

MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF ELECTRICAL AND ELECTRONICS  
ENGINEERING

---

**EE300 Summer Practice I  
Report**

---

**Student Name:**

*Halil Temurtaş*

**Student ID:**

*2094522*

**SP Beginning**

**Date:**

*03.07.2017*

**SP End Date:**

*28.07.2017*

**SP Company Name:**

*TÜRKSAT A.Ş.*

**SP Company Division:**

*Directorate of Satellite Programming*

**Supervisor Engineer:**

*Ömer Eren Can Koçulu*

**SE Contact Info:**

*ekoculu@turksat.com.tr*

.. .. 2017

# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>                               | <b>3</b>  |
| <b>2</b> | <b>Description of the Company</b>                 | <b>4</b>  |
| 2.1      | Company Name . . . . .                            | 4         |
| 2.2      | Company Location . . . . .                        | 4         |
| 2.3      | General Description of the Company . . . . .      | 4         |
| 2.4      | The Organizational Chart of the Company . . . . . | 5         |
| 2.5      | A Brief History of the Company . . . . .          | 6         |
| <b>3</b> | <b>Orientation &amp; Useful Programs</b>          | <b>7</b>  |
| 3.1      | Pomodoro Technique . . . . .                      | 7         |
| 3.1.1    | Pomotodo App . . . . .                            | 8         |
| 3.2      | Database Structure . . . . .                      | 10        |
| 3.2.1    | Airtable . . . . .                                | 10        |
| 3.3      | Wiki Pages . . . . .                              | 11        |
| 3.3.1    | Confluence Wiki . . . . .                         | 12        |
| 3.4      | V-Model & Agile Methodology . . . . .             | 12        |
| 3.4.1    | V-Model . . . . .                                 | 12        |
| 3.4.2    | Agile Methodology . . . . .                       | 14        |
| 3.4.2.1  | Roles . . . . .                                   | 14        |
| 3.5      | Version Control with Git . . . . .                | 14        |
| 3.5.1    | Github . . . . .                                  | 15        |
| 3.5.2    | Bitbucket . . . . .                               | 15        |
| <b>4</b> | <b>Solar Tracker System Project</b>               | <b>15</b> |
| 4.1      | Planning & Researching . . . . .                  | 15        |
| 4.1.1    | System Requirements . . . . .                     | 16        |
| 4.1.2    | Subsystem Requirements . . . . .                  | 16        |
| 4.1.3    | Component Requirements . . . . .                  | 17        |
| 4.1.4    | Components . . . . .                              | 17        |
| 4.2      | Training . . . . .                                | 18        |
| 4.2.1    | Training on Python . . . . .                      | 18        |
| 4.2.1.1  | Basics . . . . .                                  | 19        |
| 4.2.1.2  | Using Conditions . . . . .                        | 20        |
| 4.2.1.3  | Using Loops . . . . .                             | 20        |
| 4.2.1.4  | Defining Functions . . . . .                      | 21        |

|          |   |           |
|----------|---|-----------|
| 4.2.1.5  | Defining Classes . . . . .                  | 21        |
| 4.2.2    | Training on Raspberry Pi . . . . .          | 22        |
| 4.2.2.1  | Training on LEDs . . . . .                  | 22        |
| 4.2.2.2  | Training on LDRs . . . . .                  | 22        |
| 4.2.2.3  | Training on Servo Motors with Raspberry Pi  | 23        |
| 4.2.3    | Training on Arduino . . . . .               | 23        |
| 4.2.3.1  | Training on Servo Motors with Arduino . . . | 23        |
| 4.3      | Project Coding . . . . .                    | 24        |
| 4.3.1    | Raspberry Pi Part . . . . .                 | 24        |
| 4.3.2    | Arduino Part . . . . .                      | 26        |
| 4.4      | Implementation . . . . .                    | 28        |
| 4.4.1    | PCB Drawing & 3D Drawings . . . . .         | 28        |
| 4.4.1.1  | PCB Drawing . . . . .                       | 28        |
| 4.4.1.2  | 3D Drawings . . . . .                       | 28        |
| 4.4.2    | Construction of the Body . . . . .          | 28        |
| 4.4.2.1  | Top Layer . . . . .                         | 28        |
| 4.4.2.2  | Main Body . . . . .                         | 29        |
| 4.4.2.3  | Solar Panel . . . . .                       | 29        |
| 4.4.2.4  | Final Body . . . . .                        | 30        |
| 4.5      | Tests . . . . .                             | 30        |
| 4.5.1    | System Requirement Tests . . . . .          | 30        |
| 4.5.2    | Subsystem Requirement Tests . . . . .       | 31        |
| 4.5.3    | Component Requirement Tests . . . . .       | 31        |
| <b>5</b> | <b>After Project</b>                        | <b>31</b> |
| 5.1      | Training on MATLAB . . . . .                | 31        |
| 5.1.1    | Coursera . . . . .                          | 31        |
| 5.1.2    | Outline of the Course . . . . .             | 32        |
| 5.1.2.1  | Simple Sorting Code . . . . .               | 33        |
| 5.2      | Training on Microsoft Sharepoint . . . . .  | 35        |
| 5.2.1    | Microsoft Sharepoint . . . . .              | 35        |
| <b>6</b> | <b>Conclusion</b>                           | <b>35</b> |
| <b>7</b> | <b>References</b>                           | <b>35</b> |

# 1 Introduction

I have performed my summer practice in TÜRKSAT A.Ş. (Türksat Satellite Communications and Cable TV Operations Company - Türksat Uydu Haberleşme Kablo TV ve İşletme A.Ş). It is the sole communications satellite operator in Turkey. My internship lasted 20 days. Ömer Eren Koçulu, a mechatronics engineer in TURKSAT was our supervisor and he managed our internship program.

My internship started with an orientation program. The company and how works are handled were presented to new interns. After that, the programs and techniques we would use in our internship and our work life were introduced. Following this introduction, a project is assigned to us as a team. Our team consisted of me and two mechatronics engineering students, Abdullah Taha İzmir and Duran Arif Göçer.

The project was about solar panels that can follow sun to increase its efficiency. In order to achieve this, we were recommended to use Raspberry Pi instead of Arduino since other team were using Arduino in their project. Moreover, we could compare the efficiency of using Raspberry and Arduino at the end. For controlling Raspberry Pi, I learnt the basics of Python and Linux environment. Lastly, I studied on Matlab, MS Sharepoint after finishing project.

In this report, I start with an introduction

...  
...  
...  
...  
...  
...  
...  
...  
...  
...  
...  
...  
...  
...  
...

## 2 Description of the Company

In this chapter, I will introduce the company in four parts:

1. Company Name
2. Company Location
3. General Description of the Company
4. A Brief History of the Company

### 2.1 Company Name

TÜRKSAT A.Ş. (Türksat Satellite Communications and Cable TV Operations Company - Türksat Uydu Haberleşme Kablo TV ve İşletme A.Ş).

### 2.2 Company Location

**Address-1: Ana Kampüs:** Konya Yolu 40 KM. Gölbaşı/Ankara/Türkiye

**Address-2: Gazi Teknokent:** Bahçelievler Mahallesi, Gazi Üniv. Gölbaşı Yerleşkesi No:24, 06830 Gölbaşı/Ankara/Türkiye

**Phone:** +90 312 615 3000

**Fax:** +90 312 499 5115

### 2.3 General Description of the Company

Türksat Satellite Communications and Cable TV Operations Company is the sole communications satellite operator in Turkey. It was established on 21 December 1990 as a state-owned company named Türksat Milli Haberleşme Uyduları (Türksat National Communications Satellites) in Gölbaşı, Ankara Province; eventually incorporating the satellite services of Türk Telekomünikasyon A.Ş. and becoming Türksat A.Ş. on 22 July 2004. Türksat A.Ş. also owns 100% of the shares of Eurasiasat S.A.M., jointly established as a spin-off company with Aérospatiale in 1996 to manufacture and launch Turksat 2A (Eurasiasat 1) in 2001.

## 2.4 The Organizational Chart of the Company

The organizational chart of TÜRKSAT can be seen in Figure 1.

