# SMART ATM FINAL PROJECT REPORT

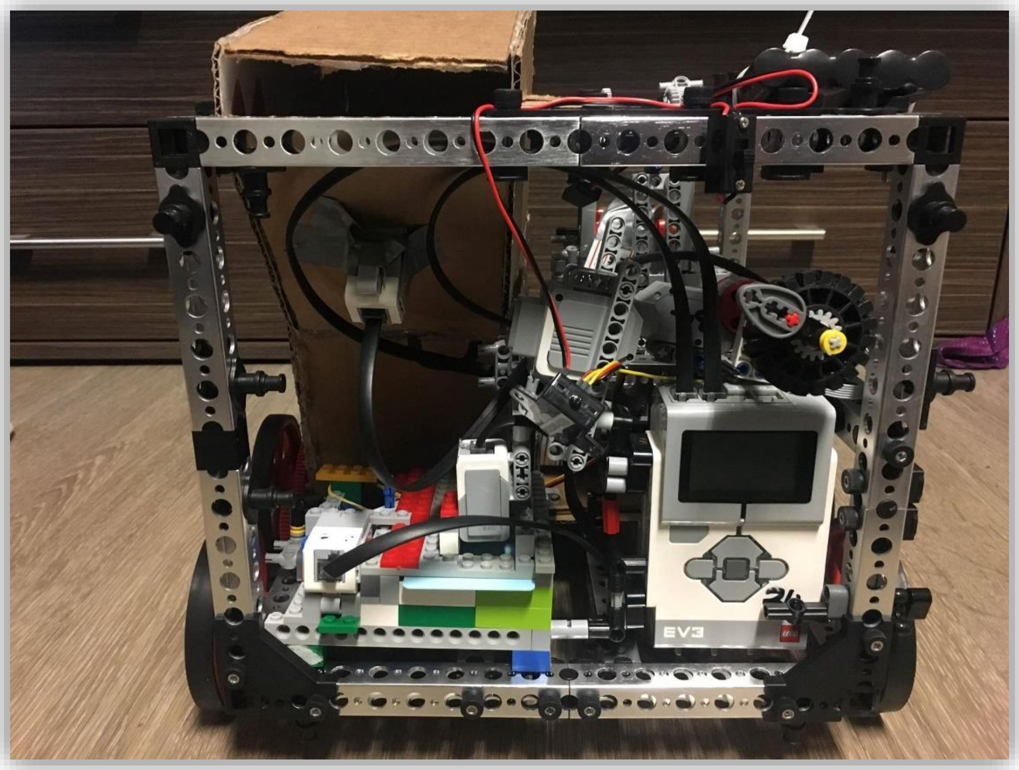
Sarim Ali, Dhruv Manani, Kyle Tam, Mahad Zaryab

Student IDs: 20767329, 20780235, 20763704, 20765196

Group: 834

Course: MTE 100

Date: Dec 3rd, 2018



## Table of Contents

## Summary

Automated Teller Machines (ATMs) are convenient electronic banking machines which allow users to perform basic transactions on their accounts without the need for human interaction [1]. However, ATMs are plagued with complicated menus and a lack of flexibility that often impacts their efficiency negatively.

The problem this project was trying to solve was the need for an easy-to-use, mobile and safe Automated Teller Machine (ATM). Thus, the scope of this project was to design a fully functional and mobile ATM machine with a simple user interface. Overall, the project was successful in satisfying all the criteria and constraints and designing a system that satisfied the need statement set during the initial stages of the project.

### Design Decision

In the Interim report, a decision matrix was used with a weighted criterion to determine the best design choice. Design #1, which involved mounting the cash withdrawing mechanism on the opposite side of the user interface, in favour of stability and storage, achieved the highest overall score as a result of being the safest and most stable system. Thus, the following report highlights the outcomes of choosing this design to create an ATM robot.

### Verification of Constraints on Design

All the constraints set in the planning stage of the project were met, with minor additions to account for a lack of project resources. The revised constraints of the project are as follows:

- Time: Design should not take more than 4 weeks to complete
- Human resources: Design should not require more than 4 people to complete the project
- Project resources: Design should not require more than the given resources that were provided for the project by the employer (University of Waterloo)
- Budget: Design should not cost more than $20 for external resources

**Verification of Criteria on Design**

All the criteria set during the planning stage of the project were satisfied, with a minor change of physical bills being switched out for tickets. The revised criteria for the project is as follows:

- Stability of system
- The efficiency of money dispensing
- Storage of bills in the system
- Safety

**Mechanical Design**

- Chassis – A robust frame for the robot was made using Tetrix components. Moreover, the base of the robot was attached to two continuous motors to allow for remote controlled movement.
- Card reader – A user-friendly card reader was made using Lego pieces, allowing the program to detect if whether a card was inserted, and the individual associated with the account.
- Withdrawing mechanism – A suspended motor capable of dispensing money based on the exact amount indicated by the user.

**Software Design**

- Storage of Data – 1-D arrays were used to store the logistics of each account, such as the balance and colour, and a 2-D array was used to store the correct pins for each account.
- Functions – A function was made for each important task, notably configuring sensors, withdrawing and depositing money, and getting the correct pin.
- Exiting mechanism – The program exited when an incorrect pin was entered three times in a row or if the user simultaneously pressed the right and left buttons.

## Introduction

The following is from the Interim Report:

"An automated teller machine (ATM) is an electronic banking outlet that allows customers to complete basic transactions without the aid of a branch representative or teller" [1]. The increasing popularity of ATMs can be attributed to their convenience, operation at virtually all hours and lack of human interaction [2]. For example, 107 billion cash withdrawals were conducted on ATMs globally in the year 2016 alone [2]. However, there are a few problems with ATMs that prevent future growth in their usage; the increasing complexity of ATM user interfaces and the lack of mobility and flexibility of ATMs prevent the technology from being more widely used.

Firstly, the increasingly complex menus and graphical user interfaces (GUIs) used by ATMs have a negative impact on the users who are stakeholders of the ATM. For instance, according to a survey done in India by S. Bhargavi, cash withdrawal accounts for 78% of all ATM usage, yet the most common ATMs nest this option behind 3 to 4 menus [3]. Furthermore, S. Bhargavi's surveys also illustrate the negative impact of increasingly complicated menus and interfaces of ATMs that often cause users to seek human help, prolonging the time spent at the ATM and perhaps making it more inconvenient than going to a local bank [3].

Moreover, the size and weight of ATMs, which is on average 150-250 pounds for a regular sized ATM, make them stationary and inflexible [4]. Lack of movement is an issue to certain stakeholders such as users, clients and customers. Lack of mobility is a problem for some users because they may not be able to or find it inconvenient to drive or walk to an ATM to withdraw money. Clients and customers could find the immobility of ATMs to be a source of extra expenses and wasted time as the ATM could not easily be moved from its initial installation position

Thus, it was the goal of this design project to create a fully functional mobile ATM with a simple and understandable user interface, and in doing such satisfy the Stakeholder's need for a simpler and more flexible ATM.

## Scope

[Continued from Interim Report] As an ATM machine, the robot must maintain the basic functionality of an ATM, in addition to the improvements. Thus, to accomplish the task of creating a mobile and simpler ATM, the robot was created with the following capabilities:

- Security: Ensures that only the account owner has access to their account and the balance associated with their credit card. This is necessary to prevent illegal transactions [5].
- Storage of Account Information: The ATM stores the information of all the accounts and credit cards that are to be used with the ATM. This includes the name of the account owner, the current balance on an account, the colour of the credit card and the PIN associated with the account.
- Transaction Processing: Once the card has been authenticated with the correct PIN associated with the account, all transactions update the account balance respectively.
- Human Interaction: The ATM robot allows users to access and modify their account balance without much complication through the use of a simple menu with easily understandable options.
- Physical Operation:
    - The ATM robot withdraws tickets of the correct monetary value.
    - Allows the user to deposit "bills" of different colours corresponding to different monetary values.
- Mobility: Should be able to move safely and freely on a two-dimensional space.

To satisfy the functionalities mentioned above, the ATM robot implements various methods of input. The table below illustrates the types of input, what they measure and the purposes they serve in satisfying the functionalities.

| Input Type | Measurement | Purpose |
|---|---|---|
| Touch Sensor | 1 or 0 | Detects if a credit card has been inserted. Also used for powering off the robot. |
| Colour Sensor | RGB values | Distinguishes between various coloured cards, each of which are associated with a certain account. A colour sensor will also be used for detecting the value of money deposited into the ATM machine. |
| File | String and Integers | For storing and updating user information |
| nMotorEncoder | Numerical value from the rotation of the motor shaft | Used for account PIN entry and selection of the amount of money to transfer. |
| EV3 Buttons | 1 or 0 from the press of a particular button | Used for the majority of user input and option selection |

*Table 1. Implementation of ATM inputs and the purpose they serve.*

As this ATM robot is mobile, it is crucial that it interacts with its environment in a safe manner. Two continuous Tetrix motors are used to allow the robot to move on a two-dimensional plane. A gear reduction has been applied to the motors to increase torque but decrease speed, allowing the wheels to move the relatively heavy ATM robot while also preventing it from moving too fast. For safety reasons, the robot does not move faster than the average walking speed of a human being.

The main tasks of the robot revolve around making the account transaction process as simple as possible. Thus, once turned on, by powering on the EV3 and Tetrix components, the ATM will remain on and be ready to operate on a user's account until both the up and down button on the EV3 are pressed together.

The success of this mobile and simple robot ATM machine was measured by its ability to complete the tasks required and meet the design criteria and constraints [6].

## Changes in Scope

Overall, the final product differed only slightly from the original scope described above. Due to design constraints and issues with the money withdrawal mechanism, the scope of autonomously withdrawing exact amounts of money was modified. Instead of having bills of varying value being withdrawn for the user to pick up, a rolling ticket mechanism was created which spun out the correct amount of tickets for the user to tear of. While this did not meet the exact original scope, it still satisfied the need to create an ATM that distributed the correct monetary value as assigned to the tickets and thus was deemed acceptable by the group. Secondly, the file input type [Refer to *Table 1*] was also changed to arrays. Due to the complexity of using files in RobotC and the near identical performance of essential tasks by using arrays, arrays were chosen over files to store user information. The remainder of the project scope remained the same, which is reflected in the final prototype.

