AALBORG UNIVERSITY
STUDENT REPORT

ED2 – P2 – ED17-B306a

Line Follower Automated Guided Vehicle for Medical Institutions

Students:

Matei Caciuleanu
Sergio Hernandez
Jegors Kirilovskis
Fikrican Ozgur
Devrat Singh

Supervisors:

D. M. Akbar Hussain
Torben Rosenørn

May 29, 2018

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Line Follower
Automated Guided Vehicle
for Medical Institutions

Theme:
Analog Instrumentation

Project Period:
Spring Semester 2018

Project Group:
ED17-B306a

Group Member(s):
Matei Caciuleanu
Sergio Hernandez
Jegors Kirilovskis
Fikrican Ozgur
Devrat Singh

Supervisor(s):
D. M. Akbar Hussain
Torben Rosenørn

Copies: 1

Number of Pages: 74

Date of completion:
May 29, 2018

Abstract:

Inside hospitals, the task of transporting goods such as pharmaceuticals, surgical equipment, food, beds, etc., from one location to another is crucial and time-consuming. It influences all the activities transpiring within the hospital and in many places, it is done manually by the hospital staff, which is wastage of human resources for such a repetitive task and has prospect for improvement. Thus, the report focuses on the ways of increasing the efficiency of medical logistics inside hospitals by automating the process. We developed a proof of concept for an automated guided vehicle based on line following which uses a network of black lines and RFID tags to go from one place to another. It can accept destination input, detect obstacles, utilize inductive charging and has an anti-theft system. Furthermore, it is capable of finding the shortest path to a destination and a monitoring system is also implemented for tracking its movement. A series of tests were conducted for the assessment of the robot and from them we could say that the vehicle is efficient in taking input from the user, obstacle detection and safety system but there is still need for improvement in scanning of RFID tags, which affected the process of shortest way detection and the infrared sensors used were sometimes unreliable. Even after considering the drawbacks, our proof of concept robot was able to traverse between two points many times.

Table of Contents

Table of Figures

Table of Tables

Preface

The project entitled "Line Follower Automated Guided Vehicle for Medical Institutions" was written by five students from the Electronics and Computer Engineering Bachelor's program at Aalborg University Esbjerg, for the P2 project in the second semester. From hereby on, every mention of "we" refers to the five co-authors listed below.

The completion of this project would not have been possible without the contribution of our supervisors, D. M. Akbar Hussain and Torben Rosenørn, who directed their efforts towards guiding us to the right path. We would like to express our gratitude for the time they took for providing their support and relevant advice.

We would also like to thank Henry Enevoldsen, Assistant Engineer at Aalborg University Esbjerg, who showed patience and understanding at the times we asked for his help with the equipment used in the project. Many components have been taken from the Electronics Laboratory of the university and we considered appropriate to specify that in the report each time that was the case.

Aalborg University, May 29, 2018

[illegible]

Matei Caciuleanu
<ccaciu17@student.aau.dk>

[illegible]

Sergio Hernandez
<shern17@student.aau.dk>

[illegible]

Jegors Kirilovskis
<jkiril17@student.aau.dk>

[illegible]

Fikrican Ozgur
<fyozgy17@student.aau.dk>

Singh

Devrat Singh
<dsingh17@student.aau.dk>

1. Introduction

Hospitals around the globe are challenged by the increased share of elderly citizens and chronically ill patients alongside expensive treatment costs. To satisfy this increasing demand for healthcare, there is a need to increase the efficiency and productivity of hospitals for increasing the patient intake rate without raising the costs for treatment dramatically [1]. One of the elements of hospitals that exhibit great prospect for improvement is medical logistics. It is the logistics of pharmaceuticals, medical supplies, food, equipment and other services that are required by the doctor or other staff for performing their tasks.

Today, healthcare sectors around the globe are working on improving the logistics in hospitals by automating the process. Denmark has already come far in the matter of optimizing logistics in hospitals through utilization of just-in-time solution which has led to a 30% increase in hospital productivity since 2003. Furthermore, many hospitals in Denmark are in the course of employing technologies like pneumatic tubes for transporting blood samples, service logistics solutions and other automation and IT technologies [1]. The automation of these basic and repetitive tasks can make the whole process of medical logistics within hospitals more effective and productive. It can streamline the traffic flow of material inside hospitals, control costs, reduce workload and the human resources involved within the sector can be transferred to other departments or patient related tasks.

Therefore, the scope of this project focuses on developing methods and models for improving the goods delivery system inside hospitals

1.1. Initiating Problem

According to Healthcare Denmark, the country is actively improving the hospital logistics using dedicated apparatus [1]. Nonetheless, we are wondering what enhancement we can bring to the current system and we aim to find out how electronics can be utilized to develop a piece of machinery that could transport items inside a hospital. The goal is to reach a solution whose functionality does not imply the alteration of the edifice. Therefore, the following question initializes the problem:

How can we develop a reliable device for optimizing transport within a Danish medical institution without changing the structure of the building?

2. Problem Analysis

2.1. Six W's Diagram

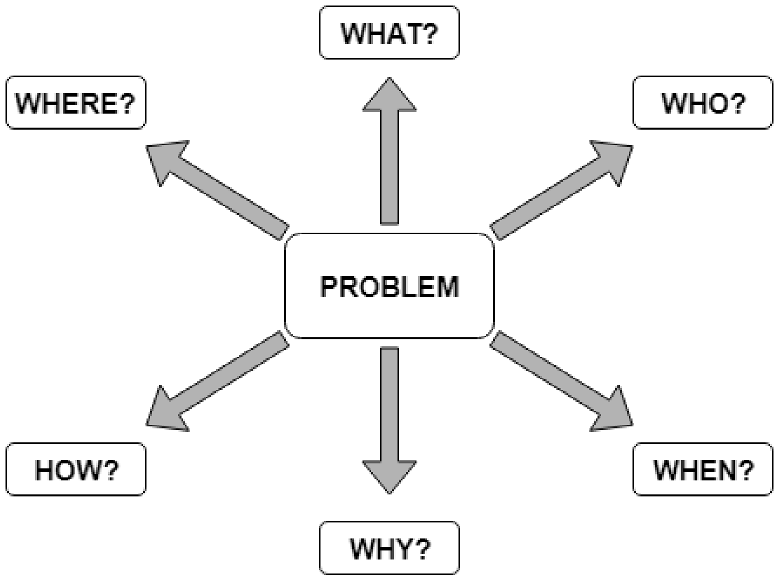


Fig. 1: 6 W's Diagram

What is the problem?

The project group attempts to create a device for delivering any kind of necessary goods (food, samples, medicine etc.) in order to improve and simplify hospital employees' workflow and optimize the delivery system inside hospitals. To achieve that, the device is expected to work automatically, in other words, without human assistance.

Who is affected by the problem?

Logically, the most affected people are the hospital employees whose duties will be performed by the device, i.e., workers whose job directly relates to logistics within the hospital.

Where does the problem occur?

The problem occurs in any Danish medical institution where the task of transporting goods is still carried out using human efforts.

When does the problem occur?

The problem occurs anytime repetitive transportation tasks are necessary and human resources are employed for that purpose. Even though some parts of logistics such as delivery of food and medicine have their own certain schedule, delivery of other goods may be needed unpredictably.

How does the problem occur?

Certain hospital employees should take some requested goods (food, documents, samples, medicine etc.) from one location and transport them to another. Furthermore, non-automated logistic systems are time-consuming, and they require human effort in order to move large quantity of materials.

Table 1 shows frequently transported goods inside Bispebjerg Hospital, as well as the number of employees involved in their delivery along with the number of hours spent on it [2]. As shown, there is a total time of almost 600 hours every week consumed for repetitive transportation tasks. Therefore, an important amount of resources can be saved by implementing an automated solution.

Table 1: Information on Medical Logistics in Bispebjerg Hospital

Transported Items	*Employees Involved*	*Time taken (Hours/Week)*
Medicine	2	50,4
Mail	4	107,1
Food	5	98
Clothes	4	130
Waste	4	195,8

Why does the problem occur?

In developed countries like Denmark the elderly citizen population is increasing. That calls for an improvement in the workflow efficiency. Medical logistics inside hospitals is a field where there is clear room for optimization. Nowadays transportation automation is popular and is being implemented in a variety of places, including hospitals.

2.2. Stakeholder Analysis

Interested Parties

The most affected part of the society, as already mentioned above, are the hospital employees whose work is related to the delivery of items within the institution. Since the automatic delivery device would perform part of the job, the hospital would benefit from more human resources that could be directed towards different tasks rather than good delivery. These people could focus on more sensible areas that require higher competences, making the overall service better for the patients.

The workload of nurses is not significantly affected by the introduction of automation in the transportation sector. Nonetheless, their interaction with the technology would be necessary, so they are also interested parties. Although doctors are part of the hospital's working force, logistic systems inside the hospital do not affect them as they are not directly responsible for supplying goods to the different rooms.

The group considers patients as an interested party as well. They will not notice radical changes in the logistics because the main idea is the same: delivering goods to rooms, including wards. However, the automation of the delivery system may affect their opinion about the quality of the service. The reduction of the staff's tasks might result in a better attention and therefore improve the patient's experience.

Actors

In this section of stakeholders, the group describes the parties that have the power to affect the technology or may be affected by it. The main actors are our supervisors, Akbar Hussain and Torben Rosenørn. The group members have to discuss their project with their supervisors frequently as they are working on it in order to receive their supervisors' opinions. The supervisors may come up with several advices as well as alterations for the project. The group members have the right to change their project in the direction that their supervisor pointed or they may choose to continue with their initial plans. However, the group members are encouraged to listen to their supervisors.

Aalborg University also has the power to affect the project, therefore it is considered as a separate actor. All the materials the group is using are provided by the university. Materials that were not available in the facilities of the university were bought using the funding the university approved for the group. This fact also means that, if the university does not approve the group's funding application for any reason, the group is highly affected by this decision and they may need to consider other components.

Managers of Danish medical institutions can be considered actors as well, since they are in charge of the financial resources of the hospital and may choose to pay for the developed technology. The technology highly depends on their decisions of whether to implement the device in their hospitals or not.

Another type of actors are the developers. The developers of the technology are five students, members of the project group ED17-B306a, who are currently on the second semester of the Electronics and Computer Engineering Bachelor's program at Aalborg University and working on the P2 project in order to solve the aforementioned initiating problem.

Table 2 lists all the aforementioned stakeholders.

Table 2: Stakeholder Table

Stakeholder	*Type*	*Role*
Transportation Personnel	Interested Parties	End Users
Nurses	Interested Parties	End Users
Patients	Interested Parties	Beneficiaries
Supervisors	Actors	Supervisors
University	Actors	Material Suppliers
Hospital Managers	Actors	Clients
ED17-B306a	Actors	Developers

2.3. Already-Existing Solutions

For more than a decade, Denmark is investing decisively in a new and modernized hospital infrastructure in order to provide better services and to maintain a sustainable healthcare system for the future. Just-in-time solutions are the key for fast treatment, giving personnel more time to focus on their primary task by removing the need for searching for equipment, colleagues or waiting for lab results. Below are demonstrated some already-existing hospital logistic solutions in Denmark for different hospital service areas. The reference for the information given in this chapter is taken from a white paper by Healthcare Denmark [1], unless another reference is given.

Patient Movements

Movement of patients between wards and treatment/operation rooms are handled through a service logistics solution (SLS) at Aalborg University Hospital (AUH). The entire staff uses the SLS at AUH on a daily basis to organize and register patient movements. SLS makes it possible for service personnel to have simple and mobile access to task lists, which assures rapid and well-timed patient movements and equipment throughout the hospital. In addition, the person who demanded the patient’s movement can pursue the status of the order and arrange the next steps since they know the exact time the patient will be there. SLS is also aware of whereabouts of all service personnel in the hospital with the use of real-time location tracking. Furthermore, it gathers significant information regarding tasks and activities. The data may be used to further optimize resource allocation and shifts. The establishment of SLS in AUH has resulted in almost 20% efficiency increase. Upon positive benefits of the service, AUH plans to broaden the use of it in areas where real-time location tracking and optimized coordination can speed up the process as well as efficiency. For example, the same technology can be potentially used to show the status of all equipment in the hospital. Thus, delays due to missing equipment or unready beds can be avoided, resulting in energy savings and resources for the hospital.

Blood Sample Delivery

In 2014, Hospital of Southern Jutland was founded by merging three small hospitals. As a consequence of high number of new patients in the hospital, faster turn-around-times (TAT) on blood sample results was needed. TAT (Turnaround Time) describes the rapidness of sample analysis in the laboratories. ToTAT (Total TAT) is a measure of the whole blood sample cycle, starting with the request of a blood sample till results are available. Nowadays, many hospitals employ the TAT approach and this process can take up to 6 hours. The ultimate purpose at Hospital of Southern Jutland was to lower the response time to maximum one hour. The problem was solved by implementing an automatic system that transports blood samples from various departments to the laboratory. The pneumatics based Tempus600 transport system is now delivering blood samples to the laboratory fully automated, IT-supported and monitored to assure that they stick with a maximum response time of one hour as decided before. This process also guarantees that samples are treated evenly, eliminating manual handling. *Previously, the staff walked from point A to B with the samples, and the process was characterized by manual handling. The dedicated transport system has resulted in a reduced amount of manual repetitive tasks and staff can stay on the department floors taking care of the patients instead,* says Susan Boye Nørregaard, Head of the Biochemistry and Immunological Department in Aabenraa.

Chemotherapy Distribution

Two years ago, a new way of delivering chemo therapies at the Hospital of Southern Jutland in Sønderborg was needed upon the relocation of hospital’s Oncology Department away from laboratory. Before the relocation, delivery times for chemotherapies took around 15 seconds since the Oncology Department was situated right above the laboratory, being in direct connection. However, with the new location of the Oncology Department, this time was extended up to 10 minutes, resulting in increased transport needs. In order to overcome this problem, hospital decided to use a MiR100 based Automated Guided Vehicle (AGV) for transportation which would also be able to navigate securely throughout the complex hospital environment. In simple terms, an AVG is an unmanned mobile robot that benefits from some sort of markers on the ground for self-driving and at the same time utilizes a navigation method to find its way. Therefore, it contains many different types of sensors on it [3].

Medicine and Food Delivery in Spain

In Spain, robots are used daily for a large variety of applications. In the Oviedo University Hospital (in Oviedo, a small city in northern Spain) a fleet of 12 isotherm robots deliver food and some medicines to the patient-rooms' doors, so the nurses can dispense them easily [4]. This is achieved by installing RFID tags under the floor that are read by the robot so it can reach certain destination successfully. The system is equipped with an independent network of lifts that can only be used by the robots so they do not block the ordinary flow of the building [5]. Although the system performs the desired task, the hospital staff has experienced some work accidents such as burns due to their height (1.90m) that makes easy to spill the content of the food trays placed on the upper part of the robot [6]. In Vigo, another Spanish city, they have developed an automatic medicine dispenser for the hospital's pharmacy, that has proven to improve the medicine delivery times and the staff's work conditions, as no worker has been fired due to its implementation [7].

Innovative Future Hospital Logistics for Denmark

Based on our research on logistics in Danish hospitals, we have seen that there already exist some advanced technologies, as demonstrated above, that are currently being used to automate some logistic processes. Denmark is currently in the act of building new highly-advanced hospitals around the country, taking into account the future role of hospitals. It is clear that a well-working interior hospital logistics plays a significant role in terms of productivity and efficiency. Therefore, it can be stated that more just-in-time logistics technologies will be needed very soon.

Below are demonstrated some ideas we considered as to how interior hospital logistics can be provided in Denmark.

2.4. Possible Solutions

In this section of the "Problem Analysis" we try to identify potential solutions to the "Initiating Problem" of the project. We can broadly classify these solutions by whether they entail any modifications in the hospital structure or not.

Pipe Network

If, supposedly, some sort of network of pipes that would connect all the rooms inside a hospital existed, medicines and other relatively small items could be transferred through the pipes. For example, a person standing in the pharmacy or medicine storage room could take requests from medical personnel and introduce the needed medicine in a pipe, which could then convey it to the desired location. This resembles the pneumatic tube technology used by the Tempus600 Vita in blood sample transport. However, no evidence of the concept being applied in the transport of other items like medicine or food was found. Placing the medicine inside the pipe could be more efficiently done using robotic arms. There are clear disadvantages involved in using this system. Apart from the infrastructure the hospital would need to invest in, is hard to believe that something like food could ever be delivered through a pipe without unwanted consequences. Other bigger items, such as packages, would not be suitable as well unless very wide tubes with very high suction capabilities were utilized.

Storage Rooms Inside Walls

Another possibility would be to completely change the storage method by building, on each floor, big deposits equipped with conveyors that transport goods to multiple designated "checkpoints" - locked containers - located across a specific floor. Goods could be temporarily stored there, before an authorized person would pick them up. This system would imply construction of small deposits in the entire building and connecting all of them to the floor's deposit. Moreover, it would require a fairly small distance between checkpoints, so that medical staff could save precious time when materials are needed. A far more challenging problem would be posed by the conveyor system itself, for which additional space should be allocated, possibly behind walls. A similar, still much more well-thought concept will most likely be implemented in The New Odense University Hospital by 2025. There will be a very large centralized storage facility to serve the entire hospital. The temporary, drawer-like storage spaces will be integrated in the building's walls and the goods will be transported "behind the scenes" using a very complex system of lifts, conveyor belts and pneumatic tubes [1].

Developing solutions such as those mentioned above is out of our reach and we did not devote too much time to research the functionality and implications of these technologies. The approach we would like to take in this project is obviously directed towards the field of electronics and preferably much less towards the civil engineering aspects. Let us now shift our attention to those solutions that could, at least theoretically, be implemented without altering the physical structure of the medical facility.

Drones

Drones are very popular nowadays and they have been already tested by restaurants and retailers in delivering food or other products [8]. While drone delivery certainly has a bright future ahead, one might wonder why drones are not utilized for medical purposes. It should come as no surprise to learn that in reality they were. There are concrete examples of drone flights carried out for medical-related transports. Most of the time, the vehicles transport rare drugs, vaccines or blood cells to remote rural areas [9]. In March, 2017, in Lugano, Switzerland, the first hospital-to-hospital sample exchange using drones in an urban area has been registered and it is very likely this will become an unexceptional event in the near future [10]. By extrapolation, this type of delivery may eventually be a commonplace not only outside, but inside medical facilities as well. Drones do not require the construction of any pathway and could be programmed to transport items from one room to another. On a closer look, the reasons become apparent. The environment will force the device to fly to a relatively small height compared to the floor, creating a risk of accidents, so one would have to live with the threat of getting hurt by spinning drone propellers or even falling drones. Another drawback is represented by the noise a drone causes while flying. A 2017 NASA study showed that people tend to find cars less annoying than drones, in terms of sound [11][12]. Moreover, long exposure to noise pollution is linked to sleep problems and heart disease [11]. Another downside would be that not all drones are capable of lifting heavy objects, so one would need to utilize either very big drones or many small ones. These aspects constitute impediments to the use drones as a solution to our problem.

Automated Guided Vehicles

Automated Guided Vehicles are currently used in healthcare systems, as described in the "Already-Existing Solutions" section. They are a very efficient solution for material handling [13] and they are well suited for product transportation within hospitals (e.g.: medicine, food, samples). In fact, it was found that they bring a high efficiency in terms of transportation, because many items can be placed in a single cargo unit [2]. There are examples of such devices that can carry up to 500 kg [14]. In the last twenty years the interest in AGVs for healthcare has grown significantly, but they are still not a commonplace in hospitals, since the existing technology is still under constant development. An advantage of these service robots is that the degree of alteration they cause to a facility can be regarded as being minimal [2]. One of the easiest to develop, most common and low-priced navigation technologies consists of following lines – pieces of colored tape – placed on the floor [3]. If designed properly in terms of size and power consumption, they do not block doors or corridors, waste too much electricity or cause obnoxious noise. They can be programmed to deliver items between deposits and wards once a request is sent by a member of the medical personnel. Alternatively, they can be set to operate on given routes, stopping at each checkpoint and wait for an authorized person to take the necessary goods and start the device again, allowing it to continue the route. They can be very well used or for moving heavy materials as well, even during night time. There is clear evidence that these vehicles can adapt to the existing logistics in hospitals [2].

From the discussion of the possible solutions, we can infer that an AGV represents the best method of increasing the efficiency of transportation inside large health facilities. Therefore, the following sections of the “Problem Analysis” will focus solely on relevant aspects regarding these vehicles.

2.5. Legislation

Any product that is intended to be implemented in a professional environment should satisfy the requirements set by the concerned standard setting body. These standards are placed to ensure the proper functioning of the product and for the health and safety of people that come in contact with it. For the use of AGVs inside hospitals the following laws and regulation should be taken into account.

Noise Level Inside Hospitals

According to the guidelines by World health organization, the noise level inside hospitals should not exceed 35 dB in areas in which patients are treated and observed. Thereby the machines like AGVs that are used inside hospitals should maintain the sound level so that any noise produced by its functioning doesn’t affect the health of the patients [15].

Low Voltage Directive

It is issued by the European commission and convers general safety regulations for electrical equipment. The directive applies to electrical equipment that are designed for use with a voltage rating between 50 and 1000 V for alternating current and between 75 to 1500 V for direct current. There are general guidelines stated in the directive that ensure the safe operation of a device such as proper display of information on the device about the assembly and manufacturer as well as testing and investigation (if necessary) of the electrical equipment, so that it doesn’t pose any danger to the user/s or to the surrounding environment. The AGV runs on an electric battery and uses the standard 230 V outlets for charging, accordingly it must adhere to the low voltage directive [16].

Machinery Directive

It is issued by European commission to regulate machinery and parts of the machinery which are intended for either commercial or industrial use. It has objectives to ensure a high level of safety and protection for machinery users and exposed persons. According to the definition specified by the machinery directive “*an assembly, fitted with or intended to be fitted with a drive system other* *than directly applied human or animal effort, consisting of linked parts or components, at least one of which moves, and which are joined together for a specific application*" the AGV lies within the scope directive and must satisfy the standards and regulations [17].

2.6. Risk and Safety

Our technology carries some risks with itself. Without a solution to them, we cannot provide the security of the vehicle, its content and most importantly, its surrounding to a satisfactory extent. Therefore, below we examined those risks and explained how we could solve them.

Collision

We cannot forget about the inconvenience of a small device moving autonomously around a hospital. If the robot is not correctly programmed it can entail risks for both patients and staff, causing crashes and even people falling to the floor. Therefore, it is very crucial that the AGV is responsive to its surrounding in order to prevent unwanted collisions. It should stop as soon as it detects an object or a human being on its way within a predetermined range. The range of the sensors must be enough, so it prevents accidents well in advance. Also, a minimum distance to the room doors must be kept so nobody hits the robot with the door accidentally. There are several technologies we can use for this. GPS sensors might be used to detect fixed objects on the way. Imminent collisions can be avoided using image recognition. LIDAR is definitely an option to measure distance to any object in the environment. Ultrasonic sensors are also applicable. In addition, the robot should also be provided with very visible colors and lights so as nobody steps on it. In case the system stops working or breaks because of human action or by itself, some signal should be delivered for its immediate fixing or replacement, as having such a device in the middle of a hospital is far from convenient and the staff might think it is waiting for someone to take its content.

Access Control

Someone might be interested in what the AGV carries inside for any reason. To prevent unwanted access to the content or to any feature of the technology, we should have security control so that only authorized people will be able to interact with the AGV. This can be done by implementing technologies such as fingerprint detection, face recognition, RFID tag, pin code and so on.

Theft

Even though access control provides security of the content to some extent, we should also consider theft of the vehicle since the robot would move by itself with no supervision. It is vital that it does not fall in the wrong hands, as it can contain certain substances like morphine that are dangerous for health and can cause a serious addiction. A reliable alarm system must be implied so that when the robot is lifted or dragged away from its work area, the authorized people will be informed immediately. This signifies the necessity of a monitoring system. As it is clear, we cannot prevent the theft but we can provide fast response to it by informing necessary authorities immediately.

Lost Communication with the Monitoring System

Monitoring the vehicle is very significant since it provides information about the vehicle all the time. In the case of lost communication, the robot should remain in its position and wait for help. Lost communication should be detected by the monitoring system and necessary people should be informed.

Sensor Malfunction

There is always a possibility that the hardware will malfunction. Since the AGV will be guided depending on sensor readings, even a small error might alter our robot's behavior. Unfortunately, there is not much we can do apart from checking sensor readings periodically. However, the chances of it happening are quite slim.

Battery

Overheating of batteries for any reason may have dangerous consequences as it may lead to an explosion. When this is the case, it is crucial to monitor battery temperature as well. A temperature sensor, which send information back to charging station, should ensure that the temperature conditions inside the robots are ideal for both battery lifespan and medicine storage. In order to charge the batteries, there should be a space in the hospital hosting charging stations where robots would go when battery reaches certain level of charge. The charging stations must be safe so if somebody touches them by accident, they do not get an electrical shock. Even after taking these precautions, it is important to mention that batteries are compact energy storages, and therefore susceptible to explosion or toxic gas emission, that may result in very undesired consequences in an environment such as a hospital.

Motors

How much power is given to the motors individually and its direction will be determined by sensor readings. We will not really have a control over it. However, we should make sure that the motors will not be exposed to any sudden power and direction changes since this can damage them. This can be ensured with a proper code flow.

Robot Getting Out of Path

If the robot moves out of the line due to a malfunction, it must be stopped since it will not be able to navigate itself. An uncontrolled moving vehicle in a hospital may cause significant harm to environment and even to people considering its weight and dimensions. Sensor readings may be used to detect if the robot is out of line or not. In case of such event, it also important to inform authorized personnel immediately to get the robot working as soon as possible. Otherwise, significant delays may be experienced in the work areas of the robot, and, if such scenario is not considered before and not enough precautions are taken, the available personnel may not be able to compensate the unavailability of the robot.

Whereabouts of the Robot

Not knowing where the robot is in the hospital when needed can be considered as a risk if fast human interaction is needed. Therefore, the monitoring system should also be fed with proper information about whereabouts of the robot. RFID tags may be used for this purpose as a secondary task.

All the risks listed above may cause the robot to malfunction and put people's health at risk. For this reason, it is crucial that we implement the aforementioned solutions. Having said that, we can conclude the robot should have: obstacle detection system, access control, theft protection and monitoring system (localization, battery temperature, theft).

2.7. Problem Statement

During the initial stage of the project, we analyzed in detail the aspect of logistics inside Danish hospitals and focused our attention on identifying solutions for increasing the efficiency of transportation within the institutions. We discussed the methods currently used for handling this administrative sector and we also suggested potential alternatives. After weighing the advantages and drawbacks of our findings, we decided to build an Automated Guided Vehicle that could convey items from one location to another inside the hospital. The AGV itself would be a mobile platform, to which containers of different weights could be attached. The purpose of the robot is clear: to increase the efficiency of delivery inside hospitals.

With all that being said, the problem that we are aiming to solve through our project can be formulated in a more specific manner that actually includes the solution we decided to develop. This constitutes the "Problem Statement" of the project and takes the form of the following question:

How can we build a microcontroller-based line follower Automated Guided Vehicle?

2.8. Project Specifications

The “Problem Analysis” not only clarified many aspects related to the topic of the project, but it also provided a basis for extracting the guidelines needed for starting the practical part, i.e., the development of the AGV. A reliable robot that would represent the solution to the problem of increasing the efficiency of transportation inside hospitals should, according to the conducted analysis, possess the following features:

- High material resistance: in a real scenario, the vehicle should be able to carry heavyweight materials, perhaps cargos reaching even 500 kg, so a highly resistant material should be used in the construction of the drive base.
- Adequate dimensions: the dimensions should be high enough to properly distribute the pressure of the cargo; the vehicle should be a mobile drive base on which different containers could be attached rather than a moving container, with a fixed storage capacity.
- Line-following: the AGV must follow the markers mounted on the floor; it needs a fixed path, not to impede the circulation of pedestrians on the hallways; the line should be thoughtfully placed as well to prevent that.
- Ability to find the shortest way to the destination: it is reasonable to assert that there would be cases in which more possibilities to reach a certain destination would exist; to save energy and keep the efficiency as high as possible, the vehicle should be able to find the shortest way to the indicated location.
- Moderate speed: the movement of the vehicle should not pose a threat to the health and safety of the people in the hospital, but at the same time it should be able to quickly deliver the items, for efficiency reasons.
- Obstacle avoiding: the robot must be equipped with the adequate means of detecting obstacles placed on its path or in its vicinity and take action accordingly to avoid any accidents.
- Position identification: the AGV must have the necessary hardware that would allow it to keep track of its position at any time and identify the destination.
- Large battery autonomy and possibility of recharge: the vehicle may have to work across long distances, 24/7, so the battery autonomy is a crucial aspect; a charging facility should exist in the hospital and the vehicle should be equipped with the hardware that allows recharge.
- Ability to monitor battery temperature: during operation, the temperature of the power source increases and excessive heat could damage the device and it may even lead to the explosion of the battery, thus the vehicle should be equipped with temperature sensors, for the safety of the patients and hospital employees.
- Theft protection: in case of the vehicle being stolen, there should be some sort of alarm informing the hospital personnel of the event.

- User interface that only allows the interaction with hospital personnel: the hospital employees using the device should be able to interact with it to indicate the destination and the speed; at the same time, there should be a safety measure against the access of unauthorized people, who are not allowed to decide upon the activity of the robot.
- Feedback system: it should be possible to receive data wirelessly from the vehicle, such as position, battery level and temperature or theft alert; a computer inside the hospital where this information is received should be always on and perhaps a designated person should supervise the activity.

