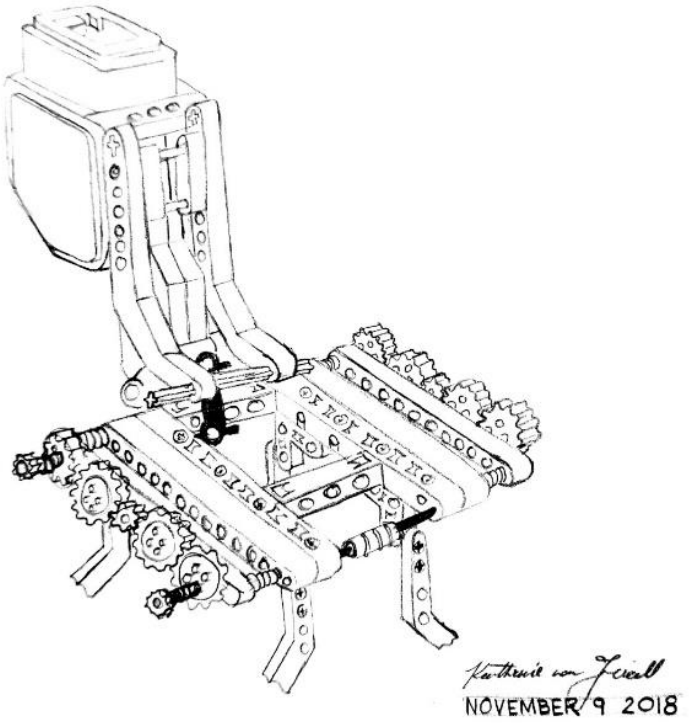
Inkbox: Final Project Report



Group 403

Urban Pistek, Julia Baribeau, Kathrine Von Friedl, Krishn Kiran Ray

20802700, 20792386, 20760403, 20759323

MTE 100, GENE 121

December 3, 2018

1. Summary

It was after careful consideration, and elaborate discussion that it was decided the project should be creating a automated temporary tattoo robot. The focus was to build a system that could consistently perform specific designs and operate in a fully autonomous fashion. Further, it was deemed necessary that the system would be easy to use, and appeal to the user. To remain within the scope of this project it was decided to only move in two dimensions. Additionally, the specialized tattoo ink that was being used did not need to be pressed onto the skin, rather it simply needed to fall onto the skin. Ensuring the safety of the user, features such as an ultrasonic checking mechanism and keeping the needle farther away from the arm were implemented. Constraints involved the use of only one ink colour, limited detail, and the area that the robot could operate in. Despite this, to meet the established criteria the drawing quality, durability and user appeal needed to be optimal. Mechanically the system involved a rectangular frame made with wood planks. Rack and pinions were made with CAD to fit the dimensions of the lego gears and controlled all the movement of a suspended inking module. Two long racks lay lengthwise with the frame and a smaller rack lay perpendicular within a internal module. Motors were connected to gears that controlled the movement along each direction. Suspended from the bottom of the module was a needle connected with a tube to a syringe. Ink was filled into the tube and syringe and was squeezed out by means of a motor-controlled syringe. The system was consistent and worked efficiently, yet there were improvements that could be made; particularly with the ink distribution mechanism. The software was designed to utilize one function that would control the movement of each axis using parameters using geographic coordinates, a distance and an angle. All movement was controlled with this function. Other necessary functions such as a zero function (module moves to a established zero position), ink control and configuring sensors were added. The main function was written so it it would wait for user input, while also checking if someone was present. Although most aspects of the robot were successful, it was evident that the ink distribution system could be improved in particular. Future designs could involve a screw method controlled by a motor. However, considering the limited time and resources available for this project, InkBot was extremely successful in meeting the projected and required goals.

Table of Contents

Table of Figures

2. Introduction

In recent decades, personal expression in the form of body art has become increasingly popular. Tattoos in particular have developed into a globally celebrated artform [1]. However, among the traditional tattoo, more temporary forms have gained popularity like henna or specialized tribal inks. To expand the practice of these temporary tattoos, companies such as Inkbox have began to manufacture tattoo alternatives inspired by various cultural traditions. The ink that Inkbox uses is made with an active ingredient extracted from the Genipa Americana fruit of South America and safely reacts with organic compounds in the epidermal layer of the skin to produce the dark blue colour of the tattoo [**source3**][**source4**]. They were inspired by the body ornamentation of the tribes in Darién Gap, Panama [**source1**]. Various cultures have been performing body art for extremely long periods of time, such as henna which has been documented to have been practiced over 9000 years ago [**source2**]. Yet, only recent years have seen the rise of temporary skin decoration from an international perspective (especially in North America). However, as inking becomes more popular, there is a lack of skilled artists to properly apply the ink. There lies the first problem InkBot aims to solve: a more accessible alternative to temporary tattoo parlors.

The use of this temporary ink gives people the opportunity to experiment without commitment. In principle, this alternative makes tattoos more accessible and reasonable for people. A portion of our primary objective centered around this idea; using knowledge regarding mechatronic systems, it was aimed to create a system that would make tattooing cheaper, safer, and more accessible. Automating the process in the long run can help make the process more convenient, efficient and possibly affordable. Further, this automation takes out the human factor. Humans experience fatigue when completing repetitive tasks. However, robots excel at repetition and therefore are capable of drawing multiple tattoos in an identical fashion. For example, drawing the same sequence of straight lines takes effort, practice and can only be performed so long before a person tires out [2]. Therefore, creating a mechatronic system to solve these limitations can be more efficient, and ensure the same, if not higher, quality of tattoo.

Moreover, the purpose of using a robotic system goes beyond efficiency and quantitative measurements. The use of a robot gives an extra flare to the process; since tattoos are considered art the use of a robot can be interpreted as a form of artistic expression. In short, our goal was to

create a machine that was more efficient and consistent in creating temporary tattoos. This is anticipated to make temporary tattoos more accessible, while also supporting body ornamentation as a globally rising artform.

While developing the system, many ideas were taken from the mechanics and form of 3D printers. In many ways InkBot's system resembles that of a 3D printer. However, it only draws in two dimensions because to draw in three dimensions is out of the scope of this project. The system consists of a frame, with two long rack and pinions running parallel lengthwise down the frame. Suspended on these rack and pinions was a module containing a motor and gear that moved another rack and pinion perpendicular to the ones running lengthwise. The entire module was controlled by another motor connected to a drivetrain permitting movement up and down these long racks. This setup gave the necessary two degrees of freedom to move. Also suspended from the module was a needle tip connected to a tube leading to a syringe. This tube and syringe was filled with the ink. A motor mounted to the frame and connected to the syringe controlled the flow of the ink. Sensors and the main brick controlling the system were mounted on the frame.

A software interface was written that took the users feedback and preformed specific checks to run the program. In particular, a function was written that controlled the direction and distance of movement for both motors. Using this function, designs were programed that the user could select through the interface. Other features such as a zero-function, checks for arm placement, and ink dispersion control were implemented to ensure the system could run smoothly and consistently and be able to respond to changes in the environment, specifically the human element.

As a result, valuable knowledge was gained relating to programming software to draw shapes, fluid controls, and mechanical movement; details regarding each component and sub component will be explored further.

3. Scope

The main function of the robot has not changed significantly through the design process. From its conception, the function of the robot was to apply semi-permanent tattoos to users. Through discussion, the tattooing area was narrowed down to just the forearm. The overall use for the robot, however, has not been altered.

The process for the application tattoos required several tasks and inputs. As stated in the interim report, the robot must receive a pattern, wait for an arm, move precisely to follow the design, and toggle the ink flow appropriately [2]. It also needs a shutdown procedure. However, in the implementation of the project, it was realized that the robot should return to a zero-point upon turning on, in case any movement occurred while the robot was off. A task was also added to check the type of user (student, professor, or TA) of the robot.

As InkBot turns on, the robot return to a zero-point, to ensure consistency of the starting point of the tattoos drawn. The robot zeros itself using two touch sensors, one on the x-axis and one on the y-axis. Each motor will run until the touch sensor in that axis is pressed, signifying that it has reached the zero-point for that axis.

Next, the robot waits for an input to check the user of the robot. Once it receives this information, it will print a message dependent on the type of user.