## Initial Project Criteria

| **Criterion Name** | **Measurement** | **Preference** |
|---|---|---|
| Stability of system (quantitative) | The time in minutes the ATM can drive without requiring any human intervention | A greater time is preferred |
| The efficiency of money dispensing (quantitative) | The number of bills the ATM can dispense in one minute | A greater number of bills is preferred |
| Storage of bills in the system (quantitative) | The number of bills the ATM can dispense before having to be reloaded | A greater number of bills stored is preferred |

| | | |
|---|---|---|
| Safety → Speed of robot's drive and control of robot (quantitative/qualitative) | Centimeters driven / minute<br>How much control does the robot have(On a scale of 1-10, with 1 being that the robot has no control, and 10 being the robot is fully controlled)? | Lower numbers are preferred for the speed<br>Higher numbers are preferred for the rating |

*Table 2. Criteria for comparing various designs.*

## Description of Initial Criteria [As stated in the Interim Report]

Stability of system: Due to the interactive nature of the design, it is highly important that the system is stable in its movement. The ideal design would be able to drive on its own without any human intervention to ensure that the parts of the system are intact. This aspect also ties into the safety criterion, as it is essential that the parts of the system do not fall off, especially when a user is in contact with it. This criterion is given a weight factor of 2, to ensure that the system is able to hold the weight of its components and perform its main task.

The efficiency of money dispensing: The efficiency of the ATM's money dispensing is also important as it reduces the wait time for the user. A greater number of bills dispensed per minute is preferred to reduce the wait time for the user, which in turn contributes to a better user experience

Storage of bills in the system: The number of bills the ATM can dispense before having to be refilled is important to once again reduce human intervention. A greater number of bills is preferred as it will reduce the number of times the ATM will have to be refilled. This is extremely important to ensure a good user experience by reducing the amount of the times an ATM would need to be closed for it to be refilled.

Safety: Referring to the interactive nature of the design, it is extremely important that the system does not pose an unsafe environment. Alongside the stability of the system, the safety of the system's surroundings will be ensured by the speed of the robot and the control that the designers have over the robot. In terms of the speed, a lower speed is preferred to ensure that the robot has sufficient time to react to its surroundings. Furthermore, it is preferred that the designers of the system have full control over the system. This may include the system being remote controlled or programming the robot's sensors to react to its surroundings. This criterion is given the highest weight factor of 3 as the entire project revolves around the idea of a safe and easy to use system, and hence safety is the most important aspect of this system.

## Changes in Criteria

As a result of the thorough planning in the intermediate stages of the project, there were not any significant changes made to the initial criteria. The most notable change made was swapping out the physical bills for tickets. This change was made due to the inability of the EV3 motor to dispense out a single bill when the motor was suspended from the upper frame of the system. This issue was countered by suspending a roll of tickets from the upper frame and having the tickets mimic the bills dispensed out of the ATM. This change allowed for an easier and more accurate dispensing of the 'money'.

## Effectiveness of Criteria

The criteria set was highly effective in guiding the project towards the completion of the scope. However, further analysis of the criteria upon completion of the project suggested that while certain criteria were essential towards the completion of the projects, others were not as effective in keeping the project on track. To begin, the stability of the system was valuable towards the success of the project, primarily because of the user-oriented nature of the project. As an ATM requires to take user input in the form of a card insertion, the entering of the account pin and selecting which action to proceed with, it is essential that the ATM requires minimal-to-no human interaction when it comes to keeping the system stable. This criterion was highly effective in focusing the project towards its main goal, the building of an interactive and mobile ATM.

Furthermore, the safety criterion was also an essential aspect of the success of the project. As previously mentioned, the nature of an ATM requires for it to be accessible by users and hence it was important for the system to be safe to use in all aspects. For the purpose of this project, safety was measured by looking at the speed of the system as well as the control of the robot. Without the presence of this criterion, the safety aspect of the need that the project needed to satisfy would have been missing.

Conversely, the efficiency criterion was not as important as the aforementioned criteria for the scope of the project. As the efficiency criterion looked at the number of bills the system could dispense in one minute, it did not aid the design process but rather served as a stretch goal. Although the efficiency criterion was not essential, it was still a good criterion to set as it expanded upon the project's scope and looked at some of the prototype's next steps.

Similarly, the storage criterion was also not valuable towards the scope of the project. As the storage criteria looked at the number of bills that the system could store, it did not impact the need that the project was targeting and like the efficiency criterion, it served as an add-on to the final design. Regardless of the criterion's importance, it was still a decent criterion to set as it suggested some important aspects to add to the actual design of the ATM.

## Initial Project Constraints

| Constraint Description | Reason for Constraint |
|---|---|
| The design must not take more than 4 weeks from the start date to build. | The Demo Day for the entire Mechatronics Class of 2023 is on November 23rd, and hence the project should be ready for viewing and grading purposes by this date |
| The design must not use more than the following resources:<br>▪ 4 EV3 motors<br>▪ 4 rubber wheels<br>▪ 2 touch sensors<br>▪ 3 colour sensors<br>▪ 1 ultrasonic sensor<br>▪ 1 Tetrix kit<br>▪ 1 Lego pieces kit | These are the only resources we were provided with, and therefore the design must use at most the resources listed. |

| | |
|---|---|
| The design should not require more than 4 people to meet the deadline | The class has been divided into groups of fours and hence the design should be feasible enough to meet the deadline with 4 people. |
| The budget for purchasing external resources (such as fake currency) should not exceed more than $20 | As most of the resources have been provided with, the only external resource required is fake currency, which should not exceed $20. |

*Table 3. Constraints on the design of the ATM.*

## Changes in Constraints

Similar to the criteria for the project, there were not any significant changes in the initial constraints. The most notable change was the addition of an extra Tetrix Kit, which was required to finish the structure of the system.

## Effectiveness of Constraints

The constraints set for the project were also highly effective in directing the project toward its intended goal. To begin, the time constraint was extremely important in ensuring that the project was completed before Demo Day. This constraint allowed for proper and structured scheduling of team meetings, which in turn allowed for the project to proceed according to the due date.

Furthermore, the resource constraint was also important in directing the project toward achieving its goal. It provided the project with a set list of resources that were available and allowed the project to stay within its scope. Without a resource constraint, there would have been no limit as to what could be used and hence would not have provided the project with a clear path towards achieving its goal. Likewise, setting a budget for additional expenses was also important as it allocated a set amount of money, $20 in this case, to be available if required. This ensured that the project stayed within its scope and that money was not being spent wastefully on items that were not essential to the project's scope.

Conversely, the human resource constraint was not as important in guiding the actual design of the project. As this project was to be done in groups organized prior to the beginning of planning, this constraint was already implied and hence did not have a major effect on the implementation of this project. However, it should be noted that if the ATM prototype was to be functionally implemented at a larger scale, then a human resource constraint would be much more significant towards the scope of the project.

## Mechanical Design and Implementation

As the main components of an ATM are encapsulated within the actual system, it was important for the design to be strong as it needed to house the main components of the system. As there were two different components which could have been used to build the chassis, the Lego pieces and the Tetrix components, an analysis was conducted within the group to decide on which components to use. After an analysis of both components, it was concluded that the sturdiness of the Tetrix components would be a better option for designing the frame of the system. Moreover, it was decided that due to the freedom of design associated with the Lego pieces, they would be used for the internal components of the system, such as the card reader and the pin mechanism.

### Chassis Design

The major component used in designing the chassis was the Tetrix beams of different lengths as they were the most robust elements available within the kit. Initially, the base of the robot was designed by connecting four beams in a rectangular shape. After doing so, two wheels were attached to the Tetrix beams at the front of the robot. Likewise, two wheels were also attached to the back on either side, however, these wheels were to be fixated with continuous motors. Due to the safety criterion of the project, a gear reduction was applied to both wheels to increase the torque and in turn, decrease the angular speed. Following the base of the robot, the Tetrix connectors and extenders were used to extend four beams upwards from the base. Following this, the chassis was completed by adding four beams in the shape of a rectangle at the top of the system. The frame was designed in a cubical shape to ensure that the centre of gravity stayed within the span of the machine when the internal elements were to be placed inside.

The chassis was the first component that was built and, thusly was the most important aspect of the design. The building of chassis allowed the project to consider two of the most important criteria, safety and stability. This created a starting point needed to complete the execution of the entire project.

## Card Reading Mechanism

The card reader is located beside the Lego EV3 brick and is constructed completely out of Lego pieces. The purpose of this implementation in the design is that it enables the user to insert and store a card into the ATM. The card reader itself is responsible for determining if the ATM is in use by checking to see if there is a card within. This is done by using the touch sensor located within the card reader; if the touch sensor is pressed down, that indicates a card is present and the ATM is in use. If at any moment the card is removed, the ATM goes back to its original state in which it asks to insert a card. To further enhance the ATM, a colour sensor is also implemented into the design of the card reader. The use of a colour sensor allows for a wider variety of cards that can be inserted into the ATM. Once a card is inserted, the colour sensor, which is suspended over the card, intakes the colour of the card. Each unique colour determines the account that is being accessed. For example, if a red card is inserted, then the account associated with red is displayed, such as Professor Hulls. The screen then would display her account information. If another card with a different colour was inserted, it would display their account information on the screen. The incorporation of the touch sensor along with the colour sensor allows the card reader to determine if a card is inserted as well as accept various cards that are associated with different accounts. [Refer to Figure 2 in the Appendix]

## Withdrawal Mechanism Design

The withdrawal mechanism located at the back right of the ATM consists of the following components:

- A rectangular cardboard base with two short cardboard walls
- A cardboard exit slot shaped like a triangular prism
- The ticket axle and roll
- The hanging motor and accompanying axle
  [Refer to Figure 3 in the Appendix]

The rectangular card-base is designed to hold a stack of bills or in the current usage, a stream of tickets originating from the ticket axle. The base allows the motor to roll the tickets or bills out of the machine through the force of friction acting on the ticket between the motor and the base. Two short cardboard walls were built to keep the stream of bills or tickets from sliding off to the sides of the machine. This facilitates the straight movement of tickets or bills as they exit the machine.

The cardboard exit slot is designed to allow the bills or tickets to exit the machine while preventing theft or tampering. The slot is angled down at the back of the machine to prevent users from looking into the machine. Moreover, the slope of the slot allows the tickets or bills to naturally slide out through the force of gravity. Furthermore, the slot is covered to prevent any access to the tickets other than those distributed by the machine.
[Refer to Figure 4 in the Appendix]

The ticket axle and roll is situated at the top of the machine, behind the motor axle. Connected to the machine using a plastic Lego axle, it is securely connected between two Tetrix bars using a pair of large gears with two shorter gears and a plastic clevis in between on the length of the axle. All of this is necessary to prevent the axle from sliding out naturally while making it still possible for the bar to be taken out should it require modification or maintenance. Connected to each of the two large grey gears are two long Lego blocks that hang down into the machine. Connecting each set of Lego blocks are two shorter red Lego blocks. In between these two blocks, there is a tireless Lego wheel suspended through an axle. This wheel serves as the rotating mechanism that distributes the tickets.
[Refer to Figure 5 in the Appendix]

The motor axle situated in front of the ticket axle is designed structurally similar to the ticket axle with slight differences. On the axle, there are three clevises and two short Lego blocks. Attached to the two Lego blocks there is a third Lego block. On the other side of this block, there are two long Lego blocks attached that reach down into the machine. At the bottom of each of these long Lego blocks, a connector is attached that links these blocks with the large motor. All of this hangs from the top of the Tetrix structure of the ATM. The motor is oriented so that it faces towards the back of the machine. To ensure that the motor does not swing wildly within the machine, the motor is zip-tied onto the base.