If an AGV had to be utilized inside a medical institution, the preceding list of specifications would apply. Nevertheless, because of the group's lack of experience regarding the development of robots or any other type of electromechanical system, a prototype would not be possible to build in the time allocated for the project. As an alternative, the practical part would consist of a proof of concept, to demonstrate that the solution is feasible. The characteristics of the proof of concept would be different than the ones of a real product. The scale will not be 1:1 and a light material would be used for constructing the drive base, otherwise the vehicle would become too heavy and hard to handle. The speed the proof of concept will operate at will be low and the battery will not provide high autonomy.

2.9. Project Delimitations

Denmark

For this project, we have decided to focus on the Danish healthcare system. The main reason behind this decision is the heterogeneity of the legal frames and quality standards across Europe in this matter. It is difficult to adapt the system to every single legislation in the continent, as there is an obvious gap in quality of service between the wealthy countries and the modest ones. The project, although useful, might not be viable for the latter, because the investment needed is considerable and it is probably more necessary in other fields such as personnel training or basic infrastructure improving. In Denmark, on the other hand, the investment in healthcare is very high in relation to the country's size, and the medical infrastructure is so advanced at this point that small improvements in logistics like the one our system aims to provide are more appreciated.

Lack of a Storage Box

As we do not know the concrete characteristics of the goods we are delivering in every case, and the addition of a storage box makes more difficult to keep the stability of the robot, we decided not to include a storage box in our proof of concept. Having such a container in the robot would mean to have a climatization system running constantly and ensuring that temperature is always in the desired range. This would imply a much higher cost of the system, as isolating materials and heat pumps come at a price, without mentioning the higher energy requirements that would inevitably lead to a higher battery capacity, with all the environmental and safety risks that that entails. Even if including such a system is an interesting challenge from an engineering perspective, time and funding constrains made its developing too likely to be left unfinished.

Incompatibility with Lifts

Cost efficiency is an important part of our project. That implies avoiding additional infrastructure with a high price tag. For that reason, the robot we aim to design cannot rely on a lift system for themselves, and adapting the normal lifts is not an option either, as a communication protocol between lifts and robot would have to be developed.

2.10. The General Idea

In the next section, the components utilized for building the proof of concept of the AGV are discussed. Table 3 provides the list of components and their functions.

Table 3: Components Used in the Project

Component	*Role*
Arduino Mega	Controlling the System
Lead Battery	Powering the System
DC Motors (2)	Movement
IR sensors (5)	Line Following
RFID Reader	Location Tracking
Ultrasonic Sensors (3)	Obstacle Detection
LCD	User Input
Keypad	
Tilt Sensor	Theft Protection
Buzzer	
Ultrasonic Sensor	
Temperature Sensor	Monitoring the Battery Temperature
Inductive Charging Module	Recharging the Battery Wirelessly

The general idea for the movement of the AGV inside a building is described in the following points and is shown in Fig. 2.

1) It is assumed that there are lines marked on the floor leading to every location the robot is supposed to go.
2) The robot travels to locations based on the input of the user and tracks these lines for guidance, thus it can only travel on top of the line.

3) Each location has an RFID tag assigned to it for unique identification and more tags are placed between two locations, along the marked line, for tracking the movement of the robot.
4) While tracking the robot, its location is shown on a map illustrating the real structure of the building.
5) Each RFID tag is a part of the process for deciding the direction of movement, especially at the intersections where there is more than one option available to proceed further.

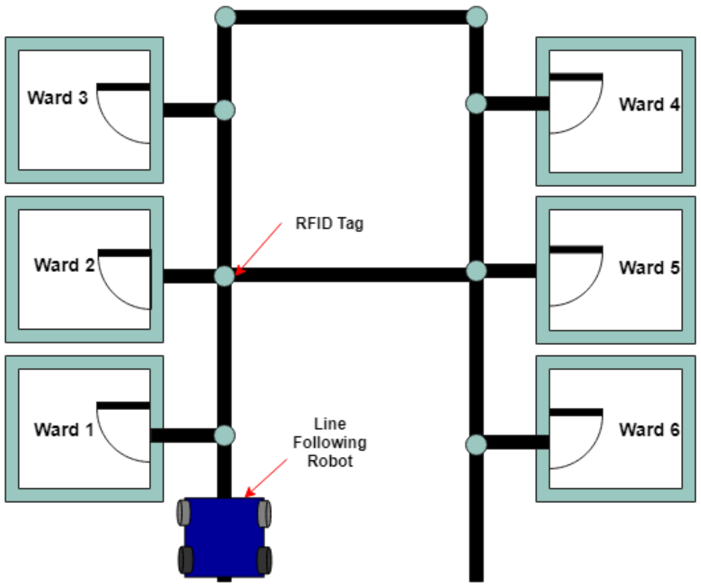


Fig. 2: General Idea for the Proof of Concept

Having just mentioned the general idea for the movement of the robot, now we want to briefly explain how the interaction between the hospital personnel and the robot will be. Fig. 3 illustrates the workflow of the robot.

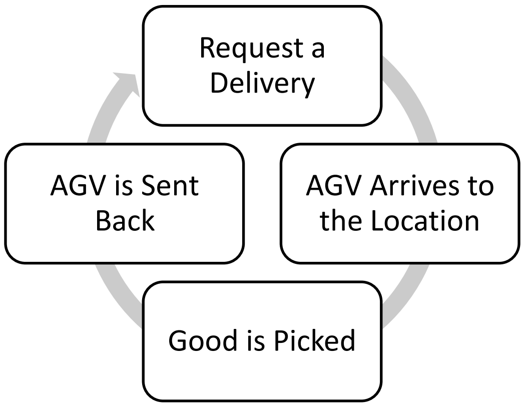


Fig. 3: Workflow for the AGV

The steps are explained below.

1. The authorized personnel requests a delivery to a specific location using the mobile application. They also indicate the good to be delivered.
2. Depending on the requested good, the AGV either directly comes to the selected location or it first picks the necessary good and then arrives to the selected location. For example, if the AGV is loaded with multiple types of medicines and an available type of medicine is requested, the AGV can directly go to the selected location. However, if the personnel requests a medicine that the AGV does not carry at that moment, then it will first go to the pharmacy of the hospital and pick the necessary medicine. The pharmacy personnel will also be informed about the request through the mobile app and they will be ready before the AGV arrives. After they place the medicine inside the AGV and approve it, the AGV goes to selected location.
3. The employee enters the pin code and picks the good. In the event that the AGV arrives to the location before the personnel who requested the delivery, it waits until the good is picked.
4. The member of personnel selects to send the AGV back to charging station. In case that some other personnel requested an available medicine to another location in the hospital, the AGV can directly go to that specific location without visiting the charging station.

Even though this work flow reflects group members' idea for this project, we need to mention that the mobile application was not included within the project scope.

3. Methods and Materials

3.1. Microcontroller

We needed a microprocessor with enough memory to store the code and with enough number of pins for sensors and motors. Initially, one Arduino Due is used for this project as it met our aforementioned desires. However, we had to switch to Arduino Mega later on due to operating voltage level of Due. Due gives and accepts 3.3 V as high voltage which was a problem for us since some of the components we used required 5 V. Mega was able to provide and accept 5V from its I/O pins. On the other hand, as a drawback of Mega, it has a memory as much as half of Due's to store the code. However, it proved to be enough in our case. The specifications for Arduino Mega (Fig. 4) are listed in Table 4 [18],[19],[20].

Table 4: Specifications for Arduino Mega

Attribute	*Value*
Processor	ATmega2560
Operating Voltage	5 V
Input Voltage	7-12 V
CPU Speed	16 MHz
Analog Input Pins	16
Digital I/O Pins	54 (14 provide PWM)
Flash Memory	256 kB
Max. Current Per I/O Pin	40 mA
Max. Current Per 5V – GND Line	200 mA

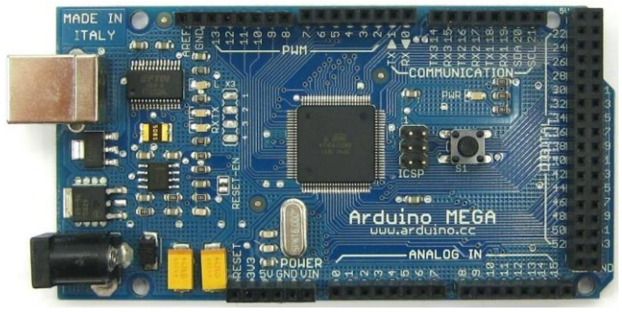


Fig. 4: Arduino Mega

3.2. Radio Frequency Identification (RFID)

RFID data collection technology is mainly used to identify, categorize and track objects. Fundamentally, a basic RFID system consists of two parts: an RFID reader and an RFID tag. The reader may also be called as a transceiver or interrogator, whereas the tag may also be called as a transponder [21].

RFID Tags

In essence, RFID tags comprise two components: an antenna for receiving and transmitting a signal, and an integrated circuit for storage and processing information. RFID tags have non-volatile memory and they are activated when passed through an electromagnetic field generated by the reader. They convert electromagnetic waves they receive into electrical current and power their microchip. There are three types of RFID tags: passive, active and battery-assisted passive. The cheapest option is the passive tag which is powered by radio energy transmitted by the reader. An active tag transmits its credentials periodically and comes with an inbuilt battery which powers the tag. A battery-assistive passive tag also contains an onboard battery. However, these tags are only activated when they are in the presence of an RFID reader [22].

RFID Readers

The purpose of an RFID reader is to handle radio communication between its antenna and tags' antennas. An RFID reader transmits signals to a tag and ask for its data in order to pass its information to outside world. In essence, an RFID reader transmits and collects information. RFID reader antennas are powered from a source. Electrical current coming from the source is converted to electromagnetic waves and radiated by the antenna to be picked up by the RFID tag[23]. The aforementioned electrical current is an alternating current, usually referred as RF (Radio Frequency) current[24]. There are three major frequency ranges for RFID systems [25]:

- Low Frequency (LF) 125 to 135 KHz
- High Frequency (HF) 13.56 MHz
- Ultra-High Frequency (UHF) 860 to 960 MHz

Low frequency RFID systems have a short reading range, slow read speed and low cost. On the other hand, high frequency RFID systems can provide a longer reading range and faster read speed, but the cost is also increased proportionally. In order for a successful communication between the reader and the tag, they need to operate in the same frequency [26].

RFID Systems

Nowadays, RFID systems are all around us. They are very small, require no power and easily attached to anything. They are implemented in various fields such as goods management/tracking, contactless payment, travel documents, security tags, timing, healthcare data, real time location systems and so on [22]. They are in high demand thanks to their reliability, efficiency and accuracy. Here is the diagram showing the basic work flow of an RFID system. This diagram applies to all types of RFID tags [21].

1. Data within the RFID tag's microchip awaits to be read.
2. The RFID reader creates an electromagnetic field.
3. Electromagnetic energy from the reader's antenna is received by the tag's antenna.

4. By using the power from reader’s electromagnetic field or, in the case of active and battery-assistive RFID tags, power from its own inbuilt battery, the tag transmits radio waves back to the reader.
5. The reader receives and decodes the radio wave as meaningful data and sends it to a computer.

At the end, we chose to use passive RFID tags among other options because of the following two reasons. First, we do not need to update stored information in the RFID tags. Our objective is to locate our robot in real time. To do so, the identification number of the tags are more than enough. Second, we are trying to keep the cost of our project as minimum as possible. And passive RFID tags are the cheapest tags in the market.

MFRC522 RFID module is used as the RFID reader for this project as following specification are found suitable to our project [27]. We do not need a long reading range. Operating distance is up to 50mm. RFID tags are placed right under the line that the robot follows. By placing the RFID reader close to ground on the robot, we do not expect to encounter any problems in terms of reading the tags successfully. In addition, the module requires 2.5 to 3.3V supply. We are able to supply this voltage through our microcontroller Arduino Mega. Also, our robot needs to read the RFID tags on the move. Therefore, fast reading speed needed. RC522 RFID module uses high frequency (13.56MHz) and this provides faster read speeds.

The technical specifications for MFRC522 RFID module (Fig. 5) and its illustration can be found in Table 5 [28].

Table 5: Specifications of MFRC522 RFID Module

Attribute	*Value*
Supply Voltage	3.3 V
Current	13 – 26 mA
Read Range	3 – 5 cm
Operating Frequency	13.56 MHz
Supported Host Interfaces	SPI/I^2C/UART
Max. Data Transfer Rate for SPI	10 Mbit/s

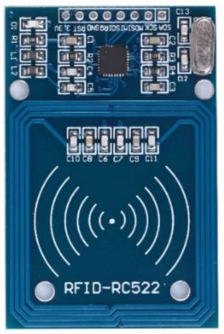


Fig. 5: MFRC522 RFID Module

3.3. Infrared Sensor (IR)

An infrared sensor is a device which emits light rays within infrared spectrum in order to sense certain aspects of its surroundings such as colour, radiation, proximity to an object, motion etc. The infrared sensor module used in the project consists of an emitter-detector pair which are collectively known as opto-coupler. The emitter is an LED which emits infrared rays whose wavelength lies between 0.75 -1000 μm and cannot be seen by the human eye, while the detector is an IR photodiode which is sensitive to the infrared light of the same wavelength as emitted by the LED.

Photodiode

The photodiode used as detector, is a semi-conductor device that can sense optical signals (IR light) within its vicinity. It is a PN junction diode and works on the principle of photogeneration, i.e., the conversion of light energy to electrical energy. Therefore, when the light rays are incident on the photodiode they are converted into electric current and the amount of current produced is directly proportional to the intensity of incident light. When there is no light incident on the diode it can be perceived of having very high resistance and as the intensity of light is increased, the resistance reduces to a low value due to rise in photo-electric current [29].

Contrast Between Black and White Surfaces

Infrared sensors can distinguish between black and white coloured surface as a result of the surface's light reflecting properties. Black coloured surface are capable of absorbing light rays in the visible spectrum and the infrared light of smaller wavelength, depending on the surface material. In the case of a white surface, all the light rays that are incident on it are reflected back. Therefore, when the emitter of the infrared sensor transmits the IR rays on a white surface all the rays are reflected onto the detector (shown in Fig. 6) and thus there is a change in light intensity which in turn leads to the change in output voltage of the sensor. In the case of black surface, the IR rays are absorbed (shown in Fig 7) and hence no change in output. In general, bright colours reflect back more light as compared to darker colours and this property is used by the sensor to distinguish different colours,

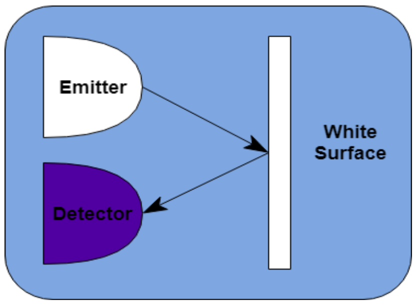


Fig. 6: Light Emitted on White Surface

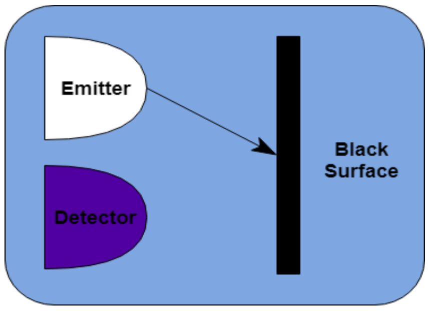


Fig. 7: Light Emitted on Black Surface

Infrared Sensor Circuit

In the project we used the FC-51 infrared sensor module, for line-following, which utilizes the circuit shown in Fig. 8. In the IR sensor circuit, the photo-diode is connected in reverse bias, i.e., the P region of the diode is connected to the ground while the N-region is connected to 5 V power supply. It is connected in series with a 10 KΩ pull-down resistor R2 which ensures that the output is defaulted to 0 V when there is no incident light. The photodiode and R2 are in voltage divider configuration and the voltage drop “V_{IN}” across R2 is used as an input to the inverting terminal of the operational amplifier while the non-inverting terminal is connected to a variable resistor (R3). The operation amplifier is connected to 5V on the positive side and to ground on the negative side and is used as an inverting comparator circuit.

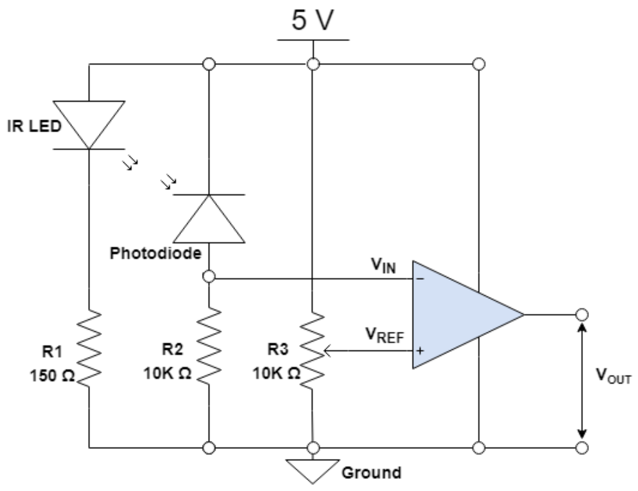


Fig. 8: Infrared Sensor Circuit

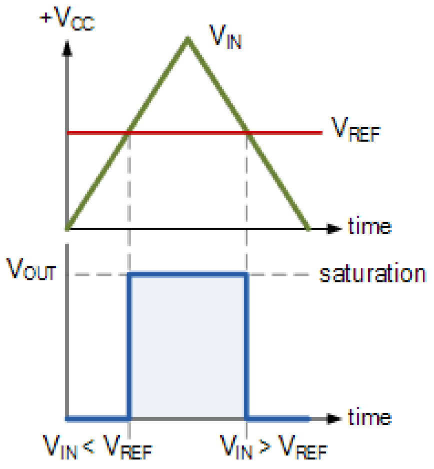


Fig. 9: Comparator Circuit Waveform

The inverting comparator circuit compares two voltages and outputs either “high” or “low” depending on the input. If the voltage on the inverting input “V_{IN}” is lesser than the voltage “V_{REF}” on the non-inverting input, then the op-amp output “V_{OUT}” will rise to a positive saturation equal to the voltage at positive side and if V_{IN} is greater than V_{REF} than V_{OUT} will drop to negative saturation level equal to the voltage at the negative side (waveform shown in Fig. 9) [30].

In the circuit seen in Fig. 8, V_{REF} is the potential drop across the variable resistor R3 which acts as a threshold voltage. Furthermore, V_{IN} is directly proportional to the intensity of light, therefore when the sensor is above a black surface V_{IN} is less than V_{REF} so the output is “High” (5 V) and for white surface V_{IN} is more than V_{REF}, thus output is “Low” (0 V) [31],[32].

The specifications of the FC-51 IR sensor (Fig. 10) can be found in Table 6 [33].

Table 6: Specifications of FC-51

Attribute	*Value*
Supply Voltage	3.3 – 5 V
Current	23 – 43 mA
Range	2 – 30 cm

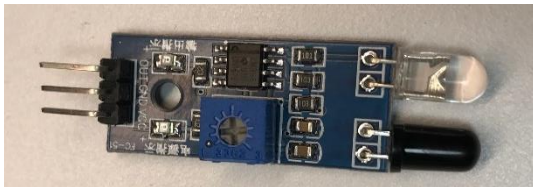


Fig. 10: FC-51 Infrared Sensor

3.4. Keypad

The keypad is an essential component in systems that require an interaction with the user. Even though nowadays touchscreen is widely used for receiving input, physical keypads continue to be sometimes preferred due to their lower price. Moreover, the environment where the device is utilized calls for the use of one option or the other. For example, in a medical institute, equipment may require quite often the interaction with a user wearing gloves. In such a scenario, a keypad overtakes touchscreen in terms of efficiency. This aspect along with the consideration of cost and size argue the choice of using a keypad in the current project.

A small keyboard is nothing more than a matrix of interconnected buttons, very common arrangements being 3 × 3 or 4 × 4. A diagram of the inner structure of a 3 × 3 keypad is shown in Fig. 11.

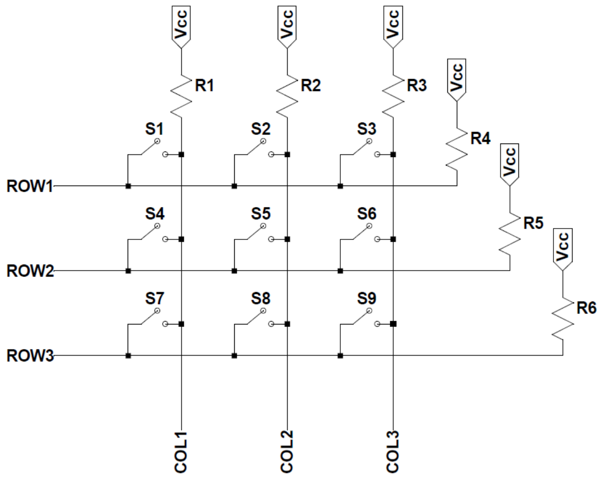


Fig. 11: Keypad Diagram

The working principle is fairly simple. The push-buttons act as momentary tactile switches: when a button is pressed, the switch closes, shorting one column pin to one row pin, and then returns to its previous state. Since there is a direct correlation between the row-column combinations and the buttons, the pressed key can be detected [34],[35].

When interfaced with a microcontroller, the column pins should be set as output pins and the row pins as input pins. Initially, all the row pins are pulled high and the digital value of the column pins is set to high as well. If an individual column pin is set to low and a switch on that column is triggered, the corresponding row pin becomes low. As soon as that happens, the microcontroller is able to identify which key has been pressed. In order to detect any key, each column pin should be set to low, the microcontroller should check for low row pins and then the same column pin should be again set to high. These three instructions must be executed inside a loop [36],[37]. The Arduino IDE provides the downloadable library *Keypad.h*, which allows users to elude the painstaking process of writing the sequence of steps described above, while improving the code readability [38]. The keypad model used in the project is Devlin 16 Key Keypad. The component is shown in Fig. 12 and its specification can be found in Table 7 [39].

Table 7: Specifications of Devlin 16 Key Keypad

Attribute	*Value*
Keypad Layout	4 × 4
Keypad Output	Matrix
Mechanical Life of Individual Keys	10^6 Operations
Supply Voltage	0.5 – 24 V
Current	10 – 20 mA

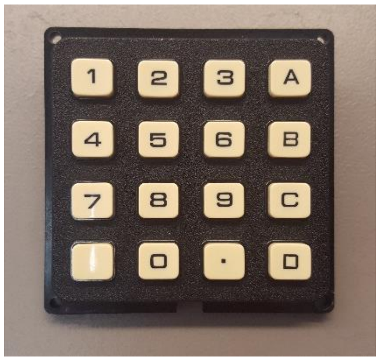


Fig. 12: Devlin 16 Key Keypad

3.5. Liquid Crystal Display (LCD)

One of the most popular ways of creating simple user interfaces for electronic projects is making use of the Liquid Crystal Display technology. The fact that it is inexpensive, power-efficient, compatible with microcontrollers like Arduino and easy to program incentivize one to opt for this type of screen. An LCD display is shown in Fig. 3. When choosing a screen, the number of characters it can display becomes an important criterion. Common combinations are 16 × 2 (two rows with a maximum of sixteen characters each) and 20 × 4 (four rows and at most twenty characters per row).

The LCD used in the project is LCD 2004 IIC/I^{2}C (see Fig. 13). It is a convenient choice for two main reasons: it has four usable lines and allows up to twenty characters to be displayed on which one of them; it comes with a daughterboard used for Inter-Integrated Circuit (I^{2}C) communication. The latter is an aspect of great importance, since it greatly simplifies the circuit needed for connecting an LCD to a microcontroller. The I^{2}C protocol allows communication between multiple devices using only two wires, thus there are just two signals involved, namely SCL (Serial Clock) and SDA (Serial Data). The specifications of the LCD are shown in Table 8 [40].

Table 8: Specification of LCD 2004 IIC/I^{2}C Module

Attribute	*Value*
Size	99 × 60 mm
No. of Characters	20 × 4
Interface	I^{2}C
Supply Voltage	5 V
Backlight Current	200 mA

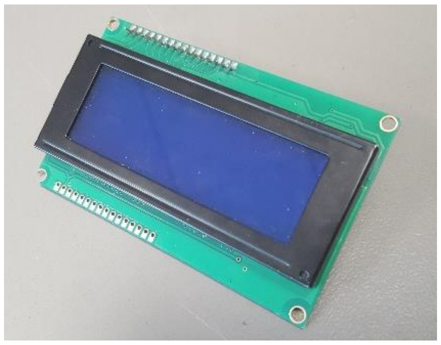


Fig. 13: LCD 2004 IIC/I2C Module

3.6. Wheels

In the mobile robot projects there are three types of wheels that are widely used: standard wheels, orientable (caster) wheels and omni wheels. Vehicles that rely solely on standard wheels are confined to having only two degrees of freedom (forward and reverse). Steering is of course not impossible, but the friction between the standard wheels and the ground during a turn would certainly hinder fast and smooth steering. Adding a caster wheel can compensate for that. Our initial intention was to use two standard wheels as driven wheels and one or two casters to allow smooth steering. However, caster wheels were found to be inappropriate for the setup used in the project. Unlike standard or omni wheels, casters need to be screwed under the drive base and they could not be placed without diminishing the stability of the device. That is how our attention shifted towards omni wheels. They add the possibility of using a third degree of freedom by having small rollers attached along the circumference of the wheel. They are designed such that the axis of each roller is perpendicular to the axis of the main wheel at any given moment [41]. The connection to the drive base can be done in the same fashion as for the standard wheels.

The selected wheels were the following: two 65 mm standard wheels (Fig. 14 (a)) and two 70 mm omni wheels with plastic rollers (Fig. 14 (b)).

(a) Standard

(b) Omni

Fig. 14: Wheels

3.7. DC Motors

A motor is defined as *a machine that supplies motive power for a vehicle or other device with moving parts* [42]. In particular, an electric motor is a device that produces movement by converting electrical energy into mechanical energy [43]. A very popular type of electric motors used in robotics is represented by the brushed DC motors (BDC). They are very common, since they are easy to drive, affordable, and can be effortlessly connected to the circuit: one has to apply a voltage difference to the two terminals of the motor to make it spin [44].

The two BDCs (model JGA25-370) utilized in the project have been taken from the Electronics Laboratory of the university and they had standard wheels and coupling already attached. In fact, these standard wheels were the ones shown in the "Wheels" section. Before making the decision of using the motors however, the group wanted to learn whether they were suitable for the project or not.

Calculations of Torque and Speed

The method for selecting the adequate motors for a mobile robot consists of the theoretical calculation of torque, which describes the amount of force required to spin the wheels, and angular speed, a measure of how fast the motor shaft is spinning [45]. This process requires an analysis of the forces that act upon the robot at any given moment. A body diagram of the vehicle is presented in Fig. 15. Because the robot would move chiefly on flat surfaces within the hospital, only this specific case is considered here, something that simplifies both the diagram and the calculation.

Newton's Second Law of Motion states that the algebraic sum of the forces acting upon a body on a specific axis is equal to the mass of the body multiplied with its acceleration on that axis. We are only interested in the forces acting along the X-axis [45]. By inspection of Fig. 15, we have:

$$\sum F_x = f = ma \quad (1)$$

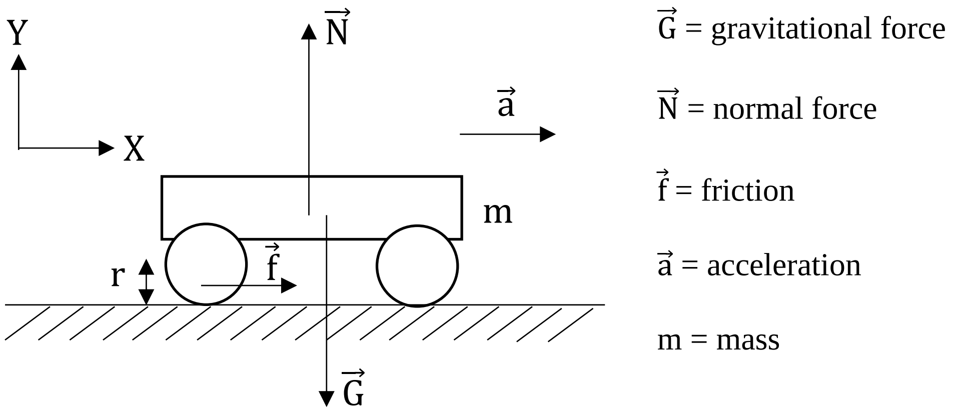


Fig. 15: Body Diagram of the AGV

The friction between the drive wheels and the surface is what causes the wheels to rotate. Torque is defined to be *the quantitative measure of the tendency of a force to cause or change rotational motion* and is equal to the cross product of force and radius [46]. The friction acting upon the wheel is tangent to its circumference, and therefore perpendicular to the motor shaft. Therefore, the torque has the formula:

$$T = fr, \text{ thus } f = \frac{T}{r} \quad (2)$$

Combining (1) and (2) yields the formula for the total torque:

$$\frac{T}{r} = ma, \text{ which means } T = rma$$

The torque of each motor is given by:

$$T_w = \frac{rma}{n}, \text{ where } n \text{ is the number of drive wheels}$$

The formulae are not yet accurate, because every system has an overall efficiency, as a consequence of the efficiency of its individual parts (i.e., motors, gears, wheels, etc.) [45], which needs to be considered. Since no motor has been selected yet, the only thing that can be done is estimate an overall efficiency. It follows that:

$$T_w = \frac{1}{eff}\frac{rma}{n}, \text{ where } eff \text{ denotes the overall efficiency (3)}$$

Given the fact that we are dealing with non-uniform linear motion, the acceleration of the system can be found by the following formula, provided that the robot was initially at rest:

$a = \frac{{v_{max}}^2}{2d}$, where is the maximum v_{max} velocity of the robot and

d is the distance covered until v_{max} is achieved (4)

Finally, the torque on each drive wheel can be found from (3) and (4):

$$T_w = \frac{1}{eff}\frac{rm}{n}\frac{{v_{max}}^2}{2d} \quad (5)$$

The relation in (5) cannot be further simplified, hence all the right-hand side variables had to be estimated according to the technical specifications of the project. We assigned the following values: m = 2.5 kg, v_{max} = 0.2 m/s, d = 0.01 m, eff = 50%. The number of drive wheels and their radius are known: n = 2 and r = 0.0325 m. After performing the calculations, the torque obtained was:

$$T_w = 0.16 \text{ Nm}$$

We can infer that, according to the calculations above, a motor that could provide a torque of around 0.16 Nm was required. The reader should notice however that some aspects are neglected in the described sequence of steps. Air drag, friction between the front wheels and the ground and the thermal-related details are left aside, for the choice of motors would become unnecessarily complicated [45]. The applied procedure should suffice for the proof of concept that represents the scope of the project.

