The Team – with Warmongler

From left to right:

Alan Daniels - Circuits

Wen Bo Li - Electromechanical

Kevin Eisa - Programming

Acknowledgements

We would like to formally thank:

Professor M. R. Emami
Tomas Mawyin
AER201 Teaching Assistants

For giving us great instruction and insightful comments;

as well as,

Creatron Inc.
Home Hardware
Canadian Tire
Active Surplus
Digikey
AER201 Design Store

For providing all that we needed to construct Warmongler;

and finally,

Our peers and everyone else whose presence, time, or actions helped shape this project and made this an unforgettable experience.

Abstract

Quality Control is an engineering pursuit which has great importance in industry. Developing autonomous machines that are capable of identifying malfunctioning products is common practice. This report presents a solution to the Request for Proposal of an LED flashlight testing machine. The requested design determines the functionality of a tray of up to nine flashlights in under 90 seconds without human interaction, aside from initiating the operation. The motivation for such a machine was in improving human quality control working conditions. Various design criteria were examined, which led to a final design in which a sliding bar mechanism was used to turn on flashlights held in a linear tray, while an array of phototransistors, buttons, and multiplexers described the functionality of the flashlights to a microcontroller. Specific design and construction was divided into three subsystems: electromechanical, circuitry, and programming. Project scheduling was done using Gantt charts and critical path analysis. Furthermore, this report describes the integration process, discusses the final functionality of the machine, and presents a Standard Operation Procedure. Lastly, possible design improvements were investigated.

Table of Contents

0. Symbols and Abbreviations

0.1 General

LED - Light emitting diode
LCD - Liquid-crystal display

0.2 Electromechanical

N - Newton
kgf - Kilogram-force
DC - Direct Current

0.3 Circuitry

V - Voltage
I - Current
R - Resistance
P - Power
V - Volts
A - Amperes
Ω - Ohms

1. Introduction

This report describes the design and realization process of an autonomous flashlight LED examining machine. The report outlines the motivations and background for designing such a machine, the objectives of the design, the division of the development process, integration, functionality, and the final outcome of the project.

The motivation for developing such a machine is for quality control purposes; a manufacturer must be able to examine the functionality of multiple flashlights in a fast and accurate manner. Manufacturing processes have a rate of producing successful products that is less than 100%. This is due to assembly environment complications that are not controllable for a reasonable budget. Assigning an employee to manually test for quality assurance is inefficient, expensive, and may cause disinterest for the employee, along with introducing the danger of repetitive strain injury [1].

The machine must accept a pre-loaded tray of flashlights, which is free of electrical components, turn the flashlights on, output the data to an LCD screen, and turn the flashlights off. Operator input starts the operation of the machine, and the operator must be able to use the LCD to access information about the number of flashlights on a tray and the functionality of each flashlight.

2. Perspective

2.1. Theory and History

The development of the assembly line [2] ultimately led to automated production. Automated production performs the same task over and over, however, small imperfections can arise and cause defects in some products from the production line. The imperfections can occur from physical imprecision, varying environmental conditions, and many other factors. Hence, it is not always feasible to have a production line that is free of error. As a result, quality control has grown as a large engineering endeavour. Examining products and discarding defective products is typically a more desirable solution for many production companies. In industry, quality control is performed on a large scale, with many products being examined quickly. Tool flashlights are an example of a device that is autonomously produced, and is susceptible to random devices having partial or complete failure. Additionally, devices such tool flashlights can break during transportation or another form of motion. Quality control can be performed by human operators, however this configuration is costly and introduces the risk of repetitive strain industry. Hence, it is desirable, in many circumstances, to develop a reasonable autonomous quality control machine. The *Warmongler* is an example of such a machine.

2.2. Survey

2.2.1. Photosensing

Determining the functionality of an LED is done using a photosensor. Many types of photosensors are available on the market, such as: photoresistors, phototransistors, and photodiodes. Other types exist, but are not regularly used in general applications because of high specificity or cost. An example of this type of photosensor is an avalanche photodiode. [3]. The resistance of a photoresistor decreases as the photoresistor is exposes to more visible light. Photoresistors are inexpensive and are simple to implement, however they change in resistance is nonlinear, and photoresistors typically have a large surface area. Photodiodes generate current that is proportional to the amount of photons that hit it. Photodiodes are more expensive and slightly more complex to implement than photoresistors. Photodiodes are extremely fast, however, since all of the photosensing is done in parallel in our machine, a microsecond or even millisecond speed increase is not significant. Phototransistors operate like regular transistors, except the base is either entirely a photodiode, or the base is supplemented by a photodiode. Phototransistors have a large gain, and can sense small increments of light intensity, but they are slower than photodiodes. Phototransistors were chosen of their seemingly close relationship with logical highs and logical lows, as opposed to a photoresistor which is more of a continuous spectrum. Furthermore, phototransistors were a compromise in cost and speed between photoresistors and photodiodes. For analyzing multiple inputs, using multiplexing is an intuitive and common solution in industry. A multiplexer takes signals to select amongst various inputs. This is a logical method of examining 3 LEDs.

In the initial design phase, the large surface area of a photoresistor was seen as a disadvantage, as it crowded circuit board space, and it made sensing all three LEDs difficult. However, in the final design, the photosensors were not directly soldered onto the circuit boards, and thus the large surface area of photoresistors would not have been a problem, it might even have been beneficial. Additionally, the nonlinear behaviour of the photoresistor was seen as a con, however, the LEDs were Bernoulli tested; they could only be on or off, their intensity was not measured. In such an application, it is possible to use a nonlinear sensor with an appropriate circuit without a problem. Thus, it appears that photoresistors would have been a good choice, and could have significantly reduced the cost of the machine. Photoresistors could have been made compatible with multiplexing also, but more testing is required to determine how that circuitry might work.

2.2.2. Turning Tool Flashlights ON and OFF

The flashlights this machine deals with are turned on and off by rotating a plastic ring on the circumference of the flashlight by approximately 20°. In industry, it is most common to apply a continuous force tangential to a part that needs to be rotated. Hence, motors were used instead of solenoids. Additionally, motors are cheaper and can often provide more force than a solenoid.

2.2.3. Actuator Driving Circuits

To drive a motor bidirectionally, it must be supplied with opposite signals. This can be done by selecting a signal to send to the motor, using a device such as a transistor or a relay. Alternatively, signals can be alternated in another method with circuitry. An example is a push and pull transistor circuit, as shown in Figure 9.1. Another commonly used circuit is the H-bridge. It is possible to manually construct an H-bridge, or use an integrated circuit such as an L298.

2.2.4. Tool Flashlight LED Sensing Machine

A machine that performs the same functions that this machine is required to do, subject to the same constraints, does not exist in industry. Although the objective of the machine is practical, the scale of the machine does not coincide with common practice in quality control. Typically, quality control machine are far larger and more autonomous, and thus different flashlight switching mechanisms are required.

3. Objectives

3.1. Project Goal

The goal of this project was to develop a machine capable of determining the functionality of a tray of up to nine tool LED flashlights. The flashlight tray was required to have no electronic parts or actuators, and the flashlights were not to be affixed to the tray though a permanent attachment. Additionally, the tray was required to be completely separable from the machine. Operation had to be controllable with a keypad. When the tray was placed inside the machine and the operation was initiated, all flashlight present on the tray had to be turned on, examined, and turned off. The tray was then made available for removal, and data about the functionality of the flashlights was stored and displayable on an LCD screen upon user command. Each flashlight could have none of its three LEDs working, two of its three LEDs working, one LED working, or no LEDs working. Furthermore, the machine was required to determine the number of flashlights that were placed onto the tray and examined.

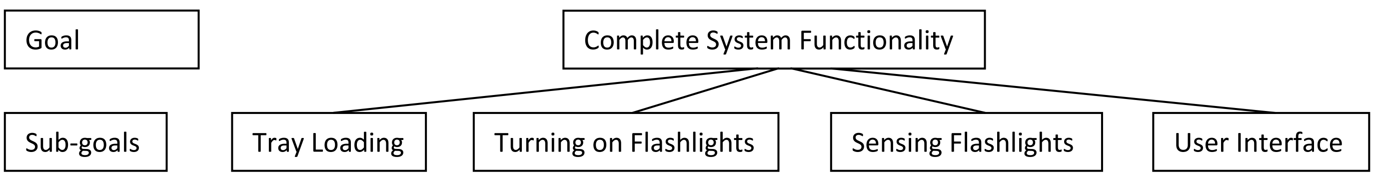


Figure 1 General Goals

3.2. Constraints

<u>Tray</u>

- Completely separable from machine
- No fixed attachments
- Quick and convenient flashlight loading process
- No electronics, actuators, or sensors

<u>Machine Physical Structure</u>

- Cannot exceed 50 x 50 x 50 cm^3
- Cannot exceed 6 kg excluding flashlights, including the power cable
- Tray must be convenient to load and unload, within a minute

Machine Functionality

- Fully autonomous during operation
- Must have the capability to display:
    - Operation time
    - Number of flashlights
    - Functionality of each flashlight
- Cannot damage flashlights
- Operation must occur in under 90 seconds
- The machine must not be hazardous in any way

The above constraints were referred to from the Request for Proposal [4]

3.3. Design Criteria

Functionality

- Minimal number of incorrectly examined flashlights
- Little susceptibility to failure
- Speed of operation
- Ease and speed of loading flashlights onto tray
- Ease and speed of loading tray
- Intuitive and convenient LCD interface

Manufacturing

- Less cost
- Ease of repairing and replacing failed components
- Minimal unnecessary complication

3.4. AHP Analysis for Robot Tray Decision

The tray of the robot was one of the most important decisions in the design process. This decision affected many components of the robot, such as the actuation mechanisms used, the design of the various physical components of the robot, and thus many decisions in the circuitry and programming portions of the design as well. Due to the importance of this solution, the analytical hierarchy process (AHP) was a decision making analysis appropriate in making this selection. There were three candidates for the tray: circular, 3x3, and 1x9. The circular tray was a design in which the flashlights are situated around the perimeter of the circular tray, in which a rotational actuation mechanism would rotate the tray in order to bring the flashlights to the sensing portion of the robot. The 3x3 design was a design in which the flashlights would be positioned in a 3x3 rectangular position, and nine actuation mechanisms would go to each flashlight to operate on them. Lastly, the 1x9 tray was a design in which the nine flashlights would be situated in a straight line, and then actuation mechanisms can operate on them when they are in such a position.

There were four criteria or objectives which were considered in the decision. First, the compactness of the robot was important in giving extra score to the robot; second, the speed of the operation was important in both meeting the operation time constraint and giving extra score to the robot; third, the complexity design was important in increasing the reliability of the robot and increasing its chances of success; and fourth, the cost of the robot was important in meeting the cost constraint on the project. These were the objectives were used in the AHP analysis.

Table 3.1 - Solution Legend

Solution Label	A	B	C
Solution	Circular	3x3	1x9

Table 3.2 - Objective Legend

Objective Label	1	2	3	4
Objective	Compactness	Speed of Operation	Design Complexity	Cost

Table 3.3 - Relative Preferences (RP) of Solutions with Respect to Objectives and Overall Solution Weights

Objective 1 RP				Normalized Objective 1 RP				
	A	B	C		A	B	C	Overall
A	1.00	0.83	5.00	A	0.42	0.40	0.50	0.44
B	1.20	1.00	4.00	B	0.50	0.48	0.40	0.46
C	0.20	0.25	1.00	C	0.08	0.12	0.10	0.10
Objective 2 RP				Normalized Objective 2 RP				
	A	B	C		A	B	C	Overall
A	1.00	1.25	0.40	A	0.23	0.22	0.24	0.23
B	0.80	1.00	0.30	B	0.19	0.18	0.18	0.18
C	2.50	3.33	1.00	C	0.58	0.60	0.59	0.59
Objective 3 RP				Normalized Objective 3 RP				
	A	B	C		A	B	C	Overall
A	1.00	3.00	0.33	A	0.23	0.30	0.22	0.25
B	0.33	1.00	0.17	B	0.08	0.10	0.11	0.10
C	3.00	6.00	1.00	C	0.69	0.60	0.67	0.65
Objective 4 RP				Normalized Objective 4 RP				
	A	B	C		A	B	C	Overall
A	1.00	5.00	3.00	A	0.65	0.80	0.35	0.60
B	0.20	1.00	4.50	B	0.13	0.16	0.53	0.27
C	0.33	0.22	1.00	C	0.22	0.04	0.12	0.12

Table 3.4 - Relative Importance (RI) of Objectives and Overall Objective Weights

Objective RI					
	1	2	3	4	
1	1.00	1.25	0.67	2.00	
2	0.80	1.00	0.80	1.70	
3	1.50	1.25	1.00	2.50	
4	0.50	0.59	0.40	1.00	
Normalized Objective RI					
	1	2	3	4	Overall
1	0.26	0.31	0.23	0.28	0.27
2	0.21	0.24	0.28	0.24	0.24
3	0.39	0.31	0.35	0.35	0.35
4	0.13	0.14	0.14	0.14	0.14

Table 3.5 - Scores of Solutions

Solution	Score
A	0.345467
B	0.239305
C	0.415228

The final weighted average shows that solution C is the most preferred solution given the objectives considered, which is the 1x9 rectangular tray solution. This was the solution selected for the final design.

4. Budget

The maximum budget for the prototype parts was $230. Labor, testing, and unused parts costs are exempt. The cost of the prototype was $229.67, as shown in Table 4.1.

Table 4.1 – Total Cost of Robot

Subsystem	**Cost**
Electromechanical	$82.98
Circuits	$71.66
Programming	$75.03
Total	**$229.67**

Please refer to sections 6.36, 7.2, and 8.2 for a detailed breakdown of cost for each subsystem.

5. Problem Division

The machine conceptualization and realization was divided into three subsystems: electromechanical, circuits, and programming. Wen Bo Li was responsible for the electromechanical subsystem, Alan Daniels for circuits, and Kevin Eisa for programming. For the first two weeks of the project, design of critical and general functionality was performed as a team. For the next six weeks, each team member designed and developed their respective subsystem. For the following four weeks, the team integrated the subsystems. Finally, in the week remaining, the team debugged the system.

5.1. Design

The goal of the design stage was to develop a functional decomposition for the problem and conceptualize a solution. This stage evaluated different methods of turning the flashlights on and off, investigated multiple tray structures, and examined other aspects of design. The end of the design stage outlined the required tasks for each subsystem; what needed to be built and how it should work with the other subsystems.

5.2. Electromechanical Tasks

The electromechanical subset of the problem consisted of the physical components of the robot, which can be broken down into static, passive dynamic, and active dynamic components that cooperate together to achieve the physical functions of the robot such as turning on and off the flashlights. The active dynamic components are those which are powered by actuators, which utilize electrical power to deliver the force needed to accomplish a task. The electromechanical task had to take into consideration the circuital and programming constraints such as available power and placement of the microcontroller board, and thus in this component required coordination between the three different tasks.

5.3. Circuitry Tasks

The objective of the circuitry was to deliver power to all components, control actuators, and sense flashlights. Powering components involved wiring design, connecting wires, and organizing components and wires in the machine. Controlling actuators consisted of designing and constructing circuits which would cause actuators, motors and solenoids in this machine, to perform in a controlled manner through interacting with the microcontroller. Sensing flashlights required using photosensing components and buttons which could describe the functionality and presence of a flashlight. Furthermore, circuitry was required to process the behavioural change in these components into signals which the PIC can process.

5.4. Programming Tasks

The objective of microcontroller was to control the actuation and sensing performed by the robot and to provide the user the means to interact with the robot. The actuation involved assigning pins for the inputs and outputs, deciding the length and order for the different steps in actuation, and saving the results of the experiment. The user interface involved giving users the ability to control the machine and see the results of the test.

6. Electromechanical Subsystem

6.1. Functional Decomposition

The physical functions were decomposed into the following components:

1. The securing of the flashlight tray and the placing of the flashlights inside it.
2. The securing of the body of the flashlights during the operation in order for the turning of the flashlights' rings to take place without turning the body itself.
3. Turning on and off the flashlights by turning the rings of the flashlights clockwise and counter-clockwise.
4. The lining of the photosensors with the lights of the flashlights.
5. The detection of the presence of flashlights.

6.2. Physical Decomposition of Components and their Assessment

The physical components of the robot were roughly divided into the following components: the tray, the tray holder, the frame of the robot, the robot stand, and the body of the robot, which contained the detailed components that will accomplish the functions required for the robot.

6.2.1. General

In fulfilling the functions required by the robot and attempting for the maximum score on the test within the constraints of the system, there were a few major considerations to be balanced in order to achieve the best overall result. One, the compactness of the robot, which limited the dimensions and weight of the robot, would have drastically increased the scoring of the robot; two, if all the mechanisms could have been accomplished in a short amount of time, this would also have increased the scoring of the robot; three, the complexity of the robot, which included the number of moving parts and the difficulty in construction, factored into the quality of the robot since this project was under a small time constraint; four, the extra number of flashlights the robot could have managed would also have significantly increased the score of the robot. Five, the cost of each of the components were best to be kept as low as possible in order to meet the general cost constraint.

Table 6.1 - Five Major Electromechanical Design Considerations

Design Consideration	Reason for Significance
Compactness	Extra scoring chances
Speed of Operation	Extra scoring chances
Complexity in Design	Improvement in feasibility
Extra Flashlight Handling	Extra scoring chances
Cost	Meeting constraint

6.2.2 Tray

The tray's function was to allow the operator to easily place the flashlights into the robot and secure the positions of the flashlights to an extent such that the robot may have operated on the flashlights with precision and reliability. In order to design for a robot which received the greatest amount of points, the best design for this particular component was to allow for the robot to stay within the 30cmx30cmx30cm dimension requirement for the compactness bonus.