Then, the robot waits for a pattern selection with a button press. There are four pre-programmed designs. In the initial stages of design, there were considerations of allowing users to create their own simple designs by selecting directions of lines to be drawn. However, this was found to create too many variables, as the movement would have to be carefully monitored to ensure the user would only be tattooing over their arm.

The robot must then move over the skin, drawing the selected pattern. This must be precisely to ensure that the design comes out as desired. To ensure patterns are drawn in a continuous manner, slow movement of the motors and tracking of position using the motor encoders had been implemented.

Once the pattern has been drawn, the robot should perform its shutdown procedure. This will require only that the robot returns to an origin point. To do so, the robot will follow the same steps as described for the zeroing during the startup. At this time, the robot will shut down.

4. Constraints and Criteria

Table 4.1 Criteria Changes Since Interim Report

	Explanation		Measurement	
	Interim	Final	Interim	Final
Drawing Quality	How closely the drawn pattern matches the intended pattern	Customer satisfaction with their tattoo	Number of inaccuracies in the pattern	Scale of 1 - 10 with a rank of 10 being best
Strength and Stability	Strength and stability of the suspension system for the x-axis gearset and motor	No change.	mm of flex in the axle running through the motor	No change.
User Appeal	InkBot system must appear welcoming, reliable, and non-intimidating	No change.	Scale of 1 to 5, with 1 being very unappealing and 5 being very appealing	No change.

Table 4.2 Constraints

	Description	Why the constraint is needed	Justification	Change from Interim
Color Restriction	No more than one colour can be used in the tattoo designs. Results are limited to monochrome tattoos, with blue-black being the only tone available.	Constraint is based in resources. InkBox, the provider of the ink solution being used, only offers ink in this one colour. The fruit from which the ink is made, Genipa Americana, has a blue-black colour.	[2]	No change.
Y-axis range	The y-axis movement must span at least 100 mm.	This is the narrowest the y-direction movement can be to allow maximum arm width.	Measure-ments taken of Lego pieces	No change.
Detail Spacing	Tattoo designs cannot have details separated by a gap smaller than 5 mm.	Avoiding merging. Merging occurs due to the tendency of the ink to attract itself in short range. This constraint will help ensure that the tattoo matches the desired pattern, increasing customer satisfaction.	Determined by testing	Deemed unnecessary.

The constraints and criteria remained mostly fixed throughout the duration of this project with change occurring in only one criterion, drawing quality. Due to the viscous nature of the ink, the rate at which it flows from the tube is inconsistent. Due to this variability, each tattoo differed slightly. Despite this, it was found that many users of InkBot were still happy with their tattoos. Many users were fond of their unique tattoos, so it was determined that customer satisfaction is better measured by a ranking given directly from the user than by how closely the tattoo matches the intended pattern.

Our constraints and criteria helped guide our project to a successful final result. The criterion for strength and stability caused a need to re-evaluate the system of movement in the x-direction. The racks were moved inwards (*Figures 4.1, 4.2*) to reduce the distance the motor axle must span, minimizing flex in the axle and increasing stability.

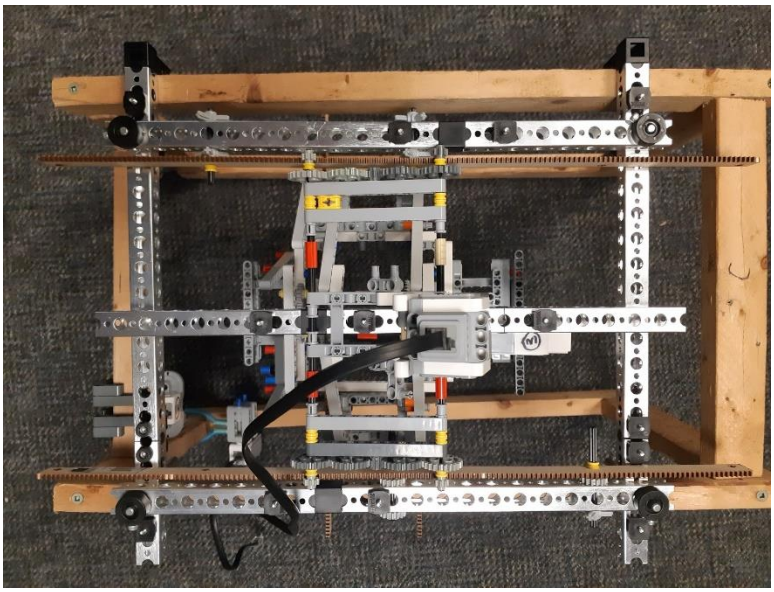


Figure 4.1

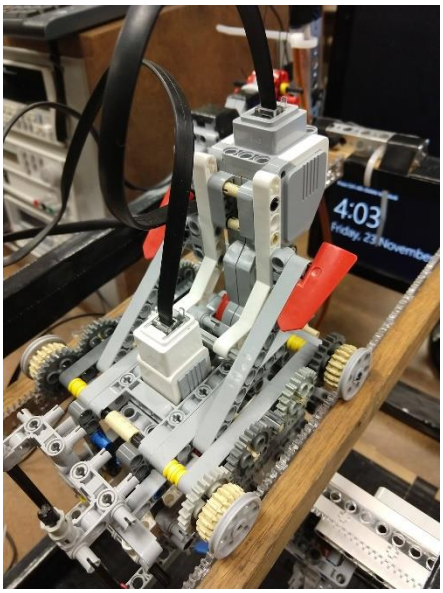


Figure 4.2

The criterion of user appeal was the motivation to cover the wooden frame with black duct tape. The smooth black coating eliminated the risk of splinters and improved the aesthetics of InkBot, making it appear more professional than bare wood. This made users of the system more comfortable with the machine, and willing to place their arm in it.

The constraint on ink colour guided our designs. When creating each of the four patterns, the constraints were kept in mind, so none of the designs were intended to have multiple colours.

The y-axis range constraint caused us to modify the system of movement in the y-direction. The design was changed from having the ink tip and support system move around on top of two racks (*Figure 4.3*) to having them suspended below a sliding rack (*Figure 4.4*). This modification allowed for a wider range of motion while retaining stability, letting us satisfy the constraint and span the full 100 mm needed.

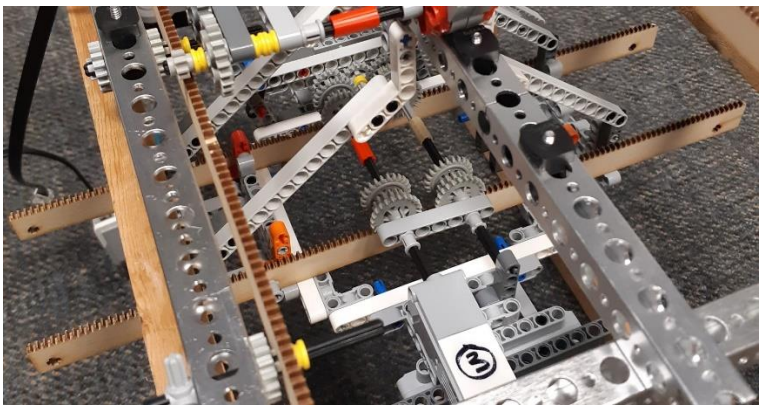


Figure 4.3

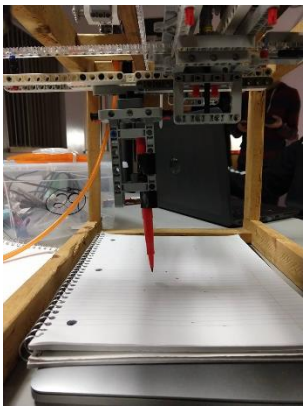


Figure 4.4

The constraint on detail spacing was not valuable when designing and implementing the project. None of the designs incorporated close details, so there was never a time when this constraint was referenced.

5. Mechanical Design and Implementation

Overall Design

The robot can be broken down into three major mechanical systems; the main frame, the internal module, and the ink distribution system.

