Evidence of extracurricular activities

This is a collection of files on my extracurricular activities throughout my bachelor's. Documents are listed back to back in the following order:

1. Summer Internship I: Internship report of my internship at SolarBiotec as embedded systems intern.

2. Summer Internship II: Internship report of my internship at TOFAS, a car manufacturer partner with Stellantis where I was a model base development intern.

3. METU VEGA Conceptual Design Report: Fire detecting UAV project design proposal report for Tubitak Teknofest Competition.

4. METU Ascend Proposal Report: Path finder UAV project design proposal report. This report is published in Turkish and this is a draft partially in English for our advisor at the time Anna Prach.

MIDDLE EAST TECHNICAL UNIVERSITY NORTHERN CYPRUS CAMPUS

ELECTRICAL AND ELECTRONICS ENGINEERING PROGRAM
EEE – 300
Summer Practice Report

Student Name:	**Barış Güzel**
Student ID:	**2315935**
Summer Practice - Start Date:	**01.08.2021**
Summer Practice - End Date:	**01.09.2021**
Report Submission Date:	**15.11.2021**
Company Name:	**Solar Biyoteknoloji LTD**
Company Division/Department:	**R&D**
Supervising/Mentoring Engineer Name and Professional Title:	**Emre Kurtulan** **Electronics R&D Engineer**
Company Address:	**Alsancak m. 1476-1 s. 9/1 Kordon Kule, Konak 35220 Izmir, TR**
Field of Practice in Electrical and Electronics Engineering	**Embedded Systems**

TABLE OF CONTENTS

1. INTRODUCTION

I have done my EEE 300 internship in SolarBiotec for one month. It is a small research and development company in Izmir/Alsancak. I worked on embedded software and hardware developments. I had the chance to experience how a research engineer works. SolarBiotec hires many interns and gives them the chance to work as an engineer before graduation, solve problems independently, and guide them. I had access to an excellent laboratory in the office. Trying and failing many times taught me many problem-solving techniques and easy and cheap ways to fix problems.

In my internship, between 01.08.2021 to 01.09.2021 took notes and pictures for preparing this summer internship report. Mr. Kurtulan was my advisor in my internship. He worked on the project I was assigned to before and made the first prototype. My responsibility was to add new features and migrate them to STM32 architecture, which will be explained further in detail with the struggles I encountered. The methods I used, and the trials I made are explained in detail in this report.

2. COMPANY DESCRIPTION

2.1. A brief history of the company

SolarBiotec was founded as a Teknogirisim (Technoentrepreneurship) company to perform the technological feasibility experiments to develop a portable instrument for molecular diagnostics. By the completion of the project, the company moved to the Izmir Technology Development Zone, to get a tax-exempt status in 2014. In 2015, Solar Biotec gained Genoma Medical Technologies to increase their experience further in Medical/Laboratory instrument design. [1]

2.2. Main Area of Business

Solar Biotechnology Ltd. is an Izmir based technology company specialized in design and manufacture of MEMS or robotic based solutions to simplify complex systems in healthcare by a multidisciplinary approach.[1]

2.3. Organizational Structure of the Company

SolarBiotec is a small R&D company. Figure 2.1 shows the organization structure of the company. SolarBiotec is a project-based company each engineer has different responsibility in different projects. They develop different projects simultaneously. Company enables university students to get field experience by hiring multiple interns throughout the year. Interns would have varied responsibility depending on the workload. I worked with all the engineers, but Emre Kurtulan was the one responsible with most of the interns like me. [1]

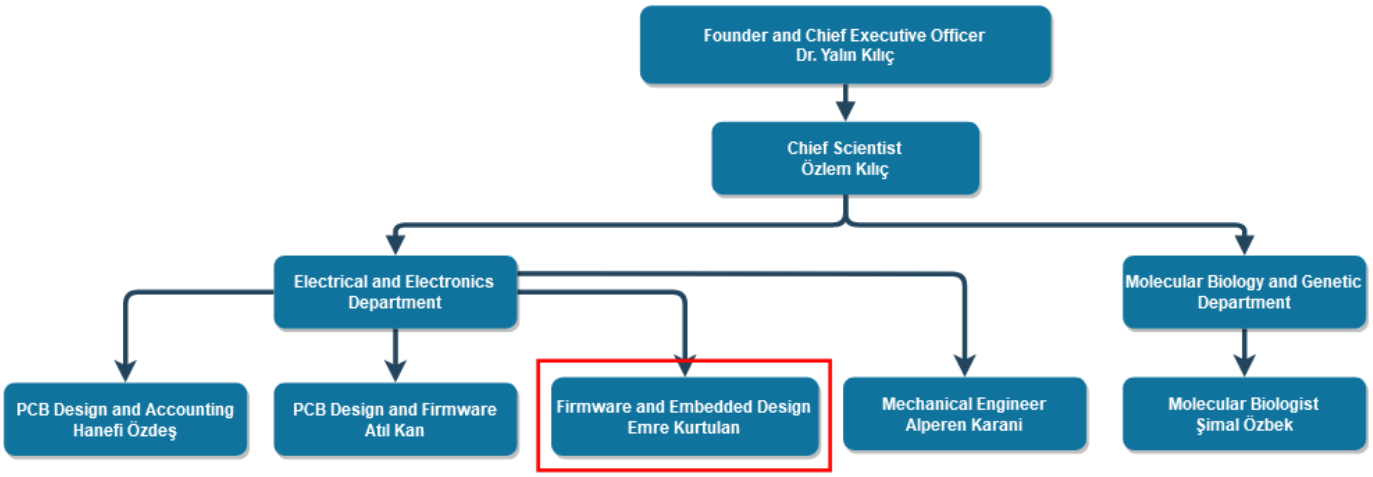


Figure 2.3-1: Organizational structure [1]

3. FIRST WEEK

3.1. Introduction

At the beginning of the first week, my advisor Mr. Kurtulan, Embedded Design Engineer, introduced me to the lab and colleagues. After Dr. Yalcin arrived, a short interview was conducted to determine which project fits my interest the most. In the following chapters, my assignment will be explained in detail. Then an account is set up in ClickUp, software for business project management.

3.2. Clickup Setup

Clickup is software that is widely used in project-based businesses. It makes the distribution of tasks easy and efficient. It was an advantage to getting used to one of the most popular apps for the purpose. I completed an introduction inside the app explaining how the company uses the app, and my first tasks are given for my time in the company as an intern. [2]

3.3. Assignment: Pressure Control Valve

In my internship, I spend my time developing a pressure control valve system for an extraction machine. Briefly, how the extraction machine works is a gas, commonly CO_2 or N, pressurized and heated in many steps to push the gas into a supercritical state. Supercritical fluids extract or dissolve liquids and gasses that are much better than gas and oil alternatives. CO_2 in a supercritical state fills another chamber with the material to be extracted. This operation has multiple steps with different pressures and temperatures for other materials. Supercritical extractors are big and expensive machines. Our purpose was to make a mini version of the product without losing functionality.

As explained earlier, the process requires different pressures and instant changes in pressure in a stable manner. Since this extractor version is meant to be small, the gas pump for pressurization is more basic than the competitors, creating unstable pressure values. The pump can only increase a certain amount of pressure at once and can exceed desired value. A valve is added to eliminate excess pressure or decrease the pressure to a much lower value instantly. Mechanical relief valves aren't fast and reliable enough because of the stability concern. Another reason an electronic system is used is mechanical relief valve pressure threshold can't be changed easily.

My assignment was to create a PID control system and test it. There was a prototype shown in figure 1 below that was functional. But only uses a proportional control algorithm with an older processor. Before my internship, the team had already drawn and printed a new controller card with an STM processor, which is faster and more commonly used than the old prototype. In this older prototype, the stepper motor is connected to the valve with a belt. An infrared sensor is used to determine a fully closed position. Another main issue with the older version was that when highly pressurized CO2 escapes, it freezes the valve, making the motor unable to rotate. Due to that loses all the pressure in the system. Another PID system for heating the valve had to be added.

I'm assigned to test the new PCB and functionality of the sensors with the new processor, control stepper motor, and program PID control of both pressure and temperature all in embedded C using STM32CUBE IDE. The workflow was: sub assignments are added, which I discussed before to Clickup, and when I finished each, I wrote short reports for each task and marked them as completed.

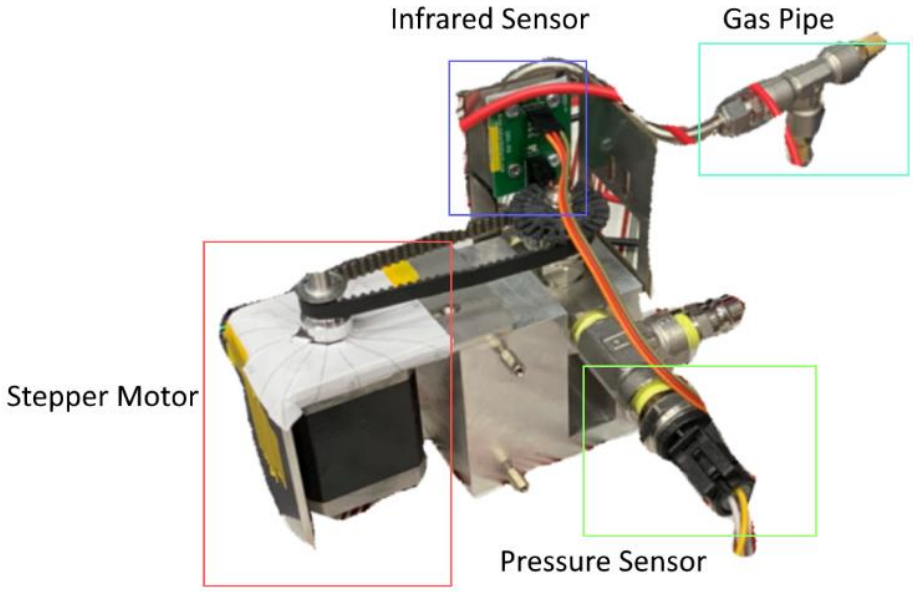


Figure 3.3-1 Older Relief Valve Prototype

3.4. Studying STM32F373 Processor and Setting Pin Layout

I started with reading manuals of both processor and motor driver (DRV 8825). STM32F373 is a widely used microcontroller, and it was the one that the company has in stock the most. Because of the semiconductor crisis all over the globe, stocks were always considered while designing. The processor is a 72MHz processor with 256k flash and 32k SRAM capacity. It has one 12bit fast ADC and three 16-bit ADC, 2 DAC. In terms of communication, it has 2 I^2C, 3 SPI, 3 UARTS, and a USB. [3] Due to the small package size of 48 pins, all of the functionalities cannot be used simultaneously. So, pin distribution was an important design consideration. Since the PCB designs are private IP, I'm unable to share the whole design, but the following figure shows how pins are distributed. In the first attempt, 2 ADC pins are used one for pressure read and one for temperature control, which I'm going to explain when I worked on temperature control in the following weeks. Two external interrupt pins are used, one for an infrared sensor to find the home location and another to determine if the valve is opened all the way through. This algorithm will be changed in the following weeks, but my first assignment was to check functionality.

Motor driver DRV8825 takes nine pins 3 for mode selection 1 PWM signal and five control signals such as direction, enable, sleep, decay mode, and reset.[4] Four pins are used for communicating with an IC, which is responsible for temperature reading.[5] Two pins are used for USB communication for plotting sensors data. I've written a short UART communication

code in C using STM32CUBE IDE, which I was already familiar with to test the basic functionality of the control card.

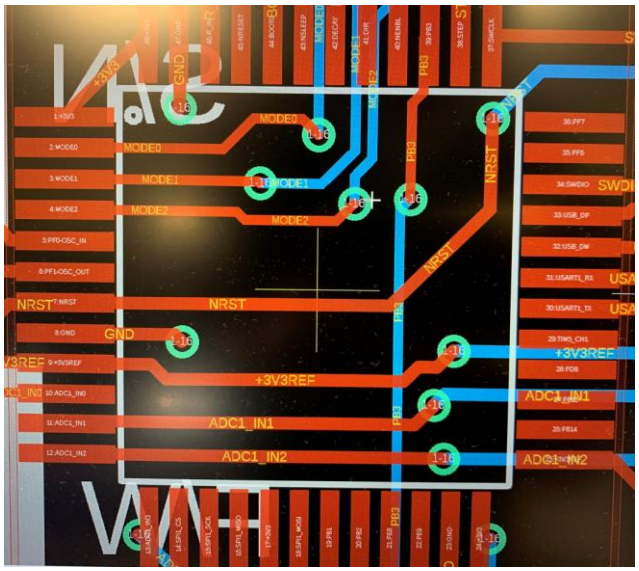


Figure 3.4-1 Pin Layout of STM32F373 Processor

3.5. Motor Control with DRV8825

One of my first assignments was to write a code utilizing DRV8825 and control the stepper motor. The functions should take rotation angle and direction as input and control the motor through the driver. DRV8825 is a driver from Texas Instruments, primarily used in 3D printer stepper motor control boards. Solarbiotech developed development boards for each of their stocked processors and test boards with breadboards for developing without printing PCBs. In figure 3.5.1 below, I'm using one of the F373 development boards to test the motor driver. The Purple board in the middle is a breakout board of DRV8825 which was designed for 3D printers.

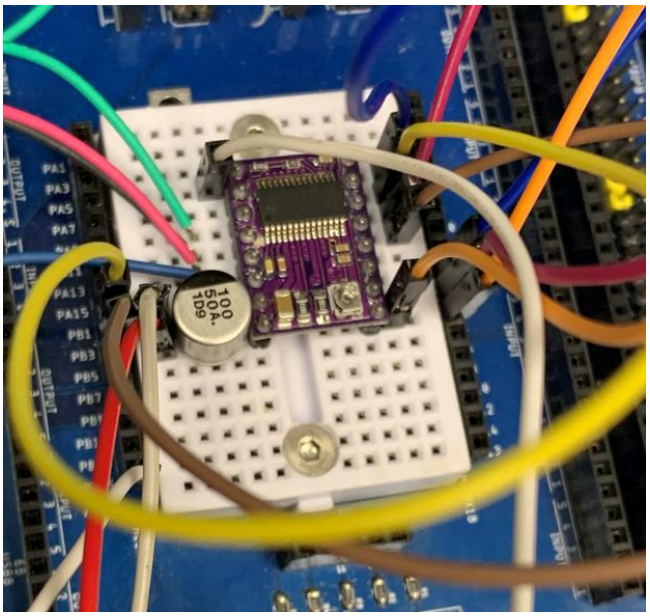


Figure 3.5-1 DRV8825 Breakout Board

To rotate a stepper motor, each coil should be activated one by one, each one called a step. In different modes, total steps for one full rotation and activated pins can vary. The most common ones are full, half, and micro-steps such as 1/8 1/32 steps, etc. Minimum step angle depends on the motor with micro-stepping 1 step can be divided into more steps meaning in 1/8 micro-step mode, four steps or cycles must pass for minimum step angle to be traveled. Check Appendix A for the micro-step table of DRV8825. A Diagram of the motor used in this project is shown in Figure 3.5.2 below with a datasheet.[6] The minimum step angle of this motor is 1.8 degrees. DRV8825 is capable of a maximum of 1/32 micro-steps. Step signal is sent to the controller, and in each rising edge, the driver sets output pins to the corresponding next step creating a graph similar to figure 3.5.3. Sleep and Reset and Enable pins are active low. For the motor to operate, Enable should be 0, and both Sleep and Reset should be set to 1. Reset resets the state table of steps, and sleep decreases the current through the motor. There were timing requirements stated in the datasheet that I considered while programming the functions.

Microstep mode is selected with three pins 1 1 1 corresponds to 1/32 and 1 1 0 1/8 etc. I created an initialization function and set the micro-step to 1/8. Because of the basic design of the motor controller, even with the maximum acceptable Step PWM signal timing, 1/32 micro step was too slow for the application. So, I chose to use 1/8 step resolution with a 4kHz PWM signal.

Frame Size	42mm
Type	High-Torque Type
Shaft Type	Single Shaft
Motor Lead Wire/Connector Assembly	-
Maximum Holding Torque	0.39N・m
Rotor Inertia J	57×10^-7kg・m^2
Gear Ratio	-
Basic Step Angle	1.8°
Rated Current	1.2A / Phase
Voltage	4.8V
Winding Resistance	4Ω / Phase
Mass: Motor	0.3kg

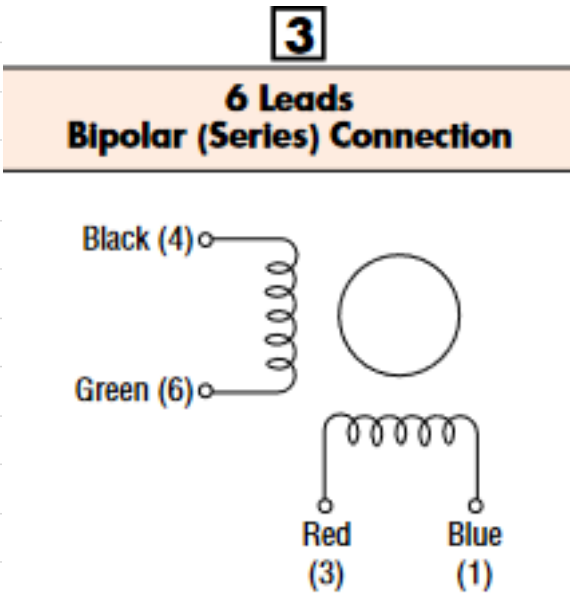


Figure 3.5-2 Motor Specs

Decay mode determines how fast the motor reacts to the steps. In figure 3.5.3, It can be seen that in fast decay mode, the H-bridge that switches works in an inverse manner to slow down the motor. I selected the fast decay mode for more precise movement. Lastly, the Dir pin is set for the direction of the motor if 1 means right 0 is left or vice versa. The initial direction depends on the pin connections.

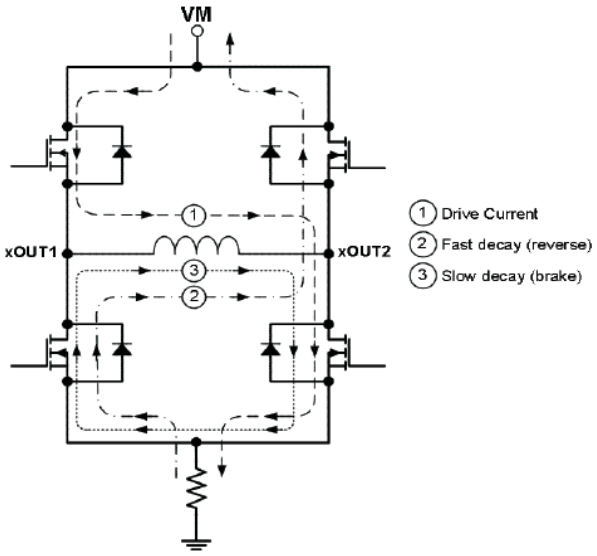


Figure 3.5-3 Decay Modes

I will explain the code by pseudo-code or by explaining the algorithm. Again, due to private intellectual property rights. In the first iteration of the motor control code, there were at

the end 16 iterations, and I created enable and disable functions where I switch enable, sleep, and reset pins as explained above. Another function sets 1/8microstep mode with fast decay called motorInit(). Motor control is done with a timer. 1/8 micro-step is consisting of 32 steps 32*4= 128 rising edges. Since the PWM signal is 4kHz, between each rising edges takes 250us. 360/128*250us*x = time required to travel the desired angle. X is the desired angle.

When the control function is called, the operation above is done with x angle input. Then the timer is set with interrupt and motor enabled. When the timer interrupt is activated under the subroutine motor, disable function is called. This unconventional method will be changed due to timing issues after multiple interrupts and operations are added, function call delays become more significant.

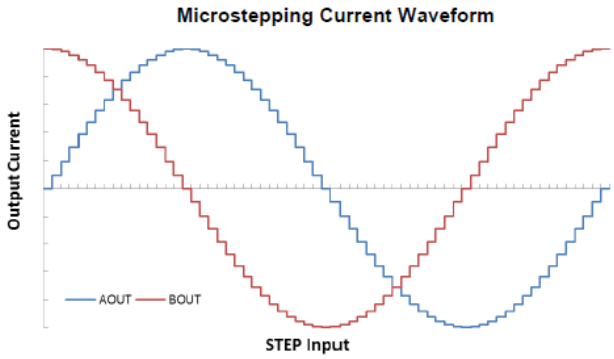


Figure 3.5-3 Microstepping Current Waveform

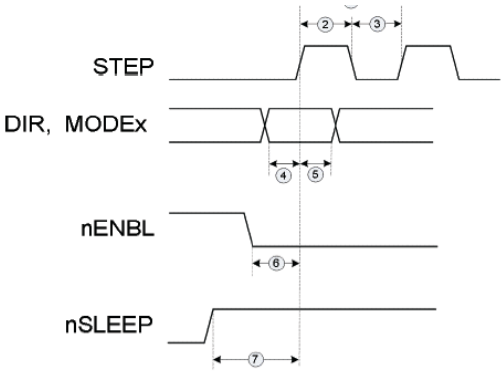


Figure 3.5-4 Timing Diagram

3.6. First Motor Test and Notes on STM32CUBE IDE

I've printed a 360-degree circle and attached a screw to the top of the motor, as shown in Figure 3.6.1 below. I have written a short code to test motor functionality. Whenever the button is pressed, the motor is expected to rotate 60 degrees. Connected the button to a pin as an external interrupt. When the interrupt set motor rotate function called which I explained its functionality earlier. Tested with different degrees of rotation, small angles such as 5 degrees were not accurate as higher values. In the following weeks, the algorithm will be changed to fix all the issues. Another critical issue was the torque. Maximum torque with acceptable speed was the objective since while gas exits the system, it makes valve rotation harder by freezing the valve itself. In figure 3.6.2, one can observe a nonlinear inverse proportion relationship between speed and torque.

