DEPARTEMENT of ELECTRICAL ENGINEERING

FACULTY of ENGINEERING AND DESIGN

EE10170

Robotics and Mechatronic Systems

CONTROL CHALLENGE REPORT

Alexandre BENOIT 199209819

18/04/2020

UNIVERSITY OF BATH

Abstract

This report illustrates the theory, results and the manufacturing of a specific control system. It put forward three tasks in order to achieve the control challenge:

- It shows the functioning of an MPU-6050
- It shows the control of a DC motor by controlling its speed and rotation
- It shows the control of a propeller which stays at a given position and levitates

Contents

Table of Figures

I- Introduction

This report will cover a range of knowledge to carry the control challenge. A complete description of the hardware, software and control will be detailed. The purpose of this control challenge is to manipulate a DC motor in levitation, where a propeller is attached, by giving it an angle to reach by itself. It is required to know how to read an angle value in the X, Y and Z axis thanks to the MPU6050. It is an IMU sensor mostly used for self-balancing robots, smartphones and more. Control the DC motor by rotating it clockwise and anticlockwise is one of the basic control movement we need to understand. The program will be written on the Arduino software and run it on the Arduino nano.

II- Hardware description

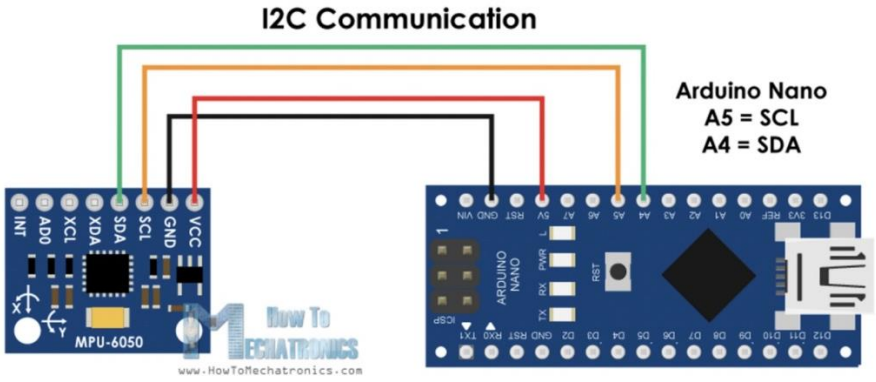

Figure 1: MPU-6050 and Arduino Nano wiring diagram

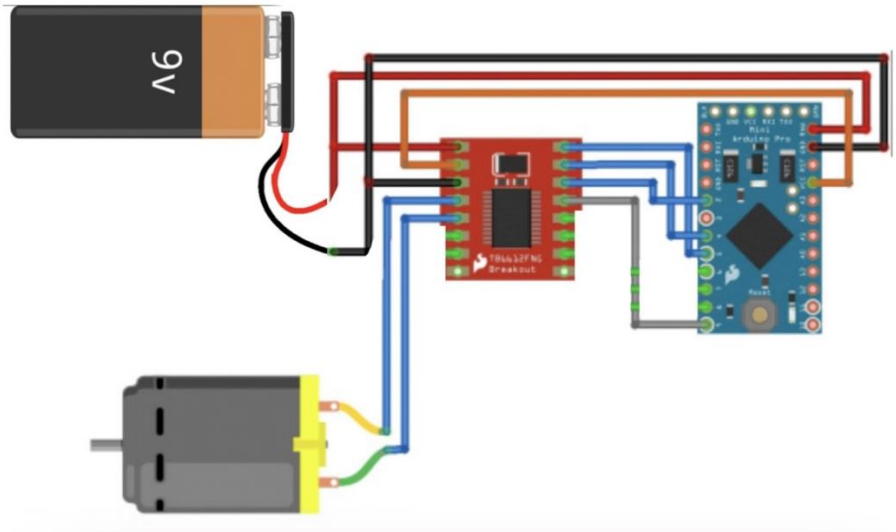

Figure 2: Motor driver and DC motor wiring diagram

1. Wiring explanation

Looking at figure 1, on the left side of the Motor Driver, the DC motor is attached to the pin A01 and A02, Vt is connected with Vin on the Arduino, the VCC to the 5V input and the Ground to the Ground. Because the current delivered by the Arduino is not enough to power the motor correctly, a 9V battery have been added and linked to the VCC and the ground. On the right side of the motor driver, the standby pin is joined to D9, AI1 and AI2 to D4 and D2 and PWMA to D5 of the Arduino.