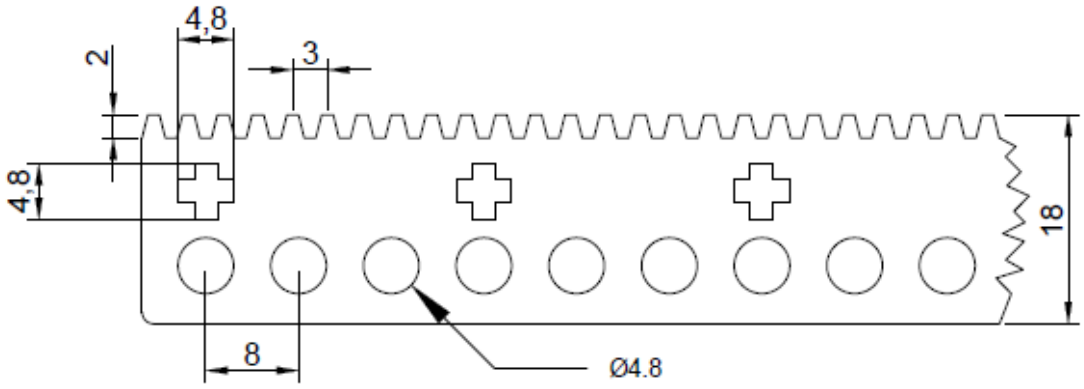
The Main Frame

The frame was built 1 ½ by ¾ inch wood planks connected with wood screws. The dimensions of the frame were 50 cm in length, 30 cm in width and 30 cm in height. These dimensions were selected so a person's arm could easily fit inside and there would be adequate space for all the mechanical components. The frame was built of wood, mainly because it is an inexpensive and easy to work with material, and it provided the necessary stability and strength. Mounted to the frame were the sensors, main lego brick and the long rack and pinion tracks that the robot would move along.

The Rack and Pinions

After discussions and research, it was determined that racks and pinions would be the best method to move the internal module. Measurements and online references were used to create a CAD model that matched the dimensions of the lego gears. In order to make the racks and pinions compatible with legos, round holes and cross holes were added lengthwise. These models were then cut out of ⅛ inch thick acrylic, and made 50 cm long to run lengthwise with the frame. The rack and pinions were mounted with wood screws through the laser cut holes into two pieces of wood. These pieces were then mounted onto short blocks of wood that were offset into the interior of the frame. This was done so that the racks and pinions were closer together; this resulted in better weight distribution for the internal module. Since most of the weight rested on two axles, shortening the length of those axles resulted in less sag and sturdier structure. These were the tracks which were then used to move the entire internal module.

Figure 5.1 AutoCAD Section of Rack and Pinion (All Dimensions in Millimeters)



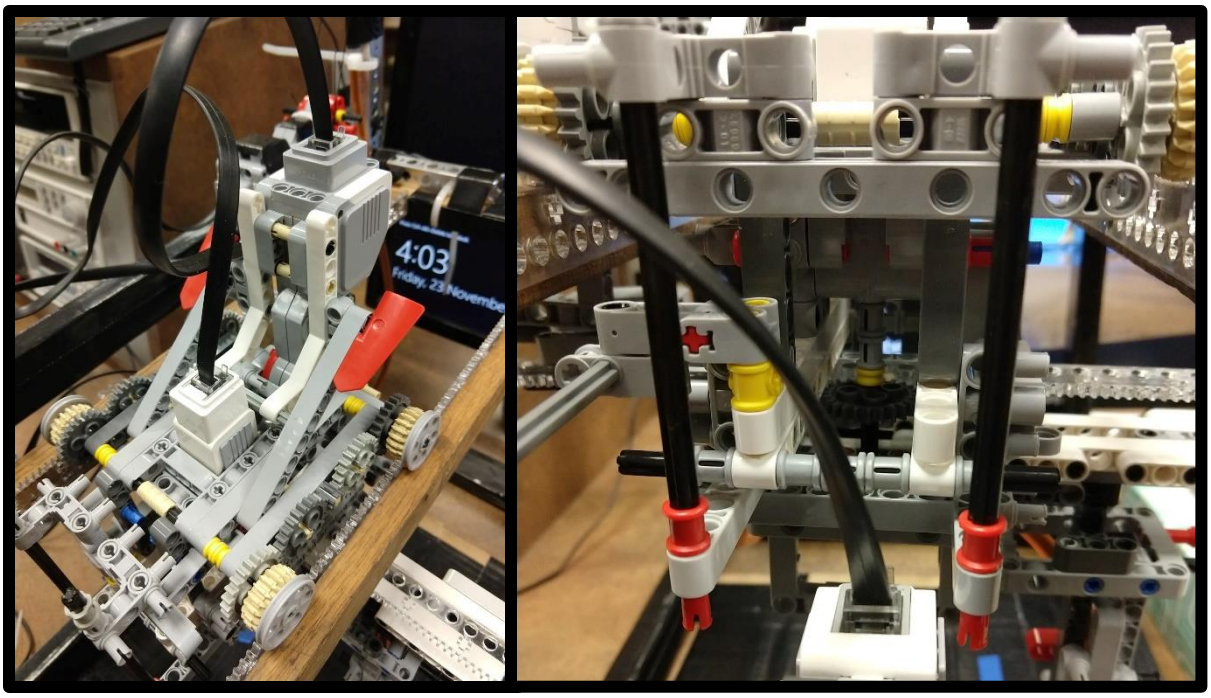
Sensor and Lego Brick Mounting

Sensors such as the ultrasonic, touch, and colour, along with the brick were mounted directly to the frame. These were fixed into position onto the frame; the brick was placed into a slot that on the top of the frame, the colour sensor was zip-tied to the frame and the touch sensor was mounted directly to the rack and pinion via legos. The colour sensor was used to identify the user via colour cards and the two touch sensors were used to establish a zero position.

The Internal Module

The majority of the mechanical components were contained within this internal module. It contained the motor that moved the entire module along the long rack and pinions (labelled as the x direction), which was connected to an axle which translated the motion through gears to the other axle. The motor was mounted to a lego frame from which the other sub components where attached. The medium motor was mounted facing downward through the center of the lego frame. This medium motor had a gear attached to the axle which moved the shorter rack and pinion in the direction perpendicular to the long rack and pinion (labelled the y direction). Attached to the rack and pinion was the mount that contained the needle which would dispense the ink. The entirety of this internal module was constructed of lego and was responsible for the mobility of the needle.

Figure 5.2 The internal module



X-Direction Movement

A large motor was mounted onto the frame of the internal module. This motor connected to an axle which moved two gears connected to the rack and pinions simultaneously. Two other gears were connected to the axle that translated the rotational motion through four gears to another axle, which had two more gear connected to the rack and pinions. Overall, there were four points of contact between the rack and pinions and internal module, all through these four gears. All the weight rested on these points, which was why the distance between the x-direction rack and pinions was made as small as possible. The axles were also supported by various lego pieces, and as a result the module had no issue supporting the weight and ran very smoothly. Circular guide pieces were placed on the exterior of the gears to prevent any lateral movement that may cause the module to run off the rack and pinions.

Y-Direction Movement

Suspended below the motor that moved the module in the x-direction was the mechanism that moved the needle in the y direction. The medium motor faced downward, from the middle of the module, and attached to the axle coming from the motor was a gear. This gear moved a rack and pinion laterally. The rack and pinion was held in position with legos and allowed to slide laterally. It was fixed in all other directions so nothing would get caught or move off the gear. Supported from this rack and pinion was a lego module that connected directly to the rack and pinion via the holes cut into it. This lego module was used to hold the needle dispensing the ink. Also mounted to one end of the rack and pinion was a touch sensor. This was used as the second point for the zero reference. The touch sensor was calibrated so it would be activated when the rack and pinion moved as far as it could to one side; this way the touch sensor did not take away from the full range of motion.

Needle Mount

At the bottom of lateral rack and pinion component was an area designated for attaching the swappable needle mount. Motivation for this design choice was to be able to quickly swap out a needle for a marker for testing. As a result, a mount was created for the needle and marker which simply snapped into place. This made testing very easy, and allowed for fast swaps and adjustments if needed.

The Ink Distribution

Mounted in the back corner on top of the frame was the system controlling the flow of ink. A syringe was strapped to a tetrix boxtube which itself was mounted to the frame. Mounted to the tetrix frame was a large motor. The syringe had a string hot glued to the top of the plunger, which was then threaded through holes drilled through the flanges on the side of the syringe. The string was then attached to a spool connected to the motor. As the motor spun, it pulled on the string which forced the syringe plunger down. Contained within the syringe was the ink used for the tattoo, then connected to the syringe was a tube leading to the needle. The tube was also filled with ink, so when the top of the syringe was pulled down a force was applied to the ink causing it to come out of the needle end. The motor moved at a slow constant speed so that the ink flowed smoothly. Through testing it was discovered that once pressure was applied, ink

continued to flow for a period after even after the motor ceased applying pressure. As a result, the motor was used to apply pressure for only a brief period for time so that the ink did not continue to flow after the design has been completed.