6.2.3. Tray Holder

The tray holder was complementary to the tray, and functioned to secure its position within the robot in order for the robot to operate onto the flashlights. For a static tray, the holder had to be able to secure and maintain the position of the tray inside the robot while allowing for the easy insertion and removal of the tray. Given these purposes were met, the tray holder was best built as light as possible to help meet both the compactness bonus criteria and also the general weight constraint for the entire robot.

6.2.4 Robot Frame and Stand

The robot frame and stand were to provide the structure and rigidity of the robot. The function of the frame not only provided both the basis upon which other components can be constructed, but also helped to shield the photosensors from outside light, which could impact the detection of light from the flashlights themselves. The stand, which was to be situated on the outside of the robot, was to provide the base upon which the programming board and circuits were to be placed, and so provide the place for the human-machine interface. The best design for the frame and stand was one which would meet the dimension requirement of the compactness bonus, provide a good rigidity to the robot, and was also cheap to build, which would have helped to meet the cost constraint.

6.2.5. Mechanism for Turning On and Off the Flashlights

The mechanism for turning on and off the flashlights was central to the solution of the problem, and was a decision upon which almost every other component was dependent.

This mechanism was to target the rings of the flashlights when the flashlight in the trays was inserted into the robot, and would rotate the rings without turning rest of the flashlight in order to turn them on or off. The mechanism would invoke all major considerations, and needed to be carefully considered in the design process.

6.2.6. Mechanism for Holding the Flashlights in Place for Turning On and Off the Flashlights

The mechanism for holding the flashlights in place functioned to prevent the flashlights' bodies from turning when the flashlights' rings are being targeted by the on/off mechanism. The more powerful this mechanism, the more reliable the on/off mechanism would have been in its function.

6.2.7 Mechanism for Detecting the Presence of Flashlights

The mechanism for the detection of flashlights functioned as the counter of the number of flashlights in the machine. Given that this served its purpose, the cheapest and least complex mechanism would have been best design for this particular component. This is a decision which depended on the on/off mechanism.

6.2.8 Mechanism for the Lining of Flashlights with Photosensors

The mechanism for lining the flashlights with the photosensors functioned to allow for the detection of the presence of light from each light bulb of the flashlights. Given that its purpose was accomplished, the cheapest and least complex mechanism would have been the best design for this particular component. This is also a decision which depended on the on/off mechanism.

6.2.9 Mechanism for the Detection of Flashlights

This mechanism would have functioned to detect the presence of individual flashlights. This mechanism could have been completely static, as a distance sensor was sufficient in detecting for the presence or absence of flashlights; this would have contributed to making the robot simpler for design. However, an active dynamic mechanism powered by an actuator was also possible, and would reduce the complexity in the circuit and programming subsystems, and possibly the cost as well.

6.3. Selected Solutions and their Progression into the Finalized Prototype Design

Table 6.2 - Robot Components and their Constituted Physical Decomposition Components and Functions

Robot Component	Constituted Physical Components	Functions Achieved
Slider and Slider Restrainer	Mechanism for Turning on/off Flashlights	Turned on/off the flashlights
Tray	Tray	Provided a place for the insertion, securing, and retrieval of flashlights
Tray Holding System	Tray Holder	Secured the tray and the flashlights during the operation
Sensor Arm	Mechanism for Holding the Flashlights in Place for Turning on/off Flashlights; Detecting Presence of Flashlights; and the Lining of Flashlights with Photosensors	Helped in turning on/off the flashlights, lined the photosensors with the flashlights, secured the flashlights during the operation, and the detection of flashlights
Robot Frame and Stand	Robot frame and Stand	Prevented outside light from interference of photosensing; provided a basis upon other components may be constructed

6.3.1 Slider and Slider Restrainer

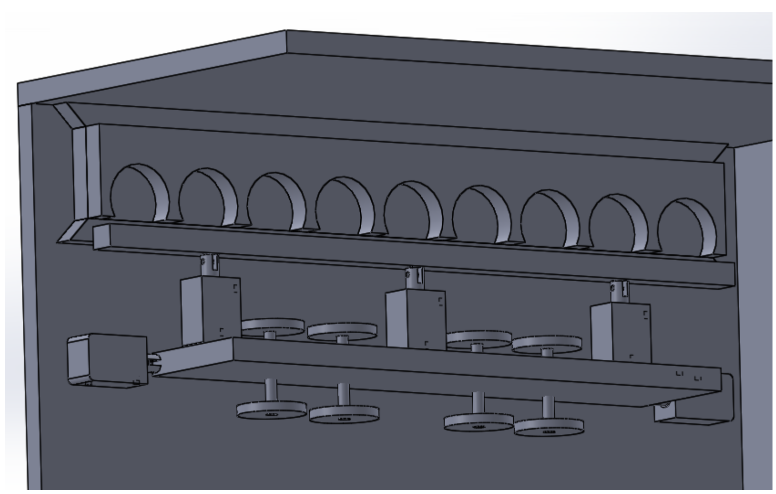


Figure 2 Original Belt Design

The slider was the selected solution for the mechanism to turn on or off the flashlights, and as the solution for the central problem of the robot, was the component upon which the solution to the rest of the robot depended. The original concept of this selected solution was a wooden body on wheels which would hold a certain number of solenoids, and which would all attach to a long rectangular piece of material with a high friction material on the other end, called the belt (figure 2). The slider itself was also attached to solenoids on either side of it. During operation, the solenoids attached to the belt would push the belt out on the flashlight rings, asserting a force on the rings in an attempt to grasp firmly onto them. Subsequently, the solenoids on either side of the slider body, which are attached to wheels, would produce a sideways movement, and push the body in a certain direction; since the body is attached to the solenoids holding the belt that are gripping onto the flashlights, the movement of the belt would rotate the flashlight rings and turn on the flashlights. The length of the belt would be long enough that it would touch all of the flashlights when utilized, meaning that it would turn all the flashlights on at once. With respect to the major considerations of the project, this solution had advantages and disadvantages. This solution would not be able to meet the compactness bonus of the project, since the belt would extend beyond the 30cm dimension limit; however, it would allow the robot to quickly turn on and off all the flashlights, which would help reduce the amount of time needed for each operation. Furthermore, the static tray reduced the complexity of the robot, and allowed the combining of the mechanisms which would detect the presence of the flashlights and the lining of the flashlights with the photosensors, which reduced another level of complexity. The handling of nine extra flashlights was not implemented due to cost and complexity concerns. For these reasons, this concept was adopted as the mechanism for turning on/off the flashlights.

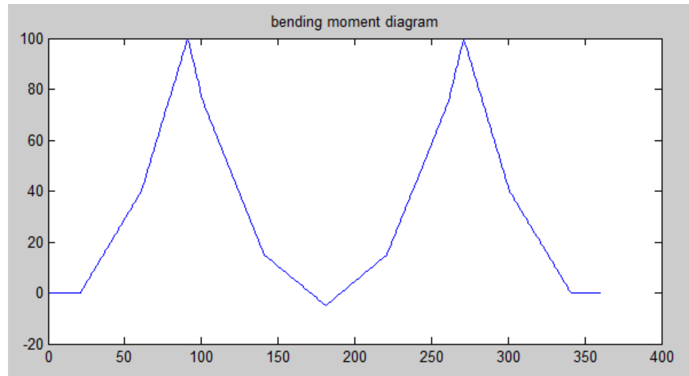
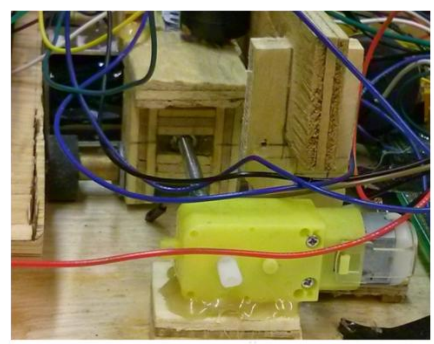


Figure 3 Bending Moment Diagram of Belt under Flashlight Loads

Originally, the slider powered by solenoids consisted of a rectangular piece of wood which would act as the body, and another long rectangular piece of wood with a rubber end attached would act as the belt. The belt held two solenoids, which are positioned in such a way that the belt bent the least upon receiving the point forces from each of the flashlights; this was calculated from a bending moment diagram (figure 3 – refer to code in appendix E). However, the design underwent major changes to best transfer the most amount of force onto the flashlights rings. The first major change was the change of the actuators which would power the slider in its sideways movement. Solenoids could have achieved that task of pushing the slider body; however, since the movement of solenoids are abrupt, it could have induced a movement which would not have fully taken advantage of the static friction of the belt on the flashlight rings. Due to this reason, and the fact that there was limited power for the solenoids, which would impact the amount of force put onto the



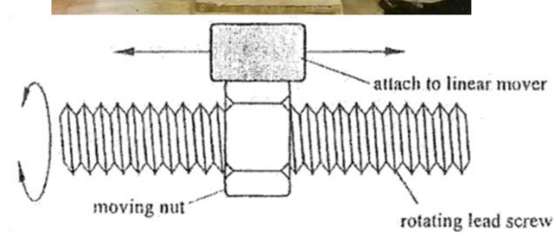


Figure 4 Leadscrew Mechanism for the Slider

flashlight rings critical for the gripping of the belt onto the rings, the design was changed to a leadscrew attached to the slider powered by a DC motor (figure 4). Calculation for the force of a leadscrew was complex and required parameters such as the coefficient of friction which were difficult to estimate (appendix A); however, testing showed that it did possess enough force to move the slider and turn on the flashlights. Furthermore, the body of the slider was changed to a box design in which the axles of the wheels were each put through both sides of the box, ensuring the balance of the slider when the heavy solenoids with the belts were attached (figure 5). It also helped to hold the solenoids more firmly by providing a surface above the solenoids onto which clamps holding the solenoids could firmly grasp. Lastly, the material on the belt underwent a dramatic change. When this system was tested, the material proved to not have enough grip onto the flashlights, and the belt would slide past the flashlight rings without engaging them. Through changing the material to sandpaper, timing belt, and then skateboard friction tape, the final material became the central portion of screws, whose threads fitted well into the flashlight grooves (figure 6). They fitted well into the grooves of the flashlights, and gripped onto them better than all previous materials. This became the final form of the slider. The slider also had an accompanied slider restrainer which provided a balancing moment to the slider when the solenoids pushed onto the flashlight rings, and prevented the movement of the slider during its operation. As shown in figure 5, a pair of wheels attached to the slider rolled on the restrainer to maintain the slider's position while minimizing friction between the two objects. The slider restrainer was constructed to be moveable so that the solenoids could have been adjusted in position to have the smallest throw. This ensured that the solenoids would have had the greatest amount of force when they were activated.

Figure 5 Solenoid Attachment and Slider Restrainer Wheel

Figure 6 Belt Mechanism

6.3.2. Tray

The tray was a rectangular piece of wood which had nine wells to hold the flashlights. The wells were situated in a straight line along the entire length of the wood, and each of the wells was a circular piece which was cut so that part of each circle was outside of the tray, exposing a small part of each flashlight. This was the tray which was compatible with the slider design, and allowed the flashlights to be aligned so that the belt could be touching all at once. Furthermore, the wells

Figure 7 Exit of Robot

secured the flashlights in positions which are precise enough for the photosensors, and also allowed for the easy insertion and retrieval of the flashlights into and out of the tray.

6.3.3. Tray Holding System

In the original concept, the tray holding system consisted of two holes in the robot frame, a pair of restrainers that limited the position of the tray, and a hinge-string system. One of the holes was designated the entrance of the robot into which the tray was to be inserted, and one of the holes was the exit of the robot out of which the tray was retrieved (figure 7). The pair of restrainers acted to keep the positioning of the tray so that it would guide the tray to its starting position each time it was inserted. Lastly, the hinge-string system would secure the position of the tray inside the two tray restrainers so that it would be in the exact position required by the robot. The mechanism consisted of two hinges that blocked the path of the tray at each hole, a stiff string which would connect the two hinges when they are in a closed position, and a spring behind the hinge at the entrance of the robot that would naturally position the entrance hinge in the closed position. When a tray is inserted, it would open the entrance hinge, but when the tray is pushed to the exit hinge of the robot, the entrance hinge would have closed, causing the stiff string to transmit the spring force to keep the exit hinge in the closed position and allowing for the correct positioning of the tray. When a new tray is inserted, it would open the entrance hinge, releasing the exit hinge and allow the exit of the first tray. In this way, the machine allowed for the easy insertion and retrieval of trays in the style of an assembly line.

Figure 8 Magnetic Holder System

The major change to the tray holding system was the change from a hinge-string system to a magnetic holder (figure 8). The holder consisted of one static magnet and one moveable magnet attached to a hinge that blocked the way of the tray at the exit of the robot. Similar to the hinge-string system, the magnetic holder allowed the operator to position the tray in the same position each time by feeling a stopping force, and would allow a new tray of flashlights to be inserted from the same entrance when the operator applies enough force one the first hinge to open the magnetic door. This achieved the same effect as desired by the hinge-string system, but was easier to construct and was more reliable. This was because no string can be completely stiff, and would always allow a slight extension when a small force is given, while unless enough force was given, two magnets would not separate and open the hinge.

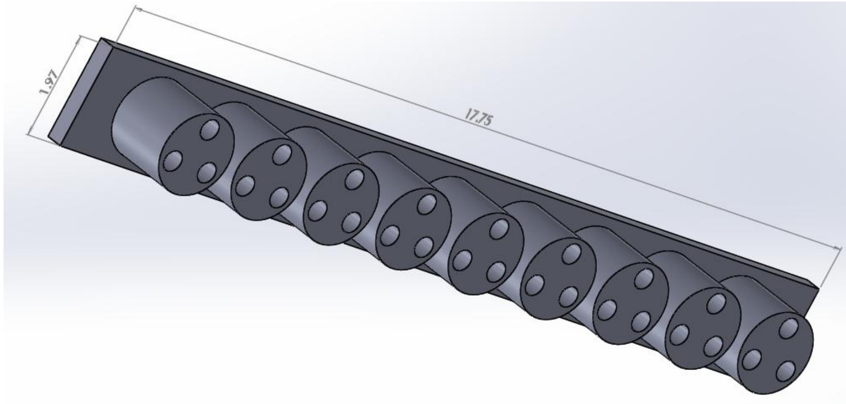


Figure 9 Original Sensor Arm Design

6.3.4. Sensor Arm


Figure 10 Sensor Holders on the Sensor Arm

The sensor arm was the selected solution for the mechanism to line the photosensors with the flashlights' light bulbs, the mechanism to detect the presence of flashlights, and the mechanism to hold the flashlights in place when they were to be turned on or off. The concept of this solution was a board consisting of a flat rectangular piece of material which would hold both the photosensors and pushbutton keys, which were the sensors selected to detect for the flashlights' presence. The board would be attached to nine sensor holders (figure 9), which each had three holes matching the positions of each flashlight's light bulb, and a support in the middle which would hold the pushbutton key. In order to activate the pushbutton keys, a solenoid attached to the board would hang above vertically and push the board down onto the flashlights. If a certain flashlight were present, then the corresponding pushbutton would press onto the flashlight, causing the key to be pressed, and the presence of the flashlight would be indicated. At the same time, the pressure that the pushbutton key exerts onto the flashlight would keep the body of the flashlights on the tray when the rings are being turned, and thus would function to hold down the body of the flashlight in during the on/off processes.

There was one major design change from the original concept, which was to replace the solenoid powering the motion of the sensor arm by a DC motor that would power a leadscrew hanging above vertically that is attached to the sensor arm. This design change was made to both change the abrupt motion of the sensor arm to one that was more controlled, and to reduce the power required by the actuators of the robot. In the abrupt motion, the force to be exerted on the pushbutton keys would be large. The slower leadscrew design would have reduced the chance of a pushbutton key being damaged. The final design is shown in figure 11.

6.3.5. Robot Frame and Stand

The frame and stand of the robot was originally designed to be as light as possible in order to meet the weight constraint, since almost all materials provided enough rigidity in structure to maintain the shape of the robot. This design was to be a rectangular box which wholly surrounded all components of the robot. The stand upon which the microcontroller board were to be situated was the top lid of the box. However, the frame in the final design was changed to use hinges that would make both the stand and two sides of the robot to be doors. This change was made to increase the ease of fixing components had there been a problem. In addition to adding hinges to these walls of the frame, the two side doors were constructed with a magnet system, which consisted of a magnet on each of the moving doors and corresponding magnets on the static walls of the robot, which kept the doors in a closed position unless a large force was applied to open them (figure 12).