Figure 3.6-1 Motor Rotation Test

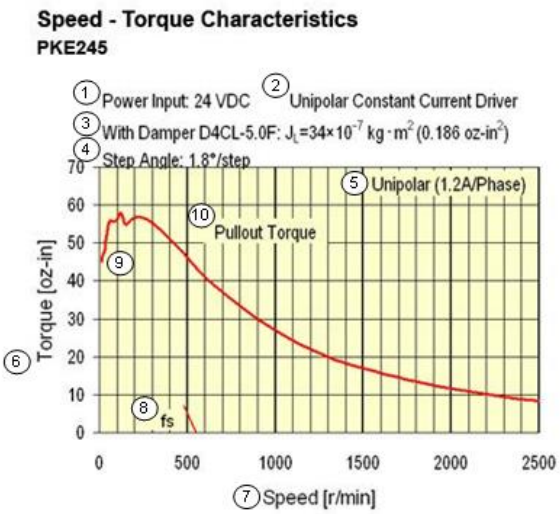


Figure 3.6-2 Speed- Torque Curve

STM32CUBE IDE was the best IDE, Integrated Development Environment, I've ever used. It makes the initialization of parameters incredibly easy. All the STM32 based processors are integrated with the software. One can click on a pin and set it as an ADC, for example. Enable interrupts, change priorities, change the clock speed, or connect external memory. These setup steps can be done manually, but most code doesn't vary from one project to another. Having the code written for you was a great convenience. Another remarkable but problematic thing was the HAL library. It is a library that handles basic tasks such as call functions for interrupt subroutines, timer register initializations, count flag functions, etc.

Most of the time, the HAL library is essential to have a fast and easy-to-read embedded software development process. But there are multiple problems I encountered. Firstly, the library is bad or, in some cases, even not documented at all. In some cases, such as using 1 ADC in multiple pins, the HAL library handles it so that the user loses the data of formerly converted pins and can only read the last pins converted data. Since it is not well documented, it took unnecessary time for me to figure out ADC interrupt flag is set for each conversion, but the library function waits for all of the conversions to end to show the last one's result, ruining the previous data.

4. SECOND WEEK

4.1. Issues with DRV8825

At the start of the second week. While trying to debug why the motor doesn't work correctly burned two motor drivers. Because of the semiconductor crisis, the driver was hard to find. Since we were not happy with the reliability of the driver, its temperature increases incredibly high in short periods of time, and ESD, Electrostatic Discharge, properties of the IC wasn't good enough. Updated and in the stock version of the driver, DRV8425 was ordered, and I changed the PCB to fit the new and old motor controller simultaneously for comparison. Sadly, my Time at SolarBiotec wasn't enough to test and compare the new controller.

4.2. Control Board Prototype Evaluation and Tests

While working on the motor driver found multiple missing capacitors or not connected pins on the designed PCB. Motor driver's capacitors were not connected to the ground. I fixed the problems by adding jumper wires or scratching the PCB surface, and disconnecting pins. Throughout the internship, I got used to altering prototype PCBs and finding solutions with limited equipment. I did know how to solder but haven't soldered many surface mount device (SMD) type components. Reported the changes I made and moved on to write codes for reading each sensor.

4.3. Soldering Exercise by Typesetting Development Boards

In the middle of the week, I soldered components on a clear development board PCB for new interns to work on the same processors. While soldering learned about different techniques to solder different components. Some components have ground pins under them. Paste solder, and a heat gun is used for soldering those kinds of ICs. An applied example is shown in the following figure [7]. It was good practice for me to get better at soldering.

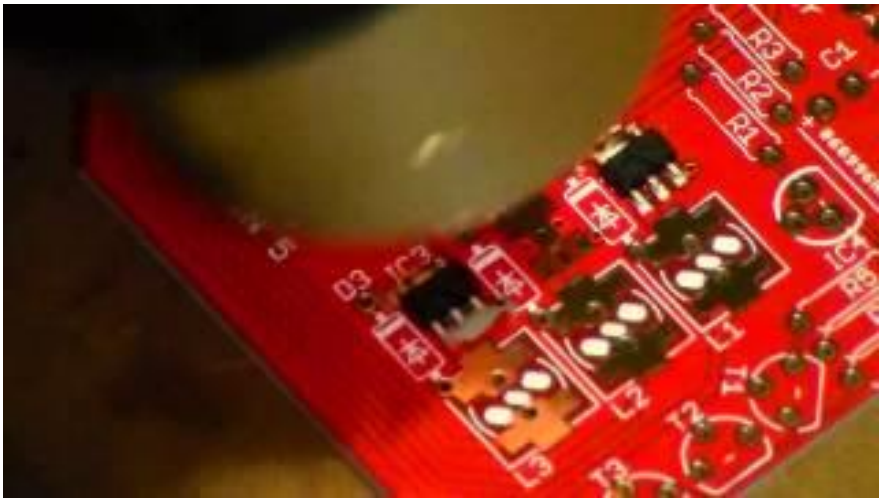


Figure 4.3-1 Soldering with Heat Gun

4.4. Reading Pressure

The old prototype PCB was operated with an Atmel microprocessor which works on 5V. STM32 processors work with 3.3V input. While converting to STM32, some voltages were incorrectly distributed. One of them was the pressure sensor. The pressure sensor was working with 5V. With a 3.3V input, the sensor couldn't measure pressure correctly in the 5000-15 PSI span. A jumper from a 5V pin on the board is connected to the input pin of the pressure sensor. Now the sensor works correctly, but at high pressures, it outputs voltages higher than 3.3V. STM32 processor cannot convert voltages to digital higher than 3.3V, and voltage high as 5V damages the input pin.

I have connected a voltage divider with two resistors to change the 5V-0V span to 3.3V-0V. And buffered the output with a voltage follower opamp. The circuit is shown in figure 4.4.1 below. I constructed a small detachable circuit using a bakelite perfboard. The processor constantly reads and converts the pressure sensors output to PSI value under interrupt subroutine.

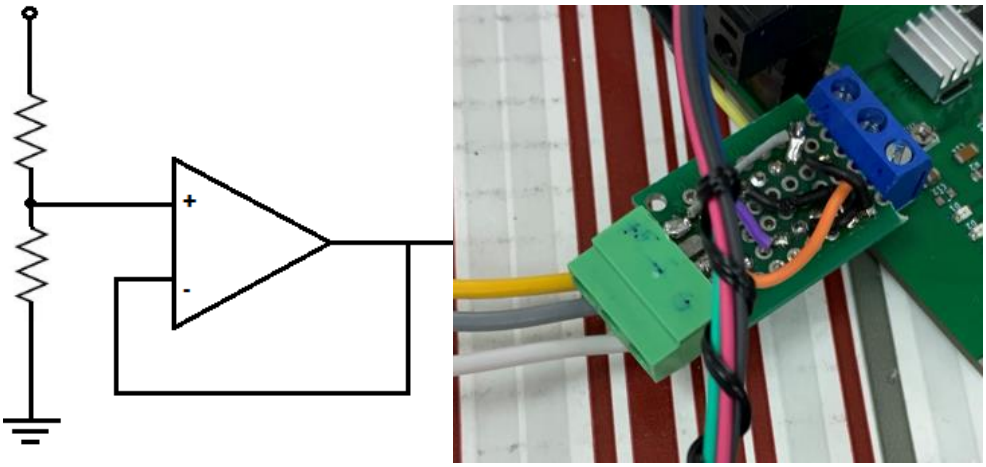


Figure 4.4-1 Voltage Follower

The valve closed manually for the experiments, and the CO2 tank is connected through the 200ml chamber in figure 4.4.2. The extraction machine will work at high pressures such as 5000 PSI, but gas tanks can hold a maximum of 1000psi. Datasheet of the sensor shows a linear voltage pressure relationship between 5000psi to 500psi. Manually measured voltages and pressure under 500psi and in MATLAB equated the curve. Added it to the code enabling the sensor to precisely read atmosphere pressure (14.7) to 5000 PSI. Between 0.5 to 5V or after the voltage division on the processor side, 0.33V to 3.3V converted to 410 to 4095 integer values through 12-bit ADC. For the values under 410 previously explained equation is used.

Figure 4.4-2 CO2 Tank

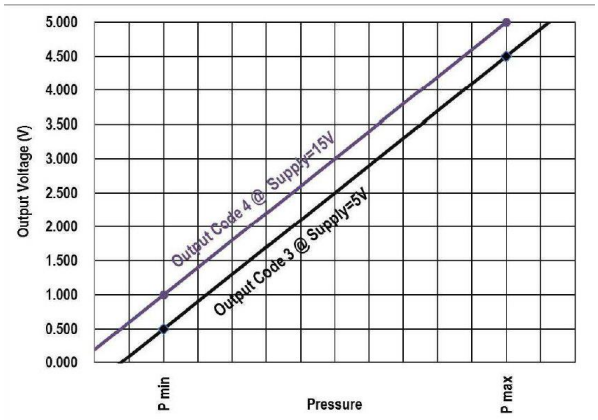


Figure 4.4-3 Pressure Transfer Curve

4.5. Infrared Sensor and Button Reads

The infrared sensor has an emitter and transmitter side. If the connection between them is blocked with a solid, it outputs 0V. Else outputs input voltage. The button works similarly by creating a physical connection when pressed, outputting input voltage and 0V when depressed. Both button and infrared mini boards were developed for the 5V system. By decreasing resistors and disconnecting, LEDs made them work in a 3.3V system.

Figure 4.5-1 Infrared Sensor Board

4.6. Optical Encoder

Motor control had still flawed. Firstly, when the motor tightly seals the valve, it couldn't open it back with multiple small degrees of rotation. For example, when the motor tries to open a tightly sealed valve by 30degrees at once, it can easily open it but sometimes turns 25 degrees or less because of the strain to loosen the valve. The worst case is making the motor turn 5 degrees 6 times a total of 30 degrees. In this case, the motor couldn't open the valve at all. 5-degree rotation didn't have enough force to rotate the valve and loosen it. Ideally, it moves just a fraction and puts stress on the valve but moves back to the exact location when the movement ends. The reason for this was that in the motorStop function, we activate sleep, decreasing current flow to the motor. As a result, its position can be easily changed by outside effects. Keeping stepper motors idle is the most current drawing case. In the application we aim, motor idles most of the time, and motor driver reaches temperatures that damage itself in short periods of time.

The first approach to the problem was to add a basic optical encoder and change the infrared sensor's purpose. An optical Encoder is a combination of a disc with small slits outside of it and an infrared sensor. In figure 3.6.1, 3D printed disc can be seen. Made the disc with a 15-degree resolution. After each time motor rotation function called the encoder starts counting spaces. When the distance traveled and inputted angle exceeds 15 degrees, the motor rotates back or forward by relying on the encoder. Most of my solutions until now weren't ideal. I'm happy that SolarBiotec engineers helped me throughout my internship only when I asked them and didn't show me the solutions directly.

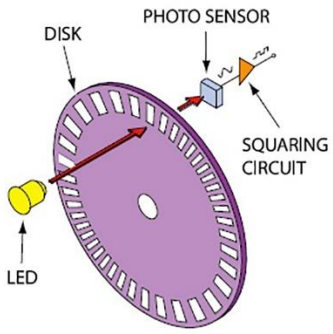


Figure 4.6-1 Rotary Encoder [8]

5. THIRD WEEK

5.1. Pressure PID Control

In the first two weeks, I migrated old sensors and motor control to the STM32 system. After passing the functionality tests, I conducted, I started working on PID control. With a mechanical engineer intern, we constructed a simulation environment inside Simulink using Simscape tools. In figure 5.1.1 block diagram of the environment can be seen. Parameters of each block are measured or taken from datasheets of mechanical equipment such as valves and chambers to create simulation close to real application. Then a feedback system was added to valve control and again, using MATLAB calculated Kp, Kd, Ki values.

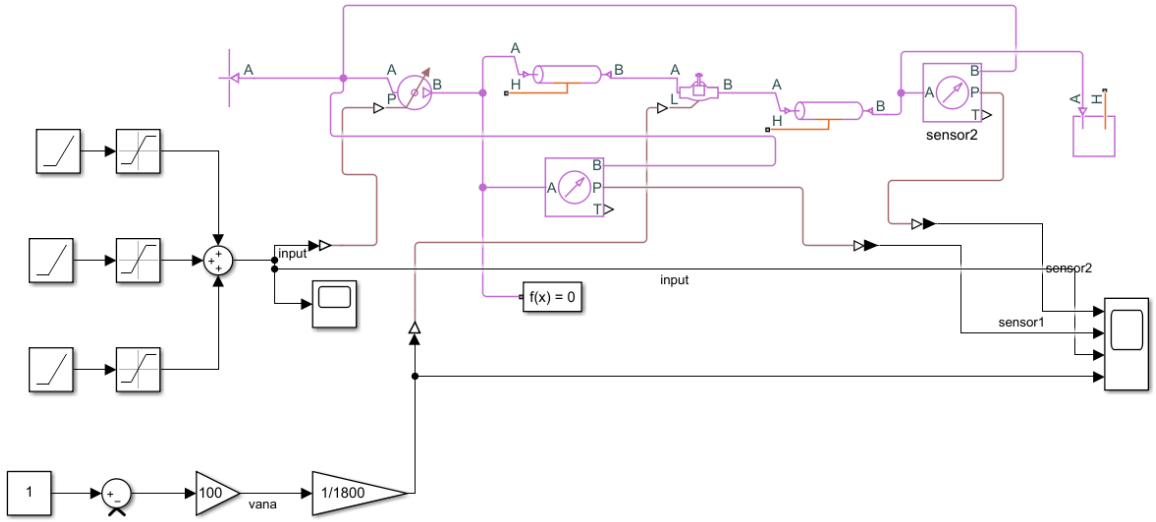


Figure 5.1-1 Pressure Regulator Simulation on Simulink

After simulating and comparing data with physical tests, I started to write a PID code. PID function runs every 0.01 seconds. As I will explain later, while controlling temperature, the mechanical engineer who worked on it advised me to do it this way. In temperature control heater works with AC signal with 50Hz frequency. I kept the period half the AC network so I could do both PIDs in one function. Two timers are set at the start of the function. One counts how long the function itself takes, which is a lot less than 0.01 seconds. It was 100 to 150 cycles in a 72MHz clock which corresponds to approximately 2us. The time spent on time will be important in AC signal control while working on temperature. The second timer is set for 0.02 seconds. When done, the interrupt is triggered and calls the PID function again. Kp, Kd, Ki values were set to 2, 0.08, 0.03. A global variable two-element array is set for storing error recent and previous error values. With pressure value as input motor rotate angle is outputted aiming desired pressure. The function is shown in figure 5.1.2. The effect of the previous value is decreased by a commonly used method called taking a rolling average. In the code, both PD and PID methods have been experimented with. As discussed earlier, we couldn't experiment on higher pressures, and it made integral parameter tuning arbitrary since pressure decrease in higher pressures cant be reflected by working on lower pressures. Eventually, PID control will be the best fit for regulating pressure.

```
void motorPID(float pressure){
    globalPressureFlag = 0;

    timeElapsed = 1.0/cycleWithoutPID;

    /* PID Calculation */
    error_2[0] =  error_2[1];
    error_2[1] = (desiredPressure - pressure);

    Ui_2_Temp = Ui_2;
    Up_2 = error_2[1] * Kp_2; //120
    Ud_2 = (error_2[1] - error_2[0]) * Kd_2 / timeElapsed; //-300
    Ui_2 = Ui_2 + (Ki_2 * error_2[1] * timeElapsed ) ;  //?

    //PID
    //globalCurrentAngle = (globalRotateAngle * 9 + (Up_2 + Ui_2 + Ud_2)) / 10;

    //PD
    globalCurrentAngle = (globalCurrentAngle * 9 + (Up_2 + Ud_2)) / 10;

    //PD
    //globalCurrentAngle = (Up_2 + Ud_2);

    globalCurrentAngle = (globalCurrentAngle > 0) ? 0:globalCurrentAngle;

    Ui_2 = (globalCurrentAngle> 360) ? Ui_2_Temp: Ui_2;

    remainingAngle = globalCurrentAngle - globalTotalAngle;
```

Figure 5.1-2 Pressure PID C Code

The motor rotates and, while rotating, changes globalCurrentAngle. This parameter shows how many degrees the valve is traveled away from the closed position. As an illustration, after PID calculations angle is set to +50 degrees to decrease pressure. Then in a new calculation, after 0.02 seconds with lesser pressure read, the new angle is set to +30 degrees. Motor control function takes remainingAngle and rotates -20 degrees. Eventually, the valve is closed when desired pressure is set, or with constant pressure, increase stabilizes in a particular valve gap.

I used USB communication for sending data to a plotting software they developed called GalvanoPlot. It is used for test equipment called galvanometer, which they develop. We tricked the software and made the data readable by the software for plotting graphs. Figure 5.1.3 above wave is the pressure data, and the below negative one is the globalCurrentAngle parameter which shows how much the valve is opened. We set desired pressure to a certain value and opened the gas tank excessively. In milliseconds valve stabilized the pressure to the desired value.

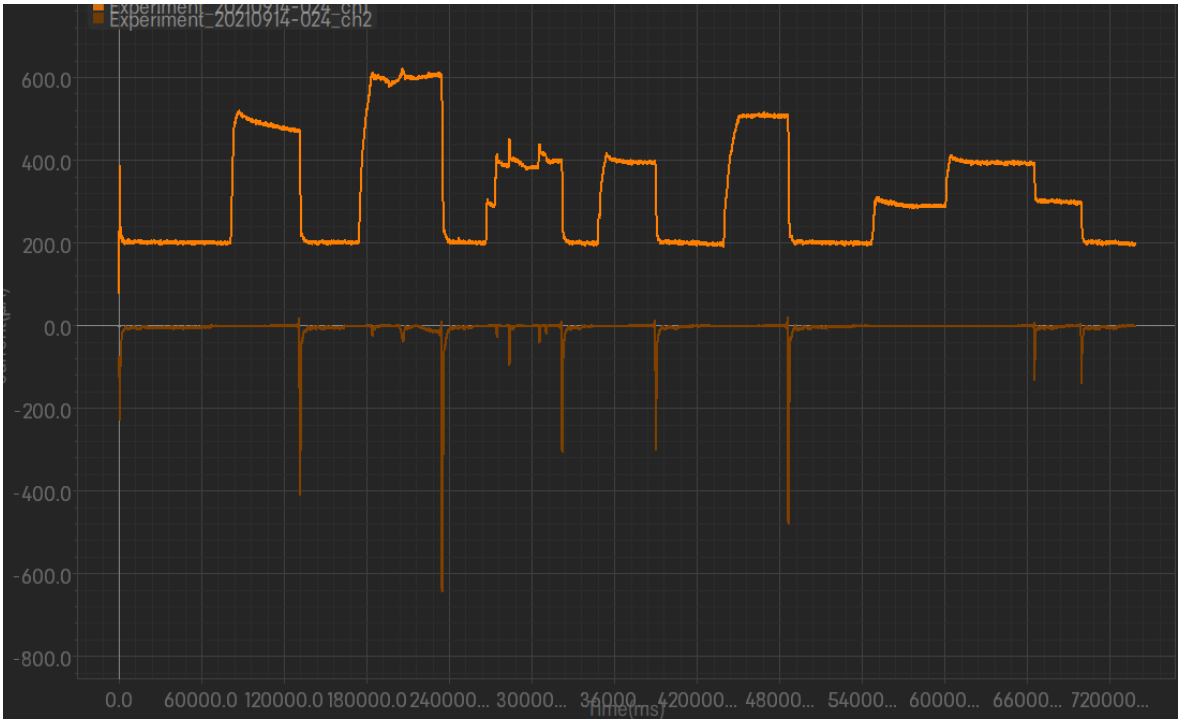


Figure 5.1-3 Pressure PID Graph

5.2. New Valve Prototype

The belt-driven motor started causing problems such as: when the valve is opened, it increases in height, and the belt comes out of its channel. The old prototype took a lot of space and material to produce. An aluminum block surrounds the new regulator system valve with slots drilled for heaters. The motor is connected directly to the valve with a thing called a coupler. (Shown in figure 5.2.1) The coupler has a rubber part at the middle that connects two metal ends to each other even when the distance between ends increases. At this point, I was working on a better way to control the motor, and we got rid of the encoder idea.

Figure 5.2-1 Coupler

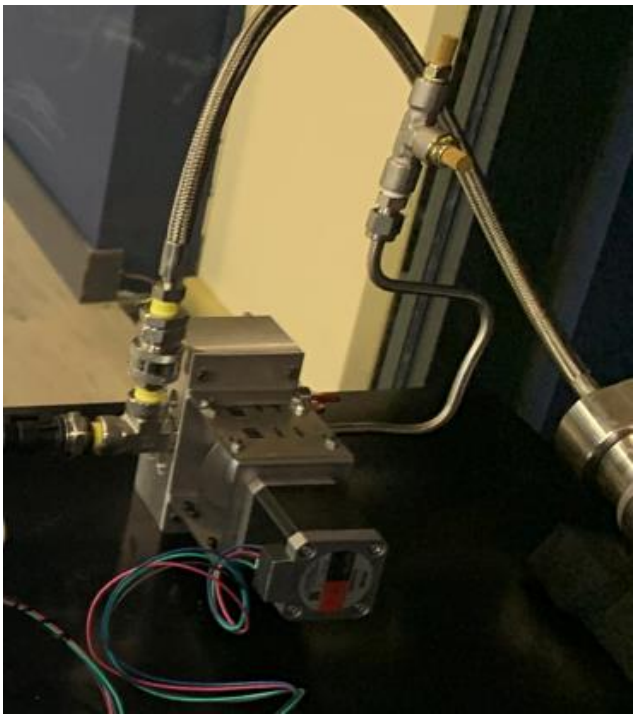


Figure 5.2-2 New Regulator Prototype

5.3. Temperature Sensor Reading

Temperature control of the valve was another essential part of my assignment, preventing the valve from freezing. I worked with a mechanical engineer intern in this part of the assignment. He was working on another project that required precise temperature control. His "know-how" accelerated the design and trial-error process. We used a sensor called Pt-100, a temperature sensor. This sensor essentially is a resistor when heated, increases its resistance, and decreases when cooled. Sensors' room temperature resistance is around 100 Ohms. MAX31865 IC is used to read the temperature. This component uses the SPI protocol. First, it should be calibrated with known resistance. The relationship between resistance and temperature is close to linear between 0^0C to 100C. Callendar-Van Dusen equation is used to convert resistance to temperature or vice versa.