The initial design originally consisted of a simplified version of the current design. In this design, the ticket axle was absent as bills were anticipated as the only type of currency to be outputted. Furthermore, the exit slot was simplified to only an inclined piece of cardboard to prevent customers from tampering the system. In addition, the motor consisted of an extra spring. This was to apply pressure on the motor which would, in turn, apply friction onto the paper bills as they were propelled out of the machine. Evidence of this setup is still observable in the current design. The initial design is especially prominent in the motor axle setup. However, due to the shift to the use of tickets, the spring on the motor was removed as the motor is no longer required to move.
[Refer to Figure 6 in the Appendix]

The complexity of the system has increased over time due to unforeseen problems with the withdrawal mechanism. Further analysis was undertaken by the team after the planning stage determined that outputting bills would be problematic. This was observed when tests were determined to see how difficult it would be to output a single bill using a single motor. It was found to be nearly impossible to output one bill as the bills stuck together in eight of the ten tests performed. Some solutions proposed included adding cardstock in between the bills or switching to a ticket system. The later was settled upon as it would be easier to maintain and reload for the client, even though it was slightly more complex to design.

## Depositing Mechanism

The depositing mechanism consists of a cardboard box with a square slot on the bottom, a rectangular slot at the front and a hinged door at the back. The box is shaped like a triangular prism attached to a square based prism. This was designed so that money would fall down an inclined ramp (the triangular section of the box) before falling into the square-based section which acted as a basket. On the inclined ramp, there is a square slot on the bottom in which a colour sensor is attached using duck tape. This sensor is faced inwards so that it can detect the colour of bills passing down the inclined ramp and update the balance accordingly. The slot at the front serves as an entrance in which money can enter the depositing mechanism. The hinged door at the back serves as a way for money to be taken out of the ATM. This functionality is especially important for the owner of the

ATM who would need a way to retrieve all cash transactions undertaken by the machine. The entire mechanism is situated on the left side of the ATM overtop the card reader.

[Refer to Figure 7 in the Appendix]

The initial design originally consisted of a rectangular box with a slot at the front, a motor hanging within and a colour sensor attached from the bottom. The slot at the front would have allowed a user to input money into the machine. The motor hanging near the slot within would have been powered on whenever a user desired to deposit money, rolling the money into the storage bay. The colour sensor would have detected the type of money and updated the balance accordingly.

[Refer to Figure 8 in the Appendix]

This design was changed because it was realized that the motor would be unnecessary. The current design relies on gravity to move the paper bills from the entrance slot into the storage bay. The former design used an extra motor and was reliant on the motor design to work correctly. Through the difficulties encountered when designing the withdrawal mechanism, it was decided that a gravity-facilitated depositing mechanism would be better than this former design.

## Sensor Attachment Design

For the project, three sensors were used whose feedback was used in programming the system. The sensors were strategically placed to obtain the optimal reading. A description of the attachment of each sensor is as follows:

- Card Reader: In order to keep track of which account the card inserted was associated with, a colour sensor was used to detect the colour of the card. This value was then stored within the code of the program. Within the card reader, a slot was made through which the colour sensor could read the colour of the card. The colour sensor itself was attached by designing a suspension out of Lego pieces which extended from the card reader itself. The placement of the sensor was essential in ensuring that it only detected the colour of the card and did not take in any other readings. To account for this, the slot through the card reader was made precisely to the dimension of the colour sensor, ensuring the colour sensor only picked up the colour of the card.

- Card Detector: An ATM cannot function without a card, hence the code for the system was structured in a way so that any individual could use the ATM only if a card was inserted. If a card was inserted and then removed in the middle of a transaction, the individual using the ATM would not be able to perform any further actions. In order to accomplish this, a touch sensor was positioned in front of a Lego lever. As the card was inserted, the card pushed on the lever which in turn pushed the touch sensor. Moreover, an elastic band was added to the centre of the lever to ensure that the lever returned to its original position when the card was removed, releasing the touch sensor
- Depositing mechanism: In order to allow users to deposit money into an account, a colour sensor was attached to the depositing slot. Bills of different amounts were associated with different colours within the code of the program. The colour sensor was also required to be positioned strategically to ensure that the sensor could accurately read the colour of the bill that was inserted. To achieve this, a hole with the dimensions of the colour sensor was cut out in the slope of the depositing box. The colour sensor was then fit through the hole and duct taped to the back of the cardboard. The entire slot was designed in such a way so that when a bill was inserted, it would slide down the slope of the slot, allowing the colour sensor to catch the colour of the bill.
- Pin Mechanism: Any secure ATM requires the user to verify their identification. In this case, a wheel was spun, causing the digits on the screen based on the motor encoder value to change. The mechanism was set up by connecting a wheel to the motor which was mounted on top of the EV3 Brick. Note that all the components, including the wheel and the motor, were designed using the Lego pieces.

## Overall Assembly

The overall assembly of the parts designed was the most challenging aspect of the mechanical implementation of the project. After the chassis was built, the main challenge was the integration between Lego pieces and Tetrix components, as the two kits were not created to mate together. This issue was countered by using Tetrix extenders and Lego axles to connect the two types of parts together. A trade-off that was made in using both kits together was that the assembly was not as stable as it would have been if it had been built with a single kit. However, the use of both components allowed for more freedom in the design. Whenever integration of the two components became impossible to overcome, it was countered by using zip ties and duct tape to secure everything together.

The card reader, built out of Lego pieces, was secured to the lower left corner of the robot. The card reader was mated onto the assembly by attaching two Lego pieces, one on each back corner of the card reader, and assembling it with Tetrix extenders of the same size onto the base of the robot. For additional support, the card reader was also connected to the Tetrix beams of the frame. This was done by connecting a Lego axle between a Tetrix hole and a Lego block hole.

The Lego EV3 brick was placed at the lower right corner of the ATM beside the card reader. It was implemented here for the purpose of making transactions easier for the user, as he/she would insert his/her card and easily look to the right at the screen of the EV3 brick to continue the transaction. The brick was also connected to the Tetrix kit in a similar fashion as to the card reader. An additional Tetrix arm was extended from the cubic Tetrix structure to support the EV3 brick from behind.

The depositing mechanism was constructed out of cardboard and is held together through the use of a glue gun. It was placed right above the card reader for the same purpose of making it easier for the user to switch between each component of the ATM. Due to the fact that it was made out of cardboard, the easiest way to secure this to the frame of the ATM was to zip tie it. This worked effectively because cardboard is rigid so the zip tie was able to prevent the depositing mechanism from moving out of place. Furthermore, the shape of the depositing mechanism was designed so that it would fit snugly in the space left over once all the pieces had been integrated into the structure.

For the withdrawal mechanism, it was placed at the back of the ATM as space at the front was used for other components of the ATM. However, to require only minimal effort from the user, the wheels of the ATM allowed the machine to spin 180 degrees so that the withdrawal mechanism could face the user. To integrate this mechanism into the structure, two axles were used to support the ticket roll and motor. Additionally, zip ties were used so that the cardboard platform and exit slot would stay in place.

Refer to Figure 9 in the Appendix for the finished assembly of the Automated Teller Machine.

## Software Design and Implementation

The software for the ATM Robot involved varying sensors and input types which meant that tasks had to be split up and combined to create a complete program. The following table illustrates the breaking down of the overall program into smaller and more manageable parts.

| Block/Task | Operation | Reason |
|---|---|---|
| Credit Card Reading Program | Waits for the user to enter a credit card and then recognize when a card is entered and read the colour of the card to determine which account it is associated with. | Could be implemented in a separate function and return a value referring to the account which corresponds to the card |
| Pin Entry and Verification System | Obtains an array of digits the user enters using through spinning a motor. Compares this array with the array representing the accounts actual pin and determines if the pin entered is correct. Increments every wrong pin entered and prevents the user from entering more than three. | Is separate from the main operation of the ATM and thus can be called when necessary. |
| Withdrawing Program | Obtains an amount the user wishes to withdraw, using EV3 buttons. Determines if this withdrawal can be executed; if so updates account balance accordingly. | A large part of the operation of the ATM and thus must be closely integrated into the main program. |

| | | |
|---|---|---|
| Depositing Program | Waits for the user to drop money down the depositing hole. Recognizes different coloured bills and determines their monetary value. Adds appropriate value to account balance. | A function can be made which determines the monetary value of the amount deposited. |
| Displaying Menu | Displays options to the user. Allows the user to select an option using EV3 buttons. | A function which main keeps returning to until the user wants to exit. |

*Table 4. Large sections of the ATM Robot Software.*

The final tasks list for the robot ATM was as follows:

- The ATM can dispense money without requiring any human intervention
- The ATM can dispense 20 tickets in one minute
- The ATM can store over 20 bills
- The ATM moves at a speed slower than the average human walking speed
- The design must use only the parts specified in the Interim Report
- The cost of additional resources should not exceed $20

This task list was only slightly altered from previous tasks list due to the design decision to use tickets for withdrawing rather than bills. Due to complications with using bills, a roller system was implemented, allowing the ATM robot to roll tickets out for the user to tear off. Each ticket represented a monetary value of $10. Thus, when withdrawing money, the mechanism for rolling out tickets of the correct monetary value did not require human intervention, but the act of tearing of the tickets did. However, the dispensing of money into an account did not change due to this change in design. Depositing still allowed the user to enter “bills” of different colours corresponding to the appropriate monetary values.