Regarding figure 2, the SCL and SDA are connected to A4 and A5 of the Arduino. The ground and VCC are joined to the same pin as the motor driver.

The potentiometer, which will be used in exercise 2, has three outputs where the far-left pin is linked to the 5V input, the middle is connected to the A0 and the far-right is attached to the ground.

2. MPU-6050 sensor

MPU6050, ADXL345 Accelerometer sensor. The GY- 521 is an accelerometer, a gyroscope and a thermometer. The device provides two pins, SCL and SDA. SCL is the serial clock. The clock signal will synchronize the data transfer between the devices on the I2C bus and a master device generates it. The SDA is what carries out the data.

The MPU 6050 is a six-axis IMU sensor and it will display six values: three values from the accelerometer and three from the gyroscope. The MPU 6050 is a MEMS (microelectromechanical systems) technology. The accelerometer and the gyroscope are located inside a single chip. The accelerometer is working on the principle of the piezoelectric process. The box where the ball is situated will move forced to gravity and touched one of the three-axis walls made of piezoelectric crystals and will create current. From there, we can determine the direction of inclination and its magnitude. The proof mass will experience a force when accelerated. If this mass is a semi-movable, suspended material, of the proof mass, on a fixed substrate that can be deflected you a force and this deflection can be measured to determine the acceleration. Furthermore, the gyroscope works on the principle of Coriolis acceleration where the motion and inclination of the fork creates a current.

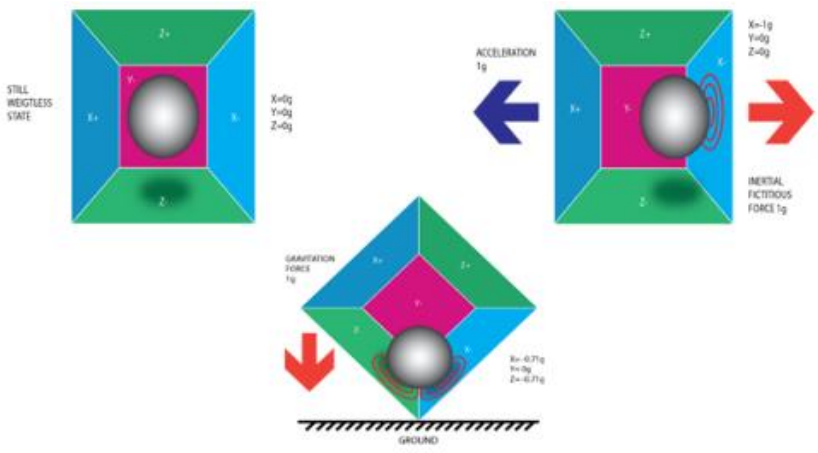

Figure 3: Accelerometer process

3. Spark Fun Motor Driver - Dual TB6612FNG

Spark Fun Motor Driver - Dual TB6612FNG: it has a supply range of **2.5V to 13.5V** and is capable of **1.2A continuous current** and **3.2A peak current.** [1] In this experience, it is powered by a 9V battery.

PIN LABEL	FUNCTION	POWER/ INPUT/OUTPUT	CHARACTERISTICS
VM	Motor voltage	power	Provide power for the motor
VCC	Logic Voltage	Power	This is the voltage to power the chip and talks to the microcontroller
GND	Ground	Power	Need to be linked to another ground
STBY	Stand by	Input	Allows the H-bridge to work when high
AIN1/2	Input 1 and 2 for channel A	Input	Determine direction

PIN LABEL	FUNCTION	POWER/ INPUT/OUTPUT	CHARACTERISTICS
PWMA	PWM (pulse width modulation) input for channel A	Input	Control the speed of the motor. It is the speed you are going to give to the motor.
A01/02	Output 1 and 2 for channel A	Output	Connect to the motor

Figure 4:Data table for the motor driver's settings

What is an H-bridge (figure 5) ? It is a simple circuit that allows us to control a DC motor backward or forward. It changes the polarity of a voltage applied to a load. The circuit is using MOFSETs. They are transistors based on a small voltage and fix to their gate. They can disconnect and connect which make the MOFSETs a switch. [9]