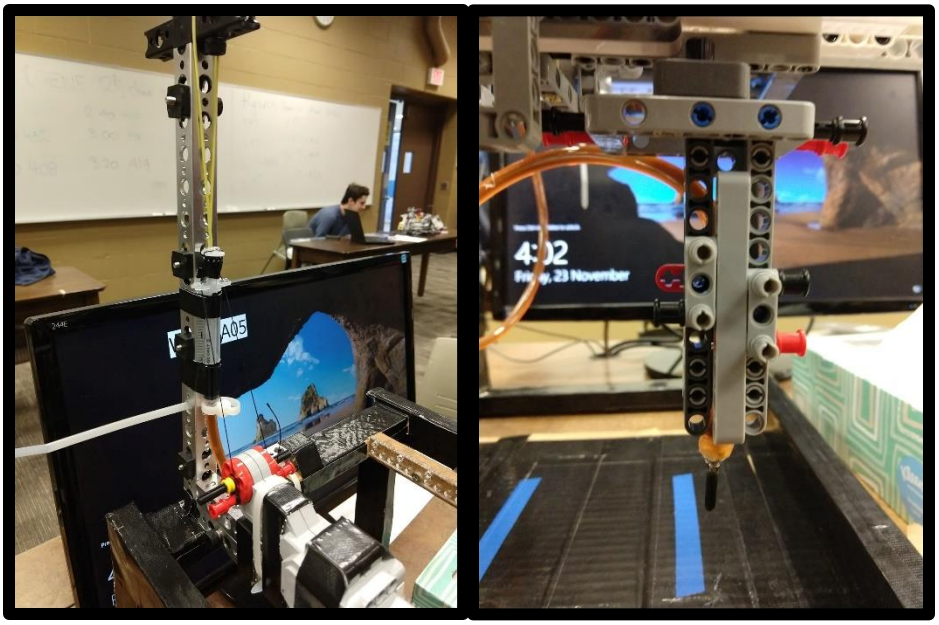


Figure 5.3 The Ink Distribution System

Design Decisions and Trade-Offs

The major design decisions that were implemented are the following:

1. Rack and Pinions for Movement
    a. **Benefit:** Since the rack and pinions needed to be fabricated they were made using AutoCAD to fit to specific dimensions. This allowed to tailor them to the exact specifications required. Consequently, they meshed perfectly with the lego gears. This close fit with the rack and pinion system ensured precise control of movement. Precision was crucial to ensure this project would be successful.
    b. **Tradeoff:** Since the rack and pinions needed to be completely fabricated from scratch, they required a fair amount of extra work compared to other methods.
2. Using a string and spool for applying pressure to the ink
    a. **Benefit:** The system was very simple to set up and worked relatively well. Since strings were used to compress the syringe the motor could be further away from the syringe, making the mounting easier. The strings were threaded through two

holes in the flanges of the syringe, then connected to the motor. The syringe was mounted to a tetrix boxtube and the motor was mounted to the frame. Keeping the two components separate made it very easy to swap out and kept the mount simple. Furthermore, this method was fairly effective at outputting the ink at a constant rate.

   b. **Tradeoff:** Although the string and spool was fairly effective, it had a few drawbacks. While the syringe was pulled it did not move the shaft pushing down on the ink inside at a constant rate. Rather, it often jumped forward in small bursts. Although this was not a major issue, sometimes it resulted in larger burst of ink to be distributed out. This made controlling the amount of pressure applied to the ink more difficult, as these jumps added extra pressure that lingered on even after the program finished.

3. Having the syringe mounted on the frame and not directly on the module
   a. **Benefit:** Having the syringe mounted on the frame, greatly simplified the complexity of the internal module. Firstly, it reduced the weight the module needed to support. The syringe itself was fairly light, but the motor would add a lot of weight. Moreover, the motor along with the ink would need to be mounted to the internal module, adding even more weight. Along with the weight, this would complicate the design, and make the module larger.
   b. **Tradeoff:** Since the syringe was mounted away from the internal module, a tube was required to connect the syringe to the needle mounted to the internal module. Using the tube complicated matters in regards to transporting the ink. Air bubbles would sometimes form in the tube, resulting in small gaps that disrupted the steady flow of ink. The tube needed to be prefilled to limit the possibility of air bubbles forming. Further, the syringe was not retracted to prevent the vacuum from sucking the ink back into the syringe from forming bubbles. This complicated the process of refilling ink.

6. Software Design and Implementation

The program was broken down into several blocks: the starting procedure, the pattern selection and drawing, and the shutdown procedure. This breakdown was determined during discussion, finding that these were logical places to define separations.

Starting Procedure

First, the sensors are configured in the function configure sensors. This function takes no parameters and returns nothing. For the robot, an ultrasonic sensor was set up as well as two touch sensors, and a colour sensor set to colour mode.

Second, the robot is reset to a zero-position. This is done using the zeroBot function. This function takes no parameters and returns no values. First, the x-direction motor is turned on in the negative direction until the touch sensor mounted on the x axis is pressed. This signifies that the robot has reached 0 in the x direction. The x-direction motor is then turned off. Then, the same thing is done for the y-axis.

Next, the program follows the function identification. There are no input parameters, as this function simply waits for a button press and takes a reading from the colour sensor. This is a function that returns an integer depending on the input it receives from the colour sensor. The sensor takes an input of a color and returns an integer depending on the colour it reads. The sensor returns one for red, two for blue, three for yellow, and zero for anything else. Once the integer value is returned, it is used to print a welcome message on the robot's display.

In the main program, the string "Welcome" is displayed on the robot's screen. However, the returned value from the identification function adds a string on to the end of this. "Professor" is added if a value of one is returned, "TA" for two, "student" for three, and an empty string for zero. For example, if the colour sensor reads red, the string "Welcome Professor" would be displayed on the robot's screen.

Table 5.2.1: Starting Procedure Programming

Function	Programmer
configure_sensors	Urban Pistek
zeroBot	Krishn Kiran Ray
Identification	Kathrine Von Friedl

Pattern Selection and Drawing

After the welcome message is displayed, the next step is to wait for user input for pattern selection. This is done with a while loop. This loop has a two-fold end condition. First, the loop can end if a pattern is selected and then drawn on the user's arm. Second, the loop is exited if 25 seconds pass without selection of a pattern. Pattern selection is done using the buttons. If the left button is pressed, a cube is drawn. If the right button is pressed, "UW" is drawn on the user's forearm. The up and down buttons select the arrow and square patterns respectively.

The patterns themselves are all functions that take no parameters and return no values. "UW" is drawn using two individual functions called drawU and drawW separately, but the other designs are all single functions (drawCube, drawArrow, and drawSquare). All five of these functions use two other functions that have been written: draw and inkOn. These functions are used to follow each of these patterns and turn the ink on and off when necessary.

The draw function takes several parameters and returns one value by reference. The parameters are a boolean for positive or negative in the x-direction (east-west), the angle of the line (between 0 and 90, inclusive), a boolean for positive or negative in the y-direction (north-south), the length of the line (in millimeters), and a boolean variable passed by reference to check if the previous line drew correctly. The function works by first checking if the user's arm is in an acceptable range, defined by two constants. The robot gets a value from the ultrasonic sensor and ensures that this value is between those two constant values determined to be a safe range through testing. If the arm is not within that safe range, then the boolean passed by reference will be set to false, which will cause all successive draw commands to be skipped as well. However, if the arm is within the range, then the function will check the booleans for north

and east. If one of these booleans is false, then a scalar value will be changed to negative one so that the motor power will be negative.

Then, the angle and length of the line are used to calculate the distance the robot must move in each direction, using trigonometry. For example, for the x-direction, the distance is calculated using length*cosine(angle). Power for each motor is calculated similarly, substituting the constant for motor power for the length, and multiplying by the scalar found when checking the boolean values. Then, the motors are set to these calculated powers until the robot has traveled the length of the line, checked using the pythagorean theorem ($x^2+y^2=length^2$). Once this distance has been traveled, the motors are turned off.

