**MARMARA UNIVERSITY**

**FACULTY OF ENGINEERING**



**Modelling of Propulsion Unit Test Platform for Cubesats**

Yunus YAMAK

Hüseyin MENTEŞE

**GRADUATION PROJECT REPORT**

Department of Electrical and Electronics Engineering

**Supervisor**

Asst. Prof. Salih BAYAR

ISTANBUL, 2023

**MARMARA UNIVERSITY**

**FACULTY OF ENGINEERING**

**Modelling of Propulsion Unit Test Platform for Cubesats**

**by**

**Yunus YAMAK**

**Hüseyin MENTEŞE**

**January 20, 2023, Istanbul**

**SUMBITTED TO THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING IN**

**PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE**

**OF**

**BACHELOR OF SCIENCE**

**AT**

**MARMARA UNIVERSITY**

**The author(s) hereby grant(s) to Marmara University permission to reproduce and to distribute publicly paper and electronic copies of this document in whole or in part and declare that the prepared document does not in any way include copying of previous work on the subject or the use of ideas, concepts, words, or structures regarding the subject without appropriate acknowledgement of the source material.**

Signature of Author(s) …………………………………………………………………………

Department of Electrical and Electronics Engineering

Certified By …………………………………………………………………………

Project Supervisor, Department of Electrical and Electronics Engineering

Accepted By …………………………………………………………………………

Head of the Department of Electrical and Electronics Engineering

# ACKNOWLEDGEMENTS

Our advisor Asst. prof. We would like to express our thanks to Salih Bayar. Also, we would like to thank our families for their unwavering support throughout this project process.

**January, 2023**

**Yunus YAMAK, Hüseyin MENTEŞE**

# TABLE OF CONTENTS

[ACKNOWLEDGEMENTS ii](#_Toc125408047)

[TABLE OF CONTENTS iii](#_Toc125408048)

[ABSTRACT iv](#_Toc125408049)

[LIST OF SYMBOLS v](#_Toc125408050)

[ABBREVIATIONS vi](#_Toc125408051)

[LIST OF FIGURES vii](#_Toc125408052)

[LIST OF TABLES x](#_Toc125408053)

[1. INTRODUCTION 1](#_Toc125408054)

[1.1. Thesis Content 2](#_Toc125408055)

[2. RESEARCH OBJECTIVE 3](#_Toc125408056)

[3. RELATED LITERATURE 4](#_Toc125408057)

[4. DESIGN 7](#_Toc125408058)

[4.1. Realistic constraints and conditions 7](#_Toc125408059)

[4.2. Cost of the design 7](#_Toc125408060)

[4.3. Engineering Standards 7](#_Toc125408061)

[4.4. Details of the design 8](#_Toc125408062)

[5. METHODS 10](#_Toc125408063)

[6. RESULTS AND DISCUSSION 35](#_Toc125408064)

[7. CONCLUSION 40](#_Toc125408065)

[REFERENCES 41](#_Toc125408066)

[APPENDICES 42](#_Toc125408067)

[Appendix A 42](#_Toc125408068)

# ABSTRACT

Our aim at the beginning of this project was to make the test unit of the controls of the combustion mechanisms in the propulsion units of CubeSat satellites. Due to time constraints, this is an article about how to control the combustion unit with simulations and gives direction. It is about the simulation of control mechanisms on Matlab and Simulink of parts such as suction tank, pressure sensor, ball valves used in the control mechanism of these units. In addition to these, it includes parts that will be used in the software part and control these parts with microcontrollers. Although these two parts may seem different from each other, they are interrelated things. In the Matlab and Simulink section, it is shown how the control mechanism works and how it checks itself according to the feedbacks. In the STM32CubeIDE and Proteus sections, it provides information about how the valves and pressure sensors are controlled by software.

# LIST OF SYMBOLS

**:** Nominal Flow Rate

**:** Nominal Pressure Drop

**:** Fluid Density

**:** Flow Coefficient

**A:** The Corresponding Flow Area

**:** Orifice Opening

**:** Maximum Orifice Opening

**:** Maximum Orifice Area

**:** Leakage Orifice Area

**:** Piston Area A

**:** Piston Area B

# ABBREVIATIONS

**LCD:** Liquid-Crystal Display

**PS:** Physical Signal

# LIST OF FIGURES

Figure 1: Choice of card type on the Proteus………………………………………………………………………17

Figure 2: Implementation of circuit with STM32F103C6……………………………………………………..17

Figure 3: Selection of card on the STM32CubeIDe……………………………………………………………….18

Figure 4: Pinout view of chosen card……………………………………………………………………………….….18

Figure 5: Enabling of .hex extension compiling……………………………………………………………………19

Figure 6: Inside of main.c at the STM32CubeIDE…………………………………………………………………19

Figure 7: Selection of .hex file for Proteus…………………………………………………………………………..20

Figure 8: Inside of selection file of .hex file for Proteus……………………………………………………….20

Figure 9: On LED………………………………………………………………………………………………………………....21

Figure 10: Off LED……………………………………………………………………………………………………………….21

Figure 11: Generic Simspace Model…………………………………………………………………………………….22

Figure 12: Library: SimscapeFluids\_lib…………………………………………………………………………………23

Figure 13: Library: SimscapeFluids\_lib/Hydraulics (Isothermal)……………………………………….….23

Figure 14: Hydraulic Fluid Block…………………………………………………………………………………………..23

Figure 15: Library Browser……………………………………………………………………………………………….….24

Figure 16: Simulink Library Browser…………………………………………………………………………………….24

Figure 17: Hydraulic Constant Pressure Source……………………………………………………………………25

Figure 18: Hydraulic Reference…………………………………………………………………………………………...25

Figure 19: Connection of Solver Configuration, Hydraulic Fluid, Hydraulic Constant Pressure Source and Hydraulic Reference Blocks……………………………………………………………………………….25

Figure 20: Double-Acting Hydraulic Cylinder (Simple)………………………………………………………….26

Figure 21: 4-Way Directional Valve……………………………………………………………………………………..26

Figure 22: Translational Reference Block…………………………………………………………………………….26

Figure 23: Connection of Double-Acting Hydraulic Cylinder and Translational Reference blocks………………………………………………………………………………………………………………………………….26

Figure 24: Connection of Double-Acting Hydraulic Cylinder and 4-way Directional Valve blocks………………………………………………………………………………………………………………………………….27

Figure 25: Mass Block………………………………………………………………………………………………………….27

Figure 26: Connection Between Mass Block and The Model in Figure 24…………………………….27

Figure 27: Ideal Translational Motion Sensor Block………………………………………………………….….28

Figure 28: Next Version of The Model in Figure 26……………………………………………………………...28

Figure 29: Connection with Ideal Translational Motion Sensor, PS-Simulink Converter and Scope Blocks……………………………………………………………………………………………………………………….29

Figure 30: Valve Actuator Block……………………………………………………………………………………….….29

Figure 31: Connection of Valve Actuator Block and S terminal of 4-Way Directional Valve Block……………………………………………………………………………………………………………………………………29

Figure 32: Connection of 4-Way Directional Valve Block and the Hydraulic Constant Pressure Source block and the Hydraulic Reference block…………………………………………………………….…..30

Figure 33: Hydraulic Pressure Sensor Block………………………………………………………………………...30

Figure 34: Version of the Model After Adding 2 Hydraulic Sensor Blocks…………………………….31

Figure 35: Connection of Hydraulic Pressure Sensor with PS-Simulink Converter and Scope.31

Figure 36: Step Block……………………………………………………………………………………………………….….32

Figure 37: Constant Block……………………………………………………………………………………………………32

Figure 38: Add Block……………………………………………………………………………………………………………32

Figure 39: Signal İnput Structure…………………………………………………………………………………………32

Figure 40: Hydraulic Pipeline Block………………………………………………………………………………….….33

Figure 41: Connection of Hydraulic Pipeline Block…………………………………………………………….…33

Figure 42: First Step Signal Parameters……………………………………………………………………………….33

Figure 43: Second Step Signal Parameters…………………………………………………………………………..34

Figure 44: Third Step Signal Parameters………………………………………………………………………………34

Figure 45: Fourth Step Signal Parameters…………………………………………………………………………...34

Figure 46: Valve Actuator Parameters………………………………………………………………………………...35

Figure 46: Valve Actuator Parameters…………………………………………………………………………….…..35

Figure 47: Hydraulic Constant Pressure Source Parameters………………………………………………..35

Figure 48: 4-Way Directional Valve Block Dialog Box of Part1……………………………………………..37

Figure 49: 4-Way Directional Valve Block Dialog Box of Part2……………………………………………..38