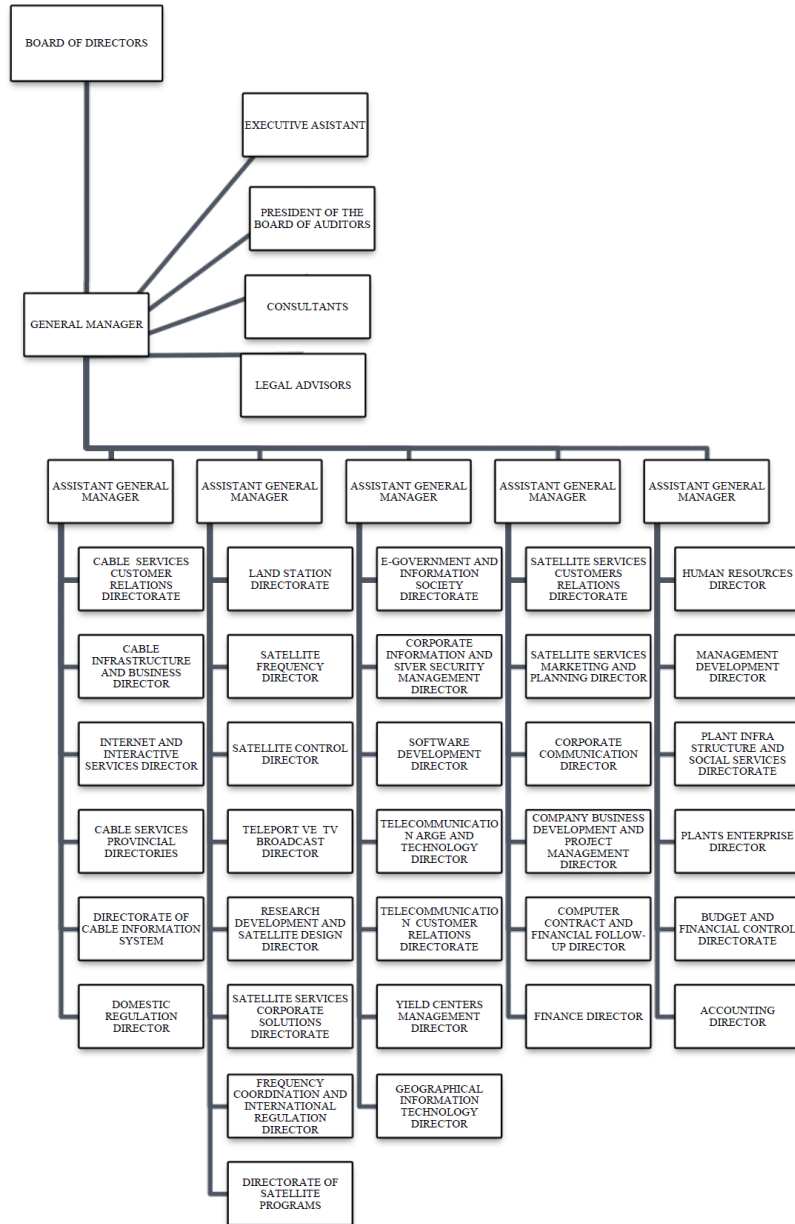


Figure 1: The Organizational Chart of TÜRKSAT

## 2.5 A Brief History of the Company

- **1968**

The Chief Engineering of Satellite Telecommunications Group was established within the General Directorate of PTT.

- **August 11th, 1994**

Turkey's Türksat 1B satellite was launched and put successfully into 42° East orbit.

- **July 10th, 1996**

Turkey's second satellite, Türksat 1C, was launched and put into 31.3° E orbit.

- **January 11th, 2001**

Türksat 2A (Eurasiasat 1) satellite manufactured by Eurasiasat company established in partnership with Türk Telekom and Alcatel company launched by Ariane 4 rocket from Kourou Base in South America.

- **July 22nd, 2004**

In order to conduct satellite communication services, which was previously conducted by Türk Telekomünikasyon A.Ş., under a new company, *Türksat A.Ş.* was founded by Law no. 5189.

- **June 13th, 2008**

Türksat 3A satellite launched from the French Guiana on June 13th, 2008 at 01:05 by Ariane 5 rocket and put into 42.0° East orbit.

- **February 14th, 2014**

Turksat 4A communication satellite launched by Proton rocket from Baikonur Cosmodrome in Kazakhstan.

- **October 16th, 2015**

Turksat 4B communication satellite launched by Proton Breeze M vehicle from Baikonur Cosmodrome in Kazakhstan and put into 50° East orbit.

## 3 Orientation & Useful Programs

Throughout my summer practice, I used several techniques and useful programs recommended by our supervisor.

In this section, I will explain these techniques and programs that I found very useful.

### 3.1 Pomodoro Technique

The Pomodoro Technique is a time management method developed by Francesco Cirillo in the late 1980s. The technique aims to increase efficiency by breaking work hours into several intervals called pomodoro. Originally 25 minutes in length, separated by short breaks, the length of these intervals can be changed to suit people's personalities. For example, I have used 40 minutes length pomodoros, 5 minutes length short breaks and 1 hour length long break after 4 or 5 pomodoros. Pomodoros (tomatoes in Italian) are named after the tomato-shaped kitchen timer that Cirillo used as a university student.

The technique is closely related to software design concepts such as incremental development and iterative and timeboxing, and has been adopted in pair programming contexts.

**There are six steps in the technique:**

1. *Decide on the task to be done.*
2. *Set the pomodoro timer (traditionally to 25 minutes).*
3. *Work on the task until the timer rings.*
4. *After the timer rings put a checkmark on a piece of paper.*
5. *If you have fewer than four checkmarks, take a short break (3–5 minutes), then go to step 2.*
6. *After four pomodoros, take a longer break (15–30 minutes), reset your checkmark count to zero, then go to step 1.*

A goal of the technique is to reduce the impact of internal and external interruptions on focus and flow. A pomodoro is indivisible which means it can not be interrupted. When interrupted during a pomodoro, either the other activity must be recorded and postponed (inform – negotiate – schedule – call back) or the pomodoro must be abandoned.



### 3.1.1 Pomotodo App

Although the creator of this technique encourages a low-tech approach that includes using a mechanical timer, paper and pencil. We have used more technological solutions called Pomotodo App in my summer practice.

The reason behind this decision was to increase efficiency even mre by using Pomotodo’s some key features like built-in to-do list & category tracking system.

The stages of planning, tracking, recording, processing and visualizing are fundamental to the technique. In the planning phase tasks are prioritized by recording them in a ”To Do Today” list. This enables users to estimate the effort tasks require. As pomodoros are completed, they are recorded, adding to a sense of accomplishment and providing raw data for self-observation and improvement. For that purpose, I have used Pomotodo’s builtin to-do list that enables user not just tracking its work but allows user to categorise work by some cathegories. Some of my to-do list objects can be seen in figure XX.

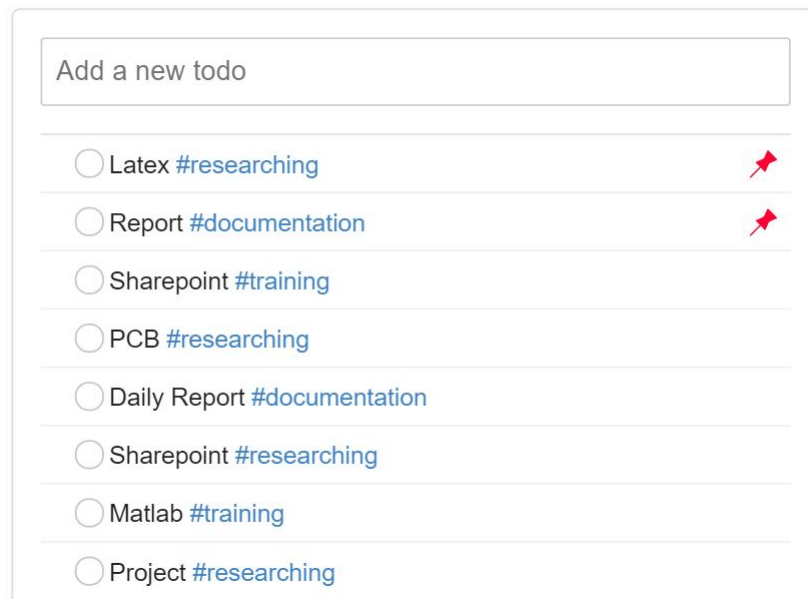


Figure 2: My To Do List in Pomotodo Web App

|                          |               |   |
|--------------------------|---------------|---|
| Jul 12 Wed               | 17:20 - 17:55 | Project #researching                            |
| Finished 10 pomos        | 16:45 - 17:20 | Project #researching                            |
| Total 5 hours 52 minutes | 16:13 - 16:38 | Raspberry #training (Manually)                  |
|                          | 15:30 - 16:10 | Raspberry Arduino Git #training (Manually)      |
|                          | 14:35 - 15:10 | Arduino #training (Manually)                    |
|                          | 13:55 - 14:30 | Raspberry Arduino Git #training (Manually)      |
|                          | 12:05 - 12:40 | Raspberry Arduino Servo #researching (Manually) |
|                          | 11:28 - 12:03 | Raspberry Servo #training (Manually)            |
|                          | 10:43 - 11:24 | Project #researching                            |
|                          | 10:03 - 10:40 | Raspberry Servo #training (Manually)            |

Figure 3: My Pomodoro History of July 12th

As can be seen in figure XX, I have used some hashtags to categorise the work I have done. As can be understood from figure, 10 pomodoros were completed at July 12th. As I mentioned earlier, I have tried to use my pomodoro length as a 40 minutes and short breaks as 5 minutes. After 5 completed pomodoros, a long break was taken. After using this hashtags, we can ingestive our work statistic for desired times. For instance, throughout my summer practice 66% of my time was spent on training. Further statics can be seen at figure XX.

