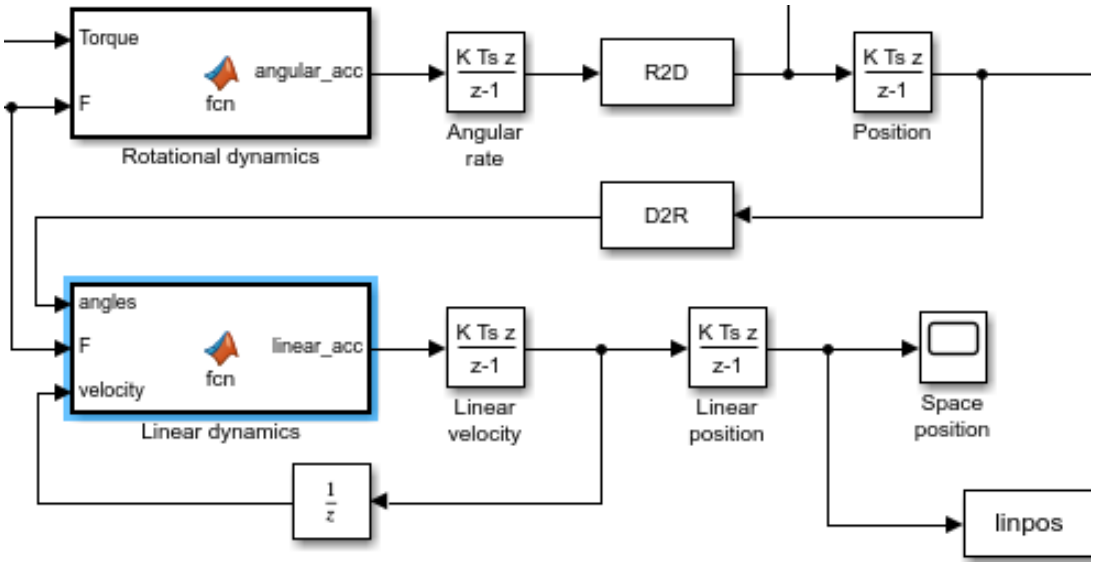
The three elemental rotations may be extrinsic (rotations about the axes xyz of the original coordinate system, which is assumed to remain motionless), or intrinsic (rotations about the axes of the rotating coordinate system XYZ, solidary with the moving body, which changes its orientation after each elemental rotation).

Euler angles are typically denoted as α, β, γ, or as we named Θ, ϕ, Ψ and N is the line of nodes defined as the intersection of the planes xy and XY (it can also be defined as the common perpendicular to the axes z and Z).

Those three angles are defined and bounded in the following way :

1. α is the angle between x and the line of nodes. It is also named as Precession angle and it is defined in the range $[0; 2\pi]$ or $[-\pi; \pi]$
2. β is the angle between z e Z. It is also named as Nutation angle, and it is defined in the range $[0; \pi]$ or $[-\frac{\pi}{2}; \frac{\pi}{2}]$
3. γ is the angle between the line of nodes and X. It is also named as Intrinsic Rotation, and it is defined in the range $[0; 2\pi]$ or $[-\pi; \pi]$

21 Plant process - Linear dynamics



The block of linear dynamics allows to trace the drone's movements in space. Through the data collected, saved and accumulated in the linpos variable on SIMULINK then it will be possible to carry out a trajectory analysis in the medium in three-dimensional space.

It is necessary to immediately notice and pay attention to convert again the angles from degrees to radians, this because the mathematical operations concerning this block are defined on angles with standardized unit radians (functions like atan2); it is not a casuality that the block is positioned as a reminder of the R2D block applied above before to the output of the rotational dynamics block.

Linear dynamics involve Euler's angles to calculate the relative position of the drone in space. The roll (ϕ) and pitch (Θ) angles describe the tilt of the quadrotor relative to the local vertical; i.e., relative to the gravity vector. The yaw angle (Ψ) describes the amount of rotation around the local vertical direction. Typically, the roll and pitch angles are used to control the direction of the quadcopter's total thrust vector.

If they are zero, the thrust vector will point directly upwards opposite to the direction of the gravity vector and that means that the quadcopter could be both moving linearly in the Z axis or hover on its position. Non-zero values of roll and pitch can be used to tilt the thrust vector so that the resultant horizontal component of the thrust vector can be used for translational motion.

Inside the block the three thrust vector components are calculated in the following way

$$F_{prop}(x) = sin(\phi) * cos(\Theta) * (F_{motor1} + F_{motor2} + F_{motor3} + F_{motor4})$$
$$F_{prop}(y) = sin(\Theta) * cos(\phi) * (F_{motor1} + F_{motor2} + F_{motor3} + F_{motor4})$$
$$F_{prop}(z) = cos(\Theta) * cos(\phi) * (F_{motor1} + F_{motor2} + F_{motor3} + F_{motor4})$$

For the yaw axis the 3D plane is projected in 2D coordinates by applying the following math

$$Angle_{xy} = \text{atan2}(F_{prop}(x), F_{prop}(y))$$
$$XY_{2D} = \sqrt{(F_{prop}(x)^2 + F_{prop}(y)^2)}$$

After that we have to check if $F_{prop}(z)$ is greater than 0 or not in order to know if we have to apply a positive Ψ angle or a negative one.

We get the final values for $F_{prop}(x)$ and $F_{prop}(y)$ applying this

$$F_{prop}(x) = XY_{2D} * cos(\pm\Psi + Angle_{xy})$$
$$F_{prop}(y) = XY_{2D} * sin(\pm\Psi + Angle_{xy})$$

If $F_{prop}(z) > 0$ then we apply a positive Ψ otherwise (apart from 0) we apply a negative Ψ.

We also have to remember that also the gravitational acceleration vector acts in the Z direction, therefore the total thrust in Z is given by

$$F_{prop}(z) = cos(\Theta) * cos(\phi) * (F_{motor1} + F_{motor2} + F_{motor3} + F_{motor4}) - mg$$

Finally we use Newton's second law to get the linear acceleration along the three directions

$$F = ma \qquad \Rightarrow \qquad a = \frac{F}{m}$$

So we finally get our three equations

$$\ddot{X} = \frac{F_{prop}(x)}{m} \qquad \ddot{Y} = \frac{F_{prop}(y)}{m} \qquad \ddot{Z} = \frac{F_{prop}(z)}{m}$$

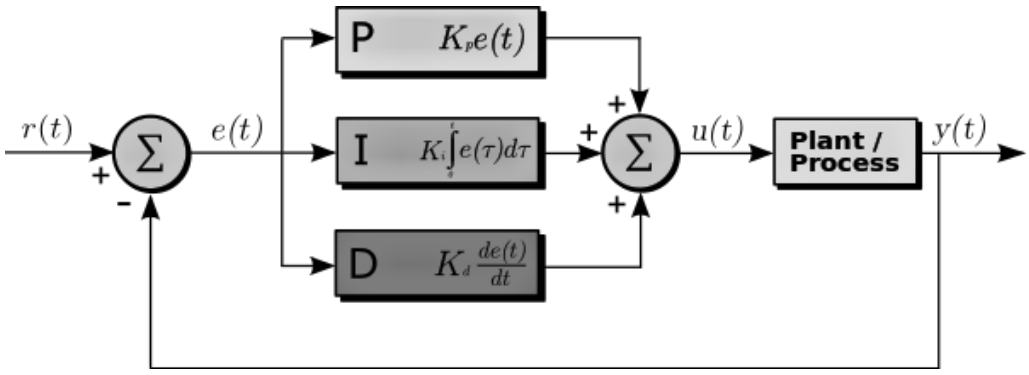
We can again, as we did for the Rotational dynamics block integrate to obtain linear velocity

$$velocity = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} \int_0^t \ddot{X} dt \\ \int_0^t \ddot{Y} dt \\ \int_0^t \ddot{Z} dt \end{bmatrix} \Rightarrow \begin{bmatrix} \sum_{k=0}^t \ddot{X} T \\ \sum_{k=0}^t \ddot{Y} T \\ \sum_{k=0}^t \ddot{Z} T \end{bmatrix}$$

and then if we integrate velocity we obtain the position in the 3D space

$$position = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \int_0^t \dot{X} dt \\ \int_0^t \dot{Y} dt \\ \int_0^t \dot{Z} dt \end{bmatrix} \Rightarrow \begin{bmatrix} \sum_{k=0}^t \dot{X} T \\ \sum_{k=0}^t \dot{Y} T \\ \sum_{k=0}^t \dot{Z} T \end{bmatrix}$$

22 PID



In control theory there are two main kinds of controllers:

1. Open loop controllers
2. Feedback loop controllers (closed loop)

Feedback systems differ from open loop systems because at each loop the system output is reported as input and is added / subtracted to the new reference passed as input as shown in the figure above.

The PID controller is one of the most used mechanisms in industrial applications to control a system in closed loop, and in particular it is a negative feedback loop controller, generally for BIBO stable plants.

Positive feedback is such defined because its operation leads the system to amplify in a positive way, thus adding a corrective value to the state of the system itself. This mechanism is unstable because often, using this approach, the system tends to diverge with respect to the target value and this is mathematically provable.

On the other side, negative feedback leads the system to make a subtractive correction, so the value generated at each iteration tends to damp the system's next state in case of error and this, under controllability and observability conditions, leads the system to converge towards the target.

22.1 Laplace transform and transfer function

T Laplace transform of a function f(t), defined for all real numbers $t \geq 0$, is the function F(s), which is a unilateral transform defined by

$$F(s) = \int_0^\infty f(t)e^{-st}dt$$

where s is a complex number frequency parameter defined by $s = \sigma + i\omega$ where σ and ω are real numbers.

The Laplace transform is really important in system analysis because mapping the system from the time domain to the frequency domain allows to greatly simplify complex calculus like integration e differentiation. In fact in the Laplace domain integration is obtained dividing by the complex variable s and opposite differentiation is obtained multiplying by s.

Laplace transforms are used to obtain the transfer function of the system which is the relation, or the ratio, between the output and the input express as frequency amplitude and so it is defined in the Laplace domain but can be reported in time domain applying the inverse transformation, however this is hardly ever used because is much simpler and effective to work using the variable s.

Transfer functions are commonly used in the analysis of systems such as single-input single-output (SISO systems) in control theory. The term is often used exclusively to refer to Linear Time Invariant (LTI) systems.

The large majority of real systems have non linear input/output characteristics, but when operated within nominal parameters have behavior close enough to the linear one so LTI system theory is an acceptable representation of the input/output behavior, and this is the case of our project since we used nominal values and we did a linear analysis of a non linear system around linearization points (we used hovering condition).

What we just said is valid of LTI-CT systems (Linear Time Invariant-Continuous Time system), so we need to apply another transformation to equivalently map what we have in the Laplace domain (continuous time) to the discrete time LTI-DT system (LTI-Discrete Time system) and that is the Z-transform.

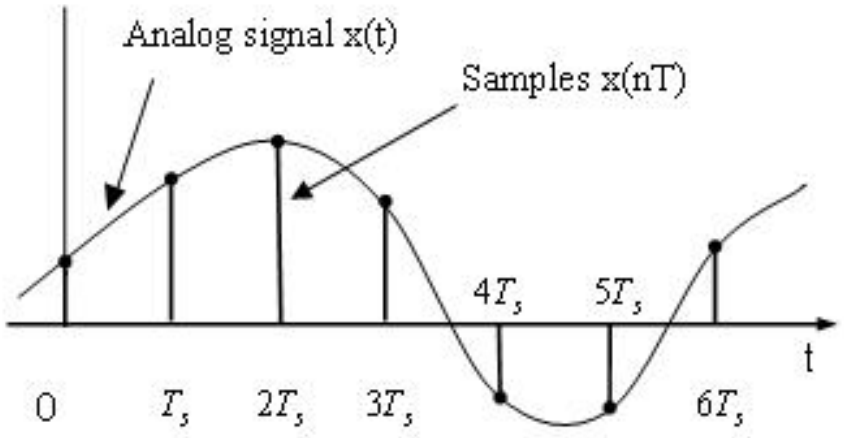
Z-transform is very important in control theory because it maps the system behaviour in the discrete time, and so, it is the only possible concrete study we can do to physically implement the controller on a computing system.

MCUs, CPUs and any kind of computer are discrete time machine because the only notion of time they have is the clock signal that is a square wave with a well defined shape, period and so frequency.

The Z-transform converts a discrete-time signal, which is a sequence of real or complex numbers, into a complex frequency domain representation. It is the discrete-time equivalent of the Laplace transform.

It was formalized by Ragazzini and Zadeh in the sampled-data control group at Columbia University in 1952 and defined (in unilateral transformation) as such geometrical series.

$$X(z) = Z\{x[n]\} = \sum_{n=0}^{\infty} x[n]z^{-n}$$



22.2 P controller

The P component, or proportional, is characterized by a multiplier K_p that is only a constant gain with respect to the calculated error that makes the quadcopter turn in the direction of the zero error (or opposite to the error). The k_p gain value, and it is the power with which compensation will take place. The higher it is, the more the flight controller will send strong inputs to the motors trying to correct the quadcopter position. To have a very stable quadcopter it is important that it is settled as high as possible without introducing high frequency oscillations. If it is too high it can cause persistent oscillation and in extreme cases these oscillations can lead the drone to be completely uncontrollable.

In real life controller is hard to have a pure proportional controller because P controller often leads to persistent steady-state error and it doesn't have any kind of consciousness about how long the error has been persisting in the systems in terms of time.

P controllers take as input the difference between the desired position, or set-point, and the feedback value so

$$e(t) = ref(t) - u(t-1)$$
$$u(t) = K_p e(t)$$

If we analyze the controller by the mean of the Laplace transform we obtain

$$U(s) = K_p E(s) \;\; \Rightarrow k_p = \frac{U(s)}{E(s)}$$

so the t.f. (transfer function) of the Proportional controller is K_p since $e(t)$ is the input of the controller and the t.f. is exactly the ratio between output and input in the Laplace domain.

22.3 I controller

The I component, or integral component, characterized by the multiplier k_i, acts over the persistence with which the variations are compensated. The I parameter determines how long the quadcopter will tend to maintain the desired inclination. The higher it is, the more the quadcopter will tend to remain in the position imparted by the pilot. It must be modified by taking small steps, generally going up 0.002 units at time. It is necessary to ensure that its modification does not induce the quadcopter to low frequency oscillations. The higher it is, the more any corrections in case of external disturbance will be maintained. An external disturbance can be, for example, a gust of wind. It is strongly recommended to set this parameter after both Kp and Kd.

Integral controller can be analyzed in the following way

$$u(t) = K_i \int_0^t e(t)dt$$

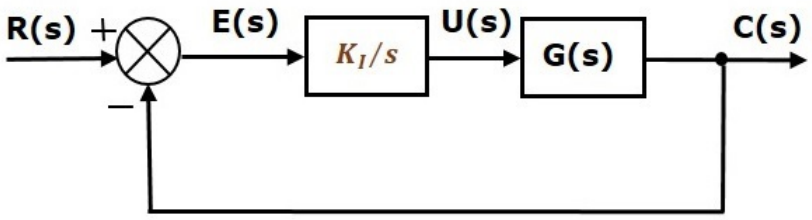
Applying the Laplace transform we obtain

$$U(s) = \frac{K_i E(s)}{s}$$

So the transfer function of the Integral controller is

$$\frac{K_i}{s} = \frac{U(s)}{E(s)}$$

where K_i is the integration constant.



A big issue that can occur when using integral control in plants with fast dynamics, like quadcopters, is the so called Integral windup. It is a typical behaviour of the integrator that accumulates a significant error while a fast setpoint change is required or presented as reference to the feedback controller.

This leads to an unreachable value for the controller that can't take effect because the actuator will saturate showing no appreciable corrective effects on the plant dynamics.

Different solution are possible in order to not income in this problem and precisely :

1. Disabling the integral function until the process variable (PV) has entered the controllable region
2. Preventing the integral term from accumulating above or below pre-determined bounds (clamp)
3. Back-calculating the integral term to constrain the process output within feasible bounds

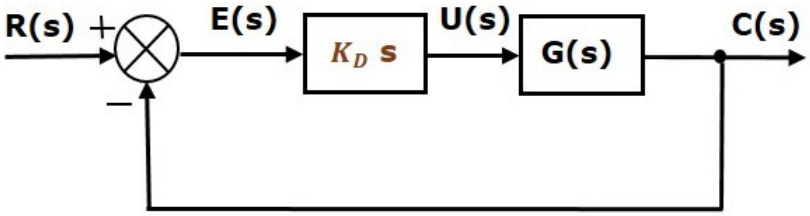
SIMULINK allows us to both put a saturator block before the controller or limit the output by setting the saturation limit inside the PID blocks but to apply the technique reported in nr. 1 it is required to implement that via MATLAB function.

In the firmware implementation we used the clamping technique. It consists in the cut-off of the integration, keeping the value constant, over or under a reference value usually symmetric with respect to the 0.

Since quadcopter dynamics are perfectly specular we decided to apply symmetrical clamping for integral values under -400 and over 400. These value have been chosen starting from the consideration that pitch, roll and yaw rest position is 1500 μs and so considering small factor correction and the fact that there is also the proportional controller acting at the same time on the system we have a bounded range for integral that covers almost all the possible PWM values and at the same time it is difficult to saturate.

22.4 D controller

The D component, or derivative component, is characterized by the multiplier K_d that influences the speed of the compensation of the error. Its behaviour is particularly highlighted in passages from fast flight to hovering and during rapid inflight-setpoint changes. In several cases it is increased or decreased proportionally to the parameter P. The setting of this parameter is recommended only after a good parameter for the multiplicative constant k_p had been found, however we are not going to use D component in the final controller for this quadcopter.



Derivative controller can be expressed as

$$u(t) = k_d \frac{\partial e(t)}{\partial t}$$

Applying the Laplace transform we obtain

$$U(s) = sk_dE(s)$$

So the transfer function of the derivative controller is

$$sk_d = \frac{U(s)}{E(s)}$$

The D part of the PID controller is approximately realizable; the ideal PID controller should not use D if the sampling time is small because the output of the PID controller severely fluctuates, resulting in shortening the life of actuators such as valves or motors because the sensitivity of the numerical derivative to noises is inversely proportional to the magnitude of the sampling time.

So, commercial PID controllers usually suppress the effects of noises by adding a strong low-pass filter to the D part or directly truncating it.

Negative effects of pure derivative controller are:

1. D-controller is very sensitive to noise in a system. For example, assuming a GPS sensor that has 1 m resolution. In real application it means that even if you stay still the consecutive position readings can differ by ± 1 m, this is a noise or disturb in a system. Now, assuming a movement of 10 km/h and a position reading at 1Hz, it means that 1s is the dt time constant, the distance to the target changes about 2.8 m each second. As can be observed, the noise amounts to more than 35% of the de(t) part of the equation. If, under these conditions, D-controller is used to stabilize the system it might (probably) happen that this instead will cause instability.

2. Derivative term is also affected by setpoint change, something that is roughly called "derivative kick". If we take the just told GPS example and we change the destination by 100 m then the consecutive calculated distance (error) will change in the middle, e.g. 50, 47, 45, 142, 139, 137 etc. Now, the de(t) will look like this: -3, -2, 97, -3, -2. Note that huge jump in derivative, not only exceeding many times the magnitude of position change, but also having opposite sign.

Concluding D-controller or derivative term can be used only under some restrictive conditions like a fast sampling time system or a system which is not heavily affected by noise.

Quadcopter are plants very subject to noise and disturbances that come from the motors EMF (Electro-Magnetic Field), from the air moved by the propeller characterized both by the flying frequencies and external factors and from the not ideal coupling of components theoretically identical each other. Therefore a pure derivative controller is for the large majority of the cases forbidden.

22.5 PI controller

This project, having an object able to move independently among the three axes X, Y and Z needs 3 controllers like the one explained above, capable of handling and correcting the error for each component of the three-dimensional space at each loop.

For each PID loop an error is calculated as the difference of the sampled value from the gyro, the one calculated in the function at the very end of the previous chapter, and the setpoint given by the mapping of the radio signal in deg / sec (the same measurement unit of the gyro).

After the calculation of the 3 components the output of the single PI controller is given by

$$u(t) = k_p e(t) + k_i \int_0^t e(t)dt \quad \Rightarrow \quad U(s) = k_p + \frac{k_i}{s} \quad \Rightarrow \quad U(z) = k_p + \frac{k_i z}{z-1}$$

The code implementation for one axis is the following

```
pid_p = Kp * error;
pid_i = pid_i + ki * error;
pid_d = kd * (error - previous_error); //we don't use this so kd = 0

total_pid = pid_p + pid_i + pid_d;

previous_error = error;
```

This calculation must be done in the same way for the 3 controllers of the three axes and must be normalized with respect to the smallest value or the largest value that the PWM pulse can assume (in this project the PWM values are all the integer values between 1100 and 2000).