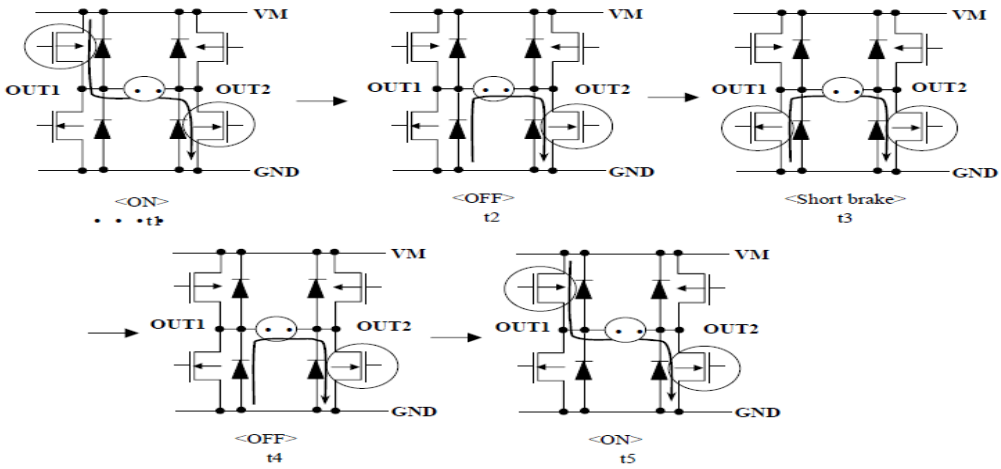

Figure 5: H-SW Operating Description

4. The physics behind the propeller

Propeller with two blades. The thrust generated by a propeller is hard to explain. We need several abstract pieces of knowledge to underline this explication. With the equation $\boldsymbol{Thrust=F=A\Delta p}$, $\boldsymbol{\Delta p= pt_e\text{-}pt_0}$, $\boldsymbol{\Delta p= .5 *}$

r * [Ve ^2 - V0 ^2], F = .5 * r * A * [Ve ^2 - V0 ^2], F = [m dot * V]e - [m dot * V]0 and **Vp = .5 [Ve + V0]** and the chart (figure 6) [6-7] , the propeller converted engine horsepower into thrust by accelerating air and creating a low-pressure differential in front of the propeller. Since air naturally moves from high to low-pressure, when your propeller is spinning, you are being pulled forward. [8] It is important to underline that the special shape of the propeller will influence down or upthrust.

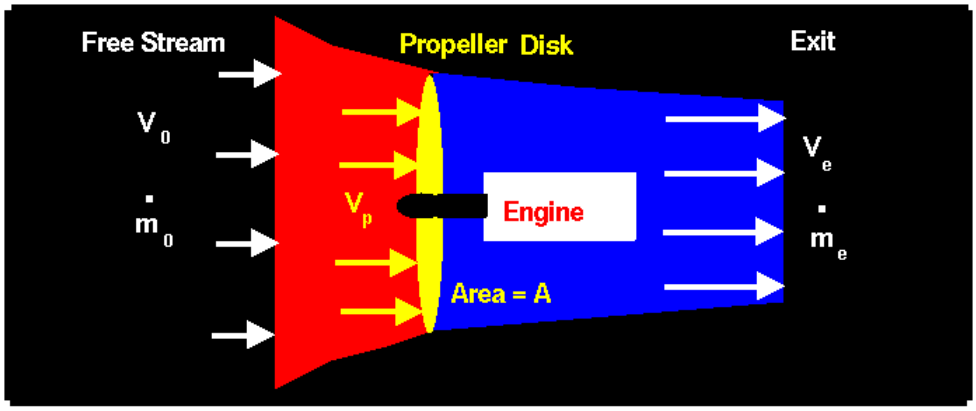

Figure 6: Propeller thrust

5. Rotary Potentiometer

Rotary Potentiometer with three outputs: it varies their resistive values while rotating the knob. When turning the knob, the internal wiper sweeps around the resistive track. The position of the wiper then provides an appropriate output signal (pin 2), which will vary between the voltage level applied to one end of the resistive track (pin 1) and that at the other (pin 3). [14]

The potentiometer is a three-wire resistive device where the continuous variable voltage output signal is proportional to the physical position of the wiper along the track. For example, 0° will be 0V and 0rpm for a motor and 270° will be 5V and 255 or -255rpm.