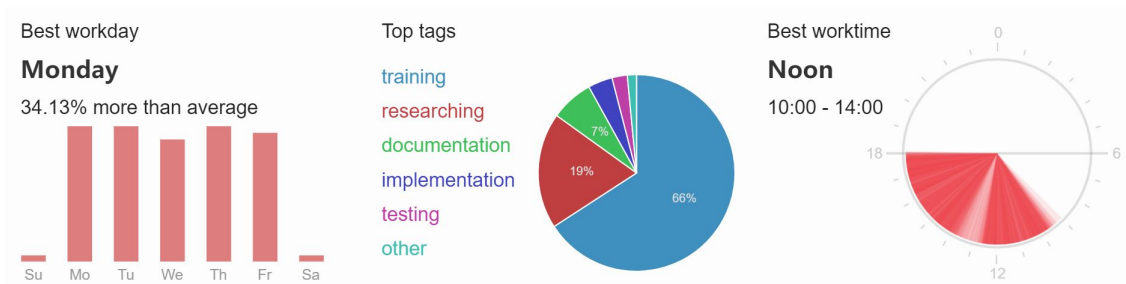


Figure 4: Some statics about my summer practice

## 3.2 Database Structure

A database is an organized collection of data. It is the collection of schemas, tables, queries, reports, views, and other objects. The data are typically organized to model aspects of reality in a way that supports processes requiring information, such as modelling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

Formally, a "database" refers to a set of related data and the way it is organized. Access to this data is usually provided by a "database management system" (DBMS) consisting of an integrated set of computer software that allows users to interact with one or more databases and provides access to all of the data contained in the database (although restrictions may exist that limit access to particular data). The DBMS provides various functions that allow entry, storage and retrieval of large quantities of information and provides ways to manage how that information is organized. Because of the close relationship between them, the term "database" is often used casually to refer to both a database and the DBMS used to manipulate it. Outside the world of professional information technology, the term database is often used to refer to any collection of related data (such as a spreadsheet or a card index). This article is concerned only with databases where the size and usage requirements necessitate use of a database management system.

### 3.2.1 Airtable

Airtable is a spreadsheet-database hybrid i.e., the features of a database are applied to a spreadsheet. The fields in an Airtable table are similar to a cell of a spreadsheet, but have types check-boxes, phone numbers, and drop-down lists, and can reference file attachments like images. Users can create a database, set up field types, add records, link tables, collaborate with a team, sort the records based on a field and publish views to external websites. When an Airtable database is created, it is automatically hosted to the cloud. The values in the fields are updated real time.

Airtable has six basic components:

**Bases:** All the information needed to create a project is contained in a Base. Bases can be built from existing templates provided by Airtable. In addition, they can also be built from scratch, from a spreadsheet or from an existing Base.

**Tables:** A table is similar to an excel spreadsheet. A Base is a collection

of tables.

An example table in a restaurant Base.

Views: Views are how we can see a table. Views can be saved for future purposes.

Fields: Each entry in a Table is a field. They are not just restricted to hold text. Airtable currently offers 16 basic field types. These are: single-line texts, long text articles, file attachments, check-boxes, single select from drop-down list, multiple-selects from drop-down lists, date and time, phone numbers, email ids, URLs, numbers, currency, percentage, auto-number, formulae and barcodes.

Records: Each row of a Table is a Record.

Team: Team is a collection of Bases in Airtable. For example, in the adjacent restaurant template which contains all the information we need to store about the restaurants. We can have a ‘Restaurants’ table to store the names of restaurants along with information about their addresses, ratings, menus, etc. We can have a view to show our favourite restaurants. Each record in the Restaurants table is kept for a particular restaurant. ‘Rating’ can be kept as a field, to help generate ‘My favourite restaurants’ view.

### 3.3 Wiki Pages

A wiki is a website on which users collaboratively modify content and structure directly from the web browser. In a typical wiki, text is written using a simplified markup language and often edited with the help of a rich-text editor.

A wiki is run using wiki software, otherwise known as a wiki engine. A wiki engine is a type of content management system, but it differs from most other such systems, including blog software, in that the content is created without any defined owner or leader, and wikis have little implicit structure, allowing structure to emerge according to the needs of the users.

There are dozens of different wiki engines in use, both standalone and part of other software, such as bug tracking systems. Some wiki engines are open source, whereas others are proprietary. Some permit control over different functions (levels of access); for example, editing rights may permit changing, adding or removing material. Others may permit access without enforcing access control. Other rules may be imposed to organize content.

### 3.3.1 Confluence Wiki

The screenshot displays two pages from a Confluence Wiki. Each page is divided into three main sections: Pomotodo Daily Report, Notes, and To-Do List.

**Page 1 (05 Jul 2017):**

- Pomotodo Daily Report:** Shows a table of activities for Wednesday, July 5th. Activities include 'Project #researching' and 'Orientation #training' with specific time slots. A total of 10 pomos (16 minutes) is reported.
- Notes:** A bulleted list of notes, including 'Worked on Airtable with other team as an orientation', 'I have searched projects similar to our project', 'Using LDR's learned.', and 'Initial thoughts on projects design finalized'.
- To-Do List:** A list of tasks with checkboxes, including 'Python' and 'Raspberry Pi', both attributed to '@Halil Temurtas'.

**Page 2 (06 Jul 2017):**

- Pomotodo Daily Report:** Shows a table of activities for Thursday, July 6th. Activities include 'Python #training' with specific time slots. A total of 8 pomos (8 minutes) is reported.
- Notes:** A bulleted list of notes, including 'Worked on Python tutorials.' and 'GitHub Desktop and PyCharm are tested.' Below the notes is a section titled 'Learn the Basics' with a list of topics like 'Hello, World', 'Variables and Types', 'Lists', etc.
- To-Do List:** A list of tasks with checkboxes, including 'Git-GitHub', attributed to '@Halil Temurtas'.

Figure 5: Body

There are dozens of different wiki engines in use, both standalone and part of other software, such as bug tracking systems. Some wiki engines are open source, whereas others are proprietary. Some permit control over different functions (levels of access); for example, editing rights may permit changing, adding or removing material. Others may permit access without enforcing access control. Other rules may be imposed to organize content.

## 3.4 V-Model & Agile Methodology

### 3.4.1 V-Model

The V-model is a graphical representation of a systems development lifecycle. It is used to produce rigorous development lifecycle models and project man-

agement models. The V-model falls into three broad categories, the German Das V-Modell, a general testing model and the US government standard. The V-model summarizes the main steps to be taken in conjunction with the corresponding deliverables within computerized system validation framework, or project life cycle development. It describes the activities to be performed and the results that have to be produced during product development.