The angular speed (in RPM) can be calculated using the formula [47]:

$$\omega = \frac{60}{2\pi}\frac{v_{max}}{r} = 59 \text{ RPM}$$

The brushed DC motor JGA25-370 has a maximum torque (stall torque) of 0.98 Nm and a maximum speed (no-load speed) of 77 RPM, larger values than the calculated ones. To better see that the motor did represent a viable option indeed, the torque-speed characteristic of the motor was plotted (see Fig. 16). The curve shows that for a value of torque of 0.16 Nm, the motor provides a speed of 64 RPM, which means the AGV would move a bit faster than required [47]. The specification of the JGA25-370 motor (Fig. 17) are shown in Table 9 [48],[49].

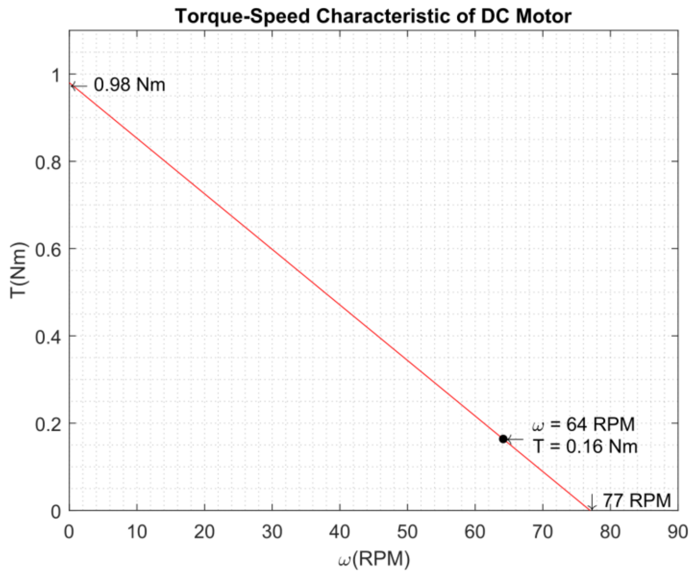


Fig. 16: Torque-Speed Curve of the DC Motor

Table 9: Specification of JGA25-370

Attribute	*Value*
Type	Brushed DC
Stall Torque	0.98 Nm
No-Load Speed	77 RPM
Supply Voltage	3 – 9 V
Current	80 – 900 mA

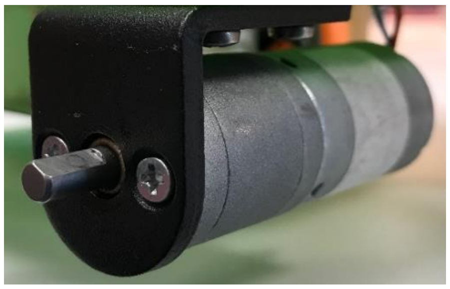


Fig. 17: JGA25-370 Brushed DC Motor

3.8. Motor Driver

One of the most crucial tasks of the microcontroller (ATmega2560 in Arduino Mega) used in the project is to control the motors depending on the input of the sensors. The motors require high voltage and current to operate but the Arduino is only capable of supplying small voltages and small amount of current. In other words, motors cannot run directly through the output of the Arduino. Accordingly, a motor driver is used as an intermediate device between the Arduino and the motors. It acts as a current amplifier and accepts low current signals from the Arduino and converts it to a high current signal which is then used to drive the motors. Furthermore, the motor driver enables us to control the direction of rotation of the motors as well as to apply brakes rather than stopping the motors by giving 0 speed, the brake function when applied stops the motor instantaneously and does not let the motors rotate due to inertia.

Motor Driver Circuit

The circuit of a motor driver consists of an H-bridge which is responsible for controlling the direction and speed of the motor. A basic H-bridge circuit consists of four switching elements with a load (motor) at the center, connected in an H-like configuration. The switching elements in the circuit are MOSFETs and an external power supply is connected to the motor driver which provides the high voltage and current required to drive the motors. The circuit uses different on-off combinations of transistors to control the current flow direction in the motor and by doing so, it controls the direction of rotation of the motor.

The H-bridge can be divided into two sides, ‘A’ and ‘B’, with each side being connected to one of the terminals of the motor. The ‘A’ side includes two FETs, A1 and A2, and the ‘B’ side consists of FETs marked as B1 and B2 ,where A1 and B1 are P-channel and A2 and B2 are N-channel MOSFETs [50].

When A1 and B2 are closed, the left lead of the motor is connected to the power supply and the right lead is connected to the ground. This results in the flow of current through the motor and causes the rotation of the motor in clockwise direction as in Fig. 18. Similarly, when A2 and B1 are closed the right lead of the motor is connected to the power supply and the left lead to the ground, as the voltage supply to terminals of the motor is reversed, the current flowing in the motor is in opposite direction and results in the anticlockwise rotation as Fig 19 illustrates [51] [52].

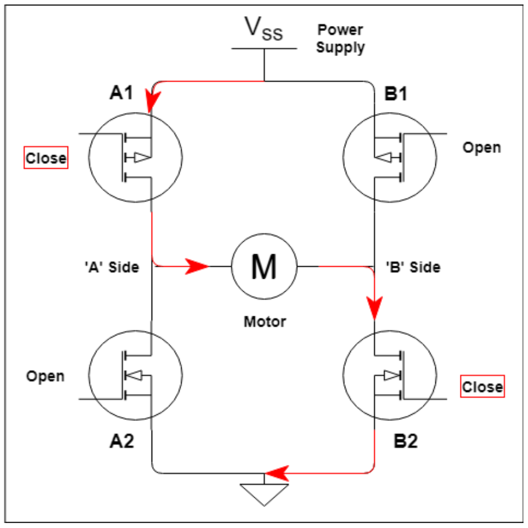


Fig. 18: Current Direction for Clockwise Rotation

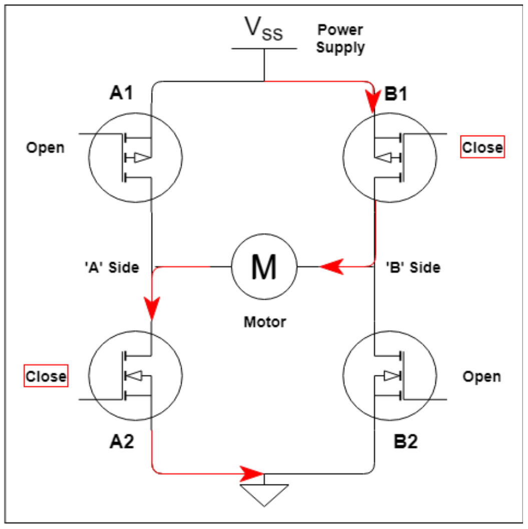


Fig. 19: Current Direction for Anticlockwise Rotation

When both the terminals of the motor are given the same potential then an electric brake is generated which does not let motor to spin in either direction. The condition can be achieved by connecting both the terminals to the power supply (see Fig. 20) or both to ground. The DC motor acts as a generator when the motor shaft spins and by connecting either the positive supply or ground to both the terminals, the generated electricity is sent back to the same supply, the result is that the motor shaft resists to spin and is kept stationary by forcing the opposite voltage into the same supply. The condition for electric brake is very useful in situations where the rotation of motors must be immediately stopped as well as for braking on a slope.

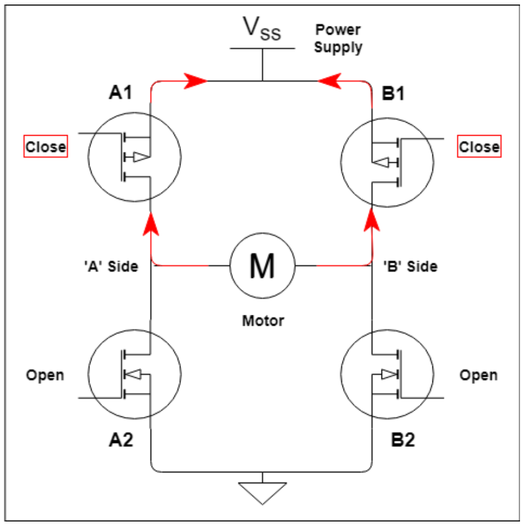


Fig. 20: Electric Brake Condition

The four possible on-off combinations of the transistors are shown in Table 10.

Table 10: Possible on-off combinations of the transistors

A1	A2	B1	B2	Terminal (A)	Terminal (B)	Motor Action
Close	Open	Open	Close	9 V	GND	Forward
Open	Close	Close	Open	GND	9V	Backward
Close	Open	Close	Open	9V	9V	Brake
Open	Open	Open	Open	0 V	0 V	Motor is OFF

The motors used in the project require a voltage of at most 9 V and maximum current of 900 mA per motor (for two motors, the maximum current is 1800 mA) but the Arduino Mega is only capable of providing a maximum voltage of 5V and maximum current of 200 mA, which is significantly less than the required amperage and voltage. To solve this problem, an Arduino Motor Shield is used as an intermediate device between the motors and the Arduino Mega. The motor shield should be connected to an external 9 V power supply and receives signal from the Arduino Mega through its digital I/O pins, to control the movement of motors. It consists of a L298 motor driver IC that allows us to control two brushed DC motors simultaneously. The operating voltage of the motor shield is 5-12 V which is sufficient for the motors and the maximum current per channel is stated as 2 A (4 A in total), enabling us to drive the motors and also leaves enough room (2200 mA) between the maximum current of the motors and one of the motor shield, as a safety precaution from spikes in current due to back EMF [53].

3.9. Pulse Width Modulation (PWM)

Pulse width modulation is a digital control method, used for generating analog output from a digital device, but this output is not technically analog, rather it is a modulated signal which is the digital equivalent to an analog voltage supplied by the device. The modulated signal is a square wave generated by switching the voltage supply between ON and OFF and it can produce the desired output voltage by controlling the duty cycle which is the percentage of time that a signal is ON during one cycle. For example, if the duty cycle of a pulse is 50 %, it means that the pulse is High for half the time of the cycle (see Fig. 21) [54],[55].

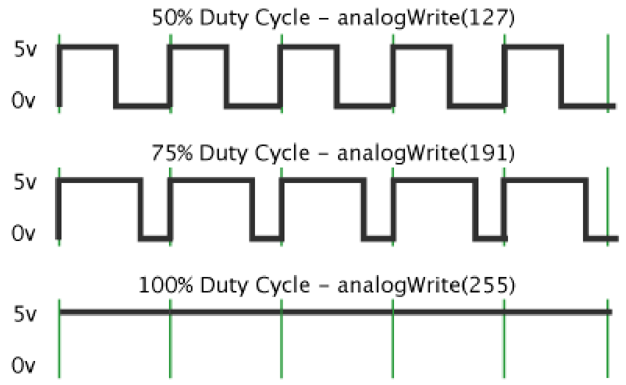


Fig. 21: Pulse Width Modulation Signals Corresponding to Different Duty Cycles

The duty cycle is defined by the following relation:

$$Duty\ cycle\ (\%) = \frac{Pulse\ width}{T} \times 100, \text{ where}$$

$Pulse\ width$ is the time for which the pulse is high and T is the period of the pulse

The output voltage is the average of the pulses and can be calculated by multiplying the duty cycle with the pulse's high value, as shown in the equation below:

$$V_{out\ average} = V_{max} \times Duty\ cycle$$

The other factor that affects PWM is frequency. Frequency reflects the number of ON/OFF cycles per second and is rated in Hz. It is equal to the inverse of the total time of one cycle (T) and thus is directly proportional to the duty cycle. If the frequency for the same pulse width is increased, it implies that the analog device receiving the signal is switched ON and OFF more times per second.

PWM is used in conjunction with the motor driver to control the rotational speed of the motors and thereby to control the movement of the robot. The PWM signal is generated by the Arduino connected to the motor driver, which allows duty cycles ranging from 0 - 255 where 0 corresponds to 0% and 255 to 100% duty cycle at a fixed frequency of 500 Hz, i.e., there are 500 ON/OFF switching cycles per second, where each cycle is 2 millisecond long. The signal is sent as a control input to the switching element (MOSFET) and by varying the duty cycles we are able to adjust the power applied to the motors. Consequently, the larger the duty cycle, the higher is the voltage at the terminals and thus the greater the speed of rotation [56],[57].

3.10. Ultrasonic Sensor

An ultrasonic sensor is a device used to measure the distance to an object. Fig. 22 shows the basic ultrasonic sensor operation. The transmitter sends out a sound wave at a specific frequency and the receiver waits for the sound wave to bounce back from the object to the ultrasonic sensor [58].

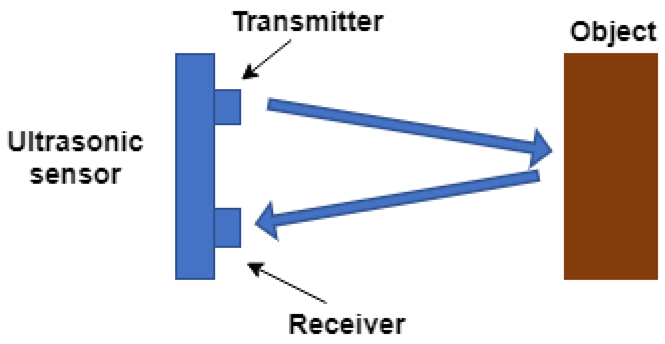


Fig. 22: Ultrasonic Sensor. Object Detection

The sensor measures only one variable, the time for which the sound wave reaches an object and bounces back. To calculate the relevant parameter, the distance between the sensor and the object, the formula shown below is used [58]:

$$Distance = \frac{Speed\ of\ Sound\ \times\ Time}{2}$$

Under normal atmospheric conditions, sound travels through the air at about 344 m/s. Division by two is required because the sensor measures the time the wave reaches the object and comes back, so the result would be the total distance travelled by the wave, i.e., twice the distance between the sensor and the object [58].

Although ultrasonic sensors can offer precise distance measurements, the possibility of not detecting a certain object still exists. Objects can be shaped or positioned in such a way that the sound wave bounces off but is deflected away from the ultrasonic sensor. Also, other objects may be too small to reflect enough of the sound wave back to the sensor to be detected.

The sensors that the group is using are HC-SR04. This model is one of the most basic ultrasonic sensors. As it is shown in Fig. 23, it has a transmitter, which emits the ultrasonic wave, and a receiver, which receives the reflected wave [59].

The group used four such sensors in the project, to serve for two purposes: obstacle detection and theft protection.

Table 11 [59] shows the important technical specifications of the HC-SR04 sensor. We found several advantages of using this sensor in the project: the fairly small dimensions of the sensor were suitable for the system; the measuring range was adequate since the longest distance the sensor could detect is 4 m and the price of the sensor was relatively low.

Table 11: HC-SR04 Sensor Specifications

Attribute	*Value*
Dimension	45 x 20 x 15 mm
Measuring Range	2 cm – 400 cm
Measuring Angle	30°
Power Supply	5 V
Working Current	15 mA

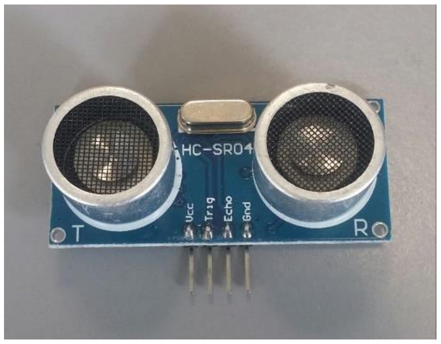


Fig. 23: HC-SR04 Ultrasonic Sensor

3.11. Tilt Sensor

A tilt sensor is basically an electrical instrument able to detect a change in tilt angle. Depending on the position (angle) of the tilt sensor compared to the ground line, it is able to distinguish between upright and rotated position. Fig. 24 shows the working principle of the given type of tilt sensor [60].

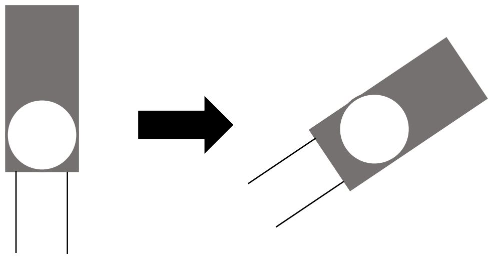


Fig. 24: Tilt Sensor Working Principle

When tilt sensor is standing completely upright, the ball falls on the bottom of the sensor. In that case, the two pins of the sensor are directly connected to each other, and current flows from one to another. When it reaches a pre-determined angle, it does not touch the poles anymore. It means that the circuit is open, and the sensor stops drawing current. Thus, the tilt sensor acts like a switch from an “on” state (upright position) to an “off” state (rotated) and vice versa [60].

The tilt sensor plays a role in the theft protection. SW-520D is the model of tilt sensor that we chose for our project. It has 2 poles and 2 metallic balls inside. Fig. 25 shows the exact model we are using in the project [60]. It can work without external power supply. It is also fairly low-profile and inexpensive, so it is appealing for a modest project, such as ours. The specifications of the sensors can be found in Table 12 [61].

Table 12. SW-520D Sensor Specifications

Attribute	*Value*
Dimension	5.2 x 5.2 x 19.5 mm
Supply Voltage	0 – 24 V
Working Current	5 mA
Lifetime	50000 switches
Switch Angle	45º

Fig. 25: SW-520D Tilt Sensor

3.12. Buzzer

The purpose of the buzzer in this project is to impede the theft of the robot by sounding an alarm. Simply, a buzzer is a sound emitting device that works upon receiving current as input. There are 3 types of buzzers: mechanical, electromechanical and piezoelectric. The group is using one piezo buzzer which was found in the Electronics Laboratory of the university. The sound of the piezo buzzer depends on the frequency it is set to work with as well as the current that is flowing through it. Even though the sound of the buzzer we used was not loud enough for the project’s purpose, we still decided to use it since implementing mechanical or electromechanical buzzers would have been complicated because of their relatively big size and weight. Table 13 [62] shows the specifications of the chosen buzzer. The buzzer is drawing only 3 mA at most and it is also low-profile. Finally, the buzzer has a frequency range of 1 KHz that gives many variations of sound.

Table 13. Specifications of RS-7800731

Attribute	*Value*
Dimension	30 x 30 x 13.5 mm
Voltage	1 – 30V
Working Current	3 mA
Max. Sound Level	90 dB
Frequency	2.3 – 3.3 KHz

Fig. 26: RS-7800731 Buzzer

3.13. Battery

The AGV built as part of the project is a mobile device and thus a battery was needed to power the system. Moreover, the fact that the vehicle should operate on a daily basis required the battery to be rechargeable.

To choose a battery with a suitable capacity, an estimation of the power needed by the circuit had to be made. Table 14 lists all the individual components that form the system as well as the estimation of their maximum power requirements, calculated by multiplying the voltage that the Arduino board could supply to each part and the maximum current, found in the data sheets of the components. In the case of motors, where the exact amount of power was known (torque multiplied by the angular speed), that precise value was used.

Table 14: Estimation of Power Requirements

Component	*Max. Power Consumption*
ATmega2560	250 mW
DC Motors (2)	2180 mW
Ultrasonic Sensors (4)	300 mW
LCD	1000 mW
Keypad	100 mW
Tilt Sensor	30 mW
Buzzer	15 mW
Temperature Sensor	7.5 mW
IR sensors (5)	379.5 mW
RFID Reader	85.8 mW
Total	**4.35 W**

As it can be seen, the total estimated power was around 4.35 W. That means a battery that would provide this power was needed. Regarding the voltage, the motors required the highest voltage level (9 V). Thus, the ideal battery would be able to supply that voltage. Then the current was calculated by dividing the power by the voltage:

$$I = \frac{P}{V} = 0.48 \text{ A}$$

The amount of time it takes for the battery to discharge was needed as well for selecting the appropriate capacity. Since the AGV is only a proof of concept, the time was chosen as being one hour. Therefore, a very good power source would have been a 9 V battery with a capacity of around 0.5 Ah.

Multiple types of battery technologies are available on the market. Assessing all of them is far beyond the scope of the project. It is worth mentioning that in electronic projects where motors or other high current components are used, the recommended battery types are Lead Acid and Nickel-Metal Hydride (NiMH), both being rechargeable [63]. The former one is suitable for large power applications (e.g.: hospital equipment). It is very economical and has a low self-discharge rate, but it is not so efficient in terms of energy density, that is, it cannot store big quantities of energy per unit volume. A typical lead battery is 12 V rated. NiMH batteries are very common, as they have high energy densities, they are lightweight, even though they have a high self-discharge rate. Regular rechargeable AA NiMH batteries are generally 1.2 V rated. To meet the voltage demands of the project, a pack of at least eight such batteries would be required, perhaps more, if one pursues an increase in capacity [64],[65],[66]. Price was one of the main criteria followed when the battery was acquired. Therefore, the team opted for UL 2.4-12 (Fig. 27), a 12 V Lead Acid battery with a capacity of 2.4 Ah. It can provide 28.8 W, far more than the circuit required, according to the estimations shown earlier. Regarding its technical specifications, they are listed in Table 15 [67].

Table 15: Specifications of UL 2.4-12

Attribute	*Value*
Type	Lead Acid
Weight	0.89 kg
Voltage	12 V
Capacity	2.4 Ah
Cycle Voltage	14.4 V – 15 V
Max. Discharge Temperature	50ºC

Fig. 27: UL 2.4-12 Lead Acid Battery

3.14. Buck Converter

As the reader probably noticed after reading the sections that discussed motors and batteries, it was mentioned that one can apply a maximum voltage of 9 V to the JGA25-370 Brushed DC motors. However, the chosen battery is labeled with 12 V and thus, directly connecting the battery to the motors may lead to temperature increase and eventually to failure. To avoid any unwanted behavior, the voltage supplied to the motors had to be stepped down to a value of 9 V.

Multiple ways of obtaining a lower voltage output from a higher voltage input have been identified. Firstly, one could use two Operational Amplifiers (op-amps) to achieve this goal: an inverter with a gain of -3/4, whose output is given to an inverter with gain -1. The method may work, but the fact that there is a need for supplying the op-amp poses a considerable problem, when only one voltage source is available.

More promising options are represented by linear regulators or switching converters. Linear regulators are inexpensive, small, not very complex devices, and yet they are very inefficient, wasting high amounts of energy as heat. Switching converters bring a very high efficiency and are better at handling temperature. Their complexity is higher, they come in bigger sizes and have higher costs. In many applications where the efficiency is an important demand, switching regulators are preferred [68]. That is the case for this project, since the battery life needs to be preserved as much as possible.

A common DC-to-DC switching converter is known as buck or step-down converter, because it converts a voltage given as input into a smaller voltage with the same polarity at the output [69]. Even though the initial intention of the project group was to design and build a buck converter, the sources found showed that carrying out this task requires extensive knowledge about electronics, involving concepts and methods that are above the scope of this project. Given that, the logical decision was to purchase a buck converter. We opted for the XL4015 LM2596 DC-to-DC adjustable step-down converter (Fig. 28). The specifications of the component are listed in Table 16 [70].

Table 16: Specifications of XL4015 LM2596

Attribute	*Value*
Input Voltage Range	4 V – 38 V
Output Voltage Range	1.25 V – 36 V
Max. Current	5 A
Efficiency	96%

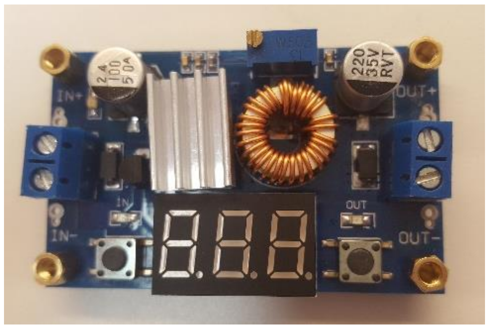


Fig. 28: XL4015 LM2596 Buck Converter

The selected model has three built-in 7-segment displays that allow the user to see the values of the input and output voltage, which is a very convenient feature.

The buck converter solved the battery issue and allowed us to calculate the actual time the battery would be able to supply power to the circuit. According to the estimations, the circuit needed a power of 4.35 W to run properly. For the converter, the following relationship exists between the input and output powers:

$$P_{IN} = \frac{P_{OUT}}{eff}\text{, where } eff \text{ is the efficiency of the converter}$$

Setting the output power to the desired value of 4.35 W, the formula gives:

$$P_{IN} = 4.53\ \text{W}$$

The battery is rated with 12 V, so the current supplied by the battery is:

$$I = \frac{P_{IN}}{V} = 0.37\ \text{A}$$

Thus, the battery only gives 0.37 A. To see the time until discharge, we divided the capacity of the battery (2400 mAh) by this value of the current:

$$t = \frac{c}{I} = 6.5\ \text{h}$$

Consequently, with the selected battery and converter, the AGV would be functional for a much longer period than the desired one.

3.15. Temperature Sensor

When leaving a battery system unattended, it is important to monitor its functioning variables, so we know whether everything is working correctly and have a reaction protocol in case it does not. One of the most important variables to take into account when measuring the battery’s state is temperature. When there is a large energy input or output, batteries tend to warm up. If this energy flow persists through time and temperature keeps rising, we might reach the temperature threshold of the battery, that, if we are using a sealed lead-acid battery, can lead to explosion. This happens because when temperature rises quickly in batteries with this technology, hydrogen and oxygen gases are produced inside of the battery. If the pressure and temperature in the battery cells is too high, this gases can break through the seal of the battery, causing an explosion [71].

Although there is a wide variety of different temperature sensors compatible with Arduino devices, our choice was DS18b20 (Fig. 29) for different reasons. The main one is its simplicity. It does not require any kind of formula or data treatment to get temperature readings, so it is nearly “plug and play”. Another solid reason is its OneWire interface, that allows precise digital temperature readings to be managed with ease by microcontrollers like Arduino with just one serial data output, as long as the correct libraries are used. These digital interfaces allow longer cables to be used without any perceptible change in the reading, as digital systems are less affected by attenuation [72]. Also, the cable-shaped format of the sensor was very convenient, as it allows us to install the integrated circuit comfortably next to the battery, even if the IC and the microcontroller are far away from each other. The specifications of the sensor are shown in Table 17 [73].

Table 17: Specifications of DS18b20

Attribute	*Value*
Supply Voltage	3 – 5.5 V
Max. Error	±0.5 °C
Max. Standby Current	1 μA
Max. Active Current	1.5 mA

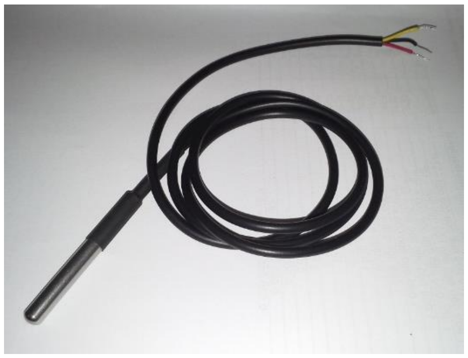


Fig. 29: DS18b20 Temperature Sensor

3.16. Inductive Charging

Theoretical Model

Creating a charging station in a public environment like a hospital entails problems: the robot may not be able to make the plug and the socket match and, when the robot is away from the station, somebody can short the plug. The most practical way of avoiding this is by using inductive charging, as energy is transmitted wirelessly between station and robot, so all the aforementioned problems are avoided.

As its name suggests, inductive charging is based on electromagnetic induction. This property defines the behaviour of electrical circuits (or any conductive surface) when they are exposed to varying external magnetic fields. [74] So, our main goal is to convert the electrical energy of the powerline into a magnetic field that will induce a current towards the battery, charging it.

In order to achieve this, the solenoid (or coil) is a fundamental piece. The electrical property that makes solenoids so special is inductance. Inductance is defined, according to A. Bettini as the *“proportionality constant between the current that certain circuit carries and the magnetic flux* *produced by it."* Therefore, the mathematical definition of inductance would be as described in equation 1:

$$L = \frac{\Phi}{I}\ (H) \qquad (1)$$

where L is inductance (in henrys), Φ is magnetic flux (in webbers) and I is current (in amperes.) Solenoids are characterized by their high inductance compared to a normal circuit, meaning that they create relatively high intensity magnetic fields when certain current runs through them. Thus, they also work the opposite way: even when they do not have power source directly connected to them, they can induce current in a circuit if the magnetic flux across them is powerful enough.

But this only applies for a short period of time, until the circuit reaches the steady regime, the reason being Faraday's law of induction, shown in equation 2:

$$\varepsilon = -\frac{d\Phi}{dt}\ (V) \qquad (2)$$

where ε is electromotive force, measured in volts and the magnetic flux is measured in webbers. This equation means that the energy that we transmit through inductive charging is simply a reaction to the variation of the magnetic flux in the coil. Thus, if the current that flows through the transmitting coil is constant (creating, therefore, a constant magnetic field), there will be no electromotive force transmitted.