$R(T) = R0(1 + aT + bT^2 + c(T - 100)T^3)$

T = temperature (^{0}C)
R(T) = resistance at T
R_0 = resistance at T = 0^0C
$\alpha = 0.00385055$
$a = 3.90830 \times 10^{-3}$
$b = -5.77500 \times 10^{-7}$
$c = -4.18301 \times 10^{-12}$ for -200^0C < T < 0^0C, 0 for 0^0C < T < $+850^0$C[5]

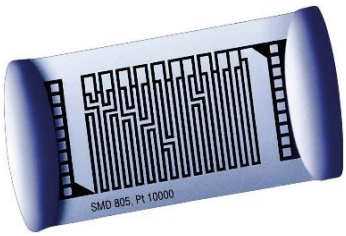


Figure 5.3-1 Pt 100 in SMD805 Package [9]

5.4. Switching to STM32F303 Processor

I discussed earlier that using one ADC with multiple pins caused problems. We switched the way we connect the infrared sensor from an external interrupt to an ADC Watchdog interrupt. In fast rotations, the infrared sensor is stuck in between logic 0 and logic 1. Sometimes it falls between noise margins of the processor, creating a false zero or one. Analog watchdog interrupt is triggered when the ADC value exits compare register values. I set it to 3V-3.3V at first. When the interrupt is called program changes the gap to 0 to 0.5V, the next time it is called changes the gap back to 3V-3.3V. This way, I manage to find a solution to the sensor's reliability issue. For temperature control, I needed another watchdog, but F373 only had one channel to work with. I tried the DMA method, a structure in STM32 processors that connects I/O directly to memory without disturbing the program. Values are read from memory directly. But for my case reading and comparing it manually wasn't fast as internal hardware, and I missed some values. STM32F373 is the most used microprocessor in the company, and there was a shortage of this processor as well; besides, it doesn't fit the requirements for my application. We had an STM32F303 processor in stock, and it had three separate ADC modules built-in. Since building and printing a new PCB for the F303 microprocessor would delay the development process, with the help of Mr. Kan, electronics engineer, we fitted F303 on a PCB designed for F373. We crossed multiple legs of the processor and used jumper cables to fit it into the spot. Figure 5.4.1 shows the board after installation. I updated the drawings of the board after successfully installing the new microprocessor.

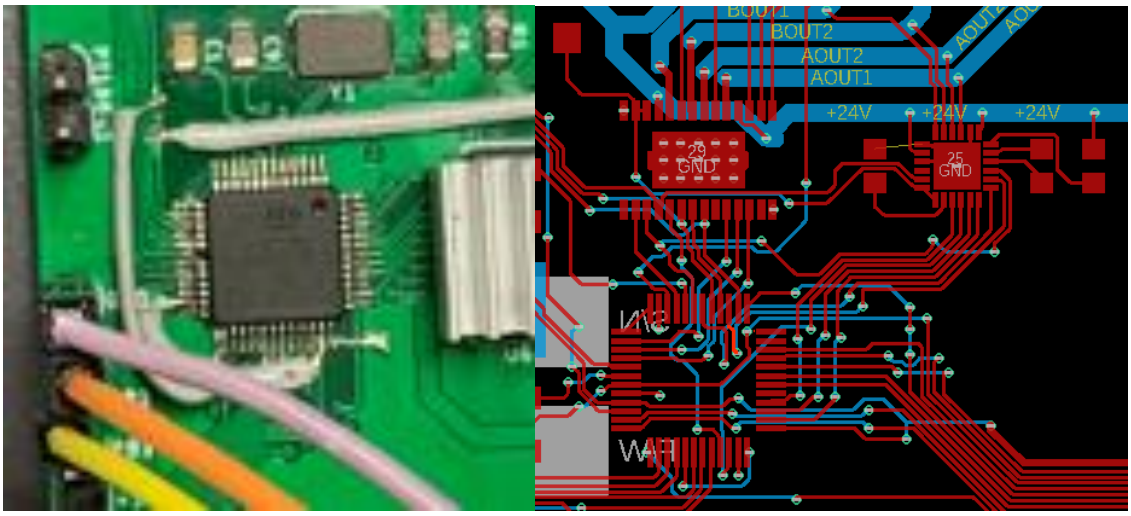


Figure 5.4-1 F303 Installation and CAD Drawings

6. FOURTH WEEK

6.1. Temperature PID Control

My last week was dedicated to solving the freeze problem. They were using 100W Cartridge type resistances. Heat on these resistances are controlled with an AC board. PCB has a full wave rectifier on it, and an IC digitalizes the signal. AC 220V(RMS) signal is converted to a DC signal, as shown in yellow in figure 6.1.2. Even though the circuit works on 5V input, it outputs waves with a maximum of 2.3V. In figure 6.1.1 blue connector is connected directly to AC, and the black connector is connected to the resistance. Brown connector consists of the yellow signal output discussed above as inputs: drive signal, 5V source, and ground to the circuit from the main control card. A Drive signal is a pulse signal generated from the microprocessor according to temperature value and digitalized AC signal. When the drive pin is set to bridge conducts until the next zero-cross point. If one wants to provide 100% power to the heater drive signal should be sent right after each zero-cross point. A lookup table is made for phase angles to which percentage it drives the heater load.

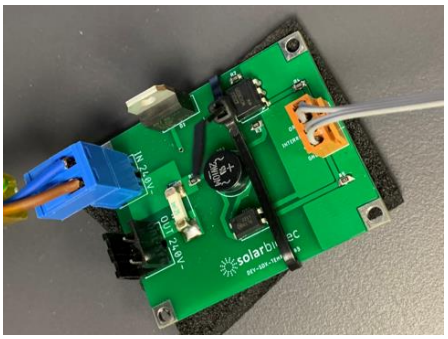


Figure 6.1-1 AC Control PCB

I have written a code similar to the pressure control code. Another engineer already did PID constant calculations while I was working on motor controls. I use an analog watchdog to detect zero-cross locations, which approximately occur every 0.01 seconds. When interrupt subroutine is called, I run both PID functions and start a timer. The timer counts both PID calculations to end right before exiting the function stores the time passed.

Figure 6.1-2 AC Control Card Signals

Temperature PID control algorithm takes temperature as input and creates an output between 0% to 100% power. As an example, we desired to keep the valve temperature at 40degrees always. When the valve opens, and it drastically decreases the temperature. At first, the heater works at 100% and decreases when it gets closer to the desired 40degrees. We manipulated the integral component to get rid of overshoots. Whenever the heater works at 100%, we equate the integral component to 0, but when the power decreases, we include the integral component back so the coefficients don't increase or decrease unnecessarily. After having the power percentage using the lookup table, we find the phase angle and, with that, time to wait until to send the drive pulse signal. If we want to send %50 power, we wait 0.005 seconds. We exclude time spent on calculations approximately 2us and set a timer. When timer interrupt is set, we send a short pulse signal, and the code waits for the next zero-cross to appear.

Figure 6.1-3 Cartridge Resistance

6.2. New Motor Control Algorithm

After the new valve design was manufactured, I went back and started to find new ways to fix angle precision problems. I constantly supplied clock or STEP signal in this case to the motor driver and operated it using enable, sleep, and reset signals. I changed the priorities of interrupts in a way that prevented missing zero cross points in temperature control. I created an interrupt subroutine that is triggered whenever the PWM signal rises. I store each rising edge, and the function disables the PWM signal when the desired angle is set. With this new algorithm, stress on the microprocessor decreased, and the program's reliability increased significantly. I still had issues with opening tightly closed valve.

6.3. Using DAC with Motor Controller

A potentiometer is connected to the pins of the motor driver to determine the maximum current flow. 3.3V supply converts into 1A for this motor. The motor should rotate in high currents to have torque, but if the current limit is set to high values while idle, the driver heats and burns. I disconnected the potentiometer and, with a jumper, connected digital to analog converter pin to the input of the driver. With this new setup, I changed the motor control function. When the motor rotates, I provide 3.3V, and the motor rotates in 1A other, and when the motor is on idle, function changes DAC output to 0.3V and motor drains only 0.3A. It fixed all the issues on tightening and loosening the valve.

6.4. Temperature Read Without MAX31865 IC and TUBITAK Report

In this project, we weren't utilizing all the capabilities of the MAX31865 IC. It was expensive for the purpose, so they assigned me to read temperature without this component. I constructed the same voltage follower circuit I used in the pressure sensor. (Figure 4.4.1) The resistor below, one that connected to opamp, is where the PT100 temperature sensor was connected. (Figure 6.4.1)

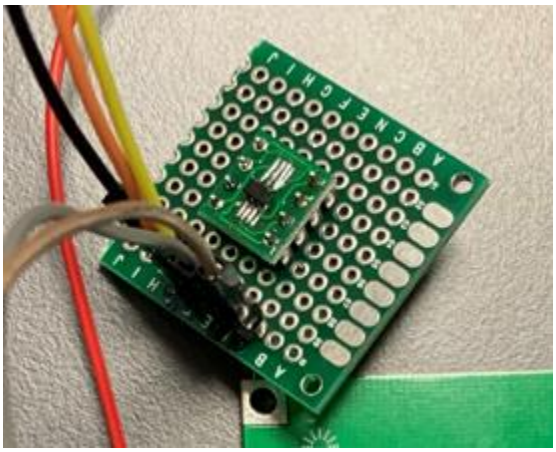


Figure 6.4-1 Voltage Follower for Temperature Sensor

I connected a 100Ohm resistor and measured voltage from the microprocessor. Now that I have my reference value by using the Callendar-Van Dusen equation that I explained in part 5.3, I converted the resistor value that I converted from the voltage to temperature. It wasn't as stable as MAX31865 IC, but with added filters, noises can be eliminated.

I've spent my last days helping engineers to prepare the report of parts I have done in my internship for the Tubitak presentation coming. We conducted multiple experiments and reported the results. Putting my knowledge from laboratory work on preparing reports and experimenting to practice was a great experience.

7. CONCLUSIONS

To conclude my report, I spent one month in SolarBiotec but learned and gained experience worth maybe six months. Having the opportunity to work as similar as a graduated engineer and having responsibilities was an excellent experience for me. I was interested in microprocessors and embedded systems before my internship, and it was the perfect fit for my interests. Before the internship, I had a little bit of knowledge on driving motors and STM32 architecture, which helped me through my internship.

I made a prototype from start to finish with only guidelines and an older version to be inspired with. Finding my own solutions and researching and every missing part of my knowledge on my own enhanced my understanding of the subjects much more than if someone directly showed me the solutions or handed me the information needed. I am happy that I contributed significantly to the final product. I learned how to document progress, how to keep version control of a code, and many other embedded system engineering skills. I got more confident in my C language skills and microprocessor architecture knowledge.

Shortly with this internship, I gained valuable experience in the area that I am most interested in.

8. REFERENCES

[1] (n.d.). SolarBiotec. https://www.solarbiotec.com/

[2] (n.d.). ClickUp™ | One app to replace them all. https://clickup.com

[3] *STM32F373xx Datasheet*. (n.d.). ST.

[4] *DRV8825 Manual*. (n.d.). Texas Instruments.

[5] *MAX31865 RTD-to-Digital Converter*. (n.d.). Maxim Integrated.

[6] *2-Phase Stepping Motors PK Series Datasheet*. (n.d.). Oriental Motors.

[7] Chris Thomas. (n.d.). *How to solder an SMD using a cheap heat gun* [Video]. YouTube. https://www.youtube.com/watch?v=4OYakUQmgd0&t=19s

[8] Schweber, B. (2019, April 11). *Rotary encoder basics and applications, Part 1: Optical encoders*. Analog IC Tips. https://www.analogictips.com/rotary-encoders-part-1-optical-encoders/

[9] 2020 Conrad Electronic. (n.d.). *Heraeus Nexensos SMD 0805 V PT100 platinum temperature sensor -50 up to +130 °C 100 Ω 3850 ppm/K SMD tape cut*. Conrad Electronic » Your Sourcing Platform. https://www.conrad.com/p/heraeus-nexensos-smd-0805-v-pt100-platinum-temperature-sensor-50-up-to-130-c-100-3850-ppmk-smd-tape-cut-181219

9. APPENDIX

9.1. Motor Driver Microstep Table

1/32 STEP	1/16 STEP	1/8 STEP	1/4 STEP	1/2 STEP	FULL STEP 70%	WINDING CURRENT A	WINDING CURRENT B	ELECTRICAL ANGLE
1	1	1	1	1		100%	0%	0
2						100%	5%	3
3	2					100%	10%	6
4						99%	15%	8
5	3	2				98%	20%	11
6						97%	24%	14
7	4					96%	29%	17
8						94%	34%	20
9	5	3	2			92%	38%	23
10						90%	43%	25
11	6					88%	47%	28
12						86%	51%	31
13	7	4				83%	56%	34
14						80%	60%	37
15	8					77%	63%	39
16						74%	67%	42
17	9	5	3	2	1	71%	71%	45
18						67%	74%	48
19	10					63%	77%	51
20						60%	80%	53
21	11	6				56%	83%	56
22						51%	86%	59
23	12					47%	88%	62
24						43%	90%	65
25	13	7	4			38%	92%	68
26						34%	94%	70
27	14					29%	96%	73
28						24%	97%	76
29	15	8				20%	98%	79
30						15%	99%	82
31	16					10%	100%	84
32						5%	100%	87
33	17	9	5	3		0%	100%	90
34						–5%	100%	93
35	18					–10%	100%	96
36						–15%	99%	98
37	19	10				–20%	98%	101
38						–24%	97%	104
39	20					–29%	96%	107

1/32 STEP	1/16 STEP	1/8 STEP	1/4 STEP	1/2 STEP	FULL STEP 70%	WINDING CURRENT A	WINDING CURRENT B	ELECTRICAL ANGLE
40						–34%	94%	110
41	21	11	6			–38%	92%	113
42						–43%	90%	115
43	22					–47%	88%	118
44						–51%	86%	121
45	23	12				–56%	83%	124
46						–60%	80%	127
47	24					–63%	77%	129
48						–67%	74%	132
49	25	13	7	4	2	–71%	71%	135
50						–74%	67%	138
51	26					–77%	63%	141
52						–80%	60%	143
53	27	14				–83%	56%	146
54						–86%	51%	149
55	28					–88%	47%	152
56						–90%	43%	155
57	29	15	8			–92%	38%	158
58						–94%	34%	160
59	30					–96%	29%	163
60						–97%	24%	166
61	31	16				–98%	20%	169
62						–99%	15%	172
63	32					–100%	10%	174
64						–100%	5%	177
65	33	17	9	5		–100%	0%	180
66						–100%	–5%	183
67	34					–100%	–10%	186
68						–99%	–15%	188
69	35	18				–98%	–20%	191
70						–97%	–24%	194
71	36					–96%	–29%	197
72						–94%	–34%	200
73	37	19	10			–92%	–38%	203
74						–90%	–43%	205
75	38					–88%	–47%	208
76						–86%	–51%	211
77	39	20				–83%	–56%	214
78						–80%	–60%	217
79	40					–77%	–63%	219
80						–74%	–67%	222
81	41	21	11	6	3	–71%	–71%	225
82						–67%	–74%	228
83	42					–63%	–77%	231
84						–60%	–80%	233
85	43	22				–56%	–83%	236
86						–51%	–86%	239

1/32 STEP	1/16 STEP	1/8 STEP	1/4 STEP	1/2 STEP	FULL STEP 70%	WINDING CURRENT A	WINDING CURRENT B	ELECTRICAL ANGLE
87	44					–47%	–88%	242
88						–43%	–90%	245
89	45	23	12			–38%	–92%	248
90						–34%	–94%	250
91	46					–29%	–96%	253
92						–24%	–97%	256
93	47	24				–20%	–98%	259
94						–15%	–99%	262
95	48					–10%	–100%	264
96						–5%	–100%	267
97	49	25	13	7		0%	–100%	270
98						5%	–100%	273
99	50					10%	–100%	276
100						15%	–99%	278
101	51	26				20%	–98%	281
102						24%	–97%	284
103	52					29%	–96%	287
104						34%	–94%	290
105	53	27	14			38%	–92%	293
106						43%	–90%	295
107	54					47%	–88%	298
108						51%	–86%	301
109	55	28				56%	–83%	304
110						60%	–80%	307
111	56					63%	–77%	309
112						67%	–74%	312
113	57	29	15	8	4	71%	–71%	315
114						74%	–67%	318
115	58					77%	–63%	321
116						80%	–60%	323
117	59	30				83%	–56%	326
118						86%	–51%	329
119	60					88%	–47%	332
120						90%	–43%	335
121	61	31	16			92%	–38%	338
122						94%	–34%	340
123	62					96%	–29%	343
124						97%	–24%	346
125	63	32				98%	–20%	349
126						99%	–15%	352
127	64					100%	–10%	354
128						100%	–5%	357

MIDDLE EAST TECHNICAL UNIVERSITY NORTHERN CYPRUS CAMPUS

ELECTRICAL AND ELECTRONICS ENGINEERING PROGRAM
EEE – 400
Summer Practice Report

Student Name:	**Barış Güzel**
Student ID:	**2315935**
Summer Practice - Start Date:	**18.07.2022**
Summer Practice - End Date:	**16.08.2022**
Report Submission Date:	**15.11.2022**
Company Name:	**TOFAŞ Türk Otomobil Fabrikası A.Ş**
Company Division/Department:	**E&E Entegrasyon**
Supervising/Mentoring Engineer Name and Professional Title:	**Ilker Acarlar** **E&E Entegrasyon Yöneticisi**
Company Address:	**İstanbul Cad. No: 574 16110, Bursa,TR**
Field of Practice in Electrical and Electronics Engineering	**Embedded Software**

TABLE OF CONTENTS

1. INTRODUCTION

I have done my EEE 400 internship in TOFAS for one month. It is a company that is part of the Stellantis group. This internship was part of my onboarding process with the TOFAS company as an embedded software engineer. I followed a newcomer engineer experience rather than a simple internship. I spent time in E&E Integration teams and all sub-teams individually and learned their workflow and responsibilities. This helped me to understand my probable job title in TOFAS. It was an experience that benefited both parties in getting to know each other.

During my internship, between 18.07.2022 and 16.08.2022 took notes and pictures to prepare this summer internship report. With each sub-team, I did little mock tasks that were already done and simple for me to understand the workflow. Spending time with senior engineers who had worked with most of the teams before was helpful. It motivated me to apply my theoretical knowledge to practice by doing as many mock tasks as possible. Trying and failing often taught me many problem-solving techniques and easy and cheap ways to fix problems. In the report, AUTOSAR architecture and workflow go along with it in every different sub-team is detailly explained.

2. COMPANY DESCRIPTION

2.1. A brief history of the company

TOFAS is an automotive company that is part of the FCA group, which recently merged with the PSA group under the Stellantis name. Stellantis comprises more than ten car companies such as Fiat, Jeep, Lancia, Peugeot, Abarth, Alfa Romeo, Ram, Citroen, Maserati, etc. TOFAS was founded in 1968 in Bursa, Turkiye, and started the production of Murat 124 in 1971. In the 80s and 90s, TOFAS produced many entry-level, accessible cars for the Turkish market and exported them worldwide. Factory transitions to fully producing vehicles, engine to chassis, in the mid-90s with the Tempra model.

In 1994 TOFAS R&D was founded and started working on global development projects with FCA and Fiat Chrysler Automobiles. Since the 2000s, Fiat Doblo and Fiorino have been Turkey's most popular commercial vehicles. TOFAS is responsible for Maserati, Alfa Romeo, and Jeep's operations in Turkey.

Mission: To improve people's quality of life by providing them with the products and services that best suit society's need for mobility.

Vision: To be a leading automotive company that shapes customer expectations and is a source of pride.

2.2. Main Area of Business

TOFAS is an automotive company that produces and develops Stellantis brands' cars. In the R&D - E&E Integration department, embedded system designs, communications, hardware designs, and testing of all components and systems are done.

2.3. Organizational Structure of the Company

TOFAS is a company that the KOC Group and FCA half own. The Board of Directors is assembled with both KOC and FCA representatives. TOFAS, as touched on before, is a factory that is part of a global corporation, Stellantis. I had my internship in the Electric and Electronics Integration division under R&D Department. Figure 2.3-1 shows the summarized organization inside TOFAS. The company enables university students to get field experience by hiring multiple interns throughout the year. Interns would have varied responsibilities depending on the workload. I mainly spent my time in software subdivisions where embedded software's developed.

Cengiz Eroglu is the CEO and Giuseppe Masciocco is R&D Director. Ali Sengur is E&E Director, and Ilker Acarlar is E&E Integration Manager. I was directly in touch with Ilker Acarlar.