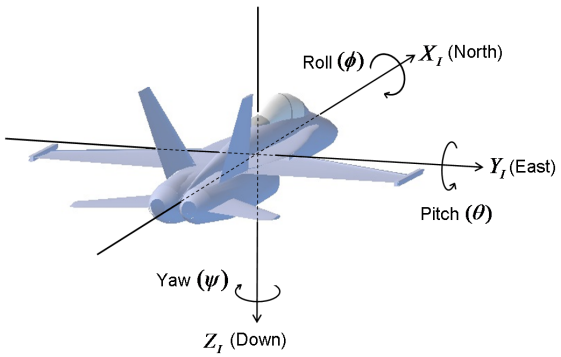
Before calculating the final pulse for each motor we need to save the throttle value in a variable; in such way we do not have to disable interrupts. Than we have to map that value within an interval $[1000, x]$, where x must be chosen to allow correction of the PID controller when the throttle (channel 3) is in its highest position.

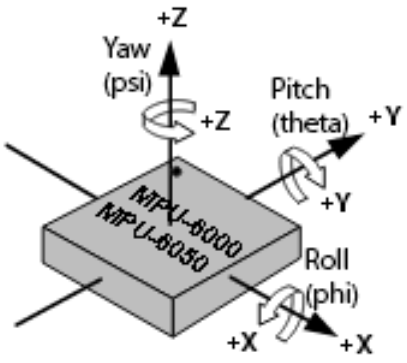
Finally, the pulse for each motor is given by the sum of the normalized throttle value with the appropriate combination of the values of the three previously calculated PID controllers as reported below

```
speedVec[MOTOR_1] = throttle - total_pid_y - total_pid_x - total_pid_z;
speedVec[MOTOR_2] = throttle - total_pid_y + total_pid_x + total_pid_z;
speedVec[MOTOR_3] = throttle + total_pid_y - total_pid_x + total_pid_z;
speedVec[MOTOR_4] = throttle + total_pid_y + total_pid_x - total_pid_z;
```

Note that the sum of the four normalized values with respect to the same interval is not a normalized value again with respect to the same interval, therefore after calculating the values of the motorPulse array they must be again normalized before passing the array to the real motors driving function.

The last computation performed by the calculate_pid() function, which is very important and cannot be absolutely forgotten, is to save the three values of the errors calculated at the beginning of the function as previous_error in order to be used in the next PID loop for the calculation of the derivative component.





One more important parameter that must be set is the orientation of each rotation verse around each axis.

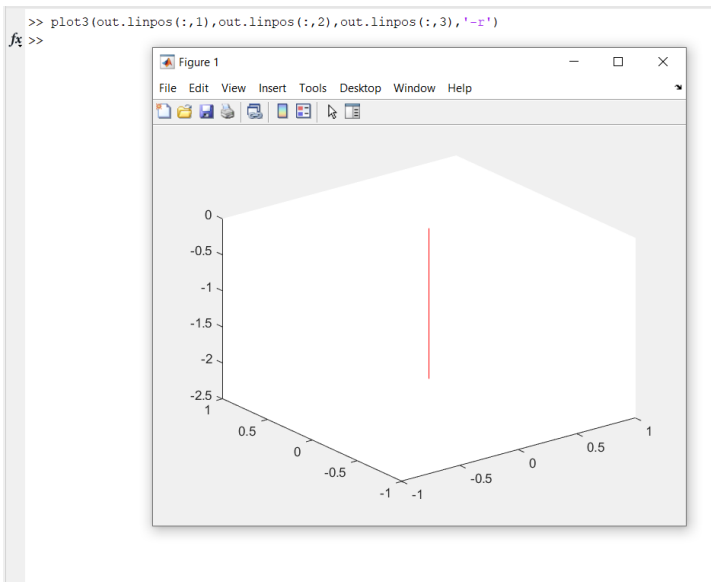
For the project we used the reference frame reported in the picture above and so since MPU_6050 Z axis positive motion is counterclockwise we need to reverse it by software multiplying the value by -1.

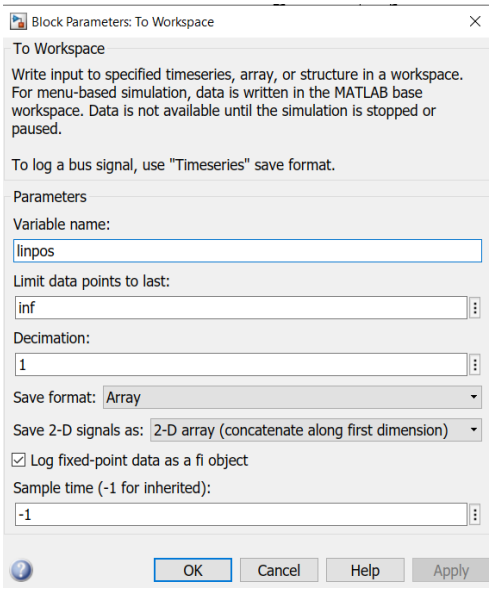
23 Simulation output in 3D MATLAB plot

Using the variable linpos we can save in a 2D array the time and the corresponding position of the model.

Then we can use these data to plot from MATLAB in a 3D graph the trajectory in which the drone has moved since the beginning to the end of the simulation.

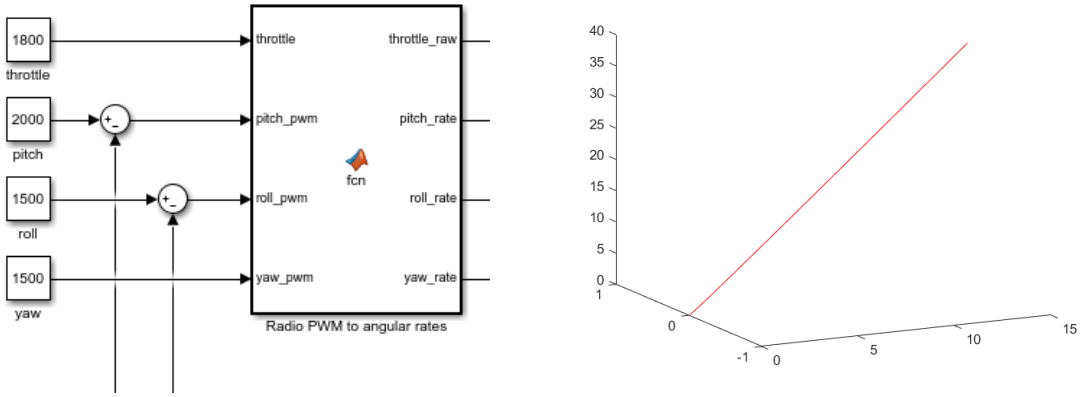
The SIMULINK block "to_workspace" is settled as follow and then we also reported the MATLAB command to plot the linear dynamics block output in 3D which corresponds to the spatial position of the quadcopter.





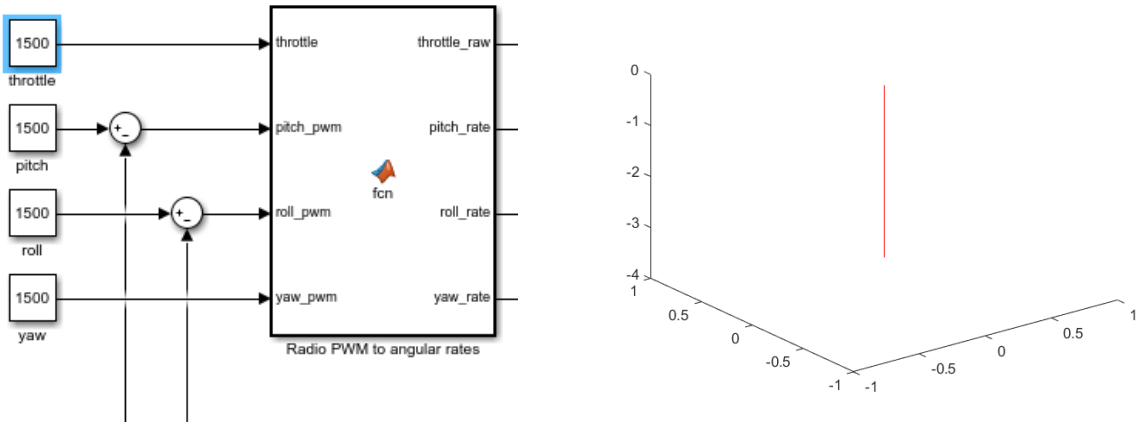
Setting the "to_workspace" block as in the right we are telling MATLAB to store simulation data as 2D double precision array. This file is similar to an .xls and can be parsed both by MATLAB itself or other tools or programming languages. For example we can plot simulation results using python matplotlib and sklearn modules.

As first test we tried to move forward with 3/4 throttle



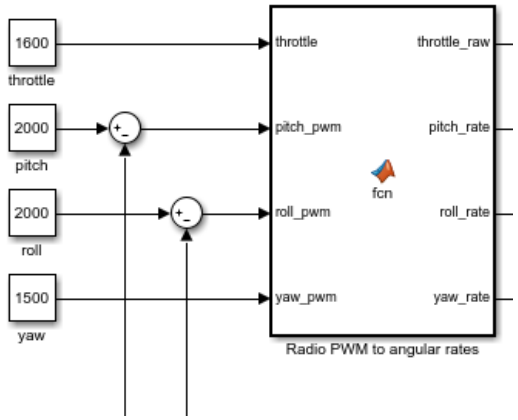
It can be observed that correctly the quadcopter moved forward at the beginning pitch until $-10°$ and the moved forward and upwards stuck at that angle which is exactly the desired setpoint.

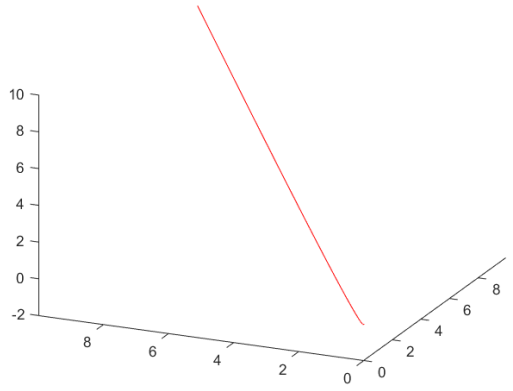
Now we can evaluate the thrust performance of the motors by applying only throttle command and see how the model behaves in the space over time.



We can notice that for half throttle the drone is going downside, which means that in real world it will not take-off, and so we can conclude that it will be better if we use more powerful motors in order to at least hover at mid throttle position.

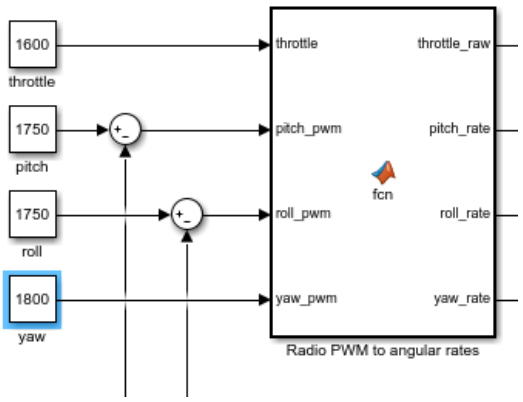
Next test is about maximum angle forward and right (it is a combined test between pitch and roll) and we expect to see the quadcopter moving at 45° with respect to the X and Y axes.

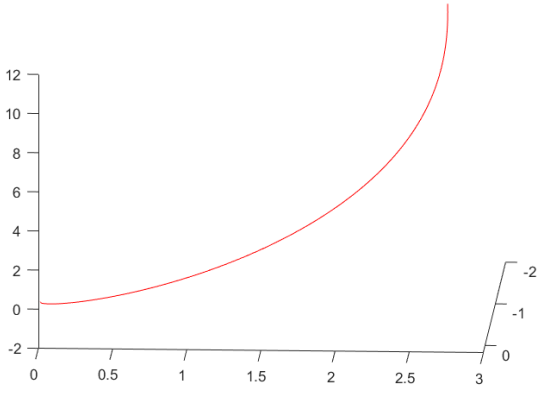




The obtained result is correct; of course the drone is also climbing because the bounding angle of ±10° for both pitch and roll is not extreme at all, so it always will produce a strong vertical component of the thrust vector.

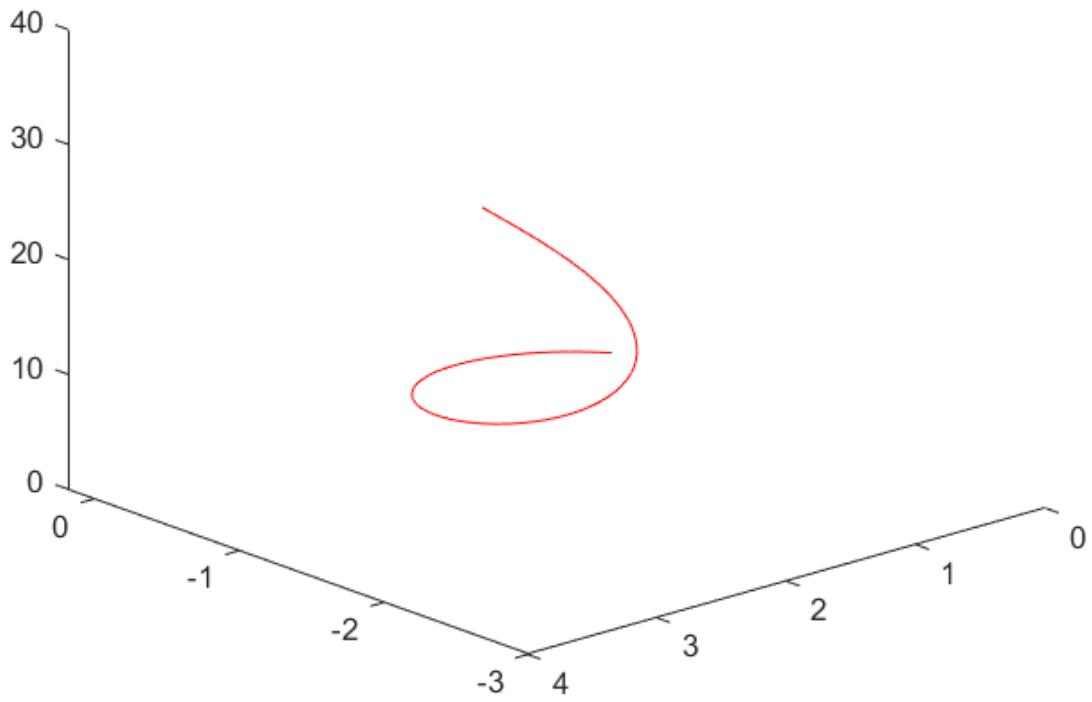
The last test is about the yaw. Since it is the proper rotation of the quadcopter around its body frame Z axis to obtain observable results on the 3D trajectory we have to model a complex maneuver of the drone that involves all 3 pitch, roll and yaw. The expected result would be part of a circle turn while climbing.





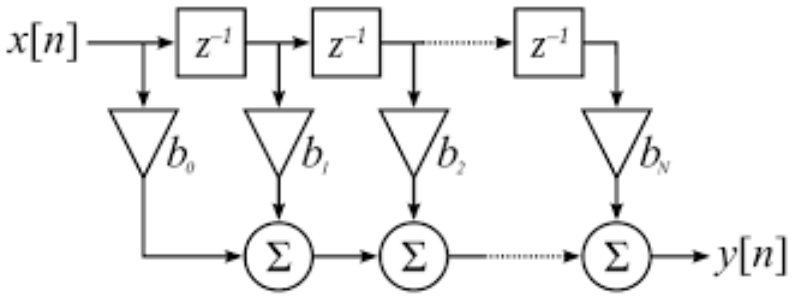
The result is correct, the full turn is interrupted only because of the simulation time too short to see the full round trajectory.

However if we take a longer simulation time we can see the output of the full climbing turn as reported below



24 Filters

24.1 FIR filters



In signal processing, a Finite Impulse Response (FIR) filter is a filter whose impulse response, or its response to any finite length input, is of finite duration, because it settles to zero in finite time. This is in contrast to Infinite Impulse Response (IIR) filters, which may have internal feedback and may continue to respond indefinitely (usually decaying).

FIR filters can be discrete-time or continuous-time both digital or analog; however we are going to use only the discrete-time digital one as DLPF (Digital Low-Pass Filter) for gyroscope data and derivative component of the PID controller.

Generally digital filters implementations are made exploiting specific processors called DSP (Digital Signal Processor) that are strongly optimized for this specific purpose and rarely are hardware realized (hardware implementation is mandatory if a continuous time filter is required).

As reported in the datasheet, our MCU, the STM32F303K8T6 has integrated on its ARM instruction set specific DSP instructions that allows us to efficiently manage the code to get high performance digital filters directly inside the firmware.

ARM instruction for DSP are the same implemented in specific purpose processors and those aim to optimize the math between "multiply and accumulate" chain at assembly language level.

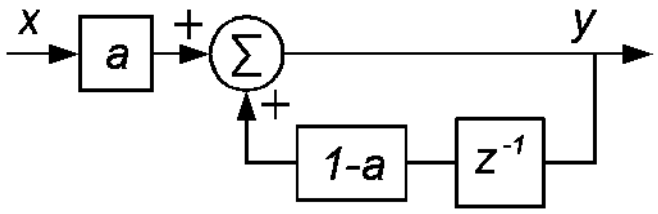
24.2 IIR filters

Infinite impulse response (IIR) is a property applying to many LTI systems. Common examples of linear time-invariant systems are most electronic and digital filters. Filters with this property are known as IIR filters, and are distinguished by having an impulse response which does not become exactly zero after a certain time, but continues indefinitely. This is in contrast to a Finite Impulse Response (FIR) in which the impulse response $h(t)$ become exactly zero at times t > T, for some finite T, being of finite duration.

In practice, the impulse response, even of IIR systems, usually approaches zero and can be neglected after a certain point in time.

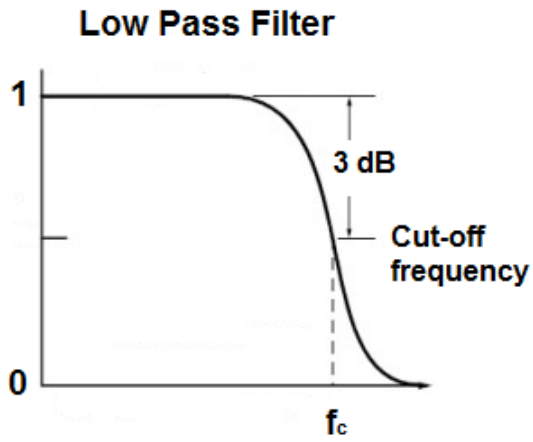
However the physical systems which give rise to IIR or FIR responses are very different. For instance, analog electronic filters composed of resistors, capacitors, and/or inductors (and perhaps linear amplifiers) are generally IIR filters. On the other hand, discrete-time filters, usually digital filters, based on a tapped delay line employing no feedback are necessarily FIR filters.

Capacitors and inductors in the analog filter have a memory behaviour with respect to the variation of the signal in time (signal time derivative) and their internal state never completely relaxes following an impulse.



The block representation shown above is an anticipation of the block diagram of the IIR single pole low pass filter used in the flight controller firmware we are going to deeply analyze in few pages.

24.3 DLPF filters



A low-pass filter (LPF) is a filter that passes signals with a frequency lower than a selected cut-off frequency and attenuates signals with frequencies higher than the cutoff frequency. The exact frequency response of the filter depends on the filter design.

Generally in RT application the use of digital filters is preferred with respect to the discrete components one because its possibility to be differently tuned and because its versatility that makes possible to change filter characteristics at cost of minimal changes in the software and in the system in general.

Kind of DLPF are the moving average, which is commonly used with time series data to smooth out short-term fluctuations and highlight longer-term trends or cycles and the n-pole low pass filter.

It is important to take care when using filters in control because every time a signal passes through a digital filtering system it takes a delay that directly depends on the filter implementation and depending on the case could be not negligible and harmful for the control system dynamics.

In the project we are going to use the built-in DLPF of MPU-6050 and we implemented a single-pole filter to better smooth gyro data.

24.4 Digital filter and causality (RT application)

It is important for Real-Time application that Digital Filters involved in signal processing and the dynamics of the controllers to be causal. We define a system causal if its output for a given time only depends on its past values.

Formally we can say that a system mapping x to y is causal if and only if, for any pair of input signals $x_1(t)$ and $x_2(t)$ and a selected t_0 we have

$$x_1(t) = x_2(t), \quad \forall t < t_0$$

Because of this definition we previously said that a true Derivative controller is in real application not feasible, because it is not causal however rearranging formulas it is possible to implement a differentiator.

Every kind of filter or control system that has to run in a RT application must be causal, this because real time requires data to be processed in "real-time mode" so every output of the system has no information about the future and it has no way to know that. Every calculation must be done only on present data and previous system outputs.