Thus, if we want a constant delivery of energy between two coils, the current in the transmitter must be always changing, in other words, we need alternating current (AC) to make the system work.

Module Choice

Although we considered building the whole wireless charging system ourselves, we decided to buy a built module, as designing requires some deep research on which frequencies are ideal for maximum power efficiency and building coils that are light and compact enough for our purpose is no easy tasks in the timespan available. For our prototype, we used the PW-WCG-01 inductive charging module. It includes both transmitter (Fig. 30) and receiver, so the only thing you need is to connect a DC power supply to the transmitter [75]. The specifications are shown in Table 18 [76].

Table 18 : Specifications of PW-WCG-01

Attribute	*Value*	
	Transmitter	Receiver
Max. Voltage	13.2 V DC	12 V DC
Max. Current	-	600 mA
Coil Inductance	30 μH	30 μH
Max. Range	2 cm	
Current (1 mm distance)	600mA	
Current (18 mm distance)	10mA	

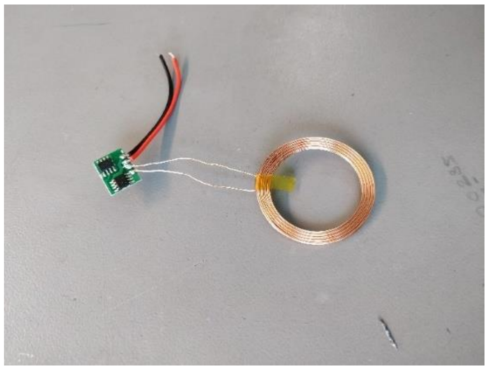


Fig. 30: Transmitter of PW-WCG-01 Wireless Charging Module

3.17. Boost Converter

Since the maximum output of the wireless charging module could only provide a voltage of 12 V and the battery required at least 14.4 V for charging, a method of stepping up the voltage was needed. The dilemma of choosing the right component for amplifying the voltage applied to the battery was solved quite easily. Op-amps were still not suitable as the observation in the "Buck Converter" section suggests, while linear regulators are designed solely for stepping down a voltage. Therefore, a boost converter was selected.

A boost or step-up converter can be considered "the opposite" of a buck regulator, since it takes at input a DC voltage and outputs a voltage of a larger value [69]. The regulator (model XL6009) was taken from the Electronics Laboratory of the university. The component can be seen in Fig. 31 and its specifications can be found in Table 19 [77].

Table 19: Specifications of XL6009

Attribute	*Value*
Input Voltage Range	3.5 V - 32 V
Output Voltage Range	5 V - 35 V
Max. Current	3 A
Efficiency	96%

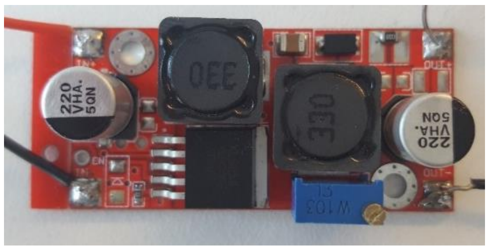


Fig. 31: XL6009 Boost Converter

4. Implementation

4.1. Keypad and LCD

The keypad was used to allow the user to interact with the AGV, as there was a need for the hospital personnel to be able to send goods to a specific room. The eight pins of the matrix keypad corresponding to the rows and columns were connected to eight distinct digital pins of the Arduino Mega. Regarding the aspect of displaying the options from which the user can choose a destination from, the LCD was utilized and it was connected to the microcontroller using only the four pins of the I^{2}C daughterboard: VCC, GND, SDA and SCL, as the Arduino Mega board has in-built SDA and SCL pins.

The Arduino library *Keypad.h* was used for programming the keypad because it allowed the use of premade, intuitive functions, such as *char getKey()*, which returns the pressed key, if any. Naturally, the keypad was programmed along with the LCD, so that users could immediately see the effect produced by pressing a certain key. The LCD code was written using the *LiquidCrystal_I2C* library. In this case as well, the functions were very easy to understand and employ. As an example, the method *setCursor(int, int)* moves the cursor to the specified row and column. Two menus the user can navigate around using the keypad were created: *mainMenu* and *oneRoomMenu*. The former allows the user to send the robot to all the wards, charging station or only one room. The third option leads to the second menu, where the possibilities are as follows: laundry, canteen, pharmacy and five different wards. Selecting an option would set the destination of the AGV to that specific location. Scrolling and selecting are possible using the letters on the right-most column of the key matrix: *A* moves the cursor one row up, *B* moves the cursor one row down, *C* is the button for selecting an option and *D* is used for going back to the previous menu, if applicable (in the *oneRoomMenu* only). Unsurprisingly, the digit keys do nothing more than printing the corresponding number on the screen when typing numbers is required.

It is important to mention that before being able to send the vehicle to a certain room, the user must enter a four-digit PIN code. The system does not allow menu navigation if the correct PIN is not inserted and thus only people who know the code (authorized hospital personnel) would be allowed access.

4.2. Wheels

As it was mentioned before, the AGV employs four wheels out of which only two are driven. The motors were connected to the standard wheels placed on the rear side of the robot, while the omni wheels were left free, in the front. When turning, the rollers mounted around the omni wheels start moving, permitting smooth steering. All the four wheels were attached to the drive base using screws. The omni wheels were shipped with no coupling and thus an adequate support for each wheel was printed. The reader recalls perhaps the small difference of about 5 mm between the wheel diameters. As already mentioned in the “DC Motors” section, the standard wheels were already connected to the motors and therefore two omni wheels of a matching size were needed, but the dimension closer to the sought one was found to be 70 mm. This impediment was easily overcome, as the only thing that had to be done was to design the support for the omni wheels such that the base would remain parallel to the ground, that is, two structures 2.5 mm shorter than the ones attached to the standard wheels.

4.3. Line Following

The main function of the robot is that it should be capable of travelling from a starting point to the desired destination without any assistance from the user and fully autonomously. To do so, a network of black lines is placed which connects all the points that the robot needs to go. The black lines act as a roadway for the robot as it is the path that it follows to reach its destination. Once the place to which the robot needs to be sent has been set by the user, the robot starts to continuously tracks the black line until it has reached its destination. To follow the line, it has to frequently adjust its position so that it always stays on top of the line.

Sensor Arrangement

For sensing the black line, the project utilizes IR sensors which are able distinguish the colour difference between the line and floor. An array of five IR sensors is placed in front of the robot as shown in Fig 32, the arrangement of the sensors is done in such a way that the middle sensor (no. 3) is on top of the black line while sensors 2 and 4 are placed at a distance of 4 cm apart from each other. The distance between sensors 2 and 4 is equal to the width of black line, in such a manner that the respective sensors are not on top of the black line but adjacent to it during straight movement. In this arrangement, the sensors 2, 3, 4 are 1 cm apart from each other and similarly, sensors 1 and 5 are placed adjacent to sensors 2 and 4 respectively, maintaining the same distance. Furthermore, the sensors are a placed at a height of 2 cm above the ground. This height was decided based on the range of IR sensor as well as by testing different heights for best readings.

Thresholds for Black and White

For the robot to know the difference between the colour of the floor and the black line, the readings for the respective colours from the IR sensors needs to be defined in the line following program. In order to do so the IR sensor readings were taken separately for the black line and the floor (presumably white in colour). The output of the sensors is also dependent on the input supply and because the sensors are powered through a battery, so when the battery discharges the input to the sensors also drops. Therefore, the sensor gives different readings for the same colour at different input voltage levels. Consequently, the readings for the black line is in the range of 144-255 and for the colour of the floor it is from 5-20. Form the range of readings for black line the smallest readings, i.e., 144, is set as an upper threshold in the program, so that any output from the sensor which is higher than this threshold is identified as the black line. Similarly, the largest reading from the floor, i.e., 20, is set as a lower threshold, so that any output below it is interpreted as the sensor being right above the floor.

Conditions for Line Following

For the robot to stay on top of the line and not to go rampant, the Arduino runs a code in a loop which tests a set of conditions that controls the movement of motors and if a certain condition is true then the code to run the motors is executed. These conditions are being tested according to the readings from the IR sensors to adjust the position of the robot using the black line as a reference which can be distinguished by the sensors.

Straight Movement

When sensor 3 is on top of the black line and the remaining four sensors are placed above the floor (as shown in Fig. 32), then the Arduino gives the command to run both the left and the right motors forward at maximum speed, which is defined in the program at a PWM value of 255.

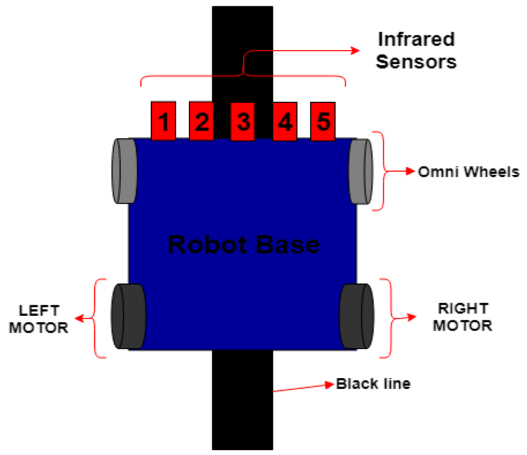


Fig. 32: Sesnor Arrangement and Straignt Movement Condition

Adjustment of Position

When the robot moves towards the right, sensor 3 is no longer on top of the line but is shifted in the right direction and now sensor 2 or both sensor 1 and 2 can be on top of the black line (Fig. 33). At this point, to adjust the position of robot back to the middle of the line, the speed of the left motor is decreased, and the speed of the right motor is at maximum which is greater than of left motor, thus the net movement of the robot is towards the left direction. This difference in speed between the left and the right motor is maintained until the middle sensor is back on top of the black line, which is the condition for straight movement. Similarly, when the robot moves towards left, sensor 4 or both sensor 4 and 5 can be above the black line (Fig. 34). To bring the robot back to the middle of the line speed of the right motor is decreased and now because the left motor has a greater speed than its counterpart, the net movement of the robot is towards the right. It is continued until the robot's position is adjusted back to the center of the line. The more is the difference in speed of rotation between the left and the right motor, the more quickly the robot will be able position itself in the middle of the line, but more difference also makes the movement of the robot look more mechanical as compared to a less speed difference.

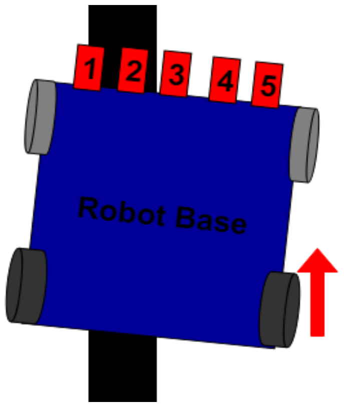


Fig. 33: Robot Shifted Towards Right

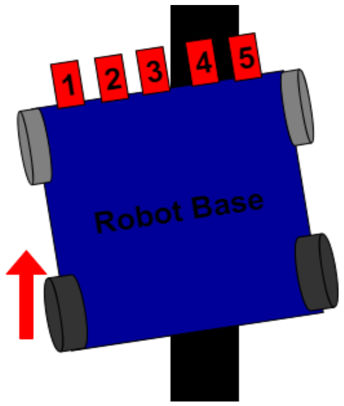


Fig. 34: Robot Shifted Towards Left

Conditions for Intersections

Apart from straight movement and the other adjustments the robot must do to stay on the line, it also needs to turn at the intersections. These intersections are the points where three or four black lines meet and at each of these intersections an RFID tag is placed which has a unique location and identification. To recognize an intersection, the code checks a set of conditions which uses the combination of sensors that are above the black line. For example, for an intersection that goes right, sensors 3, 4 and 5 are above the black line (shown in Fig. 35) and similarly, for an intersection that goes left, sensor 1, 2 and 3 are on the black line (shown in Fig. 36). At the intersection, the algorithm decides the direction of movement based on the RFID tag scanned. There are four possibilities for direction of movement that the robot can take at an intersection, it can be straight movement, 90° right turn, 90° left turn, and a 180° turn to retrace the path it came from. To take a 90° right turn, the robot spins the right motor in backward direction and the left motor in forward direction, this opposite directional spin is continued until the robot is facing towards the right and is in the middle of the line. Turning 90° left is obviously done using the same method. The rotation of right and left motor in opposite direction makes the robot turn at one point, rather than taking an arch-like turn which requires more space for turning. After the turn has been made the robot again starts to follow the line until the next intersection [78].

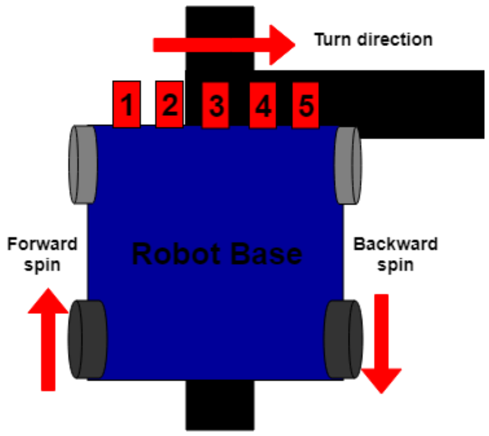


Fig. 35: Motors Spinning Direction for Right Turn

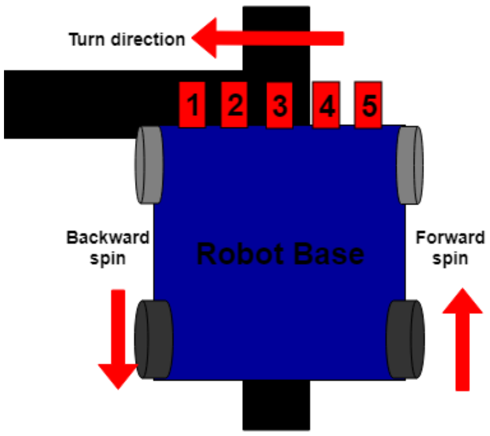


Fig. 36: Motors Spinning Direction for Left Turn

Code Diagram

The code diagram for turning is shown in Fig. 37, while the one for line following is shown in Fig. 38. Since they describe two different functions, they should be shown separately.

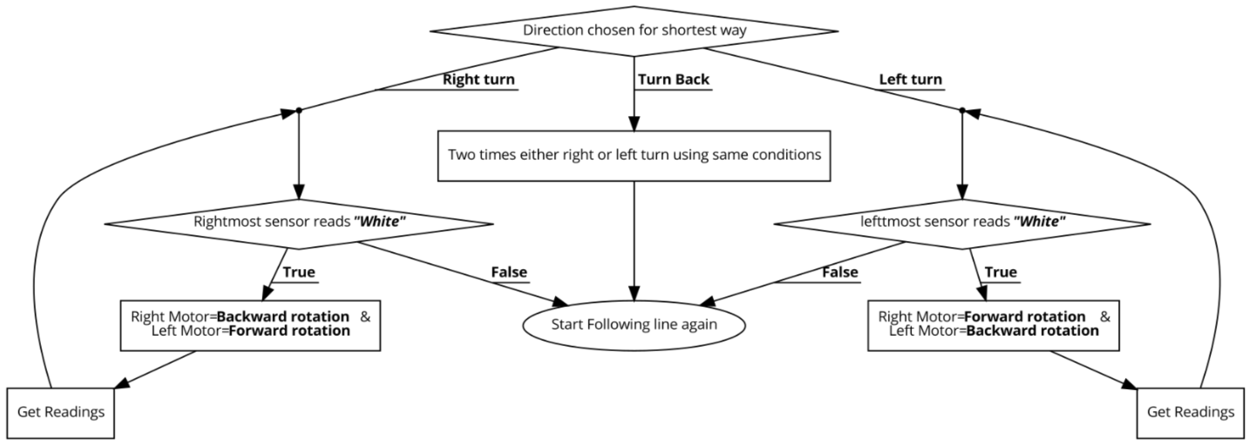

Direction chosen for shortest way
Right turn
Turn Back
Left turn
Two times either right or left turn using same conditions
Rightmost sensor reads "White"
lefttmost sensor reads "White"
True
False
False
True
Right Motor=Backward rotation & Left Motor=Forward rotation
Start Following line again
Right Motor=Forward rotation & Left Motor=Backward rotation
Get Readings
Get Readings

Fig. 37: Turning Code Diagram

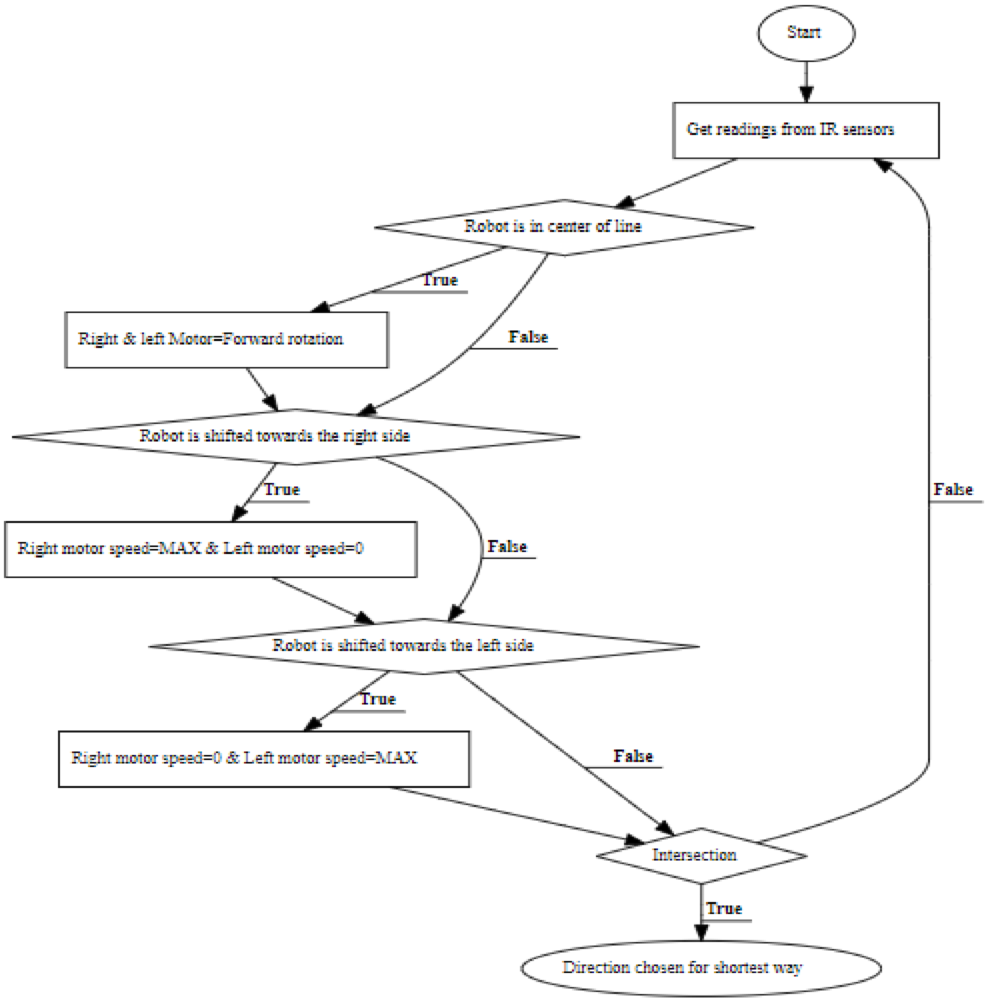

Start
Get readings from IR sensors
Robot is in center of line
True
False
Right & left Motor=Forward rotation
Robot is shifted towards the right side
True
False
Right motor speed=MAX & Left motor speed=0
Robot is shifted towards the left side
True
False
Right motor speed=0 & Left motor speed=MAX
Intersection
False
True
Direction chosen for shortest way

Fig. 38: Line Following Code Diagram

4.4. Obstacle Detection

As specified in the "Methods and Models" section, the HC-SR04 ultrasonic sensors have a sensing range of 4 m and an aperture angle of 30º. In other words, each sensor can cover an area equivalent to a circular sector with a central angle of 30º, where the radius of the circle is 4 m. The task of detecting all the obstacles that could emerge in the way of the AGV is beyond the sensing capabilities of a single sensor. For this reason, our robot employs three ultrasonic sensors for the mentioned purpose. They have been placed in the forepart of the device, since it was decided that the vehicle could only move forward. The Arduino code was written in such a way that, as soon as either of the three sensors detects an object closer than 30 cm lying within its ultrasonic beam, the electric break is activated and the robot stops.

In the described situation, having more sensors implies a bigger sensing range and thus a lower probability of collision. Notwithstanding, the number of sensors is not the only aspect that should be considered, as the arrangement of the devices is of an equal importance when one seeks minimizing the rate of potential accidents. For this project, the group members agreed to place the sensors in such a way that their beams would intersect at a distance of 20 cm from the vehicle, as shown, in a relatively crude way, in Fig 39. This was possible by positioning one sensor in the center and the others on its left and right respectively, shifted towards the exterior, with an angle of 15º with respect to the middle sensor. The distance between the midpoints of the sensors was calculated as being 5 cm. Because the distance at which each ultrasonic sensor sends the break signal to the microcontroller, i.e., 30 cm, is larger than the distance at which the ultrasonic beams are intersecting, i.e., 20 cm, the chances of not detecting an obstacle are relatively low.

No object would be able to enter the "blind spot" of the sensors without initially entering the ultrasonic beam, unless, of course, it may fall in that area. Even if it is initially seen by the sensors, an object may still move into the undetected area and a collision may occur. There are also other cases where obstacles can pass undetected, even though they are located inside the sensing area. This can happen either because of the material the object is made of or the angle at which the ultrasonic wave hits the target.

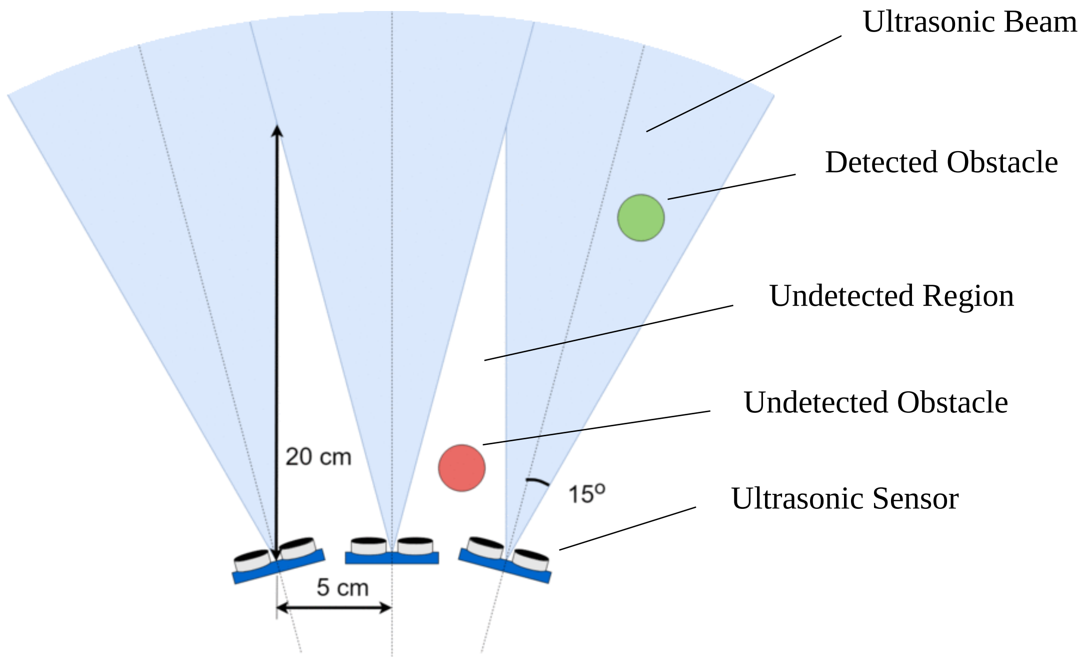


Fig. 39: Area Covered by the Ultrasonic Sensors, Top View (Drawing Not at Scale)

4.5. Theft Protection

Hospitals are public places so there is a possibility that the robot can be stolen. Consequently, a system is required which would notify the hospital staff in case of theft. For this purpose, a theft alert system was made, utilizing the distance sensing and tilt detection functionality of ultrasonic and tilt sensor respectively.

For the system, an ultrasonic sensor is placed on the robot in such a way that it is parallel to the floor and facing downwards, for constantly sensing the distance to the ground. A threshold distance is set so that if the robot is lifted up such that the distance between the ground and the robot is more than the threshold distance, the buzzer starts sounding the alarm. The threshold distance was set to be 30 cm as it was appropriate in the case of our small robot, but the distance can be adjusted easily to match the requirements of a professional environment.

The tilt sensor is used for detecting if the robot is being tilted towards any side. It is placed in upright position in such a way that if the robot is inclined by an angle of more than 45^o, similar to the ultrasonic sensor, an alarm is sounded using the buzzer.

This arrangement of sensors ensures the safety of the robot based on the perception that in the case of theft the robot would be lifted above the ground and can be tilted during the process.

4.6. Battery. Buck Converter

As the reader probably noticed in the section dedicated to batteries in the “Methods and Materials” section, the group intended to use a single battery to power the AGV. When the implementation phase of the project was reached and the battery was eventually attached to the setup, malfunction occurred. It was discovered that the issue arose because of the current limitations of the Arduino Mega. The only 3.3 V pin on the board could merely supply a maximum of 50 mA, and the five IR sensors required a total of 115 mA. The situation called for adding another voltage source to the circuit. An aspect to be considered this time was space, since the event was not at all predicted. The best option was to utilize three 1.2 V NiMH AA batteries, each with a capacity of 2400 mAh, placed in series to obtain a 3.6 V battery with the same capacity. There was no need to buy these batteries, since they were found in the Electronics Laboratory of the university.

The battery was naturally connected to the input of the buck converter. The output of the regulator was then given as input to the motor shield and implicitly to the rest of the circuit. Using the potentiometer, the push-buttons and the 7-segment displays on the converter, the input and output values of the voltage were set to the desired values, namely 12 V and 9 V, respectively.

4.7. Inductive Charging. Boost Converter

The implementation of the inductive charging in the robot was fairly simple. The receiving coil was connected to a boost converter in order to get the necessary 14.5V so the battery is charged correctly. A diode and a 130 Ω resistor were attached in series between converter and battery, to make sure that there was no overcharging and the battery was not consuming power when the charging process was finished. As for the location of the coil, we decided to place it in the rear of the robot, so it did not interfere with the sensors in the front, even if that entails that the robot must move backwards in order to find the charging station.

The transmitter coil has no fixed place, as it has to be connected to a DC power supply, so the charging must take place in the university laboratory. As the voltage and current regulation is made by the IC of the module, no additional components are needed.

4.8. Temperature Sensor

As mentioned before, the implementation of the temperature sensor was uncomplicated due to its physical format and the OneWire interface. We connect the ground and power cables to their respective Arduino pins, and the data wire to a digital pin. Although our temperature sensor supports parasitic powering (using the data connection as a power connection), we decided to ignore that feature as the readings are too unstable. So, we put a 4.7 KΩ between the data and power wires, following the recommendation of the manufacturer. Finally, we attach the thermal coating of the integrated circuit to the side of the battery, as in the top, due to the chemistry of the battery, the temperature is lower, so the reading is somehow altered [71].

4.9. Monitoring System

The monitoring system we designed consists of three sections: localization, battery temperature and information panel (all in real-time). These three parts are explained in depth in the following paragraphs.

Localization

It is important to be aware of the location of the vehicle at any time for security reasons. For this reason, we wanted to visualize the hospital corridor structure and show the real-time location of our robot on it. We designed a hallway structure (referred to as “map”) where we would test our robot. While designing the map, we tried to include various situations one can come across in a hospital such as dead-ends, paths of various length and U-shaped hallways. This way, we would see how the robot behaves in many different situations and this would give us an idea of its compatibility. Fig. 40 shows the map we designed as well as the RFID tag placement.

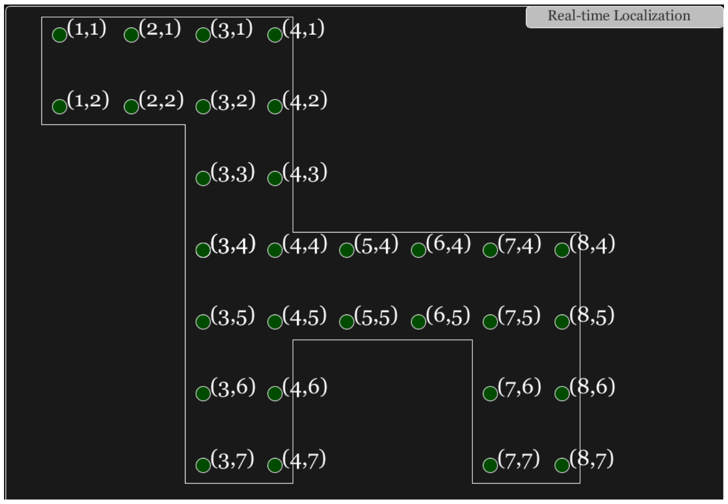


Fig. 40: RFID Tag Placement on the Hospital Floor

Not all the RFID tags are directly connected to each other on the line. Fig. 41 shows the connection between RFID tags, illustrated with the green lines. As it can be seen, our robot is capable of moving horizontal and vertical, diagonal movement is not possible. Moreover, these RFID tags form a network, meaning from one RFID tag there are at least 2 and at most 4 other RFID tags you can connect to. Knowing which RFID tags are connected is crucial for the algorithm to be able to draw the shortest way. For that reason, we created a matrix where we stored information about existing connections from each RFID tag. In addition, we did not want the robot to cross the hallway at any point it desires to. Instead, we wanted to let it cross the hallway at some certain points so that its action would be more predictable. To do so, we gave priority to some RFID tags where we wanted the robot to cross the hallway if necessary.