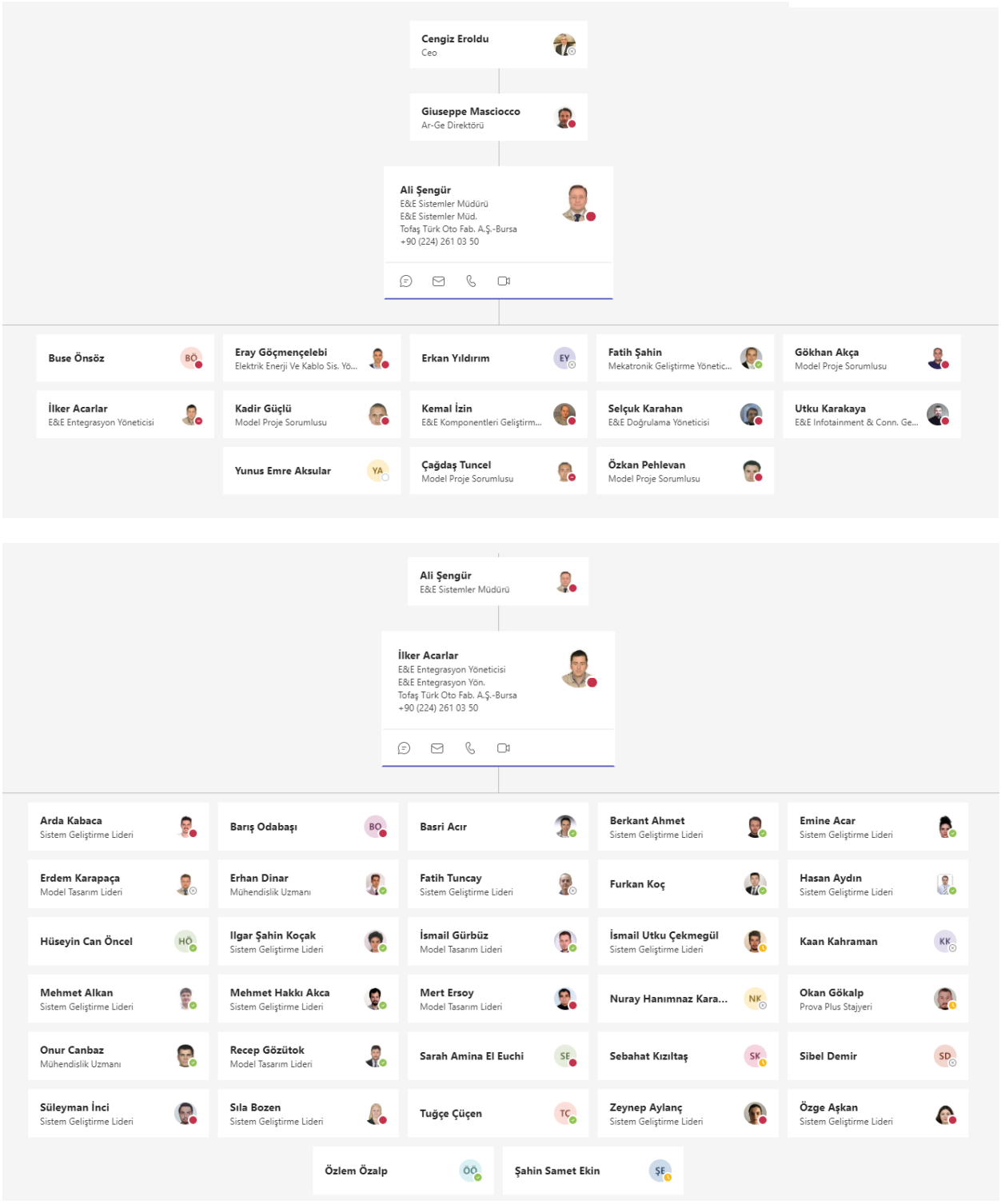


Figure 2.3-1: Organizational structure

3. FIRST WEEK

3.1. Orientation and Occupational Health and Safety Training

Many factories conduct training seminars and courses before starting an internship or job. Safety officers explained the risks in the work environment and how to avoid life-threatening incidents for the first two days. A health checkup and short factory tour are done on the second day. Then I meet with the team with whom I will be doing my internship. Mr. Acarlar, Integrations Manager, introduced me to the team.

3.2. Communications Sub team Training in CAN and LIN Coms

The TOFAS factory has training facilities for engineers and blue-collar workers. TOFAS Academy is mainly used for on-job trainings for newcomers. I spent most of my time on on-job training activities, in short OJT. Since, as I explained before in the introduction chapter, the objective was to do my orientation before starting to work as a software engineer while doing my internship. The communications sub-team did the first OJT. In communications team is responsible for supplying signals connection diagrams to other teams.

In today's cars, many different communication topologies are being used. There are at least ten different ECUs inside each car. These modules are connected with different connections, sometimes with more than one. The most common and cheaper options are LIN and CAN.

LIN is primarily used in low bandwidth-required networks. It can support bandwidths around 20KBps. In fiat cars, it is used in controlling doors, start-stop buttons, and pumps. CAN connection has two classes, one slower, which supports a maximum of 125KBps, and faster, which supports 1MBps. Better connection types like ethernet, Flexray, and MOST networks are used for higher bandwidth required applications. In TOFAS, none of the current cars are using these topologies, but research is being done in R&D. Mostly, a single bus topology is used, as shown in the figure below. Gateway modules, sometimes integrated into more extensive modules, are used to transfer data between different buses. Sometimes in more complex systems, repeaters are used.

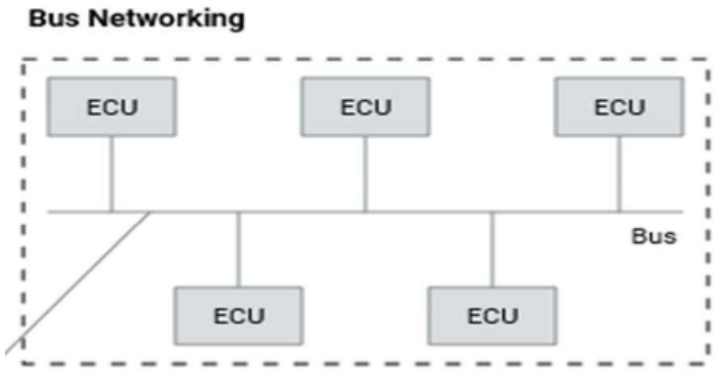


Figure 2 CAN BUS Network

LIN connection is similar to UART. It is mainly connected as a full duplex, and only one type of data is transmitted at a time on the bus. One ECU takes control of the bus and receives or transmits data. When the transmission is finished, another ECU can take control of the bus. Since this type of connection limits the throughput, ECUs can be organized and daisy-chained to create tree-like structures.

CAN fix the single bus single data issue and supports greater bandwidths. In CAN, data is packed inside a frame, and each frame has different arbitration fields to define what the message carries and from which ECU it comes. There is the start, stop bits, and CRC-generated bits to be checked on the receiving side for the integrity of the message in the CAN frame. In automotive applications, even signals within the controller units are CRC-checked. CAN is a two-wire connection, CAN High and CAN Low, that terminates with 120Ohms at both ends of the single bus. Resistors are primarily integrated into the I/O module of the ECUs connected at the end of the bus lines. With CAN, different types and lengths of data can be transmitted through the bus. Each module can read all frames and take the data that it requested from another module.

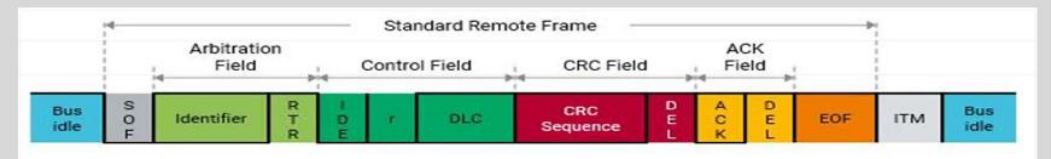


Figure 3 CAN Message Format

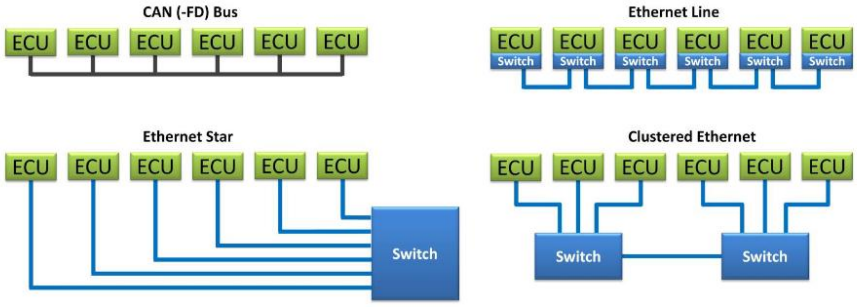


Figure 4 Different types of network topologies: Bus, Daisy chain, Star, Cluster

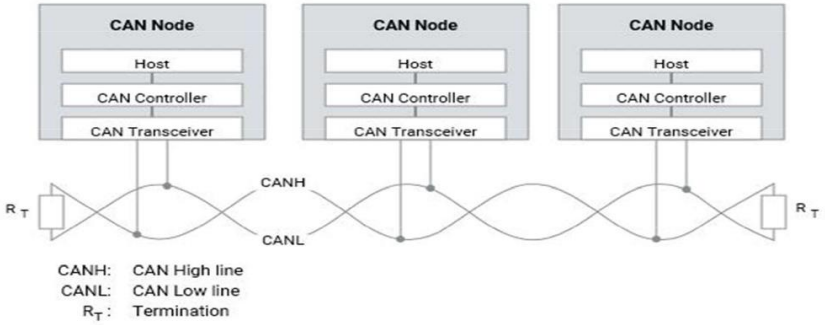


Figure 5 CAN BUS connection

Most of the data can't be fitted inside of one frame. In TOFAS, one data block is split into eight frames and transmitted. These frames don't have to be sent consecutively. Different frames can be present in the bus in between the parts of the data. In some applications, there can be a module that organizes these frames. The most easily applied type is comparing signals bitwise from most significant to least significant. 0 has priority over 1. For example, there are three different signals with different arbitration fields: 0101, 0010, and 0001. The module starts comparing bits from left to right. The signal order to be transmitted to the bus would be 0001, 0010, and 0101. In short, the smallest valued signal goes first on the bus. This receiving module can be programmed with a frequency to check the bus in fixed periods and read the exact frame it requested.

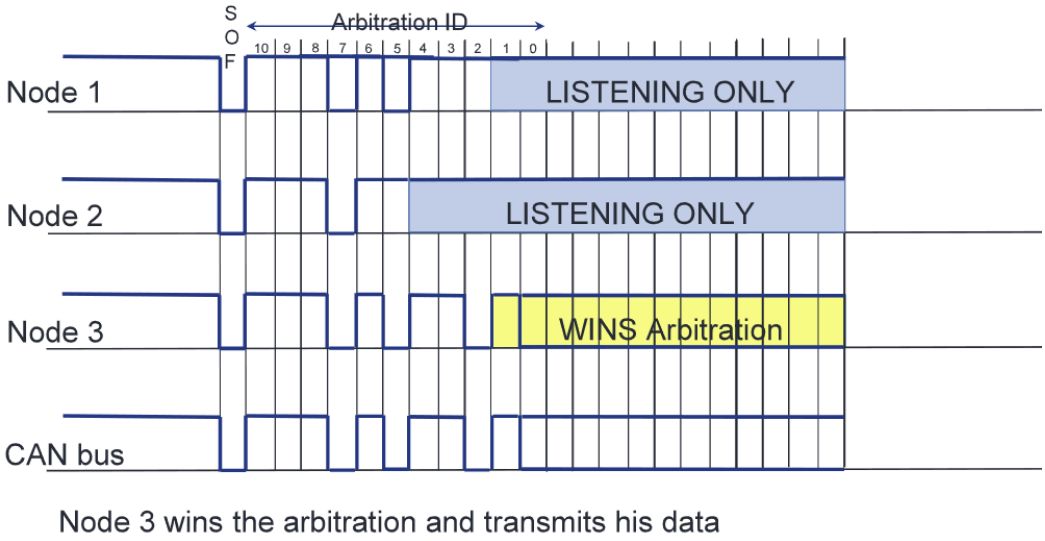


Figure 6 Arbitration Signal Waveform

Error messages, the status of each module or component can be read through the bus. Analyzing tools like CANAnalyzer are used to read messages on the bus. Diagnostic hardware is connected as an additional node to the common bus of the internal CAN line of the car. It is mainly used to detect differences from the expected behavior of the component. Expected behaviors are defined in VF, vehicle function, and files that system engineers prepare. Diagnostic analysis tools can be used to log data in physical tests, or one can transmit diagnostic request messages to the bus for running tests on a specific ECU.

3.3. CANalyzer Experiment

After studying and observing the workflow of the communication systems in the automotive industry, I helped the lead engineer, Mr. Canbaz, to conduct a test on Alfa Romeo Giulia high-performance car. It was a general validation test. We connected CANalyzer to the car, started the engine, and checked the functionality of the dashboard module and infotainment (Center console unit). There was a list of issues with both of the modules. While we conducted the tests, CANalayzer recorded all 3 CAN lines.

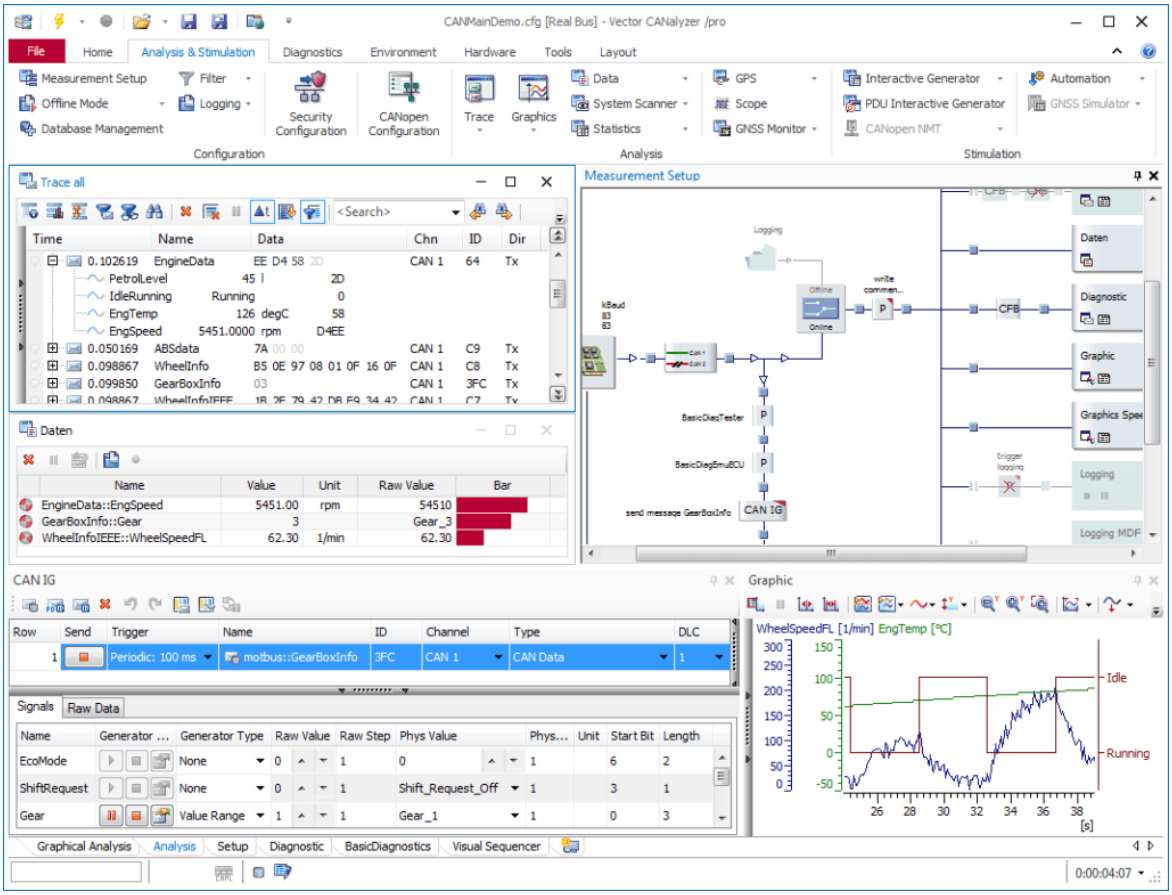


Figure 7 CANalayzer Screenshot

Using the log data, we plotted many different signals in CANalayzer. Since data is confidential, a similar image from Vector's (CANalyzer developer) website is represented in the figure above. The software can plot signal values over time, as seen on the bottom right. The software can plot the engine rpm, engine temperature, the status of each light, the car's speed, etc.. We plotted signals in the same graph that are linked to the failed function. Then compared the behavior with the VF document of the module. One of the issues was when the car is in the key-off state, we could wake the center console module just by pressing the sound button, but when the engine is started car changes status to engine-on. In the transition, the infotainment unit reset itself, closing and opening back up. This was not the expected behavior. Then we went back to the car and, this time used an internally developed tool that can overpass security gateway modules and provides us extended diagnostic signals. These are protected messages that are being masked for outside interventions. The test results and summary are sent to the supplier that manufactured the unit. A TOFAS has multiple suppliers in different parts of the car's electronics. After the supplier analyzes the logs, provide an answer to the issue. It might be a software problem that has to be fixed by the TOFAS software team, or it can be an issue manufactured by another company or supplier to fix the error within their production.

4. SECOND WEEK

4.1. System Engineering: VF Owner

Automotive Open System Architecture is a standard format in the automotive industry. Car manufacturers are communicating with tens of product suppliers while developing a technology. These suppliers are getting orders from many different car manufacturers. Having one format and similar workflow within the industry between the partners cuts cost and time, both for the manufacturer and product suppliers. In TOFAS's workflow, firstly, Stellantis analyzes and provides a problem definition. For example, an ADAS, Advanced Driving Assistance System, will be added to a new car that the company is developing. Stellantis provides industry limitations, standards, and deliverables to advanced system engineers. These engineers analyze the applicability of the request and split the task and assign it to the system engineers, VF owners. System engineers are called vehicle function owners since they are the one that defines clearly how the component should work and behave in different conditions.

In sophisticated components such as the ADAS, unit can be split and defined into five different VF files to eliminate mistakes. The document can be pictured as a datasheet. In the definitions, there are internally defined legends. Different arrows mean different types of communication. Dashed boxes mean virtual components etc. These are confidential information, so there will be no specific description of the blocks in this report. As explained before, one system is often split into multiple VFs. The connection between the functions is sometimes defined in another file, or it is supplied by the hardware engineers or directly supplied by the network engineers. VF file is the input for both the hardware integration team and the software team. These teams are responsible for achieving the expected behavior defined in the file.

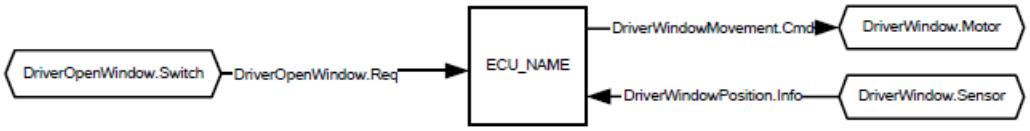


Figure 8 VF Flowchart

Software and hardware development are done in parallel. Arrows in the diagrams are transmitting circuits; hexagons are I/O units; squares are controller units etc. While hardware designers design the components, software designers use many tools to imitate hardware and develop the software in virtual environments.

When the VF deliverables are achieved, software or hardware design, along with the VF files and documents, are sent to suppliers. The supplier takes the newly added units and integrates them into ECUs. ECU is short for Electronic Control Unit. Since many car companies form Stellantis, there are not many projects that start from scratch. Both supplier and manufacturer have know-how on the component. AUTOSAR architecture reduces the complexity of the process. Every manufacturer uses some sort of controller unit, a microprocessor. On top of the

MCU, there are hardware drivers; on top of that, there is a firmware called run time environment that controls all. When a manufacturer sends a software, supplier converts it to their format and loads it to the ECU, where the loaded software works on top of RTE. It can be visualized as a primitive operating system.

I was with the system developer team for multiple days. We worked on changing the IPC, Instrument Panel Cluster VF file during that period. IPC is the electronic dashboard of the car located behind the wheel. It is responsible for displaying a lot of data and taking most of the user's inputs, and supplying it to other ECU modules. It is one of the most complex components of a car. The VF was specifically on displaying break conditions, and one feature was decided to be removed to cut cost. We analyzed the proposition, and I updated the block diagram. The engineer changed the document and sent it to the software and hardware team. In the update process, many meetings are held where the software and hardware teams explain concerns and suggest better or more suitable solutions to the system engineers. After this process of back-and-forth changes, the new file is posted, and every team is assigned to update their parts to the updated definitions.

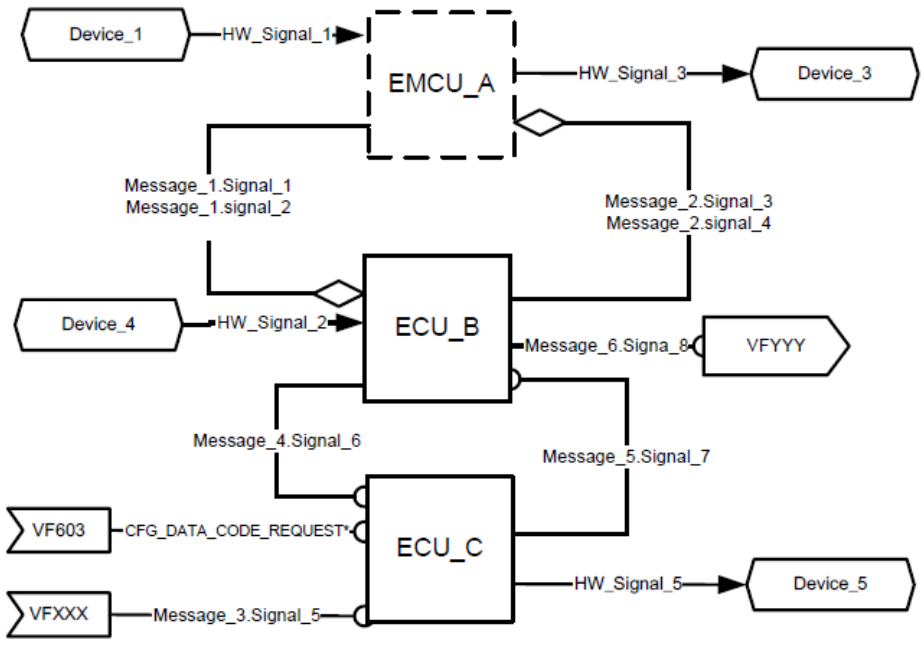


Figure 9 Complete VF File Diagram

4.2. Project Chief and Proxy Tasks

The project Chief is the project responsible in TOFAS. They ensure that all parts of the project are on time with the development schedule. They are responsible for communicating to the team regarding updates and errors or motivating the teams. It is the minimum executive title in TOFAS, and Mr. Acarlar is responsible for project chiefs.

In each project, a similar schedule is followed regardless of the topic of the project; it can be hardware, software, or something else. The mainly 3-part development process is adopted. A model in between the V-shape model and the waterfall model is used. The main difference between these two models is that in the V-shape model, testing and development are done simultaneously, while in the waterfall, each process must be followed by another. The V-shape model is an extension of the waterfall development model. Firstly, requirements are set by the advanced development team. VF file is prepared according to the criteria. Each sub-team (Hardware, software, infotainment, network, etc.) reviews and starts developing their parts of the project simultaneously. While working on the project, errors and better possible solutions are discussed. Project Chiefs are responsible for leading these discussions and coming up with answers. In the V-shape model, primarily being used in software development, testing starts right after requirements are passed the first phase. The product is sent to the supplier when the first phase is finished, and development is carried out simultaneously. In the second phase supplier or the sub-teams can request changes. These changes must be imperative that there is no other option to change the design. Sometimes in this phase, cost cuts and updates due to cuts can be done. In the second phase, testing and development of a component are done in parallel. In the last phase, real-life application tests on physical components are done, and the last updates are added.