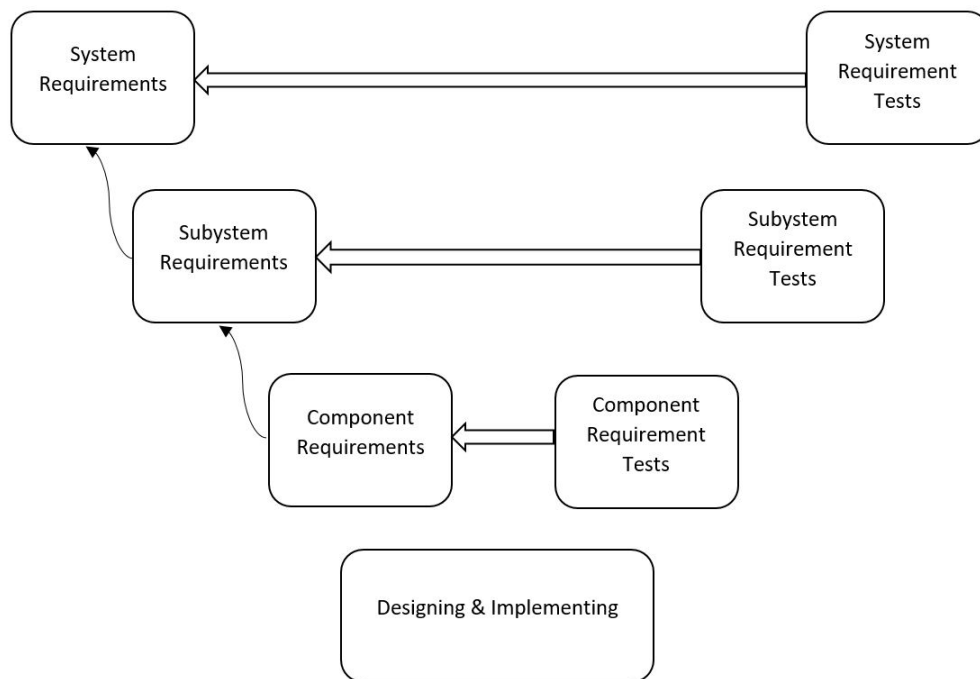


Figure 6: V-Model

The left side of the "V" represents the decomposition of requirements, and creation of system specifications. The right side of the V represents integration of parts and their validation. However, Requirements need to be validated first against the higher level requirements or user needs. Furthermore, there is also something as validation of system models (e.g. FEM). This can partially be done at the left side also. To claim that validation only occurs at the right side may not be correct. The easiest way is to say that verification is always against the requirements (technical terms) and validation always against the real world or the user needs

### 3.4.2 Agile Methodology

Agile software development describes a set of values and principles for software development under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams. It advocates adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change. The term agile (sometimes written Agile) was popularized by the Agile Manifesto, which defines those values and principles. Agile software development frameworks continue to evolve, two of the most widely used being Scrum and Kanban.

#### 3.4.2.1 Roles

**Product Owner** : The team leader, the person responsible for tracking the process.

**Scrum Master** : The person responsible for the correct execution of the process.

**Hardware Engineer** : The person or people that are responsible for designing and implementing the electrical and electronics hardware.

**Software Engineer** : The person or people that are responsible

**Structure Engineer** : The person or people that are responsible

**Test Engineer** : The person or people that are responsible

## 3.5 Version Control with Git

Git is a version control system (VCS) for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files. As a distributed revision control system it is aimed at speed, data integrity, and support for distributed, non-linear workflows. Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development. Its current maintainer since 2005 is Junio Hamano. As with most other distributed version control systems, and unlike most client-server systems, every Git directory on every computer is a full-fledged repository with complete history and full version tracking abilities, independent of network access or a central server. Like the Linux kernel,

Git is free software distributed under the terms of the GNU General Public License version 2.

### **3.5.1 Github**

### **3.5.2 Bitbucket**

Bitbucket is a web-based hosting service that is owned by Atlassian, used for source code and development projects that use either Mercurial (since launch) or Git (since October 2011) revision control systems. Bitbucket offers both commercial plans and free accounts. It offers free accounts with an unlimited number of private repositories (which can have up to five users in the case of free accounts) as of September 2010. Bitbucket integrates with other Atlassian software like Jira, HipChat, Confluence and Bamboo. It is similar to GitHub, which primarily uses Git. Bitbucket has traditionally tailored itself towards helping professional developers with private proprietary code, especially since being acquired by Atlassian in 2010. In September 2016, Bitbucket announced it had reached 5 million developers and 900,000 teams on its platform. Bitbucket has 3 deployment models: Cloud, Bitbucket Server and Data Center.

sdfdfsdfs

## **4 Solar Tracker System Project**

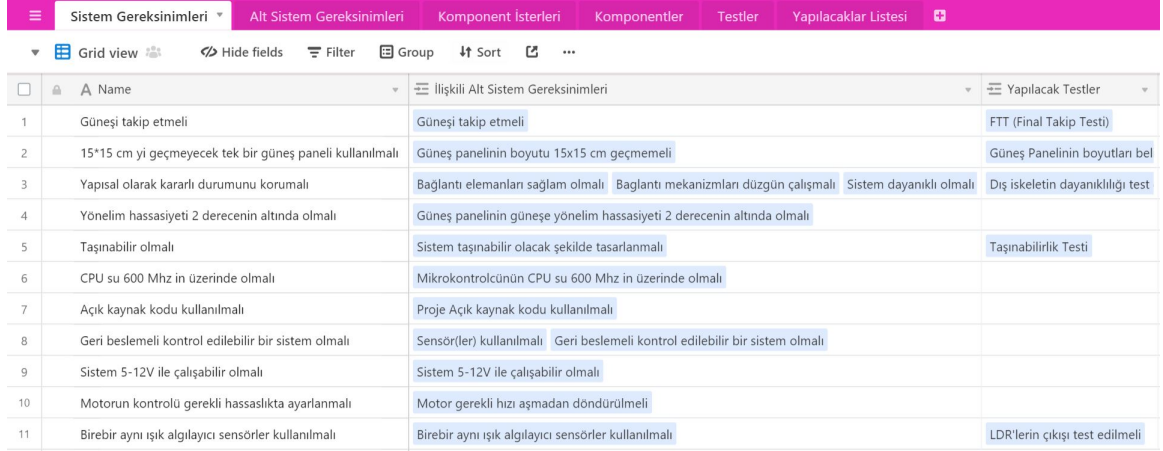
In my summer practice, I was assigned for a project with a team. For the project, we were expected to build a solar panel system that can follow the sun light to maximize its efficiency.

### **4.1 Planning & Researching**

As planning the project, we used V-model and Agile Methodology (Scrum) in order to increase efficiency and reduce time spent on the project. As mentioned earlier, using V-model required using another program. Therefore, we decided to use Airtable for tracking system requirements, subsystem requirements, tests and so on. The Interface of Airtable & System Requirements can be seen at figure X.



## 4.1.1 System Requirements



|    | Name   | İlişkili Alt Sistem Gereksinimleri  | Yapılacak Testler                |
|----|--|---|----------------------------------|
| 1  | Güneşi takip etmeli                                      | Güneşi takip etmeli   | FTT (Final Takip Testi)          |
| 2  | 15*15 cm yi geçmeyecek tek bir güneş paneli kullanılmalı | Güneş panelinin boyutu 15x15 cm geçmemeli   | Güneş Panelinin boyutları bel    |
| 3  | Yapısal olarak kararlı durumunu korumalı                 | Bağlantı elemanları sağlam olmalı Bağlantı mekanizmaları düzgün çalışmalı Sistem dayanıklı olmalı | Diş iskeletin dayanıklılığı test |
| 4  | Yönelim hassasiyeti 2 derecenin altında olmalı           | Güneş panelinin güneşe yönelim hassasiyeti 2 derecenin altında olmalı                             |                                  |
| 5  | Taşınabilir olmalı                                       | Sistem taşınabilir olacak şekilde tasarlanmalı  | Taşınabilirlik Testi             |
| 6  | CPU su 600 Mhz in üzerinde olmalı                        | Mikrokontrolcünün CPU su 600 Mhz in üzerinde olmalı   |                                  |
| 7  | Açık kaynak kodu kullanılmalı                            | Proje Açık kaynak kodu kullanılmalı   |                                  |
| 8  | Geri beslemeli kontrol edilebilir bir sistem olmalı      | Sensör(ler) kullanılmalı Geri beslemeli kontrol edilebilir bir sistem olmalı                      |                                  |
| 9  | Sistem 5-12V ile çalışabilir olmalı                      | Sistem 5-12V ile çalışabilir olmalı   |                                  |
| 10 | Motorun kontrolü gerekli hassaslıkta ayarlanmalı         | Motor gerekli hızı aşmadan döndürülmeli   |                                  |
| 11 | Birebir aynı ışık algılayıcı sensörler kullanılmalı      | Birebir aynı ışık algılayıcı sensörler kullanılmalı   | LDR'lerin çıkışı test edilmeli   |

Figure 7: The Interface of Airtable & System Requirements

Constructing V-model required to specify the requirements that defines the project. For the system requirements, we considered the most basic requirements that the project must fulfil. For instance, being portable was a primary purpose for our project and it became one system requirements. From the nature of V-model, every system requirement has one or more subsystem requirement and system requirement test that will be explained later. Our project had 11 system requirements as can be seen at *Figure 7*.