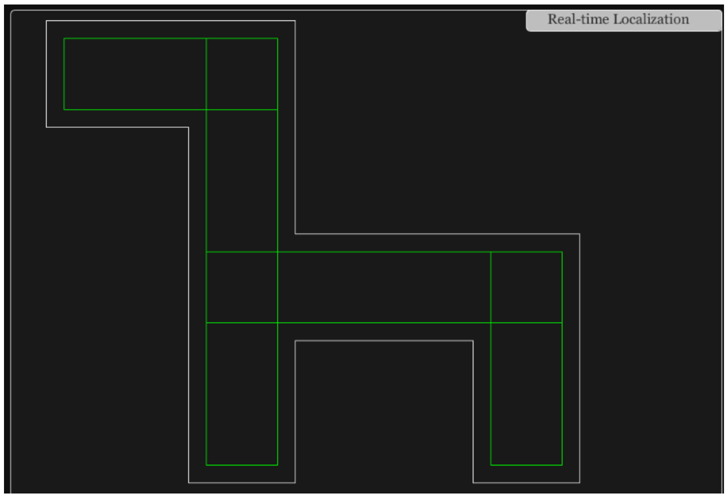


Fig. 41: RFID Connections

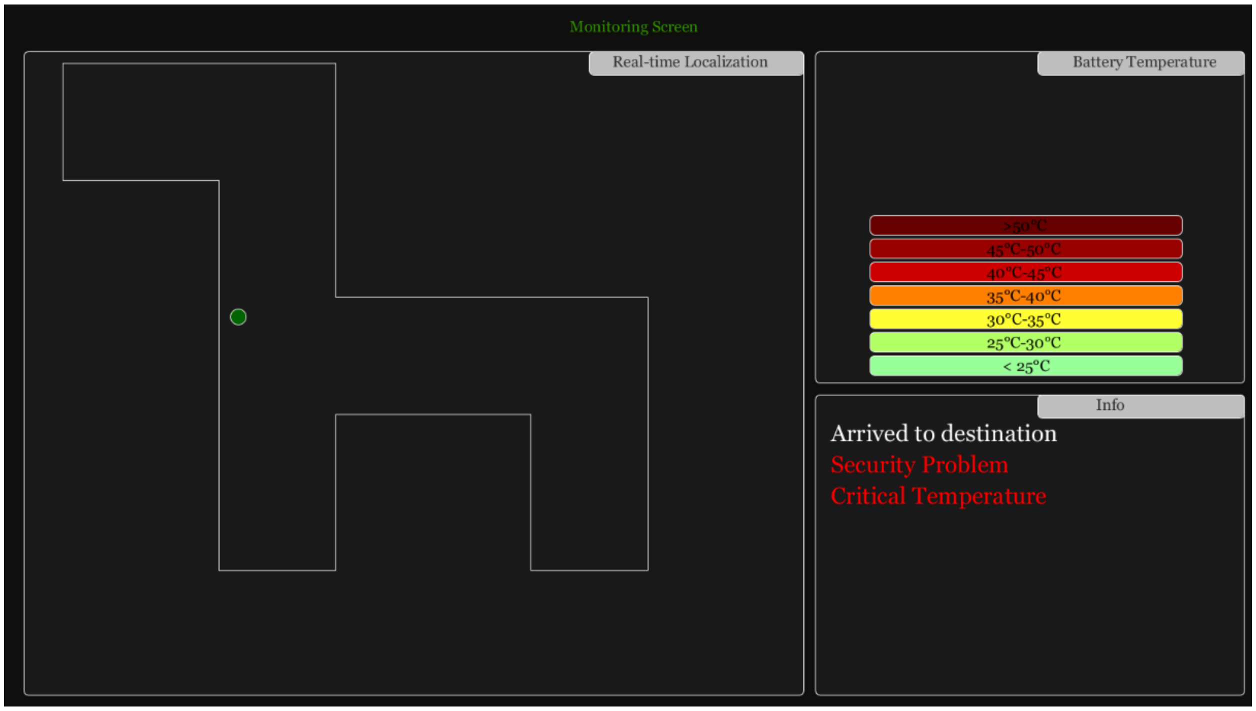
Our robot needs to find the shortest way between its current location and the selected destination in order to save time and power. To do so, we identified each RFID tag with a unique combination of two numbers (representing x and y values of its location) and placed these passive RFID tags underneath the line that the robot follows with a constant distance among them. Fig. 40 shows the given values of each RFID tag (indicated as green dots) with respect to their position on the map. The x values of the RFID tags increase as one moves right and y values of the RFID tags increase as one moves down, which means the origin is located on the upper left corner. This unique numbering of the RFID tags is the backbone of the program that finds the shortest way between two points.

Battery Temperature and Info Panel

Monitoring battery temperature is crucial to be able to avoid explosions due to overheat. We created 7 thresholds corresponding to every 5°C difference starting from 25°C until 50°C.

The info panel gives three important information. It shows whether the robot is on the move or not. It also prints the destination on the screen. In case of a security or battery temperature issue, this information is also displayed on the info panel.

The complete monitoring screen we designed is shown in Fig. 42.



Arrived to destination
Security Problem
Critical Temperature

Fig. 42: Complete AGV Monitoring Screen

On the left side of the figure, we see the 2D map of the hospital corridor where the robot operates. The green dot with a thin white frame shows the current location of the robot in action. As the robot moves and reads the passive RFID tags under the line, the green dot also changes its location on the map according to the readings. Battery temperature is always displayed on upper right corner. The info panel, situated on the right bottom corner, always updates its information. Thanks to this monitoring system, the person in charge, perhaps, can easily see some crucial information about the robot in real time.

4.10. Shortest Way Algorithm

Below is given the algorithm we developed in order to find the shortest way between two points on the map. We assume destination is already given to the robot.

1. Eliminate 2 directions that are unnecessary to move towards. For example, if the current location of the robot is (3, 4) and the destination is selected to be (5, 5), it is clear that the robot must definitely move right (because x value of the destination point is greater than x value of the current location) and down (because y value of the destination point is greater than y value of the current location). Moving left and up would be redundant since these moves would only increase the distance the robot has to travel. If x or y value(s) of two points are same and there is no direct connection between the tags, the program finds the closest intersection to the starting point where the robot may take the appropriate turn and arrive to the destination.

2. Make the comparison between determined directions in step 1 and possible connections from the current RFID tag to neighboring RFID tags. If there exist only one common direction, go to step 3. If there happens to be 2 common directions go to step 4.
3. Update your current location by following the steps below:
    a. If the chosen direction is ‘right’, increase x value of the current location by one and do not change y value.
    b. If the chosen direction is ‘left’, decrease x value of the current location by one and do not change y value.
    c. If the chosen direction is ‘down’, increase y value of the current location by one and do not change x value.
    d. If the chosen direction is ‘up’, decrease y value of the current location by one and do not change x value.

   Go to step 5.

4. Choose the first one of 2 common directions and change the current location by following step 3. Then calculate the distance between the current location and destination. Follow the same steps for the second one of 2 common directions. Compare the calculated distances and pick the shortest one to move towards. If two distances happen to be same, choose the one with priority, if applicable. If none has no priority choose one of them (it does not matter which one is chosen). Go to step 3.

5. Store the updated location as the first RFID tag for the robot to read on the shortest way to the destination. If the robot did not arrive to the destination, go to step 1. If the robot arrived, finish the program.

The flowchart of the aforementioned code is given in Fig. 43, on page 58.

Following the Shortest Way

After finding the shortest way, the next step is to make sure that AGV is facing the right direction so that it does not start following the line in the wrong direction. In order to keep track of its direction, we need to check its current direction and, based on the location of the first RFID tag to be read, decide the necessary move (e.g.: turn left, turn back, turn right, do not turn) before it starts following the line. This code is executed each time our robot reads a new RFID tag and updates its location. The flowchart of the aforementioned code is given in Fig. 44, on page 59

Reading an RFID Card

Each RFID tag is assigned with a unique identification number (UID) during the manufacturing process, used to identify each tag. After reading the tag, the AGV sends a corresponding number for the particular card to the PC. The Processing (an open-source programming language specifically developed for electronic arts) code always checks for new data sent by the AGV, and updates its variables depending on this data. This way, we were able to show the whereabouts of the AGV in real-time.

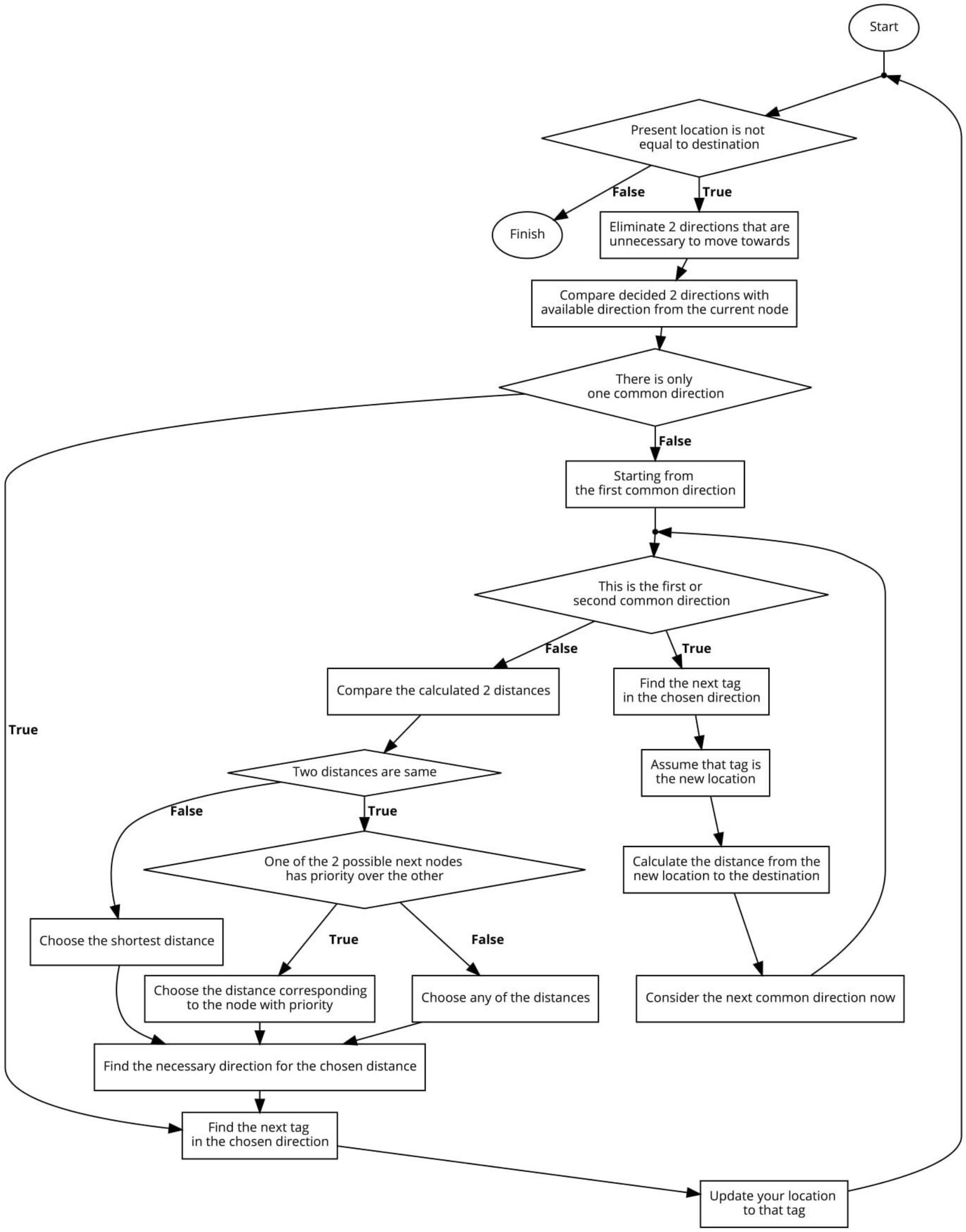

Start
Present location is not equal to destination
False
True
Finish
Eliminate 2 directions that are unnecessary to move towards
Compare decided 2 directions with available direction from the current node
There is only one common direction
False
True
Starting from the first common direction
This is the first or second common direction
False
True
Compare the calculated 2 distances
Find the next tag in the chosen direction
Assume that tag is the new location
Two distances are same
False
True
One of the 2 possible next nodes has priority over the other
Calculate the distance from the new location to the destination
Choose the shortest distance
True
False
Choose the distance corresponding to the node with priority
Choose any of the distances
Consider the next common direction now
Find the necessary direction for the chosen distance
Find the next tag in the chosen direction
Update your location to that tag

Fig. 43: Finding the Shortest Way Code Diagram

.

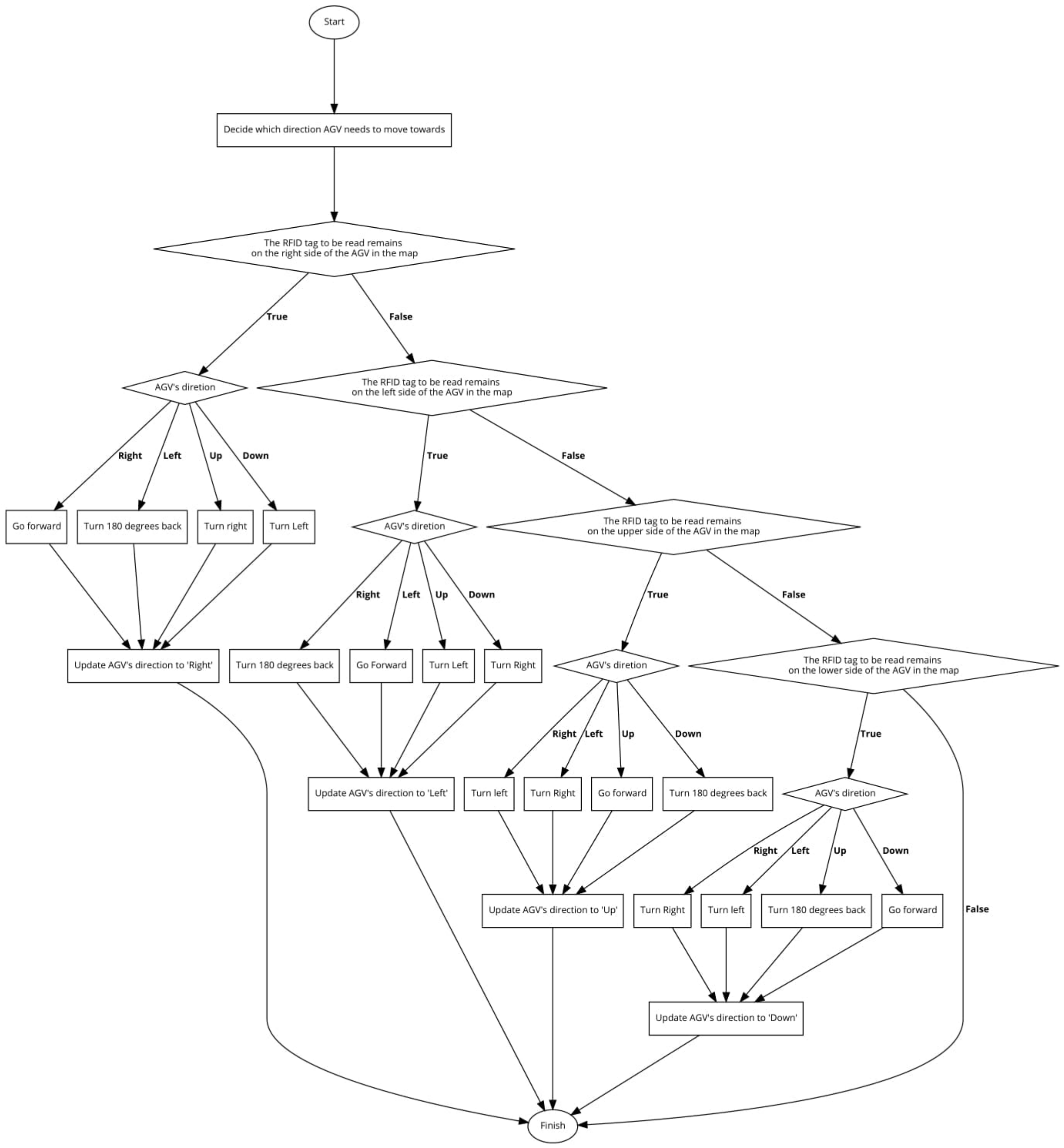
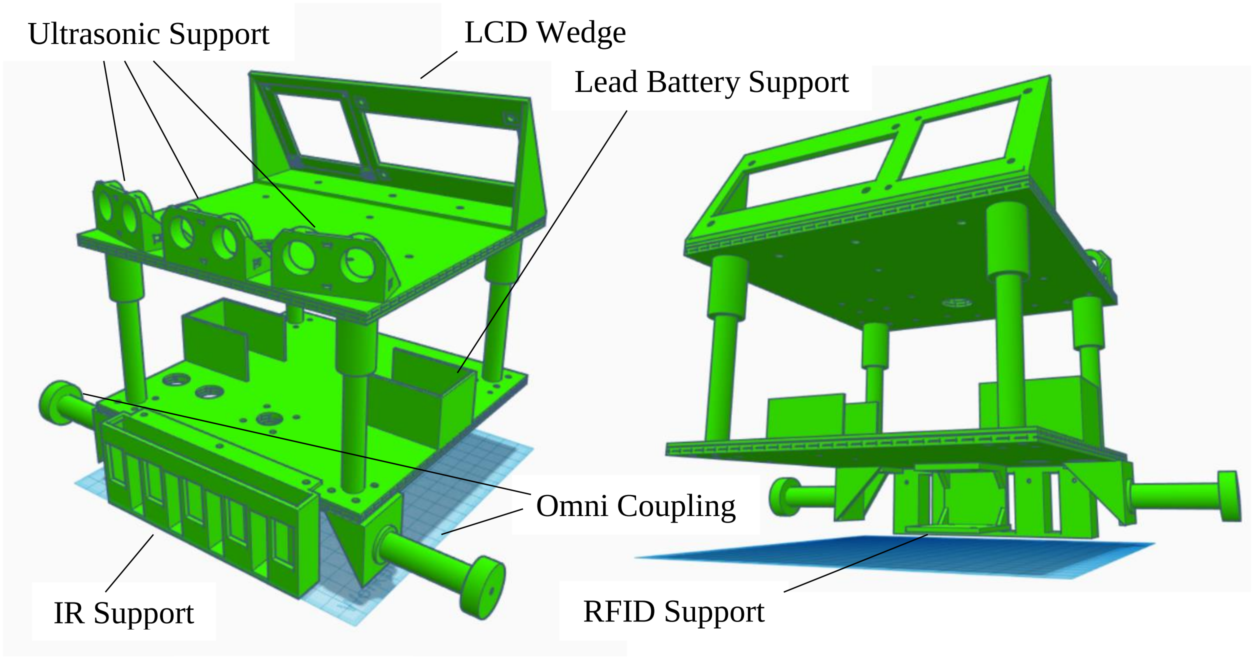

Start
Decide which direction AGV needs to move towards
The RFID tag to be read remains on the right side of the AGV in the map
True
False
AGV's diretion
The RFID tag to be read remains on the left side of the AGV in the map
Right
Left
Up
Down
Go forward
Turn 180 degrees back
Turn right
Turn Left
Update AGV's direction to 'Right'
AGV's diretion
The RFID tag to be read remains on the upper side of the AGV in the map
Turn 180 degrees back
Go Forward
Turn Left
Turn Right
Update AGV's direction to 'Left'
AGV's diretion
The RFID tag to be read remains on the lower side of the AGV in the map
Turn left
Turn Right
Go forward
Turn 180 degrees back
Update AGV's direction to 'Up'
AGV's diretion
Turn Right
Turn left
Turn 180 degrees back
Go forward
Update AGV's direction to 'Down'
Finish

Fig. 44: Flowchart for Following the Shortest Way

4.11. 3D Modeling

The weight limitations imposed by the motor calculations called for a light, yet solid drive base that would be able to carry quite heavy components, such as the battery, but would not substantially increase the required torque. Wood and aluminum are common candidates for being among the materials used for a mobile robot drive base. However, they fail to fulfill the weight criterion mentioned above.

The fact that the practical part of the project consists of a proof of concept, where many details related to a real product are ignored, gave the possibility to use the 3D printer for creating the base of the robot. Given the number of components utilized in the project and the limitations of the available printer, a one-story floor was found insufficient and therefore two floors were created: the bottom one was designed to hold the lead acid battery with the buck converter and provide connection areas for the four wheels, the IR sensors, the RFID reader, the temperature sensor, the wireless charging module and one ultrasonic sensor, while the idea behind the top part was to hold the Arduino, the three ultrasonic sensors for obstacle detection and the LCD with keypad. Apart from these two layers, other parts had to be modeled and printed, such as holders for IR sensors, RFID reader, LCD and ultrasonic sensors, coupling for the omni wheels. Even though, the base was not printed in one piece, only pictures of its complete form are shown (see Fig. 45), since this form is the relevant one.



(a) Front View **(b)** Rear View

Fig. 45: 3D Model of the AGV Drive Base in Tinkercad

4.12. Proof of Concept

In Fig. 46 the combined code diagram that describes the entire working of our vehicle can be found. Additionally, Fig. 47 shows a diagram of the full system that constitutes the proof of concept of the AGV, whereas Fig. 48 illustrates our proof of concept when all separate components are put together.

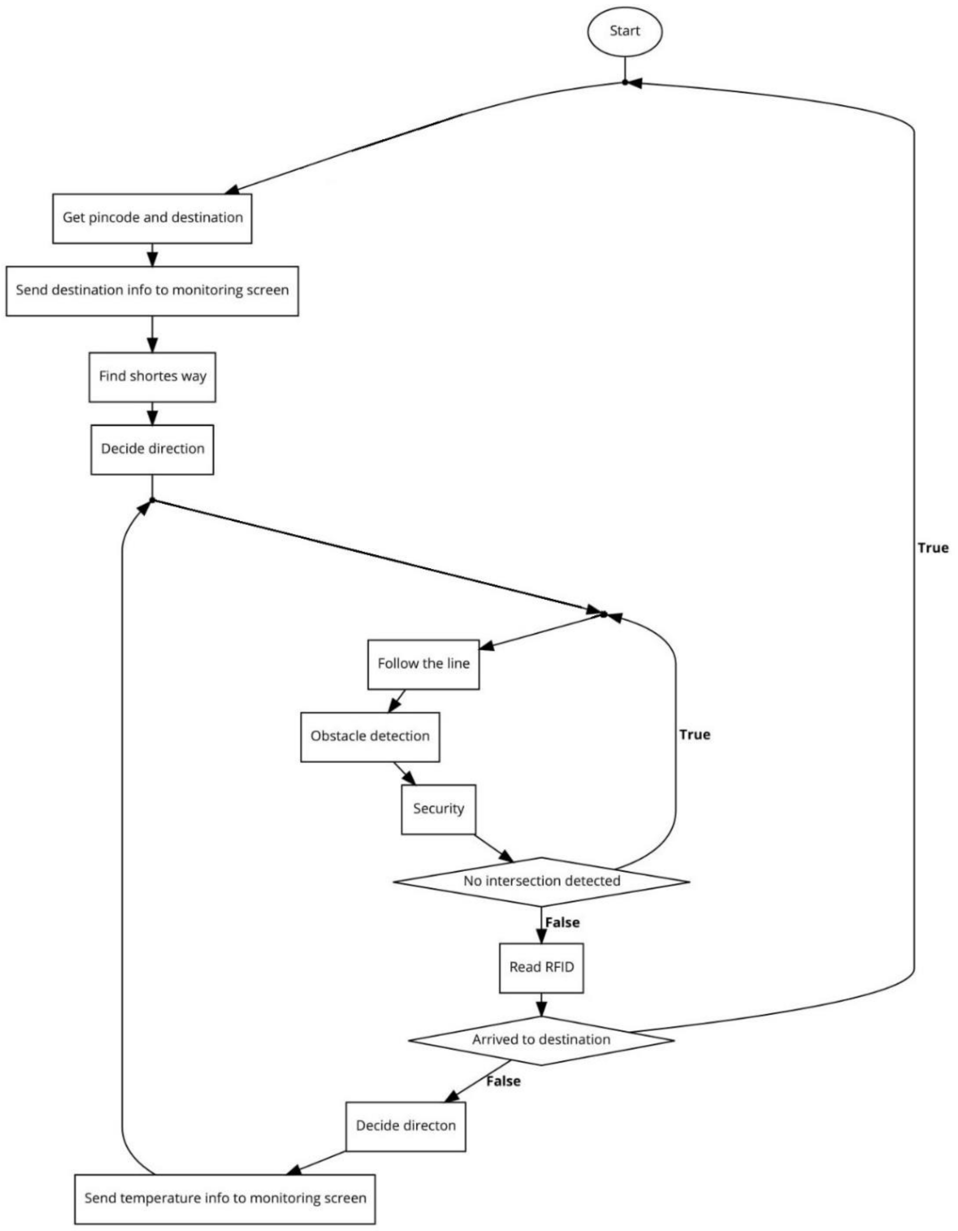


Fig. 46: Code Diagram for The System

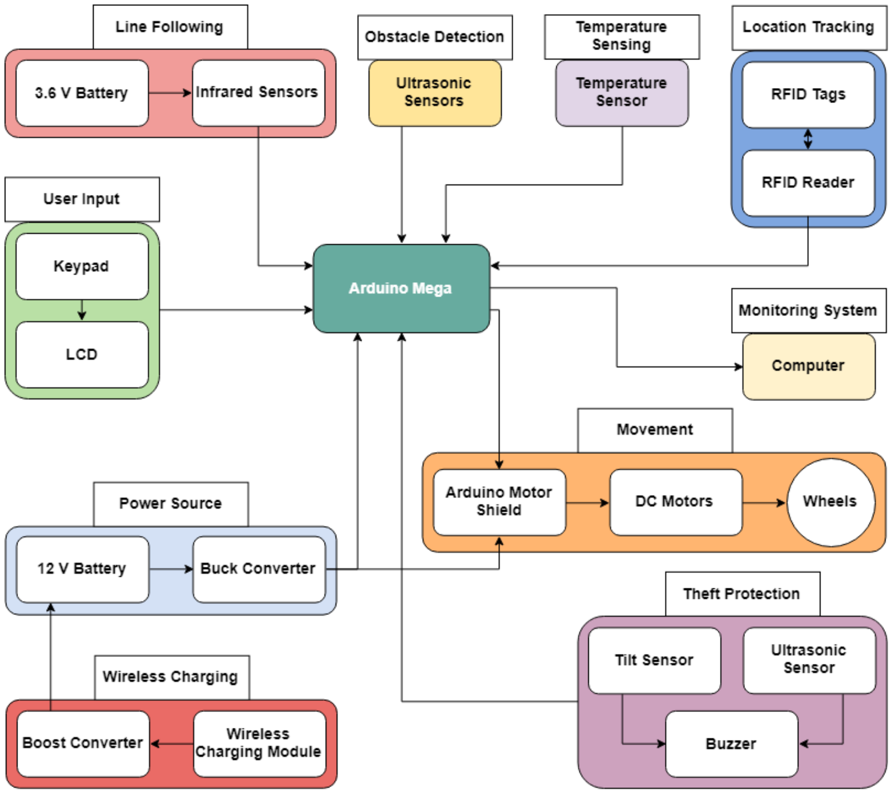
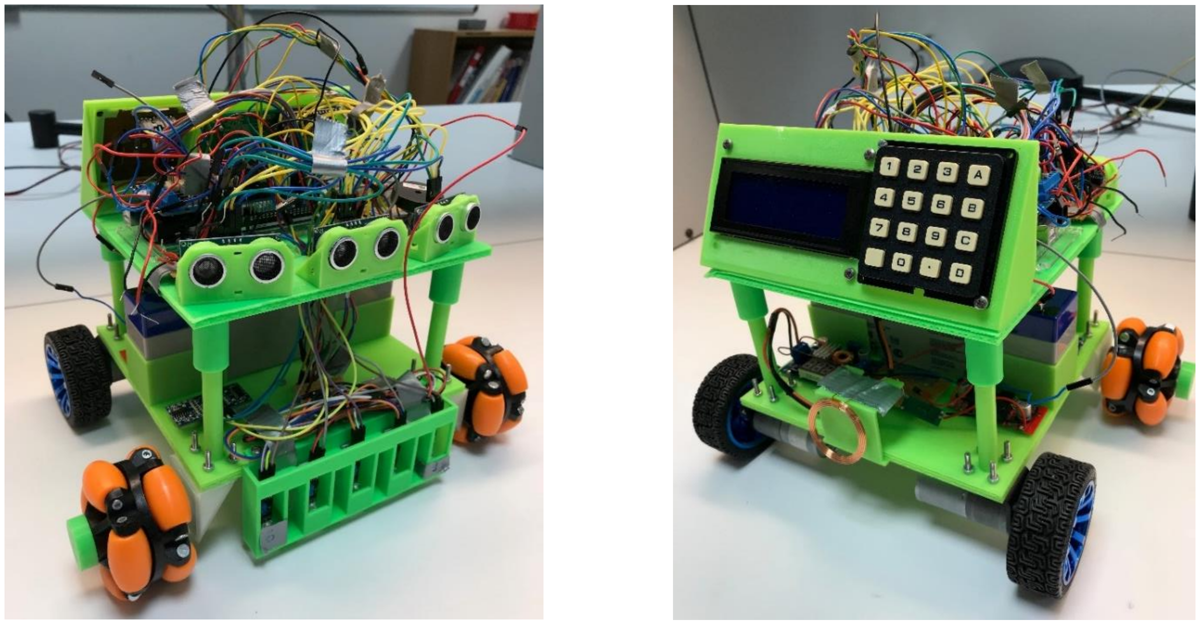


Fig. 47: Diagram of the Complete System



(a) Front View **(b)** Rear View

Fig. 48: Photos of the Proof of Concept AGV

5. Testing

We conducted two testing sessions where we found the chance to evaluate our work and improve it. They have been very helpful for the project since it allowed us to notice some errors we did not anticipate.