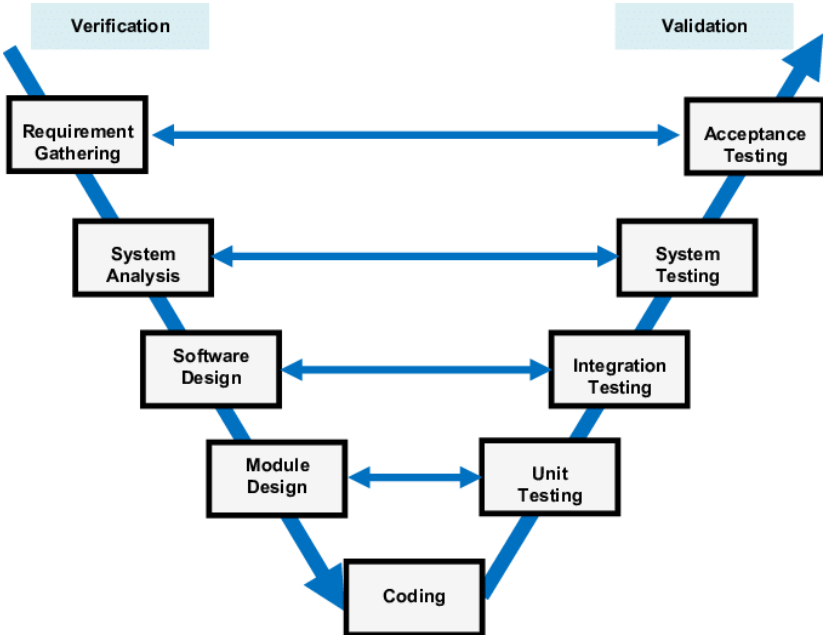


Figure 10 V-Shape Development Model

During my time with the project chiefs, there was a problem with the end-of-line test in the production line. The end-of-line test is a diagnostic test where the car sends back signals that each component is working fine and installed correctly. The reason the EOL test is failing was that the car's software was set up for another package model. The car that fails the test is Urban,

but the software was set up as Lounge. Different packages of the car in the subject can be seen in the figure below. Proxy is the term used to define different packages. Each car's body control module sends the predefined proxy message through the CAN lines to other ECUs. Since there is a disparity in the EOL test, it fails and sends back that it has missing hardware. The car's rundown of the manufacturing steps is sent to us, and we review and send it back to the correct model code. Software is flashed and changed to the correct package.

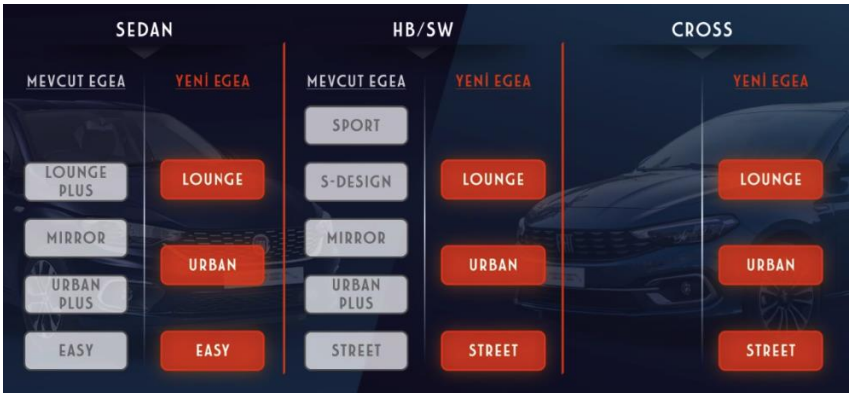


Figure 11 Fiat Egea offered packages

4.3. Powertrain Software Components

Another sub-team in the electronics integration team is the powertrain. I could spend only one day with them to observe workflow and learn. The team is responsible for controlling engine-related software development. They were mostly doing system calibrations until the start of this year. I observed the team transfer more responsibilities from the Italy headquarters throughout my internship. Powertrain software is developed and tested but mainly calibrated by the powertrain team.

In system calibrations, they calibrate already written software specifically to the climate, country, or user feedback. Software in the subject is run inside the engine control module, in short ECM, and controls internal engine components' behaviors. The team optimizes fuel efficiency. The team generates torque maps specialized for everyday use and implemented into the ECM. Again, the powertrain team is responsible for programming engine maps in some projects where multiple engine maps are set, such as Corsa (track mode), offroad mode, etc. Since engine control requires different know-how than the rest of the software development, TOFAS specialized a team for the task. An example of an engine torque map can be seen in the figure below.

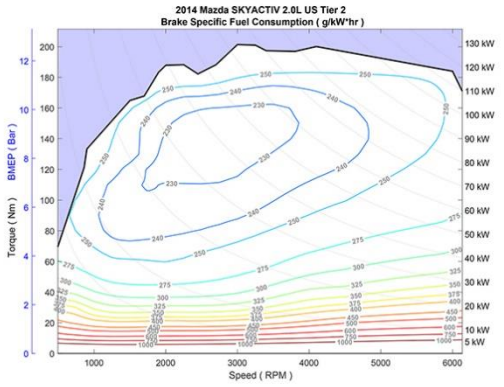


Figure 12 Torque vs Speed Graph

5. THIRD WEEK

5.1. Hardware Design & Integration

Hardware is developed modularly, where one team designs smaller circuits, and the other uses these modules to develop more complex hardware. There are many different modules, from I/O filters and buck converters to a controller for a mechanical switch mechanism. The hardware integration team is responsible for designing the modules. A simulation test for each circuit is done to determine the span of the device in terms of different values such as voltage limit, current limit, maximum thermal resistance, etc. Then this module is sent to the hardware manufacturing supplier along with the limitations document. In this step, the same circuit and its datasheet-like document are sent to different vendors, and each supplier replies with their approach to manufacturing the circuit. They are required to send a datasheet of each component that will be used and the detailed process of how it will be manufactured. Information such as which kind of soldering or what kind of circuit board will be used is listed in the replied proposal. The hardware integration engineer reviews the proposal and checks if it is a good fit for the application. The cost and lifetime of the product are both critical when deciding. Then proposals are graded blindly, and one with the greatest mark is selected to be manufactured when requested. To understand the process better, I did the explained steps one by one on a finished project.

The task was to design a 5^{th}-order low pass filter for an I/O device. Most common basic circuits are already designed and can be searched by an internal tool. Since I don't have access to the internal library, I only took the capacitor and inductor values and designed the same circuit in LTSPICE. LTSpice allows the user to do simulations with real Analog Devices components. I simulated with a variety of capacitors and inductors and listed acceptable corner frequency variation that is again stated in the library. After submitting my work to the engineer in charge, she gave me an excel sheet with different proposals. Since it is a reasonably basic circuit most important value to look for is the expected lifespan and suitability in automotive safety regulations.

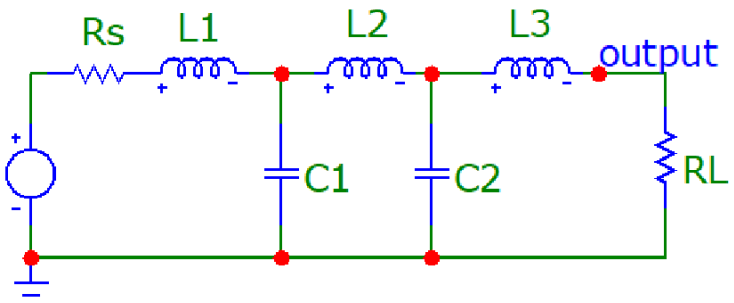


Figure 13 5th order low pass filter

Hardware developers, when needed, will use this designed module to create an I/O module. As shortly touched on, smaller circuits, are loaded into a server where the developer can take each

module as a box in MS Visio and connect it with other modules to create more complex modules. The network team defines connections between modules and signal names and types listed in MS Visio. Generated Visio diagram is just a representation of a greater module with references to real circuits. These files are named SEDs, system electrical diagrams. These files are sent to automotive suppliers such as Continental to be manufactured.

5.2. Network Systems: Cables

The network team was in touch with every other team. One of the responsibilities of the team was to design cable networks. It is called harness in the automotive industry. It is a big jungle of wires strapped together, which can be seen in the figure below. The main concern is keeping the complexity minimum while preventing signal disruptions due to closely strapped different CAN lines. Length is another limitation in which quarter wavelength transform is utilized in CAN lines. The cable must be exact multiples of the quarter of the wavelength that the signal will be passing through the wires. CAN line terminations were explained in the first week of my internship. The team primarily creates variants of already designed cable networks rather than going through all the design process.

Figure 14 Cable Harness

5.3. Infotainment Systems

I concluded my week by helping infotainment systems engineers. This team is specialized in entertainment and information units in cars. Navigation screens, radio panels in the center console, and the information part of the dashboard are in teams' responsibilities. I helped the team to calibrate the equalizer setting of the sound system of the Fiat Egea Cross model. Team is mostly doing revisions on manufactured infotainment systems and reporting back to R&D teams.

6. FOURTH WEEK

6.1. Software Team Workflow

Throughout my internship, I mainly spent my time with the software team, the team that I applied for, but for readability, I will be explaining all my experience in software development in one chapter.

The team is split into two subcategories. Testing and software component development. Each engineer is usually assigned to do both tasks. When a new software component is being developed team works as pair. One program, the SWC (Software component), and the other writes test cases simultaneously. This minimizes development errors and the fixed time when an error is detected since it is tested simultaneously.

6.2. Software Component Development

SWCs are developed in MATLAB/Simulink environment. This development process is called Model-Based Software development. Many proprietorial tools, such as the dSpace-Targetlink addon in MATLAB, are used to generate SWC. VF files are the reference to develop the software. Tests are constructed by the same function definitions. Each engineer can understand different functionality if the VF file is not clear. Working in pairs without knowing each other's work is an excellent way to test software. If both communicate through development, they can make the same mistake. VF file, as I explained before, provides functionality definitions and additional files such as a file explaining signal names and if a similar SWC is developed before the model of the previous variant is provided.

The developer uses Targetlink blocks to create logic gates and conditions and connects corresponding input-output signals. Smaller blocks get connected to each other with respect to the VF definitions. Then using dSpace addons, C language code is automatically generated. This method minimizes the code's readability problem since some engineers don't have enough knowledge in writing and commenting correctly on the code. Both developers and suppliers benefit from the model-based process. Hard coding, coding directly in C language, is minimal in TOFAS. Quick fixes or bugs due to Targetlink are fixed manually when needed. Generated code and model are then sent to the tester. Suppose the component passes the tests. SWC is sent to the supplier.

At this point, the supplier already has hardware or a simulation that can simulate the hardware TOFAS requested. AUTOSARs modular architecture comes in handy at this stage of development.

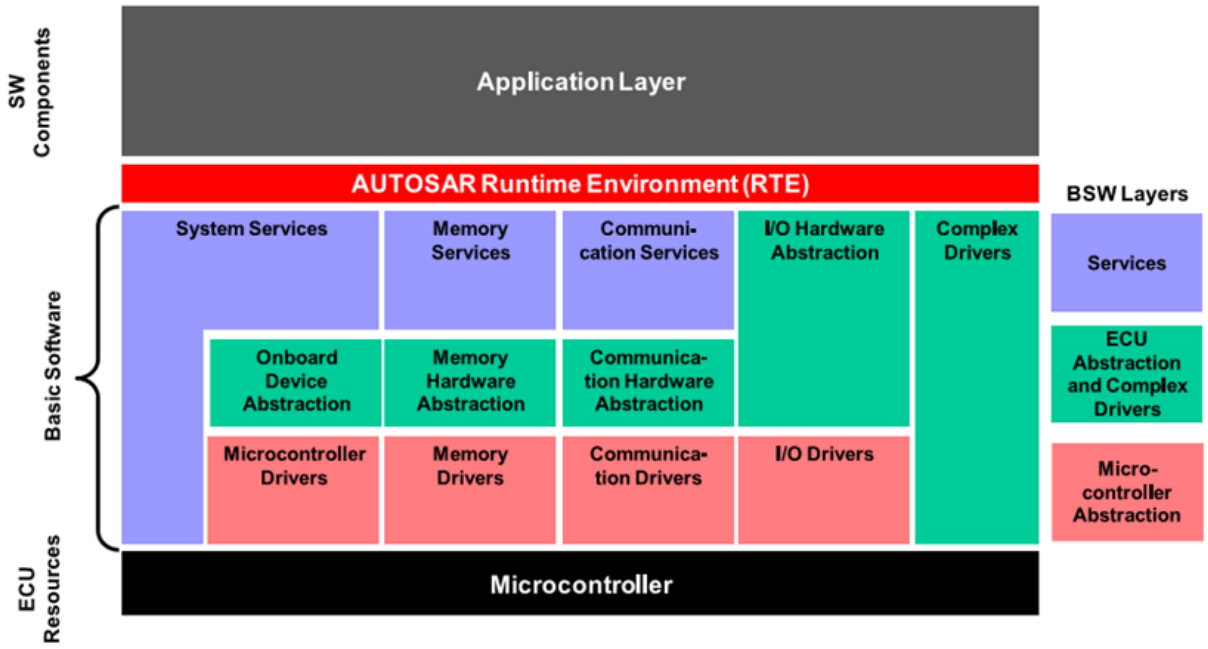


Figure 15 AUTOSAR Architecture summary

The figure above shows a summarized view of an AUTOSAR ECU. As I explained in the hardware chapter, microcontrollers and drivers are defined and requested by TOFAS to be manufactured by the supplier. Some complex drivers and abstraction layers, some of which are software-based and some are hardware to communicate with higher levels. AUTOSAR Runtime Environment RTE is an application binary interface, ABI, in operating system terms. Each hardware has firmware that is developed by the supplier for it to work with the RTE. BSW, basic software behaviors are embedded in the Targetlink addon so that developer doesn't have to directly consider drivers that run under the RTE while developing the application. In more advanced AUTOSAR architectures such as adaptive AUTOSAR. Hype vision alike architectures are utilized and ECUs can take advantage of virtualization and fully functioning operating systems. Adaptive AUTOSAR is mostly used in autonomous cars or high-end electric cars. The industry is slowly transitioning to the Adaptive AUTOSAR since most car manufacturers are leaning into EVs and smart vehicles. The following figure shows the V shape development model of the model-based AUTOSAR software development process. Even though I applied for the title due to confidentiality, I only observed the development. I joined tutorials and demos where I got experience using internal tools and learned the workflow. Luckily I got the chance to reinforce my knowledge by doing testing the rest of the week.

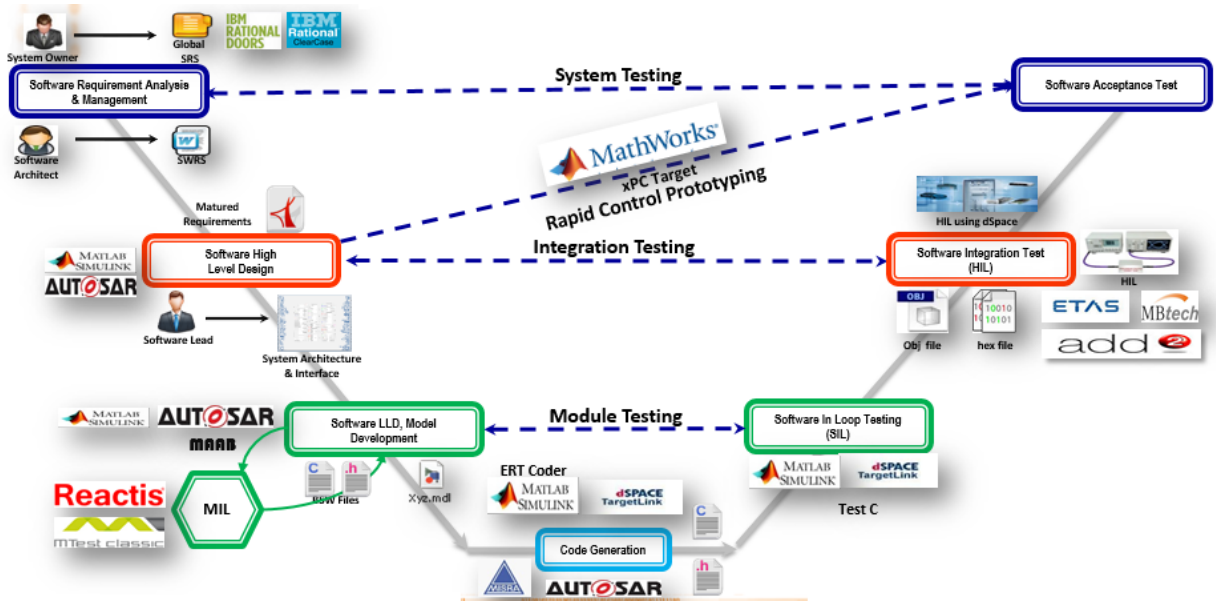


Figure 16 V-Shape model in AUTOSAR Development

6.3. Testing

There are two types of testing shown in the figure above. MIL, model in loop, and SIL, software in loop test. A similar testing system is developed for hardware development called hardware in loop in short HIL. MIL and SIL tests are automated by an internal tool. The developer is required to define variables, and by using the defined variables, functional test cases can be written. There are multiple groups of tests. Common cases and automotive musts are tested. SWC-specific test cases are added to the common case tests. An example of the common test is that if we are testing a brake system related, the SWC car key has to be in on condition. TOFAS uses an excel sheet where all cases are listed. MATLAB functions, along with internal tools added functions, are used to define test cases. The tool is loaded with the case list and the software model. The MIL test tool uses the Simulink model and runs the test codes directly on the model. Moreover, the SIL test model is converted to C language and tested as software. There might be disparities between MIL and SIL test results due to Targetlink bugs or missing definitions in the model.

The last and equally important test is the coverage test. Code coverage is where test code is tested. Generated SWCs C code is run with the test cases, and supervising software checks if every part of the C code is utilized. If not, it is a good indication that there might be missing test cases in the defined test case file. There are different types of coverage where it only checks if the condition is checked if the condition is true or relative conditions. (Ex. When one condition is set other must be both true and false in different tests.) Relative checks are not a must in the automotive industry. The aerospace industry uses relative coverage more often.

After learning detailed testing processes, engineers provided me with already tested components and asked me to write test cases. It was a cyclist warning signal generating SWC. Since it is

confidential, I will be altering the functionality of the SWC when explaining my experience. In short, I had to test nominal conditions such as if the key is on, if the responsible ECU is connected to the common bus, if the ECU is not busy with other operations, etc. Then I move on to add SWC-specific test cases. I used the VF file as the reference. If a group of sensors is sending a specific message in the same period of time, it means that there is a cyclist in the proximity of the car. The car has to indicate the driver by lighting up a tell-tale light on the dashboard. Information is also transmitted through the CAN bus and needs to be tested if it is transmitting the signal or not. I mainly used MATLAB if() function while defining test cases. Then provided my excel file to the developer, and we checked if the test case was passing and had enough coverage in terms of automotive standards. It was a fairly simple program, so coverage was high, and in two tries, I managed to make it work 100%. While writing the test, I got a better understanding of how the AUTOSAR development processes work.

An additional type of testing is doing the SIL test manually. There is Stellantis internal software that does the simulations. Software vendors such as dSpace and FESTO provide similar simulation tools for both hardware and software simulations. In SIL simulations, a mockup dashboard provides information to the tester, and a menu is used to create different conditions. The tester can set specific car speeds, turn the wipers on and off, etc. This testing method eliminates "rough edges" of the software since the tester can directly see the behaviors.

7. CONCLUSIONS

To conclude my report, I spent one month in TOFAS but learned and gained experience worth several months. I had the opportunity to work with Italian and Turkish engineers here in TOFAS. It was a privilege to communicate with other engineers in English, the language I got my education. It is always hard to get used to Turkish terms and somewhat arbitrary. I learned what AUTOSAR is and how embedded software development is done in the automotive industry. My knowledge in embedded software development was helpful even though the C language usage is minimal. The concepts I learned in my lessons still apply to model-based software development.

Besides the job title, I applied I got to experience involved in sub-teams operations with software team such as hardware and network. My knowledge of analog electronics and microwave engineering topics helped me to understand the processes in detail. Small mockup tasks they assigned me to tackle enhanced my knowledge in AUTOSAR development while utilizing my theoretical knowledge into practice.

Shortly after this internship, I gained valuable experience in an area in which I had little to no experience. Having to see how embedded software development in the automotive industry was illuminating.

8. REFERENCES

AUTOSAR. (2021, December 6). *Adaptive platform*. AUTOSAR - Enabling Innovation. https://www.autosar.org/standards/adaptive-platform/

Bullseye. (n.d.). *Code coverage analysis*. BullseyeCoverage - C++ Code Coverage Tool. https://www.bullseye.com/coverage.html

Cache://10.1109/SEAA.2011.45 - Google search. (n.d.). https://10.1109/SEAA.2011.45

dSpace. (2022, June 14). *MATLAB, Simulink, Stateflow and TargetLink*. dSPACE. https://www.dspace.com/en/inc/home/products/systems/mbd/matlabsimulinkstateflow.cfm

Software engineering | SDLC V-model. (2022, March 3). GeeksforGeeks. https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/

TOFAS. (n.d.). *Tofaş*. Tofaş. https://www.tofas.com.tr/Hakkimizda/Tarihce/Pages/default.aspx

T. Hermans, P. Ramaekers, J. Denil, P. D. Meulenaere and J. Anthonis, "Incorporation of AUTOSAR in an Embedded Systems Development Process: A Case Study," 2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications, 2011, pp. 247-250, doi: 10.1109/SEAA.2011.45.

