**T.C.**

**BAHÇEŞEHİR UNIVERSITY**



**FACULTY OF ENGINEERING AND NATURAL SCIENCES**

**CAPSTONE FINAL REPORT**

**CONSTRUCTOR ROBOT WITH A GIVEN MAP OF BUILDINGS**

**1010513**

**Selin Aksun - Mechatronics Engineering**

**Yasemin Karaca - Computer Engineering**

**Beyza Karataş - Mechatronics Engineering**

**Cem Koç - Computer Engineering**

**Assoc. Prof. Ozan Akdoğan - Mechatronics Engineering**

**Assist. Prof. Selin Nacaklı - Computer Engineering**

**ISTANBUL, June 2023**

**STUDENT DECLARATION**

By submitting this report, as partial fulfillment of the requirements of the Capstone course, the students promise on penalty of failure of the course that

* they have given credit to and declared (by citation), any work that is not their own (e.g. parts of the report that is copied/pasted from the Internet, design or construction performed by another person, etc.);
* they have not received unpermitted aid for the project design, construction, report or presentation;
* they have not falsely assigned credit for work to another student in the group, and not take credit for work done by another student in the group.

# ABSTRACT

CONSTRUCTOR ROBOT WITH A GIVEN MAP OF BUILDINGS

Selin Aksun

Yasemin Karaca

Beyza Karataş

Cem Koç

Faculty of Engineering and Natural Sciences

Advisor: Assoc. Prof. Ozan Akdoğan, Assist.Prof.Selin Nacaklı

June 2023

This project aims to develop a system for automating the construction of structures using a map of buildings. The scope of the project includes extracting data about the buildings from an input image, measuring the heights of the structures, visualizing the 3D model, and implementing a constructor robot capable of building the structures based on the map. Image processing techniques were used to analyze the input image and accurately detect the positions and colors of the rectangles in the map. The heights of the buildings will be measured in the next stages of the project, and a 3D model will be generated for visualization. The constructor robot will be designed to independently assemble parts and construct buildings based on the map's specifications. Successes of the project include accurately detecting the positions and colors of the rectangles in the map, while challenges faced included adjusting the threshold in the image processing algorithm to detect all the rectangles properly and accounting for pixel blending when obtaining color data from the image. Overall, the project has met the specified functional and performance requirements and is ready to accomplish its objective of implementing construction using a map of buildings.

**Key Words:** 3D visualization, autonomous movement, image processing, constructor robot.

# TABLE OF CONTENTS

[ABSTRACT iii](#_Toc137409421)

[TABLE OF CONTENTS iv](#_Toc137409422)

[LIST OF TABLES v](#_Toc137409423)

[LIST OF FIGURES v](#_Toc137409424)

[LIST OF ABBREVIATIONS vii](#_Toc137409425)

[1. OVERVIEW 7](#_Toc137409426)

[1.1. Identification of the need 7](#_Toc137409427)

[1.2. Definition of the problem 8](#_Toc137409428)

[1.2.1. Functional requirements 8](#_Toc137409429)

[1.2.2. Performance requirements 8](#_Toc137409430)

[1.2.3. Constraints 9](#_Toc137409431)

[1.3. Conceptual solutions 9](#_Toc137409432)

[1.3.1. Literature Review 10](#_Toc137409433)

[1.3.2. Concepts 11](#_Toc137409434)

[1.4. Physical architecture 15](#_Toc137409435)

[2. WORK PLAN 19](#_Toc137409436)

[2.1. Work Breakdown Structure (WBS) 19](#_Toc137409437)

[2.2. Responsibility Matrix (RM) 20](#_Toc137409438)

[2.3. Project Network (PN) 20](#_Toc137409439)

[2.4. Gantt chart 21](#_Toc137409440)

[2.5. Costs 22](#_Toc137409441)

[2.6. Risk Assessment 22](#_Toc137409442)

[3. SUB-SYSTEMS 23](#_Toc137409443)

[3.1. Computer Engineering 23](#_Toc137409444)

[3.1.1. Requirements 23](#_Toc137409445)

[3.1.2. Technologies and methods 24](#_Toc137409446)

[3.1.3. Conceptualization 24](#_Toc137409447)

[3.1.4. Architecture 25](#_Toc137409448)

[3.1.5. Materialization 26](#_Toc137409449)

[3.1.6. Evaluation 29](#_Toc137409450)

[3.2. Mechatronics Engineering 35](#_Toc137409451)

[3.2.1. Requirements 35](#_Toc137409452)

[3.2.2. Technologies and methods 35](#_Toc137409453)

[3.2.3. Conceptualization 36](#_Toc137409454)

[3.2.4. Physical architecture 36](#_Toc137409455)

[3.2.5. Materialization 37](#_Toc137409456)

[3.2.6. Evaluation 37](#_Toc137409457)

[4. INTEGRATION AND EVALUATION 39](#_Toc137409458)

[4.1 Integration 39](#_Toc137409459)

[4.2 Evaluation 41](#_Toc137409460)

[5. SUMMARY AND CONCLUSION 43](#_Toc137409461)

[ACKNOWLEDGEMENTS 44](#_Toc137409462)

[REFERENCES 45](#_Toc137409463)

[APPENDIX A 47](#_Toc137409464)

[APPENDIX B 48](#_Toc137409465)

[APPENDIX C 49](#_Toc137409466)

[APPENDIX D 50](#_Toc137409467)

[APPENDIX E 51](#_Toc137409468)

# LIST OF TABLES

[Table 1. Comparison of the three conceptual solutions - Mechatronics](#_heading=h.mtevi22bason) 12

[Table 2. Comparison of the two conceptual solutions -](#_heading=h.106wht8w988a) Mechatronics 13

[Table 3. Comparison of the two conceptual solutions - Computer](#_heading=h.3j2qqm3) 14

[Table 4. Comparison of two conceptual solutions for 3D output generation](#_heading=h.4i7ojhp) 15

[Table 5. Responsibility Matrix for the team](#_heading=h.3whwml4) 20

[Table 6. Costs](#_heading=h.qsh70q) 22

[Table 7. Risk Matrix](#_heading=h.3as4poj) 22

Table 8. Risk Assessment23

# LIST OF FIGURES

[Figure 1. Representation of three different concepts](#_heading=h.i7lkvbwy1n1m) 12

[Figure 2.](#_heading=h.dn4uz2xxuow0) Interface Diagram 16

[Figure 3.](#_heading=h.4i7ojhp) Process chart for computer engineering 17

[Figure 4.](#_heading=h.sxvtgxi8plu1) Process chart for the system 18

[Figure 5.](#_heading=h.ili5eriq4byn) Work breakdown structure for the project 19

Figure 6. Project network 21

Figure 7. Gantt Chart 21

Figure 8. Use Case Diagram 25

Figure 9. Flow Chart for Image Processing 26

Figure 10. Workflow of the Image Processing 27

Figure 11. Workflow of the 3D modelling 28

Figure 12. Workflow of the Arduino data retrieval 28

Figure 13. The Map of Buildings and Output Log of input 1 32

Figure 14. 3D Visualization for input 1 32

Figure 15. The Map of Buildings and Output Log of input 2 33

Figure 16. 3D Visualization for input 2 33

Figure 17. The Map of Buildings and Output Log of input 3 34

Figure 18. 3D Visualization for input 3 34

Figure 19. Physical architecture of the robot 37

Figure 20. Shaft part tat can not come out of the printer 38

Figure 21. The Map of Buildings considering the limitations 40

Figure 22. Example of JSON String 42

Figure 23. Output of parsed JSON data 42

# LIST OF ABBREVIATIONS

CR Constructor Robot

3D 3 Dimension

CMP Computer Engineering

CNC Computer Numerical Control

MCH Mechatronics Engineering

SCARA Selective Compliance Articulated Robot Arm

WBS Work Breakdown Structure

PN Project Network

UI User Interface

OCR Optical Character Recognition

VTK Visualization Tool Kit

IDE Integrated Development Environment

EEPROM Electronically Erasable Programmable Read-Only Memory

PIC Peripheral Interface Controller

JSON JavaScript Object Notation

# 1. OVERVIEW

The project's primary goal is to create structures of various heights using the provided building map. The height of each building on the map will be determined using image processing methods. This building map will be created by a constructor robot using objects of the same height. An image of a construction map will show the amount of buildings that are next to one another along with the height information for each structure. The output will be a visual display of this map, which will get formed by this map being examined with image processing methods. Information about the map including the number of structures and their related heights will be forwarded to the construction. A constructor robot that can carry objects and stack them on top of one another using the given map will be built. Three objects should be stacked on top of one another to create a building with three floors, as each object symbolizes one level. All of the buildings are going to be visible to the robot once construction is finished.

Team members and responsibilities:

Cem Koç - Planning, input generation, 3D visualization.

Yasemin Karaca - Documentation, input generation, 3D visualization

Beyza Karataş - Integration, robot logic coding, system design

Selin Aksun - Integration, cad design and manufacturing, integration and test

## 1.1. Identification of the need

This product aims to solve the needs of the construction industry such as the need for manpower, finding and managing labors and decreasing the total cost of construction projects.

 The construction industry composes a large portion of the global economy. By 2022 the market size of this industry is 8.2 trillion dollars (Statista[1]). Labor cost percentages in construction lie between %20 and %40 of the total budget of construction (Bridgit[2]). This project’s goal is to present stakeholders a faster, cheaper and more efficient solution to construction.

## 1.2. Definition of the problem

The problem in this project is that in order for a constructor robot to navigate, carry, and build the objects in a correct way, the proper and necessary processed image input should be provided. A good amount of data from the processed image should be extracted as an input and implemented by the constructor robot as required. The algorithm of image analyzing should provide a deductive visual.

While working on this project, there are few potential challenges or problems we may encounter:

1. 1. **Image processing**: Factors including lighting, resolution, and image quality can have an influence on image processing techniques. As a result, using image processing techniques to accurately calculate the heights of the buildings on the map may be difficult.
2. 2. **3D Visualization**: From the given map of building a 3D representation of the map should be generated. In this part which 3D engine will be used should be determined.
3. 3. **Construction robot:** It may be difficult to design and construct a robot that can accurately stack objects of the same height to build structures of different heights as represented by the map. The robot's movement, stability, and capacity to manipulate objects should all be taken into account.
4. 4. **Map interpretation:** The construction robot correctly interprets the map and constructs the buildings as intended should be ensured. Considering how the map will be conveyed to the robot and how it will be able to accurately follow the instructions it receives will be needed.

### 1.2.1. Functional requirements

From the given map of buildings an input data should be generated for the robot to move. Then using this data a 3D representation of the map should be created.

The construction robot should be able to accurately interpret the building map and determine the number and heights of the structures to be built. In the construction site, the robot should be able to move around and pick up objects of the same height. The robot should be able to stack the objects on top of each other to construct the buildings as specified by the map. The constructed buildings should match the specifications on the map and it should be visually confirmed by the construction robot. In case of any errors that can arise, the system should be able to provide feedback to the user.

### 1.2.2. Performance requirements

In the data processing and providing side the time constraint is neglectable because there is no need for real-time data generation while the robot is in the construction process. To create the desired building map, the robot must move objects and place them correctly.

### 1.2.3. Constraints

**Scheduling and Team size:** We have 14 weeks to develop and get this project ready with a team that is composed of 4 people and our budget is around 2000 liras. Considering these resources we are able to develop only a prototype version that can not be implemented in real life constructions.

**Economic impacts:** The labor cost can be reduced by automating the construction process. If the robot is able to construct buildings efficiently and accurately, it can reduce the need for human labor, resulting in cost savings. Developing and implementing the construction robot, along with the image processing technology, could result in a high initial cost. If there is a high market demand for the automated construction systems, there can be significant revenue. However if market demand is low, initial development costs can be more difficult to recover.

**Environmental impacts:** By minimizing the need for human labor and transportation, the carbon footprint of the construction industry can be reduced. However, there may be additional environmental impacts related to the materials and energy needed in construction robots. The environment might be damaged if the components of these systems are not energy-efficient or sustainable. Moreover, waste may be produced by the system while it is being built.