Figure 50: The Hydraulic Pipeline Dialog Box……………………………………………………………………...39

Figure 51: Ideal Translational Motion Sensor Dialog Box…………………………………………………….39

Figure 52: Part1 of The Model…………………………………………………………………………………………....40

Figure 53: Part2 of The Model…………………………………………………………………………………………....40

Figure 54: Input Signal of Valve Actuator Block……………………………………………………………………42

Figure 55: Scope of P4 (Part 1)…………………………………………………………………………………………….42

Figure 56: Scope of P5 (Part 1)…………………………………………………………………………………………….43

Figure 57: Scope of P1 (Part 2)…………………………………………………………………………………………….43

Figure 58: Scope of P2 (Part 2)…………………………………………………………………………………………….44

Figure 59: Scope of P3(Hydraulic Constant Pressure Source)……………………………………………….44

Figure 60: Scope(y4) of Position (part1)………………………………………………………………………………45

Figure 61: Scope (y) of Position (part2)…………………………………………………….............................45

Figure 62: Schematic flow diagram of a liquid fuel rocket engine with a gas pressure feed system and the inputs (red numbers) and outputs (blue numbers) of the microprocessor controlling this system………………………………………………………………………………………..………………15

# LIST OF TABLES

Table 1 Pressure-fed vs pump fed……………………………………………………………………………………….12

Table 2 Blocks and Sub-libraries of Hydraulic Constant Pressure Source and Hydraulic Reference……………………………………………………………………………………………………………………………24

Table 3 Blocks and Sub-libraries of Double-Acting Hydraulic Cylinder (Simple) and 4-Way Directional Valve…………………………………………………………………………………………………………………25

Table 4 Blocks and Sub-libraries of Mass……………………………………………………………………………..27

Table 5 Blocks and Sub-libraries of Ideal Translational Motion Sensor………………………………..28

Table 6 Blocks and Sub-libraries of Hydraulic Pressure Sensor…………………………………………….30

Table 7 Blocks and Sub-libraries of Input Signals………………………………………………………………...31

Table 8 Blocks and Sub-libraries of Hydraulic Pipeline………..…….…………………………………………32

# INTRODUCTION

Throughout human history, space and the sky have always managed to keep themselves mysteriously secret. This secret was gradually searched out with the voyage of humanity to space. The rockets we use or the satellites we send to space are of great importance for us. According to the information received from the satellites sent to space, these events arouse much more curiosity. Mankind has continued to work without slowing down to satisfy this curiosity and has come a long way until today.

The biggest failures of these satellites and vehicles sent to space usually occur in the parts called launch, which is the moment of departure. In order to prevent these errors, the actions to be taken should be error-free and should be carried out sequentially. Today, the number of these mistakes is reduced to almost zero, and great successes are achieved. In fact, these achievements have reached such extremes that they convince people to go on a vacation to space.

It is the part that we call the propulsion unit, which is one of the most important parts of the launch units. This part is the unit that ensures that the fuel is burned properly and at the desired level, and that it is at spray levels that can carry the rocket or satellite into space. It is vital that the fuel used at the time of launch leaves the fuel tank and reaches the combustion chamber, which is the combustion area. Two different methods are used to push the fuel used to the combustion area, one of which is the method we call pump-fed. The other method is the method we call pressure-fed, in which the fuel is pushed to the combustion area by means of pressure tanks. Different control mechanisms are used to ensure that this pressure tank works at the required times and within the required time. Before realizing these control mechanisms, they need to be modeled and tested. We have tried to implement this control mechanism on Matlab. In addition to the control mechanism, we have established in Matlab, we have tried to simulate how valves and pressure sensors to be used in a real test unit can be controlled with a microcontroller.

## Thesis Content

The subject we want to explain and try to explain in this thesis is to try to model the control mechanism of the propulsion units of CubeSat satellites. In this modeling process, we have modeled the entire system on Matlab and tried to talk about how the valves and sensors used in a real test unit can be controlled with a microprocessor. We have tried to use the STM32CubeIDE program and Proteus applications together so that the valves we use can be simulated while using a microprocessor. While simulating the system in Matlab, we have used the control blocks in Simulink. In the other part, the microprocessor part, we will explain that STM32CubeIDE and Proteus application can work together and try to give examples. In these examples, we consider that the codes that valves and sensors receive via STM32CubeIDE are run and simulated on Proteus with a microcontroller.

# RESEARCH OBJECTIVE

Our aim in this project is the modeling of the propulsion unit, which is of great importance in the launch phase, used in CubeSat satellites and rockets. This modeling is an example of the actual use of valves, sensors, and pressure tanks to simulate and use these materials with remote and self-operating microcontrollers, and to simulate them. While this modeling is being done, it should be designed in the most appropriate way for the purpose and cost. There are different methods for minimizing the launch costs of small satellites such as CubeSat. If we consider this minimization process for the propulsion unit, it is to ensure that different methods such as pump-fed and pressure-fed are more suitable and less costly. In this regard, the use of pressure-fed, which is more logical from our point of view, is more advantageous than pressure-fed pump-fed. It is to make the control mechanism of the propulsion unit of the CubeSat satellites by using pressure-fed and to ensure that it performs its operations at the desired level. It is to show the use of the materials used in a real experimental unit with a microcontroller and to create a simulation that will help them to be used[1].

# RELATED LITERATURE

The goal of engine design is to minimize bulk, volume, and cost while optimizing performance and dependability. There are several design philosophies for previous, present, and upcoming engine designs because of how these parameters are connected. The application determines the motor specifications and, hence, the significance of the aforementioned factors. It is unclear if optimizations are frequently required to satisfy competing needs. A definite transition from straightforward pressure-fed engine cycles to intricate turbo-pump-fed engine cycles may be seen by looking at the past, present, and next motorcycle cycles suggested in the literature. Although pressure supply systems are still extensively utilized today, turbo pump supply systems are claimed to be more effective, lighter, and typically more compact. This prompts the query of when pressure supply systems should be favoured over turbopump systems. Classical gas generators and progressive burning cycles have evolved into novel turbopump delivery cycles like expansion cycles. The expander loop is a potential technological advancement that has undergone extensive and continuing research in Europe on the forthcoming generation of highly sophisticated cryogenic engines to enhance dependability performance, restartability, and keep engine repeatability low[2].

The design of the Propulsion unit is vital for satellites. In the design of this important satellite piece, fuel and combustor must meet in ideal proportions. The combination of these two substances in ideal proportions means that the combustion is more efficient. There are two different methods that can be used when these two substances come together. The first of these methods is the method we call pressure fed. The other method is the method we call pump fed. These methods have efficient and inefficient aspects compared to each other. Before starting to compare them, it is necessary to give necessary information about the two. The pump fed is a dual-fuel rocket engine in which the fuel pump is powered electrically, so all the fuel supplied is burned directly in the main combustion chamber and is not diverted to power the pump. Pressure-fed engines are a subclass of rocket engines. It is based on the principle of pushing the fuel and oxidant into the combustion chamber by increasing the pressure in the propellant tank, usually with the help of helium. To ensure adequate flow, the pressure in the tank must exceed the pressure in the combustion chamber. This is how we can briefly describe these two methods.

We can now move on to the comparison of these two different combustion triggers. It would be a good point for us to start with the complexity part. In cases where people work together in large scale and very different fields such as satellites, it is very difficult to avoid complexity. Pressure-fed method consists of a simpler structure than pump-fed. The basic principle is to connect the required pressure tank to the desired burner and combustor and to push the combustible and combustible materials into the combustion chamber. When we come to the pump-fed part, electric pumps are used instead of pressure. These electric pumps push the system to a more complex structure. One of the reasons pump-fed is more complex is the number of parts. Pressure-fed offers a simpler design as it contains fewer parts. Another issue is the cost of the work done. We can accept that large-scale studies such as satellites and activation and launching are too expensive for normal people, but when we compare pump-fed with pressure-fed, pressure-fed remains cheaper and lower-cost than pump-fed. Continuing the comparison, the fuel combination we want and the output pressure output from this combustion should move our satellite the way we want, which is slightly better than pump-fed pressure-fed in this area. Pump-fed provides a plus advantage with this aspect. One of the disadvantages of pressure-fed is that the tolerance threshold is not very wide compared to pump-fed.

Another of our comparisons is about development costs, development risks, and development timelines. When evaluating these areas, pressure-fed stands out compared to pump-fed. While the development cost remains lower, the development time, along with the development risk, corresponds to more acceptable risks and times. When we bring all these drums together, the reliability of the established system gains great importance. It offers a more reliable design opportunity compared to pressure-fed pump-fed. When we compare the activation and closing mechanisms, the pressure-fed reacts more easily and in short times, while the pump-fed gives longer and more difficult responses.