Vector. (n.d.). *Autosar*. Vector Informatik GmbH. https://www.vector.com/int/en/know-how/autosar/

International Free Mission UAV Competition

Conceptual Design Report

TEAM NAME: VEGA UAV	
TEAM RESPONSIBLE NAME/SURNAME: Babak Danesh	
TEAM RESPONSIBLE UNIVERSITY: Middle East Technical University Northern Cyprus Campus	
AIRCRAFT TYPE: Fixed Wing	
AIR VEHICLE DEVELOPMENT	**New Aircraft**

1. ORGANIZATION SUMMARY

1.1 Team Organization

Electronics and Programming Sub-team:

Table 1 Electronics & Programming Sub-team Members

Name & Surname	Department & University	Year of Study
Hevra Petekkaya	Computer Engineering, METU NCC	3rd Year
Barış Güzel	Electrical and Electronics Engineering, METU NCC	4th Year
Mariam İmran	Computer Engineering, METU NCC	3th Year
Abdul Rafey Khan	Computer Engineering, METU NCC	1st Year

Mechanical & Aerodynamics Sub-team:

Table 2 Mechanical & Aerodynamics Sub-team Members

Name & Surname	Department & University	Year of Study
Burçin Telli	Aerospace Engineering, METU NCC	3rd Year
Muhammed Üge	Mechanical Engineering, METU NCC	3rd Year
Ersin Ateş	Aerospace Engineering, METU NCC	1st Year
Babak Danesh	Aerospace Engineering, METU NCC	3rd Year

As it can be seen from the tables, VEGA UAV team consists of 8 members from Middle East Technical University Northern Cyprus Campus. The team has two different sub-teams which are Electronics and Programming and Mechanical and Aerodynamics. Since Unmanned Aerial Systems required multidisciplinary organization, VEGA UAV includes members from different engineering branches. The team captain and communication responsible is Babak Danesh who has UAV1 pilot license from ESOGÜSEM. The organizational chart of the team has been explained below.

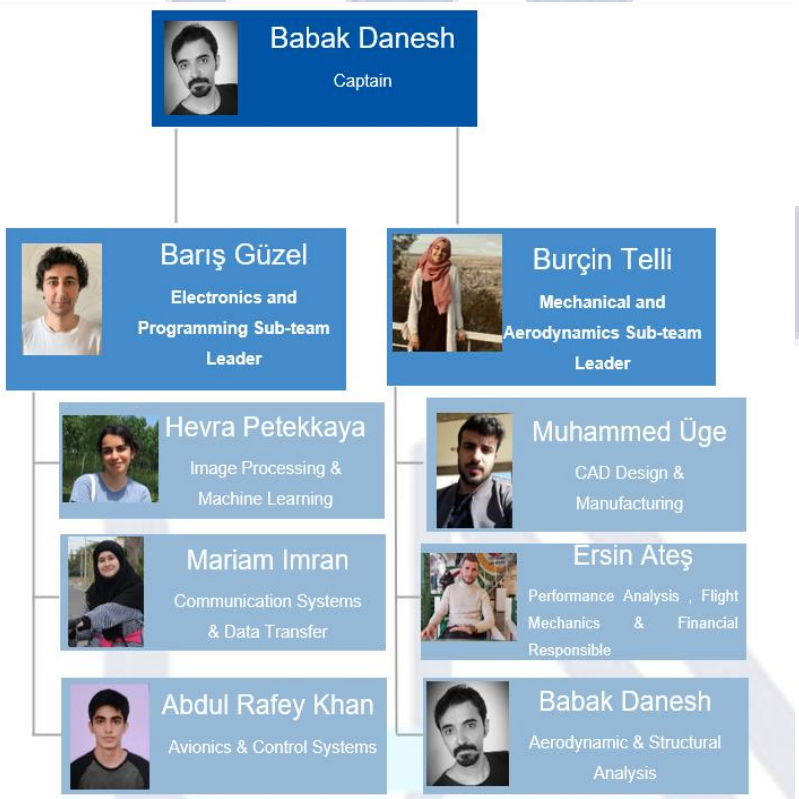


Figure 1, Work Flow Chart

It is planned to reach the competition's goals by having continuous weekly progress report presentations (PRP), monthly tasks, and mini-projects. The specific missions for each sub-team are as below:

Mechanical & Aerodynamics Sub-team:

Within the scope of the Mechanical & Aerodynamics Sub-team, it is planned to Design, Analyze, Manufacture, and Test the desired physical parts of the UAV considering optimum performance criteria and most efficient power usage.

In order to reach optimum and most efficient product, the processes described below in Figure-2 are followed by the Mechanical & Aerodynamics sub-team.

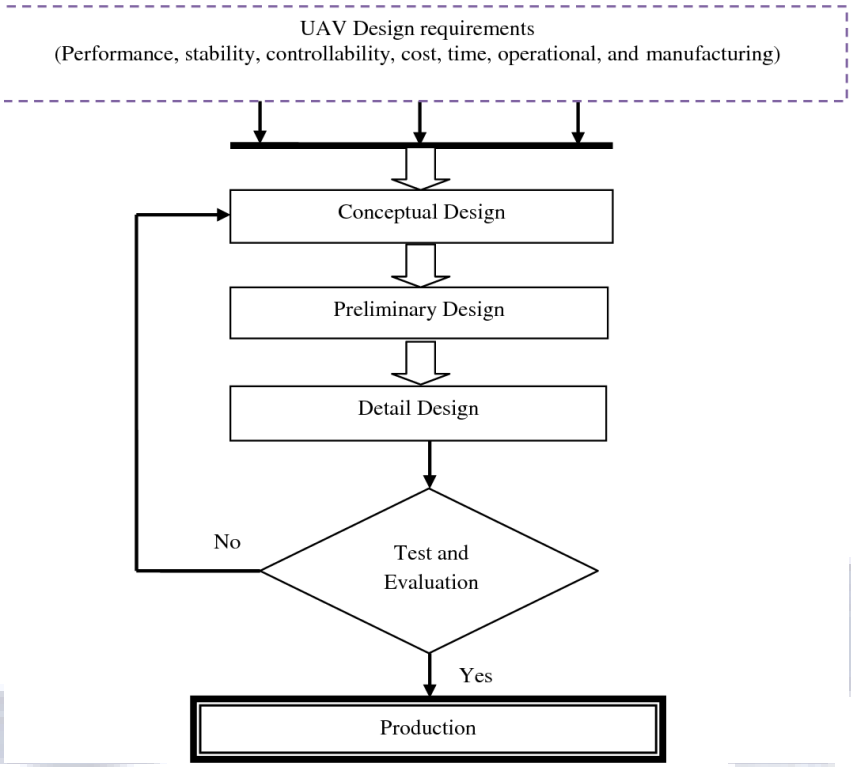


Figure 2, Mechanical & Aerodynamics Sub-team's Tasks Flow Chart

Electronics & Programming Sub-team:

Within the scope of the Electronic & Programming Sub-team, it is planned to setup up the circuit schematic, communication system, and build a Machine Learning Model that will run in the most efficient way from an algorithmic perspective. All the connections within the overall system will be tested to confirm that all the parts are working smoothly in the most efficient manner.

1.2 Workflow Chart

According to the calendar given at the Teknofest's Website, the Project Timeline of the Mechanical and Aerodynamics sub-team is planned to be like the table below:

Table 3, Mechanical & Aerodynamics Subteam's Timeline

Task / Date	Mar	Apr	May	Jun	Jul
CAD Design of The UAV	Detailed CAD Design will be done until 29th March				
Analysis	Analysis will be done until 22th March				
Manufacturing		Manufacturing will be done until 2nd May			
Structural Tests			Structural Tests will be done until 28th May		
Mission-Based Flight Tests (MBFT) and Final Changes				MBFT will be done until 5th July	

Similarly, The Timeline of the Electronic & Programming Sub-team is below:

Table 4, Electronics & Programming Sub-team's Timeline

Task	Date	Mar	Apr	May	Jun	Jul
Circuit Schematic		Detailed Design of the Circuit Schematic				
Learning deep learning frameworks and libraries		The basic building blocks needed to build an ML model will be learned				
Building the ML Model				Build the model for the Fire detection		
Fine tuning the ML Model					After a skeleton is built for the model, fine tuning will be done	
Mission-Based Flight Tests					MBFT will be done until 5th July	

2. CONCEPTUAL DESIGN

2.1 MISSION

After experiencing Forest Fire that reached to METU Northern Cyprus Campus's border (DHA,

2020) at May 2020, VEGA UAV team members decided to work on Early Fire Detecting UAV. However, due to the limitations brought by the pandemic and METU NCC's tight academic calendar, team members could not meet regularly and discuss about the team until the foundation of the team at August 2021. As the team was founded, the flight scenarios and flight path were continuously discussed during weekly meetings. Finally, all members agreed on a UAV that is capable of autonomously flying over both a predetermined and a spontaneously determined path.

It sends and receives any data/commands through a Pixhawk flight controller. While flying, it has the capability of autonomously capturing, processing, and analyzing images/video of the forest, which it is flying above, by the help of a machine learning model. The data processing happens onboard through an NVIDIA Jetson module because of latency issues that might arise from sending the images/video to the GCS. Finally, it sends any fire alerts to the ground station in case the machine learning model predicts that there is a fire.

Furthermore, in order to have a safe environment during the flight, the fires will be made in gallons and Fire Pits as shown in Figure4 & 5.

After human-assisted launch and manual take-off, operator can select the target points from Mission Planner as shown in the Figure 6 to observe the potential region for forest fire. After observing the target region, the UAV can land on its fuselage due to shock absorbers that described in section 2.2, Alleged Matters.

Figure 3 Forest Fire in METU NCC on News

Figure 4 Sample Fire Gallon for Test Flights

Figure 5 Sample Fire Pit for Test Flight

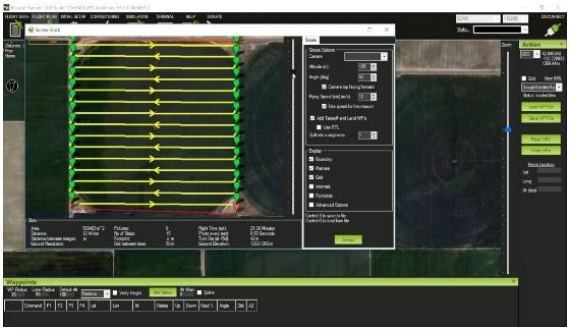


Figure 6 Sample Pointing in Mission Planner

2.2 Alleged Matters

In order to have clear description, Alleged Matters section is divided into two parts: Electronics and Programming & Mechanical and Aerodynamics.

2.2.1. Alleged Matters for Mechanical and Aerodynamics Sub-team:

After observing the possible areas for landing and takeoff of Early Fire Detection UAV and after flying with test prototypes (Figure 8) as well, it is concluded to use shock absorbers (inspired from Stalker XE) instead of landing system.

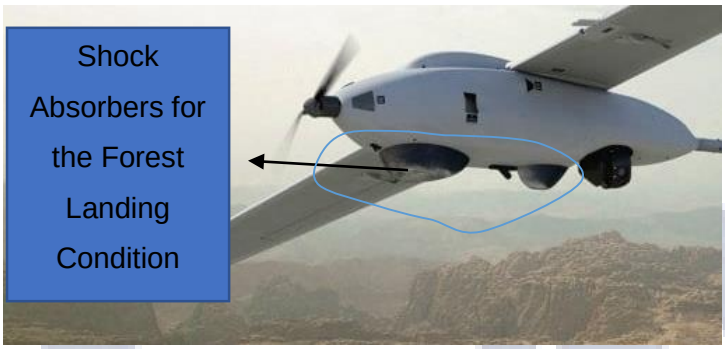


Figure 7 Shock Absorbers of Stalker XE

Figure 8, Forest Road Near to Akdeniz Forest

Moreover, in order to have stable flight, the design of UAV is inspired from Cessna172, which is the most successful pilot training aircraft in the history of aviation. After considering design parameters such as Ease of Manufacture Process, Strength of Structures, and Aerodynamic Performances Cessna172 selected as the pattern model. Considering the UAV's mission and payload, minor changes are done on design and are explained in section 2.3 Production with Aerodynamic, Mechanical and Structural Features.

2.2.2. Alleged Matters for Electronics and Programming Sub-team:

There are two approaches that we can adopt for building a Machine Learning model that can detect fire. The first one would be to train a neural network from scratch with the limited data in hand. The second one would be to perform transfer learning, meaning that a pre-trained model can be used, which was not specifically built for our aim, and add on more layers to be trained on the specific goal of fire detection.

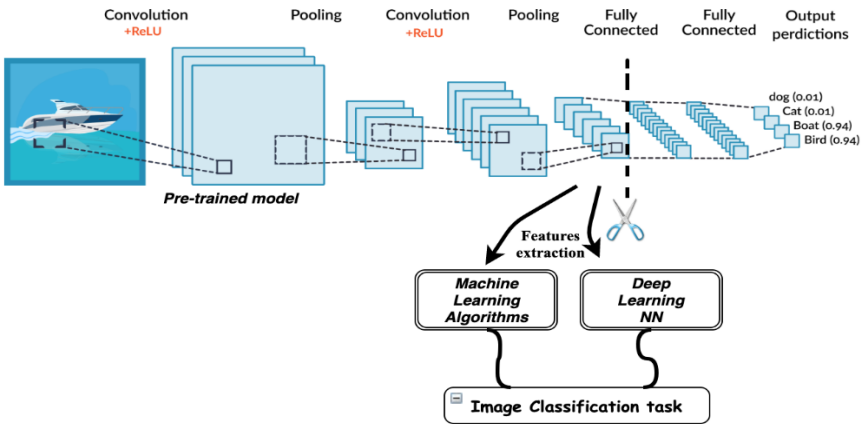


Figure 9, Convolutional Neural Network

The second approach, transfer learning, is more suitable for the fire detection problem because there is very limited amount of data, fire pictures, to be able to build a model from scratch. The advantage of the second approach would be that in the initial layers of a neural network, minimal features, such as a line, edge, shape, etc., are detected. These features are common to most models; hence, the knowledge acquired from this model can be transferred, which will be considered as a pre-trained model, to build a model to detect fire which is a more specific task.

2.3 Production with Aerodynamic, Mechanical and Structural Features

In order to have aerodynamic design, one of the useful programs is XFLR5 which can be used to extract essential data (such as $C_l - \alpha$, $C_l - C_d$ &…) for 2D and 3D geometries of the UAV. Analysis has been run in Xfoil environment with Ra= 3,100,000 & Mach=0.1. After comparison between results of different airfoils, NACA 4412 has been chosen. The sample comparison for NACA 2212 and NACA 4412 is provided in figure 11. Moreover, XFLR5 will be used for the aerodynamics analysis of the wing as well. Similarly, Ansys Fluent will be used to observe the turbulency and get data for dynamic pressure. Furthermore, Ansys Static Structural will be used to analyze the strength of the UAV.

As described in the Figure 13, after getting reliable and desired results from the analysis, the Manufacturing Process will start. The Fuselage of the UAV will be made of Balsa-Carbon Fiber composites in order to have maximum strength and lightness. The process will be done with a laser cutting machine to have balsa parts and female molds printed via 3D printer, necessary Epoxy fixtures and fibers to have carbon fiber layers. In addition, wing and stabilizers of the UAV will be manufactured from XPS-Balsa Composite. In order to manufacture the wing, the NACA-4412 aerofoil section will be cut via a laser cutting machine and montaged on the XPS. After the montage, the wing will be cut with hotwire as two separate parts and installed on the fuselage via cylinders and lock mechanisms.

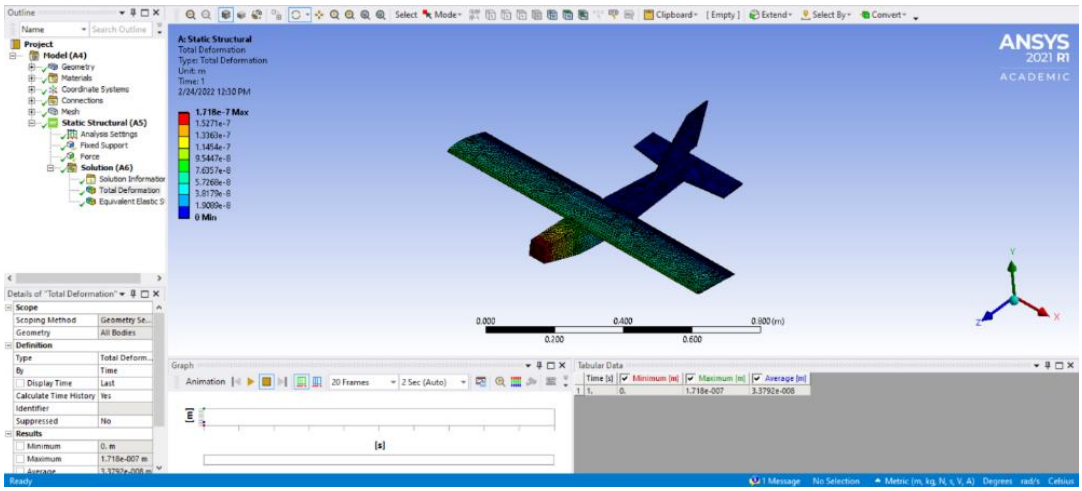


Figure 10, Sample Structural Analysis via Ansys

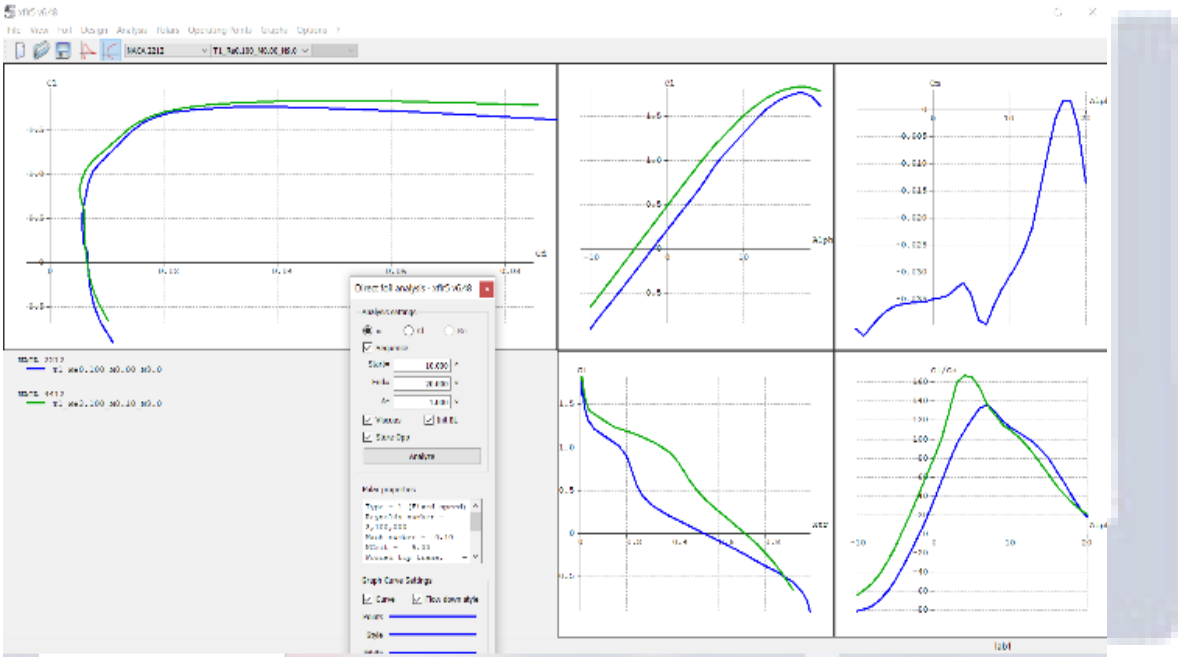


Figure 11, Comparison between NACA 2212 & 4412

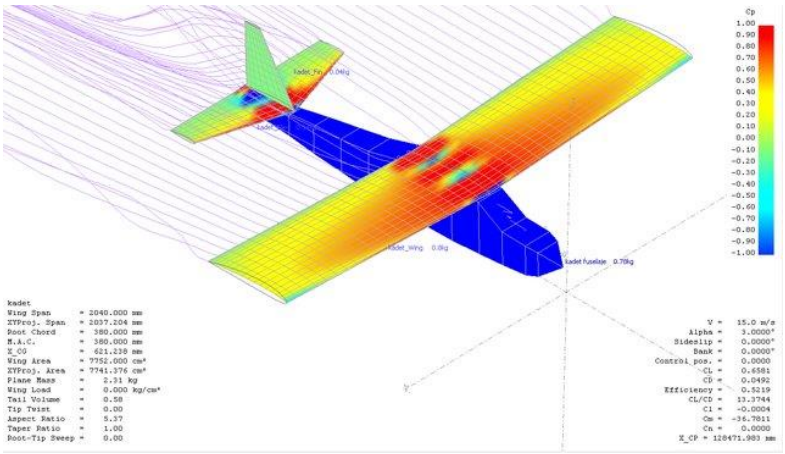


Figure 12, Sample XFLR5 aerodynamic and stability analysis

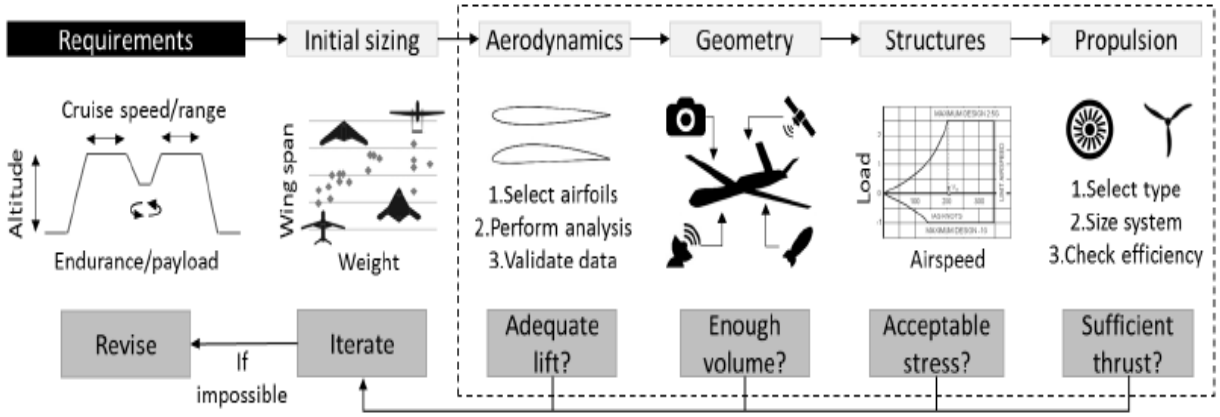


Figure 13, Design Process of The UAV

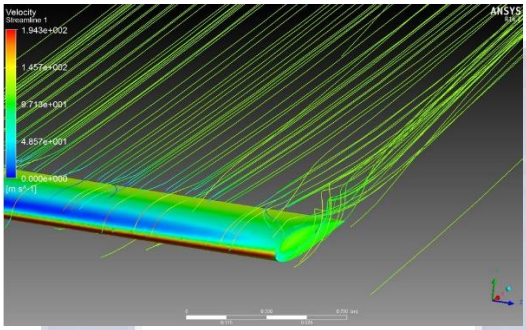


Figure 13, Aerodynamic Analysis of The Wing via Ansys

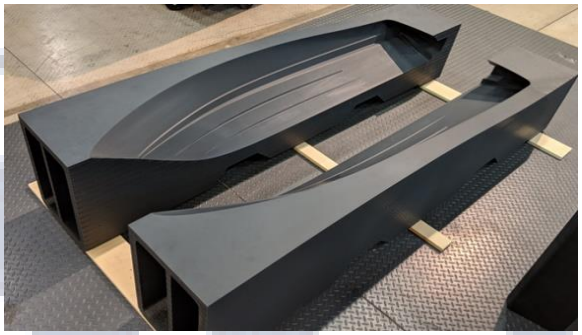


Figure14, 3D printed Mold for Carbon Fiber Manufacturing

2.4 AVIONICS

2.4.1 Circuit Schematic

Circuit Schematic is represented in the Figure 15 below. The reasons for choosing each component and their functionalities in this schematic are explained in this chapter. A 14.8V 4S battery will be powering all of the electronics. It is supplied through a fuse and a circuit breaker as stated in the competition safety requirements. A module that combines both fuse and the breaker will be used to cut power in less than 2 seconds in an emergency. Fuse is selected as 50A to prevent exceeding the selected ESC limit in any failure. A 35A ESC will be used to drive a brushless DC motor. The brushless motor will be located at the nose of the UAV. Lastly, on power systems, a DC/DC converter will supply 5V to the electronics. UAV is equipped with two processing units. The mission flight controller is Pixhawk 4, and Nvidia Jetson Nano is chosen as a secondary processing unit for fire detection. A Raspberry Pi camera will be located under the plane for fire detection. Three servo motors, one for each control surface, will be placed inside the fuselage. Servos and control surfaces are connected with metal cords. By pulling and pushing the cords, the UAV is steered. GPS, IMU(Inertial Measurement Unit), and an airspeed module will provide necessary data for autonomous flight. The flight controller will communicate with a radio Mavlink transmitter with the ground station. Jetson will transmit

images with a 3G module. The figure shows that UAV can be controlled manually by the radio connection with a controller along with the autonomous flight.