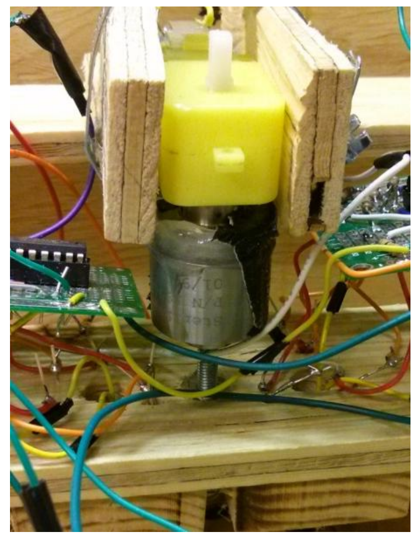


Figure 11 Sensor Arm Leadscrew Mechanism

Figure 12 Magnetic Doors

6.3.6 Budget and Materials Used

Table 6.3 Materials Used in Electromechanical Budget

Material	Quantity	Cost per Unit ($)	Total Cost ($)
2'x2' 3/8'' Pine plywood	3	1.84375	5.53
Scrap wood for sensor holders	1	1.00	1.00
Metal corner bracing	2	0.4975	0.99
Hinges with screws	7	1.245	8.715
Shenzhen DC gearhead motor (straight)	2	5.00	10.00
Leadscrew and nut	2	0.995	1.99
Solenoids	2	14.95	29.90
Wood glue	1/5	5.99	1.50
Epoxy	1/4	8.49	1.70
Wood screws	18	0.078625	1.42
Metal clamps	3	1.29	3.87
Wheels and axle	2	1.00	2.00
Slider Restrainer Wheels	2	0.50	1.00
Nuts and bolts	2	0.37375	0.75
Metal washers	4	0.4975	1.99
4.5mm to 6mm Flexible metal coupling	2	4.00	8.00
Magnets	7	0.37375	2.62
Total			82.98

6.4. Suggestions for Improvement of Subsystem

6.4.1 Improvements to the Slider System

Although the slider's design underwent many modifications, it still failed to perform its intended function. The three factors which affected the performance of this mechanism were the force applied, the friction between the belt and the flashlight rings, and the force and controlled movement required to move the slider and turn on/off the flashlights. Even though all three problems were addressed, and the last two problems were solved, the solution for the force issue was not adequate. In the original design, three solenoids were used to apply the force, which from appendix A shows that about 24N of force was applied. Under inadequate testing, however, a value of 2N was thought to provide enough force to open the flashlights. Since only seven flashlights were required to be opened in each run, the 14N of force was clearly given by the solenoids. In the design process, the number of solenoids was reduced to two in order to reduce cost and power consumption, which reduced the force to 16N. However, later testing and research showed that each flashlight actually required 3.7N of force, bringing the required total force to least 25.9N. Since the solenoids were only each able to provide around 8.25N of force each [5], giving a total of only 16.5N, this made solenoids unable to accomplish this task under the power and cost constraints. A possible design which could have solved this problem would be DC motors powering leadscrews that controlled the motion of the belt onto the solenoids. When the screw threads on the belt are attached to the flashlight rings, they would be locked in position. Under this assumption, the very large amount of force to turn a stalled motor in the opposite direction would provide enough force to engage the flashlight rings. Since DC motors required less power than the solenoids and were cheaper to implement, this would have been a feasible idea to solve this problem.

Another improvement in the slider system would be to change the DC motor that powered the movement of the slider to a more powerful one. Although the motor used in the final design was powerful enough to open flashlights, it was slowed down significantly when the nail threads on the belt attached to the flashlight rings. Since the motored was only powered for a certain amount of time, this made the distance by which the slider moved unpredictable, which caused the motor to stall in either direction, and damaged parts of the robot which were more delicate. This slight modification would have helped to keep the speed of the slider constant, and reduced such damage to the robot.

6.4.2. Improvements to the Tray

In the final design of the robot, testing showed that the belt was not able to grip onto all of the flashlight rings due to the fact that the wells in the trays were not in a precise straight line. The tolerance for the position of the wells was very low, since the thread of the nails on the belt was only about one millimetre wide. Other methods of cutting the tray, such as laser cutting, which has tolerances of much less than tenths of millimetres [7], could be used in making the tray to solve this precision issue.

6.4.3. Using Types of Wood other than Plywood

In the final design, nearly the entirety of the robot was constructed using pine plywood, which although was cheap, was not a good choice for a few components of the robot. The biggest problems this caused was for the sensor arm, which in the final design had two problems. First, the flexibility in the cantilever to which the motor was attached caused the motor to bend upwards a significant amount due to the strong stalling torque of the motor. Even though this precision problem still allowed the sensor arm to achieve its purpose, it showed how plywood was not stiff enough, and was easily bent. The second problem in the sensor arm was that the plywood which held the sensor holders was bent upon the attachment of the holders, which prevented the pushbuttons from being in the same vertical positions. For both these problems and others which required more stiffness and rigidity, other types of wood would have been a better choice over plywood.

6.4.4. Using Metal Pieces, Screws, and Nails to Attach Difference Pieces of Wood

For the majority of wood attachment of the robot in the final design, the pieces of wood were attached with wood glue. Even though this was sufficient for the robot frame, it was insufficient for other parts of the robot which required stronger attachments. This problem was most prominent in the slider, in which pieces of wood were joined to form the body of the slider that held the heavy solenoids. During the construction process, the slider body frequently bent at the wood glue attachments due to the stress on the slider body caused by the weight of the solenoids. If a flat metal piece with screws were used to secure the shape of the slider, the weight of the solenoids would have been able to be easily supported. In other parts of the robot, metal pieces and nails would have helped to maintain their structures.

7. Electrical Subsystem

7.1. Problem Assessment and Objectives

There are four primary objectives of the electrical subsystem: slider motion, sensor arm motion, photosensing, and an emergency switch that halts all motion.

7.1.1. Slider

The high-level objective of the slider is to turn the flashlights on before sensing, and turn them off after sensing. To turn the flashlights on, the slider applies a tangential frictional force to the yellow plastic rings of the flashlights, which when turned, turns on the flashlights. A small bar with a material with a large coefficient of friction on the surface facing the flashlights must be pressed against the flashlights. A strong force is required, as friction is directly proportional to this normal force. After this bar has made contact with the flashlights, the entire slider must be driven linearly in one direction. This motion will turn on the flashlights. Then, sensing is performed. Subsequently, the slider must be driven in the reverse direction, turning off the flashlights. Finally, the normal force must only now be disengaged, so that the flashlight tray will become removable once again. Signals from the microcontroller must be able to control when motion begins and when it terminates.

7.1.2. Sensor arm

The sensor arm must be physically actuated to achieve three objectives: First, to hold the flashlights down, rendering them easier to turn on. Second, to cause buttons above flashlights to be pressed, giving knowledge of which flashlights are present and of the total number of flashlights. Third, to bring the photosensors into proper position. All of these objectives are achieved by lowering the sensor arm onto the flashlights until sufficient force is applied on the buttons, and maintaining that position until operation is complete. Once again, microcontroller signals must be able to control this motion.

7.1.3. Photosensing

For each flashlight, the number of working LEDs must be determined. The flashlights are, roughly, positioned into the tray in a certain orientation, and then the sensor arm separates each of the three flashlight LEDs, so that each LED can be sensed by one photosensor. Information about all 27 possible LEDs must be sent to the microcontroller.

7.1.4. Emergency Stop Switch

The machine must have an emergency switch that stops all motion upon being pressed. Emergency shutdown can damage the microcontroller, hence avoiding interaction between the switch and the microcontroller is beneficial. However, the stop switch is a last resort mechanism, and thus preserving microcontroller functionality is not a necessity.

7.2. Electrical Design Solutions

7.2.1. Slider

Extending the bar and maintaining a large normal force will be done using a pair of solenoids. Solenoids are actuators which, when supplied with a sufficient amount of current, extend a small metal bar. The implemented solenoid has a throw of 10 mm. The current through the non-polar solenoid travels through a helically wrapped wire, which generates a magnetic field through the centre of the helix. This magnetic field interacts with the metal bar and is responsible for its motion. When current travels through the solenoid, the magnetic field causes the small metal bar to be pushed out. When no current travels through the solenoid, a spring pulls the metal bar back. Most of the magnetic field is generated within the solenoid, thus solenoids should not cause signal interference problems when placed far away from other wires and electrical components, which they are. The solenoids are connected so that they always actuate at the same time. The metal bars of the solenoids are glued to the slider's friction bar, so that it moves with the solenoid bars. Two solenoids were used because that configuration better distributes the force over the slider bar, and hence over the flashlights. More solenoids would have better force distribution, but would be too costly. The solenoids are powered directly by the 12V line of the power supply, as their internal resistance is sufficient to not cause a short circuit, and adding more resistance along their wiring path would decrease the current travelling through the solenoids, and would thus decrease the

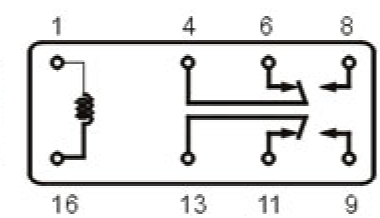


Figure 13 HLS-4078 Schematic

total force exerted on the flashlights. The solenoids are activated by a relay controlled by the microcontroller. All relays in the machine operate in the same manner. The implemented relays are HLS-4078 5V relays. The schematic for this relay model is shown in Figure 13. Connections 13, 11, and 9 are not used. When a 5V potential difference exists on both sides of the relay's solenoid, between connections 1 and 16, the wiring of the relay changes. When the 5V potential is applied, connections 4 and 8 are connected. In the absence of a sufficiently large potential, connections 4 and 6 are connected. For all relays in this machine, the connection labelled 16 is connected to ground, and 1 is connected to the emitter of a TIP142 NPN Darlington transistor. The collector of the transistor is connected to 12V. The base is connected to a pin of the PIC through a 1kΩ resistor. A transistor requires a voltage difference between the base and emitter, and this resistance is required to achieve the correct voltage difference. When the PIC outputs a high signal, the emitter of the transistor becomes 5V, activating the relay. If the PIC does not output a signal, or outputs a low signal, the relay is deactivated.

The translational motion of the slider is performed by a 12V DC motor driving a lead screw. The motor is powered directly by the power supply, either the +12V line and the -12V line. To move in the direction required to turn on the flashlights, the motor is powered by +12V. To move in the reverse direction, the motor is powered by -12V. The motor is controlled by a set of relays. One relay controls the direction of the motor, by selecting either the 12V line or -12V line of the power supply. The other relay selects whether or not the motor is running, by selecting between the output of the previous relay or nothing. The slider circuit diagram is shown in Figure 14.

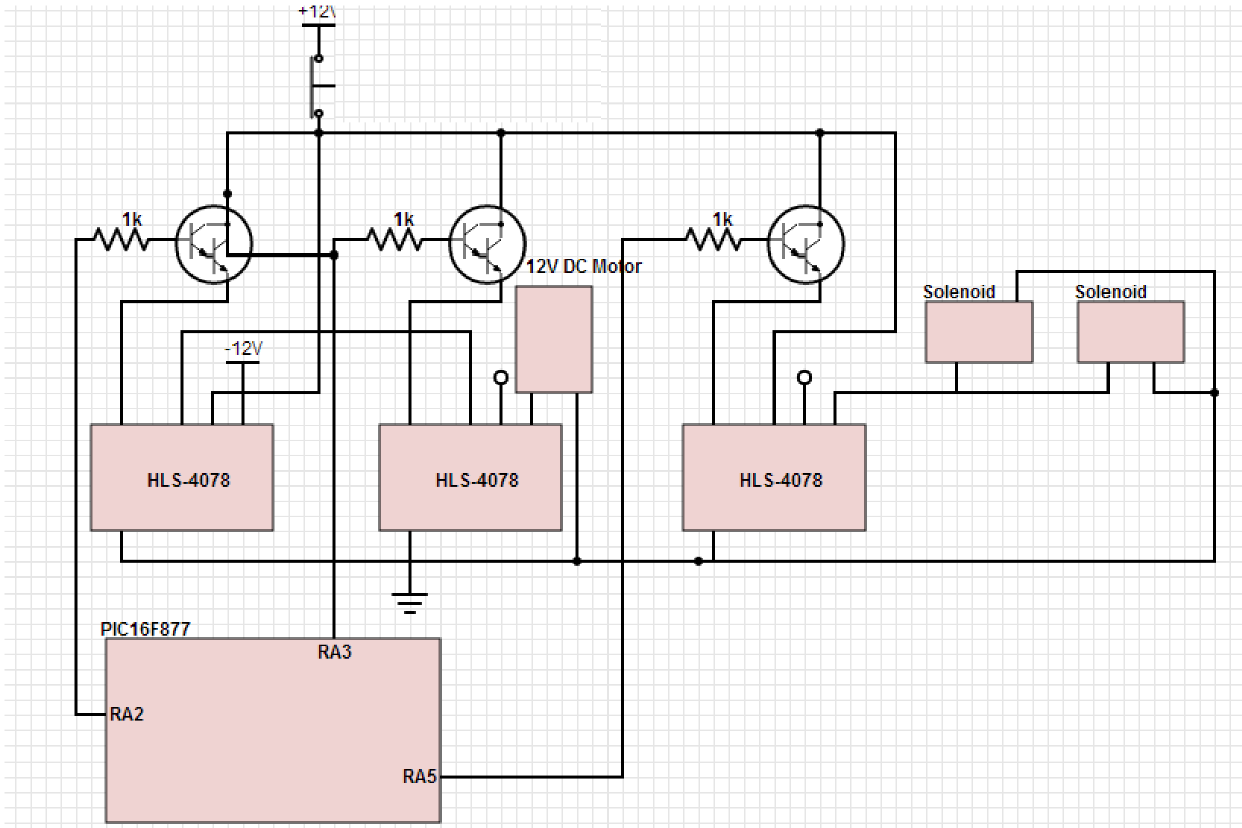


Figure 14 Slider Circuit Diagram

Table 7.1 - Slider Circuit Cost

Part	Cost
HLS-4078 Relay x 3	\$2.30 x 3 = \$6.90
TIP142 Transistor x 2	\$2.00 x 2 = \$4.00
Wiring	\$0.50
1kΩ Resistors x 3	\$0.20 x 3 = \$0.60
Printed Circuit Board	\$3.00
Total	**\$11.70**

7.2.2. Sensor arm

The sensor arm is driven downward and upward by a 12V DC motor of the same model as the slider motor. The mechanism and circuitry is identical to that of the slider. The sensor arm motor circuit is shown in Figure 15.

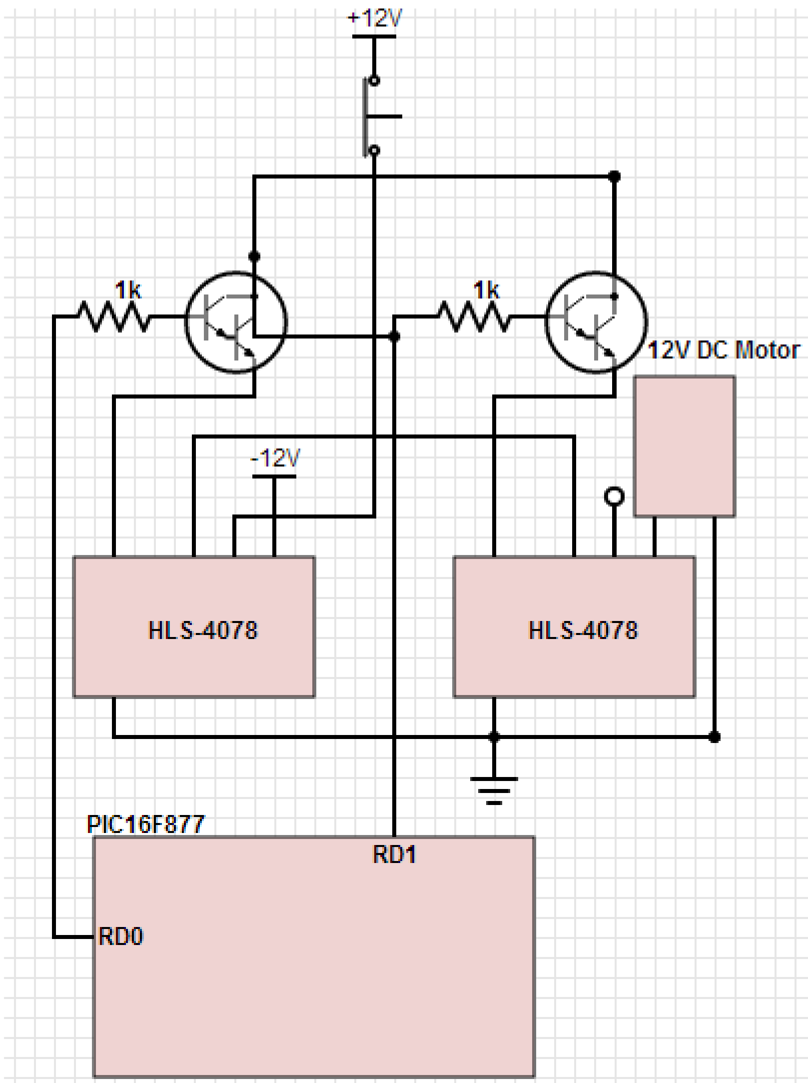


Figure 15 Sensor arm Motor Circuit Diagram

Table 7.2 - Sensor Arm Motor Circuit Cost

Part	Cost
HLS-4078 Relays x 3	\$2.30 x 2 = \$4.60
TIP142 Transistors x 2	\$2.00 x 2 = \$4.00
Wiring	\$0.50
1k Resistors x 3	\$0.20 x 3 = \$0.60
Printed Circuit Board	\$3.00
Total	**\$9.40**

7.2.3. Photosensing

Each flashlight has its own copy of the same photosensing circuit. This photosensing circuit consists of a button sub-circuit, and three photosensing sub-circuits. These four sub-circuits are connected to a multiplexer, which allows the PIC to process the data effectively. Since there are four inputs and one output, a four to one multiplexer is required. The chosen multiplexer was a 74HC153 dual four to one multiplexer. The PIC sends a two bit signal to all of the multiplexers simultaneously, acting as the select signals. The select signals choose which of four inputs to the multiplexer will be outputted to the PIC. The selector logic is shown in Table 7.3.