In digital filter sum the data to filter is always a low-pass filter, this because in discrete domain sum operation is integration and therefore integrator is a low-pass filter. Opposite to implement a high-pass filter we just need to subtract the sampled data.

A digital filter can be centered or not centered

$$\begin{aligned} centered \;\; &\Rightarrow \;\; y(n) = y(n-1) + y(n) + y(n+1) \\ Not\, centered \;\; &\Rightarrow \;\; y(n) = y(n) + y(n-1) \end{aligned}$$

24.5 Firmware gyro low pass filter

The following line are present in the firmware and implements a DLPF for gyroscope value.

Here we have the protection from unreachable data that are all those that are outside the range of the gyroscope scale value.

```
if ( gyro_raw_x >= MIN_GYRO_VALUE && gyro_raw_x <= MAX_GYRO_VALUE ) {

        gyro_pitch_input = ( gyro_pitch_input * 0.8 ) + ( gyro_raw_x * 0.2 );

}
if ( gyro_raw_y >= MIN_GYRO_VALUE && gyro_raw_y <= MAX_GYRO_VALUE ) {

        gyro_roll_input  = ( gyro_roll_input  * 0.8 ) + ( gyro_raw_y * 0.2 );

}
if ( gyro_raw_z >= MIN_GYRO_VALUE && gyro_raw_z <= MAX_GYRO_VALUE ) {

        gyro_yaw_input   = ( gyro_yaw_input   * 0.8 ) + ( gyro_raw_z * 0.2 );

}
```

The outputs of these three filters is the value of the gyroscope presented in the PI controller as feedback as

$$error = radio\_xxxx\_desired\_setpoin - gyro\_xxxx\_input$$

Such filter takes strongly into account the past value and relies on the new value only for a little portion of signal.

However the α gain can be changed to reach different application specifications.

In particular the filter just described and used in the code is a IIR (Infinite Impulse Response) Single pole Low-Pass filter.

This filter has a single design parameter, which is the decay value α.

The recurrence relation is

$$y[n] = \alpha x[n] + (1 - \alpha)y[n - 1] \qquad \alpha = \frac{T_s}{RC + T_s}$$

where $x[n]$ is the input of the filter, $y[n - 1]$ is the previous output and T_s is the sampling period or equivalently the time between two consecutive samples.

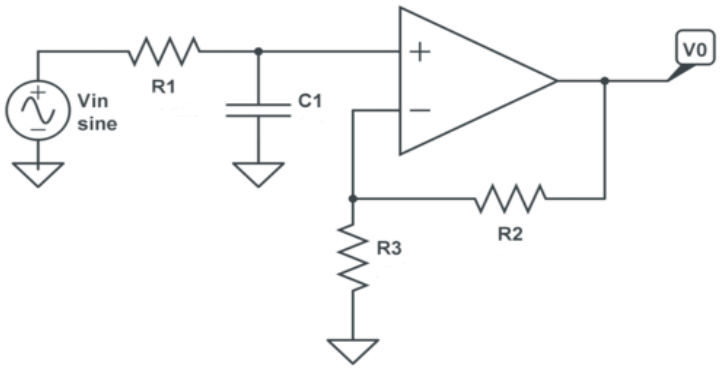
By definition the smoothing factor α must be $0 < \alpha \leq 1$. Therefore from the expression of α we can calculate the RC constant

$$RC = T_s \left(\frac{1 - \alpha}{\alpha} \right)$$

Notice that for $\alpha = 0.5$ the RC constant is exactly the same as the sampling period, whatever the period is.

The recurrence relation directly shows the effect of the filter. The previous output value of the filter, y[n - 1], is decreased with the decay factor, also named smoothing factor $1 - \alpha$. The current input value, x[n], is taken into account by adding a small fraction α of it to the output value.

$$y[n] = \alpha x[n] + (1 - \alpha)y[n - 1]$$



The impulse response for the single-pole IIR filter is defined on a sample by sample basis, as described by the recurrence relation given above.

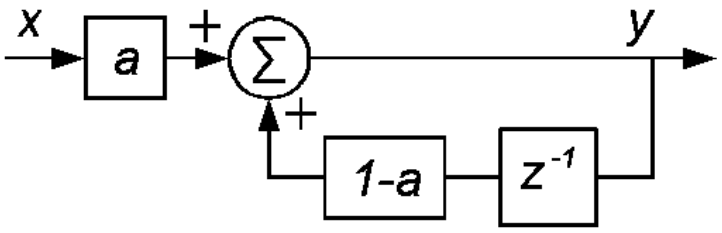
The response of this filter is completely analogous to the response of an electronic low-pass filter consisting of a single resistor and a single capacitor or to be more specific to an active electronic low pass filter (like first order Sallen-Key reported in the previous page) because we don't have any reduction of the signal being that a full digital process.

Recalling that we know from electronics that the cut-off frequency of first order filter is given by

$$f_c = \frac{1}{2\pi RC}$$

It means that for discrete time systems, as the firmware in MCUs, the knowledge of the sampling frequency (or sampling period) and the smoothing factor α make possible to know the cut-off frequency of the filter.

The filter just described is also known as Exponentially Weighted Moving Average Filter or EWMA for short and its SIMULINK DSP block (the same reported in IIR filters section) is the following



25 Complementary filter

Most IMU's have 6-DOF (Degrees Of Freedom), so there are 3 accelerometers, and 3 gyroscopes inside the unit. Remembering class it is normal to think that the IMU will be able to measure the precise position and orientation of the object it is attached to. This because theoretically an object in free space has 6DOF. So if we can measure all 6-DOF from the theory perspective we can do everything. In real world application it is not possible to act straight-forward this way. Sensors data are not good enough to be used in such way.

We used both the accelerometers and gyroscopes data for the same purpose: obtaining the angular position of the object.

The gyroscope can do this by integrating the angular velocity over time. To obtain the angular position with the accelerometer, we are going to determine the position of the total gravity vector (g-force) which is always visible on the accelerometer. This can easily be done by using an atan2 function. In both these cases, there is a big practical issue which makes the data very hard to be uses without filters.

25.1 Accelerometers problem

As an accelerometer measures all forces that are working on the object, it will also see a lot more than just the gravity vector. Every small force working on the object will disturb our measurement completely. If we are working on an actuated system (like the quadcopter), then the forces that drive the system will be visible on the sensor as well. The accelerometer data is reliable only on the long term, so a low pass filter has to be used.

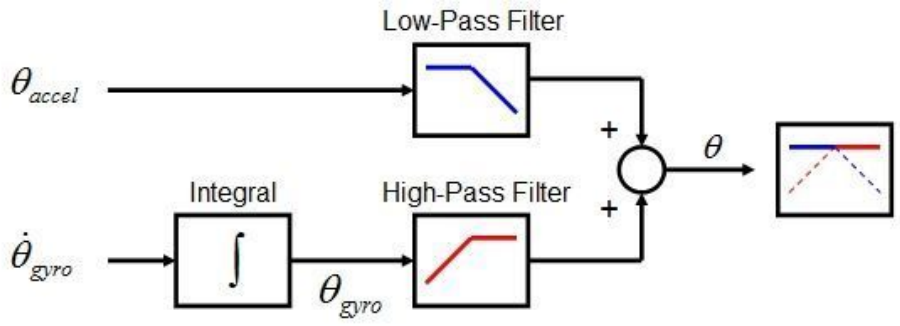
25.2 Gyroscopes problem

It was very easy to obtain an accurate measurement that was not susceptible to external forces. Bad using gyroscopes is that, because of the integration over time, the measurement has the tendency to drift, which means not returning to zero when the system went back to its original position. The gyroscope data is reliable only on the short term, as it starts to drift on the long term.

25.3 Sensor fusion

The complementary filter gives us the best of both gyroscopes and accelerometers data. On the short term, we use the data from the gyroscope, because it is very precise and not susceptible to external forces. On the long term, we use the data from the accelerometer, as it does not drift.

The filter looks as follows



$$angle = \alpha(angle + gyro\_angle * dt) + (1 - \alpha)(accel\_angle)$$

The gyroscope data is integrated every timestep with the current angle value. After this it is combined with the low-pass data from the accelerometer (already processed with atan2). The constant α can be changed to properly tune the filter.

Within every iteration the pitch and roll angle values are updated with the new

gyroscope values by means of integration over time. The filter then checks if the magnitude of the force seen by the accelerometer has a reasonable value that could be the real g-force vector. If the value is too small or too big, we know for sure that it is a disturbance we don't need to take into account. Afterwards, it will update the pitch and roll angles with the gyroscope data by taking $\alpha\%$ of the current value, and adding $(1-\alpha)\%$ of the angle calculated by the accelerometer. This will ensure that the measurement won't drift, but that it will be very accurate on the short term.

The filter is easy and light to implement making it perfect for embedded systems. It should be noted that for stabilization systems (like the quadcopter), the angle will never be very big. The atan2 function can then be approximated by using a small angle approximation. In this way, this filter could easily fit on an 8 bit system, however our MCU is 32bit and greatly support atan2 and all the other trigonometric functions taking the advantage of its built-in FPU (Floating Point Unit).

Filtered data appear as follow compared to the raw ones

26 Control system output

If the quadcopter is in ACRO mode, then the outer feedback loop has all gains set to 0, so $k_p = 0$ and the control loop is only handled by the inner loop, the one of the angular rates.

If the quadcopter is in STABILIZED mode both the feedback loops are active and the position of the vehicle is limited by the bounding angles settled by the proper choice of the k_p constant of the outer loop.

The choice of such K_p is driven by the simple formula reported below

$$K_p = \frac{500}{B_a}$$

where

1. 500 is the maximum PWM offset from the neutral position
2. B_a is the desired bounding angle

In the firmware by default we provide 4 different pre-computed bounding angles that are:

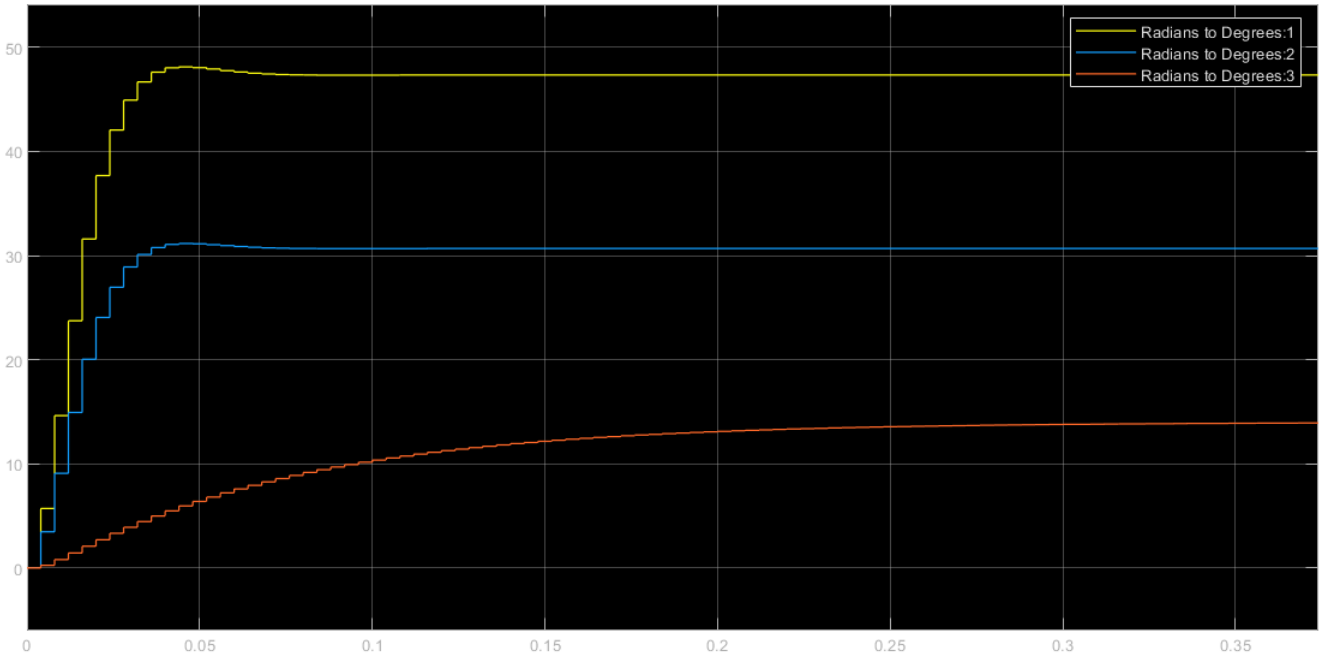
```
#define ANGLE_CORRECTION_10  50       // [-10;+10]
#define ANGLE_CORRECTION_15  33.333   // [-15;+15]
#define ANGLE_CORRECTION_35  14.28    // [-35;+35]
#define ANGLE_CORRECTION_45  11.1111  // [-45;+45]
```

One of this four values, by default is ANGLE_CORRECTION_10, is assigned in the setup procedure to the float variable angle_correction but however every angle under $[-60°; +60°]$ can be settled using the previously shown formula.

26.1 Controller outputs and flying results comments

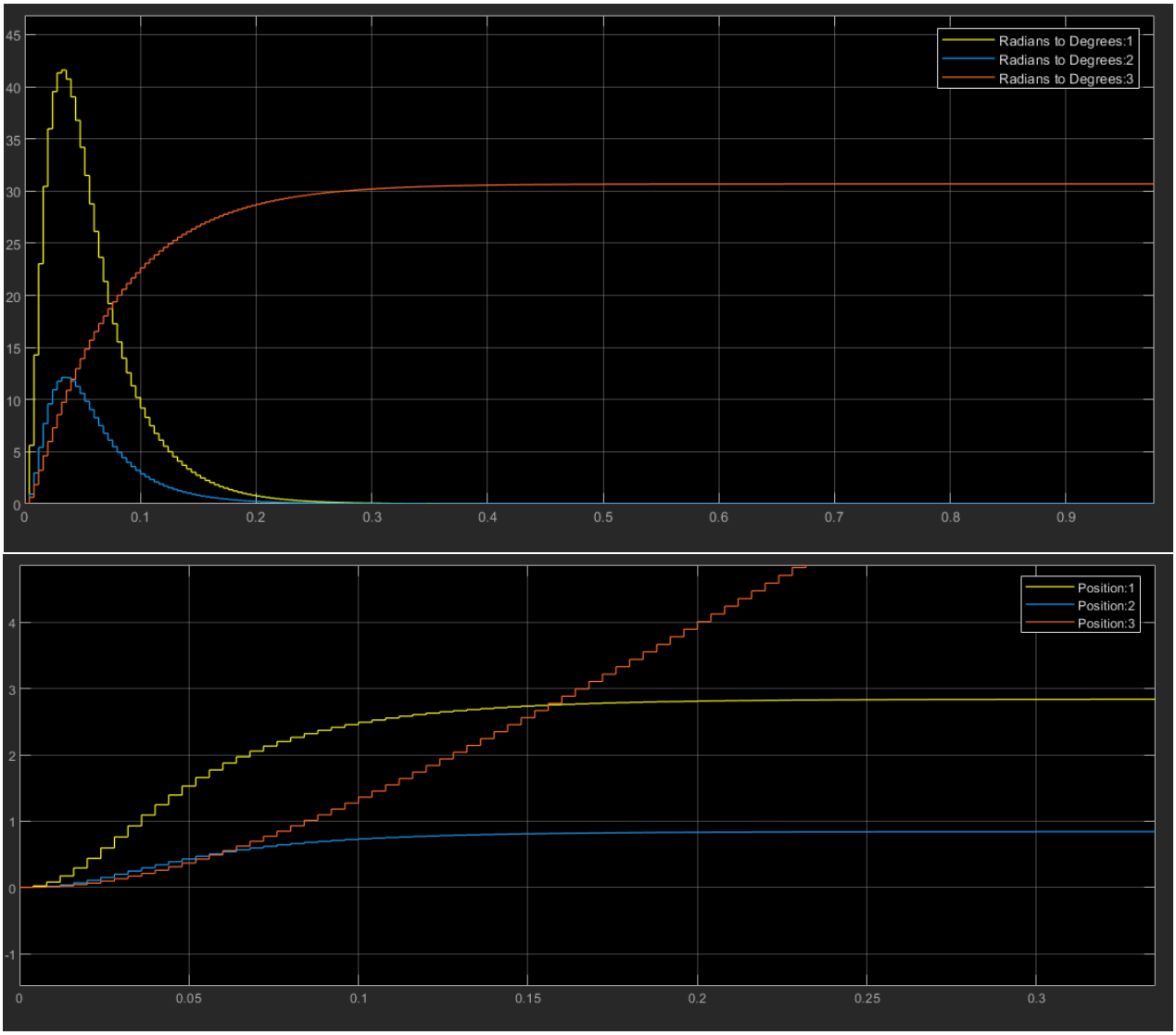
With the proper values settled we can observe using the simulation on the model described at page 54 the control output for the two modes.

As expected in acro mode the angular speed (rate) keeps increasing until the reference is reached, so this directly depends on the mapping of radio on rates.



It can be seen that there a little overshoot with respect to the desired position, however it is too small to significantly affect the controller adjustment.

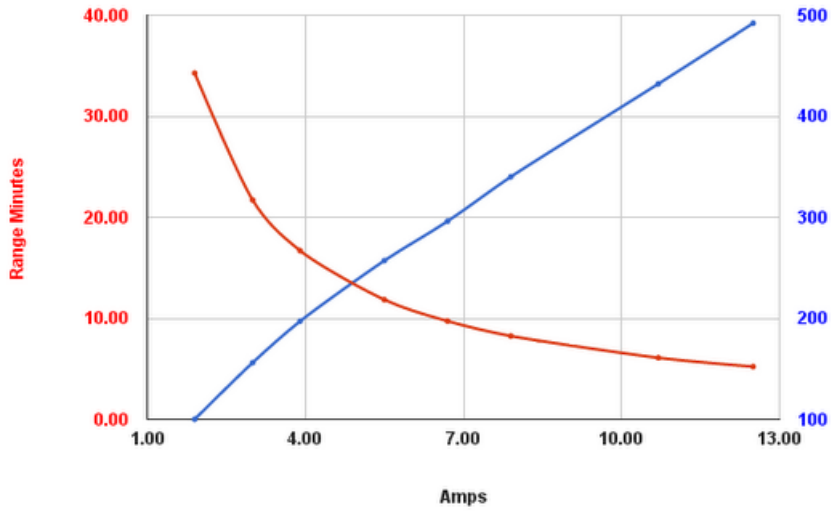
In Stabilized mode the control output is both given as before from the radio channel mapping and the bounding angle chosen from the configurator. The inner control loop, the one acting on the angular rates, is responsible for the speed on which the quadcopter moves to reach the setpoint. The outer loop, the slower one, act to limit the behaviour of the other loop by applying an implicit scale to the input given by the SAPR operator.



Analyzing the red line it is possible to see that it is very different from the other two; this because using 6-DOF controlling sensing units is not efficiently possible to control the vehicle heading. For such task sensors such compass, GPS or magnetometer are required. The yaw axis in both Acro and Stabilized mode relies only on the angular rate control while for pitch and roll it is possible to filter data to retrieve the angle in that specific axis in a specific moment.

If controllers gains are not properly chosen we can observe strange behaviours in the controller output that can lead to oscillating scenarios in which instability can lead to diverging signals with respect to the output or to converge but in a not acceptable time or with too much overshoot to guarantee a good usability for the vehicle.

It is important to notice that incorrect controller output signal not only affect the quality of the vehicle control itself but can seriously damage motors or make current consumption peaks that can significantly reduce autonomy or stress the battery with direct testable consequences.