**Social impacts:** In this project, employment in the construction industry can be affected. The impact in employment can lead to loss of jobs or displacement for some workers in the industry. The construction robot may also increase safety in the field by removing the need for human workers to carry out potentially dangerous or physically challenging operations. By enabling anyone with disabilities or mobility challenges to contribute to the construction process in methods that may not have been previously accessible, the robot may increase accessibility in the construction sector.

**Costs:** Since we only have 2000 liras to work with, we are occasionally unable to select the highest-quality materials.

## 1.3. Conceptual solutions

An algorithm can be developed to read pixels from the given map of buildings, then this data can be interpreted and used to feed game engines such as Unity[3] or Unreal Engine[4]. With the help of other 3D softwares like Blender[5] or Maya[6] a design may be created for building an environment to visualize the map. Another solution might be to directly use 3D Modeling softwares without using game engines.

### 1.3.1. Literature Review

Image processing is a technique for applying various procedures to an image in order to improve it or extract some relevant information from it. It is a form of signal processing where a picture serves as the input, and the output could be another image or characteristics or features related to that image[7]. “Image processing is a crucial step in the creation of a 3D model of a building map. Several techniques have been proposed in the literature to extract relevant information from images, such as object detection, image segmentation and pattern recognition (Szeliski,2010).[8]

The main steps of image processing are:

1. importing the image using image acquisition software;
2. analyzing and adjusting the image;
3. producing a report or improved image as a result of the analysis.

The main steps of 3D visualization are: [[9]](#_heading=h.9dcwik44x8m8)

1. acquiring 3D data
2. preprocessing and cleaning the data
3. creating a 3D model or scene
4. adding the data to the 3D model or scene
5. visualizing the data
6. interacting with the data

In our case, a building map that indicates the number of buildings with their heights will be given to CMP students, who will follow the steps above and give the improved, analyzed map as an input to the MCH students, who will generate a constructor robot that will read the map and carry the objects, putting them on top of each other according to this map. In the end, from the robot’s sight, all buildings will be visible. Image processing algorithm will extract data from the input map in a fast way, making it cost-efficient and time-efficient for the user. By using open-source software there won’t be a need for licensing fees.

A growing number of tasks can be completed by robots with great efficiency and accuracy, which is why they are becoming more and more important in the construction sector. Construction robots come in a variety of designs, each with particular benefits and drawbacks. One of the most popular industrial robots for automation is the Scara robot. They are employed in numerous tasks like assembly, welding, and material handling.

The main steps of SCARA robot are:

1. Create the robot: Sketch the design of the robot and choose the sizes of each component.
2. Assemble the robot: Put the parts together to create the robot.
3. Connect the components: Attach the power supply and the microcontroller (Arduino) to the components.
4. Use the C/C++ programming language to program the Arduino to control the robot.
5. Evaluate the robot: Check its accuracy and motions.
6. Improve the robot's performance by making any necessary modifications.

### 1.3.2. Concepts

**Concept 1 for System Design - Mechatronics**

Cartesian coordinate configuration robots have the ability to take the products over a band or from any point by taking the elbow up-down, in-out and forward-back with its three joints. The disadvantage is that it can not be used in narrow spaces because it takes up a lot of space.

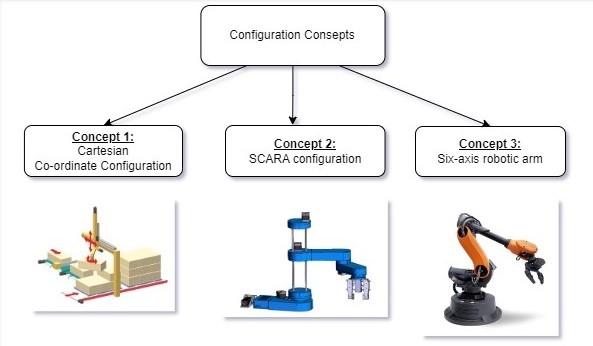
**Concept 2 for System Design- Mechatronics**

SCARA type robots have very high speeds, best repetition ability, high sensitivity and accuracy rates. It is similar to cartesian robots in that they move in three axes, but they can also rotate. While they are fast and plastic, they are more durable than cartesian robots.

**Concept 3 for System Design - Mechatronics**

Six-axis robots are mounted on a pedestal and offer the most directional movement and control of all three robots. The six axis allows the robot arm to move in the X, Y, and Z planes, as well as position itself using roll, pitch, and yaw movements. Very flexible and can mimic the motion of a human arm. Can be more compliant and a little slower than other configurations due to the nature of the design.

Table 1 compares different conceptual solutions with respect to the four most important requirements; Concept 2 is chosen for this project due to durability and availability.



**Figure 1. Representation of three different concepts**

**Table 1. Comparison of the three conceptual solutions - Mechatronics.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Concept 1 | Concept 2 | Concept 3 |
| Cost | medium | low | high |
| Complexity | medium | medium | high |
| Performance | high | medium | medium |
| Features | low | medium | high |

All of these concepts have their own advantages and disadvantages. Generally, cartesians have a rectangular work envelope, while a scara's is circular. There are other differences between them as well as several advantages and disadvantages of each type. The main disadvantage of cartesians is that they require a large volume of space to operate in. Cartesian have the largest surface area requirement of all robotic configurations. Additionally, within the limits of their working volume and quick movement capabilities, Scara robots can be quickly reached at any location on the workspace. [Six-axis robots](https://robotsdoneright.com/Articles/what-is-a-six-axis-robot.html) are the most common option for industrial applications. Cost is always a major consideration with any automation project. Six-axis robots are one of the more expensive robotic options available. This is mainly due to the number of axes. Each axis requires a motor and gearbox. Therefore, We decided to move forward with concept 2 as we thought it was more suitable for our task in terms of performance and usability.

**Concept 1 for Microcontroller - Mechatronics**

Microchip Technology's PIC microcontrollers are 8-bit microcontrollers. They are commonly used in embedded systems and are written in the C programming language. They contain a variety of characteristics and are quite versatile, making them a popular choice for a variety of applications.

**Concept 2 for Microcontroller - Mechatronics**

Arduino microcontrollers are open-source microcontrollers that run the Arduino programming language and are based on the Atmel AVR microcontroller. They are generally used for applications that need physical interaction, such as robotics, home automation, and electronic art.

**Table 2. Comparison of the two conceptual solutions - Mechatronics.**

|  |  |  |
| --- | --- | --- |
|  | Concept 1 | Concept 2 |
| Cost | low | low |
| Complexity | medium | high |
| Performance | medium | high |
| Features | medium | medium |

The main difference between PIC and Arduino is their ease of usage and flexibility. While PICs require extensive technical knowledge to program and communicate with external components, Arduino is far more user-friendly. It is built on a basic programming language and includes a variety of libraries and software tools that make generating interactive projects easy.

Arduino is a popular choice for Scara robot that we choose applications due to its ease of use and wide range of functionality.

**Concept 1 for Image Processing - Computer**

For image processing, there are many tools, both open source and commercial. In our case, these two tools are the ones that can meet our requirements at the maximum level: OpenCV[10] and Scikit-Image[11]. These two tools have many ups and downs compared to each other but in the end they are efficient image processing tools that are used in many image processing applications. These tools are libraries, each one is available for multiple programming languages and they provide robust image processing algorithms to make it easy and fast to extract information out of an image. We can choose to use either one of them to extract the needed information from the given map of buildings. It is also important to mention that both of these two tools are open source.

**Concept 2 for Image Processing - Computer**

The second option for image processing is to code our own algorithm to extract the data from an image. The programming language we choose should not matter that much since we won’t be using any image processing libraries. C++ and Java are good options. A comparison of the two conceptual solutions for the computer part can be seen in Table 3.

**Table 3. Comparison of the two conceptual solutions - Computer.**

|  |  |  |
| --- | --- | --- |
|  | Concept 1 | Concept 2 |
| Cost | low | low |
| Complexity | medium | high |
| Performance | high | high |
| Flexibility | medium | high |

Both concepts have their own strengths and weaknesses. In concept 1, using established tools that have been widely used and have a range of powerful features that makes them suitable for image processing and 3D visualization. Because of this, the chance of running into technical problems or undesired behaviors throughout the project's development may be minimized. Support and documentation is also another strength for concept 1, since the tools we will use have a widely used area. Using image processing libraries such as OpenCV or Scikit-Image can result in good performance and efficiency features.

In concept 2, by writing our own image processing algorithm, more flexibility to customize the solution to the specific needs of the project is allowed. but it may be more time consuming and require more programming knowledge. In terms of performance, optimizing the solution for performance and efficiency is high due to writing our own image processing algorithm.

Since there are different types of image files and some of them are compressed and encoded. It is a complex process to decode them and extract the pixel information. These libraries mentioned in Concept 1 are the tools to help us in these situations thus decreasing the complexity. Therefore we will choose Concept 1 for image processing.

**Concept 1 for 3D Output Generation - Computer**

For 3D output generation, MatPlotlib can be used. MatPlotlib is a 2D plotting library that is used in Python. 3D plots will be created with the ‘mplot3d’ module. Matplotlib is open-source and free to use. It is primarily used for 2D plotting libraries so it might not be the best option for advanced 3D visualization. A three-dimensional axis can be created. To add more functions such as the analysis of the data, other libraries called NumPy and Pandas can provide it. It is largely customizable so its flexibility is high. The details of the appearance can be managed and with other libraries help, it can be more advanced. Another library we can use is called Visualization Tool Kit(VTK). For 3D visualization and image processing, VTK can be used. VTK is a library of C++ classes. The visualization is flexible, and it has robust capability for 3D modeling. VTK provides classes for creating, rendering and displaying 3D structures. Positions, points, lines and orientation can be easily controlled with the classes VTK provides.

**Concept 2 for 3D Output Generation - Computer**

To have more advanced graphics, game engines can be used for 3D output generation. Most popular game engines in the market are Unity and Unreal Engine. To feed the engine with 3D models to create the environment some 3D softwares that can be used are; Blender, Maya or we can use premade assets that are available on the market.

**Table 4. Comparison of the two conceptual solutions for 3D output generation**

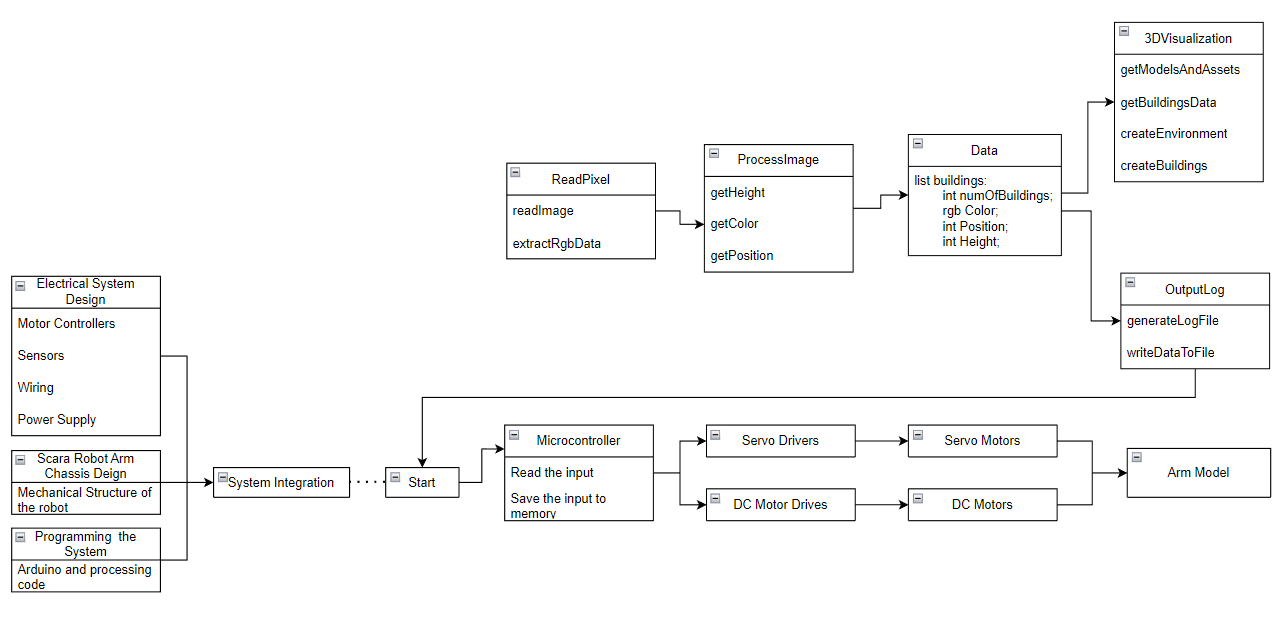
|  |  |  |
| --- | --- | --- |
|  | Concept 1 | Concept 2 |
| Cost | low | low |
| Complexity | low | medium |
| Performance | high | low |
| Flexibility | medium | high |

In concept 1, the created visualizations will be more technical but not advanced. The data will be read and analyzed easily. On the other hand, in concept 2, 3D environments are going to have more advanced graphics because rendering methods and functions are way more robust. Creating interactive and dynamic environments is possible. Therefore the environments are more customizable. We will go with concept 2.