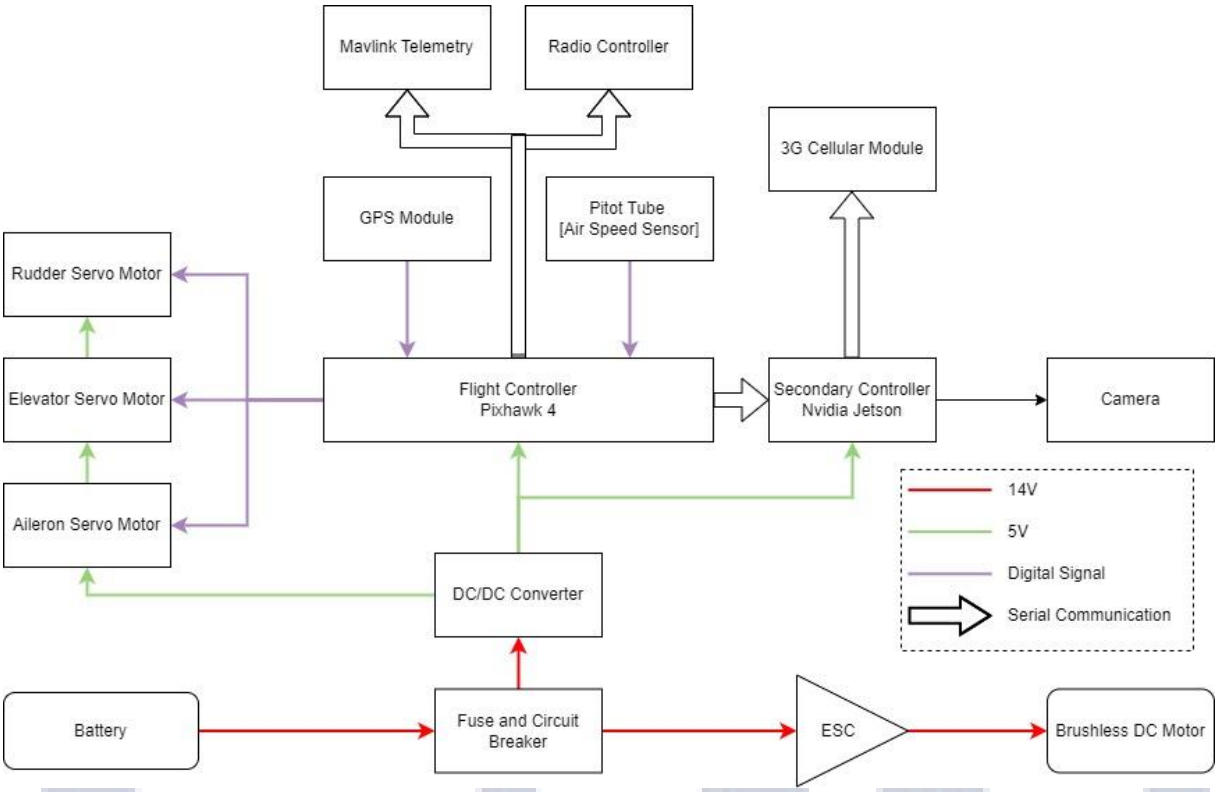


Figure 15 Circuit Schematic

2.4.2 Sensors

Sensors are what UAVs is relied on while autonomous flight. The following sensor implementations are essential to achieve a fluent autonomous system and a better manual drive. While flying autonomously, UAV will only be relying on its onboard autopilot algorithm and its sensors. The autopilot software will use the sensor data to follow the mission path precisely. GPS module will provide location data in latitude and longitude. IMU is a unit that is a combination of a gyroscope and an accelerometer. The barometer provides the altitude. Lastly, a pitot tube sensor is used to determine airspeed. Pixhawk 4 is already equipped with IMU and barometer, which are not shown in the circuit schematic. All combined, the software can determine the UAV's current state at any given time. Pixhawk's open-source autonomous flight software will be used, relying on the specified sensors above. The team will further develop the software according to the mission's requirements. Lastly, the Raspberry Pi v2 Camera is installed for fire detection purposes. The camera and machine learning setup is explained in part 2.4.5.

2.4.3 Flight Controllers

In terms of the flight controller, we will use the Pixhawk flight controller. We decided on this not

only because it is the set hardware standard for drone applications but also because of its open-hardware ability, which will help us model it to exactly our needs. We know that it is a well-tested device that we can rely on for clutch tasks.

2.4.4 Communications

To establish connections between UAV and ground station, we will be using 3 data links for the following purpose:

- RC transmission (with Controller)
- Mavlink/Telemetry transmission (with GCS)
- Image transmission

Image Transmission

For image transmission, a long-range and high bandwidth data link is required to transmit fire-detected images to the ground station in real-time. Our team has discussed several approaches, such as a dedicated Wi-Fi link, VTX transmitters, and using cellular networks. However, considering the scope of the team's budget and mentioned requirements, we have decided to leverage cellular networks and design a communication architecture that sends images to the ground station captured by the camera only if they are classified as containing the fire as predicted by our machine learning model on OBC, thus minimizing traffic and network usage.

OBC connected to a 4G LTE dongle has access to the internet and can upload images to our custom-built cloud-hosted GCS web server via HTTP. The GCS web server is hosted in the cloud as it requires a static public IP address to be publicly accessible across the internet. This web server will have an image upload API, which broadcasts the image to the GCS clients via webhooks when receiving a POST request with an image.

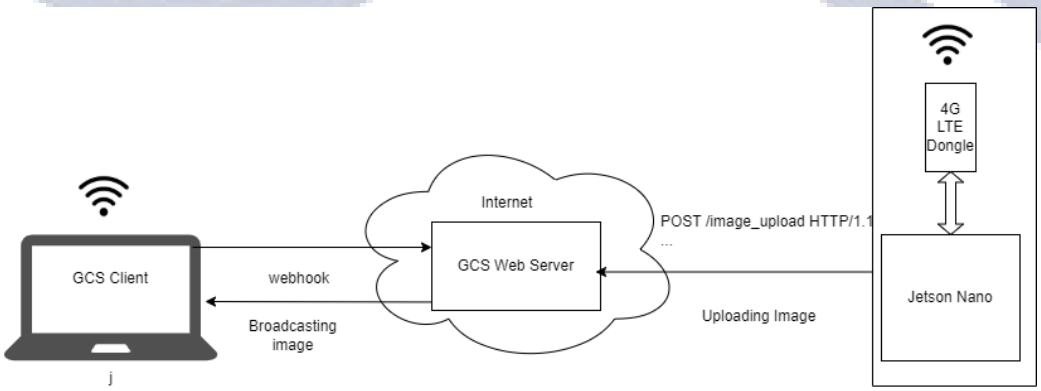


Figure 16 Image Transmission

2.4.5 Setup for the Machine Learning Model

Firstly, we plan to complete most of the processing of the data onboard the UAV. This is because the images/video captured of the forest below need to be analyzed frame by frame

by the machine learning model. For this to be done on the ground, each frame will have to be transmitted leading too many problems, the most significant of which is the latency in data transfer. This latency will also mean that the UAV will be farther away from the geolocation of the image/frame, hence greater difficulty with location accuracy. This problem is significantly reduced if the lag time for the data transfer is eliminated.

We plan to use one of NVIDIA's Jetson modules for onboard processing. The advantage of this is that these modules are designed explicitly for autonomous machines and AI applications. This helps us keep the complexity of the tasks in control and assures us that we are using a state-of-the-art and reliable machine as the backbone of our processing. As for the camera, it is planned to use the Raspberry Pi Camera module due to its compatibility with NVIDIA Jetson Nano, specifications that match the need of the ML Model, and affordability.

Figure 17, Connection of NVIDIA Jetson Nano with Raspberry Pi Camera module

2.4.6 Power Systems and Safety

Power systems are designed while considering a lightweight but powerful solution. The primary energy-consuming component is the brushless DC motor (BLCD). A motor with 850KV, maximum current is 35A, and estimated power is around 400W. The motor can be powered by 3S or 4S batteries and creates enough torque for maintaining level flight and takeoff. A 35A ESC will be used to drive the motor. BLDC motor will be driven full throttle during takeoff and maneuvering. The rest of the flight it will run around 80%. While assuming a linear relationship with the torque and power consumption, the average power is estimated as 350W. Depending on the developments, differences in actual characteristics of the motor and calculations are expected with small margins.

Other electronics and power dissipations are listed: servo motors, 3 of them located on the plane, create 1.3Kg.cm torque while consuming 1.1W. Nvidia Jetson Nano, on average, consumes 1.2W. Pixhawk 4 consumes 2W, including powering all the sensors. Total average dissipation is 350+1.1x3+1.2+2=356.5W.

Limskey 4S 10400mAh battery is selected for the mission. As stated before, keeping the weight to the minimum is vital. This battery has the power of 154Wh and 893g. Power density (Ratio of Wh to kg) is 172/5 Wh/Kg. This battery enables the UAV to fly for 26 minutes.

Power distribution will be done with a power distribution board that can provide 12V and 5V to the system while taking 14.8V input. 12V output is currently not being used, but if any component is changed to one that requires 12V, for example, a better camera, with minor change team can install a new component. A fuse and a circuit breaker will be connected right after the battery's positive terminal to cut power in less than 2 seconds. A 35A fuse is selected. The distribution board has 3A fuses on the output ends as well.

2.5 PROPULSION AND PERFORMANCE

While choosing Thrust loading value ($\frac{T}{W}$), Data from Figure 18 & 19 has been considered. After these considerations, the thrust loading (Thrust to Weight ratio) value has been chosen as 0,71. As a result of these calculations, FMS 850KV-3536 Motor was selected. According to performance analysis, it is predicted that the UAV can reach the desired cruise speed at mid-throttle with FMS 850KV-3536 Motor and 14 "x7" propeller.

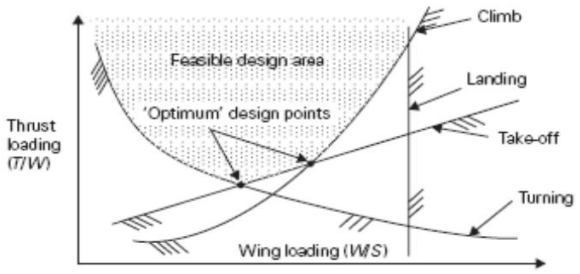


Figure18, Optimum Design Points

Flying Capabilities	Power to Weight Ratio (W/lbs.)
Electric Glider, Park Flyer & Slow Flyer	30-60
Trainers & Basic Scale Flying	60-75
Sport Flying & Improved Climbing	75-100
Limited 3D, Pattern & Racing	100-150
Full Power 3D & Pattern Aerobatics	150-220

Figure19, Power to Weight Ratio Values for Different Capabilities

Table 5, Aircraft Dimensions & Performance Specifications

	Wing	Vertical Stabilizer	Horizontal Stabilizer	Aircraft Dimensions and Performance Specifications	
Airfoil	NACA 4412	NACA 0012	NACA 0012	Overall Length(m)	**1.9**
Area (m^2)	0.39	0.10	0.07	Width(m)	**1.72**
Span (m)	1.74	0.43	0.32	Empty Weight (kg)	**1.7**
Aspect Ratio	7.76	1.85	1.7	Motor's Weight (kg)	**0.140**
Chord (m)	0.23	0.14	0.15	Payload(kg)	**2.8**
Motor	FMS 850KV 3536			Cruise Speed(m/s)	**10.1**
				Maximum Speed (m/s)	**16.4**
Propeller Size	14x7			Stall Speed(m/s)	**4.6**
Battery	Limskey 4S 10400mAh			Endurance (min)	**16**
				Minimum Turn Radius (m)	**11**

2.6 Visual Design Configuration

In order to have accurate and reliable CAD (Computer Aided Drawing) files, Autodesk Inventor Professional and Fusion360 programs have been used. Detail drawings of Shock Absorber and Payload will be provided in the Detailed Design Report, and the final views with Camera and Shock Absorber for the conceptual design are as below:

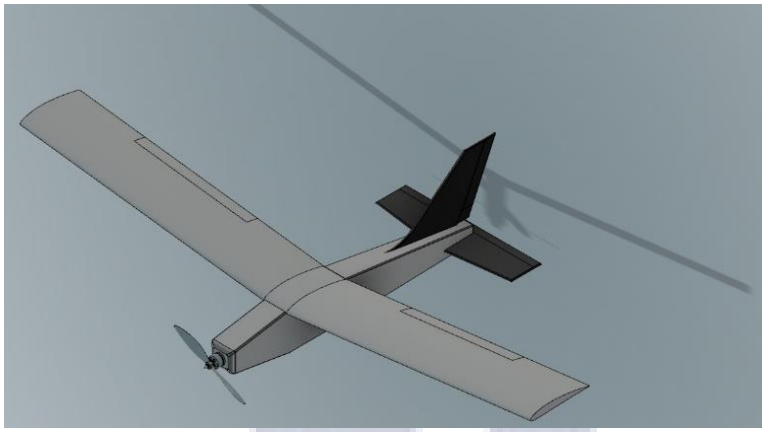


Figure 20, View of The UAV

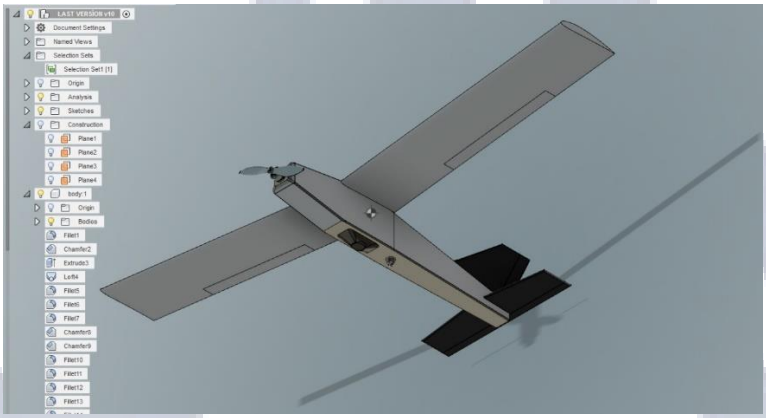


Figure 21 Payload (Camera) & Shock Absorber View of the UAV

REFERENCES

D. (2020, May 17). KKTC'de yangın büyüyor… ODTÜ Kampüsü'ne de sıçradı, öğrenciler tahliye edildi. DHA | Demirören Haber Ajansı. https://www.dha.com.tr/dunya/kktc-de-yangin-buyuyor-odtu-kampusune-de-sicradi-ogrenciler-tahliye-edildi-1771440

Rajendran, P., & Smith, H. (2013). The Development of a Small Solar Powered Electric Unmanned Aerial Vehicle Systems. Applied Mechanics and Materials, 465–466, 345–351. https://doi.org/10.4028/www.scientific.net/amm.465-466.345

Jimenez, P., Lichota, P., Agudelo, D., & Rogowski, K. (2019). Experimental Validation of Total Energy Control System for UAVs. *Energies*, *13*(1), 14. https://doi.org/10.3390/en13010014

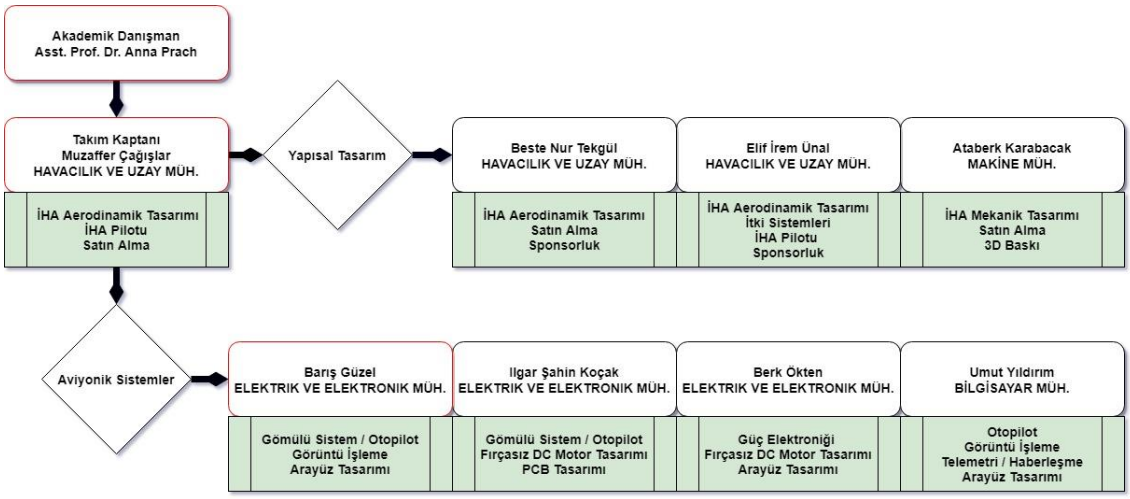
TÜBİTAK

TAKIM ADI:	**METU Ascend**
ARAÇ TÜRÜ:	**Sabit Kanat**
ÜNİVERSİTE:	**Orta Doğu Teknik Üniversitesi Kuzey Kıbrıs Kampüsü**
TAKIM KAPTANI:	**Muzaffer Çağışlar**

İÇİNDEKİLER DİZİNİ **Sayfa**

1. ORGANİZASYON ÖZETİ

1.1 Organizasyon Özeti



Şekil 1 Organizasyon Şeması

Takımımız işinde tecrübeli bir danışman gözetiminde, alanında bilgili ve hevesli 8 öğrenciden oluşmaktadır. Yapısal tasarımda çalışan 3 havacılık ve uzay mühendisliği ve 1 makine mühendisliği üyelerimiz, aldıkları teorik eğitim ile aerodinamik ve mekanik açıdan özgün ve başarılı bir tasarım çıkarmayı amaçlamaktadır. Aviyonik sistemde çalışan 3 elektrik elektronik mühendisliği öğrencisi ve 1 bilgisayar mühendisliği öğrencisi, yarışmada İHA'nın görevi başarılı bir şekilde tamamlamasını sağlayacak otopilot algoritması ve gerekli elektrik/elektronik sistemleri hazırlamakla görevlidir. Yerliliğe önem veren takımımız özgün otopilot algoritması, arayüzü ve fırçasız DC motoru ile yarışmada yerli ve milli bir İHA ile katılacaktır.

1.2 İş Akış Çizelgesi

Şekil 2 İş Akış Şeması

2. KAVRAMSAL TASARIM

2.1 Görevler için İHA Konfigürasyonu

Before designing the UAV we analized the missions and figured the key points that are necessary in order to complete two missions. First mission was to test the autonomous flight and maneuverability where the second one required a special drop-mechanism in order to release weigths at the correct time and location. After considering these important aspects of, we decided that stability and maneuverability were the critical points we should pay more attention to.