## Functions

To create an ATM robot capable of completing all the necessary tasks, many functions were created and implemented. The following is a list of functions defined by their prototypes, of the form "returnType name(parameters);" which contains a brief description of the operation of the function and its purpose:

[Refer to *Figure 1* of Appendix A for flowchart representing positioning and connection between each function and the main program]

`void displayMenu(int balance, int time);` - Dhruv Manani

- Displays, to the EV3, a menu describing the options a user can take after they have verified their pin
- Also displays the user's current balance, and their time spent at the ATM
- This function is looped once the user enters the correct pin [Refer to *Figure 1*]

`void configureSensors();` - Mahad Zaryab

- Configures/initializes all required sensors for the operation of the ATM robot
- This function is called only once at the beginning of the program [Refer to *Figure 1*]

`int readCard (int *accountColors);` - Sarim Ali

- Waits for a credit card to be inserted by waiting for a touch sensor to be pushed
- Once a card is inserted, the function reads the colour of the card through a colour sensor and determines which account the card is associated with by finding the colour in the array accountColors which is passed into the function
- The integer returned by this function represents the index value of the account the card is associated with
- This is the first function to be called once a new credit card is entered [Refer to *Figure 1*]

`int depositMoney();` - Mahad Zaryab

- While waiting for the user to press the enter button on the EV3, calculates the total amount of money the user deposits into the ATM machine
    - This is done through the use of colour sensors which recognize the monetary value of a bill that passes through the depositing mechanism
    - In this case, a red bill represents $5, a blue bill represents $10 and a green bill represents $20
- The integer returned represents the monetary value of the amount deposited, so that the account balance can be changed accordingly in main
- The function is called when the corresponding button is pressed on the displayMenu function [Refer to *Figure 1*]

`int getWithdrawAmount (int accountBalance);` - Dhruv Manani

- Allows the user to select an amount to withdraw from their account
    - The EV3 up button corresponds to increments of $10, which the down button corresponds to decrements of $10
    - The user is prevented from withdrawing amounts less than $0 and more than what is in their account, which is why accountBalance is taken in as a parameter
- The integer returned represents the monetary value of the amount withdrawn, which is passed to the withdrawMoney function to execute the withdrawal
- The function is called when the corresponding button is pressed on the displayMenu function [Refer to *Figure 1*]

`void withdrawMoney(tMotor port, int amountWithdraw);` - Kyle Tam

- Takes in a motor port and the amount to withdraw from the previous getWithdrawAmount function, and spins the motor the amount required to withdraw the exact amount of tickets that represent the amount withdrawn
- This function is called immediately after the getWithdrawAmount function [Refer to *Figure 1*]

`int getPinEntered(tMotor port, int pinDigit);` - Dhruv Manani

- While waiting for the user to press the enter button, the EV3 displays a value from 0 to 9 representing the current pin digit being entered
  - Ex. If the user is entering the first pin digit, the EV3 screen displays "Enter pin digit 1:" and the numerical value (0-9) the motor is currently spun to, ex. "4." Once enter is pressed by the user, that pin is registered as pin digit 1 in an array representing the pin they enter
- The motor port taken in is used to take user input by using nMotorEncoder. The spinning of the motor on the ATM a certain amount clockwise increments the number displayed on the screen by one; spinning counter-clockwise decrements the number by one. This number is restricted between 0 and 9 and once the user attempts to go below 0, the number goes to 9, whereas if they attempt to go above 9, it resets to 0.
- The value the user enters (determined by when they press the EV3 enter button) is passed back to main as an integer
- This function is called four times in a loop once a card is inserted, to obtain an array of 4 elements representing the pin entered by the user [Refer to F*igure 1*]

`bool isCorrectPin (int *pin, int *correctPin);` - Dhruv Manani

- Takes in two arrays as parameters, representing the pin entered by the user and the correct pin corresponding to the account. Each element of the arrays are compared to determine if the pin entered by the user is correct
- This function is called after a user enters their pin and then returns a boolean representing if the user entered the correct pin or not [Refer to F*igure 1*]

For each of the functions above refer to Appendix B for the full commented code.

## Data Storage

Originally, the data representing user information, such as their name, account balance, card colour and account pin, was to be stored in files. However, due to the complexity of using RobotC File I/O and the equivalent result through using arrays, arrays were chosen to store most of the data. In main [Refer to Appendix B] the following arrays were created to store and update data:

- A 2D array for account pins where each row index represents an account and each column index represents a pin digit (each column containing 4 values)
- A 1D array for account colours, where each index corresponds to the account and stores the colour of the card
- A 1D array storing account balances, which each index corresponding to an account
- A 1D array of strings storing the name of each account

## Design Decisions and Trade-Offs

While most of the software was created as initially intended, some mechanical issues lead to slight changes in the software. Firstly, as mentioned above, the mode of storage of data was changed from files to arrays. Secondly, due to issues with the mechanical aspect of withdrawing separate bills, a ticket roller system was implemented. This change in the mechanical design affected the *withdrawMoney* function in particular because instead of withdrawing multiple bills, it spun a motor the appropriate amount of times to roll out the number of tickets that represented the monetary value the user wished to withdraw. This was a trade-off that had to be made to implement the withdrawing functionality of the robot. Lastly, the *depositingMoney* function was also slightly modified to allow for more accurate functionality. Originally, the function was implemented in such a way that each bill would be detected for the amount of time it was above the colour sensor in the depositing compartment. However, as the colour sensor was unreliable and detected different colours from the same bill, a timer had to be implemented to ensure that one bill was not counted multiple times. While this timer made the procedure of depositing money slightly longer for the user, making them wait approximately 1 second before dropping another bill into the mechanism, this was a necessary trade-off to ensure the accuracy of depositing money.

## Testing and Problem Solving

Testing for the software was conducted by created smaller programs that involved one function and a short main program written to test the function.

For instance, for the *getPinEntered* and *isCorrectPin* functions, a short main was created which consisted of an array representing one pin. The function was repeatedly tested with this main to ensure that it was accurate and fine-tuned to yield the best user experience. This function was quite complicated and took a lot of debugging due to the nature of using *nMotorEncoder* for input. The first issue encountered by the *getPinEntered* function was that the motor did not actually spin, thus preventing the user from entering any kind of input. This issue was resolved by powering the motor to 1 and then -1 immediately after [Refer to Appendix B for the function code]. The second issue with this function was that once the user reached the value of 0 on the screen and tried to spin the motor counter-clockwise, the function would register this as a *pinDigit* and produce an array of 0s as the pin. This was resolved by carefully considering the looping conditions in the function and adding certain if statements to prevent such behaviour. A similar issue involving the 0 value was that the pin would overflow back to 0 once the user spun the motor beyond 9 (clockwise) but would not overflow to 9 once the user spun the motor beyond 0 (counter-clockwise). While this was not a large issue, it was solved after adding an if statement for when a value became less than 0.

Another issue encountered was mentioned above in the Design Decisions and Trade-Offs section; this was the issue of the depositing function. A timer had to be added to prevent inaccurate depositing data from being produced.

Lastly, an issue was also encountered with passing in arrays as parameters to the *isCorrectPin* function. After consulting Carol Hulls, pointers were used to pass in entire arrays which could be compared inside the function.

## Verification

The project was successful in meeting all the constraints that were set during the planning stage. A detailed description of each constraint and the steps taken to achieve it is as follows:

Time Constraint: As stated in the interim report, “The design must not take more than 4 weeks from the start to build” [6]. Based on this constraint, a project schedule was created, allocating several time slots within the week for project purposes. As the project was ready for Demo Day, which was 4 weeks from the start date, the time constraint was successfully met.

Resource Constraint: The resource constraint was updated within the first week of the project to account for an extra Tetrix Kit, which was required to finish the chassis of the system. From the updated resource constraint list, each part was designated a distinct function, which ensured that every function was accounted for so that the constraint for the resources was met. As only the resources listed within the update resource constraint list were used for the completion of the project, the resource constraint was met.

Human Resource Constraint: As stated in the interim report, “The design should not require more than 4 people to meet the deadline” [6]. As only the group members within Group 834 worked on the project, the human resource constraint was satisfied.

Budget Constraint: As stated in the interim report, “The budget for purchasing external resourced should not exceed more than $20” [6]. The money spent on extra resources was approximately $15, which contributed to the purchase of the tickets and cardboard. As the money spent on extra resources was less than $20, the budget constraint was satisfied.

## Project Plan

Compared to the original project plan, only a few of the deadlines such as the cash deposit, withdrawal and housing had to be adjusted due to unexpected problems encountered during building and coding. The following is an updated list of deadlines that reflect when each one was initiated and completed:

| Design | |
|---|---|
| Team Creation | Oct 12 |
| Initial Design Brainstorming (needs/requirements etc.) | Oct 12 - Oct 22 |
| Project Idea Formulated | Oct 22 |
| Design Sketches and Functionalities Determined | Oct 22 - Nov 2 |
| Project Pitch Presentation | Oct 26 |
| Hardware Request | Oct 29 |

| | |
|---|---|
| First Design Revisions | Nov 2 - Nov 9 |
| Second Design Revisions | Nov 9 - Nov 23 |
| Preliminary Design Report | Nov 12 |
| Project Demo | Nov 23 |
| Final Design Report | Dec 3 |
| Building | |
| Chasis | Nov 2 - Nov 4 |
| Card Detection Mechanism | Nov 2 - Nov 9 |
| Mechanical System Presentation | Nov 9 |
| Housing | Nov 2 - Nov 22 |
| Cash Deposit | Nov 12 - Nov 20 |
| Cash Withdrawal | Nov 12 – Nov 20 |
| Coding | |
| Card Detection Mechanism | Nov 13 - 17 |
| Software Design Meeting | Nov 13 |
| User input | Nov 13 - 14 |
| Deposit and Balance Update | Nov 14- 17 |
| Withdrawal and Balance Update | Nov 17 -18 |
| Testing | |
| Stability and Mobility Test | Nov 10 |
| User Input Mechanism Test | Nov 12 |
| Money Deposit Test | Nov 21 |
| Money Withdrawal Test | Nov 14 - Nov 22 |
| System Test | Nov 17 - 22 |

*Table 5. Project schedule and deadlines.*

See Graph 1 and Graph 2 in the Appendix for a graphical interpretation of the initial project deadlines.

In terms of roles, Sarim Ali and Kyle Tam continued to be primarily responsible for the mechanical and structural components of the project while Dhruv Manani and Mahad

Zaryab continued to be primarily responsible for the software components of the project. All team members were equally responsible for the overall design. Throughout the duration of the project, team members either worked together or split into pairs based on their responsibilities. Furthermore, testing was a task that shared equal responsibility between team members. All of these tasks were performed during allocated work times from the course or during daily team meetings after lectures that spanned a minimum of one hour.

The Gantt chart depicted in Graph 1 is structured to demonstrate the importance of each task. Sections like Design come before sections such as Coding because tasks in Coding are dependent on tasks in Design to be completed before they can begin.

Some of the tasks that took more time in comparison to other tasks included the following:

- Manufacturing of the housing and cash withdrawal
- Coding of the main program
- Debugging of the main program
- Testing of the deposit and withdrawal functions

This created some delays in the overall project. As the testing was reliant on the building to be completed first, the dates for testing the cash withdrawal and depositing had to be pushed. Furthermore, the building of the housing was reliant that everything else had been built and tested. That is why the period for this task stretched for a longer period compared to all other tests.

## Conclusion

The problem this project was trying to solve was the need for an easy-to-use, mobile and safe Automated Teller Machine (ATM). Thus, the scope of this project was to design a fully functional and mobile ATM machine with a simple user interface. Overall, the project was successful in satisfying all the criteria and constraints and designing a system that satisfied the need statement set during the initial stages of the project.

### Verification of Constraints and Criteria on Design

All the constraints set in the planning stage of the project were met, with minor additions to account for a lack of project resources. Moreover, all the criteria set during the planning stage of the project were satisfied, with a minor change of physical bills being switched out for tickets.