Table 7.3 - Selector Logic

Select Signal S_1S_0	Multiplexer Output
00	Button
01	Phototransistor 1
10	Phototransistor 2
11	Phototransistor 3

Each multiplexer outputs to a separate pin of the PIC. The button sub-circuit uses a TEPT5700 pushbutton to select between a high and a low signal, depending on whether or not the flashlight is absent or present. The phototransistors are used in common emitter configuration, which the amount of light hitting the phototransistor acting as the base. The collector of a phototransistor is connected through a 10kΩ 5V, and the emitter is connected to ground through a 330Ω resistor. A multiplexer input line is connected to the collector of each phototransistor. Hence, when little light hits the phototransistor, the multiplexer sees approximately 5V. When light from the flashlight LED hits the phototransistor current travels through the phototransistor, resulting in a large voltage drop across the 10kΩ resistor so that the multiplexer sees a low signal, nearly 0V. One iteration out of nine of the photosensing circuits has its diagram shown in Figure 16.

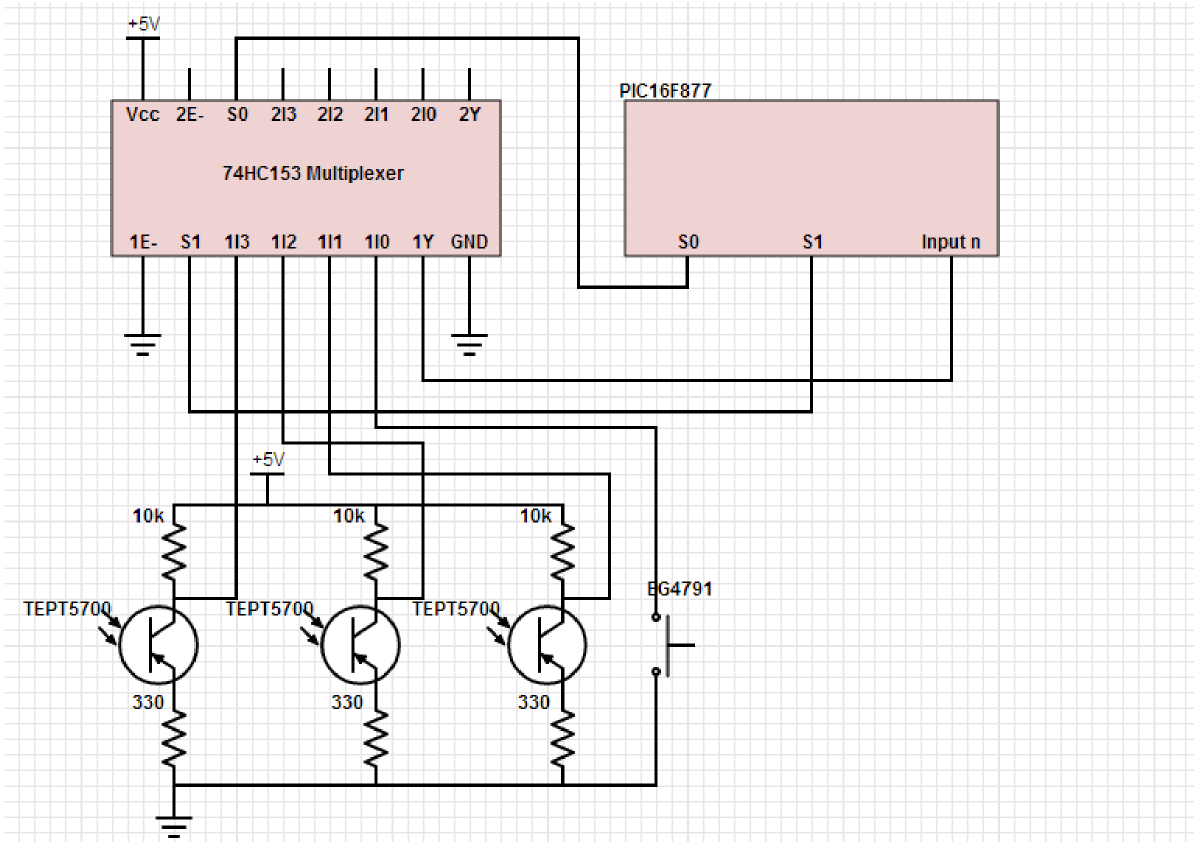


Figure 16 One Photosensing Circuit Schematic Diagram

Table 7.4 - Photosensing Circuit Cost

Part	Cost
TEPT5700 Phototransistors x 27	\$0.69 x 27 = \$18.63
EG4791 Pushbuttons x 9	\$0.75 x 9 = \$6.75
10kΩ, 330Ω Resistors x 54	\$0.15 x 54 = \$8.10
Wiring	\$2.00
74HC153 Multiplexers x 9	\$0.62 x 9 = \$5.58
Printed Circuit Boards	\$5.00
Total	**\$46.06**

7.2.4. Emergency Stop Switch

Shutting off the 12V line from the power supply will result in all of the relays closing, stopping all actuation. Hence, the emergency stop switch is a locking pushbutton which is normally closed.

Table 7.5 - Total Circuitry Cost

Component	Cost
Slider	$11.70
Sensor arm Motor	$9.40
Photosensing	$46.06
Pushbutton	$1.50
Wiring	$3.00
Total	**$71.66**

7.3. Calculations

7.3.1. Motor Power

The motors are powered directly by the +12V and -12V lines of the power supply. In direct connection with these lines, the current from the power supply is 2A and 0.5A respectively. Power is given by:

$$P = VI \tag{1}$$

where P is power, in watts, V is voltage, in volts, I is current, in amperes. Hence, the +12V power is:

$$P = 12 \times 2 \text{ W}$$
$$P = 24\text{W}$$

The -12V power is:

$$P = 12 \times 0.5 \text{ W}$$
$$P = 6\text{W}$$

7.3.2. Solenoid Power

The solenoid is given the same power as the motor when powered by the +12V line. The solenoid does not increase resistance greatly, and thus the current remains at 2A.

$$P = 12 \times 2 \text{ W}$$
$$P = 24\text{W}$$

7.3.3. Phototransistor Output Power

A transistor in common emitter configuration has voltage gain governed by:

$$\frac{V_{out}}{V_{in}} = \frac{R_c}{R_E + R_{tr}} \qquad (2)\ [R]$$

V_{out} is the voltage after the collector resistor, which is sent to the multiplexer. V_{in} is the voltage before the resistor at the collector, 5V. R_c is the resistance of the resistor at the collector, 10kΩ. R_E is the resistance of the resistor at the emitter, 330Ω. R_{tr} is the transresistance of the transistor, 230Ω for the TEPT5700. When no light hits the phototransistor, no current travels, and hence the voltage drop across the resistor, given by equation 3,

$$V = IR \qquad (3)$$

is zero, and thus V_{out} is 5V. When light from an LED hits the phototransistor, approximately 0.48mA travels through the phototransistor. Thus, there is voltage drop over the collector resistor and the phototransistor, although the latter is negligible. The voltage drop over the collector resistor is:

$$V = 0.00048 \times 10000 \text{ V}$$

$$V = 4.8\text{V}$$

Hence, the voltage seen by the multiplexer is:

$$V = V_c - V_{R,C}$$

$$V = 0.2\text{V}$$

Less than 1.35V is seen as a logical low by the multiplexer, and greater than 2.4V is seen as a high by the multiplexer, thus, these values are acceptable.

7.4. Suggestions for Improvement

7.4.1. Button Sub-circuit

The button sub-circuit uses floating values, which, despite working in some tests, are unreliable for signaling purposes. There are two alternatives: a pulldown resistor or a microswitch. A pulldown resistor uses a pushbutton to select between ground, and a controllable voltage. The circuit diagram for a pulldown resistor is shown in Figure 17. A microswitch selects between two values. A microswitch could have been used to select between 5V and ground, as shown in Figure 18. In both figures, the circular nodes represent connections to a multiplexer.

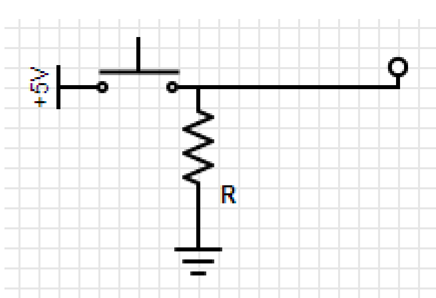


Figure 17 Pulldown Resistor

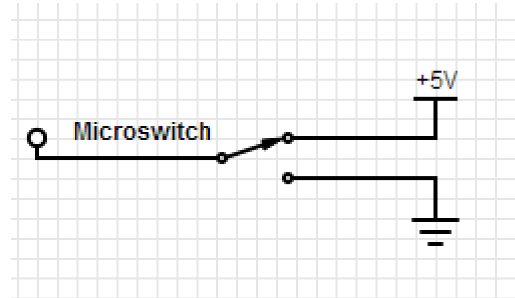


Figure 18 Microswitch

7.4.2. Motor Driving

Higher torque motor operation could have been used. Motors should have had a capacitor on both leads to smoothen the signal and help make the motor motion less sporadic. Using an alternative method of motor driving, such as an H-bridge, for example, which did not use relays, may have been worth more consideration as well as relays are expensive components.

7.4.3. Photosensing and PIC communication

Multiplexer output should be buffered before being sent to the microcontroller for processing. Buffering the multiplexer output would have resulted in more reliable photosensing, but at a slight cost. Two 74HC541 non-inverting buffers could have been used. The cost of each of these integrated circuits is $2.00. Thus, the total cost of $4.00 is sufficiently low that the benefit of implementing buffers would have outweighed the cost.

7.4.4. Wiring

The wiring could have been performed in a more organized and effective manner. Wires should be connected and attached to the side of the robot. Additionally, secure wire attaching methods should have been investigated, as soldering and using male and female connectors were prone to disconnecting on occasion. Solid wire may also have been preferable over stranded wire. Stranded wire is more durable against bending, however, stranded wire is less compatible with connectors.

7.4.5. Circuit Boards

Circuit boards could have been designed using software such as Eagle and printed. This may have reduced cost and simplified manufacturing and debugging.

8. Microcontroller

8.1. Assessment of Problem

The microcontroller had to address two major components in the operation of the robot. The first was to control the proper execution of the various steps in the actuation of the robot. The second was to provide the user the means to interact with the robot.

The first problem, the execution of actuation, had the following components to consider. Firstly, the order in which the steps of actuation were to be executed and the time allocated to complete each of these steps. Next, which pins of the PIC would be responsible for sending, and receiving signals to the various circuits controlling the actuation. After that, it had to be decided how to allocate the memory to save the data received during the actuation. More specifically which tray spots had flashlights, how many LEDs were on for each flashlight, and how long the operation of the robot took.

The second problem, the means in which the user could interact with the robot can also be split into parts. First, what the user had to do start the operation of the robot once the tray had been put in place. Next, what indications and data the user would be given during operation and finally what information the user could access after operation and how they would go about accessing it.

A secondary problem to consider in order to find the solutions to the two main problems is what PIC and components would be used on the microcontroller board to accomplish the above tasks.

8.2. Solution

The first step in the solution to the problem that was addressed was what PIC would be used and what other microcontroller board components would be used. In terms of the PIC, for this assignment two microcontrollers produced by Microchip were considered for use: the PIC16F877 and the PIC18F4620. The reason these microcontrollers were chosen as candidates was the PIC microcontrollers have one of the best cost to performance ratios in the market. Both of these PICs are in the midrange of the available PIC microcontrollers and are used in industry. The reason the new enhanced midrange PICs such as the PIC16887 were not considered is as these chips are newer, they are more prone to bugs so are not ideal to be used as a beginner.

The advantages of the two chips are as follows.

PIC16F877:

- A lot of available example code to work off of

- Has enough memory and is fast enough for the scope of this project

PIC18F4620:

- Faster than the PIC16 with higher capacity

- Unlike PIC 16 has USB and Ethernet support

- Internal clock so more I/O ports available to use

-More instruction available

For this project the PIC16F877 was selected since, as a beginner project, it was felt that the aid provided by the sample code will be more helpful than advantages provide by the PIC18. It was decided that the PIC16 will be fast enough and hold enough memory for this project. Also, the additional instructions of the PIC18 would not have been useful if the learning curve to programming was too high to overcome.

In terms of other aspects of the microcontroller board, USB support to program the PIC was achieved by using a second smaller PIC18 as a secondary chip. An external RTC clock was used since to keep time. A keypad was included to receive user inputs and a LCD display was used to convey information to the user.

The next part of the solution was to work out how the user interface would work. It was decided that when the robot first started up it would display the date followed by a welcome message. The user would then be prompted to start the robots operation by pressing any of the keys on the keypad. During the actuation first the LCD would display the start time. Next it would periodically update the time and display relevant information as it was calculated (such as the number of flashlights). When the operation is complete the end time would be displayed and the LCD would be sent to the end of operation menu. From this menu the user can check the number of working LEDs for each flashlight, the total number of flashlights, and the time it took to complete the operation.

Finally the solution to the actuation was addressed. First the PIN assignment to the actuators was determined. Pins D0 and D1 controlled the motors the lifted an dropped the top sensor arm (D0 was for direction and D1 was for on/off). Pins A2 and A3 controlled the motors that brought the slider left and right (A2 was for direction and A3 was for on/off). Pin A5 controlled pushing the solenoids in the slider in and out. Pin A0 and A1 controlled what the sensing circuits would look for (i.e. presence of flashlight (00), status of LED 1

(01), status of LED 2 (10), status of LED 3 (11). Finally C0-C2,C5-C7,B0 and B2-B3 controlled the inputs from the nine sensing circuits.

The actuation itself was as follows. The top sensing arm was brought down, the solenoids in the slider were activated and the slider was moved left. Then the sensing commands of 00, 01, 10 and 11 were sent out to determine the number of flashlights and the number of LEDs working for each flashlight (each of these pieces of data saved in memory). Then the slider was moved right the solenoids were pulled back in the top sensing arm was brought back up.

The cost of the items associated with the PIC are as follows:

Table 8.1 - Microcontroller Cost Analysis

- PIC DevBugger Development Board (without LCD, keypad, RTC and coin battery) (excluding DC adaptor)	$48.50
- LCD+Keypad (with the encoder chip)	$6
- Real-time Clock (RTC) Chip and Coin Battery	$5
- Design Kit Power Supply	$15
-12V DC connector	$0.53

8.3. Computer Programs

The important aspects of the code are highlighted below.

First it calls supporting documents and it lists the macros. A macro of interest is the AdjustPCL macro that is used to fit more lines of text and is called by the various lookup tables at the end of the code.

Then it continues to pin assignments.

Table 8.1 - Pin Assignment Table

Pin	Location	Function
A0 - A1	Sensing Bar	Selects what's being sensed
A2	Slider motor	Chooses Slider Direction
A3	Slider motor	Turns motor on and off
A5	Slider Solenoid	Turns solenoid on and off
B0,B2-B3	Sensing Bar	Sensing inputs 7-9
B1, B4-B7	Keypad	Keypad input
C0-C2,C5-C7	Sensing Bar	Sensing inputs 1-6
C3-C4	Real time clock	RTC
D0	Sensing Bar motor	Chooses direction
D1	Sensing Bar motor	Turns motor on and off
D2-D7	LCD	Output to LCD

Next the actual operation code is started starting with the display of a welcome message, followed by the date, then a waiting screen.

Once a button is pressed it saves the start time.

Then it enters the actuator start code.
The top sensing bar is brought down, the solenoids are pushed out, and the slider is moved left.

Then it enters the sensing code.
The variable for number of flashlights is created and the 00 output is sent to the bar. For each spot with the button pressed down the number of flashlights is incremented by one. Next a 01 signal is sent out and the 9 variables are created for the 9 flashlights. It should be noted that each variable in the code is assigned to its own unique part of memory. For each LED sensed the appropriate LED counter is incremented. This is repeated for the 10 and 11 outputs.

Next there is the actuator end code.
The steps of the start code are done in reverse

Finally, followed by the save end time code.
(This was the only part of the code that continued to fail tests at the time when the robot was ceased to be worked on. A solution to this issue is discussed in improvements.) In this step the final time is saved to a 2 digit number representing the number of seconds.

Then there is the end menu starting with a comprehensive 'if' statement to determine what button has been pressed (below). Followed by the appropriate information for each button. The end of the code saves the various strings and some functions used by the earlier code.

```
    cblock 0x204
    temp3:0, temp3_hi, temp3_lo
    endc
    clrf     temp3
    swapf          PORTB,W    ;Read PortB<7:4> into W<3:0>
    andlw          0x0F
    xorlw    B'0000'
    movwf    temp3
    btfsc    temp3, 0
    goto     check2
    btfsc    temp3, 1
    goto     check2
    btfsc    temp3, 2
    goto     check2
    btfsc    temp3, 3
    goto     check2
    goto     its1
check2
...
```

8.4. Improvements

Some improvements to this subsystem would be more reliable timing code the current code occasionally crashes if the ones digit in the seconds of the end time is lower than the ones digit of the seconds in the start time. This is because the final step of calculating the runtime involves subtracting these two values so if you were subtracting a larger number from a smaller number the code exhibited unpredictable behaviour. This could be solved with a rather long if statement (such as the one used in the final menu) that would send the user to one of 2 areas. One if the first value is larger and the other if the second value is larger.

Another improvement would be to save results to the EEPROM memory so a user can access the results of a previous trial. This would e useful since the user could test multiple trays at a time before checking results.

A final improvement would be an easier to navigate menu that requires less previous knowledge to operate. The current menu works fine but a more elegant menu would have been implemented before the robot was in a final state.