## 4.1.2 Subsystem Requirements

As mentioned just above, every system requirement has one or more subsystem requirement that detail the requirement. As the V-model suggests, for fulfilling the system requirements, its subsystem requirements must be fulfilled first. These subsystem requirements can be considered as secondary goals that the project trying to accomplish in order to succeed its primary goals. As can be seen at *Figure 8*, we had 14 subsystem requirements for finalizing the project.

|    | A Name  | ... | Alt Sistem Türü                     | Yapılacak Testler  |
|----|---|-----|-------------------------------------|--|
| 1  | Güneş takip etmeli  | ✓   | Elektrik-Elektronik Mekanik Yazılım | FTT (Final Takip Testi)  |
| 2  | Güneş panelinin boyutu 15x15 cm geçmemeli                             | ✓   | Mekanik                             | Güneş Panelinin boyutları belirlenmeli   |
| 3  | Sistem dayanıklı olmalı   |     | Mekanik                             | Bağlantı elemanlarının sağlamlığının test edilmesi Dış iskeletin dayanıklılığı test edilmeli |
| 4  | Bağlantı mekanizmaları düzgün çalışmalı                               |     | Mekanik                             | Bağlantı elemanlarının sağlamlığının test edilmesi   |
| 5  | Bağlantı elemanları sağlam olmalı                                     |     | Mekanik Elektrik-Elektronik         | Bağlantı elemanlarının sağlamlığının test edilmesi   |
| 6  | Güneş panelinin güneşe yönelim hassasiyeti 2 derecenin altında olmalı | ✓   | Elektrik-Elektronik Yazılım Kontrol |  |
| 7  | Sistem taşınabilir olacak şekilde tasarlanmalı                        | ✓   | Mekanik Elektrik-Elektronik         | Taşınabilirlik Testi   |
| 8  | Mikrokontrolcünün CPU su 600 Mhz in üzerinde olmalı                   |     | Elektrik-Elektronik                 |  |
| 9  | Proje Açık kaynak kodu kullanılmalı                                   | ✓   | Yazılım                             |  |
| 10 | Geri beslemeli kontrol edilebilir bir sistem olmalı                   | ✓   | Elektrik-Elektronik Kontrol Yazılım |  |
| 11 | Sensör(ler) kullanılmalı  | ✓   | Elektrik-Elektronik Yazılım         | Tasarımda sensör kullanımını denetleme   |
| 12 | Sistem 5-12V ile çalışabilir olmalı                                   | ✓   | Elektrik-Elektronik                 |  |
| 13 | Motor gerekli hızı aşmadan döndürülmeli                               | ✓   | Yazılım Elektrik-Elektronik         |  |
| 14 | Birebir aynı ışık algılayıcı sensörler kullanılmalı                   | ✓   | Mekanik                             | LDR'lerin çıkışı test edilmeli   |

Figure 8: Body

### 4.1.3 Component Requirements

Figure 9

|   | A Name  | Sağlandı | İlişkili Komponent | Komponent Testleri               |
|---|---|----------|--------------------|----------------------------------|
| 1 | Servo motor 2 dereceden daha hassas olmalı          | ✓        | Servo Motor        | DKT(derece kontrol testi)        |
| 2 | Mikrokontrolcünün CPU su 600 Mhz in üzerinde olmalı | ✓        | Mikrokontrolcü     | UMS(Uygun Mikrokontrolcü Seçimi) |
| 3 | Mikrokontrolcünü 5-12V arasında çalışabilmeli       | ✓        | Mikrokontrolcü     | UMS(Uygun Mikrokontrolcü Seçimi) |
| 4 | Servo Motor 5-12V arasında çalışabilmeli            | ✓        | Servo Motor        | SVT (Servo Voltaj Testi)         |
| 5 | Servo motor gerekli ağırlığı taşıyabilmeli          | ✓        | Servo Motor        | SAT (Servo Ağırlık Testi)        |
| 6 | Birebir aynı LDRler kullanılmalı                    | ✓        | LDR                | LDR'lerin çıkışı test edilmeli   |

Figure 9: Body

### 4.1.4 Components

Table 1

|                            | Number of used | Unit Price (TL/unit) | Cost(TL) |
|----------------------------|----------------|----------------------|----------|
| Servo Motor                | 2              | 53.74                | 107.48   |
| Microcontroller            | 1              | 158.9                | 158.90   |
| Additional microcontroller | 1              | 50.00                | 50.00    |
| Solar Panel                | 2              | 23.18                | 46.36    |
| LDR                        | 4              | 1.26                 | 5.04     |
| Jumper                     | 80             | 0.12                 | 9.60     |
| Somun                      | 10             | 0.01                 | 0.14     |
| USB Voltage Regulator      | 1              | 7.21                 | 7.21     |
| Pil yuvası                 | 1              | 1.60                 | 1.60     |
| Vida                       | 20             | 0.07                 | 1.34     |
| Standoff                   | 16             | 0.19                 | 3.04     |
| Makaron (1 meter)          | 1              | 1.26                 | 1.26     |
| Dış malzeme                | 2              | 15.00                | 30.00    |
|                            |                | TOTAL                | 421.97   |

Table 1: The components used in the project.

## 4.2 Training

Table 2

|   | Roles              | Responsible Person          |
|---|--------------------|-----------------------------|
| 1 | Product Owner      | Halil Temurtaş              |
| 2 | Scrum Master       | Eren Koçulu                 |
| 3 | Hardware Engineer  | Taha İzmir & Halil Temurtaş |
| 4 | Software Engineer  | Arif Göçer & Halil Temurtaş |
| 5 | Structure Engineer | Taha İzmir & Arif Göçer     |
| 6 | Test Engineer      | Arif Göçer & Halil Temurtaş |

Table 2: Roles

### 4.2.1 Training on Python

In order to use Raspberry Pi efficiently, I studied Python for a while from a couple of web sites. I mainly focused on Python 3 since it's more up to date than previous version. I tried different codes on Pycharm for Windows

before meeting with Linux terminal and Raspberry. Pycharm is one of the most recommended Python IDE's by communities. Here are some of my very first attempts to use Python.

#### 4.2.1.1 Basics

```
1 # Using Python for the first time!!
2 print("Hello Internship!!!")
3
4 x = 1
5 if x == 1:
6     # indented four spaces, indents works as brackets in C!
7     print("x is 1.")
8 if x==3:
9     print(23)
10
11 myint = 7
12 print(myint) # use '#' for commenting
13
14 # A sample script that uses lists:
15
16 numbers=[] # creates a list called numbers.
17 numbers.append(1) # adds '1' to numbers as first element.
18 numbers.append(2)
19 numbers.append(3)
20
21 strings=[] # creates a list called strings.
22 strings.append("hello")
23 strings.append("world")
24
25 names = ["Ali", "Ahmet", "Ayse"] # adds Ali, Ahmet and Ayse
                                # to names.
26
27 second_name=names[1]
28
29 print(numbers) # prints [1, 2, 3]
30 print(strings) # prints ['hello', 'world']
31 print("The 2nd name on the name list is %s" %second_name)
                                # prints the second name
                                # on the names list is Ahmet!
```

```

1 astring = "Hello world!"
2
3 print(astring.index("o")) # prints 4, since o appears firstly
                           # at 4th digit.
4 print(astring.count("l")) # prints 3, since l appears three
                           # times
5 print(astring[3:7])      # prints low, starting from 3rd
                           # element to 7th element (7th is
                           # not included!)
6 print(astring[3:7:2])   # prints l, starting from 3rd element
                           # to 7th element skipping one
                           # character.
7 print(astring[::-1])    # prints the string reverse.
8 print(astring.upper())  # prints the string with upper cases.
9 print(astring.lower())  # prints the string with lower cases.
10 print(astring.startswith("Hello")) # Returns True
11 print(astring.endswith("asdfasdf")) # Returns False

```

#### 4.2.1.2 Using Conditions

```

1 if < statement is="" true="" > :
2     < do something="" >
3     ....
4     ....
5 elif < another statement="" is="" true="" > :
6     < do something="" >
7     ....
8     ....
9 else:
10    < do something="" >
11    ....
12    ....