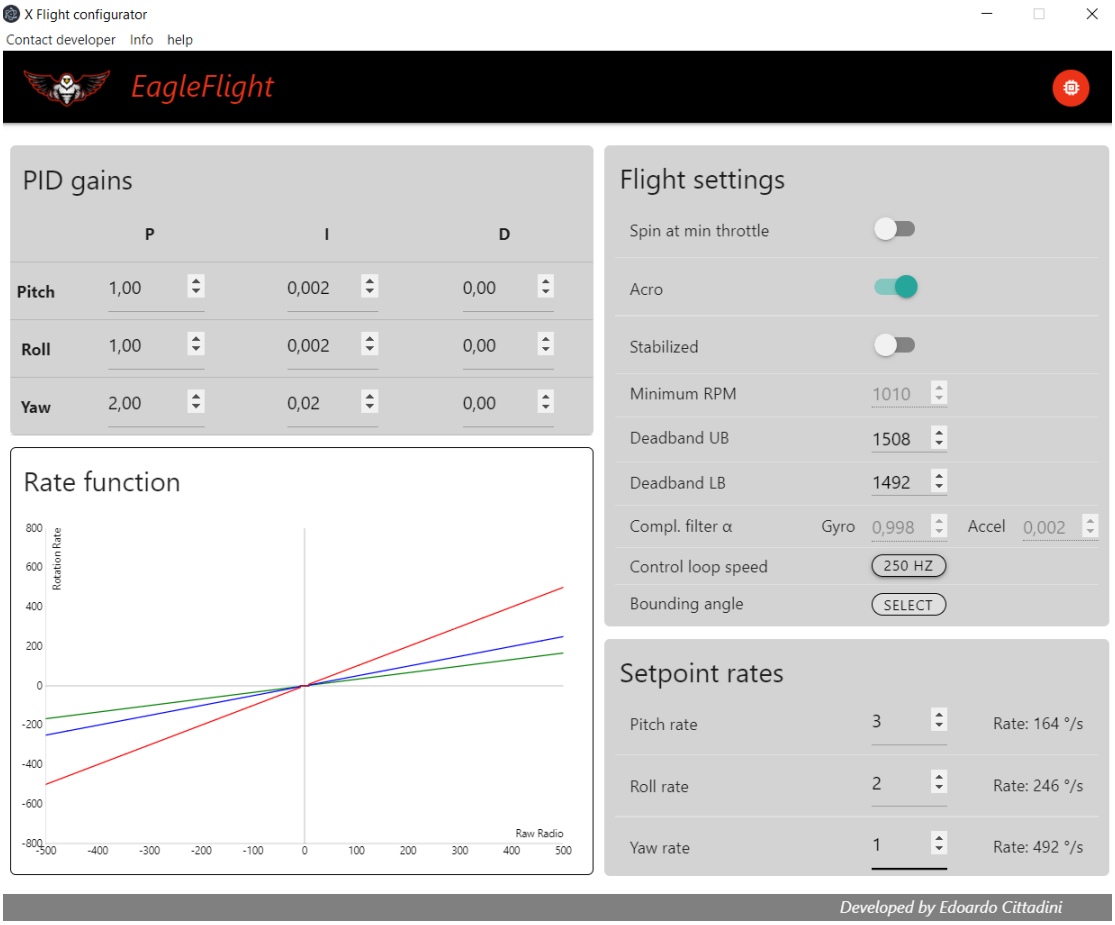


This graph better explains what just said, motor thrust is represented with the blue line while Autonomy against Current is shown with the red line.

It is easy to see that high current consumption lead to a much faster decay of the Voltage and consequently reduces a lot the flight time of the drone. In quadcopter as in other multirotor/multirotor vehicles is a very important factor to deal with.

27 Configuration program

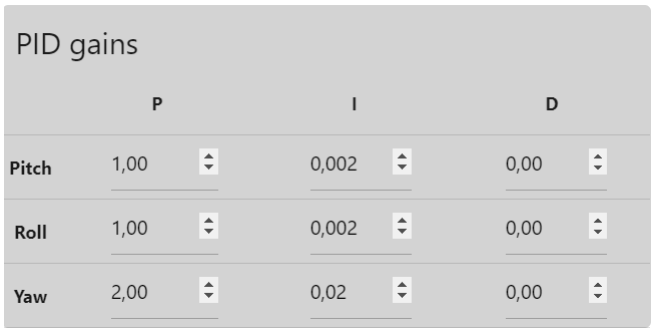
To make the controller parameters easy to access without having to operate directly at SW level, a flight configurator has been developed along the lines of those proposed by the most famous softwares such as Betaflight, iNAV, CleanFlight and RaceFlight.



The configurator is divided into 3 main control areas:

1. PID gains tuning
2. Controller internal configuration
3. Radio input normalization and rates

27.1 PID gains tuning



In this section is possible to set the proper values of the gains K for the three possible rotation the quadcopter is able to perform.

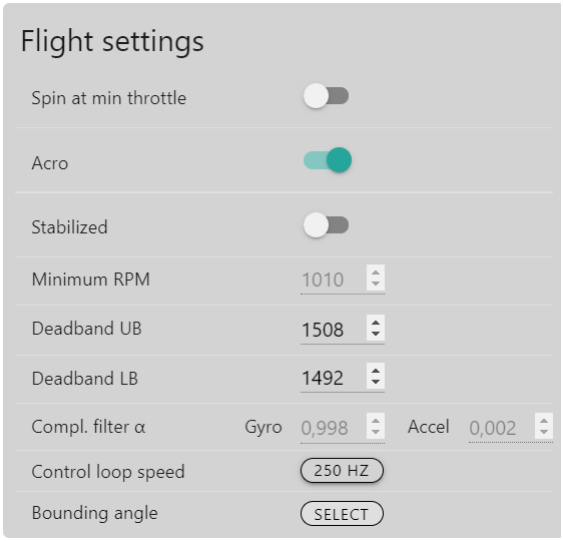
Many flight controllers, considering the intrinsic symmetry of the vehicle, constrain pitch and roll to the same multipliers; however the aim of the software of this project is to be extensible and as compatible as possible with different flight platforms therefore it has been chosen to keep independent all the values related to the degrees of freedom (DOF) controlled (in this specific case the 3 rotations).

For example, the aforementioned symmetry is lost in more unstable configurations such as the trichopter and it is precisely in these circumstances that the independence of the controllers on the axes becomes fundamental.

Furthermore, again due to the diversity of the vehicles for which the software is designed, whether they are currently configurable or if an interface for future support has been provided, the parameters P, I, D are in no way bounded by any part of the application and for this reason the insertion of coherent and meaningful values is entirely in charge of the user of the platform.

For dangerous situations, as already described in the part concerning the controls given via radio control, safety checks are provided in the flight firmware (and therefore not at the application level) that do not allow the use of the quadcopter in situations that could cause damage to the pilot or people nearby.

27.2 Controller internal configuration



In the second control panel it is possible to set the internal operating modes of the controller. The flight modes that can be implemented with gyroscope and accelerometer are 2: Acro and Stabilized.

Not all the parameters presented in this interface are useful or usable in both flight modes, in fact a distinction must be made. In Acro mode it has been explained that only the gyroscope is used and therefore the configurable values apart from the dead band for the rest position of the sticks are:

1. Rotation of the motors at min. throttle
2. Control cycle frequency

In case of "spin at min throttle" the RPM input block becomes active and editable in order to make possible for the user to set the desired minimum (idle) rotation per minute. This is a bounded free choice for security reasons in fact values lower than 1000 (us) are discarded because meaningless for the motor PWM and values over 1100 (us) are discarded because of the danger of such high RPM in an idle state without possibility of control.

Stabilized mode is more complex both because the cooperation between sensors and the filtering system involved in the calculations.

In this operating mode everything just said is valid but it is possible and mandatory to tune much more parameters that are better explained below.

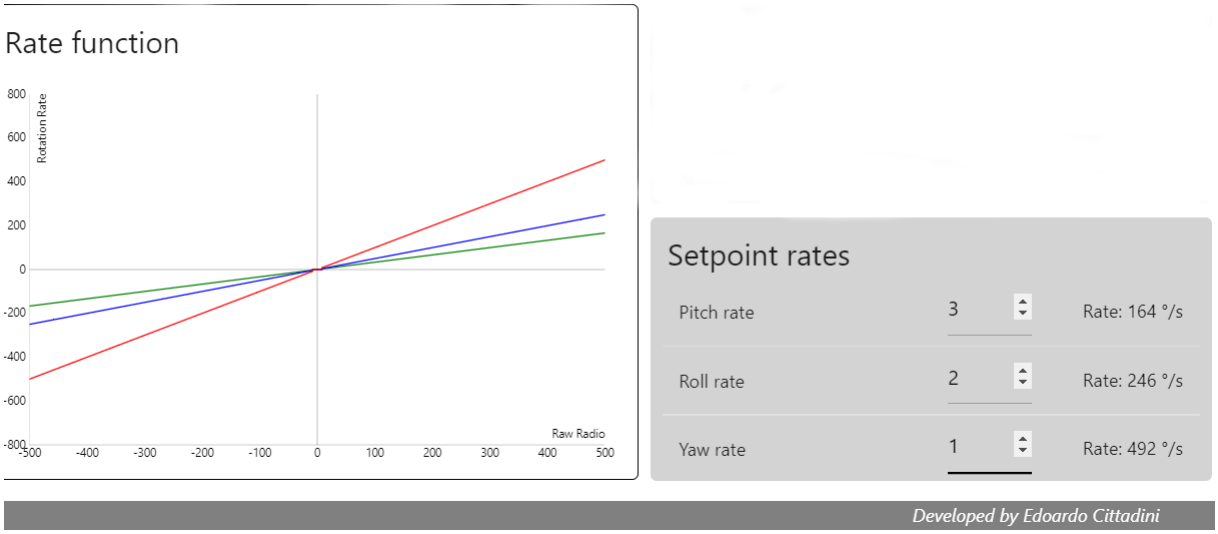
The first big difference is the $\alpha - value$ or gain for the complementary filter better explained in the previous chapter. The application automatically handles the "sum 1 behaviour" of the two multipliers, so the user can both tune the accelerometer part or the gyro one that the configurator transparently to the user act in order to maintain the coherence among parameters.

For sure extreme output are allowed (1 gyro and so 0 accel and the opposite) but in such configuration the filter is no more useful but instead it is harmful for the system.

In stabilized mode not only the automatic stabilization at 0 degrees level is enabled for pitch and roll but also an angle bound is provided. Such angle is settled by the proper choice of the gains for the outer control loop.

In this configurator, as described in page 108, 4 different values are provided. As the angle increases, the speed of movement and the speed of the vehicle also increase; however, the quadcopter is also more difficult to be controlled.

27.3 Radio input normalization and rates

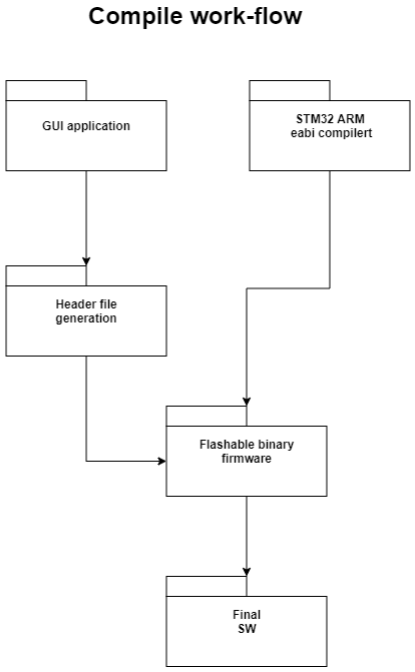


In the third panel you can set the values that define how the inputs provided by the radio have an impact on the operations of the multirotor. It is possible to define the input and conversion relationship of radio channels in many different ways; for example Betaflight uses a third degree function with variable parameters.

In the case of this software, on the other hand, a pieces linear relationship is used and to make the output modes clearer and with parameters that are immediately visually understandable, a graph updated in real-time was reported as the configurator parameters change. This allows you to see for a determined input the corresponding angular velocity on which it is mapped. The function is piecewise because the linear behaviour stops in the domain of the dead-band. In the dead-band area the mapped angular rate corresponds always to 0.

Since the values for the rates, those reported in the picture above on the right, are divisors in math functions the application automatically handles bounds for them. In particular value 0 is not allowed to avoid "division by zero" error and the maximum value can be settled was decided to be 0.2 that corresponds to an angular rate of 2460 °/s, enough for every kind of application.

27.4 STM32 C Header generation



When all desired parameters are configured the application transparently to the user is able to generate a C header file from which the Eclipse C compiler is able to retrieve data information to properly tune software and controllers values.

Every chosen value can be both generated for the first time or inferred from another static .h file called "common.h".

After the header generation, configurator puts a copy of such file in the "Inc" directory of the source project allowing the arm-none-eabi compiling toolchain to find and use it at compile time for the .elf/.hex file generation.

The .hex file is an intel standard to map compiled programs into memory while .elf (Executable Linkable File) can be used directly out of the shelf.

Every time a new header is generated by the application the old one will be overwritten.

27.5 Configuration dump and restore

In order to not have to reconfigure the whole interface for every time a little change is needed a configuration dump and restore mechanism has been provided for each time a new header file is generated.

Configuration is saved in a lightweight JSON (JavaScript Object Notation) file as follow

```
function dump_configuration(){
 let data = {
        "pitch_p"             : document.getElementById('pitch_p').value,
        "pitch_i"             : document.getElementById('pitch_i').value,
        "pitch_d"             : document.getElementById('pitch_d').value,
        "roll_p"              : document.getElementById('roll_p').value,
        "roll_i"              : document.getElementById('roll_i').value,
        "roll_d"              : document.getElementById('roll_d').value,
        "yaw_p"               : document.getElementById('yaw_p').value,
        "yaw_i"               : document.getElementById('yaw_i').value,
        "yaw_d"               : document.getElementById('yaw_d').value,
        "db_ub"               : document.getElementById('db_ub').value,
        "db_lb"               : document.getElementById('db_lb').value,
        "stabilized_toggle"   : document.getElementById('stabilized_toggle').checked,
        "acro_toggle"         : document.getElementById('acro_toggle').checked,
        "min_spin_rpm"        : document.getElementById('min_spin_rpm').checked,
        "min_rpm"             : document.getElementById('min_rpm').value,
        "gyro_a"              : document.getElementById('gyro_a').value,
        "sel_dr2"             : document.getElementById('sel_dr2').innerHTML,
        "sel_dr1"             : document.getElementById('sel_dr1').innerHTML,
        "pitch_rate"          : parseFloat(document.getElementById('pitch_rate').value).toFixed(1),
        "roll_rate"           : parseFloat(document.getElementById('roll_rate').value).toFixed(1),
        "yaw_rate"            : parseFloat(document.getElementById('yaw_rate').value).toFixed(1)
    };

  data = JSON.stringify(data);
  fs.writeFileSync('dump/config.json', data);
}
```

The last two lines show how values are saved in the traditional JSON notation (k ey : value) persistently in the disk.

Now it is reported the reverse procedure to load such saved values in the GUI at the next configuration use

```
function load_configuration(){
  let data = fs.readFileSync('dump/config.json');
  data = JSON.parse(data);

  for(var key in data){
    if(key == "stabilized_toggle" || key == "acro_toggle" || key == "min_spin_rpm"){
      document.getElementById(key).checked = data[key];
    }else if(document.getElementById(key).tagName == "INPUT"){
      document.getElementById(key).value = data[key];
    }else{
      document.getElementById(key).innerHTML = data[key];
    }
  }

}
```

Data are parsed back basing on the type of HTML element they have to match and the load/store procedure goes on and on as long as the program is used for every invocation and for every new header file generated.

28 Conclusion

In this report we explained how to develop a complete ARM-based flight controller.

We used a bottom-up approach, starting from the hardware requirements such the MPU choice up to the optimization of the flight software.

STM32 MCUs are well equipped with all the peripherals and hardware mechanisms that are needed to develop a flight control unit.

Project had been developed in 4 main steps:

1. Draw the board using CAD, print the board and subsequently tests to confirm its correct functioning
2. Build a simple mathematical model of the quadcopter and simulate the whole control logic using MATLAB/SIMULINK
3. Develop the logic that is not related to the control loop and then develop the filtering system both for gyroscope and angle estimation data
4. Implement the control loop on a microcontroller, implement the input capture mechanism, setup hardware PWM for actuation, eventually fine-tune filters coefficients and finally perform an in flight test.

This project was developed for the Digital Control Systems and Mechatronics of the master degree in Embedded Computing Systems (ECS) from University of Pisa and Scuola Superiore Sant'Anna.

The author Edoardo Cittadini would like to thank Prof. Carlo Alberto Avizzano who supervised the hardware design and Prof. Lorenzo Pollini who supervised the control, filtering and simulation of the model.

29 System report

1. Description

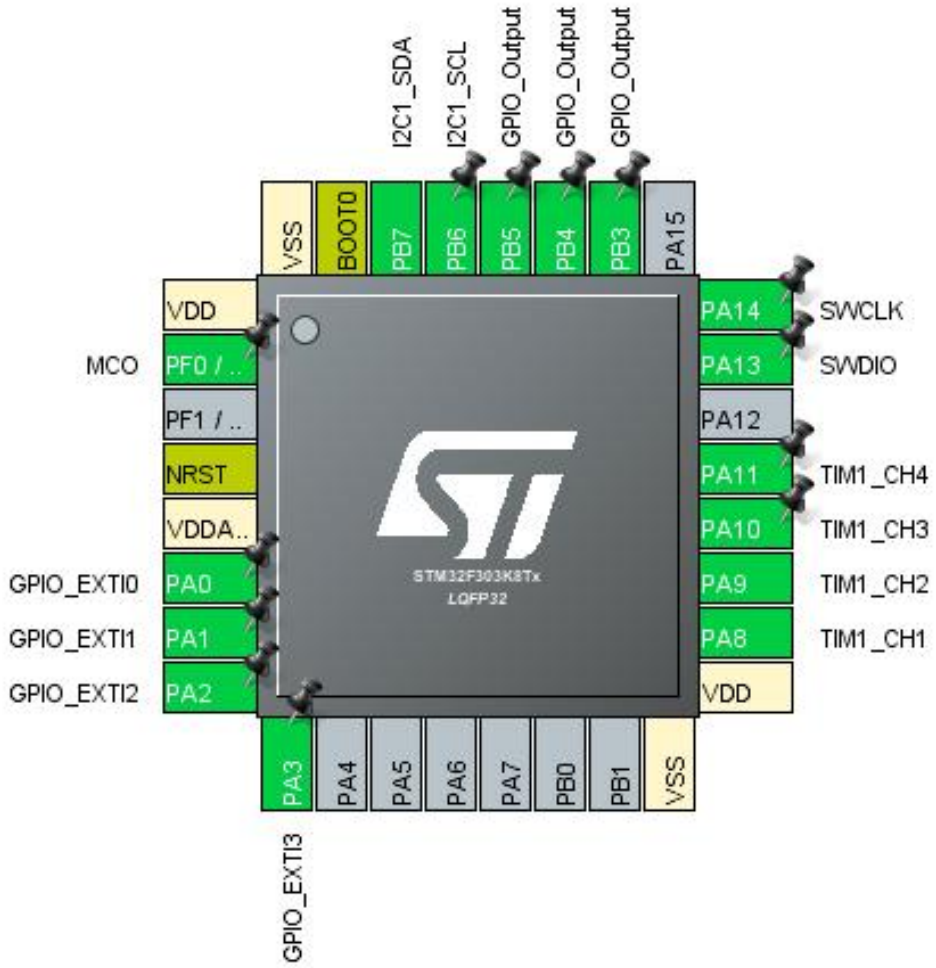
1.1. Project

Project Name	STM32F303K8T6 Drone
Board Name	NUCLEO-F303K8
Generated with:	STM32CubeMX 5.2.0
Date	11/16/2019

1.2. MCU

MCU Series	STM32F3
MCU Line	STM32F303
MCU name	STM32F303K8Tx
MCU Package	LQFP32
MCU Pin number	32