5.1. First Testing Session

When we had our prototype and code ready to function together, we set the test map in electronics lab on the floor using black tape. Our intention was to set the map wholly, but due to lack of material and not enough available area we were only able to complete 80% of the total map. Nevertheless, we decided to test our prototype within this test area (Fig. 49). We placed appropriate RFID tags under the tape at correct positions with 30 cm in between. We powered everything with an external power supply (12 V) except the IR sensors which were powered by an internal power supply of 3 NiMH AA batteries connected in series.

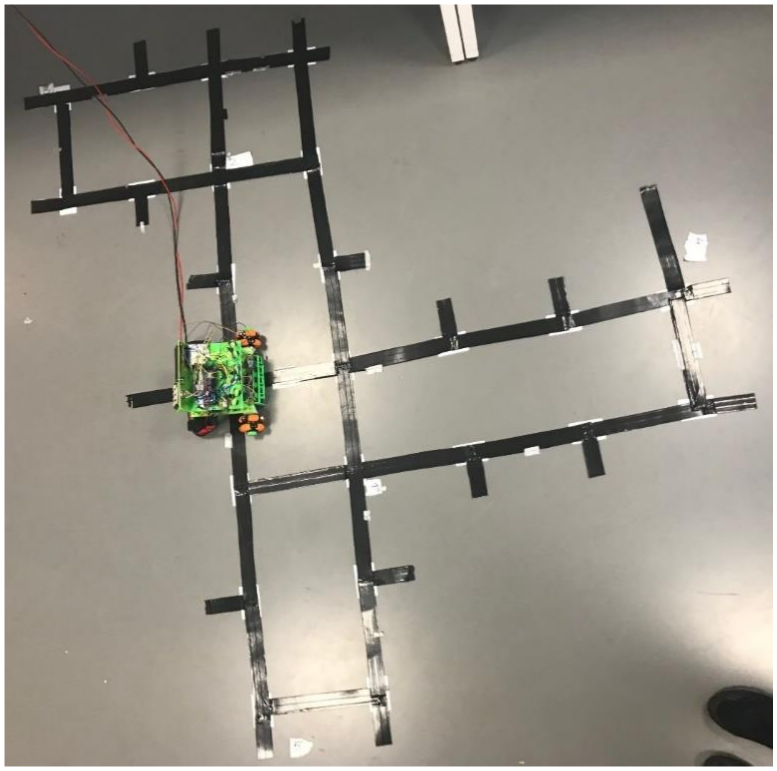


Fig. 49: Test Area 1

Upon a long session of testing, we have observed many errors. Some of them were hardware and software related errors. However, some were originating from the test field itself. All errors observed are listed below with an explanation of how we solved them.

The hardware related errors we observed during the test session are listed below.

Disconnection of Wires

We rearranged the whole wiring using small solder plates. This way, we were able overcome the disorder of wires to some extent. However, we experienced the same problem later again since not all wires were soldered and fixed at some point, for example the wires going out of mega.

Wrong Readings of IR Sensors

We had to calibrate the IR sensors with respect to the surface so that they could distinguish black tape from the dark colored floor. However, we had the same error later again since the readings were too sensitive to floor, light and voltage level.

Not Spinning Omni Wheels

The problem was mainly due to lack of friction. We tried to increase the friction by putting tape around the wheels. However, it did not solve the problem. The permanent solution would be to have a heavier robot. Since we could not increase the weight as we please (considering that we decided on the motors with respect to estimated weight which is 2.5kg), we had to neglect this problem.

Malfunctioning Tilt Sensor

Even though everything seemed correct hardware-wise, the sensor was not functioning properly. We were able to solve the problem by complementing the if-statement for the corresponding sensor.

Loose Nuts

We had to tighten them each time they were loose. Permanent solution would be using fibered nuts which we did not have in the laboratory and did not order.

The software related errors we observed during the test session are listed below.

Reading the Same RFID Tag Multiple Times

This problem was occurring as a result of detecting the intersection many times. As a result, the AGV was not able to decide the next move. The problem was solved by moving a lit bit forward right after reading the tag for the first time.

Trying to Follow the Line Even Though There Wass No Line

The reason for this problem was the lack of the condition where all IR sensors see floor's color. We tried to solve the problem by adding the above-mentioned condition. However, it did not solve the problem since there were times all IR sensors see floor's color even while following the line. Having included the new condition, the AGV was stopping when it is out of line as well as some-times it is on the line when all IR sensors see floor's color. For this reason, we did not implement the new condition. We could not spare more time to solve this problem due to time limitation.

Turning Left/Right/Back More Than Necessary

Before we were turning left/right/back for a period of time that we determined by trial and error. But this code was not working properly every time since sometimes motors were free spinning due to lack of friction. Therefore, we wanted to control the turning moves by getting feedback from IR sensors. With the new code, the prototype decides when to stop based on IR readings and this gives us a better precision.

The test field related errors we observed during the test session are listed below.

Robot not Able to Align Itself in the Middle of the Line

The problem was due to short distance between two successive RFID tags. We would solve this problem by increasing the speed difference of two motors if we were not already using one at highest speed and one at lowest speed. Therefore, we had to provide more time to the prototype to adjust its position over the line by increasing the distance between RFID tags.

Free Spinning Back Wheels

The problem occurred as a result of the thickness of the line. The real technology will follow a line painted on the floor, which means the line will not have any thickness. Therefore, we decided to ignore errors due to thickness of the line since we could not get permission to paint a map on the floor of electronics lab.

Conclusion

Even though we fixed many errors, our robot was still not functioning as we desired, mainly due to small distance between two successive RFID tags. For this reason, we were compelled to set a new test area where we left more space between the tags.

5.2. Second Testing Session

We set the second test area again in the electronics laboratory with more space between the tags, but the available space was still an issue. Fig. 50 shows the second test area. Again, we powered everything with an external power supply (12 V) except the IR sensors which were powered by 3 NiMH AA batteries connected in series.

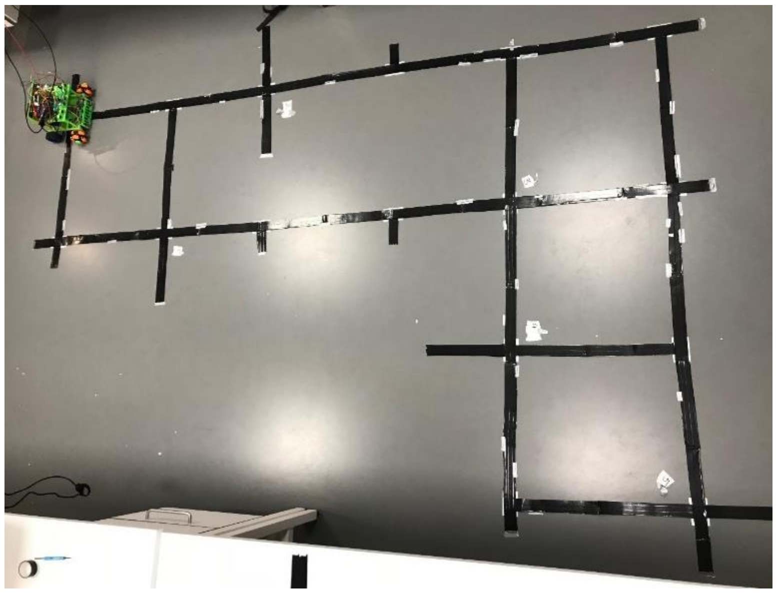


Fig. 50: Test Area 2

When everything was ready, we tested our prototype by sending it to different locations on the map. Unlike the first testing session, the AGV was able to arrive to the chosen destination most of the time and did not repeat the same errors previously solved. However, the thickness of the line was still an issue. There were times that the robot was unable to complete is turn due to the thick line. Since this error was caused by the test area itself, we decided to ignore it. The IR sensors were still problematic and sometimes they were not able to distinguish between the line and the floor. Apart from that, everything seemed to be working at a satisfactory level.

6. Discussion

In this project, we worked on building only a single AGV. When this was the case, we did not bother ourselves considering how it would be if we had multiple AGVs delivering goods simultaneously. It is clear that they would need to communicate to each other for many reasons. For example, we would not want to see two AGV's motionless due to the reason that they detect each other as an obstacle in front. Such problems may be overcome by establishing a communication protocol among the AGVs. One more important aspect would be to restrict their movement abilities on specific lines, such as letting them move only in one direction due to the fact that another AGV might be following the same line behind. This change would also possibly affect the algorithm to find the shortest way. Now, the AGV finds the shortest way to the destination right before it starts moving and it does not change this decided path due to any reason later on. However, it would be ideal to be able to change the decided path with another one which is the same length (there may be many paths between two points with same length) when the decided path overlaps with another AGV's path or when an obstacle is detected on the way. This requires having a different algorithm that can find the shortest way again when needed.

There are also some technologies we wanted to implement in our proof of concept but we could not, due to time limitations. For example, the AGV does not have any security measures for a lost communication with the monitoring screen. We would like the AGV to stop when the communication is lost since it would be impossible to monitor it anymore. The monitoring screen shows some real-time pieces of information from the AGV itself, but it does not record them for further investigation if needed. In addition, there is another security measure the AGV does not possess: it does not check if it is on the correct path to the destination. Due to an error in the code, the AGV may start following the wrong line. Since it does not have any means of self-checking the path, it will keep following the same line. In this scenario, it will not be able to arrive to the next RFID tag it needs to read and therefore keep following the line. Currently, it does not stop even when the line is over since we did not implement the condition for all IRs seeing floor's color due to the reason mentioned in the first testing session. Another point we want to discuss is the option to select speed. The AGV does not provide any option to select speed for the moment. It would be good to have an option to let the AGV move a little bit faster for urgent deliveries. Moreover, the communication between the AGV and the monitoring screen is now established through a wire, which is far from ideal. Our initial idea was to transfer data wirelessly. Unfortunately, even though we had bought the necessary wireless module, we have not used it due to time limitations. One more parameter we wanted to display on monitoring screen was the voltage level of the battery. This was also left undone for the same reason. Moreover, the theft protection system can be further improved by employing an accelerometer instead of the tilt sensor for better measurements.

A very important aspect that was mistakenly neglected is related to the mode of operation of the vehicle. The robot was designed to wait for the destination input from the user, travel to that location on the shortest way and then wait for another destination to be given again one the destination is reached. This method would probably never be useful in a medical institution, because it is highly inefficient. It erroneously assumes an employee of the hospital knows where the AGV has to go after it delivered something to their location. A more sensible mode of operation would be to allow a user to send items to more than one specific place. In that case, the AGV would travel on the shortest way to the closest location among the selected ones, wait for a member of the medical staff to pick up the necessary goods and press a button to indicate that the robot could continue its trip to the next closest location. In fact, the "All Wards" option displayed on the main menu of the LCD was supposed to be exactly that, a mode in which the AGV would travel to multiple location without waiting for new input from the user at every stop along the way. Again, due to time limitations, it was impossible for us to write this function.

Considering the paragraph above, for scheduled deliveries with large quantities to multiple locations, such as lunch delivery to patient wards, the work flow could be formulated as shown in Fig. 51. The steps are explained below.

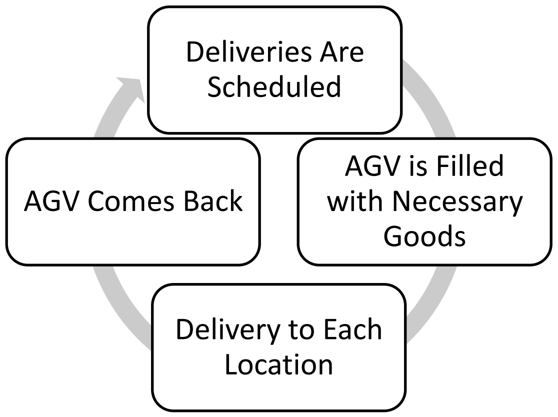


Fig. 51: Correct Work Flow for the AGV

1. Authorized personnel schedules a delivery, also indicating the type of good to be delivered as well as the delivery locations.
2. Based on the selected good, the AGV first goes to appropriate location and waits to be loaded.
3. Once the member of the personnel approves that the good is loaded into the AGV, it starts delivering the good to each selected location. Each time the AGV arrives to one of the selected locations, it waits for the command to continue delivering.
4. Once the delivery to selected locations is completed, the AGV returns to the charging station.

The infrared sensors used in the robot had a short range of detection for black and white surfaces, as a result they were placed very close to the ground which is not practical in case of real application because a clearance space is required below the robot, thus these sensors should be replaced with sensors having larger range. Furthermore, the precision of line following can be easily improved by using more IR sensors, which will provide more combinations as conditions for position adjustment. Due to unexpected development, the IR sensors were powered through a separate power supply but for future prospects the sensors should be connected to the main battery and the supply to them should be maintained at a constant voltage level through the use of a linear voltage regulator or a switching regulator.

Similar to the IR sensors, the RFID reader had a limited range and thus it was placed very close to the ground to read the RFID tags and the limited range also resulted in the robot sometimes missing the RFID tags, which affected the implementation of shortest path algorithm. Therefore, an RFID reader of larger range should be used instead.

7. Conclusion

After the research made on the logistic necessities of healthcare in Denmark and abroad, we can conclude that there is not a single answer for our initiating problem. Not only because there are different ways of optimizing logistics in a hospital, but because the needs of every medical institution vary from one another depending on their specific characteristics (location, size, internal regulations, types of patients treated, etc.). It is indeed very difficult to make a device that can adapt to most of those necessities, be attractive from a commercial point of view and not entail any alteration of the building structure. Our idea for this project was to make something as general as possible so it can be useful for the widest variety of cases. Apart from the conceptual idea behind it, the actual implementation of the proof of concept has also been troublesome: we merged different technologies in a single device, and that entailed problems. For example, the security system needs an accelerometer instead of a tilt sensor in order to be really theft proof, the IR sensors seem to have an unstable behavior and we could not manage to make the wireless communication module work. The user interface, the algorithms for line-following and finding the shortest way and the wireless charging work flawlessly though. Even if the behavior of the robot is far from perfect and some of our initial ideas could not be included, the AGV is able to carry items from A to B in a simple and efficient way.

References

[1] Healthcare Denmark, “Sustainable Hospitals - Hospital Logistics,” p. 28, 2017.

[2] A. G. Özkil, Z. Fan, S. Dwids, H. Aanaæs, J. K. Kristensen, and K. H. Christensen, “Service robots for hospitals: A case study of transportation tasks in a hospital,” *Proc. 2009 IEEE Int. Conf. Autom. Logist. ICAL 2009*, no. August, pp. 289–294, 2009.

[3] Bruce F. Field and Joseph G. Kasper, “Automated Guided Vehicle | United States Patent,” 1991.

[4] “Robots para transportar ropa, medicinas y comida en el HUCA. El Comercio.” [Online]. Available: http://www.elcomercio.es/v/20140202/asturias/robots-para-transportar-ropa-20140202.html. [Accessed: 19-Feb-2018].

[5] “Una docena de robots reparten las bandejas de comida entre las habitaciones de los pacientes.” [Online]. Available: http://www.abc.es/espana/abci-docena-robots-reparten-bandejas-5723284799001-20180131013002_video.html. [Accessed: 19-Feb-2018].

[6] “Los robots del Hospital Central «son una trampa».” [Online]. Available: http://www.elcomercio.es/asturias/201610/18/robots-hospital-central-trampa-20161018003533-v.html. [Accessed: 19-Feb-2018].

[7] “Hospital de España usa un ‘robot farmacéutico’ para dispensa.” [Online]. Available: https://miradaprofesional.com/ampliarpagina.php?id=5159. [Accessed: 19-Feb-2018].

[8] Andrew Meola, “Amazon, Domino’s and the future drone delivery market - Business Insider,” 2017. [Online]. Available: http://www.businessinsider.com/delivery-drones-market-service-2017-7?r=US&IR=T&IR=T. [Accessed: 27-Feb-2018].

[9] “Medical drones poised to take off - For Medical Professionals - Mayo Clinic,” 2015. [Online]. Available: https://www.mayoclinic.org/medical-professionals/clinical-updates/trauma/medical-drones-poised-to-take-off. [Accessed: 27-Feb-2018].

[10] James Vincent, “Swiss hospitals will start using drones to exchange lab samples - The Verge,” 2017. [Online]. Available: https://www.theverge.com/2017/3/31/15135036/drone-hospital-laboratory-delivery-swiss-post-lugano. [Accessed: 27-Feb-2018].

[11] L. Josephs, “A NASA study found drones sound more annoying to people than cars or trucks, and flying them higher won’t help — Quartz,” 2017. [Online]. Available: https://qz.com/1033675/a-nasa-study-found-drones-sound-more-annoying-to-people-than-cars-or-trucks-and-flying-them-higher-wont-help/. [Accessed: 27-Feb-2018].

[12] A. Christian and R. Cabell, “Initial Investigation into the Psychoacoustic Properties of Small Unmanned Aerial System Noise.”

[13] Suman Kumar Das, “Design and Methodology of Line Follower Automated Guided Vehicle-A Review,” *IJSTE - Int. J. Sci. Technol. Eng.*, vol. 2, no. 10, 2016.

[14] “TransCar® Automated Guided Vehicle | Swisslog.” [Online]. Available: https://www.swisslog.com/en-us/healthcare/products/material-transport/transcar-automated-guided-vehicle. [Accessed: 20-May-2018].

[15] Ecophon, “Impact of noise in healthcare Healthcare facts,” 2017.

[16] “The Low Voltage Directive (LVD) - European Commission.” [Online]. Available: http://ec.europa.eu/growth/sectors/electrical-engineering/lvd-directive_en. [Accessed: 20-May-2018].

[17] “Machinery - European Commission.” [Online]. Available: http://ec.europa.eu/growth/sectors/mechanical-engineering/machinery_en. [Accessed: 20-May-2018].

[18] “Arduino Mega 2560 Datasheet Overview.”

[19] "Arduino Due." [Online]. Available: https://store.arduino.cc/usa/arduino-due. [Accessed: 25-Apr-2018].
[20] "Arduino Playground - ArduinoPinCurrentLimitations." [Online]. Available: https://playground.arduino.cc/Main/ArduinoPinCurrentLimitations. [Accessed: 13-May-2018].
[21] Patrick J. Sweeney, "Examining the Elements of a Basic RFID System." [Online]. Available: http://www.dummies.com/education/science/science-electronics/examining-the-elements-of-a-basic-rfid-system/. [Accessed: 15-Apr-2018].
[22] Gavin Phillips, "How Does RFID Technology Work?," 2017. [Online]. Available: https://www.makeuseof.com/tag/technology-explained-how-do-rfid-tags-work/. [Accessed: 15-Apr-2018].
[23] Anil Pandey, "How do RFID tags and reader antennas work?," 2017. [Online]. Available: https://www.analogictips.com/rfid-tag-and-reader-antennas/. [Accessed: 15-Apr-2018].
[24] Ron Lessnick, "(4) What is RF current? - Quora," 2017. [Online]. Available: https://www.quora.com/What-is-RF-current. [Accessed: 15-Apr-2018].
[25] Suzanne Smiley, "RFID Tag Antennas - RFID Insider," 2017. [Online]. Available: https://blog.atlasrfidstore.com/rfid-tag-antennas. [Accessed: 15-Apr-2018].
[26] "RFID System Components." [Online]. Available: http://fireflyrfidsolutions.com/firefly-university/what-is-rfid-copy/index.html. [Accessed: 15-Apr-2018].
[27] N. Semiconductors, "MFRC522 Standard performance MIFARE and NTAG frontend."
[28] "MFRC522."
[29] "Working Principle of Photodiode, Characteristics And Applications." [Online]. Available: https://www.elprocus.com/photodiode-working-principle-applications/. [Accessed: 19-May-2018].
[30] "Op-amp Comparator and the Op-amp Comparator Circuit." [Online]. Available: https://www.electronics-tutorials.ws/opamp/op-amp-comparator.html. [Accessed: 19-May-2018].
[31] Vishwam, "How to build an IR Sensor," *Http://Maxembedded.Com/*, 2013. [Online]. Available: http://maxembedded.com/2013/08/how-to-build-an-ir-sensor/. [Accessed: 19-May-2018].
[32] "IR Sensor Circuit and Working with Applications." [Online]. Available: https://www.elprocus.com/infrared-ir-sensor-circuit-and-working/. [Accessed: 19-May-2018].
[33] "IR Infrared 2 - 30cm Obstacle Detaction Sensor Module FC-51 | QQ Online Trading." [Online]. Available: http://qqtrading.com.my/ir-infrared-obstacle-detaction-sensor-module-fc-5. [Accessed: 27-May-2018].
[34] Krishna Pattabiraman, "How to Set Up a Keypad on an Arduino - Circuit Basics," 2017. [Online]. Available: http://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/. [Accessed: 15-Apr-2018].
[35] JIMB0, "Switch Basics - learn.sparkfun.com." [Online]. Available: https://learn.sparkfun.com/tutorials/switch-basics. [Accessed: 15-Apr-2018].
[36] G. Lazaridis, "How a Key Matrix Works | PCBheaven," 2010. [Online]. Available: http://pcbheaven.com/wikipages/How_Key_Matrices_Works/. [Accessed: 15-Apr-2018].
[37] M. Margolis, *Arduino Cookbook*, First Edit. O'Reilly Media, Inc., 2011.
[38] C. Andrews, "Arduino Playground - Keypad Library," 2015. [Online]. Available: https://playground.arduino.cc/Code/Keypad. [Accessed: 16-Apr-2018].
[39] RS Components Ltd., "PKA-1606F | Devlin 16 Key Keypad | Devlin." [Online]. Available: https://uk.rs-online.com/web/p/keypads/0331304/. [Accessed: 16-Apr-2018].
[40] "LCD2004 IIC/I2C Blue Backlight - ArduinoTech.dk," 2017. [Online]. Available: https://arduinotech.dk/shop/lcd2004-iici2c-blue-backlight/. [Accessed: 21-Apr-2018].
[41] "Robot Platform | Knowledge | Types of Robot Wheels." [Online]. Available: http://www.robotplatform.com/knowledge/Classification_of_Robots/Types_of_robot_wheels.html. [Accessed: 21-Apr-2018].

[42] Sarah Jensen, “MIT School of Engineering | » What’s the difference between a motor and an engine?,” 2013. [Online]. Available: https://engineering.mit.edu/engage/ask-an-engineer/whats-the-difference-between-a-motor-and-an-engine/. [Accessed: 17-May-2018].
[43] “Chapter 10: Faraday’s Law of Induction | MIT,” .
[44] taifur, “Complete Motor Guide for Robotics,” 2015. [Online]. Available: http://www.instructables.com/id/Complete-Motor-Guide-for-Robotics/. [Accessed: 17-May-2018].
[45] B. Coleman, “Drive Motor Sizing Tutorial - RobotShop Blog,” 2012. [Online]. Available: https://www.robotshop.com/blog/en/drive-motor-sizing-tutorial-3661. [Accessed: 17-Apr-2018].
[46] M. Page, “D.C. Motor Torque/Speed Curve Tutorial:::Understanding Motor Characteristics,” 2007. [Online]. Available: http://lancet.mit.edu/motors/motors3.html#torque. [Accessed: 17-Apr-2018].
[47] A. Neal, “Tips For Selecting DC motors for your mobile robot,” *Servo Mag.*, pp. 33–37, 2010.
[48] oddWires, “JGA25-370 6V DC Gear Motor 77 RPM (Use From 3V to 9V) - oddWires.” [Online]. Available: http://www.oddwires.com/jga25-370-6v-dc-gear-motor-77-rpm-use-from-3v-to-9v/. [Accessed: 17-Apr-2018].
[49] AliExpress, “JGA25 370 Gear Motor Intelligent Robot/Car Motor (Metal Gear / Little Noise / Much Selection) Aliexpress.com | Alibaba Group.” [Online]. Available: https://www.aliexpress.com/item/JGA25-370-DC-gear-motor-for-intelligent-robot-car-motor-metal-gear-little-noise-much-selection/915695702.html. [Accessed: 18-Apr-2018].
[50] “H-Bridges – the Basics | Modular Circuits.” [Online]. Available: http://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridges-the-basics/. [Accessed: 16-May-2018].
[51] “Sign-Magnitude Drive | Modular Circuits.” [Online]. Available: http://modularcircuits.tantosonline.com/blog/articles/h-bridge-secrets/sign-magnitude-drive/. [Accessed: 16-May-2018].
[52] “Motor driver | Electrical tutorials | Mepits | Mepits.” [Online]. Available: https://www.mepits.com/tutorial/379/electrical/motor-driver. [Accessed: 16-May-2018].
[53] “Get Your Motor Running – How to select a motor driver - Motor Drive &amp; Control - Blogs - TI E2E Community.” [Online]. Available: https://e2e.ti.com/blogs_/b/motordrivecontrol/archive/2013/04/17/get-your-motor-running-how-to-select-a-motor-driver. [Accessed: 16-May-2018].
[54] “What is Pulse-Width Modulation (PWM)?” [Online]. Available: https://www.analogictips.com/pulse-width-modulation-pwm/. [Accessed: 15-May-2018].
[55] “Pulse Width Modulation - learn.sparkfun.com.” [Online]. Available: https://learn.sparkfun.com/tutorials/pulse-width-modulation. [Accessed: 15-May-2018].
[56] “Pulse Width Modulation Used for Motor Control.” [Online]. Available: https://www.electronics-tutorials.ws/blog/pulse-width-modulation.html. [Accessed: 15-May-2018].
[57] “Pulse Width Modulation | DC Motor Drives | Electronics Textbook.” [Online]. Available: https://www.allaboutcircuits.com/textbook/semiconductors/chpt-11/pulse-width-modulation/. [Accessed: 15-May-2018].
[58] “What is an Ultrasonic Sensor?” [Online]. Available: http://education.rec.ri.cmu.edu/content/electronics/boe/ultrasonic_sensor/1.html. [Accessed: 20-May-2018].
[59] “Complete Guide for Ultrasonic Sensor HC-SR04 with Arduino | Random Nerd Tutorials.” [Online]. Available: https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/. [Accessed: 20-May-2018].
[60] “How To Make A Tilt Sensor With Arduino?” [Online]. Available: https://www.electronicshub.org/arduino-tilt-sensor/. [Accessed: 21-May-2018].

[61] “Tilt Sensor SW-520D.” [Online]. Available: https://www.rhydolabz.com/sensors-direction-sensors-c-137_145/tilt-sensor-sw520d-p-2264.html. [Accessed: 21-May-2018].
[62] “Piezo buzzer RS 7800731.” [Online]. Available: https://uk.rs-online.com/web/p/piezo-buzzer-components/7800731/. [Accessed: 21-May-2018].
[63] lady ada, “Powering Motors | Adafruit Motor Shield V2 for Arduino | Adafruit Learning System.” [Online]. Available: https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino/powering-motors. [Accessed: 13-May-2018].
[64] JORDANDEE, “How to Power a Project - learn.sparkfun.com.” [Online]. Available: https://learn.sparkfun.com/tutorials/how-to-power-a-project. [Accessed: 13-May-2018].
[65] “What’s the Best Battery? | Battery University,” 2010. [Online]. Available: http://batteryuniversity.com/learn/archive/whats_the_best_battery. [Accessed: 13-May-2018].
[66] NATE, “Battery Technologies - learn.sparkfun.com.” [Online]. Available: https://learn.sparkfun.com/tutorials/battery-technologies. [Accessed: 13-May-2018].
[67] “UL2.4-12 Dimensions F1 Terminal.”
[68] Bill Schweber, “Understanding Linear Regulator Advantages | DigiKey,” 2017. [Online]. Available: https://www.digikey.com/en/articles/techzone/2017/sep/understanding-the-advantages-and-disadvantages-of-linear-regulators. [Accessed: 29-Apr-2018].
[69] Texas Instruments Incorporated, “Switching Regulator Fundamentals,” 2016.
[70] “XL4015 LM2596 5A DC-DC Adjustable Voltage Step Down - ArduinoTech.dk.” [Online]. Available: https://arduinotech.dk/shop/xl4015-lm2596-5a-dc-dc-adjustable-voltage-step-down/. [Accessed: 29-Apr-2018].
[71] “How batteries can explode - and how to avoid it - Yachting Monthly.” [Online]. Available: http://www.yachtingmonthly.com/archive/how-batteries-can-explode-and-how-to-avoid-it-3930. [Accessed: 19-May-2018].
[72] “Arduino Playground - OneWire.” [Online]. Available: https://playground.arduino.cc/Learning/OneWire. [Accessed: 17-May-2018].
[73] “DS18B20 datasheet.” [Online]. Available: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf. [Accessed: 17-May-2018].
[74] A. Bettini, *A Course in Classical Physics 3 - Electromagnetism*. 2016.
[75] “Wireless Charge Module DS.” [Online]. Available: http://elecfreaks.com/store/download/User-specified.pdf. [Accessed: 20-May-2018].
[76] “Wireless Charge Module - Datasheet.” [Online]. Available: http://elecfreaks.com/store/download/User-specified.pdf. [Accessed: 23-May-2018].
[77] “XL6009 DC-DC Adjustable Step-up Power Converter - ArduinoTech.dk.” [Online]. Available: https://arduinotech.dk/shop/xl6009-dc-dc-adjustable-step-power-converter/. [Accessed: 15-May-2018].
[78] “Arduino Line Follower Robot Code and Circuit Diagram.” [Online]. Available: https://circuitdigest.com/microcontroller-projects/line-follower-robot-using-arduino. [Accessed: 12-May-2018].