```

#### 4.2.1.3 Using Loops

```

1 temurtas = [5, 8, 3, 6]
2 for halil in temurtas:
3     print(halil) # prints every element in temurtas one by
                  # one in every loop.
4 print(temurtas) # prints [5, 8, 3, 6]

```

```

1 count=0
2 while (count<5) :
3     print(count)
4     count +=1
5 else:
6     print("count value reached %d" %(count))

```

#### 4.2.1.4 Defining Functions

```

1 def sum_two_numbers(a, b): # Defining function
2     return a + b
3 x = sum_two_numbers(1,2) # after this line x will hold the
4                             value 3!
5 print("x=%s" %x) #prints x=3

```

#### 4.2.1.5 Defining Classes

```

1 class Vehicle: # define the Vehicle class
2     name = ""
3     kind = "car"
4     color = ""
5     value = 100.00
6
7 def description(self):
8     desc_str = "%s is a %s %s worth $%.2f." %(self.name,
9                                             self.color, self.kind,
10                                            self.value)
11
12     return desc\_str
13
14 car1 = Vehicle()
15 car1.name = "Ferrari"
16 car1.color = "red"
17 car1.kind = "sport"
18 car1.value = 600000.00
19
20 car2 = Vehicle()
21 car2.name = "Jeep"
22 car2.color = "blue"
23 car2.kind = "SUV"
24 car2.value = 10000.00
25
26 print(car1.description()) # prints Ferrari is a red sport
27                             worth $600000.00.

```

```
24 print(car2.description()) # prints Jeep is a blue SUV worth
                               $10000.00.
```

As I went into detail, Python is not very difficult language to learn. In fact, aside from some indent mistake using it is very simple and clean yet powerfull in various applications.

## 4.2.2 Training on Raspberry Pi

```
1 x==4 # first line of code on raspberry pi
2 if x==4
3     print("evet")
```

### 4.2.2.1 Training on LEDs

```
1 import RPi.GPIO as GPIO
2 import time
3
4 GPIO.setmode(GPIO.BCM)
5 GPIO.setwarnings(False)
6 GPIO.setup(17,GPIO.OUT)
7 GPIO.setup(4,GPIO.OUT)
8
9 while True:
10     print "LED on"
11     GPIO.output(17,GPIO.HIGH)
12     GPIO.output(4,GPIO.LOW)
13     time.sleep(1)
14     print "LED off"
15     GPIO.output(17,GPIO.LOW)
16     GPIO.output(4,GPIO.HIGH)
17     time.sleep(1)
```

### 4.2.2.2 Training on LDRs

```
1 from gpiozero import LightSensor, Buzzer
2
3 ldr = LightSensor(4)
4 ldr2 = LightSensor(17)
5 ldr3 = LightSensor(27)
6 ldr4 = LightSensor(22)
7
```

```

8 bir=ldr.value+ldr2.value
9 iki=ldr3.value+ldr4.value
10 uc=ldr.value+ldr3.value
11 dort=ldr2.value+ldr4.value
12
13 while True:
14     print("ldr= %s" %ldr.value)
15     print("ldr2= %s" %ldr2.value)
16     print("ldr3= %s" %ldr3.value)
17     print("ldr4= %s" %ldr4.value)
18     print("bir= %s" %bir)
19     print("iki= %s" %iki)
20     print("uc= %s" %uc)
21     print("dort= %s" %dort)

```

#### 4.2.2.3 Training on Servo Motors with Raspberry Pi

```

1 # Servo Control
2 import time
3 import wiringpi
4
5 wiringpi.wiringPiSetupGpio() # use 'GPIO naming
6 wiringpi.pinMode(18, wiringpi.GPIO.PWM_OUTPUT) # set pin 18
7                                     to be a PWM output
8 wiringpi.pwmSetMode(wiringpi.GPIO.PWM_MODE_MS) # set the PWM
9                                     mode to milliseconds stype
10
11 wiringpi.pwmSetClock(192) # divide down clock
12 wiringpi.pwmSetRange(2000)
13
14 delay_period = 0.01
15
16 while True:
17     for pulse in range(50, 250, 1):
18         wiringpi.pwmWrite(18, pulse)
19         time.sleep(delay\_period)
20     for pulse in range(250, 50, -1):
21         wiringpi.pwmWrite(18, pulse)
22         time.sleep(delay\_period)

```

#### 4.2.3 Training on Arduino

##### 4.2.3.1 Training on Servo Motors with Arduino



```

1 #include <Servo.h>
2
3 Servo Servo1; // create servo named Servo1 to control a servo
4 int pos = 0; // variable to store the servo position }
5
6 void setup()
7 {
8     Servo1.attach(9); // attaches the servo on pin 9 to the servo
9     object
10 }
11
12 void loop()
13 {
14     for (pos = 0; pos <= 180; pos += 1) // goes from 0 degrees
15     to 180 degrees in steps of 1 degree
16     {
17         Servo1.write(pos); // tell servo to go to position in
18         variable 'pos'
19         delay(15); // waits 15ms for the servo to reach the
20         position
21     }
22     for (pos = 180; pos >= 0; pos -= 1) // goes from 180 degrees
23     to 0 degrees
24     {
25         Servo1.write(pos); // tell servo to go to position in
26         variable 'pos'
27         delay(15); // waits 15ms for the servo to reach the
28         position
29     }
30 }

```

## 4.3 Project Coding

### 4.3.1 Raspberry Pi Part

```

1 from gpiozero import LightSensor, Buzzer
2
3 import RPi.GPIO as GPIO
4 import time
5
6 GPIO.setmode(GPIO.BCM)
7 GPIO.setwarnings(False)
8 GPIO.setup(23, GPIO.OUT)
9 GPIO.setup(24, GPIO.OUT)
10 GPIO.setup(25, GPIO.OUT)

```

```

11 GPIO.setup(8,GPIO.OUT)
12
13 ldr = LightSensor(4)# Assign the data coming from LDR1 to ldr
14 ldr2 = LightSensor(17) # Assigns the data similarly
15 ldr3 = LightSensor(27)
16 ldr4 = LightSensor(22)
17
18 while True:
19     bir=ldr.value+ldr2.value # Total Readings of Top
20     iki=ldr3.value+ldr4.value # Total Readings of Bottom
21     uc=ldr.value+ldr3.value # Total Readings of Left
22     dort=ldr2.value+ldr4.value # Total Readings of Right
23
24     fark1=bir-iki; #
25     fark2=iki-bir;
26     fark3=uc-dort;
27     fark4=dort-uc;
28
29     print("bir= %s" %bir)
30     print("iki= %s" %iki)
31     print("uc= %s" %uc)
32     print("dort= %s" %dort)
33
34     print("fark1= %s" %fark1)
35     print("fark3= %s" %fark3)
36
37     if bir>iki and fark1>0.01:
38         GPIO.output(23,GPIO.HIGH)
39         GPIO.output(25,GPIO.LOW)
40         time.sleep(1)
41     elif iki>bir and fark2>0.01:
42         GPIO.output(25,GPIO.HIGH)
43         GPIO.output(23,GPIO.LOW)
44         time.sleep(1)
45     else :
46         GPIO.output(25,GPIO.LOW)
47         GPIO.output(23,GPIO.LOW)
48         time.sleep(1)
49
50     if uc>dort and fark3>0.01:
51         GPIO.output(24,GPIO.HIGH)
52         GPIO.output(8,GPIO.LOW)
53         time.sleep(1)
54     elif dort>uc and fark4>0.01:
55         GPIO.output(8,GPIO.HIGH)

```

```

56         GPIO.output(24, GPIO.LOW)
57         time.sleep(1)
58     else :
59         GPIO.output(24, GPIO.LOW)
60         GPIO.output(8, GPIO.LOW)
61         time.sleep(1)