### Mechanical Design

- Chassis – A robust frame for the robot was made using Tetrix components. Moreover, the base of the robot was attached to two continuous motors to allow for remote controlled movement.
- Card reader – A user-friendly card reader was made using Lego pieces, allowing the program to detect if whether a card was inserted, and the individual associated with the account.
- Withdrawing mechanism – A suspended motor capable of dispensing money based on the exact amount indicated by the user.

### Software Design

- Storage of Data – 1-D arrays were used to store the logistics of each account, such as the balance and colour, and a 2-D array was used to store the correct pins for each account.
- Functions – A function was made for each important task, notably configuring sensors, withdrawing and depositing money, and getting the correct pin.
- Exiting mechanism – The program exited when an incorrect pin was entered three times in a row or if the user simultaneously pressed the right and left buttons.

## Recommendations

The following outlines the main ideas and changes that should be applied in order to improve the design in terms of its mechanical and software aspects.

## Mechanical

- The card reader should be made stronger by using tape to stabilize its fully Lego constructed body to reduce the chance of it breaking apart.
- The money dispenser should be constructed more thoroughly by adding support material to each side of the spinning wheel in order to reduce the risk of the tickets jamming within or coming off track from the sides.
- The assembly for the frame of the ATM should be constructed with more space of the inner components to be placed in, which would allow for smoother transactions by not having everything clumped together.

To further clarify the statements presented above, they can be explained by describing the situations that occurred while running tests on the prototype.

In several trials, the card reader broke apart when a card was inserted due to the instability of the Lego pieces. A solution to this problem is to tape the exterior of the card reader so that the Lego pieces stay in contact with each other. This would provide greater resistance against future forceful card insertions.

For the money dispenser component, it was observed that the tickets that were wrapped around the spinning motor encoder slid off and came off the sides - this caused them to jam up so no tickets exited the dispenser. To avoid this, support material could be added to the side of the wheel so that it could prevent the tickets from coming of place, therefore reducing the risk of jamming.

The outer frame of the ATM should be larger as it was very hard to place components within during assembly. This caused confusion during the process as connection wires got in the way of the components. By enlarging the frame, the ATM would have greater internal space which will also increase the reliability of the system as parts would not have as such a high risk of breaking.

## Software

If there was more time to complete this ATM Robot, in addition to the mechanical changes mentioned above, some software changes could be applied to make the ATM more user-friendly and robust.

Firstly, the mode of data storage would be changed from solely arrays to arrays and files. Arrays are fine when working with small datasets such as those used during the demo of the ATM robot, however, in the real world there are many thousands of accounts and thus it becomes impractical to store all their information in arrays. Using files would not only allow for a larger amount of user data to be stored but would also speed up the operations of the ATM as it could more easily gain access to the data it needs and ignore the data it does not immediately need.

Secondly, to increase the accuracy of the card reading mechanism, instead of using the default colour values, RGB could be used. While this would increase the complexity and size of data storage and reading, it would make the ATM far more secure which is a necessity in the real world. Using RGB values would also allow for more accounts to be stored as there would be many unique combinations of colours a credit card could use, provided that the colour sensor is accurate enough.

## References

[1] Investopedia, “Automated Teller Machine - ATM,” *Investopedia*, 03-Aug-2018. [Online]. Available: https://www.investopedia.com/terms/a/atm.asp. [Accessed: Nov. 9, 2018].

[2] “ATM cash withdrawal volume grows 6 percent in 2016,” *ATM Marketplace*, 15-Dec-2017. [Online]. Available: https://www.atmmarketplace.com/news/atm-cash-withdrawal-volume-grows-6-percent-in-2016/. [Accessed: Nov. 9, 2018].

[3] S. Bhargavi, “User experience and ATMs,” *YourStory.com*, 04-May-2017. [Online]. Available: https://yourstory.com/2017/05/user-experience-atms-neglected-issues-affecting-bank-customers-india/. [Accessed: Nov. 9, 2018].

[4] “ATM Installation,” *ATM Depot*. [Online]. Available: https://www.atmdepot.com/resources/how-atm-machines-work/atm-installation-training/. [Accessed: Nov. 9, 2018].

[5] K. Al-Saleh and S. Bendak, “An Ergonomics Evaluation of Certain ATM Dimensions,” *International Journal of Occupational Safety and Ergonomics*, vol. 19, no. 3, pp. 347–353, 2013. [Accessed: Nov. 10, 2018].

[6] S. Ali, D. Manani, K. Tam and M. Zaryab, "Interim Design Report", University of Waterloo, Waterloo, 2018..

## Appendix A

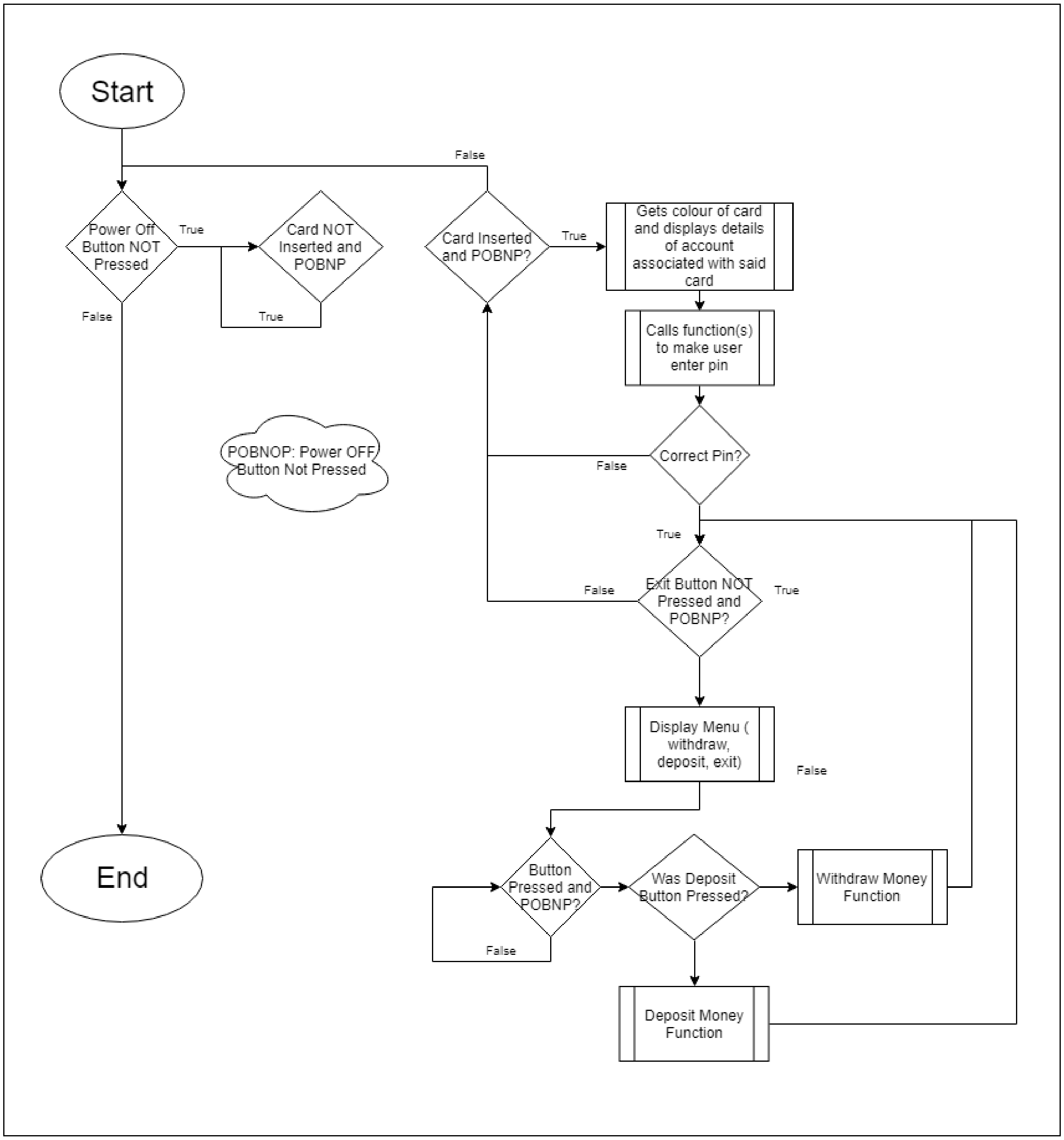

Start
False
Power Off Button NOT Pressed
True
Card NOT Inserted and POBNP
True
False
Card Inserted and POBNP?
True
Gets colour of card and displays details of account associated with said card
Calls function(s) to make user enter pin
POBNOP: Power OFF Button Not Pressed
Correct Pin?
False
True
Exit Button NOT Pressed and POBNP?
False
True
Display Menu ( withdraw, deposit, exit)
False
End
Button Pressed and POBNP?
False
Was Deposit Button Pressed?
Withdraw Money Function
Deposit Money Function