Appendices

Appendix 1. Complete Arduino Code

```
// LIBRARIES
#include < DallasTemperature.h >
#include < OneWire.h >
#include < Key.h >
#include < Keypad.h >
#include < LiquidCrystal_I2C.h >
#include < MFRC522.h >
// DEFINATIONS
#define serial_output 2
#define SENSOR_PIN4 33
#define outPin 34
#define N 4
#define SS_PIN 48
#define RST_PIN 53
#define LED5 5
#define LED2 2
#define LED6 6
#define LED4 4
// VARIABLES
// Temperature sensor
float temp;
int Alarm = 0;
OneWire oneWire(serial_output);
// Pass our data to convert it into celsius temperature
DallasTemperature sensors( & oneWire);
// Buzzer and ultrasonic sensors
const int echoPin = 29;
const int trigPin = 28;
long duration;
int distance;
// IR sensors
// A is the right motor and B is the left motor
const int dir_A = 12;
const int dir_B = 13;
const int brake_A = 9;
const int brake_B = 8;
const int PWM_A = 3;
const int PWM_B = 11;
// Above 145, all readings classify as black colour
const int upperthreshold = 145;
// Below 30, all readings classify as white colour
const int lowerthreshold = 30;
int mappedvalue1 = 0; // sensor number one
int mappedvalue2 = 0; // sensor number two
int mappedvalue3 = 0; // sensor number three
int mappedvalue4 = 0; // sensor number four
int mappedvalue5 = 0; // sensor number five
int breakvar = 1;
const int trigPin1 = 22;
const int echoPin1 = 23;
const int trigPin2 = 24;
const int echoPin2 = 25;
const int trigPin3 = 26;
const int echoPin3 = 27;
int detectDist = 20;
int distance1;
int distance2;
int distance3;
int firstWhite = 0;
int flag = 0;
// LCD and keypad
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {{'1', '2', '3', 'A'}, {'4', '5', '6', 'B'}, {'7',
 '8', '9', 'C'}, {'/', '0', '.', 'D'}};
byte rowpins[ROWS] = {45, 43, 41, 39};
byte colpins[COLS] = {44, 42, 40, 38};
Keypad kp = Keypad(makeKeymap(keys), rowpins, colpins, ROWS, COLS);
LiquidCrystal_I2C lcd(0x27, 20, 4);
// Stores the PIN that allows further interaction with the robot
const char pin[N] = {'1', '2', '3', '4'};
//Store the current location of the cursor
int cursrow;
int curscol;
//When 1, it indicates the movement of the cursor from OneRoomMenu1 to On-
eRoomMenu2
int pagechange = 0;
// RFID and shortest way
MFRC522 mfrc522(SS_PIN, RST_PIN);
// Srores xy coordinates of starting point and the destination
int route[2] = {34, 00};
// Stores the x and y values of the both starting point and destina-
tion separately
int myPoints[2][2] = {{0, 0}, {0, 0}};
```

```
// Stores the decided-direction of the ro-
bot.(1=left, 2=up, 3=right, 4=down)
// Initialized as 0, values will change in the program
int setDirection[2] = {0, 0};
// Stores present location xy coordinates based on RFID readings
int presentLocation = 34;
// Stores present location xy coordinates separately
int presentLocationMatrix[2] = {3, 4}; //
int realPresentLocationMatrix[2] = {3, 4};
// Stores destination xy coordinates separately
int destinationLocationMatrix[2] = {0, 0};
// While finding the shortest way, this variable stores temporary xy coor-
dinates of nearby RFID tags, when needed
int tempLocationMatrix[2] = {0, 0};
// Store common directions between decided-directions and available direc-
tions from the present node
int commonDirections[2] = {0, 0};
// Stores calculated coordinates of a particular point
int coordinates[2] = {0, 0};
// Stores instant direction of the robot.(1=left, 2=up, 3=right, 4=down)
int instantDirection = 3;
        //stores available directions possi-
ble from each RFID tag. (0=not available,1=left, 2=up, 3=right, 4=down)
        const int Nodes[30][4] = {{0,2,0,4},{0,2,3,4},{0,2,3,4},{0,2,0,4},
{0,2,3,0},{1,2,0,0},{0,2,0,4},{1,2,3,4},{1,0,3,0},{1,0,3,0},{1,2,3,4},{0,2
,0,4},{0,2,3,0},{1,2,0,0},{0,2,0,4},{1,2,0,4},{1,0,0,4},{1,0,3,4},{1,0,3,0
},{1,0,3,0},{1,2,3,4},{0,2,0,4},{1,2,0,4},{1,0,0,4},{1,0,3,4},{1,0,3,0},{0
,0,3,4},{0,2,3,0},{1,0,3,0},{1,2,3,4}};
         // Stores the index of the particular RFID tag in the ar-
ray called Nodes
        const int index[88]=
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 26, 27, 0, 0, 0, 0, 0, 0, 0, 0, 25, 28,
0, 0, 0, 0, 0, 0, 0, 0, 24, 29, 0, 1, 2, 3, 4, 0, 0, 0, 23, 22, 21, 20, 7,
 6, 5, 0, 0, 0, 0, 0, 0, 19, 8, 0, 0, 0, 0, 0, 0, 0, 0, 18, 9, 0, 0, 0, 0,
 0, 0, 0, 0, 17, 10, 11, 12, 0, 0, 0, 0, 0, 0, 16, 15, 14, 13};
        // Stores the found shortest way to the destina-
tion, in terms of xy coordinates.
        int shortestWay[15][2] =
{{3,4},{9,9},{9,9},{9,9},{9,9},{9,9},{9,9},{9,9},{9,9},{9,9},{9,9},{9,9},{
9,9},{9,9},{9,9}};
        int y = 1; //used as an index for array called shortestWay
        int t = 0; //used to control a while loop
        // Stores xy coordinates of each RFID tag
        const int rooms[30][2] = {{1,1},{1,2},{2,1},{2,2},{3,1},{3,2},{3,3
},{3,4},{3,5},{3,6},{3,7},{4,1},{4,2},{4,3},{4,4},{4,5},{4,6},{4,7},{5,4},
{5,5},{6,4},{6,5},{7,4},{7,5},{7,6},{7,7},{8,4},{8,5},{8,6},{8,7}};
        // Stores xy coordinates of all intersection points
        const int InterSetions[18][2] =
{{1,1},{1,2},{3,1},{3,2},{3,7},{4,7},{4,1},{4,2},{3,4},{3,5},{4,4},{4,5},{
7,4},{7,5},{7,7},{8,7},{8,4},{8,5}};
        // Stores the distance to a particular intersection point
        int closeness[18] = {0};
        // Stores the closest intersection point to the present location
        int closestIntersection[2] = {0};
        // FUNCTION PROTOTYPES
        // LCD and keypad
        //displays the blinking cursor to a desired loca-
tion and gets its coordinates
        void showCursor(int, int);
        //shows greetings menu
        void greetings();
        //clears the screen, sets the curosor to (0,0) and deac-
tivates the blinking cursor
        void clearScreen();
        //compares the pin introduced by the user with const char pin[N]
        void checkPin(char[], int);
        //selects an option when 'C' is pressed
        bool select(char);
        //goes to the previous menu when 'D' is prssed
        bool back(char);
        //asks the user to introduce the PIN
        int askPin(char[], int);
        /** "Go to" Functions **/
        /* Give the coordi-
nates of the room(s) the user sends the AVG to. * The func-
tion names are self-explaatory.*/
        void goStart();
        void allWards();
        void canteen();
        void pharmacy();
        void laundry();
        void ward1();
        void ward2();
        void ward3();
        void ward4();
        void ward5();
        /** Menues **/
        /* There are 3 menues in to-
tal: Main Menu and One Room Menu(this is the menu  * that ap-
pears when the user selects the One Room des-
```

```
tianation in the  * Main Menu and it com-
prises two pags that have to be treated as individ-
ual  * menues. Hence, we can say there are 3 mnues in total). */
       /* Display the menues. Each menu function calls the correspond-
ing navigate  * function.*/
       void mainMenu();
       void oneRoomMenu1();
       void oneRoomMenu2();
       /* Allow navigation and selection in the menues. Each navi-
gate function calls  * the corresponding moveCursor function.*/
       void navigateMainMenu(char);
       void navigateOneRoomMenu1(char);
       void navigateOneRoomMenu2(char);
       /* Allow up/down cursor movement in the menues */
       void moveCursorMainMenu(char);
       void moveCursorOneRoomMenu1(char);
       void moveCursorOneRoomMenu2(char);
       // IR sensors
       int mapValue(int); // Maps the value of sensors to 255
       void getSensorReadings(); // Gets the reading of the IR sensor
       void lineFollowing(); // Tests the condition for follow-
ing the black line
       void motorRun(char, char, int);
       // Runs the motor based on the motor ,direc-
tion ,and speed given as arguments
       void motorMovement(); // Calls the function "interSec-
tion" and "lineFollowing"
       void interSection(); // Tests the condition when the ro-
bot sees an intersection
       void turnRight90(); // Turns the robot by 90 degree to the right
       void turnLeft90(); // Turns the robot by 90 de-
gree to the left // RFID and shortest way
       void sendDestinationInfo(); // Sends destination point to the mon-
itoring screen
       void decideDirection(); // Decides the next move-
ment(left, right, forward, back) and executes it
       void decideNextNode(); // Decides the next loca-
tion the AGV should traverse to arrive to the destinaiton in the short-
est possible way
       void goToNextNode(int); // Updates the present location to the de-
cided next node
       void assumeNextNode(int); // Temporarly assumes the robot trav-
eled to a particular location by updating tempLocationMatrix
       int calculateDistance(); // Calculates distance be-
tween two points
       void FindClosestIntersection(); // Finds the closest intersec-
tion to the present location
       void findDirection(); // Finds the necessary direc-
tions to be taken to arrive to destination in the shortest way possible
       void findCoordinates(int[]); // Calculates the coordi-
nates of a given point
       void turnOffLEDs(); // Turns off the LEDs // Temperature Sensor
       void temperature(); // Sends tmperature values to monitor-
ing screen // Buzzer and ultrasonic sensors
       void security(); // Checks particular sensor readings and de-
tects anomalies, if any
       void buzzer(); // Beeps the buzzer
       void obstacleDetection(); // Includes functions for detecting ob-
stacle
       void getDistance(); // Includes functions for getting read-
ings from ultrasonic sensors
       int getReading_US(int, int); // Gets readings for each ultra-
sonic sensor
       void detectionCondition(); // Checks if the distance to the ob-
ject is critical
       void ultraSonic(int, int); // Calculates the distance to the ob-
ject in front
       void setup() {
       // Buzzer and ultrasonic sensors
           pinMode(SENSOR_PIN4, INPUT_PULLUP);
           pinMode(echoPin, INPUT);
           pinMode(trigPin, OUTPUT);
           pinMode(outPin, OUTPUT);
       // Temperature sensor
           sensors.begin();
       // IR sensors
           pinMode(dir_A, OUTPUT);
           pinMode(dir_B, OUTPUT);
           pinMode(brake_A, OUTPUT);
           pinMode(brake_B, OUTPUT);
           pinMode(trigPin1, OUTPUT);
           pinMode(echoPin1, INPUT);
           pinMode(trigPin2, OUTPUT);
           pinMode(echoPin2, INPUT);
           pinMode(trigPin3, OUTPUT);
           pinMode(echoPin3, INPUT);
       // RFID and shortest way
           Serial.begin(9600);
           SPI.begin();
           mfrc522.PCD_Init();
           pinMode(LED5, OUTPUT);
```

```
            pinMode(LED2, OUTPUT);
            pinMode(LED6, OUTPUT);
            pinMode(LED4, OUTPUT);
            turnOffLEDs();
         //LCD and keypad
            lcd.init();
            lcd.backlight();
            clearScreen();
        }
        void loop() {
                sendDestinationInfo();
                char trypin[N];
                lcd.backlight();
                checkPin(trypin, N);
                mainMenu();
                clearScreen();
                sendDestinationInfo();
                while (presentLocation != route[1]) {
                    findDirection();
                    decideNextNode();
                }
                t = 0;
                while (!t) {
                    while (1) {
                        decideDirection();
                        breakvar = 1;
                        while (breakvar) {
                            motorMovement();
                            obstacleDetection();
                            security();
                        }
                        readRFID();
                        if (realPresentLocationMatrix[0] == destinationLocationMatrix[0] && realPresentLocationMatrix[1] == destinationLocationMatrix[1]) {
                            digitalWrite(LED5, HIGH);
                            digitalWrite(LED2, HIGH);
                            digitalWrite(LED6, HIGH);
                            digitalWrite(LED4, HIGH);
                            digitalWrite(brake_A, HIGH);
                            digitalWrite(brake_B, HIGH);
                            delay(1000);
                            route[1] = 00;
                            destinationLocationMatrix[0] = 0;
                            destinationLocationMatrix[1] = 0;
                            int u;
                            int c;
                            for (u = 0; u < 15; u++) {
                                for (c = 0; c < 2; c++) {
                                    shortestWay[u][c] = 9;
                                }
                            }
                            shortestWay[0][0] = presentLocationMatrix[0];
                            shortestWay[0][1] = presentLocationMatrix[1];
                            y = 1;
                            break;
                        }
                        decideDirection();
                        temperature();
                    }
                }
            }
        // FUNCTIONS
        void sendDestinationInfo() {
            switch (route[1]) {
                case 11:
                    Serial.write(111);
                    break;
                case 12:
                    Serial.write(112);
                    break;
                case 21:
                    Serial.write(121);
                    break;
                case 22:
                    Serial.write(122);
                    break;
                case 31:
                    Serial.write(131);
                    break;
                case 32:
                    Serial.write(132);
                    break;
                case 33:
                    Serial.write(133);
                    break;
                case 34:
                    Serial.write(134);
                    break;
                case 35:
```

```
                Serial.write(135);
                break;
            case 36:
                Serial.write(136);
                break;
            case 37:
                Serial.write(137);
                break;
            case 41:
                Serial.write(141);
                break;
            case 42:
                Serial.write(142);
                break;
            case 43:
                Serial.write(143);
                break;
            case 44:
                Serial.write(144);
                break;
            case 45:
                Serial.write(145);
                break;
            case 46:
                Serial.write(146);
                break;
            case 47:
                Serial.write(147);
                break;
            case 54:
                Serial.write(154);
                break;
            case 55:
                Serial.write(155);
                break;
            case 64:
                Serial.write(164);
                break;
            case 65:
                Serial.write(165);
                break;
            case 74:
                Serial.write(174);
                break;
            case 75:
                Serial.write(175);
                break;
            case 76:
                Serial.write(176);
                break;
            case 77:
                Serial.write(177);
                break;
            case 84:
                Serial.write(184);
                break;
            case 85:
                Serial.write(185);
                break;
            case 86:
                Serial.write(186);
                break;
            case 87:
                Serial.write(187);
                break;
            default:
                Serial.write(000);
                break;
        }
    }
    void decideDirection() {
        turnOffLEDs();
        if ((shortestWay[y][0] - realPresentLocationMa-
trix[0]) > 0) {
            switch (instantDirection) {
                case 3:
                    digitalWrite(LED2, HIGH);
                    forward();
                    break;
                case 2:
                    digitalWrite(LED6, HIGH);
                    goBitForward();
                    turnRight90();
                    break;
                case 1:
                    digitalWrite(LED4, HIGH);
                    goBitForward();
                    turnLeft90();
                    digitalWrite(brake_A, HIGH);
                    digitalWrite(brake_B, HIGH);
                    delay(50);
                    turnLeft90();
```

```
                        break;
                    case 4:
                        digitalWrite(LED5, HIGH);
                        goBitForward();
                        turnLeft90();
                        break;
                }
                instantDirection = 3;
            } else if ((shortestWay[y][0] - realPresentLocationMatrix[0]) < 0) {
                switch (instantDirection) {
                    case 1:
                        digitalWrite(LED2, HIGH);
                        forward();
                        break;
                    case 4:
                        digitalWrite(LED6, HIGH);
                        goBitForward();
                        turnRight90();
                        break;
                    case 3:
                        digitalWrite(LED4, HIGH);
                        goBitForward();
                        turnLeft90();
                        digitalWrite(brake_A, HIGH);
                        digitalWrite(brake_B, HIGH);
                        delay(50);
                        turnLeft90();
                        break;
                    case 2:
                        digitalWrite(LED5, HIGH);
                        goBitForward();
                        turnLeft90();
                        break;
                }
                instantDirection = 1;
            } else if ((shortestWay[y][1] - realPresentLocationMatrix[1]) > 0) {
                switch (instantDirection) {
                    case 4:
                        digitalWrite(LED2, HIGH);
                        forward();
                        break;
                    case 3:
                        digitalWrite(LED6, HIGH);
                        goBitForward();
                        turnRight90();
                        break;
                    case 2:
                        digitalWrite(LED4, HIGH);
                        goBitForward();
                        turnLeft90();
                        digitalWrite(brake_A, HIGH);
                        digitalWrite(brake_B, HIGH);
                        delay(50);
                        turnLeft90();
                        break;
                    case 1:
                        digitalWrite(LED5, HIGH);
                        goBitForward();
                        turnLeft90();
                        break;
                }
                instantDirection = 4;
            } else if ((shortestWay[y][1] - realPresentLocationMatrix[1]) < 0) {
                switch (instantDirection) {
                    case 2:
                        digitalWrite(LED2, HIGH);
                        forward();
                        break;
                    case 1:
                        digitalWrite(LED6, HIGH);
                        goBitForward();
                        turnRight90();
                        break;
                    case 4:
                        digitalWrite(LED4, HIGH);
                        goBitForward();
                        turnLeft90();
                        digitalWrite(brake_A, HIGH);
                        digitalWrite(brake_B, HIGH);
                        delay(50);
                        turnLeft90();
                        break;
                    case 3:
                        digitalWrite(LED5, HIGH);
                        goBitForward();
                        turnLeft90();
                        break;
                }
                instantDirection = 2;
```

```
        }
    }
    void decideNextNode() {
        int i;
        int xp = 0;
        int d = index[presentLocation];
        for (i = 0; i < 4; i++) {
            int m;
            for (m = 0; m < 2; m++) {
                if (Nodes[d][i] == setDirection[m]) {
                    commonDirections[xp] = Nodes[d][i];
                    xp++;
                }
            }
        }
        if (xp > 1) {
            int distance[2];
            int m;
            int op;
            for (m = 0; m < xp; m++) {
                assumeNextNode(commonDirections[m]);
                distance[m] = calculateDistance();
                if ((tempLocationMatrix[0] == 3 && tempLocationMatrix[1] == 1) || (tempLocationMatrix[0] == 3 && tempLocationMatrix[1] == 2) || (tempLocationMatrix[0] == 4 && tempLocationMatrix[1] == 1) || (tempLocationMatrix[0] == 4 && tempLocationMatrix[1] == 2) || (tempLocationMatrix[0] == 3 && tempLocationMatrix[1] == 4) || (tempLocationMatrix[0] == 3 && tempLocationMatrix[1] == 5) || (tempLocationMatrix[0] == 4 && tempLocationMatrix[1] == 4) || (tempLocationMatrix[0] == 4 && tempLocationMatrix[1] == 5) || (tempLocationMatrix[0] == 7 && tempLocationMatrix[1] == 4) || (tempLocationMatrix[0] == 7 && tempLocationMatrix[1] == 5) || (tempLocationMatrix[0] == 8 && tempLocationMatrix[1] == 4) || (tempLocationMatrix[0] == 8 && tempLocationMatrix[1] == 5) || (tempLocationMatrix[0] == 1 && tempLocationMatrix[1] == 1) || (tempLocationMatrix[0] == 1 && tempLocationMatrix[1] == 2) || (tempLocationMatrix[0] == 3 && tempLocationMatrix[1] == 7) || (tempLocationMatrix[0] == 4 && tempLocationMatrix[1] == 7) || (tempLocationMatrix[0] == 7 && tempLocationMatrix[1] == 7) || (tempLocationMatrix[0] == 8 && tempLocationMatrix[1] == 7)) {
                    op = m;
                }
            }
            if (distance[0] > distance[1]) {
                commonDirections[0] = commonDirections[1];
                commonDirections[1] = 0;
            } else if (distance[0] < distance[1]) {
                commonDirections[0] = commonDirections[0];
                commonDirections[1] = 0;
            } else if (distance[0] == distance[1]) {
                commonDirections[0] = commonDirections[op];
                commonDirections[1] = 0;
            }
        }
        goToNextNode(commonDirections[0]);
    }
    void goToNextNode(int m) {
        switch (m) {
            case 1:
                presentLocationMatrix[0] = presentLocationMatrix[0] - 1;
                presentLocationMatrix[1] = presentLocationMatrix[1];
                break;
            case 2:
                presentLocationMatrix[0] = presentLocationMatrix[0];
                presentLocationMatrix[1] = presentLocationMatrix[1] - 1;
                break;
            case 3:
                presentLocationMatrix[0] = presentLocationMatrix[0] + 1;
                presentLocationMatrix[1] = presentLocationMatrix[1];
                break;
            case 4:
                presentLocationMatrix[0] = presentLocationMatrix[0];
                presentLocationMatrix[1] = presentLocationMatrix[1] + 1;
                break;
        }
        presentLocation = (presentLocationMatrix[0] * 10) + (presentLocationMatrix[1]);
        int next;
        int mz;
        for (mz = 0; mz < 15; mz++) {
            if (shortestWay[mz][0] == 9) {
                next = mz;
                break;
            }
        }
        shortestWay[next][0] = presentLocationMatrix[0];
        shortestWay[next][1] = presentLocationMatrix[1];
        commonDirections[0] = 0;
        commonDirections[1] = 0;
        route[0] = presentLocation;
        myPoints[0][0] = 0;
        myPoints[0][1] = 0;
        myPoints[1][0] = 0;
        myPoints[1][1] = 0;
    }
    void assumeNextNode(int m) {
        switch (m) {
            case 1:
                tempLocationMatrix[0] = presentLocationMatrix[0] - 1;
                tempLocationMatrix[1] = presentLocationMatrix[1];
                break;
            case 2:
                tempLocationMatrix[0] = presentLocationMatrix[0];
                tempLocationMatrix[1] = presentLocationMatrix[1] - 1;
                break;
            case 3:
                tempLocationMatrix[0] = presentLocationMatrix[0] + 1;
                tempLocationMatrix[1] = presentLocationMatrix[1];
                break;
            case 4:
                tempLocationMatrix[0] = presentLocationMatrix[0];
                tempLocationMatrix[1] = presentLocationMatrix[1] + 1;
                break;
        }
    }
    int calculateDistance() {
        int distance;
        int x[2];
        int y[2];
        findCoordinates(tempLocationMatrix);
        x[0] = coordinates[0];
        y[0] = coordinates[1];
        findCoordinates(destinationLocationMatrix);
        x[1] = coordinates[0];
        y[1] = coordinates[1];
        distance = abs(x[0] - x[1]) + abs(y[0] - y[1]);
        return distance;
    }
    void FindClosestIntersection() {
        int j;
        for (j = 0; j < 18; j++) {
            closeness[j] = abs(myPoints[0][0] - InterSections[j][0]) + abs(myPoints[0][1] - InterSections[j][1]);
        }
        int index = 0;
        int i;
        for (i = 1; i < 18; i++) {
            if (closeness[i] < closeness[index]) {
                index = i;
            }
        }
        closestIntersection[0] = InterSections[index][0];
        closestIntersection[1] = InterSections[index][1];
    }
    void findDirection() {
        int f;
        for (f = 0; f < 2; f++) {
            int x;
            x = route[f];
            while (!((x >= 0) && (x < 10))) {
                x -= 10;
            }
            myPoints[f][1] = x;
            int y = (route[f] - x) / 10;
            myPoints[f][0] = y;
        }
        if (((myPoints[0][1] == 6) || (myPoints[0][1] == 7)) && ((myPoints[1][1] == 6) || (myPoints[1][1] == 7)) && (abs(myPoints[0][0] - myPoints[1][0]) > 2)) {
            if ((myPoints[0][0] - myPoints[1][0]) > 0) {
                setDirection[0] = 1;
            } else {
                setDirection[0] = 3;
            }
            setDirection[1] = 2;
        } else {
            if ((myPoints[0][0] - myPoints[1][0]) > 0) {
                setDirection[0] = 1;
            } else if ((myPoints[0][0] - myPoints[1][0]) < 0) {
                setDirection[0] = 3;
            } else {
                FindClosestIntersection();
                if ((myPoints[0][0] - closestIntersection[0]) > 0) {
                    setDirection[0] = 1;
                    } else if ((myPoints[0][0] - closestIntersec-
tion[0]) < 0) {
                        setDirection[0] = 3;
                    } else {
                        setDirection[0] = 5;
                    }
                }
                if ((myPoints[0][1] - myPoints[1][1]) > 0) {
                    setDirection[1] = 2;
                } else if ((myPoints[0][1] - myPoints[1][1]) < 0) {
                    setDirection[1] = 4;
                } else {
                    FindClosestIntersection();
                    if ((myPoints[0][1] - closestIntersection[1]) > 0) {
                        setDirection[1] = 2;
                    } else if ((myPoints[0][1] - closestIntersec-
tion[1]) < 0) {
                        setDirection[1] = 4;
                    } else {
                        setDirection[1] = 5;
                    }
                }
            }
        }
        void findCoordinates(int matrix[]) {
            coordinates[0] = matrix[0] * 100;
            coordinates[1] = matrix[1] * 100;
        }
        void goStart() {
            route[1] = 34;
            destinationLocationMatrix[0] = 3;
            destinationLocationMatrix[1] = 4;
        }
        void canteen() {
            route[1] = 36;
            destinationLocationMatrix[0] = 3;
            destinationLocationMatrix[1] = 6;
        }
        void pharmacy() {
            route[1] = 74;
            destinationLocationMatrix[0] = 7;
            destinationLocationMatrix[1] = 4;
        }
        void laundry() {
            route[1] = 42;
            destinationLocationMatrix[0] = 4;
            destinationLocationMatrix[1] = 2;
        }
        void ward1() {
            route[1] = 44;
            destinationLocationMatrix[0] = 4;
            destinationLocationMatrix[1] = 4;
        }
        void ward2() {
            route[1] = 45;
            destinationLocationMatrix[0] = 4;
            destinationLocationMatrix[1] = 5;
        }
        void ward3() {
            route[1] = 36;
            destinationLocationMatrix[0] = 3;
            destinationLocationMatrix[1] = 6;
        }
        void ward4() {
            route[1] = 47;
            destinationLocationMatrix[0] = 4;
            destinationLocationMatrix[1] = 7;
        }
        void ward5() {
            route[1] = 37;
            destinationLocationMatrix[0] = 3;
            destinationLocationMatrix[1] = 7;
        }
        void mainMenu() {
            char key;
            lcd.print("Select Destination:");
            lcd.setCursor(1, 1);
            lcd.print("Charging Station");
            lcd.setCursor(1, 2);
            lcd.print("All Wards");
            lcd.setCursor(1, 3);
            lcd.print("One Room");
            lcd.setCursor(17, 3);
            lcd.print("1/1");
            showCursor(0, 1);
            navigateMainMenu(key);
        }
        void navigateMainMenu(char key) {
            while (1) {
                security();
                key = kp.getKey();
                if (select(key)) {
```

```
                clearScreen();
                switch (cursrow) {
                    case 1:
                        goStart();
                        lcd.noBacklight();
                        break;
                    case 2:
                        allWards();
                        lcd.noBacklight();
                        break;
                    case 3:
                        oneRoomMenu1();
                        break;
                    default:
                        mainMenu();
                        break;
                }
                break;
            } else moveCursorMainMenu(key);
        }
    }
    void moveCursorMainMenu(char key) {
        if (key == 'A' && cursrow > 1) {
            showCursor(curscol, cursrow - 1);
        } else if (key == 'B' && cursrow < 3) {
            showCursor(curscol, cursrow + 1);
        }
    }
    void oneRoomMenu1() {
        char key;
        lcd.setCursor(1, 0);
        lcd.print("Canteen");
        lcd.setCursor(1, 1);
        lcd.print("Pharmacy");
        lcd.setCursor(1, 2);
        lcd.print("Laundry Room");
        lcd.setCursor(1, 3);
        lcd.print("Ward 1");
        lcd.setCursor(17, 3);
        lcd.print("1/2");
        if (pagechange) {
            showCursor(0, 3);
        } else {
            showCursor(0, 0);
        }
        pagechange = 0;
        navigateOneRoomMenu1(key);
    }
    void navigateOneRoomMenu1(char key) {
        while (1) {
            security();
            key = kp.getKey();
            if (select(key)) {
                clearScreen();
                switch (cursrow) {
                    case 0:
                        canteen();
                        break;
                    case 1:
                        pharmacy();
                        break;
                    case 2:
                        laundry();
                        break;
                    case 3:
                        ward1();
                        break;
                    default:
                        oneRoomMenu1();
                        break;
                }
                lcd.noBacklight();
                break;
            } else if (back(key)) {
                clearScreen();
                mainMenu();
                break;
            } else {
                moveCursorOneRoomMenu1(key);
                if (cursrow > 3) {
                    clearScreen();
                    oneRoomMenu2();
                    break;
                }
            }
        }
    }
    void moveCursorOneRoomMenu1(char key) {
        if (key == 'A' && cursrow > 0) {
            showCursor(curscol, cursrow - 1);
        } else if (key == 'B') {
            showCursor(curscol, cursrow + 1);
        }
    }
    void oneRoomMenu2() {
        char key;
        lcd.setCursor(1, 0);
        lcd.print("Ward 2");
        lcd.setCursor(1, 1);
        lcd.print("Ward 3");
        lcd.setCursor(1, 2);
        lcd.print("Ward 4");
        lcd.setCursor(1, 3);
        lcd.print("Ward 5");
        lcd.setCursor(17, 3);
        lcd.print("2/2");
        showCursor(0, 0);
        navigateOneRoomMenu2(key);
    }
    void navigateOneRoomMenu2(char key) {
        while (1) {
            security();
            key = kp.getKey();
            if (select(key)) {
                clearScreen();
                switch (cursrow) {
                    case 0:
                        ward2();
                        break;
                    case 1:
                        ward3();
                        break;
                    case 2:
                        ward4();
                        break;
                    case 3:
                        ward5();
                        break;
                    default:
                        oneRoomMenu2();
                        break;
                }
                lcd.noBacklight();
                break;
            } else if (back(key)) {
                clearScreen();
                mainMenu();
                break;
            } else {
                moveCursorOneRoomMenu2(key);
                if (cursrow < 0) {
                    clearScreen();
                    pagechange = 1;
                    oneRoomMenu1();
                    break;
                }
            }
        }
    }
    void moveCursorOneRoomMenu2(char key) {
        if (key == 'A') {
            showCursor(curscol, cursrow - 1);
        } else if (key == 'B' && cursrow < 3) {
            showCursor(curscol, cursrow + 1);
        }
    }
    void showCursor(int col, int row) {
        lcd.setCursor(col, row);
        lcd.cursor();
        lcd.blink();
        curscol = col;
        cursrow = row;
    }
    bool select(char key) {
        return (key == 'C');
    }
    bool back(char key) {
        return (key == 'D');
    }
    void clearScreen() {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.noCursor();
        lcd.noBlink();
    }
    int askPin(char trypin[], int n) {
        lcd.print("Please insert PIN");
        lcd.setCursor(0, 1);
        lcd.print("PIN:");
        int i;
        for (i = 0; i < n; i++) {
            trypin[i] = kp.waitForKey();
            lcd.setCursor(i + 4, 1);
            lcd.print("*");
        }
        clearScreen();
    }
    void checkPin(char trypin[], int n) {
        int i, k;
        while (1) {
            security();
            k = 0;
            clearScreen();
            askPin(trypin, N);
            for (i = 0; i < n; i++) {
                if (trypin[i] == pin[i]) k++;
            }
            if (k == 4) {
                clearScreen();
                lcd.setCursor(4, 1);
                lcd.print("PIN Correct!");
                delay(1000);
                break;
            } else {
                lcd.setCursor(5, 1);
                lcd.print("Wrong PIN!");
                lcd.setCursor(5, 2);
                lcd.print("Try again!");
                delay(1000);
                continue;
            }
        }
        clearScreen();
    }
    void getSensorReadings() {
        int sensorValue1 = analogRead(A7);
        int sensorValue2 = analogRead(A8);
        int sensorValue3 = analogRead(A9);
        int sensorValue4 = analogRead(A12);
        int sensorValue5 = analogRead(A11);
        mappedvalue1 = mapValue(sensorValue1);
        mappedvalue2 = mapValue(sensorValue2);
        mappedvalue3 = mapValue(sensorValue3);
        mappedvalue4 = mapValue(sensorValue4);
        mappedvalue5 = mapValue(sensorValue5);
    }
    int mapValue(int sensorValue) {
        int mappedvalue = 0;
        mappedvalue = map(sensorValue, 0, 1023, 0, 255);
        return (mappedvalue);
    }
    void motorRun(char motor, char dir, int spd) {
        if (motor == 'R') {
            switch (dir) {
                case 'F':
                    digitalWrite(dir_A, HIGH);
                    digitalWrite(brake_A, LOW);
                    analogWrite(PWM_A, spd);
                    break;
                case 'B':
                    digitalWrite(dir_A, LOW);
                    digitalWrite(brake_A, LOW);
                    analogWrite(PWM_A, spd);
                    break;
                default:
                    digitalWrite(brake_A, HIGH);
            }
        } else if (motor == 'L') {
            switch (dir) {
                case 'F':
                    digitalWrite(dir_B, HIGH);
                    digitalWrite(brake_B, LOW);
                    analogWrite(PWM_B, spd);
                    break;
                case 'B':
                    digitalWrite(dir_B, LOW);
                    digitalWrite(brake_B, LOW);
                    analogWrite(PWM_B, spd);
                    break;
                default:
                    digitalWrite(brake_B, HIGH);
            }
        }
    }
    void motorMovement() {
        lineFollowing();
        interSection();
    }
    void lineFollowing() {
        getSensorReadings();
        if (mappedvalue3 >= upperthreshold && mappedvalue2 <= lower-
threshold && mappedvalue4 <= lowerthreshold && mappedvalue1 <= lower-
threshold && mappedvalue5 <= lowerthresh-
old) { // When sen1=W,sen2=W,sen3=B,sen4=W,sen5=W
            flag = 0;
            motorRun('R', 'F', 255);
            motorRun('L', 'F', 255);
        } else if (mappedvalue3 <= lowerthresh-
old && mappedvalue2 >= upperthreshold && mappedvalue4 <= lowerthresh-
old && mappedvalue1 <= lowerthreshold && mappedvalue5 <= lowerthresh-
old) { // When sen1=W,sen2=B,sen3=W,sen4=W,sen5=W
            flag = 0;
            motorRun('R', 'F', 255);
            motorRun('L', 'F', 0);
        } else if (mappedvalue3 >= upperthresh-
old && mappedvalue2 >= upperthreshold && mappedvalue4 <= lowerthresh-
old && mappedvalue1 <= lowerthreshold && mappedvalue5 <= lowerthresh-
old) { // When sen1=W,sen2=B,sen3=B,sen4=W,sen5=W
            flag = 0;
            motorRun('R', 'F', 255);
            motorRun('L', 'F', 0);
        } else if (mappedvalue3 <= lowerthresh-
old && mappedvalue2 <= lowerthreshold && mappedvalue4 >= upperthresh-
old && mappedvalue1 <= lowerthreshold && mappedvalue5 <= lowerthresh-
old) { // When sen1=W ,sen2=W, sen3=W, sen4=B,sen5=W
            flag = 0;
            motorRun('R', 'F', 0);
            motorRun('L', 'F', 255);
        } else if (mappedvalue3 >= upperthresh-
old && mappedvalue2 <= lowerthreshold && mappedvalue4 >= upperthresh-
old && mappedvalue1 <= lowerthreshold && mappedvalue5 <= lowerthresh-
old) { // When sen1=W,sen2=W,sen3=B,sen4=B,sen5=W
            flag = 0;
            motorRun('R', 'F', 0);
            motorRun('L', 'F', 255);
        } else if (mappedvalue3 <= lowerthresh-
old && mappedvalue2 <= lowerthreshold && mappedvalue4 >= upperthresh-
old && mappedvalue1 <= lowerthreshold && mappedvalue5 >= upperthresh-
old) { // When sen1&sen2&sen3=W,sen4&sen5=B
            flag = 0;
            motorRun('R', 'F', 0);
            motorRun('L', 'F', 255);
        } else if (mappedvalue3 <= lowerthresh-
old && mappedvalue2 >= upperthreshold && mappedvalue4 <= lowerthresh-
old && mappedvalue1 >= upperthreshold && mappedvalue5 <= lowerthresh-
old) { // When sen3&sen4&sen5=W,sen1&sen2=B
            flag = 0;
            motorRun('R', 'F', 255);
            motorRun('L', 'F', 0);
        } else if (mappedvalue1 >= upperthresh-
old && mappedvalue2 <= lowerthreshold && mappedvalue3 <= lowerthresh-
old && mappedvalue4 <= lowerthreshold && mappedvalue5 <= lowerthresh-
old) { // When sen1 sees blak while others see white
            flag = 0;
            motorRun('R', 'F', 255);
            motorRun('L', 'F', 0);
        } else if (mappedvalue5 >= upperthresh-
old && mappedvalue2 <= lowerthreshold && mappedvalue3 <= lowerthresh-
old && mappedvalue4 <= lowerthreshold && mappedvalue1 <= lowerthresh-
old) { // When sen5 sees black while others see white
            flag = 0;
            motorRun('R', 'F', 0);
            motorRun('L', 'F', 255);
        }
    }
    void interSection() {
        int m = millis();
        if (mappedvalue1 >= upperthreshold && mappedvalue2 >= upper-
threshold && mappedvalue3 >= upperthreshold && mappedvalue4 <= lower-
threshold && mappedvalue5 <= lowerthresh-
old) { // sen1&sen2&sen3=B,sen4&sen5=W
            breakvar = 0;
        } else if (mappedvalue1 <= lowerthresh-
old && mappedvalue2 <= lowerthreshold && mappedvalue3 >= upperthresh-
old && mappedvalue4 >= upperthreshold && mappedvalue5 >= upperthresh-
old) { // sen3&sen4&sen5=B,sen1&sen2=W
            breakvar = 0;
        } else if (mappedvalue1 >= upperthresh-
old && mappedvalue2 >= upperthreshold && mappedvalue3 >= upperthresh-
old && mappedvalue4 >= upperthreshold && mappedvalue5 <= lowerthresh-
old) { // sen1&sen2&sen3&sen4=B,sen5=W
            breakvar = 0;
        } else if (mappedvalue1 <= lowerthresh-
old && mappedvalue2 >= upperthreshold && mappedvalue3 >= upperthresh-
old && mappedvalue4 >= upperthreshold && mappedvalue5 >= upperthresh-
old) { // sen3&sen4&sen5&sen2=B ,sen1=W
            breakvar = 0;
        } else if (mappedvalue1 >= upperthresh-
old && mappedvalue2 >= upperthreshold && mappedvalue3 >= upperthresh-
old && mappedvalue4 >= upperthreshold && mappedvalue5 >= upperthresh-
old) { // sen3&sen4&sen5&sen2&sen1=B
            breakvar = 0;
        }
    }
    void readRFID() {
        while (1) {
            motorRun('R', 'F', 125);
            motorRun('L', 'F', 125);
            if (!mfrc522.PICC_IsNewCardPresent()) {
                break;
            } // Select one of the cards
            if (!mfrc522.PICC_ReadCardSerial()) {
                break;
            }
            String content = "";
            for (byte i = 0; i < mfrc522.uid.size; i++) {
                content.concat(String(mfrc522.uid.uid-
  Byte[i] < 0x10 ? " 0" : " "));
                content.concat(String(mfrc522.uid.uidByte[i], HEX));
            }
            content.toUpperCase();
            if (content.substring(1) == "35 12 02 15") {
                Serial.write(1);
                realPresentLocationMatrix[0] = 3;
                realPresentLocationMatrix[1] = 5;
                y++;
                t = 1;
                turnOffLEDs();
            } else if (content.substring(1) == "35 7A E4 15") {
                Serial.write(2);
                realPresentLocationMatrix[0] = 3;
                realPresentLocationMatrix[1] = 6;
                y++;
                t = 1;
                turnOffLEDs();
            } else if (content.substring(1) == "35 55 20 15") {
                Serial.write(3);
                realPresentLocationMatrix[0] = 3;
                realPresentLocationMatrix[1] = 7;
                y++;
                t = 1;
                turnOffLEDs();
            } else if (content.substring(1) == "65 0B 44 15") {
                Serial.write(4);
                realPresentLocationMatrix[0] = 4;
                realPresentLocationMatrix[1] = 7;
                y++;
                t = 1;
                turnOffLEDs();
            } else if (content.substring(1) == "35 12 89 15") {
                Serial.write(5);
                realPresentLocationMatrix[0] = 4;
                realPresentLocationMatrix[1] = 6;
                y++;
                t = 1;
                turnOffLEDs();
            } else if (content.substring(1) == "25 E1 E7 15") {
                Serial.write(6);
                realPresentLocationMatrix[0] = 4;
                realPresentLocationMatrix[1] = 5;
                y++;
                t = 1;
                turnOffLEDs();
            } else if (content.substring(1) == "65 D0 9C 15") {
                Serial.write(7);
                realPresentLocationMatrix[0] = 5;
                realPresentLocationMatrix[1] = 5;
                y++;
                t = 1;
                turnOffLEDs();
            } else if (content.substring(1) == "25 C8 DB 15") {
                Serial.write(8);
                realPresentLocationMatrix[0] = 6;
                realPresentLocationMatrix[1] = 5;
                y++;
                t = 1;
                turnOffLEDs();
            } else if (content.substring(1) == "45 56 DC 15") {
                Serial.write(9);
                realPresentLocationMatrix[0] = 7;
                realPresentLocationMatrix[1] = 5;
                y++;
                t = 1;
                turnOffLEDs();
            } else if (content.substring(1) == "94 2F 24 17") {
                Serial.write(10);
                realPresentLocationMatrix[0] = 7;
                realPresentLocationMatrix[1] = 6;
                y++;
                t = 1;
                turnOffLEDs();
            } else if (content.substring(1) == "35 6A E1 15") {
                Serial.write(11);
                realPresentLocationMatrix[0] = 7;
                realPresentLocationMatrix[1] = 7;
                y++;
                t = 1;
```

```
                    turnOffLEDs();
                } else if (content.substring(1) == "35 06 6A 15") {
                    Serial.write(12);
                    realPresentLocationMatrix[0] = 8;
                    realPresentLocationMatrix[1] = 7;
                    y++;
                    t = 1;
                    turnOffLEDs();
                } else if (content.substring(1) == "35 68 AB 15") {
                    Serial.write(13);
                    realPresentLocationMatrix[0] = 8;
                    realPresentLocationMatrix[1] = 6;
                    y++;
                    t = 1;
                    turnOffLEDs();
                } else if (content.substring(1) == "45 5A 34 15") {
                    Serial.write(14);
                    realPresentLocationMatrix[0] = 8;
                    realPresentLocationMatrix[1] = 5;
                    y++;
                    t = 1;
                    turnOffLEDs();
                } else if (content.substring(1) == "35 D8 EE 15") {
                    Serial.write(15);
                    realPresentLocationMatrix[0] = 8;
                    realPresentLocationMatrix[1] = 4;
                    y++;
                    t = 1;
                    turnOffLEDs();
                } else if (content.substring(1) == "84 D1 E0 17") {
                    Serial.write(16);
                    realPresentLocationMatrix[0] = 7;
                    realPresentLocationMatrix[1] = 4;
                    y++;
                    t = 1;
                    turnOffLEDs();
                } else if (content.substring(1) == "35 99 2A 15") {
                    Serial.write(17);
                    realPresentLocationMatrix[0] = 6;
                    realPresentLocationMatrix[1] = 4;
                    y++;
                    t = 1;
                    turnOffLEDs();
                } else if (content.substring(1) == "75 5D 51 15") {
                    Serial.write(18);
                    realPresentLocationMatrix[0] = 5;
                    realPresentLocationMatrix[1] = 4;
                    y++;
                    t = 1;
                    turnOffLEDs();
                } else if (content.substring(1) == "25 E4 D3 15") {
                    Serial.write(19);
                    realPresentLocationMatrix[0] = 4;
                    realPresentLocationMatrix[1] = 4;
                    y++;
                    t = 1;
                    turnOffLEDs();
                } else if (content.substring(1) == "25 C6 D9 15") {
                    Serial.write(20);
                    realPresentLocationMatrix[0] = 4;
                    realPresentLocationMatrix[1] = 3;
                    y++;
                    t = 1;
                    turnOffLEDs();
                } else if (content.substring(1) == "35 14 8C 15") {
                    Serial.write(21);
                    realPresentLocationMatrix[0] = 4;
                    realPresentLocationMatrix[1] = 2;
                    y++;
                    t = 1;
                    turnOffLEDs();
                } else if (content.substring(1) == "45 0B 61 15") {
                    Serial.write(22);
                    realPresentLocationMatrix[0] = 4;
                    realPresentLocationMatrix[1] = 1;
                    y++;
                    t = 1;
                    turnOffLEDs();
                } else if (content.substring(1) == "25 F3 C0 15") {
                    Serial.write(23);
                    realPresentLocationMatrix[0] = 3;
                    realPresentLocationMatrix[1] = 1;
                    y++;
                    t = 1;
                    turnOffLEDs();
                } else if (content.substring(1) == "55 A0 3D 15") {
                    Serial.write(24);
                    realPresentLocationMatrix[0] = 2;
                    realPresentLocationMatrix[1] = 1;
                    y++;
                    t = 1;
                    turnOffLEDs();
```

```
            } else if (content.substring(1) == "25 F1 40 15") {
                Serial.write(25);
                realPresentLocationMatrix[0] = 1;
                realPresentLocationMatrix[1] = 1;
                y++;
                t = 1;
                turnOffLEDs();
            } else if (content.substring(1) == "35 80 4B 15") {
                Serial.write(26);
                realPresentLocationMatrix[0] = 1;
                realPresentLocationMatrix[1] = 2;
                y++;
                t = 1;
                turnOffLEDs();
            } else if (content.substring(1) == "35 AB 50 15") {
                Serial.write(27);
                realPresentLocationMatrix[0] = 2;
                realPresentLocationMatrix[1] = 2;
                y++;
                t = 1;
                turnOffLEDs();
            } else if (content.substring(1) == "25 C6 06 15") {
                Serial.write(28);
                realPresentLocationMatrix[0] = 3;
                realPresentLocationMatrix[1] = 2;
                y++;
                t = 1;
                turnOffLEDs();
            } else if (content.substring(1) == "55 4E 52 15") {
                Serial.write(29);
                realPresentLocationMatrix[0] = 3;
                realPresentLocationMatrix[1] = 3;
                y++;
                t = 1;
                turnOffLEDs();
            } else if (content.substring(1) == "45 E3 B7 15") {
                Serial.write(30);
                realPresentLocationMatrix[0] = 3;
                realPresentLocationMatrix[1] = 4;
                y++;
                t = 1;
                turnOffLEDs();
            }
        }
    }
    void turnOffLEDs() {
        digitalWrite(LED5, LOW);
        digitalWrite(LED2, LOW);
        digitalWrite(LED6, LOW);
        digitalWrite(LED4, LOW);
    }
    void turnRight90() {
        getSensorReadings();
        while (mappedvalue5 <= lowerthreshold) {
            motorRun('R', 'B', 255);
            motorRun('L', 'F', 255);
            getSensorReadings();
        }
        getSensorReadings();
        while (mappedvalue4 <= lowerthreshold) {
            motorRun('R', 'B', 200);
            motorRun('L', 'F', 200);
            getSensorReadings();
        }
    }
    void turnLeft90() {
        getSensorReadings();
        while (mappedvalue1 <= lowerthreshold) {
            motorRun('R', 'F', 255);
            motorRun('L', 'B', 255);
            getSensorReadings();
        }
        getSensorReadings();
        while (mappedvalue3 <= lowerthreshold) {
            motorRun('R', 'F', 200);
            motorRun('L', 'B', 200);
            getSensorReadings();
        }
    }
    void goBitForward() {
        motorRun('R', 'F', 125);
        motorRun('L', 'F', 125);
        delay(650);
        digitalWrite(brake_A, HIGH);
        digitalWrite(brake_B, HIGH);
    }
    void forward() {
        motorRun('R', 'F', 125);
        motorRun('L', 'F', 125);
        delay(350);
        digitalWrite(brake_A, HIGH);
        digitalWrite(brake_B, HIGH);
    }
    void obstacleDetection() {
        getDistance();
        detectionCondition();
    }
    void getDistance() {
        distance1 = getReading_US(trigPin1, echoPin1);
        distance2 = getReading_US(trigPin2, echoPin2);
        distance3 = getReading_US(trigPin3, echoPin3);
    }
    int getReading_US(int trigPin, int echoPin) {
        long duration;
        int distance;
        digitalWrite(trigPin, LOW);
        delayMicroseconds(2);
        digitalWrite(trigPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin, LOW);
        duration = pulseIn(echoPin, HIGH); // Calculates the distance
        distance = duration * 0.034 / 2;
        return (distance);
    }
    void detectionCondition() {
        while (distance1 <= detectDist || distance2 <= detectDist || distance3 <= detectDist) {
            digitalWrite(brake_A, HIGH);
            digitalWrite(brake_B, HIGH);
            getDistance();
        }
    }
    void ultraSonic(int trigPin, int echoPin) {
        digitalWrite(trigPin, LOW);
        delayMicroseconds(2);
        digitalWrite(trigPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin, LOW);
        duration = pulseIn(echoPin, HIGH);
        distance = duration * 0.034 / 2;
    }
    void buzzer() {
        tone(outPin, 2300);
        delay(500);
        noTone(outPin);
        delay(500);
    }
    void security() {
        if (digitalRead(SENSOR_PIN4) == HIGH) {
            Serial.write(201);
            digitalWrite(outPin, LOW);
        } else {
            digitalWrite(outPin, HIGH);
            buzzer();
            Serial.write(200);
        }
        ultraSonic(trigPin, echoPin);
        if (distance > 100) {
            Serial.write(201);
            digitalWrite(outPin, HIGH);
            buzzer();
            Serial.write(200);
        } else {
            digitalWrite(outPin, LOW);
        }
    }
    void temperature() {
        sensors.requestTemperatures(); // Ask for a reading
       // This "0" is because we are using the first IC in the bus, as this command allows multiplexation of more than one sensor
        temp = sensors.getTempCByIndex(0);
        if (temp < 25.0) {
            Serial.write(40);
        } else if (temp >= 25.0 && temp < 30.0) {
            Serial.write(41);
        } else if (temp >= 30.0 && temp < 35.0) {
            Serial.write(42);
        } else if (temp >= 35.0 && temp < 40.0) {
            Serial.write(43);
        } else if (temp >= 40.0 && temp < 45.0) {
            Serial.write(44);
        } else if (temp >= 45.0 && temp < 50.0) {
            Serial.write(45);
        } else if (temp >= 50.0) {
            Serial.write(46);
        }
    }
```