2. Pinout Configuration

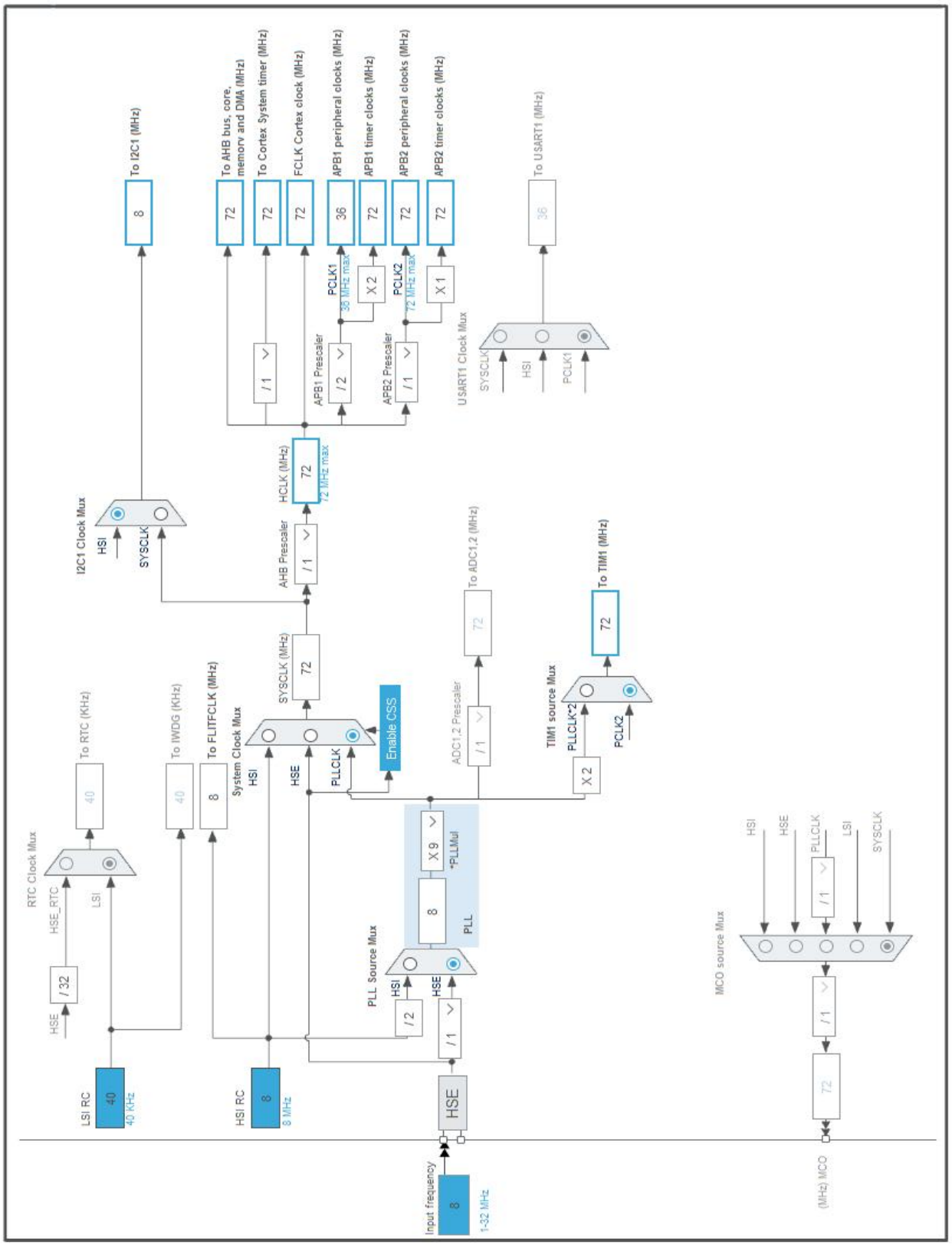


3. Pins Configuration

Pin Number LQFP32	Pin Name (function after reset)	Pin Type	Alternate Function(s)	Label
1	VDD	Power		
2	PF0 / OSC_IN	I/O	RCC_OSC_IN	MCO
4	NRST	Reset		
5	VDDA/VREF+	Power		
6	PA0	I/O	GPIO_EXTI0	
7	PA1	I/O	GPIO_EXTI1	
8	PA2	I/O	GPIO_EXTI2	
9	PA3	I/O	GPIO_EXTI3	
16	VSS	Power		
17	VDD	Power		
18	PA8	I/O	TIM1_CH1	
19	PA9	I/O	TIM1_CH2	
20	PA10	I/O	TIM1_CH3	
21	PA11	I/O	TIM1_CH4	
23	PA13	I/O	SYS_JTMS-SWDIO	SWDIO
24	PA14	I/O	SYS_JTCK-SWCLK	SWCLK
26	PB3 *	I/O	GPIO_Output	
27	PB4 *	I/O	GPIO_Output	
28	PB5 *	I/O	GPIO_Output	
29	PB6	I/O	I2C1_SCL	
30	PB7	I/O	I2C1_SDA	
31	BOOT0	Boot		
32	VSS	Power		

* The pin is affected with an I/O function

4. Clock Tree Configuration



5. Software Project

5.1. Project Settings

Name	Value
Project Name	STM32F303K8T6 Drone
Project Folder	C:\Users\edoardo\Desktop\STM32F303K8T6 Drone
Toolchain / IDE	SW4STM32
Firmware Package Name and Version	STM32Cube FW_F3 V1.10.0

5.2. Code Generation Settings

Name	Value
STM32Cube Firmware Library Package	Copy only the necessary library files
Generate peripheral initialization as a pair of '.c/.h' files	No
Backup previously generated files when re-generating	No
Delete previously generated files when not re-generated	Yes
Set all free pins as analog (to optimize the power consumption)	No

6. Power Consumption Calculator report

6.1. Microcontroller Selection

Series	STM32F3
Line	STM32F303
MCU	STM32F303K8Tx
Datasheet	025083_Rev5

6.2. Parameter Selection

Temperature	25
Vdd	3.6

7. IPs and Middleware Configuration

7.1. I2C1

I2C: I2C

7.1.1. Parameter Settings:

Timing configuration:

I2C Speed Mode	**Fast Mode ***
I2C Speed Frequency (KHz)	400
Rise Time (ns)	0
Fall Time (ns)	0
Coefficient of Digital Filter	0
Analog Filter	Enabled
Timing	**0x0000020B ***

Slave Features:

Clock No Stretch Mode	Disabled
General Call Address Detection	Disabled
Primary Address Length selection	7-bit
Dual Address Acknowledged	Disabled
Primary slave address	0

7.2. RCC

High Speed Clock (HSE): BYPASS Clock Source

7.2.1. Parameter Settings:

System Parameters:

VDD voltage (V)	3.3
Prefetch Buffer	Enabled
Flash Latency(WS)	2 WS (3 CPU cycle)

RCC Parameters:

HSI Calibration Value	16
HSE Startup Timout Value (ms)	100
LSE Startup Timout Value (ms)	5000

7.3. SYS

Debug: Serial Wire
Timebase Source: SysTick

7.4. TIM1

Clock Source : Internal Clock
Channel1: PWM Generation CH1
Channel2: PWM Generation CH2
Channel3: PWM Generation CH3
Channel4: PWM Generation CH4

7.4.1. Parameter Settings:

Counter Settings:

Prescaler (PSC - 16 bits value)	**7 ***
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	**2039 ***
Internal Clock Division (CKD)	No Division
Repetition Counter (RCR - 16 bits value)	0
auto-reload preload	Disable

Trigger Output (TRGO) Parameters:

Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection TRGO	Reset (UG bit from TIMx_EGR)
Trigger Event Selection TRGO2	Reset (UG bit from TIMx_EGR)

Break And Dead Time management - BRK Configuration:

BRK State	Disable
BRK Polarity	High
BRK Filter (4 bits value)	0

Break And Dead Time management - BRK2 Configuration:

BRK2 State	Disable
BRK2 Polarity	High
BRK2 Filter (4 bits value)	0

Break And Dead Time management - Output Configuration:

Automatic Output State	Disable
Off State Selection for Run Mode (OSSR)	Disable
Off State Selection for Idle Mode (OSSI)	Disable
Lock Configuration	Off

Clear Input:

Clear Input Source	Disable

PWM Generation Channel 1:

Mode	PWM mode 1

Pulse (16 bits value)	0
Fast Mode	Disable
CH Polarity	High
CH Idle State	Reset

PWM Generation Channel 2:

Mode	PWM mode 1
Pulse (16 bits value)	0
Fast Mode	Disable
CH Polarity	High
CH Idle State	Reset

PWM Generation Channel 3:

Mode	PWM mode 1
Pulse (16 bits value)	0
Fast Mode	Disable
CH Polarity	High
CH Idle State	Reset

PWM Generation Channel 4:

Mode	PWM mode 1
Pulse (16 bits value)	0
Fast Mode	Disable
CH Polarity	High
CH Idle State	Reset

*** User modified value**

8. System Configuration

8.1. GPIO configuration

IP	Pin	Signal	GPIO mode	GPIO pull/up pull down	Max Speed	User Label
I2C1	PB6	I2C1_SCL	Alternate Function Open Drain	**No pull up pull down ***	**High ***	
	PB7	I2C1_SDA	Alternate Function Open Drain	**No pull up pull down ***	**High ***	
RCC	PF0 / OSC_IN	RCC_OSC_IN	n/a	n/a	n/a	MCO
SYS	PA13	SYS_JTMS-SWDIO	n/a	n/a	n/a	SWDIO
	PA14	SYS_JTCK-SWCLK	n/a	n/a	n/a	SWCLK
TIM1	PA8	TIM1_CH1	Alternate Function Push Pull	No pull up pull down	Low	
	PA9	TIM1_CH2	Alternate Function Push Pull	No pull up pull down	Low	
	PA10	TIM1_CH3	Alternate Function Push Pull	No pull up pull down	Low	
	PA11	TIM1_CH4	Alternate Function Push Pull	No pull up pull down	Low	
GPIO	PA0	GPIO_EXTI0	**External Interrupt Mode with Rising/Falling edge**	No pull up pull down	n/a	
	PA1	GPIO_EXTI1	**External Interrupt Mode with Falling edge trigger detection**	No pull up pull down	n/a	
	PA2	GPIO_EXTI2	**External Interrupt Mode with Falling edge trigger detection**	No pull up pull down	n/a	
	PA3	GPIO_EXTI3	**External Interrupt Mode with Falling edge trigger detection**	No pull up pull down	n/a	
	PB3	GPIO_Output	Output Push Pull	No pull up pull down	Low	
	PB4	GPIO_Output	Output Push Pull	No pull up pull down	Low	
	PB5	GPIO_Output	Output Push Pull	No pull up pull down	Low	

8.2. DMA configuration

nothing configured in DMA service

8.3. NVIC configuration

Interrupt Table	Enable	Preenmption Priority	SubPriority
Non maskable interrupt	true	0	0
Hard fault interrupt	true	0	0
Memory management fault	true	0	0
Pre-fetch fault, memory access fault	true	0	0
Undefined instruction or illegal state	true	0	0
System service call via SWI instruction	true	0	0
Debug monitor	true	0	0
Pendable request for system service	true	0	0
System tick timer	true	0	0
EXTI line 0 interrupt	true	0	0
EXTI line 1 interrupt	true	0	0
EXTI line 2 and touch sense controller	true	0	0
EXTI line 3 interrupt	true	0	0
PVD interrupt through EXTI line 16	unused		
Flash global interrupt	unused		
RCC global interrupt	unused		
TIM1 break and TIM15 interrupts	unused		
TIM1 update and TIM16 interrupts	unused		
TIM1 trigger and commutation and TIM17 interrupts	unused		
TIM1 capture compare interrupt	unused		
I2C1 event global interrupt / I2C1 wake-up interrupt through EXT line 23	unused		
I2C1 error interrupt	unused		
Floating point unit interrupt	unused		

*** User modified value**

9. Software Pack Report

30 APC 8045 propeller data

====== PERFORMANCE DATA (versus advance ratio and MPH) ======

DEFINITIONS:

J=V/nD (advance ratio)

Ct=T/(rho * n**2 * D**4) (thrust coef.)

Cp=P/(rho * n**3 * D**5) (power coef.)

Pe=Ct*J/Cp (efficiency)

V (model speed in MPH)

PROP RPM = 3000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1202	0.0520	0.004	0.078	0.141
0.7	0.03	0.0660	0.1193	0.0527	0.004	0.079	0.140
1.3	0.06	0.1289	0.1181	0.0535	0.004	0.080	0.139
2.0	0.09	0.1887	0.1168	0.0542	0.004	0.081	0.137
2.7	0.12	0.2454	0.1153	0.0548	0.004	0.082	0.135
3.3	0.15	0.2990	0.1136	0.0554	0.004	0.083	0.133
4.0	0.18	0.3494	0.1117	0.0560	0.004	0.084	0.131
4.6	0.20	0.3968	0.1096	0.0564	0.004	0.084	0.129
5.3	0.23	0.4412	0.1071	0.0567	0.004	0.085	0.126
6.0	0.26	0.4827	0.1041	0.0566	0.004	0.085	0.122
6.6	0.29	0.5210	0.1007	0.0564	0.004	0.084	0.118

7.3	0.32	0.5561	0.0969	0.0559	0.004	0.084	0.114
8.0	0.35	0.5880	0.0927	0.0552	0.004	0.083	0.109
8.6	0.38	0.6167	0.0882	0.0542	0.004	0.081	0.104
9.3	0.41	0.6422	0.0834	0.0530	0.004	0.079	0.098
9.9	0.44	0.6643	0.0785	0.0517	0.004	0.077	0.092
10.6	0.47	0.6834	0.0735	0.0502	0.004	0.075	0.086
11.3	0.50	0.6998	0.0692	0.0490	0.003	0.073	0.081
11.9	0.53	0.7139	0.0647	0.0476	0.003	0.071	0.076
12.6	0.55	0.7268	0.0609	0.0465	0.003	0.069	0.072
13.3	0.58	0.7366	0.0555	0.0440	0.003	0.066	0.065
13.9	0.61	0.7438	0.0499	0.0411	0.003	0.061	0.059
14.6	0.64	0.7482	0.0442	0.0379	0.003	0.057	0.052
15.2	0.67	0.7488	0.0382	0.0342	0.002	0.051	0.045
15.9	0.70	0.7432	0.0321	0.0303	0.002	0.045	0.038
16.6	0.73	0.7276	0.0259	0.0259	0.002	0.039	0.030
17.2	0.76	0.6924	0.0195	0.0214	0.002	0.032	0.023
17.9	0.79	0.6134	0.0131	0.0168	0.001	0.025	0.015
18.6	0.82	0.4373	0.0065	0.0122	0.001	0.018	0.008
19.2	0.85	-0.0100	-0.0001	0.0078	0.001	0.012	0.000

PROP RPM = 4000

V	J	Pe	Ct	Cp	PWR	Torque	Thrust
(mph)	(Adv Ratio)				(Hp)	(In-Lbf)	(Lbf)
0.0	0.00	0.0000	0.1203	0.0516	0.009	0.137	0.251
0.9	0.03	0.0651	0.1193	0.0524	0.009	0.139	0.249

	1.7	0.06	0.1272	0.1182	0.0531	0.009
0.141	0.247					
	2.6	0.09	0.1862	0.1169	0.0539	0.009
0.143	0.244					
	3.5	0.11	0.2421	0.1155	0.0546	0.009
0.145	0.241					
	4.3	0.14	0.2949	0.1138	0.0552	0.009
0.147	0.238					
	5.2	0.17	0.3447	0.1120	0.0557	0.009
0.148	0.234					
	6.1	0.20	0.3913	0.1099	0.0562	0.009
0.149	0.229					
	6.9	0.23	0.4350	0.1074	0.0565	0.010
0.150	0.224					
	7.8	0.26	0.4758	0.1047	0.0566	0.010
0.150	0.218					
	8.7	0.29	0.5136	0.1015	0.0565	0.010
0.150	0.212					
	9.5	0.31	0.5484	0.0981	0.0562	0.009
0.150	0.205					
	10.4	0.34	0.5801	0.0942	0.0557	0.009
0.148	0.197					
	11.3	0.37	0.6089	0.0901	0.0550	0.009
0.146	0.188					
	12.1	0.40	0.6347	0.0856	0.0540	0.009
0.144	0.179					
	13.0	0.43	0.6575	0.0810	0.0529	0.009
0.140	0.169					
	13.9	0.46	0.6774	0.0763	0.0515	0.009
0.137	0.159					
	14.7	0.49	0.6947	0.0714	0.0500	0.008
0.133	0.149					
	15.6	0.51	0.7094	0.0664	0.0482	0.008
0.128	0.139					
	16.5	0.54	0.7218	0.0612	0.0461	0.008
0.122	0.128					
	17.3	0.57	0.7317	0.0558	0.0436	0.007
0.116	0.117					
	18.2	0.60	0.7390	0.0503	0.0408	0.007
0.109	0.105					
	19.1	0.63	0.7438	0.0445	0.0377	0.006
0.100	0.093					
	19.9	0.66	0.7450	0.0386	0.0341	0.006
0.091	0.081					
	20.8	0.69	0.7404	0.0324	0.0301	0.005
0.080	0.068					
	21.7	0.71	0.7262	0.0262	0.0258	0.004
0.068	0.055					
	22.5	0.74	0.6952	0.0198	0.0211	0.004
0.056	0.041					
	23.4	0.77	0.6224	0.0133	0.0165	0.003
0.044	0.028					
	24.3	0.80	0.4528	0.0067	0.0118	0.002
0.031	0.014					
	25.1	0.83	-0.0055	0.0000	0.0072	0.001
0.019	0.000					

PROP RPM = 5000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1204	0.0518	0.017	0.215	0.393
1.1	0.03	0.0641	0.1194	0.0525	0.017	0.218	0.390
2.1	0.06	0.1253	0.1184	0.0533	0.018	0.221	0.386
3.2	0.08	0.1835	0.1171	0.0540	0.018	0.224	0.382
4.3	0.11	0.2388	0.1157	0.0546	0.018	0.227	0.377
5.3	0.14	0.2911	0.1141	0.0552	0.018	0.229	0.372
6.4	0.17	0.3405	0.1122	0.0558	0.018	0.232	0.366
7.5	0.20	0.3869	0.1102	0.0562	0.019	0.233	0.359
8.5	0.23	0.4304	0.1077	0.0565	0.019	0.234	0.351
9.6	0.25	0.4711	0.1050	0.0565	0.019	0.235	0.342
10.7	0.28	0.5089	0.1018	0.0564	0.019	0.234	0.332
11.7	0.31	0.5437	0.0984	0.0561	0.018	0.233	0.321
12.8	0.34	0.5756	0.0946	0.0556	0.018	0.231	0.308
13.9	0.37	0.6044	0.0905	0.0549	0.018	0.228	0.295
15.0	0.39	0.6304	0.0860	0.0539	0.018	0.224	0.281
16.0	0.42	0.6534	0.0814	0.0527	0.017	0.219	0.266
17.1	0.45	0.6735	0.0767	0.0514	0.017	0.213	0.250
18.2	0.48	0.6909	0.0719	0.0499	0.016	0.207	0.234
19.2	0.51	0.7058	0.0669	0.0481	0.016	0.200	0.218
20.3	0.54	0.7183	0.0617	0.0460	0.015	0.191	0.201
21.4	0.56	0.7284	0.0564	0.0436	0.014	0.181	0.184

	22.4	0.59	0.7359	0.0508	0.0409	0.013
0.170	0.166					
	23.5	0.62	0.7410	0.0451	0.0378	0.012
0.157	0.147					
	24.6	0.65	0.7426	0.0392	0.0342	0.011
0.142	0.128					
	25.6	0.68	0.7388	0.0330	0.0302	0.010
0.126	0.108					
	26.7	0.70	0.7259	0.0266	0.0259	0.009
0.107	0.087					
	27.8	0.73	0.6964	0.0201	0.0212	0.007
0.088	0.066					
	28.8	0.76	0.6286	0.0135	0.0164	0.005
0.068	0.044					
	29.9	0.79	0.4632	0.0068	0.0115	0.004
0.048	0.022					
	31.0	0.82	-0.0074	-0.0001	0.0069	0.002
0.028	0.000					

PROP RPM = 6000

V	J	Pe	Ct	Cp	PWR	Torque	Thrust
(mph)	(Adv Ratio)				(Hp)	(In-Lbf)	(Lbf)
0.0	0.00	0.0000	0.1205	0.0519	0.030	0.311	0.566
1.3	0.03	0.0631	0.1196	0.0527	0.030	0.315	0.562
2.5	0.06	0.1233	0.1185	0.0534	0.030	0.319	0.557
3.8	0.08	0.1808	0.1173	0.0541	0.031	0.324	0.551
5.1	0.11	0.2353	0.1160	0.0548	0.031	0.327	0.545
6.3	0.14	0.2871	0.1144	0.0553	0.032	0.331	0.537
7.6	0.17	0.3360	0.1126	0.0559	0.032	0.334	0.529
8.8	0.19	0.3821	0.1105	0.0563	0.032	0.337	0.519
10.1	0.22	0.4254	0.1082	0.0565	0.032	0.338	0.508
11.4	0.25	0.4659	0.1054	0.0566	0.032	0.338	0.495
12.6	0.28	0.5036	0.1024	0.0565	0.032	0.338	0.481
13.9	0.31	0.5385	0.0989	0.0561	0.032	0.336	0.465

15.2	0.33	0.5704	0.0952	0.0556	0.032	0.333
0.447						
16.4	0.36	0.5995	0.0911	0.0549	0.031	0.328
0.428						
17.7	0.39	0.6256	0.0868	0.0540	0.031	0.323
0.408						
18.9	0.42	0.6489	0.0822	0.0528	0.030	0.316
0.386						
20.2	0.44	0.6692	0.0776	0.0515	0.029	0.308
0.364						
21.5	0.47	0.6869	0.0728	0.0500	0.028	0.299
0.342						
22.7	0.50	0.7021	0.0678	0.0483	0.027	0.289
0.318						
24.0	0.53	0.7149	0.0627	0.0463	0.026	0.277
0.294						
25.3	0.56	0.7252	0.0574	0.0440	0.025	0.263
0.270						
26.5	0.58	0.7331	0.0519	0.0413	0.024	0.247
0.244						
27.8	0.61	0.7386	0.0462	0.0382	0.022	0.229
0.217						
29.0	0.64	0.7409	0.0402	0.0347	0.020	0.208
0.189						
30.3	0.67	0.7381	0.0341	0.0308	0.018	0.184
0.160						
31.6	0.69	0.7269	0.0276	0.0263	0.015	0.157
0.129						
32.8	0.72	0.6993	0.0208	0.0215	0.012	0.129
0.098						
34.1	0.75	0.6364	0.0140	0.0165	0.009	0.099
0.066						
35.4	0.78	0.4780	0.0070	0.0114	0.007	0.068
0.033						
36.6	0.81	-0.0021	0.0000	0.0066	0.004	0.039
0.000						