As we mentioned before for launched satellites, the increase in mass and volume gives us negative feedback. The method that can be used to prevent this is pump feeding. Pressure-fed pushes more space and mass to be used. In addition to all these comparisons, the biggest problem we will encounter will be the possibility of error. The error we will encounter with pressure-fed can be described as easier and benign. Although we cannot say the same for pump-fed, we are likely to face a devastating error[3].

|  |  |  |
| --- | --- | --- |
| **Characteristic** | **Pressure-fed** | **Pump-fed** |
| Complexity | Simple | Complex |
| Recurring cost | Inexpensive | Expensive |
| Specific impulse | Good | Slightly higher |
| Development cost | Low | High |
| Development risk | Low | Moderate |
| Development schedule | Short | Long |
| Reliability | High | Low |
| Number of parts | Few | Many |
| Tolerances | Low | Very high |
| Start up / shut down | Easy and short | Difficult and long |
| Tank stiffness | High | Low |
| Handling robutness | High | Low |
| Cross-sectional area | High | Moderate |
| Failure type | Benign | Catastrophic |
| Weight | Heavy | Light |

Table 1: Pressure-fed vs pump-fed

While researching the modeling of the pressure fed system with Matlab, the MathWorks Help Center tab on Matlab's site helped me solve the problems I encountered while modeling the system[4].

# DESIGN

## Realistic constraints and conditions

Realistic constraints and conditions vary in space for many different reasons. To start with the simplest, most gases undergo changes in their volumes due to temperature changes. These pressure changes can also cause problems in optimal combustion rates. To avoid these, it would be a better approach to simulate before a real test unit is built. These results can be obtained by trial-and-error methods, but it can put the project in a difficult situation in terms of both time and cost. There are maximum and minimum operating temperatures and pressure values that the valves, sensors, and tanks to be used can withstand. In order to adjust these values optimally, the calculations must be done correctly. Since we cannot reflect the external factors in space too much in Matlab and Simulink, more rounded calculations should not be made. When we come to the STM32CubeIDE and Proteus part, the input and output pins of the microprocessor used may not be sufficient for such a wide-ranging control mechanism. For this control mechanism, designing a microcontroller with the necessary output and input pins may yield healthier results.

## Cost of the design

Since we used the free version of the Matlab application provided for students, the cost of this part is zero. In the part where STM32CubeIDE and Proteus are used together, STM32CubeIDE is a free application, and the trial period of Proteus has been used. The total cost of these two different paths is zero.

## Engineering Standards

One of the necessary things for engineering is to use materials according to certain standards and to emphasize human health. The programs we used while doing this project were made according to certain standards. STM32CubeIDE and Protues programs are leading programs for simulation and embedded systems engineering. While using the Proteus program, it is necessary to know the basic electronics as well as embedded software engineering such as electrical circuits. In the STM32CubeIDE program, we have carried out our work by adhering to the software sources and the software codes given as an example by STM32CubeIDE. Proteus application is of great importance for processes such as PCB design, but it is also used in simulation processes. If it is necessary to use a real test unit for this measurement, the products used must be appropriately selected. The valves, sensors and tanks used must be made up of protectors that are suitable for use and produced in accordance with certain results[5].

Simpace and Simulink programs are used while modeling the system in this project. Simulink is a program that "models, simulates, and evaluates dynamic models," according to the MathWorks website[6]. Simulink is a multi-domain graphical simulation and design environment for time-varying systems. Simulink is an effective tool for examining system behavior over time. Simscape uses the Simulink platform to model and develop physical systems using the Physical Networks technique. Simulink provides Simscape as a block library[7].Each system is shown in the Physical Networks method as being made up of functional components that communicate with one another by swapping energy via their ports. Each of the connecting ports indicates an energy flow, simulating the physical connections that exist between parts. Simscape offers a library of building blocks for several kinds of physical system parts, including hydraulic, electrical, and mechanical systems. This makes it possible for any Simscape model to be easily interchangeable[7].

## Details of the design

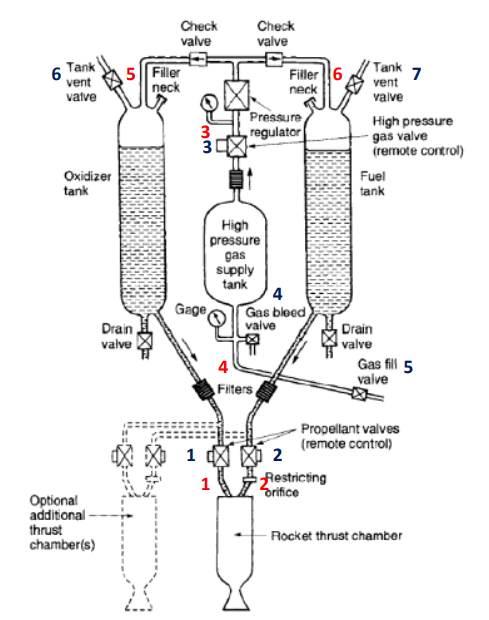


Figure 64: Schematic flow diagram of a liquid fuel rocket engine with a gas pressure feed system and the inputs (red numbers) and outputs (blue numbers) of the microprocessor controlling this system

While we were modelling, we carried out our studies based on the picture above. This modeling was done using pressure-fed, which causes less problems than pump-fed and is more cost-effective. We set out to make this test unit one-to-one. However, due to time constraints, we have taken a different direction and decided to model this test unit and simulate the valves and sensors to be used.

For Proteus and STM32CubeIDE applications to work together, the input and output pins must be marked according to the board to be used by the STM32CubeIDE. After these input and output pins are determined, the necessary software codes should be written in the main.c file and the input and output pins decided before should be taken into account while writing these codes. After these steps are done properly, the written codes should be debugged, and it should be ensured that no errors or warnings are encountered. After this debugging process, the debug file with .hex extension created by the STM32CubeIDe application will be run in the Proteus application. After these processes, the circuits of the parts to be used on Proteus should be created properly. Input and output pins must be connected exactly the same as the codes created in the STM32Cube section. If the transactions have been done correctly up to this point, it should be passed to the operating part of the card. This card should be run with the .hex file we mentioned before. After all the operations are done correctly, it will be seen that the simulation process created in Proteus works in accordance with the written codes.

# METHODS

When we come to method part, we can use go with several way. We try to use two different perspectives, first one is using MATLAB to modelling propulsion unit of satellite. Other one is the modelling propulsion unit with STM32CubeIDE and Proteus Design Suite. If we start with STM32CubeIDE and Proteus Design Suite, we have tried to follow a path that is created by two different applications working together in this method. Proteus is a very advanced program that lets you create electronic circuits, print circuits, and even create computer animations. Besides Proteus, The STM32CubeIDE software development tool is great for working on STM32 microcontrollers and microprocessors on Windows, Linux, and macOS. It has a lot of features for developing code for these devices, including peripheral configuration, code generation, code compilation, and debugging.

Before starting these processes, we must first open the Proteus application. After the application is opened, we must select the board we want to trade. In the software, we have several boards from several companies, since we will use the card that belongs to the STM company, we must enter the model number of the card in the search section and select the card.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 1. Choice of card type on the Proteus.

After selecting the suitable card for our project, we run our first test with a simple LED on/off circuit to check if the Proteus and STM32CubeIDE programs work together properly.

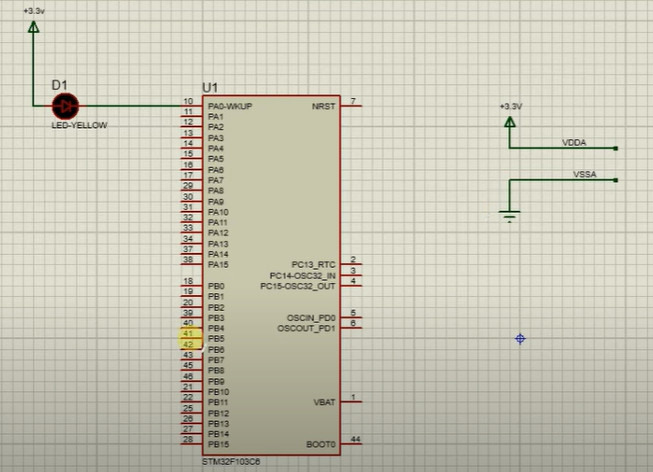


Figure 2. Implementation of circuit with STM32F103C6.