8.5. Figures

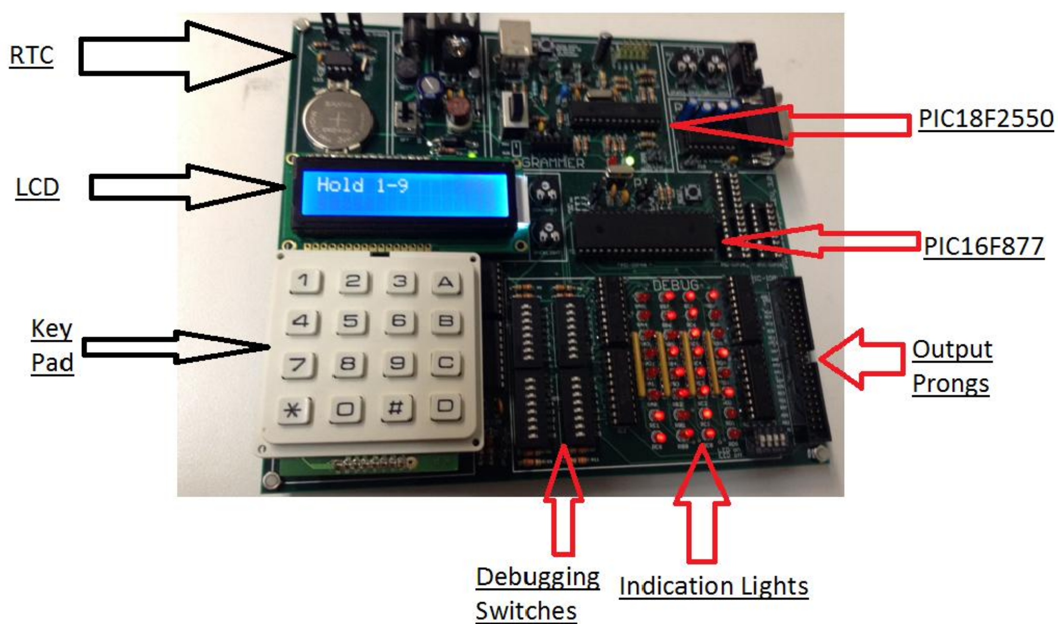


Figure 19 DevBugger Rev 3.0

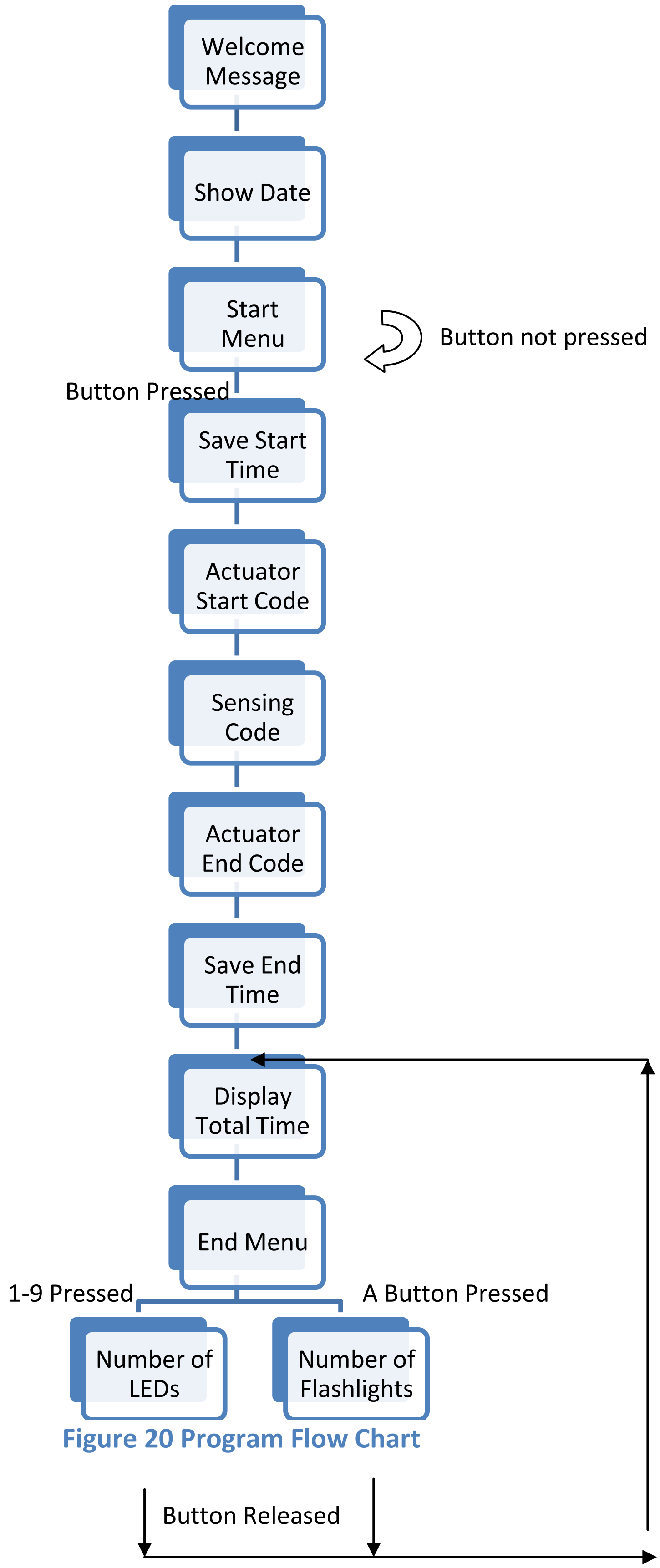


Figure 20 Program Flow Chart

9. Integration

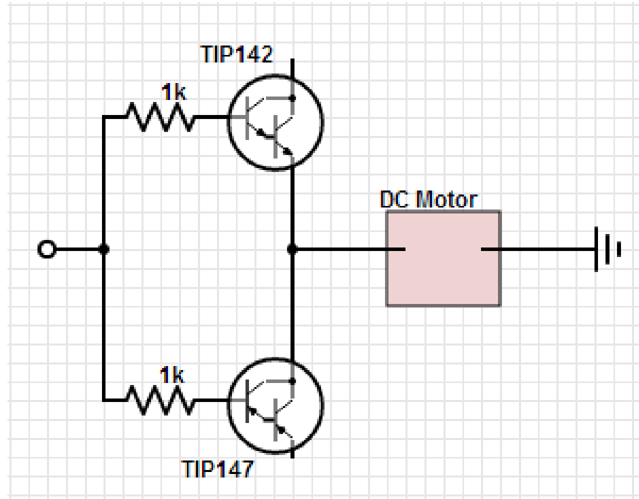


Figure 21 Push-Pull Transistor Circuit

Initial integration began in week 9. The slider was test-driven by the Helishun DC motor powered by previous circuit iterations. It was discovered that the previous circuit iteration had problems driving the motor when the motor had a significant load. In fact, the old circuit was incorrectly designed. This circuit is shown in Figure 19. Furthermore, it was noticed that the power of the solenoids were quite weak when their strokes were not very small. An adjustment system was constructed during this stage as detailed in the electromechanical system's solution section.

A photosensing circuit for one flashlight was testing with the PIC code, and modifications were made to both. The circuit, which used transistors in emitter follower mode, resulting in floating values being used as input for the multiplexer. The circuit was redesigned into its current form, and the PIC code had to be changed to accommodate the change that a low value now meant that a working LED is present, whereas it used to mean a working LED was not present.

Final integration began in week 11. The objective of this stage of integration was to complete actuation and photosensing. The slider was controlled by the circuits and the PIC, and its motor and relays worked as intended. However, the slider had problems turning on all of the flashlights. This is because the force needed to apply enough friction for the rotation of the flashlight rings to occur was miscalculated, as discussed in the electromechanical section under improvements. At this stage, calibration was performed by changing the code to alter the run time of each motor in each direction. The objective was to have the motor drive the slider sufficiently far in the initial direction to turn on the flashlights, and to have the motor drive the slider the same distance in the reverse direction. Calibration was also performed by adjusting the position of the slider. Photosensing of multiple flashlights was also performed at this time. The microcontroller outputted the select signals to all of the multiplexers and received all of the information about each flashlight. The problem with the button circuit was discovered at this junction, in which some buttons maintained floating values after being activated and then released. Furthermore, the phototransistor circuits were not always successful at detecting the presence of a working LED light. However, due to the time needed to solve each of these problems, and the little amount of time remaining, the team decided to terminate the project at this stage.

10. Improvement Suggestions

The majority of minor improvement suggestions are organized by subsystem and discussed with their respective subsystems. A few minor improvements that are associated mainly with the integration process will be discussed here, along with major design changes.

The wiring configuration and process could have been improved. Wiring was color coded to a degree, but the color coding could have been more effectively done. Additionally, the wiring could have been housed in a certain area so that the wires would not be free to move all over the machine. This could have been achieved through a physical structure such as a hollow container along one wall of the machine. The attachment of the wires to the microcontroller should have been made more secure and more assembly-friendly.

A combined effort across all three subsystems could have been made to develop a jam reactive mechanism. A combination of electromechanical work and circuitry could be implemented to detect jamming, such as a position sensor. Upon detection of jamming, a signal would be sent to the microcontroller. The microcontroller would then attempt to resolve the jam through a subroutine, such as telling the motor to jerk.

A better power supply should have been used. An alternative power supply could have provided more current, resulting in stronger motor motion, and more normal force from the solenoids. Additionally, an alternative power supply could have provided the same amount of current for +12V and -12V, so that the motors would operate with equal strength in both directions. This would be more precise as it would eliminate the need for PIC calibration.

A major change that is plausible would be to alter the function of the slider. Instead of the entire slider being driven by a motor, only the bar that turns on the flashlights should move. The relays would push the tray into the bar, and springs underneath the bar would push the tray back into its initial position after the flashlights are turned off. Driving the bar without the slider would put less strain on the motor and the bar would move more forcefully. However, it may be more difficult for the solenoids to supply adequate force when pressing the tray into the bar, instead of the bar into the tray, as the tray weighs more.

11. Schedule

11.1 Critical Path Analysis

Table 11.1 - Table of Tasks

Activity Code	Activity Description	Duration (Week)	Preceded By
A	Functional Decomposition	1	None
B	Tray Design	1	A
C	Securing and Turning FL	1	A
D	Sensing	1	A
	Microcontroller		
E	Run Example Programs	2	None
F	Complete Pseudocode	1	C, D
G	Key Pad and LCD	2	E, F
H	Timer	1	G
I	Actuator Outputs	1	G
J	Sensing Inputs, Outputs	1	G
K	Debugging	1	H, I, J
L	Additional PIC Functionality	1	K
	Circuits		
M	Design Photosensing Circuit	2	D
N	Design Motor Driving Circuit	2	C
O	Design Solenoid Circuit	2	C
P	Soldering and Testing	3	M,N, O
	Electromechanical		
Q	Solid Works Drawings	1	B, C, D
R	Material and Actuator Selection	1	Q
S	Build Tray	1	R
T	Build Frame and Stand	1	R
V	Build Tray Holding System	1	S, T
W	Building Slider	1	V
X	Building Sensor Arm	1	S
Y	Build Slider Restraining System	1	W
Z	Testing	1	X, Y
	Integration		
AA	Debugging Photosensing (M and C)	1	J, P
AB	Debugging Actuators (M and C)	1	I, P
AC	Physical Attachment (C and E)	1	AA, AB, Z
AD	Debugging All (M, C and E)	1	AC, L
AE	Additional Functionality	1	AD
AF	Finalization	1	AE

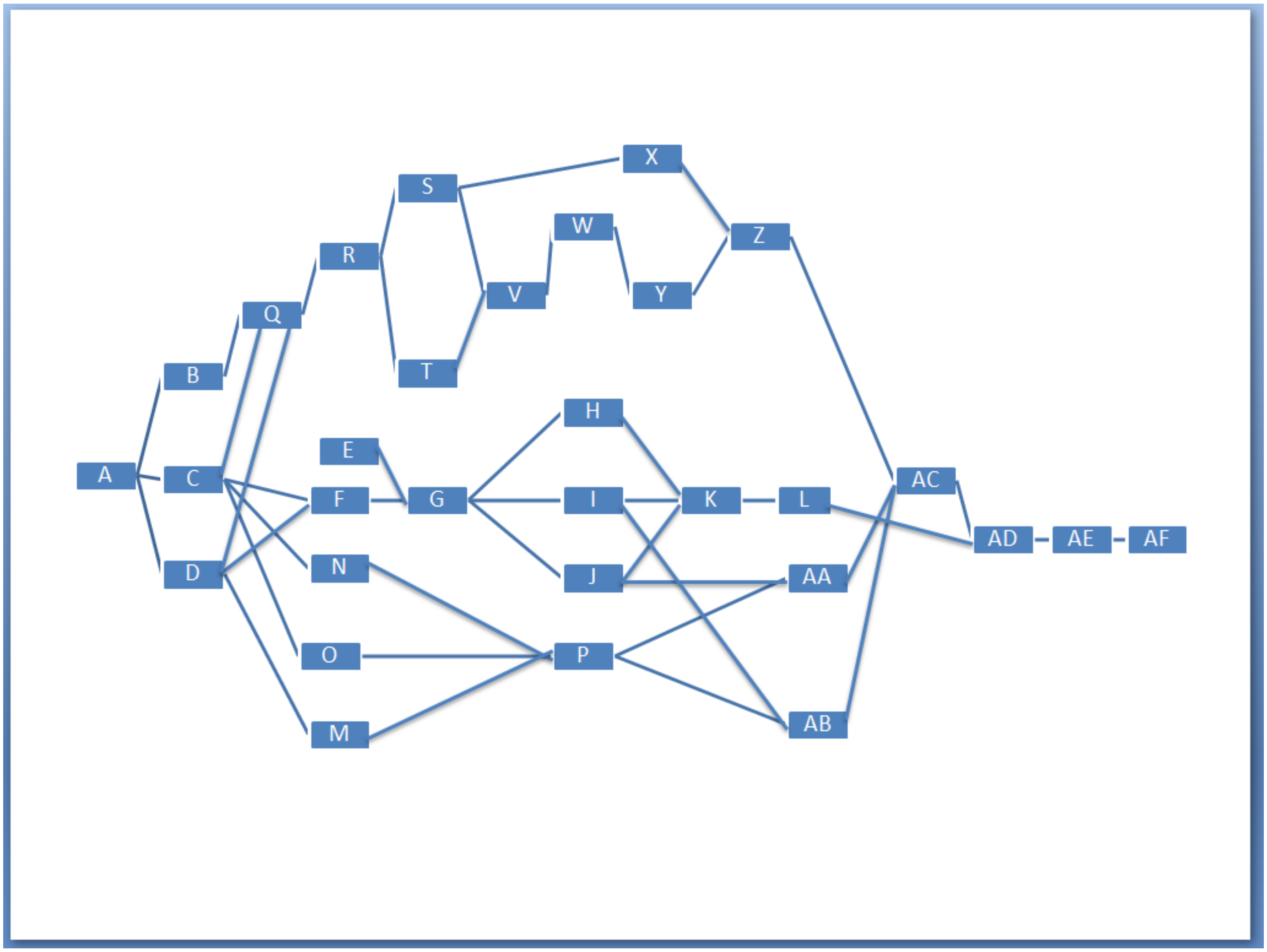


Figure 22 Network of Tasks

Table 11.1 - Chart of Possible Paths

Path	Length
A, B, Q, R, S, X, Z, AC, AD, AE, AF	11
A, B, Q, R, S, V, W, Y, Z, AC, AD, AE, AF	13
A, B, Q, R, T, V, W, Y, Z, AC, AD, AE, AF	13
A, C, Q, R, S, X, Z, AC, AD, AE, AF	11
A, C, Q, R, S, V, W, Y, Z, AC, AD, AE, AF	13
A, C, Q, R, T, V, W, Y, Z, AC, AD, AE, AF	13
A, D, Q, R, S, X, Z, AC, AD, AE, AF	11
A, D, Q, R, S, V, W, Y, Z, AC, AD, AE, AF	13
A, D, Q, R, T, V, W, Y, Z, AC, AD, AE, AF	13
A, C, F, G, H, K, L, AD, AE, AF	11
A, C, F, G, I, K, L, AD, AE, AF	11
A, C, F, G, J, K, L, AD, AE, AF	11
A, C, F, G, I, AB, AC, AD, AE, AF	11
A, C, F, G, J, AA, AC, AD, AE, AF	11
A, D, F, G, H, K, L, AD, AE, AF	11
A, D, F, G, I, K, L, AD, AE, AF	11
A, D, F, G, J, K, L, AD, AE, AF	11
A, D, F, G, I, AB, AC, AD, AE, AF	11
A, D, F, G, J, AA, AC, AD, AE, AF	11
E, G, H, K, L, AD, AE, AF	9
E, G, I, K, L, AD, AE, AF	9
E, G, J, K, L, AD, AE, AF	9
E, G, I, AB, AC, AD, AE, AF	9
E, G, J, AA, AC, AD, AE, AF	9
A, C, N, P, AA, AC, AD, AE, AF	12
A, C, N, P, AB, AC, AD, AE, AF	12
A, C, O, P, AA, AC, AD, AE, AF	12
A, C, O, P, AB, AC, AD, AE, AF	12
A, D, M, P, AA, AC, AD, AE, AF	12
A, D, M, P, AB, AC, AD, AE, AF	12

The Critical Path is: A, (B, C, D), Q, R, (S, T), V, W, Y, Z, AC, AD, AE, AF
A delay in any of these activities would have caused a delay in the project completion. It should be noted that this path is the path that involved the electromechanical subsystem, which suggested that ensuring that the electromechanical subsystem's progress ensured the quickest completion time of the project.