```

### 4.3.2 Arduino Part

```

1  #include <Servo.h>
2
3  Servo servo1;
4  Servo servo2;
5
6  int in_rasp1 =3;
7  int in_rasp2 =4;
8  int in_rasp3 =5;
9  int in_rasp4 =6;
10
11 int read1=0;
12 int read2=0;
13 int read3=0;
14 int read4=0;
15
16 void setup()
17 {
18     servo1.attach(9);
19     servo1.writeMicroseconds(1475);
20     servo2.attach(10);
21     servo2.writeMicroseconds(1475);
22
23     pinMode(in_rasp1 , INPUT);
24     pinMode(in_rasp2 , INPUT);
25     pinMode(in_rasp3 , INPUT);
26     pinMode(in_rasp4 , INPUT);
27 }
28 void loop() {
29     read1 =digitalRead(in_rasp1);
30     read2 =digitalRead(in_rasp2);
31     read3 =digitalRead(in_rasp3);
32     read4 =digitalRead(in_rasp4);
33
34     if (read1 == HIGH)
35     {

```

```

36     servo1.writeMicroseconds(1515);
37     delay(42);
38     servo1.writeMicroseconds(1475);
39     delay(200);
40 }
41 else if (read2 == HIGH)
42 {
43     servo1 .writeMicroseconds(1425);
44     delay(24);
45     servo1.writeMicroseconds(1475);
46     delay(100);
47 }
48 else
49 {
50     delay(24);
51     servo1.writeMicroseconds(1475);
52     delay(24);
53 }
54 if (read3 == HIGH)
55 {
56     servo2.writeMicroseconds(1515);
57     delay(42);
58     servo2.writeMicroseconds(1475);
59     delay(100);
60 }
61 else if (read4 == HIGH)
62 {
63     servo2.writeMicroseconds(1425);
64     delay(24);
65     servo2.writeMicroseconds(1475);
66     delay(100);
67 }
68 else
69 {
70     delay(24);
71     servo2.writeMicroseconds(1475);
72     delay(24);
73 }
74 }

```

## 4.4 Implementation

### 4.4.1 PCB Drawing & 3D Drawings

#### 4.4.1.1 PCB Drawing

#### 4.4.1.2 3D Drawings

### 4.4.2 Construction of the Body

#### 4.4.2.1 Top Layer

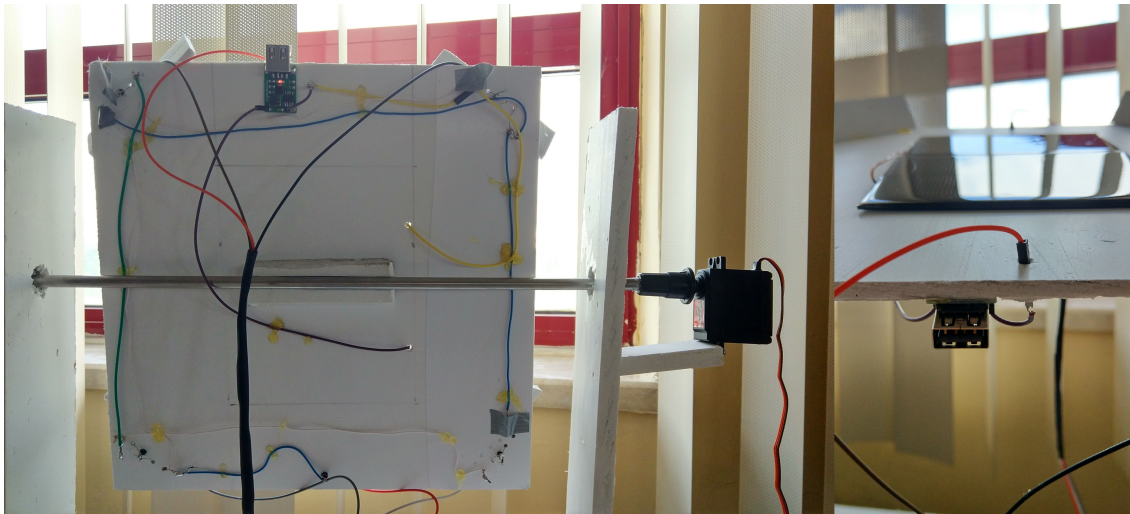


Figure 10: Top Layer

#### 4.4.2.2 Main Body

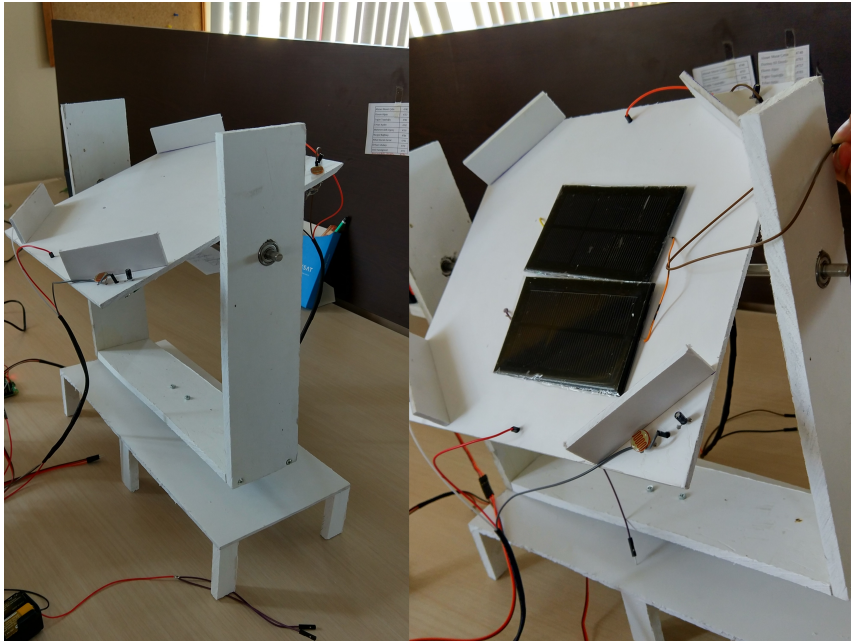


Figure 11: Body

#### 4.4.2.3 Solar Panel

..  
..  
..  
..  
..

#### 4.4.2.4 Final Body

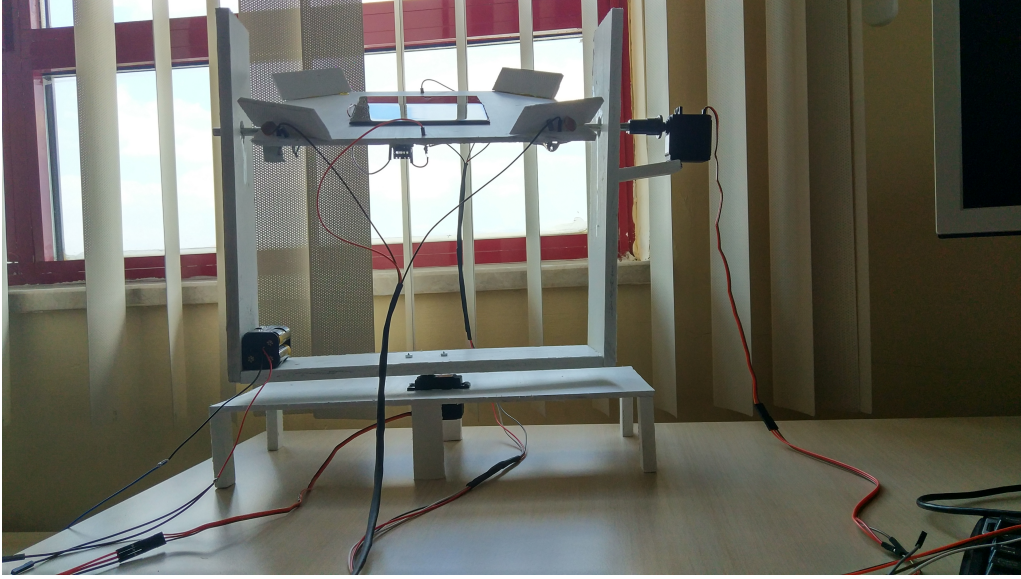


Figure 12: Final Body

### 4.5 Tests

#### 4.5.1 System Requirement Tests

| <input type="checkbox"/> | <input type="lock"/> | A Name                                    | <input checked="" type="checkbox"/> ... | ▼ Test Tipi  | ≡ Test Adımları                    | ≡ Notes                 |
|--------------------------|----------------------|---|---|--------------|------------------------------------|-------------------------|
| 1                        |                      | Güneş Panelinin boyutları belirlenmeli    |   | Sistem Testi | 1.Güneş Panelinin boyutları ...    | İstenilen panelin ...   |
| 2                        |                      | Dış iskeletin dayanıklılığı test edilmeli |   | Sistem Testi | 1. Bir saat boyunca kendi halin... |                         |
| 3                        |                      | FTT (Final Takip Testi)                   |   | Sistem Testi | 1. Sistem güneşli bir alana ...    |                         |
| 4                        |                      | Taşınabilirlik Testi                      |   | Sistem Testi | 1. Sistem çalıştırılır....         | Raspberry şu aşamada... |