After we set up our LED on and off circuit, we run the STM32CubeIDE software with our card to make it work in an optimized way with the STM32CubeIDE program. After the software is opened, we go to the file section in the application and choose to create a new project, then we select our card, which we also used in Proteus software, in STM32CubeIDE.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3. Selection of card on the STM32CubeIDe.

After these operations, we complete our adjustment to decide how our pins will act as input or output from the pin output screen where we configure the pin of your card in the STM32CubeIDE program.

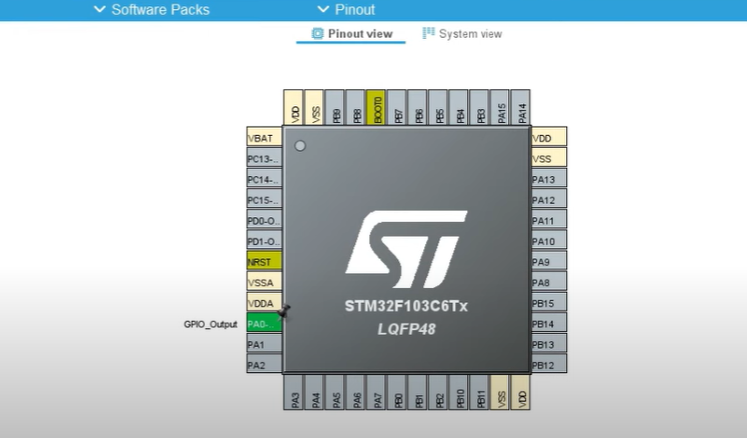


Figure 4. Pinout view of chosen card.

Then, we activate the internal crystal setting so that our card works at high speed. For Proteus and our STM32CubeIDe program to work together, the Proteus program uses the hexadecimal compiled code of the code to be compiled in the STM32CubeIDE. We also activate the hexadecimal compilation process to generate this code.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5. Enabling of .hex extension compiling.

After the settings, we run our project with the build option to create the part where we wrote our code. We can write our code in the file called main.c, which is opened in the application. There are too many predefined fields on this page. Most of these fields contain some guidelines not to be changed by the user. We write the code we will write in the place designated as the beginning and end of the user code, which is one of the places reserved for the user.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6. Inside of main.c at the STM32CubeIDE.

The code we have written gives an output for the pin we have marked. This output process is voltage and is set to be full for half a second and empty for half a second. Our LED lights up when a full voltage is given as an output and stays off when an empty output is given. After writing the code in the fields given for the user, we generate the code we have written. This generating process is for the creation of our hex file and for compiling the written codes. Our work in the STM32CubeIDE part of this compilation process is finished. After checking that the code, we have written works without any errors and warnings after compilation, we will proceed to the process of running this hex file in the Proteus program. After coming to the Proteus screen, we must double-click on the card we have placed in the application and go to the place where the files of the STM32CubeIDE application are saved, and select our file with the .hex extension to run the code created by STM32CubeIDE.

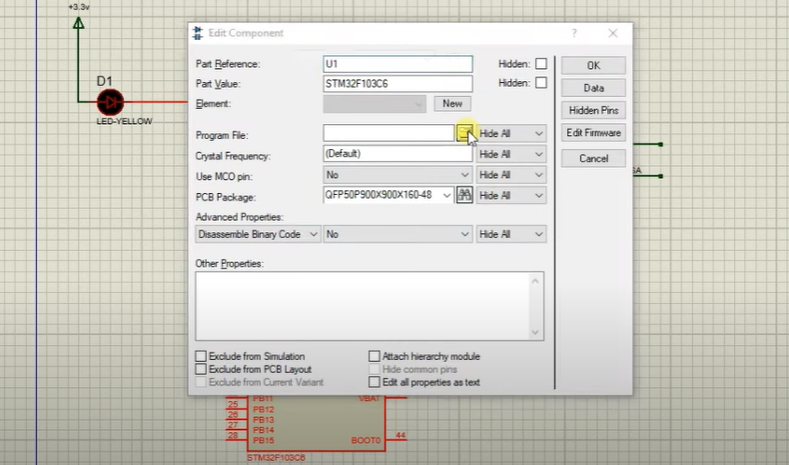


Figure 7. Selection of .hex file for Proteus.

As you can see figure, we select the file that is extension with .hex, this file will used by our card.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 8. Inside of selection file of .hex file for Proteus.

After these processes, we continue the process by pressing the play button on the bottom left of the Proteus application. After performing these operations in order and properly, you will see our LED flashing. Now, we can see LED is on.

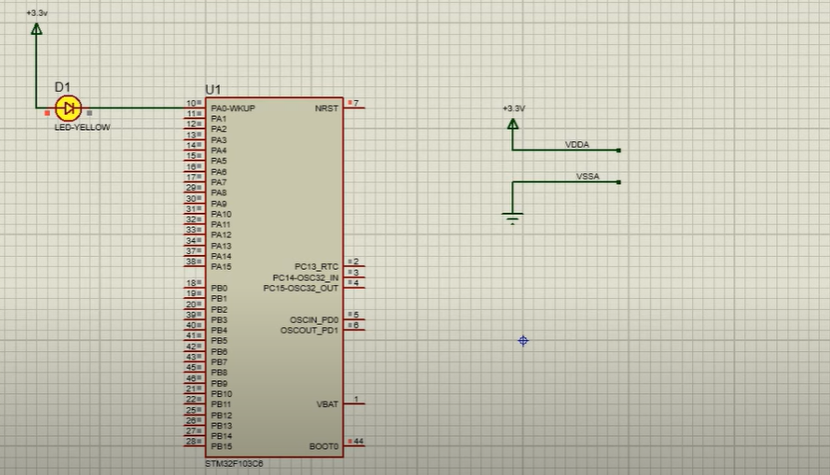


Figure 9. On LED.

After some time-interval which is set by user, we can see that is our LED is off at the below figure 10.

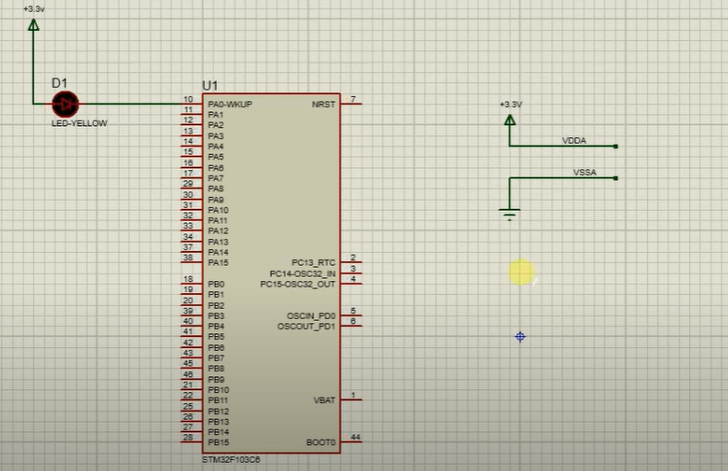


Figure 10. Off LED.

We can use the processes we have done in this order in other materials used in the propulsion unit such as pressure sensor, servo motor, temperature sensor. Servo motor stands for the motors in the ball valves we have used. These motors can be used remotely and by themselves, thanks to the microprocessor. For other materials, although not exactly the same, you can add and use their equivalents in the Proteus application, and their controls are made possible thanks to the codes we wrote in STM32CubeIDE. In addition to all these works, we can take printouts like the pressure sensor or over the servo motor and print these printouts on the LCD panel. These outputs will affect how our system will behave, and we can optimize our system according to these outputs and ensure that it works properly.

Pressure feed system modeling will be done through MATLAB. For this, MATLAB is opened first. Then the command "ssc\_new" is written on the MATLAB command line. This command opens a generic simspace model. This model is shown in Figure 11.