PROP RPM = 7000

V	J	Pe	Ct	Cp	PWR	Torque	Thrust
(mph)	(Adv Ratio)				(Hp)	(In-Lbf)	(Lbf)
0.0	0.00	0.0000	0.1207	0.0522	0.047	0.425	0.772
1.5	0.03	0.0625	0.1198	0.0529	0.048	0.431	0.766
2.9	0.06	0.1223	0.1188	0.0536	0.048	0.436	0.759

4.4	0.08	0.1793	0.1176	0.0543	0.049	0.442	0.752
5.9	0.11	0.2336	0.1162	0.0549	0.050	0.447	0.743
7.3	0.14	0.2851	0.1147	0.0555	0.050	0.452	0.733
8.8	0.17	0.3338	0.1129	0.0560	0.051	0.456	0.722
10.2	0.19	0.3797	0.1109	0.0564	0.051	0.459	0.709
11.7	0.22	0.4229	0.1085	0.0567	0.051	0.461	0.694
13.2	0.25	0.4634	0.1058	0.0567	0.051	0.462	0.677
14.6	0.28	0.5012	0.1027	0.0566	0.051	0.461	0.657
16.1	0.30	0.5361	0.0993	0.0562	0.051	0.458	0.635
17.6	0.33	0.5681	0.0956	0.0557	0.050	0.454	0.611
19.0	0.36	0.5972	0.0915	0.0550	0.050	0.448	0.585
20.5	0.39	0.6235	0.0872	0.0540	0.049	0.440	0.557
22.0	0.41	0.6468	0.0826	0.0529	0.048	0.430	0.528
23.4	0.44	0.6673	0.0779	0.0516	0.047	0.420	0.498
24.9	0.47	0.6851	0.0731	0.0501	0.045	0.408	0.468
26.3	0.50	0.7004	0.0681	0.0483	0.044	0.393	0.436
27.8	0.52	0.7132	0.0630	0.0463	0.042	0.377	0.403
29.3	0.55	0.7237	0.0577	0.0440	0.040	0.358	0.369
30.7	0.58	0.7316	0.0522	0.0413	0.037	0.337	0.334
32.2	0.61	0.7372	0.0464	0.0382	0.035	0.311	0.297
33.7	0.63	0.7395	0.0405	0.0347	0.031	0.283	0.259
35.1	0.66	0.7367	0.0342	0.0307	0.028	0.250	0.218
36.6	0.69	0.7258	0.0276	0.0263	0.024	0.214	0.177
38.1	0.72	0.6985	0.0209	0.0215	0.019	0.175	0.134
39.5	0.75	0.6359	0.0140	0.0165	0.015	0.134	0.090
41.0	0.77	0.4770	0.0071	0.0114	0.010	0.093	0.045
42.4	0.80	-0.0008	0.0000	0.0069	0.006	0.056	0.000

PROP RPM = 8000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1207	0.0520	0.070	0.552	1.008
1.7	0.03	0.0628	0.1198	0.0527	0.071	0.560	1.001
3.3	0.06	0.1229	0.1188	0.0534	0.072	0.567	0.992
5.0	0.08	0.1802	0.1176	0.0540	0.073	0.575	0.982
6.7	0.11	0.2348	0.1163	0.0547	0.074	0.581	0.971
8.4	0.14	0.2866	0.1147	0.0552	0.075	0.587	0.958
10.0	0.17	0.3355	0.1130	0.0558	0.075	0.593	0.944
11.7	0.19	0.3816	0.1110	0.0562	0.076	0.597	0.927
13.4	0.22	0.4250	0.1087	0.0564	0.076	0.600	0.907
15.1	0.25	0.4657	0.1060	0.0565	0.076	0.601	0.885
16.7	0.28	0.5036	0.1029	0.0564	0.076	0.599	0.859
18.4	0.30	0.5388	0.0994	0.0560	0.076	0.596	0.830
20.1	0.33	0.5711	0.0957	0.0555	0.075	0.590	0.799
21.7	0.36	0.6007	0.0916	0.0547	0.074	0.582	0.765
23.4	0.39	0.6274	0.0872	0.0537	0.072	0.571	0.728
25.1	0.41	0.6513	0.0826	0.0525	0.071	0.558	0.690
26.8	0.44	0.6724	0.0779	0.0511	0.069	0.544	0.651
28.4	0.47	0.6909	0.0730	0.0496	0.067	0.527	0.610
30.1	0.50	0.7069	0.0680	0.0478	0.064	0.508	0.568
31.8	0.52	0.7204	0.0628	0.0457	0.062	0.486	0.524
33.4	0.55	0.7314	0.0574	0.0433	0.058	0.461	0.479
35.1	0.58	0.7399	0.0518	0.0405	0.055	0.431	0.432

36.8	0.61	0.7457	0.0458	0.0373	0.050	0.397
0.383						
38.5	0.63	0.7478	0.0397	0.0337	0.045	0.358
0.332						
40.1	0.66	0.7443	0.0333	0.0296	0.040	0.315
0.278						
41.8	0.69	0.7322	0.0267	0.0252	0.034	0.268
0.223						
43.5	0.72	0.7028	0.0202	0.0206	0.028	0.219
0.168						
45.2	0.75	0.6373	0.0135	0.0158	0.021	0.168
0.113						
46.8	0.77	0.4722	0.0068	0.0111	0.015	0.118
0.057						
48.5	0.80	-0.0001	0.0000	0.0070	0.009	0.074
0.000						

PROP RPM = 9000

V	J	Pe	Ct	Cp	PWR	Torque	Thrust
(mph)	(Adv Ratio)				(Hp)	(In-Lbf)	(Lbf)
0.0	0.00	0.0000	0.1209	0.0512	0.098	0.689	1.278
1.9	0.03	0.0640	0.1200	0.0520	0.100	0.699	1.268
3.8	0.06	0.1250	0.1190	0.0527	0.101	0.710	1.257
5.7	0.08	0.1831	0.1178	0.0535	0.103	0.720	1.245
7.6	0.11	0.2383	0.1165	0.0542	0.104	0.729	1.231
9.4	0.14	0.2906	0.1149	0.0548	0.105	0.737	1.215
11.3	0.17	0.3400	0.1132	0.0553	0.106	0.745	1.196
13.2	0.19	0.3864	0.1112	0.0558	0.107	0.751	1.175
15.1	0.22	0.4301	0.1088	0.0561	0.108	0.755	1.150
17.0	0.25	0.4710	0.1061	0.0562	0.108	0.756	1.121
18.9	0.28	0.5093	0.1030	0.0560	0.108	0.754	1.088
20.8	0.30	0.5448	0.0995	0.0556	0.107	0.749	1.051
22.7	0.33	0.5776	0.0956	0.0551	0.106	0.741	1.011

24.6	0.36	0.6075	0.0915	0.0543	0.104	0.730
0.967						
26.4	0.39	0.6348	0.0871	0.0532	0.102	0.716
0.920						
28.3	0.42	0.6594	0.0824	0.0519	0.100	0.699
0.871						
30.2	0.44	0.6812	0.0776	0.0505	0.097	0.679
0.820						
32.1	0.47	0.7004	0.0726	0.0488	0.094	0.657
0.768						
34.0	0.50	0.7172	0.0674	0.0469	0.090	0.631
0.713						
35.9	0.53	0.7313	0.0621	0.0447	0.086	0.602
0.657						
37.8	0.55	0.7428	0.0567	0.0423	0.081	0.569
0.599						
39.7	0.58	0.7517	0.0509	0.0394	0.076	0.530
0.538						
41.6	0.61	0.7574	0.0449	0.0361	0.069	0.486
0.475						
43.4	0.64	0.7589	0.0387	0.0325	0.062	0.437
0.409						
45.3	0.66	0.7548	0.0324	0.0285	0.055	0.384
0.342						
47.2	0.69	0.7411	0.0259	0.0242	0.046	0.326
0.274						
49.1	0.72	0.7109	0.0197	0.0199	0.038	0.268
0.208						
51.0	0.75	0.6422	0.0131	0.0153	0.029	0.206
0.139						
52.9	0.78	0.4723	0.0066	0.0109	0.021	0.146
0.070						
54.8	0.80	-0.0018	0.0000	0.0068	0.013	0.091
0.000						

PROP RPM = 10000

V	J	Pe	Ct	Cp	PWR	Torque	Thrust
(mph)	(Adv Ratio)				(Hp)	(In-Lbf)	(Lbf)
0.0	0.00	0.0000	0.1210	0.0505	0.133	0.839	1.579
2.1	0.03	0.0650	0.1202	0.0513	0.135	0.853	1.568
4.2	0.06	0.1268	0.1192	0.0522	0.138	0.867	1.555
6.3	0.08	0.1855	0.1180	0.0530	0.140	0.880	1.540

	8.4	0.11	0.2411	0.1167	0.0537	0.142
0.892	1.523					
	10.5	0.14	0.2938	0.1152	0.0544	0.143
0.904	1.503					
	12.6	0.17	0.3434	0.1134	0.0550	0.145
0.914	1.480					
	14.7	0.19	0.3901	0.1114	0.0555	0.146
0.922	1.454					
	16.8	0.22	0.4339	0.1091	0.0558	0.147
0.927	1.423					
	18.9	0.25	0.4751	0.1063	0.0559	0.147
0.929	1.387					
	21.0	0.28	0.5136	0.1032	0.0558	0.147
0.926	1.346					
	23.1	0.31	0.5494	0.0997	0.0554	0.146
0.920	1.300					
	25.2	0.33	0.5825	0.0958	0.0548	0.144
0.910	1.250					
	27.3	0.36	0.6130	0.0916	0.0539	0.142
0.896	1.195					
	29.4	0.39	0.6408	0.0871	0.0528	0.139
0.878	1.137					
	31.5	0.42	0.6661	0.0824	0.0515	0.136
0.855	1.075					
	33.6	0.44	0.6886	0.0775	0.0500	0.132
0.831	1.012					
	35.7	0.47	0.7086	0.0725	0.0483	0.127
0.802	0.946					
	37.8	0.50	0.7262	0.0672	0.0462	0.122
0.768	0.877					
	39.9	0.53	0.7410	0.0618	0.0440	0.116
0.731	0.807					
	42.1	0.56	0.7531	0.0562	0.0414	0.109
0.689	0.734					
	44.2	0.58	0.7625	0.0503	0.0385	0.101
0.639	0.657					
	46.3	0.61	0.7681	0.0443	0.0352	0.093
0.585	0.578					
	48.4	0.64	0.7695	0.0382	0.0317	0.084
0.526	0.498					
	50.5	0.67	0.7654	0.0320	0.0279	0.073
0.463	0.418					
	52.6	0.69	0.7519	0.0258	0.0238	0.063
0.396	0.337					
	54.7	0.72	0.7201	0.0195	0.0195	0.052
0.325	0.254					
	56.8	0.75	0.6505	0.0131	0.0151	0.040
0.250	0.171					
	58.9	0.78	0.4777	0.0066	0.0107	0.028
0.178	0.086					
	61.0	0.80	-0.0012	0.0000	0.0067	0.018
0.111	0.000					

PROP RPM = 11000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1212	0.0499	0.175	1.003	1.914
2.3	0.03	0.0659	0.1204	0.0508	0.178	1.021	1.901
4.6	0.06	0.1285	0.1194	0.0517	0.181	1.039	1.885
7.0	0.08	0.1879	0.1183	0.0526	0.184	1.056	1.867
9.3	0.11	0.2439	0.1170	0.0534	0.187	1.073	1.847
11.6	0.14	0.2969	0.1154	0.0541	0.190	1.087	1.823
13.9	0.17	0.3468	0.1137	0.0547	0.192	1.100	1.795
16.2	0.19	0.3936	0.1117	0.0553	0.194	1.111	1.764
18.5	0.22	0.4377	0.1094	0.0556	0.195	1.118	1.727
20.9	0.25	0.4790	0.1066	0.0557	0.196	1.120	1.683
23.2	0.28	0.5178	0.1035	0.0556	0.195	1.117	1.633
25.5	0.31	0.5539	0.0999	0.0552	0.194	1.110	1.577
27.8	0.33	0.5873	0.0960	0.0546	0.191	1.097	1.515
30.1	0.36	0.6182	0.0918	0.0537	0.188	1.079	1.449
32.5	0.39	0.6466	0.0872	0.0525	0.184	1.056	1.377
34.8	0.42	0.6725	0.0824	0.0511	0.179	1.028	1.301
37.1	0.45	0.6958	0.0775	0.0496	0.174	0.996	1.223
39.4	0.47	0.7165	0.0724	0.0478	0.168	0.961	1.143
41.7	0.50	0.7349	0.0670	0.0457	0.160	0.918	1.058
44.1	0.53	0.7504	0.0615	0.0433	0.152	0.871	0.971
46.4	0.56	0.7631	0.0558	0.0407	0.143	0.818	0.881
48.7	0.58	0.7729	0.0499	0.0377	0.132	0.758	0.788
51.0	0.61	0.7785	0.0439	0.0345	0.121	0.694	0.693

53.3	0.64	0.7797	0.0378	0.0310	0.109	0.623
0.597						
55.6	0.67	0.7753	0.0317	0.0273	0.096	0.548
0.500						
58.0	0.70	0.7610	0.0255	0.0233	0.082	0.469
0.403						
60.3	0.72	0.7283	0.0193	0.0191	0.067	0.385
0.304						
62.6	0.75	0.6561	0.0129	0.0147	0.052	0.296
0.203						
64.9	0.78	0.4767	0.0064	0.0105	0.037	0.210
0.101						
67.2	0.81	-0.0002	0.0000	0.0066	0.023	0.133
0.000						

PROP RPM = 12000

V	J	Pe	Ct	Cp	PWR	Torque	Thrust
(mph)	(Adv Ratio)				(Hp)	(In-Lbf)	(Lbf)
0.0	0.00	0.0000	0.1215	0.0494	0.225	1.181	2.283
2.5	0.03	0.0669	0.1207	0.0503	0.229	1.204	2.267
5.1	0.06	0.1302	0.1197	0.0513	0.234	1.227	2.249
7.6	0.08	0.1900	0.1186	0.0522	0.238	1.249	2.228
10.1	0.11	0.2465	0.1173	0.0531	0.242	1.270	2.204
12.7	0.14	0.2997	0.1158	0.0539	0.245	1.289	2.176
15.2	0.17	0.3498	0.1141	0.0546	0.249	1.305	2.143
17.7	0.20	0.3968	0.1121	0.0551	0.251	1.319	2.106
20.3	0.22	0.4410	0.1098	0.0555	0.253	1.328	2.062
22.8	0.25	0.4825	0.1070	0.0557	0.254	1.331	2.010
25.3	0.28	0.5214	0.1038	0.0555	0.253	1.328	1.950
27.9	0.31	0.5578	0.1002	0.0551	0.251	1.318	1.883
30.4	0.33	0.5915	0.0962	0.0544	0.248	1.302	1.808
33.0	0.36	0.6228	0.0920	0.0535	0.244	1.281	1.728

35.5	0.39	0.6518	0.0873	0.0523	0.238	1.252	1.641
38.0	0.42	0.6783	0.0825	0.0509	0.232	1.217	1.550
40.6	0.45	0.7023	0.0775	0.0492	0.224	1.178	1.456
43.1	0.47	0.7238	0.0724	0.0474	0.216	1.134	1.359
45.6	0.50	0.7430	0.0669	0.0452	0.206	1.081	1.257
48.2	0.53	0.7591	0.0613	0.0428	0.195	1.024	1.152
50.7	0.56	0.7725	0.0556	0.0401	0.183	0.960	1.044
53.2	0.59	0.7828	0.0497	0.0372	0.169	0.890	0.934
55.8	0.61	0.7886	0.0438	0.0341	0.155	0.815	0.823
58.3	0.64	0.7899	0.0377	0.0306	0.140	0.733	0.709
60.8	0.67	0.7853	0.0316	0.0269	0.123	0.644	0.594
63.4	0.70	0.7704	0.0254	0.0230	0.105	0.550	0.477
65.9	0.72	0.7366	0.0191	0.0188	0.086	0.450	0.359
68.4	0.75	0.6645	0.0129	0.0146	0.066	0.349	0.242
71.0	0.78	0.4869	0.0065	0.0104	0.047	0.249	0.122
73.5	0.81	-0.0019	0.0000	0.0065	0.029	0.155	0.000

PROP RPM = 13000

V	J	Pe	Ct	Cp	PWR	Torque	Thrust
(mph)	(Adv Ratio)				(Hp)	(In-Lbf)	(Lbf)
0.0	0.00	0.0000	0.1221	0.0517	0.299	1.450	2.693
2.8	0.03	0.0648	0.1213	0.0524	0.304	1.471	2.675
5.5	0.06	0.1268	0.1203	0.0531	0.308	1.492	2.654
8.3	0.08	0.1859	0.1192	0.0538	0.312	1.512	2.629
11.0	0.11	0.2422	0.1179	0.0545	0.316	1.531	2.601

	13.8	0.14	0.2957	0.1164	0.0551	0.319
1.547	2.567					
	16.5	0.17	0.3464	0.1147	0.0556	0.322
1.561	2.529					
	19.3	0.20	0.3943	0.1127	0.0560	0.324
1.572	2.485					
	22.1	0.22	0.4397	0.1103	0.0562	0.325
1.578	2.433					
	24.8	0.25	0.4825	0.1075	0.0561	0.325
1.576	2.371					
	27.6	0.28	0.5227	0.1042	0.0558	0.323
1.567	2.299					
	30.3	0.31	0.5601	0.1006	0.0553	0.320
1.552	2.217					
	33.1	0.34	0.5946	0.0965	0.0545	0.316
1.531	2.128					
	35.8	0.36	0.6265	0.0921	0.0535	0.310
1.503	2.032					
	38.6	0.39	0.6559	0.0874	0.0522	0.303
1.467	1.928					
	41.4	0.42	0.6827	0.0825	0.0507	0.294
1.425	1.820					
	44.1	0.45	0.7071	0.0775	0.0491	0.284
1.378	1.708					
	46.9	0.48	0.7290	0.0722	0.0471	0.273
1.323	1.592					
	49.6	0.50	0.7486	0.0667	0.0449	0.260
1.260	1.470					
	52.4	0.53	0.7650	0.0610	0.0424	0.246
1.191	1.346					
	55.1	0.56	0.7787	0.0552	0.0397	0.230
1.115	1.218					
	57.9	0.59	0.7892	0.0494	0.0368	0.213
1.033	1.089					
	60.7	0.62	0.7950	0.0435	0.0337	0.195
0.945	0.958					
	63.4	0.64	0.7966	0.0374	0.0303	0.175
0.850	0.826					
	66.2	0.67	0.7920	0.0313	0.0266	0.154
0.746	0.691					
	68.9	0.70	0.7772	0.0252	0.0227	0.131
0.637	0.556					
	71.7	0.73	0.7437	0.0190	0.0186	0.108
0.523	0.420					
	74.4	0.76	0.6691	0.0128	0.0144	0.083
0.405	0.281					
	77.2	0.78	0.4861	0.0064	0.0103	0.060
0.290	0.141					
	80.0	0.81	-0.0013	0.0000	0.0066	0.038
0.184	0.000					

PROP RPM = 13999

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1232	0.0577	0.417	1.879	3.152
3.0	0.03	0.0594	0.1224	0.0578	0.418	1.882	3.131
6.0	0.06	0.1177	0.1214	0.0579	0.419	1.884	3.106
8.9	0.08	0.1747	0.1203	0.0579	0.419	1.887	3.077
11.9	0.11	0.2302	0.1190	0.0580	0.420	1.889	3.043
14.9	0.14	0.2839	0.1174	0.0580	0.420	1.890	3.003
17.9	0.17	0.3357	0.1156	0.0580	0.419	1.887	2.957
20.8	0.20	0.3855	0.1136	0.0579	0.419	1.884	2.905
23.8	0.22	0.4330	0.1112	0.0576	0.417	1.876	2.843
26.8	0.25	0.4782	0.1083	0.0572	0.413	1.861	2.769
29.8	0.28	0.5208	0.1049	0.0565	0.409	1.840	2.682
32.7	0.31	0.5600	0.1011	0.0557	0.403	1.814	2.586
35.7	0.34	0.5958	0.0970	0.0548	0.396	1.784	2.480
38.7	0.36	0.6283	0.0925	0.0537	0.388	1.748	2.366
41.7	0.39	0.6582	0.0877	0.0523	0.379	1.704	2.243
44.6	0.42	0.6854	0.0827	0.0508	0.367	1.654	2.116
47.6	0.45	0.7101	0.0776	0.0491	0.355	1.598	1.985
50.6	0.48	0.7323	0.0723	0.0471	0.341	1.533	1.849
53.6	0.50	0.7522	0.0667	0.0448	0.324	1.458	1.706
56.5	0.53	0.7691	0.0610	0.0423	0.306	1.377	1.561
59.5	0.56	0.7831	0.0552	0.0395	0.286	1.288	1.412
62.5	0.59	0.7941	0.0494	0.0366	0.265	1.192	1.262
65.5	0.62	0.8004	0.0434	0.0335	0.242	1.090	1.111
68.4	0.65	0.8025	0.0374	0.0301	0.217	0.979	0.956