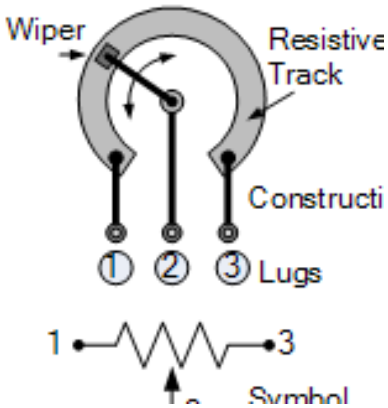

Figure 7: Inside a potentiometer

6. Arduino NANO

Arduino NANO is a breadboard-friendly board based on the ATmega328P. It will make the connection between the computer program and the components you wish to link the Arduino.

III- Software and control description

1. Reading the angle

To run the MPU6050, a library called **wire.h** is used for I2C communication. It is a popular electronic device which is required to ensure a connection between a master and multiple slave device. The ADX345 has a unique device address and addition internal registers addresses for the X, Y and Z-axis.

In the code, *Wire.begin* initiates the Wire library and the *beginTransmission (*MPU_addr) begins the transmission to the I2C slave, which is the accelerometer sensor. *Wire.write()* function is employed to ask the data we need from the two registers for the X-axis. The *wire. read's* function will read the bytes from two registers of the X-axis. To calculate the angle, the formula in figure 8.

```
Angle_X= RAD_TO_DEG * (atan2(-yAng, -zAng)+PI);
Angle_Y= RAD_TO_DEG * (atan2(-xAng, -zAng)+PI);
Angle_Z= RAD_TO_DEG * (atan2(-yAng, -xAng)+PI);
```

Figure 8: Angle calculation

The *atan2()'s* function computes the principal value of the arc tangent of y and x, using the signs of both arguments to determine the quadrant of the return value. The returned value is in the range [-pi, +pi] radians. Then all the angle values are displayed on the serial monitor. It is possible if the function *Serial.print* is used

to display the text and *serial.println* to demonstrate the value of the angle. The angle displayed, has been previously calculated in the program. Here are some values while moving the MPU (figure 9).

```
AngleX= 30.40
AngleY= 350.62
AngleZ= 105.73
-----------------------------------------
AngleX= 316.86
AngleY= 340.81
AngleZ= 249.62
-----------------------------------------
AngleX= 285.83
AngleY= 309.88
AngleZ= 251.26
-----------------------------------------
AngleX= 19.78
AngleY= 338.21
AngleZ= 138.04
-----------------------------------------
AngleX= 3.01
AngleY= 30.73
AngleZ= 5.06
-----------------------------------------
AngleX= 358.70
AngleY= 325.93
AngleZ= 181.92
-----------------------------------------
AngleX= 37.49
AngleY= 38.20
AngleZ= 44.27
-----------------------------------------
AngleX= 337.48
AngleY= 331.24
AngleZ= 217.07
-----------------------------------------
AngleX= 33.70
AngleY= 3.54
AngleZ= 84.70
```

Figure 9: Random angle values for x, y and z-axis while moving the MPU-6050

2. Controlling the motor

To run the motor, the code must know on which pins are located the direction and speed control. The code asks a user to enter his preferred direction (clockwise or anticlockwise) and control the speed with a potentiometer. To ask the user to enter a text, the function *char* is used where the variable *ch* is incremented to the value the user is going to enter. To read the potentiometer value, we attributed it to the pin A0. The function *analogread(A0)* reads the position of the potentiometer. As previously explained, the position will give a certain speed to the motor. The potentiometer's value is assigned to the *sensor_value*. The latter gives its value to the function *MotorSpeed1* which is the speed of the motor. Then as we configured pins as an output with *pinMode*, the function *digitalwrite* has two possibilities, HIGH or LOW. By referring to the table of how to control the channels, we can set a HIGH or LOW value for AI1/2 and STBY (figure 7). Of course, the three channels that interest us are CCW, CW and STOP. With this, it is essential to think about the delay we are using. The delay used is 1 ms in order to limit speed jumps and motor bug while changing potentiometer's values.

In1	In2	PWM	Out1	Out2	Mode
H	H	H/L	L	L	Short brake
L	H	H	L	H	CCW
L	H	L	L	L	Short brake
H	L	H	H	L	CW
H	L	L	L	L	Short brake
L	L	H	OFF	OFF	Stop

Figure 10: Channels control motor

3. Control challenge

For this final exercise, a PID control process enable to create (figure 11) a levitation self-control by a chosen angle.