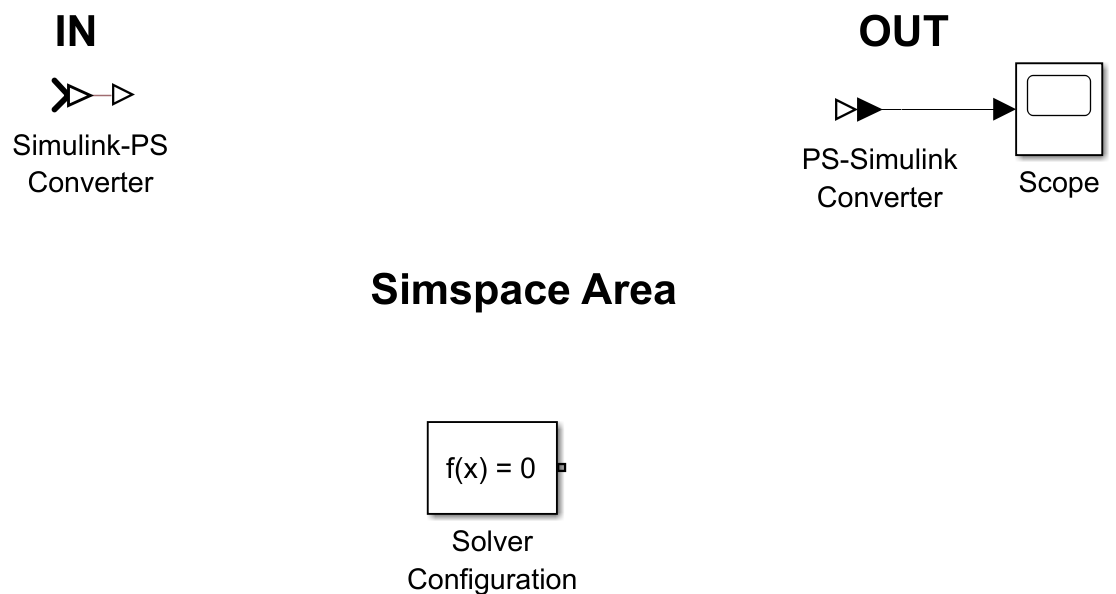


Figure 11: Generic Simspace Model

To transmit data between the Simscape and Simulink domains, there are blocks called Simulink-PS Converter and PS-Simulink Converter. The scope block allows displaying time domain signals. Before you can start simulating, your model needs some solver parameters, which are specified in the Solver Configuration block.

Then, Simspace fluids library is opened by typing "SimscapeFluids\_lib" command on MATLAB command line. This library is shown in Figure 12 below.

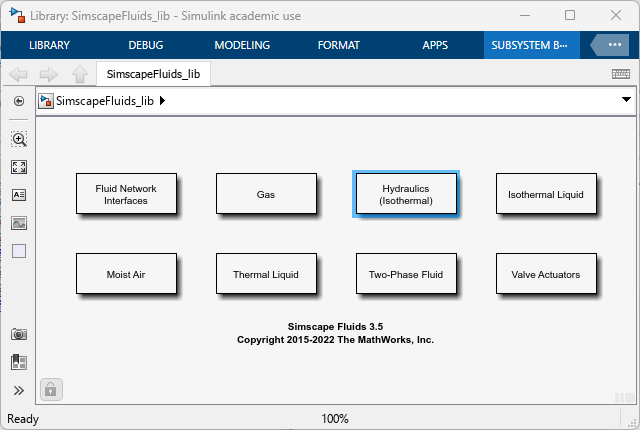


Figure 12: Library: SimscapeFluids\_lib

Then click on the Hydraulics (Isothermal) box.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 13: Library: SimscapeFluids\_lib/Hydraulics (Isothermal)

After these operations, the Hydraulic Fluid block is accessed by double-clicking on the Hydraulic Utilities library in the Hydraulics (Isothermal) library. The Hydraulic Fluid block is added to the model in Figure 14 by dragging it with the mouse.



Figure 14: Hydraulic Fluid Block

With the block shown in Figure 14, you can determine the physical properties of the hydraulic fluid. These physical properties are density, viscosity, and bulk modulus.



Figure 15: Library Browser

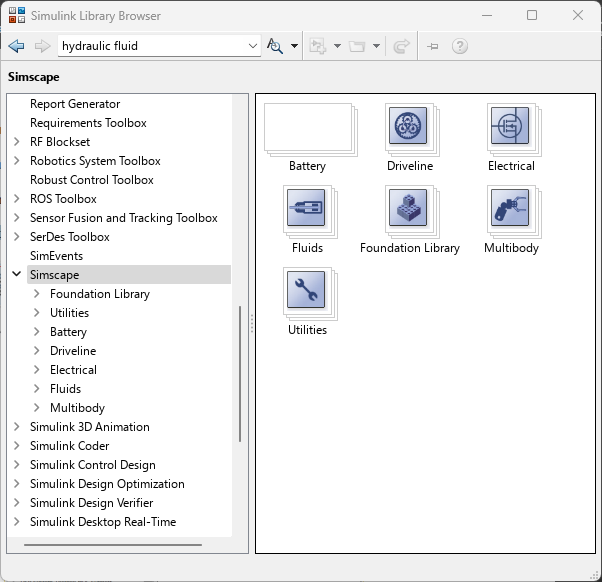


Figure 16: Simulink Library Browser

Then in the canvas window with the model, click the Library Browser button shown in Figure 15 to open the Simulink Library Browser in Figure 16. In the next step, the Hydraulic Constant Pressure Source block, and the Hydraulic Reference block from the Foundation library in the Simscape library of the Simulink Library Browser are found and added to the model. Table 1 shows from which sub-libraries these blocks can be found.

|  |  |
| --- | --- |
| **Block** | **From Sublibrary** |
| Hydraulic Constant Pressure Source | Foundation Library/ Hydraulic/ Hydraulic Sources |
| Hydraulic Reference | Foundation Library/ Hydraulic/ Hydraulic Elements |

Table 2: Blocks and Sub-libraries of Hydraulic Constant Pressure Source and Hydraulic Reference

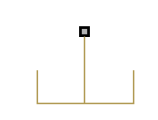
 

Figure 17: Hydraulic Constant Pressure Source Figure 18: Hydraulic Reference

In the next step, Solver Configuration, Hydraulic Fluid, Hydraulic Constant Pressure Source and Hydraulic Reference blocks are connected as in Figure 19.

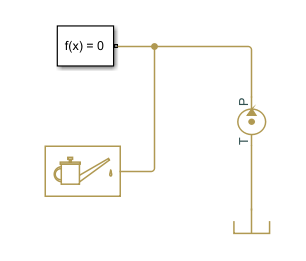


Figure 19: Connection of Solver Configuration, Hydraulic Fluid, Hydraulic Constant Pressure Source and Hydraulic Reference Blocks

In the next step, Double-Acting Hydraulic Cylinder (Simple) and 4-Way Directional Valve blocks from the Fluids library in the Simscape library are dragged into the model. Table 2 shows from which sub-libraries these blocks can be found.

|  |  |
| --- | --- |
| **Block** | **From Sublibrary** |
| Double-Acting Hydraulic Cylinder (Simple) | Hydraulics (Isothermal/ Hydraulic Cylinders |
| 4-Way Directional Valve | Hydraulics (Isothermal)/ Valves /Directional Valves |

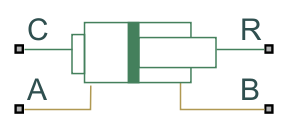
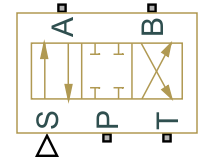
Table 3: Blocks and Sub-libraries of Double-Acting Hydraulic Cylinder (Simple) and 4-Way Directional Valve  

Figure 20: Double-Acting Hydraulic Cylinder (Simple) Figure 21: 4-Way Directional Valve

Real components have physical connections between each other. These links are represented by block links. The fluid reservoir is connected to the cylinder by way of the valve, which is then joined to the pump.

In the next step, the Translational Reference block (block shown in Figure 22), which is the mechanical translation reference point, is connected to the Double-Acting Hydraulic Cylinder block. These two blocks are connected as in Figure 23.

metin, anten içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 22: Translational Reference Block

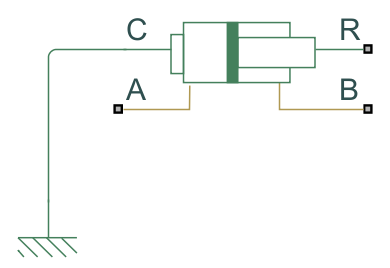


Figure 23: Connection of Double-Acting Hydraulic Cylinder and Translational Reference blocks

In the next step, 4-way Directional Valve and Double-Acting Hydraulic Cylinder blocks are connected. This connection is shown in Figure 24.

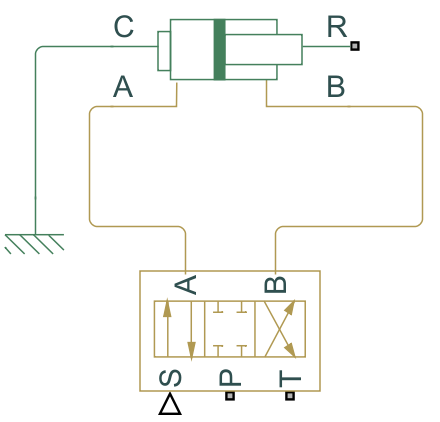


Figure 24: Connection of Double-Acting Hydraulic Cylinder and 4-way Directional Valve blocks

In the next step, the Mass block is connected to the model in Figure 26. The sub-libraries of the Mass block are shown in Table 3.