*Figure 1: Flow Chart of Software Design for ATM Robot.*

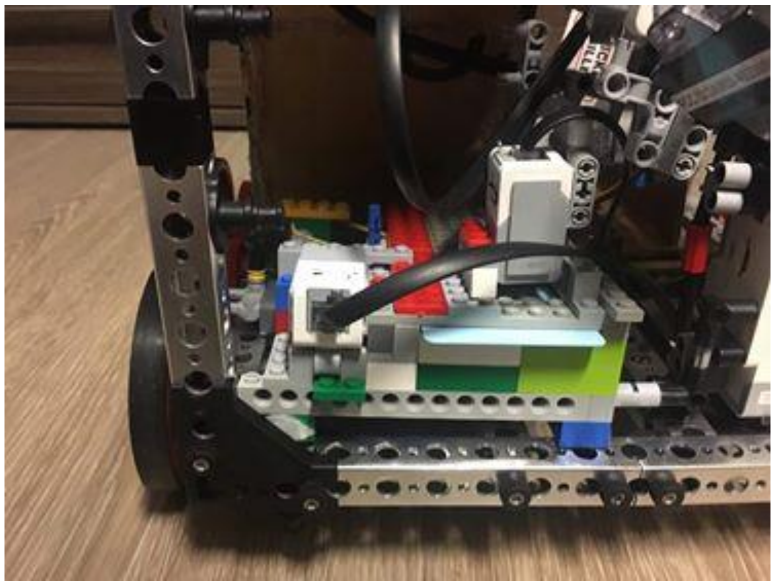


*Figure 2: Overview of card reader mechanism*

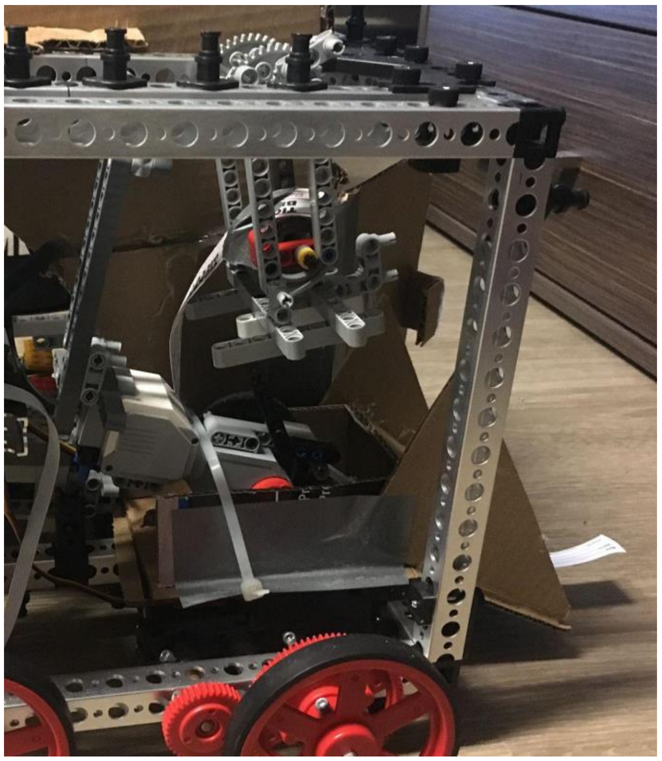


*Figure 3: Overview of the withdrawal mechanism*

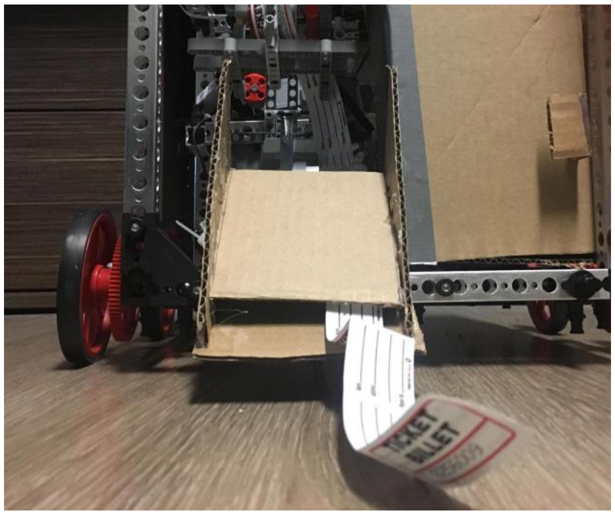


*Figure 4: Detailed view of ticket withdrawing slot*

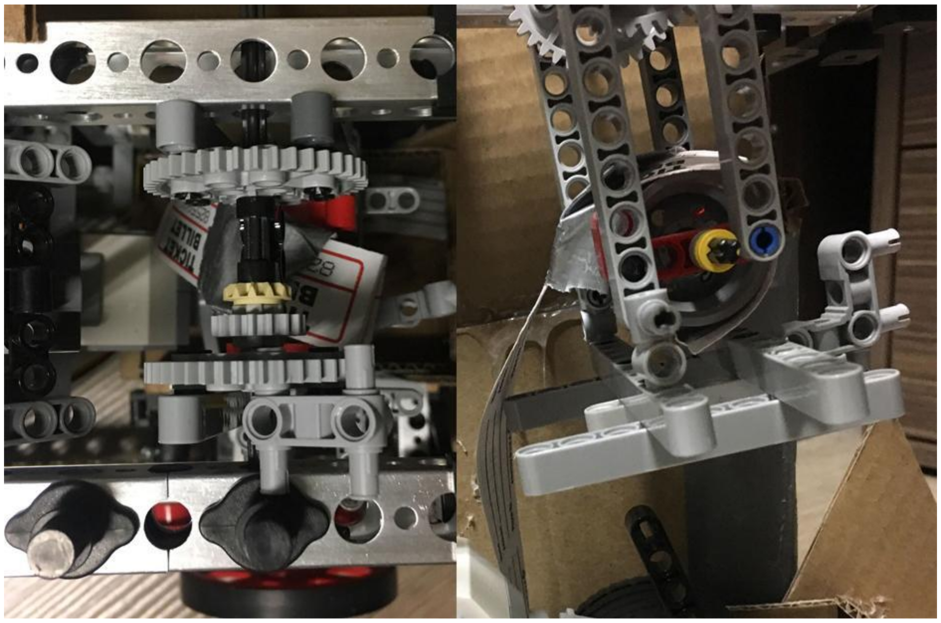


*Figure 5: Top and front view of the ticket rolling axle.*

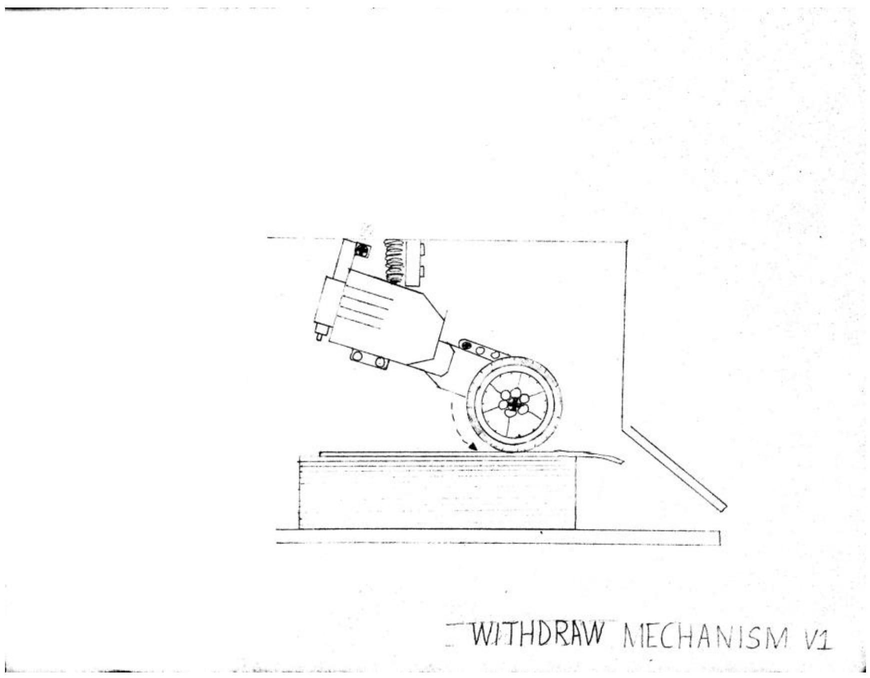


*Figure 6: Drawing of the initial design of the withdrawing mechanism.*

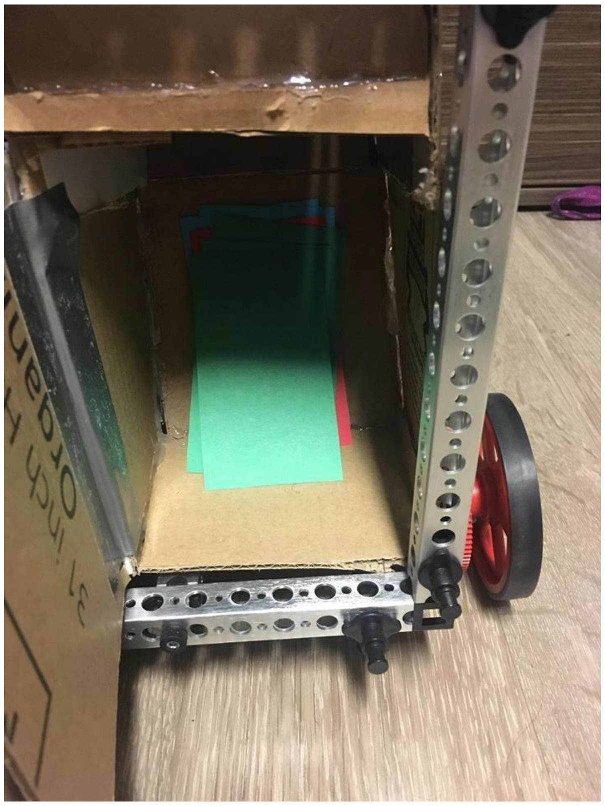


*Figure 7: Interior of the depositing mechanism with money stored at the bottom of the machine.*