Figure 13: Body

## 4.5.2 Subsystem Requirement Tests

| <input type="checkbox"/> |  | Name   | <input checked="" type="checkbox"/> ... | Test Tipi      | Test Adımları                     | Notes |
|--------------------------|--|--|---|----------------|-----------------------------------|-------|
| 1                        |  | Bağlantı elemanlarının sağlamlığının test edilmesi |   | Alt Sistem ... | 1. Lehimler gerekli sağlamlık ... |       |
| 2                        |  | Tasarımda sensör kullanımını denetleme             | <input checked="" type="checkbox"/>     | Alt Sistem ... | 1. Tasarım sensör içeriyor mu ... |       |

Figure 14: Body

## 4.5.3 Component Requirement Tests

| <input type="checkbox"/> |  | Name                             | <input checked="" type="checkbox"/> ... | Test Tipi   | Test Adımları                           | Notes                  |
|--------------------------|--|----------------------------------|---|-------------|---|------------------------|
| 1                        |  | SAT (Servo Ağırlık Testi)        | <input checked="" type="checkbox"/>     | Komponen... | 1. Servo mototrun üzerine bir ağırlı... | Satın alınan servo ... |
| 2                        |  | LDR'lerin çıkışı test edilmeli   | <input checked="" type="checkbox"/>     | Komponen... | 1. Raspberry üzerinden gerekli ko...    |                        |
| 3                        |  | SVT (Servo Voltaj Testi)         | <input checked="" type="checkbox"/>     | Komponen... | 1. Servo gerekli kod yardımıyla ...     |                        |
| 4                        |  | DKT(derece kontrol testi)        | <input checked="" type="checkbox"/>     | Komponen... |   |                        |
| 5                        |  | UMS(Uygun Mikrokontrolcü Seçimi) | <input checked="" type="checkbox"/>     | Komponen... |   |                        |

Figure 15: Body

# 5 After Project

## 5.1 Training on MATLAB

After finishing the project earlier than expected, I was asked to study for educational purposes. Firstly, PCB designing and Solidworks modelling were my priorities since I was not able to do both during the project. Due to limited time, I did not choose either. Since I know the basics, I have chosen Matlab to study on it.

### 5.1.1 Coursera

For that purpose, I have enrolled a course on Coursera. Coursera is.... ...

...

...



### 5.1.2 Outline of the Course

In the first two weeks of the course program, as can be seen from figure XX, Matlab environment and basic operators were introduced. Since I know them already, I have watched the video lectures in a few hours. After that,

| Lesson 1: The MATLAB Environment   | Lesson 2: Matrices and Operators   | Lesson 3: Functions   |
|--|--|---|
| <ul style="list-style-type: none"> <li>Lesson 1: The MATLAB Environment 10 min</li> <li>Introduction 12 min</li> <li>The MATLAB Environment 23 min</li> <li><b>MATLAB Online 27 min</b></li> <li>MATLAB as a Calculator 14 min</li> <li>Syntax and Semantics 5 min</li> <li>Help 8 min</li> <li>Plotting 19 min</li> </ul> | <ul style="list-style-type: none"> <li>Lesson 2: Matrices and Operators 10 min</li> <li>Introduction to Matrices and Operators 11 min</li> <li>The Colon Operator 8 min</li> <li>Accessing Parts of a Matrix 21 min</li> <li>Combining and Transforming Matrices 10 min</li> <li>Arithmetic Part 1 18 min</li> <li>Arithmetic Part 2 11 min</li> <li>Operator Precedence 13 min</li> </ul> | <ul style="list-style-type: none"> <li>Lesson 3: Functions 10 min</li> <li>Introduction to Functions 5 min</li> <li>Function I/O 22 min</li> <li>Formal Definition of Functions 2 min</li> <li>Subfunctions 6 min</li> <li>Scope 5 min</li> <li>Advantages of Functions 2 min</li> <li>Scripts 4 min</li> <li>Problem Solving 52 min</li> </ul> |

Figure 16: The Syllabus of Matlab Course for First 3 Weeks

| Lesson 4: Programmer's Toolbox  | Lesson 5: Selection  | Lesson 6: Loops  |
|---|--|--|
| <ul style="list-style-type: none"> <li>Lesson 4: Programmer's Toolbox 10 min</li> <li>Introduction to Programmer's Toolbox 7 min</li> <li>Matrix Building 15 min</li> <li>Input / Output 20 min</li> <li>Plotting 17 min</li> <li>Debugging 22 min</li> </ul> | <ul style="list-style-type: none"> <li>Lesson 5: Selection 10 min</li> <li>Selection 11 min</li> <li>If-Statements, Continued 8 min</li> <li>Relational and Logical Operators 34 min</li> <li>Nested If-Statements 2 min</li> <li>Variable Number of Function Arguments 6 min</li> <li>Robustness 8 min</li> <li>Persistent Variables 6 min</li> </ul> | <ul style="list-style-type: none"> <li><b>Lesson 6: Loops 10 min</b></li> <li>For-Loops 36 min</li> <li>While-Loops 20 min</li> <li>Break Statements 29 min</li> <li>Logical Indexing 37 min</li> <li>Preallocation 8 min</li> </ul> |

Figure 17: The Syllabus of Matlab Course for 4-5-6 Weeks

| Lesson 7: Data Types   | Lesson 8: File I/O   |
|--|--|
| <ul style="list-style-type: none"> <li>Lesson 7: Data Types 10 min</li> <li>Introduction to Data Types 20 min</li> <li>Strings 29 min</li> <li>Structs 14 min</li> <li>Cells 21 min</li> </ul> | <ul style="list-style-type: none"> <li>Lesson 8: File I/O 10 min</li> <li>File Input/Output 15 min</li> <li>Excel Files 9 min</li> <li>Text Files 12 min</li> <li>Binary Files 38 min</li> </ul> |

Figure 18: The Syllabus of Matlab Course for Last 2 Weeks

### 5.1.2.1 Simple Sorting Code

```

1 function [a b c] = sort3(A)
2 a1 = A(1)
3 a2 = A(2)
4 a3 = A(3)
5
6 if a1 <= a2
7     if a2 <= a3
8         a = a1
9         b = a2
10        c = a3
11    else
12        e = a3
13        a3 = a2
14        a2 = e
15
16    if a1 <= a2
17        a = a1
18        b = a2

```

```

19         c = a3
20     else
21         w = a2
22         a2 = a1
23         a1 = w
24         a = a1
25         b = a2
26         c = a3
27     end
28 end
29 else
30     w = a2
31     a2 = a1
32     a1 = w
33     if a2 >= a3
34         e = a3
35         a3 = a2
36         a2 = e
37         if a1 <= a2
38             a = a1
39             b = a2
40             c = a3
41         else
42             w = a2
43             a2 = a1
44             a1 = w
45             a = a1
46             b = a2
47             c = a3
48         end
49     else
50         a = a1
51         b = a2
52         c = a3
53     end
54 end
55 end
56 }

```

## 5.2 Training on Microsoft Sharepoint

### 5.2.1 Microsoft Sharepoint

SharePoint is a web-based, collaborative platform that integrates with Microsoft Office. Launched in 2001, SharePoint is primarily sold as a document management and storage system, but the product is highly configurable and usage varies substantially between organizations. Microsoft states that SharePoint has 190 million users across 200,000 customer organizations.

## 6 Conclusion

For the project, we were expected to built a solar panel system that can follow the sun light to maximize its efficiency. As planning the project, we used V-model and Agile methodology. As mentioned earlier, using V-model required using another program. We have used Airtable for tracking system requirements, subsystem requirements, tests and so on. The Interface of Airtable & System Requirements can be seen at figure X.

## 7 References

<https://bitbucket.org/temurtas/pi/>  
[https://bitbucket.org/temurtas/staj\\_matlab](https://bitbucket.org/temurtas/staj_matlab)  
[https://bitbucket.org/temurtas/ee300\\_report](https://bitbucket.org/temurtas/ee300_report)  
<https://pomotodo.com/app/>