Figure 25: Mass Block

|  |  |
| --- | --- |
| **Block** | **From Sublibrary** |
| Mass | Simscape/ Foundation/ Mechanical/ Translational Elements |

Table 4: Blocks and Sub-libraries of Mass

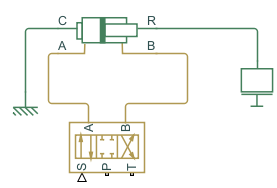


Figure 26: Connection Between Mass Block and The Model in Figure 24

In the next step, the Ideal Translational Motion Sensor block is added to the model in Figure 28. The sub-libraries of Ideal Translational Motion Sensor block are shown in Table 4. After this block is added to the model, the R and C terminals of the Ideal Translational Motion Sensor block and the R and C terminals of the Double-Acting Hydraulic Cylinder (Simple) block are connected to each other separately. After adding the Ideal Translational Motion Sensor, the next version of the model is shown in Figure 28.

|  |  |
| --- | --- |
| **Block** | **From Sublibrary** |
| Ideal Translational Motion Sensor | Simscape/ Foundation/ Mechanical/ Mechanical Sensors |

Table 5: Blocks and Sub-libraries of Ideal Translational Motion Sensor

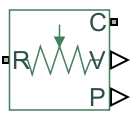


Figure 27: Ideal Translational Motion Sensor Block

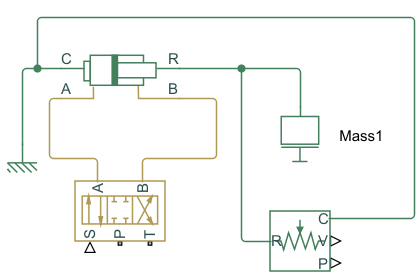


Figure 28: Next Version of The Model in Figure 26

In the next step, the Ideal Translational Motion Sensor is connected with PS-Simulink Converter and Scope blocks as in Figure 29. The P terminal in the Ideal Translational Motion Sensor block represents the position. The V terminal in the Ideal Translational Motion Sensor block represents the velocity.

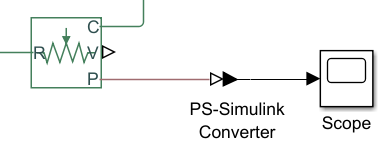


Figure 29: Connection with Ideal Translational Motion Sensor, PS-Simulink Converter and Scope Blocks

In the next step, Valve Actuator Block is added to the model so that the signal that initiates the operation of the valve can be sent to the valve. This signal is sent by connecting the Simulink-PS Converter and the Valve Actuator block. Valve Actuator Block is connected with S terminal of 4-Way Directional Valve Block.



Figure 30: Valve Actuator Block

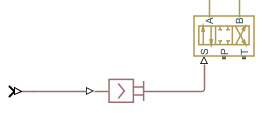


Figure 31: Connection of Valve Actuator Block and S terminal of 4-Way Directional Valve Block

In the next step, the 4-Way Directional Valve is connected to the Hydraulic Constant Pressure Source block and the Hydraulic Reference block as shown in Figure 32.

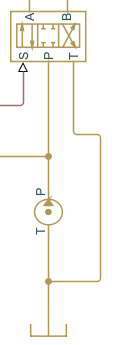


Figure 32: Connection of 4-Way Directional Valve Block and the Hydraulic Constant Pressure Source block and the Hydraulic Reference block

In the next step, the Hydraulic Pressure Sensor block is added to the model to measure cylinder chamber pressures. The sub-libraries of this block are shown in Table 5. The model in Figure X has two Hydraulic Pressure Sensors. One is between interface A of the hydraulic cylinder and the hydraulic reference (interface B), and the other is between interface B and the hydraulic reference (interface B) of the hydraulic cylinder.

|  |  |
| --- | --- |
| **Block** | **From Sub-library** |
| Hydraulic Pressure Sensor | Simscape/ Foundation/ Hydraulic/ Hydraulic Sensors |

Table 6: Blocks and Sub-libraries of Hydraulic Pressure Sensor

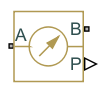


Figure 33: Hydraulic Pressure Sensor Block

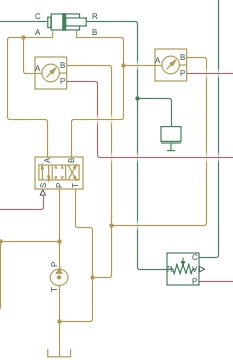


Figure 34: Version of the Model After Adding 2 Hydraulic Sensor Blocks

The next step is to connect 2 Hydraulic Pressure Sensors to the Scope using the PS-Simulink Converter block, as shown in Figure X. Using a PS-Simulink Converter as in Figure 35, both Hydraulic Pressure Sensors are connected to Scopes. The current converter in the Model is copied as much as necessary and added to the model.



Figure 35: Connection of Hydraulic Pressure Sensor with PS-Simulink Converter and Scope

Add block, Step block and Constant block are added to the model to give signal input and a structure is created as in Figure 39. Sub-libraries of Add Blog, Step blog and Constant blog are shown in Table 6.

|  |  |
| --- | --- |
| **Block** | **From Sub-library** |
| Add Block | Simulink/ Math Operations |
| Step Block | Simulink/ Sources |
| Constant Block | Simulink/ Sources |

Table 7: Blocks and Sub-libraries of Input Signals

kare içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, saat içeren bir resim

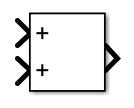
Açıklama otomatik olarak oluşturuldu 

Figure 36: Step Block Figure 37: Constant Block Figure 38: Add Block

metin, ışık içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 39: Signal İnput Structure

The constant block represents the parameter of the valve at the zero point. The initial constant value is set to zero. This parameter is adjusted later so that the cylinder can be held steady when there is a zero-input signal. Step blocks are used as step input commands for proportional control valve. These parameter values are adjusted in later stages.

The Hydraulic Pipeline block is used to model hydraulic piping with circular cross-sections. This block is located between the A and B terminals of the Double-Acting Hydraulic Cylinder (Simple) and 4-Way Directional Valve blocks. The sub-libraries of this block are shown in Table 7.

|  |  |
| --- | --- |
| **Block** | **From Sublibrary** |
| Hydraulic Pipeline | Simscape/ Fluids/ Hydraulics (Isothermal)/ Pipelines |

Table 8: Blocks and Sub-libraries of Hydraulic Pipeline



Figure 40: Hydraulic Pipeline Block

Two Hydraulic Cylinders are added to the model and connected to the A and B interfaces of the Hydraulic Cylinder and the corresponding A and B interfaces of the 4-Way Directional Valve, as in Figure 40. These blocks can be rotated with the keyboard shortcut “Ctrl+R”.

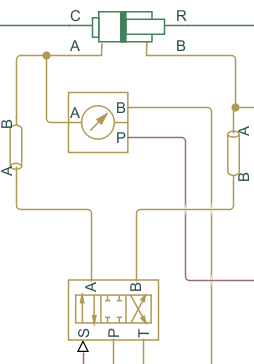


Figure 41: Connection of Hydraulic Pipeline Block

Double-clicking on the blocks opens the parameter input tabs. These windows are used to set system parameters. Step blocks for timing valve commands are set as in Figure 42, Figure 43, Figure 44 and Figure 45, and the usable voltage range is set to -10 to +10 Volts.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 42: First Step Signal Parameters

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 43: Second Step Signal Parameters

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 44: Third Step Signal Parameters

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 45: Fourth Step Signal Parameters

Valve actuator parameters are set as in Figure 46.

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 46: Valve Actuator Parameters

The maximum Valve stroke is set to 0.005 m. If we apply 10 V input, the actuator reaches a full 0.005m pulse. Therefore, the actuator gain is found to be 0.005/10 m or, in theory, 0.005/10 m/V. The time constant value becomes 0.002 s (2 ms).

Pressure parameter of Hydraulic Constant Pressure Source block is entered as 120e5 as in Figure 47. This value represents an ideal pressure source with a constant pressure of 120 bar. In Matlab, 120 bar (Pa) is entered as 120e5.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 47: Hydraulic Constant Pressure Source Parameters

The system consists of two parts, in the details of the design described in the Detail of Design part, before proceeding to adjusting the parameters of the 4-Way Directional Valve. Therefore, the parameters of two 4-Way Directional Valves need to be adjusted.