## 1.4. Physical architecture

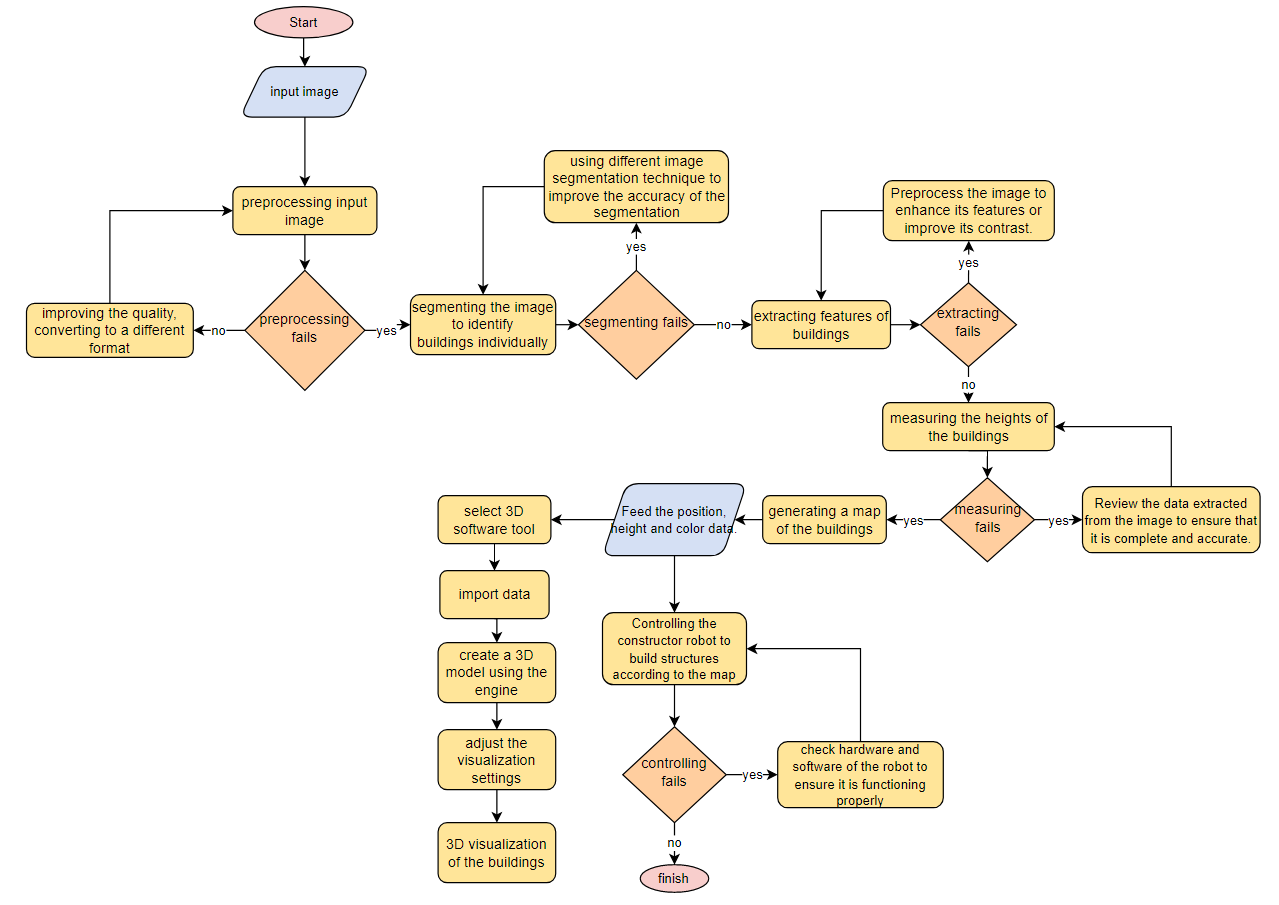
The project consists of two sub-systems: the builder robot and the image processing. Breaking down these two concepts, in the image processing part we have; information extraction from a given image by first reading pixels of the image and then interpreting them to scrape the information. Then this information will be transferred to two processes. First process is generation of the output log file to feed the robot with the data. Second process is the generation of 3D visualization to represent the data in a 3D environment. As for the constructor robot part a design will be realized taking into account the size, shape, materials and purpose of the robot. We will use Arduino which is a microcontroller board that provides the main processing power for the robot. The servo motor, which moves the robotic arm, will be managed by it. The robot's software consists of writing code for the Arduino to control the servo motor and creating a user interface to direct its movements. These two subsystems will be integrated by using the output log file and microprocessor. A data transfer protocol will migrate the data from the file. First Arduino will be connected via usb port to a PC then using Arduino IDE the data extracted from the file will be written to EEPROM which is a section of Arduino’s memory that is not volatile if Arduino loses power. When the migration of data is done these two subsystems are integrated.

The Interface diagram that illustrates the relationship between subsystems can be seen in figure 2.



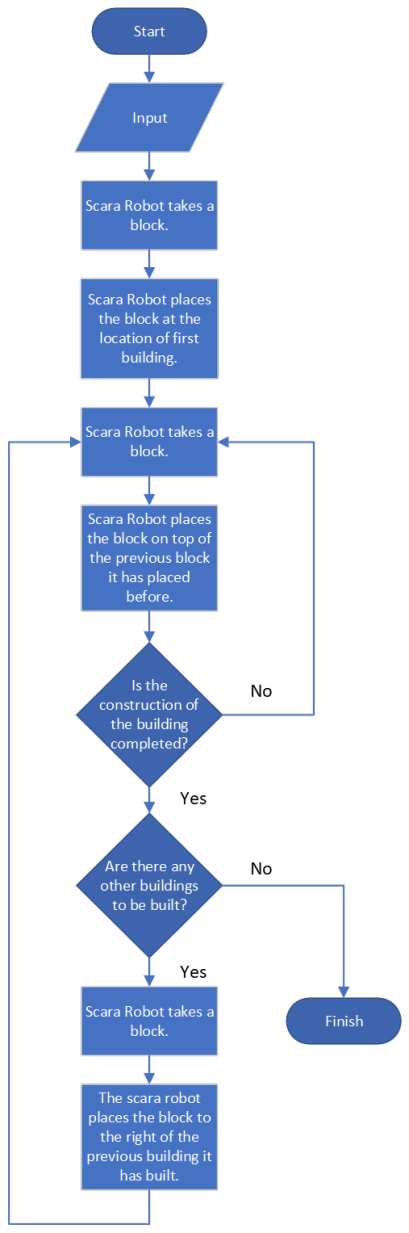
**Figure 2. Interface Diagram**

The process chart of the computer part of the project can be seen in figure 3.



**Figure 3. Process chart for computer engineering**

The process chart for mechatronics can be seen in Figure 4.

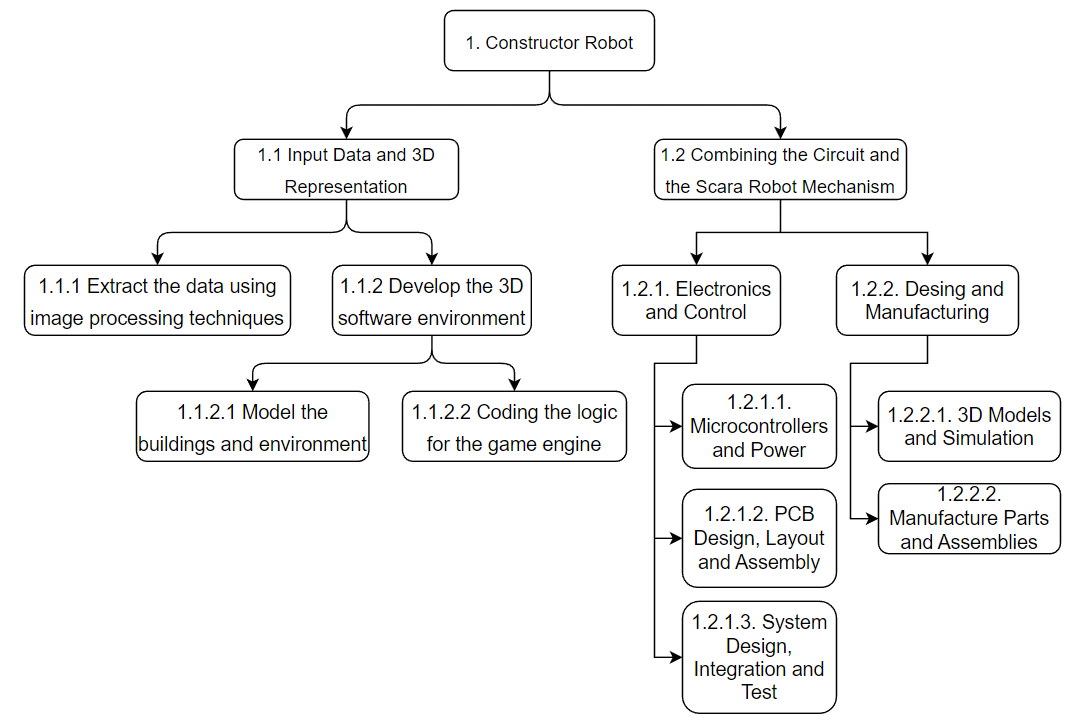


**Figure 4. Process chart for the system**

# 2. WORK PLAN

## 2.1. Work Breakdown Structure (WBS)

1. Work Breakdown Structure(WBS) offers a thorough hierarchical representation of the project’s structure. WBS can be seen in Figure 5.



**Figure 5. Work breakdown structure for the project**

## 2.2. Responsibility Matrix (RM)

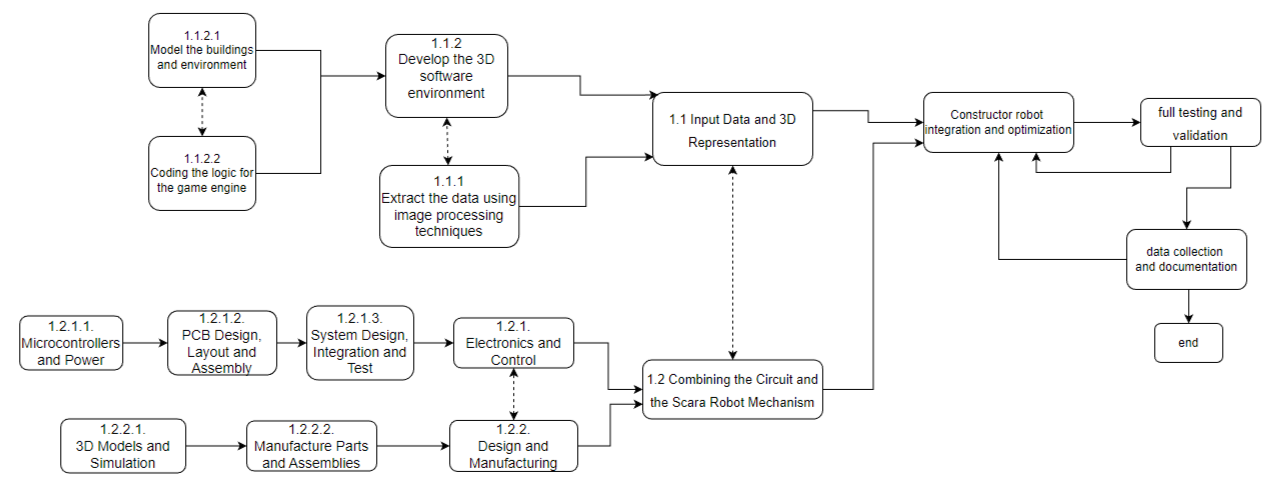
Responsibility matrix given in Table 5 represents the tasks assigned to the students in the project.