71.4	0.67	0.7984	0.0313	0.0264	0.191	0.859
0.800						
74.4	0.70	0.7840	0.0252	0.0225	0.163	0.733
0.644						
77.4	0.73	0.7507	0.0190	0.0185	0.134	0.601
0.486						
80.3	0.76	0.6756	0.0127	0.0143	0.103	0.465
0.326						
83.3	0.79	0.4892	0.0064	0.0102	0.074	0.333
0.163						
86.3	0.81	-0.0032	0.0000	0.0065	0.047	0.211
-0.001						

PROP RPM = 15000

V	J	Pe	Ct	Cp	PWR	Torque	Thrust
(mph)	(Adv Ratio)				(Hp)	(In-Lbf)	(Lbf)
0.0	0.00	0.0000	0.1244	0.0638	0.568	2.385	3.653
3.2	0.03	0.0551	0.1236	0.0632	0.563	2.364	3.629
6.4	0.06	0.1103	0.1226	0.0627	0.557	2.342	3.600
9.6	0.08	0.1654	0.1214	0.0621	0.552	2.321	3.565
12.8	0.11	0.2200	0.1201	0.0616	0.548	2.301	3.525
16.0	0.14	0.2738	0.1185	0.0610	0.543	2.280	3.478
19.2	0.17	0.3266	0.1166	0.0604	0.537	2.258	3.424
22.4	0.20	0.3780	0.1145	0.0598	0.532	2.234	3.361
25.6	0.23	0.4277	0.1120	0.0591	0.526	2.208	3.288
28.8	0.25	0.4751	0.1090	0.0582	0.518	2.176	3.201
32.0	0.28	0.5199	0.1055	0.0572	0.509	2.139	3.098
35.2	0.31	0.5609	0.1016	0.0562	0.500	2.100	2.983
38.4	0.34	0.5979	0.0974	0.0551	0.490	2.059	2.858
41.6	0.37	0.6311	0.0928	0.0539	0.479	2.014	2.724
44.8	0.39	0.6614	0.0879	0.0524	0.466	1.960	2.580

48.1	0.42	0.6889	0.0828	0.0508	0.452	1.899	2.430
51.3	0.45	0.7139	0.0776	0.0490	0.436	1.832	2.277
54.5	0.48	0.7364	0.0722	0.0470	0.418	1.756	2.119
57.7	0.51	0.7565	0.0665	0.0446	0.397	1.669	1.954
60.9	0.54	0.7737	0.0608	0.0421	0.375	1.574	1.785
64.1	0.56	0.7880	0.0550	0.0393	0.350	1.470	1.614
67.3	0.59	0.7992	0.0491	0.0364	0.324	1.360	1.441
70.5	0.62	0.8058	0.0431	0.0332	0.295	1.240	1.266
73.7	0.65	0.8082	0.0370	0.0297	0.264	1.110	1.087
76.9	0.68	0.8040	0.0309	0.0260	0.231	0.972	0.907
80.1	0.70	0.7891	0.0248	0.0221	0.197	0.827	0.727
83.3	0.73	0.7543	0.0186	0.0181	0.161	0.675	0.545
86.5	0.76	0.6738	0.0123	0.0139	0.124	0.520	0.362
89.7	0.79	0.4888	0.0063	0.0102	0.091	0.382	0.186
92.9	0.82	-0.0045	0.0000	0.0065	0.058	0.243	-0.001

PROP RPM = 16000

V	J	Pe	Ct	Cp	PWR	Torque	Thrust
(mph)	(Adv Ratio)				(Hp)	(In-Lbf)	(Lbf)
0.0	0.00	0.0000	0.1256	0.0702	0.757	2.984	4.196
3.4	0.03	0.0509	0.1248	0.0689	0.744	2.932	4.169
6.8	0.06	0.1029	0.1239	0.0677	0.731	2.881	4.138
10.2	0.08	0.1558	0.1227	0.0665	0.718	2.827	4.097
13.6	0.11	0.2091	0.1213	0.0653	0.705	2.778	4.053
17.1	0.14	0.2625	0.1197	0.0642	0.693	2.728	3.998

	20.5	0.17	0.3158	0.1178	0.0630	0.680
2.679	3.935					
	23.9	0.20	0.3683	0.1156	0.0619	0.668
2.631	3.863					
	27.3	0.23	0.4196	0.1131	0.0607	0.655
2.581	3.778					
	30.7	0.25	0.4690	0.1101	0.0594	0.642
2.528	3.677					
	34.1	0.28	0.5160	0.1065	0.0581	0.627
2.471	3.558					
	37.5	0.31	0.5588	0.1025	0.0568	0.613
2.415	3.424					
	40.9	0.34	0.5969	0.0982	0.0555	0.600
2.362	3.279					
	44.3	0.37	0.6309	0.0935	0.0542	0.585
2.306	3.123					
	47.8	0.39	0.6616	0.0885	0.0527	0.569
2.242	2.958					
	51.2	0.42	0.6895	0.0834	0.0511	0.551
2.172	2.786					
	54.6	0.45	0.7148	0.0782	0.0492	0.532
2.094	2.611					
	58.0	0.48	0.7377	0.0727	0.0472	0.509
2.006	2.430					
	61.4	0.51	0.7583	0.0671	0.0449	0.484
1.908	2.243					
	64.8	0.53	0.7760	0.0614	0.0423	0.457
1.800	2.052					
	68.2	0.56	0.7909	0.0556	0.0395	0.427
1.681	1.856					
	71.6	0.59	0.8028	0.0497	0.0366	0.395
1.555	1.659					
	75.0	0.62	0.8103	0.0437	0.0334	0.360
1.419	1.459					
	78.5	0.65	0.8135	0.0376	0.0299	0.323
1.272	1.256					
	81.9	0.68	0.8108	0.0316	0.0263	0.284
1.119	1.055					
	85.3	0.70	0.7982	0.0256	0.0225	0.243
0.958	0.854					
	88.7	0.73	0.7673	0.0195	0.0186	0.200
0.789	0.650					
	92.1	0.76	0.6949	0.0132	0.0144	0.156
0.615	0.441					
	95.5	0.79	0.5179	0.0069	0.0105	0.113
0.447	0.231					
	98.9	0.82	-0.0055	0.0000	0.0063	0.069
0.270	-0.001					

PROP RPM = 17000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1268	0.0769	0.996	3.691	4.783
3.6	0.03	0.0475	0.1261	0.0750	0.971	3.600	4.754
7.3	0.06	0.0968	0.1251	0.0730	0.946	3.506	4.718
10.9	0.08	0.1477	0.1239	0.0711	0.921	3.414	4.673
14.6	0.11	0.2000	0.1225	0.0692	0.897	3.324	4.621
18.2	0.14	0.2532	0.1209	0.0674	0.873	3.238	4.559
21.8	0.17	0.3069	0.1190	0.0657	0.851	3.155	4.486
25.5	0.20	0.3606	0.1167	0.0640	0.829	3.073	4.402
29.1	0.23	0.4138	0.1141	0.0623	0.807	2.993	4.304
32.7	0.25	0.4653	0.1110	0.0606	0.785	2.912	4.185
36.4	0.28	0.5144	0.1073	0.0589	0.763	2.829	4.047
40.0	0.31	0.5590	0.1032	0.0574	0.743	2.754	3.891
43.7	0.34	0.5983	0.0987	0.0559	0.724	2.685	3.722
47.3	0.37	0.6329	0.0939	0.0545	0.706	2.617	3.542
50.9	0.40	0.6640	0.0889	0.0529	0.686	2.542	3.352
54.6	0.42	0.6921	0.0837	0.0512	0.663	2.460	3.156
58.2	0.45	0.7177	0.0784	0.0493	0.639	2.369	2.955
61.8	0.48	0.7409	0.0729	0.0472	0.612	2.268	2.748
65.5	0.51	0.7617	0.0672	0.0449	0.581	2.155	2.535
69.1	0.54	0.7798	0.0615	0.0423	0.548	2.031	2.318
72.8	0.56	0.7950	0.0556	0.0395	0.511	1.896	2.096
76.4	0.59	0.8072	0.0496	0.0365	0.472	1.751	1.872
80.0	0.62	0.8151	0.0436	0.0332	0.430	1.596	1.644
83.7	0.65	0.8188	0.0375	0.0298	0.386	1.430	1.415
87.3	0.68	0.8161	0.0314	0.0261	0.338	1.252	1.184

90.9	0.71	0.8035	0.0254	0.0223	0.289	1.073	0.959
94.6	0.73	0.7708	0.0191	0.0182	0.236	0.874	0.720
98.2	0.76	0.6944	0.0129	0.0142	0.184	0.681	0.487
101.9	0.79	0.5041	0.0065	0.0102	0.132	0.489	0.245
105.5	0.82	0.0005	0.0000	0.0064	0.083	0.308	0.000

PROP RPM = 18000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1280	0.0838	1.288	4.511	5.412
3.9	0.03	0.0443	0.1273	0.0812	1.249	4.372	5.381
7.7	0.06	0.0909	0.1264	0.0786	1.208	4.230	5.342
11.6	0.08	0.1398	0.1252	0.0760	1.168	4.088	5.293
15.4	0.11	0.1908	0.1238	0.0734	1.128	3.950	5.233
19.3	0.14	0.2435	0.1221	0.0709	1.090	3.818	5.163
23.1	0.17	0.2974	0.1202	0.0686	1.054	3.691	5.081
27.0	0.20	0.3520	0.1179	0.0663	1.019	3.569	4.984
30.8	0.23	0.4066	0.1153	0.0641	0.986	3.451	4.873
34.7	0.25	0.4600	0.1121	0.0620	0.953	3.337	4.738
38.6	0.28	0.5112	0.1083	0.0599	0.921	3.225	4.579
42.4	0.31	0.5577	0.1041	0.0580	0.892	3.125	4.400
46.3	0.34	0.5982	0.0995	0.0564	0.868	3.038	4.206
50.1	0.37	0.6335	0.0946	0.0549	0.844	2.956	4.001
54.0	0.40	0.6649	0.0895	0.0533	0.819	2.868	3.784
57.8	0.42	0.6934	0.0842	0.0515	0.792	2.773	3.561

61.7	0.45	0.7193	0.0788	0.0496	0.762	2.669	3.333
65.5	0.48	0.7428	0.0733	0.0474	0.729	2.553	3.099
69.4	0.51	0.7640	0.0676	0.0451	0.693	2.425	2.859
73.3	0.54	0.7824	0.0619	0.0425	0.653	2.286	2.615
77.1	0.57	0.7981	0.0560	0.0397	0.610	2.135	2.367
81.0	0.59	0.8108	0.0501	0.0367	0.564	1.975	2.118
84.8	0.62	0.8194	0.0440	0.0334	0.514	1.799	1.861
88.7	0.65	0.8237	0.0379	0.0299	0.460	1.611	1.602
92.5	0.68	0.8220	0.0319	0.0263	0.404	1.416	1.347
96.4	0.71	0.8099	0.0257	0.0224	0.344	1.206	1.085
100.3	0.74	0.7783	0.0194	0.0183	0.281	0.984	0.818
104.1	0.76	0.7022	0.0130	0.0142	0.218	0.764	0.552
108.0	0.79	0.5112	0.0066	0.0102	0.156	0.547	0.278
111.8	0.82	-0.0005	0.0000	0.0064	0.099	0.345	0.000

PROP RPM = 19000

V	J	Pe	Ct	Cp	PWR	Torque	Thrust
(mph)	(Adv Ratio)				(Hp)	(In-Lbf)	(Lbf)
0.0	0.00	0.0000	0.1292	0.0910	1.644	5.455	6.085
4.1	0.03	0.0415	0.1285	0.0877	1.585	5.258	6.054
8.1	0.06	0.0857	0.1276	0.0843	1.525	5.059	6.012
12.2	0.08	0.1326	0.1265	0.0810	1.465	4.858	5.958
16.3	0.11	0.1823	0.1251	0.0777	1.405	4.659	5.891
20.4	0.14	0.2343	0.1234	0.0745	1.348	4.470	5.812
24.4	0.17	0.2882	0.1214	0.0716	1.294	4.292	5.719

	28.5	0.20	0.3434	0.1191	0.0687	1.242
4.121	5.610					
	32.6	0.23	0.3993	0.1164	0.0660	1.194
3.959	5.484					
	36.7	0.25	0.4545	0.1132	0.0635	1.147
3.806	5.332					
	40.7	0.28	0.5076	0.1094	0.0610	1.103
3.658	5.152					
	44.8	0.31	0.5562	0.1050	0.0588	1.063
3.527	4.948					
	48.9	0.34	0.5979	0.1004	0.0570	1.031
3.420	4.728					
	53.0	0.37	0.6338	0.0954	0.0554	1.002
3.323	4.495					
	57.0	0.40	0.6656	0.0902	0.0537	0.971
3.221	4.249					
	61.1	0.42	0.6944	0.0849	0.0519	0.938
3.113	3.998					
	65.2	0.45	0.7206	0.0794	0.0499	0.902
2.993	3.741					
	69.3	0.48	0.7444	0.0738	0.0477	0.863
2.863	3.478					
	73.3	0.51	0.7660	0.0682	0.0453	0.820
2.719	3.210					
	77.4	0.54	0.7849	0.0624	0.0427	0.773
2.563	2.937					
	81.5	0.57	0.8011	0.0564	0.0399	0.721
2.392	2.658					
	85.6	0.59	0.8143	0.0504	0.0368	0.665
2.206	2.373					
	89.6	0.62	0.8235	0.0442	0.0334	0.605
2.005	2.083					
	93.7	0.65	0.8284	0.0380	0.0299	0.540
1.791	1.789					
	97.8	0.68	0.8271	0.0319	0.0262	0.473
1.569	1.500					
	101.9	0.71	0.8151	0.0259	0.0225	0.407
1.350	1.221					
	105.9	0.74	0.7841	0.0196	0.0184	0.332
1.102	0.922					
	110.0	0.76	0.7057	0.0131	0.0142	0.257
0.853	0.619					
	114.1	0.79	0.5112	0.0066	0.0103	0.186
0.615	0.312					
	118.2	0.82	0.0005	0.0000	0.0065	0.118
0.392	0.000					

PROP RPM = 20000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1304	0.0991	2.089	6.584	6.808
4.3	0.03	0.0387	0.1298	0.0950	2.004	6.315	6.776
8.6	0.06	0.0804	0.1290	0.0909	1.918	6.043	6.734
12.9	0.08	0.1252	0.1279	0.0869	1.832	5.772	6.677
17.2	0.11	0.1733	0.1265	0.0827	1.745	5.498	6.605
21.5	0.14	0.2246	0.1249	0.0787	1.660	5.232	6.517
25.8	0.17	0.2784	0.1229	0.0750	1.582	4.985	6.413
30.0	0.20	0.3342	0.1205	0.0715	1.508	4.752	6.290
34.3	0.23	0.3913	0.1178	0.0682	1.438	4.533	6.147
38.6	0.25	0.4483	0.1145	0.0651	1.374	4.329	5.978
42.9	0.28	0.5033	0.1106	0.0623	1.313	4.137	5.774
47.2	0.31	0.5540	0.1062	0.0597	1.259	3.968	5.542
51.5	0.34	0.5970	0.1014	0.0577	1.217	3.836	5.292
55.8	0.37	0.6336	0.0964	0.0560	1.181	3.721	5.029
60.1	0.40	0.6658	0.0911	0.0542	1.144	3.605	4.753
64.4	0.42	0.6949	0.0857	0.0524	1.105	3.481	4.472
68.7	0.45	0.7214	0.0802	0.0504	1.062	3.347	4.184
73.0	0.48	0.7456	0.0746	0.0481	1.015	3.200	3.891
77.3	0.51	0.7675	0.0688	0.0457	0.964	3.037	3.591
81.5	0.54	0.7869	0.0629	0.0430	0.908	2.860	3.285
85.8	0.57	0.8037	0.0569	0.0401	0.846	2.667	2.971
90.1	0.59	0.8173	0.0509	0.0371	0.781	2.462	2.657
94.4	0.62	0.8272	0.0448	0.0338	0.712	2.243	2.338
98.7	0.65	0.8329	0.0386	0.0302	0.637	2.009	2.017
103.0	0.68	0.8325	0.0325	0.0265	0.559	1.763	1.696
107.3	0.71	0.8218	0.0262	0.0226	0.476	1.499	1.366

111.6	0.74	0.7882	0.0197	0.0184	0.389	1.225	1.029
115.9	0.76	0.7017	0.0131	0.0143	0.301	0.950	0.684
120.2	0.79	0.5015	0.0065	0.0103	0.217	0.683	0.339
124.5	0.82	0.0003	0.0000	0.0070	0.149	0.468	0.000

PROP RPM = 21000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1318	0.1083	2.644	7.935	7.584
4.5	0.03	0.0360	0.1312	0.1034	2.525	7.578	7.550
9.0	0.06	0.0751	0.1305	0.0985	2.404	7.215	7.508
13.5	0.09	0.1178	0.1295	0.0935	2.283	6.852	7.453
18.0	0.11	0.1641	0.1282	0.0886	2.162	6.488	7.376
22.5	0.14	0.2144	0.1265	0.0836	2.042	6.127	7.280
27.1	0.17	0.2682	0.1245	0.0790	1.928	5.786	7.164
31.6	0.20	0.3244	0.1221	0.0747	1.824	5.474	7.027
36.1	0.23	0.3825	0.1193	0.0707	1.727	5.183	6.867
40.6	0.26	0.4414	0.1161	0.0671	1.638	4.915	6.678
45.1	0.28	0.4984	0.1121	0.0637	1.556	4.670	6.448
49.6	0.31	0.5514	0.1075	0.0608	1.484	4.453	6.185
54.1	0.34	0.5959	0.1026	0.0586	1.429	4.290	5.902
58.6	0.37	0.6332	0.0974	0.0567	1.384	4.154	5.605
63.1	0.40	0.6657	0.0921	0.0549	1.340	4.020	5.297
67.6	0.43	0.6951	0.0866	0.0530	1.293	3.881	4.983
72.2	0.45	0.7220	0.0810	0.0509	1.243	3.729	4.662

76.7	0.48	0.7465	0.0753	0.0486	1.187	3.564	4.335
81.2	0.51	0.7688	0.0695	0.0462	1.127	3.381	4.001
85.7	0.54	0.7887	0.0636	0.0434	1.060	3.181	3.659
90.2	0.57	0.8060	0.0576	0.0405	0.988	2.966	3.312
94.7	0.60	0.8201	0.0515	0.0374	0.913	2.739	2.963
99.2	0.62	0.8307	0.0453	0.0340	0.831	2.493	2.608
103.7	0.65	0.8370	0.0392	0.0305	0.745	2.237	2.256
108.2	0.68	0.8367	0.0329	0.0267	0.652	1.957	1.891
112.7	0.71	0.8242	0.0264	0.0227	0.555	1.665	1.521
117.3	0.74	0.7677	0.0200	0.0192	0.468	1.406	1.150
121.8	0.77	0.6908	0.0131	0.0145	0.354	1.062	0.753
126.3	0.79	0.4752	0.0065	0.0108	0.263	0.790	0.371
130.8	0.82	0.0105	0.0001	0.0080	0.194	0.583	0.006

PROP RPM = 22000

V	J	Pe	Ct	Cp	PWR	Torque	Thrust
(mph)	(Adv Ratio)				(Hp)	(In-Lbf)	(Lbf)
0.0	0.00	0.0000	0.1333	0.1194	3.351	9.599	8.419
4.7	0.03	0.0332	0.1328	0.1136	3.187	9.131	8.385
9.5	0.06	0.0696	0.1321	0.1076	3.021	8.654	8.341
14.2	0.09	0.1098	0.1312	0.1016	2.852	8.171	8.287
18.9	0.11	0.1544	0.1301	0.0955	2.681	7.680	8.214
23.6	0.14	0.2031	0.1284	0.0896	2.516	7.207	8.111
28.4	0.17	0.2566	0.1264	0.0838	2.352	6.739	7.983
33.1	0.20	0.3135	0.1240	0.0785	2.204	6.313	7.831