We need to use equations to set the parameters of the 4-Way Directional Valve. For a narrow (i.e. mostly turbulent flow) orifice, the flow rate can be defined as in the equation below.

The variables in this equation are defined as: nominal flow rate (), nominal pressure drop (), fluid density (), flow coefficient () and the corresponding flow area ().

In the above equation, when we take the corresponding flow area variable to the left of the equation and other variables to the right of the equation, we get the following equation that will allow us to find the corresponding flow area variable.

The values used in part 1 of the model for leakage of a particular proportional control valve are as follows:

* (Density of Liquid Oxygen=)( Since Liquid Oxygen is not in the library, the item with the closest density value is selected.)

The values used in part 2 of the model for leakage of a particular proportional control valve are as follows:

* (Density of jet fuel)

The leakage area parameters in part1 and part2 of the model are as follows:

* Leakage area parameter of part1 =
* Leakage area parameter of part2 =

These values are found separately from the corresponding flow area equation.

Before entering the values in the Model Parametrization tab in the 4-Way Directional Valve block dialog box, we need to calculate these values. Simscape's way of modeling orifice area (maximum opening area in parameter tab) is calculated with the following equation.

The variables in this equation are defined as:

* orifice opening () []
* maximum orifice opening () []
* maximum orifice area () []
* leakage orifice area () []

The leakage orifice area is used for a “closed orifice”.

The maximum opening area is calculated with the following equation.

The maximum opening area parameters in part1 and part2 of the model are as follows:

* Maximum opening area of part1 =
* Maximum opening area of part2 =

As shown in Figure 48 and Figure 49, the values found for the parameters of 4-Way Directional Valves in part1 and part2 of the model are entered into the 4-Way Directional Valve block dialog box.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 48: 4-Way Directional Valve Block Dialog Box of Part1

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 49: 4-Way Directional Valve Block Dialog Box of Part2

In the next step, the parameters of the Double-Acting Hydraulic Cylinder are set. The parameters of Double-Acting Hydraulic Cylinder are found by the following equations:

* Piston area A:
* Piston area B:
* Piston stroke: 1m (maximum stroke)
* Piston initial distance from cap A: 0.8m (initial position of piston)

The variable D in the above equation represents the cylinder diameter and its value is 32mm. The variable d in the above equation represents the rod diameter and its value is 20mm. When we replace the values, the values of the parameters are found as follows:

* Piston area A:
* Piston area B:

Hydraulic Pipeline parameters are updated by entering the following values. The value of the pipe internal diameter parameter from the Hydraulic Pipeline parameters is updated to 0.012m and the value of the pipe length parameter to 0.75. The value of the pipe internal diameter parameter from the Hydraulic Pipeline parameters is updated to 0.012m and the value of the pipe length parameter to 0.75. These entered values are displayed in the Hydraulic Pipeline dialog box as in Figure 50.

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 50: The Hydraulic Pipeline Dialog Box

In the next step, the initial position parameter of the Ideal Translational Motion Sensor block is set to 0.8m as shown in Figure 51.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 51: Ideal Translational Motion Sensor Dialog Box

The final version of the pressure fed system modeling consists of two parts, Part1 and Part2. Part1 of the model is shown in Figure 52 below and Part2 of the model is shown in Figure 53 below.

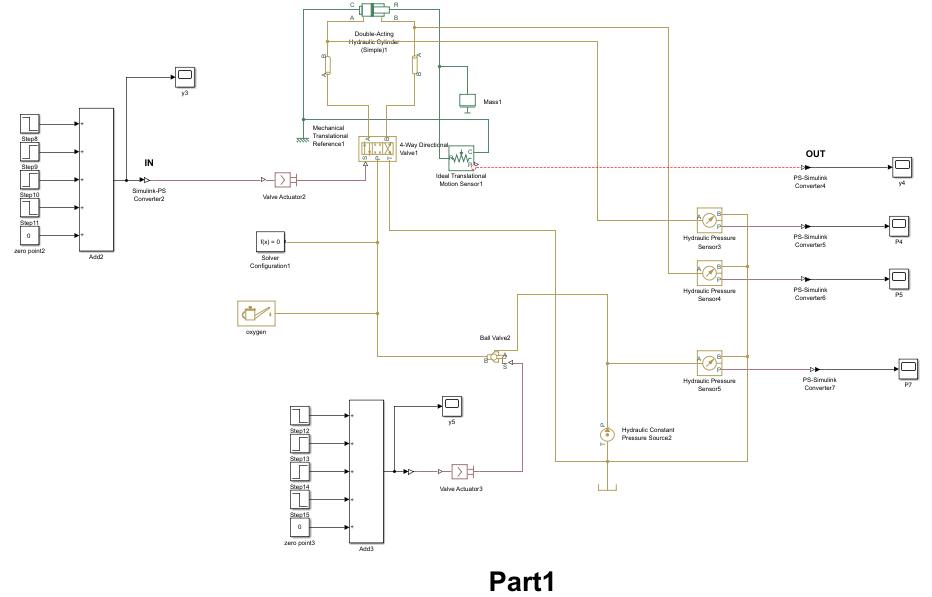


Figure 52: Part1 of The Model (Liquid Oxygen Part)

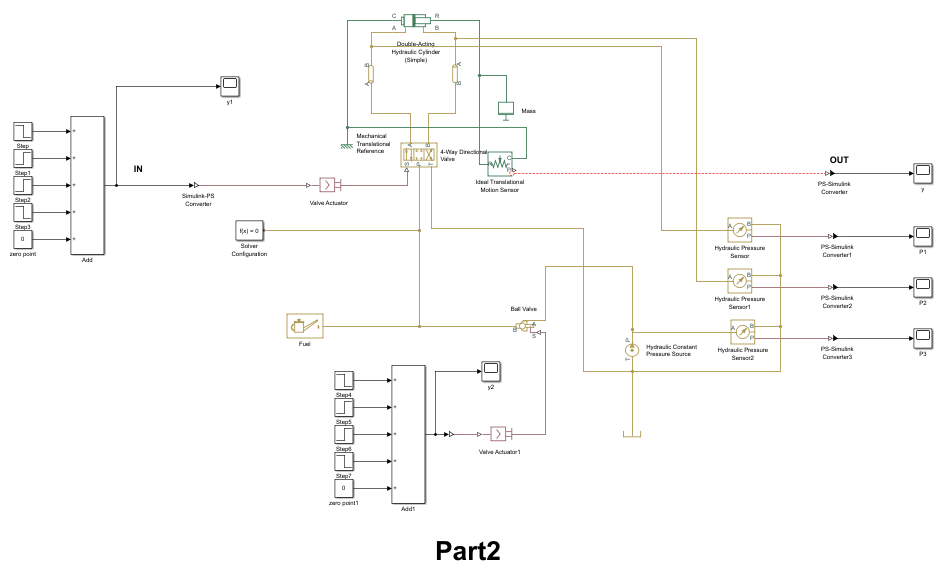


Figure 53: Part2 of The Model (Jet Fuel Part)

# RESULTS AND DISCUSSION

When we come to the result and discussion part, we can say that the installed systems work successfully. We tested that STM32CubeIDE, and Proteus programs can work together easily and that this optimal operation can be used in different areas. We have seen that the control mechanism of any system can be used by adapting it to different variables. By working with two different applications such as STM32CubeIDE and Proteus, we have the opportunity to avoid the errors that we will encounter while simulating in real life. Errors that we may encounter while testing real test units can be large enough to lead to death. In a real system, mishaps such as valves and pressure sensors not being properly connected or adjusted have severe consequences. Even if all the parts are connected properly, it also gives the opportunity to prevent accidents that will affect the system and devastating levels such as undesirable external factors and high-pressure explosions. The simulated test units of the propulsion units of the cubesat satellites that we have built in Matlab and Simulink applications also give the chance to prevent such errors. While preventing these errors, it also gives us information about how a real test unit can be designed.

When we come to the result and discussion part of the modeling of the pressure feeding system, we can say that the modeled system has partially achieved its purpose. We tested the modeling of the pressure feeding system with MATLAB and Simulink programs. After modeling this system, some results were obtained by simulating it. The obtained results can be listed as follows: Output of the input signal that activates the valves, monitoring the pressure in the constant pressure source with Scope, observing the pressure change in the system using pressure sensors, and observing the position of liquid oxygen and jet fuel in the system with motion sensors.

In Figure 54 below, the output of the signal produced by the structure created to generate a signal with the combination of Add block, Constant block and Step blocks is observed with Scope. This signal passes through the Valve Actuator block and enables the valves to be activated.