The inkOn function simply toggles the ink motor. It does this using a boolean parameter which turns the ink on if it is true or off if it is false. It does not return any values. The ink is turned on by turning on the motor attached to the syringe. It is turned off by turning the ink motor in reverse for 500 milliseconds at 5 times the forward power then turning off the motor.

At the end of each of the patterns, the function inkEnd is called. This function takes no parameters and returns no values. The ink motor is set to -25 power for 500 milliseconds so that the ink is pulled back into the tubing.

Table 5.2.2: Pattern Selection and Drawing Programming

Function	**Programmer**
drawU	Kathrine Von Friedl
drawW	Kathrine Von Friedl
drawCube	Urban Pistek
drawArrow	Julia Baribeau
drawSquare	Krishn Kiran Ray
draw	Julia Baribeau
inkOn	Kathrine Von Friedl
inkEnd	Krishn Kiran Ray

Shutdown Procedure

The shutdown procedure is fairly short and simple. First, the robot is zeroed using zeroBot. Then, a string is displayed on the screen. After a short wait, the program ends.

Table 5.2.3: Main Function Programming

Function	**Programmer**
main	Krishn Kiran Ray

Design Decisions

Several design decisions were made regarding the software. Though these were made to improve the code, some came at a cost.

First, it was decided to zero the robot in two places. This was to account for any movement the robot underwent while it was off. This also ensured that at the beginning of every design, the robot was always at the origin point.

Second, a variable was created that would be passed in by reference to the draw function as a way to ensure that lines would not continue to be drawn if the user's arm was removed. This variable allows this because, if it is changed to false, then all subsequent calls of the draw function will not draw the line and instead skip to the end. This will cause the robot to skip to the end of the program and shut down.

It had also been considered to add an option for users to create their own unique design by providing input that would draw their lines. However, this would require that the robot be controlled on all four sides to ensure that no user-created pattern would cause the robot to go outside of the correct drawing area or get caught against the frame in one direction and cause the motors to burn out.

Testing

Owing to the involved nature of our design, testing was performed only after most of a mechanical system created. The rack and pinion had to set up on the first axis before there the movement could be tested. This was also done for the other axis. Once these two axes were set up it became very simple to test each of the drawing patterns.

This began by testing the drawing in each of the dimensions individually. A marker was connected to the internal module with a paper underneath to be draw on. To check that results were valid lines drawn by the draw function were checked if they were in the correct orientation and length. Then, the same check was performed for lines at angles, ensuring angles were correct as well. For all of these tests, the tolerance of error was variation by ±1 millimeter in position and straightness.

Then, similar tests were performed with completed patterns. Often, it was found that the robot was drawing in the wrong direction for a desired line, which meant that the angle or the direction of line needed to be changed to a negative value. However, with some trial and error, the patterns ended up meeting the desired tolerance.

Once the patterns were working correctly with the marker, ink was added. This was a complication, as the ink did not flow as well as anticipated. This resulted in a need for repeated tests with varying powers for the motors controlling the ink and movement of the module. This complication was not completely fixed. Though there was significant improvement in the quality of the drawn designs, completely clean lines were achieved with a pen attachment but not the ink.

7. Verification

Table 7.1 Testing

	How the Constraint was Met
Color Restriction	Upon receiving a sponsorship from Inkbox, only black freehand ink was loaded into the multiple syringe attachments. For this reason, all tattoos featured the black/blue ink.
Y-axis range	The system spanned 100mm due to its mechanical design which featured a moving base attached to gear set that had a maximum span of approximately 35. This value was determined by using a pen apparatus mounted to the bottom of the robot (where the ink distribution tube sat), and testing how far the rack and pinion system in the y direction could move before experiencing interference.
Detail Spacing	InkBot had achieved this constraint, despite it being made obsolete. Individual functions used to construct the four designs (cube, UW, square and arrow) had no checks for lines less than 5mm. Therefore, CAD was implemented to draw out designs with no lines coming within 5 mm (unless merging). This was overseen by human inspection. Using these designs, functions were written to draw lines identical to those used in AutoCAD. Line length and angles were kept constant, in doing so only the AutoCAD drawings needed to meet physical constraints, as the code was written with respect to the drawn files.

8. Project Plan

Table 8.1 Assigned Tasks

Assigned Tasks

Kathrine:
• Arranged sponsorship with InkBox
• Graphics and Physical System Designer
  ○ Created detailed sketches of design prototypes
• Mechanical Fabrication
  ○ Created the main drivetrain for the internal module to move in the x direction
• Colour Identification, Draw U, Draw W, InkOn Functions

Krishn:
• Code Design
  ○ Conceptualized overall flow charts
  ○ Responsible for merging all functions into main
• Syringe System Assembly
  ○ Modified syringes for use
• Robot Maintenance and storage
• Zero, Draw Square, Ink End Functions

Julia:
• Code Design
  ○ Tested the code to ensure it worked well
  ○ Created the draw function which was responsible for all movement
• Syringe System Design
  ○ Prefilled syringes and tubes
  ○ Syringe mounting and tube attachments

- Finished Frame Design and Implementation
- Draw and Arrow Functions

Urban:

- Mechanical Design
    - Constructed y directional (lateral) movement component of the internal module
- Rack & Pinions
    - AutoCAD Model
    - Got required materials and laser cut
    - Mounted to frame
- Frame fabrication
    - Used saws to cut wood, joined together with wood screws
- Configure Sensors, Draw Cube Functions

Revisions to Project Plan

For the most part the project plan stayed consistently the same, although small changes were created, either because new discoveries lead to a better way of doing things or time constraints.

1. Initially the plan was to use some kind of screw method for the ink distribution system. A threaded bolt was to be mounted to the rubber seal on the inside of the syringe. Then a nut needed to be mounted to the opening of the syringe, the bolt would pass through this nut. Next a motor needed to be attached to the bolt to spin it and the whole system had to be fixed into position while it could still up and and down. During the initial prototyping of this it was discovered how complicated this approach would be. It was decided that a simpler solution would be appropriate.
2. At first there were three designs to choose from; UW, square and cube. During testing there were struggles with determining the timing and the flow rate of the ink. It was hypothesized that an arrow design could work well with the main issues that InkBot was facing; sometimes the ink came out in a very curly pattern.

Therefore the arrow was coded and implemented, and through testing it proved to work well even if aspects of the robot did not work well. Through testing and optimization these issues were minimized, however the arrow design was still kept for its reliability.

Comments on Projected VS Actual Timeline

Overall the actual completion of tasks compared to the planned finish date was relatively consistent. Yet, there were a few delays and changes to the order of completion of tasks. Firstly, the software design was finished on time and tested appropriately. The main mechanical design, completion of the internal module and rack and pinions was also on schedule. Delays were present in regards to the syringe system. Firstly, the materials had to be acquired which were not readily available; syringes were found at a pharmacy and the tubing along with zip ties and duct tape were found at a home depot. After the materials were acquired, the first prototype of the system was constructed quickly; however, there were multiple revisions before the concept was finalized. Also, testing of the full system was delayed slightly because it took some time before InkBox responded with the sponsorship that provided the project with more test ink.

9. Recommendations and Conclusions

9.1 Mechanical Recommendations

If given the opportunity to expand the mechanical design of InkBot, significant improvements would be made to the ink distribution system. During the assembly, testing and demoing of InkBot, it was evident that there were three significant problems with the way in which ink was applied. These problems included ink overflow after designs were completed, air bubbles being present in the tubing causing designs to come out partially complete, and the precision of speed at which ink is poured.

The first concern was an overflow of ink that came as a result from the insufficient means of reverting ink flow. The final design of InkBot depended on a syringe working in the opposite direction (being pulled up) to effectively retract the ink back into the tube. While this was a good idea in theory to revert ink flow, the way in which this idea was implemented had flaws. To draw back the ink, a rubber band was attached to the end of the plunger of the syringe. When motor C, which was attached to the end of the syringe was set to a power of 0, the elastic potential energy of the rubber band was the only force acting on the plunger in the upward direction. This was not nearly a significant amount of force for the syringe to effectively retract ink. To improve the performance of this system, more rubber bands could have been added to increase the amount of elastic potential energy. Another way to solve this method is to modify the string which was attached to the syringe. If the string was run through a topmost suspension, and was also tied to the side extensions of the syringe, the string could have been used for both upward and downward movement of the syringe. This is achieved by using a system similar to threads which may be gripped and therefore pushed and pulled.