11.2 Gantt Chart

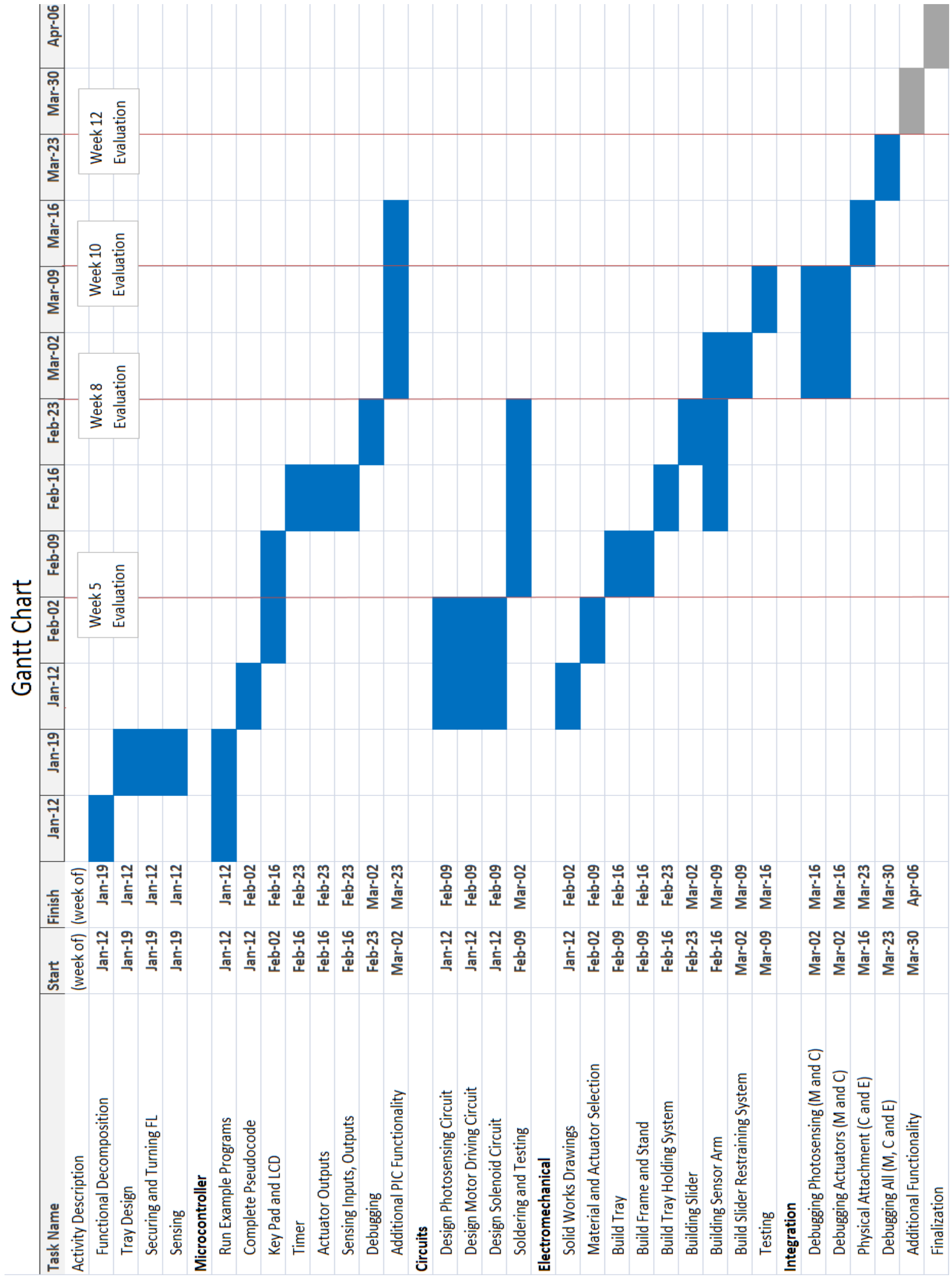


Figure 23 Gantt Chart

12. Conclusion

Research into the relevant industry was done, criteria were set forth, and a conceptual design was created. The team divided into electromechanical, circuit, and programming subsystems and realized the conceptual design through integration. The design solution to the Request for Proposal described in this report achieved most of the required functionality. Ultimately, the machine was incapable of consistently turning on all of the flashlights, and additionally had sporadic sensing errors. Inability to turn on the flashlights arose from the design not supplying sufficient tangential force to the flashlight plastic rings. This could have been remedied by implementing an alternative power supply, using more solenoids, or using a different design. Sensing error arose from assembly imperfections and incorrect pushbutton circuits. Alternative pushbutton circuits were outlined. The machine also had some issues determining the run time of the operation.

The design process could have been improved by greater team communication, earlier integration, and less black-boxing of other subsystems or parts. Many subsystem specific issues only became clear after testing with other subsystems. Additionally, more significant design changes, such as replacing the solenoids, could have been investigated if problems had been discovered earlier.

Above fixing functionality, the design could be improved by taking in multiple trays of flashlights without human work. For example, trays could be fed into the machine through a conveyor belt. The photosensing mechanism could be replaced by a cheaper alternative, such as using photo resistors. Motor driving circuits could have been replaced with more effective alternatives, such as implementing capacitors. Signals could have been regulated more, especially between the multiplexers and the microcontroller. A more intuitive and informative user interface could have been constructed, perhaps through connecting to a computer. Electromechanical components could have been more precise to minimize randomness. Lastly, better integration could have resulting in overall smoother operation.

13. Description of Overall Machine

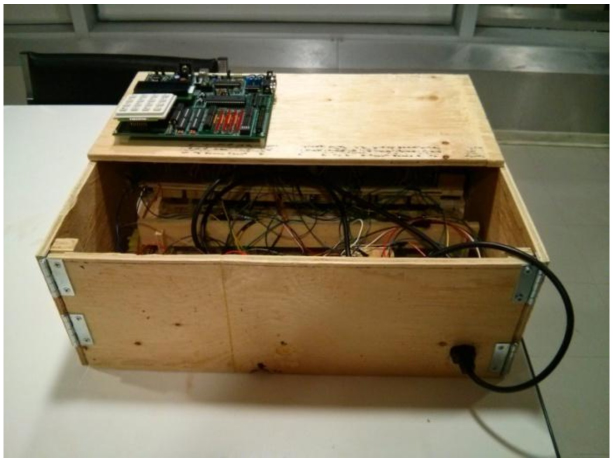


Figure 24 "Warmongler"

The *Warmongler* is a proof-of-concept prototype of a tool LED flashlight testing machine capable of verifying the functionality of the LEDs of these tool flashlights in a simulated assembly line. The machine is a 19.5’’x16’’x7.375’’ rectangular prism shaped robot that can receive a tray of a maximum of nine LED tool flashlights and subsequently test for the functionality of each of the three LEDs for each flashlights. The operator of the robot can then verify the state of each flashlight through an LCD interface, check for the total number of flashlights inserted into the tray, and then retrieve the tray with the flashlights turned off.

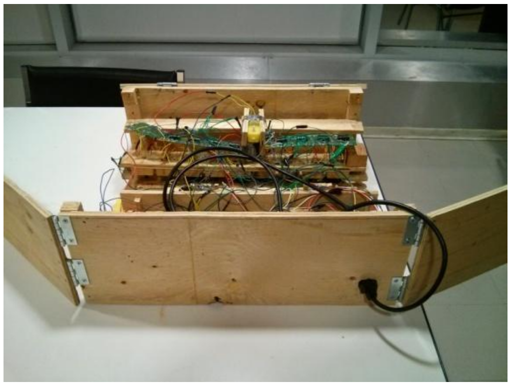


Figure 25 Hinged Doors

Warmongler is a prototype whose static, passive dynamic, and active dynamic mechanisms work together with photosensing, signal sensing, and actuation circuitry, along with programmed logic to perform its flashlight testing functions. The frame of the robot consists of three magnetic doors, which simultaneously allows for the access to the interior of the robot for debugging, repairs, or replacements, and the protection of the interior components from outside exposure. The tray, which accepts flashlights in a certain orientation, can be inserted through a hole on the entrance side of the machine, which is then guided in the inside machine in one dimension to the position required for the testing of the LED lights. A magnetic holder near the exit side of the machine can then exert a restraining force on the inserted tray and keep it in the right position for the remaining dimension. When the machine is just operated, a sensor arm above the tray is also activated by a leadscrew mechanism actuated by a DC motor, which lowers a board of nine sensor holders, each with a button in the middle and three phototransistors positioned in the orientation of the three LEDs. This action activates any button which is pressed onto existing flashlights, and at the same time secures the positions of the flashlights through a vertical force. After this action is complete, the solenoids on the slider of the machine activate a belt that pushes screw threads into the inserted flashlights with a force. This action secures the screw threads within the rotatable rings of the flashlights, which determines the on/off state for each flashlight. Soon after, a leadscrew mechanism on the side of the slider activates, and pulls the slider in one direction. This motion also pulls on the belt, whose grip onto the flashlight rings means that it will rotate and turn on the flashlights. At this point, any working LEDs will be turned on, and this information is transferred through the photosensing circuit to the DevBugger, which contains the programmed logic for processing of this information. This is the half way point of operation, and the robot reverses these actions before the operation's end. A short time after the opening of the flashlights, the slider's leadscrew mechanism reverses the position of the slider, and now pushes it along with the belt which is attached to the flashlights' rings to turn them off. Thereafter, the solenoids release their force onto the belt, and the leadscrew mechanism of the sensor arm lifts the board with the sensor holders up, thus releasing the flashlights the vertical pressure previously exerted by the button sensors. At this point, the operation will be finished, and the tray is able to be removed by the insertion of a new tray that can push the hold tray out of the machine, ready to be operated again.

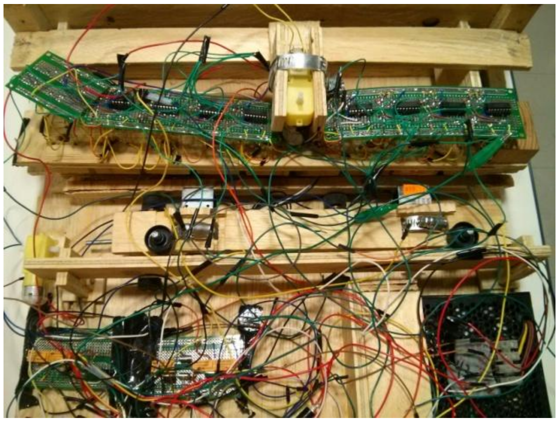


Figure 26 Internal Components

The *Warmongler* is equipped with a power supply which can be powered by a cable through a hole in the side of the robot. It is also equipped with a red stop button, which when pressed, will stop all actuation mechanisms, regardless of their state of operation, and bring any activities to a halt.

14. Standard Operating Procedure

14.1. Scope

This Standard Operating Procedure (SOP) pertains to usage of the machine by an operator.

14.2. Objective

This SOP outlines the proper operation of the machine and explains the entire functionality of the user interface.

14.3. Procedure

1. Load the desired number of flashlights onto the tray by placing one flashlight in any well, with one LED aligned with the arrow indicated on each well of the tray.
2. Slide the tray into the slit at the bottom of the machine, while ensuring that the tray is held flat.
3. Continue to push the tray into the machine until notable resistance is felt. The tray is now in its operating position.
4. Press any key on the keypad to begin operation. Wait until the LCD screen displays the number of flashlights, the end time, and then "Hold 1-9".
5. Hold the 'A' key to view the number of flashlights.
6. Hold a key from 1 to 9 to view the number of working LEDs for the flashlight in the corresponding well, numbered from left to right. Zero working LEDs will be displayed for a well without a flashlight.
7. Release the pressed key to display the total run time and return to "Hold 1-9".
8. At any point during operation, pressing the red emergency stop button will terminate all motion.

14.4. Safety

The entire of this machine contains moving parts and electrical wiring. Do not disassemble. Do not slide anything other than the tray into the tray slot on the machine.

15. References

[1] C. Scott. *Repetitive Strain* Injury. [Online]. Available: http://web.eecs.umich.edu/~cscott/rsi.html

[2] Encyclopedia Britannica. *The Assembly Line*. [Online]. Available: http://www.britannica.com/EBchecked/topic/648000/history-of-the-organization-of-work/67046/The-assembly-line

[3] Perkin Elmer. *Avalanche Photodiode.* [Online]. Available: http://www.perkinelmer.com/CMSResources/Images/44-6538APP_AvalanchePhotodiodesUsersGuide.pdf

[4] R. Emami. *Multidisciplinary Engineering Design.* "Request for Proposal #1".

[5] Sparkfun. *Solenoid - 36v.* [Online]. Available: https://www.sparkfun.com/products/10391

[6] Huayuexin Precise Ware Co. Ltd. *Toy Motor*. [Online]. Available: http://hyx-gears.manufacturer.globalsources.com/si/6008826884986/pdtl/Geared-motor/1084105044/Toy-Motor.htm/

[7] Practical Machinist. *Laser Cutting Tolerances.* [Online]. Available: http://www.practicalmachinist.com/vb/general/laser-cutting-tolerances-210793/

[8] Daycounter, Inc. *Solenoid (Electromagnet) Force Calculators*. [Online]. Available: http://www.daycounter.com/Calculators/Magnets/Solenoid-Force-Calculator.phtml

[9] Meadinfo. *ACME Lead Screw Torque Calculator.* [Online]. Available: http://www.meadinfo.org/2009/07/acme-lead-screw-torque-calculator.html

[10] A. Williams. *PIC Instructions.* [Online]. Available: http://tutor.al-williams.com/pic-inst.html

[11] WinPicProg. *PIC Tutorial One - LED's*. [Online]. Available: http://www.winpicprog.co.uk/pic_tutorial1.htm

Appendix A: Electromechanical Calculations

Solenoid Calculations

Question: What is the force supplied by solenoids onto the flashlights?

Assuming the chosen stroke is just less than 2mm, the solenoid datasheet gives a force of 2.4 kgf under 36V operation. Assuming the solenoids are operating under the assumption of 12V, the ratio of this to the default 36V would give the force supplied by the solenoid. This is because equations (1), which is the force of a solenoid, and (2), Ohm's Law, implies (3), the proportionality between the force and the voltage.

$$F = \frac{2\mu_0 nIA}{g} \text{ (1) [8]} \qquad I = \frac{V}{R} \text{(2)} \qquad F \propto V \text{ (3)}$$

Where μ_0 is the magnetic permeability of free space, n is the number of coils in the wire, I is the current of the system, A is the area of the solenoid, g is the size of the gap between the metal rod and the solenoid, V is the voltage given to the solenoid, and R is the resistance of the solenoid.

In the final design, 2 solenoids were used to supply the force under the conditions as stated, which gives the following calculation for the total force.

$$2.4\frac{kgf}{solenoid} \times \frac{12V}{36V} \times 2\ solenoids = 1.6\ kgf \approx 16N$$

Answer: the force supplied by the solenoids is about 16N, where each solenoid supplies about 8N of force.

Leadscrew Mechanism Calculations

Question: What is the lifting and lowering force of the leadscrew?

Assuming an operating voltage of 12V for the Shenzhen DC motor, the datasheet in appendix C had a torque of 5kgf·cm[6]. However, in order to calculate the force of the leadscrew mechanism, many other parameters are needed as listed in equation (4).

$$T_{raise} = \frac{Fd_m}{2}\left(\frac{l + \pi f d_m}{\pi f d_m - fl}\right) \text{(4)[9]}$$

$$T_{lower} = \frac{Fd_m}{2}\left(\frac{\pi f d_m - l}{\pi d_m + fl}\right) \text{ (5)[9]}$$

Where T is the torque, F is the force on the load, d_m is the mean diameter of the screw, l is the lead, and f is the coefficient of friction of the screw material.

Although it was possible to estimate the force, the lead and coefficient of friction were difficult to determine. Thus, experiments were done instead to see if the force from the motors were enough. These experiments showed that the force was enough.

Answer: the force of the leadscrew was undetermined, but experiments showed that it was enough to carry its loads.