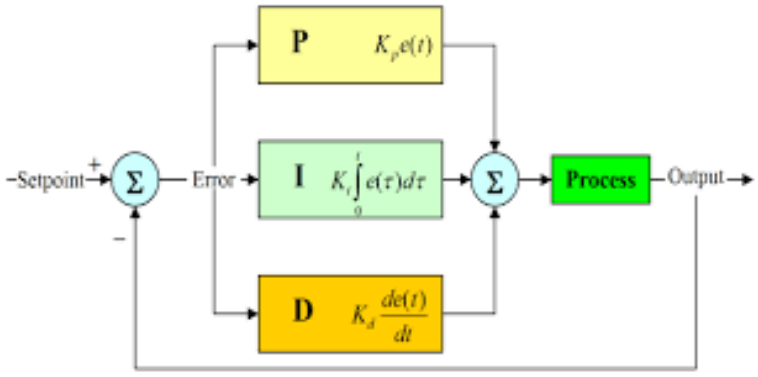

Figure 11: PID control block diagram

From the previous exercises, angle and motor code have not changed a lot. In this code, the main focus is on PID formulas, the error and the total angle.

Firstly, new variables are introduced as the elapsed time, time and previous time which will help us calculate the total angle. In the void setup, the function *millis* returns the number of milliseconds passed since the Arduino board began running the current program. To calculate the *Accangle*, Euler's formula is really helpful. The atan function is used to calculate the arctangent. To get the degree angle for X and Y (respectively total_angle[0] and [1]), the gyro_angle has to be calculated. It is obtained by dividing the reading angle by 131 (which is a value a data-sheet give us). Then the total angle is obtained by this formula TotAngle[0] = 0.98 * (TotAngle[0] + GyroAngle[0] * elapsedTime) + 0.02 * AccAngle[0]. This formula is a filter that will allow us to have a precise angle. As we know, the purpose of this exercise is to manage the error which is the key to process the PID control. This error will be the subtraction of the demand angle and

the measured angle. The value of the error will be multiplied or divided by several constant named Kp, Kd and Ki. Fix these values is essential as they are unique for each device (test it, see and change your device). Then 3 constants have to be set: proportional, integral and derivative. PID is associated to the motor speed value calculated by this mathematical formula: [11] [13]

$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(t')\,dt' + K_\mathrm{d} \frac{de(t)}{dt},$$

Figure 12: PID control mathematical formula

IV- Performance analysis

1. Sensor noise

One of the significant issues we had was the noise produced by the motor and vibration. It affected the MPU which gave strange values for the angles. To avoid this problem, we can think about isolating vibration with some foam. We can use mathematical formulas that will play the role of a high pass or low pass filter or merely adding an electrical component like a capacitor which will use the part of the mathematical formula. Noise interferes with electronic devices. One of the best filters we can use is the Kalman filter, which is applied in many situations, but it is tough to understand, which is why we used a complementary filter. It is a high-pass filter for the gyroscope and a low pass filter for the accelerometer

2. Battery issue

Another issue that appeared during this experience was the discharge of my 9V battery. Since we ran a few times this experience, it lost energy. Here are two problems: it will cost you a bit of money and it affects your result, especially when you are trying to fix these PID constants.

3. Delay

Also, the code we made was sometimes crashing and I was not sure my code was working for my demand angle. I decided to put a long delay which will show if my system is working and respecting my expectations. During this time, the motor will lift up and reach a steady levitation stabilization. The stabilisation will be allowed thanks to the closed-loop PID control system, as shown on the graph figure 12).[12]

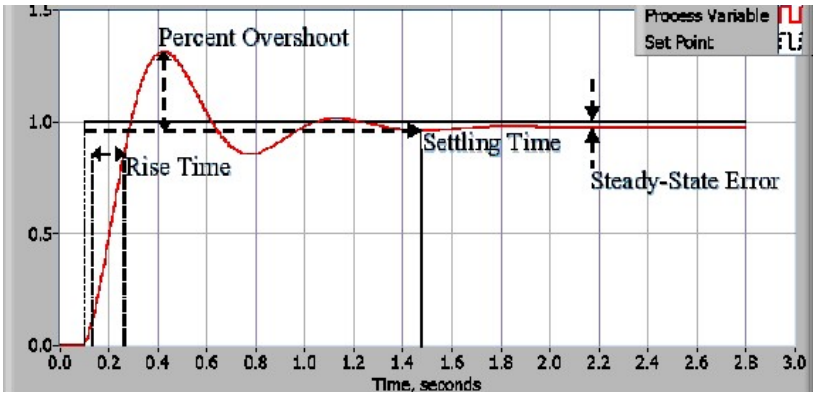

Figure 13: Response of a typical PID closed-loop system

4. Make the structure

What was interesting in this project was the originality that we could show to illustrate this control challenge. To lift the propeller positioned on the arm, we built a wooded structure with a base and two adjacent blocks (figure 13).