Air bubbles also presented several difficulties in the completion of tattooed designs; this came as a result of frequent refills in which air was introduced to the system. One possible way of addressing this problem would be to use larger capacity syringes that can be prepared with a greater volume of ink. In this way, air bubble frequency would significantly decrease as refills are reduced. The ideal solution would be to install a one way valve that could be manually blocked while the system is operational. In doing so, there would be no leakage of ink even as pressure increases (when the tube assumes the collapsed position).

The last challenge InkBot faced was the accuracy and speed at which ink left the tube. Motor C, which was responsible for controlling the syringe was operating too quickly at certain instances even when set to power 1.

A possible solution to this would be to use a screw-in plunger. The plunger of the syringe would function exactly the same except instead of operating using strings and a rubber band, the vertical movement would be controlled by a shaft extending from a suspended motor which features a threaded extension. This extension would pass through a nut and as the motor would rotate, the shaft would spin causing an upward movement to occur in the syringe body. In doing so, the motor would distribute ink at a much slower and controlled pace.

Combining these ideas together, the ideal mechanical modification would be a redesigned ink distribution module. A screw method shaft will screw into a large syringe body with a one way valve at the topside of the plunger. This design is hypothesized to greatly decrease the severity of the three aforementioned ink distributing concerns.

9.2 Software Recommendations

There were some hypothesized stretch design choices which weren't pursued due to the time frame of the project.

One idea had been to make InkBot a customizable tattooing machine. Using a colour sensor to scan combinations of coloured paper sheets, users could construct words or patterns. Each colour would correspond to a specific character. This idea was initially proposed during the design phase of InkBot but was omitted due to complexity and the limited amount of time available to program and test this feature. This would be done by assigning a limited amount of values temporarily to an array (user input). The array would then be read and appropriate functions for various characters would be called respectively.

To make InkBot more industry focused and user friendly it would be ideal for users to see a larger display. For this reason, InkBot could be outfitted with a program that constantly updates a file with real time visuals of the inking design being produced. This monitor may also include all text displayed on the EV3 brick featuring design selection and greetings.

Given the resources, the ultrasonic sensor used for arm detection would have been replaced with a touch sensor. It was discovered during testing that the ultrasonic was sometimes unreliable; it would read a distance far greater than it should have, causing the system to assume

an arm is not present and go into shutdown mode. One hypothesis of why this would happen is due to the slant of a user's upper arm deflecting the ultrasonic waves away from the sensor. A touch sensor to determine arm placement would be much more consistent and reliable, making our system more robust for industry use.

10. Conclusions

Overall, InkBot was a success. The project was successfully designed and implemented a solution to the initial problem: creating an autonomous, reliable temporary-tattoo system which is safe for the user.

The system satisfied all of the initial criteria and constraints. It output quality drawings which users enjoyed, was adequately strong and stable, and was appealing to users as a non-threatening device. Each tattoo consisted of one colour, fell within the proper range for average width of a forearm, and had safely spaced details.

The mechanical design made appropriate use of wood, laser-cut acrylic racks and pinions, lego pieces, and tetrix pieces. The ink tip had controlled motion with two degrees of freedom. Though there was some difficulty controlling the ink precisely, tattoos were created with satisfactory ink distribution.

The software used functions to maintain efficiency and user input to create a welcoming, personable user interface. The startup and shutdown procedures ensured consistency in the placement of each tattoo, and the draw function allowed for logical design of each of the four tattoo patterns.

A lot was accomplished in the design and development of InkBot. Many students and staff members alike gathered to the system to receive a tattoo, and a reminder of the InkBot project will linger on UW's skin for approximately two weeks until the ink fades.

11. References

[1] W. Heywood, BA(Hons), "Who Gets Tattoos? Demographic and Behavioral Correlates of Ever Being Tattooed in a Representative Sample of Men and Women," Annals of Epidemiology, vol. 22, no. 1, January 2012, pp. 51-56. [Online] Available: Annals of Epidemiology, https://www.annalsofepidemiology.org/articleS1047-2797(11)00287-0/pdf Accessed: Nov. 11, 2018.

[2] *Inkbox Freehand Ink;* SDS [Print]; Inkbox Ink Inc.: Toronto, ON, Canada, Jan 12, 2018. Accessed: Nov. 23, 2018.

[3] Inkbox Ink, Inc, "Freehand Ink," Inkbox Ink. [Online]. Available: Inkbox Ink, https://inkbox.com/freehand-ink#shopFreehand Accessed: Nov. 11, 2018.

[4] I. Moazzam, "A brief history of henna," *The Express Tribune*, 07-Aug-2014. [Online]. Available: https://tribune.com.pk/story/741476/a-brief-history-of-henna/. [Accessed: 02-Dec-2018].

[5] J. Baribeau, et al., "Final Project Interim Report: IntBot," University of Waterloo: Waterloo. Rep. # 1, 2018.