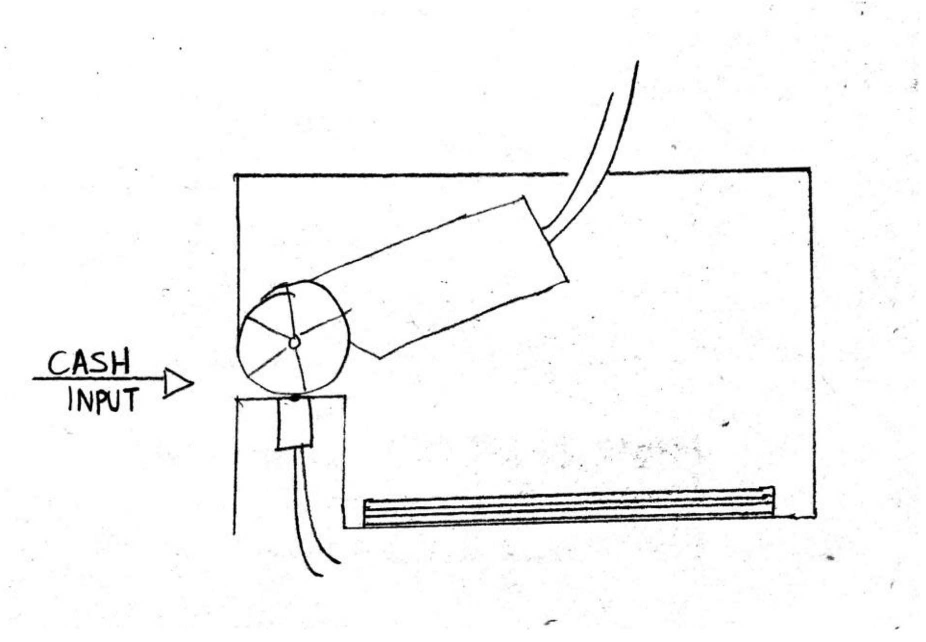


*Figure 8: Drawing of the initial design of the depositing mechanism.*

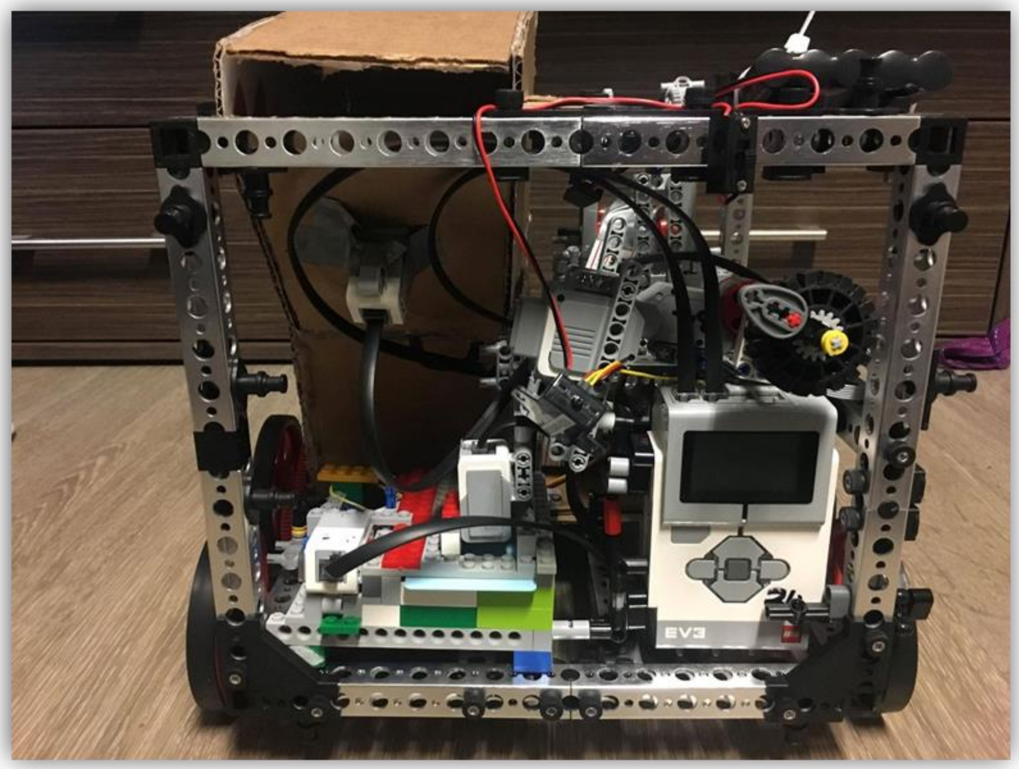
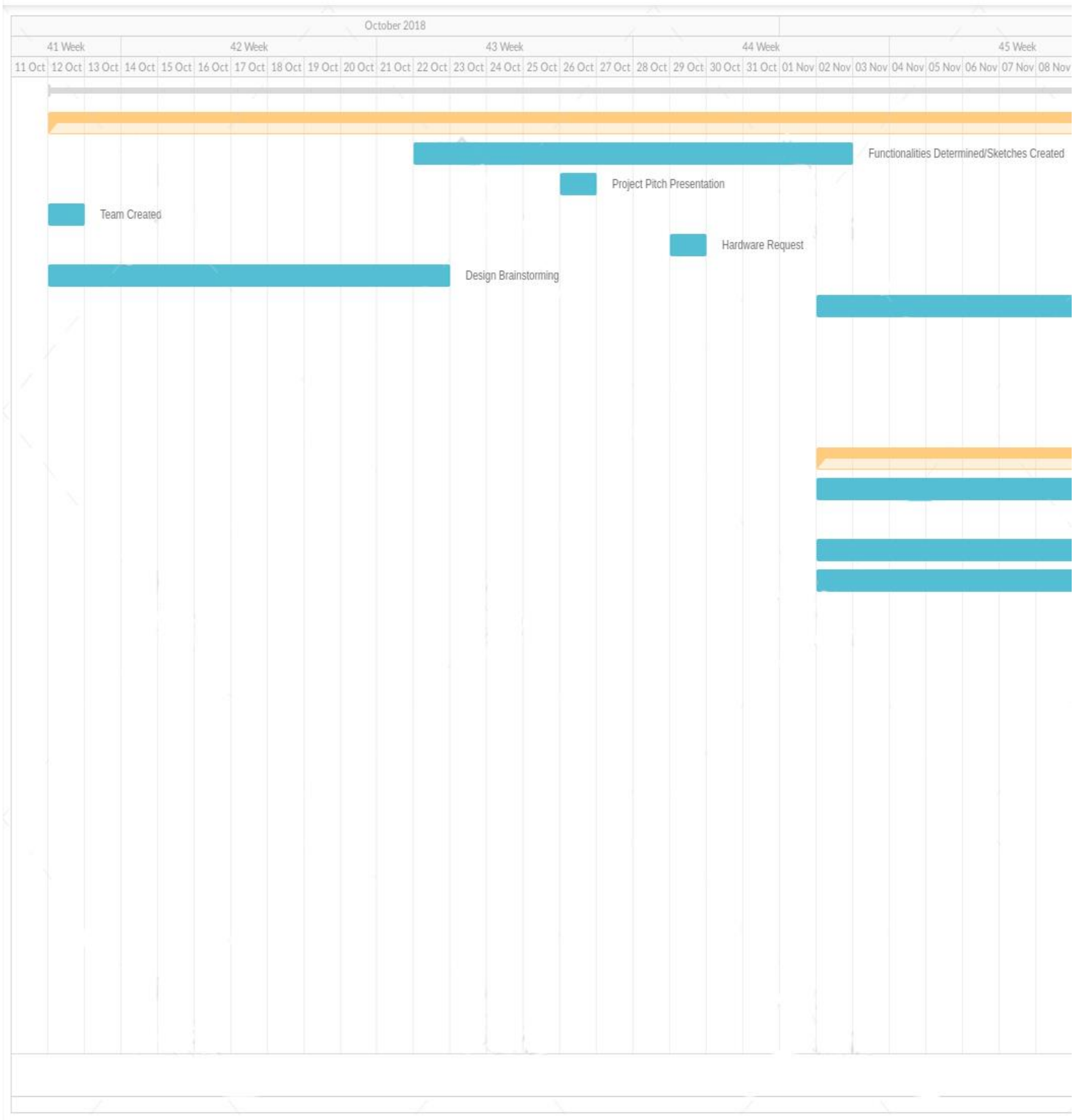
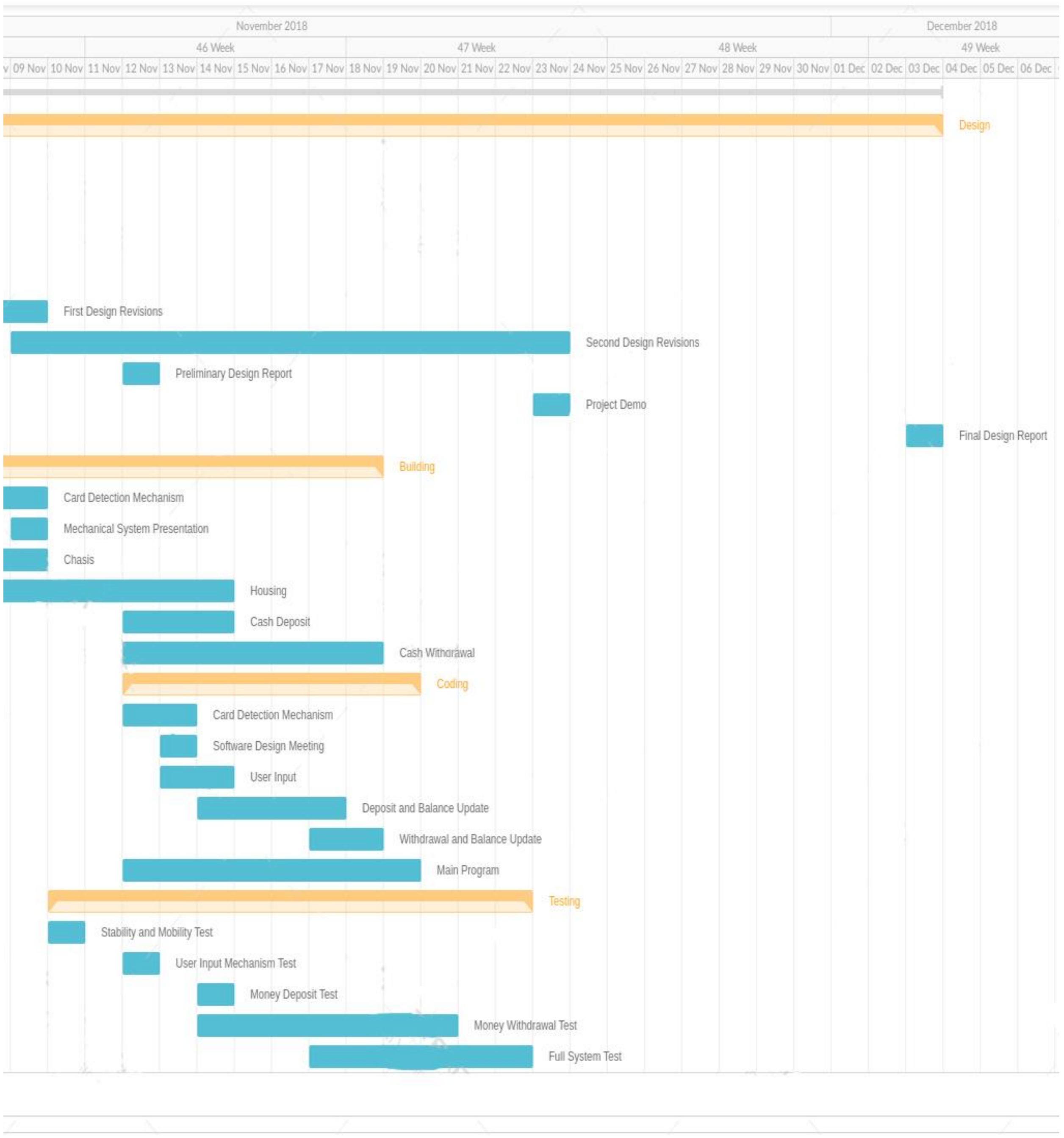