Appendix 2. Processing Code

```
//Import serial communication library
import processing.serial.*;
Serial myPort;
int xPos = 300;
int yPos = 400;
int temp = 43;
int colorr = 100;
int destination = 100;
int security = 0;
PFont font;
void setup() {
    fullScreen();
    String portName = Serial.list()[0];
    myPort = new Serial(this, portName, 9600);
}
void draw() {
    font = createFont("Georgia", 20);
    textFont(font);
    backImage();
    stroke(255);
    dot();
    battery();
    printInfo();
}
void battery() {
    if (temp == 1) {
        levelOne();
    } else if (temp == 2) {
        levelOne();
        levelTwo();
    } else if (temp == 3) {
        levelOne();
        levelTwo();
        levelThree();
    } else if (temp == 4) {
        levelOne();
        levelTwo();
        levelThree();
        levelFour();
    } else if (temp == 5) {
        levelOne();
        levelTwo();
        levelThree();
        levelFour();
        levelFive();
    } else if (temp == 6) {
        levelOne();
        levelTwo();
        levelThree();
        levelFour();
        levelFive();
        levelSix();
    } else if (temp == 7) {
        levelOne();
        levelTwo();
        levelThree();
        levelFour();
        levelFive();
        levelSix();
        levelSeven();
    }
}
void levelSeven() {
    fill(102, 0, 0);
    rect(1110, 270, 400, 25, 7);
    fill(0, 0, 0);
    text(">50°C", 1280, 290);
}
void levelSix() {
    fill(153, 0, 0);
    rect(1110, 300, 400, 25, 7);
    fill(0, 0, 0);
    text("45°C-50°C", 1260, 320);
}
void levelFive() {
    fill(204, 0, 0);
    rect(1110, 330, 400, 25, 7);
    fill(0, 0, 0);
    text("40°C-45°C", 1260, 350);
}
void levelFour() {
    fill(255, 128, 0);
    rect(1110, 360, 400, 25, 7);
    fill(0, 0, 0);
    text("35°C-40°C", 1260, 380);
}
void levelThree() {
    fill(255, 255, 51);
    rect(1110, 390, 400, 25, 7);
```

```
  fill(0, 0, 0);
  text("30°C-35°C", 1260, 410);
}
void levelTwo() {
  fill(178, 255, 102);
  rect(1110, 420, 400, 25, 7);
  fill(0, 0, 0);
  text("25°C-30°C", 1260, 440);
}
void levelOne() {
      fill(153, 255, 153);
      rect(1110, 450, 400, 25, 7);
      fill(0, 0, 0);
      text("< 25°C", 1280, 470);
    }
    void dot() {
      fill(0, colorr, 0);
      ellipse(xPos, yPos, 20, 20);
    }
    void backImage() {
      background(15);
      fill(25, 25, 25);
      rect(25, 60, 1000, 825, 7);
      rect(1040, 60, 550, 425, 7);
      rect(1040, 500, 550, 385, 7);
      fill(190, 190, 190);
      rect(750, 60, 275, 30, 7);
      fill(50, 50, 50);
      text("Real-time Localization", 780, 80);
      fill(190, 190, 190);
      rect(1325, 60, 265, 30, 7);
      fill(50, 50, 50);
      text("Battery Temperature", 1370, 80);
      fill(190, 190, 190);
      rect(1325, 500, 265, 30, 7);
      fill(50, 50, 50);
      text("Info", 1400, 520);
      stroke(250);
      line(75, 75, 425, 75); //A
      line(75, 75, 75, 225); //B
      line(75, 225, 275, 225); //C
      line(275, 225, 275, 725); //D
      line(275, 725, 425, 725); //E
      line(425, 725, 425, 525); //F
      line(425, 75, 425, 375); //G
      line(425, 375, 825, 375); //H
      line(425, 525, 675, 525); //I
      line(675, 525, 675, 725); //J
      line(675, 725, 825, 725); //K
      line(825, 375, 825, 725); //L
      fill(50, 150, 0);
      text("Monitoring Screen", 725, 35);
    }
    void serialEvent(Serial myPort) {
      int x = myPort.read();
      switch (x) {
        case 1:
          xPos = 300;
          yPos = 500;
          break;
        case 2:
          xPos = 300;
          yPos = 600;
          break;
        case 3:
          xPos = 300;
          yPos = 700;
          break;
        case 4:
          xPos = 400;
          yPos = 700;
          break;
        case 5:
          xPos = 400;
          yPos = 600;
          break;
        case 6:
          xPos = 400;
          yPos = 500;
          break;
        case 7:
          xPos = 500;
          yPos = 500;
          break;
        case 8:
          xPos = 600;
          yPos = 500;
          break;
        case 9:
          xPos = 700;
          yPos = 500;
          break;
```

```
            case 10:
                xPos = 700;
                yPos = 600;
                break;
            case 11:
                xPos = 700;
                yPos = 700;
                break;
            case 12:
                xPos = 800;
                yPos = 700;
                break;
            case 13:
                xPos = 800;
                yPos = 600;
                break;
            case 14:
                xPos = 800;
                yPos = 500;
                break;
            case 15:
                xPos = 800;
                yPos = 400;
                break;
            case 16:
                xPos = 700;
                yPos = 400;
                break;
            case 17:
                xPos = 600;
                yPos = 400;
                break;
            case 18:
                xPos = 500;
                yPos = 400;
                break;
            case 19:
                xPos = 400;
                yPos = 400;
                break;
            case 20:
                xPos = 400;
                yPos = 300;
                break;
            case 21:
                xPos = 400;
                yPos = 200;
                break;
            case 22:
                xPos = 400;
                yPos = 100;
                break;
            case 23:
                xPos = 300;
                yPos = 100;
                break;
            case 24:
                xPos = 200;
                yPos = 100;
                break;
            case 25:
                xPos = 100;
                yPos = 100;
                break;
            case 26:
                xPos = 100;
                yPos = 200;
                break;
            case 27:
                xPos = 200;
                yPos = 200;
                break;
            case 28:
                xPos = 300;
                yPos = 200;
                break;
            case 29:
                xPos = 300;
                yPos = 300;
                break;
            case 30:
                xPos = 300;
                yPos = 400;
                break;
            case 40:
                temp = 1;
                break;
            case 41:
                temp = 2;
                break;
            case 42:
                temp = 3;
```

```
                    break;
                case 43:
                    temp = 4;
                    break;
                case 44:
                    temp = 5;
                    break;
                case 45:
                    temp = 6;
                    break;
                case 46:
                    temp = 7;
                    break;
                case 200:
                    security = 200;
                    break;
                case 201:
                    security = 201;
                    break;
                default:
                    destination = x;
                    break;
            }
        }
        void printInfo() {
            font = createFont("Georgia", 30);
            textFont(font);
            if (destination == 0) {
                fill(255, 255, 255);
                text("Arrived to destination", 1060, 560);
            } else {
                fill(255, 255, 255);
                text("Going to location:", 1060, 560);
                text(destination - 100, 1300, 560);
            }
            if (security == 200) {
                fill(255, 0, 0);
                text("Security Problem", 1060, 600);
            } else {
                fill(25, 25, 25);
                text("Security Problem", 1060, 600);
            }
            if (temp == 6 || temp == 7) {
                fill(255, 0, 0);
                text("Critical Temperature", 1060, 640);
            } else {
                fill(25, 25, 25);
                text("Critical Temperature", 1060, 640);
            }
        }
```