**Table 5. Responsibility Matrix for the team**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Yasemin** | **Cem** | **Selin** | **Beyza** |
| **Documentation** | **R** | S | S | S |
| **Integration** | S | S | **R** | **R** |
| **Planning** | S | **R** | S | S |
| **Data Processing** | **R** | **R** |  |  |
| **Cad Design and Manufacturing** |  |  | **R** | **R** |
| **Integration and Test** |  |  | **R** | **R** |
| **Data Visualization** | **R** | **R** |  |  |
| **Robot Logic Coding** |  |  | **R** | **R** |
| **System Design** |  |  | **R** | **R** |

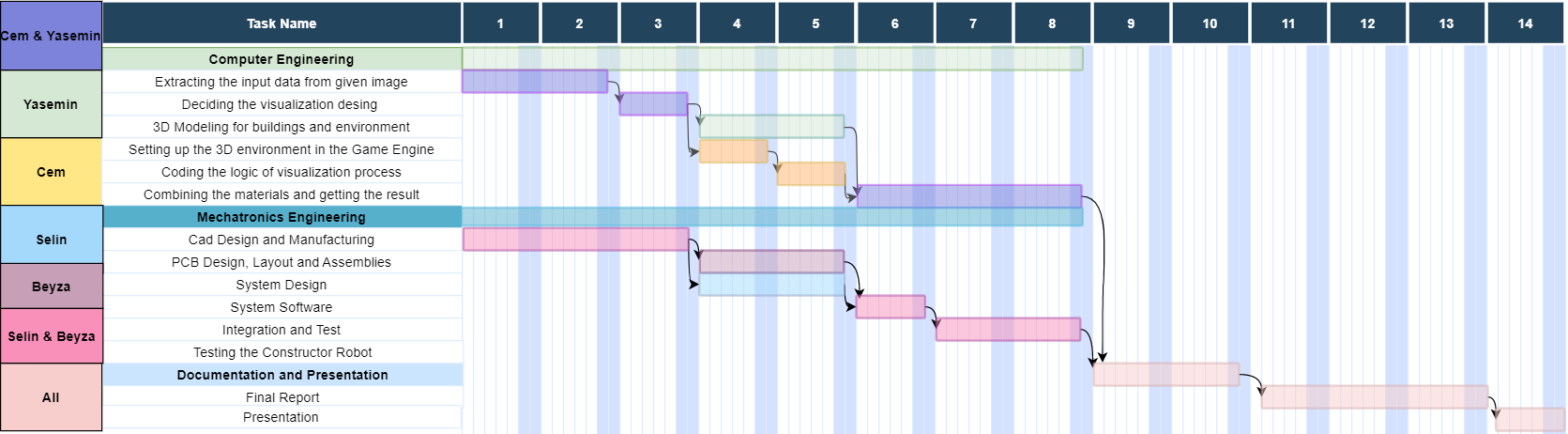
## 2.3. Project Network (PN)

A project network is a graphical representation of the dependencies between the tasks of a project. The project network that illustrates the connection between tasks can be seen in Figure 6.



**Figure 6. The Project Network.**

## 2.4. Gantt chart

1. For project management, a gantt chart is provided to represent the schedule of the project. The main procedure for getting the finished constructor robot is shown and a detailed schedule of how the crew's weekly responsibilities are partitioned is shown in Figure 7.
2. 

**Figure 7. Gantt Chart**

## 2.5. Costs

Costs are given in Table 6 below.

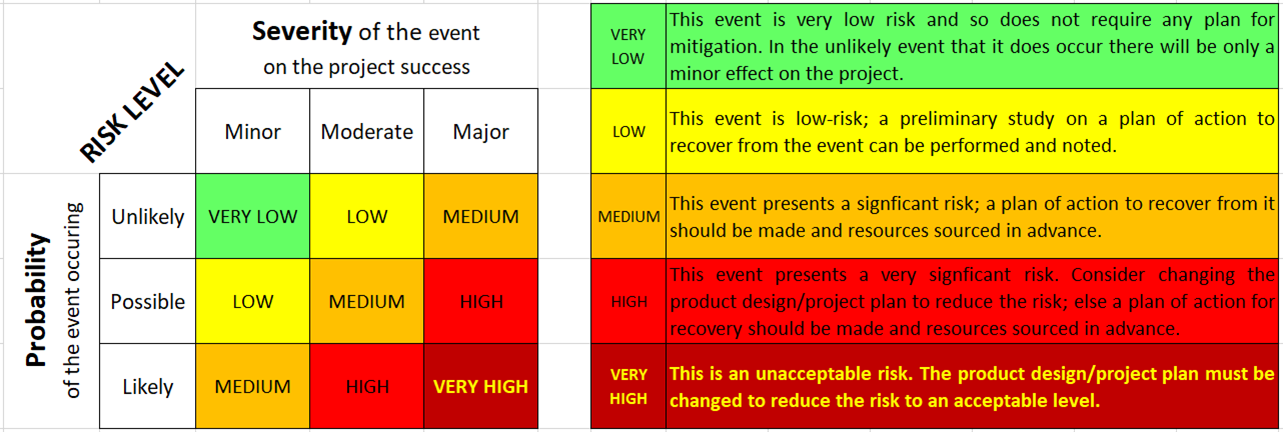
**Table 6. Costs**

|  |  |
| --- | --- |
| **Electro-mechanical** | |
| 2 x NEMA 17 DC Step Motor | 477 |
| MG996R Servo Motor | 159,04 |
| 3D Printed Parts | 600 |
| Mills | 150 |
| Assembling Parts (Screw, Bearings, Bolts and Nuts etc.) | 300 |
|  | |
| **Communications and Control** | |
| Arduino UNO R3 | 108 |
| DC Power Supply 12V 6A | 288,23 |
| A4988 Motor Driver | 32 |
| Micro Servo | 49 |
|  | |
| **System Total** | **2263,27** |

## 2.6. Risk Assessment

The problems that can occur whilst the preparation and the production of the project are shown in the risk table. Table 7 represents risk matrix.

**Table 7. Risk Matrix**

******

The purpose of a risk assessment is to recognize any potential hazards that may hinder the success of a project and to implement measures to either reduce or control those risks. Risk assessment of the project is shown in Table 8.

**Table 8. Risk Assessment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Failure event** | **Probability** | **Severity** | **Risk level** | **Plan of action** |
| Microprocessor  failure | **Unlikely**  This component is known to be reliable. | Major  Would require replacing. | **MEDIUM** | Have a spare microprocessor at hand. |
| Corrupted or Wrong Input Data | **Possible**  Slightest error in the input generating process would cause the machine to work undesirably. | Minor  Revaluation of the code is needed | **LOW** | We need to make sure that our algorithm is working correctly an it is ready to consider all possible scenarios |
| Maintenance | **Possible**  Robot or equipment may require repair while working, which can impact project timeline or budget. | Minor | **LOW** | Performing regular checks, cleanings, and repairs to make sure everything is performing at its best. Having back-up equipments. |
| Not Enough Power Output | **Unlikely**  We may specify the necessary parts and their power outputs with careful preparation in advance, enabling the robot to function as intended. | Major | **MEDIUM** | Disassembling the parts, acquiring new equipment that meets our demands, and then correctly reassembling them. |

# 3. SUB-SYSTEMS

## 3.1. Computer Engineering

Computer subsystems are coding an image processing method according to the given map of buildings and representing it in 3D visualization.

### 3.1.1. Requirements

* Before providing the constructor robot with input, the map should be analyzed and the buildings should be visually displayed.
* By image processing techniques, the height of all buildings should be determined based on the number that is given on top of each building's representative rectangles.
* The building map image should be processed and analyzed quickly and correctly.
* Based on the data obtained from the image, creating a 3D model should be provided.
* The data from the output log file should be transferred to the construction robot

### 3.1.2. Technologies and methods

For 3D visualization apps like Unreal Engine, Unity, Blender, Maya can be used.

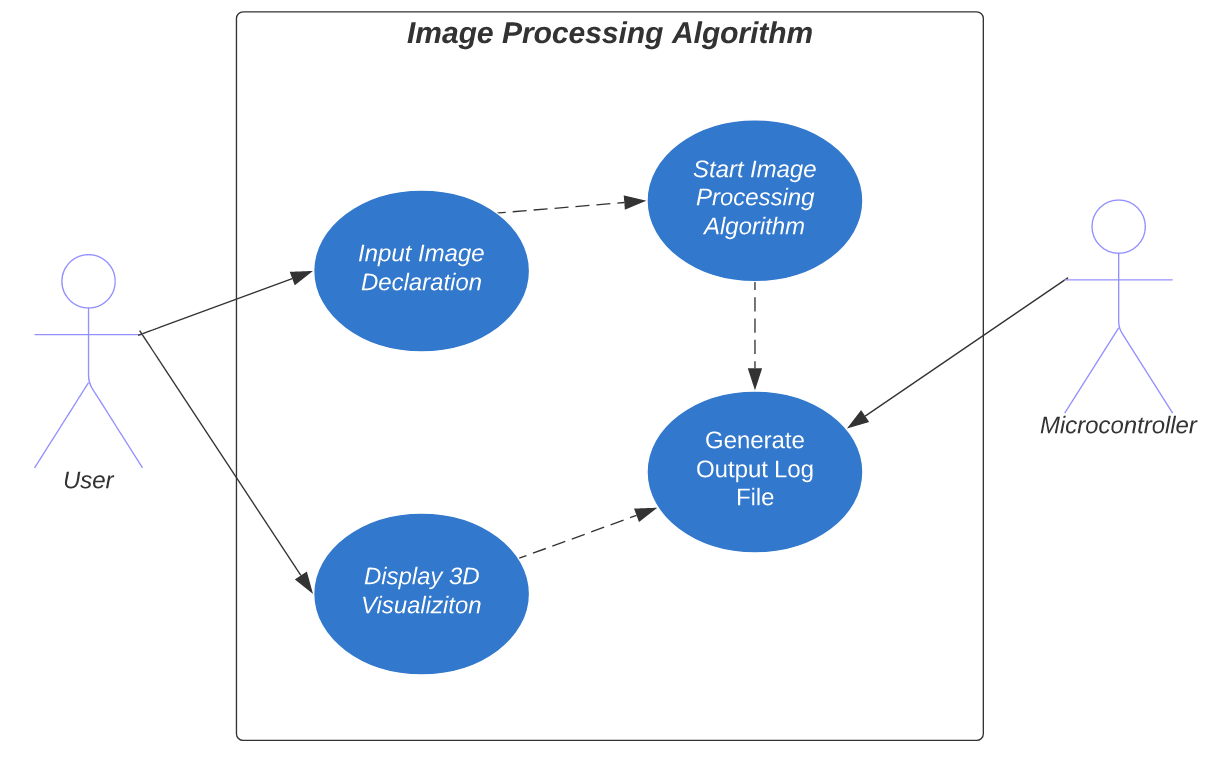
For generating the input data depending on the game engine that we are going to use C# or C++ can be used. Another option may be extracting the data, without the game engine with a different programming language such as python and feeding the data to the microcontroller and the game engine later.