11. Appendices

11.1 Appendix 1: Flowcharts

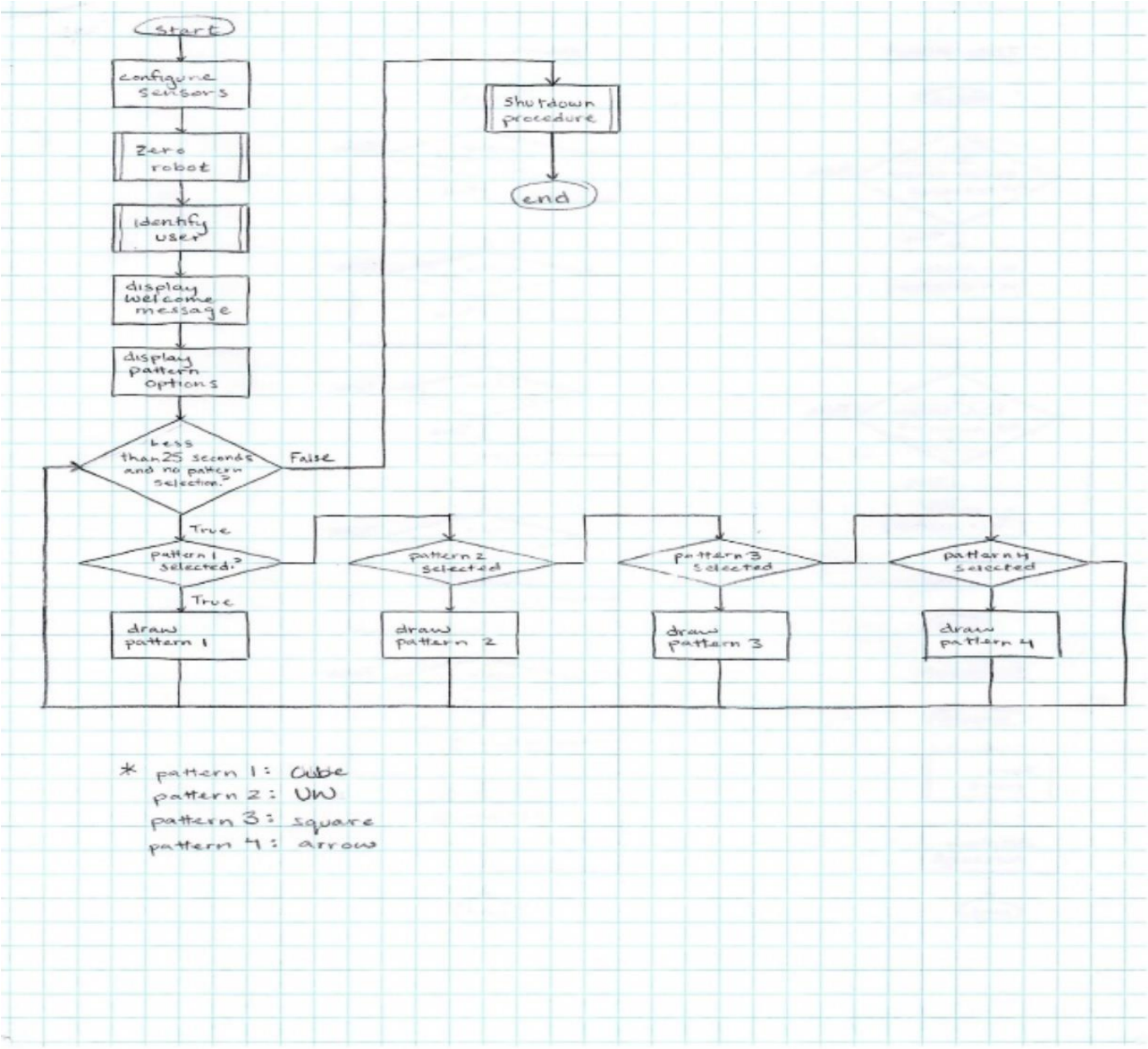


Figure 11.1.1 Main Flowchart

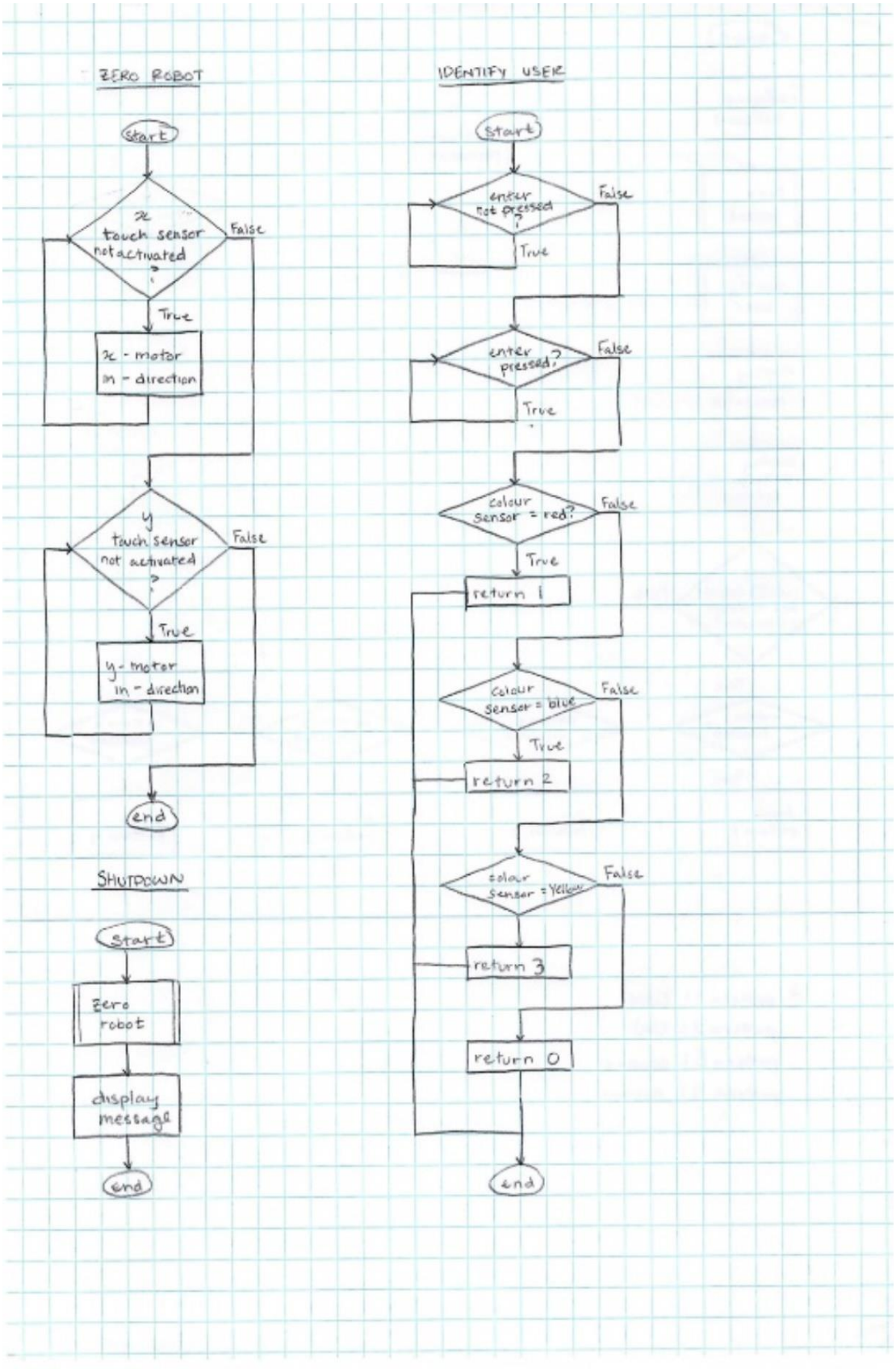


Figure 11.1.2 Startup Flowcharts

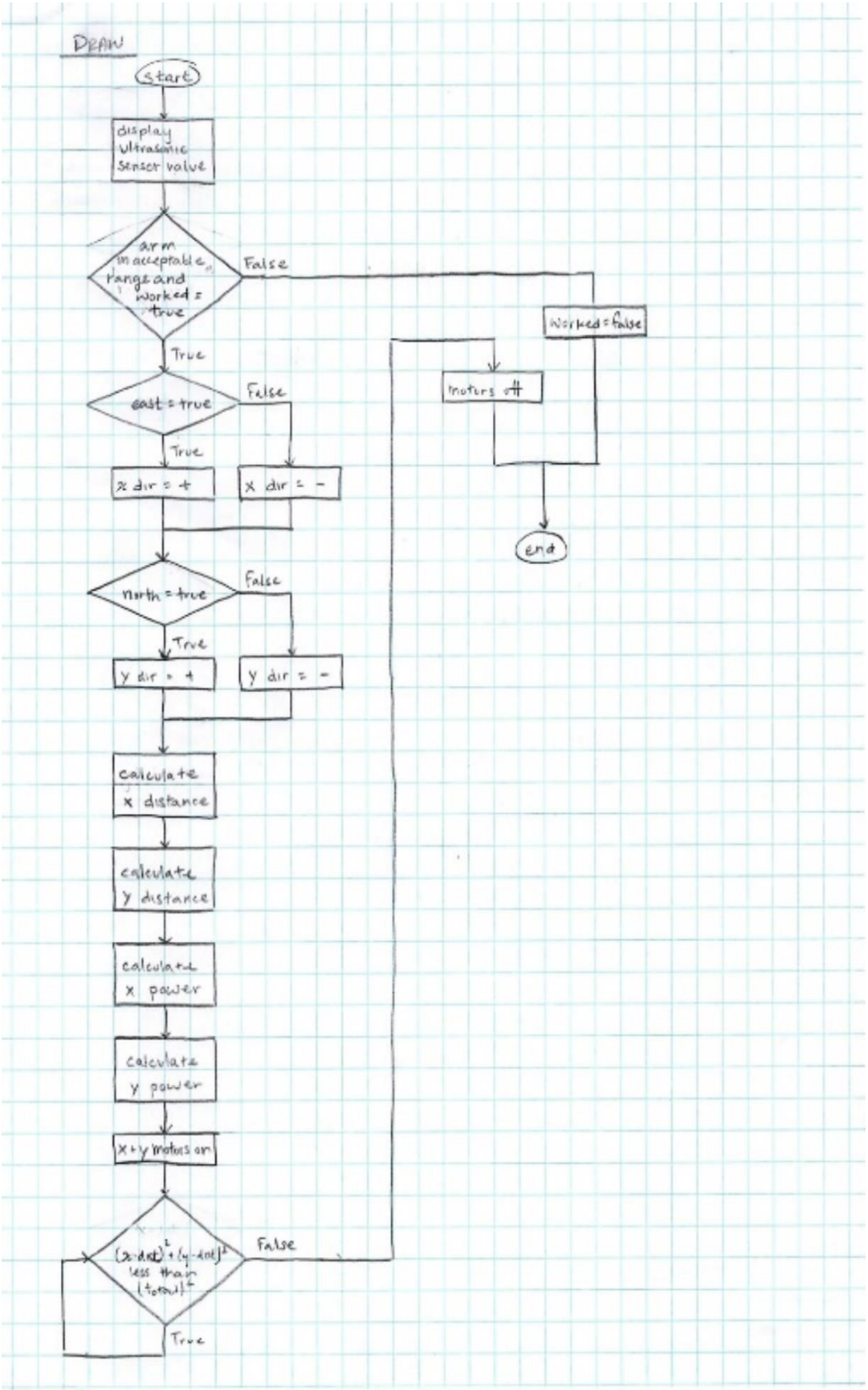


Figure 11.1.3 Draw Function Flowchar

11.2 Appendix 2: RobotC Code for InkBot

```
/*NOTE TO READER OR PROGRAMMER
motorA is the X direction (the longer rack)
motorB is the Y direction (the shorter rack)
motorC ink motor*/

const int INKPOWER = 2;
const int BASEPOWER = 20;
const int ARMTHRESHOLD = 17;
const int ARMTHRESHOLD_CLOSE = 5;

void configureSensors()
{
           SensorType[S1] = sensorEV3_Ultrasonic;
           wait1Msec(50);
           SensorType[S2] = sensorEV3_Touch;
           wait1Msec(50);
           SensorType[S3] = sensorEV3_Touch;
           wait1Msec(50);
           SensorType[S4] = sensorEV3_Color;
           wait1Msec(50);
           SensorMode[S4] = modeEV3Color_Color;
           wait1Msec(50);
}

int identification()
{
     while (!getButtonPress (buttonEnter))
     {}
     while (getButtonPress (buttonEnter))
     {}

     if (SensorValue[S4] == (int) colorRed)
          return 1;
     else if (SensorValue[S4] == (int) colorBlue)
          return 2;
     else if (SensorValue[S4] == (int) colorYellow)
          return 3;
     else
          return 0;
}


float degreesToMM(int degrees)
```

```
{
	return ((float)degrees/15);
}

float rad(float deg)
{
	return deg*PI/180.0;
}

// Draws a line segment of a specified length in a cardinal direction
void draw(bool east, int angle, bool north, float dist, bool & worked)
{
	displayBigTextLine(4, "%f", SensorValue[S1]);
	if(SensorValue[S1] < ARMTHRESHOLD &&
	   SensorValue[S1] > ARMTHRESHOLD_CLOSE && worked)
	{
		nMotorEncoder[motorA]=0;
		nMotorEncoder[motorB]=0;
		nMotorEncoder[motorC]=0;
		int dirX, dirY;
		float Xdist, Ydist, powerX, powerY;

		if(east)
			dirX = 1;
		else
			dirX = -1;
		if(north)
			dirY = -1;
		else
			dirY = 1;

		Xdist = dirX * dist * cos(rad(angle));
		Ydist = dirY * dist * sin(rad(angle));

		powerX = dirX * BASEPOWER*cos(rad(angle));
		powerY = dirY * BASEPOWER*sin(rad(angle));

		motor[motorA] = powerX;
		motor[motorB] = powerY;

		while((pow((degreesToMM(nMotorEncoder[motorA])),2) +
		       pow((degreesToMM(nMotorEncoder[motorB])),2)) <
		       dist*dist)
		{}
```

```
            motor[motorA] = motor[motorB] = 0;
            worked = true;
      }
      else
            worked = false;
}

void inkOn (bool inkMotor)
{
      if (inkMotor == true && SensorValue[S1] < ARMTHRESHOLD &&
          SensorValue[S1] > ARMTHRESHOLD_CLOSE)
      {
            motor[motorC]= INKPOWER;
            wait1Msec(500);
      }
      else
      {
            motor[motorC]= -5*INKPOWER;
            wait1Msec(500);
            motor[motorC]= 0;
      }

     wait1Msec(50);
}

void inkEnd()
{
      motor[motorC] = -25;
      wait1Msec(500);
      motor[motorC] = 0;
}

//This code writes UW
void drawU()
{
      motor[motorA]=motor[motorB]=motor[motorC]=0;
      bool worked = true;
      draw(true, 90, false, 40, worked);
      draw(true, 0, true, 30, worked);
      draw(true, 90, true, 7.5, worked);
      inkOn(true);
      wait1Msec(6500);

      draw(false, 90, false, 7.5, worked);
      draw(false, 45, false, 3.5, worked);
```

```
	draw(false, 0, true, 5, worked);
	draw(false, 45, true, 3.5, worked);
	draw(false, 90, true, 7.5, worked);