37.8	0.23	0.3726	0.1212	0.0737	2.070	5.930
7.651						
42.5	0.26	0.4332	0.1178	0.0694	1.948	5.581
7.441						
47.3	0.28	0.4924	0.1137	0.0655	1.838	5.267
7.183						
52.0	0.31	0.5479	0.1090	0.0621	1.742	4.990
6.885						
56.7	0.34	0.5940	0.1040	0.0595	1.671	4.788
6.565						
61.4	0.37	0.6321	0.0987	0.0575	1.615	4.627
6.232						
66.2	0.40	0.6650	0.0932	0.0556	1.562	4.474
5.887						
70.9	0.43	0.6947	0.0877	0.0537	1.507	4.317
5.538						
75.6	0.45	0.7219	0.0821	0.0516	1.447	4.146
5.182						
80.3	0.48	0.7468	0.0763	0.0493	1.382	3.960
4.819						
85.1	0.51	0.7694	0.0705	0.0467	1.312	3.758
4.450						
89.8	0.54	0.7898	0.0645	0.0440	1.235	3.537
4.073						
94.5	0.57	0.8076	0.0584	0.0410	1.152	3.299
3.690						
99.2	0.60	0.8224	0.0523	0.0379	1.063	3.046
3.305						
104.0	0.62	0.8330	0.0461	0.0345	0.969	2.776
2.911						
108.7	0.65	0.8386	0.0398	0.0310	0.870	2.492
2.517						
113.4	0.68	0.8330	0.0334	0.0273	0.766	2.196
2.111						
118.1	0.71	0.8184	0.0271	0.0234	0.658	1.884
1.709						
122.9	0.74	0.7776	0.0201	0.0190	0.534	1.529
1.267						
127.6	0.77	0.6715	0.0139	0.0158	0.445	1.274
0.878						
132.3	0.79	0.4518	0.0072	0.0126	0.354	1.015
0.453						
137.0	0.82	-0.0188	-0.0002	0.0090	0.252	0.721
-0.013						

PROP RPM = 23000

V	J	Pe	Ct	Cp	PWR	Torque
Thrust						

(mph)	(Adv Ratio)				(Hp) (In-Lbf)	(Lbf)
0.0	0.00	0.0000	0.1350	0.1332	4.272	11.707
9.320	4.9	0.03	0.0301	0.1345	0.1262	4.049
11.095	9.286	9.8	0.06	0.0634	0.1339	0.1192
3.823	10.477	9.244	14.8	0.08	0.1007	0.1331
0.1120	3.593	9.845	9.190	19.7	0.11	0.1424
0.1322	0.1048	3.360	9.207	9.122	24.6	0.14
0.1894	0.1309	0.0975	3.128	8.570	9.033	29.5
0.17	0.2412	0.1289	0.0905	2.903	7.955	8.896
34.4	0.20	0.2987	0.1264	0.0836	2.683	7.351
8.727	39.4	0.23	0.3590	0.1235	0.0777	2.492
6.830	8.527	44.3	0.25	0.4211	0.1202	0.0725
2.325	6.371	8.294	49.2	0.28	0.4825	0.1160
0.0679	2.177	5.966	8.008	54.1	0.31	0.5406
0.1112	0.0639	2.048	5.612	7.673	59.0	0.34
0.5893	0.1059	0.0609	1.953	5.351	7.311	63.9
0.37	0.6284	0.1005	0.0587	1.883	5.160	6.939
68.9	0.40	0.6619	0.0950	0.0567	1.820	4.986
6.559	73.8	0.42	0.6919	0.0894	0.0547	1.755
4.810	6.173	78.7	0.45	0.7194	0.0837	0.0526
1.686	4.620	5.780	83.6	0.48	0.7446	0.0780
0.0503	1.612	4.418	5.383	88.5	0.51	0.7677
0.0721	0.0477	1.531	4.194	4.976	93.5	0.54
0.7887	0.0660	0.0449	1.440	3.947	4.558	98.4
0.56	0.8070	0.0599	0.0419	1.343	3.681	4.132
103.3	0.59	0.8212	0.0536	0.0387	1.240	3.399
3.698	108.2	0.62	0.8310	0.0472	0.0353	1.132
3.102	3.260	113.1	0.65	0.8354	0.0408	0.0317
1.018	2.789	2.819	118.1	0.68	0.8281	0.0344
0.0282	0.904	2.476	2.377	123.0	0.71	0.8065
0.0279	0.0244	0.784	2.148	1.928	127.9	0.73
0.7529	0.0212	0.0207	0.664	1.820	1.466	

132.8	0.76	0.6542	0.0136	0.0158	0.507	1.390
0.937						
137.7	0.79	0.4125	0.0063	0.0120	0.385	1.055
0.432						
142.6	0.82	0.0004	0.0000	0.0103	0.331	0.908
0.000						

PROP RPM = 24000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1375	0.1613	5.878	15.435	10.333
5.1	0.03	0.0253	0.1371	0.1516	5.524	14.507	10.301
10.2	0.06	0.0539	0.1365	0.1418	5.168	13.571	10.260
15.3	0.08	0.0865	0.1358	0.1319	4.808	12.626	10.209
20.4	0.11	0.1238	0.1350	0.1221	4.450	11.685	10.146
25.5	0.14	0.1673	0.1340	0.1121	4.086	10.729	10.070
30.5	0.17	0.2182	0.1328	0.1022	3.725	9.781	9.979
35.6	0.20	0.2748	0.1305	0.0931	3.392	8.909	9.809
40.7	0.22	0.3381	0.1274	0.0844	3.076	8.078	9.576
45.8	0.25	0.4035	0.1238	0.0773	2.817	7.399	9.304
50.9	0.28	0.4673	0.1195	0.0716	2.609	6.852	8.981
56.0	0.31	0.5281	0.1143	0.0666	2.428	6.376	8.587
61.1	0.34	0.5816	0.1085	0.0627	2.285	6.001	8.157
66.2	0.36	0.6224	0.1030	0.0603	2.196	5.766	7.744
71.3	0.39	0.6562	0.0975	0.0582	2.122	5.574	7.328
76.4	0.42	0.6863	0.0919	0.0562	2.048	5.379	6.904
81.5	0.45	0.7141	0.0861	0.0540	1.969	5.170	6.473
86.5	0.48	0.7397	0.0803	0.0517	1.883	4.944	6.034

91.6	0.50	0.7633	0.0743	0.0491	1.788	4.696
5.586						
96.7	0.53	0.7847	0.0682	0.0463	1.686	4.427
5.129						
101.8	0.56	0.8026	0.0621	0.0433	1.578	4.145
4.665						
106.9	0.59	0.8162	0.0559	0.0402	1.467	3.851
4.199						
112.0	0.62	0.8255	0.0495	0.0369	1.346	3.535
3.721						
117.1	0.64	0.8288	0.0431	0.0335	1.220	3.203
3.237						
122.2	0.67	0.8209	0.0365	0.0298	1.088	2.856
2.740						
127.3	0.70	0.7967	0.0292	0.0256	0.934	2.452
2.192						
132.4	0.73	0.7340	0.0226	0.0224	0.817	2.145
1.699						
137.5	0.76	0.6381	0.0146	0.0173	0.631	1.656
1.098						
142.5	0.78	0.4101	0.0071	0.0135	0.493	1.294
0.532						
147.6	0.81	-0.0423	-0.0006	0.0108	0.392	1.031
-0.042						

PROP RPM = 25000

V	J	Pe	Ct	Cp	PWR	Torque	Thrust
(mph)	(Adv Ratio)				(Hp)	(In-Lbf)	(Lbf)
0.0	0.00	0.0000	0.1416	0.2341	9.644	24.312	11.548
5.3	0.03	0.0180	0.1411	0.2181	8.984	22.649	11.504
10.6	0.06	0.0389	0.1404	0.2014	8.293	20.907	11.449
15.9	0.08	0.0634	0.1396	0.1843	7.590	19.134	11.381
21.1	0.11	0.0927	0.1385	0.1669	6.872	17.325	11.294
26.4	0.14	0.1279	0.1372	0.1496	6.163	15.537	11.189
31.7	0.17	0.1712	0.1357	0.1327	5.465	13.777	11.064
37.0	0.20	0.2256	0.1339	0.1159	4.775	12.037	10.919
42.3	0.22	0.2942	0.1316	0.0999	4.113	10.369	10.730

47.6	0.25	0.3750	0.1280	0.0857	3.531	8.901	10.437
52.9	0.28	0.4540	0.1231	0.0756	3.116	7.855	10.036
58.1	0.31	0.5193	0.1175	0.0694	2.860	7.209	9.578
63.4	0.33	0.5748	0.1118	0.0651	2.683	6.763	9.117
68.7	0.36	0.6160	0.1062	0.0626	2.576	6.495	8.662
74.0	0.39	0.6502	0.1005	0.0604	2.487	6.271	8.196
79.3	0.42	0.6806	0.0947	0.0582	2.399	6.048	7.722
84.6	0.45	0.7084	0.0888	0.0560	2.305	5.811	7.241
89.9	0.47	0.7340	0.0827	0.0535	2.203	5.553	6.748
95.1	0.50	0.7589	0.0764	0.0506	2.084	5.254	6.233
100.4	0.53	0.7799	0.0701	0.0477	1.964	4.950	5.718
105.7	0.56	0.7968	0.0638	0.0447	1.840	4.639	5.201
111.0	0.59	0.8095	0.0575	0.0416	1.713	4.320	4.686
116.3	0.61	0.8176	0.0510	0.0383	1.578	3.978	4.160
121.6	0.64	0.8184	0.0443	0.0348	1.432	3.610	3.615
126.9	0.67	0.8063	0.0374	0.0310	1.279	3.224	3.048
132.1	0.70	0.7715	0.0302	0.0273	1.126	2.839	2.466
137.4	0.73	0.7173	0.0227	0.0230	0.946	2.386	1.852
142.7	0.75	0.6017	0.0151	0.0189	0.778	1.961	1.230
148.0	0.78	0.3825	0.0072	0.0147	0.607	1.529	0.588
153.3	0.81	0.0278	0.0004	0.0129	0.531	1.339	0.036

PROP RPM = 26000

V	J	Pe	Ct	Cp	PWR	Torque	Thrust
(mph)	(Adv Ratio)				(Hp)	(In-Lbf)	(Lbf)

0.0	0.00	0.0000	0.1386	0.1798	8.332	20.197
12.227						
5.5	0.03	0.0224	0.1383	0.1719	7.962	19.300
12.200						
11.0	0.06	0.0467	0.1380	0.1643	7.613	18.454
12.174						
16.4	0.08	0.0734	0.1376	0.1566	7.253	17.582
12.138						
21.9	0.11	0.1027	0.1371	0.1485	6.882	16.682
12.093						
27.4	0.14	0.1353	0.1365	0.1403	6.499	15.754
12.038						
32.9	0.17	0.1711	0.1358	0.1325	6.140	14.884
11.982						
38.4	0.19	0.2135	0.1348	0.1229	5.696	13.807
11.890						
43.8	0.22	0.2615	0.1337	0.1138	5.270	12.775
11.789						
49.3	0.25	0.3161	0.1322	0.1047	4.853	11.763
11.663						
54.8	0.28	0.3836	0.1290	0.0935	4.332	10.502
11.375						
60.3	0.31	0.4689	0.1241	0.0810	3.751	9.093
10.944						
65.8	0.33	0.5459	0.1181	0.0722	3.345	8.109
10.414						
71.2	0.36	0.5935	0.1117	0.0681	3.154	7.646
9.855						
76.7	0.39	0.6309	0.1054	0.0650	3.014	7.305
9.294						
82.2	0.42	0.6654	0.0989	0.0620	2.873	6.964
8.723						
87.7	0.45	0.6979	0.0921	0.0587	2.722	6.598
8.126						
93.2	0.47	0.7241	0.0859	0.0561	2.600	6.302
7.579						
98.6	0.50	0.7420	0.0801	0.0540	2.503	6.067
7.061						
104.1	0.53	0.7614	0.0736	0.0511	2.367	5.737
6.491						
109.6	0.56	0.7795	0.0671	0.0479	2.220	5.382
5.923						
115.1	0.58	0.7932	0.0604	0.0445	2.062	4.999
5.331						
120.5	0.61	0.8030	0.0533	0.0406	1.883	4.564
4.703						
126.0	0.64	0.8042	0.0463	0.0369	1.708	4.140
4.087						
131.5	0.67	0.7919	0.0391	0.0330	1.529	3.706
3.452						
137.0	0.70	0.7657	0.0319	0.0290	1.344	3.257
2.816						
142.5	0.72	0.7117	0.0241	0.0245	1.134	2.749
2.125						
147.9	0.75	0.5823	0.0158	0.0204	0.944	2.287
1.393						

153.4	0.78	0.3936	0.0077	0.0152	0.703	1.703	0.676
158.9	0.81	-0.0083	-0.0001	0.0126	0.585	1.418	-0.011

PROP RPM = 26999

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1392	0.2046	10.614	24.776	13.242
5.7	0.03	0.0198	0.1389	0.1950	10.115	23.611	13.208
11.4	0.06	0.0417	0.1384	0.1846	9.575	22.351	13.161
17.1	0.08	0.0662	0.1378	0.1738	9.017	21.048	13.103
22.8	0.11	0.0938	0.1370	0.1626	8.438	19.696	13.032
28.5	0.14	0.1254	0.1361	0.1510	7.833	18.284	12.946
34.1	0.17	0.1620	0.1350	0.1391	7.219	16.852	12.846
39.8	0.19	0.2050	0.1338	0.1272	6.598	15.402	12.731
45.5	0.22	0.2563	0.1325	0.1150	5.968	13.930	12.600
51.2	0.25	0.3187	0.1309	0.1029	5.336	12.457	12.453
56.9	0.28	0.3961	0.1290	0.0906	4.701	10.973	12.269
62.6	0.31	0.4645	0.1249	0.0823	4.270	9.968	11.882
68.3	0.33	0.5293	0.1199	0.0756	3.924	9.159	11.405
74.0	0.36	0.5762	0.1147	0.0720	3.735	8.718	10.908
79.7	0.39	0.6133	0.1092	0.0693	3.597	8.395	10.383
85.4	0.42	0.6448	0.1031	0.0667	3.461	8.080	9.804
91.1	0.45	0.6726	0.0968	0.0641	3.324	7.760	9.208
96.7	0.47	0.6976	0.0905	0.0614	3.184	7.433	8.611
102.4	0.50	0.7195	0.0841	0.0585	3.037	7.090	8.001

108.1	0.53	0.7384	0.0778	0.0557	2.891	6.748	7.403
113.8	0.56	0.7545	0.0713	0.0526	2.730	6.371	6.785
119.5	0.58	0.7600	0.0647	0.0497	2.579	6.021	6.151
125.2	0.61	0.7671	0.0572	0.0456	2.367	5.525	5.438
130.9	0.64	0.7644	0.0494	0.0414	2.147	5.012	4.702
136.6	0.67	0.7468	0.0410	0.0367	1.903	4.441	3.901
142.3	0.70	0.7095	0.0331	0.0325	1.685	3.933	3.150
148.0	0.72	0.6411	0.0246	0.0278	1.440	3.362	2.340
153.7	0.75	0.5316	0.0159	0.0225	1.165	2.719	1.512
159.3	0.78	0.3006	0.0071	0.0185	0.961	2.244	0.680
165.0	0.81	-0.0713	-0.0013	0.0144	0.745	1.740	-0.121

PROP RPM = 28000

V	J	Pe	Ct	Cp	PWR	Torque	Thrust
(mph)	(Adv Ratio)				(Hp)	(In-Lbf)	(Lbf)
0.0	0.00	0.0000	0.1346	0.1501	8.685	19.549	13.770
5.9	0.03	0.0257	0.1344	0.1457	8.433	18.983	13.744
11.8	0.06	0.0529	0.1340	0.1411	8.166	18.381	13.708
17.7	0.08	0.0818	0.1336	0.1363	7.889	17.757	13.664
23.6	0.11	0.1128	0.1330	0.1313	7.599	17.105	13.610
29.5	0.14	0.1461	0.1324	0.1262	7.302	16.437	13.545
35.4	0.17	0.1822	0.1316	0.1207	6.986	15.725	13.465
41.3	0.19	0.2211	0.1308	0.1153	6.670	15.014	13.376
47.3	0.22	0.2636	0.1297	0.1097	6.345	14.283	13.271
53.2	0.25	0.3100	0.1286	0.1039	6.014	13.537	13.153

59.1	0.28	0.3824	0.1253	0.0912	5.279	11.883	12.816
65.0	0.31	0.4301	0.1247	0.0888	5.140	11.569	12.760
70.9	0.33	0.5259	0.1163	0.0739	4.275	9.623	11.895
76.8	0.36	0.5713	0.1142	0.0724	4.188	9.427	11.685
82.7	0.39	0.6107	0.1080	0.0690	3.991	8.984	11.052
88.6	0.42	0.6426	0.1018	0.0662	3.829	8.619	10.414
94.5	0.45	0.6703	0.0956	0.0636	3.678	8.279	9.782
100.4	0.47	0.6932	0.0897	0.0612	3.544	7.976	9.174
106.3	0.50	0.7130	0.0836	0.0587	3.399	7.651	8.548
112.2	0.53	0.7293	0.0772	0.0560	3.240	7.293	7.895
118.1	0.56	0.7419	0.0706	0.0530	3.065	6.899	7.218
124.0	0.58	0.7541	0.0633	0.0491	2.842	6.397	6.479
130.0	0.61	0.7557	0.0562	0.0456	2.637	5.935	5.750
135.9	0.64	0.7481	0.0489	0.0419	2.422	5.452	5.002
141.8	0.67	0.7251	0.0414	0.0381	2.206	4.965	4.230
147.7	0.70	0.6844	0.0337	0.0343	1.983	4.464	3.447
153.6	0.72	0.6064	0.0247	0.0295	1.708	3.844	2.529
159.5	0.75	0.5066	0.0174	0.0258	1.493	3.361	1.779
165.4	0.78	0.3110	0.0073	0.0182	1.053	2.371	0.743
171.3	0.81	-0.0715	-0.0012	0.0132	0.764	1.720	-0.120

PROP RPM = 29000

V	J	Pe	Ct	Cp	PWR	Torque	Thrust
(mph)	(Adv Ratio)				(Hp)	(In-Lbf)	(Lbf)
0.0	0.00	0.0000	0.1338	0.1485	9.549	20.752	14.686

6.0	0.03	0.0254	0.1334	0.1440	9.257	20.118
14.639						
12.0	0.05	0.0523	0.1329	0.1392	8.949	19.448
14.583						
18.0	0.08	0.0807	0.1323	0.1348	8.664	18.829
14.521						
24.1	0.11	0.1111	0.1316	0.1298	8.344	18.133
14.446						
30.1	0.14	0.1477	0.1292	0.1197	7.697	16.728
14.173						
36.1	0.16	0.1845	0.1281	0.1141	7.333	15.937
14.055						
42.1	0.19	0.2239	0.1270	0.1088	6.991	15.194
13.941						
48.1	0.22	0.2554	0.1278	0.1096	7.046	15.312
14.025						
54.1	0.25	0.2996	0.1265	0.1040	6.688	14.534
13.880						
60.2	0.27	0.3550	0.1248	0.0963	6.189	13.451
13.697						
66.2	0.30	0.4236	0.1228	0.0873	5.615	12.202
13.478						
72.2	0.33	0.4812	0.1199	0.0818	5.262	11.436
13.155						
78.2	0.36	0.5359	0.1151	0.0765	4.916	10.684
12.633						
84.2	0.38	0.5788	0.1101	0.0729	4.686	10.183
12.077						
90.2	0.41	0.6147	0.1045	0.0698	4.490	9.758
11.471						
96.2	0.44	0.6408	0.0987	0.0675	4.338	9.427
10.830						
102.3	0.47	0.6651	0.0927	0.0649	4.171	9.065
10.174						
108.3	0.49	0.6858	0.0865	0.0622	3.998	8.689
9.497						
114.3	0.52	0.7039	0.0802	0.0592	3.809	8.278
8.797						
120.3	0.55	0.7189	0.0735	0.0560	3.599	7.821
8.065						
126.3	0.57	0.7284	0.0663	0.0523	3.363	7.310
7.273						
132.3	0.60	0.7316	0.0587	0.0483	3.108	6.754
6.443						
138.3	0.63	0.7277	0.0510	0.0441	2.835	6.160
5.591						
144.4	0.66	0.7094	0.0430	0.0399	2.562	5.568
4.721						
150.4	0.68	0.6735	0.0348	0.0354	2.275	4.944
3.821						
156.4	0.71	0.6114	0.0264	0.0308	1.977	4.297
2.899						
162.4	0.74	0.5124	0.0177	0.0255	1.642	3.569
1.943						
168.4	0.77	0.3257	0.0089	0.0209	1.341	2.914
0.972						

174.4 0.79 0.0255 0.0005 0.0169 1.087
2.361 0.060