Appendix B: Code

The following code was the final version used for the robot:

```
;*************************************
****************************
; Keypad/LCD Test Code
; Assembler : mpasm.exe
    list p=16f877           ; list
directive to define processor
    #include <p16f877.inc>      ;
processor specific variable definitions
    __CONFIG _CP_OFF & _WDT_OFF &
_BODEN_ON & _PWRTE_ON & _HS_OSC
& _WRT_ENABLE_ON & _CPD_OFF &
_LVP_OFF

    #include <lcd.inc>
                    ;Import LCD
control functions from lcd.asm
    #include <rtc_macros.inc>

        udata_shr
  COUNTH        res 1       ;const
used in delay
        COUNTM  res     1
        ;const used in delay
        COUNTL  res     1
        ;const used in delay
        Table_Counter       res
        1

    ORG     0x0000  ;RESET vector
must always be at 0x00
    goto    init    ;Just jump to the
main code section.

;*************************************
*
; Display macro
;*************************************
*
Display macro       Message
            local       loop_
            local       end_
            clrf
        Table_Counter
            clrw

loop_   movf        Table_Counter,W
            call
        Message
            xorlw
        B'00000000' ;check WORK
reg to see if 0 is returned
            btfsc
        STATUS,Z
                    goto
        end_
            call
        WR_DATA
            incf
        Table_Counter,F
            goto        loop_
end_
            endm

AdjustPCL macro     TableEntries
    local   end_
    cblock 0x170
    temp:0, temp_hi, temp_lo
    endc

    movwf temp
    movlw HIGH TableEntries
    movwf PCLATH
    movf  temp, w
    addlw LOW TableEntries
    btfsc STATUS, C
    incf PCLATH, f
    movwf PCL

end_
    endm


    code    ;'code' lets the linker
decide where in
        ;program memory to put
these instructions.
init
    clrf    INTCON      ; No interrupts

    bsf     STATUS,RP0      ; select
bank 1
    clrf    TRISA       ; All port A is
output
    movlw   b'11111111'     ; Set
required keypad inputs
    movwf   TRISB
    movlw   b'11111111'     ; Set
required inputs
    movwf   TRISC
    clrf    TRISD       ;
    movlw   b'00000111'     ; Set
required inputs
    movwf   TRISE

    ;Set SDA and SCL to high-Z first as
required for I2C
                bsf
TRISC,4
                bsf
TRISC,3

    bcf     STATUS,RP0      ; select
bank 0
    clrf    PORTA
    clrf    PORTB
    clrf    PORTC
    clrf    PORTD
    clrf    PORTE

    call    i2c_common_setup

    call    InitLCD ;Initialize the LCD
(code in lcd.asm; imported by lcd.inc)

Hi      Display
        Welcome_Msg1
    call            HalfS
    call            HalfS
    call    Clear_Display
    Display     Welcome_Msga
    call            Switch_Lines
    call    showdate
    call            HalfS
    call            HalfS
    call            Clear_Display
    Display
        Welcome_Msg1
    call            Switch_Lines
    Display     Welcome_Msg2
    btfss           PORTB,1     ;Wait
until data is available from the keypad
    goto            $-1
    btfsc           PORTB,1     ;Wait
until key is released
    goto            $-1
    call            Clear_Display
    Display
        Welcome_Msgb
    call            Switch_Lines
    call    showtime
;Save Start Time
    cblock 0x256
    startmin:0, startmin_hi,
startmin_lo
    endc
    clrf    startmin_hi
    clrf    startmin_lo
    clrw

                rtc_read    0x01
                ;Read Address
0x01 from DS1307---min
                movfw       0x77
                movwf
        startmin_hi
                movfw       0x78
                movwf
        startmin_lo

    cblock 0x264
    startsec:0, startsec_hi, startsec_lo
    endc
    clrf    startsec_hi
    clrf    startsec_lo
    clrw

                ;Get seconds
                rtc_read    0x00
                ;Read Address
0x00 from DS1307---seconds
                movfw       0x77
                movwf
        startsec_hi
                movfw       0x78
                movwf
        startsec_lo

    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    nop

    call            Clear_Display
```

```
    Display
        Welcome_Msg3
    call            Switch_Lines
    call    showtime
    call            HalfS
    call            HalfS
```

;Actuator Start Code

```
    movlw   B'00000011'
    movwf   PORTD
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS

    movlw   B'00000000'
    movwf   PORTD
    call            HalfS
    call            HalfS

    movlw   B'00100000'
    movwf   PORTA
    call            HalfS
    call            HalfS

    movlw   B'00101000'
    movwf   PORTA
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
```

;Sensing Code

```
    cblock 0x252
    numflash:0, numflash_hi,
numflash_lo
    endc
    clrf    numflash_hi
    clrf    numflash_lo
    clrw
    addlw   B'00000000'
    addwf   numflash


    cblock 0x248
    flash1:0, flash1_hi, flash1_lo
    endc
    clrf    flash1

    cblock 0x244
    flash2:0, flash2_hi, flash2_lo
    endc
    clrf    flash2

    cblock 0x240
    flash3:0, flash3_hi, flash3_lo
    endc
    clrf    flash3

    cblock 0x236
    flash4:0, flash4_hi, flash4_lo
    endc
    clrf    flash4

    cblock 0x232
    flash5:0, flash5_hi, flash5_lo
    endc
    clrf    flash5

    cblock 0x228
    flash6:0, flash6_hi, flash6_lo
    endc
    clrf    flash6

    cblock 0x224
    flash7:0, flash7_hi, flash7_lo
    endc
    clrf    flash7

    cblock 0x216
    flash8:0, flash8_hi, flash8_lo
    endc
    clrf    flash8

    cblock 0x208
    flash9:0, flash9_hi, flash9_lo
    endc
    clrf    flash9

    cblock 0x212
    flasht:0, flasht_hi, flasht_lo
    endc
    clrf    flasht

    movlw           B'00100000'
    movwf           PORTA
                    ;set all bits on

    movf            PORTC,W
;Read PortC<7:4> into W<3:0>
    xorlw   B'11111111'
    andlw   B'00000001'
    ;swapf  flash1, W
    addwf   numflash, F

    movf            PORTC,W
;Read PortC<7:4> into W<3:0>
    xorlw   B'11111111'
    andlw   B'00000010'
    addwf   flasht, F
    btfsc   flasht, 1
    goto    f2ay
    goto    f2an

f2ay    clrw
    movlw   b'00000001'
    addwf   numflash, F
f2an    clrf    flasht

    movf            PORTC,W
;Read PortC<7:4> into W<3:0>
    xorlw   B'11111111'
    andlw   B'00000100'
    addwf   flasht, F
    btfsc   flasht, 2
    goto    f3ay
    goto    f3an

f3ay    clrw
    movlw   b'00000001'
    addwf   numflash, F
f3an    clrf    flasht

    movf            PORTC,W
;Read PortC<7:4> into W<3:0>
    xorlw   B'11111111'
    andlw   B'00100000'
    addwf   flasht, F
    btfsc   flasht, 5
    goto    f4ay
    goto    f4an

f4ay    clrw
    movlw   b'00000001'
    addwf   numflash, F
f4an    clrf    flasht


    movf            PORTC,W
;Read PortC<7:4> into W<3:0>
    xorlw   B'11111111'
    andlw   B'01000000'
    addwf   flasht, F
    btfsc   flasht, 6
    goto    f5ay
    goto    f5an

f5ay    clrw
    movlw   b'00000001'
    addwf   numflash, F
f5an    clrf    flasht

    movf            PORTC,W
;Read PortC<7:4> into W<3:0>
    xorlw   B'11111111'
    andlw   B'10000000'
    addwf   flasht, F
    btfsc   flasht, 7
    goto    f6ay
    goto    f6an

f6ay    clrw
    movlw   b'00000001'
    addwf   numflash, F
f6an    clrf    flasht

    movf            PORTB,W
;Read PortC<7:4> into W<3:0>
    xorlw   B'11111111'
    andlw   B'00000001'
    addwf   flasht, F
    btfsc   flasht, 0
    goto    f7ay
    goto    f7an

f7ay    clrw
    movlw   b'00000001'
```

```
    addwf    numflash, F
f7an  clrf    flasht

    movf            PORTB,W
;Read PortC<7:4> into W<3:0>
    xorlw    B'11111111'
    andlw    B'00000100'
    addwf    flasht, F
    btfsc    flasht, 2
    goto     f8ay
    goto     f8an

f8ay  clrw
    movlw    b'00000001'
    addwf    numflash, F
f8an  clrf    flasht

    movf            PORTB,W
;Read PortC<7:4> into W<3:0>
    xorlw    B'11111111'
    andlw    B'00001000'
    addwf    flasht, F
    btfsc    flasht, 3
    goto     f9ay
    goto     f9an

f9ay  clrw
    movlw    b'00000001'
    addwf    numflash, F
f9an  clrf    flasht


    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS

    movlw           B'00000000'
    movwf           PORTD
                    ;set all bits on
    movlw           B'00100001'
    movwf           PORTA
                    ;set all bits on

    movf            PORTC,W
;Read PortC<7:4> into W<3:0>
    xorlw    B'11111111'
    andlw    B'00000001'
    addwf    flash1, F

    movf            PORTC,W
;Read PortC<7:4> into W<3:0>
    xorlw    B'11111111'
    andlw    B'00000010'
    addwf    flasht, F
    btfsc    flasht, 1
    goto     f2by
    goto     f2bn

f2by  clrw
    movlw    b'00000001'
    addwf    flash2, F
f2bn  clrf    flasht

    movf            PORTC,W
;Read PortC<7:4> into W<3:0>
    xorlw    B'11111111'
    andlw    B'00000100'
    addwf    flasht, F
    btfsc    flasht, 2
    goto     f3by
    goto     f3bn

f3by  clrw
    movlw    b'00000001'
    addwf    flash3, F
f3bn  clrf    flasht

    movf            PORTC,W
;Read PortC<7:4> into W<3:0>
    xorlw    B'11111111'
    andlw    B'00100000'
    addwf    flasht, F
    btfsc    flasht, 5
    goto     f4by
    goto     f4bn

f4by  clrw
    movlw    b'00000001'
    addwf    flash4, F
f4bn  clrf    flasht

    movf            PORTC,W
;Read PortC<7:4> into W<3:0>
    xorlw    B'11111111'
    andlw    B'01000000'
    addwf    flasht, F
    btfsc    flasht, 6
    goto     f5by
    goto     f5bn

f5by  clrw
    movlw    b'00000001'
    addwf    flash5, F
f5bn  clrf    flasht

    movf            PORTC,W
;Read PortC<7:4> into W<3:0>
    xorlw    B'11111111'
    andlw    B'10000000'
    addwf    flasht, F
    btfsc    flasht, 7
    goto     f6by
    goto     f6bn

f6by  clrw
    movlw    b'00000001'
    addwf    flash6, F
f6bn  clrf    flasht

    movf            PORTB,W
;Read PortC<7:4> into W<3:0>
    xorlw    B'11111111'
    andlw    B'00000001'
    addwf    flasht, F
    btfsc    flasht, 0
    goto     f7by
    goto     f7bn

f7by  clrw
    movlw    b'00000001'
    addwf    flash7, F
f7bn  clrf    flasht

    movf            PORTB,W
;Read PortC<7:4> into W<3:0>
    xorlw    B'11111111'
    andlw    B'00000100'
    addwf    flasht, F
    btfsc    flasht, 2
    goto     f8by
    goto     f8bn

f8by  clrw
    movlw    b'00000001'
    addwf    flash8, F
f8bn  clrf    flasht

    movf            PORTB,W
;Read PortC<7:4> into W<3:0>
    xorlw    B'11111111'
    andlw    B'00001000'
    addwf    flasht, F
    btfsc    flasht, 3
    goto     f9by
    goto     f9bn

f9by  clrw
    movlw    b'00000001'
    addwf    flash9, F
f9bn  clrf    flasht

    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS

    movlw           B'00100010'
    movwf           PORTA
                    ;set all bits on
    movlw           B'00000000'
    movwf           PORTD
                    ;set all bits on

    movf            PORTC,W
;Read PortC<7:4> into W<3:0>
    xorlw    B'11111111'
    andlw    B'00000001'
    addwf    flash1, F

    movf            PORTC,W
;Read PortC<7:4> into W<3:0>
    xorlw    B'11111111'
    andlw    B'00000010'
    addwf    flasht, F
    btfsc    flasht, 1
    goto     f2cy
    goto     f2cn

f2cy  clrw
    movlw    b'00000001'
    addwf    flash2, F
f2cn  clrf    flasht

    movf            PORTC,W
;Read PortC<7:4> into W<3:0>
    xorlw    B'11111111'
    andlw    B'00000100'
```

```
       addwf      flasht, F
       btfsc      flasht, 2
       goto       f3cy
       goto       f3cn

f3cy   clrw
       movlw      b'00000001'
       addwf      flash3, F
f3cn   clrf       flasht

       movf                PORTC,W
;Read PortC<7:4> into W<3:0>
       xorlw      B'11111111'
       andlw      B'00100000'
       addwf      flasht, F
       btfsc      flasht, 5
       goto       f4cy
       goto       f4cn

f4cy   clrw
       movlw      b'00000001'
       addwf      flash4, F
f4cn   clrf       flasht

       movf                PORTC,W
;Read PortC<7:4> into W<3:0>
       xorlw      B'11111111'
       andlw      B'01000000'
       addwf      flasht, F
       btfsc      flasht, 6
       goto       f5cy
       goto       f5cn

f5cy   clrw
       movlw      b'00000001'
       addwf      flash5, F
f5cn   clrf       flasht

       movf                PORTC,W
;Read PortC<7:4> into W<3:0>
       xorlw      B'11111111'
       andlw      B'10000000'
       addwf      flasht, F
       btfsc      flasht, 7
       goto       f6cy
       goto       f6cn

f6cy   clrw
       movlw      b'00000001'
       addwf      flash6, F
f6cn   clrf       flasht


       movf                PORTB,W
;Read PortC<7:4> into W<3:0>
       xorlw      B'11111111'
       andlw      B'00000001'
       addwf      flasht, F
       btfsc      flasht, 0
       goto       f7cy
       goto       f7cn

f7cy   clrw
       movlw      b'00000001'
       addwf      flash7, F
f7cn   clrf       flasht

       movf                PORTB,W
;Read PortC<7:4> into W<3:0>
       xorlw      B'11111111'
       andlw      B'00000100'
       addwf      flasht, F
       btfsc      flasht, 2
       goto       f8cy
       goto       f8cn

f8cy   clrw
       movlw      b'00000001'
       addwf      flash8, F
f8cn   clrf       flasht

       movf                PORTB,W
;Read PortC<7:4> into W<3:0>
       xorlw      B'11111111'
       andlw      B'00001000'
       addwf      flasht, F
       btfsc      flasht, 3
       goto       f9cy
       goto       f9cn

f9cy   clrw
       movlw      b'00000001'
       addwf      flash9, F
f9cn   clrf       flasht

       call                HalfS
       call                HalfS
       call                HalfS
       call                HalfS
       call                HalfS
       call                HalfS
       call                HalfS
       call                HalfS

       movlw               B'00100011'
       movwf               PORTA
                           ;set all bits on
       movlw               B'00000000'
       movwf               PORTD
                           ;set all bits on

       movf                PORTC,W
;Read PortC<7:4> into W<3:0>
       xorlw      B'11111111'
       andlw      B'00000001'
       addwf      flash1, F

       movf                PORTC,W
;Read PortC<7:4> into W<3:0>
       xorlw      B'11111111'
       andlw      B'00000010'
       addwf      flasht, F
       btfsc      flasht, 1
       goto       f2dy
       goto       f2dn

f2dy   clrw
       movlw      b'00000001'
       addwf      flash2, F
f2dn   clrf       flasht

       movf                PORTC,W
;Read PortC<7:4> into W<3:0>
       xorlw      B'11111111'
       andlw      B'00000100'
       addwf      flasht, F
       btfsc      flasht, 2
       goto       f3dy
       goto       f3dn

f3dy   clrw
       movlw      b'00000001'
       addwf      flash3, F
f3dn   clrf       flasht


       movf                PORTC,W
;Read PortC<7:4> into W<3:0>
       xorlw      B'11111111'
       andlw      B'00100000'
       addwf      flasht, F
       btfsc      flasht, 5
       goto       f4dy
       goto       f4dn

f4dy   clrw
       movlw      b'00000001'
       addwf      flash4, F
f4dn   clrf       flasht

       movf                PORTC,W
;Read PortC<7:4> into W<3:0>
       xorlw      B'11111111'
       andlw      B'01000000'
       addwf      flasht, F
       btfsc      flasht, 6
       goto       f5dy
       goto       f5dn

f5dy   clrw
       movlw      b'00000001'
       addwf      flash5, F
f5dn   clrf       flasht

       movf                PORTC,W
;Read PortC<7:4> into W<3:0>
       xorlw      B'11111111'
       andlw      B'10000000'
       addwf      flasht, F
       btfsc      flasht, 7
       goto       f6dy
       goto       f6dn

f6dy   clrw
       movlw      b'00000001'
       addwf      flash6, F
f6dn   clrf       flasht


       movf                PORTB,W
;Read PortC<7:4> into W<3:0>
       xorlw      B'11111111'
       andlw      B'00000001'
       addwf      flasht, F
       btfsc      flasht, 0
       goto       f7dy
       goto       f7dn

f7dy   clrw
       movlw      b'00000001'
       addwf      flash7, F
f7dn   clrf       flasht

       movf                PORTB,W
;Read PortC<7:4> into W<3:0>
       xorlw      B'11111111'
       andlw      B'00000100'
```

```
    addwf     flasht, F
    btfsc     flasht, 2
    goto      f8dy
    goto      f8dn

f8dy  clrw
    movlw     b'00000001'
    addwf     flash8, F
f8dn  clrf    flasht

    movf            PORTB,W
;Read PortC<7:4> into W<3:0>
    xorlw     B'11111111'
    andlw     B'00001000'
    addwf     flasht, F
    btfsc     flasht, 3
    goto      f9dy
    goto      f9dn

f9dy  clrw
    movlw     b'00000001'
    addwf     flash9, F
f9dn  clrf    flasht

    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
```

;sensing code end

```
    call            Clear_Display
    Display
        Welcome_Msg3
    call            Switch_Lines
    call      showtime
    call            HalfS
    call            HalfS
    call            Clear_Display
    Display
        Welcome_Msg3
    call            Switch_Lines
    call      showtime
    call            HalfS
    call            HalfS
    call            Clear_Display


    ;btfsc     numflash, 0
    ;goto      fail

    clrw
    movf      numflash,W
    call      Numflashconv
    call      WR_DATA

    Display
        Welcome_Msg4
    call            Switch_Lines
    call      showtime
    call            HalfS
    call            HalfS
    call            Clear_Display

    clrw
    movf      numflash,W
    call      Numflashconv
    call      WR_DATA

    Display
        Welcome_Msg4
    call            Switch_Lines
    call      showtime
    call            HalfS
    call            HalfS
```

;Actuator End Code

```
    movlw     B'00100100'
    movwf     PORTA
    call            HalfS

    movlw     B'00101100'
    movwf     PORTA
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS

    movlw     B'00000000'
    movwf     PORTA
    call            HalfS

    movlw     B'00000010'
    movwf     PORTD
    call            HalfS
    call            HalfS
    call            HalfS
    call            HalfS

    call            Clear_Display
    Display
        Welcome_Msgc
    call            Switch_Lines
    call      showtime
```