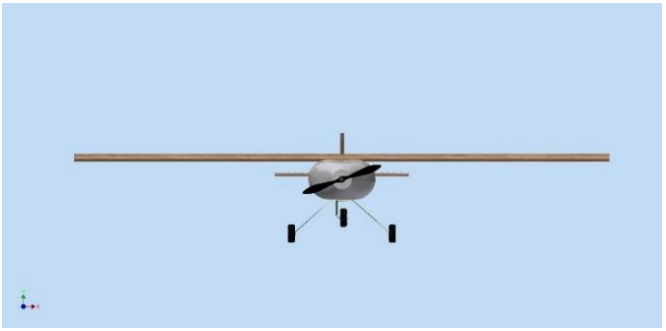
Firstly, we considered conventional airplane configuration for easy manufacturing purposes. Also, it is more convenient model to perform the expected mission. As a result, we decided to use conventional airplane configuration. Secondly, we considered various types of wings and created a list where we could eliminate that wouldn't fit our goals. After long discussions, we were left with three types of wings which are; low wing, mid wing and high wing. We decided to use high wing because it would produce more amount of lift than mid and low wing, it would be more advantegous considering the necessary space for the elements and weights in the fuselage. So, we chose high wing structure for our UAV. Thirdly, we discussed types of tail configurations such as; conventional Y-tail, V-tail and twin-boom tail. Among these three options the most advantegous one is conventional Y-tail because it has the ease and efficiency over the control systems where it is one of the key points we must achieve. Also, it is more convenient to manfacture so, we chose conventional Y-tail for our tail design. Lastly, we considered number of propellers and their locations. We examined several examples and concluded that it will be enough to use one propellar since our UAV will be low-speed fixed wing UAV and it will provide sufficient maneuverability. For the location, we analized two options where the propeller would be on the front or the rear. The former we decided on because if propeller put on rear, it creates distorted airflow which create vibration and uncontrollability. After deciding on main configurations we investigated certain airfoils which are Kline-Fogleman Airfoils (KF) and NACA2411. After analizing their advantages and disadvantages we decided that it will be more convinient to use NACA2411 because KF airfoils are more suitable for gliders. NACA2411 airfoil fitted the goals of the missions, we agreed to use in our UAV. Two types of landing gears were discussed which are conventional type and the tricaycle. In order to protect the tail when landing, we chose tricycle. As we analyzed and decided on main parts of the UAV, we also searched what would be more adventageous for image processing and location data. We found out that it would be more fast and accurate if used a camera so, we will have the camera located in our fuselage. Moreover, we debated on locating the GPS inside the fuselage or outside of the fuselage. The option that will benefit us is locating the GPS outside of the body which eliminate the possibility of signal distruption. After evaluating the requirements of the missions, the important points and the configurations that we chose to benefit us, we proceeded to design the UAV.

2.2 Gövde ve Mekanik Sistemler

We designed all parts of the UAV by ourselves. We created the fuselage as a combination of plates produced from sheet balsa or carbon fiber and covered with airplane coating film or flexible plywood. Our fuselage will consist of 2 floors; at the bottom floor, drop mechanism and camera will take place, other electronic parts

Şekil 3 İHA Yan Profili

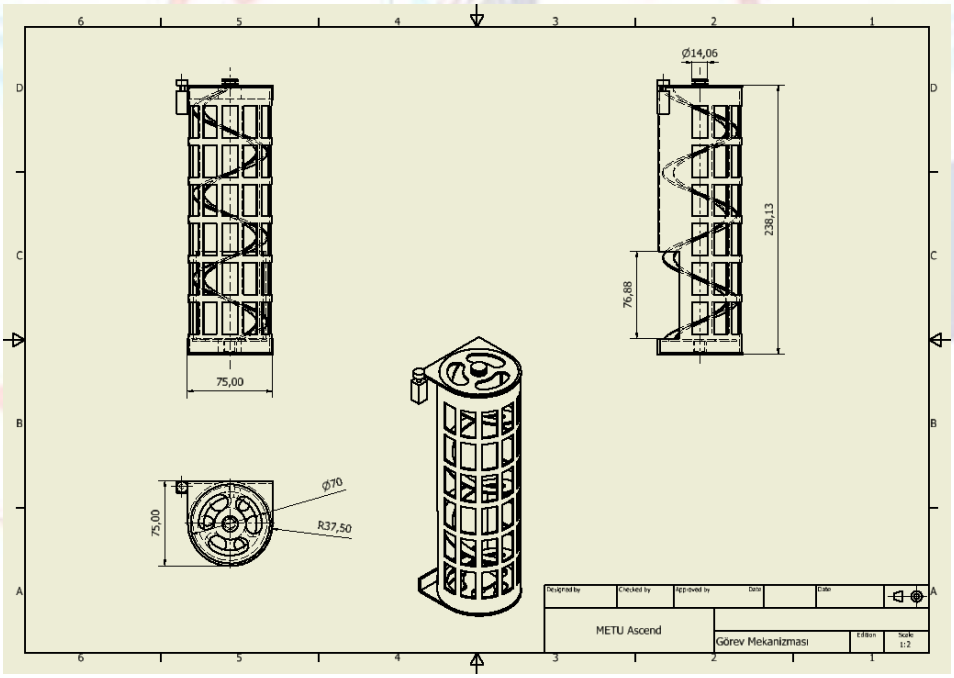
like flight card, sensors, battery, etc. of the plane will take place on the top floor. We designed the nose to drop-down if opening the fuselage would be necessary, and by doing this , we plan to easily reach inside of the fuselage. For the wings, we picked NACA 2411 for the main wing and NACA 0009 for the vertical and horizontal stabilizers. Both tail and main wing also will be covered with airplane coating films. The wings will be attached with 2 rods to the fuselage; also, the horizontal stabilizer will attach to the fuselage from the back.



Şekil 4 İHA Ön Profili

We also designed the ball drop mechanism, and we will produce it from a 3D printer. The material opptions for the ball drop mechanism are PLA and ABS. PLA is cheaper than ABS but heavier. The nose and back parts of the fuselage will also be produced from a 3D printer. We made the nose shape as smooth as possible because of the aerodynamic requirements.

2.3 Görev Mekanizması Sistemi



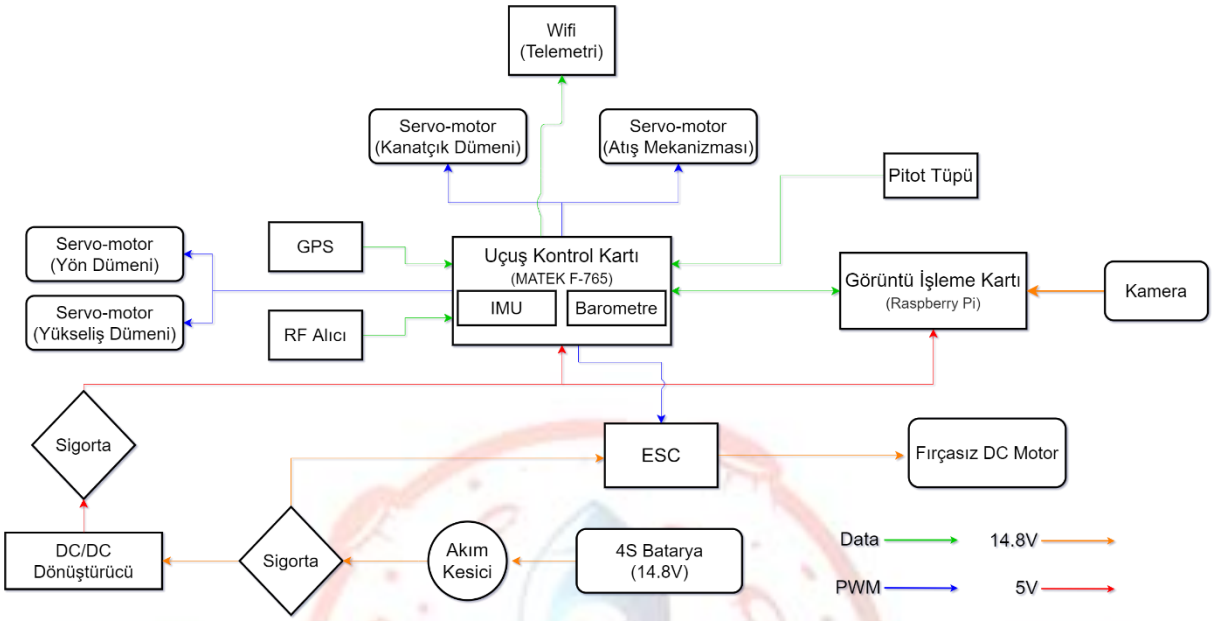
Şekil 5 Atış Mekanizması Teknik Çizimi

The task mechanism will be produced in 3D printer using ABS filament. Thus, it is aimed to be both durable and light. The total weight of the task mechanism is calculated as 96 grams. The mechanism will be located at the bottom of the fuselage.

The task mechanism is in the form of a cylindrical tube and has a capacity to accommodate 2 loads. The inner part in a spiral structure allows the loads to move easily inside the tube. When the spiral part moves counterclockwise, the loads are sent into the mechanism, while when it rotates clockwise, the loads are released from the mechanism. The spiral part is connected to the servo motor with a belt and provides its rotation with the energy it receives from the servo.

2.4 Elektrik Elektronik Kontrol ve Güç Sistemleri

2.4.1 Devre Şeması



Şekil 6 İHA Devre Şeması

UAV will be powered with a 14.8V battery. Before distributing power to the system, a circuit breaker will be placed between the electronics and battery to ensure one can cut the power immediately. ESC will be connected to the battery directly, and it will control brushless DC motor. The rest of the system will be fed through a DC/DC converter. The flight controller (MATEK F-765) and Raspberry Pi will be provided with 5V.

GPS, airspeed sensor, and barometer will be used to determine the plane's location, direction, and altitude. While flight controller control the maneuvers using servo-motors, ESC will control the speed of the brushless motor with the trapezoidal signal provided by the flight controller. Raspberry Pi will be used to process the camera's image and send the drop zone location to the flight controller to maneuver the plane. UAV will communicate through a wifi module to provide telemetry to ground station.

2.4.2 Sensörler

According to the competition rules, an autonomous flight is required additional to manual drive. To achieve a fluent autonomous system and a better manual drive, the following sensor implementations are essential.

While flying autonomously, UAV will only be relying on its onboard autopilot algorithm and its sensors. The autopilot software will use the sensor data to follow the mission path precisely. For that, GPS, IMU, and airspeed sensors will be fused with Kalman Filter, which will determine the plane's location and direction. Again with a Kalman Filter barometer and IMU will be used to check the altitude of the aircraft. The sensors' input data will be first filtered with lowpass or the appropriate filter to increase precision before processing their data.

2.4.3 Uçuş Kontrolcüsü

Controlling the UAV autonomously requires an operational unit, a microprocessor, to operate. Compared three options that are suitable for the missions with different properties in the table below. Only flight-specific microcontroller units are compared since their circuit layout is customized specifically for being used with flight electronics.

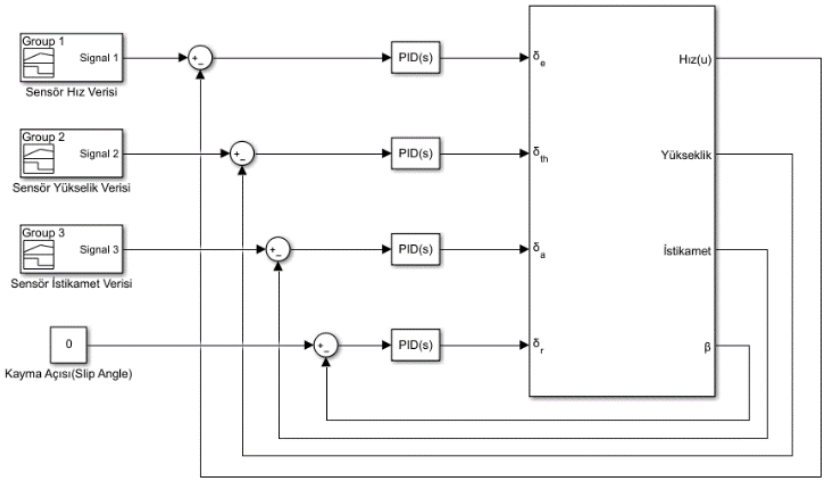
We plan to write the original autopilot control algorithm to compete in the “Yerlilik” category and help advance in milli teknoloji hamlesi. Selecting a controller, while concerning accessibility and easy to write code, we have chosen MATEKF765. It has the same processor and similar properties as commonly used Pixhawk. Since we will not use any additional features of pixhawk and F405 has an older processor, MATEK F765 will be the best option.

Uçuş Kontrolcüsü	İşlemci	Bağlantı Desteği	Haberleşme	Hazır Yazılım	Ağırlık(g)	Fiyat
Pixhawk PX4	STM32F765 32bit *(ARM M7 Cortex* *216MHz* *2MB Flash* *512KB SRAM)*	IMU(Dahili) Barometre(Dahili) GPS(Dahili) Hız Sensörü 16 PWM Servo Desteği 2 Fırçasız Motor Desteği	8 UART 3 I²C 4 SPI 2 CAN	PX4 Ardupilot Mavlink	15.8	1392TL
MATEK F405 Wing	STM32F405 32bit *(ARM M4 Cortex* *168MHz* *192KB SRAM* *1M Flash)*	IMU (Dahili) Barometre (Dahili) GPS Hız Sensörü 7 PWM Servo Desteği 2 Fırçasız Motor Desteği	6 UART 2 I²C	iNAV Mavlink	26	390TL
MATEK F765 Wing	STM32F765VI 32bit *(ARM M7 Cortex* *216MHz* *2MB Flash* *512KB SRAM)*	IMU (Dahili) Barometre (Dahili) GPS Hız Sensörü 12 PWM Servo Desteği 2 Fırçasız Motor Desteği	7 UART 2 I²C 1 SPI4	iNAV Ardupilot Mavlink	25	600TL

Tablo 1 Uçuş Kontrolcüsü Karşılaştırma Tablosu

2.4.4 Otopilot Kontrol Algoritması

As stated in 2.4.3, we are planning to write an original flight control algorithm. We are developing the control algorithm using Matlab/Simulink. Exemplified control schematic is shown in the figure below. Corresponding signals will be fed to the control system. Sensor data will be meshed for determining velocity, altitude, and heading data. A non-linear UAV model will be constructed for tuning PID coefficients. The control algorithm, communication with the ground station, communication with the raspberry pi, and failsafe mode will be written in c language and flashed into Matek F765.



$\delta_e =$ Yükseliş Dümeni Açısı (Elevator)

$\delta_{th} = Motor\ Hızı\ (Throttle)$

$\delta_a =$ Kanatçık Açısı (Aileron)

$\delta_r = Yön\ Dümeni\ Açısı\ (Rudder)$

Şekil 7 Otonom Uçuş Kontrol Şeması

2.4.5 Radyo Haberleşme

For manual communication with the UAV, we will use radio communication. We will handle authentication for the ground station and the UAV to guarantee that the signals will not interfere.

We will be using a radio controller with multiple channels so that if we have a problem with a given frequency at the competition area, we can switch to the other channels. Signals will be encrypted/decrypted to eliminate interference. Aim is to reach full autonomy with the UAV, so it is not expected to use any manual control in missions, but still, manual control can be used in landings and take-offs. For this reason, we will mostly be receiving data from the UAV to the ground station through the wifi module to see details of UAV, such as speed, altitude, and location.

2.4.6 Görüntü İşleme

In the second mission we need to drop a load to a red dropzone whose location is not determined before the competition. To tackle the problem we are planning to use a camera for video feed, a Raspberry Pi development card and write an image processing code in python. We chosen a camera and image processing program over relying on gps location that is not accurate enough. Collecting the exact location manually would be arbitary since this competition is trying to mimic rescue payload drop scenario.

Raspberry Pi and flight controller will be communicating through out the mission flights. When redzone is detected by the camera raspberry pi collects the moments telemetry data and calculates the location of the dropzone and sends it back to flight controller.

2.4.7 Sigorta ve Akım Kesici

The fuse is used to protect the circuit elements from being damaged by preventing an undesired amount of current to flows through the system. A circuit breaker is a switch that cuts the current in the circuit with a button; when pressed cuts the circuit from the battery. The fuse module, which consists of a fuse and circuit breaker, will be directly connected with the battery, and the current will be transmitted by passing through the module. The button and the fuse will be placed on top of the fuselage as stated in safety rules.

2.4.8 Güç Sistemleri

Voltage regulator and ESC were deemed necessary as power modules to be used in the UAV. We have chosen Matek Mini Power Distribution Board with 5V / 12V outputs is connected to the 14.8V battery or stability and different voltage distribituons.

While selecting the ESC for the brushless motor, the ESC current value is selected so that it is 10% -20% more than operating value, taking into account the burst current of the motor. With a safety margin, 60A ESC will be selected.

2.4.9 Batarya Sistemleri

For battery there are two reliable chooses which are Li-Ion and Li-Po. Li-po battery is chosen due to its robust and light weight characteristics are suitable for an UAV. Based on the flight time of 25 minutes, the capacity of the battery will be selected between 7000-10000 mAh and the C rating between 30-60.

The optimal battery values will be determined according to the results after the tests. Battery pack will be determined after the brushless motor is designed, which has the most effect on the power consumption.

2.5 İtki ve Taşıma Hesapları

The components of our UAV skeleton consists of a fuselage, rectangular wing located on top of the fuselage, a conventional tail connected to the fuselage with two rods and a ball drop mechanism inside the lower base of the fuselage. Which made the skeleton weight 1.286kg total. In addition to our skeleton we also placed the Raspberry Pi, servo motors, ESC, battery, power module and fuse module inside the top base of our fuselage according to the center of gravity of our UAV (which should be located near the wing). The camera required for the missions were placed on the bottom of the fuselage, in front of the lid of the ball drop mechanism. According to our design, we considered the aerodynamic effectiveness and located our motor and propeller in front of the fuselage. With all these components the take-off weight of our UAV is approximately 2.8 kg.

Brushless DC Motor	KV	Estimated Propeller Size	Voltage	Estimated Current	Estimated Wattage	Estimated Thrust	Estimated RPM
3820	960	12x8E	12.6 V	49 A	617 W	2.5 kg	9,170
3820	1200	10x6	12.6 V	47 A	592 W	2.0 kg	11,900

Our brushless DC motor, which is in the design phase, is expected to be close to the specifications mentioned above. These values will be used when calculating the estimated power consumption. From the two options above, motor properties with a thrust of 2.5 kg have been selected to be used in power consumption calculations. Depending on the developments, differences in engine characteristics are expected.

During the flight, motor throttle will be approximately used at %50 and estimated power consumption will be 350 W.

Servo motors to be used in UAV provide 1,3 Kg.cm torque and 0.12 / 60 degree speed at 4.8 V. There will be 4 servo motors in UAV and each will consume approximately 1 W.

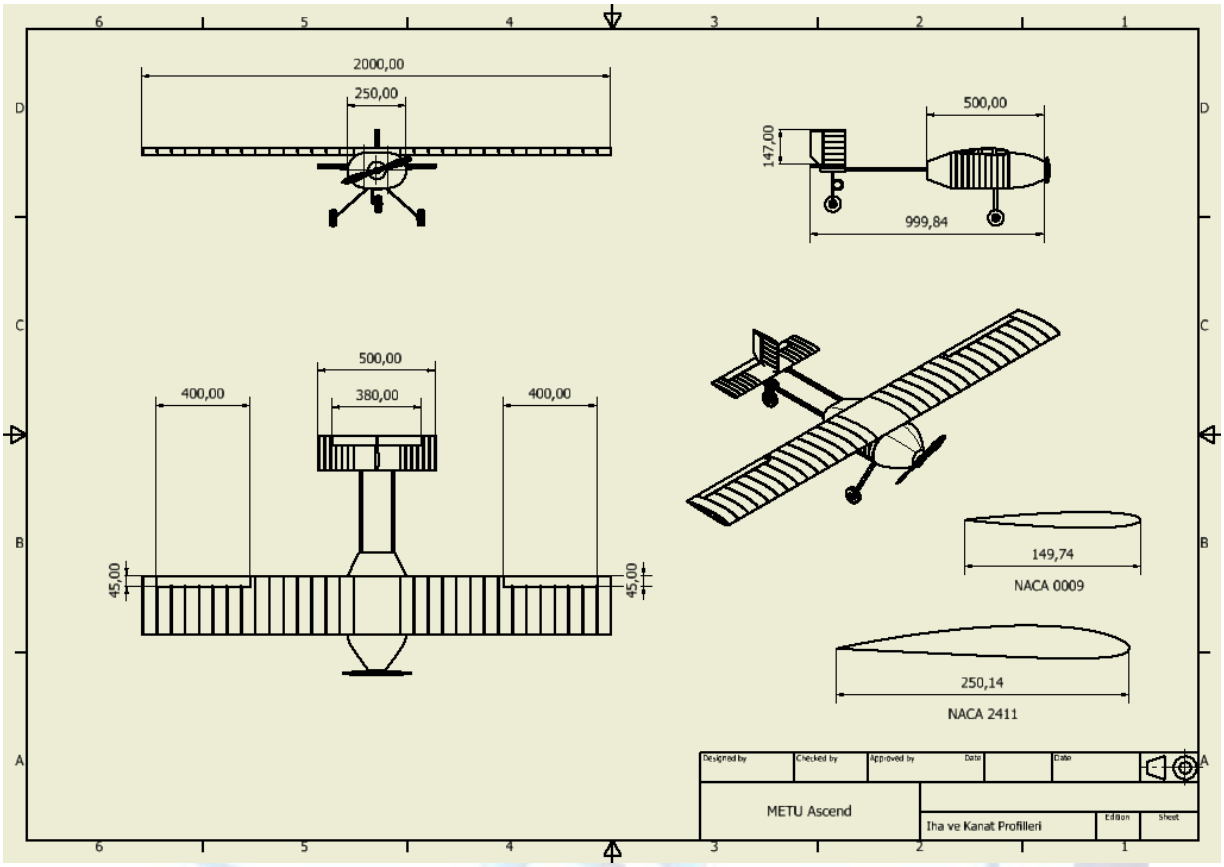
Total power consumption = Motor + Rasberry Pi 4 4GB + Matek F765-Wing + Camera + Servo

Total power consumption = 350W + 5W + 5W + 3W + 4W = 367 W

Currently Limskey 4S,14.8V, 10400mah,30C,893g battery has been selected for estimated calculations. It has a power of 10.4*14.8 = 153.92Wh and power density of 153.92/0.893= 172.363 Wh/kg

Estimated total power is 367W and battery has power of 153.92 Wh, so estimated flight time is 25.1 minutes.

2.6 Görsel Tasarım Konfigürasyonu



Şekil 8 İHA Teknik Çizimi

REFFERENCES WILL BE ADDED

Retrived from: https://blog.ravpower.com/2017/06/lithium-ion-vs-lithium-polymer-batteries/