### 3.1.3. Conceptualization

Our aim is to make it easy for constructor robots to read the map of buildings by creating a 3D visualization. As mentioned in section 1.4, the image processing subsystem has components such as data extraction from a given map by reading pixels and interpreting them to scrape the information, generation of the output log file to feed the constructor robot with data and generation of 3D visualization to represent the data in a 3D environment.

As shown in figure 2 the interface diagram for the computer engineering part is provided. ReadImage is responsible for reading the pixels of the building map, allowing the system to analyze the image and extract relevant data. ExtractRGBdata function is responsible for RGB color information extraction from the input map by using the information from ReadImage function. ProcessImage takes the information from the ReadPixel component and processes it to make it usable by the constructor robot having functions for getting the height, color and the position of the buildings. The data component stores the information extracted from the processImage component, it stores the information of each building in the map such as int numOfBuildings, int Height, int Position and rgb Color. The 3D visualization component is responsible for creating a 3D model of the final construction map. It includes functions such as getting the 3D model and assets of the buildings, getting the buildings data with getBuildingsData function, creating environment and creating buildings. Output log component generates log files that contain data about the system’s performance, such as the time taken to process the image, the number of buildings detected, and any errors that occurred during the process. These components work together to give it as an input to the constructor robot who will interpret the map of buildings accurately and efficiently. Since OCR is a library to extract text from the map of buildings, to analyze the heights of the buildings, the getHeight function will use the OCR library. Lastly, the log file will be an input for the constructor robot’s microcontroller.

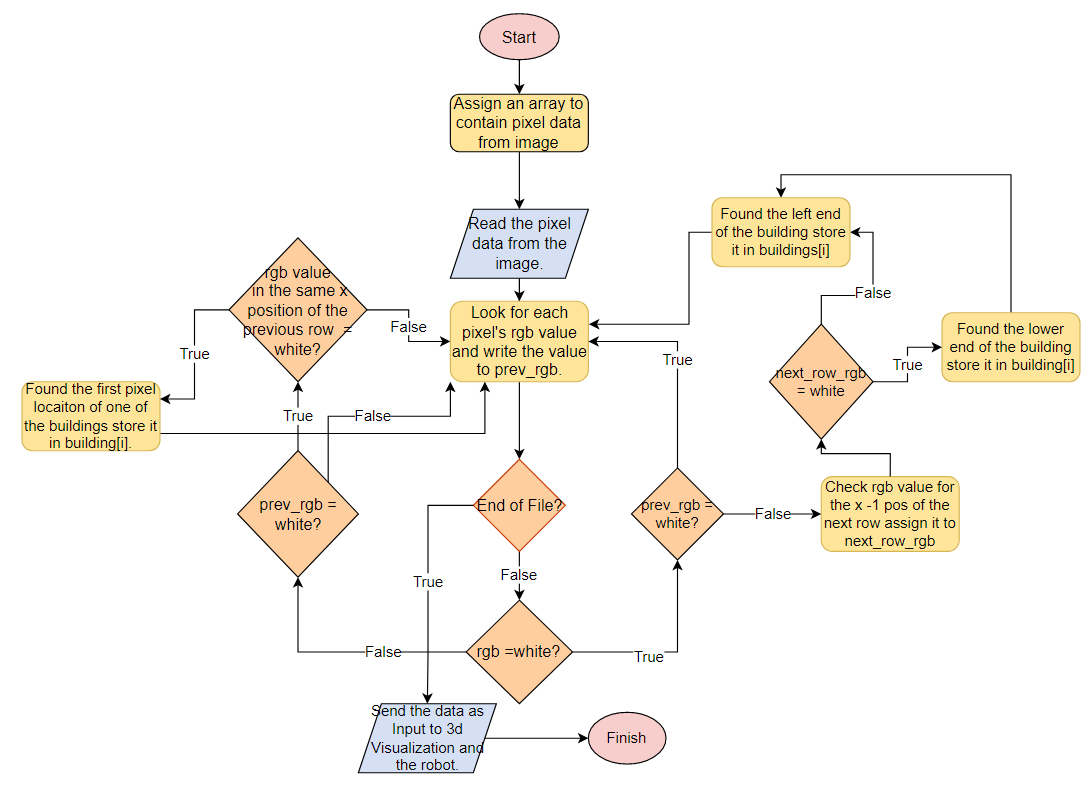
The use case diagram for the image processing algorithm is shown in Figure 8 below.



**Figure 8. Use Case Diagram**

### 3.1.4. Architecture

In the 3D representation to represent the data, some landscape materials can be added between buildings and other materials like roads, road signs. Briefly aesthetic concerns regarding the end result of 3D visualization should be discussed. In Figure 9 the flow chart of the image processing part is illustrated.

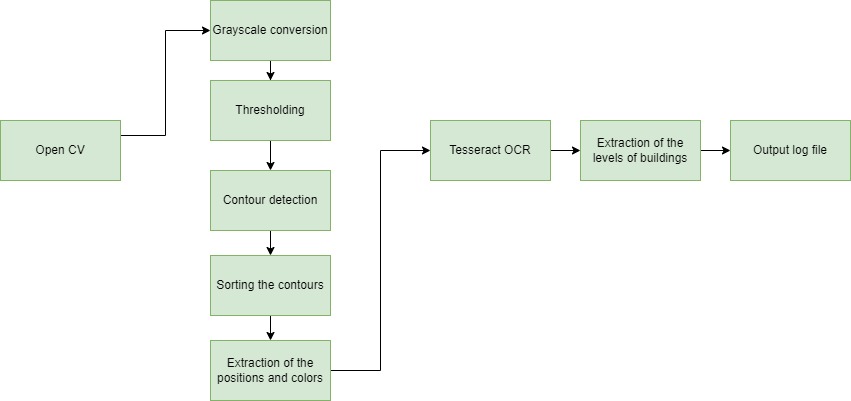


**Figure 9. Flow Chart for Image Processing**

### 3.1.5. Materialization

We first decided to use C++ with Visual Studio. We had to integrate OpenCV by downloading the project from the web and then associating the necessary files with our project in Visual Studio. We successfully integrated OpenCV into our project and managed to detect coordinates as well as color data for buildings in the given image. For the last requirement, we needed an OCR engine. During our research, we encountered Tesseract, a widely used open-source OCR engine. We tried to install Tesseract, but when we downloaded it, we had an executable program in our hands. We were able to use it as a program, but when it came to integrating it into our project in Visual Studio, this was not the way to go. Then we discovered that we had to build the project from the source code. Following instructions, we found that by using vcpkg, a free C/C++ package manager, we could be able to build Tesseract into our project. After getting our hands dirty with all these tools, the process failed, and we couldn't integrate the program into our project. Instead of trying harder and forcing a solution, we thought that managing packages was way easier in Python. If OpenCV and Teserract had packages built for Python, all we had to do was run pip to install them, and luckily they did.

In the Figure 10, the workflow of the image processing is represented.

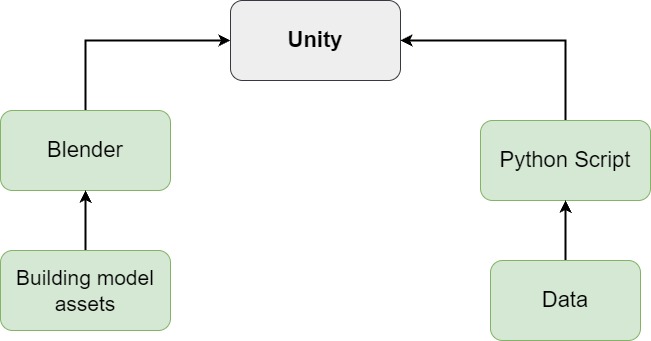


**Figure 10. Workflow of the Image Processing**

For the 3D representation of the buildings, we started with installing and creating the environment in Blender, which provides a powerful system for setting up and editing materials. Once it was set up, we were able to use the features of it for modeling, editing, adding details and rendering easily.

While models were being prepared on parallel we were working on Unity. There were some libraries and packages that we had to use in order to achieve the desired functionality. Two main things that we had to implement which Unity doesn’t support on its own were; reading JSON data and executing a process that is locally stored. To execute a process we used a C# class named process. This class provides access to local processes and enables starting and stopping them. For parsing the JSON data we used Newtonsoft’s Json.NET framework [31]. This is a third party free JSON framework for .NET applications. Using this framework we were able to read the JSON string into a C# Dictionary (C# Dictionary is a data structure consisting of key, value pairs).

In the Figure 11, the workflow of the 3D modeling is illustrated.

****

**Figure 11. Workflow of the 3D modelling**

As for the integration part, We obtained an Arduino Uno microcontroller and we connected it to a windows machine. What we had to achieve was to parse JSON string using the microcontroller. We found a free third-party library that has the functionality that we need. The library is named ArduinoJson by Benoit Blanchon [32]. We sent the data to Arduino and executed the algorithm to parse the data on the microcontroller using the library. To ensure that we are able to extract data accurately we printed the data to the Serial Monitor and the result showed that the process was a success. The workflow of the Arduino data retrieval is seen in Figure 12.

metin, ekran görüntüsü, diyagram, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Figure 12. Workflow of the Arduino data retrieval**

### 3.1.6. Evaluation

After the environment was set, we were ready to begin processing the image. In order to obtain the positions of the rectangles we need to detect edges by finding the contours of the image. Contours are the curves created by following a pattern that has continuous points that has same color or intensity [30]. In order to let OpenCV generate the best contours, it is recommended to apply some preprocessing techniques to the image. First, we need to get a grayscaled version of the image. OpenCV's cvtColor library can help us with that. After getting the grayscaled image, we applied a conversion to a binary image by setting a threshold that colors everything under the threshold black and everything above white. Doing this resulted in an image with a white background and black rectangles, but OpenCV's contour finding algorithm works in a way that, finds white objects in a black background. So we passed the invert argument to the function and got an image that has a black background with white rectangles. After all these steps, the next thing to do was to find contours. So we used the OpenCV method to find the contours and the result we got was promising but not enough. OpenCV contoured all the shapes including the numbers written inside the reactangles. After reading the documentation and understanding how OpenCV generates contours we found out that there is a hierarchical structure between contours. Every contour or we can say shape, that has another shape in it, is the parent of the inner contour. Passing an argument that only takes external contours to the method that finds contours, solved the problem. Now the output was looking better with only the positions of the rectangles and nothing else but still something was not right. Though in the input image we had 5 rectangles, in the code we were only able to locate the positions of 4 of them. When we looked at the input image, we observed that the missing rectangle had a unique attribute, which was its color. It had a much brighter color compared to all the other rectangles. We thought that the problem was most likely caused by this phenomenon. In order to identify the problem, we executed the code line by line and we found that we were losing that rectangle at the conversion to the byte image part. In the corresponding OpenCV function for conversion, there was an argument to set the threshold. So we increased the threshold to have brighter pixels to be included in the conversion and that fixed the problem. With that we were now able to find the positions of each rectangle. The next thing to do was getting the color information of the rectangles. We were trying to obtain the color data from the pixel position where we located the rectangles. Later on we found out that since the position of the rectangle corresponds to their edges, there would be pixel blending in that area. That means not all of the pixels were going to be either the color of the rectangle or the background's color. There were going to be some pixel values that are in between. To solve that, we added a margin to the position to get the color data, and by doing that, our program was ready to find the rectangles and identify the color of them.

The next step was to find the levels of the buildings. In order to measure them, we used the Tesseract OCR library. We first started by sorting the rectangles based on their x coordinates and their contours. Then we moved on to defining the region of interest (ROI) in the grayscale image corresponding to the contour. Using OCR techniques, the text presented on top of the building map image was extracted, level of the buildings were visible accurately in the output.