	draw(false, 0, true, 2.5, worked);
}

void drawW()
{
	bool worked = true;
	draw(false, 76, false, 10, worked);
	draw(false, 63, true, 5.5, worked);
	inkEnd();
	draw(false, 63, false, 5.5, worked);
	draw(false, 76, true, 10, worked);
}

void drawCube()
{
	bool worked = true;
	draw(true, 90, false, 40, worked);
	draw(true, 0, true, 30, worked);
	inkOn (true);
	wait1Msec(7000);

	draw(false, 0 , true, 10, worked);
	draw(false, 90 , true , 10, worked);
	draw(true, 0 , false, 10, worked);
	draw(true, 90 , false, 10, worked);

	draw(true, 45, true, 5, worked);
	draw(true, 90, true, 10, worked);
	draw(false, 45, false, 5, worked);
	inkEnd();
	draw(true, 45, true, 5, worked);

	draw(false, 0, false, 10, worked);
	draw(false, 45, false, 5, worked);

	motor[motorA] = motor[motorB] = motor[motorC] = 0;
}




void drawArrow()
```

```
{
	bool worked = true;
	draw(true, 90, false, 20, worked);

	inkOn(true);
	wait1Msec(8000);
	draw(true, 0, true, 40, worked);
	draw(false, 25, false, 6, worked);
	inkEnd();
	draw(true, 25, true, 6, worked);
	draw(false, 25, true, 6, worked);

	motor[motorA] = motor[motorB] = motor[motorC] = 0;
}

void drawSquare()
{
	bool worked = true;
	draw(true, 90, false, 40, worked);
	draw(true, 0, true, 30, worked);
	inkOn (true);
	wait1Msec(6500);

	draw(false, 0 , true, 10, worked);
	draw(false, 90 , true , 10, worked);
	inkEnd();
	draw(true, 0 , false, 10, worked);
	draw(true, 90 , false, 10, worked);
}

void zeroBot()
{
	// reset bot to 0 in x
	motor[motorA] = -10;
	while (SensorValue[S2] == 0)
	{}
	motor[motorA] = 0;

	// reset bot to 0 in y
	motor[motorB] = -10;
	while (SensorValue[S3] == 0)
	{}
	motor[motorB] = 0;
}
```

```
void shutdownProcedure()
{
	inkOn(false);
	zeroBot();
	displayString(6,"Shutdown");
	wait1Msec(5000);
}

task main ()
{
	configureSensors();
	zeroBot();
	int userInt = 0;
	string userStr = "";

	displayString (1, "Place your colour ID card");
	displayString (2, "and press ENTER button.");
	userInt = identification();

	if (userInt == 1)
		userStr = "Professor";
	else if (userInt == 2)
		userStr = "TA";
	else if(userInt == 3)
		userStr = "student";
	else
		userStr = "";

	displayString (3, "Welcome %s", userStr);
	wait1Msec(3000);

	eraseDisplay();

	displayString(1, "PRESS LEFT FOR CUBE");
	displayString(2, "PRESS RIGHT FOR UW");
	displayString(3, "PRESS UP FOR ARROW");
	displayString(4, "PRESS DOWN FOR SQUARE");

	bool waitForPress = true;

	time1[T1] = 0;
	while (waitForPress && time1[T1] < 25000)
	{
		if (getButtonPress(buttonLeft))
		{
```

```
            eraseDisplay();
            while (getButtonPress(buttonLeft))
            {}
            drawCube();
            waitForPress = false;

        }
        else if (getButtonPress(buttonRight))
        {
            eraseDisplay();
            while (getButtonPress(buttonRight))
            {}
            drawU();
            drawW();
            waitForPress = false;
        }

        else if (getButtonPress(buttonDown))
        {
            eraseDisplay();
            while (getButtonPress(buttonDown))
            {}
            drawSquare();
            waitForPress = false;

        }

        else if (getButtonPress(buttonUp))
        {
            eraseDisplay();
            while (getButtonPress(buttonUp))
            {}
            drawArrow();
            waitForPress = false;
        }
        shutdownProcedure();
}
```