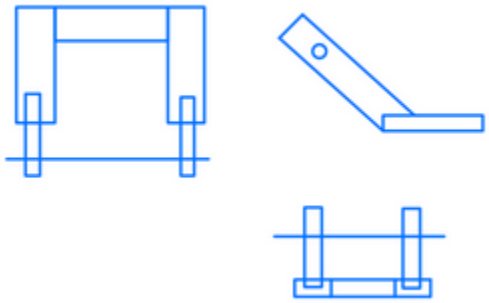

Figure 14: Wood structure's sketch

It will allow us to tape the base on the edge of a table and leave the arm suspended and hold by an axis. The axis goes through holes drilled in the wood and the plastic arm. The problem I met was the fragility of the wood and the plastic. While drilling the wood, I broke several pieces as I did not find the right position for the hole. Make sure the pressure applied would not break the wood, drill the hole in the middle of the wood.

Regarding the plastic arm, as the pressure applied by a drilling machine might break the structure, we took a nail measuring the axle diameter and burned the end of it. After that, we applied the end of the nail and it will be melted and drilled a hole thanks to the heat (it is a bit dodgy, but we get by as best you can). If you worry friction between the axel and the plastic arm will affect the lift of the wing, it doesn't matter as you will fix yourself the speed of the motor to reach your expectations.

5. Precaution

Please be careful when building your circuit. Do not link wires while the current aliments the breadboard or any component. I had made this mistake at the beginning and burnt everything. It was a beginner mistake and cost me quite a lot of money. We learn from our mistakes.

6. Performance description

After following the step of this report, you should run the program and see your propeller going at the right angle of attack. As the exercise does not ask us to be 100% precise, it would be great to add a range for your desired angle. If your desired angle is 90°, leave the propeller a margin of +-2 °.

7. Other solutions

As the PID control has some defaults due to long time delays, we could use the PSD controller. PSD controller supplies what PID alone lacks, that is, zero steady-state error (stability and large overshoots).

V- Conclusion

This project is a good overview of a simple task where a motor needs to control by itself its speed by giving it an angle he needs to reach. We have here all the tools to undertake this project. We know how to deal with the hardware, software, control and performance. We talked about some issues we must avoid and how to sort out our problem eventually.

Above all, this project is useful as we understood a useful concept: the PID control system which is essential for the stability of drones for example. I realised the importance of mastering this concept while I saw this video https://www.youtube.com/watch?v=w2itwFJCgFQ. We can see how difficult and precise it has to be to fulfil this kind of experience. This project encouraged my desire to know more in the drone universe.

VI- References

[1]-https://cdn.sparkfun.com/assets/learn_tutorials/5/6/2/TB6612FNG_bb_1.jpg

[2]-https://howtomechatronics.com/tutorials/arduino/how-to-track-orientation-with-arduino-and-adxl345-accelerometer/

[3]-https://howtomechatronics.com/tutorials/arduino/how-i2c-communication-works-and-how-to-use-it-with-arduino/

[4]-https://maker.pro/arduino/tutorial/how-to-interface-arduino-and-the-mpu-6050-sensor

[5]-https://www.electronics-tutorials.ws/resistor/potentiometer.html

[6]-https://www.grc.nasa.gov/www/k-12/airplane/propanl.html

[7]-https://www.grc.nasa.gov/www/k-12/airplane/propth.html

[8]-https://www.boldmethod.com/learn-to-fly/systems/how-a-propeller-works-and-generates-thrust-as-it-turns/

[9]-https://www.build-electronic-circuits.com/h-bridge/

[10]- PSD Based Levitation Control of Copter Motor Arm, Muhammad Ameen Bhatti1, Awais Ahmed Khoso1, Niaz Muhammad Nohri1, Nawaz Shareef Dahri1, 1 Department of Electrical Engineering, Mehran University Engineering and Technology Jamshoro,Pakistan

[11]- http://www.electronoobs.com/eng_robotica_tut6_1.php

[12]- https://www.ni.com/fr-fr/innovations/white-papers/06/pid-theory-explained.html

[13]- https://en.wikipedia.org/wiki/PID_controller

[14]- https://learn.thestempedia.com/courses/introductory-course-on-arduino/lessons/conditional-programming-in-arduino-ide/topic/analog-input-how-potentiometer-works/