After the output log was done, 3D modelling of the buildings was next. We started by creating the 3D models of the buildings using Blender. The input map guided us to understand the design and structure of the buildings. A 3D scene was created to accurately model and represent the building’s environment. We made it look more realistic by adding features such as windows include glasses with reflections, doors, door knobs and roofs. These elements were modeled using shapes such as cubes, cylinders and planes which were then scaled, rotated and positioned. The colors and the levels of the buildings were represented accurately according to the input map. Once the modeling process was complete next up on the list was Unity integration.

We imported the models to Unity, then turned them into a prefab (a prefabricated game object that makes it easier to spawn object into the scene). The program that will be coded in Unity was going to be the final product. We wanted to create an user interface to enable the user to input the image and get the scene without further implementation. To achieve that, the program needs to be able to run the python script externally but without any user interference. To solve this problem we used the process class of C#. This process class, enables you to run executables through the shell. We provided the process class with three parameters; the path of the python executable, the path to the python script and the input image. Using these 3 parameters, the program executes the python script and the python script, writes the output data into a secure folder on the machine. After the data is written and ready, we started the process of scene generation. First we had to parse the JSON string. Unity has a class that enables you to read and assign JSON data into a class with the same field variables that are present on the JSON string. This class comes in handy for game development purposes but for the job at hand, we had to write the JSON data into a C# Dictionary. After some research we found a framework that can do the job for us with the built-in JSON parser. We implemented it into the code but it was not recognized by the program we checked to see if we made any mistake in importing the framework but we couldn’t find any. Then we tought of switching to Visual Studio rather than using Visual Studio Code for script editing. With that transition the problem was solved Visual Studio recognized the framework and we parsed the JSON string into a Dictionary. Now we were able to use the generated output and we were ready to generate the scene.

One thing to consider when generating a scene is relative world position. We obtain the position data from python which corresponds to the pixel position of the rectangles in the input image. For example, we get x position as 230px, y position as 345px. In order this information to have a meaning, we need the size of the image, then we can have a relative understanding of the positions of the buildings. So what we did was to also read the size of the image in Unity. Then we wrote a function that maps a number from an interval to another. For example, for the X coordinates we mapped the value from (0, Image Width) interval to (-0.5, 0.5). This way we were able to generate buildings on a platform considering the size and orientation of the platform. With that, we are able to dynamically strecth and squeeze the platform and the buildings’ positions change accordingly.

Other things to consider were; the color of the buildings and the number of floors that a building has. We were able to overcome these matters using traditional methods. We used a switch case block to decide which building model to be generated based on the number of floors parameter and for the color we used Unity’s built-in color class and fed it with the hex-color parameter. Then we generated a material that has the color that we want and assigned it to the rendere component of the building.

After image processing and 3D modeling, we moved on to Arduino. At the Arduino part we didn’t encounter any major problems since the main task is to read information which is formatted in a pretty common JSON method. We were able to find a library and implement it into the code sent the code to the microcontroller and the microcontroller was able to parse the JSON string and extract the needed information.

As seen in Figure 13, 15 and 17 an input map is illustrated with their output logs that will be generated and fed to the robot. And in Figure 14, 16 and 18 a 3D representation output example is illustrated.

metin, ekran görüntüsü, diyagram, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Figure 13. The Map of Buildings and Output Log of input 1**

ekran görüntüsü, piksel içeren bir resim

Açıklama otomatik olarak oluşturuldu**Figure 14. 3D Visualization for input 1**

**metin, ekran görüntüsü, diyagram, dikdörtgen içeren bir resim

Açıklama otomatik olarak oluşturulduFigure 15. The Map of Buildings and Output Log of input 2**

3B modelleme, ekran görüntüsü, video oyunu yazılımı, piksel içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Figure 16. 3D Visualization for input 2**

metin, ekran görüntüsü, dikdörtgen, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu  **Figure 17. The Map of Buildings and Output Log of input 3**

oyuncak, çizgi film, piksel içeren bir resim

Açıklama otomatik olarak oluşturuldu**Figure 18. 3D Visualization for input 3**

In the image shown in Figure 13, Figure 15 and Figure 17 the numbers given represent the height of the buildings. There are few ways to read them with image processing methods.

**OCR:** Extracting text from an image can be allowed by Optical Character Recognition(OCR) technique. Using OCR, converting the numerical information in the image into a form that can be easily processed and analyzed can be achieved. [25]

**Image segmentation**: In order to separate individual buildings from the environment or other objects in the input image, image segmentation techniques are used. Image segmentation can divide the input image into clear regions or segments, each of which corresponds to a different object. [26]

**Feature extraction**: After identifying the individual buildings in the input image, Using feature extraction techniques can extract specific features or characteristics of these buildings, such as their shape, size, and color. These features can help to differentiate between buildings of different heights and to determine the height of each building. [27]

**Measurement:** Measurement techniques can determine the height of each building by using techniques such as scale estimation or perspective correction to recoup for any distortions in the input image, and to accurately measure the size and shape of the buildings.

We used the Optical Character Recognition method.

## 3.2. Mechatronics Engineering

Mechatronics subsystems are electrical system design, scara robot arm chassis design, programming the system and system integration.

### 3.2.1. Requirements

* Constructor robot should be able to construct at least 2 floor buildings.
* There should be 2 buildings next to each other from the robot's point of view.
* Building block dimensions are 21x21x21 cm.

### 3.2.2. Technologies and methods

Constructor robot based on the SCARA robot technology. The most important part in a constructor robot is the motor. We decided to use servo motors for controlling the speed, acceleration and location of the robot. Constructor robot should be able to build at least 2 floor buildings, because of that the design of the robot should be based on this feature. So, the length of the robot should be 20 centimeters to build a structure of this size. Since we determined the maximum building floor as 3, we planned to make the length of the robot as 20 centimeters.

### 3.2.3. Conceptualization

In section 1.3.2. we reviewed three different robot concepts and decided to build a SCARA type robotic arm. We prefer to use an Arduino UNO in comparison with Raspberry Pi Pico because we are familiar and experienced with the Arduino ecosystem from MCH 1003 course. Also, Arduino has larger libraries and communities than others. Since our project is going to build on an Arduino UNO platform, using a CNC shield for motor drivers will be useful in case of cabling and connections. One of the most important parts of our project is the step motor. In our task, we need powerful motors to carry blocks and buildings. So we decided to use NEMA17 size, 40 N.cm tork step motors. For grabbing the building blocks we design a grabbing hand on the working end of the robot arm. Grabbing hand powered by a MG996 servo motor which has 9,4 kgf · cm (4,8 V) tork.

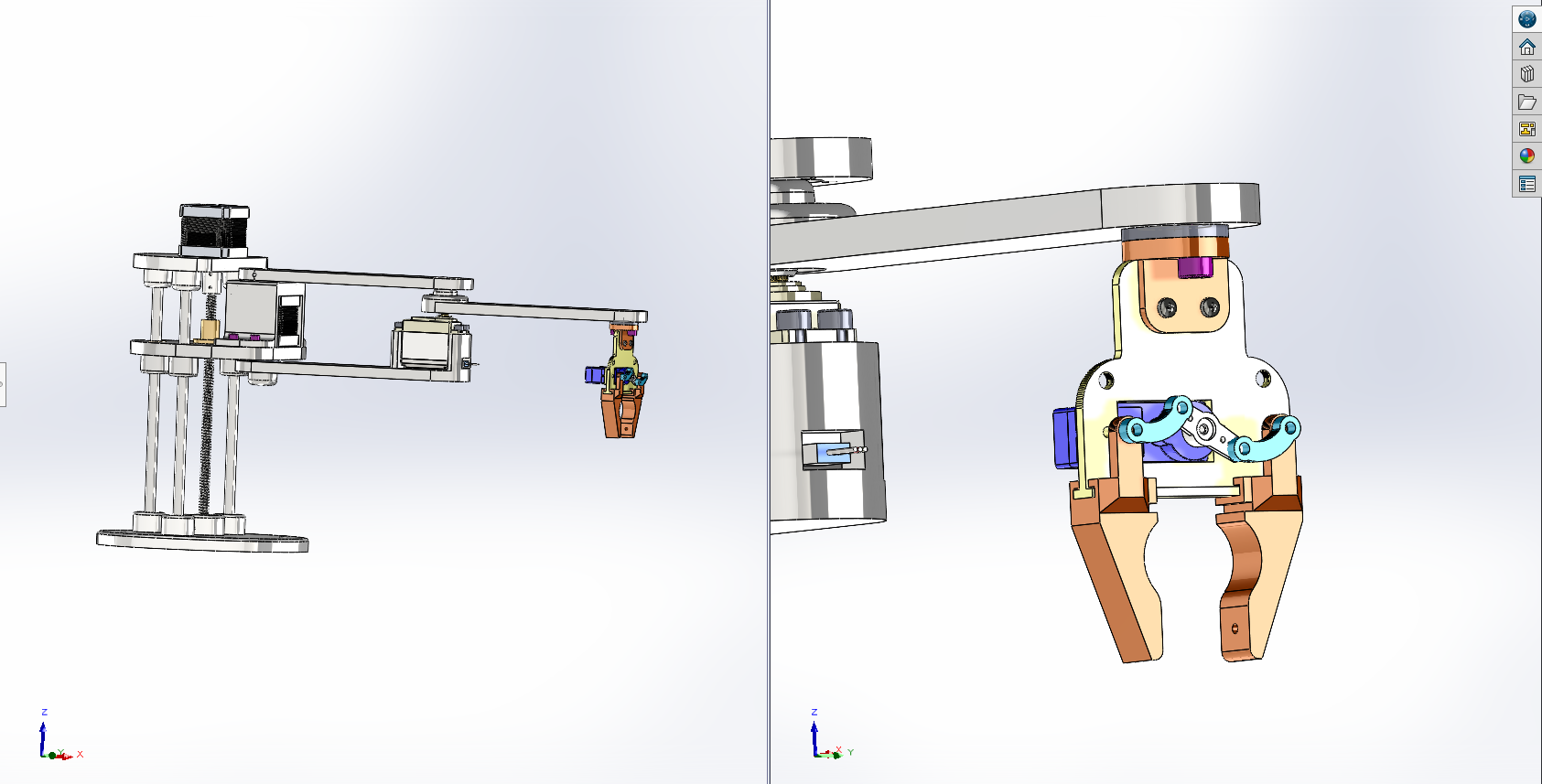
### 3.2.4. Physical architecture

The base, the arms, the end effector, and the controller make up the four essential parts of a SCARA robot's physical architecture. The base serves as the robot's structure. It must be strong enough to withstand the total mass of the robot and its parts.

The robot's arms, which are the major components that rotate, are what provide the end effector its motion. The tool used to perform the required task is called an end effector, which is attached to the end of the arm. It will be gripper.

The robot's controller, which serves as its brain, tells the robot how to move its arms and end effector.

For the movement of the arms, were planned as 2 Stepper Motors and Servo Motor. Servo Motor receives information about current position and adjusts motion accordingly. Signals from the robot's control system that specify the desired position and velocity of each joint are sent to the servo motors attached to them. The software will be positioned according to their situation in the place where it is located. And we will also use stepper motors to provide vertical motion controls.



**Figure 19. Physical architecture of the robot.**

### 3.2.5. Materialization

1. The general steps a SCARA robot materialization process involves are as follows:
2. Design: The drawing we made on Solidworks, where we designed the dimensions and features such as the gripper for the robot to perform the desired functions.
3. Component Selection: After the design was finished, we had to decide which motors, actuators, joints, controllers, and other hardware would work best for the robot.
4. Assembly: In accordance with the design specifications, this connects the robot's arms, joints, and end components, such as grippers or tools.
5. Electrical system design: Determining Motor Specifications, Wiring and Connections, Testing and Integration
6. Programming the system: Arduino and processing code. The movements, tasks, and interactions of the robot with its environment are all described in programming.

### 3.2.6. Evaluation

A SCARA robot has a control system, motors, a transmission system, position sensors, end effectors, and a power supply as its subsystems. While the motors give the robot the power to move, the control system is in charge of giving it motion and direction commands. The position sensors collect data from the robot arm so that the control system can keep track of its present position and velocity. The transmission system is in charge of transferring power from the motor to the robot arm. The power supply powers the robot, and the end effectors are the tools like grippers that are attached to its arm.