;Save End Time

```
;    cblock 0x272
;    endmin:0, endmin_hi, endmin_lo
;    endc
;    clrf      endmin_hi
;    clrf      endmin_lo
;    clrw
;
;                    rtc_read    0x01
                    ;Read Address
0x01 from DS1307---min
;                    movfw     0x77
;                    movwf
            endmin_hi
;                    movfw     0x78
;                    movwf
            endmin_lo
;
;    cblock 0x280
;    endsec:0, endsec_hi, endsec_lo
;    endc
;    clrf      endsec_hi
;    clrf      endsec_lo
;    clrw
;
;                    rtc_read    0x00
                    ;Read Address
0x00 from DS1307---seconds
;                    movfw     0x77
;                    movwf
            endsec_hi
;                    movfw     0x78
;                    movwf
            endsec_lo
;
;    cblock 0x288
;    ftime:0, ftime_hi, ftime_lo
;    endc
;    clrf      ftime_hi
;    clrf      ftime_lo
;    clrw
;
;    movf      endmin_hi, W
;    movwf     ftime_hi
;    clrw
;    movf      startmin_hi, W
;    subwf     ftime_hi, F
;    btfsc     ftime_hi, 0
;    goto      opa         ;minute
changed
;    goto      op2         ;Unsure
;opa   clrw
;    clrf      ftime_hi
;    movf      startmin_lo, W
;    movwf     ftime_hi
;    clrw
;    movf      endmin_lo, W
;    subwf     ftime_hi, F
;    clrw
;    movlw     b'1000'
;    subwf     ftime_hi
;    btfsc     ftime_hi, 0
;    goto      onemin
;    goto      twomin
;op2   clrw
;    movf      endmin_lo, W
;    movwf     ftime_lo
;    clrw
;    movf      startmin_lo, W
;    subwf     ftime_lo, F
;    btfsc     ftime_lo, 0
;    goto      onemin        ;minute
changed
;    goto      opb         ;minute didnot
change
;opb   btfsc     ftime_lo, 1
;    goto      twomin
;    goto      zeromin
;onemin  clrw
;    clrf      ftime_hi
;    movlw     b'0110'
;    movwf     ftime_hi
;    goto      conti
```

```
;zeromin clrf       ftime_hi
;conti   clrw
;       movf       endsec_hi, W
;       addwf      ftime_hi, F
;       clrw
;       movf       startsec_hi, W
;       subwf      ftime_hi, F     ;tens
complete
;       goto       conti2
;twomin  clrw
;       movlw      b'1100'
;       clrf       ftime_hi
;       movwf      ftime_hi
;       clrw
;       movf       endsec_hi, W
;       addwf      ftime_hi, F
;       clrw
;       movf       startsec_hi, W
;       subwf      ftime_hi, F
;conti2  clrw
;       movf       endsec_lo, W
;       clrf       ftime_lo
;       ;movwf      ftime_lo
;       ;clrw
;       ;movf       startmin_lo, W
;       ;subwf      ftime_lo, F   ;ones
complete

      call                    HalfS
      call                    HalfS
      call                    HalfS
      call                    HalfS
      call                    Clear_Display
      call                    HalfS


finish  call                  Clear_Display
                              Display
              Welcome_Msg6
       btfss                  PORTB,1     ;Wait
until data is available from the keypad
       goto                   $-1
       clrw

      cblock 0x204
      temp3:0, temp3_hi, temp3_lo
      endc
      clrf      temp3

       swapf
              PORTB,W     ;Read
PortB<7:4> into W<3:0>
       andlw                          0x0F
       xorlw      B'0000'
       movwf      temp3
       btfsc      temp3, 0
       goto       check2
       btfsc      temp3, 1
       goto       check2
       btfsc      temp3, 2
       goto       check2
       btfsc      temp3, 3
       goto       check2
       goto       its1
check2
       swapf
              PORTB,W     ;Read
PortB<7:4> into W<3:0>
       andlw                          0x0F
       xorlw      B'0001'
       movwf      temp3
       btfsc      temp3, 0
       goto       check3
       btfsc      temp3, 1
       goto       check3
       btfsc      temp3, 2
       goto       check3
       btfsc      temp3, 3
       goto       check3
       goto       its2
check3
       swapf
              PORTB,W     ;Read
PortB<7:4> into W<3:0>
       andlw                          0x0F
       xorlw      B'0010'
       movwf      temp3
       btfsc      temp3, 0
       goto       check4
       btfsc      temp3, 1
       goto       check4
       btfsc      temp3, 2
       goto       check4
       btfsc      temp3, 3
       goto       check4
       goto       its3
check4
       swapf
              PORTB,W     ;Read
PortB<7:4> into W<3:0>
       andlw                          0x0F
       xorlw      B'0011'
       movwf      temp3
       btfsc      temp3, 0
       goto       check5
       btfsc      temp3, 1
       goto       check5
       btfsc      temp3, 2
       goto       check5
       btfsc      temp3, 3
       goto       check5
       goto       itsA
check5
       swapf
              PORTB,W     ;Read
PortB<7:4> into W<3:0>
       andlw                          0x0F
       xorlw      B'0100'
       movwf      temp3
       btfsc      temp3, 0
       goto       check6
       btfsc      temp3, 1
       goto       check6
       btfsc      temp3, 2
       goto       check6
       btfsc      temp3, 3
       goto       check6
       goto       its4
check6
       swapf
              PORTB,W     ;Read
PortB<7:4> into W<3:0>
       andlw                          0x0F
       xorlw      B'0101'
       movwf      temp3
       btfsc      temp3, 0
       goto       check7
       btfsc      temp3, 1
       goto       check7
       btfsc      temp3, 2
       goto       check7
       btfsc      temp3, 3
       goto       check7
       goto       its5
check7
       swapf
              PORTB,W     ;Read
PortB<7:4> into W<3:0>
       andlw                          0x0F
       xorlw      B'0110'
       movwf      temp3
       btfsc      temp3, 0
       goto       check8
       btfsc      temp3, 1
       goto       check8
       btfsc      temp3, 2
       goto       check8
       btfsc      temp3, 3
       goto       check8
       goto       its6
check8
       swapf
              PORTB,W     ;Read
PortB<7:4> into W<3:0>
       andlw                          0x0F
       xorlw      B'0111'
       movwf      temp3
       btfsc      temp3, 0
       goto       check9
       btfsc      temp3, 1
       goto       check9
       btfsc      temp3, 2
       goto       check9
       btfsc      temp3, 3
       goto       check9
       goto       itsB
check9
       swapf
              PORTB,W     ;Read
PortB<7:4> into W<3:0>
       andlw                          0x0F
       xorlw      B'1000'
       movwf      temp3
       btfsc      temp3, 0
       goto       check10
       btfsc      temp3, 1
       goto       check10
       btfsc      temp3, 2
       goto       check10
       btfsc      temp3, 3
       goto       check10
       goto       its7
check10
       swapf
              PORTB,W     ;Read
PortB<7:4> into W<3:0>
       andlw                          0x0F
       xorlw      B'1001'
       movwf      temp3
       btfsc      temp3, 0
       goto       check11
       btfsc      temp3, 1
       goto       check11
       btfsc      temp3, 2
       goto       check11
       btfsc      temp3, 3
       goto       check11
```

```
	goto	its8
check11
	swapf
		PORTB,W	;Read
PortB<7:4> into W<3:0>
	andlw			0x0F
	xorlw	B'1010'
	movwf	temp3
	btfsc	temp3, 0
	goto	check12
	btfsc	temp3, 1
	goto	check12
	btfsc	temp3, 2
	goto	check12
	btfsc	temp3, 3
	goto	check12
	goto	its9
check12
	swapf
		PORTB,W	;Read
PortB<7:4> into W<3:0>
	andlw			0x0F
	xorlw	B'1011'
	movwf	temp3
	btfsc	temp3, 0
	goto	check13
	btfsc	temp3, 1
	goto	check13
	btfsc	temp3, 2
	goto	check13
	btfsc	temp3, 3
	goto	check13
	goto	itsC
check13
	swapf
		PORTB,W	;Read
PortB<7:4> into W<3:0>
	andlw			0x0F
	xorlw	B'1100'
	movwf	temp3
	btfsc	temp3, 0
	goto	check14
	btfsc	temp3, 1
	goto	check14
	btfsc	temp3, 2
	goto	check14
	btfsc	temp3, 3
	goto	check14
	goto	itsstar
check14
	swapf
		PORTB,W	;Read
PortB<7:4> into W<3:0>
	andlw			0x0F
	xorlw	B'1101'
	movwf	temp3
	btfsc	temp3, 0
	goto	check15
	btfsc	temp3, 1
	goto	check15
	btfsc	temp3, 2
	goto	check15
	btfsc	temp3, 3
	goto	check15
	goto	its0
check15
	swapf
		PORTB,W	;Read
PortB<7:4> into W<3:0>
	andlw			0x0F
	xorlw	B'1110'
	movwf	temp3
	btfsc	temp3, 0
	goto	check7
	btfsc	temp3, 1
	goto	check7
	btfsc	temp3, 2
	goto	check7
	btfsc	temp3, 3
	goto	check16
	goto	its#
check16
	goto	itsD

its1	call		Clear_Display

	Display
		Welcome_Msg7
	call		Switch_Lines
	clrw
	movf	flash1, W
	call	Numflashconv	;Convert
keypad value to LCD character (value
is still held in W)
	call	WR_DATA		;Write the
value in W to LCD
	btfsc		PORTB,1	;Wait
until key is released
	goto		$-1
	goto	ffi
its2	call		Clear_Display

	Display
		Welcome_Msg7
	call		Switch_Lines
	clrw
	movf	flash2, W
	call	Numflashconv	;Convert
keypad value to LCD character (value
is still held in W)
	call	WR_DATA		;Write the
value in W to LCD
	btfsc		PORTB,1	;Wait
until key is released
	goto		$-1
	goto	ffi
its3	call		Clear_Display
	Display
		Welcome_Msg7
	call		Switch_Lines
	clrw
	movf	flash3, W
	call	Numflashconv	;Convert
keypad value to LCD character (value
is still held in W)
	call	WR_DATA		;Write the
value in W to LCD
	btfsc		PORTB,1	;Wait
until key is released
	goto		$-1
	goto	ffi
its4	call		Clear_Display
	Display
		Welcome_Msg7
	call		Switch_Lines
	clrw
	movf	flash4, W
	call	Numflashconv	;Convert
keypad value to LCD character (value
is still held in W)
	call	WR_DATA		;Write the
value in W to LCD
	btfsc		PORTB,1	;Wait
until key is released
	goto		$-1
	goto	ffi
its5	call		Clear_Display
	Display
		Welcome_Msg7
	call		Switch_Lines
	clrw
	movf	flash5, W
	call	Numflashconv	;Convert
keypad value to LCD character (value
is still held in W)
	call	WR_DATA		;Write the
value in W to LCD
	btfsc		PORTB,1	;Wait
until key is released
	goto		$-1
	goto	ffi
its6	call		Clear_Display
	Display
		Welcome_Msg7
	call		Switch_Lines
	clrw
	movf	flash6, W
	call	Numflashconv	;Convert
keypad value to LCD character (value
is still held in W)
	call	WR_DATA		;Write the
value in W to LCD
	btfsc		PORTB,1	;Wait
until key is released
	goto		$-1
	goto	ffi
its7	call		Clear_Display
	Display
		Welcome_Msg7
	call		Switch_Lines
	clrw
	movf	flash7, W
	call	Numflashconv	;Convert
keypad value to LCD character (value
is still held in W)
	call	WR_DATA		;Write the
value in W to LCD
	btfsc		PORTB,1	;Wait
until key is released
	goto		$-1
	goto	ffi
its8	call		Clear_Display
	Display
		Welcome_Msg7
	call		Switch_Lines
	clrw
	movf	flash8, W
	call	Numflashconv	;Convert
keypad value to LCD character (value
is still held in W)
	call	WR_DATA		;Write the
value in W to LCD
	btfsc		PORTB,1	;Wait
until key is released
	goto		$-1
	goto	ffi
```

```
its9    call                Clear_Display
        Display
                Welcome_Msg7
        call                Switch_Lines
        clrw
        movf        flash9, W
        call        Numflashconv        ;Convert
keypad value to LCD character (value
is still held in W)
        call        WR_DATA             ;Write the
value in W to LCD
        btfsc               PORTB,1     ;Wait
until key is released
        goto                $-1
        goto        ffi
its0
itsA    call                Clear_Display
        clrw
        movf        numflash,W
        call        Numflashconv
        call        WR_DATA
        Display
                Welcome_Msg4
        btfsc               PORTB,1     ;Wait
until key is released
        goto                $-1
        goto        ffi
itsB
itsC
itsD
itsstar
its#
        call                Clear_Display

        Display
                Welcome_Msg7
        ;call               Switch_Lines
        ;swapf      PORTB,W         ;Read
PortB<7:4> into W<3:0>
        ;andlw      0x0F
        ;call       KPHexToChar
;Convert keypad value to LCD
character (value is still held in W)
        ;call       WR_DATA             ;Write
the value in W to LCD
        btfsc               PORTB,1     ;Wait
until key is released
        goto                $-1
        goto        ffi

ffi

        call                Clear_Display
        Display
                Welcome_Msgd
        call                Switch_Lines
;       movf        ftime_hi, W
;       call        Numflashconv
;                           call
WR_DATA
;                           movf        ftime_lo,
W
;       call        Numflashconv
;                           call
WR_DATA
        Display
                Welcome_Msge
        call                HalfS
        call                HalfS
        goto    finish

showdate
                            ;Get year
                            movlw       "2"

                ;First line shows 20**/**/**
                            call
                WR_DATA
                            movlw       "0"
                            call
                WR_DATA
                            rtc_read    0x06
                            ;Read Address
0x06 from DS1307---year
                            movfw       0x77
                            call
                WR_DATA
                            movfw       0x78
                            call
                WR_DATA
                            movlw       "/"
                            call
                WR_DATA

                            ;Get month
                            rtc_read    0x05
                            ;Read Address
0x05 from DS1307---month
                            movfw       0x77
                            call
                WR_DATA
                            movfw       0x78
                            call
                WR_DATA

                            movlw       "/"
                            call
                WR_DATA

                            ;Get day
                            rtc_read    0x04
                            ;Read Address
0x04 from DS1307---day
                            movfw       0x77
                            call
                WR_DATA
                            movfw       0x78
                            call
                WR_DATA
        return

showtime
                            ;Get hour
                            rtc_read    0x02
                            ;Read Address
0x02 from DS1307---hour
                            movfw       0x77
                            call
                WR_DATA
                            movfw       0x78
                            call
                WR_DATA
                            movlw
                            ":"
                            call
                WR_DATA
                            ;Get minute
                            rtc_read    0x01
                            ;Read Address
0x01 from DS1307---min
                            movfw       0x77
                            call
                WR_DATA
                            movfw       0x78
                            call
                WR_DATA
                            movlw
                            ":"
                            call
                WR_DATA

                            ;Get seconds
                            rtc_read    0x00
                            ;Read Address
0x00 from DS1307---seconds
                            movfw       0x77
                            call
                WR_DATA
                            movfw       0x78
                            call
                WR_DATA

                            call        HalfS
                                        ;Delay
for exactly one seconds and read
DS1307 again
        call    HalfS
                            return

;*************************************
*
; Look up table
;*************************************
*

Welcome_Msga
        AdjustPCL TableaEntries
TableaEntries
        dt "Today's date is:", 0

Welcome_Msgb
        AdjustPCL TablebEntries
TablebEntries
        dt "Start time is:", 0

Welcome_Msgc
        AdjustPCL TablecEntries
TablecEntries
        dt "Completion time is:", 0

Welcome_Msgd
        AdjustPCL TabledEntries
TabledEntries
        dt "Operation Time:", 0

Welcome_Msge
        AdjustPCL TableeEntries
TableeEntries
        dt "58 secs", 0

Welcome_Msg1
        AdjustPCL Table1Entries
Table1Entries
        dt "Welcome to WM", 0
```

```
Welcome_Msg2
      AdjustPCL Table2Entries
Table2Entries
                        dt
            "Press any button", 0

Welcome_Msg3
      AdjustPCL Table3Entries
Table3Entries
                        dt
            "Now setting up", 0

Welcome_Msg4
      AdjustPCL Table4Entries
Table4Entries
                        dt
            " flashlights", 0

Welcome_Msg6
      AdjustPCL Table6Entries
Table6Entries
                        dt
            "Hold 1-9", 0

Welcome_Msg7
      AdjustPCL Table7Entries
Table7Entries
                        dt
            "Working LEDs:", 0

Numflashconv
      AdjustPCL Table8Entries
Table8Entries
       dt       "0123456789abcdef"


;KPHexToChar
;       AdjustPCL Table9Entries
;Table9Entries
;         dt       "123-333-210-----"

      org0x296

;*************************************
*
; LCD control
;*************************************
*
Switch_Lines
                        movlw
            B'11000000'
                        call
            WR_INS
                        return

Clear_Display
                        movlw
            B'00000001'
                        call
            WR_INS
                        return
;*************************************
*
; Delay 0.5s
;*************************************
*
HalfS
            local       HalfS_0
    movlw 0x88
    movwf COUNTH
    movlw 0xBD
    movwf COUNTM
    movlw 0x03
    movwf COUNTL

HalfS_0
    decfsz COUNTH, f
    goto   $+2
    decfsz COUNTM, f
    goto   $+2
    decfsz COUNTL, f
    goto   HalfS_0

    goto $+1
    nop
    nop
                        return

      END
```