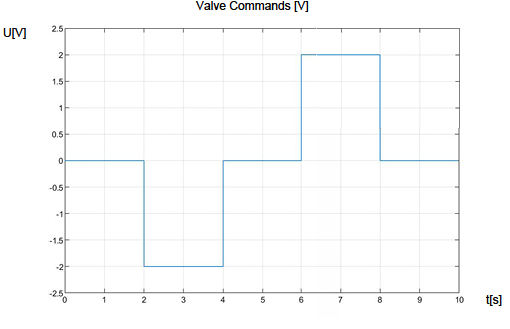


Figure 54: Input Signal of Valve Actuator Block

In the figures (Figure 55, Figure 56, Figure 57, and Figure 58 that time-varying pressure graphs) below, pressure changes in the system can be observed with the pressure sensors used in the model. In the model for which the system is designed, two types of materials are used, namely liquid oxygen and jet fuel. With these observed pressure changes, pressure changes that only one material is exposed to in the system can be observed, while the pressure changes experienced by two different materials can also be compared.

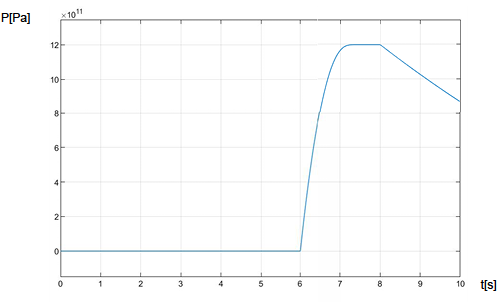


Figure 55: Scope of P4 (Part 1)

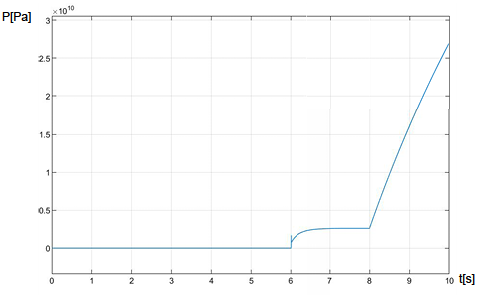


Figure 56: Scope of P5 (Part 1)

In Figure 55 and Figure 56, the pressure changes that the liquid oxygen is exposed to in the relevant parts of the system are shown.

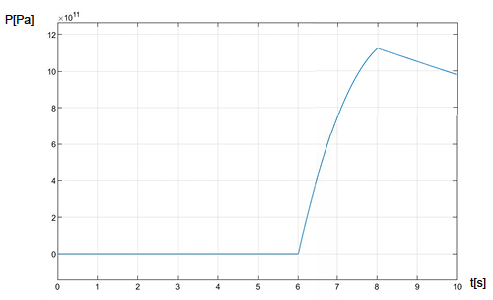


Figure 57: Scope of P1 (Part 2)

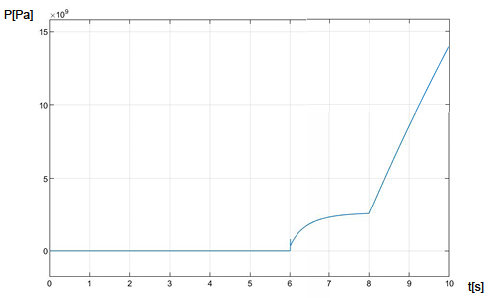


Figure 58: Scope of P2 (Part 2)

In Figure 57 and Figure 58, the pressure changes that the jet fuel is exposed to in the relevant parts of the system are shown.

In Figure 59, the pressure of the constant pressure source measured by the pressure sensor in the model is displayed on the Scope.

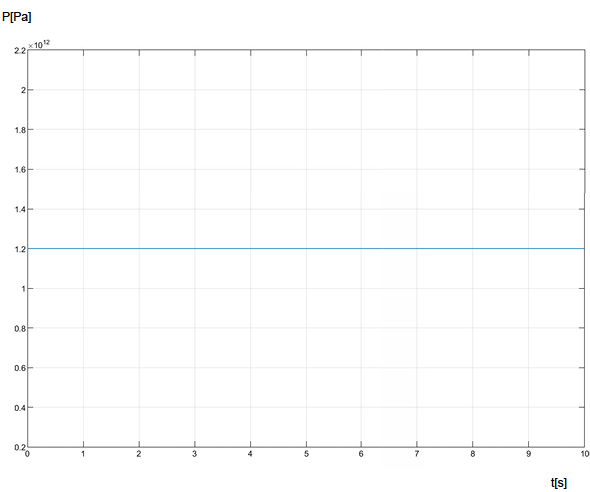


Figure 59: Scope of P3(Hydraulic Constant Pressure Source)

Figure 60 shows the time-dependent change in position of the liquid oxygen material.

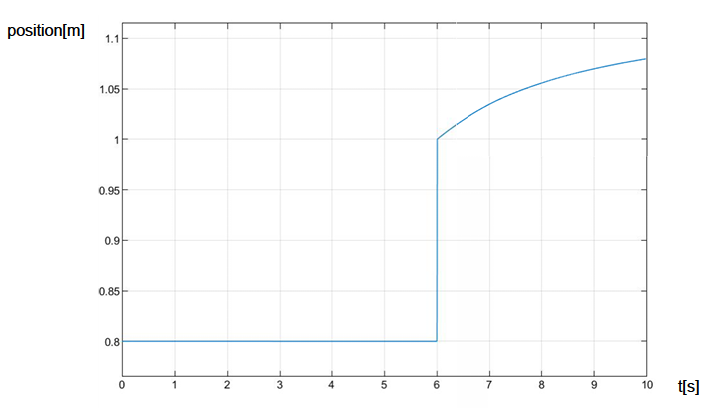


Figure 60: Scope(y4) of Position (part1)

Figure 61 shows the time-dependent change in position of the jet fuel material.

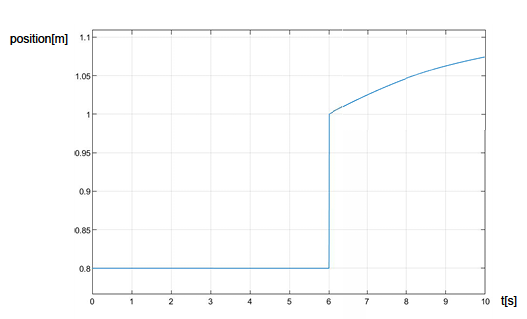


Figure 61: Scope (y) of Position (part2)

# CONCLUSION

When we came to the conclusion part, we learned that using STM32CubeIDE and Proteus program together is an efficient method for simulation. By using the two programs together, we get that valves and sensors used in a real test unit can be simulated. We use servo motors as two-way dc motors in ball valves. For the sensors, we simulated on equivalent sensors in Proteus. In addition, simulations were carried out with the help of Matlab and Simulink applications, taking into account the modeling and necessary calculations of a real test unit before it was implemented. Momentary pressure changes were observed and necessary measures were taken to prevent them from causing any errors in the system. With these simulations, we have prevented possible errors and accidents.

# REFERENCES

[1] Comparison of pump fed vs. pressure fed rocket systems. Pump fed saves up to 90% on tankage weight at similar cost. Edge. (n.d.). Retrieved January 20, 2023, from http://edge.rit.edu/edge/P07109/public/Design%20I/Published%20Research%20Documents/pfcalcs.pdf

[2] A. Lekeux, J. Pander, and J. Mansouri, “From qualification to 4th year of exploitation: Hm7b on esc a,” in 45th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, 2-5 August 2009, Denver, Colorado, 2009. AIAA 2009-5036.

[3] S. Dr. Chakroborty and T. P. Dr. Bauer, “Using pressure-fed propulsion technology to lower space transportation costs,” in 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Fort Lauderdale, Florida, 2004.

[4] Matlab. MATLAB Documentation. (n.d.). Retrieved January 20, 2023, from https://www.mathworks.com/help/matlab/

[5] Piping Material Specification. Engineering standard - Piping Material Specification - Valve Tag Numbering. (n.d.). Retrieved January 20, 2023, from https://www.wermac.org/documents/engineering\_standard\_part5.html

[6] Makers of Matlab and Simulink. MathWorks. (n.d.). Retrieved January 20, 2023, from https://www.mathworks.com/

[7] NASA. (n.d.). General Pressurization Model in Simscape - NASA Technical Reports Server (NTRS). NASA. Retrieved January 20, 2023, from https://ntrs.nasa.gov/citations/20100035692

# APPENDICES

## Appendix A

The Simulink model and all other components of the project are available at the GitHub link below.

* [Project Link](https://github.com/YunusYamakMU/GraduationProject)