For design part we decided to make robots in smaller sizes beacuse of the 3D parts. Since it would cause financial difficulties, we went down in the design. This wasted our time. And the extraction of some parts that we shrunk caused trouble. While we had limited time to remove the spindle, it did not come out of the printer properly and it was difficult for us to find a product with an 8mm diameter. We lost a lot of time as a result of editing the drawing twice and the results of the 3D printer. We could not assemble the robot completely due to the parts problems we experienced (parts do not fit together, they are damaged as a result of forcing).

iç mekan, mobilya, masa, bilgisayar monitörü içeren bir resim

Açıklama otomatik olarak oluşturuldu

makine, iç mekan, kablo, alet içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Figure 20. Shaft part that can not come out of the printer**

# 4. INTEGRATION AND EVALUATION

## 4.1 Integration

As illustrated in figure 2 Image processing subsystem’s output is a log file to help navigate the robot. Building the constructor robot subsystem’s final product is a robot that is able to move and place objects. In order to integrate these two subsystems, the data that is stored in the output log file needs to be migrated to the microprocessor's memory unit. This process can be achieved by plugging Arduino to a PC via an USB port and using Arduino IDE, data can be written to the memory of Arduino. This data needs to endure for a long period of time and it shouldn’t be affected by power loss. Atmega328P is the micro-controller that is integrated in the Arduino board that we are going to use and this micro-controller provides a memory unit named EEPROM that is non volatile and safe in case of power loss. Using Arduino IDE and EEPROM library the output log file will be written to memory and this way two subsystems are integrated.

Figure 2 represents the interface diagram. For the computer part of this project, as seen in figure 2, the read pixel algorithm will consist of reading the image and extracting the rgb. The pixel reading is integrated with the image processing interface. The image processing interface analysed and extracted data from the image such as its height, color and position. The data stores information about buildings, which consists of the number of buildings, their color, position and height. The 3D visualization interface is integrated with the data, it will consist of getting models, assets and building’s data and creating environments and buildings. And lastly, the log output interface contains generating a log file and writing data to a file.

For the mechatronics part of the project being able to store and carry out the robot's preprogrammed movements, the controller should be able to communicate with the motor and servo controllers to move the robot arm. Microcontroller will read the given input and control the motion.

To process the input image, we needed to use the OpenCV library. Initially, we decided to code in C++; although integrating OpenCV was easy in C++, for the further processing of the project, we needed an OCR engine. During our research, we found out about a widely used OCR engine called Tesseract. While trying to integrate Tesseract into Visual Studio, we encountered some problems and struggled to integrate it into Visual Studio; therefore, we decided to continue with Python, which is way easier to manage the package and can be done by running pip to install the packages. After setting our tools, as seen in figure 3, we started by receiving and preprocessing the input image. The phase was successful, so we continued with segmenting the image. After accomplishing the segmenting stage, we extracted the features of the buildings.

Later in the project process, since the extraction of data of the buildings has been accomplished, we continued with measuring the heights of the buildings with the Tesseract OCR engine, which was done by extracting the text from the image function. After we extracted the necessary information we wrote it to a file in JSON format. All this is handled by a python script. The next thing to do is to generate a 3D scene based on that data. To accomplish that, we used Unity engine. In Unity engine we coded the logic in C#. There are many different scripts that handle different tasks. Most important ones are; the Python Bridge (a script that executes the python script by giving parameters to it), Input Scene Manager (a script that allows user to input necessary file and program paths) and Scene Generator (generates the 3D scene based on the input the user provided).

Following the measuring process, a map of buildings was generated to feed the position, height, and color data. With the last data we have, we will continue creating 3D visualizations of the buildings.

Since the mechatronics part of the project was limited, we used the input map in Figure 21 for integration.

Post-it notu, dikdörtgen, kare, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Figure 21. The Map of Buildings considering the limitations**

For the Arduino part, we had to ensure that we are able to send the data to arduino from the Serial Communication Port. To achieve this, we connected the Arduino to a machine and found an Arduino library that can parse JSON string. The library that we used was ArduinoJson by Benoit Blanchon. Using the built-in functions that the library has we parsed the JSON string that contains building information, sent the data to Arduino and to ensure that we are able to transfer the data to Arduino we printed the variables to the Serial Monitor and the process was a success.

Figure 6 illustrates PN and Figure 7 illustrates Gantt chart. The representation of these figures illustrates the scheduling of the planned integration. As we can see in Figure 6, we started with coding the logic for the game engine and moved on with modeling the buildings and the environments. And developing the 3D software environment was necessary to combine with extracting the data using image processing techniques for input data and 3D representation. All of these phases was integrated with a 3D software tool.

Figure 7 represents the Gantt chart, we had 14 weeks to complete this project. Computer engineering students started with providing an input building map to the mechatronics students. In the first weeks, CMP students extracted the input data. During this process, MCH students will deal with the cad design and manufacturing of the robot. In the following weeks, CMP students decided on the visualization design, 3D modeling for buildings and environment was set up and the logic of the visualization process was coded through a game engine. MCH students will work on the layout and assembly of the robot and the system design. Following the plan, CMP students combined the materials and received the result. MCH students will finish the software of the system and move on to the integration and testing parts. To have a successful and efficient integration, the data migration should be confirmed. As for the last stages of the project, both departments worked on documentation and presentation for the final report.

## 4.2 Evaluation

In order to evaluate the performance of our system for generating input data, we performed several experiments using different images with varying levels of complexity. The experiments were designed to test the ability of our system to accurately interpret the building map and determine the colors and positions of the structures to be built. First, we started with a simple image with five rectangles of equal size and height. While the algorithm was succesful in locating the positions of 4 of the rectangles, one of the rectangle was missing. We fixed this problem by increasing the threshold to include brighter pixels, which resulted in the identification of all the rectangles. Another issue we faced while processing the image was the difficulty in obtaining the color data fort he rectangles. However, we overcame the problem by adding a margin to the position to capture the color information. The accuracy and precision of the image processing and measuring techniques was are included in the performance requirements. So far, details of the input image is accurate. Functional requirements include the capacity to use image processing techniques to extract details about the buildings from an input image. The image was converted to grayscale and generated into a binary image through thresholding succesfuly. Overall, our image processing techniques satisfied the functional and performance requirements so far. It was able to accurately interpret the building map, determine the positions and the color of the buildings to be structured.

3D modeling part ended up being a satisfactory process. We were able to generate a scene based on the input and the algorithm was a flexible and robust one since, it was able to work with different image sizes and different scene platform sizes. The scene is being generated in a fair amount of time, generally the process is done under 1.5 seconds. Since we were experienced with the 3D softwares we used, the errors that we’ve encountered were solved in a short amount of time. In the end, we have a program that takes a basic 2D input and generates a 3D movable environment with camera movement integration.

For the Arduino part we were able to send the data to the Arduino microcontroller through Serial Port. We sent the data in JSON format and we accomplished parsing the JSON string inside the microcontroller using a third party library. So we made sure that, all the data was readable and ready for the microcontroller to process. Figure 22 and 23 illustrates the process of parsing JSON data.



**Figure 22. Example of JSON String**

metin, yazı tipi, ekran görüntüsü, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Figure 23. Output of parsed JSON data**

Scara robots involve assembling and welding components such as base, arms, end effectors, sensors, cables and motors. The robot needs to be calibrated and tested before it is put into service. There can be a need for extra customizations, depending on the application. For instance, additional programming may be required so that the robot can correctly identify and retrieve objects when picking up and placing them.

Overall, the development of the robot is a difficult process that requires accuracy and close attention to detail. To ensure proper operation, the robot needs to be constructed and calibrated correctly.

# 5. SUMMARY AND CONCLUSION

In summary, using image processing techniques, information about the buildings from an input image was effectively extracted and this information was used by the 3D softwares to generate 3D representation of the image. At the final stage of the product input data is being processed fast and accurate enough to satisfy requirements. In this project there were parts that we were less experienced and solving errors took more time compared to, errors generated at the parts that we were more expreienced at. We learned as we developed throught the whole project. In the end, on the Image Processing side our program is able to generate an output considering height, positin and color information from the input image that can be transferred to a microcontroller and generate a 3D scene consisting of the buildings representings the ones on the input image. Using that data, a constructor robot that will build structures based on the map will be implemented. A type of robot called constructive robot is made to assemble parts independently to create structures in a particular environment. For the assembly of the parts, they must be perfect. The biggest problem we have experienced in this regard was that the gear mile cannot sit on the piece.

In conclusion the project is successful because it accurately and efficiently built the structures in accordance with the map. Using image processing techniques, the map was analyzed and floors of each building was determined correctly. In order to construct the buildings, the construction robot is capable of picking up objects of the same height, stack them on top of one another, and visually check that the finished structures followed the map's specifications. The map was represented in 3D. The built buildings is visualized in a realistic and detailed form owing to 3D visualization software. As for the integration part, the microcontroller is able to read and parse the generated output. The robot can learn the layout of the buildings by using a map, and it can use that information to plan effective routes and navigate the area. A constructor robot can successfully navigate and construct a given map of buildings with the right design on solidworks with given information, combination of materials and programming. While doing the project, there was a problem or disruption in most parts. But by giving all our efforts as much as possible, we continued by learning new things to solve the problems and we did our best.

# ACKNOWLEDGEMENTS

We wish to thank our adviser Assist. Prof. Selin Nacaklı, Assoc. Prof. Ozan Akdoğan.

This work was partly/wholly funded by Bahçeşehir University

# REFERENCES

1. Statista, <<https://www.statista.com/statistics/1290105/global-construction-market-size-with-forecasts/>>, Last Accessed 29.12.2022
2. Bridgit,<<https://gobridgit.com/blog/how-to-calculate-construction-labor-cost/#:~:text=According%20to%20the%20industry-standard,payroll%20taxes>>,Last accessed 29.12.2022
3. Unity, <<https://unity.com>>, Last Accessed: 29.12.2002
4. Unreal Engine, <<https://www.unrealengine.com/en-US>>, Last Accessed: 29.12.2022
5. Blender, <<https://www.blender.org>>, Last Accessed: 29.12.2022
6. Maya,<<https://www.autodesk.com/products/maya/overview?term=1-YEAR&tab=subscription>>, Last Accessed, 29.12.2002
7. University of Tartu 2014, Assoc. Prof. Dr. Gholamreze Anbarjafari, <<https://sisu.ut.ee/imageprocessing/book/1>>
8. Szeliski, R. (2010). Computer vision: algorithms and applications. Springer Science & Business Media.
9. Wikipedia 2022, <<https://en.wikipedia.org/wiki/Visualization_(graphics)>>, Last Accessed: 29.12.2022
10. OpenCV, <<https://opencv.org>>, Last Accessed: 29.12.2022
11. Scikit-learn, <[https://scikit-learn.org](https://scikit-learn.org/)>, Last Accessed: 29.12.2022
12. 3D Visualization Output Example, <https://forums.unrealengine.com/t/low-poly-suburbs/576135>, Last Accessed 29.12.2022
13. Scara robot and Cartesian robot

[Scara vs. Cartesian Robots: Selecting the Right Type for Your Applications](https://www.automate.org/industry-insights/scara-vs-cartesian-robots-selecting-the-right-type-for-your-applications) Last Accessed 20.12.2022