*Figure 9: Finished Assembly of the Automated Teller Machine*



*Graph 1. Gantt chart of the timeline of the project (October - MidNovember)*



*Graph 2. Gantt chart of the timeline of the project (MidNovember - December)*

# Appendix B

```
/* The following code was written by:
Dhruv Manani, Mahad Zaryab, Sarim Ali and Kyle Tam
[2018] All rights reserved.
*/

// declaring constants for account data:
const int MAXROW = 4;
const int MAXCOL = 4;

// function prototypes:
void displayMenu(int balance, int time);
void configureSensors();
int readCard (int *accountColors);
int depositMoney();
int getWithdrawAmount (int accountBalance);
void withdrawMoney(tMotor port, int amountWithdraw);
int getPinEntered(tMotor port, int pinDigit);
bool isCorrectPin (int *pin, int *correctPin);

task main ()
{
    // declaring account num and timer to be global variables
    int accountNum = 0;
    int timer = 0;

    configureSensors();
    /* assocating each account with a color for the bank card, an account
       balance, an account pin, and an account name by use of parallel
       arrays */
    int accountPins[MAXROW][MAXCOL] = {{1,2,3,4}, {5,6,7,8}, {1,3,5,7},
                                        {1, 0, 1, 0}};
    int accountColors[MAXROW] = {(int)colorBlue, (int)colorRed,
                                  (int) colorBlack, (int) colorWhite};
    int accountBalance[MAXROW] = {100, 100, 100, 138000};
    string accountNames[MAXROW] = {"Matt", "Carol", "Thomas", "Carol"};

    bool intruder = false; // declaring a bool to check for intruders

    while ((!getButtonPress(buttonUp) || !getButtonPress(buttonDown))
            && !intruder)
    {
        // opening messages and display
        displayBigTextLine(2, "  Welcome to");
        displayBigTextLine(4, "  the ATM");
        wait1Msec(1000);
        displayBigTextLine(8, "  Insert Card");
        displayBigTextLine(10, "  <<<<<<<<<<");
        wait1Msec(500);

        // This loop waits for a pin to be entered
        while (SensorValue[S1] == 1  && (!getButtonPress(buttonUp)
                   || !getButtonPress(buttonDown)) && !intruder)
        {
            eraseDisplay();
            int pinEntered[MAXCOL] = {0, 0, 0, 0};

            accountNum = readCard(accountColors);

            displayBigTextLine(3, "Welcome %s", accountNames[accountNum]);
            wait1Msec(1500);
            eraseDisplay();
            displayBigTextLine(3, "Enter Pin: ");
            wait1Msec(2000);
            eraseDisplay();

            int numAttempts = 3;
        bool pinVerified = false;

        // This loop allows for a pin to be entered in three times.
        // The pin is checked against the correct pin and if the pin
        // is incorrect more than three times, an intruder is declared
        // to be in presence

        while (numAttempts > 0 && !pinVerified && (!getButtonPress
               (buttonUp) || !getButtonPress(buttonDown)) &&
                SensorValue[S1] == 1)
        {
            for (int digits = 0; digits < MAXCOL; digits ++)
                pinEntered[digits] = getPinEntered(motorA, digits);


            if (isCorrectPin(pinEntered, &accountPins[accountNum][0]))
            {
                pinVerified = true;
                displayBigTextLine(3, "Pin Verifed");
                wait1Msec(2000);
            }

            else
            {
                numAttempts--;
                displayBigTextLine(3, "Incorrect Pin");
                displayBigTextLine(5, "You have %d", numAttempts);
                displayBigTextLine(7, "attempts", numAttempts);
                wait1Msec(2000);

            }

        }

        if (pinVerified)
        {
            time1[T1] = 0;
            while (SensorValue[S1] == 1 && (!getButtonPress(buttonUp)
                                        || !getButtonPress(buttonDown)))
            {
                // Timer is displayed simoltaneously
                timer = time1[T1]/1000;
                displayMenu(accountBalance[accountNum], timer);
                 /* Button up is associated with withdrawing money
                    and. Button down is associated with depositing
                    money
                 */
                if (getButtonPress(buttonUp))
                {
                    int withdrawAmount = getWithdrawAmount(
                                         accountBalance[accountNum]);
                    withdrawMoney(motorB, withdrawAmount);
                    accountBalance[accountNum] -= withdrawAmount;
                }
                else if (getButtonPress(buttonDown))
                {
                    eraseDisplay();
                    int amountDeposited = depositMoney();
                    accountBalance[accountNum] += amountDeposited;
                    displayBigTextLine(3, "Amount Deposited: %d",
                                       amountDeposited);
                }
                eraseDisplay();
            }

            // exit message
```

```
                    displayBigTextLine(2, "Your session");
                    displayBigTextLine(4, "today was:");
                    displayBigTextLine(6, "%d seconds", timer);
                    displayBigTextLine(8, "Good Day %s",
                                          accountNames[accountNum]);
                    wait1Msec(3000);
                    eraseDisplay();
                }

                else
                {
                    for (int numBlink = 0; numBlink < 4 ; numBlink ++)
                    {
                        displayBigTextLine(3, "Intruder Alert!");
                        wait1Msec(600);
                        eraseDisplay();
                        wait1Msec(200);
                    }

                    intruder = true;
                }
            }
        }
         /* if up and down button are pushed simultaneously, then the
            program ends and a message is displayed */
        for (int numBlink = 0; numBlink < 4 ; numBlink ++)
        {
            displayBigTextLine(3, "Shutting Down");
            wait1Msec(1000);
            eraseDisplay();
            wait1Msec(500);
        }
    }

    // display the Menu options onto display for user to choose from
    void displayMenu(int balance, int time)
    {
        displayBigTextLine(2, "Withdraw:UP");
        displayBigTextLine(4, "Deposit:DOWN");
        displayBigTextLine(6, "Exit - Remove");
        displayBigTextLine(8, "  Credit Card");
        displayBigTextLine(10, "Bal:%d", balance);
        displayBigTextLine(12, "Time: %d secs", time);
        wait1Msec(10);
    }

    // configures the sensors for the start-up
    void configureSensors()
    {
        SensorType[S1] = sensorEV3_Touch;
        SensorType[S2] = sensorEV3_Color;
        wait1Msec(50);
        SensorMode[S2] = modeEV3Color_Color;
        wait1Msec(50);
        SensorType[S3] = sensorEV3_Color;
        wait1Msec(50);
        SensorMode[S3] = modeEV3Color_Color;
        wait1Msec (50);
    }


    /* reads the card color and returns the account number associated with
       it
    */

    int readCard(int *accountColors)
```

```
{
    int colorCard = SensorValue[S2];
    int accountNumber = 0;
    for (int index = 0; index < MAXROW; index++)
    {
        if (accountColors[index] == colorCard)
            accountNumber = index;
    }

    return accountNumber;
}

/* Gets the amount to withdraw from the user and checks it against the
   account balance. If the amount requested to withdraw is less than
   the account balance, a zero is returned and a corresponding message
   is displayed in main. If the amount requested to withdraw is valid,
   then that amount is returned to main which is used to withdraw the
   money
*/

int getWithdrawAmount (int accountBalance)
{

    int amountWithdrawing = 0;
    bool backButtonPressed = false;

    while (!getButtonPress(buttonEnter) && SensorValue[S1] == 1 &&
            (!getButtonPress(buttonUp) || !getButtonPress(buttonDown))
             && !backButtonPressed)
    {

        int change = 0;

        while (getButtonPress(buttonUp))
            change = 10;

        while (getButtonPress(buttonDown))
            if (amountWithdrawing > 10)
            change = -10;

        while (getButtonPress(buttonBack))
            backButtonPressed = true;

        amountWithdrawing += change;

        eraseDisplay();

        displayBigTextLine(3, "Withraw: %.2f", amountWithdrawing);
        displayBigTextLine(5, "Press ENTER");
        displayBigTextLine(7, "to Confirm");
        wait1Msec(100);

        eraseDisplay();
    }

    if (amountWithdrawing <= accountBalance && !backButtonPressed)
        return amountWithdrawing;

    else if (amountWithdrawing > accountBalance)
    {
        displayBigTextLine (3, "Insufficient");
        displayBigTextLine (5, "funds in");
        displayBigTextLine (7, "account");
        wait1Msec(2000);
        eraseDisplay();
    }

    return 0;
}


/* This function is for the depositing mechanism of the robot. The
   color sensor reads the color of the bill and adds to the amount
   deposited accordingly. The amount deposited value is then returned
   to main to increment the account balance
*/

int depositMoney()
{
    int amountDeposit = 0;
    while (!getButtonPress(buttonEnter) && SensorValue[S1] == 1 &&
          (!getButtonPress(buttonUp) || !getButtonPress(buttonDown)))
    {
        displayBigTextLine(3, "Depositing...");
        wait1Msec(10);
        int change = 0;
        eraseDisplay();

        if (SensorValue[S3] == (int) colorRed)
        {
            displayBigTextLine(3, "$5 Detected...");
            wait1Msec(1000);
            change = 5; // I changed it in main
        }

        if (SensorValue [S3] == (int) colorBlue)
        {
            displayBigTextLine(3, "$10 Detected...");
            wait1Msec(1000);
            change = 10;
        }

        if (SensorValue [S3] == (int) colorGreen)
        {
            displayBigTextLine(3, "$20 Detected...");
            wait1Msec(1000);
            change = 20;
        }

        amountDeposit += change;
        eraseDisplay();

    }
    return amountDeposit;
}


/* This function spins the motor according to the number of bills. The
   encoder limit was determined based on multiple trials for one bill.
   This process is then repeated depending on the number of bills;
*/
void withdrawMoney(tMotor port, int amountWithdraw)
{
    nMotorEncoder[port] = 0;
    int num10Bills = amountWithdraw/10;

    for(int count = 0; count < num10Bills; count ++)
    {
        motor[port] = -10;

        while (nMotorEncoder[port] > -105*num10Bills)
        {
                displayBigTextLine(3, "Withdraw: $%.2f ", (float)
                                   amountWithdraw);
                wait1Msec(10);
                eraseDisplay();
            }

            motor[port] = 0;
        }
}

// this function check to see if whether the pin entered is correct
bool isCorrectPin (int *pin, int *correctPin)
{
    bool incorrectPin = false;
    for (int index = 0; index < MAXCOL && !incorrectPin; index ++)
    {
        if (pin[index] != correctPin[index])
            incorrectPin = true;
    }

    return !incorrectPin;
}

/* the following function is to recieve a pin digit from the user
   by using a knob they can spin (wheel/motor)*/
int getPinEntered(tMotor port, int pinDigit)
{
    nMotorEncoder[port] = 0;
    bool buttonPressed = getButtonPress(buttonEnter);
    /* this will be used to determine when the user wants to save the
    pin value shown on the screen */
    int savedPinValue = 0;
    int pinValue = 0;
    while (!buttonPressed && (!getButtonPress(buttonUp)
     || !getButtonPress(buttonDown)))
    {
        int pinValue = -nMotorEncoder[port]/40;
        motor[port] = 1;
        eraseDisplay();
        motor[port] = -1;
        /* the above, setting the motor to 1 and -1 immediately after
           is required to prevent the motor from braking (which
           prevents the motor from being manually rotated)
        */
        displayBigTextLine(3, "Pin Digit %d: %d", pinDigit + 1, pinValue);
        wait1Msec(1);

        if (pinValue < 0)
        {
            nMotorEncoder[port] -= 350;
        }

        if (pinValue > 9)
        {
            nMotorEncoder[port] += 400;
        }

        /* the above conditions prevent the user from entering values
           below zero or above nine which should not be pin digits
        */

        while (getButtonPress(buttonEnter))
        {
            buttonPressed = true;
            savedPinValue = pinValue;
        }

        /* the above waits for the user to press the enter button so
           it can store the pin digit displayed on the screen
        */
    }
    motor[port] = 0;
    return savedPinValue;
}
```