1. Scara robot and Six-axis robot [The Basics Of 6-Axis, SCARA, Delta And Collaborative Robots](https://www.crossco.com/resources/technical/robotics-101-6-axis-vs-scara-vs-delta-vs-collaborative/) Last Accessed 31.12.2022
2. [What’s the Difference Between Six-Axis, SCARA, and Cartesian Robots? - SDC Automation](https://sdcautomation.com/whats-the-difference-between-six-axis-scara-and-cartesian-robots/) Last Accessed 31.12.2022
3. [12V 6A Kapalı Tip Switch Mode Adaptör Uygun Fiyatıyla Satın Al - Direnc.net®](https://www.direnc.net/12v-6a-kapali-tip-switch-mode-adaptor) Last Accessed 30.12.2022
4. [MG996 Metal Servo Motor Uygun Fiyatıyla Satın Al - Direnc.net®](https://www.direnc.net/mg996-metal-servo-motor) Last Accessed 30.12.2022
5. [Nema 17 HS4401 Step Motor Uygun Fiyatıyla Satın Al - Direnc.net®](https://www.direnc.net/nema17-hs4401-step-motor) Last Accessed 30.12.2022
6. [MICRO SWITCH UZUN MAKARALI - 8,00 TL](https://www.kartalotomasyon.com.tr/urun/micro-switch-uzun-makarali) Last Accessed 30.12.2022
7. [3D Yazıcı Filament Fiyatları ve Çeşitleri %44 İndirim - ABS, PLA, PETG](https://www.robotistan.com/3d-filament) Last Accessed 30.12.2022
8. [10 Mm Krom Kaplı İndüksiyonlu Taşlanmış Mil - MERMAK'TA KAMPANYA](https://www.cnc-marketi.com/urun/10-mm-krom-kapli-induksiyonlu-taslanmis-mil) Last Accessed 30.12.2022
9. [Arduino UNO CNC Shield Uygun Fiyatıyla Satın Al](https://www.direnc.net/arduino-uno-cnc-shield) Last Accessed 30.12.2022
10. [Arduino Uno R3 SMD CH340 Chip - Klon (USB Kablo Dahil) Uygun Fiyatıyla Satın Al - Direnc.net®](https://www.direnc.net/arduino-uno-r3-smd) Last Accessed 30.12.2022
11. [A4988 Step Motor Sürücü Devresi Uygun Fiyatıyla Satın Al - Direnc.net®](https://www.direnc.net/a4988-step-motor-surucu-devresi) Last Accessed 30.12.2022
12. AWS, < <https://aws.amazon.com/tr/what-is/ocr/>>,Last Accessed: 15.01.2023
13. Wikipedia, <<https://en.wikipedia.org/wiki/Image_segmentation>>, Last Accessed: 15.01.2023
14. Wikipedia, <<https://en.wikipedia.org/wiki/Feature_extraction> >, Last Accessed: 15.01.2023
15. Pypi, < <https://pypi.org/project/pytesseract/>>, Last Accessed: 06.05.2023
16. Pypi, < <https://pypi.org/project/opencv-python/>>, Last Accessed: 06.05.2023
17. OpenCV Docs, <<https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html> >, Last Accessed: 06.05.2023
18. Newtonsoft, < <https://www.newtonsoft.com/json>>, Last Accessed: 11.06.2023
19. ArduinoJson, < <https://arduinojson.org>>, Last Accessed: 11.06.2023

# APPENDIX A

import sys

import pytesseract

import cv2

import json

def rgb\_to\_hex(rgb):

    return '#' + '%02x%02x%02x' % rgb

image\_path = sys.argv[1]

output\_dir = sys.argv[2]

img = cv2.imread(image\_path)

size\_of\_image = img.shape

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  # Convert image to grayscale

\_, thresh = cv2.threshold(gray, 210, 255, cv2.THRESH\_BINARY\_INV)  # Set a high threshold as 210 to detect bright-colored

# rectangles

contours, \_ = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  # Use RETR\_EXTERNAL to exclude

# numbers inside the rectangle by getting only parent contours

buildings\_dict = {}

for i, contour in enumerate(contours):

    x, y, w, h = cv2.boundingRect(contour)

    color\_data = img[y + 10, x + 10]  # add 10 to get away from the edges and get the certain color data.

    color\_data\_rgb = (color\_data[2], color\_data[1], color\_data[0])

    color\_hex = rgb\_to\_hex(color\_data\_rgb)  # Color data in hexadecimal

    roi = gray[y:y + h, x:x + w]

    number\_text = pytesseract.image\_to\_string(roi, config='--psm 6').strip()

    dict\_key = "Building" + str(i)

    buildings\_dict[dict\_key] = {'PosX': str(x), 'PosY': str(y), 'LengthX': str(w), 'LengthY': str(h),

                                'Color': color\_hex, 'Floors': number\_text}

buildings\_dict['image\_height'] = {'height': size\_of\_image[0]}

buildings\_dict['image\_width'] = {'width': size\_of\_image[1]}

buildings\_json = json.dumps(buildings\_dict, indent=4)

print(buildings\_json)

with open(output\_dir + "buildings\_data.json", "w") as outfile:

    outfile.write(buildings\_json)

# APPENDIX B

void setup() {

  char input[] =  "{\"Building2\": {\"PosX\": \"654\", \"PosY\": \"256\", \"LengthX\": \"129\", \"LengthY\": \"113\", \"Color\": \"#feaec9\", \"Floors\": \"4\"}}";

  StaticJsonDocument<192> doc;

  DeserializationError error = deserializeJson(doc, input);

  if (error) {

    Serial.print(F("deserializeJson() failed: "));

    Serial.println(error.f\_str());

    return;

  }

  JsonObject Building2 = doc["Building2"];

  const char\* Building2\_PosX = Building2["PosX"]; // "654"

  const char\* Building2\_PosY = Building2["PosY"]; // "256"

  const char\* Building2\_LengthX = Building2["LengthX"]; // "129"

  const char\* Building2\_LengthY = Building2["LengthY"]; // "113"

  const char\* Building2\_Color = Building2["Color"]; // "#feaec9"

  const char\* Building2\_Floors = Building2["Floors"]; // "4"

  Serial.begin(9600);

  Serial.println(Building2\_PosX);

  Serial.println(Building2\_PosY);

  Serial.println(Building2\_LengthX);

  Serial.println(Building2\_LengthY);

  Serial.println(Building2\_Color);

  Serial.println(Building2\_Floors);

}

void loop() {

  // put your main code here, to run repeatedly:

}

# APPENDIX C

using UnityEngine;

using UnityEditor;

using UnityEngine.UI;

using System.IO;

using UnityEngine.SceneManagement;

public class UIManager : MonoBehaviour

{

    string pythonPath;

    string imagePath;

    string outPath;

    public InputField pythonInputField;

    public InputField imageInputField;

    public InputField outFileInputField;

    public void GenerateSceneButton()

    {

        bool inputFieldsFilled = pythonInputField.text.Length > 0 && imageInputField.text.Length > 0 && outFileInputField.text.Length > 0;

        if (!inputFieldsFilled) return;

        string pythonScriptPath = Application.dataPath + "/Resources/ImageDataExtractor.py";

        Debug.Log(pythonScriptPath);

        string pythonExePath = pythonInputField.text;

        string imagePath = imageInputField.text;

        string outFilePath = outFileInputField.text;

        string buildingsInfo = PythonBridge.RunPythonScript(pythonScriptPath, pythonExePath, imagePath, outFilePath);

        string path = Application.persistentDataPath + "/BuildingInfo.json";

        Debug.Log("Wrote the file into: " + path);

        File.WriteAllText(path, buildingsInfo);

        SceneManager.LoadScene(1);

    }

    public void PythonBrowseButton()

    {

        pythonPath = EditorUtility.OpenFilePanel("Choose python executable", "", "exe");

        pythonInputField.text = pythonPath;

    }

    public void ImageBrowseButton()

    {

        imagePath = EditorUtility.OpenFilePanel("Choose the source image", "", "png,jpg");

        imageInputField.text = imagePath;

    }

    public void OutputFileBrowseButton()

    {

        outPath = EditorUtility.OpenFolderPanel("Choose Output File Directory", "", "");

        outFileInputField.text = outPath;

    }

}

# APPENDIX D

using System.Diagnostics;

public static class PythonBridge

{

    public static string RunPythonScript(string scrpitPath, string pythonPath, string imagePath, string outFilePath)

    {

        var psi = new ProcessStartInfo();

        psi.FileName = pythonPath;

        var scrpit = scrpitPath;

        var imagePathMessage = imagePath;

        var outFilePathMessage = outFilePath;

        psi.Arguments = $"\"{scrpit}\" \"{imagePathMessage}\" \"{outFilePathMessage}\"";

        psi.UseShellExecute = false;

        psi.CreateNoWindow = true;

        psi.RedirectStandardOutput = true;

        psi.RedirectStandardError = true;

        var errors ="";

        var results = "";

        using(var process = Process.Start(psi))

        {

            errors = process.StandardError.ReadToEnd();

            results = process.StandardOutput.ReadToEnd();

        }

        UnityEngine.Debug.Log("ERRORS:");

        UnityEngine.Debug.Log(errors);

        UnityEngine.Debug.Log("RESULT:");

        UnityEngine.Debug.Log(results);

        return results;

    }

}

# APPENDIX E

using System.IO;

using System.Collections.Generic;

using UnityEngine;

using Newtonsoft.Json;

using Unity.VisualScripting;

public class SceneGenerator : MonoBehaviour

{

    private Dictionary<string, Dictionary<string, string>> \_buildingsDict;

    public GameObject oneFloorBuilding;

    public GameObject twoFloorBuilding;

    public GameObject threeFloorBuilding;

    public GameObject fourFloorBuilding;

    public GameObject fiveFloorBuilding;

    public Material defaultMaterial;

    public GameObject baseGround;

    void Start()

    {

        string buildingsJsonPath = Application.persistentDataPath + "/BuildingInfo.json";

        var buildings\_json = File.ReadAllText(buildingsJsonPath);

        \_buildingsDict = JsonConvert.DeserializeObject < Dictionary<string, Dictionary<string, string>>>(buildings\_json);

        float imageSizeX = int.Parse(\_buildingsDict["image\_width"]["width"]);

        float imageSizeY = int.Parse(\_buildingsDict["image\_height"]["height"]);

        foreach (string key in \_buildingsDict.Keys)

        {

            if(key != "image\_width" && key != "image\_height")

            {

                Debug.Log("HELLO");

                float posX = int.Parse(\_buildingsDict[key]["PosX"]);

                float posY = int.Parse(\_buildingsDict[key]["PosY"]);

                GameObject building;

                switch (int.Parse(\_buildingsDict[key]["Floors"])){

                    case 1:

                        building = oneFloorBuilding;

                        break;

                    case 2:

                        building = twoFloorBuilding;

                        break;

                    case 3:

                        building = threeFloorBuilding;

                        break;

                    case 4:

                        building = fourFloorBuilding;

                        break;

                    case 5:

                        building = fiveFloorBuilding;

                        break;

                    default:

                        building = oneFloorBuilding;

                        break;

                }

                GameObject instantiatedBuilding = Instantiate<GameObject>(building, new Vector3(posX, 50, posY), Quaternion.identity);

                instantiatedBuilding.transform.parent = baseGround.transform;

                Color color;

                UnityEngine.ColorUtility.TryParseHtmlString(\_buildingsDict[key]["Color"], out color);

                Debug.Log(\_buildingsDict[key]["Color"]);

                Debug.Log(color.ToHexString());

                Material mat = defaultMaterial;

                mat.color = color;

                instantiatedBuilding.GetComponentInChildren<Renderer>().material = mat;

                Debug.Log(instantiatedBuilding.GetComponentInChildren<Renderer>().material.color.ToHexString());

                Debug.Log("PosX was: " + posX);

                Debug.Log("PosY was: " + posY);

                posX = ShiftNumber(posX, imageSizeX, 0, 0.5f, -0.5f);

                posY = ShiftNumber(posY, imageSizeY, 0, 0.5f, -0.5f);

                Debug.Log("PosX is: " + posX);

                Debug.Log("PosY is: " + posY);

                instantiatedBuilding.transform.localPosition = new Vector3(posX, 0, -posY);

            }

        }

    }

    private float ShiftNumber(float num, float oldMax, float oldMin, float newMax, float newMin)

    {

        float OldRange = (oldMax - oldMin);

        float NewRange = (newMax - newMin);

        return (((num - oldMin) \* NewRange) / OldRange) + newMin;

    }

}